Assignment 5 SSY316

Matthew Newson, Erik Norlin

December 2023

Exercise 1

1.

See appendix.

2.

Figure 1 shows a histogram of the approximated distribution and the curves of the proposal distribution, and the target distribution with 1000 number of samples. The quality of the approximation obtained is OK, and the weighted samples aligns somewhat with the target distribution. However, increasing the number of samples may increase the alignment of the bins with the target curve.

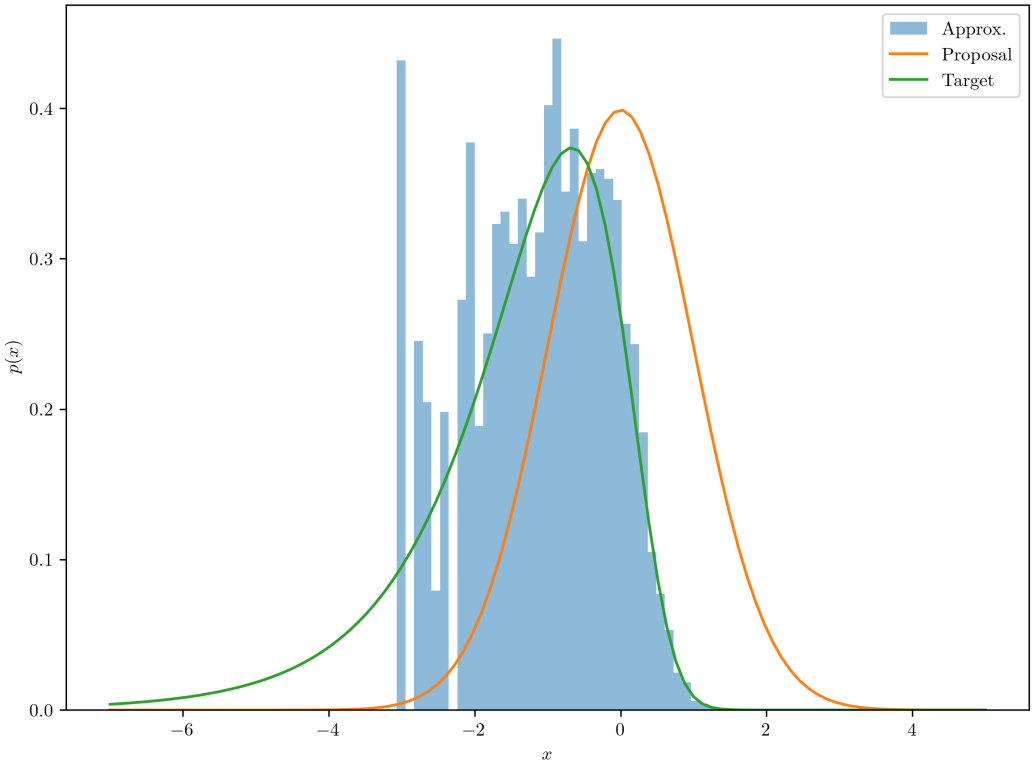

Figure 1: Approximated distribution, proposal distribution and target distribution with 1000 samples.

3.

Figure 2 shows the same as as above but this time with 10000 number of samples. As can be seen, the quality of the Monte Carlo approximation is improved.

