NERS544: (Introduction to) Monte Carlo Methods

Assignment 6 and 7: **Reflection** 2-week assignment, weight = 2 Fall 2016

Revision: November 3, 2016
Due: November 17, 2016 before class

Alex Bielajew, 2927 Cooley, bielajew@umich.edu

This assignment will involve surface with mirror-reflection properties (See section 10.7 in the book), as well as totally absorbing surfaces.

Consider a two dimensional square box centered at the origin with sides $2L$ in length. Also centered at the origin is a circle of radius R. For this assignment you should show results for $L = 1$ and $R = 0.2$, although your code should be generalizable to run for any L and R. (This may be useful when you are debugging your code as well.)

You will consider two source types, a "volume source" that emits particles uniformly and isotropically throughout the square. Those generated within the absorbing circle are "absorbed on the spot", at their point of creation. Otherwise they are transported inside the box until they reflect from one of its sides, or gets absorbed on the surface of the circle. (Graphical output helps a lot with debugging the code.) In addition, if a particle is about to take a step larger than $100L$, its transport is halted. (This is a great time saver.)

Tally the total pathlength each particle history accumulates following the example output attached to this description.

For the graphs given, the tally mesh is described by:

```
M = 256; % Mesh size of the distribution tally
if     (surfaceSource) pathlog = logspace(log10(L-R),2,M); % L-R is the minimum
elseif (volumeSource)  pathlog = logspace(-1,2,M); % Minimum can be zero
end
```

As well, your code must visually display the particle tracks for the first 100 histories. This will be demonstrated in class.

Submit your code by email.

Discussion. This is "food for thought" only.

- The connection of this problem to reactor physics should be apparent. The construction of this problem is over-simplified for this application, yet it contains features that are common, in particular the periodic boundary conditions (infinite. At this point in your Monte Carlo development, it should be clear how to add different absorption or scattering constants for the "fuel bundle" (the circle) and the moderator (everything else).
- Pathlength distributions are very important for nuclear and radiological applications because the ability for a particle to interact with the medium it is being transported in, is proportional to the number of atoms or nuclei it encounters along its path.
- It is a mathematically interesting problem, one that is challenging to express mathematically, but easy to describe, and the limits if its behavior, easy to figure out. Hence, it is an ideal candidate for numerical study. For example, once could ask:
  - What happens as the ratio R/L is varies. The limits are easy to express. For example, when R/L approaches $\sqrt{2}$, there is no space left for particles to be transported. As R/L approaches 0 the pathlengths become infinite. What exactly do the distributions look like, for different R/L.

– One could provide the pathlength distributions for the 0 walls struck, 1 wall struck and so on. The 0-wall and 1-wall might be doable analytically. More than that may be prohibitive.
– How would the distributions change if the transport medium was weakly absorbing, or the walls only partially mirrored? Leakage could be modeled!
– How about 3D and more-D behaviors?

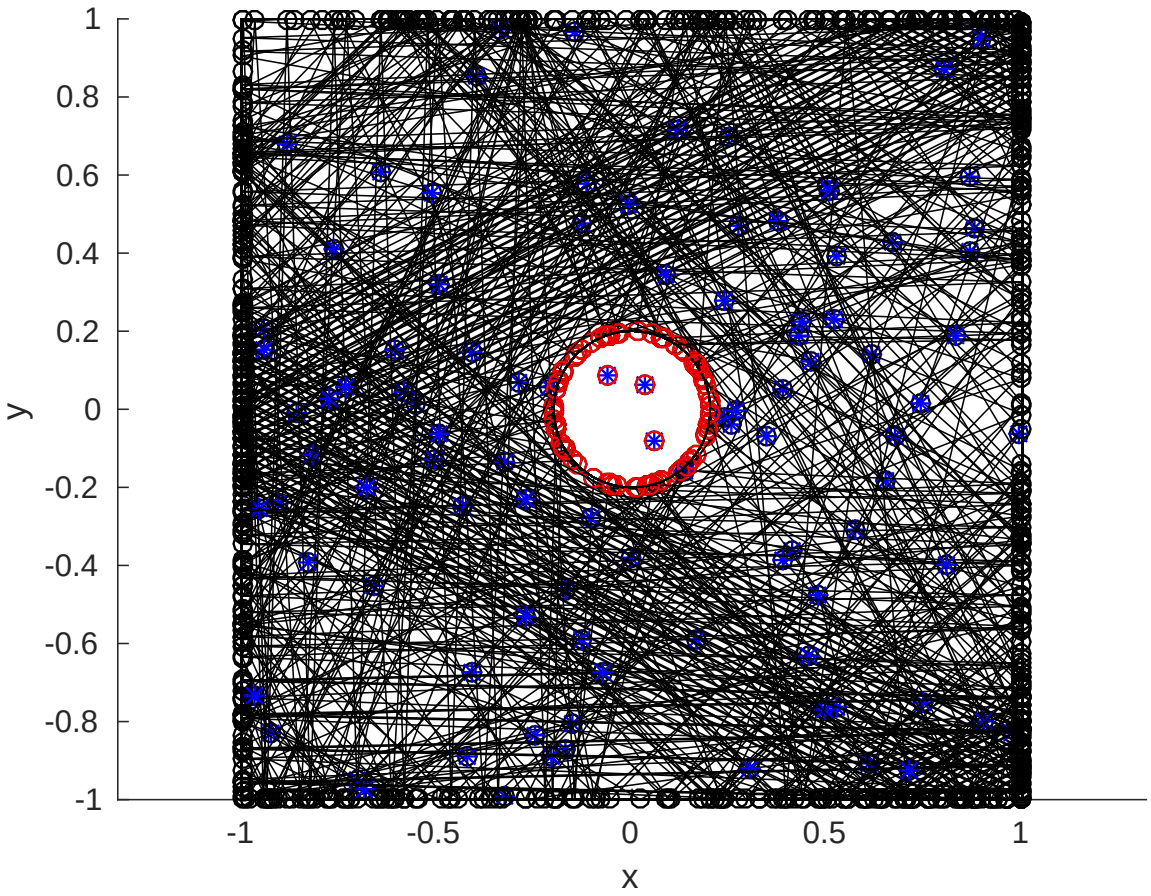


Figure 1: Volume source, 2×2 box, absorber $R = 0.2$, tracks of 100 histories. A particle's starting location is represented with a blue asterisk, its final location by a red circle. Note the particles that started and ended immediately in the interior of the absorber.

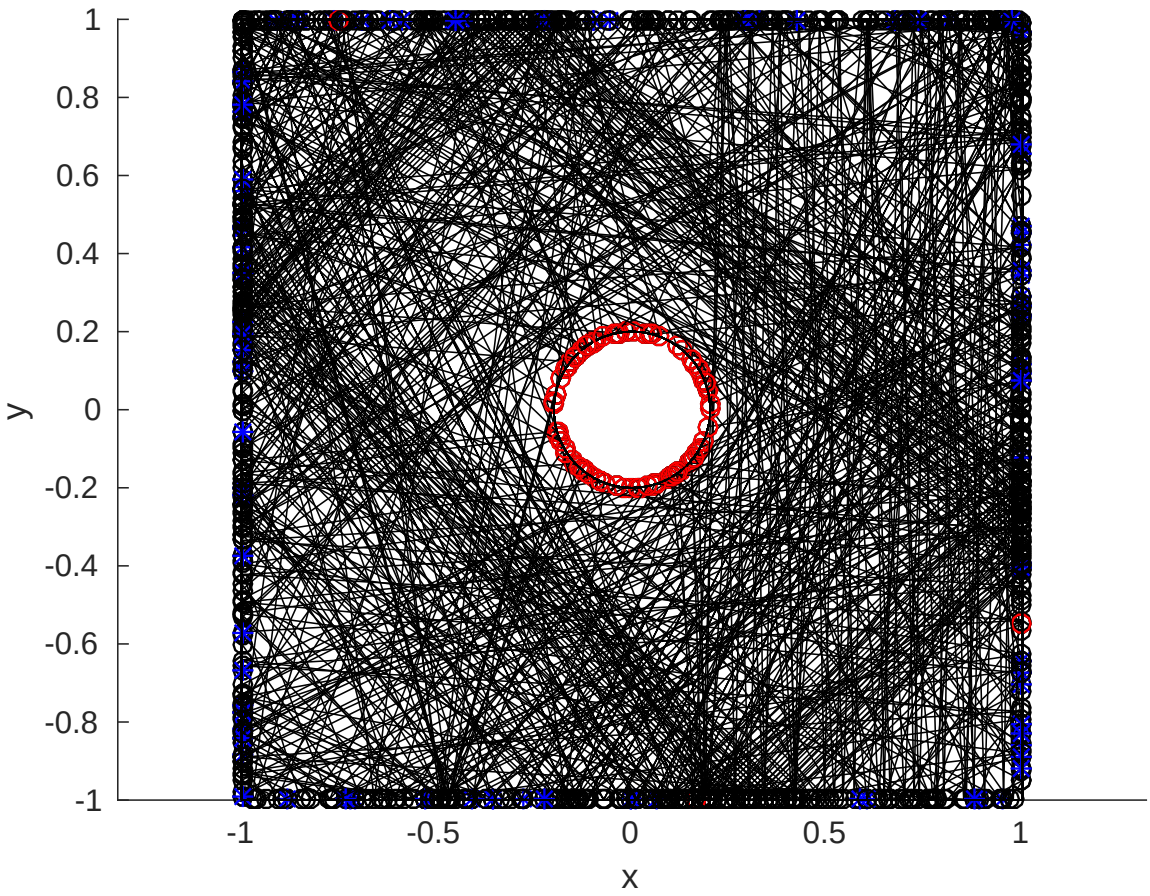


Figure 2: Surface source, 2×2 box, absorber $R = 0.2$, tracks of 100 histories. A particle's starting location is represented with a blue asterisk, its final location by a red circle. Note that two of the particles ended at the edge of the box due to the pathlength restriction.

