LAB REPORT: LAB 1-2
Simulation of Brownian motion
MODELING OF PHYSICAL SYSTEMS

Patryk Gaczyski

Monday 12th March, 2018

Abstract

This report consist specification for provided solution to given problem of simulation and analysis of Brownian motion. It describes tools used to properly simulate and visualize this physical phenomenon (MATLAB) as well as statistical operations needed to prove correctness of developed model such as autocorrelation or square mean of displacement.

1 Aim

The aim of this laboratory was to simulate and analyze Brownian motion and its properties using various selected methods.

2 Methods

For the laboratory purpose various tools and methods were selected to achieve good simulation results and easy in depth analysis.

2.1 Software

To simulate, compute and visually represent Brownian motion MATLAB software were involved.

2.2 Statistical methods

- A random walk model was used for good approximation of Brownian motion;
- A mean square of displacement was used to measure spatial extent of particle;
- A autocorrelation was used to show property named as “self similarity”;
- A histogram of particle position to illustrate the time evolution of particles density.

3 Results

3.1 simulating Brownian motion

To simulate Brownian motion, matlab function has been developed. The function does take 3 arguments, as follows: number of dimensions to carry simulation in, number of particle movements to simulate and number of particles to use.

Listing 1: Function that simulates Brownian motion

```
function [] = brownian(dims, num_of_movements,
    num_of_particles)
    figure;
    for i = 1:num_of_particles
        % initiaze variables
        N = num_of_movements;
        X(1) = 0;
        Y(1) = 0;
        Z(1) = 0;
        % generate random vectors
        for i=2:N
            X(i) = X(i-1) + randn();
            Y(i) = Y(i-1) + randn();
            Z(i) = Z(i-1) + randn();
        end
        % plot the result
        if dims == 1
            plot(Y);
            ylabel('x coordinate');
            xlabel('timestamp');
        elseif dims == 2
            plot(X,Y);
            xlabel('x coordinate');
            ylabel('y coordinate');
        elseif dims == 3
            plot3(X,Y,Z);
            xlabel('x coordinate');
            ylabel('y coordinate');
            zlabel('z coordinate')
        end
        hold on;
    end
end
```

Results of this function call for different sets of arguments are shown on Figure 3.1

