Practical Numerical Training UKNum

Random numbers, Monte Carlo methods

H. Klahr
Max Planck Institute for Astronomy, Heidelberg

Program:

1) Random Number Generator
2) Transformation Method
3) Monte Carlo Integration

1.0 Random Numbers

Applications

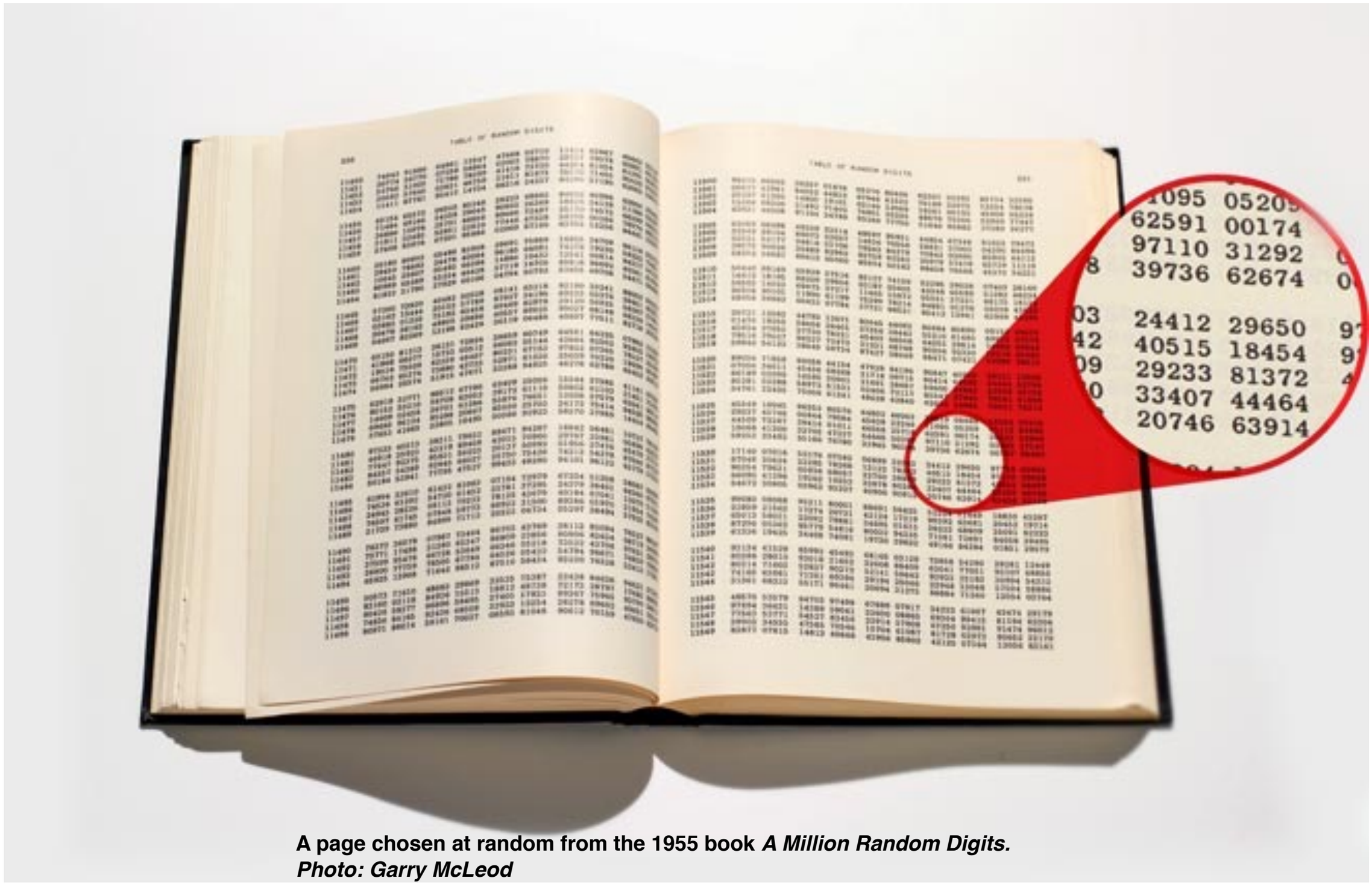
- *Gambling*
- *Physics Simulations*
- *Monte Carlo Methods*
- *Kryptography*
- *...*

Generating Random I

- *Physical Methods ("real" random numbers)*
  - *dice*
  - *Radioactive decay*
    (Time between two decay events)
  - *Noise*
    (f.i. radio frequencies in the earth atmosphere)

- *Advantage: "really" random*
- *Drawback: slow, difficult to control systematic effects.*

1955: A Million Random Digits



A page chosen at random from the 1955 book *A Million Random Digits.*
Photo: Garry McLeod

Generating Random II

- *Numerical Methods ("Pseudo" R.N.)*
  - *so called "Pseudo random number generators" PRNG*
  - *Algorithms that generate numbers in which there is apparently no pattern.*
  - *Start with "seed" to generate sequence of numbers. Deterministic!*
  - *Same seed produces same sequence (important for replicability).*
  - *After N-steps the sequence will start all over.*

Numerical Pseudo-Random-Numbers

- *Advantage: simple and fast.*

- *Drawback: One has to test the sequence if it really is "random".*
  - *f.i. no correlation between pairs of numbers, no preferred numbers, etc.*
  - *Statistical tests.*

- *Quality of RNG depends on application (f.i. length of sequence).*

Numerical RNG

- *There are RNGs that fullfill all needs…*
- *… but also bad ones:*
  - *Historic bad example:*
    randu
  - *IBM mainframes in the1960s. Widely used.*

Result: Many wrong physical results. See below…

1.1 Equal distributed RNG

Why important

- *“Uniform Deviates”*

- *Based on Uniform Deviates one can generate all other distributions via Transformation methods (f.i. normal distribution, Gaussian, Exponential … etc.).*

