A numerical approach to anomalous diffusion on a random comblike structure
TFY4235 Computational Physics Final Exam

Simen Mikalsen Martinussen

August 15, 2013

Abstract

In this exercise we perform a numerical experiment on anomal diffusion of random walkers on a comblike structure. The model does not appear to have been explored experimentally in previous literature. For 132 different comb teeth lengths, we have made RMS measurements of the displacement of 200000 random walkers. The results were found to disagree with previous theoretical work[2]. Anomal diffusion was observed, but to a lesser degree than predicted.

Contents

1 Introduction

In this paper, we discuss a model for anomalous diffusion introduced by Havlin et al.[2] In this model, a random walker moves along a comb-like structure where the length of the teeth is determined stochastically. We numerically examine the relationship between the walked distance and the probability distribution determining tooth length. The goal is to investigate Havlin's expected value for the diffusion exponent and determine if it is correct or not. If it is not, a better expression will be found.

2 Theory

In this section we describe the model system and introduce the algorithms used.

2.1 The model system

The basis of this system is a comb-like structure. It consists of a backbone of an infinite number of nodes $i-1, i, i+1$ and so on, equally spaced by a distance ξ. For simplicity, we will set distance between any two connected nodes equal to 1. From each of these nodes, a side chain is connected. The number of nodes L in each side chain is determined by a probability distribution that for large values of L behaves as

$$P(L) = 1 - L^{-\gamma}, \gamma > 0 \tag{1}$$

We will use this distribution without regard for the requirement that $L \gg 1$. Note that this allows for zero-length side chains, especially for large values of γ. This is interpreted as a backbone node with no side chain. The side chain nodes are denoted $j-1, j, j+1$ and so forth, the lowest allowed value being 0.

2.2 Procedure

A random walker is introduced to the system at node $i = 0$. At any node, it will move to either connected node with equal probability. Thus, when sitting on the backbone with a side chain underneath, it has $1/3$ chance of going either right, left or down. When it is on the side chain or on a spine node without a side chain, it has equal odds of going in either direction, and at the end of a side chain, there's nowhere to go but up.

The walker's state is thus determined by the coordinates (i, j). At $(i, 0)$ it is at the backbone, and at $(i, L(i))$ it is at the end of a side chain, $L(i)$ being the length of the side chain connected to backbone node i. The state after n steps have been made is (i_n, j_n).

The principal topic of this paper is the expected behavior of i_n. To investigate this, a large number of walkers are allowed to roam along the comb. As the probability of moving in either direction is equal to the other, $\langle i_n \rangle$ must be equal to 0. We thus decide to investigate

$$r_n = \left\langle i_n^2 \right\rangle^{1/2} \tag{2}$$

A fundamental result in analyses of diffusion on simple one-dimensional structures with no potential is[4]

$$r_n = \left\langle i_n^2 \right\rangle^{1/2} \sim n^{1/2} \tag{3}$$

This is termed *normal diffusion.* Havlin *et al.* have investigated the comb system theoretically, using two different mean field approaches. In their analyses, they have taken a statistical approach looking at the expected waiting time between two visits to a node. At $\gamma = 1$ there is a discontinuity, and they draw the conclusion

$$r_n = \left\langle i_n^2 \right\rangle^{1/2} \sim n^{1/d_\omega}, d_\omega = \begin{cases} \frac{4}{1+\gamma}, & 0 < \gamma < 1 \\ 2, & \gamma \geq 1 \end{cases} \tag{4}$$

The case where $d_\omega \neq 2$ is called anomal diffusion.

3 Implementation

This section discusses the setup of the numerical experiment and walks the reader through the structure of the computations.

3.1 The setup

The comb structure is represented by a one-dimensional array of integers of a predetermined length. Each integer represents the length of the connected side chain, 0 being no side chain. The side chains are generated as follows: For each element in the array, a random $\rho \in [0, 1]$ is chosen. L is then set to the integer component of

$$\rho^{-1/\gamma} - 1 \tag{5}$$

Three examples of generated combs are shown in figure 1.

