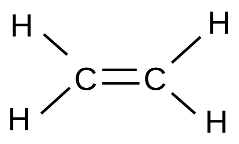
CHAPTER 3 - RANDOM WALKS AND RANDOM NUMBERS

Many physical characteristics of polymers, such as their elasticity, can be understood in terms of the properties of linear and branched chains, both in isolation and in networks. In this section, we concentrate on isolated linear chains. In Secs. 3.1-3.4, a short introduction to the chemical structure of polymers is presented, followed by an analytical treatment of some aspects of linear chains. Many attributes of chains, particularly when steric and other interactions among chain elements are important, have been found first or exclusively by computer simulation. Central to such simulations are random numbers and the Monte Carlo method. We present an introduction to random number generators in Sec. 3.5-3.7. As a project, we investigate the end-to-end distribution of chain lengths in Sec. 3.8. The particular model that we use for the chain does not require us to use the "importance sampling" aspect of the Monte Carlo approach, and so we delay the treatment of the Monte Carlo method until Chaps. 4 and 5. Lastly, more in-depth treatment of flexible chains is given in App. C.

3.1 Polymers

We start with a simple example, the polymerization of ethylene:



(3.1)

What happens when ethylene *monomers* link together to form a polyethylene *polymer*? The reaction can be written in the form

$$CH_2 = CH_2 + CH_2 = CH_2 + \ldots \;\rightarrow\; CH_2 - CH_2 - CH_2 - CH_2 \ldots \qquad (3.2)$$

where the chain on the right hand side of Eq. (3.2) is a long alkane. In the reaction, the total number of C - H bonds remains the same, but each C=C double bond is replaced by two C-C single bonds. One of the single bonds is between carbons in the original monomers, and one is between carbons on different monomers.

A typical bond energy for double bonds between carbons is 6.3 eV, meaning that it takes 6.3 eV to break the bond. But it takes a little more than half of this energy - typically 3.6 eV - to break a single carbon-carbon bond. Thus, there is (2 x 3.6) - 6.3 = 0.9 eV liberated when a double bond is replaced by two single bonds between carbons. Note that there is an energy barrier against polymerization, or else ethylene would polymerize spontaneously.

Polyethylene is a *linear* polymer, a term which implies a linear, unbranched chain geometry even if there are small groups like CH_3 attached to the chain at intervals. However, there are also *non-linear* or *branched* polymers in which a chain divides at a junction and continues as two or more chains. For example, the addition of urea (NH_2 - CO - NH_2) and formaldehyde (H_2CO) in a 1:1 ratio leads to a linear polymer

$$\text{- NH - CO - NH - CH}_2\text{ - NH - CO - NH - CH}_2\text{ - } + H_2O \qquad (3.3)$$

However, if excess formaldehyde is added, then branching occurs and the amine groups can be connected to three chains:

$$\begin{array}{ll} \text{- NH - CO - NH - CH}_2\text{ - } & \text{N - CO - NH - CH}_2\text{ -} \\ & | \\ & \text{CH}_2 \\ & | \\ & \text{NH - CO - NH - CH}_2\text{ -} \end{array} \qquad (3.4)$$

In some cases, chains can be linked to form a *network*. For example, double bonds on different polyisoprene chains can be linked together by sulfur to form a network in a chemical process known as *vulcanization*.

A polymer chain can be composed of hundreds or thousands of monomers. At each link in a saturated chain, there is at best moderate resistance to rotation: the chains are relatively floppy and many isoenergetic configurations are present at room temperature. This situation is different from ionic crystals or metals in which the atoms oscillate about well-defined positions. The properties of highly flexible materials are a natural area of application for statistical mechanics. Many interesting characteristics of rubber, such as the observation first made by John Gough in 1805 that natural rubber contracts when heated, arise from the entropic properties of polymeric chains. Statistical mechanics explains Gough's observations by demonstrating that a flexible chain behaves like a spring with a spring constant that increases linearly with temperature.

3.2 Random walks

The simplest model for the geometry of a single polymeric molecule is one in which the polymer is represented by a chain of N vectors added tip-to-tail, each vector representing a bond or monomer. We assume for the moment that each monomer has the same length a, and that the vector describing a particular monomer i is $\boldsymbol{a}_i$. The contour length l_c along the chain is then

$$l_c = Na, \tag{3.5}$$

and the end-to-end displacement $\boldsymbol{r}_{ee}$ is just the vector sum of the individual vectors $\boldsymbol{a}_i$:

$$\boldsymbol{r}_{ee} = \Sigma_i \, \boldsymbol{a}_i \, . \tag{3.6}$$

The addition of individual vectors to obtain $\boldsymbol{r}_{ee}$ is illustrated in Fig. 3.1. Taking the ensemble average over all chains with the same number of monomers N, the end-to-end displacement squared $<\boldsymbol{r}_{ee}^2>$ is

$$<\boldsymbol{r}_{ee}^2> = \Sigma_i \, \Sigma_j <\boldsymbol{a}_i \bullet \boldsymbol{a}_j>. \tag{3.7}$$

