CEE 618 Scientific Parallel Computing (Lecture 13): Random Number Generation

Albert S. Kim

Department of Civil and Environmental Engineering
University of Hawai'i at Manoa
2540 Dole Street, Holmes 383, Honolulu, Hawaii 96822

Table of Contents

Outline

Uniform deviates

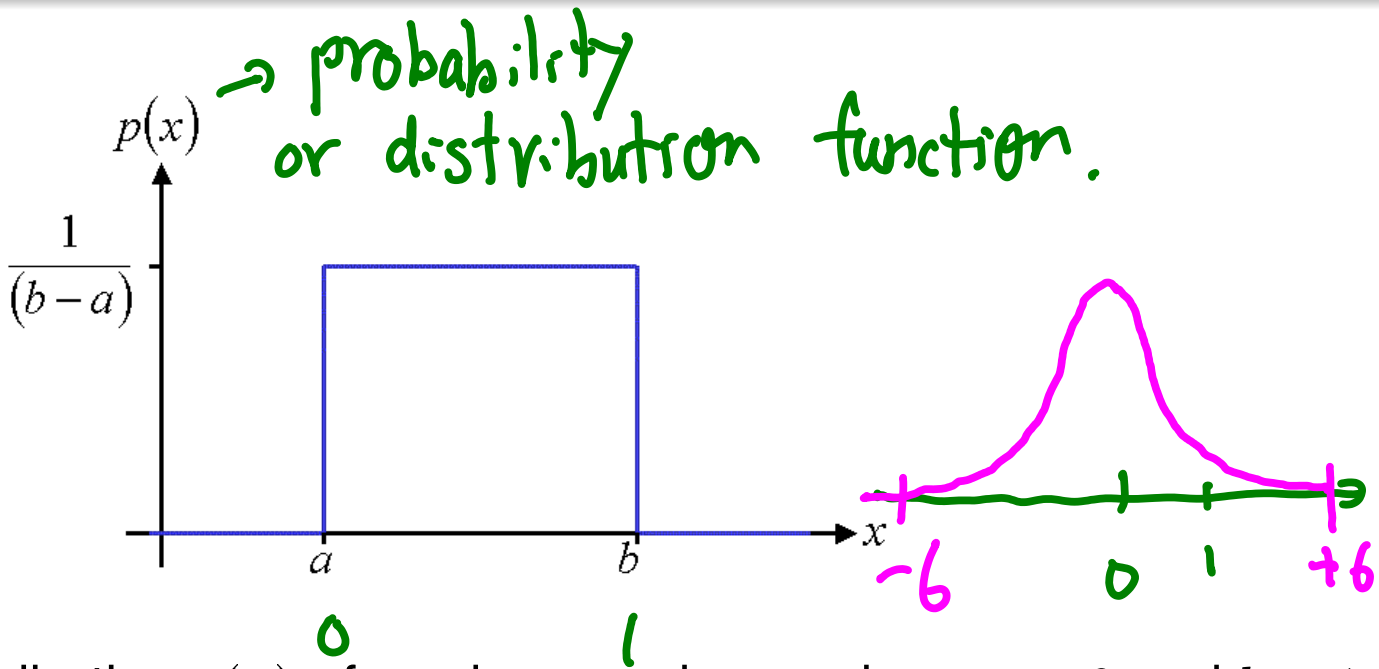


Figure: Uniform distribution $p(x)$ of random number x where $a = 0$ and $b = 1$.

1. Random numbers that lie within a specific rage (typically **0 to 1**), with any one number in the range just as likely as any other.
2. Other types of random numbers follow specific distribution such as **normal (Gaussian)** random numbers.
3. Source: Chapter 7, section 1, "Numerical Recipes in FORTRAN (NRF)". (Postscript files are freely downloadable at `http://www.haoli.org/nr/bookf.html`).