The walker is represented by two integers x and y, representing the walker's position along the comb and along the sidechains. Initially, y is equal to 0 and x is equal to the number of nodes in the chain divided by two. That is, the walker is placed in the middle of the comb.

The number of iterations is set to one half of the length of the array. This is to ensure that the walker will never move outside the comb, in the case that it would move in the same direction at every step. For a comb with a side chain at every node and 1000 iterations, the chance of this event is $1/3^{1000} \approx 1/10^{477}$, but the simulation is so computationally cheap that taking this precaution is viable. This brings to mind the following quote:

"*In testing primality of very large numbers chosen at random, the chance of stumbling upon a value that fools the Fermat test is less than the chance that cosmic radiation will cause the computer to make an error in carrying out a "correct" algorithm. Considering an algorithm to be inadequate for the first reason but not for the second illustrates the difference between mathematics and engineering.*" (Abelson and Sussman[1])

Another option we considered was adding nodes to the comb on the fly whenever the walker would move onto an unexplored node. However, this idea was discarded as it would introduce complexity and more room for bugs, and because creating a large comb is acceptably cheap, requiring only one call to rand and pow for each node. We valued simplicity and robustness far higher than efficiency in this project.

3.2 Procedure

For each iteration, the process of moving the walker can be described by a series of choices. Each time there is a choice between several directions, there is an equal probability of moving in each direction. This is illustrated in figure 2.

3.3 Parameters and analysis

The number of γ values was set to a total of 132. These were generated by declaring the lowest γ to be 0.1, and letting the next be 3% higher and so on, up to a highest value of 5. We considered spacing the values evenly, but this plan was discarded in favor of the exponential increase. The area $0 < \gamma < 1$ is most interesting, and the exponential increase favored it, while still spending some resources on the high γ limit, where normal diffusion was to be expected.

