Computational Statistics
M.Sc. Mathematical Engineering
Prof. A. Manzoni

A.A. 2021 – 2022
Politecnico di Milano

Hands-On 2
Random Variable Generation - 2

Problem 2.1 (Discrete inversion method)
We want to sample a discrete random variable X taking values $x_1 < x_2 < \ldots < x_n$ with probability

$$P(X = x_i) = p_i, \qquad \sum_{i=1}^{n} p_i = 1.$$

Let us denote by

$$F_i = \sum_{j=1}^{i} p_j = P(X \leq x_i);$$

1. Write a `Matlab` function which, given an integer $n > 0$ and the vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{p} = (p_1, p_2, \ldots, p_n)$, returns the vector $\mathbf{y}$ containing n samples from the discrete variable X.
2. Consider the random variable X such that

$$P(X = 1/5) = 0.1, \quad P(X = 1/4) = 0.4, \quad P(X = 1/3) = 0.05,$$

$$P(X = 1/2) = 0.25, \quad P(X = 1) = 0.2.$$

Draw 1000 samples from this variable and verify that the sampling strategy is correct, comparing sample mean and variance and the empirical CDF with the theoretical counterparts

1. A simple implementation is the following one:

```
%   Y=discrete(x,p,n)
%   x: array of possible values
%   p: probabilities associated to the values in x
%   n: number of samples to be drawn

function Y=discrete(x,p,n)
   Y = zeros(1,n);
   F = cumsum(p);
   U = rand(1,n);
   for i=1:n
       j=1;
       while U(i)>F(j)
           j=j+1;
       end
       Y(i)=x(j);
   end
return
```

An alternative implementation can be obtained using the command[1] `histc`:

```
function Y=discrete(x,p,n)
   U = rand(1,n);
   [N,Z] = histcounts(U, [ -inf , cumsum(p)]);
   Y = x(Z);
return
```

[1] `N = histc(X,EDGES)`, for vector `X`, counts the number of values in `X` that fall between the elements in the `EDGES` vector (which must contain monotonically non-decreasing values).

To check that the algorithm is correct, we can use the following commands (see Figure 1):

```
x=[1/5, 1/4, 1/3, 1/2, 1];
p=[.1, .4, .05, .25, .2];
n=1000;
Y = discrete(x,p,n);
mean_exact = x*p'
   mean_exact =    0.4617

mean_sample = mean(Y)
   mean_sample =   0.4635

var_exact = x.^2*p' - mean_exact^2
   var_exact =     0.0839

var_sample = var(Y)
   var_sample =    0.0866

figure(1); N=hist(Y,x); bar(x,N/n); %histogram
figure(2); cdfplot(Y);   % empirical distribution
```

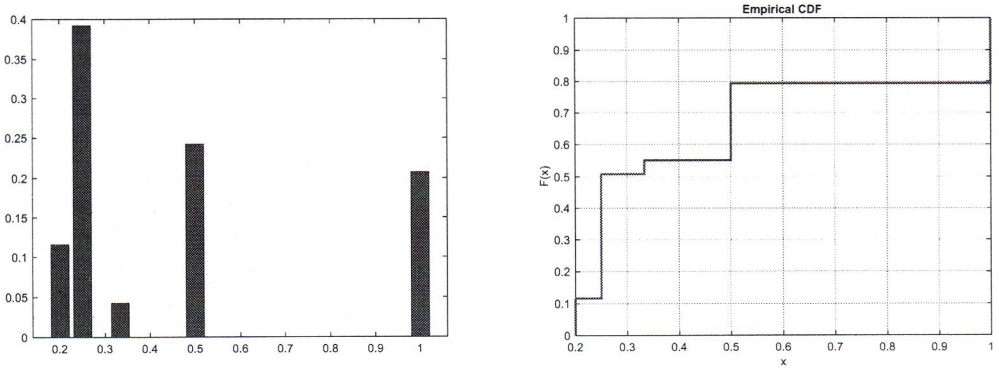


Figure 1: Histogram (left) and empirical CDF (right) of X for 10^3 values sampled from X. Sampled values are indeed very close to the exact ones.

Problem 2.2 (Binomial variable)
We want to sample from $X \sim Bin(m, p)$ for which

$$p_i = P(X = i) = \binom{m}{i} p^i (1-p)^{m-i}, \qquad i = 0, 1, \ldots, m$$

using the discrete inversion method.