The sums on the right hand side of Eq. (3.7) may or may not involve restrictions. We consider two (of many) possibilities:

(i) *Freely-jointed ideal chain*

If a given bond vector $\boldsymbol{a}_{i+1}$ can have any orientation independent of its neighboring vector $\boldsymbol{a}_i$, then the ensemble average of $\boldsymbol{a}_i \cdot \boldsymbol{a}_j$ should vanish if $i \neq j$:

$$<\boldsymbol{a}_i \cdot \boldsymbol{a}_j> = 0 \qquad \text{if } i \neq j. \tag{3.8}$$

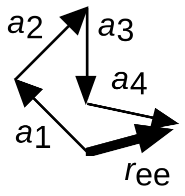


Fig. 3.1 End-to-end displacement $\boldsymbol{r}_{ee}$ for a chain whose elements have a common length and random orientation.

Thus, the only terms which survive in the double sum in Eq. (3.7) are the diagonal elements $i = j$, each of which equal a^2. Thus, for a freely jointed chain

$$<\boldsymbol{r}_{ee}{}^2> = Na^2. \qquad (3.9)$$

(ii) *Freely rotating chain at fixed bond angle*

Suppose that the monomers are free to rotate about a given bond, but that the polar angle θ between one bond and the next is fixed. That is

$$<\boldsymbol{a}_i \cdot \boldsymbol{a}_{i+1}> = \boldsymbol{a}_i \cdot \boldsymbol{a}_{i+1} = -a^2 \cos\theta. \qquad (3.10)$$

This is approximately the situation in alkane chains, if one ignores the long range steric interaction between elements on the chain. The definition of the bond angle θ is that if $\boldsymbol{a}_i$ and $\boldsymbol{a}_{i+1}$ point in the same direction, then θ is 180^o, and if they point in opposite directions, then θ is zero. Although Eq. (3.10) applies to neighboring monomers, a recursion relation based upon Eq. (3.10) can be used to find $\boldsymbol{a}_i \cdot \boldsymbol{a}_{i+k}$ for any k. Consider the orientations of the vectors indicated in Fig. 3.2. Clearly, the average projection of $\boldsymbol{a}_{i+1}$ on vector $\boldsymbol{a}_i$ is just $-\boldsymbol{a}_i\cos\theta$. Thus, the average projection of $\boldsymbol{a}_{i+2}$ on $\boldsymbol{a}_{i+1}$ is $-\boldsymbol{a}_{i+1}\cos\theta$, so that $<\boldsymbol{a}_i \bullet \boldsymbol{a}_{i+2}> = (-\cos\theta)<\boldsymbol{a}_i \bullet \boldsymbol{a}_{i+1}> = a^2\ (-\cos\theta)^2$. This argument can be repeated as neccesary to give

$$<\boldsymbol{a}_i \bullet \boldsymbol{a}_{i+k}> = a^2\ (-\cos\theta)^k. \qquad (3.11)$$

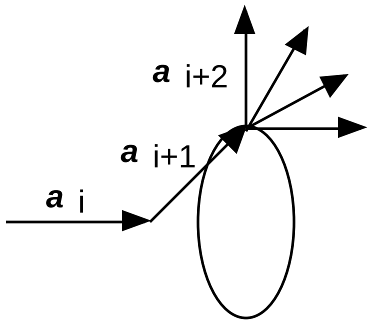


Fig. 3.2 Allowed orientations for three elements in a chain where the bond angle between neighboring elements is fixed, but the azimuthal angle ranges from 0 to 2π.

When Eq. (3.7) is evaluated using Eq. (3.11), one finds at large N

$$\langle \boldsymbol{r}_{ee}^2 \rangle = Na^2 \,(1 - \cos\theta) / (1 + \cos\theta). \qquad (3.12)$$

Details of the proof can be found in PHYS 883 lectures.

Now, Eq. (3.12) has the same scaling exponent for $\langle \boldsymbol{r}_{ee}^2 \rangle^{1/2}$ as a function of N as does Eq. (3.9), namely $N^{1/2}$. However, the length scale of the scaling is different. Eq. (3.12) reduces to (3.9) when the chain is measured on length scales of $a\,[(1 - \cos\theta) / (1 + \cos\theta)]^{1/2}$. Hence, there is a length scale on which all chains without self-avoidance appear ideal. One way of defining the length scale (see below) is through an effective bond length B_{eff} such that $\langle \boldsymbol{r}_{ee}^2 \rangle = NB_{\mathrm{eff}}^2$. For freely rotating chains, we have just established that

$$B_{\mathrm{eff}} = a\,[(1 - \cos\theta) / (1 + \cos\theta)]^{1/2}. \qquad (3.13)$$

Applying this to alkane chains with $\theta = 109.5^{\mathrm{o}}$ gives $B_{\mathrm{eff}} = \sqrt{2}\, a$.