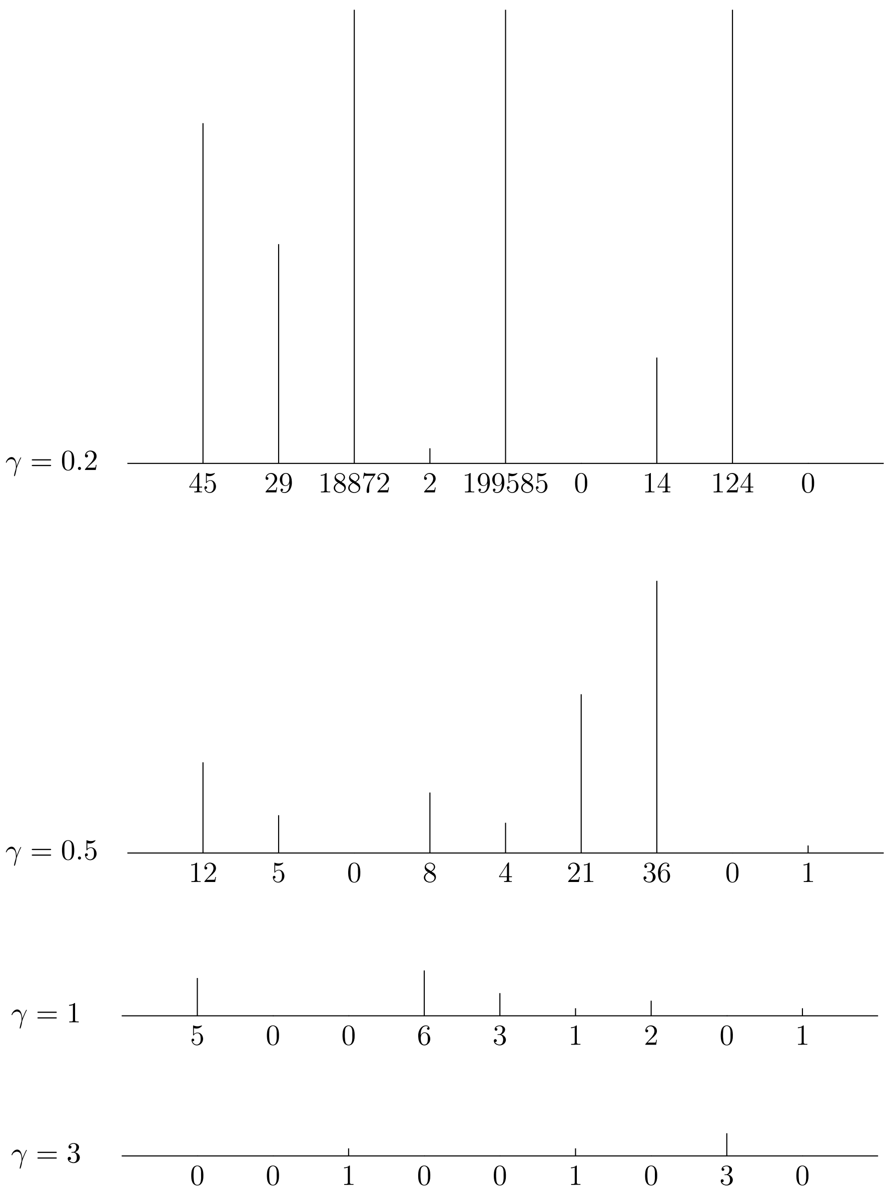

Figure 1: 9 nodes in sample combs with different γ.

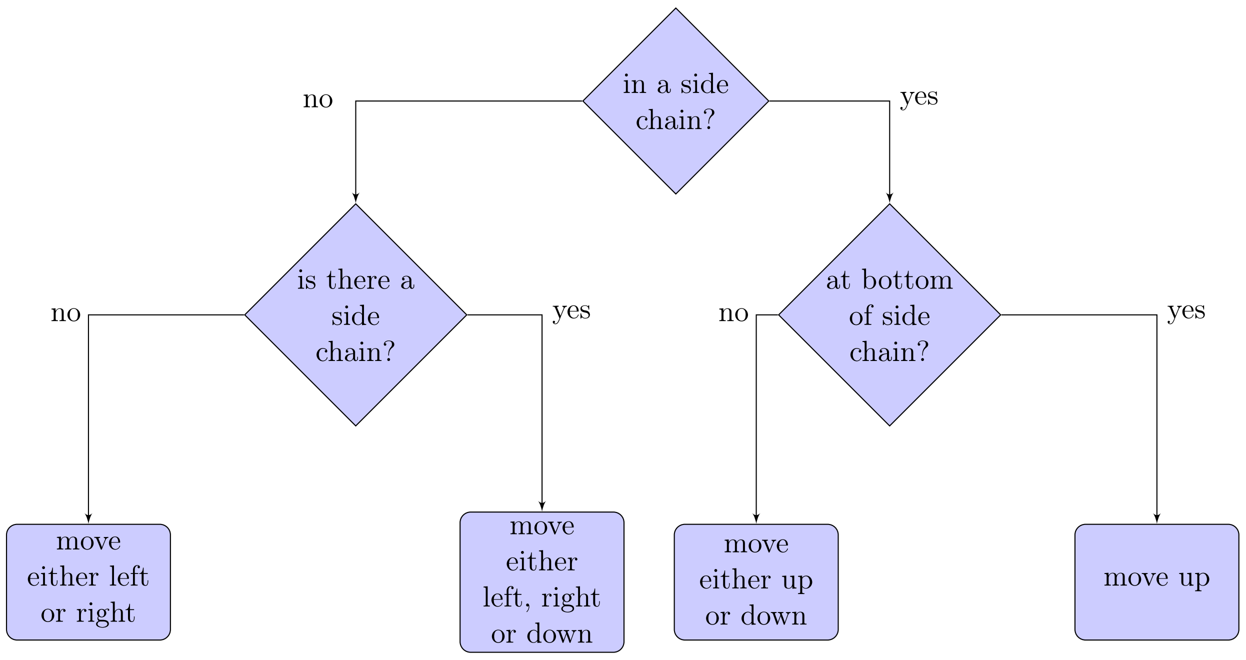

Figure 2: Flow chart illustrating the walker-moving algorithm.

