RANDOM NUMBERS

Pseudorandom Numbers

- Computers use *pseudorandom numbers* produced by (very long period) integer sequences which appear to be random.
- Early (1950s) pseudorandom number sequences were generated from linear congruential generators (LCGs) of the form $x_n = ax_{n-1} (\text{mod } m)$;
  e.g. $(m, a, x_0) = (31, 3, 1)$, $(m, a) = (2^{13} - 1, 17) = (8191, 17)$;
  $(m, a) = (2^{31} - 1, 742938295) = (2147483647, 742938295)$;
  goal is to choose good a and m; uniform $[0, 1)$ (pseudo)random numbers are x_n/m.

```
>> x(1)=1; for i = 1:30, x(i+1) = mod(x(i)*3,31); end, disp(x)
 1     3     9    27    19    26    16    17    20    29    25
13     8    24    10    30    28    22     4    12     5    15
14    11     2     6    18    23     7    21     1
>> disp(x/31)
 0.032258 0.096774 0.29032 0.87097 0.6129  0.83871 0.51613 0.54839
  0.64516 0.93548  0.80645 0.41935 0.25806 0.77419 0.32258 0.96774
  0.90323 0.70968  0.12903 0.3871  0.16129 0.48387 0.45161 0.35484
  0.064516 0.19355 0.58065 0.74194 0.22581 0.67742 0.032258
```

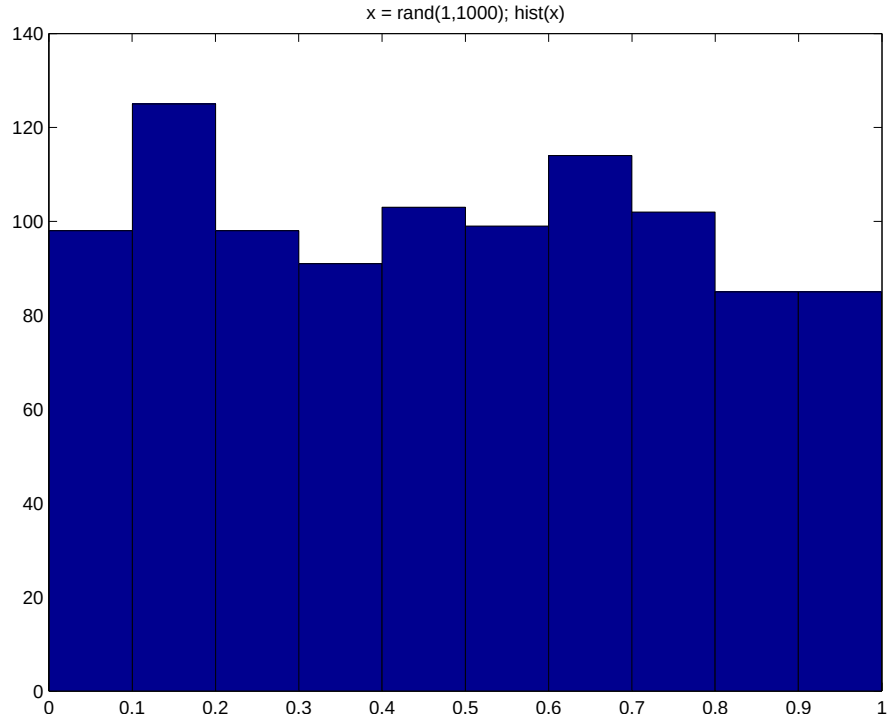
- Modern pseudorandom number generators (PNGs) use complicated generalizations of LCGs, with very long periods, and they pass sophisticated statistical tests for randomness; e.g. Matlab function *rand* has default period $(2^{19937} - 1)/2$.

