Random Number Generation

Nuts and Bolts of Simulation

Radu Trîmbițaș

Faculty of Math. and CS

1st Semester 2010-2011

Introduction

- All the randomness required by the model is *simulated* by a random number generator (RNG)
- The output of a RNG is *assumed* to be a sequence of i.i.d. r.v. $U(0,1)$
- These random numbers are transformed as needed to simulate r.v. from different probability distributions
- The validity of transformation methods depend strongly on i.i.d. U(0,1) assumption
- This assumption is *false*, since RNG are simple deterministic programs trying to fool the users by producing a deterministic sequence that looks random

Properties of Random Numbers

- Two important statistical properties:
  - Uniformity
  - Independence.
- Random Number, R_i, must be independently drawn from a uniform distribution with pdf and cdf:

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

$$F(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x \leq 1 \\ 1, & x > 1 \end{cases}$$

$$E(X) = \frac{1}{2}, V(X) = \frac{1}{12}$$

Properties

Consequences of the uniformity and independence properties:

1. if $[0, 1]$ is divided into n clases of equal length, the expected number of observations in each interval is N/n, where N is the total number of observations
2. The probability of observing a value in a particular interval is independent of the previous value drawn

Generation of Pseudo-Random Numbers

- "Pseudo", because generating numbers using a known method removes the potential for true randomness.
- Goal: To produce a sequence of numbers in [0,1] that simulates, or imitates, the ideal properties of random numbers (RN).
- Problems or errors (departure from ideal randomness)
  1. generated numbers may not be u.d.
  2. discrete instead of continuous values
  3. mean and/or variance too high or too low
  4. dependence
     1. autocorrelation
     2. number successively higher or lower than adjacent numbers
     3. several numbers above the mean followed by several numbers below the mean

Generation of Pseudo-Random Numbers II

Important considerations in RN routines:

1. fast - a large number of RN is required - total cost maintained reasonable
2. portable - to different computers, OS, and programming languages
3. have sufficiently long period - period must be longer than the numbers of events to be generated
4. replicable - given the starting point (or conditions) it should be possible to generate the same set of random numbers, independent of the system to be simulated; useful for debugging and comparison
5. Closely approximate the ideal statistical properties of uniformity and independence.

Techniques for Generating Random Numbers

- Inventing techniques that seems to generate RN is easy
- Inventing techniques that really do produces sequences that appear to be independent, u.d. RN is incredibly difficult
- Techniques:
  1. Linear Congruential Method (LCM).
  2. Combined Linear Congruential Generators (CLCG).
  3. Feedback Shift Register Generators (FSRG)
  4. Random-Number Streams.

Linear Congruential Method

- To produce a sequence of integers, X_1, X_2, ... between 0 and $m-1$ by following a recursive relationship:

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

  a multiplier, c increment, m modulus, X_0 seed, called *Linear Congruential Generator* (LCG)

- $c = 0$ *Multiplicative Congruential Generator* (MCG)
- The selection of the values for a, c, m, and X_0 drastically affects the statistical properties and the cycle length.
- The random integers are being generated $[0, m-1]$, and to convert the integers to random numbers:

$$R_i = \frac{X_i}{m}, \qquad i = 1, 2, \ldots$$

Example

- Use $X0 = 27$, $a = 17$, $c = 43$, and $m = 100$.
- The X_i and R_i values are:

$$\begin{aligned}
X1 &= (17 \cdot 27 + 43) \bmod 100 = 502 \bmod 100 = 2, \qquad R1 = 0.02;\\
X2 &= (17 \cdot 2 + 32) \bmod 100 = 77, \qquad R2 = 0.77;\\
X3 &= (17 \cdot 77 + 32) \bmod 100 = 52, \qquad R3 = 0.52;\\
&\ldots
\end{aligned}$$

Characteristics of a Good Generator

- Maximum Density
  - Such that the values assumed by R_i, $i = 1, 2, \ldots$, leave no large gaps on $[0, 1]$
  - Problem: Instead of continuous, each R_i is discrete
  - Solution: a very large integer for modulus m
  - Approximation appears to be of little consequence
- Maximum Period
  - To achieve maximum density and avoid cycling.
  - Achieved by: proper choice of a, c, m, and X_0.
- Most digital computers use a binary representation of numbers
  - Speed and efficiency are aided by a modulus, m, to be (or close to) a power of 2.

