

*CDA6530: Performance Models of Computers and Networks*

***Chapter 6: Generating Random Number  
and Random Variables***

# Objective

---

- ❑ Use computers to simulate stochastic processes
- ❑ Learn how to generate random variables
  - ❑ Discrete r.v.
  - ❑ Continuous r.v.
- ❑ Basis for many system simulations

# Pseudo Random Number Generation (PRNG)

---

- $x_n = a x_{n-1} \bmod m$ 
  - Multiplicative congruential generator
  - $x_n = \{0, 1, \dots, m-1\}$
  - $x_n/m$  is used to approx. distr.  $U(0,1)$
  - $x_0$  is the initial “seed”
- **Requirements:**
  - No. of variables that can be generated before repetition begins is large
  - For any seed, the resultant sequence has the “appearance” of being independent
  - The values can be computed efficiently on a computer

- 
- 
- $x_n = a x_{n-1} \pmod{m}$
  - $m$  should be a large prime number
  - For a 32-bit machine (1 bit is sign)
    - $m = 2^{31} - 1 = 2,147,483,647$
    - $a = 7^5 = 16,807$
  - For a 36-bit machine
    - $m = 2^{35} - 31$
    - $a = 5^5$
  - $x_n = (ax_{n-1} + c) \pmod{m}$ 
    - Mixed congruential generator

# *In C Programming Language*

---

- ❑ **Int rand(void)**
  - ❑ Return int value between 0 and RAND\_MAX
  - ❑ RAND\_MAX default value may vary between implementations but it is granted to be at least 32767
- ❑ **X=rand()**
  - ❑  $X=\{0, 1, \dots, \text{RAND\_MAX}\}$
- ❑  **$X = \text{rand()} \% m + n$** 
  - ❑  $X=\{n, n+1, \dots, m+n-1\}$
  - ❑ Suitable for small m;
  - ❑ Lower numbers are more likely picked

# *(0,1) Uniform Distribution*

---

- ❑ U(0,1) is the basis for random variable generation
- ❑ C code (at least what I use):

```
Double rand01(){  
    double temp;  
    temp = double( rand()+0.5 ) /  
    (double(RAND_MAX) + 1.0);  
    return temp;  
}
```

# Monte Carlo Approach ----

## Use Random Number to Evaluate Integral

---

---

$$\theta = \int_0^1 g(x) dx \qquad \theta = E[g(U)]$$

- U is uniform distr. r.v. (0,1)
- Why?

$$E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x) dx$$

$$f_U(x) = 1 \quad \text{if } 0 < x < 1$$

- 
- 
- $U_1, U_2, \dots, U_k$  are independent generated uniform distr. (0,1)
    - $g(U_1), \dots, g(U_k)$  are independent
    - Law of large number:

$$\sum_{i=1}^k \frac{g(U_i)}{k} \rightarrow E[g(U)] = \theta \text{ as } k \rightarrow \infty$$



$$\theta = \int_a^b g(x) dx$$

---

---

- Substitution:  $y=(x-a)/(b-a)$ ,  $dy = dx/(b-a)$

$$\theta = \int_0^1 (b-a) \cdot g(a + (b-a)y) dy = \int_0^1 h(y) dy$$

$$h(y) = (b-a) \cdot g(a + (b-a)y)$$

$$\theta = \int_0^1 \int_0^1 \cdots \int_0^1 g(x_1, \cdots, x_n) dx_1 dx_2 \cdots dx_n$$

---

---

$$\theta = E[g(U_1, \cdots, U_n)]$$

- ❑ **Generate many  $g(\dots)$**
- ❑ **Compute average value**
  - ❑ which is equal to  $\theta$

# Generate Discrete Random Variables

## ---- Inverse Transform Method

---

---

- r.v.  $X$ :  $P(X = x_j) = p_j, \quad j=0,1,\dots$
- We generate a PRNG value  $U \sim U(0,1)$ 
  - For  $0 < a < b < 1$ ,  $P(a \leq U < b) = b - a$ , thus

$$P(X = x_j) = P\left(\sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^j p_i\right) = p_j$$

$$X = \begin{cases} x_0 & \text{if } U < p_0 \\ x_1 & \text{if } p_0 \leq U < p_0 + p_1 \\ \vdots & \\ x_j & \text{if } \sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^j p_i \\ \vdots & \end{cases}$$

# Example

---

- ❑ A loaded dice:
  - ❑  $P(1)=0.1$ ;  $P(2)=0.1$ ;  $P(3)=0.15$ ;  $P(4)=0.15$
  - ❑  $P(5)=2$ ;  $P(6)=0.3$
- ❑ Generate a series of random variable

