Inna Williams
##

Section 9.1

##
****

> restart;

```
LCG := proc(m, a, b, x0, N)
        local s, i;
        s[0] := x0;
        for i from 1 to N do s[i] := (a·s[i-1] + b) mod m;
          end do;
        return(seq(s[i], i = 1..N));
       end proc:
```

Computer Problem 3

****

(a) Using calculus, find the area bounded by the two parabolas
P1(x) = x2 - x + 1/2 and
P2(x)= -x2 + x + 1/2.

> $actual\_value := \int_0^1 \left(\left(-x^2 + x + \frac{1}{2} \right) - \left(x^2 - x + \frac{1}{2} \right) \right) dx$

$$actual\_value := \frac{1}{3} \quad (1)$$

Answer: The area bounded by the two parabolas is Area = $\frac{1}{3}$

****

(b) Estimate the area as a Type 1 Monte Carlo simulation, by finding
the average value of P2(x) - P1(x) on [0,1]. Find estimates for n = 10 i for 2 ≤ i ≤ 6.
Appling minimal standard generator

$$\frac{1}{3} \quad (2)$$

> $f := x \rightarrow x^2 - x + \frac{1}{2}$

$$f := x \mapsto x^2 - x + \frac{1}{2} \quad (3)$$

> $g := x \rightarrow -x^2 + x + \frac{1}{2}$

$$g := x \mapsto -x^2 + x + \frac{1}{2} \quad (4)$$

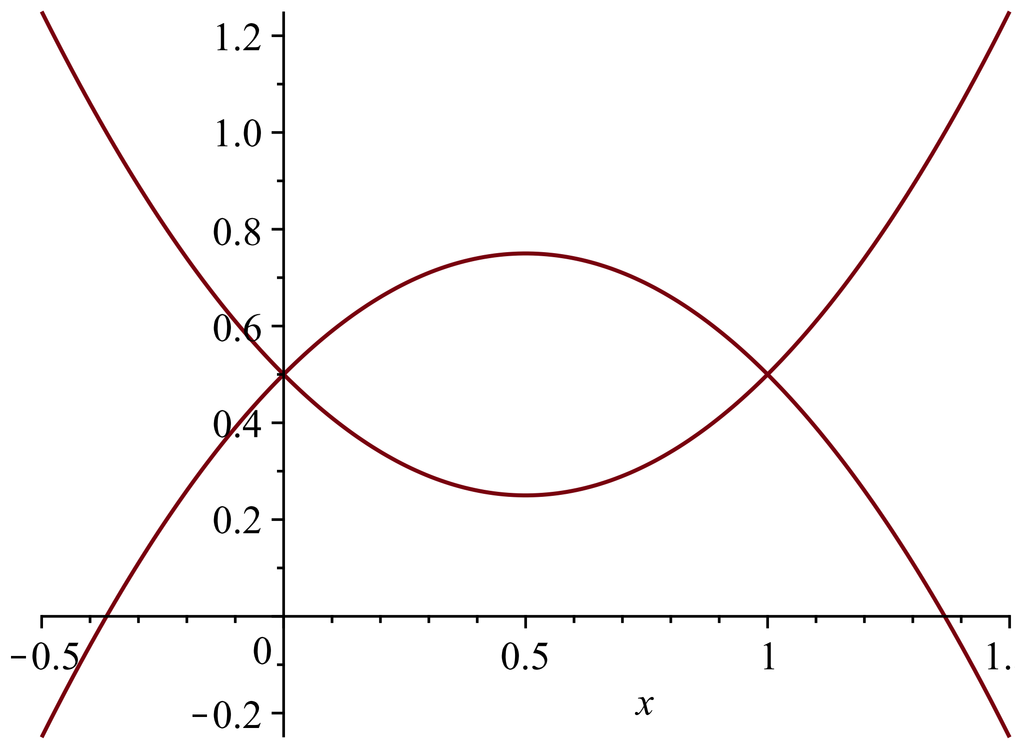
> $with(plots):$

> $g1 := plot(f(x), x = -0.5..1.5):$

> $g2 := plot(g(x), x = -0.5..1.5):$

>

> $display(g1, g2, scaling = constrained)$



> $with(plots):$

> $i := [2, 3, 4, 5, 6]$

$$i := [2, 3, 4, 5, 6] \tag{5}$$

> $x := 10$

$$x := 10 \tag{6}$$

> $\boldsymbol{n} := [x^{i[1]}, x^{i[2]}, x^{i[3]}, x^{i[4]}, x^{i[5]}]$

$$n := [100, 1000, 10000, 100000, 1000000] \tag{7}$$

> $a := n \rightarrow \boldsymbol{evalf}\left(\frac{\boldsymbol{LCG}(2^{31}-1, 7^5, 0, 2, n)}{2^{31}-1}\right)$

$$a := n \mapsto evalf\left(\frac{LCG(2147483647, 16807, 0, 2, n)}{2147483647}\right) \tag{8}$$

> $xvalues := [[\boldsymbol{a}(n[1])], [a(n[2])], [a(n[3])], [a(n[4])], [a(n[5])]]:$

>

>

> $f := x \rightarrow x^2 - x + \frac{1}{2}$

$$f := x \mapsto x^2 - x + \frac{1}{2} \qquad (9)$$

> $g := x \rightarrow -x^2 + x + \frac{1}{2}$

$$g := x \mapsto -x^2 + x + \frac{1}{2} \qquad (10)$$