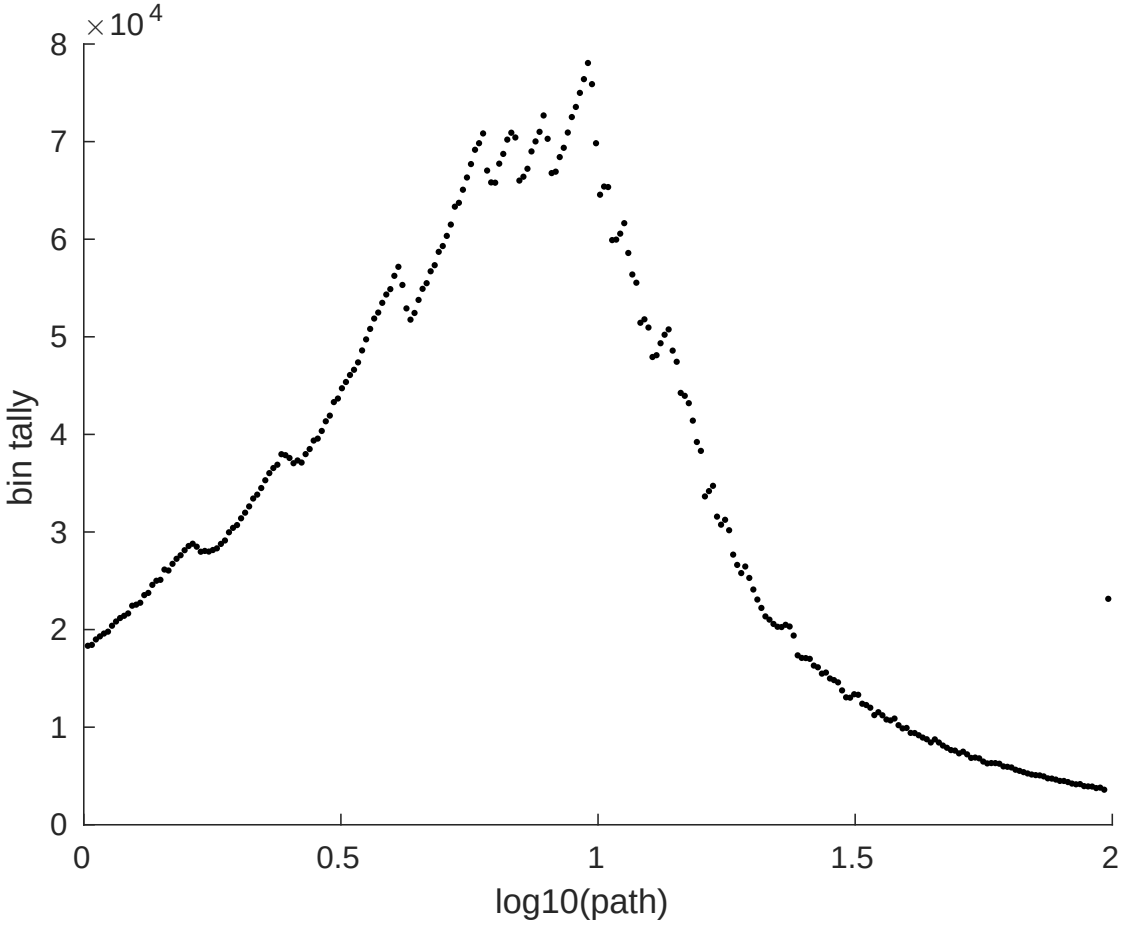


Figure 3: Volume source, 2×2 box, absorber $R = 0.2$, 10^7 histories.

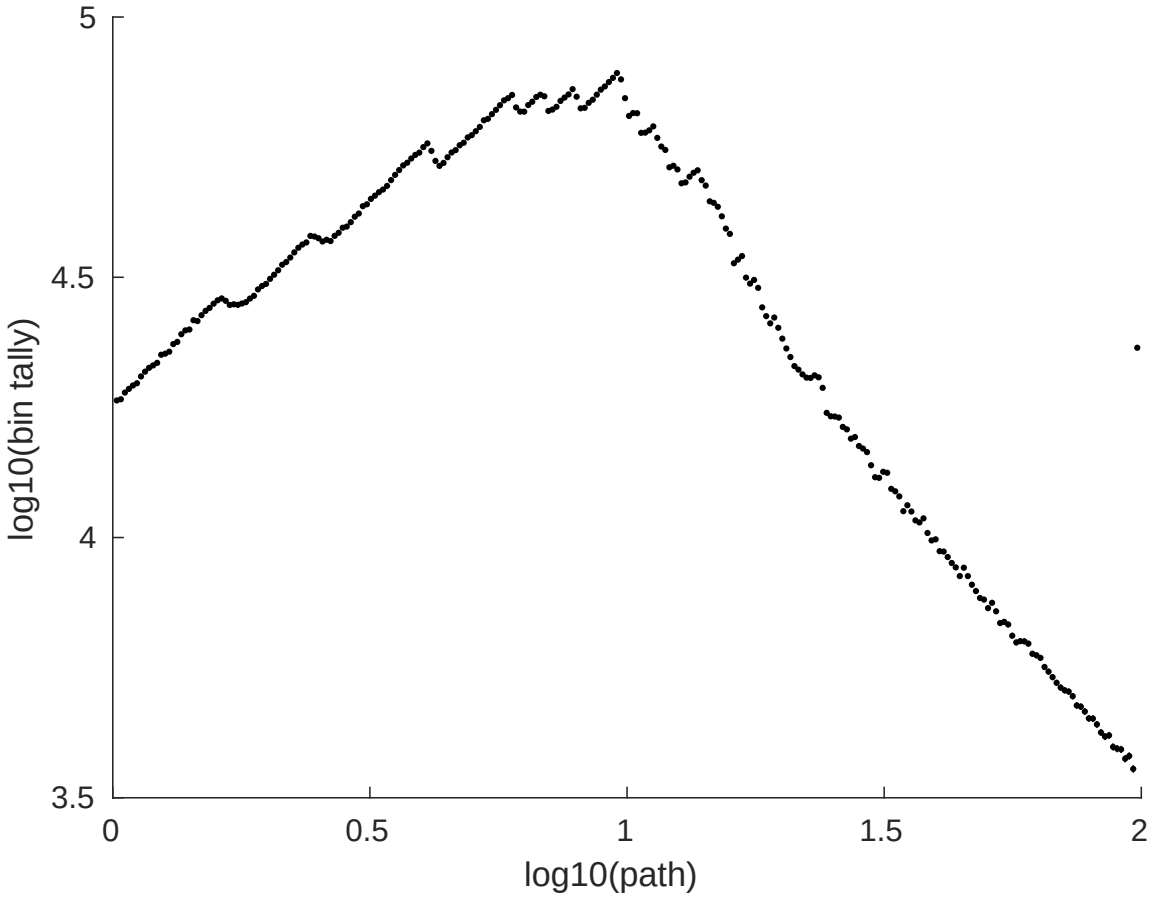


Figure 4: Same as before Figure 3, but with a logarithmic ordinate axis.