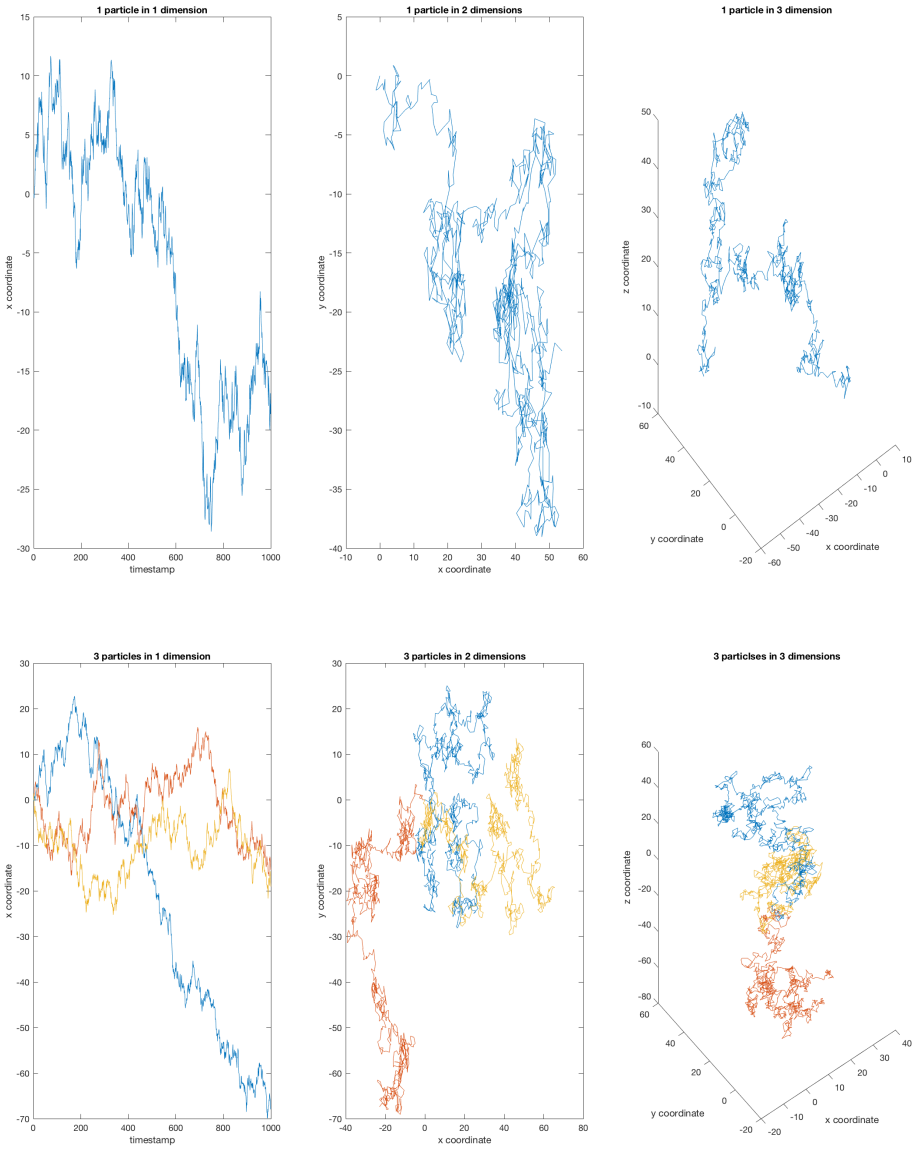

Figure 3.1: Brownian motion simulation for different number of dimensions and number of particles

3.2 Square mean of displacement and time

Mean square displacement is a measurement of how far particle goes over time from previous position (in this case). To calculate that value another MATLAB function has been written. It takes similar set of arguments as previous function and plots mean square displacement over time.

Function `square_means` can be found on listing 2

Listing 2: Function that calculates square mean of displacement

```
function [] = square_mean(dims, num_of_movements,
    num_of_particles)
    N = num_of_movements;
    num_of_parts = num_of_particles;
    x = zeros(num_of_parts, 1);
    y = zeros(num_of_parts, 1);
    z = zeros(num_of_parts, 1);
    for i=2:N
        x = [x x(:,i-1)+randn(num_of_parts, 1)];
        y = [y y(:,i-1)+randn(num_of_parts, 1)];
        z = [z z(:,i-1)+randn(num_of_parts, 1)];
    end

    if dims == 1
            plot(sum(x.^2)/num_of_parts');
            xlabel('timestamp');
            ylabel('Square mean of displacement in 1d');
            title(['1 dimension for ' num2str(num_of_parts)
                ' particles']);
        elseif dims == 2
            plot(sum(x.^2 + y.^2)/num_of_parts');
            xlabel('timestamp');
            ylabel('Square mean of displacement in 2d');
            title(['2 dimensions for ' num2str(num_of_parts
                ) ' particles']);
        elseif dims == 3
            plot(sum(x.^2 + y.^2 + z.^2)/num_of_parts');
            xlabel('timestamp');
            ylabel('Square mean of displacement in 3d');
            title(['3 dimensions for ' num2str(num_of_parts
                ) ' particles']);
    end
end
```