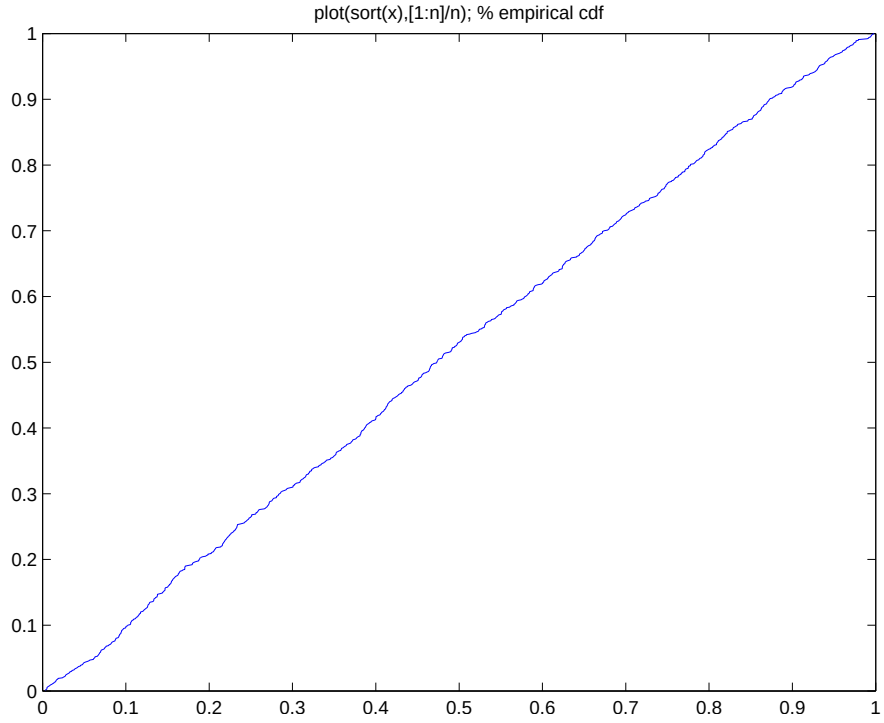
```
RAND   Uniformly distributed pseudo-random numbers.
   R = RAND(N) returns an N-by-N matrix containing pseudo-random values
    drawn from a uniform distribution on the unit interval.
   RAND(M,N) or RAND([M,N]) returns an M-by-N matrix. RAND with no arguments returns a scalar.

   You can use any one of three generator algorithms, as follows:
   RAND(METHOD,S) causes RAND to use the generator determined by METHOD, and initializes that
    generator. S is a scalar integer value from 0 to 2^32-1, or the output of RAND(METHOD).
   METHOD is one of the following strings:
    'twister' - Use the Mersenne Twister algorithm by Nishimura and Matsumoto, the default in
                MATLAB Versions 7.4 and later. This method generates double precision values in
                the closed interval [2^(-53), 1-2^(-53)], with a period of (2^19937-1)/2.
    'state'   - Use modified Marsaglia's Subtract-with-Borrow algorithm, the default in MATLAB
                Versions 5 through 7.3. This method can generate all the double precision
                values in the interval [2^(-53), 1-2^(-53)], and, theoretically, can generate
                over 2^1492 values before repeating itself.
    'seed'    - Use a multiplicative congruential algorithm, the default in MATLAB Version 4.
                This method generates values in [1/(2^31-1), 1-1/(2^31-1)], with a period 2^31-2.
   The sequence of numbers produced by RAND is determined by the internal state of the generator.
   Setting the generator to the same fixed state allows computations to be repeated.
   Setting the generator to different states leads to unique computations. Since MATLAB resets
    the state at start-up, RAND will generate the same sequence of numbers in each
    session unless the state is changed.
   Examples:
       Return RAND to its default initial state:          rand('twister',5489)
       Initialize RAND to a different state each time:    rand('twister',sum(100*clock))
       Generate uniform values from the interval [a, b]: r = a + (b-a).*rand(100,1);
       Generate integers uniform on the set 1:n:         r = ceil(n.*rand(100,1));
   For a full description of the Mersenne Twister algorithm, see
   http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html.
```

Matlab experiments to produce histogram and empirical cdf for rand.