- *Typical uniform rational numbers in the interval 0 to 1 (depending algorithm including or excluding 0 and 1).*

System-Supplied uniform PRNG

- *Available in many languages and libraries.*
  - *Random class in Java, Apple CarbonLib, glibc (used by gcc), Microsoft Visual/Quick C/C++, etc.*
- *Often low (or undefined) Quality.*
- *Often not portable. -> New platform?*

- *Better: portable PRNG (written in FORTRAN or C).*

Application

- *Typical Calling Sequence (x: REAL, iseed: INTEGER)*

```
x=ran(iseed)
```

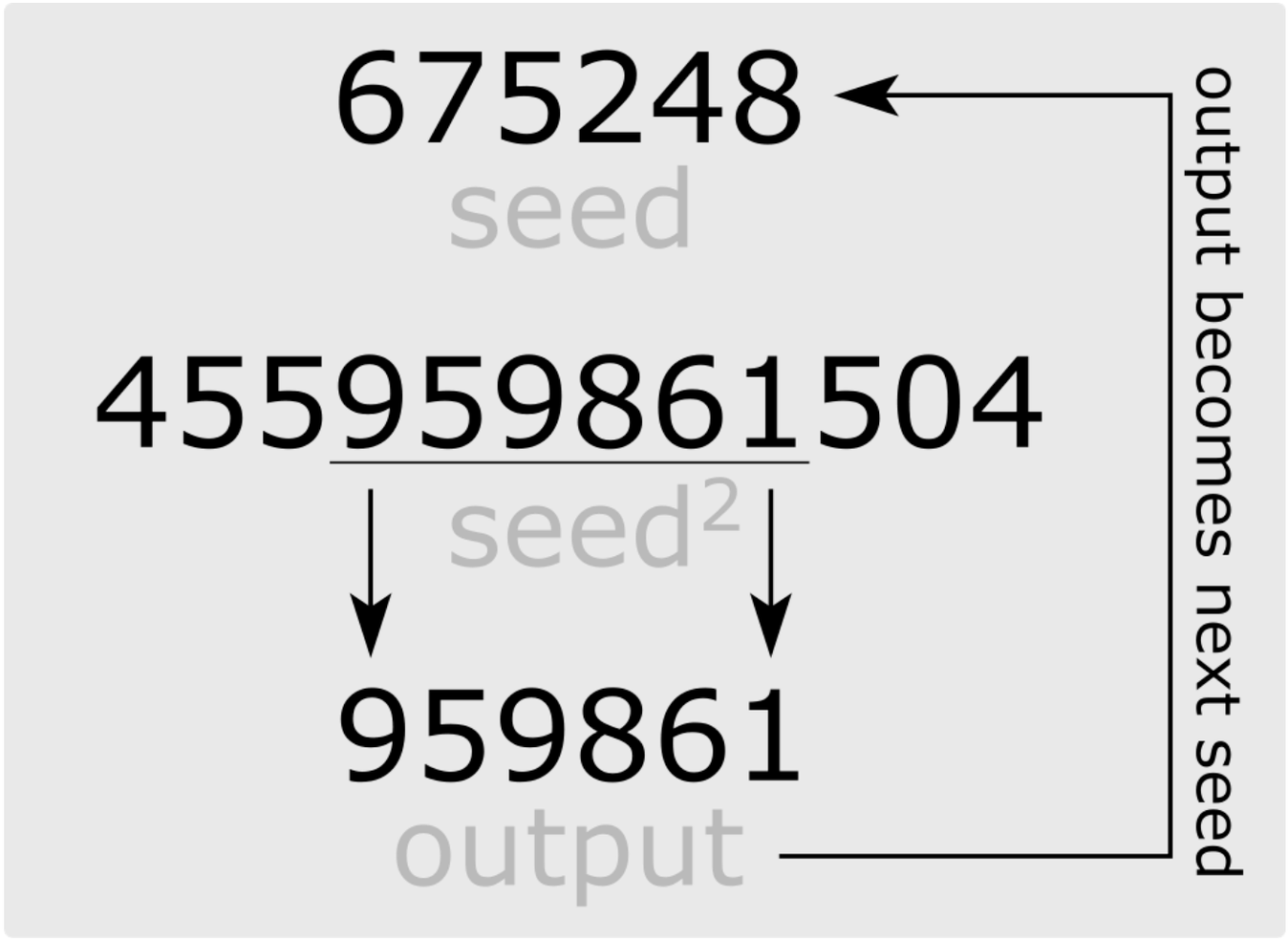
- *First call*
  - *Initialisation with arbitrary seed. Depends on algorithm, sometimes neg. Odd number. For same seed same sequence. Returns new random number x and new value for iseed.*
- *Next call:*
  - *Use last returned iseed. (!)*

PRNG Algorithmen

- *Middle square method*
- *Kongruence generator*
  - *linear*
  - *multiplicativ*
- *Mersenne-Twister*
- *...*

Middle Square method:

- *One of the first PRNG.*
  - *1946: John von Neumann.*



- *Only of historic importance:*
  (Short sequence, crashes at zero, etc.)

