



SHARIF
UNIVERSITY OF
TECHNOLOGY



Computer Simulation

Dr. Bardia Safaei

Chapter Six: Random Number Generation



Purpose and Overview



SHARIF
UNIVERSITY OF
TECHNOLOGY

- Random numbers have a pivotal role in any discrete system simulation environment
 - The main reason for using random numbers is to model events that may occur during the system operation
- In this chapter, we concentrate on the following:
 - Well-known techniques for generating random numbers
 - Introducing the subsequent testing for randomness:
 - Frequency test
 - Autocorrelation test

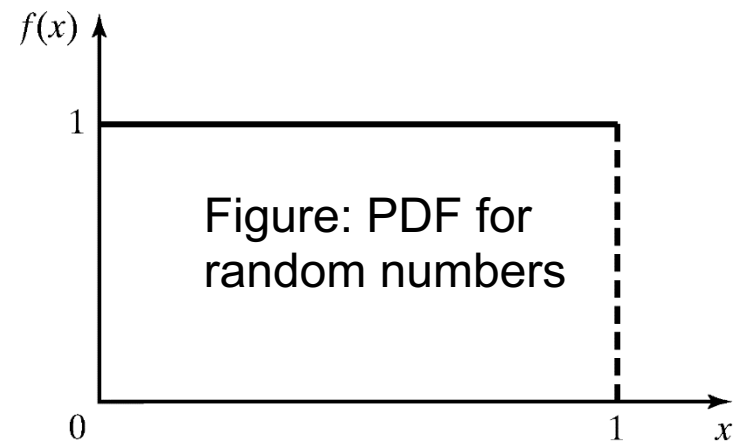


Properties of Random Numbers



- A sequence of random numbers $\{R_1, R_2, R_3, \dots\}$ must have the two following important statistical properties:
 - **Uniformity:** Probability of their individual generation is identical to other generated random numbers
 - **Independence:** Generating a random number does not affect generating other numbers
- Any random number, R_i , must be independently drawn from a uniform distribution with PDF:

$$f(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$
$$E(R) = \int_0^1 x dx = \frac{x^2}{2} \Big|_0^1 = \frac{1}{2}$$



Generation of Pseudo-Random Numbers



SHARIF
UNIVERSITY OF
TECHNOLOGY



Google Authenticator

- Why pseudo?
 - Because generating numbers using a **true random number generator** removes the capability of reproducing the numbers for comparative studies
- The main goal of pseudo random generator algorithms
 - To produce a sequence of numbers in $[0,1]$ that simulates, or imitates, the ideal properties of random numbers (RN)
- It seems that generating random numbers is a simple task, but this is not true
- Important considerations in RN routines:
 - Fast
 - Compatible with different computers
 - Have sufficiently long cycle
 - Replicable
 - Closely approximate the ideal statistical properties of uniformity and independence



What We Learn Regarding the Generation of Random Numbers In This Lecture



SHARIF
UNIVERSITY OF
TECHNOLOGY

- Techniques for random number generation:
 - Linear Congruential Method (LCM)
 - Most widely-used technique
 - Combined Linear Congruential Generators (CLCG)
- Random-Number Streams



Linear Congruential Method



- In LCM, we first produce a sequence of integers, $\{X_1, X_2, \dots\}$ between 0 and $m-1$
 - This is done by the following **recursive** relationship:

$$X_{i+1} = (aX_i + c) \bmod m, \quad i = 0, 1, 2, \dots$$

The multiplier The increment The modulus

- Where, X_0 is known as the initial value or **seed number**
 - Since we are using modulus m , integer values range in $[0, m-1]$
- The selection of the values for a , c , m , and X_0 drastically affects the **statistical properties** and the **cycle length** of the random numbers
- To convert the integers to random numbers: $R_i = \frac{X_i}{m}, \quad i = 1, 2, \dots$

Example



- Assume $X_0 = 27$, $a = 17$, $c = 43$, and $m = 100$
- Generate three random numbers using the LCM technique
 - The X_i and R_i values are:
 - $X_0 = 27$
 - $X_1 = (17 \times 27 + 43) \bmod 100 = 502 \bmod 100 = 2 \rightarrow R1 = 0.02$
 - $X_2 = (17 \times 2 + 43) \bmod 100 = 77 \rightarrow R2 = 0.77$
 - $X_3 = (17 \times 77 + 43) \bmod 100 = 52 \rightarrow R3 = 0.52$
 - ...

Maximum Density



- In addition to aspects of good generators, which we discussed before, there are two other **specifications**, which represent the **degree of uniformity** and **independence** in the generated random numbers by any technique including LCM
- Maximum density
 - Illustrates the gap between the generated values $R_i, i \in \{1, 2, \dots\}$
 - Algorithm must not leave large gaps on $[0, 1] \rightarrow$ No sparseness
 - Numbers should cover the entire $[0, 1]$ interval with small gaps
 - What is the reason for this sparseness?
 - Instead of continuous, R_i values are discrete, because R_i is generated based on X_i , where X_i is obtained from a discrete set $\{0, 1, \dots, m - 1\}$
 - So, there is a limited number of possible values for $R_i \in \{0, \frac{1}{m}, \dots, \frac{m-1}{m}\}$
 - Solution: a very large integer for modulus m

Maximum Period (1)



■ Maximum density (Cont.)