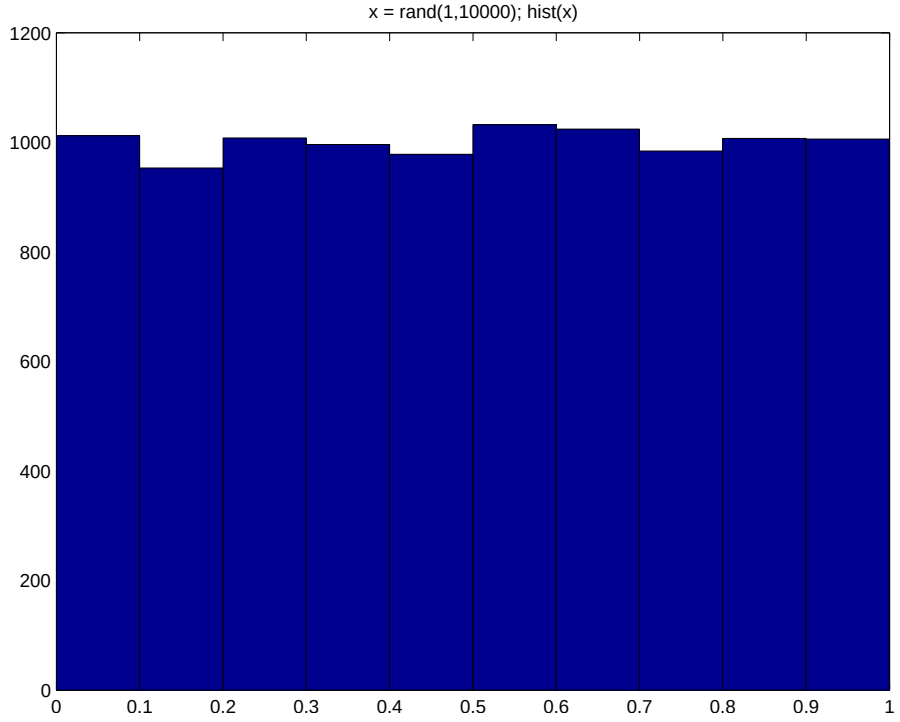
```
n = 1000; x = rand(1,n); disp([mean(x),var(x)])
      0.51002     0.081386
n = 1000; x = rand(1,n); disp([mean(x),var(x)])
      0.49821     0.083091
hist(x), plot(sort(x),[1:n]/n); % hist and empirical cdf
```