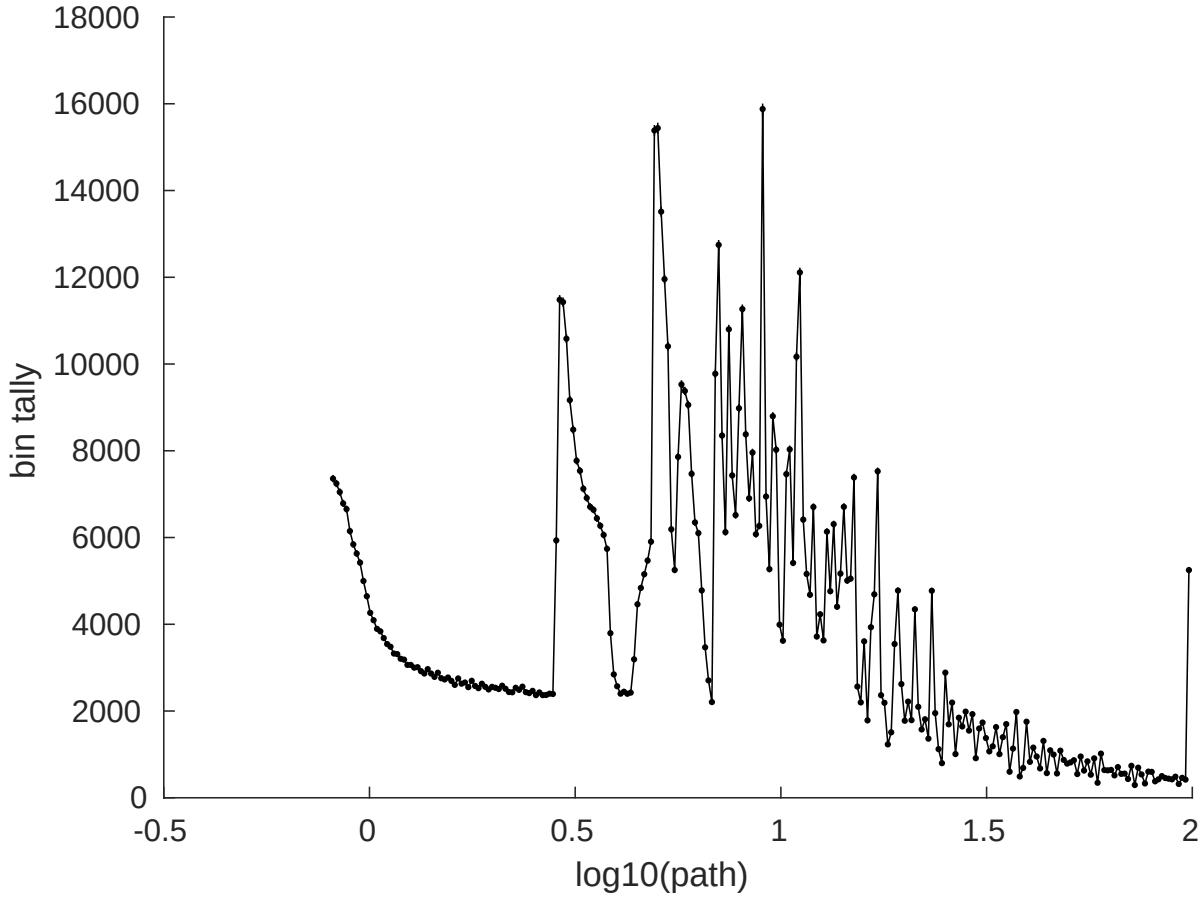


Figure 5: Surface source, 2×2 box, absorber $R = 0.2$, 10^6 histories.

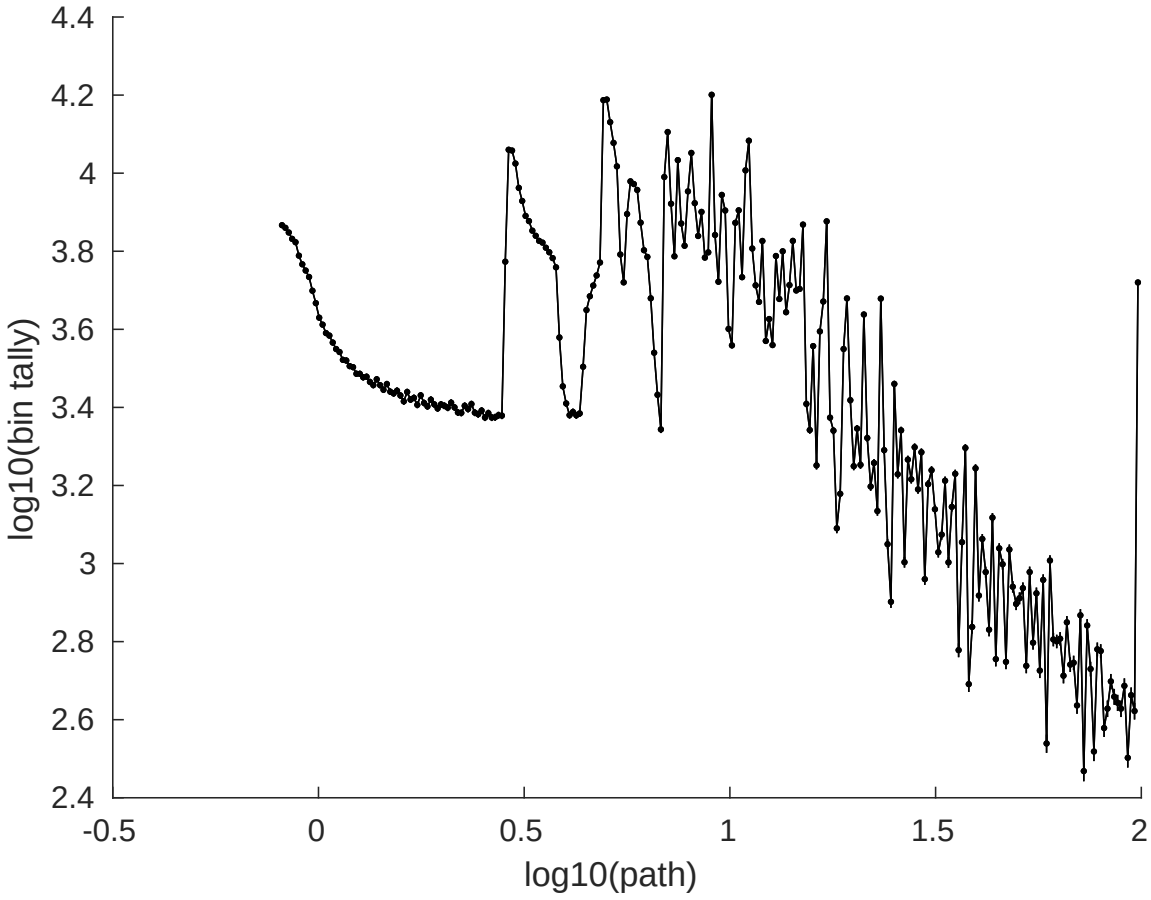


Figure 6: Same as before Figure 5, but with a logarithmic ordinate axis.

Lecture 09

Quadrature in N dimensions (11.2)

Comparison of deterministic & Monte Carlo methods

Preamble

A formal solution to the transport equation is

(11.4) $$\Psi(x,p,s) = \int ds' \int dp' G(x,p,x',p',s) Q(x',p')$$

- $\Psi(x,p,s)$: x = vector of pos'n; p = vector of momentum; s = how far along the particle is on its path
- G = Green's Function (contains all the physics)
- Q = source

Conclusion MC particle simulation could be solved if we know G and did the numerical integrations.

It is 6 dimensions, in generality

So, let's consider integration as an abstract process.

11.2 Convergence

Consider a D dimensional integral

(11.11) $$I = \int_D du\, H(u)$$ ← some function

11.2.1 One dimension

midpoint average

(11.12) $$I = \int_{u_{min}}^{u_{max}} du\, H(u) = \sum_{i=1}^{N_{cell}} \int_{u_i - \frac{\Delta}{2}}^{u_i + \frac{\Delta}{2}} du\, H(u)$$

split into N_{cell} cells

Write $H(u)$ in terms of a Taylor expansion about u_i, for the i^{th} cell

i.e.

(11.13) $$H(u) = H(u_i) + (u-u_i) H'(u_i) + \frac{1}{2}(u-u_i)^2 H''(u_i)$$

(11.13) ⇝ (11.12) ⇒

ΔI, error estimate

$$I = \Delta u \sum_{i=1}^{N_{cell}} \left(H(u_i) + \underbrace{\frac{(\Delta u)^2}{24} H''(u_i)}_{\Delta I} \right) \ldots$$

The $u - u_i$ term vanished due to anti-symmetry about u_i

estimate of error

$$(11.15)\quad \frac{\Delta I}{I} \cong \frac{1}{24}\,\frac{(u_{max}-u_{min})^2}{N_{cell}^2}\,\frac{\sum_{i=1}^{N_{cell}} H''(u_i)}{\sum_{i=1}^{N_{cell}} H(u_i)}$$

Two dimensions

$$(11.19)\quad \frac{\Delta I}{I} \cong \frac{1}{24}\,\frac{1}{N_{cell}}\left\{\text{long expression}\right\}$$

D dimensions

$$(11.22)\quad \frac{\Delta I}{I} \cong \frac{1}{24}\,\frac{1}{N_{cell}^{2/D}}\left\{\text{very long expression}\right\}$$

Convergence of Monte Carlo Solutions (11.3)

T_{mc} is some sort of tally

$$\frac{\Delta T_{mc}}{T_{mc}} \cong \frac{1}{\sqrt{N_{hist}}}\,\frac{\sigma_{mc}}{T_{mc}}$$

σ_{mc}: width of the prob. dist. of T_{mc}

After some work }

intrinsic variances

$$\frac{t_{NMC}}{t_{MC}} = \left(\frac{\alpha_{NMC}}{\alpha_{MC}}\right) \frac{\left[\sigma_{NMC}/\overline{T}_{NMC}\right]^{D/2}}{\left[\sigma_{MC}/\overline{T}_{MC}\right]} \varepsilon^{(4-D)/2}$$

- $\frac{t_{NMC}}{t_{MC}}$: ratio of time
- $\left(\frac{\alpha_{NMC}}{\alpha_{MC}}\right)$: algorithmic efficiency
- $\frac{\left[\sigma_{NMC}/\overline{T}_{NMC}\right]^{D/2}}{\left[\sigma_{MC}/\overline{T}_{MC}\right]}$: fully value

should have $\overline{T}_{NMC} \approx \overline{T}_{MC}$

$\varepsilon = \frac{\Delta T}{T}$, desired accuracy

Depending on ( ) : [ ] / [ ] factors

MC is suggested to be more efficient for $D > 4$

Note MC type calculations are $D = 6$

3 for $\vec{x}$

3 for $\vec{p}$

We will not endeavor to explain the theory behind random number generation, merely give some guidelines for good use. The operative phrase to be used when considering RNG's is "use extreme caution". **DO USE** an RNG that is known to work well and is widely tested. **DO NOT FIDDLE** with RNG's unless you understand thoroughly the underlying mathematics and have the ability to test the new RNG thoroughly. **DO NOT TRUST** RNG's that come bundled with standard mathematical packages. For example, DEC's `RAN` RNG (a system utility) and IBM's `RANDU` (part of the SSP mathematical package) are known to give strong triplet correlations. This would affect, for example, the "random" seeding of an isotropic distribution of point sources in a 3-dimensional object. A picture of an artefact generated by these RNG's is given in Figure 3.1. This is known as the "spectral" property of LCRNG's.

