HW2, name: Jorge Monzon Diaz, email:jmonzon000@citymail.cuny.edu

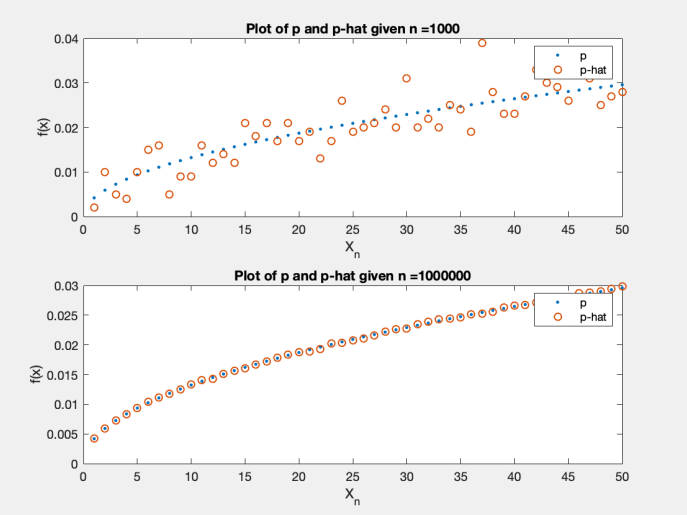
1)

a)

$$1 = c * sqrt(1) + c * sqrt(2) + \cdots + c * sqrt(50)$$

$$c = \frac{1}{\sum_{x=1}^{50} \sqrt{x}} = 0.0042$$

b) My graph shows that p-hat oscillates around the values of p meaning that the frequency of falling into each of 50 partitions of p is close to c*p(x) for any given x. This convergence becomes even greater as the sample size increases as shown by the direct relationship in the second graph of n=1e6.

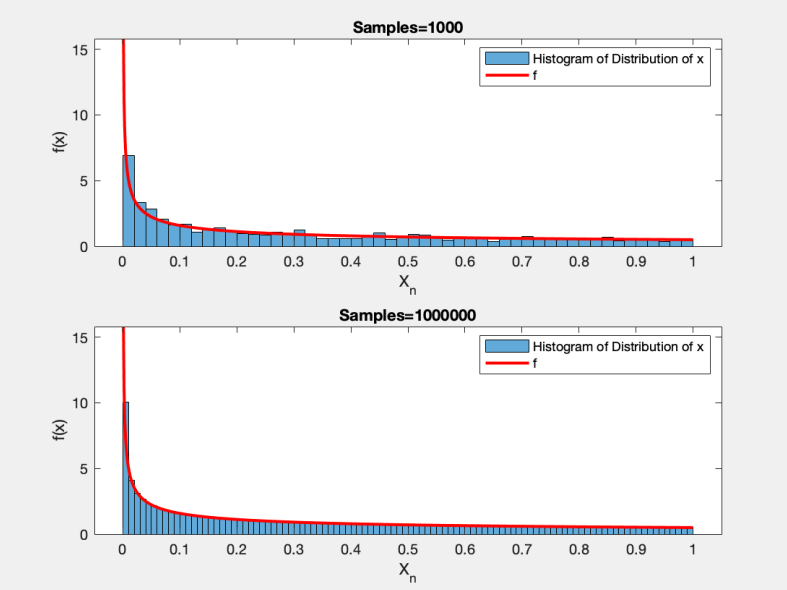


2) See attached

3)

a) See attached

b) This graph shows the correct relationship between graphing the pdf f(x) and sampling x as u^2 and creating a histogram, they are equivalent.