We also considered reducing the spacing between the values, but after reviewing the results of the chosen parameters, we deemed them good enough. In this scheme, 78 out of 132 values were between 0 and 1, with a further 24 under 2.

For each value of γ, 20000 combs were generated. For each comb, 10 walkers were used. While this ratio of combs to walkers may seem excessive, preliminary findings with a more or less equal comb to walker ratio showed bumps and waves in the expected distances walked. These aberrations were not consistent between different γ values.

We suspected that many walkers were falling into the same long side chain or clump of side chains, thus behaving similarly to each other rather than diffusing independently. The solution was to create a new environment very often, so that the same trap would be observed few times in the results. This approach was most effective. Placing the walkers at random nodes along the comb instead of placing them at the same starting point was considered. However, this was judged as interfering with the minimalistic approach to the code, and although a great number of optimizations were not even tested, the program was more than fast enough. For each value of γ, the simulation took about 12 seconds on NTNU's computer `caracal.stud.ntnu.no`.

The number of steps for each walker was set to 1000. Correspondingly, the size of the system was set to 2000 nodes along the spine. This was deemed sufficient early in the experiment. It is obvious from figure 4 that the slope of the different curves are approximately equal at the endpoint, so the areas showing the greatest differences between the curves are well covered.

The observant reader may note that although the `C++` program prints the data by line, the `MATLAB` script works on it by column. This is because the data files were transposed with an `awk` command. I did not write it myself.