Maximum period I

Theorem (Hull and Dobell,1962)

The LCG defined by (1) has the full period iff the following three conditions hold:

(a) $(m, c) = 1$

(b) q *prime,* $q|m \Longrightarrow q|a-1$

(c) $4|m \Longrightarrow 4|a-1$

Corollary

$X_{i+1} = (aX_i + c) \bmod 2^n$ *($c, n > 1$) has the full period if c is odd and $a = 4k+1$ for some k.*

Theorem

$X_{i+1} = aX_i \bmod 2^n$ *had period at most 2^{n-2}. This can be achieved when X_0 is odd and $a = 8k+3$ or $a = 8k+5$ for some k.*

Maximum period II

- For $m = 2^b$, and $c \neq 0$, maximum period is $P = m = 2^b$ – achieved when $(c, m) = 1$, $a = 4k + 1$, k integer
- For $m = 2^b$, and $c = 0$, maximum period is $P = m/4 = 2^{b-2}$ – achieved when X_0 odd and $a = 8k + 3$ or $a = 8k + 5$, k natural
- For m prime and $c = 0$, $P = m - 1$ – achieved when $\min\{k : m|a^k - 1\} = m - 1$

Combined Linear Congruential Generators

- Reason: Longer period generator is needed because of the increasing complexity of stimulated systems.
- Approach: Combine two or more multiplicative congruential generators.
- Let $X_{i,1}, X_{i,2}, \ldots, X_{i,k}$, be the ith output from k different multiplicative congruential generators.
  - The jth generator:
  - Has prime modulus m_j and multiplier a_j and period is $m_j - 1$
  - Produces integers $X_{i,j}$ is approx ˜Uniform on integers in $[1, m-1]$
    - $W_{i,j} = X_{i,j} - 1$ is approx ˜Uniform on integers in $[1, m-2]$

Combined Linear Congruential Generators II

- Suggested form $X_{i,1}$, $X_{i,2}$, …, $X_{i,k}$ be k MCG with modulus m_i, multiplier a_i and period $m_i - 1$:

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

with

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

- the maximum period is

$$P = \frac{(m_1 - 1)(m_2 - 1)\dots(m_k - 1)}{2^{k-1}}$$

Example

- For 32-bit computers, L'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 $a2 = 20,692$. The algorithm becomes:

Step 1 $X_{1,0}$ in the range $[1; 2,147,483,562]$ for the 1st generator, $X_{2,0}$ in the range$[1; 2,147,483,398]$ for the 2nd generator. Set $j = 0$

Step 2 For each individual generator

$$X_{1,j+1} = 40,014 X_{1,j} \bmod 2,147,483,563$$
$$X_{2,j+1} = 40,692 X_{1,j} \bmod 2,147,483,399$$

Step 3 $X_{j+1} = (X_{1,j+1} - X_{2,j+1}) \bmod 2,147,483,562.$

Step 4 Return

$$R_{j+1} = \begin{cases} \dfrac{X_{j+1}}{2,147,483,563} & X_{j+1} > 0 \\ \dfrac{2,147,483,562}{2,147,483,563} & X_{j+1} = 0 \end{cases}$$

Feedback Shift Register Generators

- Tausworthe, 1965
- Define a sequence of binary digits by

$$b_i = (c_1 b_{i-1} + c_2 b_{i-2} + \cdots + c_q b_{i-q}) \bmod 2, \qquad c_i \in \{0,1\}, c_q = 1 \tag{2}$$