Kongruence generator

- *Widely distributed*
  - *Not perfect but simple, quality depends on used parameters, can be improved by some tricks to a useful RNG.*
- *Very fast.*
- *Small demand in memory (cache).*

Linear Kongruence generator

- *Generates sequence of integers: I_1, I_2, I_3 ..., all between 0 and m-1 with m a "large" number.*

- *Recursive definition:*

$$I_{j+1} = aI_j + c \pmod{m}$$

- *m=Modulus (Integer, m>0)*
- *a=Multiplier/Faktor (Integer, 0< a < m)*
- *c=Increment (Integer, 0<= c <m)*

Linear Kongruence generator II

- *Repeats at least after m calls.*
  - *But for bad parameters even earlier!*
- *Good parameters: all possible numbers 0 to m-1 will occur once (Pseudo random permutation).*
- *seed then only determines, where the Sequence is started!*

Linear Kongruence generator III

- *Equal distributed between 0 and 1 is then I_{j+1}/m.*
  - *All Numbers smaller than 1, but once in m calls gives 0.*

- *Requirements for m, a, and c (“Satz von Knuth”) for maximising length of sequence (=m):*
  - *The Increment c up to modulus m has no common dividend.*
  - *All prime factors m divide a-1.*
  - *If m can be divided by 4 then also a-1.*

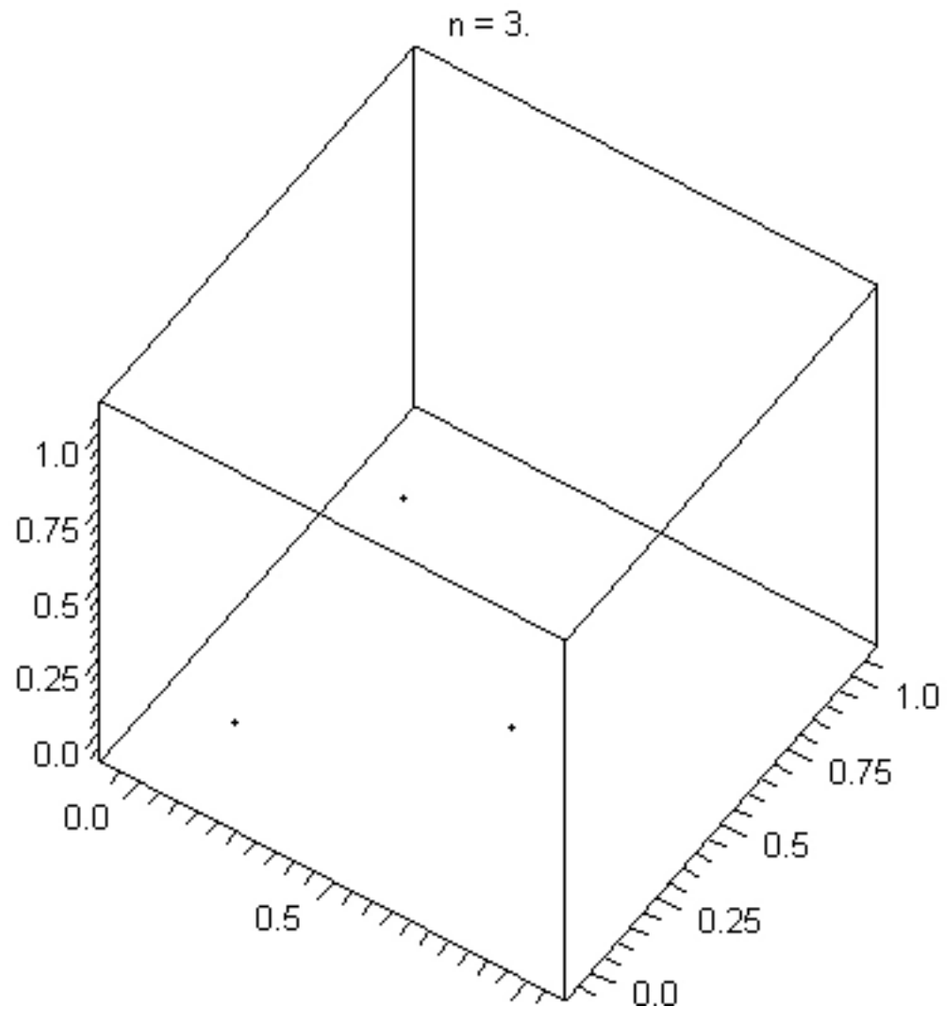
- *Illustration: Mathematica* *"Linear Congruential Generators" from The Wolfram Demonstrations Project http://demonstrations.wolfram.com/LinearCongruentialGenerators/ von Joe Bolte*

Linear Kongruence generator IV

- *Drawbacks*
  - *Sequential correlation (relation between RN pairs), manifested in hyper planes(Satz von Marsaglia):*

    Plotting a sequence of k numbers in a k dimensional space (k1,k2,k3), then the points do not fill the volume, but lie on k-1 dimensional hyper planes. Maximally $m^{1/k}$ Hyper planes, for "bad" values of m,a and c even less.

  - *Different likelihood for different bits within numbers…*



Hyper planes

(There planes k=3)

Multiplicative Kongruence generator

- *Aka Park-Miller PRNG.*
- *Special case: c=0, then:*

$$I_{j+1} = aI_j \pmod{m}$$

- *m and a have to be chosen very carefully. Park and Miller suggest:*

$$a = 7^5 = 16807 \qquad m = 2^{31} - 1 = 2147483647$$

- *So called "Minimal Standard" MINSTD Generator.*
- *Widely distributed, but should not be used for professional application.*

Implementation of MINST

- *Problem: Multiplications of a and I_j lead for some I_j to Integer > 2^{32} (4 byte=32 bit, standard Fortran Integer, C long int)*
  - *Can be remedied with Schrage's Algorithm for approximative Factorisation of m.*
- *Numerical Recipes von Press et al., Cambridge University Press.*
  http://www.nrbook.com/a/

MINSTD, 64 bit Integers

```
FUNCTION ran0long(idum)
!"Minimal"random number generator of Park and Miller.
!Returns a uniform randomd eviate between 0.0 and 1.0.
!Set or reset idum to any integer value(except 0)
!to initialize the sequence; idum must not bealtered
!between calls for successive deviates in a sequence.
!version for long Integer length (8 bytes, 64-bit)
IMPLICIT NONE
INTEGER*8 idum,IA,IM  !long length 8 bytes
REAL ran0long,AM
PARAMETER (IA=16807,IM=2147483647,AM=1./IM)


IF(idum==0)STOP'idum=0 not allowed as seed'
idum=mod(IA*idum,IM)
if (idum.lt.0) idum=idum+IM
ran0long=AM*idum

return
END FUNCTION ran0long
```

Press et al.