- As we increase m , algorithm **can** better cover $[0,1]$, and the distance between two consecutive R_i fall
- Note: Why we say can? Because increasing the value of m **must be** accompanied with more numbers being generated
 - Example: If $m=10$, then you may sufficiently cover $[0,1]$ with generating 10 numbers, but generating same amount with $m=100$ would not suffice

■ Maximum period

- To achieve maximum density and avoid cycling
- This is achieved by proper choice of a , c , m , and X_0

■ Most computers use a binary representation of numbers

- Therefore, the value of m is also considered as power of 2 (or close)
 - Based on the processing capabilities of the host, m could be high

Maximum Period (2)



- Assume $c \neq 0$:
 - In this case, LCM is called as **Mixed Congruential Method**
 - If m is power of 2 (2^b), and c is prime to m (their gcd is 1)
 - And if $a = 1 + 4k$ ($k \in \{0,1,2, \dots\}$)
 - Then, period of this algorithm is: $P = m = 2^b$
- Assume $c = 0$:
 - In this case, LCM is called as **Multiplicative Congruential Method**
 - If X_0 is odd, and m is power of 2 (2^b)
 - And a could be written in either of following formats:
 - $a = 3 + 8k$ or $a = 5 + 8k$ ($k \in \{0,1,2, \dots\}$)
 - Then, period of this algorithm is: $P = m/4 = 2^{b-2}$
 - If m is a prime number, and a comes with a specification that $a^k - 1 \bmod m = 0$ (k must be the minimum possible value)
 - We can proof that $k = m - 1$, and the period is: $P = k = m - 1$



Combined Linear Congruential Generators (1)



SHARIF
UNIVERSITY OF
TECHNOLOGY

- Due to the increasing complexity of systems, longer period generators is required for their simulation
 - Example: aviation, which we need to consider any bit-level failures
 - Even $P = 2^{31} - 1$ may not suffice
- So, we need to increase P , but how?
 - Combine two or more multiplicative congruential generators
- Let $X_{i,1}, X_{i,2}, \dots, X_{i,k}$ be the i th output from k different multiplicative congruential generators with the following considerations for the j th generator:
 - It has a prime modulus m_j , and multiplier a_j , which its period is $m_j - 1$
 - Then, this generator produces integers $X_{i,j}$, which are approximately distributed uniformly on $[1, m_j - 1]$
 - We could say $W_{i,j} = X_{i,j} - 1$ are approximately uniform on $[0, m - 2]$



Combined Linear Congruential Generators (2)



- The mixed generator can now generate its own numbers based on the following equation:

$$X_i = \left(\sum_{j=1}^k (-1)^{j-1} X_{i,j} \right) \bmod m_1 - 1$$

- As you can see, the -1 coefficient puts +, and – every other $X_{i,j}$ s
- Example: if $k=2$ (two generators are combined)

- $X_i = (-1)^0 X_{i,1} + (-1)^1 X_{i,2} = X_{i,1} - X_{i,2}$

- Then, the final random numbers are generated as follows:

$$R_i = \begin{cases} \frac{X_i}{m_1}, X_i > 0 \\ \frac{m_1 - 1}{m_1}, X_i = 0 \end{cases}$$

- The period of this generator: $P = \frac{(m_1 - 1)(m_2 - 1) \dots (m_k - 1)}{2^{k-1}}$



Example (1)

- For 32-bit computers, Ecuyer [1988] suggests combining $k=2$ generators with $m_1 = 2,147,483,563$, $a_1 = 40,014$, $m_2 = 2,147,483,399$, and $a_2 = 20,692$
- The following algorithm is executed to generate random numbers:
 - Step 1: Seed selection
 - For the 1st generator, $X_{0,1}$ is in the range $[1, 2,147,483,562]$
 - For the 2nd generator, $X_{0,2}$ is in the range $[1, 2,147,483,398]$
 - Step 2: Calculating $X_{i,j}$ values in every generator
 - 1st generator: $X_{i+1,1} = 40,014X_{i,1} \bmod 2,147,483,563$
 - 2nd generator: $X_{i+1,2} = 20,692 X_{i,2} \bmod 2,147,483,399$
 - Step 3: Combining
 - $X_i = (X_{i,1} - X_{i,2}) \bmod 2,147,483,562$

Example (2)



- Step 4: Obtaining R_i values

$$R_i = \begin{cases} \frac{X_i}{2,147,483,563}, & X_i > 0 \\ \frac{2,147,483,562}{2,147,483,563}, & X_i = 0 \end{cases}$$