> $yvalues := [map(g - f,\ xvalues[1]),\ map(g - f,\ xvalues[2]), map(g - f,\ xvalues[3]),\ map(g - f,\ xvalues[4]), map(g - f,\ xvalues[5])]:$

> $est := Vector([Statistics[Mean](yvalues[1]),\quad Statistics[Mean](yvalues[2]),\ Statistics[Mean](yvalues[3]),\quad Statistics[Mean](yvalues[4]),\ Statistics[Mean](yvalues[5])])$

$$est := \begin{bmatrix} 0.314482999598000 \\ 0.328785470515799 \\ 0.330532571431401 \\ 0.332468241228428 \\ 0.333371363431653 \end{bmatrix} \qquad (11)$$

```
> Type_1_MonteCarlo_Estimate_And_Error = Array([[est[1],est[1]
-actual_value],[est[2],est[2]-actual_value],[est[3],est[1]
-actual_value],[est[4],est[4]-actual_value],[est[5], est[5]-
actual_value]])
```

$$Type\_1\_MonteCarlo\_Estimate\_And\_Error \qquad (12)$$

$$= \begin{bmatrix} 0.314482999598000 & -0.0188503337353334 \\ 0.328785470515799 & -0.00454786281753383 \\ 0.330532571431401 & -0.0188503337353334 \\ 0.332468241228428 & -0.000865092104905429 \\ 0.333371363431653 & 0.0000380300983196524 \end{bmatrix}$$

Answer:

$$\boldsymbol{Type\_1\_MonteCarlo\_Estimate\_And\_Error} : \begin{bmatrix} \mathbf{0.327289911864000} & \mathbf{-0.00604342146933323} \\ \mathbf{0.342494092737600} & \mathbf{0.00916075940426658} \\ \mathbf{0.332705280767700} & \mathbf{-0.00604342146933323} \\ \mathbf{0.333610414931441} & \mathbf{0.000277081598107243} \\ \mathbf{0.333505165935823} & \mathbf{0.000171832602489708} \end{bmatrix}$$

****

(c) Same as (b), but estimate as a Type 2 Monte Carlo problem: Find the proportion of points in the square [0,1] × [0,1] that lie between the parabolas. Compare the efficiency of the two

Monte Carlo approaches.
****

> $a := n \rightarrow \mathbf{evalf}\left(\frac{\mathbf{LCG}(2^{31}-1, 7^5, 0, 3, n)}{2^{31}-1}\right)$

$$a := n \mapsto evalf\left(\frac{LCG(2147483647, 16807, 0, 3, n)}{2147483647}\right) \quad \textbf{(13)}$$

> $\mathbf{xvalues} := [[\mathbf{a}(n[1])], [a(n[2])], [a(n[3])], [a(n[4])], [a(n[5])]]:$

