CHAPTER 1 A Monte-Carlo Calculation of Pi

1.1 Experimental Basis - "Tossing Toothpicks"

Monte-Carlo(MC) techniques are numerical algorithms that utilize (pseudo) random numbers to perform mathematical calculations and to model physical systems or simulate an experimental procedure. In the present case the experiment is a rather simple one which itself uses a stochastic process- one that has an inherent element of randomness in it - to obtain an estimate for π. The experimental procedure is as follows:

- On a large piece of styrofoam board draw parallel lines that are separated by a distance equal to the length of a standard (round) toothpick.
- "sprinkle" a large number of toothpicks onto the board, N.
- Record the number of toothpicks that cross any of the parallel lines, N_c.
- Calculate an estimation for π using the relationship., $\pi \simeq \frac{2N}{N_c}$

The outcome of such experiments is usually quite good, providing values of π with an error of approximately 5%. In figure 1.1 for example, 175 toothpicks were used and 107 crossed a line, yielding a value of 3.271 for π. This represents an error of only 4.1%! The nature of the expected error will be discussed at greater length below. For now it should be clear that as the number of toothpicks used is increased, the error achieved should decrease.

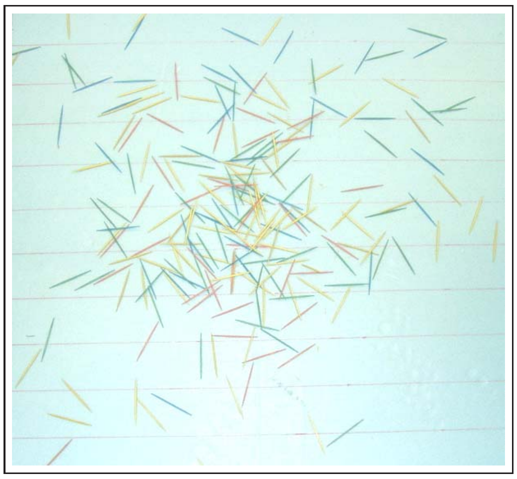


Figure 1.1: The value of π may be approximated quite well using statistically based methods. Here $\pi \simeq \frac{2N}{N_c}$, where N is the total number of toothpicks and N_c is the number crossing any of the *parallel* lines.

1.2 An Analytic Model

The following interpretation of the experiment is attributed to Hans Betthe[1] and assumes that the distribution of the toothpicks is random. This being the case one may assume that the angle that any given toothpick makes with any of the lines is also random. For simplicity imagine a toothpick that just touches a line as shown below.

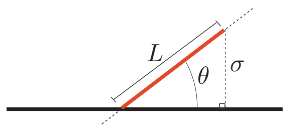


Figure 1.2: The distribution of the toothpicks may be described in terms of the perpendicular distance from the line, $\sigma = L\sin\theta$.

The distribution of the toothpicks may be described in terms of the perpendicular distance from the line, $\sigma = L\sin\theta$, and the angle θ may vary on the closed interval $[0, \pi]$. Hence the total effective "area" of the distribution is given by,

$$A = L\pi. \tag{1.1}$$

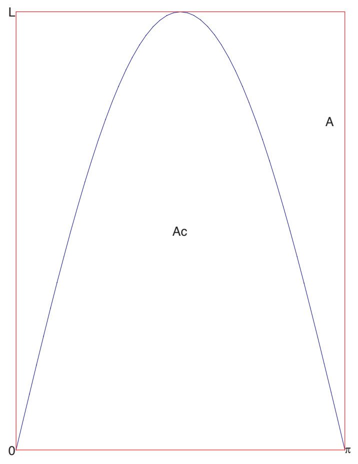
The total "crossing area" due to the angular distribution is simply,

$$\begin{aligned} A_c &= \int \sigma(L,\theta)d\theta \\ &= L\int_0^{\pi} \sin\theta \; d\theta \; . \\ &= -L\cos\theta\Big|_0^{\pi} = 2L \end{aligned} \tag{1.2}$$

These areas are more mathematical than physical, but may be displayed graphically for illustration purposes. To do this use `MAPLE` code similar to the following:

```
with(plots):
with(plottools):
bplot:=plot([[0,0],[0,1],[Pi,1],[Pi,0],[0,0]]):
cplot:=plot(1*sin(theta),theta=0..Pi,color=blue):
tplot:=PLOT(TEXT([evalf(Pi),0],"p",ALIGNRIGHT,FONT(SYMBOL)),
              TEXT([0,0],"0",ALIGNLEFT),
              TEXT([0,1],"L",ALIGNLEFT),
              TEXT([evalf(Pi)/2,1/2],"Ac"),
              TEXT([3,3/4],"A"),
              AXESSTYLE(NONE)):
display(bplot,cplot,tplot);
```

[1]One of the fathers of the atomic bomb.