- in practice, only two c_j coefficients are nonzero

$$b_i = (b_{i-r} + b_{i-q}) \bmod 2, \qquad 0 < r < q \tag{3}$$

- addition modulo 2 is equivalent to xor $\oplus$
- $b_1, \dots, b_q$ must be given

Feedback Shift Register Generators II

- To form a sequence of binary integers W_1, W_2, ... we group ℓ consecutive b_i's and consider this as a number in base 2

$$\begin{aligned} W_1 &= b_1 b_2 \cdots b_\ell \\ W_i &= b_{(i-1)\ell+1} b_{(i-1)\ell+2} \cdots b_{i\ell}, \qquad i = 2, 3, \ldots \end{aligned}$$

- The recurrence is

$$W_i = W_{i-r} \oplus W_{i-q}$$

- finally

$$U_i = \frac{W_i}{2^\ell}$$

- if $(\ell, 2^q - 1) = 1$ the period is $2^q - 1$

Feedback Shift Register Generators III

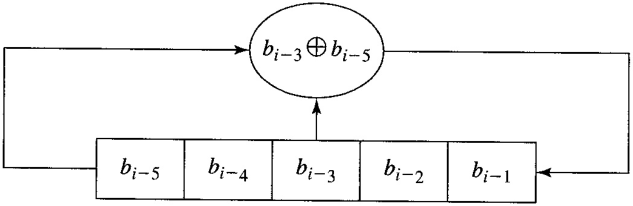


Figure: A LFSR for (3) with $r = 3$ and $q = 5$

- The name comes from a switching circuit linear feedback shift register LFSR

Feedback Shift Register Generators IV

- *generalized feedback shift register* GFSR (Lewis and Payne, 1973)
- to obtain a sequence of ℓ-bit binary integer Y_1, Y_2, ... the sequence produced by (3) is used to fill the first leftmost bit of the integers beeing formed
- That is $b_1, b_2, \ldots$ are used for the first position, $b_{1+d}, b_{2+d}, \ldots$ will fill the second position, ..., $b_{1+(\ell-1)d}$, $b_{2+(\ell-1)d}$, ... will fill the ℓ-th position
- The recurrence

$$Y_i = Y_{i-r} \oplus Y_{i-q}$$

- period $2^q - 1$ if Y_1, Y_2, ..., Y_q are linear independent with coefficients in $\mathbb{Z}_2$.

Feedback Shift Register Generators IV I

- *twisted GFSR* (TGSFR) (Matsumoto and Kurita, 1992, 1994)

$$\begin{aligned} Y_i &= Y_{i-r} \oplus AY_{i-q} \\ U_i &= TY_i \end{aligned}$$

  where Y_i's are $\ell \times 1$ vectors and A and T are $\ell \times \ell$ matrices (with binary elements)
- period $2^{q\ell} - 1$ $(q\ell = 800)$
- if $A = I_\ell$ GFSR
- *Mersenne twister* (Matsumoto, Nishimura 1998) a TGSFR with period $2^{q\ell-p} - 1$ ($q\ell - p = 19937$, the period is a Mersenne prime, i.e. a prime of the form $2^p - 1$, for some p)

$$\begin{aligned} Y_i &= Y_{i-r} \oplus A\left(Y_{i-q}^{(\ell-p)} | Y_{i-q+1}^{(p)}\right) \\ U_i &= TY_i \end{aligned}$$

Random-Numbers Streams

- The seed for a linear congruential random-number generator:
  - Is the integer value X_0 that initializes the random-number sequence.
  - Any value in the sequence can be used to "seed" the generator.
- A random-number stream:
  - Refers to a starting seed taken from the sequence $X_0, X_1, \ldots, X_P$.
  - If the streams are b values apart, then stream i could defined by starting seed:

$$S_i = X_{b(i-1)}$$

  - Older generators: $b = 10^5$; Newer generators: $b = 10^{37}$.
- A single random-number generator with k streams can act like k distinct virtual random-number generators
- To compare two or more alternative systems.
  - Advantageous to dedicate portions of the pseudo-random number sequence to the same purpose in each of the simulated systems.

