

UCF



Stands For Opportunity

CDA6530: Performance Models of Computers and Networks

***Using Simulation for Statistical Analysis
and Verification***

---- supplement to Random Variable
Generation Lecture

Objective

- ❑ We have learned how to generate random variables in simulation, but
- ❑ How can we use them?
- ❑ What is the purpose for such simulation?

Example: Generate discrete R.V.

- A loaded dice, r.v. X : number shown up
 - $P(X=1) = 0.05$, $P(X=2) = 0.1$, $P(X=3) = 0.15$,
 $P(X=4) = 0.18$, $P(X=5) = 0.22$, $P(X=6) = 0.3$
- Q1: Simulate to generate 100 samples

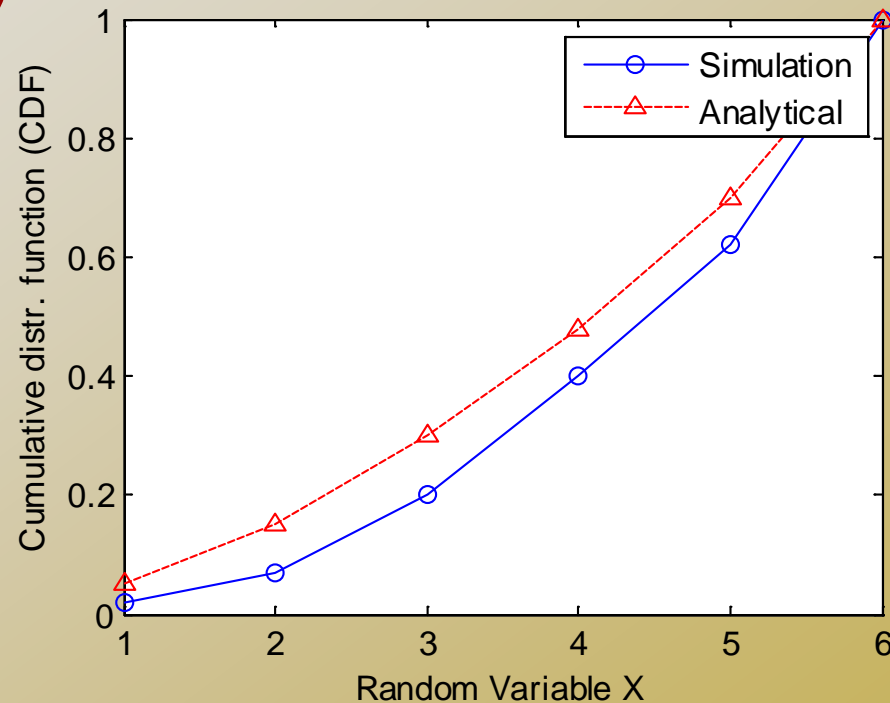
$$X = \begin{cases} x_0 & \text{if } U < p_0 \\ x_1 & \text{if } p_0 \leq U < p_0 + p_1 \\ \vdots & \\ x_j & \text{if } \sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^j p_i \\ \vdots & \end{cases} \quad X = \begin{cases} 1 & \text{if } U < 0.05 \\ 2 & \text{if } 0.05 \leq U < 0.15 \\ 3 & \text{if } 0.15 \leq U < 0.3 \\ 4 & \text{if } 0.3 \leq U < 0.48 \\ 5 & \text{if } 0.48 \leq U < 0.7 \\ 6 & \text{if } 0.7 \leq U \end{cases}$$

Code in Matlab

Draw CDF (cumulative distr. function)

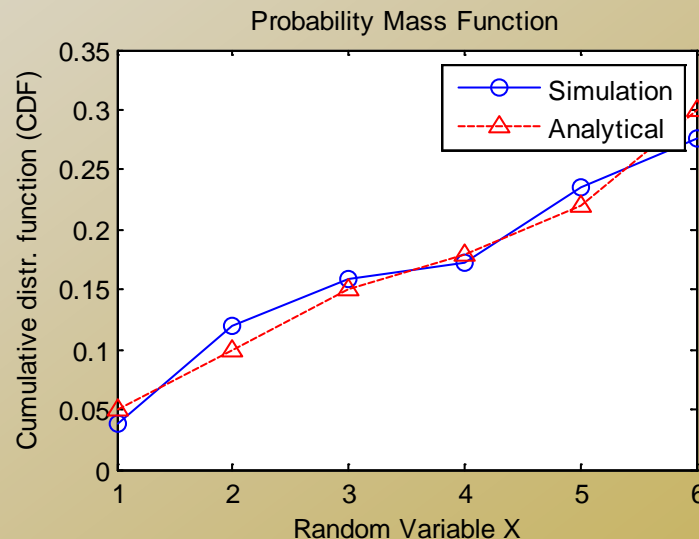
- - Remember $F(x) = P(X \leq x)$
 - For our question, r.v. X has 6 different sample values, thus we could derive from simulation:
 $F(1), F(2), F(3), F(4), F(5), F(6)$
 - $F(6) = 1$, so we need to derive the other five
 - $F(x) = m/n$ where
 - n : # of sample values we generate
 - m : # of sample r.v. values $\leq x$