> $arearatio := \mathbf{proc}(l)$
> **local** i, k, n;
> $i := 1; k := 0; n := nops(l);$
> **while** $i \le n-1$ **do**
> **if** $l[i+1] \ge f(l[i])$ **and** $l[i+1] \le g(l[i])$
> **then** $k := k+1;$
> **end if**;
> $i := i+1;$
> **end do**;
> **return** $\left(\frac{k}{\mathrm{n}}\right)$;
> **end proc**:

>

>

> $est := Vector([evalf(arearatio(xvalues[1])), evalf(arearatio(xvalues[2])), evalf(arearatio(xvalues[3])), evalf(arearatio(xvalues[4])), evalf(arearatio(xvalues[5]))])$

$$est := \begin{bmatrix} 0.2900000000 \\ 0.3370000000 \\ 0.3473000000 \\ 0.3370700000 \\ 0.3333460000 \end{bmatrix} \quad \textbf{(14)}$$

> $Type\_2\_MonteCarlo\_Estimate\_And\_Error = Array([[est[1], est[1]-actual\_value], [est[2], est[2]-actual\_value], [est[3], est[1]-actual\_value], [est[4], est[4]-actual\_value], [est[5], est[5]-actual\_value]])$

$$Type\_2\_MonteCarlo\_Estimate\_And\_Error = \begin{bmatrix} 0.2900000000 & -0.0433333333 \\ 0.3370000000 & 0.0036666667 \\ 0.3473000000 & -0.0433333333 \\ 0.3370700000 & 0.0037366667 \\ 0.3333460000 & 0.0000126667 \end{bmatrix} \quad \textbf{(15)}$$

Answer:

$$Type\_2\_MonteCarlo\_Estimate\_And\_Error = \left(\begin{bmatrix} 0.2900000000 & -0.0433333333 \\ 0.3370000000 & 0.0036666667 \\ 0.3473000000 & -0.0433333333 \\ 0.3370700000 & 0.0037366667 \\ 0.3333460000 & 0.0000126667 \end{bmatrix}\right)$$

\>

###

5. Use n = 104 pseudo-random points to estimate the interior area of the ellipses

(a) 13x2 + 34xy + 25y2 ≤ 1 in -1 ≤ x,y ≤ 1

Compare your estimate with the

correct areas (a) π/6

###

\> *restart*;

\> *restart*; *with*(*plots*, *implicitplot*) :

\> $A := 13; B := 34; C := 25; F := -1; D1 := 0; E := 0;$

$$A := 13$$
$$B := 34$$
$$C := 25$$
$$F := -1$$
$$D1 := 0$$
$$E := 0 \qquad (16)$$

\> $a :=$

$$evalf\left(\frac{1}{(B^2 - 4\,A\cdot C)}\left(-\left(2\cdot(A\cdot E + C\cdot D1 - B\cdot D1\cdot E + (B^2 - 4\cdot A\cdot C)\cdot F)\cdot\left(A + C + ((A-C)^2 + B^2)^{\frac{1}{2}}\right)\right)^{\frac{1}{2}}\right)\right)$$

$$a := 1.014173944 \qquad (17)$$

\> $b :=$

$$evalf\left(\frac{1}{(B^2 - 4\,A\cdot C)}\left(-\left(2\cdot(A\cdot E + C\cdot D1 - B\cdot D1\cdot E + (B^2 - 4\cdot A\cdot C)\cdot F)\cdot\left(A + C - ((A-C)^2 + B^2)^{\frac{1}{2}}\right)\right)^{\frac{1}{2}}\right)\right)$$