- Step 5: Set $i=i+1$, go back to step 2

- This generator has a period equal to: $\frac{(m_1-1)(m_2-1)}{2} \sim 2 \times 10^{18}$
 - As mentioned before, even this period is not enough for many simulations
- Note: In combined generators, j generators are used
 - In the previous example, if we used 1 generator, the period would be 2×10^9 , and if we used 3 generators, the period would be 2×10^{27}
 - Therefore, if you need higher period, combine more generators

Random-Number Streams (1)



- The seed X_0 in LCM, is the integer value that initializes the random-number sequence
 - After the period, algorithm starts generating the same numbers starting from X_0 again
 - In the generated sequence, $\{X_0, X_1, X_2, \dots, X_P, X_0, X_1, \dots\}$, any value can be used as the seed value
 - While the generated sequence will be still the same
- A random-number stream:
 - Refers to **separated sequences** from a general sequence $\{X_0, X_1, X_2, \dots, X_P\}$, starting from a specific seed taken from this sequence, and ending to another number
 - Every stream could be considered as an output for separate generators
 - They follow the essential specifications for random numbers if their main generator supports them, i.e., uniformity and independence

Random-Number Streams (2)



- If the streams are composed of b numbers, then stream i could be defined by its starting seed: $S_i = X_{b \times (i-1)}$
 - Where S_i indicates the seed for stream i
 - Which values could be assigned to i ?
- Older generators use $b = 10^5$, while newer generators use $b = 10^{37}$
- Other important notes:
 - A single random number generator with k streams can be used just like k distinct virtual random number generators
 - To have a fair comparison between two or more systems, we must use the **same streams**, even if their source is identical
 - When to use streams?
 - Period of the main generator is high (10^{18}), while we don't need all of them → numbers are divided into several smaller streams, e.g., 10000 numbers



Tests for Random Numbers



- R_i must be tested for uniformity, and independence
- Tests are categorized in two groups:
 - Testing for uniformity:

Generated R_i are uniformly distributed on $[0,1]$ $H0: R_i \sim U[0,1]$

Generated R_i are not uniformly distributed on $[0,1]$ $H1: R_i \sim U[0,1]$

- Failure to reject the null hypothesis ($H0$) means that evidence of non-uniformity has not been detected
 - So, **more tests** are required to prove R_i s are not distributed uniformly

- Testing for independence:

Generated R_i are independent $H0: R_i \sim \text{independently}$

Generated R_i are not independent $H1: R_i \sim \text{independently}$

- Failure to reject the null hypothesis ($H0$) means that evidence of dependence has not been detected

Level of Significance



- Regarding the level of dependability in acceptance, and rejection of hypothesizes testing techniques, there is a concept called **level of significance**
 - Level of significance indicates the probability of rejecting H_0 when it is true:
$$\alpha = P(\text{Reject } H_0 | H_0 \text{ is True})$$
 - Consider a set of random numbers, which are uniformly distributed, but our test indicates they are not
 - This mistake could be measured with a probability value α
- This is acceptable as we know that any test has a level of error, and precision

When to Use These Tests?



- If a well-known simulator or random-number generator is used, it is probably unnecessary to test
 - Cooja, iFogSim, CloudSim, NS2, NS3, ...
- When it is necessary?
 - If the generator is not explicitly known or documented
 - It is designed for the first time
 - We are not sure that appropriate testing is conducted before
- Types of tests:
 - **Theoretical tests:** Evaluate the choices of m , a , and c without actually generating any numbers
 - **Empirical tests:** Based on the structure of the algorithm, sequences of numbers are produced, and then, the numbers are tested
 - Must be conducted several times for high level of dependability



Frequency Tests



- These family of tests are used to test the **uniformity** of the produced random numbers
- Two different methods:
 - Kolmogorov-Smirnov test
 - Chi-square test
- Both of these tests try to determine the level of accordance between the distribution of random numbers with the uniform distribution
- Both of these tests are applied based on the H_0 hypothesis
 - Assumes that the accordance is high enough, and the random number generator produces uniform numbers

Kolmogorov-Smirnov Test (1)



- Compares the continuous CDF, $F(x)$, of the uniform distribution with the empirical CDF obtained from the N sample observations represented with $S_N(x)$
 - N here indicates the number of generated random numbers
 - We know that for uniform distribution: $F(x) = x, 0 \leq x \leq 1$
 - If the generated numbers are $\{R_1, R_2, \dots, R_N\}$, then the empirical CDF, $S_N(x)$ is defined as follows:

$$S_N(x) = \frac{\text{Number of } R_i, \text{ which are } \leq x}{N}$$

- Kolmogorov-Smirnov works based on the distance between $F(x)$, and $S_N(x)$:

$$D = \text{Max}|F(x) - S_N(x)|$$

Kolmogorov-Smirnov Test (2)