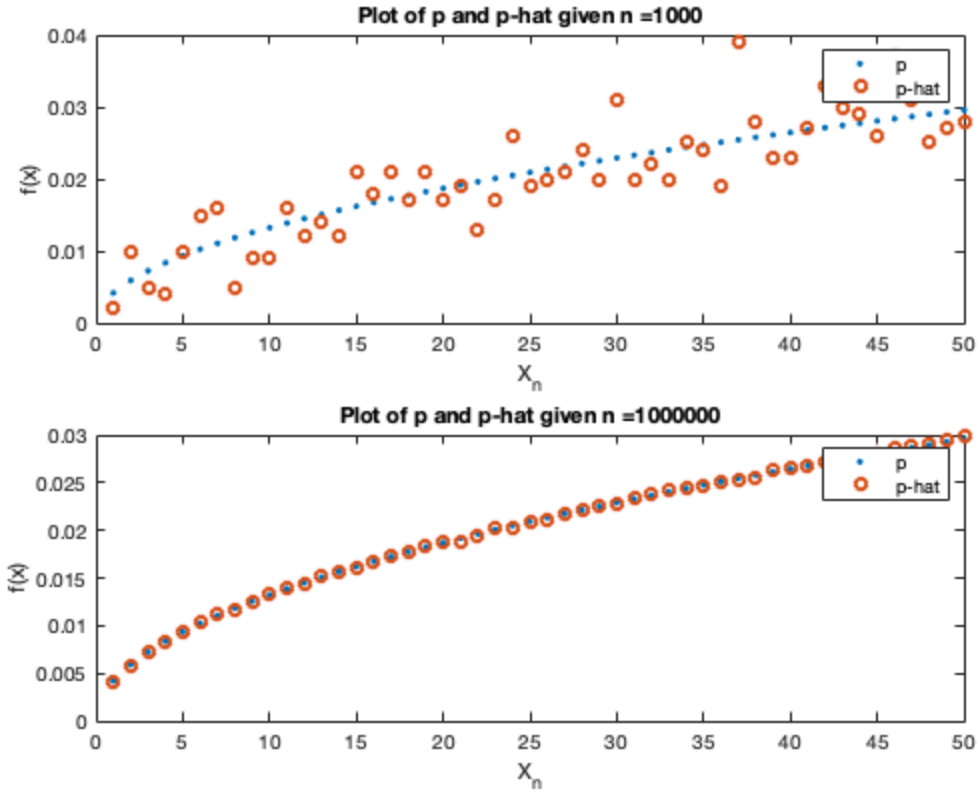


HW2, name: Jorge Monzon Diaz, email:jmonzon000@citymail.cuny.edu

Table of Contents

Problem 1

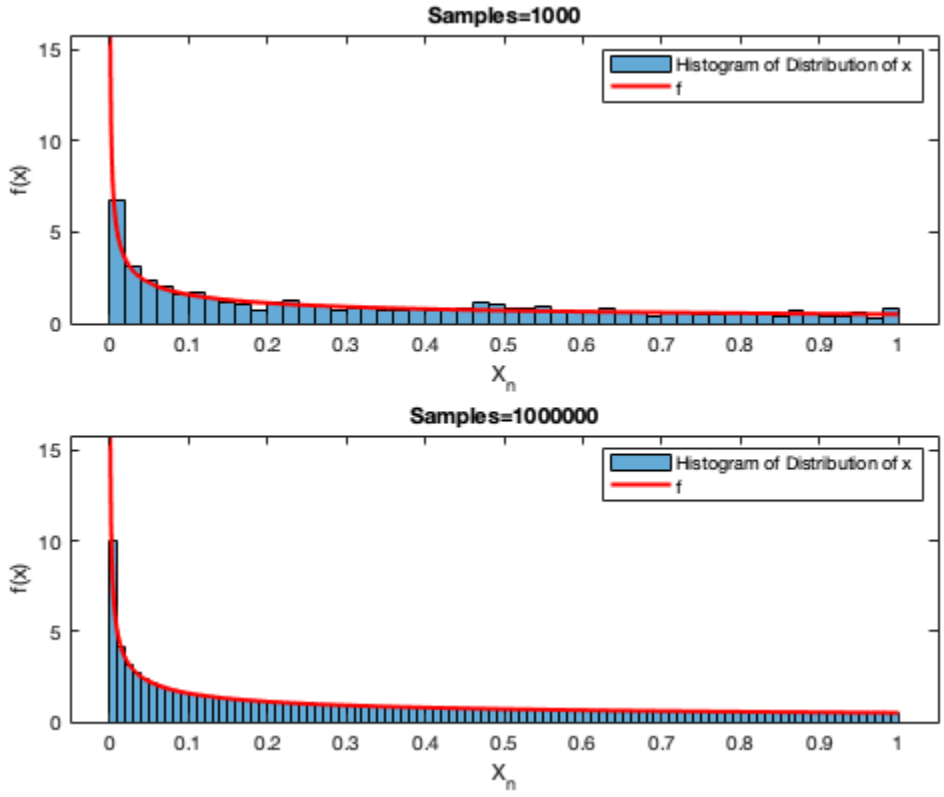
```
close all
clear all
hold on
figure(1)

c=1/sum(sqrt(1:50)); %solve for c

n=[1e3,1e6]; %store number of samples
p=c*sqrt(1:50); %calculate p

for i=1:2 %find and graph for p and p-hat for both sample sizes n1,
 and n2
    [phat]=histc(drand(1,n(i),p),1:50); %make histogram using samples
 computed by drand
    phat=phat/n(i); %normalize data
    subplot(2,1,i)
    plot(1:50,p,'.',1:50,phat,'o')
    title(strcat('Plot of p and p-hat given n = ',int2str(n(i))))
    legend('p','p-hat')
    xlabel('X_n')
    ylabel('f(x)')
end
```



