Imperial College London

COURSEWORK 2

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Stochastic Simulation

Author:
Richard Fu (CID: 01852979)

Date: November 29, 2022

1 QUESTION 1

1.1

From Lecture Notes, we get:

$$p(y) = \mathcal{N}(y; 0, 2) = \frac{1}{2\sqrt{\pi}} \exp\left(-\frac{x^2}{4}\right). \tag{1}$$

Evaluating at $y = 9$:

$$p(y = 9) = \frac{1}{2\sqrt{\pi}} \exp\left(-\frac{9^2}{4}\right) = 4.528 \times 10^{-10}. \tag{2}$$

1.2

Using test function $\varphi(x) = p(y = 9|x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(9-x)^2}{2}\right)$:

$$\bar{\varphi} = \mathbb{E}_p[\varphi(x)] = \int \varphi(x)p(x)dx \tag{3}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} \varphi(X_i)dx = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(9-x_i)^2}{2}\right) = \hat{\varphi}^N_{MC} \tag{4}$$

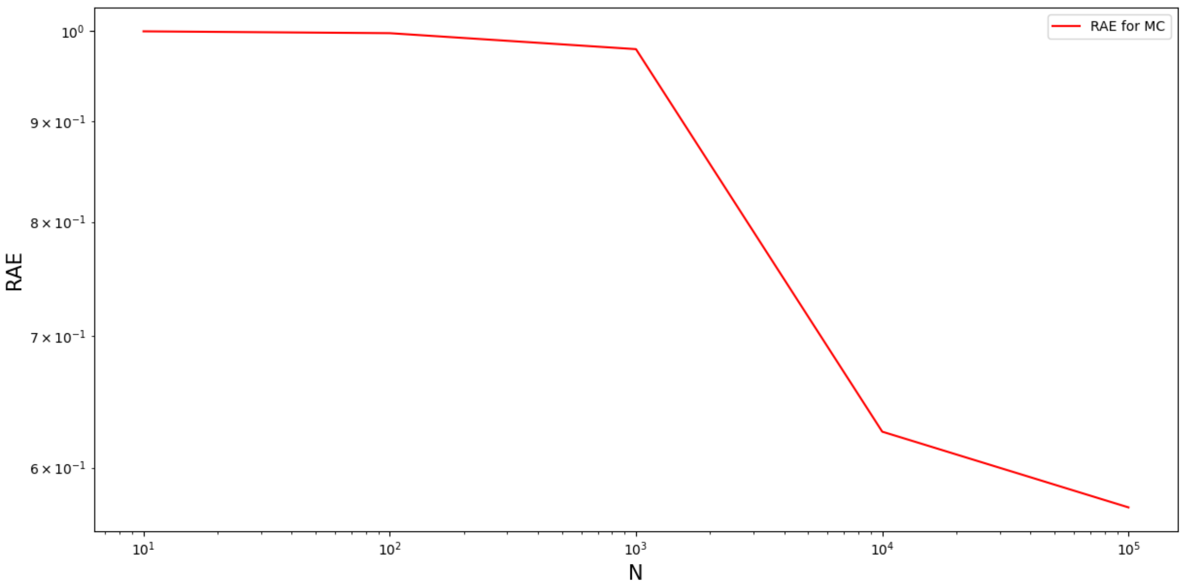

Figure 1

1.3

Again using test function $\varphi(x) = p(y=9|x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(9-x)^2}{2}\right)$:

$$\hat{\varphi}_{IS}^{N} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{w_i}\varphi(\mathrm{X_i}), \quad \text{where } \mathrm{w_i} = \frac{\mathrm{p(X_i)}}{\mathrm{q(X_i)}} \quad \text{i=1,...,N.} \tag{5}$$

$$\tag{6}$$

Now we compute this IS estimator to estimate $p(y=9)$:

$$p(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{-x^2}{2}\right), \tag{7}$$

$$q(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-6)^2}{2}\right), \tag{8}$$

$$\implies \frac{p(x)}{q(x)} = \exp(-6x+18). \tag{9}$$

$$\implies \hat{\varphi}_{IS}^{N} = \frac{1}{N}\sum_{i=1}^{N}\exp(-6x_i+18)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(9-x_i)^2}{2}\right) \tag{10}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x_i^2-6x_i+45)}{2}\right) \tag{11}$$