The ratio of the total area to the crossing area is then shown to be proportional to π. I. e.,

$$\frac{A}{A_c} = \frac{L\pi}{2L} = \frac{\pi}{2} \tag{1.3}$$

The argument then becomes that if the distribution is indeed random, the ratio of the number of toothpicks that cross a line to the total number of toothpicks should be proportional to the ratio of the respective distribution areas. That is, $\frac{N}{N_c} \propto \frac{A}{A_c}$ so that one may write,

$$\boxed{\pi \simeq \frac{2N}{N_c}}$$

1.3 Computer Simulation - “Throwing Stones”

1.3.1 The Idea

In order to simulate the statistical calculation of π accomplished by tossing toothpicks a large amount of time might be devoted to thinking about how to provide a graphical representation of the situation. Instead the idea employed here is to utilize the same basic concept in a way that side steps some of the complications that might arise. The method used here is adapted from [**?**], and utilizes the distribution of random points over a unit circle inscribed by a square to determine a value for π. A typical run is shown in figure 1.3 below.

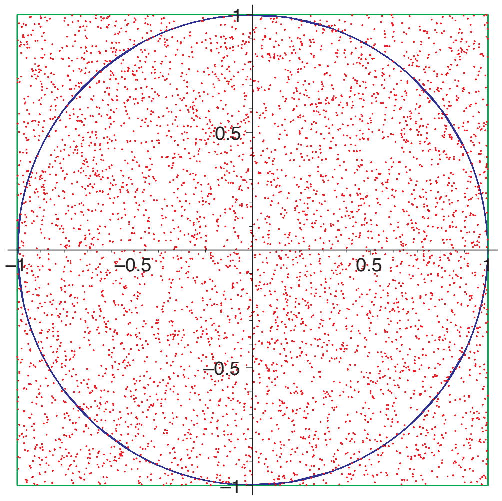


Figure 1.3: By determining the number of ordered pairs that lie inside the unit circle one may estimate the value of π. Here 5000 points are shown yielding a value for of 3.1504.

The ratio of the area of the inscribed circle to that of the square is simply, $\frac{\pi(1)^2}{2^2} = \pi/4$. If one then assumes that the pairs are uniformly distributed, then this ratio should be well approximated by the ratio of the number of pairs that “fall” within the circle to the total number. I. e.,

$$\boxed{\pi \simeq \frac{4N_c}{N}}$$

With the concept established one may design an algoritmic solution as follows:

- Generate a `N` pairs of random numbers drawn from the interval $[-1, 1]$.
- Determine the number, `Nc` of pairs that fall inside of the unit circle.
- Calculate an estimate for π.
- Calculate the error associated with the distribution.

This approach may be implemented in a straightforward manner. For example, one may implement it in FORTRAN using a simple `DO-Loop` construct as the following code fragment illustrates.[2]

```
...
DO i = 1, N
   x = Rand()*2-1
   y = Rand()*2-1
   WRITE(50,*) x, y
   IF ((x*x + y*y) .LE. 1.0) THEN
      Nc = Nc + 1
   ELSE
      Ns = Ns + 1
   END IF
   est = 4.0*Nc/(Nc+Ns)
   sum = sum + est
   mu = sum/(Nc+Ns)
```

[2]The complete program listing may be found at the end of the chapter.

```
         var = (var + (mu-est)**2)/(Nc+Ns)
         dev = SQRT(var)
         ...
      END DO
      ...
```

Typical output from this implementation is shown in figure 1.4.

