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

SHARIF UNIVERSITY OF TECHNOLOGY

- A sequence of random numbers $\{R_1, R_2, R_3, \ldots\}$ 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:

$$f(x) = \begin{cases} 1, & 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$$

$$E(R) = \int_0^1 x\,dx = \left.\frac{x^2}{2}\right|_0^1 = \frac{1}{2}$$

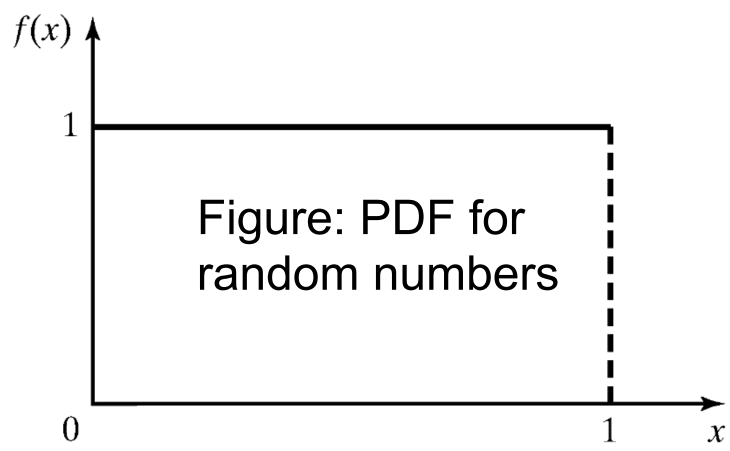


Figure: PDF for random numbers

Generation of Pseudo-Random Numbers

SHARIF UNIVERSITY OF TECHNOLOGY

- Why pseudo?
  - Because generating numbers using a **true random number generator** removes the capability of reproducing the numbers for comparative studies

Google Authticator

- 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, $\{X_1, X_2, \ldots\}$ between 0 and m-1
  - This is done by the following recursive relationship:

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

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 directly affects the **statistical properties** and the **cycle length** of the random numbers
- To convert the integers to random numbers: $R_i = \dfrac{X_i}{m}, \quad i = 1,2,\ldots$

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:
    - $X_0 = 27$
    - $X_1 = (17\times 27 + 43)\ mod\ 100 = 502\ mod\ 100 = \mathbf{2} \rightarrow R1 = 0.02$
    - $X_2 = (17\times \mathbf{2} + 43)\ mod\ 100 = 77 \rightarrow R2 = 0.77$
    - $X_3 = (17\times 77 + 43)\ mod\ 100 = 52 \rightarrow R3 = 0.52$
  - Calculate the next three random numbers based on the given parameters

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
- **1) 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] → 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
- **2) Maximum period**
  - To achieve maximum density and avoid cycling
  - This is achieved by proper choice of a, c, m, and X_0
    - Based on the processing capabilities of the host, m could be high
      - Most computers use a binary representation of numbers
      - Therefore, the value of m is also considered as power of 2 (or close)

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: $\boldsymbol{P = m = 2^b}$
- 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: $\boldsymbol{P = m/4 = 2^{b-2}}$
  - If m is a prime number, and k is the minimum possible value, where $(a^k - 1) \bmod \mathrm{m} = 0$
    - It could be seen that $k = m - 1$, and the period is: $\boldsymbol{P = k = m - 1}$

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 ith output from k different multiplicative congruential generators with the following considerations for the jth 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 $[0, m_j - 2]$

Combined Linear Congruential Generators (2)

SHARIF UNIVERSITY OF TECHNOLOGY

- 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)
    - $X_i = (-1)^0 X_{i,1} + (-1)^1 X_{i,2} = \mathrm{X}_{\mathrm{i},1} - \mathrm{X}_{\mathrm{i},2}$
- Then, the final random numbers are generated as follows:

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

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

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: $X_{i+1,1} = 40{,}014 X_{i,1} \; mod \; 2{,}147{,}483{,}563$
    - 2nd generator: $X_{i+1,2} = 20{,}692 \; X_{i,2} \; mod \; 2{,}147{,}483{,}399$
  - Step 3: Combining
    - $X_i = (X_{i,1} - X_{i,2}) \; mod \; 2{,}147{,}483{,}562$

Example (2)