Tests for Random Numbers

- Two categories:
  - Testing for uniformity:

$$H_0 : R_i \sim U[0,1]$$
$$H_1 : R_i \nsim U[0,1]$$

    Failure to reject the null hypothesis, H_0, means that evidence of non-uniformity has not been detected.
  - Testing for independence:

$$H_0 : R_i \ independent$$
$$H_1 : R_i \neg independent$$

    Failure to reject the null hypothesis, H_0, means that evidence of dependence has not been detected.
- Level of significance α, the probability of rejecting H_0 when it is true:

$$\alpha = P\left(reject\ H_0 | H_0\ is\ true\right)$$

Tests for Random Numbers II

- When to use these tests:
  - If a well-known simulation languages or random-number generators is used, it is probably unnecessary to test
  - If the generator is not explicitly known or documented, e.g., spreadsheet programs, symbolic/numerical calculators, tests should be applied to many sample numbers.
- Types of tests:
  - Theoretical tests: evaluate the choices of m, a, and c without actually generating any numbers
  - Empirical tests: applied to actual sequences of numbers produced. Our emphasis.

Frequency Tests

- Test of uniformity
- Two different methods:
  - Kolmogorov-Smirnov test
  - Chi-square test

Kolmogorov-Smirnov test I

Let X be a continuous characteristic and F its theoretical cdf. We wish to test the null hypothesis $H_0 : F = F_0$ versus one of the alternative
1. $H_a : F \neq F_0$ (two-tailed test)
2. $H_a : F > F_0$ (upper-tailed test)
3. $H_a : F < F_0$ (lower-tailed test)
Test statistics

$$D_n = \sup_{x\in\mathbb{R}}\{|\overline{F}_n(x) - F_0(x)|\}$$

$$D_n^+ = \sup_{x\in\mathbb{R}}\{\overline{F}_n(x) - F_0(x)\}$$

$$D_n^- = \sup_{x\in\mathbb{R}}\{F_0(x) - \overline{F}_n(x)\}$$

where $\overline{F}_n$ is the empirical cdf.

Kolmogorov-Smirnov test II

Kolmogorov's theorem $\Rightarrow$

$$\lim_{n\to\infty} P(\sqrt{n}D_n \leq x | H_0) = K(x) = \sum_{k=-\infty}^{+\infty} (-1)^k e^{-2k^2x^2}, \quad x > 0.$$

Also,

$$\lim_{n\to\infty} P(\sqrt{n}D_n^+ \leq x) = \lim_{n\to\infty} (\sqrt{n}D_n^- \leq x) = K^{\pm}(x) = 1 - e^{-2x^2}, \quad x > 0$$

$K^{\pm}$ is called χ-law (not χ^2) with 2 degrees of freedom. So, for $\alpha \in (0,1)$ fixed we compute the quantiles $k_{1-\alpha}$ and $k_{1-\alpha}^{\pm}$ such that

$$P(\sqrt{n}D_n \leq k_{1-\alpha}) = 1 - \alpha, \text{i.e. } K(k_{1-\alpha}) = 1 - \alpha,$$

for a two-tailed test, and

$$P(\sqrt{n}D_n^+ \leq k_{1-\alpha}^{\pm}) = 1 - \alpha \text{ and } P(\sqrt{n}D_n^- \leq k_{1-\alpha}^{\pm}) = 1 - \alpha$$

Kolmogorov-Smirnov test III

that is $K^{\pm}(k^{\pm}_{1-\alpha}) = 1 - \alpha$ for one-tailed tests.
As a conclusion, H_0 should not be rejected when

$$\sqrt{n}d_n < k_{1-\alpha} \quad \text{for} \quad H_a : F = F_0$$

$$\sqrt{n}d_n^+ < k^{\pm}_{1-\alpha} \quad \text{for} \quad H_a : F > F_0$$
$$\sqrt{n}d_n^- < k^{\pm}_{1-\alpha} \quad \text{for} \quad H_a : F < F_0$$