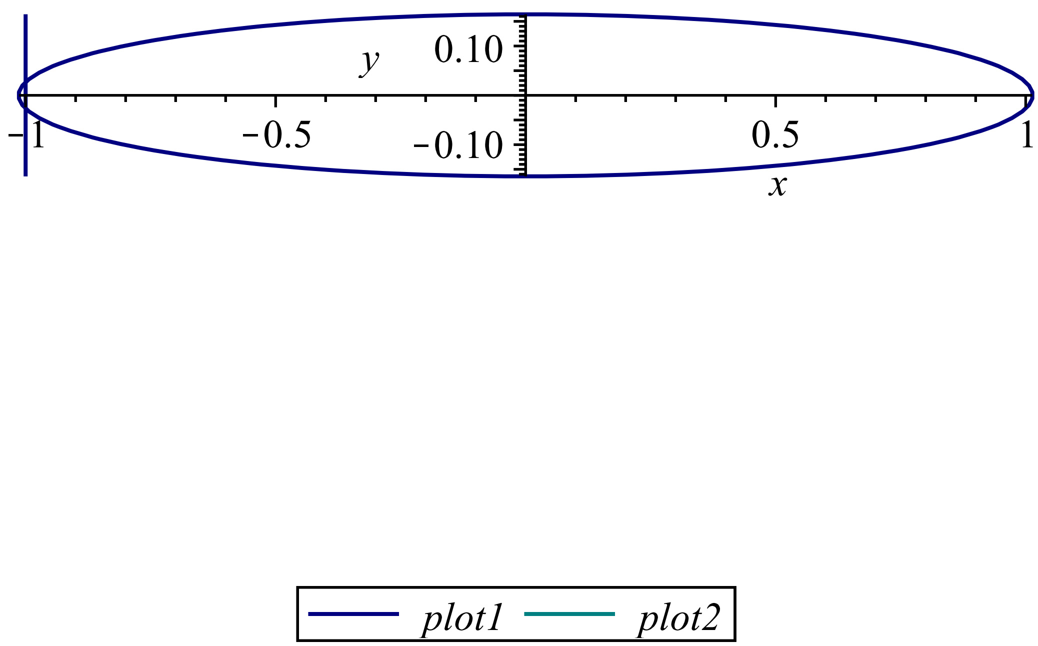
(18)

$$b := 0.1643373587 \quad \textbf{(18)}$$

> $g1(x, y) := \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1;$

$$g1 := (x, y) \rightarrow \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad \textbf{(19)}$$

> $implicitplot([g1(x, y),\ \ y = 1,\ \ x = -1], x = -a..a, y = -b..b, color = ["NavyBlue", "Teal"], legend = [plot1, plot2], scaling = constrained)$



>

```
LCG :=proc(m, a, b, x0, N)
          local s, i;
          s_0 := x0;
          for i from 1 to N do s_i := (a·s_{i-1} + b) mod m;
             end do;
          return(seq(s_i, i=1..N));
        end proc:
```

> $i := [4, 6]$

$$i := [4, 6] \quad \textbf{(20)}$$

> $x := 10$

$$x := 10 \quad \textbf{(21)}$$

> $n := [x^{i[1]}, x^{i[2]}]$

$$n := [10000, 1000000] \quad \textbf{(22)}$$

>

>

>

> $MyRandGen := (M :: posint, r :: positive) \rightarrow$
> $RandomTools\text{:-}Generate(list(float(range = -r..r, method = uniform), M))$

$$MyRandGen := (M::\mathbb{Z}^+, r::positive) \mapsto RandomTools{:}-Generate(list(float(range = -r..r, method = uniform), M)) \quad \textbf{(23)}$$

```
> areaOfTheEllipce := proc(l, m, a, b)
        local i, k, n;
        i := 1; k := 0; n := nops(l);
        while i ≤ n − 1 do
        if abs( f(l[i], m[i]) ) ≤ 1
           then k := k + 1;
        end if;
        i := i + 1;
         end do;
       area_theoretical_value := π/6;
      coefficient_calculated_value := evalf(k/n);
      area_of_the_rectangle_value := evalf(2 * a * 2 * b);
      area_of_the_ellipce_calculated := coefficient_calculated_value
   * area_of_the_rectangle_value;
      absolute_error := abs( area_of_the_ellipce_calculated - area_theoretical_value);
     print(" n = ", n);
     print(" area_theoretical_value = ", area_theoretical_value, "=",
   evalf(area_theoretical_value));

      print(" area_of_the_ellipce _calculated  = ", area_of_the_ellipce_calculated)
      print("absolute_error = ", absolute_error)
  end proc:
```