Output for various sets of input parameters are shown on Figures 3.2 and 3.3

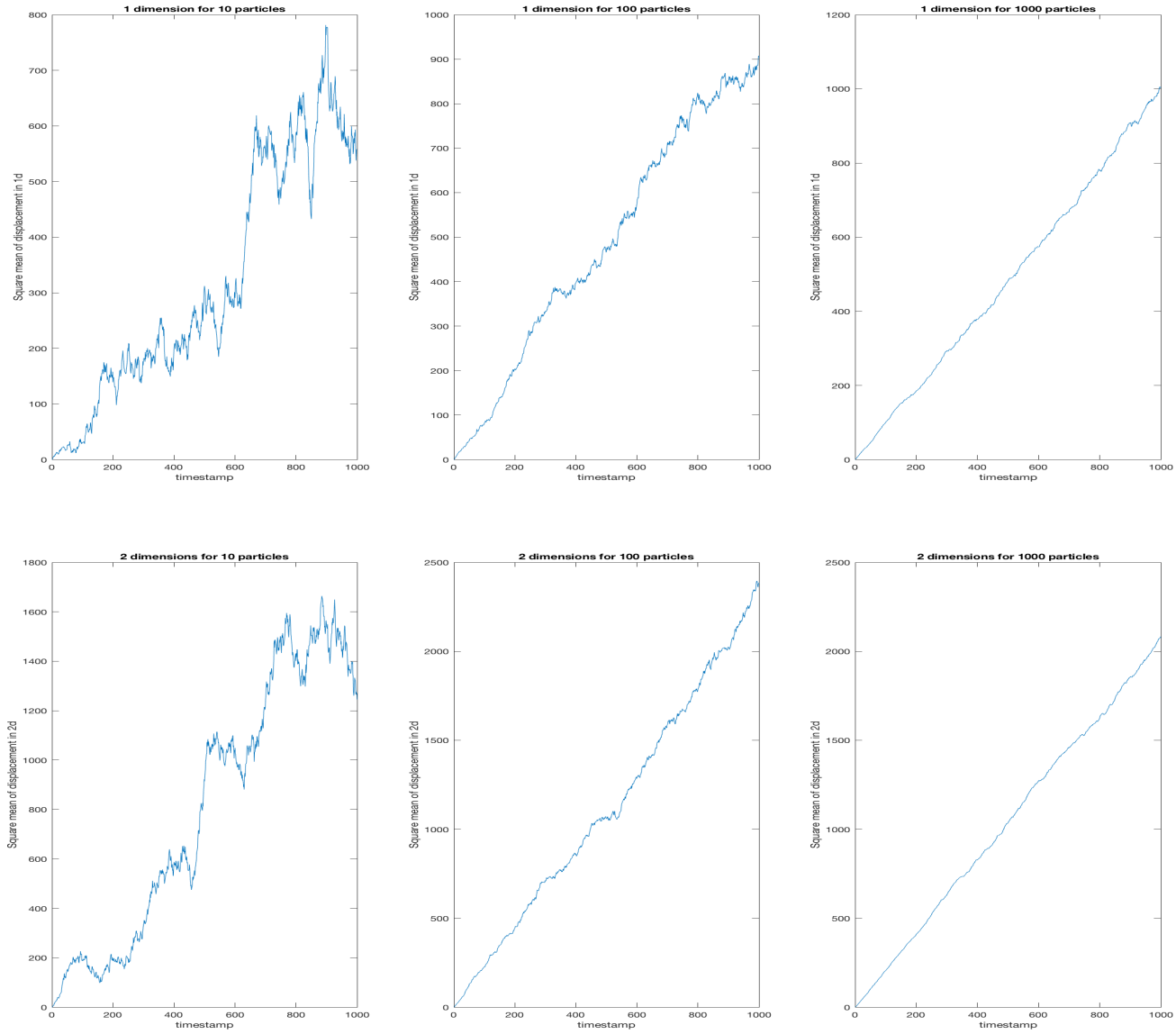

Figure 3.2: Mean square displacement for various number of particles in 1d and 2d

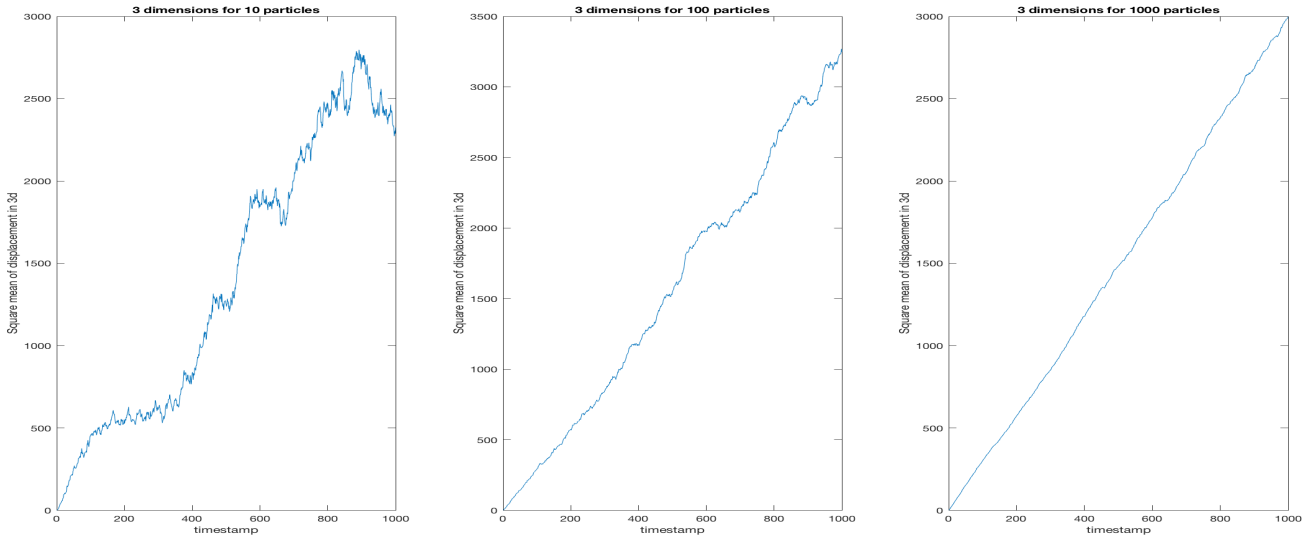

Figure 3.3: Mean square displacement for various number of particles in 3d

From this carts, it is clearly visible that, with growing number of particles, relationship between timestamp and square mean of displacement is getting more and more linear-ish.