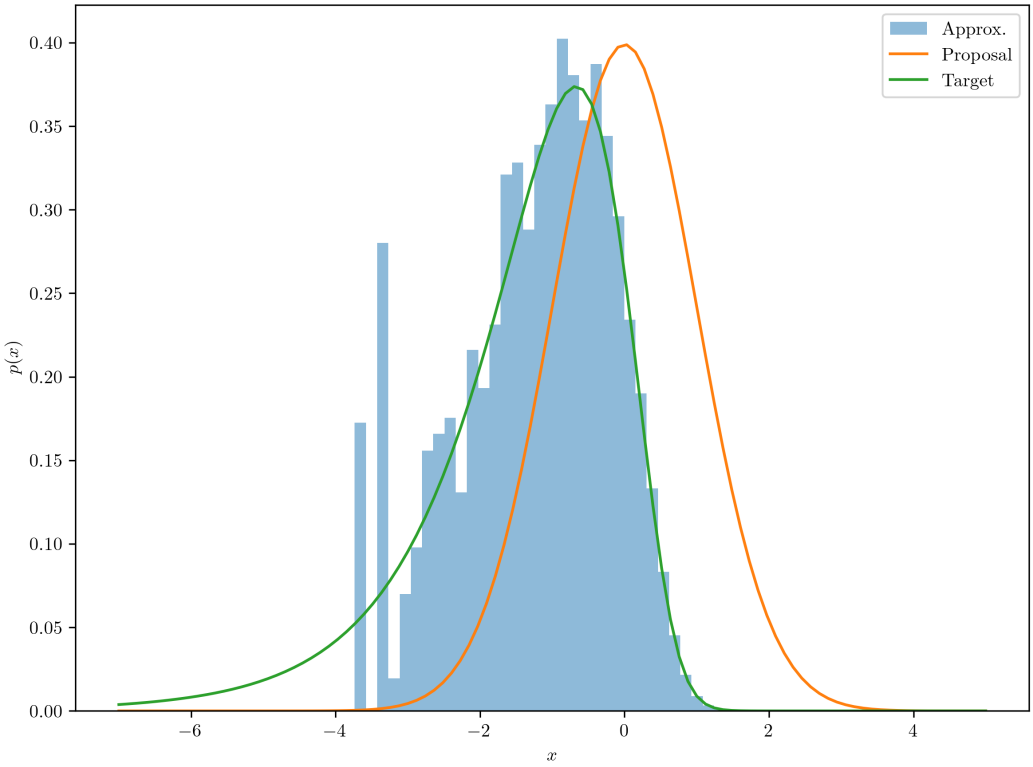

Figure 2: Approximated distribution, proposal distribution and target distribution with 10000 samples.

4.

Figure 3 shows the same as above but this time with the mean and variance of the proposal distribution as -2 and 4. Modifying the proposal distribution like this improves the approximation as well.

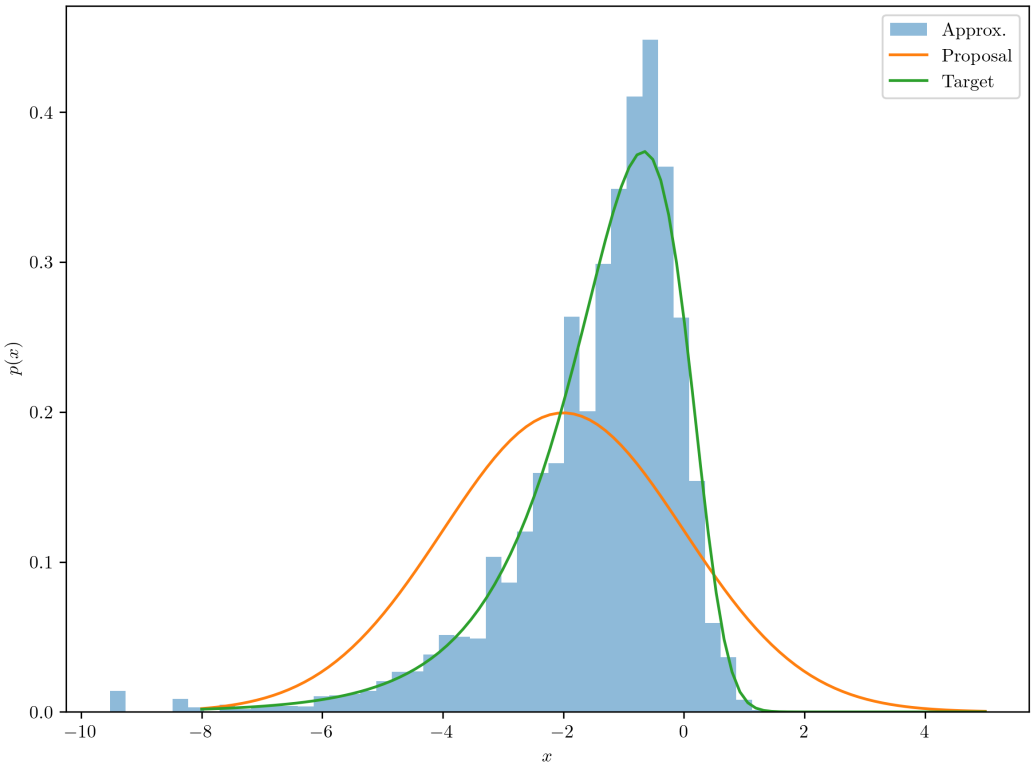

Figure 3: Approximated distribution, proposal distribution and target distribution with 1000 samples.

5.

Figure 4 shows estimates of the mean and variance of the approximation for different number of samples. It can be observed that the mean and variance barely deviates with increasing number of samples. The estimated means and variances stay around -1.41 and 1.98 as theory shows.