MINSTD: sample output

seed = 1

Call No.	x	idum
1	7.82636926E-06	16807
2	0.13153780	282475249
3	0.75560534	1622650073
....		
2147483645	0.65550071	1407677000
$2^{31}-2$ 2147483646	4.65661287E-10	1
2147483647	7.82636926E-06	16807

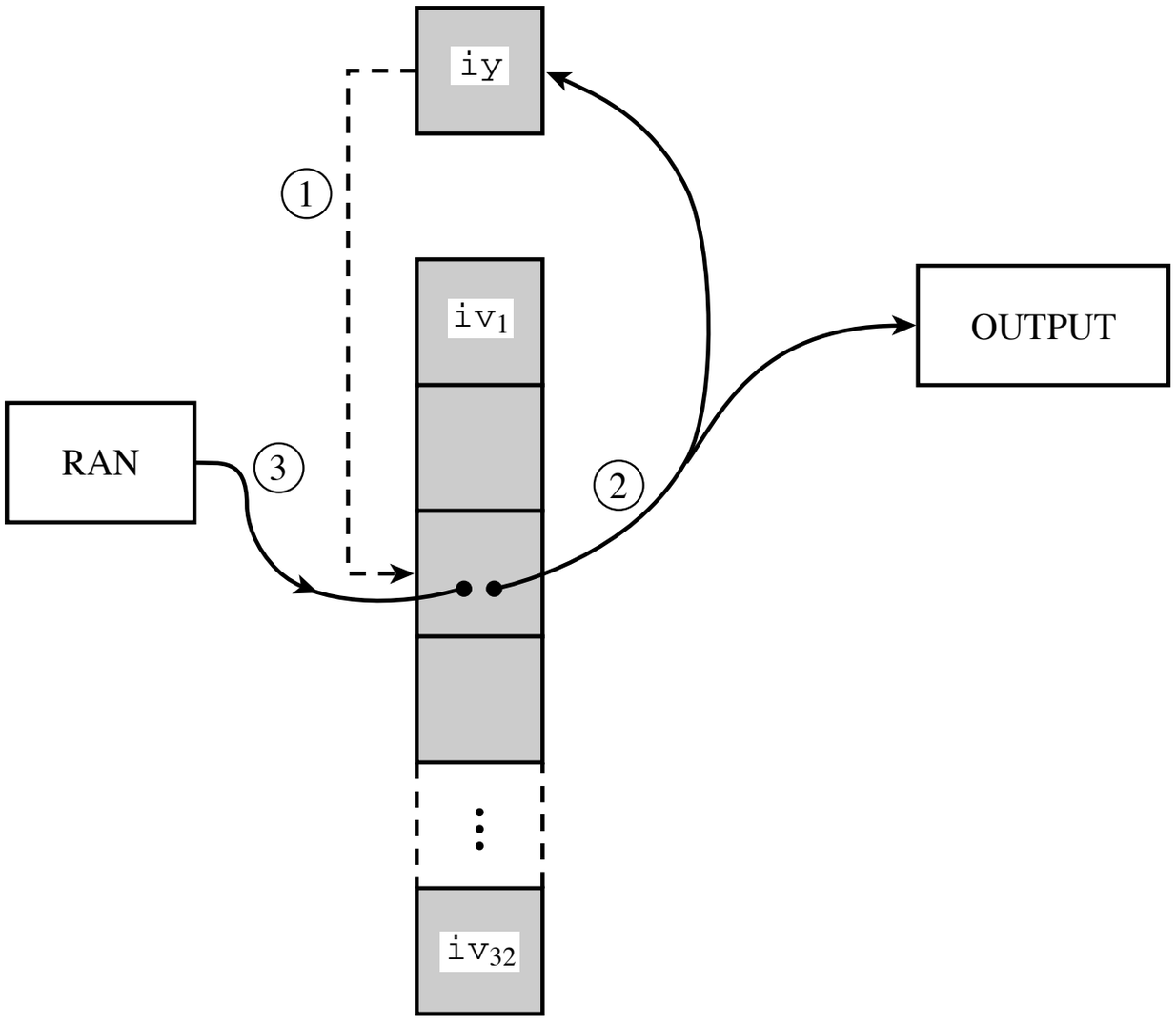
$2^{31} \approx 2\ 10^9$

- *MINSTD Problems:*
  - *Low order serial correlations. Example:*

    Once in 10^6 calls, $x < 10^{-6}$ ausgegeben (which is OK), Once in 10^6 calls, $x < 10^{-6}$ ausgegeben (which is OK), but next number is always < 0.0168, which is not OK.

MINSTD with BD-shuffling

- *Simple method after Bays and Durham, to prevent low order serial correlations:*



Press et al.

RN: I_j, generatend at position j in sequence, is not directly used but randomly after, on average, (j+32) calls later.