3.3 Distribution of $\boldsymbol{r}_{ee}$

Chains which can self-intersect themselves show ideal scaling behavior ($\langle \boldsymbol{r}_{ee}^2 \rangle = NB_{\mathrm{eff}}^2$) and have a particularly simple form for the distribution of end-to-end vectors $\boldsymbol{r}_{ee}$. To obtain this form, let us break up the three dimensional problem into three one-dimensional problems by considering the distribution for $r_{ee,x}$, which is the projection of $\boldsymbol{r}_{ee}$ on the x-axis. The situation is shown in Fig. 3.3.

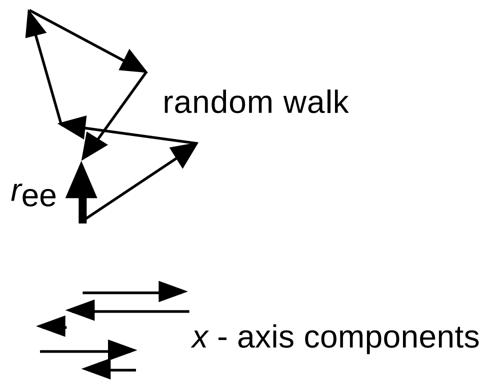


Fig. 3.3 Projection of a two-dimensional random walk onto the x-axis.

The x-component of the end-to-end displacement is just the sum of the individual monomer vector projections on the x-axis:

$$r_{ee,x} = \Sigma_i \, a_{i,x} \tag{3.14}$$

For ideal chains, $a_{i,x}$ is uncorrelated with $a_{i+1,x}$ and the situation is equivalent to the one-dimensional random walk with steps of variable size. If the number of steps is large, then the distribution with variable step length is the same as the distribution with fixed step length l_x fixed at $<a_x^2>^{1/2}$. Now $<a_x^2>^{1/2}$ refers to the expectation of the projection of the individual steps on the x-axis. Since we expect that $<a_x^2> = <a_y^2> = <a_z^2>$, then

$$l_x^2 = <a_x^2> = a^2/3. \tag{3.15}$$

For the sake of making the notation a little less cumbersome in the following few equations, let us define

$$x \equiv r_{ee,x}. \tag{3.16}$$

In a one-dimensional random walk of fixed step size, the end-to-end displacement x obeys the binomial distribution. In the continuum limit, the binomial distribution becomes Gaussian, and the probability of finding a walk with displacement between x and $x + dx$ is just

$$P(x)dx = (2\pi\sigma^2)^{-1/2} \exp(-x^2/2\sigma^2) \, dx \tag{3.17}$$

where $P(x)$ is the probability density (probability per unit length) and σ is

$$\sigma^2 = Nl_x^2 = Na^2/3. \tag{3.18}$$

This distribution assumes that the chain starts at the origin, and is normalized to unity

$$\int P(x)dx = 1. \tag{3.19}$$

The expectations of the displacements are what we anticipate:

$$<r_{ee,x}> = <x> = \int xP(x)dx = 0 \tag{3.20}$$

and

$$<r_{ee,x}{}^2> = <x^2> = \int x^2 P(x) dx = \sigma^2 = Na^2/3. \tag{3.21}$$

The one-dimensional probability density $P(x)$ can be easily generalized to the three dimensional probability density $P(x,y,z)$. The probability of finding the end-to-end displacement in a volume $dxdydz$ centered on the position (x,y,z) is $P(x,y,z)dxdydz$, where $P(x,y,z)$ is the probability per unit volume given by

$$\begin{aligned} P(x,y,z) &= P(x)P(y)P(z) \\ &= (2\pi\sigma^2)^{-3/2} \exp[-(x^2+y^2+z^2)/2\sigma^2]. \end{aligned} \tag{3.22}$$

Eq. (3.22) says that the most likely set of coordinates for the tip of the chain is (0,0,0). That is, the most likely single place that the tip of the chain will be found is at the tail of the chain, which sits at the coordinate origin by definition. Eq. (3.22) does *not* say that the most likely chain displacement is zero.