1. Find a recursive relation linking p_{i+1} to p_i, $= 0, 1, \ldots, m$.
2. Exploit the relation above to implement in `Matlab` a function which samples a Binomial variable using the discrete inversion method.
3. Take $m = 5$, $p = 0.3$ and verify that the sampling is correct by comparing the empirical and the exact CDFs (for the exact CDF, use the `binocdf` command of the Statistical toolbox).
4. Recalling that $P(X = j)$ is the probability of having j successess over m independent trials, where the probability of success is p, write a `Matlab` function which draw a sample from $X \sim Bin(m, p)$ by sampling m Bernoulli variables and counting the number of successes.
5. Using the commands `clock` and `etime` compare the computational times required to draw 10000 samples from a $Binom(5, .3)$ variable using the two implementations above. Which is the more efficient option?

1. We find that

$$p_{i+1} = \frac{m!}{(i+1)!(m-i-1)!} p^{i+1}(1-p)^{m-i-1} = \frac{m-1}{i+1}\frac{p}{1-p}p_i, \qquad i = 1, 2, \ldots, m.$$

2. A possible implementation is the following one:

```
%   Y=binomial(m,p,n)
%   n: number of samples to be drawn
%   m,p: parameters of the Bin(m,p) distribution

function Y=binomial(m,p,n)

Y=zeros(1,n);
p0=(1-p)^m;
c=p/(1-p);
U=rand(1,n);

for i=1:n
    j=0;
    pi=p0; F=pi;
    while U(i)>F
        pi=c*(m-j)*pi/(j+1);
        F=F+pi;
        j=j+1;
    end
    Y(i)=j;
end
return
```

3. The mean and the variance of a variable $X \sim Binom(5, 0.3)$ are, respectively,

$$\mathbb{E}[X] = mp = 1.5, \qquad \mathrm{Var}[X] = mp(1-p) = 1.05.$$

From the sampling we find

```
Y=binomial(5,.3,1000);
mean_Y = mean(Y)
    mean_Y = 1.4620

var_Y = var(Y)
   var_Y = 1.0756

plot([0:5],binocdf([0:5],5,.3),'r*');  % exact CDF
hold on; cdfplot(Y)                    % empirical CDF
hold off
```

Also in this case, the Statistical toolbox of `Matlab` contains the function `binornd` which allows an efficient sampling of a binomial variable.

4. An alternative implementation is the following one:

```
function Y=binomial1(m,p,n)

Y=zeros(1,n);
U=rand(n,m);

for i=1:n
   X=U(i,:)>(1-p);
   Y(i)=sum(X);
end

return
```

5. We can check the time required by a list of commands as follows:

```
t0 = clock;
''list of commads''
exec_time =  etime (clock, t0)
```

Hence, by the following commands:

```
n=10000; m=5; p=.3;
t0=clock; binomial(m,p,n); s1=etime(clock,t0)
   s1 = 5.7400e-04

t0=clock; binomial1(m,p,n); s2=etime(clock,t0)
   s2 = 0.0052
```

The second implementation takes a longer CPU time, because of the larger number of variables to sample. Indeed, for each realization of the Binomial variable, we need to sample m times from a Bernoulli variable. In the spirit of the class... previous times should be obtained by averaging over a certain amount of runs... :-)

Problem 2.3 (Poisson variable)
We want to draw a sample from a Poisson variable $X \sim Poisson(\lambda)$,

$$p_i = P(X = i) = e^{-\lambda}\frac{\lambda^i}{i!}, \qquad i = 0, 1, 2, \ldots$$

using the inversion method. To do this, exploit the recursive relation

$$p_0 = e^{-\lambda}, \qquad p_{i+1} = \frac{\lambda}{i+1}p_i.$$

1. Write a `Matlab` function to sample a Poisson variable using the inversion method, employing the recursive relation of the previous point to compute the CDF.
2. Set $\lambda = 3$ and verify that the sampling strategy is correct, comparing sample mean and variance and empirical CDF with the theoretical counterparts (the exact CDF can be computed by using the command `poisscdf` of the Statistical toolbox in `Matlab`).