- Sampling distribution of D is known
 - Tabulated in table A.8 of the textbook
 - Known as **critical value**
 - A function of N , and level of significance
 - Example: check it for $N=10$, and $\alpha = 0.05$
- Generally, a more powerful test is recommended

D_α

Level of significance

$D_\alpha = 0.410$

Degrees of Freedom (N)	$D_{0.10}$	$D_{0.05}$	$D_{0.01}$
1	0.950	0.975	0.995
2	0.776	0.842	0.929
3	0.642	0.708	0.828
4	0.564	0.624	0.733
5	0.510	0.565	0.669
6	0.470	0.521	0.618
7	0.438	0.486	0.577
8	0.411	0.457	0.543
9	0.388	0.432	0.514
10	0.368	0.410	0.490
11	0.352	0.391	0.468
12	0.338	0.375	0.450
13	0.325	0.361	0.433
14	0.314	0.349	0.418
15	0.304	0.338	0.404
16	0.295	0.328	0.392
17	0.286	0.318	0.381
18	0.278	0.309	0.371
19	0.272	0.301	0.363
20	0.264	0.294	0.356
25	0.24	0.27	0.32
30	0.22	0.24	0.29
35	0.21	0.23	0.27
Over 35	$\frac{1.22}{\sqrt{N}}$	$\frac{1.36}{\sqrt{N}}$	$\frac{1.63}{\sqrt{N}}$



Example (1)

- Suppose a generator has generated $N=5$ numbers: **0.44, 0.81, 0.14, 0.05, 0.93**
 - We want to test its uniformity with Kolmogorov-Smirnov
 - To simplify our calculations, we use a table
- 1st step: Sort R_i in ascending manner (rows **1**)
- 2nd step: Insert values corresponding to $\frac{i}{N} - R_i$, and $R_i - \frac{i-1}{n}$
 - Let's define the maximum value of row **3**, as D^+
 - $D^+ = \text{Max} \left\{ \frac{i}{N} - R_i \right\} = 0.26$
 - And, the maximum value in row **4** as D^-
 - $D^- = \text{Max} \left\{ R_i - \frac{i-1}{N} \right\} = 0.21$

1	$R_{(i)}$	0.05	0.14	0.44	0.81	0.93
2	i/N	0.20	0.40	0.60	0.80	1.00
3	$i/N - R_{(i)}$	0.15	0.26	0.16	-	0.07
4	$R_{(i)} - (i-1)/N$	0.05	-	0.04	0.21	0.13



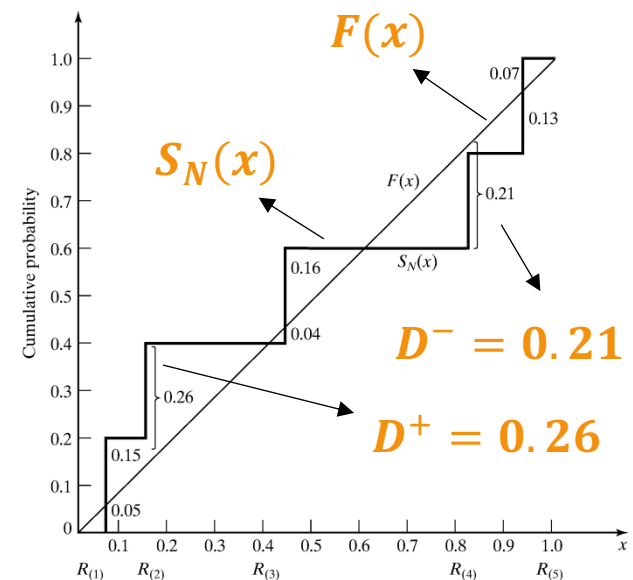
Example (2)

- 3rd step: Obtain D based on: $D = \text{Max}\{D^+, D^-\}$
 - $D = \text{Max}\{0.26, 0.21\} = 0.26$
- 4th step: Based on the specified level of significance (α), and the value of N, we determine D_α from the table
 - Assuming $\alpha = 0.05$, since $N=5$, $D_{0.05} = 0.565$
- 5th step: comparing D and D_α (critical value)
 - If $D > D_\alpha$, the null hypothesis H_0 will be rejected
 - Indicating that the uniformity of the random numbers is rejected
 - If $D \leq D_\alpha$, the null hypothesis H_0 is not rejected
 - We need more tests to reject it
 - But, with probability $1 - \alpha$, the uniformity hypothesis is correct
- Here, since $D < D_\alpha = 0.565 \rightarrow \mathbf{H_0 \text{ is not rejected}}$

Example (3)



- Since $\alpha = 0.05$, there is a 5% chance that we mistakenly reject the uniformity of these numbers
 - With 95% chance, if numbers are uniform, the test says so
- As you can see, the value of α , and the number of produced numbers have significant impact on the outcome and precision of your test
 - According to the table in slide 22, increasing N will reduce the value of D_α
 - So, D (from what we calculate) must get lower values in order to not reject the uniformity hypothesis
 - In other words, with increasing N, we expect that steps of $S_N(x)$ get closer to the uniform distribution line



