PHYS 516: Methods of Computational Physics

ASSIGNMENT 7- RANDOM WALKS

Anup V Kanale

April 12, 2017

# 1 Stock price simulation

This section illustrates the application of MC simulation to a stochastic problem– Stock price in this case specifically.

## 1.1 MC simulations of stock price based on Black-Scholes Model

A program was written in C to perform Monte Carlo (MC) simulations of a stock price, $S$, as a function of time, $t$, assuming that $S$ follows a discrete stochastic equation

$$dS = \mu S dt + \sigma \varepsilon S \sqrt{dt} \tag{1}$$

where $dt$ is the time discretization unit and $\varepsilon$ is a random variable following the normal distribution with unit variance. The code is attached in the appendix at the end of the report.

## 1.2 Temporal variation of stock price

A simulation was run to see the variation of stock price over time. There are 100 random walker (or investors), the starting stock price was chosen as $20.0 and a 14 % per annum ($\mu = 0.14$) expected return with a standard deviation of 20% per annum ($\sigma = 0.20$) was used. Let dt = 0.00274 year ( 1 day). The MC simulation was run for 1 year (365 steps).

The variation of the stock price is as shown in figure 1.2. The matlab script for plotting is included in the Appendix.

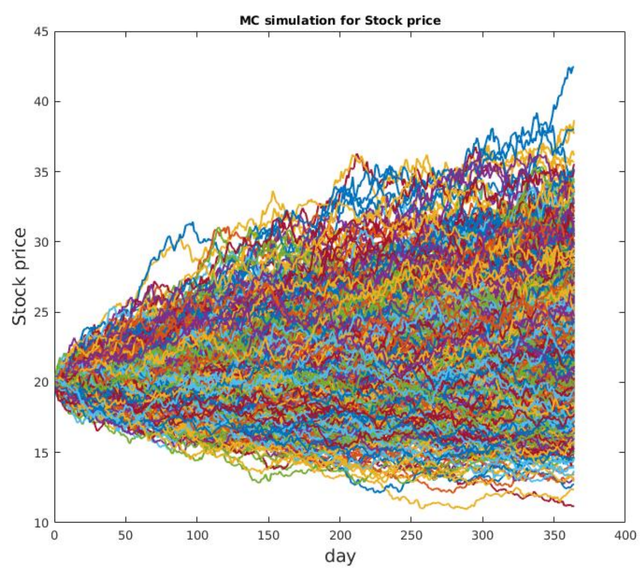


Figure 1: Variation of stock price over the year for 1000 investors

### 1.3 Histogram of stock price

The MC simulations were repeated 1,000 times with the same parameters, but with different random number seeds. The histogram below in figure 1.3 shows the number of random walkers (or investors) at each given price of stock at the end of the year.

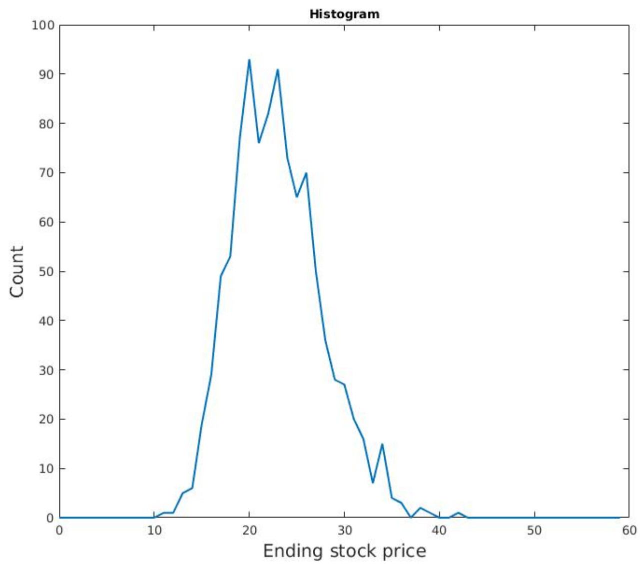


Figure 2: Histogram at the end of the year for 1000 investors

As the number of investors is increased, the histogram gradually becomes a smooth Gaussian, as is seen from figure 1.3

