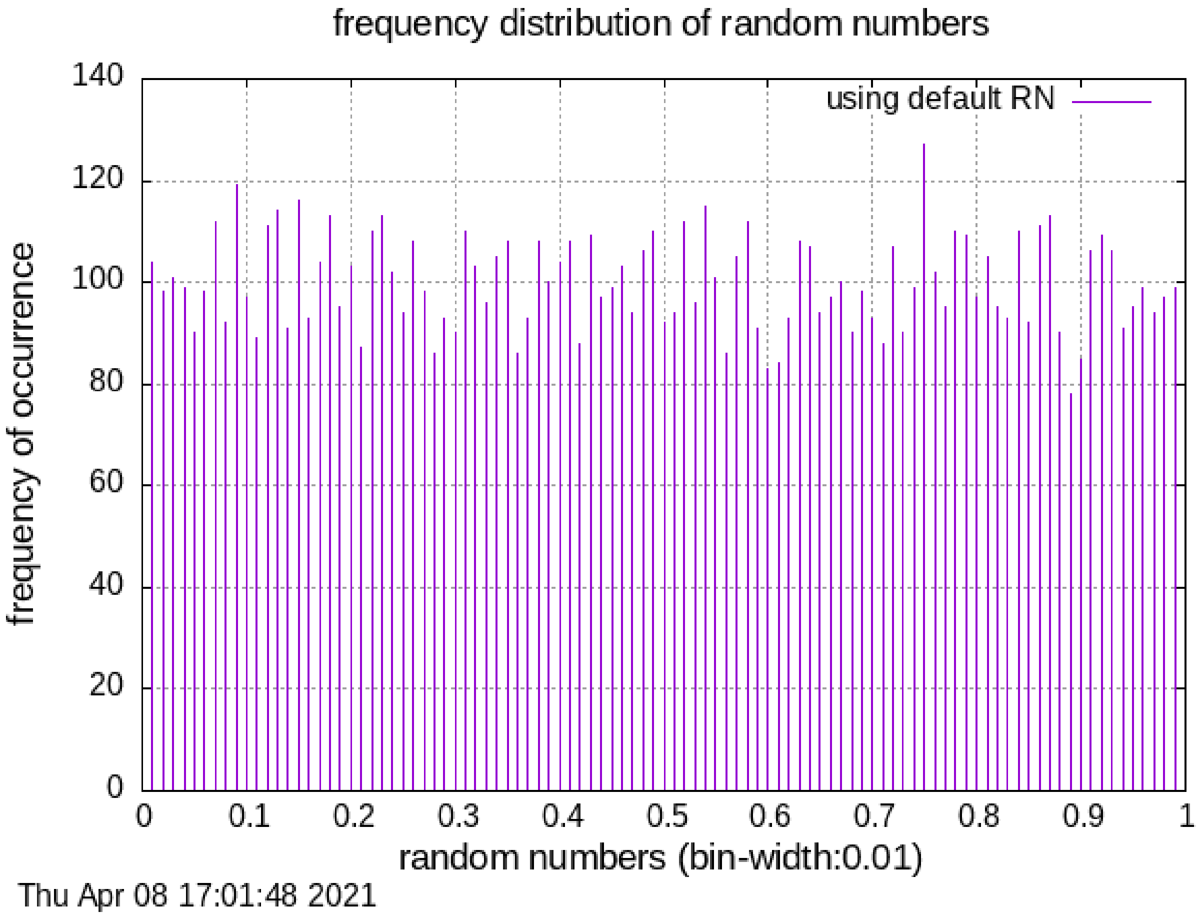
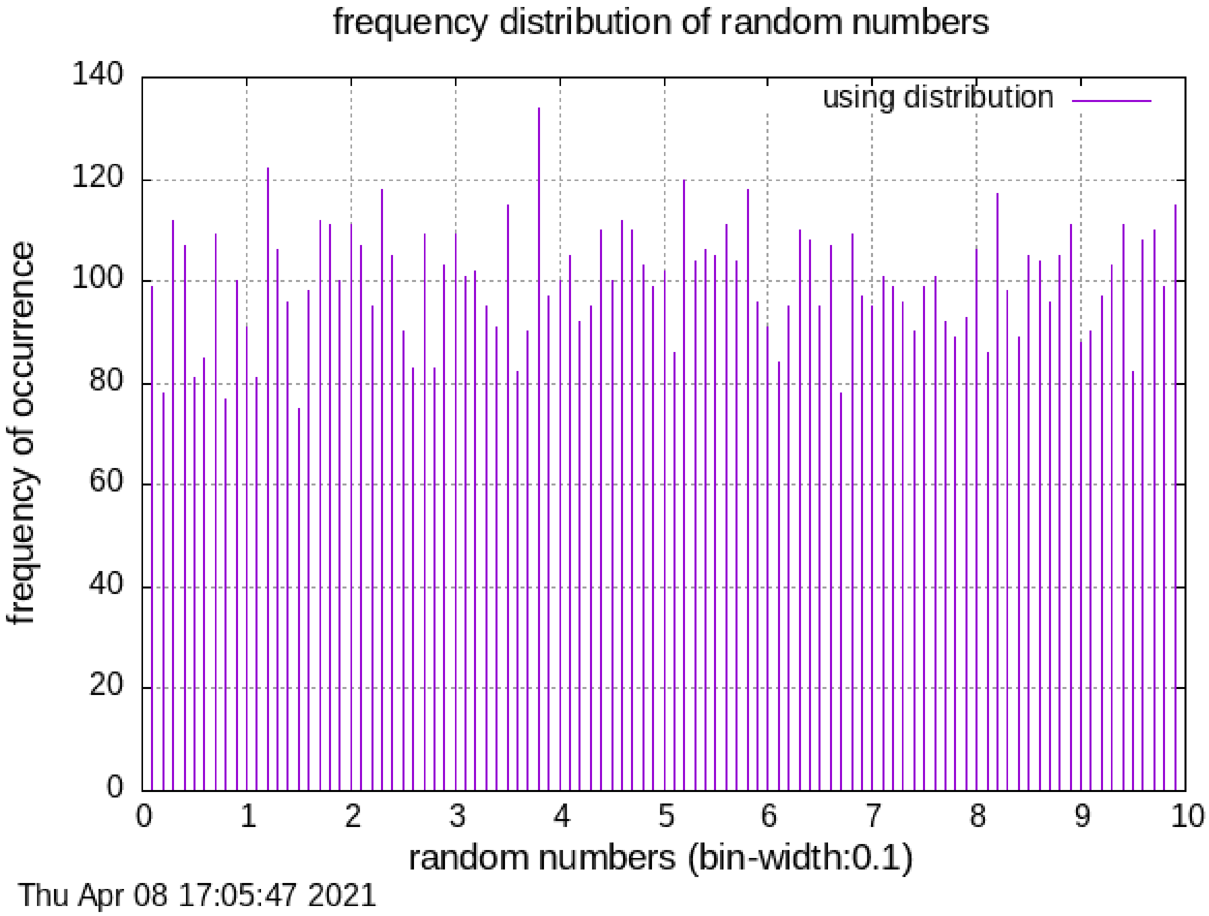
ANT LAB Assignment 07 Monte Carlo Application

PROBLEM 1 :