For a practical implementation we follow a probability based approach. It is a good practice to sort the sample values in ascending order:

$$x_1 < x_2 < \cdots < x_n.$$

In this case, for the value of test statistics one gets

$$d_n^+ = \max_{k=\overline{1,n}} \{\overline{F}_n(x_k) - F_0(x_k)\} = \max_{k=\overline{1,n}} \left\{ \frac{k}{n} - F_0(x_k) \right\}$$

Kolmogorov-Smirnov test IV

$$d_n^- = \{F_0(x_k) - \overline{F}_n(x_k - 0)\} = \max_{k=\overline{1,n}} \left\{ F_0(x_k) - \frac{k-1}{n} \right\}$$

$$d_n = \max_{k=\overline{1,n}} \{|\overline{F}_n(x_k) - F_0(x_k)|\} = \max\{d_n^+, d_n^-\}$$

For grouped data we can employ the test using class limits.
In our case $F_0(x) = x$ for $x \in [0, 1]$.

Example

- Example: Suppose 5 generated numbers are 0.44, 0.81, 0.14,0.05, 0.93.
- Sort them: 0.05, 0.14, 0.44, 0.81, 0.93
- k/N: 0.20, 0.40, 0.60, 0.80, 1.00
- $k/N - R_k$: 0.15, 0.26, 0.16, 0, 0.07
- $R_k - (k-1)/N$: 0.05, 0, 0.04, 0.21, 0.13
- $d^+ = 0.26$, $d^- = 0.21$, $d_N = \max(d^+, d^-) = 0.26$
- $K(0.26) = .326508528e-4 < 1-\alpha$, fail to reject H_0

Chi-square test I

- **Chi-square test** uses the sample statistic:

$$X_0^2 = \sum_{i=0}^{n} \frac{(O_i - E_i)^2}{E_i}$$

  E_i # of expected in the ith class, O_i # of observed in the ith class
- Approximately the chi-square distribution with $n-1$ degrees of freedom $\chi^2(n-1)$
- For the uniform distribution, E_i, the expected number in the each class is:

$$E_i = \frac{N}{n}$$

  where N is the total # of observation
- Valid only for large samples, e.g. $N \geq 50$, at least 5 observations in a class

Chi-square test II

- For large values of k we can use the approximation

$$\chi^2_{k-1,1-\alpha} \approx (k-1)\left[1 - \frac{2}{9(k-1)} + z_{1-\alpha}\sqrt{\frac{2}{9(k-1)}}\right]$$

- **Serial test** - tests if the nonoverlapping d-tuples $\mathbf{U}_1 = (U_1, U_2, \ldots, U_d)$, $\mathbf{U}_2 = (U_{d+1}, U_{d+2}, \ldots, U_{2d})$, $\ldots$ are IID uniform random vectors on $[0,1]^d$
- Divide $[0,1]$ into k subintervals of equal length and generate $\mathbf{U}_1, \ldots, \mathbf{U}_n$ (nd U_i's)
- The test statistics is

$$X^2(d) = \frac{k^d}{n}\sum_{j_1=1}^{d}\sum_{j_2=1}^{d}\cdots\sum_{j_d=1}^{d}\left(f_{j_1 j_2 \ldots j_d} - \frac{n}{k^d}\right)^2,$$

where $f_{j_1 j_2 \ldots j_d}$ is the frequency of class $I_{j_1 j_2 \ldots j_d}$

Chi-square test III

- $X^2(d)$ is assymptotic chi-square distributed with $k^d - 1$ degrees of freedom $\chi^2(k^d - 1)$