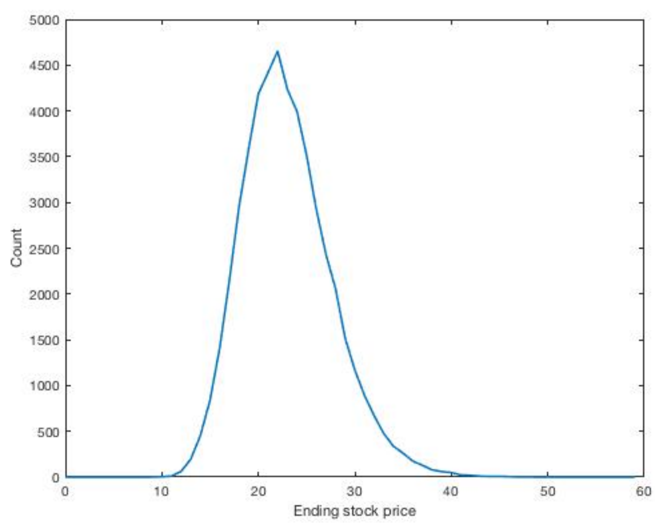


Figure 3: Histogram at the end of the year for 50000 investors

# 2 Quantum Monte Carlo simulation

This section illustrates the quantum Monte Carlo technique. The difference from Part I is that while simulating the electron wave function, in addition to the diffusion term, a birth/death term exists in the Imaginary-time Schrodinger Equation.

## 2.1 Quantum Monte Carlo Simulation of 1D Diffusion problem

A program was written to simulate diffusion using quantum Monte Carlo simulation of a 1D electron wave function, in a harmonic potential given by $V(x) = x^2/2$ (in the atomic unit). The program is attached in appendix A. The energy as a function of MC steps is shown in figure 2.1