>

>

> $f := (x, y) \rightarrow \frac{x^2}{a^2} + \frac{y^2}{b^2}$

$$f := (x, y) \mapsto \frac{x^2}{a^2} + \frac{y^2}{b^2} \quad \textbf{(24)}$$

Answer: In the following 4 statement is the answer for 10000 amd 1000000

> $\boldsymbol{xvalues := MyRandGen(n[1],\ a) : yvalues := MyRandGen(n[1],\ b) :}$

>

> $\boldsymbol{areaOfTheEllipce(xvalues, yvalues, a, b) :}$

" n = ", 10000

$$" area\_theoretical\_value = ", \frac{\pi}{6}, "=", 0.5235987758$$

$$" area\_of\_the\_ellipce \_calculated = ", 0.5234000017$$

$$"absolute\_error = ", 0.0001987740 \quad (25)$$

> $xvalues := MyRandGen(n[2], a) : yvalues := MyRandGen(n[2], b) :$

>

> $areaOfTheEllipce(xvalues, yvalues, a, b) :$

$$" n = ", 1000000$$

$$" area\_theoretical\_value = ", \frac{\pi}{6}, "=", 0.5235987758$$

$$" area\_of\_the\_ellipce \_calculated = ", 0.5234286683$$

$$"absolute\_error = ", 0.0001701074 \quad (26)$$

>

##

9. Implement the questionable random number generator from Exercise 5, and draw the plot analogous to Figure 9.3.

##

Randu Number Generator

$m=2^{48}-1, \quad a=2^{24}+3, \quad b=0, \quad x0=1$

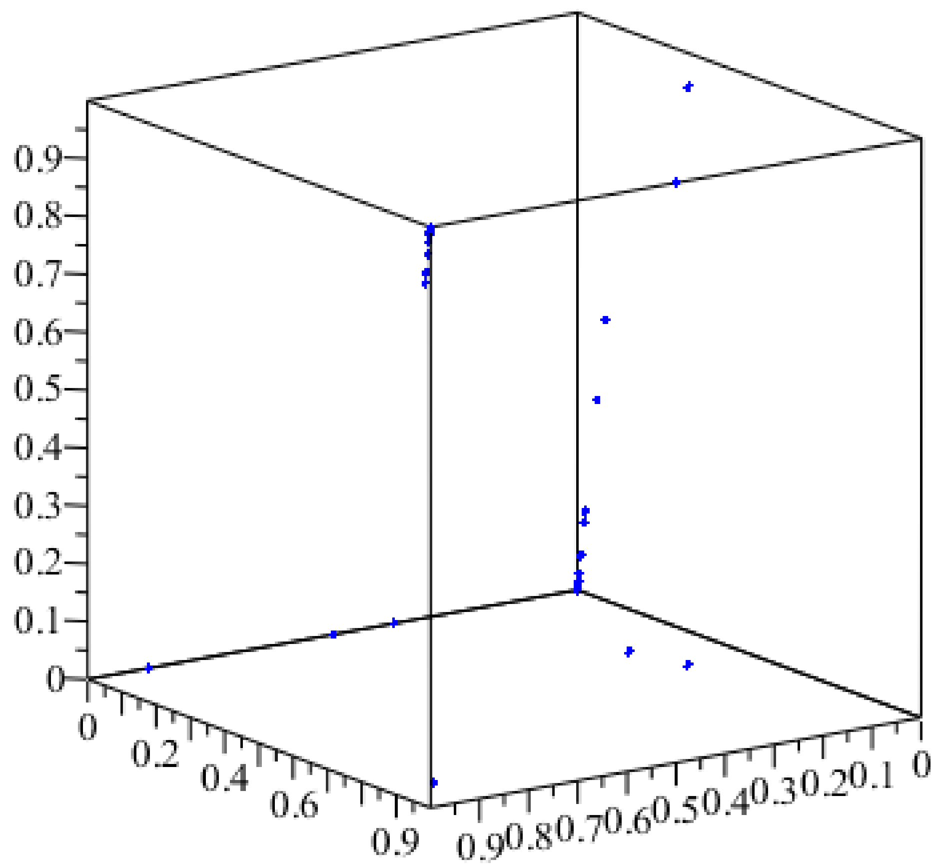
> *restart*;

```
LCG := proc(m, a, b, x0, N)
        local s, i;
        s[0] := x0;
        for i from 1 to N do s[i] := (a·s[i-1] + b) mod m;
           end do;
        return(seq(s[i], i = 1 ..N));
       end proc:
```

