# MATH 218 Ch1: Experiments, Models, and Probabilities

MATLAB (software package by Mathworks) will be used to simulate experiments with random outcomes. For this purpose, a *pseudo-random number generator* which produces a sequence of random numbers between 0 and 1 can be utilized, i.e. *rand(m, n)* produces an *m* x *n* array of uniformly distributed pseudo-random numbers.

For a Matlab simulation we first generate a vector R of N random numbers:

```
>> N = 100;
>> R = rand(1, N);
```

To simulate an experiment that contains an event with probability *p*, each random number *r* produced by above command will be tested such that if $r < p$, event occurs; otherwise it does not occur.
For example, to simulate a coin flipping experiment which yields heads with probability 0.4, tails with probability 0.5, or lands on its edge with probability 0.1, and 100 simulations are needed:
In this case, we generate vector *X* as a function of R to represent 3 possible outcomes: $X(i)=1$ if flip *i* was heads, $X(i)=2$ if flip *i* was tails, and $X(i)=3$ if flip *i* landed on the edge.

```
>> X = (R <= 0.4) + (2*(R > 0.4).*(R <= 0.9)) + (3*(R > 0.9));
>> [N, Y] = hist(X, 1:3);
```

Plot the number of occurances of each outcome *i* for *N*=100 trails of this experiment

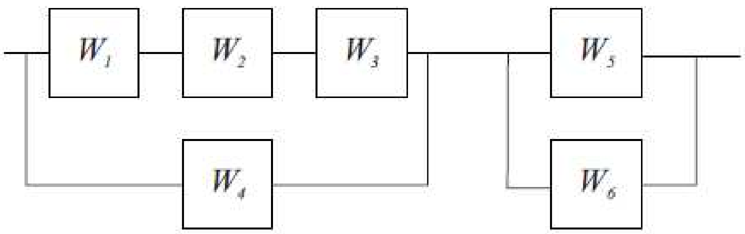
```
>> figure, bar(Y, N)
>> xlabel('Outcome i'), ylabel('Number of occurances of outcome i');
```

Next, the following Matlab function can be used to simulate two-coin experiment in Example 1.27 in textbook:

```
>> function [C, H]  = twocoin(n);  % n: number of trials
        C = ceil(2*rand(n, 1)); % C(i) = 1 if coin 1, C(i) = 2 if coin 2 is chosen for trial i
        P = 1 – (C/4);  % P(i) = 0.75 if C(i) = 1, otherwise if C(i) = 2
        H = (rand(n, 1) < P);
   end
```

This function generates vectors *C* and *H* for *n* trials of this experiment. $C(i)$ indicates which coin (biased =1 or fair =2) is chosen, and $H(i)$ is the simulated result of a coin flip with heads, $H(i) = 1$ occurring with probability $P(i)$.

Another simulation example is given below about the reliability of a 6 component system which has the following configuration:




To simulate 100 trials of the six-component test (such that each component works with probability q), we use the following Matlab function:

```
function N=reliable6(n,q);
% n is the number of 6 component devices
%N is the number of working devices
W=rand(n,6)>q;
D=(W(:,1)&W(:,2)&W(:,3))|W(:,4);
D=D&(W(:,5)|W(:,6));
N=sum(D);
```

The n×6 matrix W is a logical matrix such that W(i,j)=1 if component j of device i works properly. Note that D(i)=1 if device i works. Otherwise, D(i)=0. Hence, N is the number of working devices.
The result of 10 repetitions of the 100 trials for q=0.2 is given below:

```
>> for n=1:10, w(n)=reliable6(100,0.2); end
>> w
w =
82 87 87 92 91 85 85 83 90 89
>>
```

While the probability the device works is actually 0.8663.

For **discrete random variables**, the PMFs and CDFs for the families of random variables can be computed as follows:
For finite discrete random variable *X* defined by a set of sample values $S_X = \{s_1,...,s_n\}$ with corresponding probabilities $p_i = P_X(s_i) = P[X = si]$, $\mathbf{p} = [p_1 \ldots p_n]'$, the following Matlab function returns $y_i = P_X(x_i)$:

```
function pmf=finitepmf(sx,px,x)
% finite random variable X:
% vector sx of sample space
% elements {sx(1),sx(2), ...}
% vector px of probabilities
% px(i)=P[X=sx(i)]
% Output is the vector
% pmf: pmf(i)=P[X=x(i)]
pmf=zeros(size(x(:)));
for i=1:length(x)
    pmf(i)= sum(px(find(sx==x(i))));
end
```

By applying its definition, the CDF can be calculated simply as

```
function cdf=finitecdf(s,p,x)
% finite random variable X:
% vector sx of sample space
% elements {sx(1),sx(2), ...}
% vector px of probabilities
% px(i)=P[X=sx(i)]
% Output is the vector
% cdf: cdf(i)=P[X=x(i)]
cdf=[];
for i=1:length(x)
    pxi= sum(p(find(s<=x(i))));
    cdf=[cdf; pxi];
end
```

Note that the CDF can also be computed by summing the PMF as

```
>> cdf=cumsum(pmf);
```

The sample values of random variables can be generated based on the calculation of the CDF first. For example, the m samples of a binomial (n, p) random variable can be generated as

```
function x=binomialrv(n,p,m)
% m binomial(n,p) samples
r=rand(m,1);
cdf=binomialcdf(n,p,0:n);
x=count(cdf,r);
```