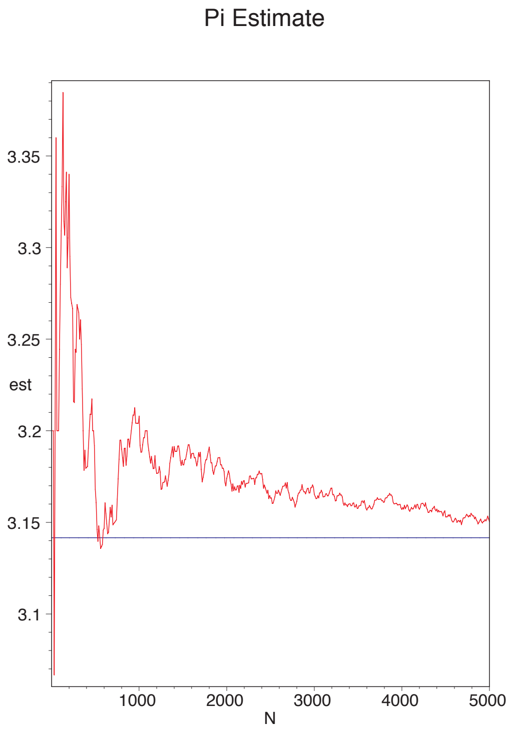


Figure 1.4: Here 5000 random points are used to obtain an estimate for π. The statistical distribution of values has a deviation of less than one part in a thousand.

Notice that the estimates vary greatly for small numbers but converge rapidly. Furthermore, some of the values reached early on(at 550 the value is 3.1418) during this run seem better than those achieved later (at 5000 the value was 3.1503). This is due to the statistical fluctuation of the actual data generated during a particular run. In the present case the accepted value of pi could of course be used to "break out" of the loop when ever the desired accuracy was reached, but more generally the value sought in a calculation is not known and a statistical treatment of the error is most appropriate.

1.3.2 The Nature of Statistical Error

The traditional measure of the error associated with a statistical distribution of data is the standard deviation, σ. Recall that the variance of a discrete distribution is given by,

$$variance \equiv \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(\mu - x_i)^2, \tag{1.4}$$

where $\mu \equiv \frac{1}{N}\sum_{i=1}^{N} x_i$ is the average of the distribution. Hence one may write,

$$\boxed{\sigma \propto \frac{1}{\sqrt{N}}}$$

for the (standard) deviation. In other words, the quantity $\sqrt{N}\sigma$ should be proportional to a constant. What this means is that for large numbers, and for "well behaved" distributions, a plot of the σ versus $1/\sqrt{N}$ should approach that of a straight line. This behavior can be seen in figure 1.5 shown below.(Note that the region of interest is that approaching the origin, where N is large.)

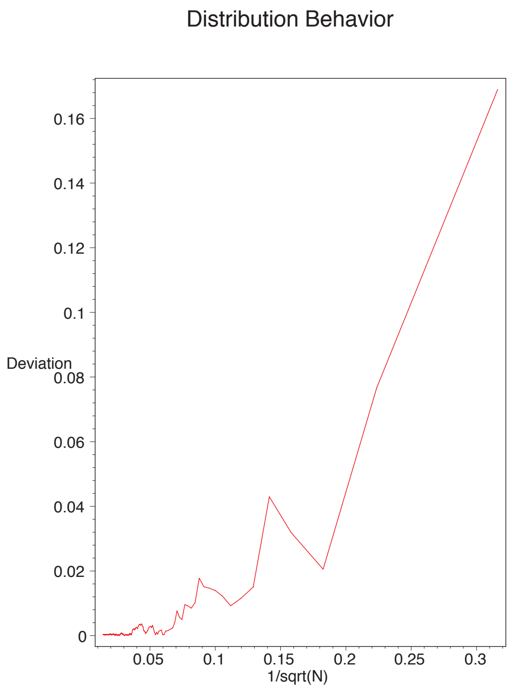


Figure 1.5: The standard deviation of a well behaved statistical distribution should approach a straight line for large values.

1.3.3 Pseudo-Random Numbers

The only processes that are truly random are those that occur in nature. A naturally occurring process that has an element of randomness in it is often referred to as a *stochastic process*. In order to model such phenomena it is necessary to generate numbers that are good approximations to truly random numbers. Numbers generated in this fashion are said to *pseudo-random*. This is because all (mathematical) techniques of generating "numbers at random" are inherently algorithmic. That is, one normally chooses a "seed" value and then applies some operation (such as modulo arithmetic) and then operates on the result.