In [ ]:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
    int i,j,N=10000;
    double r[N],y[N];
    double a=0,b=10;
    srand(time(0));
    for(i=0;i<N;++i) {
        r[i]=((double)rand()/RAND_MAX);
        y[i]=(b-a)*r[i]+a;
    }
    //frequncy distribution within bin width
    double h1=0.01; //bin width for [0:1]
    int bin1=100;   //100 intervals of width 0.01
    int rf[bin1];
    for(j=0;j<bin1;++j) {
        rf[j]=0;
        for(i=0;i<N;i++) {
            //frequncy of RN within bin width
            if((r[i]>=j*h1)&&(r[i]<(j+1)*h1)) {
                rf[j]++;
        }}
    }
    // stroing frequncy distribution
    FILE*fp=NULL;
    fp=fopen("1a.txt","w");
    for(j=0;j<bin1;++j) {
        fprintf(fp,"%lf\t%d\n",j*h1,rf[j]);
    }
    //frequncy distribution within bin width
    double h2=0.1; //width of interval
    int bin2=100; //100 intervals of width 0.1 in [0:10]
    int yf[bin2];
    for(j=0;j<bin2;++j) {
        yf[j]=0;
        for(i=0;i<N;i++) {
            //frequncy of RN within bin width
            if((y[i]>=j*h2)&&(y[i]<(j+1)*h2)) {
                yf[j]++;
        }}
    }
    // stroing frequncy distribution
    FILE*fp2=NULL;
    fp2=fopen("test.txt","w");
    for(j=0;j<bin2;++j) {
        fprintf(fp2,"%lf\t%d\n",j*h2,yf[j]);
    }
}
```

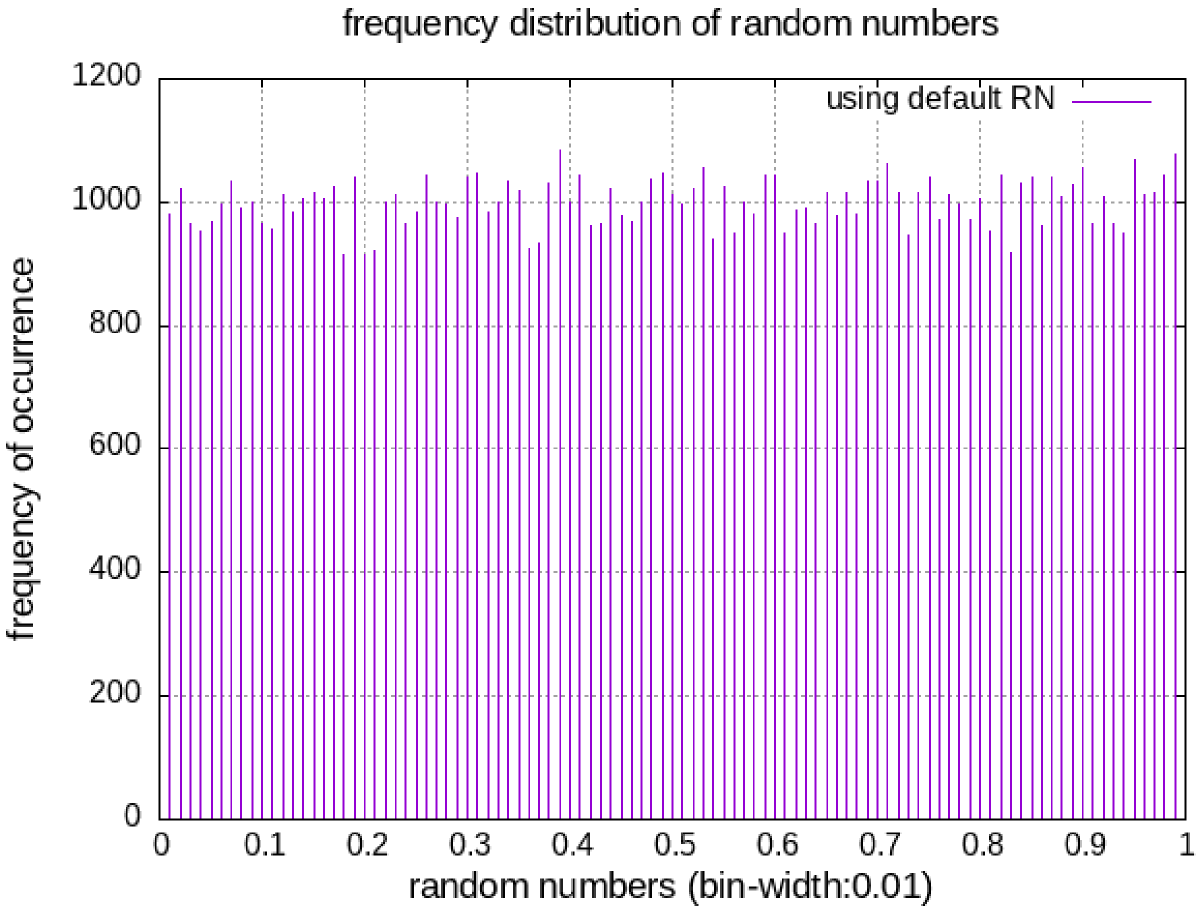

frequency distribution of random numbers
using default RN
frequency of occurrence
0
20
40
60
80
100
120
140
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
random numbers (bin-width:0.01)
Thu Apr 08 17:01:48 2021

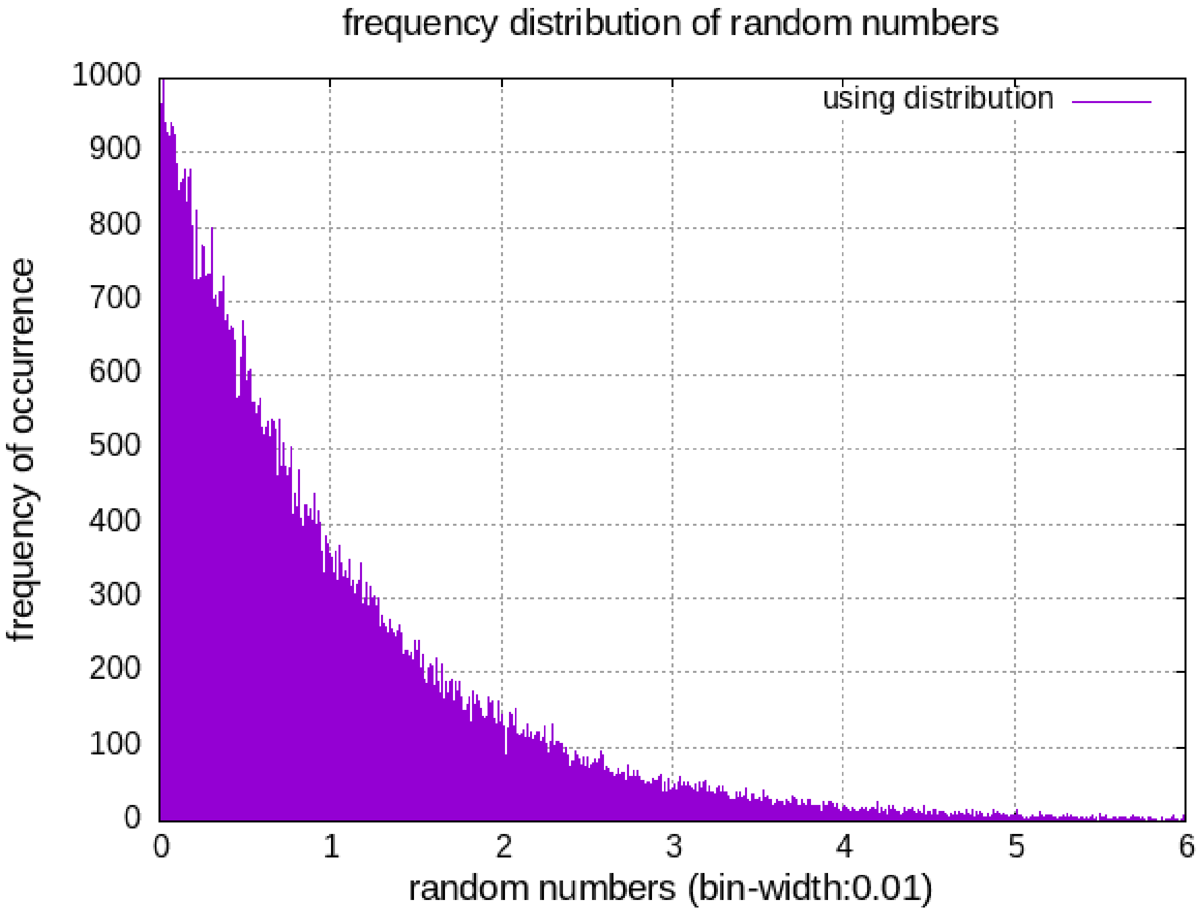

frequency distribution of random numbers
using distribution
frequency of occurrence
0
20
40
60
80
100
120
140
0
1
2
3
4
5
6
7
8
9
10
random numbers (bin-width:0.1)
Thu Apr 08 17:05:47 2021

PROBLEM 2 :

In [ ]:

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

int main()
{
    int i,j,N=100000;
    double r[N],x[N];
    double a=0,b=10;
    srand(time(0));
    for(i=0;i<N;++i) {
        r[i]=((double)rand()/RAND_MAX);
        x[i]=-log(1-r[i]);
    }
    //frequncy distribution within bin width
    double h1=0.01; //bin width for [0:1]
    int bin1=100;   //100 intervals of width 0.01
    int rf[bin1];
    for(j=0;j<bin1;++j) {
        rf[j]=0;
        for(i=0;i<N;i++) {
            //frequncy of RN within bin width
            if((r[i]>=j*h1)&&(r[i]<(j+1)*h1)) {
                rf[j]++;
            }
        }
    }
    // stroing frequncy distribution
    FILE*fp=NULL;
    fp=fopen("2a.txt","w");
    for(j=0;j<bin1;++j) {
        fprintf(fp,"%lf\t%d\n",j*h1,rf[j]);
    }
    //frequncy distribution within bin width
    double h2=0.01; //width of interval
    int bin2=100*6; //100 intervals of width 0.1 in [0:10]
    int xf[bin2];
    for(j=0;j<bin2;++j) {
        xf[j]=0;
        for(i=0;i<N;i++) {
            //frequncy of RN within bin width
            if((x[i]>=j*h2)&&(x[i]<(j+1)*h2)) {
                xf[j]++;
            }
        }
    }
    // stroing frequncy distribution
    FILE*fp2=NULL;
    fp2=fopen("2b.txt","w");
    for(j=0;j<bin2;++j) {
        fprintf(fp2,"%lf\t%d\n",j*h2,xf[j]);
    }
}
```

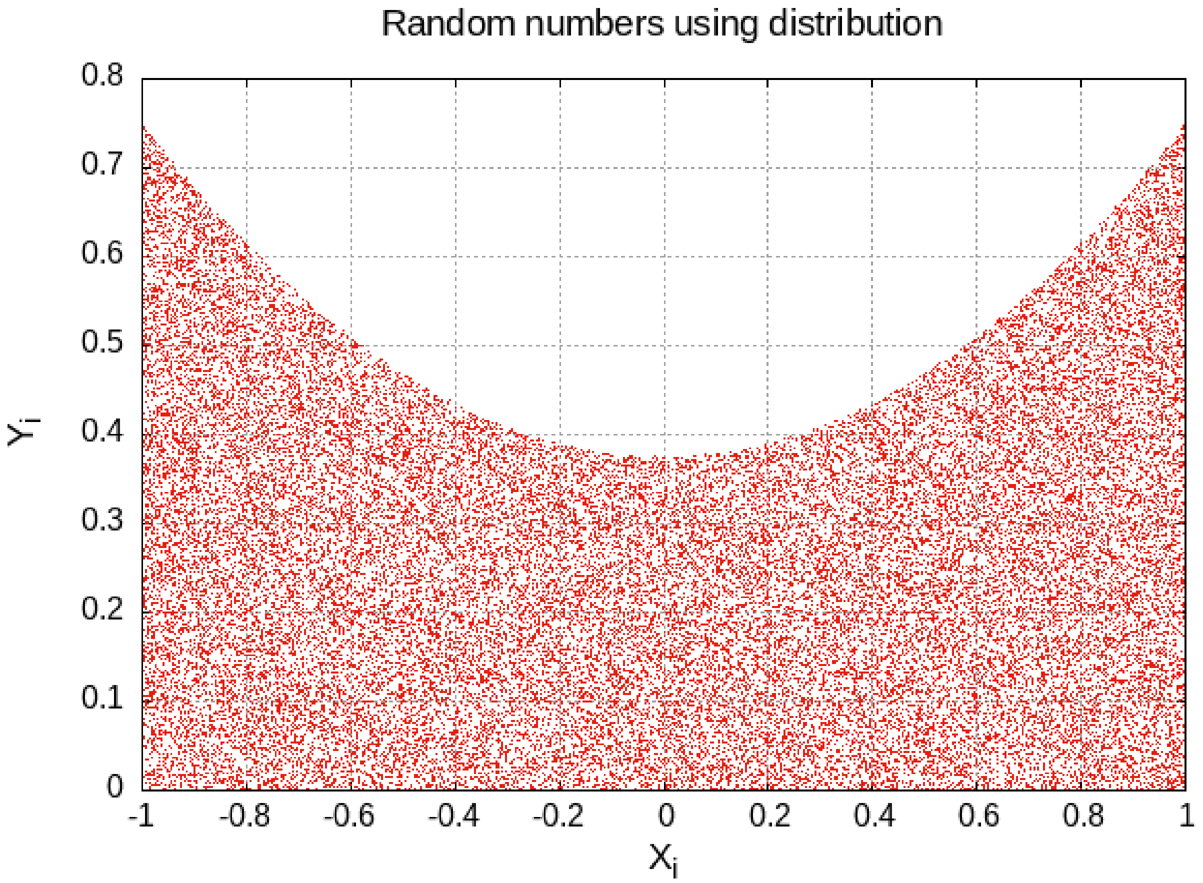

frequency distribution of random numbers
using default RN
frequency of occurrence
0
200
400
600
800
1000
1200
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
random numbers (bin-width:0.01)