1. The idea is essentially the same as the one exploited when sampling from a Binomial distribution. A possible implementation is the following one:

```
%   Y=poisson(lambda,n)
%   n: number of samples to be drawn
%   lambda: parameter of the Poisson distribution

function Y=poisson(lambda,n)

Y=zeros(1,n);
p0=exp(-lambda);
U=rand(1,n);

for i=1:n
    j=0;
    p=p0; F=p;
    while U(i)>F
        p=lambda*p/(j+1);
        F=F+p;
        j=j+1;
    end
    Y(i)=j;
end
return
```

2.
```
n=1000;
Y=poisson(3,n);
mean_Y = mean(Y)
   mean_Y =.  2.9740

var_Y = var(Y)
   var_Y = 3.0824

x=[0:10];
figure(1); N=hist(Y,x); bar(x,N/n); hold on;  % histogram
plot([0:10],poisspdf(x,3),'r*'); hold off;

figure(2); cdfplot(Y); hold on;
plot(x,poisscdf(x,3),'r*'); hold off;
```

Sample mean and variance are very close to the exact value λ. The histogram and the cumulative distribution function are compared in Figure 2 with the exact values. The `poisspdf` and `poisscdf` functions are part of the `Matlab` Statistics Toolbox and allow you to calculate the exact values of the probability density and cumulative distribution functions. Also, still in the toolbox, the `poissrnd` function is available which allows you to sample a Poisson variable very efficiently.

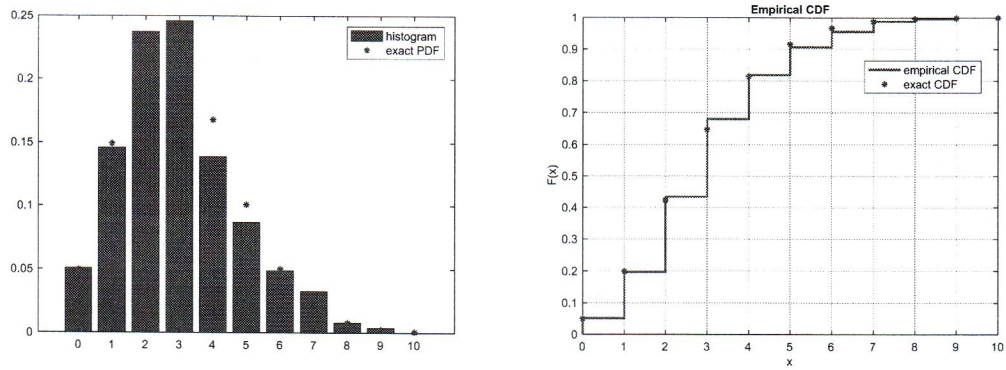


Figure 2: Left: histogram and exact PDF of a Poisson distribution with $\lambda = 3$. Right: exact and empirical CDFs.

Problem 2.4 (Box-Muller transformation for sampling $N(0,1)$)

1. Write a `Matlab` function that generates a vector of n random values sampled from a normal distribution $N(0,1)$, using the Box Muller transformation.
2. Use the function written in the previous point to generate 10^4 samples from a normal distribution of mean 4 and standard deviation 2. Verify that the sampling is correct by comparing the histogram and the empirical distribution function with the corresponding theoretical quantities.

1. The `Matlab` code is as follows:

```
function v=box_muller(n)

X=rand(1,n/2);
Y=rand(1,n/2);

s=sqrt(-2*log(X));
v=[s.*cos(2*pi*Y), s.*sin(2*pi*Y)];

return
```

2. To sample from $Y \sim N(\mu, \sigma^2)$, with $\mu = 4$ and $\sigma = 2$ we can first sample from $X \sim N(0,1)$, then use the affine transformation $Y = \sigma X + \mu$.

```
n=10000; mu=4; s=2;
X=box_muller(n);
Y=s*X+mu;
```

To check that the result is correct, we can compare the histogram with the exact PDF, or the empirical CDF with the theoretical CDF. In this respect, `Matlab` provides wht. function

$$\texttt{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

From the function `erf` we can easily obtain the CDF of $X \sim N(0,1)$:

$$F(x) = P(X \leq x) = 1 - P(X > x) = 1 - \frac{1}{2} P(|X| > x) = \frac{1}{2}(1 + P(|X| \leq x))$$

so that

$$\begin{aligned} F(x) = & \ \tfrac{1}{2}\left(1 + 2\int_0^x \tfrac{1}{\sqrt{2\pi}} e^{-y^2/2} dy\right) = \{ \text{ setting } t = y/\sqrt{2} \ \} \\ = & \ \tfrac{1}{2}\left(1 + 2\int_0^{x/\sqrt{2}} \tfrac{1}{\sqrt{\pi}} e^{-t^2} dt\right) = \tfrac{1}{2}\left(1 + \texttt{erf}(x/\sqrt{2})\right). \end{aligned}$$