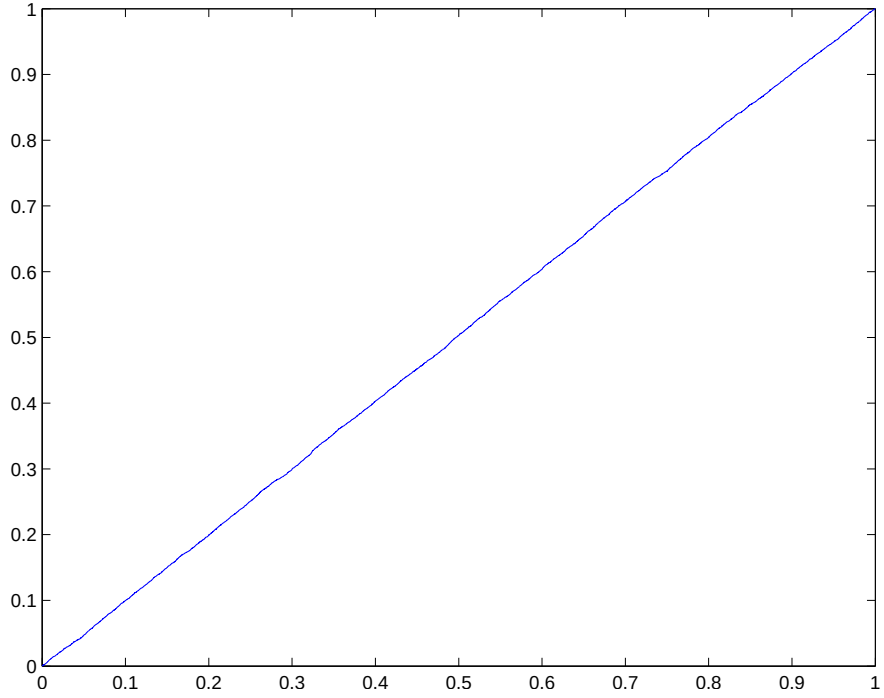




Matlab experiments to produce histogram and empirical cdf for rand.

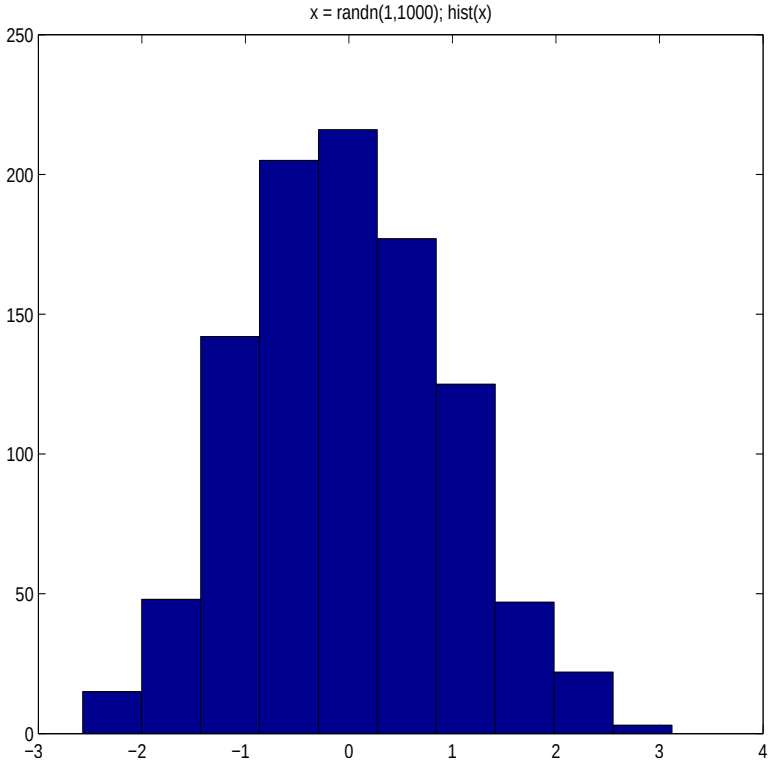
```
n = 10000; x = rand(1,n); disp([mean(x),var(x)])
      0.49963     0.083208
hist(x), plot(sort(x),[1:n]/n); % empirical cdf
```