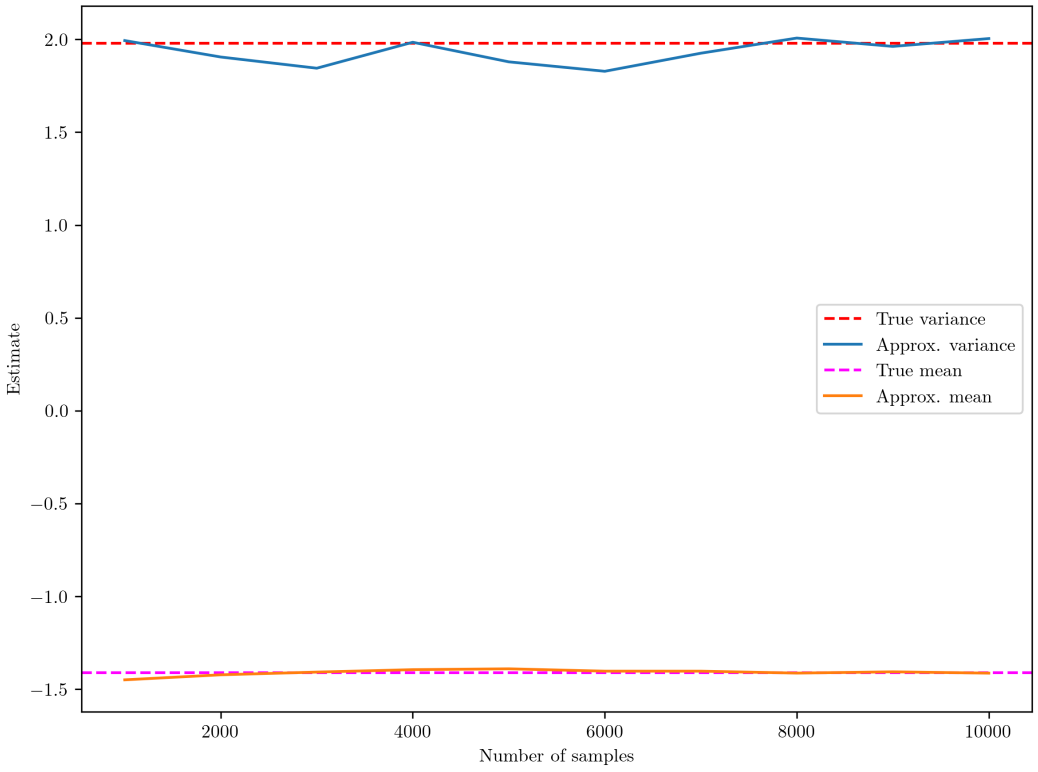

Figure 4: Mean and variance estimates for different number of samples.

6.

Figure 5 shows similarly as above but the proposal distribution as a uniform distribution between -4 and 1. This is not a valid importance sampler because the target distribution is defined as $p(x) > 0$ where the proposal distribution is undefined $q(x) < -4$ and $q(x) > 1$.

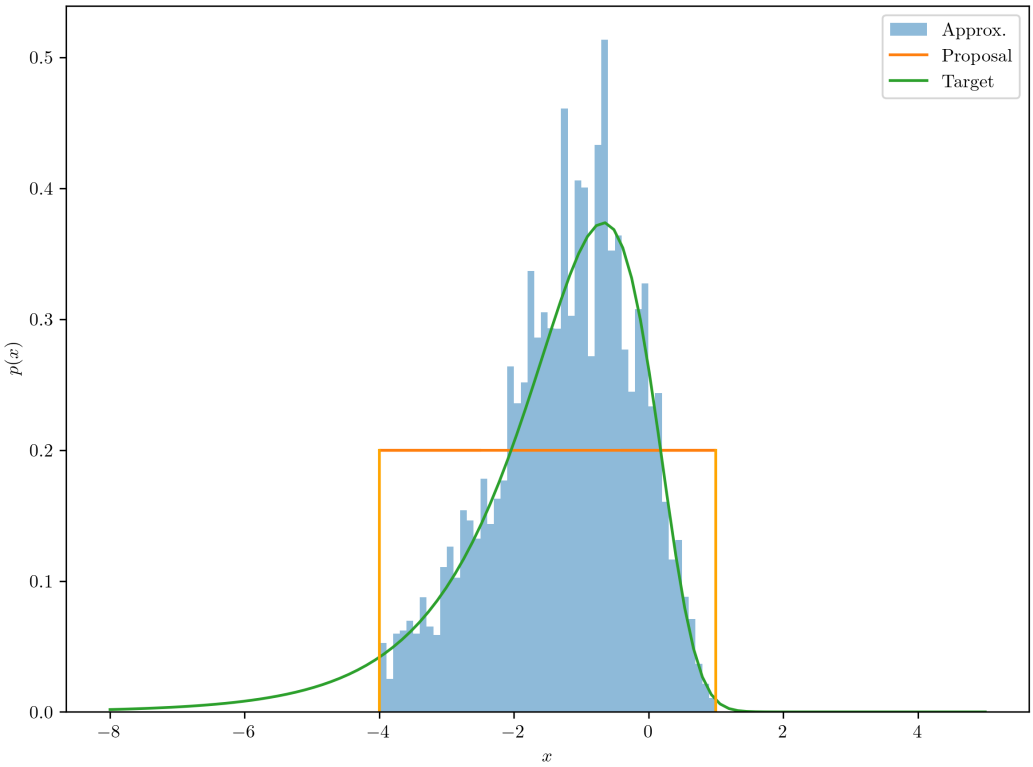

Figure 5: Approximated distribution, proposal distribution and target distribution with 1000 samples.

