CDA6530: Performance Models of Computers and Networks

Chapter 8: Statistical Simulation --- Discrete-Time Simulation
Simulation Studies

- Models with analytical formulas
  - Calculate the numerical solutions
    - Differential equations ---- Matlab Simulink
      - Or directly solve if has closed formula solutions
    - Discrete equations --- program code to solve
  - The mean value formulas for stochastic events
    - Solutions are only for the mean values
  - If you derive models in your paper, you must use real simulation to verify that your analytical formulas are accurate
Simulation Studies

- Models without analytical formulas
  - Monte Carlo simulation
    - Generate a large number of random samples
    - Aggregate all samples to generate final result
    - Example: use U(0,1) to compute integral
  - Discrete-time simulation
    - Divide time into many small steps
    - Update system states step-by-step
    - Approximate, assume system unchanged during a time step
  - Discrete event simulation (DES)
    - Accurate
    - Event-driven
Discrete-Time Simulation

- System is assumed to change only at each discrete time tick
  - Smaller time tick, more accurate simulation for a continuous-time physical system
  - At time \( k \), all nodes’ status are only affected by system status at \( k-1 \)

- Why use it?
  - Simpler than DES to code and understand
  - Fast, if system states change very quickly (or many events happening in short time period)
Discrete-Time Simulation

While (simulation not complete) {
   1). Time tick: k ++;
   2). For system’s node i (i=1, 2, ...)
      3). Simulate what could happen for node i during the last time step (k-1 \rightarrow k) based on all nodes status at k-1
      4). Update the state of node i if something happens to it
   5). Output time tick k’s system’s states (e.g., status of every node in the system)
}

UCF  Stands For Opportunity
Discrete-Time Simulation

- Note: when computing system node $i$’s state at time tick $k$, it should be determined only by all other system nodes’ states at time tick $k-1$
  - Be careful in step 4): DO NOT use node $j$’s newly updated value at current round
    - Newly updated value represents state at the beginning of next round.
Discrete-Time Simulation

- An example: one line of nodes
  \[ X_i(t) = (U - 0.5) + \frac{(X_{i-1}(t-1) + X_{i+1}(t-1))}{2} \]

Simul_N = 1000; n=100; X = ones(n,1);
for k=1:Simul_N,
    U = rand(n,1);
    X(1) = (U(1) - 0.5) + X(2);
    for i=2:n-1,
        X(i) = (U(i) - 0.5) + (X(i-1) + X(i+1)) / 2;
    end
    X(n) = (U(n) - 0.5) + X(n-1);
    % display or save X value for time k
end

What’s Wrong?
Corrected Code:

```matlab
Simul_N = 1000; n=100; X = ones(n,1);
Prior_X = ones(n,1);
for t=1:Simul_N,
    U = rand(n,1);
    Prior_X = X;  /* save last time’s data */
    X(1) = (U(1) - 0.5) + Prior_X(2);
    for i=2:n-1,
        X(i) = (U(i) - 0.5) + (Prior_X(i-1) + Prior_X(i+1)) / 2;
    end
    X(n) = (U(n) - 0.5) + Prior_X(n-1);
    % display or save X value for time k
end```

Another way to do the correct coding:

Simul_N = 1000; n=100; X = ones(n,Simul_N);
% X(i, t) is the value of node i at time t.
for t=2:Simul_N,
    U = rand(n,1);
    X(1, t) = (U(1) - 0.5) + X(2,t-1);
    for i=2:n-1,
        X(i,t) = (U(i) - 0.5) + (X(i-1, t-1) + X(i+1,t-1)) / 2;
    end
    X(n,t) = (U(n) - 0.5) + X(n-1, t-1);
    % display or save X value for time k
end
Example: Discrete-Time Markov Chain Simulation

- Simulate N steps
- For each step, use random number U to determine which state to jump to
  - Similar to discrete r.v. generation
- \( \pi(i) = \frac{m_i}{N} \)
  - 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.?

\[ P = \begin{bmatrix} 1 - \alpha & \alpha \\ \beta & 1 - \beta \end{bmatrix} \]

\[
\begin{align*}
\pi_0 &= (1 - \alpha)\pi_0 + \beta\pi_1 \\
\pi_1 &= \alpha\pi_0 + (1 - \beta)\pi_1 \\
\pi_0 + \pi_1 &= 1
\end{align*}
\]

\[
\begin{align*}
\pi_0 &= \frac{\beta}{\alpha + \beta} \\
\pi_1 &= \frac{\alpha}{\alpha + \beta}
\end{align*}
\]
Simulation result (100 time steps)

- `bar([Pi_theory Pi_simulation]);`
- `Pi_theory` and `Pi_simulation` are column vectors
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 ($i=1,2,...$)
- $F(x_i) = P(X \leq x_i) = m/n$
  - $n$: # 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 = histc(Y, 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
% 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');