Basic algorithm

$$I_{j+1} = \text{mod}\,(a I_j + c, m)$$

- I_j is an integer, which ranges from **0 to** $m - 1$.
- m is a **modulus**, (usually) the largest number that a machine can provide. For example, $m = 2^{32}$ for old **32 bit** computers.
- a and c are arbitrary positive integers, called the **multiplier** and **increment**, respectively.
- IBM mainframe computers' infamous random number generator used $a = 65539$.
- For our study, we can use $c = 618$ (course number) or any integer of your own choice.
- The initial integer I_0 is called the "seed", arbitrarily chosen, e.g. $I_0 = 1$.

Simple practice using Excel

	A	B	C	D
1				
2	a=	65539		
3	m=	4294967296		
4	c=	618		
5				
6			Integer	Random number
7		0	1	0.0000000002328306
8		1	66157	0.0000154033768922
9		2	40896945	0.0095220620278269
10		3	285286269	0.0664233856368810
11		4	1384145121	0.3222713994327930
12		5	1482827021	0.3452475697267800
13		6	775123345	0.1804724673274900
14		7	4230698781	0.9850363202858710
15		8	1268713409	0.2953953596297650
16		9	3936229805	0.9164749190676950
17		10	3649523569	0.8497209216002370
18		11	3691442365	0.8594809018541120
19		12	2228343969	0.5188267605844880
20		13	1662419021	0.3870620906818660
21		14	2844820305	0.6623613426927470
22		15	2147650653	0.5000388838816430
23		16	207923073	0.0484108629170805
24		17	3434019053	0.7995448664296410
25		18	1593437489	0.3710010761860760
26		19	169789949	0.0395323031116277
27		20	3898171489	0.9076137768570330

χ^2 test of random number generator

- Generate $N_{\rm gen}$ random numbers, R_i, for $i = 1 \ldots N_{\rm gen} (= 10^8)$.
- Use axes $(x, y) = (R_i, R_{i+1})$.
- Divide each axis by the number of bins, $N_{\rm bin} (= 5)$, so that the total number of bin in the $x - y$ plane is $N_{\rm bin}^2 = 25$.
- Then, the mean value of random numbers in each bin is $n = N_{\rm gen}/N_{\rm bin}^2$.
- Count how man random numbers belong to each bin, i.e., b_{ij}, for $i, j = 1 \ldots N_{\rm bin}$, i.e., $\{b_{11}, b_{12}, \ldots, b_{15}, b_{21}, \ldots\}$.
- Calculate χ^2 as

$$\chi^2 = \frac{\sum_{i=1}^{N_{\rm bin}} (b_{ij} - n)^2}{n}$$

- The degree of freedom is $d_f = (N_{\rm bin} - 1) \times (N_{\rm bin} - 1) = 4 \times 4 = 16$.
- Check χ^2 table to test the data set (typically) with 95% confidence level, denoted as $\chi^2_{0.05}$. Pass if $\chi^2 < \chi^2_{0.05}$

