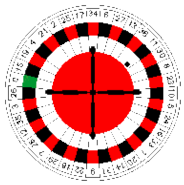


Random Walks

Learning Objective

Now that we have an idea of how to use the computer to generate pseudo-random numbers, we explore how to use these numbers to incorporate the element of chance into simulations. We do this first by simulating a random walk and in another module by simulating an atom decaying spontaneously. Both applications are good examples of how a computer can simulate nature and thereby permit the computer to be used as a virtual experimental laboratory.

Science and Math content objective:

- Students will better understand how chance enters into natural processes.
- Students will see how an approximate mathematical description can be developed of a natural process containing chance.
- Students will learn how a computer can simulate nature and thereby permit the computer to be used as a virtual experimental laboratory.

Scientific skills objective:

- Students will practice the following scientific skills:
  - Graphing (visualizing) data as *(x,y)* plots.
  - Discerning relations between variables in data containing statistical fluctuations.
  - Testing the reasonability, if not correctness, of a simulation by comparing it with a mathematical relation.

Activities

In this lesson, students will:

- Develop a mathematical model for a stochastic process.
- Develop an algorithm to simulate a stochastic process.
- Use a computer program to generate random numbers.
- Examine the random walks produced by a simulation.

- Test the reasonability, if not correctness, of a simulation by comparing it with a mathematical relation.

Related Video Lectures (Upper-Division Undergraduate)

1. Random Numbers for Monte Carlo
(science.oregonstate.edu/~rubin/Books/eBookWorking/VideoLecs/MonteCarlo/MonteCarlo.html)
2. Monte Carlo Simulations
(http://science.oregonstate.edu/~rubin/Books/eBookWorking/VideoLecs/MonteCarloApps/MonteCarloApps.html)

A Random Walk (Real World Phenomena)

Imagine a perfume bottle opened in the front of a classroom and the fragrance soon drifting throughout the room. The fragrance spreads because some molecules evaporate from the bottle and then collide randomly with other molecules in the air, eventually reaching your nose even though you are hidden in the last row. We wish to develop a model for this process which we can then use as the basis for a computer simulation of a random walk. Once we have tested the simulation, we can virtually "see"what the random walk taken by a perfume molecule looks like, and be able to predict the distance the perfume's fragrance travels as a function of time. The model we shall develop to describe the path traveled by a molecule is called a *random walk*. "Random" because it is chance collisions that determine the direction in which a perfume molecule travels, and "walk" because it takes a series of "steps" for the molecule to get from here to there. The same model has been used to simulate the search path of a foraging animal, the fortune of a gambler, and the accumulation of error in computer calculations, among others [1-9].

It is probably true that not all processes modeled as a random walk are truly random, in the sense that random means there is no way of predicting the next step from knowledge of the previous step. However, all these processes do contain an element of *chance* (what mathematicians call *stochastic processes*), and so a random walk, which has chance as a key element, is at least a good starting model for them.

An Aside on Root Mean Square Averages

Before we develop our computer model of a random walk, which is quite simple, we need to develop a simple mathematical model. Then we can use this mathematical model to see if the computer simulation is behaving in a way that we would expect nature's random walk to behave.

The math problem is to predict how many collisions, on the average, a perfume molecule makes in traveling a distance *R*. You are given the fact that, on the average, a molecule travels a distance *r* between collisions. (The velocities of the molecules and distance traveled between collisions increases with temperature, but we shall assume that the temperature is constant.)

Before we can proceed we need to be a bit more precise about what we mean by *average.* In common usage, we may use the word "average" to denote what in statistics is called the *mean*. For example, let's say *N* students take an exam, and the grade for student *i* on the exam is x_i. Then the mean or average score for the exam is

(1) $$\bar{x} \; = \; < x > = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Sometimes the numbers we want to take the average of can have negative as well as positive values (we hope your test scores do not have the former!). While it is perfectly acceptable to take the average of positive and negative numbers, sometimes we are more interested in what might be called the average *size* of the numbers, irrespective of their sign. In that case we would use another type of average known as the *root mean square* or *rms* average. In an rms average we first find the mean of the square of the numbers