Exercise 2

Here we consider the following Markov chain whose stationary distribution is a standard normal distribution

$$x[k+1] = 0.9x[k] + v[k], v[k] \propto \mathcal{N}(0, 0.19)$$

1.

The equation above is a Markov chain because it satisfies the Markov property, which is that the future state depends only on the present state, and not on the past, i.e.,

$$p(x[k+1] \mid x[k], x[k-1], ..., x[1]) = p(x[k+1] \mid x[k])$$

2.

Figure 6 shows a comparison between $\mathcal{N}(0, 1)$ and an MCMC simulation of the Markov chain for 300 samples. We can observe that the MCMC simulation roughly approximates to $\mathcal{N}(0, 1)$ where the mean and variance of the approximated distribution is 0.03 and 1.28.

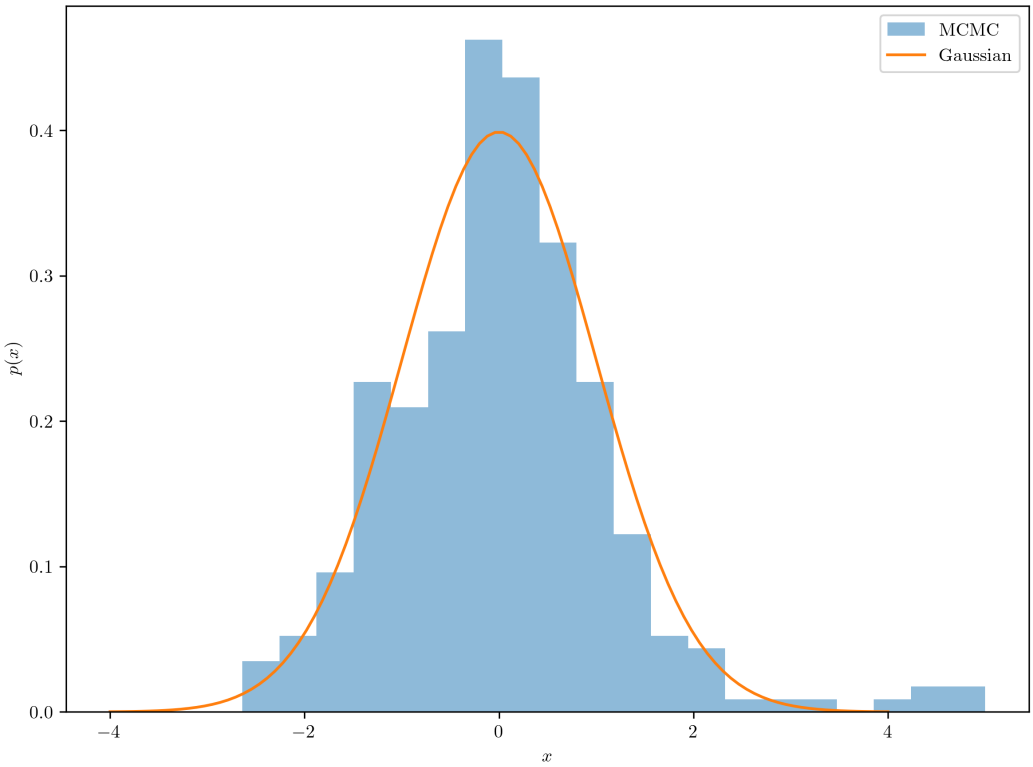

Figure 6: $\mathcal{N}(0, 1)$ and the approximated distribution from the MCMC simulation.

3.

Figure 7 shows how the numerically estimated state x change over the 300 iterations of the MCMC simulation. The burn-in period where up until the Markov chain starts to converge to its stationary distribution looks like to be around 20 iterations.