SHARIF UNIVERSITY OF TECHNOLOGY

- Step 4: Obtaining R_i values

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

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

- This generator has a period equal to: $\frac{(m_1-1)(m_2-1)}{2} \sim 2\times10^{18}$
  - 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

SHARIF UNIVERSITY OF TECHNOLOGY

- 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] $H0: \quad Ri \sim U[0,1]$

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

    - 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_is are not distributed uniformly
  - Testing for independence:

    Generated R_i are independent $H0: \quad Ri \sim independently$

    Generated R_i are not independent $H1: \quad Ri \sim independently$

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

Level of Significance

SHARIF UNIVERSITY OF TECHNOLOGY

- 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 H0 when it is true:

$$\alpha = P(Reject\ H0|H0\ is\ True)$$

  - 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, OMNET++, 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 H0 hypothesis
  - Assumes that the accordance is high enough, and the random number generator produces uniform numbers

Kolmogorov-Smirnov Test (1)

SHARIF UNIVERSITY OF TECHNOLOGY

- 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: $F(x) = x, \quad 0 \le x \le 1$
  - If the generated numbers are $\{R_1, R_2, \ldots, R_N\}$, then the empirical CDF, $S_N(x)$ is defined as follows:

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

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

$$D = Max|F(x) - S_N(x)|$$

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 $\alpha = 0.05$
  - Critical value will be used to check the H0 hypothesis
    - Explained in the example
- 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)

SHARIF UNIVERSITY OF TECHNOLOGY

- 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 (row 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^+
    - $D^+ = Max\left\{\frac{i}{N} - R_i\right\} = 0.26$
  - And, the maximum value in row 4 as D^-
    - $D^- = Max\left\{R_i - \frac{i-1}{N}\right\} = 0.21$

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

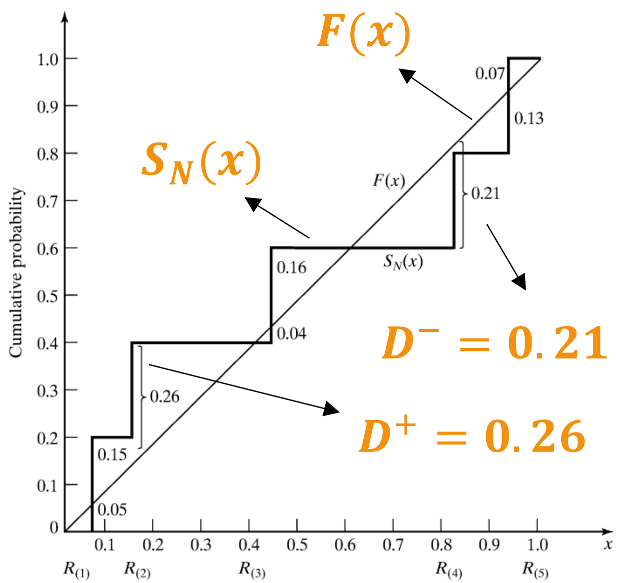
Example (2)

SHARIF UNIVERSITY OF TECHNOLOGY

- 3rd step: Obtain D based on: $D = Max\{D^+, D^-\}$
  - $D = Max\{0.26,0.21\} = 0.26$
- 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 H0 will be rejected
    - Indicating that the uniformity of the random numbers is rejected
  - If $D \leq D_\alpha$, the null hypothesis H0 is not rejected
    - We need more tests to reject it
    - But, with probability $1 - \alpha$, the uniformity hypothesis is correct
- Here, since $D < D_\alpha = 0.565$ → **H0 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 ith class
  - E.g., 12 numbers are placed in the 5th interval [0.5,0.6)

Interval	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

- 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

Classification in Chi-square Test (3)

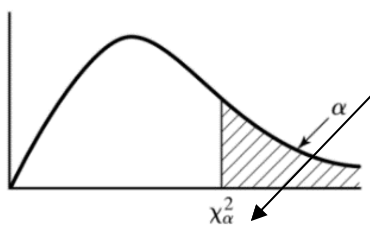
- It could be shown that the distribution of x_0^2 values are approximately supporting the chi-square distribution with **n-1 degrees of freedom,** where n indicates the number of classes
  - Its critical values are tabulated in table A.8 as a function of degree of freedom, and level of significance
    - n is represented with v
      - E.g., $10\ classes \rightarrow v = 9, \alpha = 0.05$

v	$\chi^2_{0.005}$	$\chi^2_{0.01}$	$\chi^2_{0.025}$	$\chi^2_{0.05}$	$\chi^2_{0.10}$
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	49.6	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