3.3 Time evolution of particles density

To demonstrate the particles density evolution over time histogram has been used. However, it is suitable only for 1d and 2d data, so only these two are going to be analyzed. Matlab code used to generate such plots is shown on Listing 3

Listing 3: Function that calculates particles density evolution

```
1  function [] = density(dims, num_of_particles)
2      num_of_parts = num_of_particles;
3      x = zeros(num_of_parts, 1);
4      y = zeros(num_of_parts, 1);
5      z = zeros(num_of_parts, 1);
6
7      for i=2:num_of_parts
8          x = [x x(:,i-1)+randn(num_of_parts, 1)];
9          y = [y y(:,i-1)+randn(num_of_parts, 1)];
10         z = [z z(:,i-1)+randn(num_of_parts, 1)];
11     end
12
```

```
13
14
15      if dims == 1
16              figure;
17              histogram(x);
18              xlabel('x coordinate');
19              ylabel('number of particles');
20      elseif dims == 2
21              figure;
22              histogram2(x,y);
23              xlabel('x coordinate');
24              ylabel('y coordinate');
25              zlabel('number of particles');
26      end
27  end
```

Such code execution gave following results:

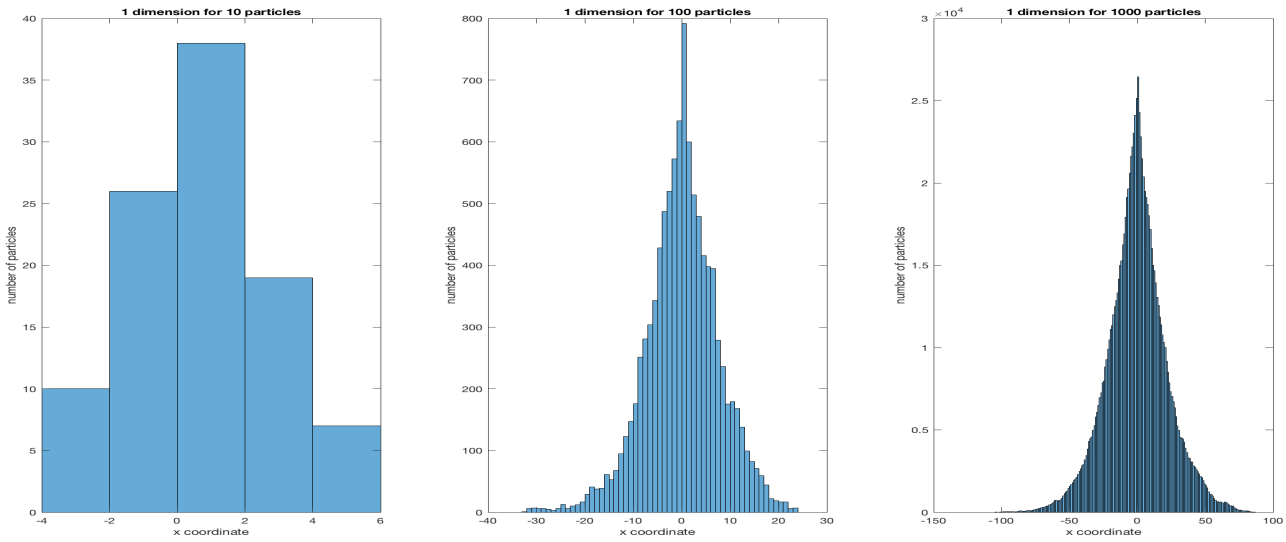