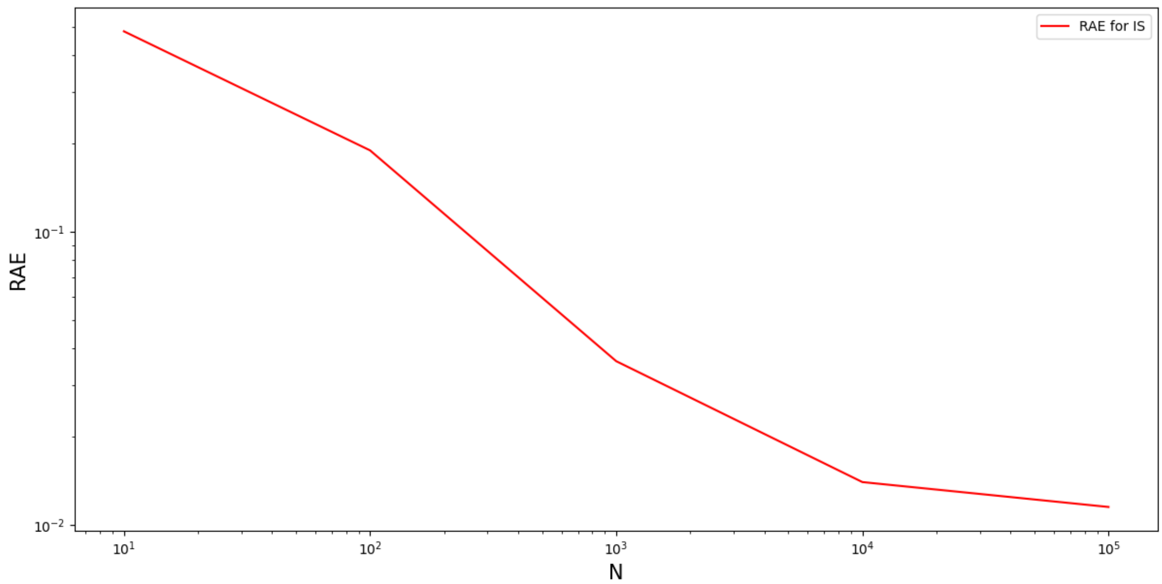

Figure 2

1.4

Comparing the RAE for MC and IS with each other:

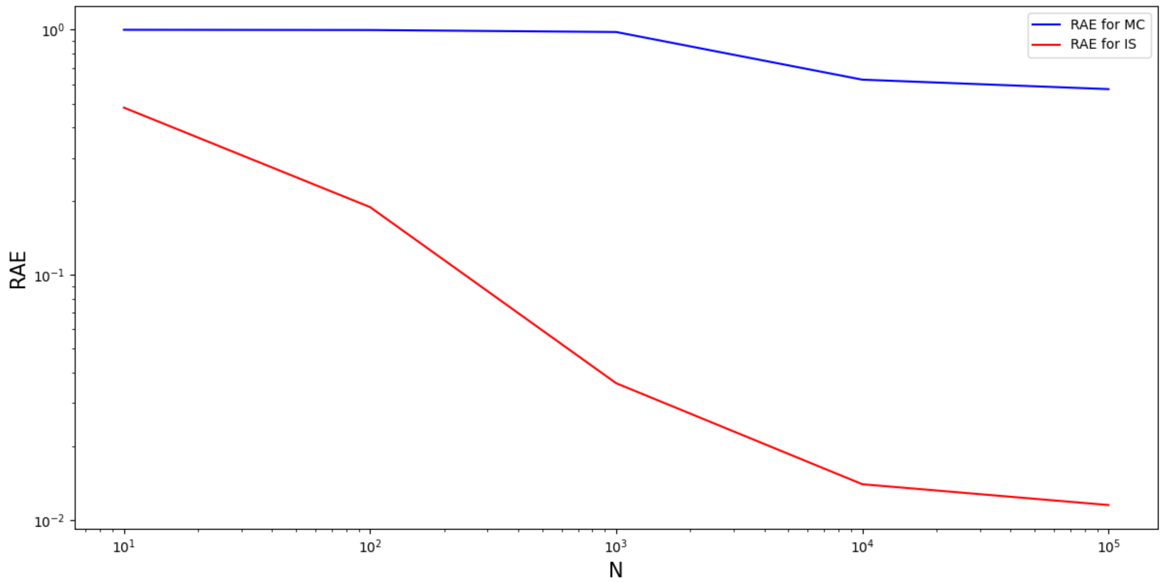

Figure 3

In Figure 3 we can see that the RAE is smaller when using IS compared to when using MC, so it is more accurate estimator.
As N increases, we can see that the RAE for IS also decreases quicker than the RAE for MC.
When using the MC estimator, we sample from $p(x) = \mathcal{N}(x; 0, 1)$ we typically get values close to 0, and so when we evaluate $p(y = 9|x)$ we will get very small values. Taking an average of these small values in the MC estimator will cause errors to occur.
Using the IS estimator, we sample from $q(x) = \mathcal{N}(x; 6, 1)$ we draw a lot more samples around 9, but take a weighted average in the IS estimator, which causes our accuracy to be better than the MC estimator.

Code Listing 1: Code for question 1

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

### Q1 Part 2
phi = lambda x: 1/np.sqrt(2*np.pi) * np.exp(-(9-x)**2 / 2) # phi(x)
true_value = 1/(2* np.sqrt(np.pi)) * np.exp(-9**2 / 4)

N_array = np.array([10, 100, 1000, 10000, 100000])
RAE_array_mc = np.array([])
for N in N_array:
    p_x = np.random.normal(0, 1, N) # sample from p(x) N times
    phi_mc = 1/N * np.sum(phi(p_x))
    RAE_mc = np.abs(phi_mc - true_value) / np.abs(true_value)
    print(phi_mc, true_value)
    RAE_array_mc = np.append(RAE_array_mc, RAE_mc)

plt.figure(figsize=(20,7))
plt.loglog(N_array, RAE_array_mc, "r-", label="RAE for MC")
plt.legend(loc="upper right")
plt.xlabel("N", fontsize=15)
plt.ylabel("RAE", fontsize=15)


### Q1 Part 3
w_phi = lambda x: 1/np.sqrt(2*np.pi) * np.exp(-(x**2 - 6*x + 45)/2) #
                                    w(x)phi(x)
RAE_array_is = np.array([])
for N in N_array:
    q_x = np.random.normal(6, 1, N) # sample from q(x) N times
    phi_is = 1/N * np.sum(w_phi(q_x))
    RAE_is = np.abs(phi_is - true_value) / np.abs(true_value)
    RAE_array_is = np.append(RAE_array_is, RAE_is)

plt.figure(figsize=(20,7))
plt.loglog(N_array, RAE_array_is, "r-", label="RAE for IS")
plt.legend(loc="upper right")
plt.xlabel("N", fontsize=15)
plt.ylabel("RAE", fontsize=15)