The Uniform Distribution

The most obvious way of assessing whether or not a particular set of numbers is (nearly) random is to construct a histogram showing the possible outcomes along with there respective probabilities. As an example, consider tossing a single die. If the die is not weighted, then each face has a equal probability $(\frac{1}{6})$ of coming up. **If the possible outcomes of a random process all occur with an equal probability the distribution is said to be uniform.** To see if this indeed the case, one may roll the die a large number of times and see if this hypothesis holds true. In figure 1.6 below, one sees that in a particular run a single die was thrown one-thousand times. While the probabilities of each occurrence $(1 \ldots 6)$ are not exactly the same, they are similar to one another and the distribution of outcomes does not display any obvious signs of weighting.

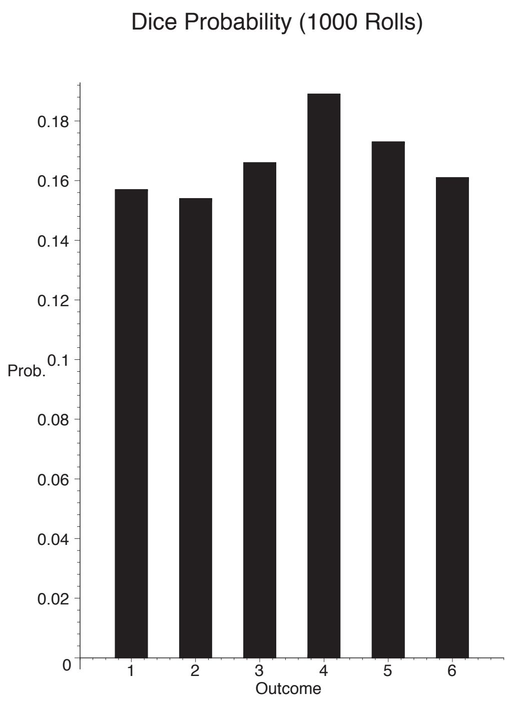


Figure 1.6: The distribution of outcomes for tossing a single (fair) die show that the outcomes are nearly equal in probability- indicating a uniform distribution.

Most modern programming languages include a built-in random number generator, such as FORTRAN's Rand() function shown in the code fragment above. This is true of `MAPLE` as well. To generate a (uniformly distributed) random number on the interval $[a, b]$ a `MAPLE` user would call the **rand** procedure in the following manner:

```
a:=0: b:=10:
r:=rand(a..b):
r();
r();
```

$$9$$
$$4$$

Note that **calls to the rand function must be made with an integer range specification.** This does not mean that other real (or complex) numbers can not be generated however. For instance to generate random numbers on the closed interval, $[-1, 1]$ one might use code similar to the following:

```
Digits:=30:
s:=10^(Digits):
unit_rand:=rand(-s..s)/s:
unit_rand();
unit_rand();
```

$$.358391135537108795991637819437$$
$$-.125690257132768639497457917923$$

Other Useful Distributions

While many stochastic processes are best modeled using uniformly distributed random numbers, it is sometimes wise to "pull" the numbers from a distribution which reflects a physically significant distribution. In this case, one acknowledges from the beginning that the probability of each outcome may not be similar, and is hoping to weight them in a way. **When the possible outcomes of a random process occur with an unequal probability the distribution is said to be nonuniform.**

Some of the most important distribution functions used in the physical sciences are,

$$\begin{aligned}
\text{Binomial}: \quad P_B(x) &= \frac{n!}{x!(n-x)!}p^x(1-p)^{n-x} && (1.5a)\\
\text{Poissonian}: \quad P_P(x) &= \frac{\mu^x e^{-\mu}}{x!} && (1.5b)\\
\text{Gaussian}: \quad P_G(x) &= \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}} && (1.5c)\\
\text{Lorentzian}: \quad P_L(x) &= \frac{1}{\pi}\frac{\Gamma/2}{(x-\mu)^2+(\Gamma/2)^2} && (1.5d)
\end{aligned}$$