To find the characteristics of the chain displacement, we must take into account that many coordinate positions have the same r. The probability for the chain having end-to-end displacement between r and $r + dr$ is $P_{\text{rad}}(r)dr$, where $P_{\text{rad}}(r)$ is the probability per unit length obtained from

$$P(x,y,z)dxdydz = P_{\text{rad}}(r)dr \tag{3.23}$$

so that

$$P_{\text{rad}}(r) = 4\pi r^2 (2\pi\sigma^2)^{-3/2} \exp(-r^2/2\sigma^2). \tag{3.24}$$

We can take the derivative of $P_{\text{rad}}(r)$ with repect to r to find the most likely value of r_{ee}, and take the usual expectations. A summary of the results for ideal chains in three dimensions is

$$r_{\text{ee, most likely}} = (2/3)^{1/2} N^{1/2} a \tag{3.25}$$

$$<r_{ee}> = (8/3\pi)^{1/2} N^{1/2} a \tag{3.26}$$

$$<r_{ee}{}^2> = Na^2. \tag{3.27}$$

3.4 Self-avoiding Chains

While the "ideal chains" of Sec. 3.2 may intersect themselves, physical systems have an excluded volume that enforces self-avoidance of the chain. This steric interaction among the chain elements is important for chains in 1-, 2- and 3-dimensional systems. Consider the simple situation in which a chain lies along the x-axis. Self-avoidance forbids the chain from reversing on itself from one step to the next, so that the end-to-end distance must be just the contour length Na. But Eq. (3.9) shows that r_{ee} for ideal chains scales like $N^{1/2}$, *independent of embedding dimension*. Thus, we conclude that in one dimension, self-avoidance of the chain dramatically affects its scaling properties: N^1 for self-avoiding chains and $N^{1/2}$ for ideal chains. Similar conclusions can be drawn for chains in 2 and 3 dimensions, although the scaling exponents are different.

A simple model for the length-scaling exponent of self-avoiding chains was proposed by Flory (1953). The calculation evaluates the power-law dependence of the free energy on the effective chain size r and the number of segments N at both large and small r. Minimizing the free energy yields r as a function of N. Since our goal is to extract the scaling exponent, we do not pay close attention to numerical factors like $4\pi/3$, and we use r to represent the effective size of the chain, as characterized by an end-to-end length or a root-mean-square radius. Also, our model chains have no explicit energy scale other than the temperature. Following de Gennes (1979), the behavior of the free energy is evaluated in two regimes:

(i) *Short distances*. Steric repulsion between the chain segments causes the chain to swell compared to an ideal chain. The repulsive energy experienced by one segment, through its interaction with other segments, is proportional to the concentration of segments, roughly N / r^{d} for a chain in a d - dimensional space. Thus, the total repulsive energy experienced by all N segments is proportional to N^2 / r^{d}. Now, the repulsive energy also will be proportional to the excluded volume of the segment-segment interaction, which we characterize by a parameter v_{ex}. Taking the excluded volume as a hard-core interaction, then the energy scale of the interaction is set by the temperature $k_{\mathrm{B}}T$. Thus, the steric contribution to the free energy should behave like

$$F = k_{\mathrm{B}}T\, v_{\mathrm{ex}}\, N^2/r^{\mathrm{d}}, \qquad (3.28)$$

where all constants have been absorbed into v_{ex}.

(ii) *Long distances*: As a chain is stretched, the number of configurations that it can adopt at a fixed end-to-end distance decreases rapidly. As shown in Sec. 3.3, the probability of finding a given end-to-end distance r for an ideal chain decays exponentially as $\exp(-dr^2/2Na^2)$, where a is the elementary segment length. Recalling that the entropy S is proportional to the logarithm of the probability, then to within a constant

$$S/k_B = - dr^2/2Na^2. \tag{3.29}$$

The entropic contribution to the free energy at long distances can be found through $F = E - TS$. Combining Eqs. (3.28) and (3.29) and discarding overall normalization constants, the free energy of the self-avoiding chain behaves like

$$F = k_B T v_{ex} N^2/r^d + k_B T dr^2/2Na^2. \tag{3.30}$$

This expression shows that there is a penalty for pushing the chain elements close together (small r) and there is a penalty for stretching out the chains (large r). The value for r that minimizes F can be found by taking the derivative of Eq. (3.30) with respect to r, holding other quantities fixed, and this value scales like

$$r \approx N^{3/(2+d)}. \tag{3.31}$$

The scaling behavior of Eq. (3.31) is expected for any length scale r that characterizes the linear dimension of the system as a whole, such as the end-to-end distance r_{ee} and the root mean square radius of the system (which is equal to $<r_{ee}^2>^{1/2}/\sqrt{6}$). The exponent on the right hand side of Eq. (3.31) is called the Flory exponent.

A different scaling exponent is expected if elements on the chain are attracted strongly to each other even though the chain is self-avoiding. If the chain forms a dense ball (like a liquid drop) then the volume of the ball V should be proportional to the number of chain segments N; *i.e.* $V \sim N^1$. But the "volume" of a sphere in d-dimensional space is proportional to R^d, so that $R \sim N^{1/d}$ for dense chains.

3.5 Random sequences

Many aspects of computational problems in physics deal with random processes, or random sampling. Examples include:

- •polymer configurations (Sec. 3.1)
- •integration by random sampling (Chap. 1)
- •non-deterministic scattering processes.