### Q1 Part 4
plt.figure(figsize=(20,7))
plt.loglog(N_array, RAE_array_mc, "b-", label="RAE for MC")
plt.loglog(N_array, RAE_array_is, "r-", label="RAE for IS")
plt.legend(loc="upper right")
plt.xlabel("N", fontsize=15)
plt.ylabel("RAE", fontsize=15)
plt.show()
```

2 QUESTION 2

2.1

Prior distribution:

$$p(x) = \mathcal{N}(x; \mu_x, \sigma_x^2). \tag{12}$$

Posterior distribution:

$$p(x|y_1, y_2, y_3, s_1, s_2, s_3) \propto p(x) \prod_{i=0}^{2} p(y_i|x, s_i). \tag{13}$$

$$\text{Let } \bar{p}_*(x) = p(x) \prod_{i=0}^{2} p(y_i|x, s_i). \tag{14}$$

Random walk proposal:

$$q(x', x) = \mathcal{N}(x', x, \sigma_q^2) \tag{15}$$

Calculate acceptance ratio and log-acceptance ratio:

$$
\begin{aligned}
r(x, x') &= \frac{\bar{p}_*(x')q(x, x')}{\bar{p}_*(x)q(x', x)} && (16)\\
&= \frac{\bar{p}_*(x')\frac{1}{\sqrt{2\pi\sigma_q^2}}\exp\left(-\frac{(x-x')^2}{2\sigma_q^2}\right)}{\bar{p}_*(x)\frac{1}{\sqrt{2\pi\sigma_q^2}}\exp\left(-\frac{(x'-x)^2}{2\sigma_q^2}\right)} && (17)\\
&= \frac{\bar{p}_*(x')}{\bar{p}_*(x)} && (18)\\
\mathrm{log(r(x, x'))} &= \mathrm{log(\bar{p}_*(x')) - log(\bar{p}_*(x))} && (19)\\
&= \mathrm{log(\bar{p}(x')) - log(\bar{p}(x))} + \sum_{\mathrm{i=0}}^{2}\mathrm{(log(p(y_i|x', s_i)) - log(p(y_i|x, s_i)))} && (20)\\
&= \frac{(x-\mu_x)^2 - (x'-\mu_x)^2}{2\sigma_x^2} + \sum_{i=0}^{2}\left(\frac{(y_i - \|x - s_i\|)^2 - (y_i - \|x' - s_i\|)^2}{2\sigma_y^2}\right) && (21)
\end{aligned}
$$

The MH algorithm, given starting point X_0 and the number of samples N:
For n=1,...,N:

1) Sample $X' \sim q(x'|X_{n-1})$
2) Sample $U \sim Uniform(0, 1)$
3) Calculate log-acceptance probability $\mathrm{log(r(x_{n-1}, x'))}$
4) If $\mathrm{log(U) \leq log(r(x_{n-1}, x'))}$, accept sample X' and set $X_n = X'$
5) Otherwise reject X' and set $X_n = X_{n-1}$

End for
6) Discard first `burnin` samples and return remaining samples

2.2

Code Listing 2: Code for question 2

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

### Q2 Part 2
N = 1000000
sigma_y = 1
mu_x = 0
sigma_x = 10
s_arr = np.array([-1, 2, 5])
y_arr = np.array([4.44, 2.51, 0.73])
x_true = 4

def log_acceptance_prob(x, x_prime):