4 Results

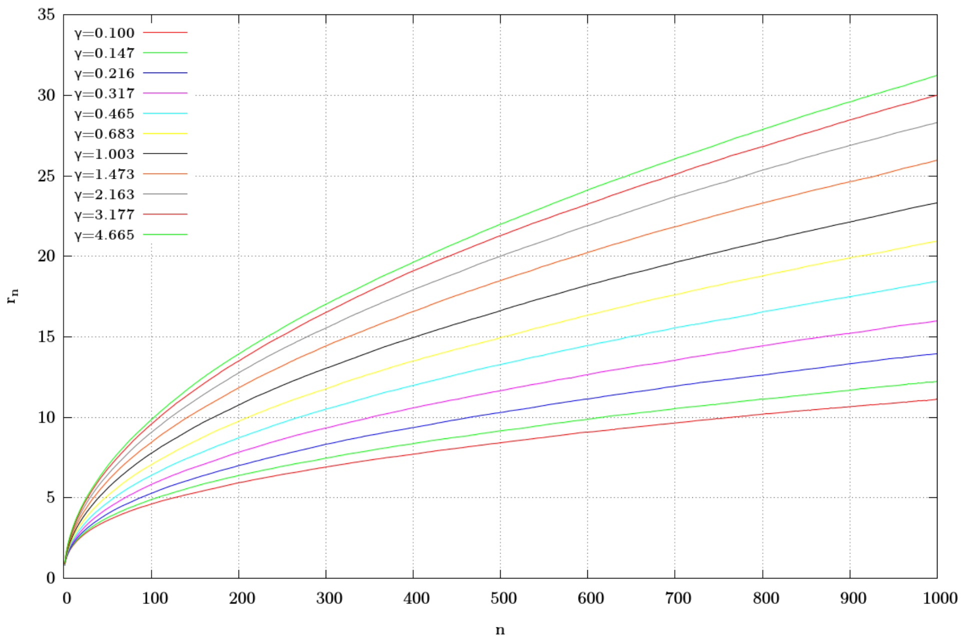

Figure 3: 11 plots of r_n sampled over every n from 1 to 1000

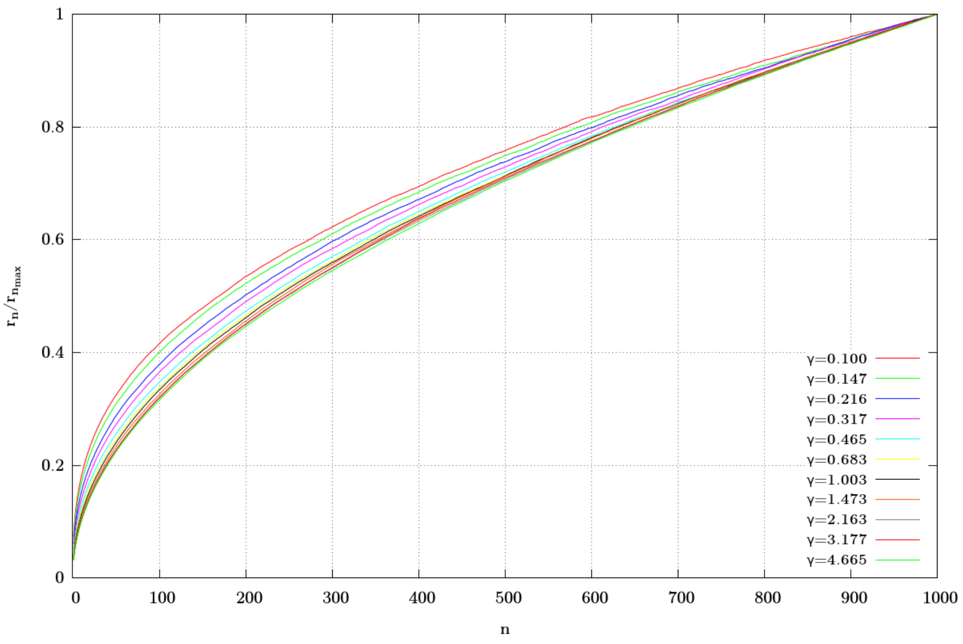

Figure 4: 11 plots of r_n sampled over every n from 1 to 1000, normalized to their endpoints

Figure 3 shows the RMS displacement of the walkers over time. It demonstrates how longer side chains causes the walkers to stray from the spine and not go very far in the same number of iterations as their cousins on the almost side chain-less combs. However, the difference in the curvature of the paths is better demonstrated in figure 4. Note that the paths that are at the bottom in figure 3 are at the top in figure 4.