> $ulist := evalf\left(\frac{LCG(2^{48}-1, 2^{24}+3, 0, 1, 1000)}{2^{48}-1}\right) :$

> $dim := nops([ulist]) : xyz := [seq([ulist_i, ulist_{i+1}, ulist_{i+2}], i = 1 ..dim - 2)] :$

> $with(plots) : pointplot3d(xyz, axes = boxed, color = blue)$



###

Randu Number Generator

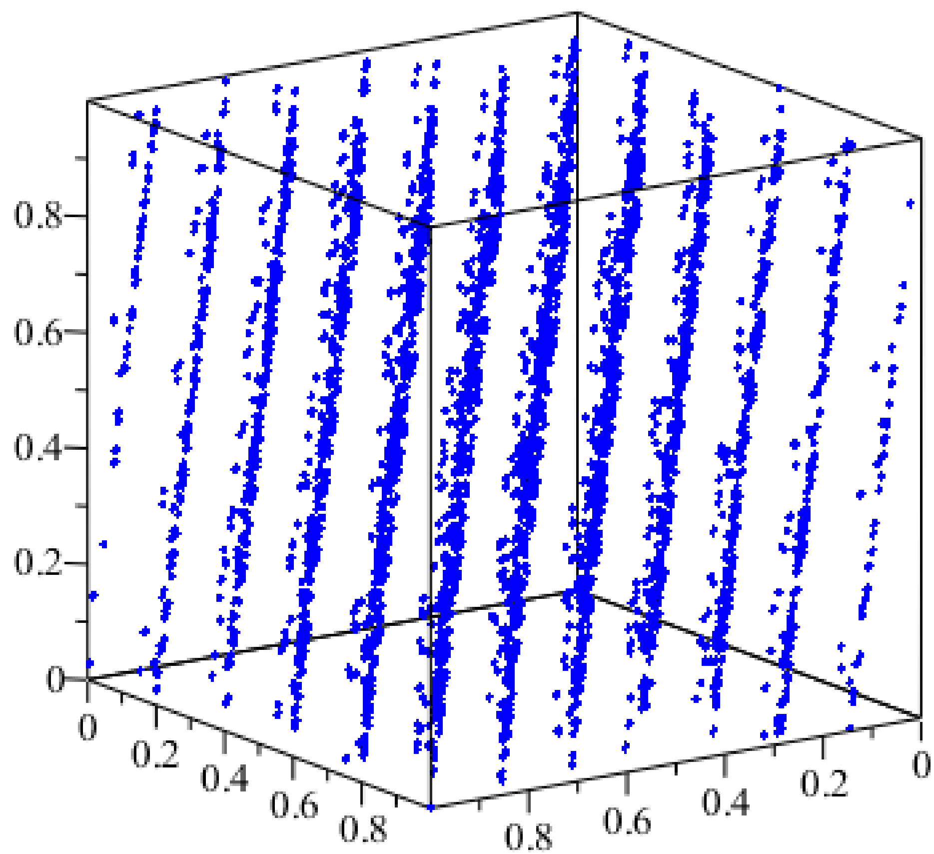
m=$2^{48}-1$, $a=2^{16}+3$, $b=0$, $x0=1$

$$281474976710655, a = 65539, b = 0, x0 = 1 \tag{27}$$

> $ulist := evalf\left(\frac{LCG(2^{48}-1, 2^{16}+3, 0, 1, 10000)}{2^{48}-1}\right):$

> $dim := nops([ulist]) : xyz := [seq([ulist_i, ulist_{i+1}, ulist_{i+2}], i = 1..dim-2)]:$