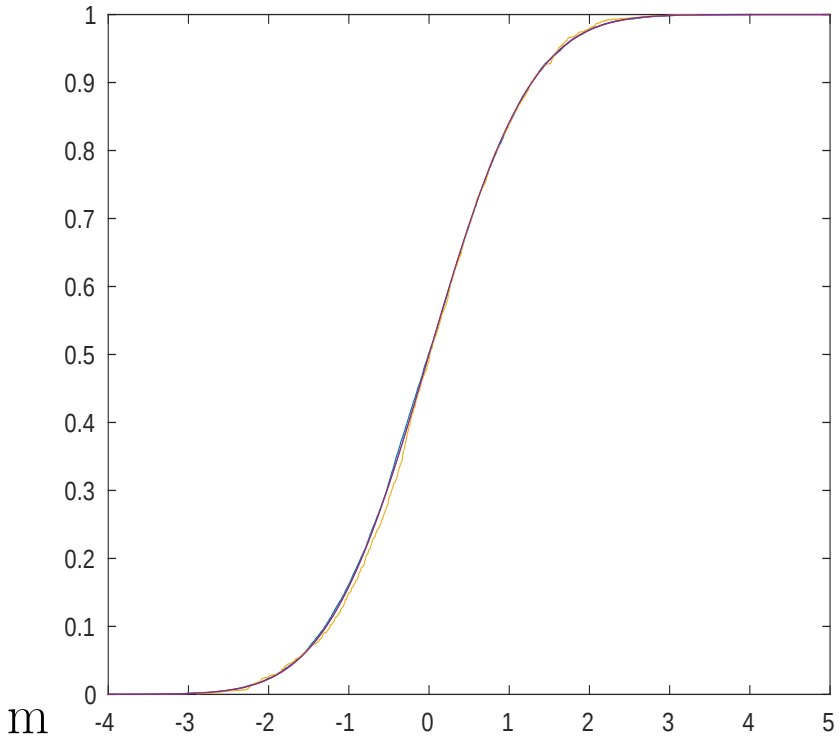




Matlab experiments to produce histogram and empirical cdf for randn.

```
n = 1000; x = randn(1,n); disp([mean(x),var(x)])
   -0.0065036       0.90672
n = 1000; x = randn(1,n); disp([mean(x),var(x)])
    -0.02224       0.97058
hist(x), plot(sort(x),[1:n]/n); % empirical cdf
hold on, z = [-4:.01:5]; plot(z,normcdf(z)) % actual cdf
```