The gathering of random numbers into planes is a well-known artefact of RNG's. Marsaglia's classic paper [Mar68] entitled "Random numbers fall mainly in the planes", describes how random numbers gather into $(n-1)$-dimensional hyperplanes in n-space. Good RNG's either maximise the number of planes that are constructed to give the illusion of randomness or practically eliminate this artefact entirely. One must be aware of this behaviour in case anomalies do occur. In some cases, despite the shortcoming of RNG's, no anomalies are detected. An example of this is the same data that produced the obvious artefact in Figure 3.1 but displayed with a 10° rotation about the z-axis does not exhibit the artefact. This is shown in Figure 3.2.

3.1 Linear congruential random number generators

Most computer architectures support 32-bit 2's-complement integer arithmetic[1]. The following equation describes a linear congruential random number generator (LCRNG) suitable for machines that employ 2's-complement integer arithmetic:

$$X_{n+1} = \mathrm{mod}(aX_n + c, 2^{32}) \ . \tag{3.1}$$

This LCRNG generates a 32-bit string of random bits X_{n+1} from another representation one step earlier in the cycle, X_n. Upon multiplication or addition, the high-order bits (greater than position 32) are simply lost leaving the low-order bits scrambled in a pseudo-random

[1] In 32-bit 2's-complement integer arithmetic
$00000000000000000000000000000000 = 0$
$00000000000000000000000000000001 = 1$
$00000000000000000000000000000010 = 2 \cdots$
$01111111111111111111111111111111 = 2^{31} - 1 = 2147483647$
$10000000000000000000000000000000 = -2^{31} = -2147483648$
$10000000000000000000000000000001 = -2^{31} + 1 = -2147483647$
$10000000000000000000000000000010 = -2^{31} + 2 = -2147483646 \cdots$
$11111111111111111111111111111111 = -1.$

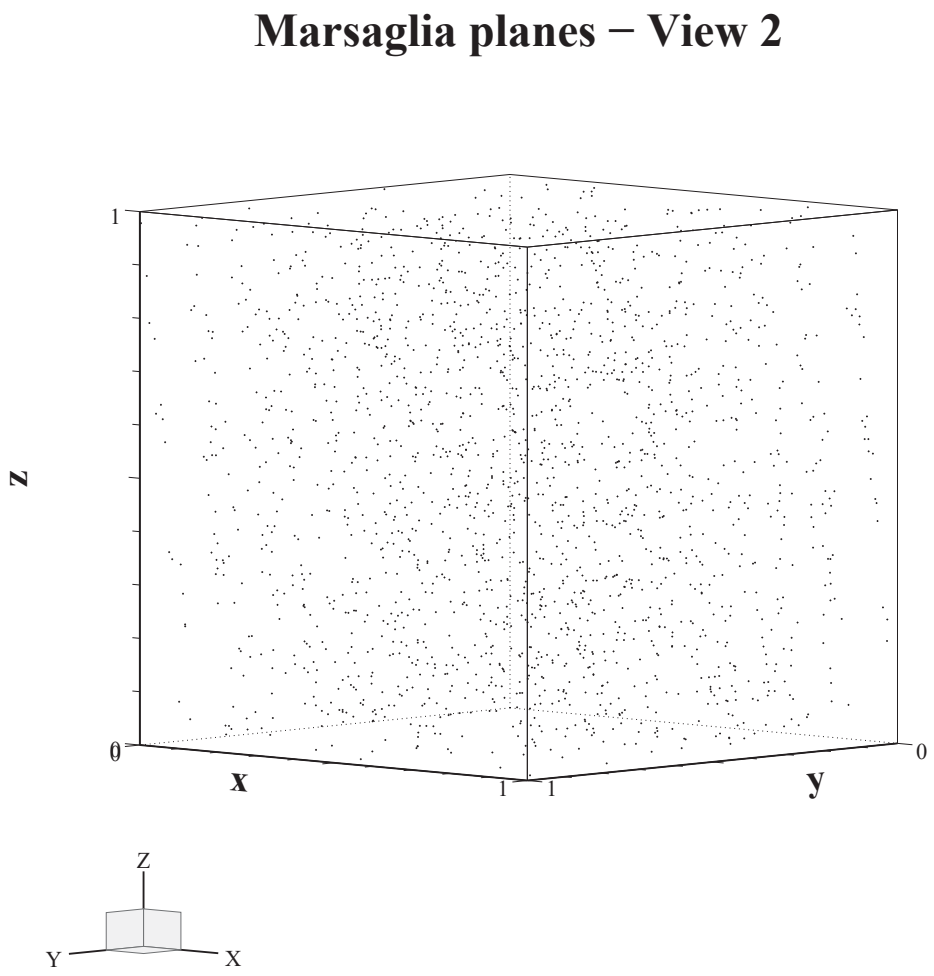


Figure 3.2: The identical data in Figure 3.1 but rotated by 10° about the z-axis.

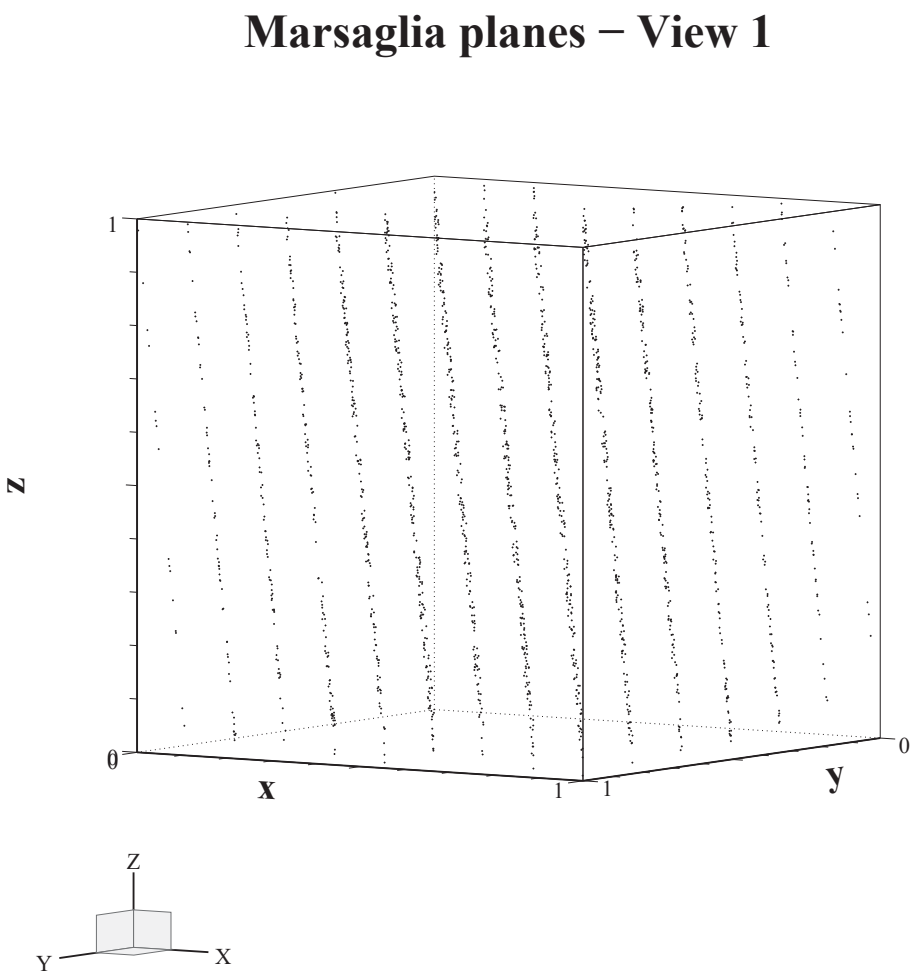


Figure 3.1: The gathering of random numbers into two-dimensional planes when a three-dimensional cube is seeded.

Chapter 6

Oddities: Random number and precision problems

Now that we understand about random number generators, sampling and error estimation, it is time for a brief respite to consider some of the oddities one might encounter during Monte Carlo calculations. These oddities are related to artefacts associated with random number generation and machine precision.

6.1 Random number artefacts

Consider the determination of the value of π one obtains by throwing random "darts" at a circle inscribed within a square. This is depicted in Figure 6.1. The ratio of the number of darts within the circle to the total number of darts within the square should be $\pi/4$.

For a small number of iterations, the result converges as expected. This is shown in Figure 6.2 where the ratio $4N_{\rm in}/(N\pi)$ is plotted up to 10^4 cycles along with the 1-σ error bars predicted by the "binary statistics" method. The estimated mean goes over and under the theoretical prediction, takes an excursion in the overprediction direction and eventually begins to settle down.