Hence,

$$F_Y(x) = P(Y \leq x) = P\left(X \leq \frac{x-\mu}{\sigma}\right) = \frac{1}{2}\left(1 + \texttt{erf}\left(\frac{x-\mu}{\sqrt{2}}\right)\right).$$

We can check that the sampling is correct through the following code (see Figure 3):

```
n=10000; mu=4; s=2;
X=box_muller(n);
Y=s*X+mu;

dx=0.4; x=[-2:dx:10];

figure(1); N=hist(Y,x); bar(x,N/(n*dx),1); hold on;
x=[-2:0.01:10]; pdf= @(x) exp(-(x-mu).^2/(2*s.^2))/(sqrt(2*pi)*s);
plot(x,pdf(x),'r'); hold off;

figure(2); cdfplot(Y); hold on;
cdf = @(x) (1+erf((x-mu)/(sqrt(2)*s)))/2;
plot(x, cdf(x),'r--'); hold off;
```

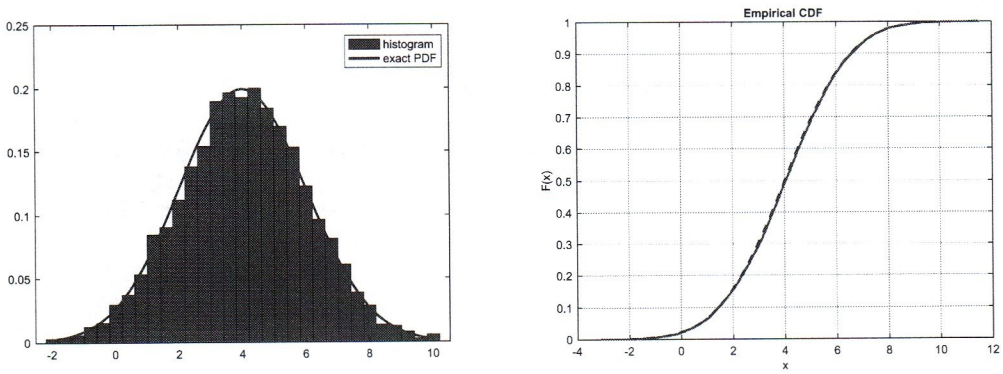


Figure 3: Histogram (left) and empirical CDF (right) for 10^4 values sampled from $X \sim N(4,4)$.

Problem 2.5 (Composition method)
Let X be a random variable with PDF

$$f(x) = (1-\alpha)f_1(x) + \alpha f_2(x), \qquad \int f_i(x)dx = 1, \quad i = 1,2, \quad \alpha > 0.$$

Let moreover be $Y \sim Be(\alpha)$ and X_1, X_2 two variables with density $f_1(x)$, $f_2(x)$, respectively. We can sample X using the following strategy:

- sample Y ;
- if $Y = 0$, then sample X_1 and set $X = X_1$, else sample X_2 and set $X = X_2$.

1. Show that the variable X sampled in this way has the desired distribution.
2. Sample, using the composition method, the variable X whose density is

$$f(x) = \frac{1}{4\sqrt{2\pi}} e^{-\frac{(x+1)^2}{8}} + \frac{1}{4\sqrt{2\pi}} e^{-\frac{(x-2)^2}{8}}$$

and show that the sampling is correct.

3. Sample, using the composition method, the variable Y whose CDF is

$$F_Y(x) = \begin{cases} x/3 & \text{if } 0 \le x \le 1/2 \\ (x+2)/3 & \text{if } 1/2 < x \le 1 \end{cases}$$

and show that the sampling is correct comparing the drawn samples with the empirical CDF.

1. The PDF of X is given by

$$f(x) = P(X = x|Y = 0)P(Y = 0) + P(X = x|Y = 1)P(Y = 1) = (1-\alpha)f_1(x) + \alpha f_2(x)$$

and corresponds to the desired distribution.