frequency distribution of random numbers
using distribution
frequency of occurrence
0
100
200
300
400
500
600
700
800
900
1000
0
1
2
3
4
5
6
random numbers (bin-width:0.01)

PROBLEM 3 :

In [ ]:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
// generating random numbers b/w range
float randnum(float min, float max) {
    float random = ((float)rand())/(float)RAND_MAX;
    return (max-min)*random + min;
}
// defining the probability density function
float f(float x) {
    return (3/8.0)*(1+x*x);
}
int main()
{
    int i,j,N=100000;
    float x[N],y[N];
    float fmax=3/4.0;
    srand(time(0));
    for(i=0;i<N;++i) {
        x[i]=randnum(-1,1);
        y[i]=randnum(0,fmax);
    }
    int Naccept=0;
    float X[N],Y[N];
    for(i=0;i<N;i++) {
        if(y[i]<=f(x[i])) {
            X[i]=x[i];
            Y[i]=y[i];
            Naccept++;
        }
    }
    //printf("%d\n",Naccept);
    FILE*fp=NULL;
    fp=fopen("3a.txt","w");
    for(i=0;i<Naccept;++i) {
        fprintf(fp,"%f\t%f\n",X[i],Y[i]);
    }
}
```



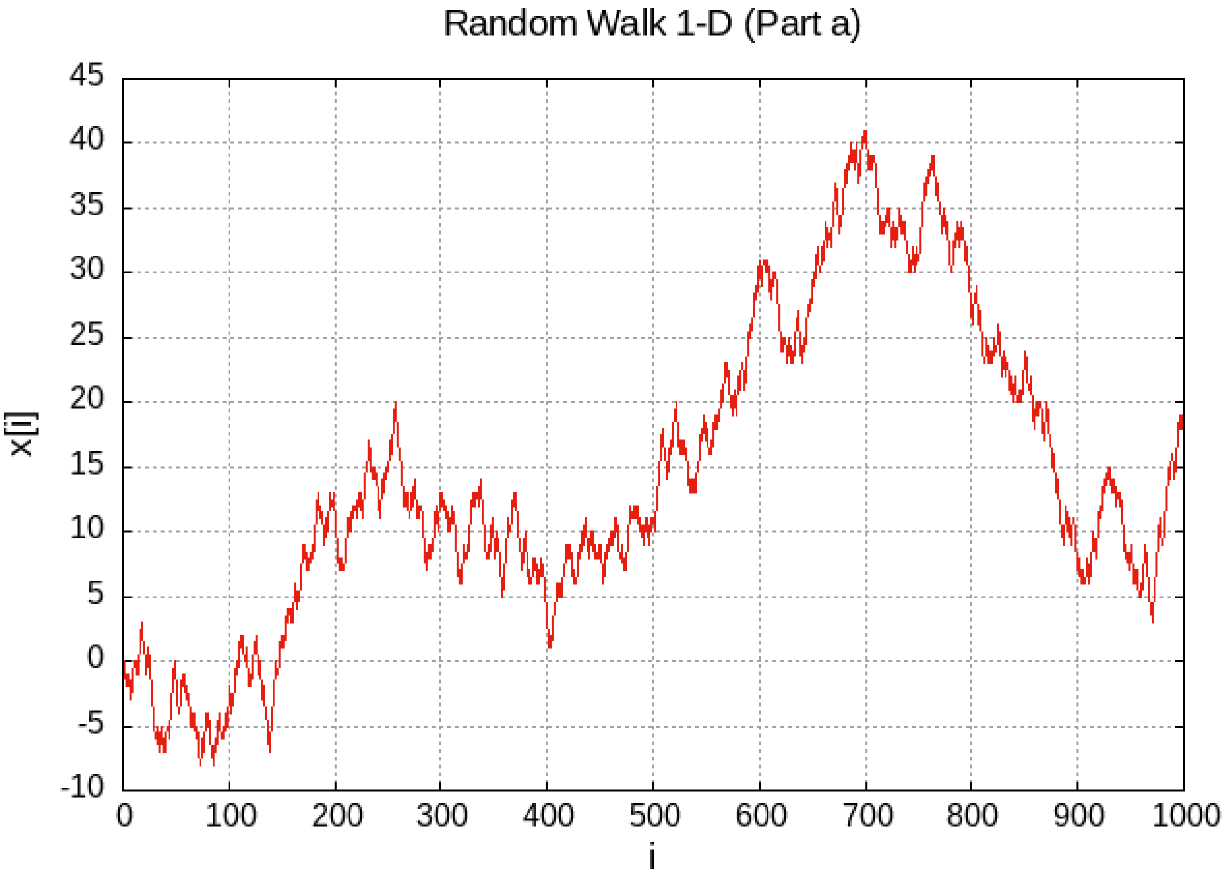
PROBLEM 4 : (Part a,b,c)

In [ ]:

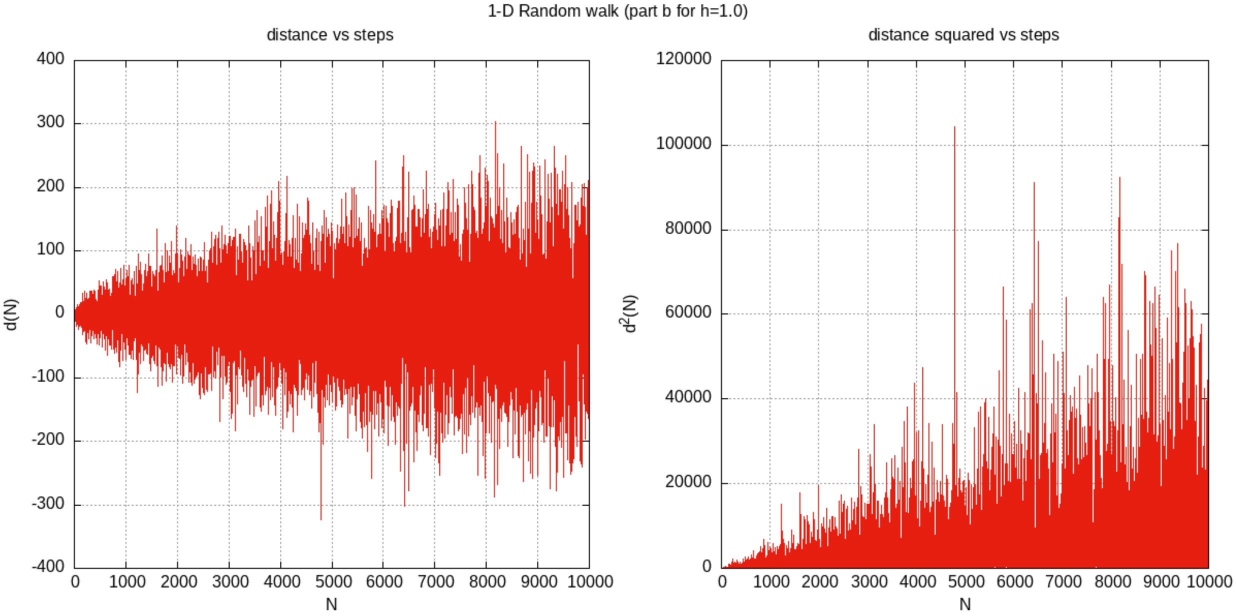
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
// generating uniform random numbers b/w [0:1] N times
void uni(int N,double h,double x[N])
{
    double u[N];
    x[0]=0.0;
    for (int i=1;i<N;++i) {
        u[i]=((double)rand()/(double)RAND_MAX);
            if(u[i]<0.5)
                x[i] = x[i-1]-h;
            else
                x[i] = x[i-1]+h;
    }
}
//calculating distance dN = x[N]-x[0]
double dN(int N,double x[N]) {
    return (x[N-1]-x[0]);
}
int main()
{
    int i,N=1000;
    double x[10000],h;
    srand(time(0));

    // part a
    h=1.0;
    FILE*fp=NULL;
    fp=fopen("4a.txt","w");
    uni(N,h,x);
    for(i=0;i<N;++i) {
        fprintf(fp,"%d\t%lf\n",i+1,x[i]);
    }
    printf("part a. The actual distance traveled:%.2f\n",dN(N,x));

    // // part b and c
    FILE*fp1=NULL;
    FILE*fp2=NULL;
    FILE*fp3=NULL;
    FILE*fp4=NULL;
    FILE*fp5=NULL;
    fp1=fopen("4ch.1.txt","w");
    fp2=fopen("4ch1.txt","w");
    fp3=fopen("4ch2.txt","w");
    fp4=fopen("4ch10.txt","w");
    fp5=fopen("4ch50.txt","w");
    for (N=10;N<=10000;++N)
    {
        uni(N,0.1,x);
        fprintf(fp1,"%d\t%lf\t%lf\n",N,dN(N,x),dN(N,x)*dN(N,x));
        uni(N,1.0,x);
        fprintf(fp2,"%d\t%lf\t%lf\n",N,dN(N,x),dN(N,x)*dN(N,x));
        uni(N,2.0,x);
        fprintf(fp3,"%d\t%lf\t%lf\n",N,dN(N,x),dN(N,x)*dN(N,x));
        uni(N,10.0,x);
        fprintf(fp4,"%d\t%lf\t%lf\n",N,dN(N,x),dN(N,x)*dN(N,x));
        uni(N,50.0,x);
        fprintf(fp5,"%d\t%lf\t%lf\n",N,dN(N,x),dN(N,x)*dN(N,x));
    }
}
```