> $with(plots) : pointplot3d(xyz, axes = boxed, color = blue)$



##

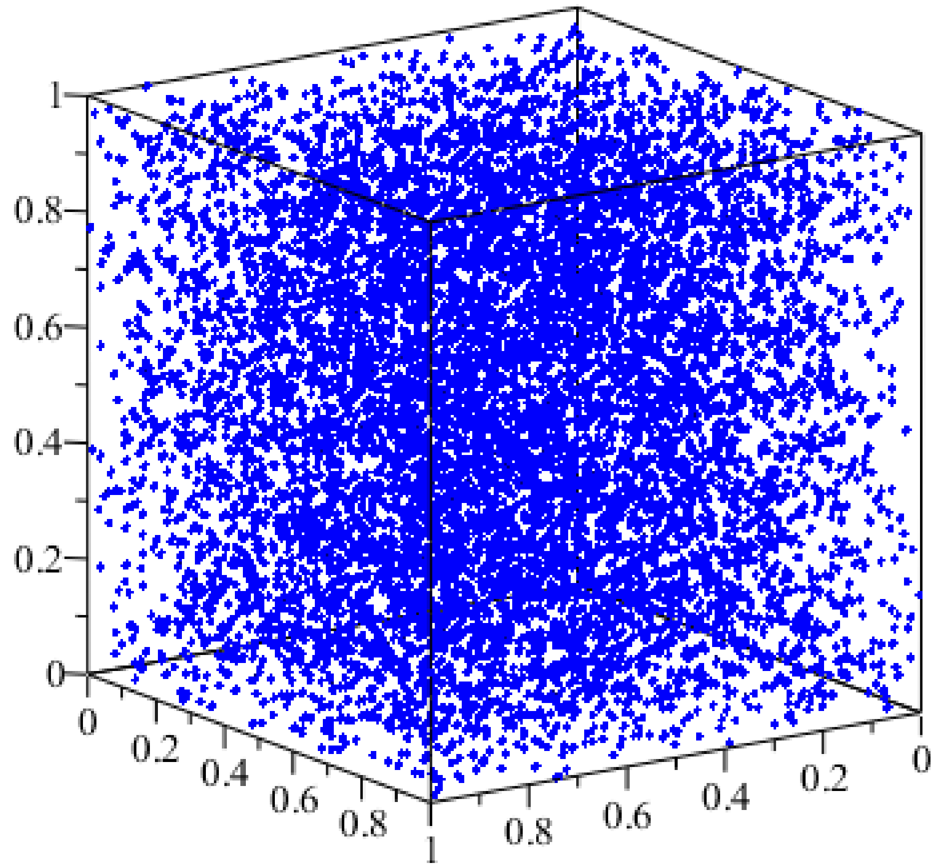
>

Minimal Rnadom Number Generator

> $\boldsymbol{ulist} := evalf\left(\frac{\boldsymbol{LCG}(2^{31}-1, 7^5, 0, 3, 10000)}{2^{31}-1}\right):$

> $dim := nops([ulist]) : xyz := [seq([ulist_i, ulist_{i+1}, ulist_{i+2}], i=1..dim-2)]:$

> $with(plots) : pointplot3d(xyz, axes=boxed, color=blue)$



##

Section 9.3

##

1. Design a Monte Carlo simulation to estimate the probability of a random walk reaching the top a of the given interval [-b,a]. Carry out n = 10000 random walks. Calculate the error by comparing with the correct answer. (a) [-2,5] (b) [-5,3] (c) [-8,3]

```
>
> restart; with(Statistics) :

random := proc(a, b, p)
      local count, pos, x, X;
      count := 0; pos := 0 ;
      X := RandomVariable(Bernoulli(p)) :
      while pos < a and pos > b do
         x := Sample(X, 1);
         if x[1] = 1 then pos := pos + 1
             else pos := pos − 1;
```

```
            end if;
           count := count + 1;
          end do;
          return(pos);
         end proc:
```

```
proc(a, b, p)                                                         (28)
    local count, pos, x, X;
    count := 0;
    pos := 0;
    X := Statistics:-RandomVariable(Bernoulli(p));
    while pos < a and b < pos do
        x := Statistics:-Sample(X, 1);
        if x[1] = 1 then pos := pos + 1 else pos := pos − 1 end if;
        count := count + 1
    end do;
    return pos
end proc
```