    first_terms = ((x-mu_x)**2 - (x_prime-mu_x)**2) / (2*sigma_x**2)
    sum_term = np.sum([((y_arr[i] - np.abs(x - s_arr[i]))**2 - (y_arr[i] - np.abs(x_prime - s_arr[i]))**2) / (2*sigma_y**2) for i in range(3)])
    return first_terms + sum_term

def sample_MH(x0, sigma_q, N):
    x = x0
    accepted_samples = np.array([])
    for n in range(N):
        x_prime = np.random.normal(x, sigma_q)
        u = np.random.uniform(0, 1)
        prob = log_acceptance_prob(x, x_prime)
        if np.log(u) <= prob:
            accepted_samples = np.append(accepted_samples, x_prime)
            x = x_prime
    return accepted_samples

sample_a = sample_MH(10, 0.1, N)
burnin_a = 1000
burnin_b = 40000
sample_b = sample_MH(10, 0.01, N)
plt.figure(figsize=(20,7))
plt.title("Histogram for $\sigma_q = 0.1, \sigma_y = 1$", fontsize=15)
plt.axvline(x_true, color="k", label="true value", linewidth=2)
plt.hist(sample_a[burnin_a:], bins=50, density=True, label="posterior", alpha=0.5, color=[0.8, 0, 0])
plt.legend(loc="upper right")

plt.figure(figsize=(20,7))
plt.title("Histogram for $\sigma_q = 0.01, \sigma_y = 1$", fontsize=15)
plt.axvline(x_true, color="k", label="true value", linewidth=2)
plt.hist(sample_b[burnin_b:], bins=50, density=True, label="posterior", alpha=0.5, color=[0.8, 0, 0])
plt.legend(loc="upper right")
```

```
### Q2 Part 3
sigma_y = 0.1
y_arr = np.array([5.01, 1.97, 1.02])
sample_c = sample_MH(10, 0.1, N)
burnin_c = 100
plt.figure(figsize=(20,7))
plt.title("Histogram for $\sigma_q = 0.1, \sigma_y = 0.1$", fontsize=
    15)
plt.axvline(x_true, color="k", label="true value", linewidth=2)
plt.hist(sample_c[burnin_c:], bins=50, density=True, label="posterior
    ", alpha=0.5, color=[0.8, 0, 0])
plt.legend(loc="upper right")

### Plotting the samples
plt.figure(figsize=(20,7))
plt.title("Sample plots", fontsize=15)
plt.plot(sample_a, "r-", label = "$\sigma_q = 0.1, \sigma_y = 1$")
plt.plot(sample_b, "b-", label = "$\sigma_q = 0.01, \sigma_y = 1$")
plt.plot(sample_c, "g-", label = "$\sigma_q = 0.1, \sigma_y = 0.1$")
plt.legend(loc="upper right")
plt.show()
```