The *Binomial* and *Poissonian* distributions are *discrete* distributions. This is because for the Binomial distribution, n is an integer and p is a real number such that $0 < p \leq 1$. For the Poissonian distribution the average value, μ is given by, $\mu = np$. While the Binomial distribution may be most appropriate for calculating permutational probabilities, the Poissonian distribution may be used to model discrete stochastic phenomena such as nuclear decay and digital packet switching. The parameter, σ indicates the (standard) deviation and $\Gamma = 2.354\sigma$ is the full-width at half-maximum(FWHM) value. The intensity profiles of laser light and the (ideal) distribution of grades, for instance, obey Gaussian statistics and the responses of many optical devices are well modeled with a Lorentzian using a Cauchy distribution function.[3]

In figure 1.7 each of the distributions given in equations (1.5) have been plotted for comparison. Note that while each is plotted over the same range, the Binomial and Poissonian functions have been plotted using points to remind the reader that they are in fact discrete distributions. The parameters, $n = 2$, and $p = \frac{1}{4}$ were used. Observe that the Binomial and Poissonian distributions are not symmetric about the mean value, $\mu = \frac{1}{2}$, but both the Gaussian and Lorentzian distributions are.

[3]The general Cauchy distribution function is given by, $P(x) = \frac{1}{\pi}\frac{b}{(x-\mu)^2+b^2}$.

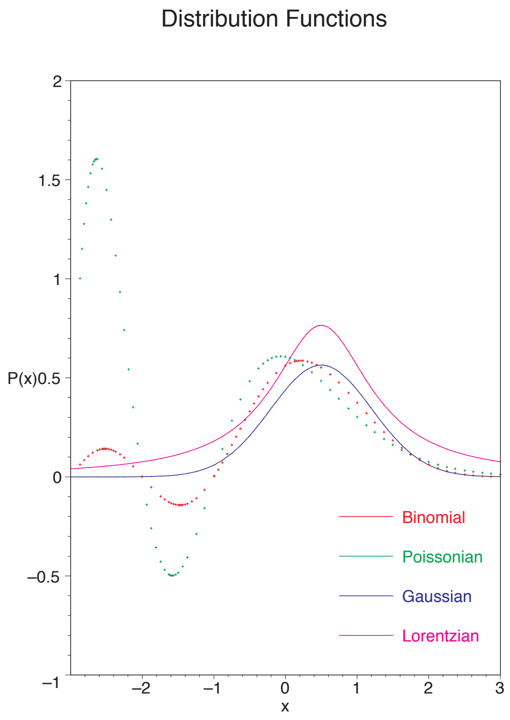


Figure 1.7: When modeling stochastic processes it is often necessary to draw any (pseudo)random numbers being used from a nonuniform distribution function which may have some physical significance.

These distributions are all available using `MAPLE` 's statistics package.[4] As an example of how one might generate a sequence of numbers drawn from each of them, consider the following `MAPLE` session:

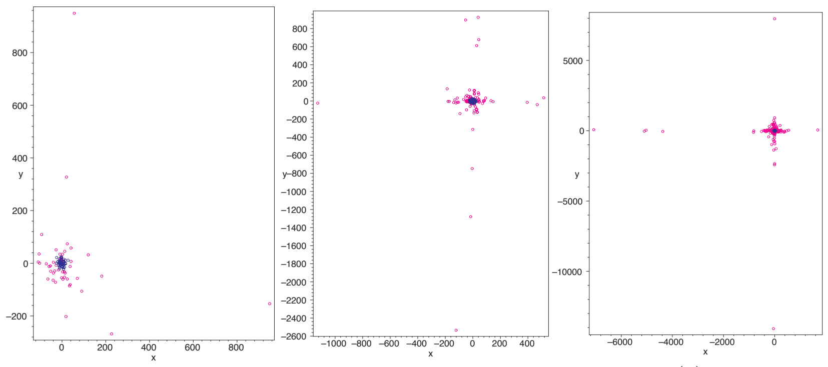
```
with(stats):
Digits:=4:
n:=2: mu:=1/2: sig:=1/sqrt(2): gam:=evalf(2.354*sig):
Binomial:=random[binomiald[n,mu]]:
Binomial(5);
Poissonian:=random[poisson[mu]]:
Poissonian(5);
Gaussian:=random[normald[mu,sig]]:
Gaussian(5);
Lorentzian:=random[cauchy[mu,gam]]:
Lorentzian(5);
```

$$0, 1.0, 2.0, 0, 2.0$$
$$0, 2.0, 0, 1.0, 0$$
$$.3253, 1.632, .8747, .9528, -.0221$$
$$.5822, 1.051, 3.302, 6.968, 2.602$$