# Generate a Poisson Random Variable

---

---

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

- Use following recursive formula to save computation:

$$p_{i+1} = \frac{\lambda}{i+1} p_i$$

# Generating a Random Permutation

---

- Generate a permutation of  $\{1, \dots, n\}$
- $\text{Int}(kU) + 1$ :
  - uniformly pick from  $\{1, 2, \dots, k\}$
- **Algorithm:**
  - $P_1, P_2, \dots, P_n$  is a permutation of  $1, 2, \dots, n$  (e.g., we can let  $P_j = j, j = 1, \dots, n$ )
  - Set  $k = n$
  - Generate  $U$ , let  $I = \text{Int}(kU) + 1$
  - Interchange the value of  $P_I$  and  $P_k$
  - Let  $k = k - 1$  and if  $k > 1$  goto Step 3
  - $P_1, P_2, \dots, P_n$  is a generated random permutation

Example: permute (10, 20, 30, 40, 50)

# *Some Other Approaches*

---

- ❑ Acceptance-Rejection approach
- ❑ Composition approach
  - ❑ They all assume we have already generated a random variable first (not U)
  - ❑ Not very useful considering our simulation purpose

# Generate Continuous Random Variables

## ---- Inverse Transform Method

---

- r.v.  $X$ :  $F(x) = P(X \leq x)$
- r.v.  $Y$ :  $Y = F^{-1}(U)$ 
  - $Y$  has distribution of  $F$ . ( $Y \stackrel{\text{st}}{=} X$ )
- $P(Y \leq x) = P(F^{-1}(U) \leq x)$ 
  - $= P(F(F^{-1}(U)) \leq F(x))$
  - $= P(U \leq F(x))$
  - $= P(X \leq x)$
- Why? Because  $0 < F(x) < 1$  and the CDF of a uniform  $F_U(y) = y$  for all  $y \in [0; 1]$



# Generate Exponential Random Variable

---

---

$$F(x) = 1 - e^{-\lambda x}$$

$$U = 1 - e^{-\lambda x}$$

$$e^{-\lambda x} = 1 - U$$

$$x = -\ln(1 - U)/\lambda$$

$$F^{-1}(U) = -\ln(1 - U)/\lambda$$

# Generate Normal Random Variable --- Polar method

- The theory is complicated, we only list the algorithm here:
  - Objective: Generate a pair of independent standard normal r.v.  $\sim N(0, 1)$ 
    - Step 1: Generate (0,1) random number  $U_1$  and  $U_2$
    - Step 2: Set  $V_1 = 2U_1 - 1$ ,  $V_2 = 2U_2 - 1$   $S = V_1^2 + V_2^2$
    - Step 3: If  $S > 1$ , return to Step 1.
    - Step 4: Return two standard normal r.v.:

$$X = \sqrt{\frac{-2 \ln S}{S}} V_1, \quad Y = \sqrt{\frac{-2 \ln S}{S}} V_2$$

<b>z</b>	<b>F(x)</b>	<b>z</b>	<b>F(x)</b>	<b>z</b>	<b>F(x)</b>
-2.5	0.006	-1	0.159	0.5	0.691
-2.4	0.008	-0.9	0.184	0.6	0.726
-2.3	0.011	-0.8	0.212	0.7	0.758
-2.2	0.014	-0.7	0.242	0.8	0.788
-2.1	0.018	-0.6	0.274	0.9	0.816
-2	0.023	-0.5	0.309	1	0.841
-1.9	0.029	-0.4	0.345	1.1	0.864
-1.8	0.036	-0.3	0.382	1.2	0.885
-1.7	0.045	-0.2	0.421	1.3	0.903
-1.6	0.055	-0.1	0.46	1.4	0.919
-1.5	0.067	0	0.5	1.5	0.933
-1.4	0.081	0.1	0.54	1.6	0.945
-1.3	0.097	0.2	0.579	1.7	0.955
-1.2	0.115	0.3	0.618	1.8	0.964
-1.1	0.136	0.4	0.655	1.9	0.971

- ❑ **Another approximate method- Table lookup**
  - ❑ Treat Normal distr. r.v.  $X$  as discrete r.v.
  - ❑ Generate a  $U$ , check  $U$  with  $F(x)$  in table, get  $z$

# Generate Normal Random Variable

---

- ❑ Polar method generates a pair of standard normal r.v.s  $X \sim N(0,1)$
- ❑ What about generating r.v.  $Y \sim N(\mu, \sigma^2)$ ?

- ❑  $Y = \sigma X + \mu$