Problem 3

```
figure(2)
clear all


n=[1e3,1e6]; %n values
x=linspace(0,1,1000); %make x-values for line plot
y=1/2.*1./sqrt(x); %make y-values for line plot

x1=rand(1,n(1)); x2=rand(1,n(2)); %generate random numbers


f=x1.^2; %x as u^2
f2=x2.^2;
axis equal
subplot(2,1,1)
histogram(f,50,'normalization','pdf') %histogram of samples
hold on
plot(x,y, 'LineWidth', 2, 'Color', 'r') %plot exact f(x)
title(strcat('Samples=',int2str(n(1))))
legend('Histogram of Distribution of x','f')
xlabel('X_n')
ylabel('f(x)')
```

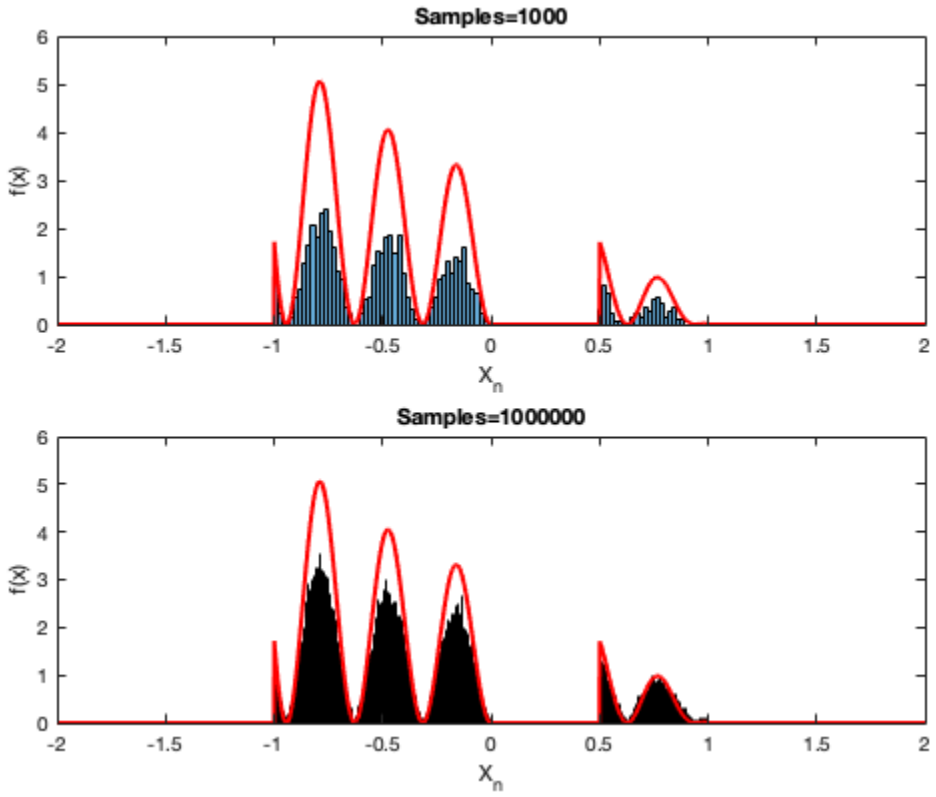
```
subplot(2,1,2)
histogram(f2,'normalization','pdf') %histogram of samples
hold on
plot(x,y, 'LineWidth', 2, 'Color', 'r') %plot exact f(x)
title(strcat('Samples=',int2str(n(2))))
legend('Histogram of Distribution of x','f')
xlabel('X_n')
ylabel('f(x)')
```



Problem 4

```
figure(3)
clear all

%store sample sizes
n=[1e3,1e6];

for i=1:length(n)
    m=max(hw2_fn_g(linspace(-1,1,n(i)))); %compute c

    x = zeros(1,n(i)); % preallocate memory
    k=1;
    while k<=n(i) %pull sample-size amount of samples
        u = 2.*rand-1; %compute u1
        v = m.*rand; %compute u2
```