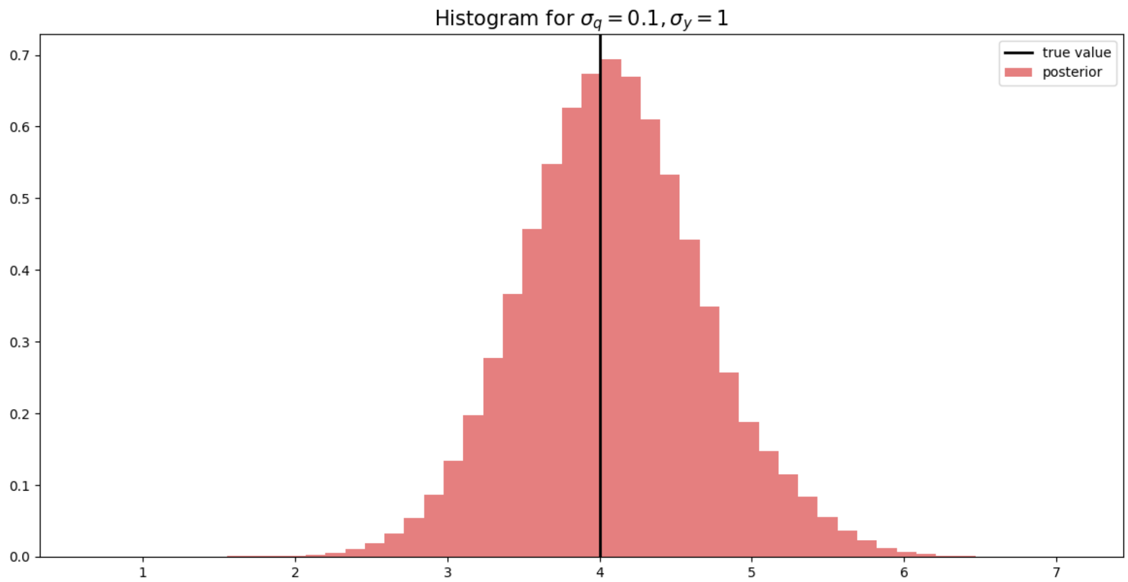

Figure 4

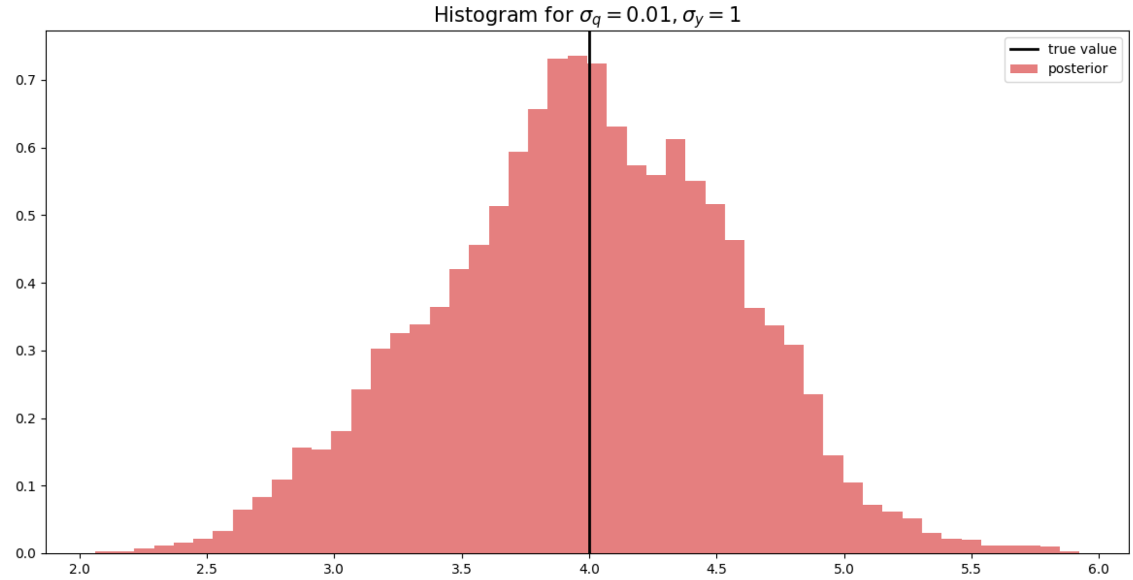

Figure 5

From Figure 7 we can see that the `burnin` period for $\sigma_q = 0.1$ (red), is a lot shorter than the `burnin` period for $\sigma_q = 0.01$ (blue).
The posteriors for both σ_q's appear to be Gaussian distributions centered around 4.