- ❑ The CDF curve for our example in one simulation run (compared with analytical results)



Draw PMF (Probability Mass Function)

- Pmf: $P(X=1) = 0.05$, $P(X=2) = 0.1$, $P(X=3) = 0.15$,
 $P(X=4) = 0.18$, $P(X=5) = 0.22$, $P(X=6) = 0.3$
- $PMF(x) = m/n$
 - n : # of sample values we generate
 - m : # of sample r.v. values = x



Matlab Code

```
% lecture 15, statics-matlab.ppt , Page 3
```

```
% generate discrete r.v.
sampleN = 1000;
P = [0.05 0.1 0.15 0.18 0.22 0.3];
Xvalue = [1 2 3 4 5 6];
n =length(P);
SumP = zeros(n, 1);
for i=1:n,
    SumP(i) = sum(P(1:i));
end
U = rand(sampleN,1);
Sample = zeros(sampleN,1);
for i=1:sampleN,
    for j=1:n,
        if U(i)< SumP(j),
            Sample(i) = Xvalue(j);
            break;
        end
    end
end
```

```
% draw CDF, Page 5
```

```
F = zeros(n,1);
for i = 1:n,
    for j=1:sampleN,
        if Sample(j)<= Xvalue(i),
            F(i) = F(i) + 1;
        end
    end
end
```

```
F = F./sampleN;
plot(F, '-ob');
xlabel('Random Variable X'); ylabel('Cumulative distr.
function (CDF)');
hold on;
plot(SumP, '--^r');
legend('Simulation', 'Analytical');
title('CDF');
```

```
% draw pmf, Page 6
```

```
PMF = zeros(n,1);
for i=1:n,
    for j=1:sampleN,
        if Sample(j) == Xvalue(i),
            PMF(i)= PMF(i)+1;
        end
    end
end
PMF = PMF./sampleN;
figure;
plot(PMF, '-ob');
xlabel('Random Variable X'); ylabel('Cumulative distr.
function (CDF)');
hold on;
plot(P, '--^r');
legend('Simulation', 'Analytical');
title('Probability Mass Function');
```

Continuous R.V.

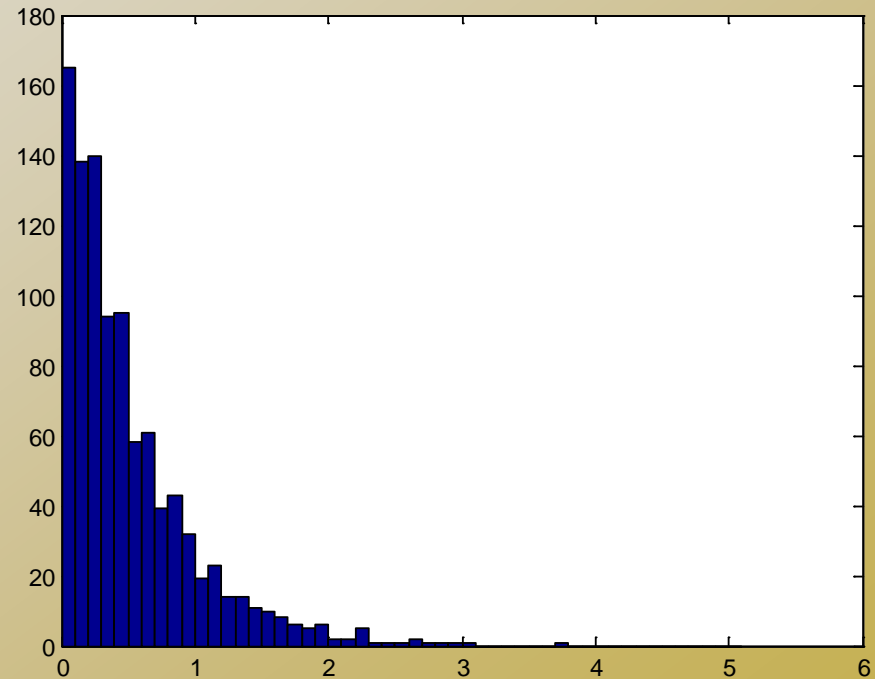
- Use inverse transform method:
 - One value of $U \rightarrow$ one r.v. sample
- Normal distr. use the polar method to generate
- How to draw CDF?
 - Problem: r.v. x could be any value
 - Solve: determine x_i points to draw with fixed interval ($i=1,2,\dots$)
 - $F(x_i) = P(X \leq x_i) = m/n$
 - n : # of samples generated
 - m : # of sample values $\leq x_i$

Continuous R.V.