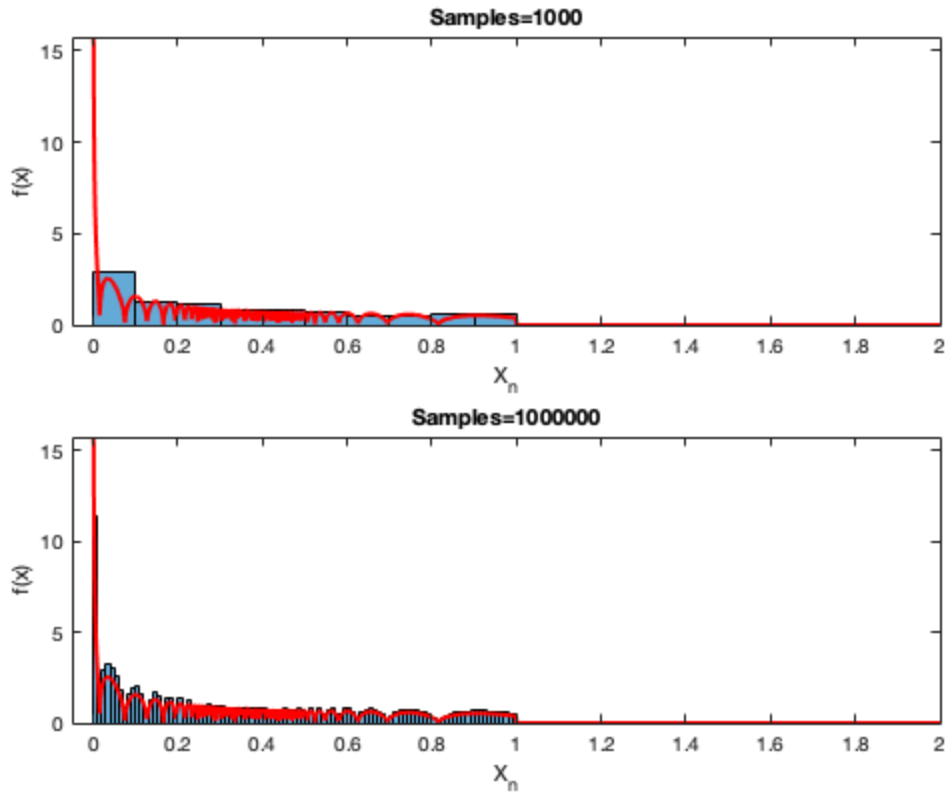
```
        if v <= hw2_fn_g(u) %if u2<=f(u1) store u1
            x(k) = u;
            k=k+1;
        end
    end


    subplot(2,1,i)
    histogram(x,n(i)./10,'normalization','pdf') %histogram of u1
    hold on
    plot( -2:.001:2, hw2_fn_g(-2:.001:2), 'LineWidth', 2, 'Color', 'r'
 )
    title(strcat('Samples=',int2str(n(i))))
    xlabel('X_n')
    ylabel('f(x)')

end
```



Problem 5

```
figure(4)
clear all

%store sample sizes
n=[1e3,1e6];
```

```
for i=1:length(n)
    m=max(hw2_fn_q(linspace(0,1,n(i)))); %compute c

    x = zeros(1,n(i)); % preallocate memory
    k=1;
    while k<=n(i) %pull sample-size amount of samples
        u = rand; %compute u1
        v = m.*rand; %compute u2
        if v <= hw2_fn_q(u) %if u2<=f(u1) store u1
            x(k) = u;
            k=k+1;
        end
    end


    subplot(2,1,i)
    histogram(x,'normalization','pdf') %histogram of u1
    hold on
    plot( .001:.001:2, hw2_fn_q(.001:.001:2), 'LineWidth',
 2, 'Color', 'r' )
    title(strcat('Samples=',int2str(n(i))))
    xlabel('X_n')
    ylabel('f(x)')

end
```