(2) $$\overline{x^2} = \; < x^2 > = \frac{1}{N}\sum_{i=1}^{N} x^2.$$

Since only positive numbers are being averaged in Equation (2), there is no cancellation of terms in $\overline{x^2}$, but it is a distance squared and not a distance. To obtain a measure of the distance, we take the square root of $\overline{x^2}$,

(3) $$x_{rms} \; = \; \sqrt{\overline{x^2}} \; = \; \sqrt{\tfrac{1}{N}\textstyle\sum_{i=1}^{N} x^2}.$$

Figure 1. *Left:* A schematic of the N steps taken in a random walk that ends at a distance R from the origin. *Right:* The actual steps taken in a simulation of a 3-D random walk.

This quantity, the root mean square average of *x,* serves as a measure of the average magnitude of *x*, regardless of what sign *x* may take. Equation (3) is not as bad as it looks. It just means take the average of x^2, and then take the square root of the average.

Random Walk Mathematical Model

Many areas of science make use of a mathematical model of a random walk that predicts the average distance traveled in a walk of *N* steps. In order to verify the validity of our simulated random walk, we will compare the mathematical and simulated results. The mathematical model tells us that the average distance *R* from the origin (see Figure 1 Left) at which the walker ends up after *N* steps of length 1, is equal to $\sqrt{N}$. Of course if you add up the lengths of each step the total distance traveled is *N*, but since the steps are not all in a straight line, we can't just add up their lengths to calculate the distance from the starting point.

Here we present a simple model for a random walk in two dimensions, that is, on a flat surface. The same basic model can be applied to walks in one or three dimensions, or be extended with more sophisticated methods of including chance.

We assume that a "walker" takes sequential steps, with the direction of each step independent of the direction of the previous step. As seen in Figures 1 and 2, the walker starts at the origin and take *N* steps in the *x*-*y* plane, each of length 1. The first step has a horizontal (*x*) component of Δx_1, and a vertical (*y*) component of Δy_1. The second step has a horizontal component Δx_2, and a vertical component Δy_2, while the last step has a horizontal component Δx_N, and a vertical component Δy_N. Although each step may be in a different direction, the distances along the *x* and *y* axes just add algebraically. Accordingly, ΔX, the total *x* distance from the origin is

$$\Delta X = \Delta x_1 + \Delta x_2 + \ldots + \Delta x_N . \tag{4}$$

Likewise, ΔY, the total y distance is

$$\Delta Y = \Delta y_1 + \Delta y_2 + \ldots + \Delta y_N . \tag{5}$$

The radial distance *R* from the starting point after *N* steps (Figure 1 Left) is the hypotenuse of a right triangle with sides of ΔX and ΔY (Figure 2). We use Pythagoras's theorem to find *R*:

$$(6) \qquad R(N)^2 = \Delta X^2 + \Delta Y^2$$

$$(7) \qquad R(N) = \sqrt{\Delta X^2 + \Delta Y^2},$$

where the argument *N* is included to indicate that *R* is a function of the number of steps *N.*

Equation (6) gives us the distance traveled in a random walk of *N* steps. Since random walks have chance entering at each step, it is likely that different walks of *N* steps will result in different values for the length *R*. However, we expect that if we *average* over many walks, all with the same number *N* of steps*,* then we should obtain an average in which some of the random fluctuations have been removed.

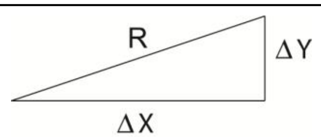


Figure 2. A right triangle with sides equal to the horizontal and vertical components of the length *R* of a random walk.

So how do we go about taking an average value for *R?* First let's make it clear that we are keeping the number of steps *N* in a walk constant while we take the average. To be explicit, let's say that we average over *M* different walks, all with *N* steps. Since the steps in our random walk are just as likely to go to the right as to the left, or to go up as often as down, if the number of walks *M* is large, then the average over all the walks of $\Delta X(N)$ = 0. Likewise, the average of ΔY would also vanish. Well, that won't do! The walker does end up some finite distance from the starting point every time, and we should be able to make at least an approximate prediction of that distance.