- How to draw pdf (probability density function)?
 - In Matlab, use `histc()` and `bar()`
 - $N = \text{histc}(Y, \text{Edge})$ for vector Y , counts the number of values in Y that fall between the elements in the `Edge` vector (which must contain monotonically non-decreasing values). N is a `Length(Edge)` vector containing these counts.
 - Use `bar(Edge,N, 'histc')` to plot the curve
 - The curve plot will have the same curve pattern as $f(x)$, but not the same Y-axis values

Pdf example of continuous R.V.

```
% exponential distribution pdf
lambda = 2; sampleN = 1000;
Sample = zeros(1, sampleN);
U = rand(1, sampleN);
for i=1:sampleN,
    Sample(i) = -log(1-U(i))/lambda;
end
Edge = 0:0.1:5;
N = histc(Sample, Edge);
bar(Edge, N, 'histc');
```



Analytical Results

- ❑ Use the formula of the distribution to directly calculate $F(x_i)$
- ❑ How to calculate integration in Matlab?
 - ❑ Use function `quad()`

```
Q = quad(@myfun,0,2);
```

```
%where myfun.m is the M-file function:
```

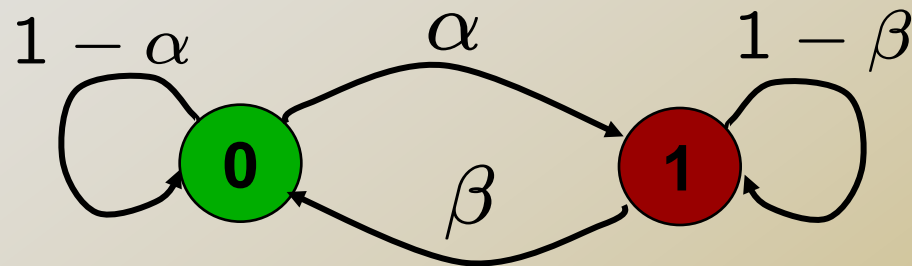
```
function y = myfun(x)
```

```
    y = 1./(x.^3-2*x-5);
```

Markov Chain Simulation

- ❑ **Discrete-time Markov Chain**
 - ❑ Simulate N steps
 - ❑ For each step, use random number U to determine which state to jump to
 - ❑ Similar to discrete r.v. generation
 - ❑ $\pi(i) = m_i/N$
 - ❑ N: # of simulated steps
 - ❑ m_i : number of steps when the system stays in state i.

Discrete-time Markov Chain Example



- **Markov on-off model (or 0-1 model)**
- **Q: the steady-state prob.?**

$$\mathbf{P} = \begin{bmatrix} 1 - \alpha & \alpha \\ \beta & 1 - \beta \end{bmatrix}$$

$$\begin{cases} \pi_0 = (1 - \alpha)\pi_0 + \beta\pi_1 \\ \pi_1 = \alpha\pi_0 + (1 - \beta)\pi_1 \\ \pi_0 + \pi_1 = 1 \end{cases} \Rightarrow \begin{cases} \pi_0 = \frac{\beta}{\alpha + \beta} \\ \pi_1 = \frac{\alpha}{\alpha + \beta} \end{cases}$$

Matlab Code

```
alpha= 0.6; beta = 0.9;
Pi_theory = [beta/(alpha+beta) alpha/(alpha+beta)];
sampleN= 100; stateNum = 2; % 2 states
U = rand(sampleN,1);
init_state = randi(2); %either 1 or 2
State=zeros(stateNum, 1);
State(init_state) = 1; % count the initial state
currentState = init_state;
for i=1:sampleN,
    if currentState ==1,
        if U(i) > alpha,
            nextState = 1;
        else
            nextState = 2;
        end
    else % currentState = 2
        if U(i) > beta,
            nextState = 2;
        else
            nextState = 1;
        end
    end
end
```

```
State(nextState) =State(nextState)+1; % count the current
                                % state
    currentState = nextState;
end
Pi_simulation = State./sampleN;
figure;
plot(Pi_theory, 'r-^'); hold on; plot(Pi_simulation, 'k:o');
legend('theory', 'simulation');
```

Markov Chain Simulation

- ❑ **Continuous-time Markov Chain**
 - ❑ Method #1:
 - ❑ Determine how long current state lasts
 - ❑ Generate exponential distr. r.v. X for the staying time
 - ❑ Determine which state the system jumps to
 - ❑ Similar to discrete-time Markov Chain jump simulation
 - ❑ Method #2:
 - ❑ Determine the jumping out time for each jump out link (everyone is expo. Distr.)
 - ❑ The staying time is the shortest jumping out time
 - ❑ The outgoing state corresponds to this shortest jumping out time
 - ❑ Method #2 is more intuitive reasonable

-
- $\pi(i) = \sum t_k(i) / T$
 - T: overall simulated time
 - $t_k(i)$: the time when the system is in state i for the k-th time