STAT 515
Homework #9 WITH SOLUTIONS

1. Customers arrive at a post office at a Poisson rate of 8 per hour. There is a single person serving customers, and service times are exponentially distributed (and independent) with mean 5 minutes. Suppose that an arriving customer will decide to wait in line if and only if there are three or fewer people already in the post office (which means two or fewer people in line). Of interest is $E(L)$, the expected total number of potential customers lost, during a single eight-hour day.

   (a) Simulate this eight-hour process 10,000 times, then find the sample mean

   $$\hat{\mu}_1 = \frac{1}{10^4} \sum_{i=1}^{10^4} L_i,$$

   where L_i is the number lost in the ith simulation.

   Solution: I'll adapt code from the solution to Homework #7 for this purpose:

```
> R <- matrix(c(-8, 12, 0, 0, 0, 8, -20, 12, 0, 0, 0, 8, -20,
+               12, 0, 0, 0, 8, -20, 12, 0, 0, 0, 8, -12),
+          5, 5)  # Set up the transition matrix
> T <- rep(0, 10000) # This is where we'll store time in state 5
> L <- rep(0, 10000) # This is where we'll store number of lost customers
> maxTime <- 8 # This is the cutoff time.
> for (count in 1:10000) {
+   currentState <- 1
+   currentTime <- 0
+   finished <- FALSE # We'll set this to TRUE when it's time to stop.
+   while (!finished) {
+ #    currentState <- states[i]
+ #    currentTime <- times[i]
+     deltaTime <- rexp(1, rate = -R[currentState, currentState])
+     if (currentTime + deltaTime > maxTime) {
+       # Now we need to finish this chain
+       deltaTime <- maxTime - currentTime
+       finished <- TRUE
+     }
+     if (currentState == 5) {
+       T[count] <- T[count] + deltaTime
+       L[count] <- L[count] + rpois(1, 8*deltaTime)
+     }
+     currentTime <- currentTime + deltaTime
+     possibleMoves <- (1:5)[-currentState]
+     currentState <- sample(possibleMoves, 1, prob=R[currentState,possibleMoves])
+   }
+ }
```

   Now we can calculate the sample mean of the L_i:

```
> mean(L)
[1] 4.5894
```

   (b) Prove that $E(L) = 8E(T)$, where T is the total amount of time spent in the state "four people in the post office". (Hint: Use conditioning.)

   Solution: Whenever in that state, all customer arrivals are losses. which means that conditional on T, L is Poisson($8T$). Therefore,

   $$E(L) = E[E(L \mid T)] = E[8T] = 8E(T).$$

(c) For the simulation in part (a), calculate

$$\hat{\mu}_2 = \frac{8}{10^4} \sum_{i=1}^{10^4} T_i,$$

where T_i is the total time spent in the four-people state in the ith simulation.

Solution: Since we kept track, we can simply find the sample mean of the T_i and multiply by 8:

```
> 8 * mean(T)
[1] 4.603994
```

Both here and in part (a), we should really report estimates of precision as well. We'll skip that here, but at least the variances are reported in part (e).

(d) Both μ_1 and μ_2 are estimators of the same quantity, so which one is better? In a statistical sense, "better" often means "smaller variance". Use the conditional variance formula to prove that

$$\text{Var } \hat{\mu}_2 < \text{Var } \hat{\mu}_1.$$

Solution: It suffices to show that $\text{Var } L > \text{Var}\,(8T)$. This follows immediately from

$$\text{Var } L = \text{E}\left[\text{Var}\,(L \mid T)\right] + \text{Var}\left[E(L \mid T)\right] = E[8T] + \text{Var}\,(8T) > \text{Var}\,(8T).$$

(e) Calculate two separate 95% confidence intervals for μ based on $\hat{\mu}_1$ and $\hat{\mu}_2$. Do your results agree with the finding of part (d)?

Solution: The comparison below shows that the sample variances agree with the theory:

```
> c(var(L), var(8*T))
[1] 13.80799  9.24100
```

2. Describe two different algorithms for simulating from a beta$(3, 2)$ density, one based on an inversion method and the other based on a rejection method. Try simulating the same number of variables (something more than a million) using each method. Does one method appear to be more efficient than the other? Explain every step. If you are using R, you might find the timing function `system.time` useful if you want to compare the algorithms based on their total time.

As a check on your simulated values, make sure that the sample mean and variance of your variables are close to the theoretical mean and variance of 3/5 and 1/25.

Solution: The beta$(3, 2)$ density equals $cx^2(1-x)$ for $x \in (0, 1)$, and therefore the cdf is $F(x) = cx^3/3 - cx^4/4$, which implies that $c = 12$. Therefore, $F(x) = 4x^3 - 3x^4$ for $x \in (0, 1)$. The inversion method involves simulating $U \sim \text{unit}(0, 1)$ and then taking X to be the unique value in $(0, 1)$ such that $F(X) = U$.