MINSTD mit BD-shuffling

```
      FUNCTION ran1(idum)
      INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV
      REAL ran1,AM,EPS,RNMX
      PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,
     *     NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
```

"Minimal" random number generator of Park and Miller 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 j,k,iv(NTAB),iy
      SAVE iv,iy
      DATA iv /NTAB*0/, iy /0/
      if (idum.le.0.or.iy.eq.0) then  Initialize.
        idum=max(-idum,1)             Be sure to prevent idum = 0.
        do 11 j=NTAB+8,1,-1           Load the shuffle table (after 8 warm-ups).
          k=idum/IQ
          idum=IA*(idum-k*IQ)-IR*k
          if (idum.lt.0) idum=idum+IM
          if (j.le.NTAB) iv(j)=idum
        enddo 11
        iy=iv(1)
      endif
      k=idum/IQ                       Start here when not initializing.
      idum=IA*(idum-k*IQ)-IR*k        Compute idum=mod(IA*idum,IM) without overflows by
      if (idum.lt.0) idum=idum+IM       Schrage's method.
      j=1+iy/NDIV                     Will be in the range 1:NTAB.
      iy=iv(j)                        Output previously stored value and refill the shuffle ta-
      iv(j)=idum                        ble.
      ran1=min(AM*iy,RNMX)            Because users don't expect endpoint values.
      return
      END
```

Press et al.

MINSTD with BD-shuffling

- *Perfect up to m/20 (about: 10^8 calls).*
- *Longer Sequences need PRNGs that combine multiple Sequences with different length. Then even a sequence of 2.3×10^{18}*

randu

- *Multiplicativ Kongruence-generator. Recursion:*

$$V_{j+1} = 65539 \cdot V_j \mod 2^{31}$$ *With V_0 odd*

- *Range [1:2^{31}-1]*
- *Rational number in range]0,1[*

$$X_j = \frac{V_j}{2^{31}}$$

- *Chosen parameters (with 32 bit Integer) thus mod 2^{31} and also multiplication with 65539=2^{16}+3 is fast from the hardware side.*

randu - continued II

- *Problem: follow 3 steps (*mod 2^{31}*)*

$$x_{k+2} = (2^{16} + 3)x_{k+1} = (2^{16} + 3)^2 x_k$$

$$x_{k+2} = (2^{32} + 6 \cdot 2^{16} + 9)x_k = [6 \cdot (2^{16} + 3) - 9]x_k$$

As 2^{32} mod $2^{31} = 0$

- *Then*

$$x_{k+2} = 6x_{k+1} - 9x_k$$

Oops!!!

2. Non equal RNG: Transformation Method

Statistical Distribution

- *X is given via (cumulative distribution function, CDF) F(x) beschrieben.*