Tests for Autocorrelation

- Testing the autocorrelation between every m numbers (m is a.k.a. the lag), starting with the ith number
  - The autocorrelation ρ_{im} between numbers: R_i, R_{i+m}, R_{i+2m}, $\ldots$, $R_{i+(M+1)m}$
  - M is the largest integer such that $i + (M+1)m \leq N$
- Hypothesis:

$$\begin{aligned} H_0 &: \ \rho_{im} = 0 \\ H_1 &: \ \rho_{im} \neq 0 \end{aligned}$$

- If the values are uncorrelated:
  - For large values of M, the distribution of the estimator of ρ_{im}, denoted $\hat{\rho}_{im}$ is approximately normal.

Tests for Autocorrelation

- Test statistics is:

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

- Z_0 is distributed normally with mean $= 0$ and variance $= 1$, and:

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

$$\widehat{\sigma} = \frac{\sqrt{13M+7}}{12\,(M+1)}$$

- If $\rho_{im} > 0$, the subsequence has positive autocorrelation – High random numbers tend to be followed by high ones, and vice versa.
- If $\rho_{im} < 0$, the subsequence has negative autocorrelation – Low random numbers tend to be followed by high ones, and vice versa.

Example

- Test whether the 3rd, 8th, 13th, and so on, for the following output

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

- Hence, $\alpha = 0.05$, $i = 3$, $m = 5$, $N = 30$, and $M = 4$
- $$\begin{aligned} \widehat{\rho}_{35} &= -0.1945 \\ \widehat{\sigma} &= 0.128 \\ Z_0 &= -1.516 < z_{0.025} = 1.96 \end{aligned}$$

  fail to reject

Runs Tests for Independence I

- Consider H_0: $R_1, R_2, \ldots, R_n$ are independent.
- **Runs Tests**. Consider some examples of coin tossing:

  A) H, T, H, T, H, T, H, T, H, T,. . . (negative correlation)
  B) H, H, H, H, H, T, T, T, T, T,. . . (positive correlation)
  C) H, H, H, T, T, H, T, T, H, T,. . . ("just right")

- A *run* is a series of similar observations.
- In A above, the runs are: "H", "T", "H", "T",. . . . (many runs)
- In B, the runs are: "HHHHH", "TTTTT", ... (very few runs)
- In C: "HHH", "TT", "H", "TT",. . . . (medium number of runs)
- A runs test will reject the null hypothesis of independence if there are "too many" or "too few" runs, whatever that means.

Runs Tests for Independence II

- Runs Test "Up and Down". Consider the following sequence of uniforms.

$$.41, .68, .89, .84, .74, .91, .55, .71, .36, .30, .09\ldots$$

- If the uniform increases, put a $+$; if it decreases, put a - (like H's and T's). Get the sequence $+ +$ - - $+$ - $+$ - - - - . . .
- Here are the associated runs: $++$, $-$, $+$, -, $+$, - - -, . . .
- So do we have too many or two few runs?

Runs Tests for Independence III

- Let A denote the total number of runs "up and down" out of n observations. ($A = 6$ in the above example.)
- Amazing Fact: If n is large (at least 20) and the R_j 's are actually independent, then

$$A \sim N\left(\frac{2n-1}{3}, \frac{16n-29}{90}\right)$$

- So if $n = 100$, we would expect around 67 runs!
- We'll reject the null hypothesis if A is too big or small. The standardized test statistic is

$$Z_0 = \frac{A - E(A)}{\sqrt{V(A)}}$$

- Thus, we reject H_0 if $|Z_0| > z_{\alpha/2}$. E.g., if $\alpha = 0.05$, we reject if $|Z_0| > 1.96$

Example

- Suppose that $n = 100$. Then

$$A \sim N\left(\frac{199}{3}, \frac{1571}{90}\right)$$

  So we could expect to see $66.7 \pm 1.96\sqrt{17.5} \approx [58.5, 74.9]$ runs.

- If we see anything out of that range, we'll reject.