To understand why one may wish to draw random numbers from a particular distribution, consider that the Lorentzian distribution is said to "heavy" tails. This means that the likelihood of choosing a number at random that is far from the mean of the distribution is

[4]For a complete list of the distributions in the statistics package type `?stats[distribution]` at the command prompt.

greater than when using the Gaussian distribution.



(a) (b) (c)

Figure 1.8: (a) 50, (b) 100, and (c) 500, random pairs have been drawn from Gaussian (blue) and Lorentzian (magenta) distribution functions. Here $\mu = 0$ and $\sigma = 10$.

This effect may be quite dramatic. Observe that in figure 1.8 the "spread" of random points drawn from a Lorentzian distribution increases dramatically as the number of points is increased. These plots were obtained with the following code,

```
with(plots):
with(stats):
mu:=0: sig:=10: gam:= 2.354*sig:
Grand:=random[normald[mu,sig]]: Lrand:=random[cauchy[mu,gam]]:
T:=500: for i from 1 to T do
    G[i]:=[Grand(),Grand()]:
    L[i]:=[Lrand(),Lrand()]:
od:
Gseq:=[seq(G[i],i=1..T)]:
Lseq:=[seq(L[i],i=1..T)]:
Gplot:=plot(Gseq,style=point,color=blue,symbol=circle):
Lplot:=plot(Lseq,style=point,color=magenta,symbol=circle):
display(Gplot,Lplot,axes=boxed,labels=["x","y"]);
```

and are consistent with the assertion that for a normal (Gaussian) distribution, 95% of the datum should fall within one standard deviation of the mean, 98.5% should fall within two standard deviations of the mean, and 99.9% should fall within three standard deviations of the mean. The mean here is zero and a standard deviation of $\sigma = 10$ was assumed. The probability of occurences outside of this region is therefore very low. On the other hand, one sees that the probability of drawing datum far from the mean value is greatly increased when using a Lorentzian distribution.

The nonuniform nature of such distributions may be desirable in some applications, but not in others. In the present case, the choice of a uniform distribution seems most appropriate because, while the distribution of toothpicks in figure 1.1 is not perfectly uniform, they appear to be nearly so. In addition, the underlying assumption of the equivalence of the ratio of the areas of the square to the inscribed circle, to that of the ratio of the total number of toothpicks to the number crossing any line, is most simply made when assuming a uniform distribution.

QUIZ Write a `MAPLE` procedure that will generate N pairs of points that fall inside of a rectangle of width, a and height, b. For example, draw the *x-coordinates* values from $[0, a]$ and the *y-coordinates* values from $[0, b]$. The procedure should also draw a “box” (with a color of your choosing) around these points and label the corners, lower-left(“`LL`”), lower-right(“`LR`”), upper-right(“`UR`”), and upper-left(“`UL`”) using the plottools package to place the text. Call your procedure using values of $a = 10$, $b = 5$ and $N = 100$. Once you have obtained this “static” result, construct an animation sequence consisting of ten frames- showing succesively larger amounts of the total data set. Export your document in HTML format and publish it on your local webserver.

1.4 Project Assignment

As shown in figure 1.3 in section 1.3.1 the basic requirements for this simulation are the production of uniformly distributed random points in the region, $[-r_c, r_c]$, where r_c is the radius of a circle,[5] along with a method for determine where the points fall. With this a statistical value for π may be calculated. The principle goals of the simulation are to construct a `MAPLE` procedure that will produce an animation sequence which displays "frames" similar to figure 1.3. The number of points, `N`, used and the number of frames, `Nf`, should be among the parameters that are specified in the procedural definition. While particular implementations may vary, a simple method for creating the lists of points to be plotted in each frame is,

```
for i from 1 to N/Nf do
    L[i]:=[ seq( [r(),r()], j=1..i*Nf ) ]:
od:
```

where r is a random number on the appropriate interval. The number of points falling within the unit circle may be incremented using a simple if-then construct. (See the FORTRAN implementation in section 1.5 for ideas on how to implemnt this in `MAPLE` .)