Since

$x_0^2 = 3.4 < x_{0.05}^2 = 16.9$

H0 is not rejected



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**
    - Accordingly, one of the testing parameters here is denoted with $\boldsymbol{\rho_{im}}$
- Hence, 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 → Selected RNs for our test = $R_3, R_8, R_{13}, R_{18}, R_{23}, R_{28}$

Tests for Autocorrelation (2)

SHARIF UNIVERSITY OF TECHNOLOGY

- There are two hypothesis for autocorrelation testing:

Generated R_i are independent $\quad H0: \quad \rho_{im} = 0$

Generated R_i are correlated $\quad H1: \quad \rho_{im} \neq 0$

- Assuming that the values are independent, for large values of M, we could say that the distribution of $\boldsymbol{\rho_{im}}$ estimator (denoted by $\hat{\boldsymbol{\rho}}_{\boldsymbol{im}}$) 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:

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

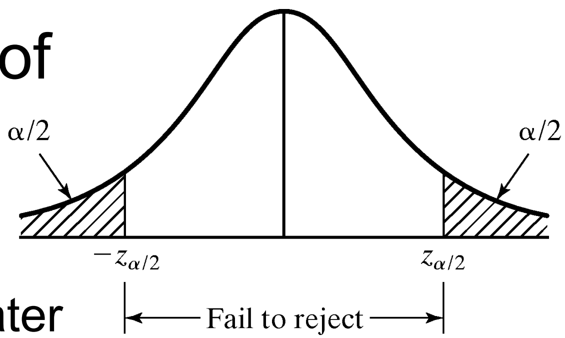
Tests for Autocorrelation (3)

- 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, $i = 3, m = 5,\ N = 30$
  - $i + (M + 1)m \le N \rightarrow M = 4$

$$\hat{\rho}_{35} = \frac{1}{4+1}\begin{bmatrix}(0.23)(0.28) + (0.28)(0.33) + (0.33)(0.27) \\ +(0.27)(0.05) + (0.05)(0.36)\end{bmatrix} - 0.25 = -0.1945$$

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

$$Z_0 = -\frac{0.1945}{0.128} = -1.519$$

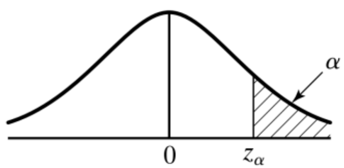
30 Generated Numbers

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

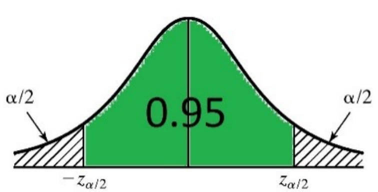
Example (2)