However, some difficulties are evident for large cycles as shown in Figures 6.3. and 6.4

There are some classic signals indicated in Figure 6.3 that the random number generator is cycling. The first piece of evidence is that the result exhibits a periodic structure. The random number generator employed in this study is a multiplicative congruential random number generator (MCRNG) with a sequence length of $2^{30} = 1,073,741,824$. Since two random numbers are employed, the periodic structure occurs over a period of $2^{29} = 536,870,912$. Another curious anomaly is that the result is close to unity (actually to within a few parts

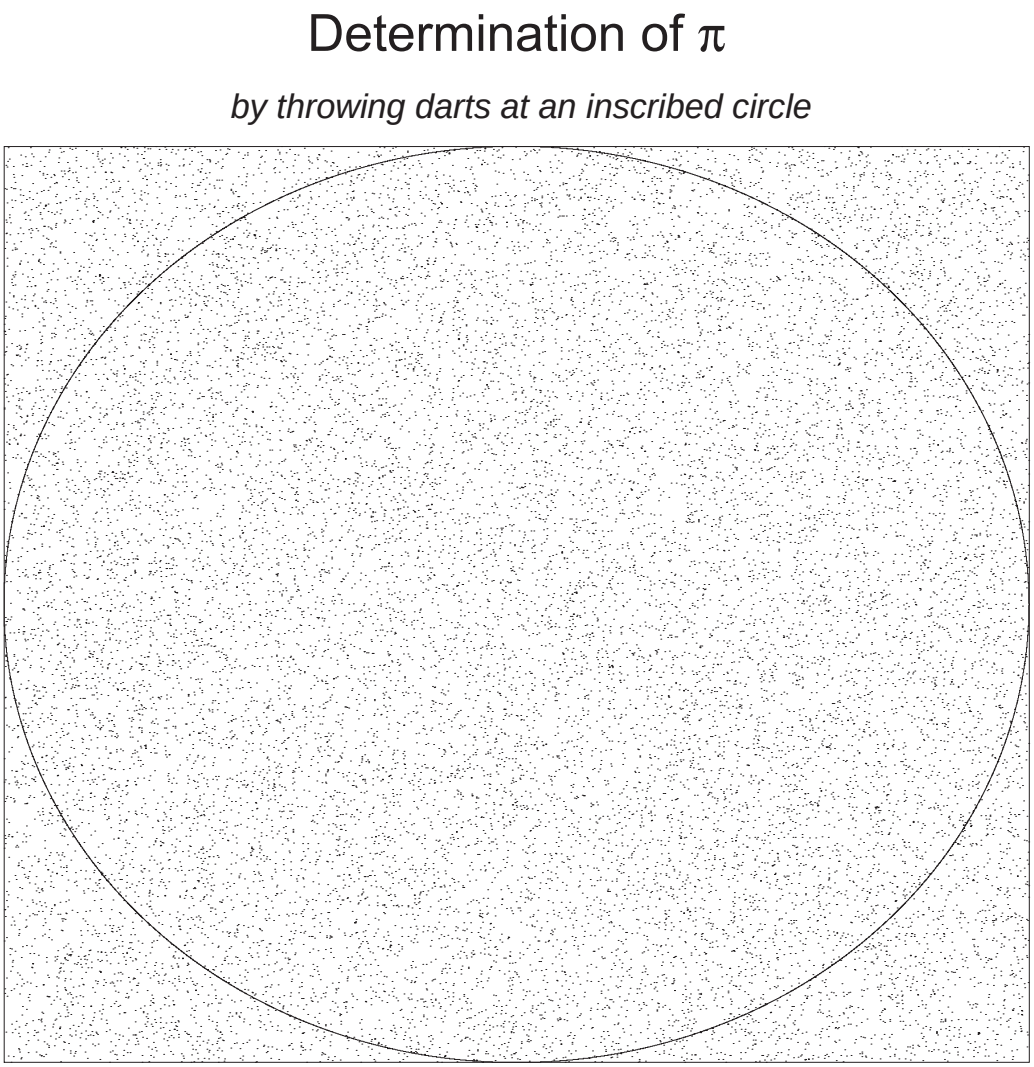


Figure 6.1: Random darts are thrown at a square with an inscribed circle. The ratio of the number of darts within the circle to the total number of darts within the square should be $\pi/4$.

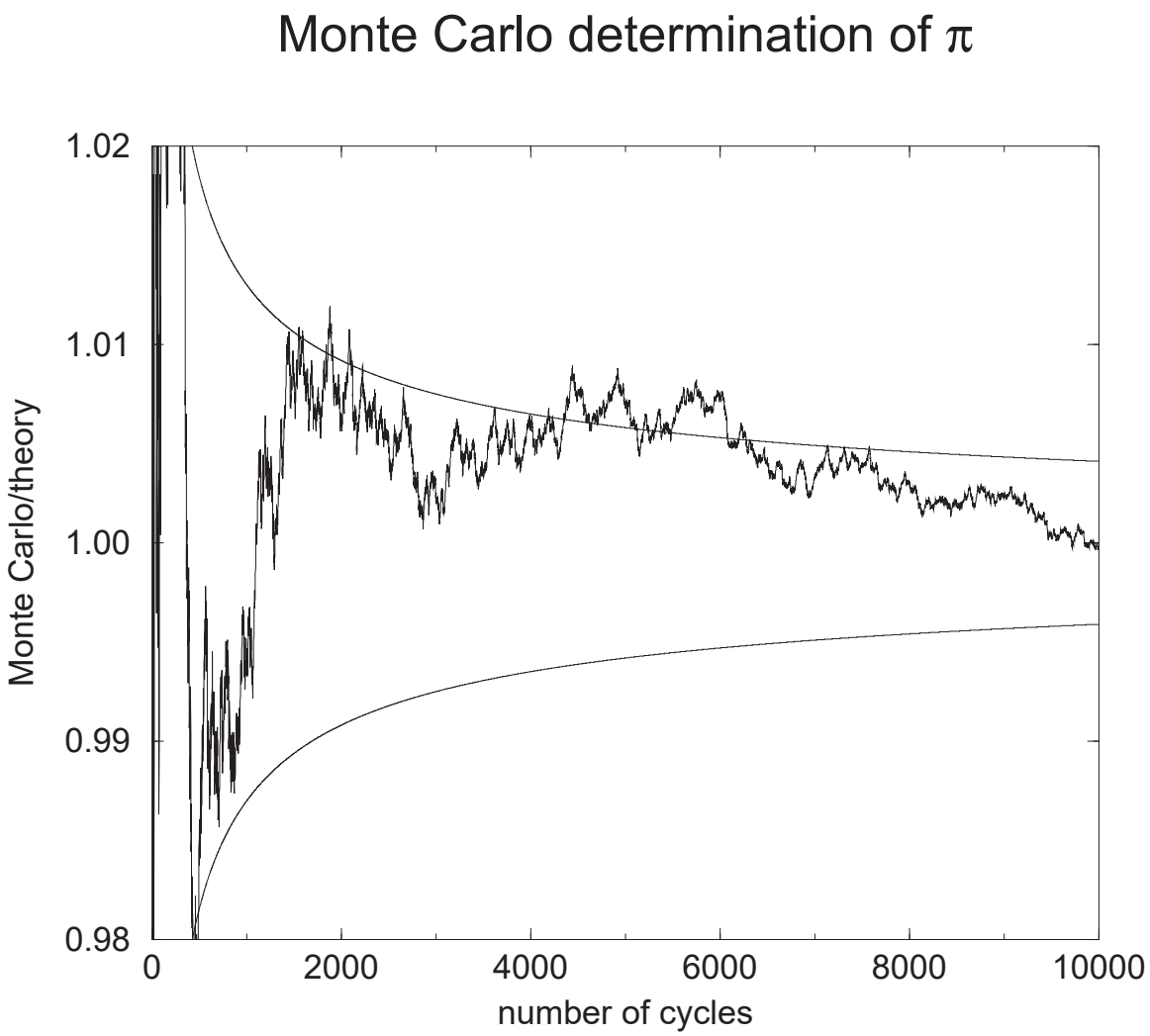


Figure 6.2: Random darts are thrown at a square with an inscribed circle. The Monte Carlo prediction divided by the theoretical prediction with the associated $1 \pm \sigma$ prediction.

