Jeremy Marcus Tan
Math 71.4: Scientific Computing II
Problem Set 4
April 28, 2024

1. Evaluate
$$\int_0^1 \int_0^\pi \frac{32xy\sin(y)}{(1+x^4)(1+cos^2(y))}\,dy\,dx$$
   (a) via Monte Carlo integration using two independent uniform random variables, using $N = 1000$.
   (b) via Monte Carlo integration using two independent random variables
   $$X \sim U(0,1)$$
   and
   $$Y \sim N(\pi/2, 1)$$
   using $N = 1000$.
   (c) Determine the empirical relative efficiency of the two estimators (the ratio of the variances of the estimates), using your results in b and c. Which of the estimators is more efficient?

Solution.

(a) I used the basic Monte Carlo Integration method on two dimensions to evaluate the integral. We have to consider the bounds of the integrals: 0,1 and 0,π.
Let
$$f(x,y) = \frac{32xy\sin(y)}{(1+x^4)(1+cos^2(y))}.$$
   i. Generate uniform random variables $u_1 \sim U(0,1)$ and $u_2 \sim U(0,\pi)$.
   ii. Return $\pi f(u_1, u_2)$.

The implementation was done on the attached Jupyter Notebook.

(b) For this item, I used the Monte Carlo Integration Method with weighted samples. We take two random variates, $x \sim U(0,1)$ and $y \sim N(\pi/2, 1)$.
Since y could have values outside the support of the pdf, we define the function
$$f(x,y) = \begin{cases} \frac{32xy\sin(y)}{(1+x^4)(1+cos^2(y))} & y \in [0,\pi] \\ 0 & y \notin [0,\pi] \end{cases}$$
Since $y \sim N(\pi/2, 1)$, it has pdf
$$g(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y-\frac{\pi}{2})^2}$$
In our simulation, we do the following:
   i. Generate $x \sim U(0,1)$ and $y \sim N(\pi/2, 1)$.
   ii. If $y \in [0,\pi]$, return $f(x,y)/g(y)$, otherwise, return 0.

(c) We run the simulation $N = 1000$ times.
We get the following results:

Method	Sample Mean	Variance
Basic	31.2947	830.7767
Weighted	31.2692	666.1347

The relative efficiency of the basic Monte Carlo estimator (T_1) and the weighted Monte Carlo estimator (T_2) is

$$e(T_1, T_2) = \frac{\text{var}(T_2)}{\text{var}(T_1)} = \frac{666.1347}{830.7767} = 0.8018$$

Since $e(T_1, T_2) < 1$, the weighted Monte Carlo method is more efficient as it has a lower variance.

2. We now define a Markov chain $X_0, X_1, ...$ with state space $E = \{x_0, x_1, ..., x_n\}$ whose stationary distribution is that of X.

Suppose we have initial state $X_0 = x_0$. To generate the next state:

- Given the current state $X_t = i$, propose the next state j according to a proposal function $g(j|i)$. (this means $g(j|i)$ is a distribution on E and we draw j from the distribution $g(j|i)$).
- Let $\alpha_{ij} = \min(1, \frac{f(j)g(i|j)}{f(i)g(j|i)})$.
- With probability α_{ij}, let $X_{t+1} = j$ (accept the proposed state), and with probability $1 - \alpha_{ij}$, let $X_{t+1} = i$ (reject the proposed state).

(a) Suppose $i \neq j$. Based on the algorithm presented above, explain why

$$p_{ij} = g(j|i)\alpha_{ij} \text{ and } p_{ji} = g(i|j)\alpha_{ji}.$$

(b) Assume $i \neq j$. Explain why choosing $\alpha_{ij} = \min(1, \frac{f(j)g(i|j)}{f(i)g(j|i)})$ satisfies the condition

$$\frac{\alpha_{ij}}{\alpha_{ji}} = \frac{f(j)g(i|j)}{f(i)g(j|i)}, \text{ for all } i, j \in E$$

(c) Let $i, j \in \mathbb{R}$. If we let g_i be the pdf of $N(i, \sigma)$ and let g_j be the pdf of $N(j, \sigma)$, show that

$$g_i(j) = g_j(i)$$

(d) Implement the Metropolis-Hastings algorithm and use it to generate a single realization of the Markov chain $X_0, X_1, ...$ until time $T = 10000$. Then use the python code to plot the generated samples alongside the target pdf.