2. We can write

$$f(x) = \frac{1}{2}f_1(x) + \frac{1}{2}f_2(x), \qquad \begin{cases} f_1(x) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x+1)^2}{8}} \\ f_2(x) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-2)^2}{8}}; \end{cases}$$

X_1 and X_2 correspond to two Gaussian random variables with mean $\mu = -1$ and 2 and variance $\sigma^2 = 4$. If $Z \sim N(0,1)$ is a standard Gaussian random variable, which can be sampled either by the command `randn` of `Matlab` (or using the `box_muller` function) we have

$$X_1 = 2Z + 1, \quad X_2 = 2Z + 2.$$

The `composition` function implements the desired sampling. To verify the correctness of the sample, we compare the histogram with the exact probability density. We can take the interval $[-1-3\sigma, 2+3\sigma]$ as a support for the histogram (see Fig. 4, left).

```
n=1000;
X=composition(n);
figure(1)
dx=1; x=[-7+dx/2:dx:8-dx/2];
P=hist(X,x)/(n*dx); bar(x,P,1); hold on;
p='exp(-(x+1).^2/8)/(4*sqrt(2*pi)) + exp(-(x-2).^2/8)/(4*sqrt(2*pi))';
x=[-7:.01:8]; plot(x,eval(p),'r'); hold off
```

3. In the case of Y, we have

$$F_Y(x) = \frac{1}{3}F_1(x) + \frac{2}{3}F_2(x), \qquad \begin{cases} F_1(x) = x & 0 \le x \le 1 \\ F_2(x) = H(x-1/2) & 0 \le x \le 1 \end{cases}$$

where $H(x)$ is the Heaviside function. Then, we can obtain

$$f_Y(x) = \frac{1}{3}f_1(x) + \frac{2}{3}f_2(x), \qquad \begin{cases} f_1(x) = 1 & 0 \le x \le 1 \\ f_2(x) = \delta(x - 1/2) & 0 \le x \le 1 \end{cases}$$

where $\delta(x)$ is the Dirac mass (at the origin). X_1 is a uniform random variable over $[0, 1]$, while X_2 is a concentrated Dirac mass at $x = 1/2$, that is, it takes value $1/2$ with probability 1. The function `composition1.m` implements the desired sampling (see Fig. 4, right).

```
Y=composition1(n);
figure(2)
cdfplot(Y); hold on
F = @(x)  x/3 + 2*heaviside(x-1/2)/3;
x=[0:.01:1]; plot(x,F(x),'r'); hold off
```

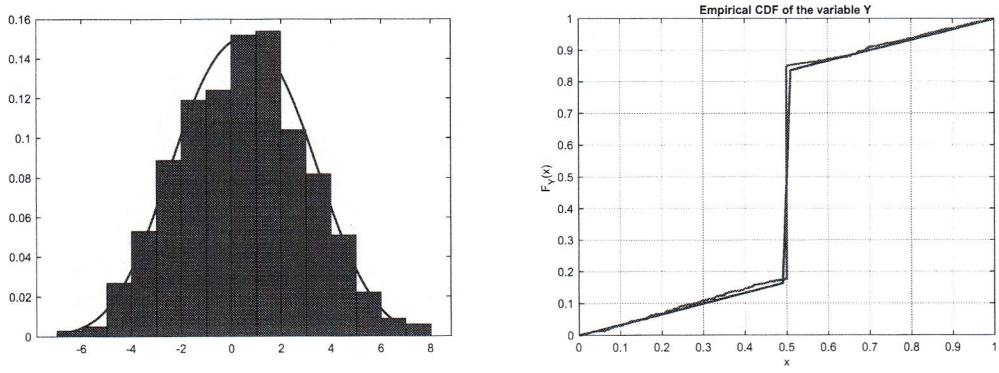


Figure 4: Histogram and exact PDF of X (left), empirical and exact CDF of Y (right).

Problem 2.6 (Acceptance-Rejection method)
Let X be a continuous random variable with values in $[0, 1]$, having probability density $f(x) = 6x(1-x)$.

1. Sample X using the acceptance-rejection method, using a uniform variable in $[0, 1]$ as the proposal distribution Y. Calculate the probability of rejection and properly verify the accuracy of the sampling.
2. Repeat the previous point taking a variable Y with probability density $g(x) = \frac{\pi}{2}\sin(\pi x)$ is chosen as the proposal distribution. Which of the two methods is more efficient?