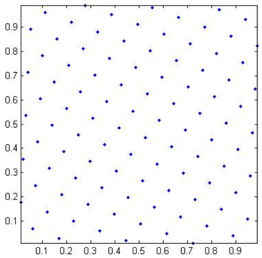
Theoretical tests I

- Theoretical tests do not require generation, they indicate how well a generator can perform by looking at its structure and defining constants - global tests (empirical tests are local).
- for a full period LCG the average of U_i's over an entire cycle is $\frac{1}{2} - \frac{1}{2m}$ and the variance is $\frac{1}{12} - \frac{1}{12m^2}$.
- Marsaglia "random numbers fall mainly in the planes" i.e. if U_1, U_2, $\dots$ is the sequence generated by a LCG, the overlaping d-tuples $(U_1, U_2, \dots, U_d)$, $(U_2, U_3, \dots, U_{d+1})$, $\dots$ fall in a relative small number of $(d-1)$-dimensional hyperplane passing through $[0,1]^d$.
- Generation 100 pairs for two full period LCGs (Figure 3) and 2000 triples and plotting of lattice structure (Figure 2).

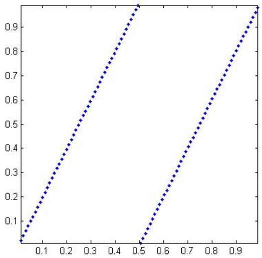
Theoretical tests II

The following geometric quantities are of interest.

- Minimum number of hyperplanes (in all directions). Find the multiplier that maximizes this number.
- Maximum distance between parallel hyperplanes. Find the multiplier that minimizes this number.
- Minimum Euclidean distance between adjacent k-tuples. Find the multiplier that maximizes this number



2D lattice structure for LCG with $m = 101$ and $a = 18$



2D lattice structure for LCG with $m = 101$ and $a = 18$

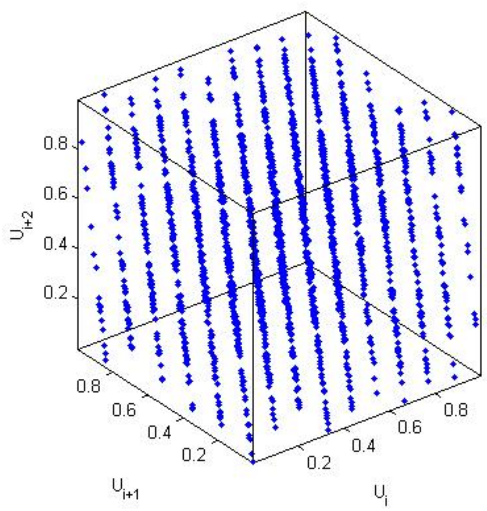


Figure: 3D lattice structure for LCG RANDU with $m = 2^{31}$ and $a = 65539$

Shortcomings

- The test is not very sensitive for small values of M, particularly when the numbers being tests are on the low side.
- Problem when “fishing” for autocorrelation by performing numerous tests:
  - If $\alpha = 0.05$, there is a probability of 0.05 of rejecting a true hypothesis.
  - If 10 independent sequences are examined,
    - The probability of finding no significant autocorrelation, by chance alone, is $0.95^{10} = 0.60$.
    - Hence, the probability of detecting significant autocorrelation when it does not exist $= 40\%$

References

- Averill M. Law, *Simulation Modeling and Analysis*, McGraw-Hill, 2007
- J. Banks, J. S. Carson II, B. L. Nelson, D. M. Nicol, *Discrete-Event System Simulation*, Prentice Hall, 2005
- J. Banks (ed), *Handbook of Simulation*, Wiley, 1998, Chapter 5
- G. S. Fishman, *Monte Carlo. Concepts, Algorithms and Applications*, Springer, 1996
- P. L'Écuyer, Random Number Generation, in Handbook of Computational Statistics, Gentle, Haerdle, Morita (eds.), Springer, 2004