This then is where our old friend, the root-mean-square average of Equation (3) comes to our rescue. Since R^2 is always positive, we can take its average over *M* walks, and then take the square root of that average to get a measure of the average distance from the origin after *N* steps:

$$(8) \qquad \overline{R^2}(N) = \frac{1}{M}\sum_{j=1}^{M}(\overline{R^2})_j$$

$$(9) \qquad R_{rms}(N) = \sqrt{\overline{R^2}(N)}$$

where $\overline{R^2}_j$ indicates one of the j^{th}calculated value of the average $\overline{R^2}$.

If this seems a little confusing, do not worry too much about it now because we'll come back to when we evaluate Equation (8) as part of our simulation. Now we have to do a little algebra, which may seem complicated at first, but then becomes rather simple. Let's take our expression Equation (6) for R^2 and substitute Equations (4) and (5) for ΔX and ΔY:

$$(10) \qquad R(N)^2 = (\Delta X_1 + \Delta X_2 + \ \ldots + \Delta X_N\ )^2 +$$

$$(11) \qquad = (\Delta X_1)^2 + (\Delta X_2)^2 + \ \ldots + (\Delta X_N)^2 + \ \Delta X_1\, \Delta X_2 + \ \ldots + \Delta X_1\, \Delta X_N$$
$$+ (\Delta Y_1)^2 + (\Delta Y_2)^2 + \ \ldots + (\Delta Y_N)^2 + \ \Delta Y_1\, \Delta Y_2 + \ \ldots + \Delta Y_1\, \Delta Y_N.$$

Now Equation (11) is unquestionably rather messy looking, but not to worry. If we take the average of $R(N)^2$ for a large number *M* of different walks (all with *N* steps*),* then all of the cross terms like $\Delta Y_1\, \Delta Y_2$ will vanish (or average out to a small number) since they are all just as likely to be negative as positive. In contrast, the squared terms like $(\Delta X_1)^2$ are always positive and so do not vanish when we take the average. We are thus left with a much simpler approximation for the average value of $R(N)^2$:

$$(12) \qquad \overline{R^2}(N) \approx \ (\Delta X_1)^2 + (\Delta X_2)^2 + \ \ldots + (\Delta X_N)^2 + \ (\Delta Y_1)^2 + (\Delta Y_2)^2 + \ \ldots + (\Delta Y_N)^2$$

$$(13) \qquad = [(\Delta X_1)^2 + (\Delta Y_1)^2] + \ [(\Delta X_2)^2 + \ (\Delta Y_2)^2] + \cdots + [(\Delta X_N)^2 + (\Delta Y_N)^2].$$

But note, each of the sums in (12) is just the length of the corresponding step in the random walk, which we have already said equals 1. So we have *N* terms each equal to 1, and when we add them all up we get the simple result

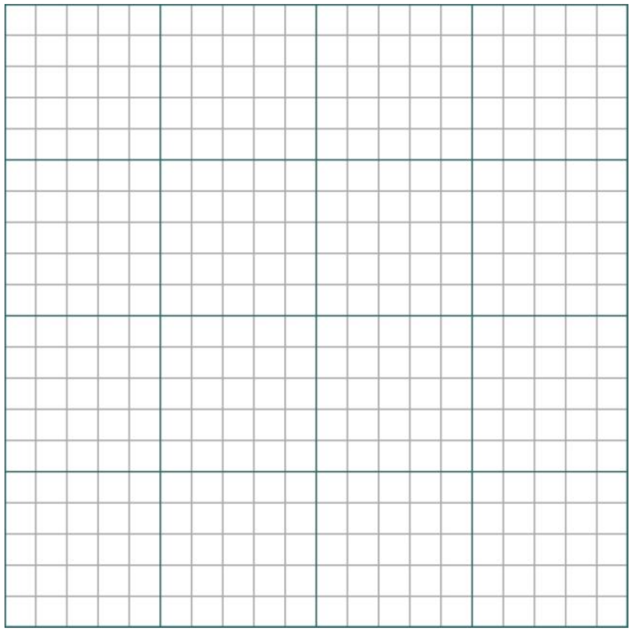
$$(14) \qquad \overline{R^2}(N) \approx N.$$