1. In this case we have $g(y) = 1$ and $f(x) = 6x(1-x) = K\tilde{f}(x) = \tilde{f}(x)$ with $K = 1$. Since we want to bound $f(x)$ by $Cg(x)$, we can take as constant

$$C = \max_{x\in[0,1]} \frac{f(x)}{g(x)} = 3/2.$$

The probability that a point is accepted is equal to $1/C = 2/3$, that is, the rejection probability is $1-1/C = 1/3$. 33% of the points will then be rejected. The function `acceptance_rejection.m` implements the method:

```
%   X = acceptance_rejection(n)
%   n: number of samples to draw

function X=acceptance_rejection(n)

X = zeros(1,n);
i = 0;                  % counter of accepted samples
```

the one with lower rejection probability (we don't want to waste points)

⇓

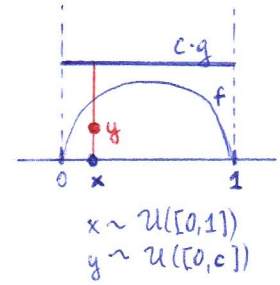
we want a good proposal (= one that follows f in a good way) but it has to be samplable

```
m = 0;                    % counter of total samples
c = 3/2;                  % constant c = max(f(x)/g(x))
f = @(x)  6*x.*(1-x);     % density f(x)

while i<n
    m=m+1;
    x=rand;      % sample the proposal (abscissa)
    y=c*rand;    % sample the ordinate
    if y<=f(x)
        i=i+1;   % found a good sample, retained
        X(i)=x;
    end
end

fprintf('rejected points: %i \n',m-i);
fprintf('rejection amount: %6.2f \n', (m-i)/m);

return
```



To verify that it is correct, we can run the following commands (see Figure 5):

```
X=acceptance_rejection(n);
   rejected points: 515
   rejection amount: 0.34

figure(1) % histogram
dx=.05; x=[dx/2:dx:1-dx/2];
P=hist(X,x)/(n*dx); bar(x,P,1); hold on
x=[0:0.01:1]; plot(x,6*x.*(1-x),'r'); hold off

figure(2); % CDF
cdfplot(X); hold on;
cdf=@(x) x.^2.*(3-2*x);
plot(x,cdf(x),'r'); hold off;
```

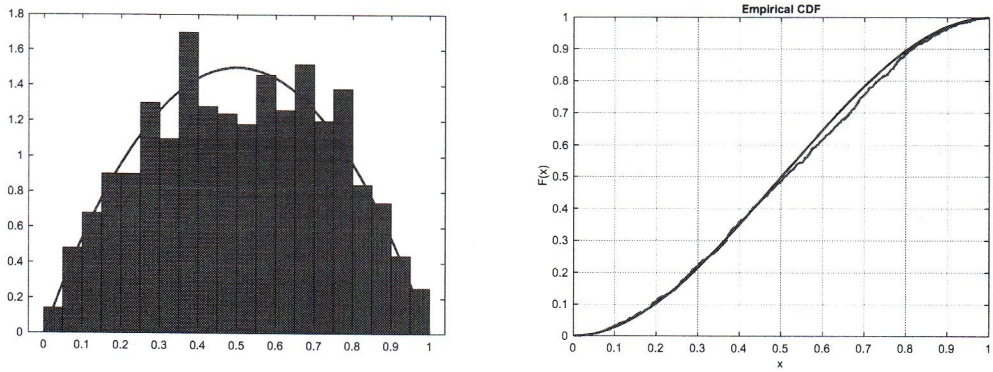


Figure 5: Histogram and exact PDF of $X \sim f(x) = 6x(1-x)$ (left); empirical and exact CDF of X (right).

2. The proposal distribution Y has density $g(x) = \frac{\pi}{2}\sin(\pi x)$, and can be sampled through the inversion method, generating $U \sim \mathcal{U}(0,1)$ and setting $Y = F^{-1}(U)$. For the case at hand,

$$F(x) = \frac{\pi}{2}\int_0^x \sin(\pi y)dy = \frac{\pi}{2}\left(\frac{1}{\pi} - \frac{1}{\pi}\cos(\pi x)\right) = \frac{1}{2}(1-\cos(\pi x))$$

so that

$$u = F(x) \;\Leftrightarrow\; x = F^{-1}(u) = \frac{1}{\pi}\arccos(1-2u).$$

The constant C can be taken as $C = 12/\pi^2$. In this case, the rejection probability is $1 - 1/C \approx 0.1775$, that is, on average, 17.7% of the points will be rejected. The probability of rejection is smaller than in the case at the previous point. In `Matlab` :

```
Y=acceptance_rejection1(n);

   punti rigettati: 254
   percentuale di rigetto:   0.20
```