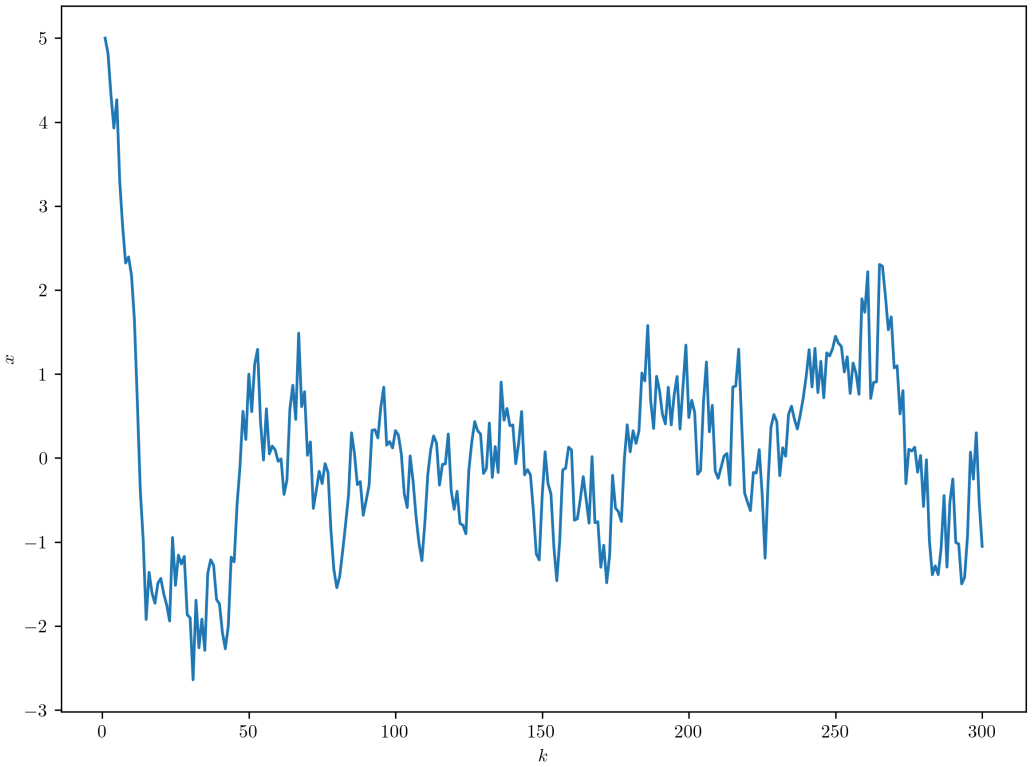

Figure 7: How the state x changes over iterations k.

4.

We prove here that the mean and variance for the stationary distribution of the Markov chain is 0 and 1. At stationarity we have that $x[k+1] = x[k]$, hence

$$\begin{aligned}
\mathbb{E}[x[k]] &= \mathbb{E}[0.9x[k] + v[k]] \\
&= 0.9\mathbb{E}[x[k]] + \mathbb{E}[v[k]] \\
&= 0.9\mathbb{E}[x[k]] + 0 \\
\Rightarrow \mathbb{E}[x[k]] &= 0
\end{aligned}$$

$$\begin{aligned}
\mathrm{Var}[x[k]] &= \mathrm{Var}[[0.9x[k] + v[k]] \\
&= 0.9^2\mathrm{Var}[x[k]] + \mathrm{Var}[v[k]] \\
&= 0.9^2\mathrm{Var}[x[k]] + 0.19 \\
\Rightarrow 0.19\mathrm{Var}[x[k]] &= 0.19 \\
\Rightarrow \mathrm{Var}[x[k]] &= 1
\end{aligned}$$

Exercise 3

1.

We have

$$\mathrm{KL}(p(x)\|q(x)) = -\int p(x)\ln q(x)\,dx + \int p(x)\ln p(x)\,dx.$$

The first term can be written as

$$\begin{aligned}
-\int p(x)\ln q(x)\,dx &= \int \mathcal{N}\left(x;\mu,\sigma^2\right)\frac{1}{2}\left[\ln(2\pi s^2)+\frac{(x-m)^2}{s^2}\right]dx \\
&= \int \mathcal{N}\left(x;\mu,\sigma^2\right)\frac{1}{2}\left[\ln(2\pi s^2)+\frac{x^2-2xm+m^2}{s^2}\right]dx \\
&= \int \mathcal{N}\left(x;\mu,\sigma^2\right)\frac{1}{2}\left[\ln(2\pi s^2)+\frac{(x-\mu)^2+2x(\mu-m)+m^2-\mu^2}{s^2}\right]dx \\
&= \frac{1}{2}\left[\ln(2\pi s^2)+\frac{\sigma^2+2\mu(\mu-m)+m^2-\mu^2}{s^2}\right] \\
&= \frac{1}{2}\left[\ln(2\pi s^2)+\frac{\sigma^2+(\mu-m)^2}{s^2}\right]
\end{aligned}$$

By replacing m and s with μ and σ, for the second term we get that

$$\int p(x)\ln p(x)\,dx = -\frac{1}{2}\left[\ln(2\pi\sigma^2)+1\right],$$

which gives

$$\mathrm{KL}(p(x)\|q(x)) = \frac{1}{2}\left[\ln\left(\frac{s^2}{\sigma^2}\right)+\frac{\sigma^2+(\mu-m)^2}{s^2}-1\right]$$

2.