<u>**Solution**</u>.

(a) The transition probability p_{ij} is the probability that the next state will be j given that the current state is i.

The proposal function g gives us the probability that a state will be proposed:

$$g(j|i) = P(\text{state } j \text{ is proposed if we are currently at state } i)$$

From the algorithm, α_{ij} is the probability that the algorithm will accept the proposed state j .

Thus, we multiply the probability that the algorithm will propose state j ($g(j|i)$) by the probability that the algorithm will accept the proposed state j (α_{ij})) to get the transition probability p_{ij}.

$$p_{ij} = g(j|i)\alpha_{ij}.$$

By the same argument, if the current state is j, then we multiply $g(i|j)$ and α_{ji} to get

$$p_{ji} = g(i|j)\alpha_{ji}.$$

(b) We show that if $\alpha_{ij} < 1$, then $\alpha_{ji} = 1$, and vice-versa.

Suppose that $\alpha_{ij} < 1$.

Then, $\alpha_{ij} = \frac{f(j)g(i|j)}{f(i)g(j|i)} < 1$.

Thus, the reciprocal, $\frac{f(i)g(j|i)}{f(j)g(i|j)} > 1$. So

$$\alpha_{ji} = \min(1, \frac{f(i)g(j|i)}{f(j)g(i|j)}) = 1$$

Thus, for this case ($\alpha_{ij} < 1$), we have

$$\frac{\alpha_{ij}}{\alpha_{ji}} = \frac{\frac{f(j)g(i|j)}{f(i)g(j|i)}}{1} = \frac{f(j)g(i|j)}{f(i)g(j|i)}.$$

Suppose that $\alpha_{ji} < 1$.

By similar reasoning to the first case, we get that $\frac{f(j)g(i|j)}{f(i)g(j|i)} > 1$. Therefore, $\alpha_{ij} = 1$.

For this case ($\alpha_{ji} < 1$), we have

$$\frac{\alpha_{ij}}{\alpha_{ji}} = \frac{1}{\frac{f(i)g(j|i)}{f(j)g(i|j)}} = \frac{f(j)g(i|j)}{f(i)g(j|i)}.$$

(c) Since g_i and g_j are normally distributed, the variables have pdfs:

$$g_i(j) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{j-i}{\sigma})^2}$$

$$g_j(i) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{i-j}{\sigma})^2}$$

The functions only differ in the power of the exponential.

Since $(j-i)^2 = j^2 - 2ij + i^2 = (i-j)^2$,

$$(\frac{j-i}{\sigma})^2 = (\frac{i-j}{\sigma})^2$$

Thus, $g_i(j) = g_j(i)$.

(d) The implementation of the Metropolis-Hastings algorithm is in the attached jupyter notebook.

Code Explanation.

- The algorithm will continue running until the length of the states list reaches 10,001, corresponding to $T = 10000$.

- The proposed state j was selected from random.normalvariate(i,1), where i is the current state.
- The probability of selecting the proposed state α_{ij} was computed using the min function.
- Lastly, a random number was generated uniformly from (0,1), and then compared against α_{ij} to determine if the algorithm would accept or reject the proposed state j.

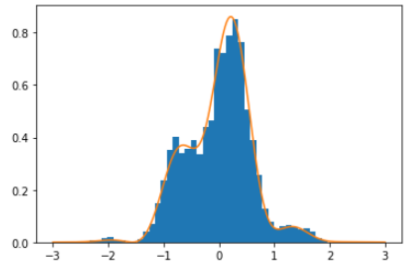

Figure 1: Histogram of the Sample States and the target pdf

Figure 1 shows that the samples generated by the Metropolis-Hastings algorithm match the target pdf.