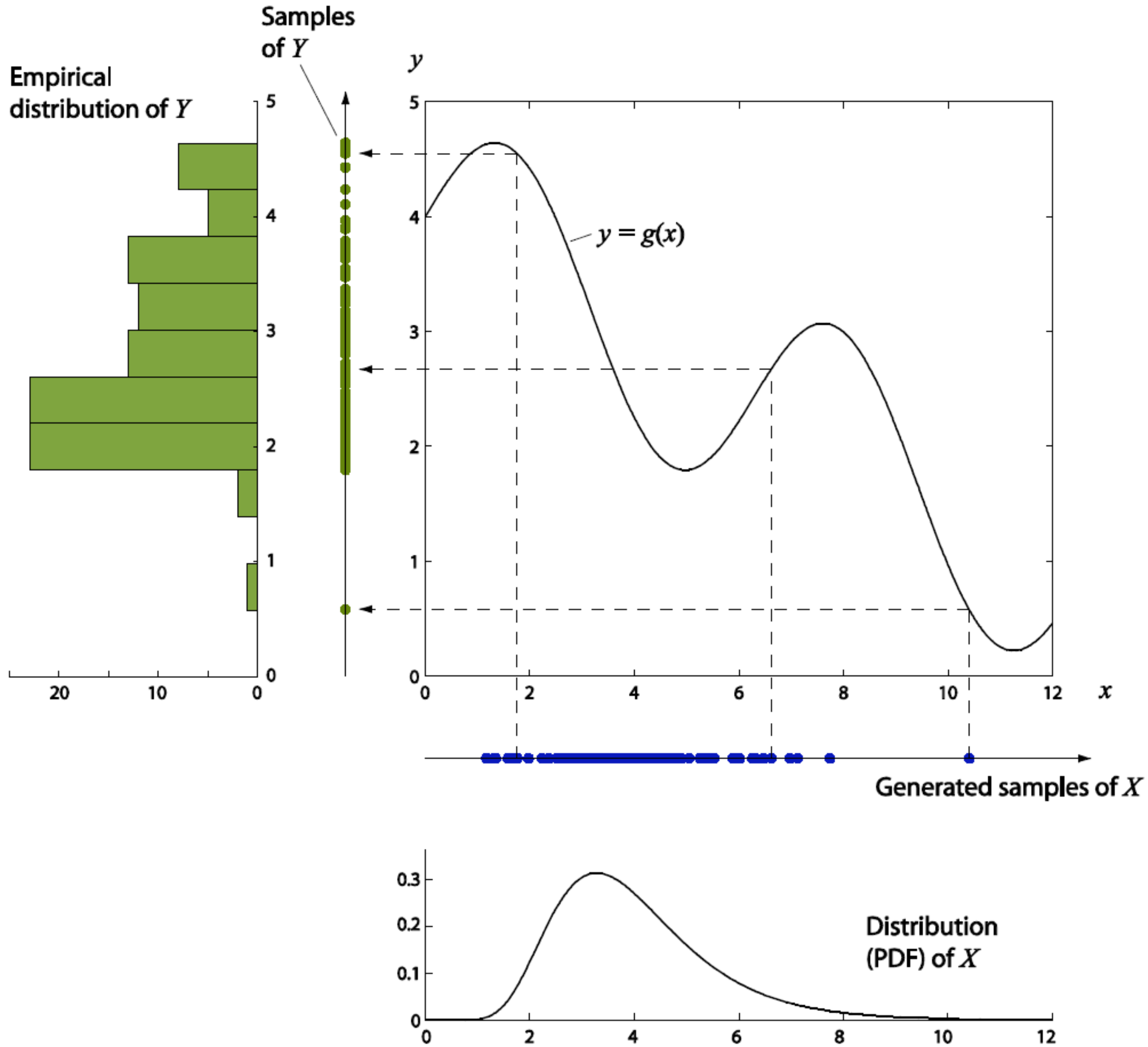
Monte Carlo simulation

Steps

- Generation of random samples $\mathbf{x}^{(i)}$, $i = 1,\ldots,n_s$ of the input variables $\mathbf{X}$
- Evaluation of the function at the samples $\mathbf{y}^{(i)} = g(\mathbf{x}^{(i)})$
- Analysis of the generated samples $\mathbf{y}^{(i)}$ of $\mathbf{Y}$