Chi-square Test



- A more powerful testing approach, which operates based on classifying the data, and then using the following criterion:

$$X_0^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i}$$

- Where n indicates the number of classes
 - O_i represents the number of observations in ith class
 - E_i represents the number of observations in every class, if the data was truly uniform ($E_i = \frac{n}{N}$)
- Only valid for large samples, e.g. $N \geq 50$
 - This is the reason we must classify the samples



Classification in Chi-square Test (1)



- Assume that 100 numbers are produced with any RN generator algorithm

- They are illustrated in the table
- We need to test their uniformity with Chi-square test

0.34	0.90	0.25	0.89	0.87	0.44	0.12	0.21	0.46	0.67
0.83	0.76	0.79	0.64	0.70	0.81	0.94	0.74	0.22	0.74
0.96	0.99	0.77	0.67	0.56	0.41	0.52	0.73	0.99	0.02
0.47	0.30	0.17	0.82	0.56	0.05	0.45	0.31	0.78	0.05
0.79	0.71	0.23	0.19	0.82	0.93	0.65	0.37	0.39	0.42
0.99	0.17	0.99	0.46	0.05	0.66	0.10	0.42	0.18	0.49
0.37	0.51	0.54	0.01	0.81	0.28	0.69	0.34	0.75	0.49
0.72	0.43	0.56	0.97	0.30	0.94	0.96	0.58	0.73	0.05
0.06	0.39	0.84	0.24	0.40	0.64	0.40	0.19	0.79	0.62
0.18	0.26	0.97	0.88	0.64	0.47	0.60	0.11	0.29	0.78

- Assumptions:

- α is assumed to be 0.05
- 10 intervals (with identical length) are used for our classification, e.g., $[0,0.1)$, $[0.1,0.2)$, ..., $[0.9,1]$

- According to the definitions in the previous slide, and the equation for x_0^2 , we create a table and calculate all the values accordingly

Classification in Chi-square Test (2)



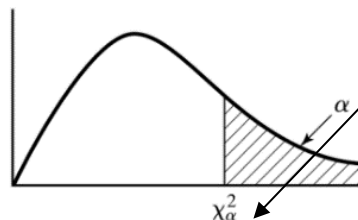
- First column indicates every class (from 1st to 10th)
- O_i indicates that how many numbers out of N, are placed within the i th class
 - E.g., 12 numbers are placed in the 5th interval [0.5,0.6)
- Regarding E_i
 - If these numbers were uniform, we expected that every interval is composed of equal number of RNs, i.e., $100/10 = 10$
 - This is a rational expectation, because in uniform distribution, everything is distributed equally between 0, and 1
- Three more columns are added to simplify our calculations

<i>Interval</i>	O_i	E_i	$O_i - E_i$	$(O_i - E_i)^2$	$\frac{(O_i - E_i)^2}{E_i}$
1	8	10	-2	4	0.4
2	8	10	-2	4	0.4
3	10	10	0	0	0.0
4	9	10	-1	1	0.1
5	12	10	2	4	0.4
6	8	10	-2	4	0.4
7	10	10	0	0	0.0
8	14	10	4	16	1.6
9	10	10	0	0	0.0
10	11	10	1	1	0.1
	100	100	0		3.4

Classification in Chi-square Test (3)



- It could be shown that the distribution of χ_0^2 values are approximately supporting the chi-square distribution with **n-1 degrees of freedom**, where n indicates the number of classes
 - Its critical values are tabulated in table A.8 as a function of degree of freedom, and level of significance
 - n is represented with ν
 - E.g., 10 classes $\rightarrow \nu = 9, \alpha = 0.05$
- Since
 $\chi_0^2 = 3.4 < \chi_{0.05}^2 = 16.9$
H0 is not rejected