Again, we have the Kullback-Leibler divergence as

$$\begin{aligned}
KL(p||q) = (p(x)\|q(x)) &= -\int p(x)\ln q(x)\,dx + \int p(x)\ln p(x)\,dx \\
&= \mathbb{E}_p[\ln(p)-\ln(q)] \\
&= \mathbb{E}_p\left[\frac{1}{2}\ln\frac{|S|}{|\Sigma|}-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)+\frac{1}{2}(x-m)^TS^{-1}(x-m)\right] \\
&= \frac{1}{2}\ln\frac{|S|}{|\Sigma|}-\frac{1}{2}\mathbb{E}_p\left[(x-\mu)^T\Sigma^{-1}(x-\mu)\right]+\frac{1}{2}\,\mathbb{E}_p\left[(x-m)^TS^{-1}(x-m)\right]
\end{aligned}$$

We rewrite the second term as

$$\begin{aligned}
&\Rightarrow \frac{1}{2}\mathbb{E}_p[tr\{(x-\mu)^T(x-\mu)\Sigma^{-1}\}] \\
&= \frac{1}{2}tr\{\mathbb{E}_p[(x-\mu)^T(x-\mu)]\Sigma^{-1}\} \\
&= \frac{1}{2}tr\{\Sigma\Sigma^{-1}\} \\
&= \frac{1}{2}tr\{I_k\} \\
&= \frac{k}{2}
\end{aligned}$$

Simplifying the third term yields

$$\frac{1}{2}\mathbb{E}_p\left[(x-m)^TS^{-1}(x-m)\right] = \frac{1}{2}\left[(\mu-m)^TS^{-1}(\mu-m)+tr\{S^{-1}\Sigma\}\right]$$

Finally, we have that

$$KL(p||q) = \frac{1}{2}\left[\ln\frac{|S|}{|\Sigma|}-k+(\mu-m)^TS^{-1}(\mu-m)+tr\{S^{-1}\Sigma\}\right]$$

Exercise 4

1.

The joint distribution is

$$p(y_{1:N}, x_{1:N}, \gamma) = p(\gamma)p(x_0)\prod_{n=1}^{N} p(x_n|x_{n-1}, \gamma)p(y_n|x_n)$$

where

$$\begin{aligned} p(x_0) &= \mathcal{N}(x_0; \mu_0, \beta_0^{-1}) \\ p(\gamma) &= \mathcal{N}(\gamma; 0, \beta_{\gamma 0}^{-1}) \\ p(x_n|x_{n-1}, \gamma) &= \mathcal{N}(x_n; x_{n-1}, \gamma, \beta_v^{-1}) \\ p(y_n|x_n) &= \mathcal{N}(y_n; x_n, \beta_e^{-1}) \end{aligned}$$

2.

To estimate the posterior $q(\gamma)$ using variational inference we define the mean field equation for $q(\gamma)$ as

$$\begin{aligned} \ln q(\gamma) &= \mathbb{E}_{\{q_n\}_{n=0}^N}\left[\ln p(y_{1:N}, x_{0:N}, \gamma)\right] + \text{const.} \\ &= \mathbb{E}_{\{q_n\}_{n=0}^N}\left[\ln p(\gamma) + \ln p(x_0) + \sum_{n=1}^{N} \ln p(x_n; x_{n-1}, \gamma) + \ln p(y_n; x_n)\right] + \text{const.} \\ &= \mathbb{E}_{\{q_n\}_{n=0}^N}\left[\ln p(\gamma) + \sum_{n=1}^{N} \ln p(x_n; x_{n-1}, \gamma)\right] + \text{const.} \\ &= \mathbb{E}_{\{q_n\}_{n=0}^N}\left[\ln p(\gamma) + \sum_{n=1}^{N}\left[\ln\left(\frac{\beta_v}{\sqrt{2\pi}}\right) - \frac{\beta_v}{2}(x_n - x_{n-1}\gamma)^2\right]\right] + \text{const.} \\ &= \ln p(\gamma) - \frac{\beta_v}{2}\sum_{n=1}^{N} \mathbb{E}_{q_n, q_{n-1}}\left[(x_n - x_{n-1}\gamma)^2\right] + \text{const.} \\ &= \ln p(\gamma) - \frac{\beta_v}{2}\sum_{n=1}^{N} \mathbb{E}_{q_n, q_{n-1}}\left[-2x_n x_{n-1}\gamma - x_{n-1}^2\gamma^2\right] + \text{const.} \\ &= \ln p(\gamma) - \frac{\beta_v}{2}\sum_{n=1}^{N}\left(-2\bar{x}_n\bar{x}_{n-1}\gamma - \overline{x_{n-1}^2}\gamma^2\right) + \text{const.} \\ &= \ln p(\gamma) - \frac{\beta_v}{2}\sum_{n=1}^{N}\overline{x_{n-1}^2}\left(\gamma^2 - \frac{\bar{x}_n\bar{x}_{n-1}}{\overline{x_{n-1}^2}}\right)^2 + \text{const.} \\ &= \ln p(\gamma) + \sum_{n=1}^{N}\ln\mathcal{N}\left(\gamma; \frac{\bar{x}_n\bar{x}_{n-1}}{\overline{x_{n-1}^2}}, (\beta_v\overline{x_{n-1}^2})^{-1}\right) + \text{const.} \end{aligned}$$

