Introduction of Pseudo-Random Number Generator

True Random Number and Pseudo-Random Number

- True Random Number Sequence
  - Not predictable
    - Cannot predict the next number of the sequence based on the current numbers
  - Difficult to be generated using software
    - Software has only deterministic operations
  - Can be generated using hardware
    - Based on microscopic phenomena such as thermal noise
- Pseudo-Random Number Sequence
  - Sequence of number determined by a small set of initial values
  - The number follows a certain distribution (usually uniform)
  - Predictable
    - Next number of the sequence is determined by the current state
  - Can be generated using software

Pseudo-Random Number Generation Algorithm

- Middle Square Method
  - Start from an n digit number
  - Calculate square of an n digit number, resulting a 2n digit number
  - Use the middle n digit of the 2n digit number as current number
  - Use the current n digit random number to generate next number
  - Example:

1^{st} 1111

$2^{nd} 1111^2$ = 01234321 -> 2343

3^{rd} 2343^2 = 05489649 -> 4896

…

Pseudo-Random Number Generation Algorithm Cont'd

- Better algorithm

  Select unsign number: *IA*, *IM*,*IC*

  start with a current state: *current_state*

  *next_state = cur * IA + IC*

  t1 = next _state & (IM-1)

  output = t1 / IM

  current_state = next_state

  Note: & is bit-wise and operation

Pseudo-Random Number Generation Algorithm Cont'd

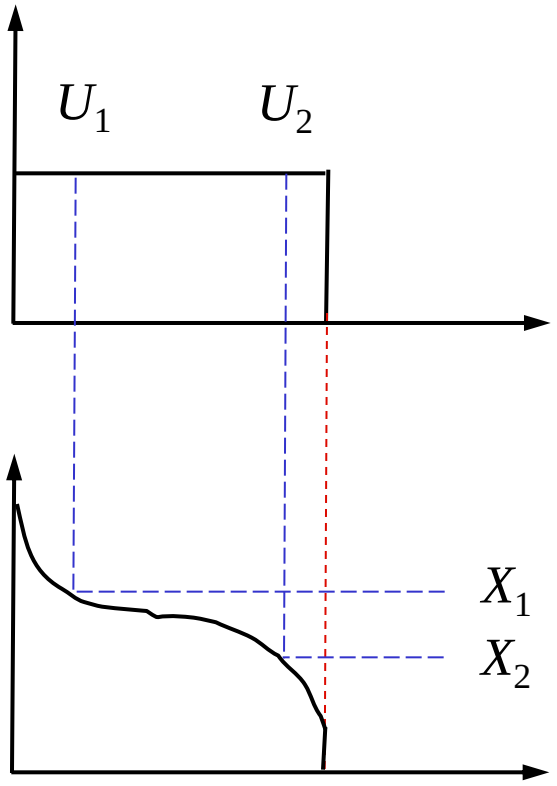
- Other algorithms
  - Yarrow algorithm
  - Mersenne twister
    - Best psudo-number generation algorithm
    - Applied in Matlab 'rand()' function
    - http://en.wikipedia.org/wiki/Mersenne_twister

Problem of Pseudo Random Sequence

- Problem: Always produce the same sequence thereafter when initialized with the initial state
  - Solve: Use true-random number as starting state.
  - Example: Use time as random seed
- Problem: Always repeat after a certain length
  - Solve: Make the repeat period long enough to prevent repeat of sequence.
  - Example: Mersenne twister achieves period 2^{19937}.

Generate Samples of Arbitrary Distribution

- Given CDF_X of random variable X with any arbitrary distribution, generate samples of X
- Method
  - Generate uniform pseudo random samples $(U_1, U_2, \ldots U_N) \in (0,1)$
  - Obtain samples of X by $X_i = CDF_X^{-1}(U_i)$
    - **Proof** $CDF(X_i) = P\{X_i < CDF_X^{-1}(U_i)\} = P\{U_i < CDF(X)\} = CDF\{X\}$
- Matlab functions
  - 'randn()', 'lognrnd()', 'random()'



Generate Correlated Random Samples

- Given joint Gaussian random vector $\boldsymbol{X}=(X_1, X_2, \ldots X_n)^{\mathrm{T}}$ with mean vector $\boldsymbol{M}=\mathrm{E}[\boldsymbol{X}]$ and covariance matrix $\boldsymbol{C}=\mathrm{E}[\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}]$
  - Generate samples for $\boldsymbol{X}$
    - Note: covariance matrix $\boldsymbol{C}$ is positive semi-definite
- Method
  - Perform eigenvalue decomposition of covariance matrix $\boldsymbol{C}=\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}}$
  - Generate samples of independent standard Gaussian random vector $\boldsymbol{Y}=(\mathrm{Y}_1, Y_2, \ldots Y_n)^{\mathrm{T}}$
  - $\boldsymbol{X}=\boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{Y}$ are the samples of correlated Gaussian random vector
- Matlab functions
  - ‘normrnd()’, ‘lognrnd()’
- Correlated non-Gaussian samples
  - Generated correlated non-Gaussian samples is very difficult
  - No efficient way to achieve

Quasi-Random Sequence

- Quasi-random sequence
  - Low discrepancy array
  - Converge faster than pure random sequence in low dimensional cases
  - Not work for very high dimensional case

- Algorithms
  - Sobal
  - Halton