Sirindhorn International Institute of Technology

Thammasat University at Rangsit

School of Information, Computer and Communication Technology

ET601: Problem Set 6 Solution

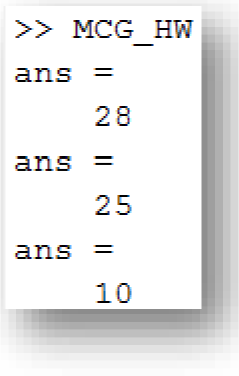
Due date: February 27, 2013 (Thursday), 1 PM

1. Consider the Multiplicative Congruential Generator (MCG) with the recursion $x_n = ax_{n-1} \bmod m$. We have seen in class that when a = 3 and m = 7, the resulting sequence has maximal length (of the cycle).
   Here, we want to explore this property when the value of *a* is changed. We will consider a = 1, 2, 3,… , 100.
   Write a MATLAB script to answer the following questions.
   a. How many of the *a* above give maximal length sequence?
      Ans: 28.
   b. Among the hundred *a* values that we test, how many are prime numbers?
      Hint: The function `isprime` may be useful here.
      Ans: 25.
   c. Among the *a* values that give maximal length sequence, how many are prime numbers?
      Ans: 10.

   The MATLAB script is provided below along with the results displayed in the command window.

```
a = 3; m = 7;
x0 = 1; % Since we only check for maximal length cycle,
        % it does not matter where we start.
N = 100; % Consider a = 1, 2, ..., N
A = zeros(1,N); % for saving the period of each case
for a = 1:N
    x = zeros(1,m-1); % The maximum period is m-1, so we may look at only
                      % m-1 of these. If the sequence never repeats,
                      % the memebrs are unique.
    x(1) = x0;
    for k = 2:(m-1)
        x(k) = mod(a*x(k-1),m);
    end
    A(a) = length(unique(x));
```

```
end
Period = A;
MaximalLengthI = (A == (m-1)); % Find the a that gives maximal length
PrmI = isprime(1:N); % Find the a that is a prime number.
sum(MaximalLengthI) % (a)
sum(PrmI) % (b)
sum(MaximalLengthI.*PrmI) % (c)
```



```
>> MCG_HW
ans =
    28
ans =
    25
ans =
    10
```

2. In class we have seen the script which creates n observations (realizations) sampled from the pmf

$$p_X(x) = \begin{cases} 1/6, & x = 3, \\ 1/3, & x = 4, \\ 1/2, & x = 8, \\ 0, & \text{otherwise.} \end{cases}$$

The script is shown in the box below.

   a. What is the relationship between the `dum` variable and `hist(X,S_X)` a few lines below?
   b. Replace the expression "`hist(X,S_X)/n;`" in the code by another expression which utilize the `dum` variable instead of `hist(X,S_X)`.

```
Clear all; close all;

S_X = [3 4 8];
p_X = [1/6 1/3 1/2];

n = 1e6;

F_X = cumsum(p_X);
U = rand(1,n);
[dum,V] = histc(U,[0 F_X/F_X(end)]);
X = S_X(V);

rf = hist(X,S_X)/n;
stem(S_X,rf,'rx')
hold on
stem(S_X,p_X,'bo')
```

```
xlim([min(S_X)-1,max(S_X)+1])
legend('Rel. freq. from sim.'...
    ,'pmf p_X(x)')
xlabel('x')
grid on
```

Solution:

a. They are almost the same.
   Recall that both `hist` and `histc` count the number of entries whose values falls inside the bins. In this case, the *k*th bin corresponds to the k^{th} value in the support of the random variable. When the RV *U* falls into the *k*th bin, the RV *X* takes the value x_k.
   The bins from `histc` also have an extra bin at the right edge, this give us one more bin than the `hist` command. This extra bin will not have any entry because the rand command never produces number 1.
   Therefore,

   `dum = [hist(X,S_X) 0]`

   or

   `hist(X,S_X) = dum(1:(end-1)).`

b. Usually, when we try to generate the RV *X*, the `dum` variable is not used. However, here, we also want to plot the histogram of *X* to check that it actually gives the correct pmf.
   In the script provided, the histogram is generated directly by the usual `hist` command. However, the counting has already been performed inside the `dum` variable already. We simply need to eliminate the extra bin.
   To do this, we simply replace `hist(X,S_X)` by `dum(1:(end-1))`.