Monte Carlo simulation



Generating (pseudo-)random samples

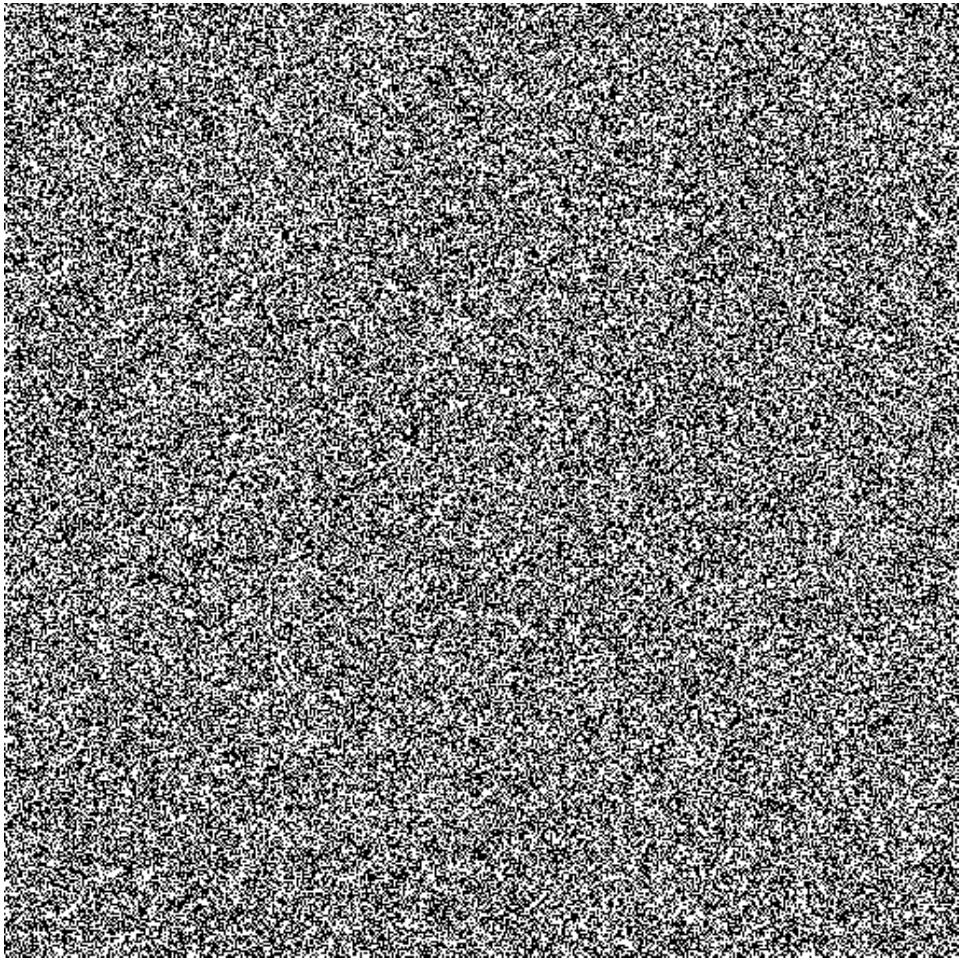
Pseudo-random number generators produce samples from the uniform distribution in $[0, 1]$

```
rand(m,n,…)
```