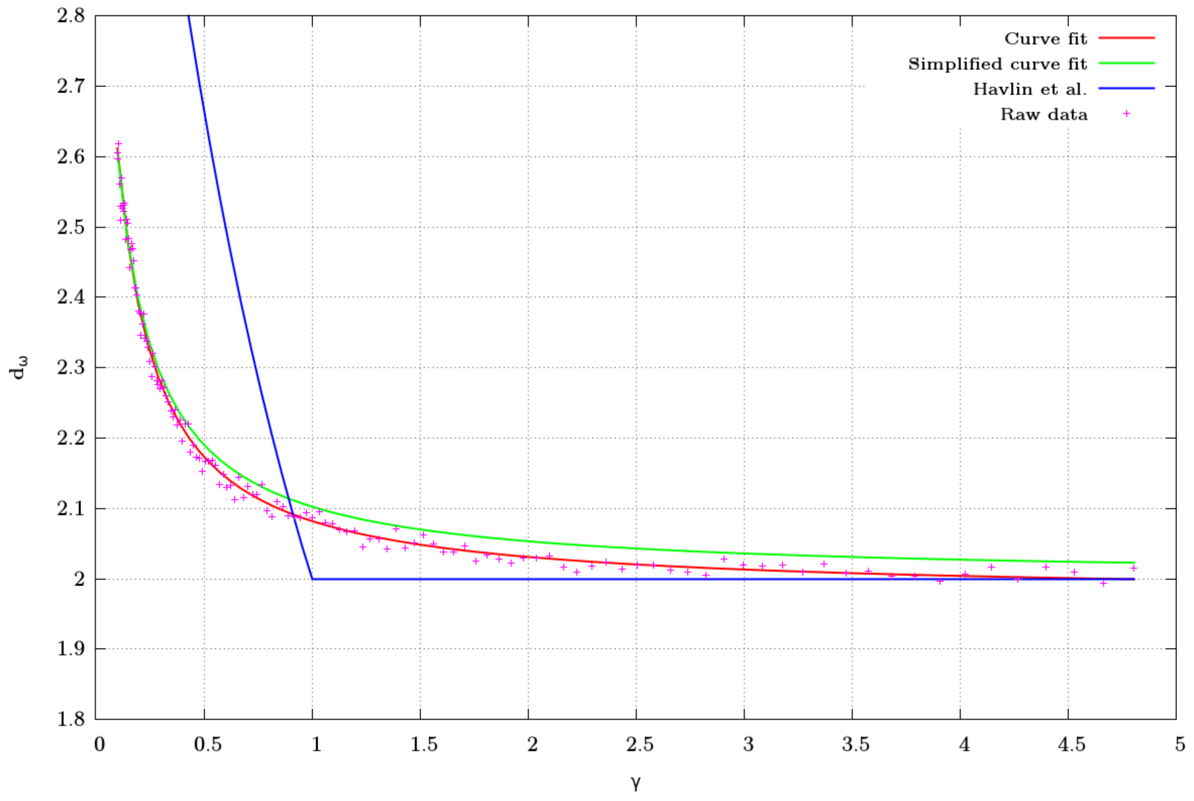

Figure 5: The measured diffusion exponents as a function of γ, along with Havlin's calculated values

Each of the paths was analyzed by the script `fitter.m` shown in the appendix. This script uses the builtin `MATLAB` function `fit()` in order to fit the supplied data to a function of the form $a \cdot x^b$. In symmetry with equation 4, $1/b$ is of interest, as it is the diffusion exponent d_ω. Figure 5 shows the calculated values of d_ω as a function of γ, along with a fitted curve and Havlin's expected curve. A simplification of the fitted curve is also included. The diffusion exponents were found to coincide well with the curve

$$d_\omega = \frac{0.1146}{0.08028 + \gamma} + 1.976 \tag{6}$$

It is tempting to approximate this expression by the much simpler $\frac{1}{\frac{3}{4}+9\gamma} + 2$.

5 Discussion

The primary purpose of the experiment was to test equation 4. Our results, most clearly shown in figure 5, contradicts the work of Havlin *et al.* In our view, they significantly overestimate d_ω for $\gamma < 1$, and are overzealous in reducing it to a constant 2 everywhere else. In their defense, equation 5 is only supposed to be a good approximation for $L \gg 1$, while we have here used it even for mostly-empty combs. As no better expression of the probability distribution for large γ has been given, we must admit that the question is unclear when approaching this limit. However, the results found herein indicate anomalous diffusion even at $\gamma = 1$. Our sample comb shown in figure 1 shows that at this point the combs are far from side chain free.

The most interesting disagreement occurs in the $\gamma \to 0$ limit. In Havlin's expression, $d_\omega = 4$, while it has here been found to be somewhat lower at 3.4. Do note that no measurements were made below $\gamma = 0.1$, so the γ^{-1} behavior might well break down at this point. In that case, applying linear regression to the curve approximates $d_\omega = 2.8$, significantly lower than the other estimates. The choice of this limit was somewhat arbitrary, and it would have been interesting to experiment further. We regret that this realization was found too late for there to be time to re-do the experiment and analysis.

The approximated curve in figure 5 is very tempting as it is much simpler, but deviates by a few percent from the exact solution for large γ. If the comb model is mostly interesting for systems with significant anomalous diffusion, it might be worthwhile to use the simplified expression and declare, as Havlin does, diffusion to be normal above $\gamma = 1$.