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

$$-1.96 \leq -1.519 \leq +1.96$$



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



$$1 - \frac{\alpha}{2} = 0.975$$

z_α	0.00	0.01	0.02	0.03	0.04	z_α
0.0	0.500 00	0.503 99	0.507 98	0.511 97	0.515 95	**0.0**
0.1	0.539 83	0.543 79	0.547 76	0.551 72	0.555 67	**0.1**
0.2	0.579 26	0.583 17	0.587 06	0.590 95	0.594 83	**0.2**
0.3	0.617 91	0.621 72	0.625 51	0.629 30	0.633 07	**0.3**
0.4	0.655 42	0.659 10	0.662 76	0.666 40	0.670 03	**0.4**
0.5	0.691 46	0.694 97	0.698 47	0.701 94	0.705 40	**0.5**
0.6	0.725 75	0.729 07	0.732 37	0.735 65	0.738 91	**0.6**
0.7	0.758 03	0.761 15	0.764 24	0.767 30	0.770 35	**0.7**
0.8	0.788 14	0.791 03	0.793 89	0.796 73	0.799 54	**0.8**
0.9	0.815 94	0.818 59	0.821 21	0.823 81	0.826 39	**0.9**
1.0	0.841 34	0.843 75	0.846 13	0.848 49	0.850 83	**1.0**
1.1	0.864 33	0.866 50	0.868 64	0.870 76	0.872 85	**1.1**
1.2	0.884 93	0.886 86	0.888 77	0.890 65	0.892 51	**1.2**
1.3	0.903 20	0.904 90	0.906 58	0.908 24	0.909 88	**1.3**
1.4	0.919 24	0.920 73	0.922 19	0.923 64	0.925 06	**1.4**
1.5	0.933 19	0.934 48	0.935 74	0.936 99	0.938 22	**1.5**
1.6	0.945 20	0.946 30	0.947 38	0.948 45	0.949 50	**1.6**
1.7	0.955 43	0.956 37	0.957 28	0.958 18	0.959 07	**1.7**
1.8	0.964 07	0.964 85	0.965 62	0.966 37	0.967 11	**1.8**
1.9	0.971 28	0.971 93	0.972 57	0.973 20	0.973 81	**1.9**
2.0	0.977 25	0.977 78	0.978 31	0.978 82	0.979 32	**2.0**
2.1	0.982 14	0.982 57	0.983 00	0.983 41	0.983 82	**2.1**
2.2	0.986 10	0.986 45	0.986 79	0.987 13	0.987 45	**2.2**
2.3	0.989 28	0.989 56	0.989 83	0.990 10	0.990 36	**2.3**
2.4	0.991 80	0.992 02	0.992 24	0.992 45	0.992 66	**2.4**
2.5	0.993 79	0.993 96	0.994 13	0.994 30	0.994 46	**2.5**
2.6	0.995 34	0.995 47	0.995 60	0.995 73	0.995 85	**2.6**
2.7	0.996 53	0.996 64	0.996 74	0.996 83	0.996 93	**2.7**
2.8	0.997 44	0.997 52	0.997 60	0.997 67	0.997 74	**2.8**
2.9	0.998 13	0.998 19	0.998 25	0.998 31	0.998 36	**2.9**
3.0	0.998 65	0.998 69	0.998 74	0.998 78	0.998 82	**3.0**
3.1	0.999 03	0.999 06	0.999 10	0.999 13	0.999 16	**3.1**
3.2	0.999 31	0.999 34	0.999 36	0.999 38	0.999 40	**3.2**
3.3	0.999 52	0.999 53	0.999 55	0.999 57	0.999 58	**3.3**
3.4	0.999 66	0.999 68	0.999 69	0.999 70	0.999 71	**3.4**
3.5	0.999 77	0.999 78	0.999 78	0.999 79	0.999 80	**3.5**
3.6	0.999 84	0.999 85	0.999 85	0.999 86	0.999 86	**3.6**
3.7	0.999 89	0.999 90	0.999 90	0.999 90	0.999 91	**3.7**
3.8	0.999 93	0.999 93	0.999 93	0.999 94	0.999 94	**3.8**
3.9	0.999 95	0.999 95	0.999 96	0.999 96	0.999 96	**3.9**