3. In class we have seen the script GenRV_Geo_while.m which creates *n* observations (realizations) sampled from the geometric$_1$ pmf with parameter *p*. The script is shown in the table below. Instead of using the `histc` command to find the bin index that the uniformly distributed on (0,1) falls into, this implementation is based on the use of `while` loop to successively construct the boundaries of the bins. The script also verifies the generated values by comparing the relative frequencies with the theoretical pmf.

   Modify the script so that it creates *n* observations (realizations) sampled from the Poisson pmf with parameter α.

```
p = 1/3; n = 1e6;
U = rand(1,n);
V = zeros(size(U)); %Preallocation
for i = 1:n % Consider individual element in the U vector
    k = 1; % The first bin. k is the bin index
    pk = p; F = p; % Boundary of the first bin
    while (U(i) > F) % Check whether U(i) is beyond
                     % the current bin
        k = k+1; % Consider the next bin
        pk = pk*(1-p); % Width of the next bin
        F = F + pk; % Boundary of the next bin
    end
    V(i) = k;
end
X = V;
```

```
X_Range = 1:10; % Only plots the first 10 values
p_X = p*(1-p).^(X_Range-1); % The theoretical pmf
rf = hist(X,X_Range)/n;
stem(X_Range,rf,'rx')
hold on
stem(X_Range,p_X,'bo')
xlim([min(X_Range)-1,max(X_Range)+1])
legend('Rel. freq. from sim.'...
    ,'pmf p_X(x)')
xlabel('x')
grid on
```

The figure below show the expected resulting plots when $n = 10^6$ and $\alpha = 2$.

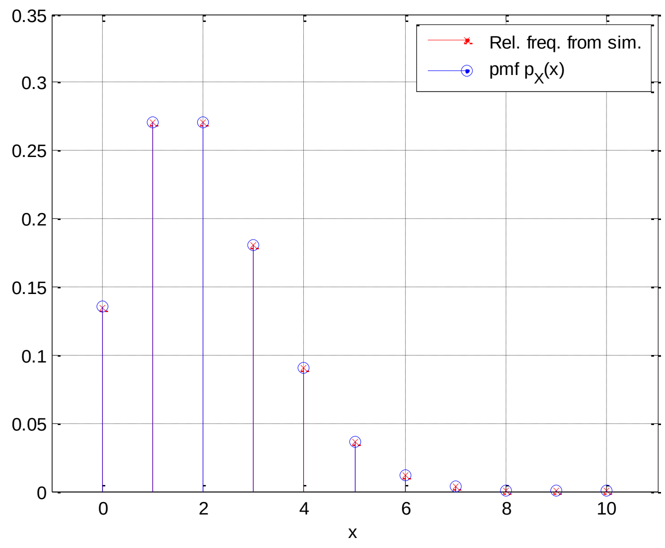


Figure 1

Solution:

For Poisson RV, $\dfrac{p_X(k)}{p_X(k-1)} = \dfrac{e^{-\alpha}\dfrac{\alpha^k}{k!}}{e^{-\alpha}\dfrac{\alpha^{k-1}}{(k-1)!}} = \dfrac{\alpha}{k}$.

The MATLAB script is provided below. Note that the support for Poisson RV also includes 0.

```
clear all; close all;

% Generate n observations (realizations) sampled
% from the Poisson pmf with parameter alpha.
% Instead of using the histc command to find
% the bin index that the uniformly distributed
% on (0,1) falls into, the implementation is based on
% the use of while loop to successively construct the
% boundaries of the bins.
% The script also verifies the generated values by comparing
% the relatives frequencies with the theoretical pmf.

alpha = 2; n = 1e6;

U = rand(1,n);
V = zeros(size(U)); %Preallocation
for i = 1:n % Consider individual element in the U vector
    k = 0; % The first bin. k is the bin index
    pk = exp(-alpha); F = pk; % Boundary of the first bin
    while (U(i) > F) % Check whether U(i) is beyond
                     % the current bin
        k = k+1; % Consider the next bin
        pk = pk*alpha/k; % Width of the next bin
        F = F + pk; % Boundary of the next bin
    end
    V(i) = k;
end
X = V;

X_Range = 0:10; % Only plots the first 10 values
p_X = exp(-alpha)*(alpha.^(X_Range))./factorial(X_Range); % The
theoretical pmf
rf = hist(X,X_Range)/n;
stem(X_Range,rf,'rx')
hold on
stem(X_Range,p_X,'bo')

legend('Rel. freq. from sim.'...
    ,'pmf p_X(x)')
xlabel('x')
grid on
```