OUTPUT:

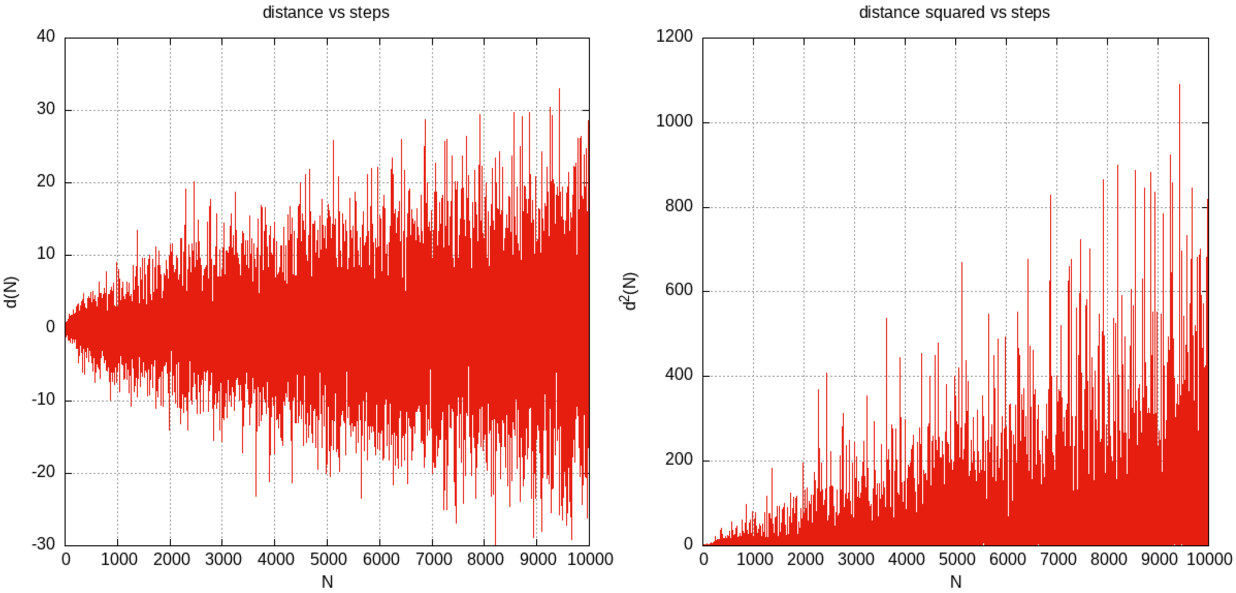
```
part a. The actual distance traveled:23.00
```



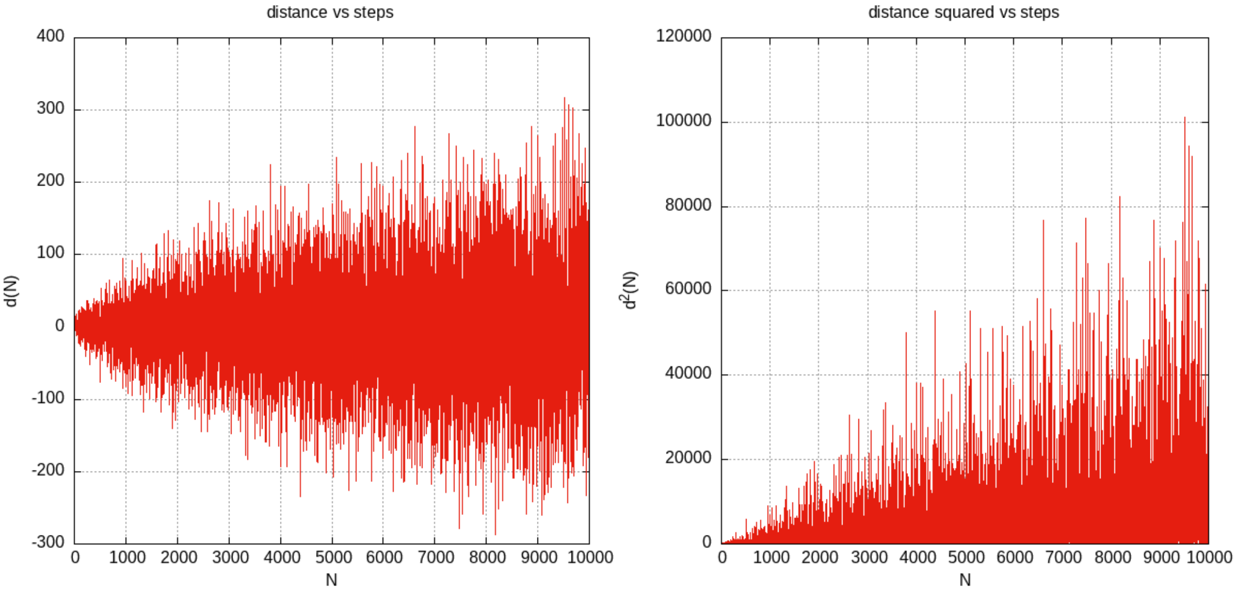
Thu Apr 15 22:47:35 2021



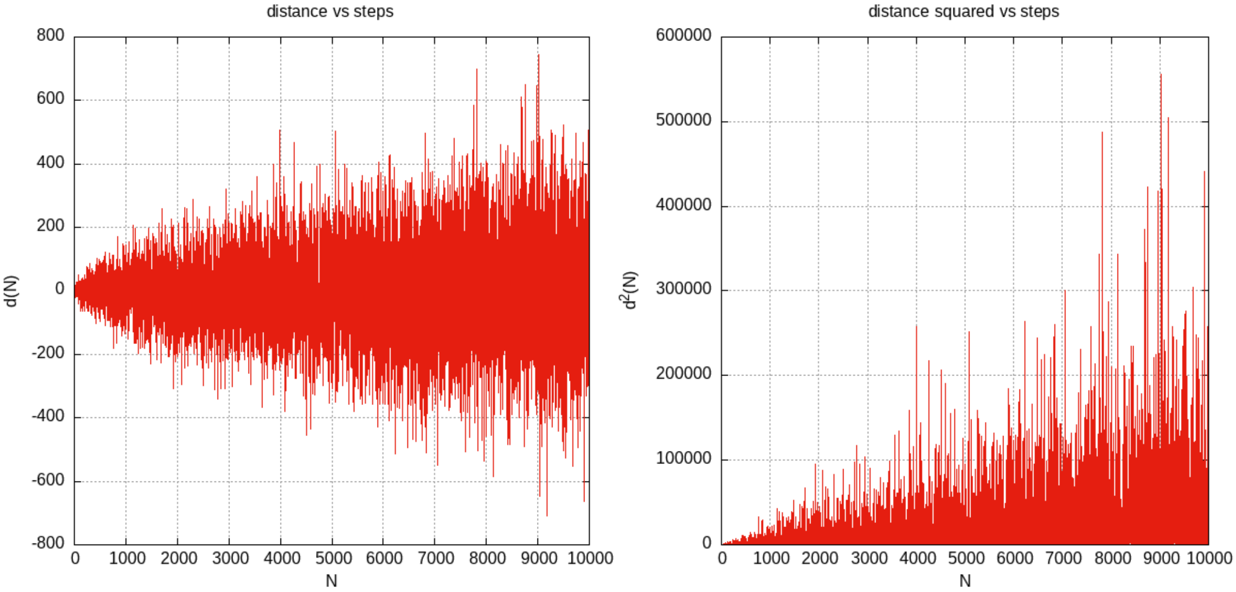
1-D Random walk (part c for h=0.1)