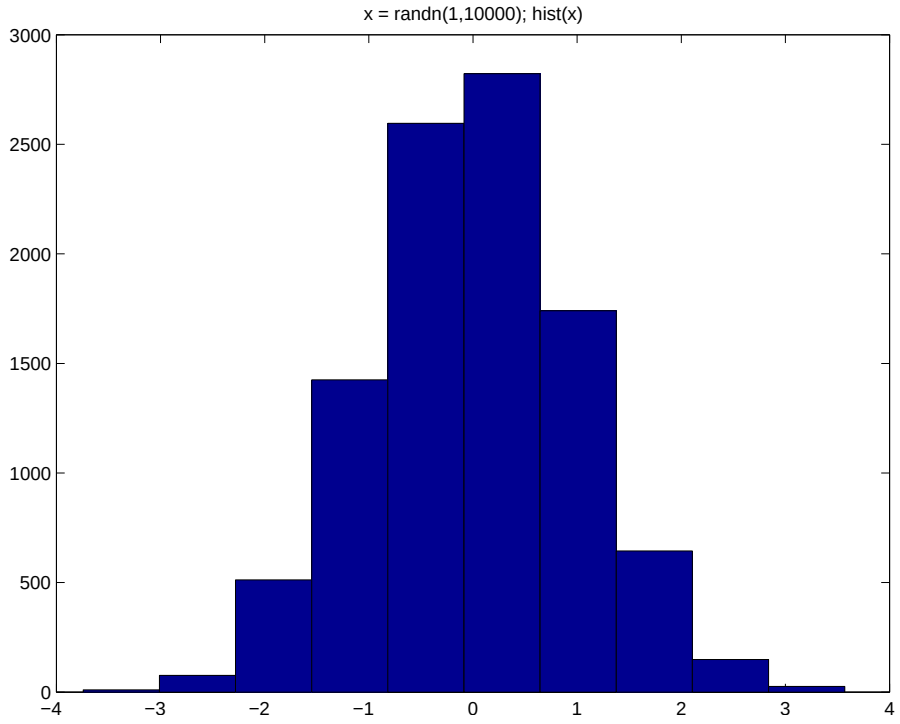


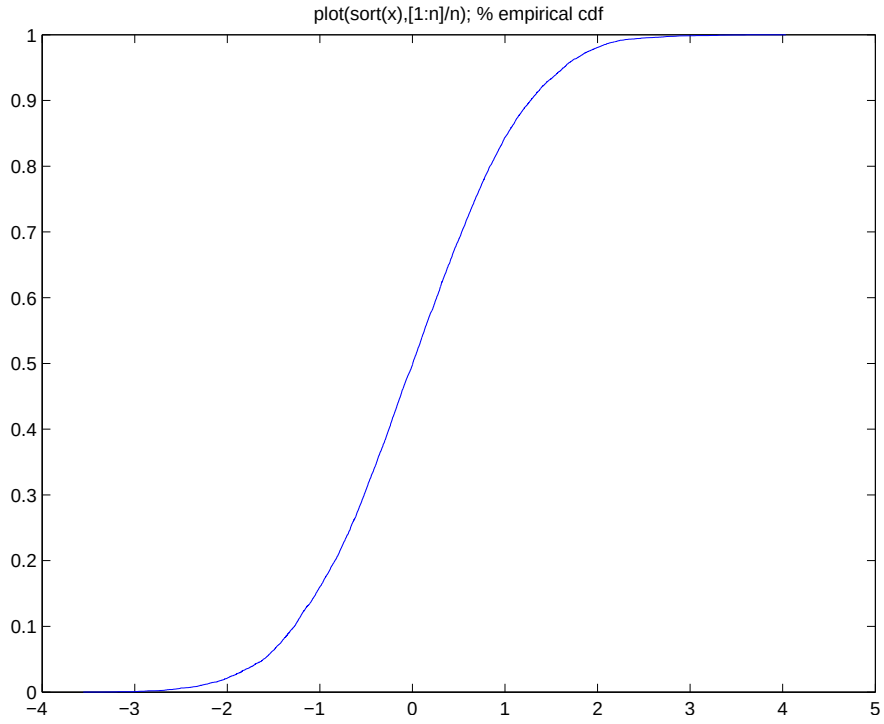


m

Matlab experiments to produce histogram and empirical cdf for randn.

```
n = 10000; x = randn(1,n); disp([mean(x),var(x)])
    -0.004009        1.0133
hist(x), plot(sort(x),[1:n]/n); % empirical cdf
```





Monte Carlo (MC) Method for Integrals: a simple application of PNGs.

- One-dim. integrals: let $\theta = \int_0^1 g(x)dx$, so $\theta = E[g(U)]$,
  with $U \in (0,1)$ uniformly ($U \sim uni(0,1)$).
  An MC Algorithm for θ:
  a) for $i = 1, 2, \ldots, K$, choose $U_i \sim uni(0,1)$;
  b) let $S_K = \frac{1}{K}\sum_{i=1}^{K} g(U_i)$.
  Laws of large numbers imply that $\lim_{K\to\infty} S_K = \theta$.
  Central Limit Theorem implies $P\{|S_K - \theta| < \frac{2\sigma}{\sqrt{K}}\} \approx .95$.
  Note: $\sigma^2 = \int_0^1 (g(x)-\theta)^2 dx \approx \hat{\sigma}_K^2 = \frac{1}{K-1}\sum_{i=1}^{K}(S_K - g(U_i))^2$.
  Matlab: with $g(x)$ defined as a vector valued function,

```
U = rand(1,K); disp([mean(g(U)) 2*std(g(U))/sqrt(K)])
```

  produces estimate for θ and error (with 95% confidence).
  Example: Matlab MC Method for $\int_0^1 e^{-x^2/2}dx$:

```
for K = 10.^[2 4 6], X = rand(1,K);
  G = exp(-X.*X/2); disp([K mean(G) 2*std(G)/sqrt(K)])
end
```

  Output

```
         100      0.83487     0.024718
       10000      0.85338    0.0024103
     1000000      0.85558   0.00024276
```

  Note: exact $\theta = \sqrt{2\pi}(\Phi(1) - \Phi(0)) \approx 0.8556243918921 49$.