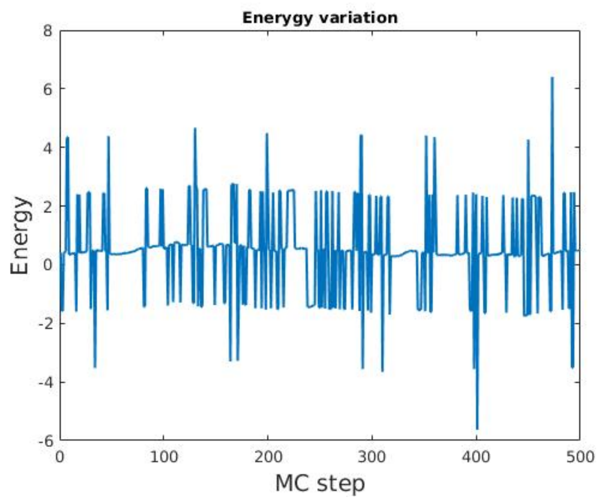


Figure 4: Energy variation with MC step

## 2.2 Histogram

The histogram of random walkers is shown in figure 2.2

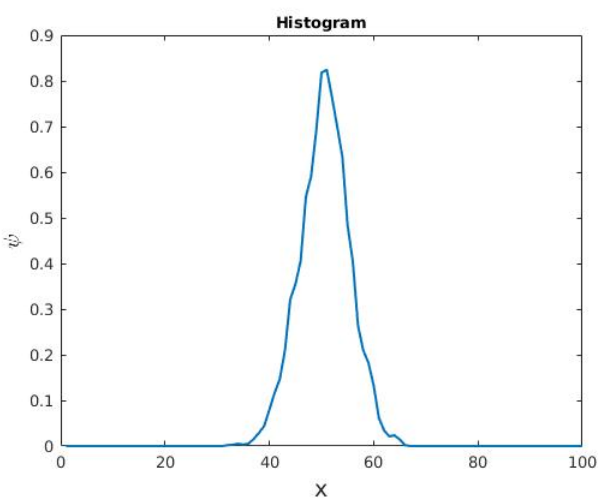


Figure 5: Histogram of random walkers

# A C Programs for simulation

```c
// Program to perform Monte Carlo Simulations of stock-price using the
// Black-Scholes equation. Based on the random walk example as in diffuse.c

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

double rand_normal();

int main(){
  int N; /* Number of walkers */
  int day,investor,k;
  int hist[60];
  double mu = 0.14; // expected return p.a in %
  double dt=0.00274;
  double sigma=0.2; // Standard deviation of returns
  double stockPrice;

  FILE *fp1, *fp2;
  char filename[50];
  char prefix[17] = "stockData_walker";
  char suffix[4] = ".txt";
  char strN[5];

  /* Input parameters */
  printf("Input the number of walkers\n");
  scanf("%d",&N);

  for (k=0; k<60; k++)
    hist[k] = 0;


  for (investor=1; investor<=N; investor++) {
    sprintf(filename, "%s%d%s", prefix,investor,suffix);
    fp1 = fopen(filename, "w");

    stockPrice = 20;
    for (day=1; day<365; day++){
      stockPrice += stockPrice*(mu*dt + sigma* sqrt(dt)*rand_normal());
      fprintf(fp1, "%d %f  \n", day, stockPrice);
    }
```

```
    k = round(stockPrice);
    hist[k] += 1;
    fclose(fp1);
  }


  fp2 = fopen("histogramData.txt", "w");
  for (k=0; k<60; k++)
    fprintf(fp2, "%d %d \n", k, hist[k]);
  fclose(fp2);
}

double rand_normal() {
  double r1,r2,eps;
  r1 = rand()/(double) RAND_MAX;
  r2 = rand()/(double) RAND_MAX;
  eps = sqrt(-2* log(r1))*cos(2*M_PI*r2);
  return eps;
}
```

stockPriceMC.c

```
/*
Quantum Monte Carlo Algorithm
-------------------------------------------------------
1. Place N0 walkers at the initial set of positions xi.
2. Compute the reference energy, Vref = Îči V(xi)/N0.
3. For each walker,
  a) Randomly move the walker to the right or left by a fixed step
      length âĹĘs.
  b) Compute ÎŤV = V(x) âĹŠ Vref and a random number r in the range
      [0, 1]. If ÎŤV > 0 and r <
  ÎŤVÎŤÏĎ, then remove the walker. If ÎŤV <0 and r < âĹŠÎŤVÎŤÏĎ, then
      add another walker at x.
  Otherwise, just leave the walker at x.
4. Compute the mean potential energy (6) and the actual number of
    random walkers. The new
reference potential is given by Eq. (7). The average ãĂĹVãĂĽ is an
    estimate of the ground state
energy.
5. Repeat steps 3âĹŠ4 until the estimates of the ground state energy
    ãĂĹVãĂĽ have reached a steady
state value with only random fluctuations. Average ãĂĹVãĂĽ over many
    Monte Carlo steps to
compute the ground state energy.
*/


```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

// define function prototypes
void initialize();
void walk();
void data();

// Constants
#define N0 50   //initial walkers
#define mcs 500 //MC steps
#define MAXX 200
#define maxpsi 100

double ds=0.1;  //random walk step
double dt=0.01; //time step (=ds^2)

// Variables
int N;      //# of walkers
double Vref; //ref energy

double x[MAXX+1];// position of each random walker
double psi[maxpsi+1]; // histogram

int main() {
  int step;
  FILE *fp1, *fp2;
  double psiSum = 0.0;

  srand((unsigned)time((long *)0)); /* Initialize the random-number
      sequence */

  initialize();
  for(step=1; step<=0.4*mcs; step++){
    walk();     // thermalization
  }

  fp1 = fopen("vRef.txt", "w");
  for (step=1; step<=mcs; step++) {
    walk();
    fprintf(fp1, "%d %f \n",step, Vref);
    data();
  }
  fclose(fp1);

  for (int i=1; i<=maxpsi; i++){
```

```c
    psiSum += psi[i]*psi[i]*0.2;
  }
  psiSum = sqrt(psiSum);

  fp2 = fopen("qmcHistogram.txt", "w");
  for (int i=1; i<=maxpsi; i++){
    psi[i] = psi[i]/psiSum;
    fprintf(fp2, "%d %f \n", i, psi[i]);
  }
  fclose(fp2);

  return 0;
}

// Function definitions
void initialize() {
  int i;
  N = N0;