Chi Square Distribution Table

d.f.	$\chi^2_{.25}$	$\chi^2_{.10}$	$\chi^2_{.05}$	$\chi^2_{.025}$	$\chi^2_{.010}$	$\chi^2_{.005}$	$\chi^2_{.001}$
1	1.32	2.71	3.84	5.02	6.63	7.88	10.8
2	2.77	4.61	5.99	7.38	9.21	10.6	13.8
3	4.11	6.25	7.81	9.35	11.3	12.8	16.3
4	5.39	7.78	9.49	11.1	13.3	14.9	18.5
5	6.63	9.24	11.1	12.8	15.1	16.7	20.5
6	7.84	10.6	12.6	14.4	16.8	18.5	22.5
7	9.04	12.0	14.1	16.0	18.5	20.3	24.3
8	10.2	13.4	15.5	17.5	20.1	22.0	26.1
9	11.4	14.7	16.9	19.0	21.7	23.6	27.9
10	12.5	16.0	18.3	20.5	23.2	25.2	29.6
11	13.7	17.3	19.7	21.9	24.7	26.8	31.3
12	14.8	18.5	21.0	23.3	26.2	28.3	32.9
13	16.0	19.8	22.4	24.7	27.7	29.8	34.5
14	17.1	21.1	23.7	26.1	29.1	31.3	36.1
15	18.2	22.3	25.0	27.5	30.6	32.8	37.7
16	19.4	23.5	26.3	28.8	32.0	34.3	39.3
17	20.5	24.8	27.6	30.2	33.4	35.7	40.8
18	21.6	26.0	28.9	31.5	34.8	37.2	42.3
19	22.7	27.2	30.1	32.9	36.2	38.6	32.8
20	23.8	28.4	31.4	34.2	37.6	40.0	45.3
21	24.9	29.6	32.7	35.5	38.9	41.4	46.8
22	26.0	30.8	33.9	36.8	40.3	42.8	48.3
23	27.1	32.0	35.2	38.1	41.6	44.2	49.7
24	28.2	33.2	36.4	39.4	32.0	45.6	51.2
25	29.3	34.4	37.7	40.6	44.3	46.9	52.6
26	30.4	35.6	38.9	41.9	45.6	48.3	54.1
27	31.5	36.7	40.1	43.2	47.0	49.6	55.5
28	32.6	37.9	41.3	44.5	48.3	51.0	56.9
29	33.7	39.1	42.6	45.7	49.6	52.3	58.3
30	34.8	40.3	43.8	47.0	50.9	53.7	59.7
40	45.6	51.8	55.8	59.3	63.7	66.8	73.4
50	56.3	63.2	67.5	71.4	76.2	79.5	86.7
60	67.0	74.4	79.1	83.3	88.4	92.0	99.6
70	77.6	85.5	90.5	95.0	100	104	112
80	88.1	96.6	102	107	112	116	125
90	98.6	108	113	118	124	128	137
100	109	118	124	130	136	140	149

Table from Ronald J. Wonnacott and Thomas H. Wonnacott,
Statistics: Discovering its Power, New York: John Wiley and Sons, 1982, p.352.

Program ran0.f

```
      FUNCTION ran0(idum)
      INTEGER idum,IA,IM,IQ,IR,MASK
      REAL ran0,AM
      PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,
     *IR=2836,MASK=123459876)
      INTEGER k
      idum=ieor(idum,MASK)
      k=idum/IQ
      idum=IA*(idum-k*IQ)-IR*k
      if (idum.lt.0) idum=idum+IM
      ran0=AM*idum
      idum=ieor(idum,MASK)
      return
      END
```

- χ^2 is more than **30** while $\chi^2_{0.05} = 26.3$ with $d_f = 16$.
- Therefore, ran0 fails to pass χ^2 test.
- For details, refer to “Numerical Recipes in FORTRAN” Section 7.1.

Main part of program to calculate χ^2

```
x = ran0(idum)
i = int(real(Nbin)*x + 1.0)
do k = 1, Ngen - 1
   x = ran0(idum)
   j = int(real(Nbin)*x + 1.0)
   bin(i,j) = bin(i,j) + 1
   i = j
enddo

open(11,file='ranbin.dat')
do i = 1, Nbin
   write(11,"(100(i10,2x))") (bin(i,j), j = 1, Nbin)
enddo

chi2 = 0.0
do  i = 1, Nbin
   do  j = 1, Nbin
      chi2 = chi2 + (bin(i,j) - binavg )**2
   enddo
enddo
chi2 = chi2 / binavg

write(11,*)
write(11,*) "chi-square is ", chi2
close(11)
```

random num. gen.

- Note: to test other random number generators, change the function to calculate x in the first and fourth lines of the above code.

ran2.f from "Numerical Recipes in FORTRAN"

"We think that, within the limits of its floating-point precision, ran2 provides perfect random numbers; a practical definition of "perfect" is that we will pay $1000 to the first reader who convinces us otherwise. "

```
      FUNCTION ran2(idum)
      INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
      REAL ran2,AM,EPS,RNMX
      PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1,
     *     IA1=40014,IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,
     *     IR2=3791,NTAB=32,NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
```

Long period ($> 2 \times 10^{18}$) random number generator of L'Ecuyer with Bays-Durham shuffle and added safeguards. Returns a uniform random deviate between 0.0 and 1.0 (exclusive of the endpoint values). Call with idum a negative integer to initialize; thereafter, do not alter idum between successive deviates in a sequence. RNMX should approximate the largest floating value that is less than 1.

```
      INTEGER idum2,j,k,iv(NTAB),iy
      SAVE iv,iy,idum2
      DATA idum2/123456789/, iv/NTAB*0/, iy/0/
      if (idum.le.0) then                    Initialize.
        idum=max(-idum,1)                    Be sure to prevent idum = 0.
        idum2=idum
        do 11 j=NTAB+8,1,-1                  Load the shuffle table (after 8 warm-ups).
          k=idum/IQ1
```

Note: FORTRAN 90 has a intrinsic subroutine, random_number(x).