where

$$\bar{x}_n = \mathbb{E}_{q_n}[x_n], \text{ and } \overline{x_n^2} = \mathbb{E}_{q_n}[x_n^2]$$

This gives

$$q(\gamma) = \mathcal{N}(\gamma; \bar{\gamma}, \beta_\gamma^{-1})$$

where

$$\beta_\gamma = \beta_{\gamma 0} + \beta_v \sum_{n=1}^{N} \overline{x_{n-1}^2}$$

$$\bar{\gamma} = \frac{1}{\beta_\gamma} \left(\sum_{n=1}^{N} \frac{\bar{x}_n \bar{x}_{n-1}}{\overline{x_{n-1}^2}} \right) = \frac{\beta_v}{\beta_\gamma} \left(\sum_{n=1}^{N} \bar{x}_n \bar{x}_{n-1} \right)$$

3.

The update equations for $q(\gamma)$ will be the same as derived above, but instead $\bar{x}_n \bar{x}_{n-1}$ will be replaced with $\overline{x_n x_{n-1}} = \mathbb{E}_{\{q_n, q_{n-1}\}}[x_n x_{n-1}]$.

Appendix

```python
def evaluate_gaussian(x, mu, sigma):
    p = 1/np.sqrt(2*np.pi*sigma**2)*np.exp(-((x-mu)**2)/(2*sigma**2))
    return p

def evaluate_johnsons(x):
    p = np.sqrt(2)/np.sqrt(np.pi*(1+(x-1)**2))*np.exp(-0.5*(3+2*np.arcsinh(x-1))**2)
    return p

path = r'C:\Users\erikn\skola\SSY316-Advanced-Probabilistic-Machine-Learning\assignment-5\report-images'
save = False
plot = False

mu = -2
sigma2 = 4
sigma = np.sqrt(sigma2)
L = 1000

z = np.random.normal(mu, sigma, L)
prop = evaluate_gaussian(z, mu, sigma)
target = evaluate_johnsons(z)
weight = target / prop

x = np.linspace(-8,5,100)
prop = evaluate_gaussian(x, mu, sigma)
target = evaluate_johnsons(x)

if plot:
    fig = plt.figure(figsize=(8,6))
    ax = plt.axes()
    ax.hist(z, weights=weight, density=True, bins=50, alpha=0.5, label='Approx.')
    ax.plot(x, prop, label='Proposal')
    ax.plot(x, target, label='Target')
    ax.set_xlabel('$x$')
    ax.set_ylabel('$p(x)$')
    ax.legend(loc="upper right")
    fig.tight_layout()
    if save:
        fig.savefig(path+r'\E1.4.png', dpi=300)
    plt.show()

mean = np.sum(z*weight)/L
var = np.sum(weight*(z-mean)**2)/L
ic(L, mean, var)
```