In addition to the animation sequence, plots of the statistical value of π (as a function of the total number of points) should be plotted (c. f., figure 1.4 and a plot of the (standard) deviation (similar to that shown in figure 1.5) should be made. To produce these plots two lists of ordered pairs of points will have to be constructed. In the case of plotting the value of π, the current value of the number of points (I. e., a loop-indicator, i.) should be the first element of the pair and the calculated value (here referred to as '`est`') should be the second element. To accomplish this one may add code similar to the following to the body of the do-loop that is being used to calculate the estimate:

```
error[i]:=[i,est[i]]:
```

Once the loop is finished the points may be "zipped up" using a seq command, so that they may be plotted.

1.4.1 Extensions

A Three-Dimensional Implementation

The statistical method for estimating a value of π may be generalized using a three-dimensional implementation. Random points may be generated within a cube that inscribes a sphere and if a uniform distribution is used, the ratio of the volume of the sphere to that of the cube should be proportional to the ratio of the number of pairs falling within the sphere (N_s) to the total number. Rudimentary analysis shows that,

$$\boxed{\pi \propto \frac{6N_s}{N}}$$

A representation of such a simulation is illustrated in figure 1.9 below.

[5]The choice of $r_c = 1$ was arbitrary.

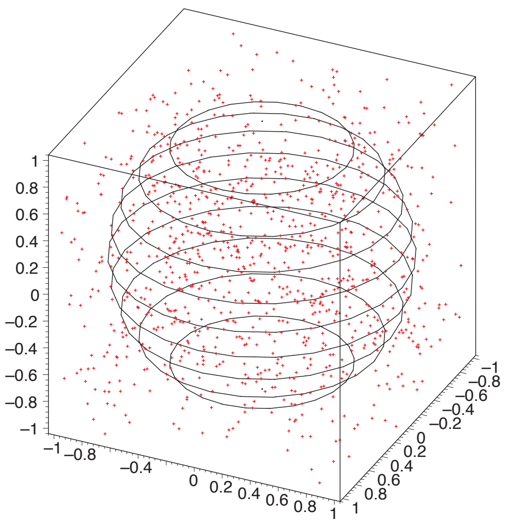
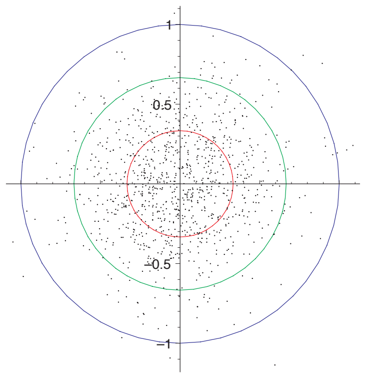


Figure 1.9: Here 1000 (uniform) random pairs have been plotted within a unit cube. Statistical arguments show that $\pi \simeq \frac{6N_s}{N}$, where N_s is the number of pairs falling within the unit sphere.

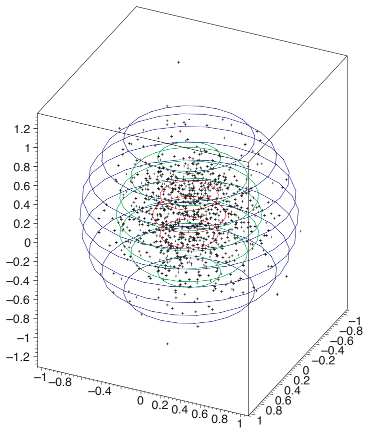
Modification of the procedural solution to the two-dimensional simulation should be carried out and an estimate for the value of π using (at least) 5000 pairs of numbers should be obtained. When the procedure is called, the number of points and frames should be variable and an animation sequence along with the (final) estimate for π and the standard deviation of the distribution should be output. Graphs of the successive values of these quantities are not required.