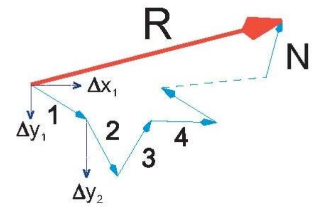
Equation (13) states that the average distance squared after a random walk of *N* steps of length 1 is *N.* If we take the square root of both sides of Equation (13) we obtain the desired expression for the root-mean-square, or rms, radius:

$$(15) \qquad R_{rms}(N) = \sqrt{\overline{R^2}(N)} \approx \sqrt{N}.$$

This is the simple result that characterizes a random walk. To summarize, if the walk is random, then we expect that on the average the walker is just as likely to be on the left as on the right, or as likely to be up as down, or in other words,

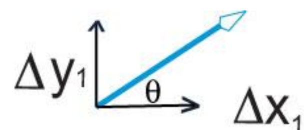
$$(16) \qquad \overline{\Delta X} \ \approx \overline{\Delta Y} \ \approx 0.$$





However, even though all directions are equally likely, the more steps that the walker takes, the further it gets from the origin, with the rms distance from the origin after *N* steps growing like the square root of *N.* In practice, we expect simulations to agree with equation (14) only when the number of steps is large and only when the number of paths used to take the average is large. Enough talk, let's simulate a random walk and see what we get.

Random Walk Simulation Algorithm



Consider again Figure 1 left. Our random walk simulation is rather simple:

i. Start with a walker at the origin, $(x,y) = (0,0)$.

ii. Choose a random direction $0 \leq \theta \leq 2\pi$ for the first step of the walk.

iii. Move the walker one step in that random direction.

iv. Increase the values for ΔX and ΔY by the values $(\Delta x = \cos\theta, \Delta y = \sin\theta)$ for that step.

v. Repeat steps ii-iv for *N* steps, with the cumulative values for ΔX and ΔY being updated for each step.

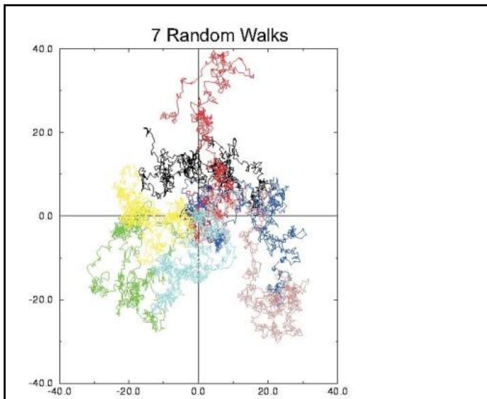


Figure 3. Paths followed in 7 random walks.

Hand Exercise: Random Walk Simulation by Hand

Here are five random angles in radian measure between 0 and 2π: 2.74, 1.59, 4.70, 4.04, 0.08. Take a piece of graph paper, or print out an electronic one, and use these numbers to draw five steps in a 2-D random walk starting at the origin. Measure the value of *R* for this walk and compare to Equation (15). A realistic walk might take thousands or millions of steps; be glad you do not have to draw that by hand.

Random Walk Simulation Algorithm

Part I, Go Take a Walk

1. Locate the random-number generator in whatever software system you are using.
2. If possible, determine how to use a different seed for each random walk. If you start your simulation from scratch each time, it will give the same results each time, and the averaging will not work properly. If you just continue using the generator already in use, then the sequence of random numbers probably will not repeat.
3. Since each step is of length 1, for step *i*, $\Delta x_i = \cos\theta_i$ and $\Delta y_i = \sin\theta_i$.
4. Sometimes you can tell your random number generator the range of values which you wish your random number r_i to span. If so choose the range $0 \leq r_i \leq 1$. In that case a random angle that covers the enture 2-D plane is just:

   (16) $$\theta_i = 2\pi r_i$$
5. Plot a line from the end of the previous step to the end of the new step, like in Figure 3. (Usually this is done just by plotting the points and having the plotting program connect the dots.)
6. Update your values of ΔX and ΔY, the total distances covered in the *x* and *y* directions, to include the latest step.
7. Continue the walk and the plot until your walker has made *N* steps (*N* = 200 is a reasonable value to start with, although larger values are better).
8. When your walker has completed *N* steps, calculate the value of R^2 according to Equation (6). Record (preferably on the computer) the values of *N*, R^2, ΔX and ΔY. Note that ΔX and ΔY are the total lengths in the *x* and *y* directions, and can be positive or negative.
9. Start your walker again at the origin. Using a different sequence of random numbers than in the previous walk. Have your walker complete another walk of *N* steps, and again record the values of *N*, R^2, ΔX and ΔY when the walk is over.
10. With the same value of *N* that you have been using, complete a total of at least 100 different random walks (more if your walk takes more than 200 steps).
11. Calculate and record the mean value of R^2 for all the walks according to Equations (8) and (9).
12. Next calculate the average values of ΔX and ΔY for these 100 walks.
13. If your walks are random, we would expect that $\overline{\Delta X} \approx \overline{\Delta Y} \approx 0$. You should obtain a "small" value for these quantities, if not exactly zero. To be more scientific about what we mean by

"small", calculate $\Delta X/\sqrt{R^2}$ and $\Delta Y/\sqrt{R^2}$. These are now dimensionless and measured in the proper relative scale to be compared to 1. For 100 walks we would expect small to be $\simeq$ 0.1; for 10,000 walks we would expect small to be $\simeq$ 0.01. (This is based on the statistical result that statistical deviations of relative values behave like $1/\sqrt{N}$.)

14. Use the values of R^2 obtained for each walk to obtain a value for the rms radius for this *N* value according to Equation (9), $R_{rms}(N) = \sqrt{\overline{R^2}(N)}$.

Part II, Does the Simulation Agree with the Math?

The mathematics predicts that if the walks are random, then the root-mean square distance from the origin, averaged over many walks, should approximately agree with Equations (15) and (16)

$$R_{rms}(N) = \sqrt{\overline{R^2}(N)} \approx \sqrt{N} \tag{15}$$

$$\overline{\Delta X} \approx \overline{\Delta Y} \approx 0\,. \tag{16}$$

Equation (16) says that the average length of the vector from the origin vanishes, $\bar{R}(N) \approx 0$, since positive and negative values for the components are equally likely. However, equation (15) tells us that the average distance from the origin does not vanish. The verification that your simulation agree with theses relations, at least approximately and within statistical errors, would give us confidence that the pseudorandom numbers we are generating are a good representation of truly random numbers, and that our random walk does indeed simulate those walks occurring in nature.

The key concept to understand in the test of (15) is that *there are two separate and independent parts* involved.

1. First, for a fixed number of steps *N*, you need to average R^2 over many walks in order to obtain and average value $\overline{R^2}(N)$ as a function of the number of steps.
2. Second, you need to vary the number of steps *N* and calculate $\overline{R^2}(N)$ for repeated walks at this new value of *N.* These walks must be different from those in part 1 in order to maximize randomness.

After you have completed all the averages, you take the square root of the mean squared value $R^2(N)$ for each *N* and see how well Equation (15) is satisfied.

3. Make a plot of $R_{rms}(N)$ versus $\sqrt{N}$. If your results scatter about a straight line, then you have verified that your simulation agrees with the mathematical model, and presumably nature.

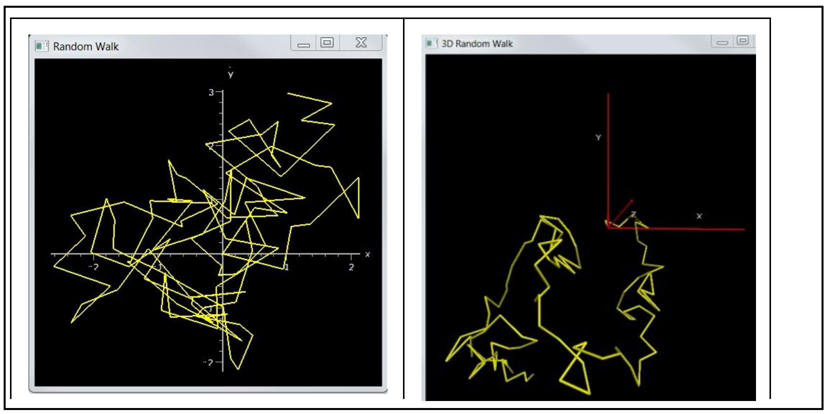
Excel Implementation

TBA

Python Implementation (WalkAngle.py)

The Python code **`WalkAngle.py`** conducts a single random walk in two dimensions and plots out the result.