  // Assign position to N0 random walkers
  for (i=1; i<=N; i++)
    x[i] = -1+2*rand()/(double) RAND_MAX ; //uniform random number # in [-1,1]

  // Calculate reference energy
  for (i=1; i<=N; i++)
    Vref += pow(x[i],2)/2; // = <V>, this is the harmonic potential x_i^2/2
  Vref = Vref/N;
}

void walk() {
  int i;
  int Nin = N;
  double Vavg, dV, Vsum=0.0;

  for (i=Nin; i>=1; i--) { //Going ulta!!
  // Random Walk as in diffuse.c
    if(rand()%2==0) x[i] += ds;
    else x[i] -= ds;

    // Birth or death
    dV = pow(x[i],2)/2 - Vref;
    if (dV<0.0)
      if (rand()/(double)RAND_MAX < -dV*dt){
        N++;
        x[N] = x[i];
        Vsum += 2*pow(x[i],2)/2;} //here V(x_i) = x_i^2/2}
```

```
      else
        Vsum += pow(x[i],2)/2;
    else
      if ( rand()/(double)RAND_MAX < dV*dt){
        x[i] = x[N];
        N--;}
      else
        Vsum += pow(x[i],2)/2;
  }

  Vavg = Vsum/N;
  Vref = Vavg - (N-N0)/(double)(N0*dt);
}

void data(){
  double xshift, binsize;
  int bin;

  binsize = 2*ds;
  xshift = binsize*maxpsi*0.5;
  for(int i=1; i<=N; i++){
    bin = (x[i] + xshift + 0.5*binsize)/binsize;
    psi[(int)bin] += 1;
  }
}
```

quantumMC.c

## B Matlab scripts for plotting

```
clc;
close all; clear all;

N = 50000;
fs = 14; % Font Size
%% Scatter Plots for Stock prices
%------------------------------------------------
for walker=1:N
    filename = strcat('stockData_walker', int2str(walker), '.txt');
    stockData = dlmread(filename);
    day = stockData(:,1);
    price = stockData(:,2);
    plot(day, price, 'LineWidth', 1.5);
    hold on;
    delete(filename);
end

```

```
xlabel('day','FontSize', fs); ylabel('Stock price','FontSize', fs);
title('MC simulation for Stock price', 'FontSize', fs-4);
hold off;

%% Histogram data
%---------------------------------------------------
histData = dlmread('histogramData.txt');
investor = histData(:,1);
count = histData(:,2);
figure();
plot(investor, count, 'LineWidth', 1.5);
delete('histogramData.txt');

xlabel('Ending stock price','FontSize', fs); ylabel('Count','FontSize', fs);
title('Histogram', 'FontSize', fs-4);
```

stockPricePlots.m

```
% Quantum Monte Carlo Plots
%----------------------------------
clc;
close all; clear;
fs = 14; % Font Size

% Energy Plot
%------------------------------------
energyData = dlmread('vRef.txt');
mcstep = energyData(:,1);
energy = energyData(:,2);
figure(1);
plot(mcstep, energy,'LineWidth', 1.5);

xlabel('MC step','FontSize', fs); ylabel('Energy','FontSize', fs);
title('Enerygy variation', 'FontSize', fs-4);

% Histogram
%-----------------------
histData = dlmread('qmcHistogram.txt');
x = histData(:,1);
psi = histData(:,2);
figure(2);
plot(x, psi,'LineWidth', 1.5);

xlabel('x','FontSize', fs); ylabel('\psi','FontSize', fs);
title('Histogram', 'FontSize', fs-4);
```

qmcPlots.m