6 Conclusion

We have shown the claims made by Havlin *et al.* to be incorrect. The dependency between γ and d_ω has been shown to be

$$d_\omega = \frac{0.1146}{0.08028 + \gamma} + 1.976 \tag{7}$$

which can be simplified to

$$d_\omega = \frac{1}{3/4 + 9\gamma} + 2 \tag{8}$$

for low γ.

References

[1] Harold Abelson and Gerald J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, USA, 2nd edition, 1996. ISBN 0262011530.

[2] S Havlin, Je Kiefer, and Gh Weiss. Anomalous diffusion on a random comb-like structure. *Physical Review A*, 36(3):1403–1408, Aug 1 1987. ISSN 1050-2947. doi: {10.1103/physreva.36.1403}.

[3] Jonathan Leffler. unix - transpose a file in bash - stack overflow, Nov 2009 . URL `http://stackoverflow.com/questions/1729824/transpose-a-file-in-bash`.

[4] P. Nelson. *Biological Physics (Updated Edition)*. W. H. Freeman, 2003. ISBN 9781429280754.

A Program listing

A.1 main.cpp

```
#include <iostream>
#include <random>
#include <time.h>
#include <vector>

using namespace std;

int main(){
  int maxSteps = 1000;
  int noWalks = 10;
  int noCombs = 20000;

  vector<int> comb;
  vector<double> distances(maxSteps,0);
  int xPos, yPos = 0;

  int length = maxSteps*2;
  int starting = length/2;

  srand(time(NULL));
  //Print line numbers to appease Gnuplot
  for (int i = 1; i < 1001; i++){
    cout << i << ' ';
  }

  //Cycle through gammas
  for (double gamma = 0.1; gamma < 5; gamma *= 1.03){
    double totWalked = 0;

    //Generating Comb
    for (int combNo = 0; combNo < noCombs; combNo++){
      comb.clear();
      for (int i = 0; i < length; i++){
        comb.push_back (pow((float)rand()/(float)RAND_MAX, -1/gamma) - 1);
      }

      for (int walkNo = 0; walkNo < noWalks; walkNo++){
        //Initializing walk
        xPos = starting;
        yPos = 0;
        //Single steps
        for (int step = 0; step < maxSteps; step++){
          if (yPos){
            if (yPos == comb [xPos]){
              yPos--;
            }else{
              yPos += (rand()%2 - 0.5) * 2;
            }
          }else{
            if (comb [yPos]){
              if (not (rand()%3)){
                yPos++;
              }else{
                xPos += (rand()%2 - 0.5) * 2;
              }
            }else{
              xPos += (rand()%2 - 0.5) * 2;
            }
          }
          //Add up the square of distances moved for each step
          distances[step] += pow(xPos - starting,2);
        }
      }
    }
    //Print the RMS distances by step before next gamma
    for (int i = 0; i < 1000; i++){
      cout << pow(distances[i]/(noCombs*noWalks),0.5) << ' ';
      distances[i] = 0;
    }
    cout << endl;
  }
  return 0;
}
```

A.2 fitter.m

```
A = importdata('transposeddata5.dat');
A = A(1:1000,:);

options = fitoptions;

for i = 2:133
    gammas(i-1) = 0.1*1.03^(i-2);
    this =  fit(A(:,1),A(:,i),'a*x^b');
    disp(i);
    fits(i-1) = this.b;
end

invfits = 1./fits;
plot(gammas, fits);
```

A.3 transposer (bash script)

```
awk '
{
    for (i=1; i<=NF; i++)  {
        a[NR,i] = $i
    }
}
NF>p { p = NF }
END {
    for(j=1; j<=p; j++) {
        str=a[1,j]
        for(i=2; i<=NR; i++){
            str=str" "a[i,j];
        }
        print str
    }
}' inputfile > outputfile
```

Retrieved from[3].