The generation of random numbers for computational use in attacking these problems is a simple, but important component of computational physics that has drawn a great deal of attention. The central problem is that a computer does not generate truly random numbers, but rather a sequence of numbers whose values may have minimal correlation under certain circumstances.

A typical random number generator involves an algorithm which takes a number n_i in a sequence, and generates a number n_{i+1}. For example, suppose that our computer has 3 bit numbers. An algorithm might involve multiplying the number n_i by 3, and subtracting 7 repeatedly from the product until n_{i+1} lies between 0 and 6. Say we start with $n_i = 2$:

i = iteration	n_i	n_{i+1}
0	2	6
1	6	4
2	4	5
3	5	1
4	1	3
5	3	2
6	2	6

Certainly, this random number generator samples all numbers between 1 and 6, and produces a sequence which is neither 1, 2, 3, 4, 5, 6 nor its inverse. Note that if the initial number is chosen to be zero, then all subsequent $n_i = 0$ as well.

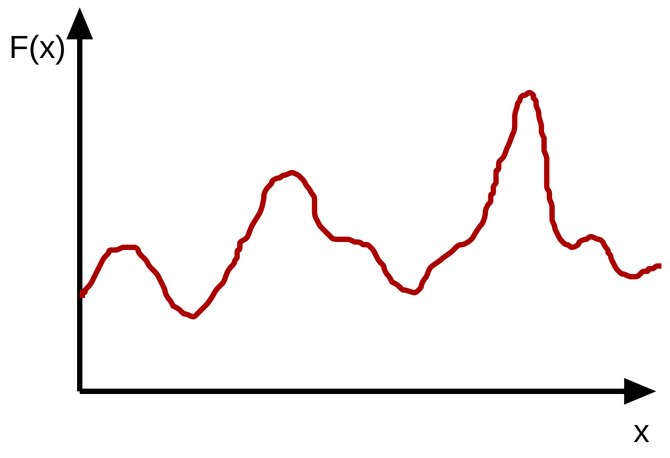
This random number generator has some obvious drawbacks:

(i) In a computer, numbers are represented by a finite number of digits, so that a sequence of random numbers may be periodic (even if the period is large). In our example, the sequence repeats itself after six numbers have been generated.

(ii) There may be correlations between successive numbers in the sequence. Again, in our example, numbers greater than 3 tend to be followed by numbers greater than 3 while numbers less than 4 tend to be followed by numbers less than 4.

It may be that neither of these effects is important for the application of interest. But in simulations involving perhaps 10^9 calls to a random number generator, one has to at least be aware that one's random number generator may possess properties (i) and (ii), and that these properties may affect the results of the calculation. We present the following examples to illustrate the problems that can be introduced by inappropriate random number generators.

Example A - Periodicity problems You have a strongly oscillating function of one variable, and you wish to obtain the mean value of the function by random sampling, using only the 3-bit random number generator decribed above.

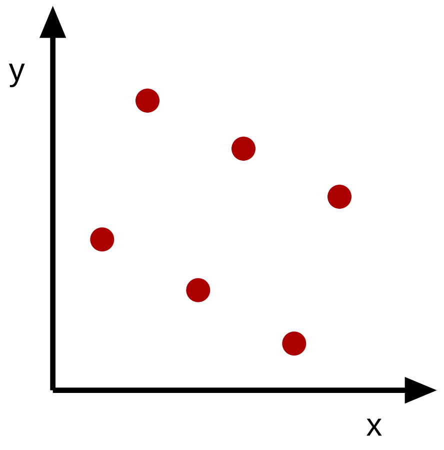


After six calls to the example sequence, the mean value would be

$$<F> = [\, F(2) + F(6) + F(4) + F(5) + F(1) + F(3) + F(2)\,] \,/\, 6, \qquad (3.32)$$

where x = 1...6 represent 6 equally spaced values of x. Now, given the structure of the function $F(x)$, one would be lucky to obtain an accurate mean by sampling it at only six values of x. Equally important, it doesn't matter how many times the sample is taken - the sample contains only the same six values of $F(x)$ because our 3-bit random sequence has a period of 6. Thus, one must choose a random number generator whose period is long compared to the numerical structure of the problem.

Example B - Correlation problems You have a function of two variables, x and y. Again, you want to construct a mean by randomly sampling x and y. If successive calls to the 3-bit random number generator are used to generate (x,y) pairs, then the sampling of the xy plane may not be uniform. For simplicity, let's use our sequence to produce 6 pairs of numbers $(x,y) = (n_i,n_{i+1})$ = (2,6), (6,4), (4,5), (5,1), (1,3), (3,2):



While the individual values of x and y may be properly sampled, the pairs of values are not properly sampled. Indeed, one can see that the (x,y) pairs fall on *lines* (or *planes* in higher dimensions). The reason for this behavior is that successive numbers in the random sequnce are correlated.

The two examples show both the utility of random sampling, and also the potential pitfalls of simple random number generators. There are a number of approaches that can be used to lessen the undesirable effects of simple random number generators. We mention these first in general terms before moving to specific algorithms in Sec. 3.6.