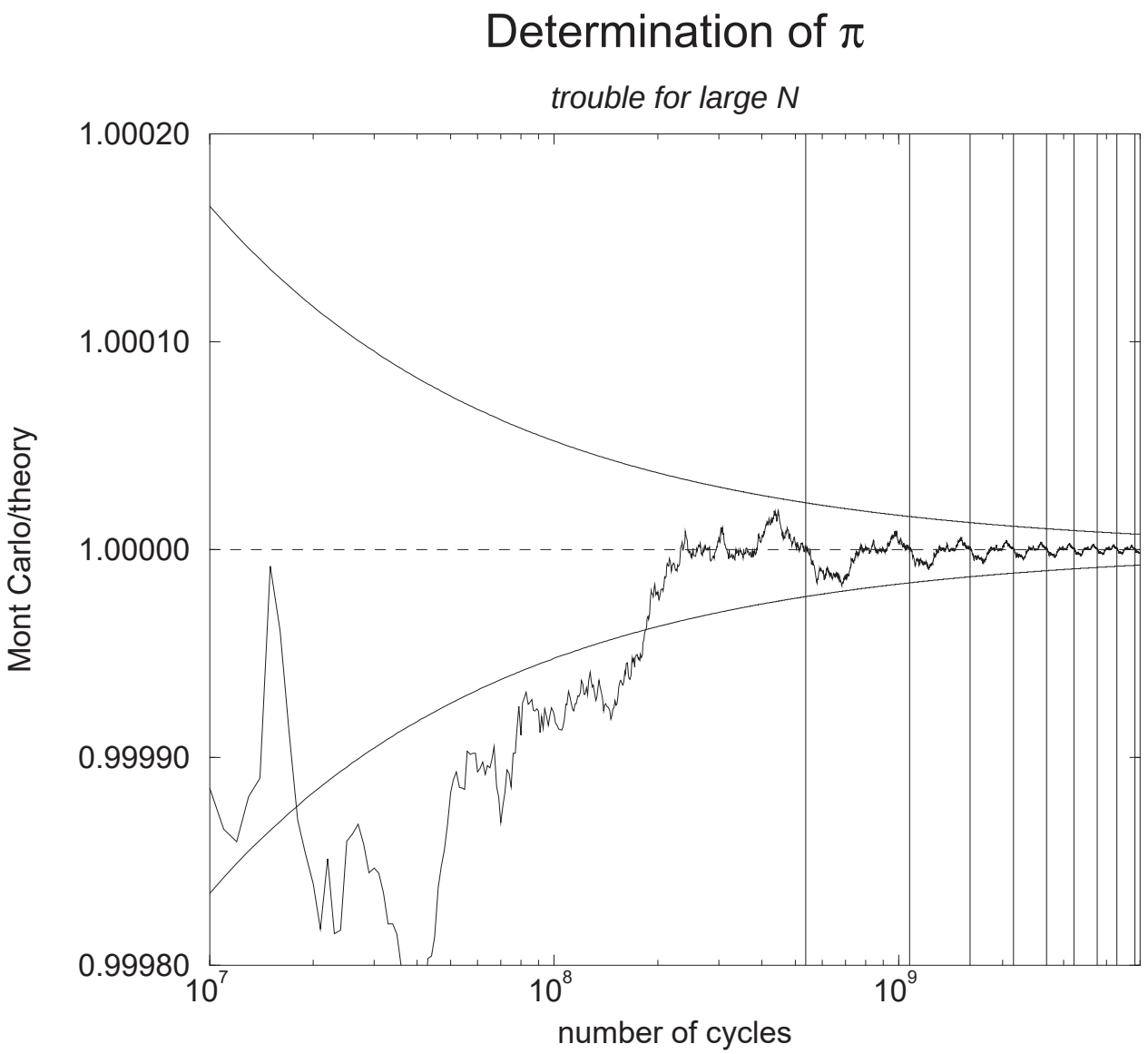


Figure 6.3: Random darts are thrown at a square with an inscribed circle. Large cycle behaviour of the Monte Carlo π experiment. The vertical lines are drawn where the random number generator begins a new cycle.

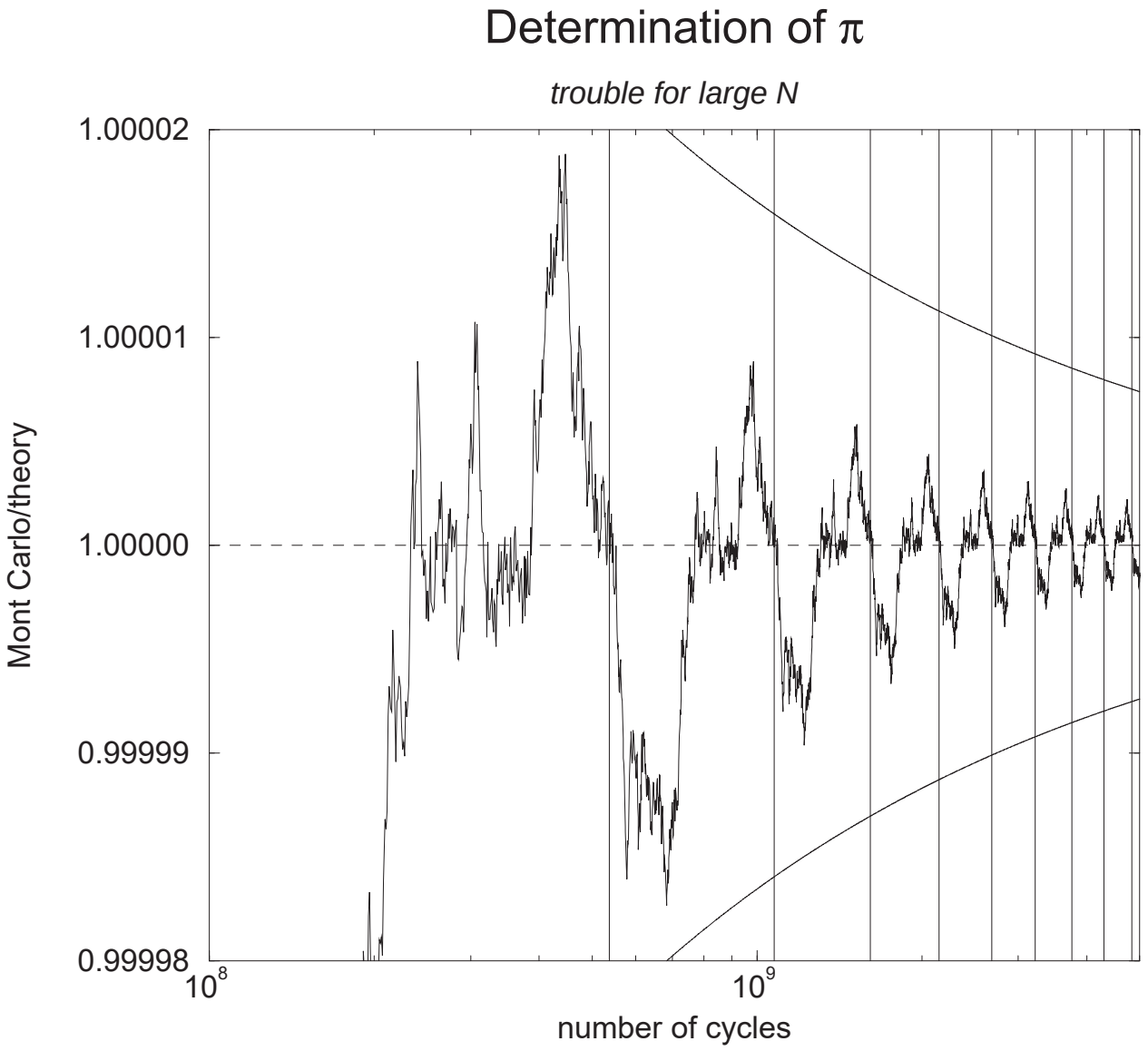


Figure 6.4: A zoom-in on the large cycle behaviour of previous figure.

in 10^7) at the point where the cycle restarts[1]. As a result, the calculated value appears to be well below the 1σ bounds predicted by the Central Limit theorem in the range shown. These are all strong signals that the random number has been "looped". It is never wise to use more than a fraction, say $1/10^{\text{th}}$ of the sequence. Note that the latter half of the sequence anti-correlates with the first half. This could lead to spurious results if a sequence is exhausted.

It is also false to conclude: *"Despite the periodic structure, the result converges to the correct answer."* **Wrong!** We just happened to be lucky in this case! The result converged to about $1 + 5 \times 10^{-7}$ after one complete cycle, nearly the correct answer but not **the** correct answer. The "error term" after one cycle just happens to be very small for this application. If we ran this application for about 12×10^9 cycles, we would note a "false convergence" to $1+5\times10^{-7}$ whereas the 1σ bounds would be smaller and converging on unity.

An example of "false convergence" is given in Figure 6.5 which is the same example except that a large number of random numbers were thrown away after each sample of π, as if to simulate many random numbers being employed in a different aspect of a calculation. Although the example is somewhat extreme, it depicts clearly an anomalous result that will never converge to the correct answer.

The object of this lesson is to warn against using random number generators beyond a fraction of their sequence length.

The signals that you have cycled the random number generator are:

- The tally exhibits a period structure.
- The tally converges in a way that is contrary to Central Limit predictions, assuming that the second moment of the tally exists.
- The presence of false convergence, which may be very difficult to detect.

[1]This is due to 2D space being nearly uniformly filled by this MCRNG.