1. You need to modify the program so that it repeats the calculation multiple times for this same value of *N*, and then calculates the average value of R^2.
2. Once you have that, you need to modify the program again so that it calculates $\overline{R^2}(N)$ for other values of *N,* saving the values.
3. Then you need to make a plot of $R_{rms}(N)$ versus $\sqrt{N}$.




```
""" From "A SURVEY OF COMPUTATIONAL PHYSICS", Python eBook Version
   by RH Landau, MJ Paez, and CC Bordeianu
   Copyright Princeton University Press, Princeton, 2012; Book  Copyright R
Landau,
   Oregon State Unv, MJ Paez, Univ Antioquia, C Bordeianu, Univ Bucharest,
2012.
   Support by National Science Foundation, Oregon State Univ, Microsoft Corp"""
```

```
# WalkAngle.py  Random walk with graph

from visual.graph import *
import random

random.seed(None)                  # Seed generator, None => system clock
jmax = 100
x    = 0.;          y = 0.                                 # Start at origin

graph1 = gdisplay(width=500, height=500, title='Random Walk', xtitle='x',
                  ytitle='y')
pts = gcurve(color = color.yellow)

for i in range(0, jmax + 1):
    pts.plot(pos = (x, y) )                          # Plot points
    theta = 2.*math.pi*random.random()               # 0 =< angle =< 2 pi
    x += cos(theta)                                  # -1 =< x =< 1
    y += sin(theta)                                  # -1 =< y =< 1
    pts.plot(pos = (x, y))
    rate(100)
print("This walk's distance R = ", sqrt(x*x + y*y)
```

Python Implementation (WalkAngle3D.py)

The 3-D implementation of a random walk is similar to the 2-D one, but walks in an extra dimension and makes a 3-D plot of the walk. Accordingly, it is more complicated (it uses arrays, that is, variables with indices). This 3-D walk would need the same modification as the 2-D walk to test the simulation.

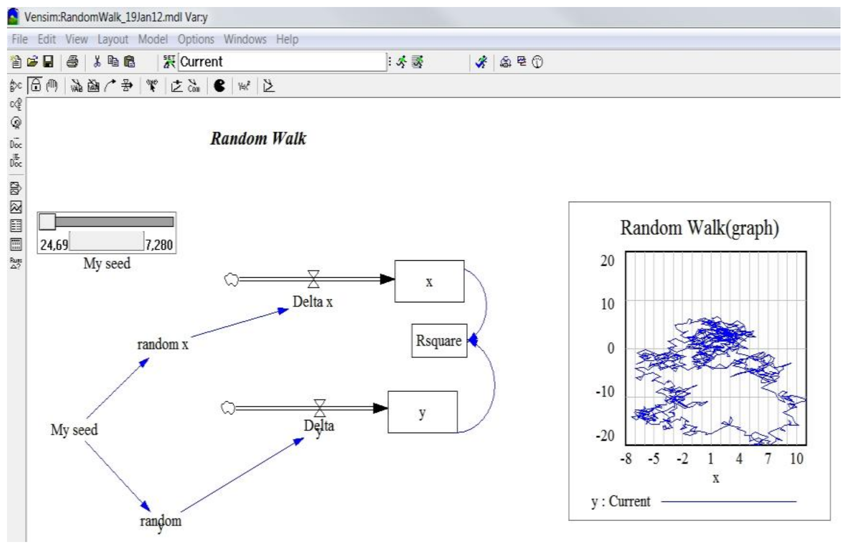
```
""" From "A SURVEY OF COMPUTATIONAL PHYSICS", Python eBook Version
by RH Landau, MJ Paez, and CC Bordeianu
Copyright Princeton University Press, Princeton, 2012; Book  Copyright R Landau,
Oregon State Unv, MJ Paez, Univ Antioquia, C Bordeianu, Univ Bucharest, 2012.
Support by National Science Foundation , Oregon State Univ, Microsoft Corp"""

# Walk3DAngle.py  3-D Random walk with 3-D graph
from visual import *
```

```
import random

random.seed(None)                    # Seed generator, None => system clock
jmax = 1000
xx =yy = zz =0.0                                         # Start at origin

graph1 = display(x=0,y=0,width = 600, height = 600, title = '3D Random Walk',
                  forward=(-0.6,-0.5,-1))
#  Curve, its parameters and labels
pts   = curve(x=list(range(0, 100)), radius=10.0,color=color.yellow)
xax   = curve(x=list(range(0,1500)), color=color.red, pos=[(0,0,0),(1500,0,0)],
radius=10.)
yax   = curve(x=list(range(0,1500)), color=color.red, pos=[(0,0,0),(0,1500,0)],
radius=10.)
zax   = curve(x=list(range(0,1500)), color=color.red, pos=[(0,0,0),(0,0,1500)],
radius=10.)
xname = label( text = "X", pos = (1000, 150,0), box=0)
yname = label( text = "Y", pos = (-100,1000,0), box=0)
zname = label( text = "Z", pos = (100, 0,1000), box=0)

pts.x[0] =  pts.y[0] =  pts.z[0] =0            # Starting point
for i in range(1, 100):
    theta = 2.*math.pi*random.random()              # 0 =< theta =< 2 pi
    phi   =    math.pi*random.random()              # 0 =< phi =< pi
    xx   += sin(phi)*cos(theta)           # -1 =< x =< 1
    yy   += sin(phi)*sin(theta)           # -1 =< y =< 1
    zz   += cos(phi)                    # -1 =< z =< 1
    pts.x[i] = 200*xx - 100
    pts.y[i] = 200*yy - 100
    pts.z[i] = 200*zz - 100
    rate(100)
print("This walk's distance R =", sqrt(xx*xx + yy*yy + zz*zz))
```