They are deterministic sequences that must initiated by a seed value

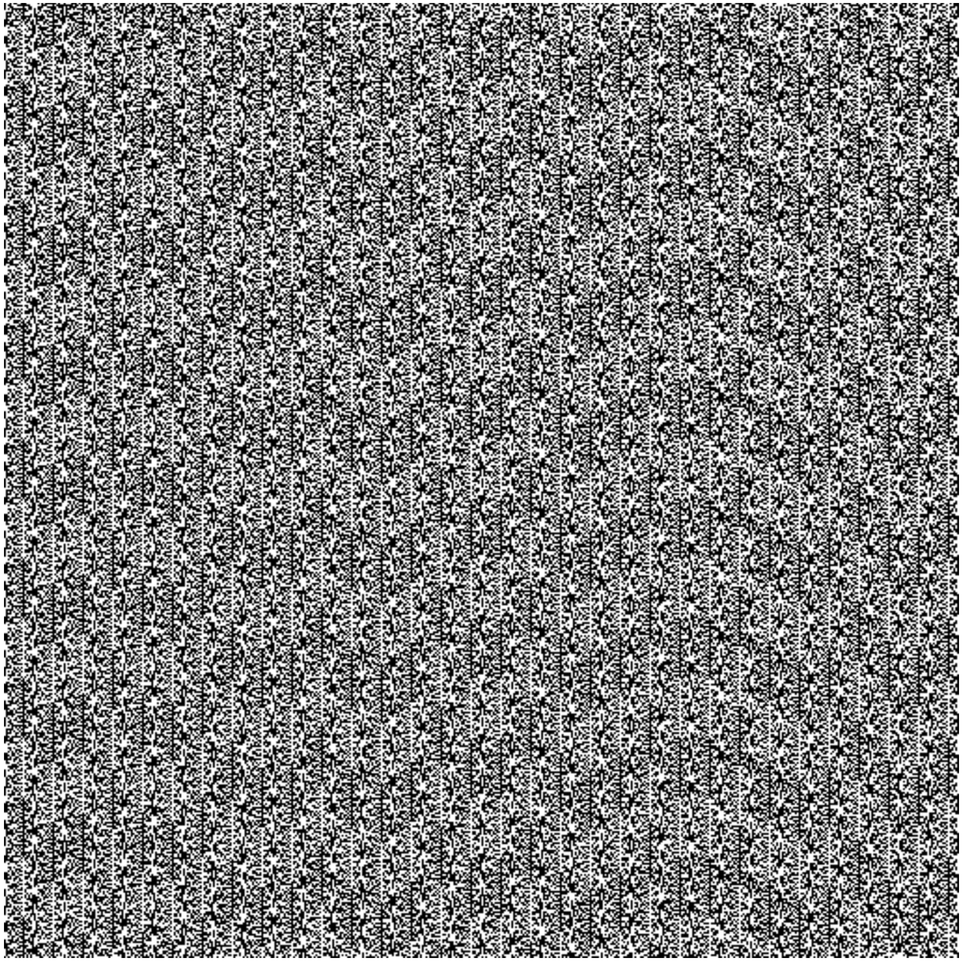
```
rng(seed,'twister')
```

True random numbers vs. pseudo-random numbers



generated by random.org

True random numbers vs. pseudo-random numbers



generated by PHP/Windows pseudo-random number generator

(from www.boallen.com)

Generating samples from an arbitrary distribution

Generation of samples from an arbitrary random variable with CDF $F_X(x)$

- Generate a sample $v^{(i)}$ uniformly distributed in $[0, 1]$
- Require that the samples $v^{(i)}$ and $x^{(i)}$ have the same CDF value

$$F_V(v^{(i)}) = F_X(x^{(i)})$$

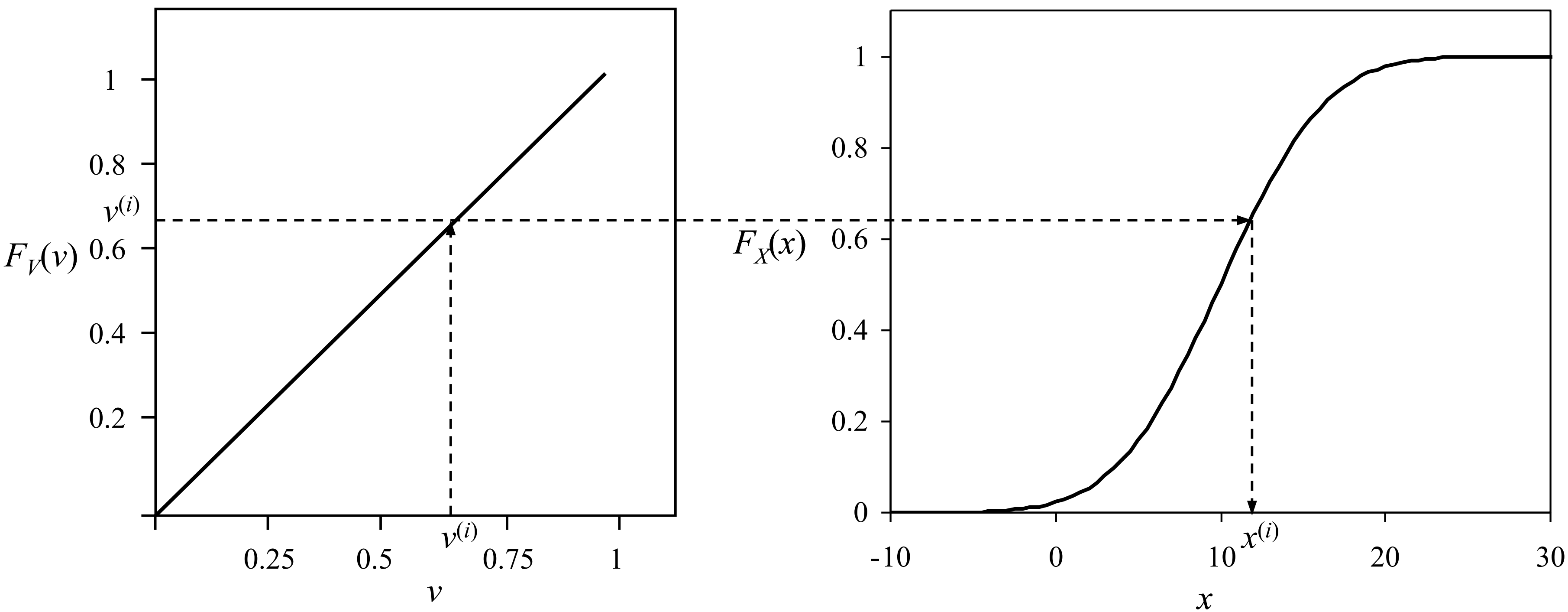
$$v^{(i)} = F_X(x^{(i)}) \Leftrightarrow x^{(i)} = F_X^{-1}(v^{(i)})$$