Random number generator to be tested

Perform χ^2 test for

1. ran0.f
2. ran2.f
3. subroutine random_number (x)
4. any code(s) that you find in Internet or books (excluding “NRF”).

Box and Muller's Algorithm

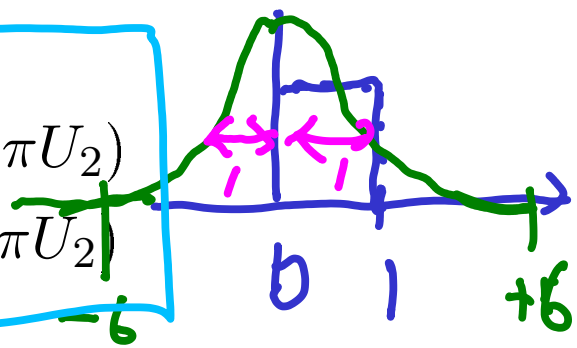
1958

Let U_1 and U_2 be independent random variables from the same rectangular density function on the interval $(0, 1)$, i.e., $0 < (U_1, U_2) < 1$. Consider the random variables:

Levy flight.

$$X_1 = \sqrt{-2 \ln U_1} \cos(2\pi U_2) \quad (1)$$

$$X_2 = \sqrt{-2 \ln U_1} \sin(2\pi U_2) \quad (2)$$



Then X_1 and X_2 will be a pair of independent random variables from the same normal distribution on the interval $(-\infty, +\infty)$, i.e., $-\infty < (X_1, X_2) < +\infty$, with **zero mean and unit variance**.

$$\langle X_i \rangle = 0 \quad \Rightarrow \text{zero mean} \quad (3)$$

$$\langle X_i^2 \rangle = 1 \quad \Rightarrow \text{unit variance} \quad (4)$$

variance = (stdev)^2

```
      SUBROUTINE GaussianBoxMuller2R(Weight,X1,X2)
C
C     Box, G.E.P, M.E. Muller 1958;
C     "A note on the generation of random normal deviates",
C     Annals Math. Stat, V. 29, pp. 610-611
C
      IMPLICIT NONE
      DOUBLE PRECISION U1, U2, X1, X2, Weight, PI
      PARAMETER(PI=3.1415926535897932)
      CALL RANDOM_NUMBER(X1)
      CALL RANDOM_NUMBER(X2)
      U1 = X1
      U2 = X2
      X1= Weight * DSQRT(-2.D0 * DLOG(U1) ) * DCOS(2.0D0*PI*U2)
      X2= Weight * DSQRT(-2.D0 * DLOG(U1) ) * DSIN(2.0D0*PI*U2)
      RETURN
      END
```

change sec

Summation Algorithm

- This method involves two steps and the generation of 12 uniform random variates:
  1. generate 12 uniform random variables, $U_1, \cdots, U_{12}$ in range $(0, 1)$;
  2. calculate

$$X = \sum_{i=1}^{12} U_i - 6$$

- This method yields numbers X which are sampled from an approximately normal distribution (by virtue of the central limit theorem of probability).
- Clearly, random variates outside the range $(-6, +6)$ will never be generated using this method, but it is adequate for most purposes, and is quite fast.

```
DOUBLE PRECISION FUNCTION GaussianRN12(Weight,X)
IMPLICIT NONE
DOUBLE PRECISION X, Xsum, Weight
INTEGER I
Xsum = 0.0D0
DO I = 1, 12
    CALL RANDOM_NUMBER(X)
    Xsum = Xsum + X
END DO
GaussianRN12 = Weight * ( Xsum - 6.D0 )
RETURN
END
```