RANDOM NUMBERS CONTINUED

MC Method for Integrals Continued

- Some other integrals:
  a) for other finite intervals (a, b) use

$$\theta = \int_a^b f(y)dy = (b-a)\int_0^1 f(a+(b-a)x)dx,$$

so $\theta \approx \frac{b-a}{K}\sum_{i=1}^{K} f(a+(b-a)U_i)$.

Example: Matlab MC Method for $\int_1^4 \sin(y)dy \approx 1.193946$

```
for K = 10.^[2 4 6], Y = 1 + 3*rand(1,K);
  F = 3*sin(Y); disp([K mean(F) 2*std(F)/sqrt(K)])
end
```

Output

```
         100      0.99835       0.36258
       10000       1.1997      0.034553
     1000000       1.1927     0.0034727
```

Compare exact answer: $\cos(1) - \cos(4) \approx 1.19394592673175$.

RANDOM NUMBERS CONTINUED

b) for unbounded (a, b) use a transformation to $(0, 1)$, so

$$\theta = \int_0^\infty f(y)dy = \int_0^1 f(t(x))t'(x)dx,$$

with some $t(x) \mid [0, 1) \mapsto [0, \infty)$; e.g. $y(x) = (1 - x)/x, \quad y'(x) = -1/x^2$, so

$$\theta = \int_0^1 \frac{f((1/x - 1)}{x^2}dx \approx \frac{1}{K}\sum_{i=1}^{K} \frac{f(1/U_i - 1)}{U_i^2}.$$

Example: Matlab MC Method for $\int_0^\infty y(1 + y^2)^{-2}dy = 1/2$.

```
for K = 10.^[2 4 6], U = rand(1,K);
  Y = 1./U - 1; F = Y./(U.*(1+Y.^2)).^2;
  disp([K mean(F) 2*std(F)/sqrt(K)])
end
```

Output:

```
       100        0.53378462    0.065752947
     10000        0.49976694    0.006761543
   1000000        0.50008421    0.000672028
```

RANDOM NUMBERS CONTINUED

c) for **multiple integrals**, with $\mathbf{x} = (x_1, x_2, \ldots, x_n)$; if $U_{ij} \sim uni(0,1)$,

$$\theta = \int_0^1 \int_0^1 \cdots \int_0^1 g(\mathbf{x}) dx_1 dx_2 \cdots dx_n \approx \frac{1}{K} \sum_{j=1}^{K} g(\mathbf{U}_j).$$

Example: estimating π; if $(X, Y) \sim uni((-1,1) \times (-1,1))$,
$P\{X^2 + Y^2 \le 1\} = \frac{\pi}{4} = \theta$, the ratio of circle to square areas;

$$\theta = \frac{1}{4} \int_{-1}^{1} \int_{-1}^{1} I(x^2 + y^2 \le 1) dx dy,$$

where *indicator function* $I(v) = \begin{cases} 1 & \text{for true } v \\ 0 & \text{otherwise} \end{cases}$.

If $U_{ij} \sim uni(0,1)$, $\theta \approx \frac{1}{K} \sum_{j=1}^{K} I\big((2U_{1j} - 1)^2 + (2U_{2j} - 1)^2 \le 1\big)$; Matlab code

```
for K = 10.^[2 4 6], Y = 2*rand(2,K) - 1;
  F = sum(Y.*Y) < 1; disp([K mean(F) 2*std(F)/sqrt(K)])
end
```

Output:

```
     100        0.76       0.085847
   10000      0.7857      0.0082071
 1000000      0.7855     0.00082095
```

Note: $\theta \approx 0.7853981633974483$, $\sigma = \sqrt{\theta(1-\theta)} \approx .4105$; θ is a Bernoulli RV.