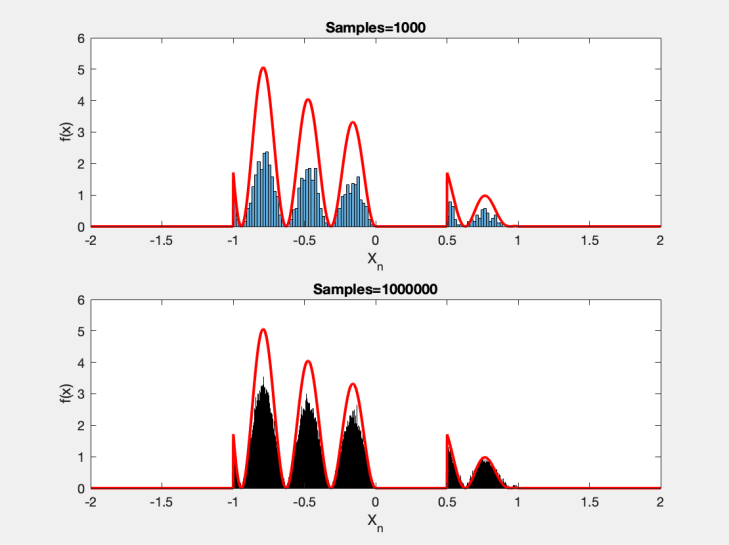
functions

```
function g=hw2_fn_g(x)
% The function g for HW #3.
% Note that the input x can be a vector
% For example, the following command
%   plot( -2:.001:2, hw3_fn_g(-2:.001:2) ) will plot the function g(x)
%       on the domain [-2,2]
    g=(sin(10*x)).^2 .* abs( x.^3 + 2.*x - 3 ) .* ( ( x>-1 & x<0 ) |
 ( x>1/2 & x<1 ) );
end

function q=hw2_fn_q(x)
% The function g for HW #3.
% Note that the input x can be a vector
% For example, the following command
%   plot( -2:.001:2, hw3_fn_g(-2:.001:2) ) will plot the function g(x)
%       on the domain [-2,2]
    q=1./(2.*sqrt(x)) .* sqrt(abs(sin(10./(1+log(abs(x)))))) .*
 ( ( x>0 & x<1 ) );
end

%given a finite pmf (p) normalized to 1, return an mxn array computing
%which partition of p that mxn would fall in
function x=drand(m,n,p)
    x=zeros(m,n); % preallocate memory
    for i=1:m
        for j=1:n
            u=rand;
            f=0; %initialize partitions
            for k=1:length(p) %you can have as many partitions as
 values for p
                f=f+p(k); %add p(k)
                if u<=f %if u falls between 0-p_n(k) store n or add
 more p(k) until it does
                    break
                end
            end
            x(i,j)=k; %store which partition (p_n) that u fell in
        end
    end
    return
end
```

Published with MATLAB® R2018b

4)

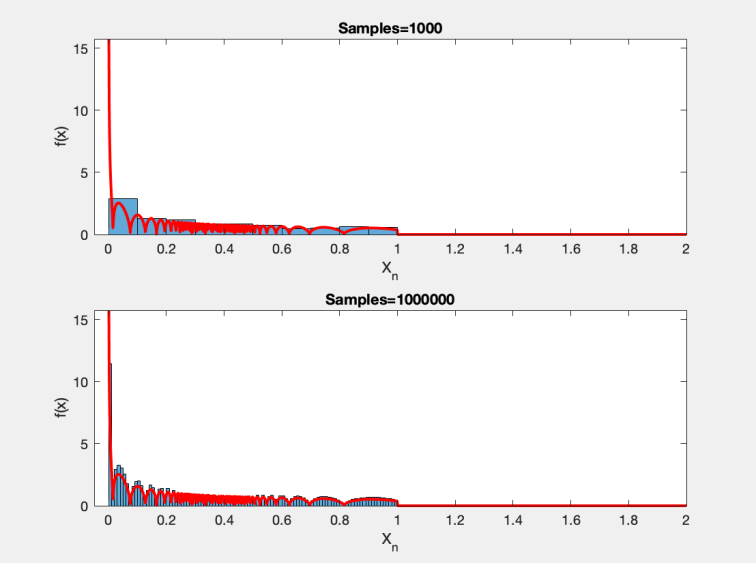
a) The accept/rejection method works as specified, it doesn't completely fill in the parabolic shapes because the probability of filling out that large area being smaller than that of completely filling in smaller regions. One could increase the height of the bars by decreasing bin-width to fit the data better, but then the bars would surpass the line in other spots. A bin-width of sample-size/10 gives a nice visualization of what the data is doing. The histogram of this data shows that randomly sampled values of f(x) converge to the shape of the distribution of g(x).



b) c=1/5.0626

5)

a) This graph successfully showcases the accept/reject method, the code just runs very slow because it is very resource intensive to run this complicated function over a million times until you generate 1e3+1e6 valid samples for both graphs combined. My initial code was vectorized so it only took a couple seconds to run, but I noticed it wasn't pulling enough samples as I was not actually obtaining 1e3 or 1e6 samples but rather just running 1e3 or 1e6 tests because of the rejection rate.



b) c=1/419.3731