- *Then F(x) is probability, that X has value between -∞ and x:*

$$F(x) = P(X \leq x) \quad \text{for} \quad -\infty < x < \infty$$

- *Requirement for CDF:*
  - *F(-∞)=0, F(∞)=1*
  - *Monotonically raising*
  - *Continuous*

Statistical Distribution

- *Probably for X to be exactly x_i is 0.*
- *probability density function PDF: f(x) means therefor that X is in the infinitesimal interval of dx: f(x)dx.*

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(x)dx,$$

- *f(x) thus is derivative of F(x).*

Statistical Distribution

- *Probability for X in [a,b] is*

$$P(a \leq X \leq b) = \int_a^b f(x)dx.$$

- *If g(x) a bijective Function of X, then it is also a random distribution. Expected value E of g(X), E(g(x)) is then*
- $$E(g(X)) = \int_{-\infty}^{\infty} g(x)f(x)dx$$

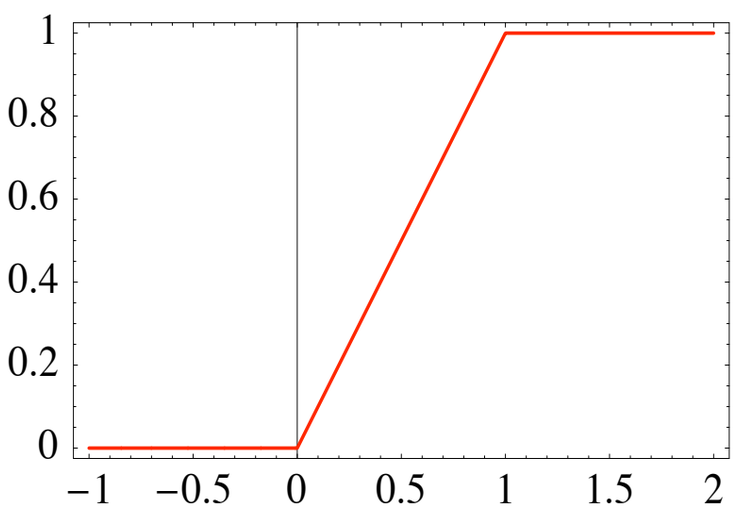
- *Special case if X is the expected value*

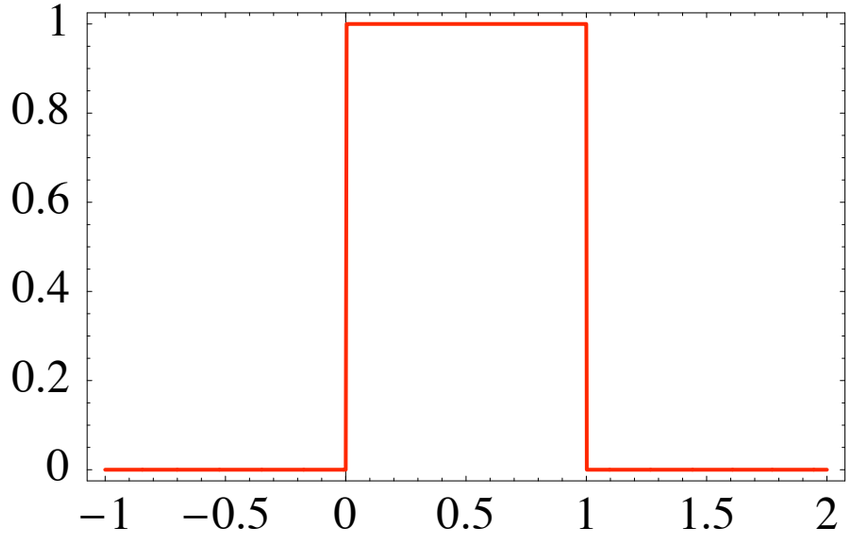
$$E(X) \doteq \mu_x = \int_{-\infty}^{\infty} xf(x)dx$$

Equal distribution in [a,b]

- *Then CDF and PDF*

$$F(x) = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } a \le x \le b \\ 1 & \text{if } x > b. \end{cases} \qquad f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \le x \le b \\ 0 & \text{otherwise} \end{cases}$$





- *Simple case: (a+b)/2, but then we can test the general Eq.:*

$$E(X) = \mu_x = \frac{1}{b-a}\int_a^b x dx = \frac{a+b}{2}$$

Equal distribution in Intervall [0,1]

- *PRNG special case Equal distribution aka standard uniform deviate (SUD).*
- *Simple PDF =1,i.e. f(x)dx=dx inside interval otherwise 0.*
- *Very useful to generate arbitrary distributions with transformation method.*

Transformation methode I

- *If X, follows a SUD. What is then Y, which samples y_i following f(y) ?*
- *Following fundamental law of probabilities:*

$$|f(y)dy| = |f(x)dx|$$

Thus (as f(x)=1 for SUD)

$$f(y) = f(x)\left|\frac{dx}{dy}\right| = \left|\frac{dx}{dy}\right|$$

Transformation-methode II

$$f(y) = f(x)\left|\frac{dx}{dy}\right| = \left|\frac{dx}{dy}\right|$$

- *Solution if D.E. is x=F(y), with F(y) the integral of f(y).*

- *Transformation of SUD from X to Y with f(y) :*

$$y(x) = F^{-1}(x)$$

With F^{-1} the inverse function of F.
So one has to be able to integrate f!

Example

- *Equal distribution Y in Intervall [a,b], and SUD X.*
- *Suggestion: y=x(b-a)+a.*
- *But use Transformations-method:*