Periodicity In the sample generator, the period for the 3-bit integers is 6, since the sequence does not include the numbers 0 and 7. Most computers have 32-bit architecture, so the period can be made closer to 2^{32} with a suitable choice for the multiplier (which is 3 in the example). A further useful trick is to intermittently skip a segment of the sequence by using the computer's clock to specify the size of the segment to be skipped. However, this may introduce machine-dependent calls into the code, which thereby render the code less portable between machines or operating systems.

Correlations Much work has been done on the choice of multiplier (3 in the example) and modulus (7 in the example) to reduce the presence of correlations. We report some choices in the next section.

3.6 Uniform deviates

The random numbers described in Sec. 3.5 are examples of uniform deviates: the numbers lie uniformly over some range. A given number is equally likely to fall anywhere between 1 and 6 in the example. In Sec. 3.7, this is contrasted to non-uniform distributions in which numbers in some range are more likely to appear in the sequence than numbers in a different range.

The specific example that is introduced in Sec. 3.5 is of the form

$$N_{i+1} = aN_i \pmod m \tag{3.33}$$

where we set the **multiplier** *a* to 3, and the **modulus** *m* to 7. This generator is an example of a general class of **linear congruential generators**

$$N_{i+1} = aN_i + c \pmod m \tag{3.34}$$

which includes a constant **increment** *c*. How "good" the generator is depends on the choice of *a*, *m*, and *c*. Tests of the "goodness" of a generator are described in Press *et al*. (1992), and in more detail in Park and Miller (1988).

The implementation of Eqs. (3.33) - (3.34) may be slightly problematic, in that the product aN_i may exceed the maximum value allowed for a 32-bit integer. Press *et al*. (1992) describe a trick developed by Schrage (1979) to avoid this overflow problem. An implementation of Eq. (3.34) by Park and Miller (1988), based on Schrage (1979), is the following [taken from Press *et al*. (1992)]:

```
#define IA 16807
#define IM 2147483647
#define AM (1.0 / IM)
#define IQ 127773
#define IR 2836
#define MASK 123459876

/* a long integer idum must be given an initial value (not equal to MASK)
before calling the function ranpm; idum must not be altered between
calls to ranpm */

float ranpm(long *idum)
/* generates uniform random deviate between 0.0 and 1.0 */
{
  long k;
  float ans;
  *idum ^= MASK;
  k = (*idum) / IQ;
  *idum = IA * (*idum - k * IQ) - IR * k;
  if(*idum < 0) *idum += IM;
  ans = AM * (*idum);
  *idum ^= MASK;
  return ans;
}
```

The period of **ranpm** is 2^{31} - 2, or about 2 x 10^9. Park and Miller propose other combinations of *a* and *r* that can be used with $m = 2^{31}-1 = 2147483647$:

IM	IA	IQ	IR
2147483647	16807	127773	2836
2147483647	48271	44488	3399
2147483647	69621	30845	23902

At present, no other combinations should be used. The author has used the first set extensively, and found only miniscule correlations, whose negative consequences can be avoided with careful coding. Press *et al*. (1992) discuss several modifications to this simple algorithm which can be used to reduce correlations, without dramatically increasing execution time. They also discuss several "quick and dirty" algorithms that run faster than **ranpm** of Park and Miller.

3.7 Non-uniform distributions

There are many situations in which one wants to sample a non-uniform distribution. For example:

- the decay times for an ensemble of radioactive nuclei obey an exponential distribution
- the velocities of a Maxwell-Boltzmann ensemble of molecules obey a normal distribution (along a given direction).

With some modification, uniform deviates can be used to generate these other distributions.

Consider the distribution of decay times for a sample of radioactive nuclei. At any given time, the number of nuclei that decay is proportional to the number of radioactive nuclei in the sample. As time goes on, there are fewer and fewer radioactive nuclei left in the sample, and hence the number of nuclei decaying per unit time decreases (assuming that the products of the decay are not themselves radioactive). It can be shown that the number of radioactive nuclei $N(t)$ remaining in the sample at time t follow an exponential function

$$N(t) = N(t{=}0)\ \exp(-\lambda t) \tag{3.35}$$

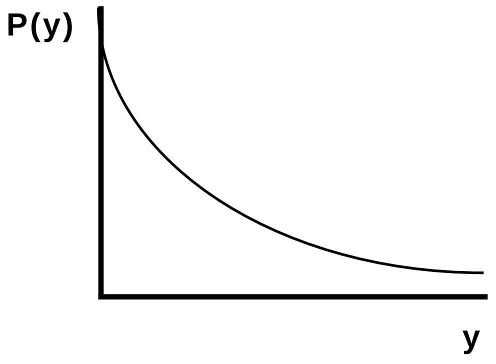
where λ is called the decay constant. To simulate radioactive decay, one would have to choose the decay times of nuclei at random, from an exponential distribution. The probability distribution $P(y)$ for a given nucleus to decay at a reduced time $y = \lambda t$ over a given range in reduced time dy is then:

$$P(y)dy = \exp(-y)dy, \tag{3.36}$$

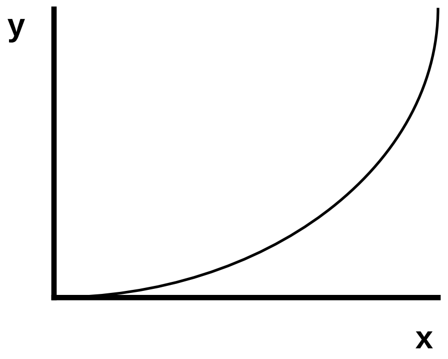
which must be normalized to unity, since the nucleus in question has to decay eventually:

$$\int P(y)dy = 1. \tag{3.37}$$

What we want to do is choose a random value of a number $0 < x < 1$, and put it in a function that produces a value of the reduced time *y*, such that the distribution of *y* obeys Eq. (3.36). If the probability distribution in *y* appears like



then the function that generates *y* from a specific values of the uniform deviate *x* must look like



This graph shows that many values of *x* produce small values of *y*, while few values of *x* produce large values of *y*, as demanded by Eq. (3.36). To put the graph into mathematical terms,

[*probability that the random number has a value less than* x]
= [*probability that the reduced time has a value less than* y].

Since *x* is uniformly distributed between 0 and 1, then

[*probability that the random number has a value less than* x] = *x*.

Since *y* is not uniformly distributed, then

[*probability that the reduced time has a value less than* y]

$$= \int_0^y P(y) dy$$

Thus

$$x = \int_0^y P(y) dy \tag{3.38}$$

Now, if we define the value of the integral to be $F(y)$,

$$F(y) \equiv \int_0^y P(y) dy \tag{3.39}$$

then

$$x = F(y) \tag{3.40a}$$

$$y = F^{-1}(x), \tag{3.40b}$$

where $F^{-1}(x)$ is the inverse of the function $F(x)$, not its reciprocal, and y is understood to be a function of x in Eq. (3.40b).

In our example, if y is to be distributed according to an exponential

$$P(y)\ \mathrm{d}y = \exp(-y)\ \mathrm{d}y, \tag{3.41}$$

over the range $0 \leq y < \infty$, then y can be generated from

$$y = -\ln(x), \tag{3.42}$$

where $\ln(x)$ is the natural logarithm of x, and x is uniformly distributed over $0 < x \leq 1$.

Another frequently-used distribution is the normal distribution

$$P(y)\ \mathrm{d}y = (2\pi)^{-1/2} \exp(-y^2/2)\ \mathrm{d}y, \tag{3.43}$$

over the range $0 \leq y < \infty$. The so-called *Box-Muller* method for generating random deviates with a normal distribution is

$$y = [\ -2 \ln(x_1)\ ]^{1/2} \cos(2\pi x_2) \qquad (3.44a)$$

or

$$y = [\ -2 \ln(x_1)\ ]^{1/2} \sin(2\pi x_2) \qquad (3.44b)$$

where both x_1 and x_2 are uniformly distributed over $0 < x \leq 1$. A fast implementation of Eq. (3.44a) and (3.44b) is given by Press *et al.* (1992).

There are many more distributions that arise in simulations beyond the two that we have discussed. A few points to note:
1. Values in a restricted region of y may be generated by changing the integration range in Eq. (3.38) to $y_1 < y < y_2$ while keeping the probability distribution properly normalized.
2. For some distributions, Eq. (3.38) must be integrated numerically. In such a case, it may be more efficient to generate and store an array of y-values in a separate routine that numerically integrates Eq. (3.38) once, and then randomly select from the array during the execution of the code.

References

S. K. Park and K. W. Miller, *Communications of the A.C.M.* **31**:1192-1201 (1988).

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd edition (Cambridge Univ. Press, 1992) Chap. 7.

L. Schrage, *A.C.M. Transactions on Mathematical Software* **5**:132-138 (1979).

3.8 Project 3 - Phantom chain

The configurations of linear chains are a good starting point for many simulation studies. Although the system can be studied using equations-of-motion techniques, the random motion approach that we take is more robust but a little slower. The algorithm for interaction among the chain elements is set up in a such a way as to have no energy scale, with the result that the "importance sampling" aspect of true Monte Carlo studies can be deferred to Chapter 4.

The chains investigated in this project are permitted to intersect themselves, and are called phantom chains. The chain configurations are equivalent to random walks. Now, there are more direct ways of generating a set of random walks than the chain simulation that we perform here, but coding a somewhat grand simulation of random walks in this project makes the next project (Chapter 4) much simpler.