2.3

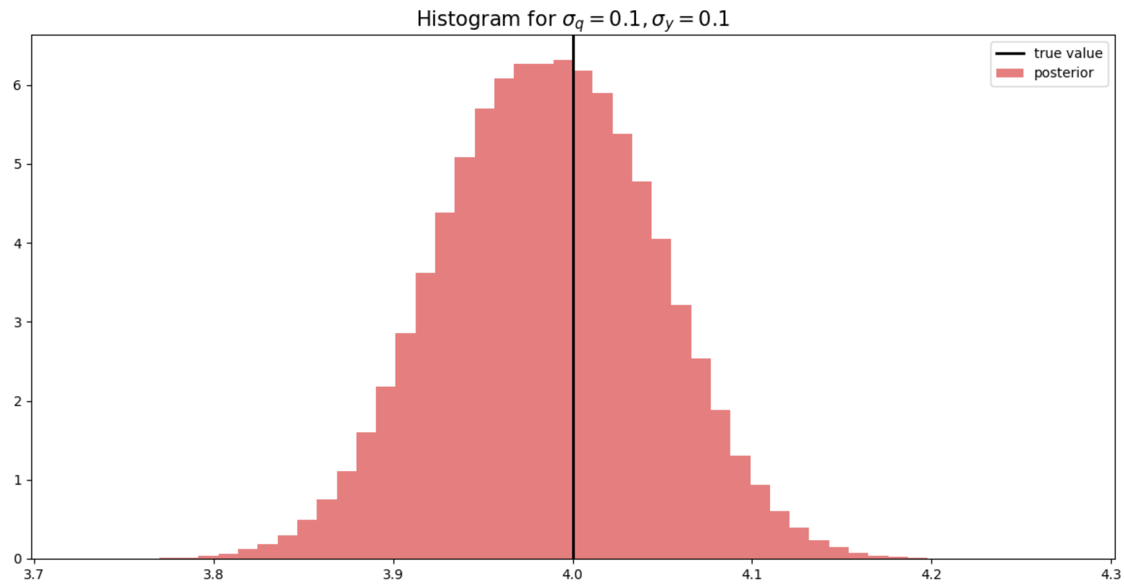

Figure 6

In Figure 6 we can see that again the posterior for $\sigma_y = 0.1$ appears to be Gaussian centered around 4.
We can see however that with this new σ_y and y_i data, we have more precise observations with less noise (variance), and so our posterior has also very small variance; it suggests our true value given the observations is around 4 with high probability.
In Figure 7 we can see the samples corresponding to $\sigma_y = 0.1$(green). We can see these samples deviate very little from 4 compared to the previous noisier observations we used.

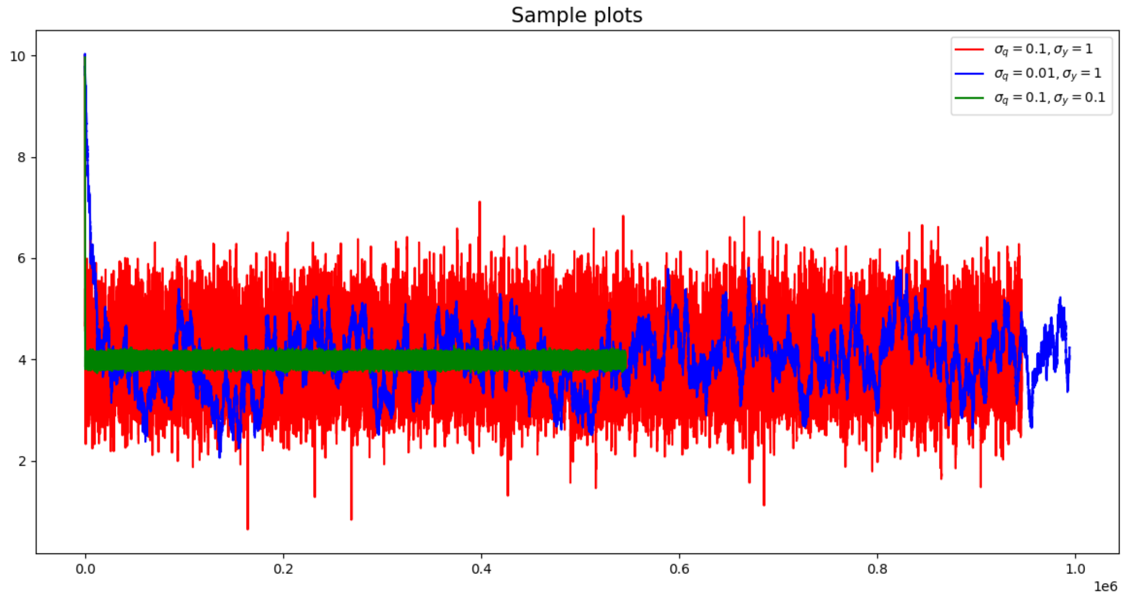

Figure 7