ν	$\chi_{0.005}^2$	$\chi_{0.01}^2$	$\chi_{0.025}^2$	$\chi_{0.05}^2$	$\chi_{0.10}^2$
1	7.88	6.63	5.02	3.84	2.71
2	10.60	9.21	7.38	5.99	4.61
3	12.84	11.34	9.35	7.81	6.25
4	14.96	13.28	11.14	9.49	7.78
5	16.7	15.1	12.8	11.1	9.2
6	18.5	16.8	14.4	12.6	10.6
7	20.3	18.5	16.0	14.1	12.0
8	22.0	20.1	17.5	15.5	13.4
9	23.6	21.7	19.0	16.9	14.7
10	25.2	23.2	20.5	18.3	16.0
11	26.8	24.7	21.9	19.7	17.3
12	28.3	26.2	23.3	21.0	18.5
13	29.8	27.7	24.7	22.4	19.8
14	31.3	29.1	26.1	23.7	21.1
15	32.8	30.6	27.5	25.0	22.3
16	34.3	32.0	28.8	26.3	23.5
17	35.7	33.4	30.2	27.6	24.8
18	37.2	34.8	31.5	28.9	26.0
19	38.6	36.2	32.9	30.1	27.2
20	40.0	37.6	34.2	31.4	28.4
21	41.4	38.9	35.5	32.7	29.6
22	42.8	40.3	36.8	33.9	30.8
23	44.2	41.6	38.1	35.2	32.0
24	45.6	43.0	39.4	36.4	33.2
25	47.0	44.3	40.6	37.7	34.4
26	48.3	45.6	41.9	38.9	35.6
27	49.6	47.0	43.2	40.1	36.7
28	51.0	48.3	44.5	41.3	37.9
29	52.3	49.6	45.7	42.6	39.1
30	53.7	50.9	47.0	43.8	40.3
40	66.8	63.7	59.3	55.8	51.8
50	79.5	76.2	71.4	67.5	63.2
60	92.0	88.4	83.3	79.1	74.4
70	104.2	100.4	95.0	90.5	85.5
80	116.3	112.3	106.6	101.9	96.6
90	128.3	124.1	118.1	113.1	107.6
100	140.2	135.8	129.6	124.3	118.5

Tests for Autocorrelation (1)



- Another essential aspect of random numbers is their independency, which is tested by autocorrelation techniques
 - Test begins with the **ith number**, and selects a set of numbers based on a **lag value m**
 - The testing metric here is denoted with ρ_{im}
- Accordingly, the autocorrelation test is conducted between the following numbers:
$$\{R_i, R_{i+m}, R_{i+2m}, \dots, R_{i+(M+1)m}\}$$
 - Where M is the largest integer such that $i + (M + 1)m \leq N$
- Example: assume we start with the 3rd number, and lag is 5
 - If $N=30 \rightarrow$ Selected RNs for our test = $R_3, R_8, R_{13}, R_{18}, R_{23}, R_{28}$

Tests for Autocorrelation (2)



- There are two hypothesis for autocorrelation testing:

Generated R_i are independent $H_0: \rho_{im} = 0$

Generated R_i are correlated $H_1: \rho_{im} \neq 0$

- Assuming that the values are independent, for large values of M , we could say that the distribution of ρ_{im} estimator (denoted by $\hat{\rho}_{im}$) is approximately **normal**

- Therefore, we should compare the calculated statistical criterion with this distribution

- The statistical criterion for the autocorrelation test is defined as follows:

$$Z_0 = \frac{\hat{\rho}_{im}}{\hat{\sigma}_{\hat{\rho}_{im}}}$$

Tests for Autocorrelation (3)



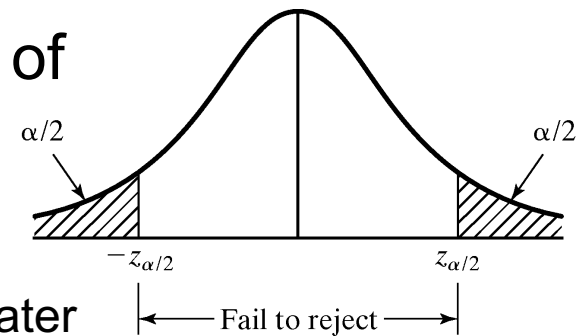
- Z_0 is distributed normally with mean=0 and variance=1
 - To obtain Z_0 , we need to first calculate two parameters:

$$\hat{\rho}_{im} = \frac{1}{M+1} \left[\sum_{k=0}^M R_{i+km} R_{i+(k+1)m} \right] - 0.25$$

$$\hat{\sigma}_{\rho_{im}} = \frac{\sqrt{13M+7}}{12(M+1)}$$

- If $-Z_{\frac{\alpha}{2}} \leq Z_0 \leq +Z_{\frac{\alpha}{2}}$ then, the hypothesis of independence cannot be rejected

- α represents the level of significance
- $Z_{\frac{\alpha}{2}}$ is obtained from the table, which we discuss later



Positive and Negative Autocorrelation



- If $\rho_{im} > 0$, the sequence of numbers has **positive autocorrelation**
 - This means high random numbers tend to be followed by high ones, and low numbers are followed by low numbers
- If $\rho_{im} < 0$, the sequence of numbers has **negative autocorrelation**
 - Indicates that low random numbers tend to be followed by high ones, and vice versa
- In either case, if $|\rho_{im}|$ is high, the random numbers have higher autocorrelation, while lower $|\rho_{im}|$ indicates lower autocorrelation



Example (1)