Physical system

The system to be simulated is a single chain that can move in three dimensions. Each vertex on the chain is moved randomly subject to a step-function potential with neighboring elements on the chain. The process is repeated at least 10^6 times per vertex so that the chain explores the full configuration space available to it.

Simulation parameters

The number of vertices on the chain is defined as N and there are $n_{\mathrm{seg}} = N - 1$ segments on the chains. The interaction potential between nearest-neighbor chain vertices is

$$V_{\mathrm{nn}}(r) = 0 \qquad \text{for } a \le r \le \sqrt{2}\, a \qquad (3.45)$$

$$V_{\mathrm{nn}}(r) = \infty \qquad \text{for } r < a \text{ or } r > \sqrt{2}\, a \qquad (3.46)$$

In other words, the parameter a is the distance of closest approach between neighboring vertices (*i.e.*, the hard core diameter). The average intervertex separation b along the chain should be approximately

$$b = (1 + \sqrt{2})a / 2 \qquad (3.47)$$

In other words, there is both a minimum and maximum length to the bond joining neighboring vertices. Note that $\langle |r| \rangle / a = (9/4) / (2\sqrt{2} - 1) = 1.23$ for radial vectors distributed randomly in three dimensions between a and $\sqrt{2}a$.

Code

1. Choose a value for the number of vertices N between 10 and 30.

2. For simplicity, start the chains on a straight line, with average separation between neighboring vertices of $b \sim 1.2a$. Unfortunately, it takes a long time for a chain to relax from this initialization. If you have the time or the inclination, choose a circular shape for the initial chain configuration.

3. Use V_{nn} from (3.45) and (3.46) for the potential between neighboring vertices. The code will run more efficiently if you avoid square roots and use r^2 rather than r, in evaluating V_{nn}:

$$V_{nn}(r) = 0 \quad \text{for } a^2 \leq r^2 \leq 2a^2 \tag{3.48}$$

$$V_{nn}(r) = \infty \quad \text{for } r^2 < a^2 \text{ or } r^2 > 2a^2 \tag{3.49}$$

4. Try to move each vertex in turn, in each of the x, y and z directions. Place a limit on the maximum move in a given direction of +/- ds, where $ds = a/10$ is a typical choice. If ds is made too large, then the self-avoidance algorithm of the Project 4 will not work.

5. If the attempted move violates the constraints (3.45) - (3.46), then reject the move and try to move the next vertex on the chain.

6. Successive chain configurations are highly correlated, and it takes some time for a chain to "forget" its recent history. Hence, many moves must be made between each statistically independent chain configuration. A typical time scale for relaxation is the Rouse time

$$t_R = N^2 / (ds/a)^2, \tag{3.50}$$

where t_R is measured in sweeps. Each sweep isr a total of N trial moves, one for each vertex.

Analysis

1. Don't analyse all chain configurations, because they are not statistically independent. Analyse only a set of configurations separated by t_R steps per vertex and ignore the rest. Further, throw away the first 10 t_R configurations, since they follow the evolution of the chain as it relaxes from its highly unlikely initial configuration of a straight line.

2. Pay special attention to the end-to-end distance r_{ee}. Perform an ensemble average over 100-200 configurations (each separated by t_R steps) to calculate

(i) $\langle r_{ee} \rangle$

(ii) $\langle r_{ee}{}^2 \rangle$

(iii) $\langle r_{ee}{}^2 \rangle - \langle r_{ee} \rangle^2$

Report

Your report should include the following elements:
•a description of the scaling law expected for an ideal chain
•an outline of your code
•your data for $<r_{ee}^2>$ as a function of $n_{seg} = N - 1$ (include all of part 2 analysis, above)
•an analysis of the combined data from the whole class to extract the scaling exponent ν in $<r_{ee}^2> \sim N^{\nu}$.
•a copy of your code

Programming hints

The projects of this course are arranged so that codes are successively built from one week to the next. It is particularly important for diagnosing errors that codes be kept as simple and transparent as possible. The author's code for this problem is less than 100 lines long, including analysis and (circular) initialization, and has only three functions. Schematically, the code looks like:

```
/* global variables for x[i] ,y[i] ,z[i], N */
void initial(void);        /* initialization */
void moves(void);          /* put one complete sweep over vertices inside moves */
long seed;                 /* make the seed global */
float ran0(long *idum);

void main(void) {
  int j,k;
  initial();               /* include an initial value for seed */
  for(j=0; j<100; j++) {
    for(k=0; k<5000; k++) moves();
    /* analysis of current configuration */
  }
  /* construct averages */
  /* write out analysis */
}
```

In the next project (4), you:
•will add another function to generate neighbor lists
•will add more elements to **moves**.
In the following weeks, you will add periodic boundary conditions and system size rescaling, which will also require functions. The bottom line is that within a few weeks, your basic code will have functions aplenty, all manipulating the same data **x**, **y** and **z**.