- *Be x=F(y)=(y-a)/(b-a), and solve for y auf (reverse -> F^{-1}), thus y(x)=x(b-a)+a*

Example 2: Uniform in lg

- *Y be uniform in* $\log_{10}$ *within [a,b] thus* lg*(x) is uniform in [*lg*(a),*lg*(b)]*

$$F(x) = \begin{cases} 0 & \text{if } x < a \\ \frac{\log(x)-\log(a)}{\log(b)-\log(a)} & \text{if } a \le x \le b \\ 1 & \text{if } x > b. \end{cases}$$

- *Transformation - method (x=F(y) -> inverse),*

$$y = 10^{x(\log(b)-\log(a))+\log(a)}.$$

Generalisation

- *Can be used in multiple dimensions. (Aka Box Muller Transformation)*
- *Not all distribution functions can be integrated. Then use the Rejection method instead.*
- *See discussion in Numerical Recipes: Press et al.*

3. Monte Carlo Methods

Idea Monte Carlo Integration

- *Calculate complex Integrals of Function f in multidimensional Volume V.*
- *Distribute N RND $x_1,..,x_N$ uniform in V.*
- *MC theorem says we can then estimate the Integral as:*

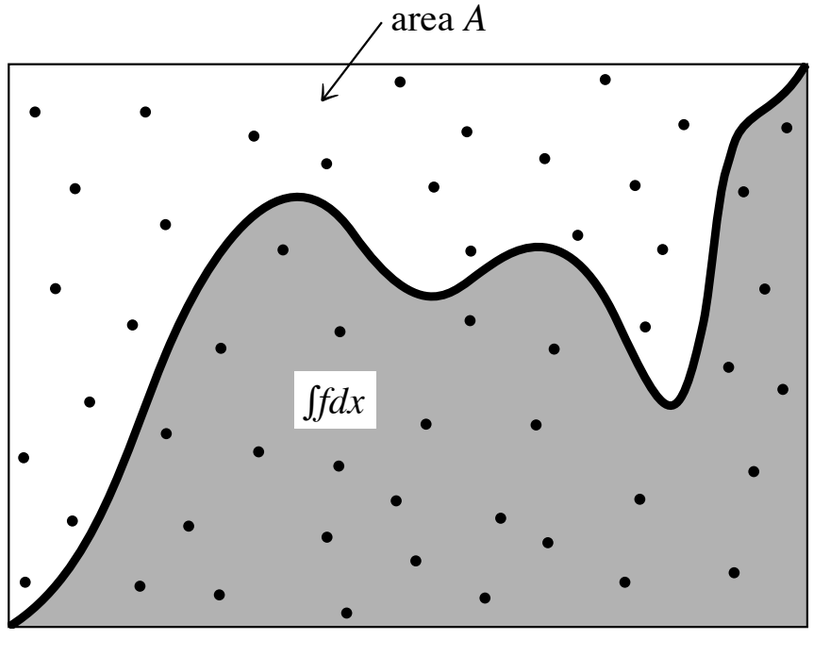
$$\int f \, dV \approx V \langle f \rangle \pm V \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}}$$

- *With <> the arithmetic mean over N sample points.*

$$\langle f \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} f(x_i) \qquad \langle f^2 \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} f^2(x_i)$$

Monte Carlo Integration II

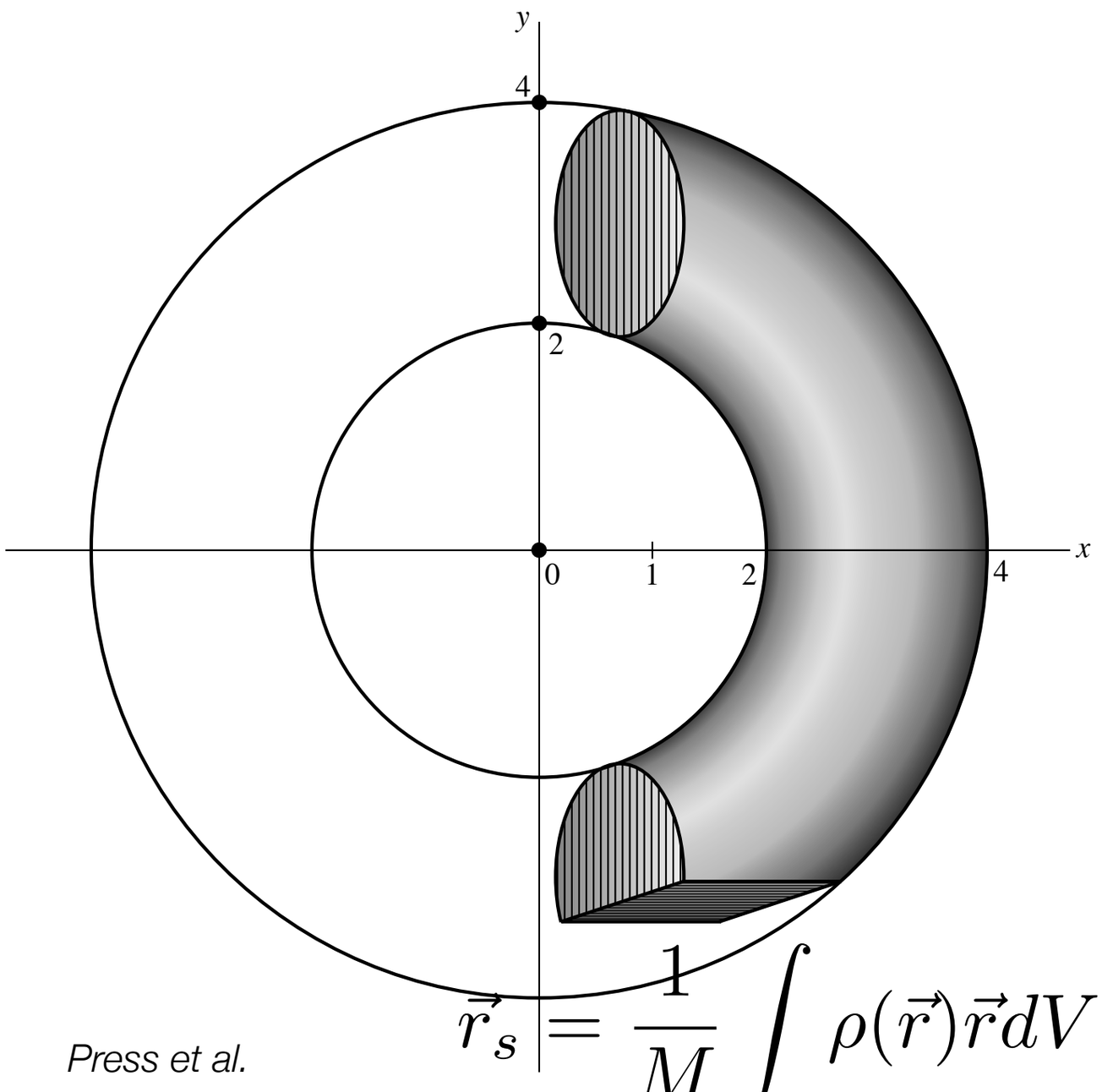
- *In general difficult to have points evenly in V , f.i. complex shape of V.*
- *Define minimal Volume W, that contains V and then set f only for points inside V and =0 for outside V (rejection method).*



Press et al.

Example

- *Center of mass for arbitrary shaped body:*



$$\vec{r}_s = \frac{1}{M} \int \rho(\vec{r}) \vec{r} dV$$

Press et al.

Torus, with certain cuts…

Difficult to integrate analytically.

For MC simple.

Example II

- *Body defined as*

$$z^2 + \left(\sqrt{x^2 + y^2} - 3\right)^2 \leq 1$$

(Torus outer radius = 4, inner radius =3) and planes

$$x \geq 1 \qquad y \geq -3$$

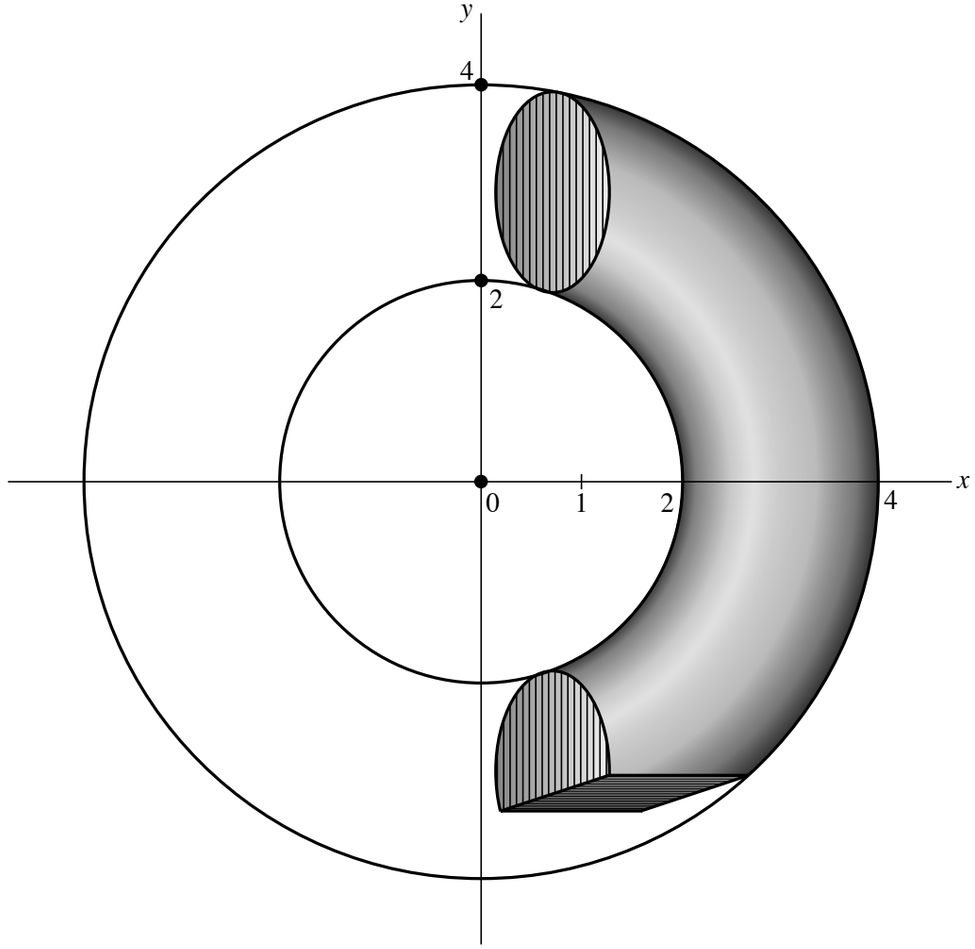
- *Find the Volume for the Integrals*

$$\int \rho \, dx \, dy \, dz \qquad \int x\rho \, dx \, dy \, dz \qquad \int y\rho \, dx \, dy \, dz \qquad \int z\rho \, dx \, dy \, dz$$

- *Assume constant density ρ.*

Example III

- *minimal Volume W, in which we are sampling points uniform in a box of 1 to 4 in x, -3 to 4 in y, and -1 to 1 in z.*