The rejection method simulates $X \sim \text{unit}(0, 1)$ and then accepts X with probability $f(X)/K$, where K is chosen to be the smallest value such that $K \geq f(x)$ for $x \in (0, 1)$. Since the maximum of $f(x)$ is 16/9, we take $K = 16/9$. Incidentally, this choice of K means that on average, 9/16 of the proposed X values will be accepted. Thus, if we use 2 million proposals, we should have more than a million accepted.

Here is some code that tests the rejection method (with timing). We'll do this method first so that we can generate exactly the same number of variables with the inversion method.

```
> system.time({
+   x1 <- runif(2e6)
+   u <- runif(2e6)
+   x1 <- x1[u < 12*x1^2*(1-x1)]
+   nAccepted <- length(x1)
+ })
   user  system elapsed
  0.189   0.056   2.967
```

Next, we test the inversion method using a numerical inversion:

```
> F <- function(x) 4*x^3-3*x^4 # Set up F and its numerical inverse function:
> FminusU <- function(x, u) F(x) - u
> Finv <- function(u) uniroot(FminusU, interval=c(0,1), u=u)$root
> system.time({  # Now run the test:
+   u <- runif(nAccepted)
+   x2 <- sapply(u, Finv)
+ })
   user  system elapsed
 84.871   0.139  85.156
```

Although it is not required, we can also test the third method we discussed in class:

```
> system.time({
+   u <- matrix(runif(5*nAccepted), ncol=5)
+   x3 <- apply(u, 1, function(a) log(prod(a[1:3]))/log(prod(a)))
+ })
   user  system elapsed
 11.257   0.351  16.393
```

Finally, a sanity check to make sure that the means and variances look close to the true values:

```
> c(mean(x1), mean(x2), mean(x3))
[1] 0.5831477 0.5997402 0.6002727
> c(var(x1), var(x2), var(x3))
[1] 0.04865153 0.04001232 0.04000663
```

We see that the rejection method is by far the most efficient of the three methods, at least in terms of computing time.

3. Suppose we want to conduct a simple hypothesis test to see whether the correlation ρ between X and Y is greater than zero. Let us assume that X and Y come from a bivariate normal distribution, and a sample of 30 points $(X_1, Y_1), \ldots, (X_{30}, Y_{30})$ gives a sample correlation of $\hat{\rho} = 0.3$. We wish to find the p-value for this sample correlation.

   To find the p-value, we need to know the null distribution of the sample correlation. In this case, the exact null distribution (i.e., the distribution of $\hat{\rho}$ when $\rho = 0$) is known, but it is quite complicated. Instead, we shall approximate the p-value using the fact (not proven here) that the distribution of $\hat{\rho}$ depends only on ρ and the sample size n.

   The p-value in this problem is defined to be $P(\hat{\rho} > 0.3 \mid \rho = 0)$. Use 10^5 Monte Carlo samples of size 30 to obtain a 95% confidence interval for the p-value.

   Solution: Since the distribution of $\hat{\rho}$ only depends on the true ρ and the sample size, all we need to do is draw bivariate normal samples of size 30 from a distribution with true correlation zero:

```
> a <- array(rnorm(1e5 * 30* 2), c(1e5, 30, 2))
> f <- function(x) cor(x[,1], x[,2])
> rhoHat <- apply(a, 1, f)
```

   Now the p-value is just the sample proportion of $\hat{\rho}$ values larger than or equal to 0.3:

```
> pval <- mean(rhoHat >= 0.3)
> pval + c(-1.96, 1.96) * sqrt(pval * (1-pval) / 1e5)
[1] 0.05113738 0.05390262
```

   It appears that the p-value is just a bit larger than 0.05. It is possible to estimate the p-value to any desired precision using a larger simulation.

4. Suppose (X_1, X_2) are bivariate normal with $EX_1 = EX_2 = 0$ and covariance matrix

$$\Sigma = \begin{bmatrix} 4 & 4 \\ 4 & 9 \end{bmatrix}.$$

   Use a Monte Carlo method to estimate $P(\max\{|X_1|, |X_2|\} < 1)$ to such a precision that a 99% confidence interval for the true value has a width of no more than 1/1000.

Solution: In the worst-case, a 99% confidence interval has width equal to $2.58/\sqrt{n}$, so if we use a sample size larger than 6.7×10^6 then we'll ensure that the confidence interval is no wider than 1/1000:

```
> Sigma <- matrix(c(4, 4, 4, 9), 2, 2)
> e <- eigen (Sigma, symmetric=TRUE) # The algorithm is slightly more efficient
>                                    # if symmetric=TRUE is given.
> X <- e$vec %*% (t(e$vec) * sqrt(e$val))
> n <- 7e6
> Y <- X %*% matrix(rnorm(2*n), nrow=2)
> phat <- mean( abs(Y[1,]) < 1 & abs(Y[2,]) < 1)
> phat + c(-1,1) * qnorm(.995) * sqrt(phat * (1-phat) / n)
[1] 0.1282625 0.1289143
```

We may thus state with a high degree of certainty that the true probability is 0.128, correct to three decimal places.