Vensim Implementation (RandomWalk.mdl)

```
(01)   Delta x=
              random x
       Units: **undefined**
(02)   Delta y=
              random y
       Units: **undefined**
(03)   FINAL TIME  = 1000
       Units: Month
       The final time for the simulation.
(04)   INITIAL TIME  = 0
       Units: Month
       The initial time for the simulation.
(05)   My seed=
              654322
       Units: **undefined**
(06)   random x=
              RANDOM UNIFORM( -1 , 1 , My seed )
       Units: **undefined**
(07)   random y=
              RANDOM UNIFORM( -1 , 1 , My seed )
       Units: **undefined**
```

```
(08)    Rsquare=
             x*x+y*y
        Units: **undefined**
(09)    SAVEPER  =
              TIME STEP
        Units: Month [0,?]
        The frequency with which output is stored.
(10)    TIME STEP  = 1
        Units: Month [0,?]
        The time step for the simulation.
(11)    x= INTEG (Delta x,  0)
        Units: **undefined**
(12)    y= INTEG ( Delta y, 0)
        Units: **undefined**
```



By nature of their construction, computers are deterministic and so cannot generate truly random numbers. Consequently, the so called “random” numbers generated by computers are not truly random, and if you look hard enough you can verify that they are correlated to each other. Although it may be quite a bit of work, if we know one random number in the sequence as well as the preceding numbers, it is always possible to determine the next one. For this reason, computers are said to generate *pseudorandom numbers* (yet with our incurable laziness we won't bother saying “pseudo “ all the time).

Glossary References

1. Van Kampen N. G., Stochastic Processes in Physics and Chemistry, revised and enlarged edition (North-Holland, Amsterdam) 1992.

2. Redner S., A Guide to First-Passage Process (Cambridge University Press, Cambridge, UK) 2001.

3. Goel N. W. and Richter-Dyn N., Stochastic Models in Biology (Academic Press, New York) 1974.

4. Doi M. and Edwards S. F., The Theory of Polymer Dynamics (Clarendon Press, Oxford) 1986

5. De Gennes P. G., Scaling Concepts in Polymer Physics (Cornell University Press, Ithaca and London) 1979.

6. Risken H., The Fokker–Planck Equation (Springer, Berlin) 1984.

7. Weiss G. H., Aspects and Applications of the Random Walk (North-Holland, Amsterdam) 1994.

8. Cox D. R., Renewal Theory (Methuen, London) 1962.

9. Dana Mackenzie, *Taking the Measure of the Wildest Danc*