```
Program mcint
IMPLICIT NONE
INTEGER              ::n,idum,j
REAL   ::den,sw,swx,swy,swz,varw,varx,vary,varz,vol
REAL   ::w,x,y,z,dw,dx,dy,dz,ran2

READ(*,*) n    !the number of sample points desired.
den=1.         !the constant value of the density.
sw=0.          !Zero the various sums to be accumulated.
swx=0.
swy=0.
swz=0.
varw=0.
varx=0.
vary=0.
varz=0.
vol=3.*7.*2.   !Volume of the sampled region.
do j=1,n
   x=1.+3.*ran2(idum) !Pick a point randomly in the sampled region.
   y=-3.+7.*ran2(idum)
   z=-1.+2.*ran2(idum)
   if (z**2+(sqrt(x**2+y**2)-3.)**2.le.1.)then   !Is it in the torus?
      sw=sw+den                        !If so, add to the various cumulants.
      swx=swx+x*den
      swy=swy+y*den
      swz=swz+z*den
      varw=varw+den**2
      varx=varx+(x*den)**2
      vary=vary+(y*den)**2
      varz=varz+(z*den)**2
   endif
enddo
```

```
w=vol*sw/n  !The values of the integrals
x=vol*swx/n
y=vol*swy/n
z=vol*swz/n
dw=vol*sqrt((varw/n-(sw/n)**2)/n)   !and their corresponding error estimates
dx=vol*sqrt((varx/n-(swx/n)**2)/n)
dy=vol*sqrt((vary/n-(swy/n)**2)/n)
dz=vol*sqrt((varz/n-(swz/n)**2)/n)

WRITE(*,*)'w,dw',w,dw
WRITE(*,*)'x,dx',x,dx
WRITE(*,*)'y,dy',y,dy
WRITE(*,*)'z,dz',z,dz

END Program mcint
```

$$\vec{r}_s = \frac{1}{M}\int \rho(\vec{r})\vec{r}dV$$

- *Results for n=1000000*

w,dw 22.107918 2.09707543E-02 (Masse des Körpers; Umgebender Quader: 3x7x2=42)
x,dx 53.275669 5.50482012E-02
y,dy 3.4985492 5.61315641E-02
z,dz -1.39091080E-02 1.53482482E-02

- *Center of mass*

$$\vec{r}_s = \begin{pmatrix} 2.4098 \\ 0.1582 \\ -0.0006 \end{pmatrix}$$

At least plausible (z-component =0 … Symmetry!)

- **Exercise 1, 6 points:** The infamous `randu`.
  Write a program code with the (not-so) random number generator randu following the recurrence:

$$I_{j+1} \;=\; (65539\; I_j) \left(\mathrm{mod}\; 2^{31}\right)$$

  (see Lecture for details). Then to obtain a random number x_i drawn from the interval $(0, 1)$ use the normalisation $x_i = u_i/2^{31}$. Create some 100 000 random number, starting with an initial seed $u_0 = 1$ and plot the consecutive triples (x_i, x_{i+1}, x_{i+2}) in a 3-dimensional plot, e.g., using the `splot` command from gnuplot. Count the number of 2D planes by viewing the data in different projections. What is the number of planes for `randu`?

- **Exercise 2, 8 points:** Transformation method.
  Write a program code to generate random numbers with an exponential probability distribution function (PDF) $\rho(y) = e^{-y}$ in the interval $y_{\mathrm{min}} = 0$ to $y_{\mathrm{max}} = 5$ using the transformation:

$$y = -\ln(1 - x) \quad \Leftrightarrow \quad x = e^{-y}.$$

  Use an random number generator with uniform PDF of your choice or the one offered in the Lecture. Show that your resulting distribution of random numbers indeed follows an exponential one.

- **Exercise 3, 6 points:** Monte-Carlo Integration.
  Approximate the value of π using the Monte-Carlo technique by integrating the area of a square with side length a and a circle of radius $1/2a$. Use the equation:

$$\pi = 4\frac{A_c}{A_s} \approx 4\frac{N_c}{N_s},$$

  where A_c and A_s is the area of the square and the circle, respectively. How does the precision of the result scale with the number of points used in the integration?