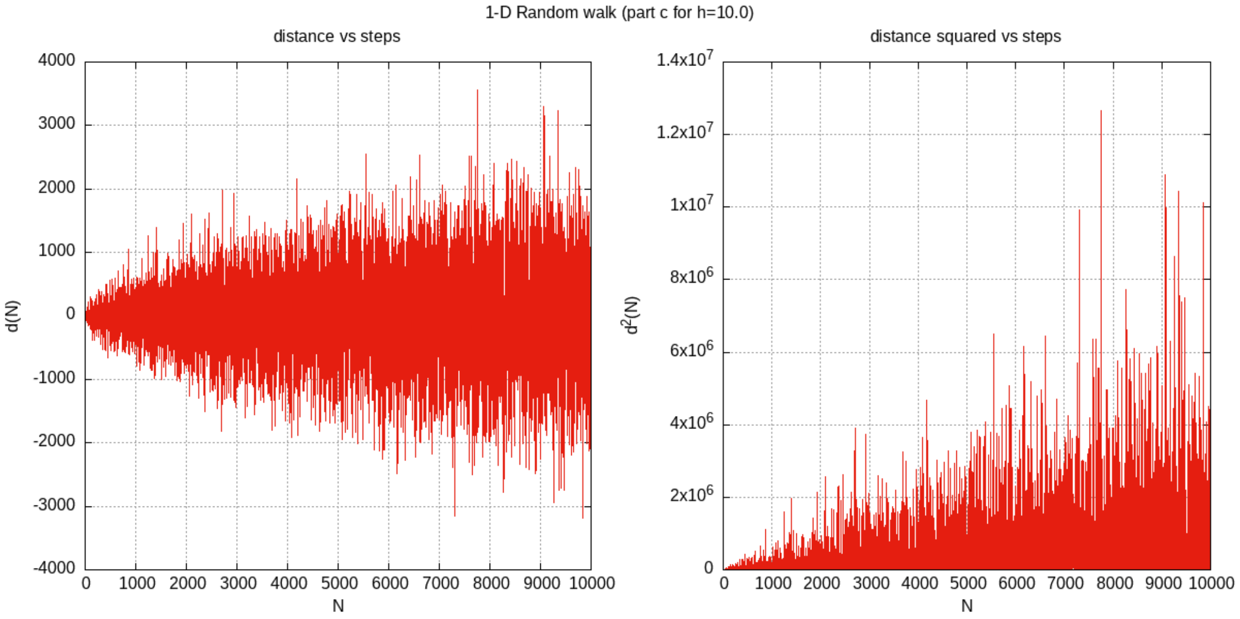


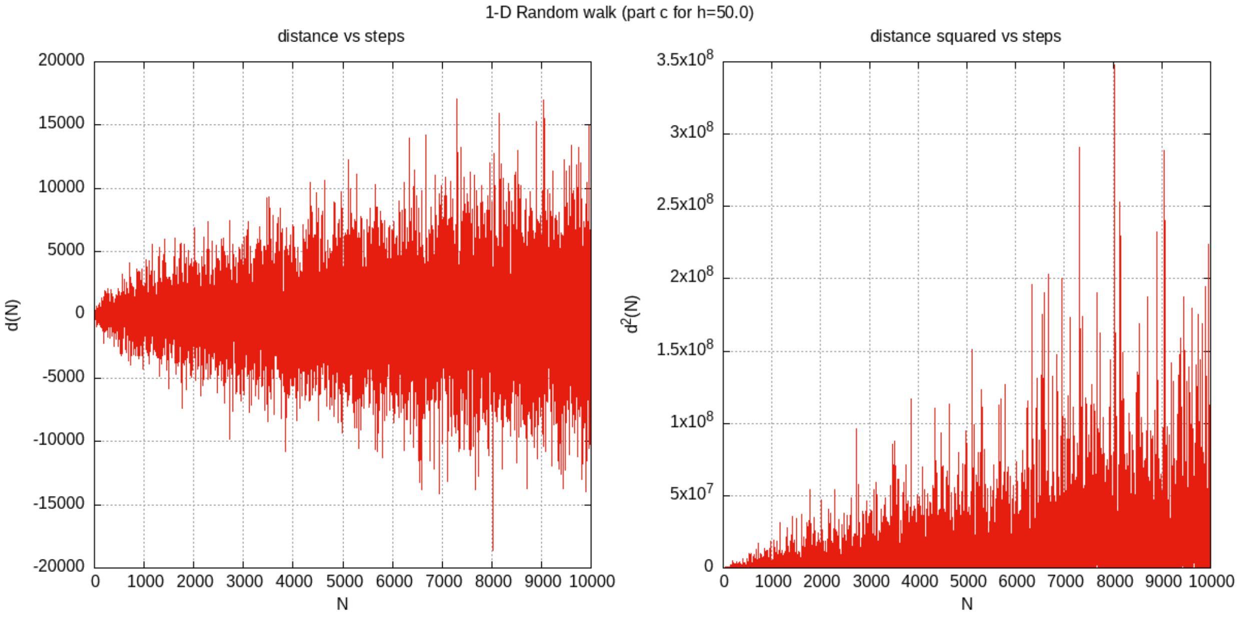
1-D Random walk (part c for h=1.0)



1-D Random walk (part c for h=2.0)