Figure 3.4: Histograms of particle density evolution in 1d

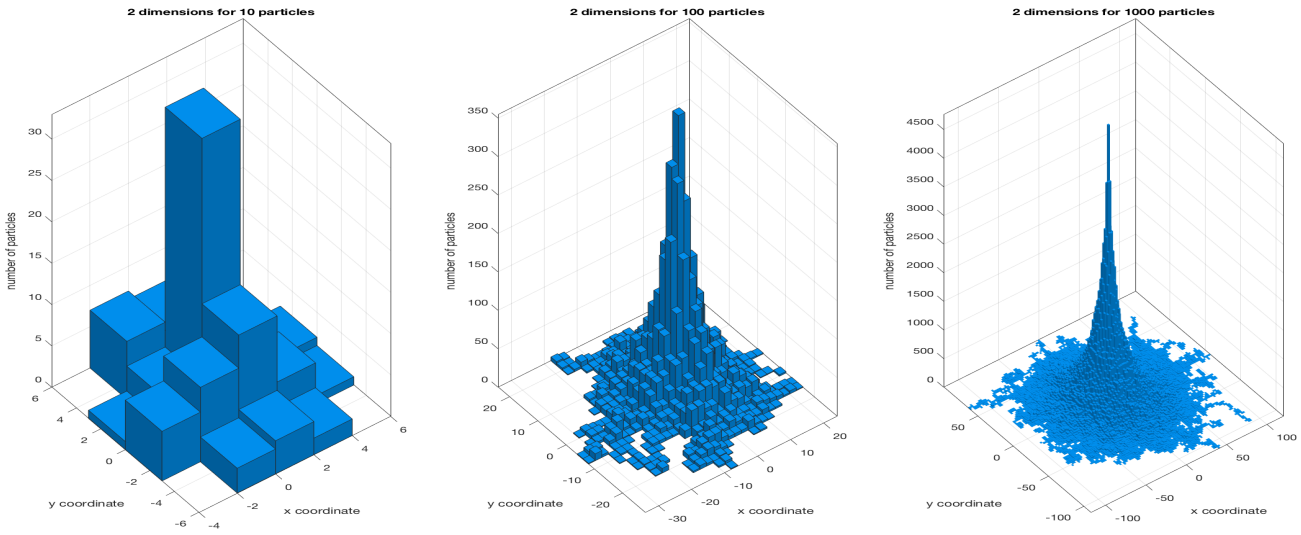

Figure 3.5: Histograms of particle density evolution in 2d

Concluding from given histograms, it can be assumed that this data seek to normal distribution (Gausian distribution) despite number of dimensions taken in to account.

3.4 Autocorrelation property

To inspect autocorrelation property of Brownian motion it can be shown in comparison with autocorrelation of random data. For this purpose, simple MATLAB script were prepared, and can be read at listing 4

Listing 4: Script that outputs autocorrelation for Brownian motion and random vector of data

```
figure;
rc = [];
random_values = rand(1, 1000);
for i = -50:50
    rc = [rc corr(random_values(100:900)', random_values
       (100+i:900+i)')];
end
plot(rc);
xlabel('random position');
ylabel('auto correlation value');
title('Auto correlation of random values');

x = [0];
```

```
13 for i=2:1000
14     x = [x x(i-1)+randn()];
15 end
16
17 bc = []
18 for i = -50:50
19     bc = [bc corr(x(100:900)', x(100+i:900+i)')];
20 end
21
22 figure;
23 plot(bc);
24 xlabel('random walk position');
25 ylabel('auto correlation value');
26 title('Auto correlation of Brownian motion');
```