Using a Nonuniform Distribution

As mentioned at several points in the text, the statistical argument employed here requires that uniformly distributed random numbers be used. If this is not done the proportion of pairs falling inside or outside of the circle may be weighted unevenly. As an example of this, consider the cases illustrated in figure 1.10, where two- and three-dimensional distributions of random pairs drawn from a Gaussian (normal) distribution with a mean of zero and a standard deviation equal to one-third. These values were chosen so that nearly all of the data (99.9%) would fall within a unit circle and/or sphere. Note that in either case, pairs fall outside of these regions.



(a)



(b)

Figure 1.10: Here 1000 (random) pairs have been plotted in both (a) two-dimensions and (b) three-dimensions using a normal(Gaussian) distribution with $\mu = 0$ and $\sigma = \frac{1}{3}$. The circles(spheres) show the boundaries of the first(red), second(green), and third(blue) moments about the mean.

Consider whether or not one might "smooth out" the data so that a modified statistical argument might still be used to estimate the value of π. In forming your opinions consider that if 99.9% of the points fall within the unit circle, then 0.1% of the points fall within a rectangle, whose dimensions are determined by the coordinates of the "outlying pairs." This situation is shown in figure 1.11 below.

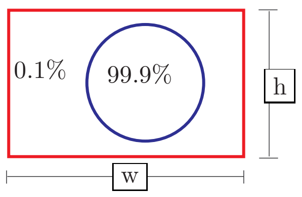


Figure 1.11: A modification of the statistical argument may be used to achieve an estimate for π using random pairs drawn from the nonuniform, Gaussian distribution.

If the numbers of points falling in each of these regions is weighted appropriately, a modified rule for calculating an estimate for π may be obtained. To determine the width and height of the rectangle, one may search the total list of pairs using the `min` and `max` functions along with the `op` function. I. e., using the lists, `L[i]`, mentioned above one may define the height and width as follows:

```
w:=max(op(L[i][1]))-min(op(L[i][1])):
h:=max(op(L[i][2]))-min(op(L[i][2])):
```

Hence the total area of the rectangle is $A_r = hw$, so that the ratio of the area of the circle to the rectangle is,

$$\boxed{\frac{A_c}{A_r} = \frac{\pi}{hw}}$$

The appropriate proportionality, in terms of the numbers of pairs falling in each region, is left to be determined by the student.

1.5 Program Listing

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      A program to calculate the value of pi using a monte-carlo       c
c      technique. Specifically, we define a unit circle, with its       c
c      center at the origin, and a ‘unit box’ that inscribes the circle.c
c      We then pick two (pseudo) random points, x and y, on [-1,1] and  c
c      check to see if they fall within the union of the area of the    c
c      unit circle and unit box. If the random numbers are uniformly    c
c      distributed on [-1,1], then the ratio of the areas of the unit   c
c      circle and unit box is proportional to the number of points      c
c      falling in each. The (standard) deviation of the distribution is c
c      also calculated.                                                 c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      PROGRAM monte_pi

      REAL delta, pi
      INTEGER N
      PARAMETER(N=5000)
      INTEGER Nc, Ns
      REAL est, sum, mu, var, dev

c**************************** open data files
c     ‘pi.dat’ will store the succesive values of pi corresponding to
c     the total number of points.
c     ‘error.dat’ will output the number of points needed to achieve
c     a given error.
c****************************
      Open(50, FILE=’throws.dat’)
      Open(60, FILE=’pi.dat’)
      Open(70, FILE=’error.dat’)
c**************************** initialize variables
      Nc = 0
      Ns = 0
      est = 0.0
      sum = 0.0
      mu = 0.0
      var = 0.0
      dev = 1.0
c*************************** iterate until accuracy achieved
      DO i = 1, N
         x = Rand()*2-1
         y = Rand()*2-1
         WRITE(50,*) x, y
         IF ((x*x + y*y) .LE. 1.0) THEN
            Nc = Nc + 1
         ELSE
            Ns = Ns + 1
         END IF
         est = 4.0*Nc/(Nc+Ns)
         sum = sum + est
         mu = sum/(Nc+Ns)
         var = (var + (mu-est)**2)/(Nc+Ns)
         dev = SQRT(var)
         IF (MOD(i,10).EQ.0) THEN
               WRITE(60,*) i, est
               WRITE(70,*) 1/SQRT(Real(i)), dev
         END IF
      END DO
c**************************** close data files
      CLOSE(50)
      CLOSE(60)
      STOP
      END
```