z_α	0.05	$\boxed{0.06}$	0.07	0.08	0.09	z_α
0.0	0.519 94	0.523 92	0.527 90	0.531 88	0.535 86	**0.0**
0.1	0.559 62	0.563 56	0.567 49	0.571 42	0.575 34	**0.1**
0.2	0.598 71	0.602 57	0.606 42	0.610 26	0.614 09	**0.2**
0.3	0.636 83	0.640 58	0.644 31	0.648 03	0.651 73	**0.3**
0.4	0.673 64	0.677 24	0.680 82	0.684 38	0.687 93	**0.4**
0.5	0.708 84	0.712 26	0.715 66	0.719 04	0.722 40	**0.5**
0.6	0.742 15	0.745 37	0.748 57	0.751 75	0.754 90	**0.6**
0.7	0.773 37	0.776 37	0.779 35	0.782 30	0.785 23	**0.7**
0.8	0.802 34	0.805 10	0.807 85	0.810 57	0.813 27	**0.8**
0.9	0.824 94	0.831 47	0.833 97	0.836 46	0.838 91	**0.9**
1.0	0.853 14	0.855 43	0.857 69	0.859 93	0.862 14	**1.0**
1.1	0.874 93	0.876 97	0.879 00	0.881 00	0.882 97	**1.1**
1.2	0.894 35	0.896 16	0.897 96	0.899 73	0.901 47	**1.2**
1.3	0.911 49	0.913 08	0.914 65	0.916 21	0.917 73	**1.3**
1.4	0.926 47	0.927 85	0.929 22	0.930 56	0.931 89	**1.4**
1.5	0.939 43	0.940 62	0.941 79	0.942 95	0.944 08	**1.5**
1.6	0.950 53	0.951 54	0.952 54	0.953 52	0.954 48	**1.6**
1.7	0.959 94	0.960 80	0.961 64	0.962 46	0.963 27	**1.7**
1.8	0.967 84	0.968 56	0.969 26	0.969 95	0.970 62	**1.8**
$\boxed{\mathbf{1.9}}$	0.974 41	$\boxed{0.975\ 00}$	0.975 58	0.976 15	0.976 70	**1.9**
2.0	0.979 82	0.980 30	0.980 77	0.981 24	0.981 69	**2.0**
2.1	0.984 22	0.984 61	0.985 00	0.985 37	0.985 74	**2.1**
2.2	0.987 78	0.988 09	0.988 40	0.988 70	0.988 99	**2.2**
2.3	0.990 61	0.990 86	0.991 11	0.991 34	0.991 58	**2.3**
2.4	0.992 86	0.993 05	0.993 24	0.993 43	0.993 61	**2.4**
2.5	0.994 61	0.994 77	0.994 92	0.995 06	0.995 20	**2.5**
2.6	0.995 98	0.996 09	0.996 21	0.996 32	0.996 43	**2.6**
2.7	0.997 02	0.997 11	0.997 20	0.997 28	0.997 36	**2.7**
2.8	0.997 81	0.997 88	0.997 95	0.998 01	0.998 07	**2.8**
2.9	0.998 41	0.998 46	0.998 51	0.998 56	0.998 61	**2.9**
3.0	0.998 86	0.998 89	0.998 93	0.998 97	0.999 00	**3.0**
3.1	0.999 18	0.999 21	0.999 24	0.999 26	0.999 29	**3.1**
3.2	0.999 42	0.999 44	0.999 46	0.999 48	0.999 50	**3.2**
3.3	0.999 60	0.999 61	0.999 62	0.999 64	0.999 65	**3.3**
3.4	0.999 72	0.999 73	0.999 74	0.999 75	0.999 76	**3.4**
3.5	0.999 81	0.999 81	0.999 82	0.999 83	0.999 83	**3.5**
3.6	0.999 87	0.999 87	0.999 88	0.999 88	0.999 89	**3.6**
3.7	0.999 91	0.999 92	0.999 92	0.999 92	0.999 92	**3.7**
3.8	0.999 94	0.999 94	0.999 95	0.999 95	0.999 95	**3.8**
3.9	0.999 96	0.999 96	0.999 96	0.999 97	0.999 97	**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 different sequences could be selected** from the entire generated random numbers to be tested
  - If in many sequences the test could not reject H0, then you can conclude with **more certainty** that they are independent

Shortcomings of Autocorrelation Test

SHARIF
UNIVERSITY OF
TECHNOLOGY

- 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 $\hat{\boldsymbol{\rho}}_{\boldsymbol{im}}$ are 0 → $\hat{\boldsymbol{\rho}}_{\boldsymbol{im}} = \mathbf{-0.25}$ → $Z_0 = -1.95$ → 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 (1)

- 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

Summary (2)

- There are many advanced versions of PRNG algorithms:
  - Mersenne Twister (MT)
    - A widely used PRNG
    - Long period $(2^{19937} - 1)$
    - High-quality randomness
    - Fast generation speed
  - Xorshift Generators
    - Simple but effective
  - PCG (Permuted Congruential Generator)
    - Good statistical quality and high performance
  - Well Equi-distributed Long-period Linear (WELL) Generators
  - CMWC (Complementary Multiply with Carry)

Summary (3)

- There are many advanced versions of PRNG algorithms:
  - ChaCha20
    - It is known for its simplicity, speed, and cryptographic security
  - Philox
    - Philox is a family of cryptographic pseudorandom number generators designed for parallel computation
    - Used in applications requiring parallelism
      - Such as **GPU computing**
  - Threefry
    - Designed for parallel computation
    - It is used in scientific computing
  - AES-CTR (Advanced Encryption Standard in Counter Mode)
    - Is primarily designed for encryption
    - But is also used as a secure and efficient PRNG