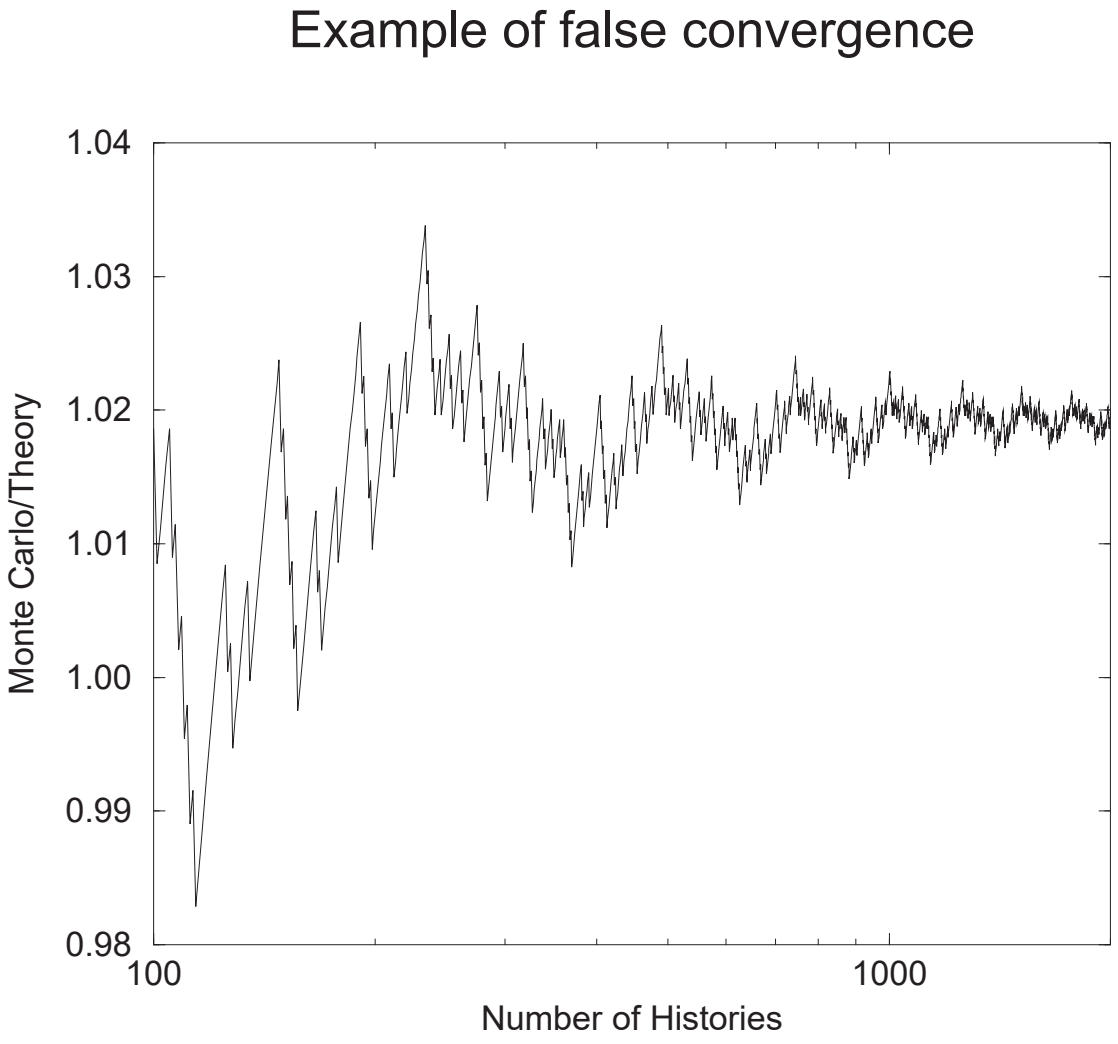


Figure 6.5: An example of false convergence.

6.2 Accumulation errors

Consider the summation:

$$s = \sum_{i=1}^{N} (1/N) \ . \tag{6.1}$$

Of course, mathematically the result is $s \equiv 1$. Numerically, however, it is a different story. The result of s *vs.* N is give in Figure 6.6.

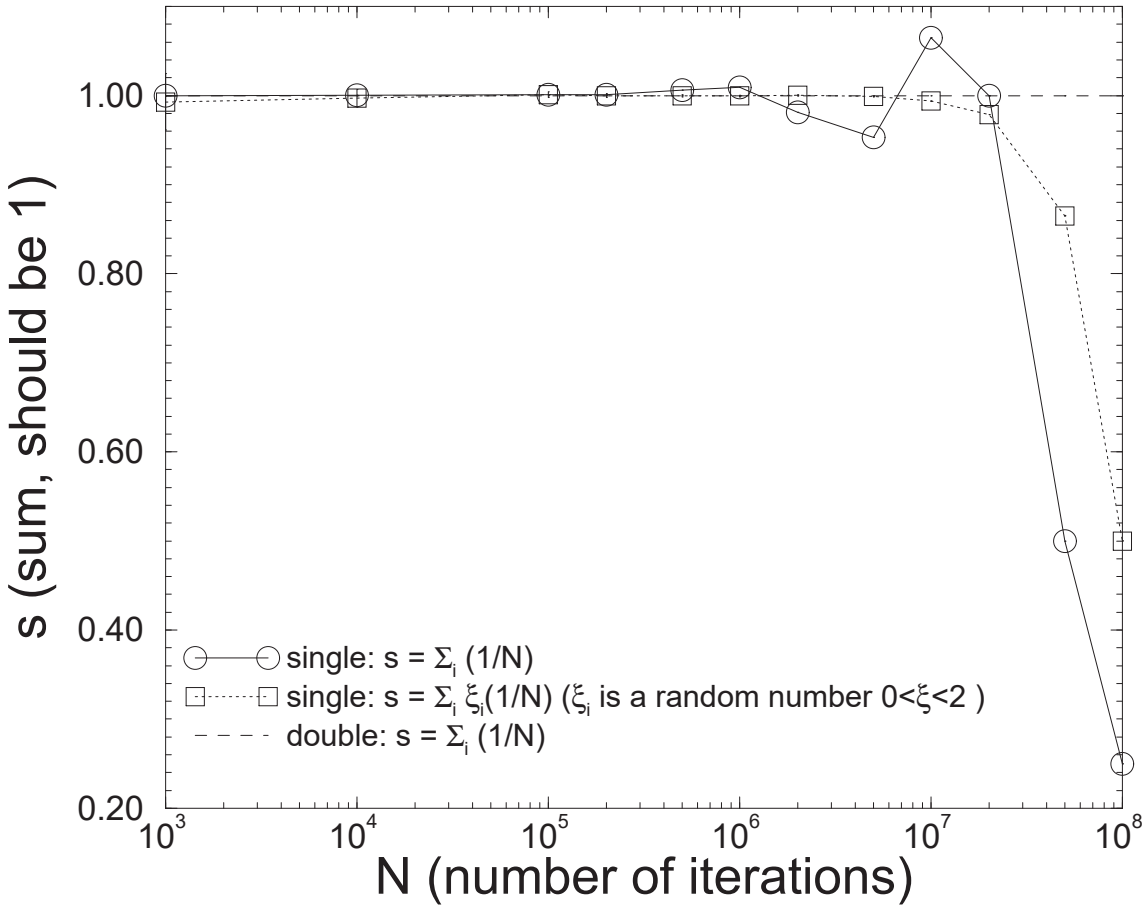


Figure 6.6: An example of constant and random accumulation errors in single-precision arithmetic.

We note that an accumumulation error is seen starting from about 10^6 iterations when the

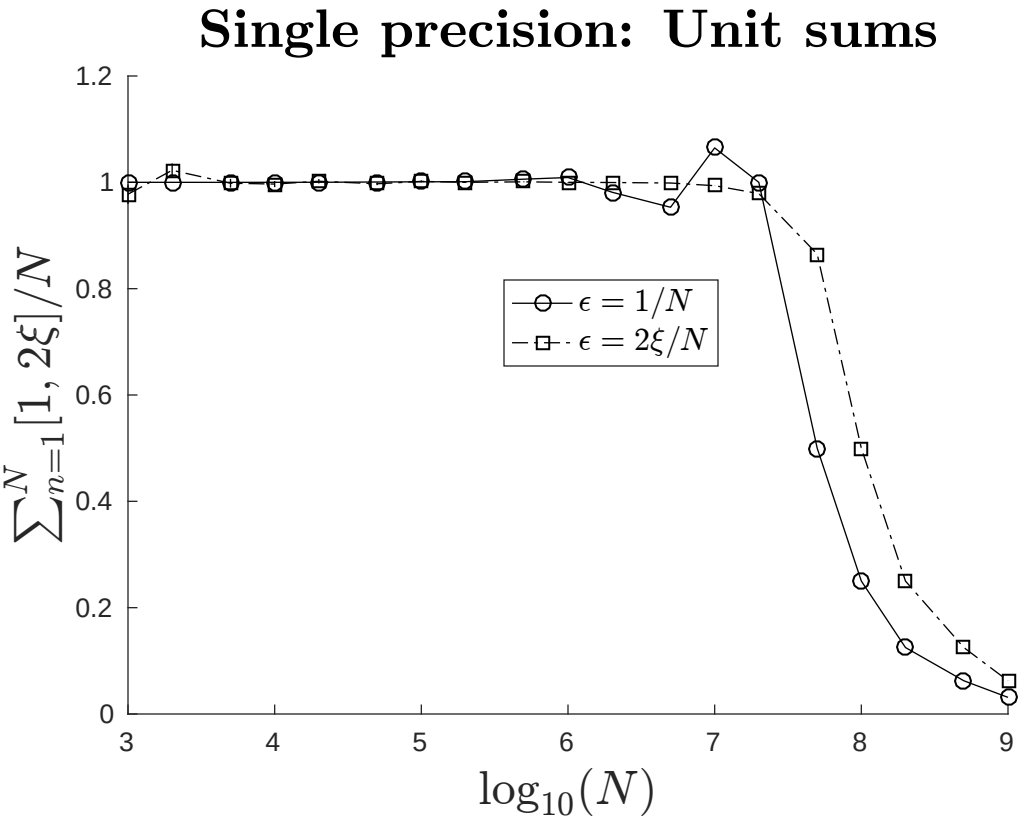
accumulation is done using single precision Fortran, a 32-bit representation of floating-point numbers. The shape of this curve is the result of constant accumulated round-off error, can be positive or negative, but eventually underestimates due to truncation error. The precise shape of the artefact is probably machine dependent. The reason for the underestimate at large value of N is because $1 + 10^{-8} \equiv 1$ in single precision arithmetic.