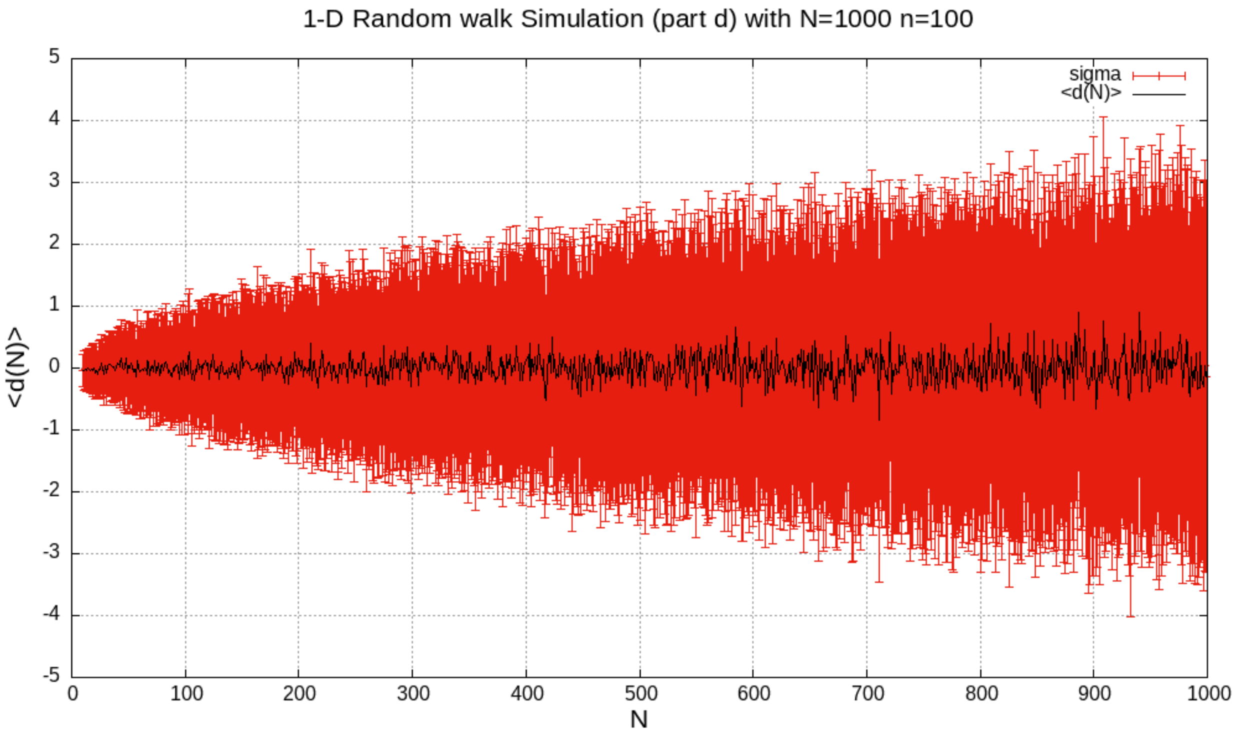


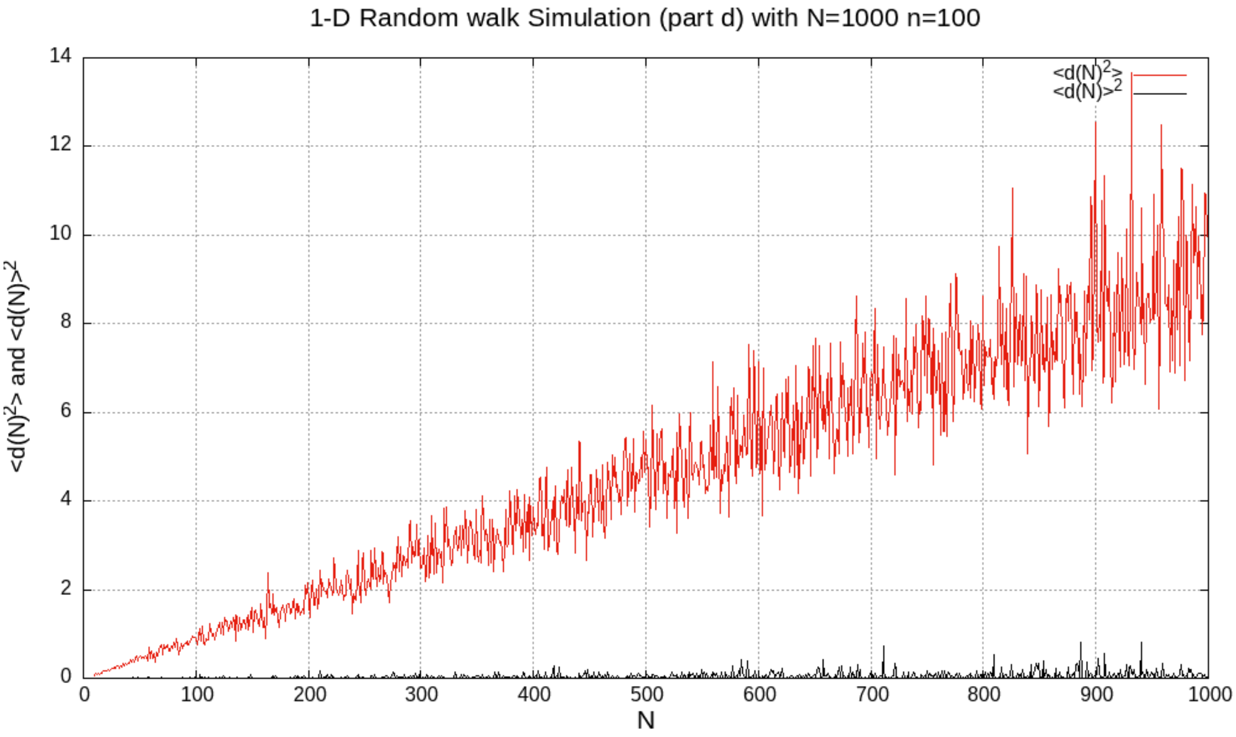


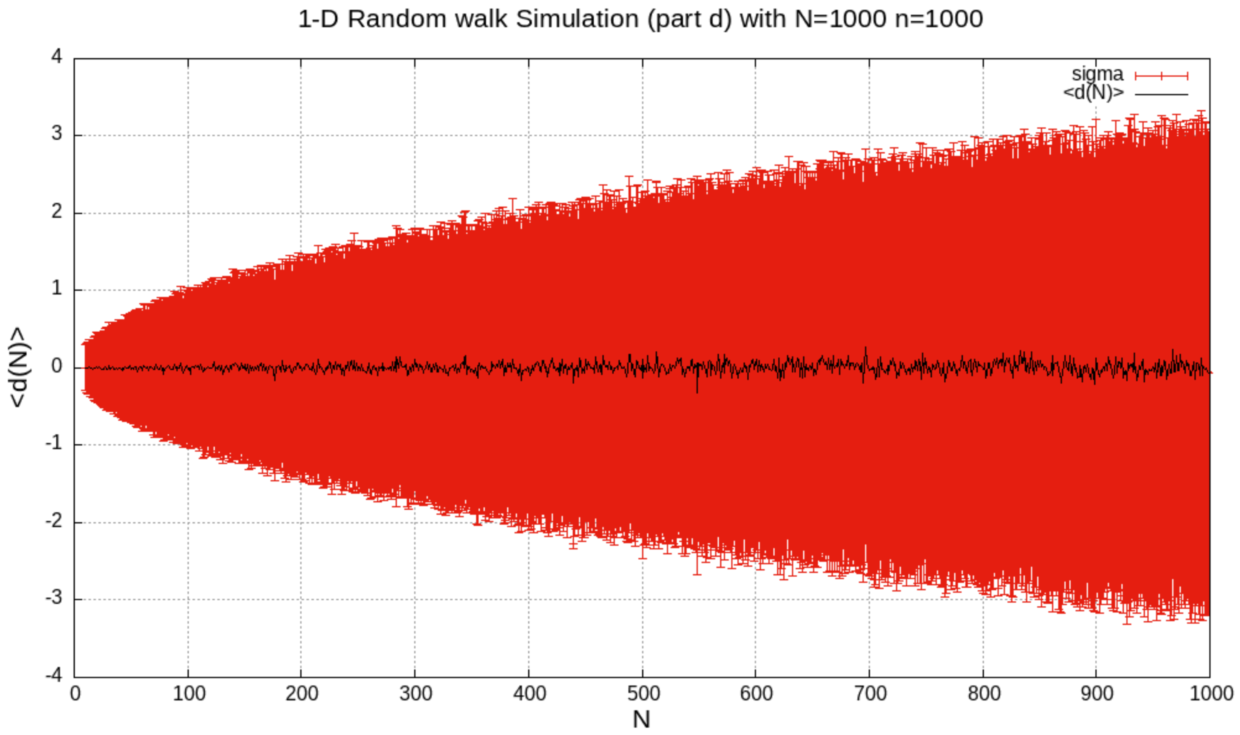
PROBLEM 4 : (Part d)

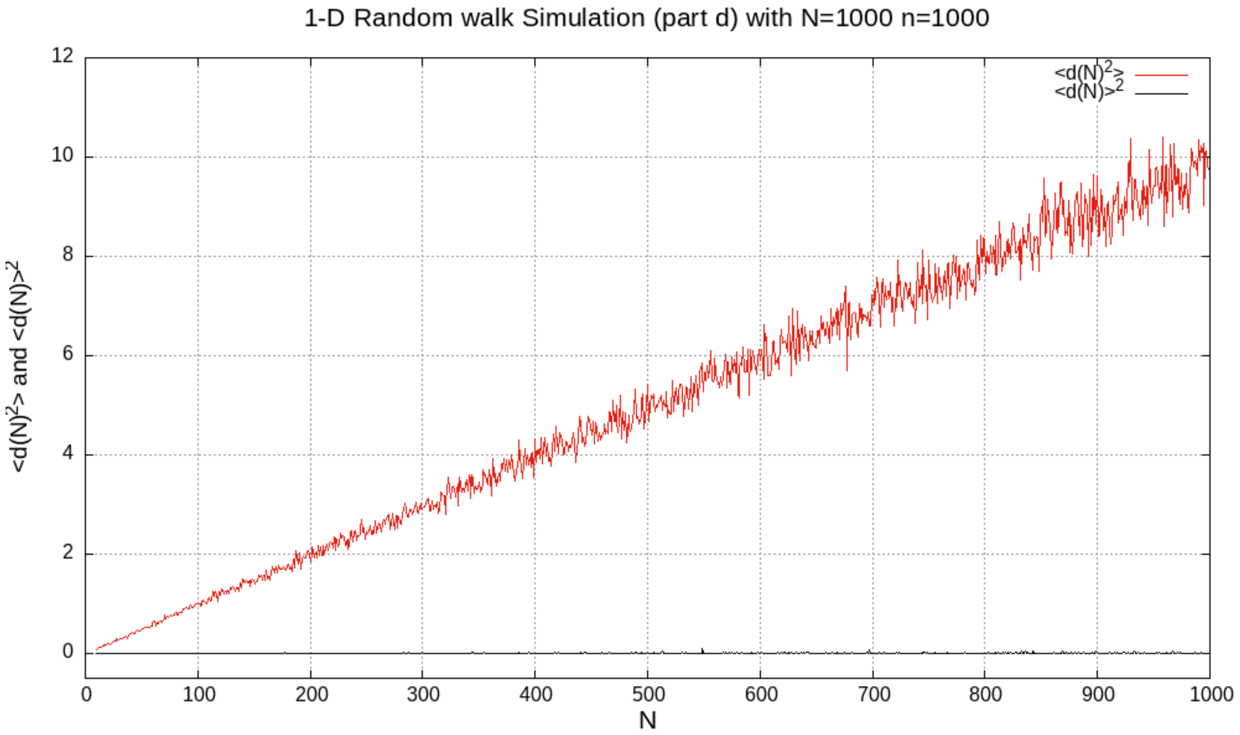
In [ ]:

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

// generating uniform random numbers b/w [0:1] N times
void uni(int N,double x[N])
{
    double u[N],h=0.1;
    x[0]=0.0;
    for (int i=1;i<N;++i) {
        u[i]=((double)rand()/(double)RAND_MAX);
            if(u[i]<0.5)
                x[i] = x[i-1]-h;
            else
                x[i] = x[i-1]+h;
    }
}
// calculating mean of an array
double mean(int N,double arr[N]) {
    double sum=0;
    for (int i=0;i<N;++i)
        sum += arr[i];
    return sum/N;
}
// calculating dispersion relation of an array
double sigma(int N,double arr[N]) {
    double avg,std=0;
    avg=mean(N,arr);
    for (int i=0;i<N;++i)
        std += pow((arr[i]-avg),2);
    return sqrt(std/(N-1));
}
//calculating distance dN = x[N]-x[0]
double dN(int N,double x[N]) {
    return (x[N-1]-x[0]);
}
int main()
{
    int i,j,N=1000,n=100;
    double x[N];
    srand(time(0));

    FILE*fp=NULL;
    fp=fopen("4d1.txt","w");
    FILE*fp1=NULL;
    fp1=fopen("4d2.txt","w");

    double dn[N],sig[N],dn2[N],d1[N],d2[N];
    int k=0;
    for (i=10;i<N;i++) {
        for (j=10;j<n;j++) {
            uni(i,x);
            d1[j]=dN(i,x); //storing 100 values of dN per N
            d2[j]=d1[j]*d1[j];
        }
        dn[k] = mean(n,d1);
        sig[k] = sigma(n,d1);
        dn2[k] = mean(n,d2);
        fprintf(fp,"%d\t%.4lf\t%.4lf\n",i, dn[k],sig[k]);
        fprintf(fp1,"%d\t%lf\t%lf\n",i, dn2[k], dn[k]*dn[k]);
        k++;
    }
}
```


1-D Random walk Simulation (part d) with N=1000 n=100
sigma
<d(N)>
<d(N)>
N
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
100
200
300
400
500
600
700
800
900
1000


1-D Random walk Simulation (part d) with N=1000 n=100
<d(N)^2>
<d(N)>^2
<d(N)^2> and <d(N)>^2
N
14
12
10
8
6
4
2
0
0
100
200
300
400
500
600
700
800
900
1000





PROBLEM 5 : (Part a,b,c)

In [ ]:

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

int c,c1,c2;
double dx,dy,dr;
//calculating distance dN = x[N]-x[0]
double dN(int N,double arr[N]) {
    return (arr[N-1]-arr[0]);
}
// generating uniform random numbers b/w [0:1] N times
void rwalk(int N,double h,double K,double x[N],double y[N])
{
    int i,j=1,k=1;
    double u[N];
    c1=0,c2=0; // initial value count
    x[0]=0.0;y[0]=0.0;
    for (int i=0;i<N;++i) {
        u[i]=((double)rand()/(double)RAND_MAX);
        //printf("u[%d]:\t%lf\n",i,u[i]);
        if(u[i]<=0.25) {
            x[j]=x[j-1]-h;
            c1++;j++;
        }
        else if (u[i]>0.25 && u[i]<=0.5) {
            x[j]=x[j-1]+h;
            c1++;j++;
        }
        else if(u[i]>0.5 && u[i]<0.75) {
            y[k]=y[k-1]+K;
            c2++;k++;
        }
        else {
            y[k]=y[k-1]-K;
            c2++;k++;
        }
    }
    // rejecting extra value in either array
    if (c1>c2)
        c=c2;
    else
        c=c1;

    dx=dN(c,x);
    dy=dN(c,y);
    dr=sqrt(dx*dx+dy*dy);
}

int main()
{
    int i,j,N=1000;
    double x[10000],y[10000];
    srand(time(0));
    double h=1.0,k=1.0;

    // part a
    FILE*fp=NULL;
    fp=fopen("5a.txt","w");
    rwalk(N,h,k,x,y);
    for(i=0;i<c;++i) {
        fprintf(fp,"%d\t%lf\t%lf\n",i+1,x[i],y[i]);
    }
    printf("for part a\ndx=%.2f\n",dx);
    printf("dy=%.2f\n",dy);
    printf("dx+dy=%.2f\n",dx+dy);
    printf("dr=%.2f\n",dr);

        // // part b and c
    FILE*fp1=NULL;
    FILE*fp2=NULL;
    FILE*fp3=NULL;
    FILE*fp4=NULL;
    FILE*fp5=NULL;
    fp1=fopen("5ch.1.txt","w");
    fp2=fopen("5ch1.txt","w");
    fp3=fopen("5ch2.txt","w");
    fp4=fopen("5ch10.txt","w");
    fp5=fopen("5ch50.txt","w");
    for (N=10;N<=10000;++N)
    {
        rwalk(N,0.1,0.1,x,y);
        fprintf(fp1,"%d\t%lf\t%lf\t%lf\t%lf\t%lf\n",N,dx,dy,dx+dy,dr,dr*dr);
        rwalk(N,1.0,1.0,x,y);
        fprintf(fp2,"%d\t%lf\t%lf\t%lf\t%lf\t%lf\n",N,dx,dy,dx+dy,dr,dr*dr);
        rwalk(N,2.0,2.0,x,y);
        fprintf(fp3,"%d\t%lf\t%lf\t%lf\t%lf\t%lf\n",N,dx,dy,dx+dy,dr,dr*dr);
        rwalk(N,10.0,10.0,x,y);
        fprintf(fp4,"%d\t%lf\t%lf\t%lf\t%lf\t%lf\n",N,dx,dy,dx+dy,dr,dr*dr);
        rwalk(N,50.0,50.0,x,y);
        fprintf(fp5,"%d\t%lf\t%lf\t%lf\t%lf\t%lf\n",N,dx,dy,dx+dy,dr,dr*dr);
    }
}
```