Listing 1: Python script for Exercise 1.1.

```python
def evaluate_gaussian(x, mu, sigma):
    p = 1/np.sqrt(2*np.pi*sigma**2)*np.exp(-((x-mu)**2)/(2*sigma**2))
    return p

def evaluate_johnsons(x):
    p = np.sqrt(2)/np.sqrt(np.pi*(1+(x-1)**2))*np.exp(-0.5*(3+2*np.arcsinh(x-1))**2)
    return p

path = r'C:\Users\erikn\skola\SSY316-Advanced-Probabilistic-Machine-Learning\assignment-5\report-images'
save = False
plot = False

Ls = np.linspace(1000, 10000, 10)
means = []
vars= []

for L in Ls:
    mu = -2
    sigma2 = 4
    sigma = np.sqrt(sigma2)
    # L = 1000

    z = np.random.normal(mu, sigma, int(L))
    prop = evaluate_gaussian(z, mu, sigma)
    target = evaluate_johnsons(z)
    weight = target / prop

    x = np.linspace(-8,5,100)
    prop = evaluate_gaussian(x, mu, sigma)
    target = evaluate_johnsons(x)

    mean = np.sum(z*weight)/L
    var = np.sum(weight*(z-mean)**2)/L
    means.append(mean)
    vars.append(var)

if plot:
    fig = plt.figure(figsize=(8,6))
    ax = plt.axes()
    ax.axhline(1.98, linestyle='--', color='red', label='True variance')
    ax.plot(Ls, vars, label='Approx. variance')
    ax.axhline(-1.41, linestyle='--', color='magenta', label='True mean')
    ax.plot(Ls, means, label='Approx. mean')
    ax.set_xlabel('Number of samples')
    ax.set_ylabel('Estimate')
    ax.legend(loc="right")
    fig.tight_layout()
    if save:
        fig.savefig(path+r'\E1.5.png', dpi=300)
    plt.show()
```

Listing 2: Python script for Exercise 1.2.

```python
def evaluate_gaussian(x, mu, sigma):
    p = 1/np.sqrt(2*np.pi*sigma**2)*np.exp(-((x-mu)**2)/(2*sigma**2))
    return p

def evaluate_johnsons(x):
    p = np.sqrt(2)/np.sqrt(np.pi*(1+(x-1)**2))*np.exp(-0.5*(3+2*np.arcsinh(x-1))**2)
    return p

path = r'C:\Users\erikn\skola\SSY316-Advanced-Probabilistic-Machine-Learning\assignment-5\report-images'
save = True
plot = True

mu = -2
sigma2 = 4
sigma = np.sqrt(sigma2)
L = 1000

low = -4
high = 1
ymax = 1 / (high - low)

z = np.random.uniform(low, high, L)
prop = np.full_like(z, ymax)
target = evaluate_johnsons(z)
weight = target / prop

x = np.linspace(-8,5,100)
# prop = np.full_like(x, 1 / (high - low))
target = evaluate_johnsons(x)

if plot:
    fig = plt.figure(figsize=(8,6))
    ax = plt.axes()
    ax.hist(z, weights=weight, density=True, bins=50, alpha=0.5, label='Approx.')
    ax.plot(z, prop, label='Proposal')
    ax.plot(x, target, label='Target')
    ax.plot([low, low], [0, ymax], color='orange')
    ax.plot([high, high], [0, ymax], color='orange')
    ax.set_xlabel('$x$')
    ax.set_ylabel('$p(x)$')
    ax.legend(loc="upper right")
    fig.tight_layout()
    if save:
        fig.savefig(path+r'\E1.6.png', dpi=300)
    plt.show()

mean = np.sum(z*weight)/L
var = np.sum(weight*(z-mean)**2)/L
ic(L, mean, var)
```

Listing 3: Python script for Exercise 1.3.

```python
def evaluate_gaussian(x, mu, sigma):
    p = 1/np.sqrt(2*np.pi*sigma**2)*np.exp(-((x-mu)**2)/(2*sigma**2))
    return p

path = r'C:\Users\erikn\skola\SSY316-Advanced-Probabilistic-Machine-Learning\assignment-5\report-images'
save = False
plot = False

x = [5]

for k in range(299):
    x_next = 0.9*x[k] + np.random.normal(0, np.sqrt(0.19))
    x.append(x_next)

if plot:
    xp = np.linspace(-4,4,100)
    y = evaluate_gaussian(xp, 0, 1)

    fig = plt.figure(figsize=(8,6))
    ax = plt.axes()
    ax.hist(x, density=True, bins=20, alpha=0.5, label='MCMC')
    ax.plot(xp, y, label='Gaussian')
    ax.set_xlabel('$x$')
    ax.set_ylabel('$p(x)$')
    ax.legend(loc="upper right")
    fig.tight_layout()
    if save:
        fig.savefig(path+r'\E2.2.png', dpi=300)
    plt.show()

    k = np.linspace(1,300,300)

    fig = plt.figure(figsize=(8,6))
    ax = plt.axes()
    ax.plot(k, x)
    ax.set_xlabel('$k$')
    ax.set_ylabel('$x$')
    fig.tight_layout()
    if save:
        fig.savefig(path+r'\E2.3.png', dpi=300)
    plt.show()
    # mean and var (0.69700374324875, 1.2313778458425388)

ic(np.mean(x), np.var(x))
```

Listing 4: Python script for Exercise 2.2 and 2.3.