All requared Data will be int output

All requared Data will be int output **(29)**

```
> probabilityReachingTop :=proc(a, b, N, p)
   local v := 0;
   local esb := 0;
   local esa := 0;
    local i := 1;
  while i ≤ N do v := random(a, b, p);
   if v ≤ b then esb := esb + 1
   elif v ≤ a then esa := esa + 1
  end if;
  i := i + 1
   end do;
  print("Number Of Reaching    a   = ", esa);
  print("Number Of Reaching   b  = ", esb);
  calculated := evalf(esa/N);
  theoretical_value := abs(b) / abs(a-b);
  p1 := abs(b);
  p2 := abs(a) + abs(b);

  if p > 0.5 or p < 0.5 then
      p1 := abs(b);
      p2 := abs(a) + abs(b);
      p3 := (1-p) / p;
      theoretical_value := ((p3^p1) - 1) / ((p3^p2) - 1);
    else;
     theoretical_value := abs(b) / abs(a-b);
```

```
 end if;
absolute_error := abs(theoretical_value-calculated);
print("theoretical  probability=", theoretical_value, "=", evalf(theoretical_value));
absolute_error := abs(theoretical_value − calculated);
print("calculated probability =", calculated);
print("error=", absolute_error);

end proc:
```

(a) [-2,5]

```
> probabilityReachingTop(5, -2, 10000,0.5)
```

"Number Of Reaching a = ", 2846

"Number Of Reaching b = ", 7154

"theoretical probability=", $\frac{2}{7}$, "=", 0.2857142857

"calculated probability=", 0.2846000000

"error=", 0.0011142857 **(30)**

(b) [-5,3]

> **probabilityReachingTop**(**3, −5, 10000, 0.5**)

"Number Of Reaching a = ", 6190

"Number Of Reaching b = ", 3810

"theoretical probability=", $\frac{5}{8}$, "=", 0.6250000000

"calculated probability=", 0.6190000000

"error=", 0.0060000000 **(31)**

(c) [-8,3]

> ***probabilityReachingTop***(**3, −8, 10000, 0.5**)

"Number Of Reaching a = ", 7332

"Number Of Reaching b = ", 2668

"theoretical probability=", $\frac{8}{11}$, "=", 0.7272727273

"calculated probability=", 0.7332000000

"error=", 0.0059272727 **(32)**

>

##

3. In a biased random walk, the probability of going up one unit is 0 < p <1, and the probability of going down one unit is q = 1 - p. Design a Monte Carlo simulation with n = 10000 to find the probability that the biased random walk with p = 0.7 on the interval in Computer Problem 1 reaches the top. Calculate the error by comparing with the correct answer [(q/p)b - 1]/[(q/p)a+b - 1] for p != q.

The same code will be used as in number 1. Only with p=0.7
and probability will be calculates as:

```
theoretical_value_of_probability := ((((1 - p)/p)^abs(b))-1)/((((1-p)/p)^abs(a + b))-1);
    All requared Data will be int output
```

All requared Data will be int output **(33)**

(a) [-2,5]

```
> probabilityReachingTop(5, -2, 10000,0.7)
```

"Number Of Reaching a = ", 8183

"Number Of Reaching b = ", 1817

"theoretical probability=", 0.8185001387, "=", 0.8185001387

"calculated probability=", 0.8183000000

"error=", 0.0002001387 **(34)**

(b) [-5,3]

```
> probabilityReachingTop(3, -5, 10000, 0.7)
```

"Number Of Reaching a = ", 9861

"Number Of Reaching b = ", 139

"theoretical probability=", 0.9866646754, "=", 0.9866646754

"calculated probability=", 0.9861000000

"error=", 0.0005646754 **(35)**

(c) [-8,3]

```
> probabilityReachingTop(3, -8, 10000, 0.7)
```

"Number Of Reaching a = ", 9990

"Number Of Reaching b = ", 10

"theoretical probability=", 0.9989513813, "=", 0.9989513813

"calculated probability=", 0.9990000000

"error=", 0.0000486187 **(36)**

```
>
```