OUTPUT:

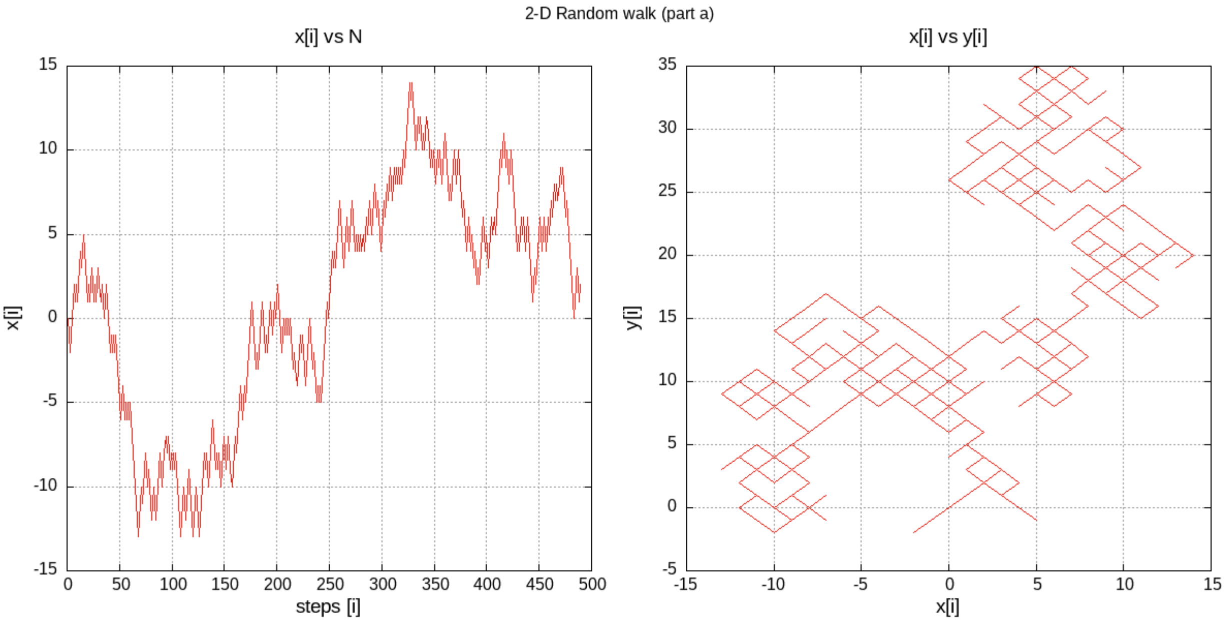
for part a

dx=2.00

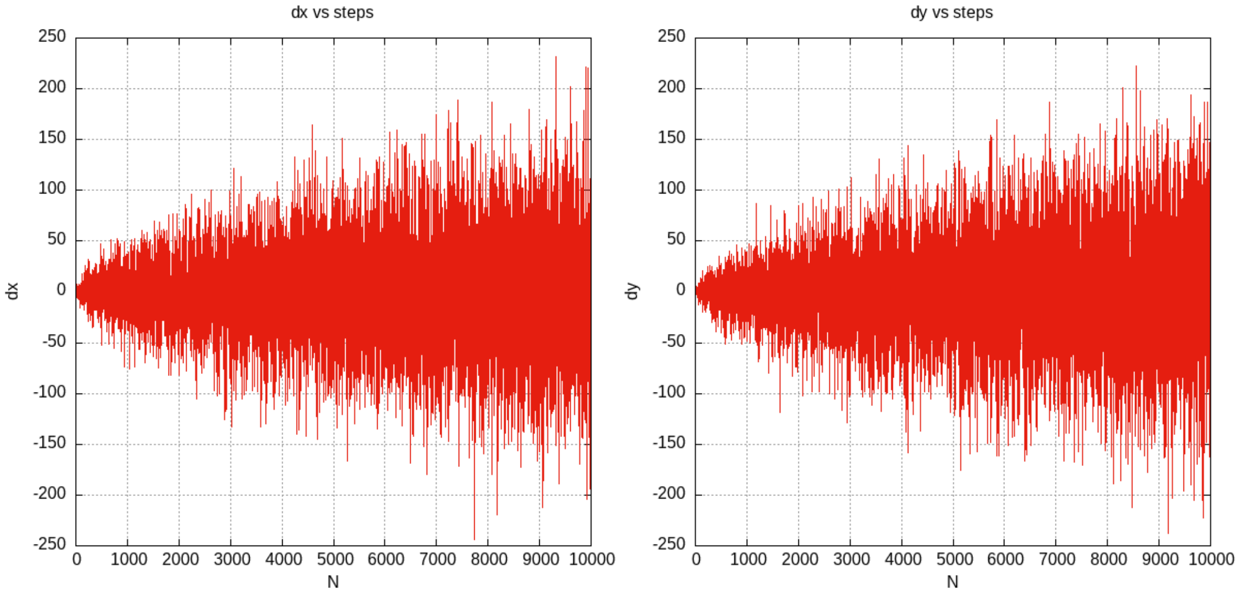
dy=24.00

dx+dy=26.00

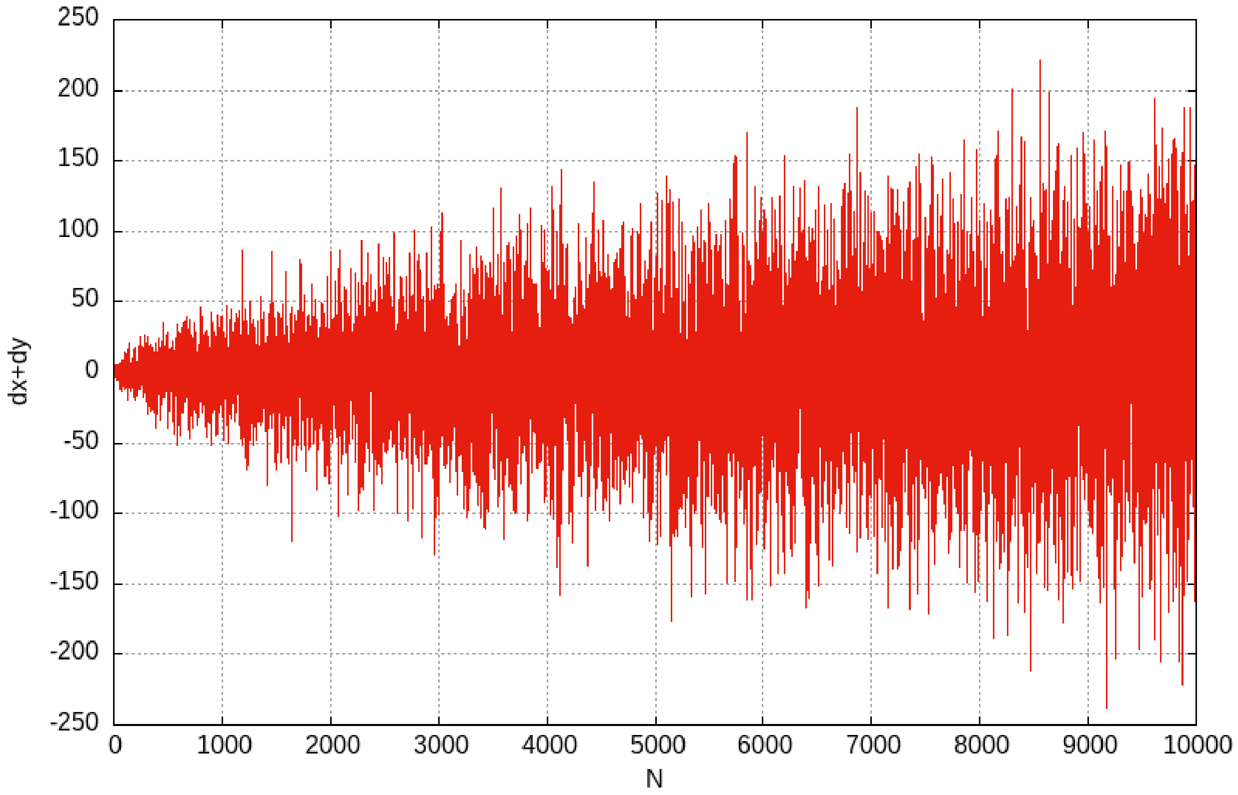
dr=24.08

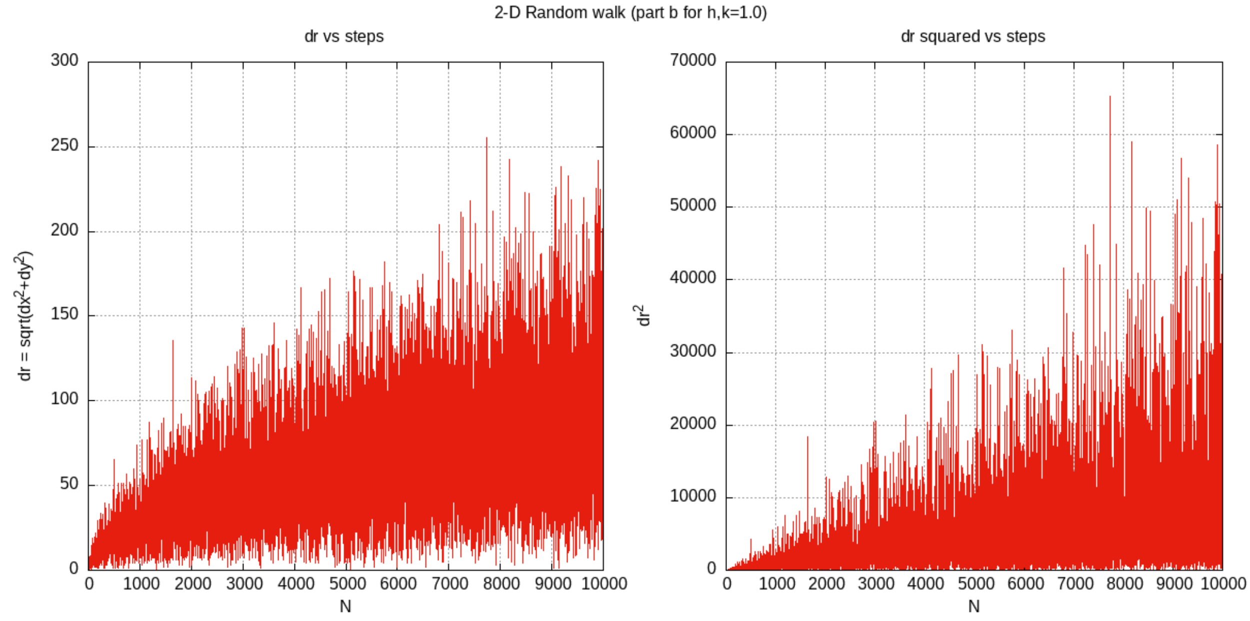


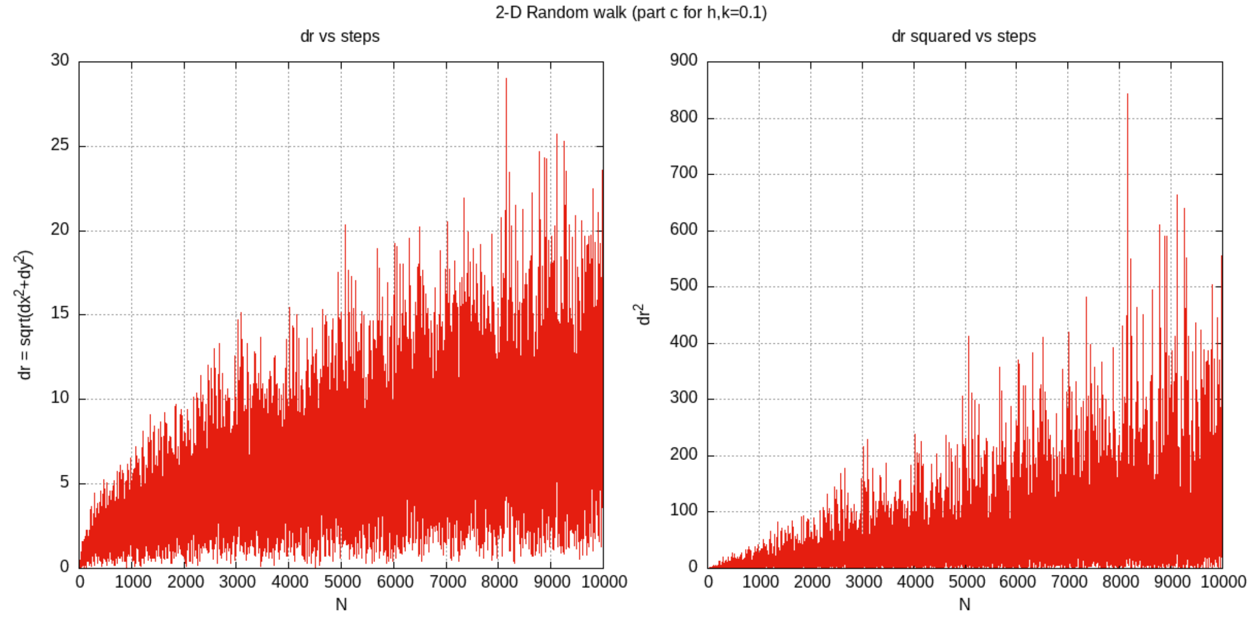