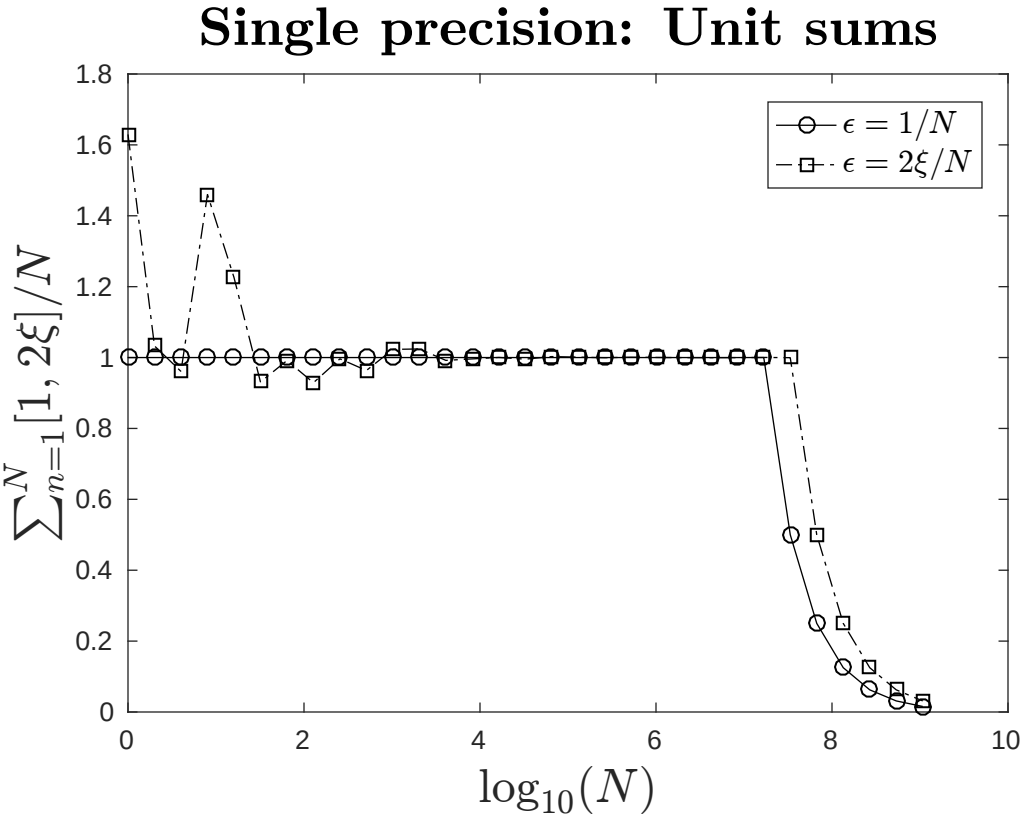
Another expression of the similar thing is:

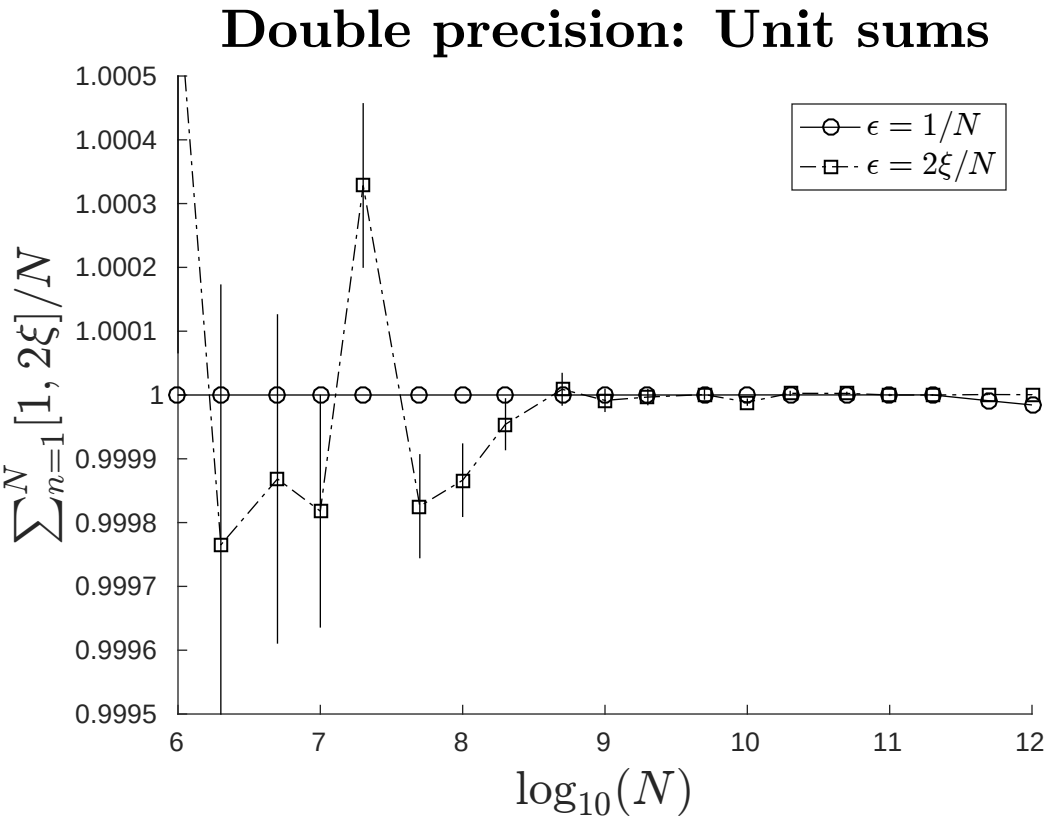
$$s = \sum_{i=1}^{N} (2r/N) \ . \tag{6.2}$$

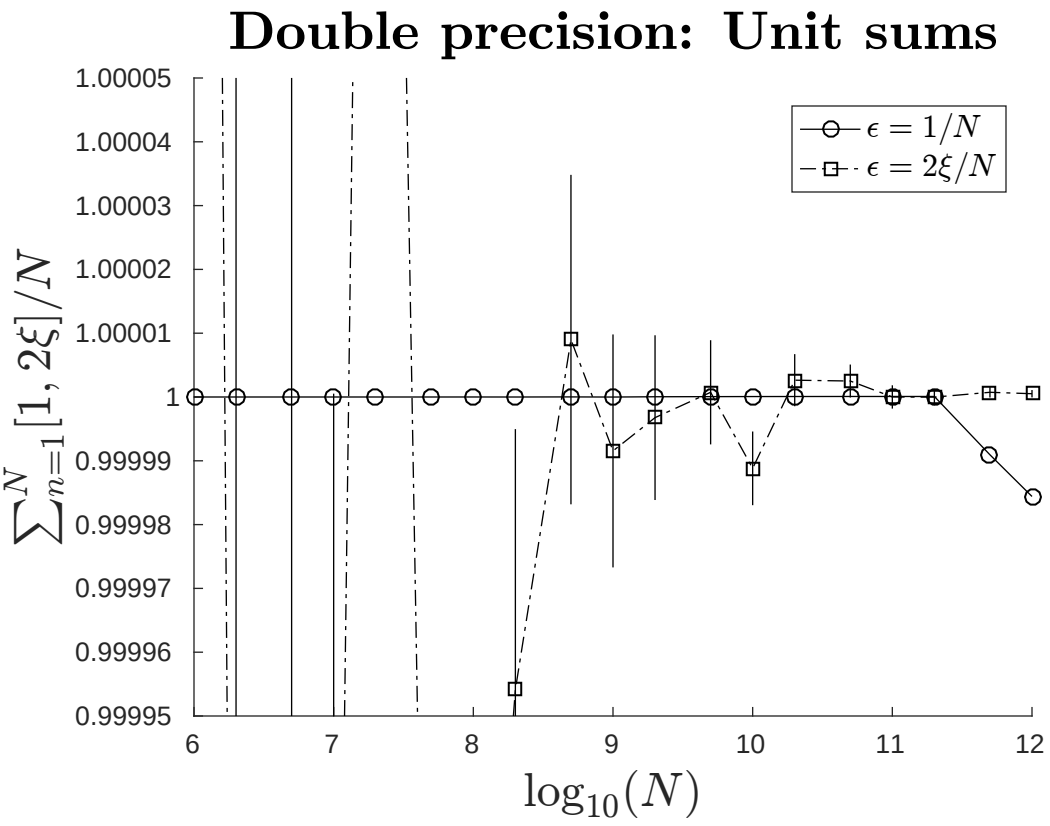
where r is a random number uniformly distributed on $[0, 1]$. Since $\langle r \rangle = 1/2$, $s \equiv 1$ mathematically as well. However, a numerical evaulation exhibits some accumulation error starting from about 10^7 iterations. This is also seen in Figure 6.6. The shape of this curve is the result of random accumulation error and is probably common to all single-precision architectures. A double precision accumulation is shown as well. For double precision, $1 + 10^{-8} = 1.00000001$ and no accumulation error is evident. Double precision errors would start at about 10^{15} to 10^{16} iterations, a realm where no application has dared to go (yet).

The obvious solution to this problem is: Use double precision! However, there are good reasons for using single precison numbers. On some architectures, single precision arithmetic is faster than double precision. (There are counter examples to this as well!) Double precision numbers also take more computer storage. Fetching and storing them can take longer than for single precision numbers. A good rule is to develop your application in double precision. Then, if fast exacution or computer storage become critical to your application, consider single precision for some, if not all of your calculation. However, you must be aware of the shortcomings (pun intended!) of single precision variables and use them with caution.


Single precision: Unit sums
1.2
1
0.8
0.6
0.4
0.2
0
3
4
5
6
7
8
9
$\sum_{n=1}^{N}[1, 2\xi]/N$
$\log_{10}(N)$
$\epsilon = 1/N$
$\epsilon = 2\xi/N$


Single precision: Unit sums
$\epsilon = 1/N$
$\epsilon = 2\xi/N$
$\sum_{n=1}^{N}[1, 2\xi]/N$
$\log_{10}(N)$
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
2
4
6
8
10


Double precision: Unit sums
$\epsilon = 1/N$
$\epsilon = 2\xi/N$
$\sum_{n=1}^{N}[1, 2\xi]/N$
1.0005
1.0004
1.0003
1.0002
1.0001
1
0.9999
0.9998
0.9997
0.9996
0.9995
6
7
8
9
10
11
12
$\log_{10}(N)$


Double precision: Unit sums
1.00005
1.00004
1.00003
1.00002
1.00001
1
0.99999
0.99998
0.99997
0.99996
0.99995
$\sum_{n=1}^{N}[1, 2\xi]/N$
6
7
8
9
10
11
12
$\log_{10}(N)$
$\epsilon = 1/N$
$\epsilon = 2\xi/N$