- We have produced 30 random numbers
 - Test the autocorrelation for the 3rd, 8th, 13th, and so on
- Accordingly, $i = 3, m = 5, N = 30$
 - $i + (M + 1)m \leq N \rightarrow M = 4$

$$\hat{\rho}_{35} = \frac{1}{4+1} \left[(0.23)(0.28) + (0.25)(0.33) + (0.33)(0.27) + (0.28)(0.05) + (0.05)(0.36) \right] - 0.25 = -0.1945$$

$$\hat{\sigma}_{\rho_{35}} = \frac{\sqrt{13(4)+7}}{12(4+1)} = 0.128$$

$$Z_0 = -\frac{0.1945}{0.1280} = -1.516$$

30 Generated Numbers

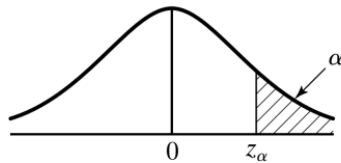
0.12	0.01	0.23	0.28	0.89	0.31	0.64	0.28	0.83	0.93
0.99	0.15	0.33	0.35	0.91	0.41	0.60	0.27	0.75	0.88
0.68	0.49	0.05	0.43	0.95	0.58	0.19	0.36	0.69	0.87

Example (2)

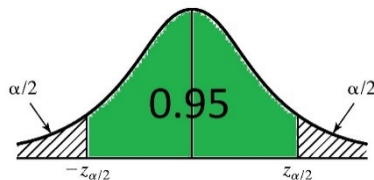


- From Table A.3, $Z_{0.025} = 1.96$, hence, the hypothesis is not rejected:

$$-1.96 \leq -1.516 \leq +1.96$$



$$\phi(z_\alpha) = \int_{-\infty}^{z_\alpha} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} du = 1 - \alpha$$



$$\frac{\alpha}{2} = 0.975$$

z_α	0.00	0.01	0.02	0.03	0.04	z_α
0.0	0.5000	0.5039	0.5078	0.5117	0.5159	0.0
0.1	0.5398	0.5437	0.5476	0.5517	0.5557	0.1
0.2	0.5798	0.5837	0.5876	0.5909	0.5948	0.2
0.3	0.6179	0.6217	0.6255	0.6293	0.6330	0.3
0.4	0.6554	0.6591	0.6627	0.6664	0.6700	0.4
0.5	0.6914	0.6949	0.6984	0.7019	0.7054	0.5
0.6	0.7257	0.7291	0.7324	0.7356	0.7389	0.6
0.7	0.7580	0.7613	0.7644	0.7674	0.7704	0.7
0.8	0.7881	0.7913	0.7943	0.7973	0.7994	0.8
0.9	0.8159	0.8189	0.8219	0.8248	0.8269	0.9
1.0	0.8413	0.8438	0.8461	0.8483	0.8504	1.0
1.1	0.8643	0.8665	0.8686	0.8707	0.8728	1.1
1.2	0.8843	0.8864	0.8886	0.8907	0.8928	1.2
1.3	0.9032	0.9049	0.9068	0.9087	0.9106	1.3
1.4	0.9192	0.9207	0.9222	0.9236	0.9250	1.4
1.5	0.9332	0.9348	0.9364	0.9379	0.9394	1.5
1.6	0.9452	0.9467	0.9481	0.9495	0.9509	1.6
1.7	0.9554	0.9568	0.9582	0.9596	0.9609	1.7
1.8	0.9647	0.9660	0.9673	0.9686	0.9698	1.8
1.9	0.9712	0.9724	0.9735	0.9746	0.9757	1.9
2.0	0.9772	0.9781	0.9790	0.9799	0.9808	2.0
2.1	0.9824	0.9832	0.9841	0.9849	0.9857	2.1
2.2	0.9868	0.9876	0.9884	0.9891	0.9898	2.2
2.3	0.9899	0.9906	0.9913	0.9919	0.9926	2.3
2.4	0.9927	0.9932	0.9937	0.9941	0.9946	2.4
2.5	0.9949	0.9953	0.9957	0.9960	0.9964	2.5
2.6	0.9967	0.9970	0.9973	0.9976	0.9979	2.6
2.7	0.9981	0.9983	0.9985	0.9987	0.9989	2.7
2.8	0.9990	0.9991	0.9992	0.9993	0.9994	2.8
2.9	0.9995	0.9995	0.9996	0.9996	0.9997	2.9
3.0	0.9997	0.9997	0.9997	0.9997	0.9997	3.0
3.1	0.9997	0.9997	0.9997	0.9997	0.9997	3.1
3.2	0.9997	0.9997	0.9997	0.9997	0.9997	3.2
3.3	0.9997	0.9997	0.9997	0.9997	0.9997	3.3
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	3.4
3.5	0.9997	0.9997	0.9997	0.9997	0.9997	3.5
3.6	0.9997	0.9997	0.9997	0.9997	0.9997	3.6
3.7	0.9997	0.9997	0.9997	0.9997	0.9997	3.7
3.8	0.9997	0.9997	0.9997	0.9997	0.9997	3.8
3.9	0.9997	0.9997	0.9997	0.9997	0.9997	3.9