Assuming a strictly increasing CDF $F_X(x)$

```
namernd(par1,par2,m,n,…)
```

Generating samples from an arbitrary distribution

Sampling from a normal distribution



Statistical analysis of generated samples

- Analysis of the samples $\mathbf{y}^{(i)}$
  - Compute statistics (sample mean, sample standard deviation,…)
  - Plot graphical summaries (histograms, cumulative frequency diagrams, …)
  - e.g. Expected value of a function

$$
\begin{aligned}
\mathrm{E}[g(\mathbf{X})] &= \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} g(\mathbf{x}) f(\mathbf{x})\, dx_1\, dx_2 \dots dx_n \\
&\approx \frac{1}{n_s} \sum_{i=1}^{n_s} g(\mathbf{x}^{(i)}) \\
&= \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{y}^{(i)}
\end{aligned}
$$

Replace multi-fold integral with single summation!

Statistical analysis of generated samples

- Analysis of the samples $\mathbf{y}^{(i)}$
  - e.g. Estimation of densities

Kernel density estimation

$$\hat{f}_Y(y) = \frac{1}{n_s h} \sum_{i=1}^{n_s} K\left(\frac{y - y^{(i)}}{h} \right)$$

$K(y)$: Kernel

h : Bandwidth parameter

Accuracy of Monte Carlo simulation

Estimate of the expected value $\mathrm{E}[Y]$

- The MCS estimate is equal to the sample mean

$$\bar{Y} = \frac{1}{n_s}\sum_{i=1}^{n_s} y^{(i)}$$

- Each sample y_i is a random variable with mean μ_Y and standard deviation σ_Y
- Therefore, the sample mean is also a random variable

$$\mathrm{E}[\bar{Y}] = \frac{1}{n_s}\sum_{i=1}^{n_s}\mathrm{E}[Y_i] = \frac{1}{n_s}\sum_{i=1}^{n_s}\mu_Y = \mu_Y \qquad \text{[Unbiased estimator]}$$

$$\mathrm{Var}[\bar{Y}] = \frac{1}{n_s^{\,2}}\sum_{i=1}^{n_s}\mathrm{Var}[Y_i] = \frac{\sigma_Y^{\,2}}{n_s}$$

Accuracy of Monte Carlo simulation

Estimate of the expected value $\mathrm{E}[Y]$

- Standard deviation of the estimate

$$\sigma_{\mu_Y,MCS} = \frac{\sigma_Y}{\sqrt{n_s}}$$

- Coefficient of variation of the estimate

$$\delta_{\mu_Y,MCS} = \frac{\sigma_Y}{\mu_Y \sqrt{n_s}} = \frac{\delta_Y}{\sqrt{n_s}}$$