Output of this script can be found on figure 3.6 and 3.7

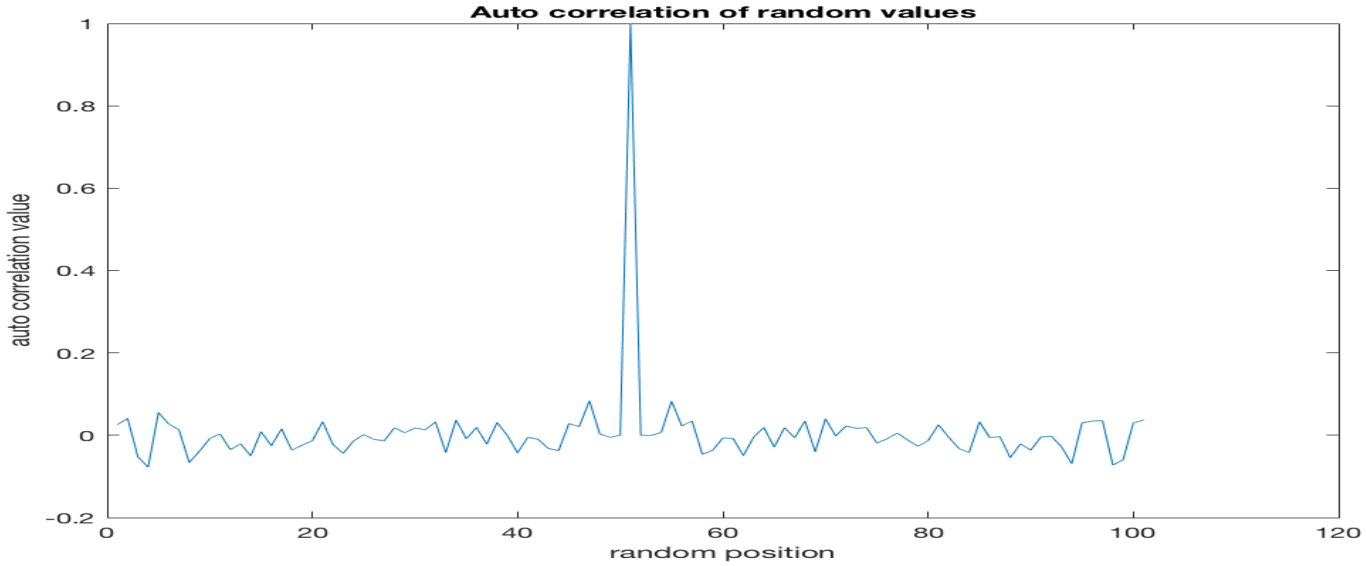

Figure 3.6: Autocorrelation of random vector of data

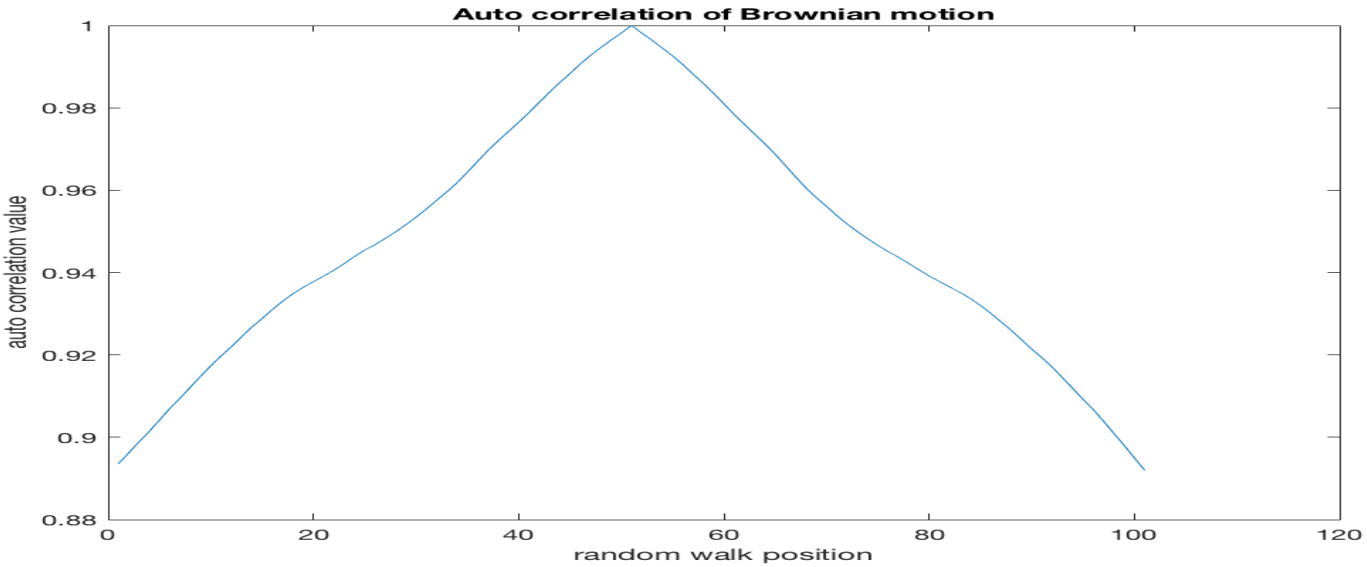

Figure 3.7: Autocorrelation of vector containing Brownian motion data

Comparing these two charts, it is observable that Brownian motion autocorrelation factor is much higher then random data, getting close to 1 and not dropping below 0.88.

4 Conclusion

In this report Brownian motion and its properties was covered. Various matlab scripts were written to illustrate random walk itself, its square mean of displacement, time evolution of particles density and its autocorrelation property. It is called "random" walk, however, as the analysis shown it undergoes some well known phenomenons like normal distribution or self similarity.