z_α	0.05	0.06	0.07	0.08	0.09	z_α
0.0	0.5199	0.5239	0.5279	0.5318	0.5357	0.0
0.1	0.5596	0.5636	0.5675	0.5714	0.5753	0.1
0.2	0.5987	0.6026	0.6064	0.6102	0.6140	0.2
0.3	0.6368	0.6406	0.6443	0.6480	0.6517	0.3
0.4	0.6736	0.6772	0.6808	0.6843	0.6878	0.4
0.5	0.7088	0.7123	0.7157	0.7190	0.7224	0.5
0.6	0.7421	0.7454	0.7486	0.7517	0.7549	0.6
0.7	0.7737	0.7767	0.7796	0.7825	0.7854	0.7
0.8	0.8023	0.8051	0.8078	0.8104	0.8129	0.8
0.9	0.8249	0.8314	0.8337	0.8360	0.8381	0.9
1.0	0.8531	0.8554	0.8577	0.8599	0.8621	1.0
1.1	0.8749	0.8771	0.8793	0.8814	0.8836	1.1
1.2	0.8943	0.8964	0.8985	0.8997	0.9017	1.2
1.3	0.9119	0.9138	0.9156	0.9174	0.9191	1.3
1.4	0.9267	0.9285	0.9299	0.9316	0.9331	1.4
1.5	0.9394	0.9406	0.9419	0.9431	0.9443	1.5
1.6	0.9503	0.9515	0.9527	0.9538	0.9549	1.6
1.7	0.9599	0.9610	0.9621	0.9632	0.9643	1.7
1.8	0.9688	0.9698	0.9708	0.9718	0.9728	1.8
1.9	0.9772	0.9781	0.9790	0.9799	0.9808	1.9
2.0	0.9824	0.9832	0.9841	0.9849	0.9857	2.0
2.1	0.9868	0.9876	0.9884	0.9891	0.9898	2.1
2.2	0.9899	0.9906	0.9913	0.9919	0.9926	2.2
2.3	0.9927	0.9932	0.9937	0.9941	0.9946	2.3
2.4	0.9949	0.9953	0.9957	0.9960	0.9964	2.4
2.5	0.9967	0.9970	0.9973	0.9976	0.9979	2.5
2.6	0.9981	0.9983	0.9985	0.9987	0.9989	2.6
2.7	0.9990	0.9991	0.9992	0.9993	0.9994	2.7
2.8	0.9995	0.9995	0.9996	0.9996	0.9997	2.8
2.9	0.9997	0.9997	0.9997	0.9997	0.9997	2.9
3.0	0.9997	0.9997	0.9997	0.9997	0.9997	3.0
3.1	0.9997	0.9997	0.9997	0.9997	0.9997	3.1
3.2	0.9997	0.9997	0.9997	0.9997	0.9997	3.2
3.3	0.9997	0.9997	0.9997	0.9997	0.9997	3.3
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	3.4
3.5	0.9997	0.9997	0.9997	0.9997	0.9997	3.5
3.6	0.9997	0.9997	0.9997	0.9997	0.9997	3.6
3.7	0.9997	0.9997	0.9997	0.9997	0.9997	3.7
3.8	0.9997	0.9997	0.9997	0.9997	0.9997	3.8
3.9	0.9997	0.9997	0.9997	0.9997	0.9997	3.9

Example (3)



- In the previous example, if we obtain $Z_0 = 1.99$, with the same level of significance, the independency hypothesis would be rejected
 - We may have started from the 4th number
 - We may started from the 5th number
 - We may used a lag value equal to 3
 -
- Therefore, we could see that many sequences could be selected from the entire generated random numbers to be tested
 - If in many sequences the test could not reject H_0 , then you can conclude with more certainty that they are independent

Shortcomings of Autocorrelation Test



- It is not very sensitive for small values of M , particularly when the numbers being tested are on the low side
 - Example: Assume that all the numbers used to calculate $\hat{\rho}_{im}$ are 0 $\rightarrow \hat{\rho}_{im} = -0.25 \rightarrow Z_0 = -1.95 \rightarrow$ independence is not rejected while we know is not true
- The **fishing problem** when performing numerous tests:
 - Recall: $\alpha = 0.05$ indicates that there is a 5% chance of rejecting a true hypothesis for the selected set of numbers
 - With 95%, a true hypothesis is accepted
 - We know that different sets of numbers could be selected for the test specifically when N is high (based on i , m , and M)
 - Now assume if 10 independent sequences are examined
 - The probability of not rejecting the true independence is $0.95^{10} = 0.60$
 - Hence, the probability of detecting a false autocorrelation would be 40% ☹



Summary



- In this chapter, we described:
 - Generation of random numbers
 - Testing for uniformity and independence
- Caution:
 - Even generators that have been used for years (some of which still in use) are found to be inadequate
 - Those generators that have been designed by us must be tested for sure to pass independency and uniformity
 - Also, even if generated numbers pass all the tests, some underlying patterns might have gone undetected
 - This chapter provided only the basic stuff