Latin Hypercube Sampling

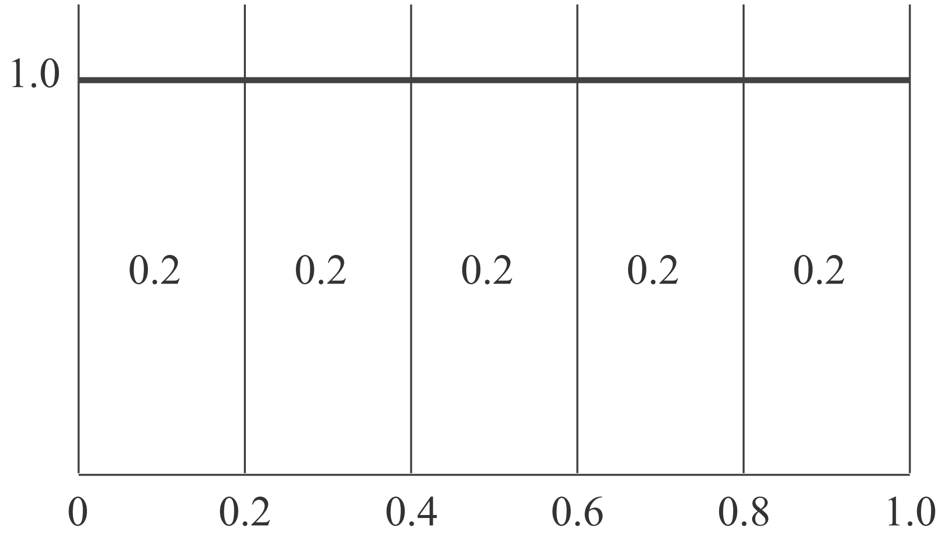
- Divide the range of each variable X_i into n_s non-overlapping intervals with equal probability $1/n_s$
- Generate one sample from the PDF of each interval
- Permute (shuffle) the samples of each variable X_i randomly

Note: Efficiency decreases with increase of dimension because the shuffling becomes the dominant source of randomness

Latin Hypercube Sampling

Sampling from a two dimensional uniform distribution

- Divide the range of each variable and sample each interval



Latin Hypercube Sampling

Sampling from a two dimensional uniform distribution

- Shuffle the samples

  Permute randomly the sequence $(1,2,3,4,5)$ for each component

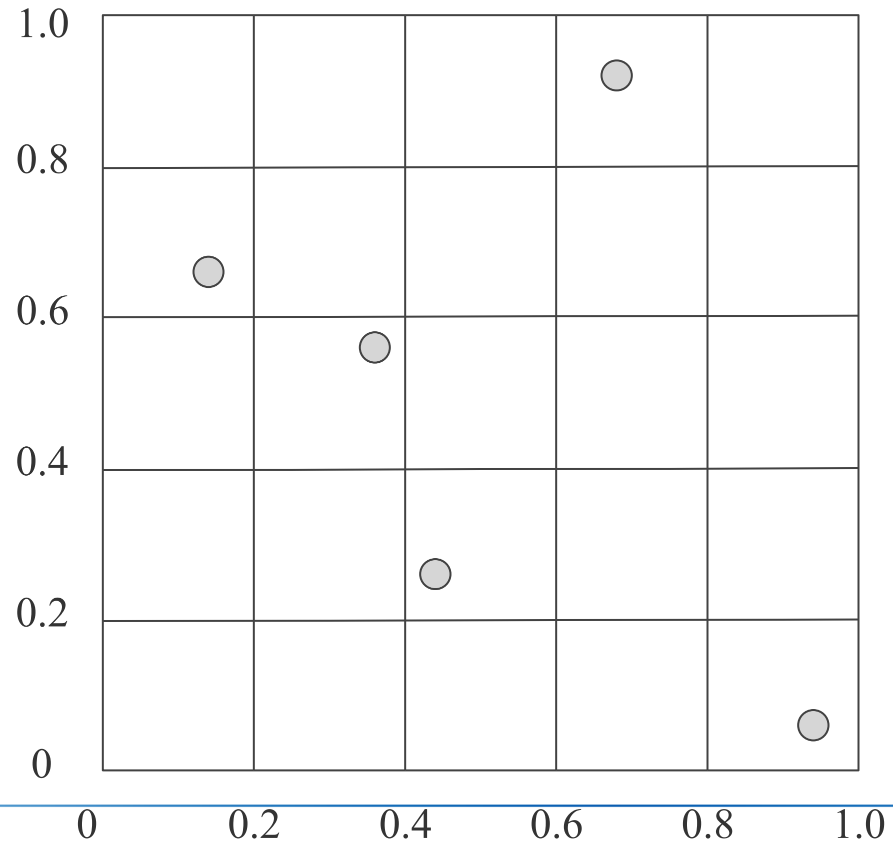
  1st component: $(3,5,2,1,4)$

  2nd component: $(2,1,3,4,5)$

Latin Hypercube Sampling

Sampling from a two dimensional uniform distribution

- Shuffle the samples



Latin Hypercube Sampling

Sampling from a d dimensional uniform distribution

- Generate a matrix $\bar{\mathbf{V}}$ of $[n_s \times d]$ samples from the standard uniform distribution
- Generate a matrix $\mathbf{P}$ with dimensions $[n_s \times d]$ such that each column is a random permutation of $1,\ldots,n_s$
- Apply:

$$\tilde{\mathbf{V}} = \frac{1}{n_s}\left(\mathbf{P} - \bar{\mathbf{V}}\right)$$

Latin Hypercube Sampling

Sampling from a d dimensional uniform distribution

```
lhsdesign(n,d);
```

Quasi-random sampling

- Low discrepancy sequences – minimize discrepancy between sequence and uniform samples, e.g. Halton, Sobol, Niederreiter sequences
- QRS does not attempt to simulate randomness but aims at filling the space uniformly
- Randomness can be approximated by
  - Scramble
  - Skip
  - Leap

Note: Efficiency in space filling decreases with increase of dimension

Quasi-random sampling

Example

Initial sequence: 1 2 3 4 5 6 7 8 9 10

- Scramble

  1 4 5 3 8 9 7 6 10 2
- Skip

  1 4 5 3 8 9 7 6 10 2
- Leap

  1 4 5 3 8 9 7 6 10 2

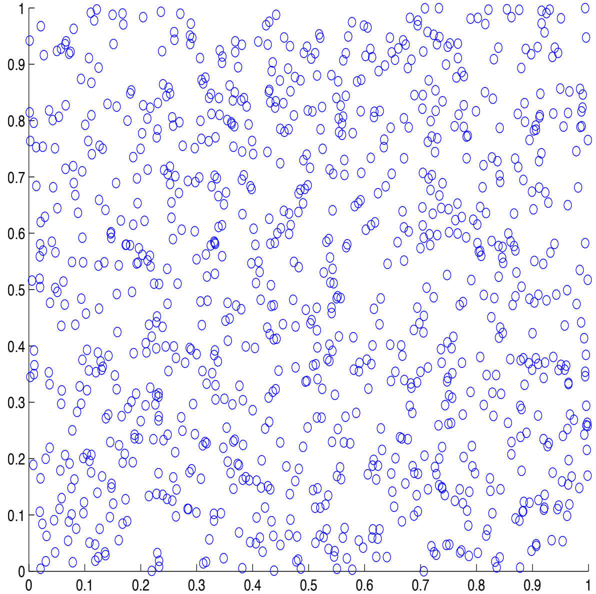
Quasi-random sampling

Sampling from a uniform distribution

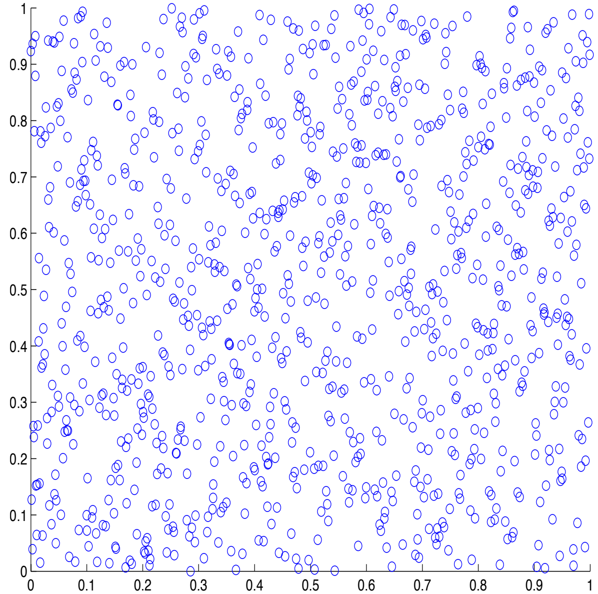
```
p = haltonset(d,'Skip',1e3,'Leap',1e2);

p = scramble(p,’RR2’);

p(1:n,:);
```

Comparison of sampling methods

Pseudo random sampling



Latin hypercube sampling



Quasi random sampling