2-D Random walk (part b:dx dy vs N)

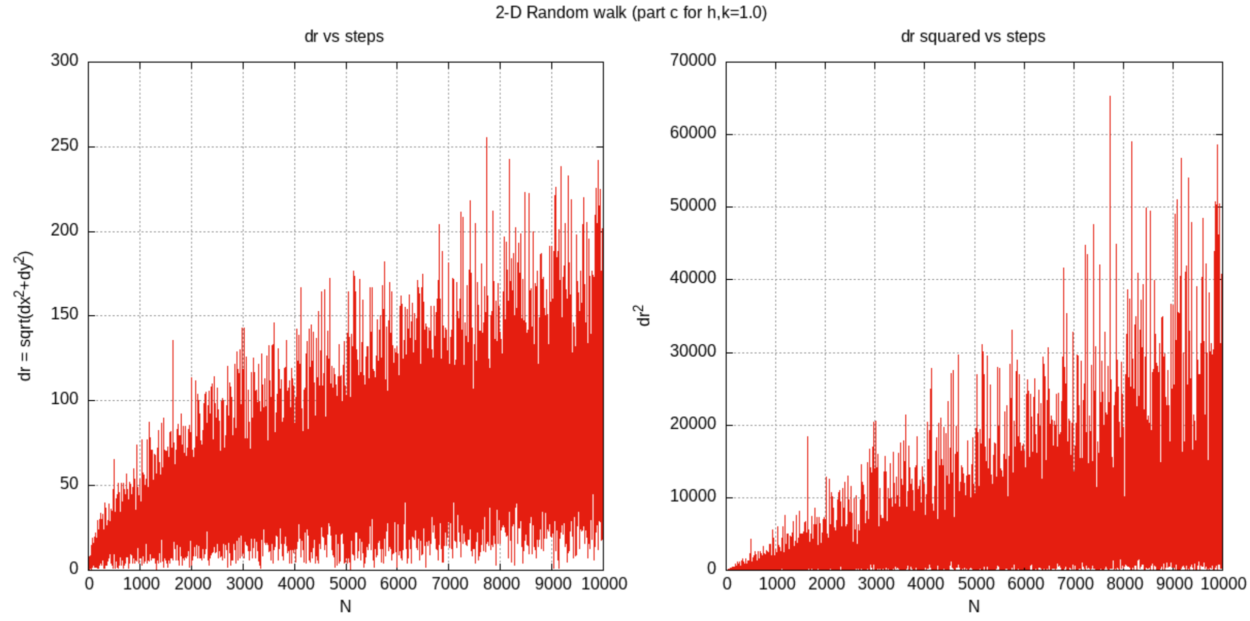


2-D Random walk (part b: total=dx+dy vs steps)

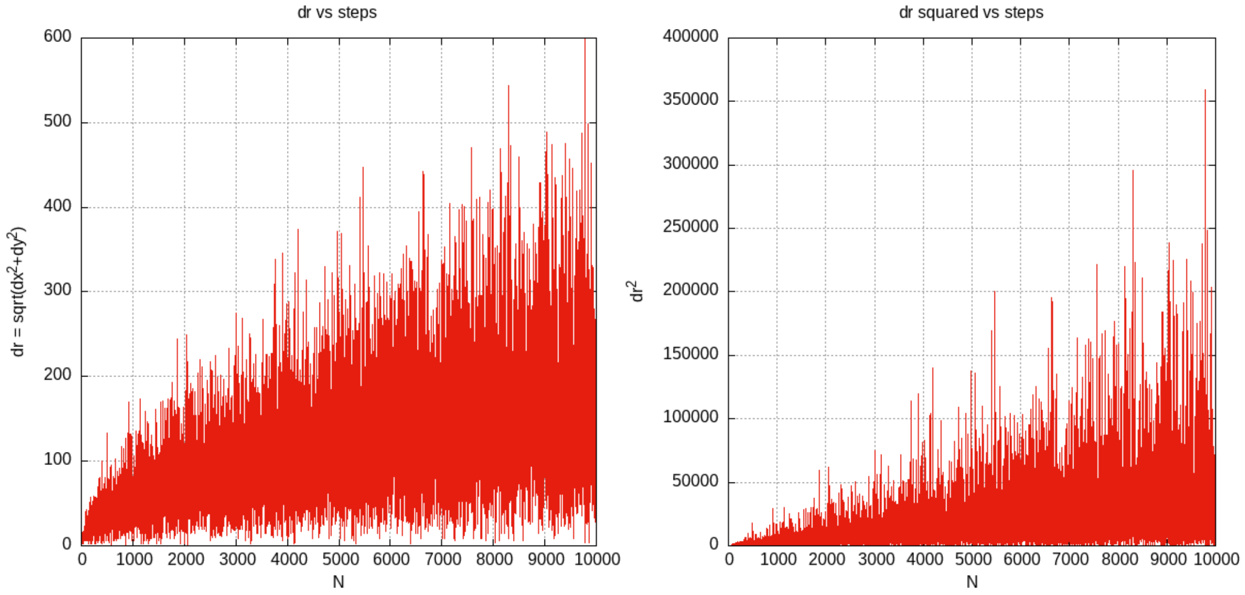



2-D Random walk (part b for h,k=1.0)
dr vs steps
dr squared vs steps
dr = sqrt(dx^2+dy^2)
dr^2
0
50
100
150
200
250
300
0
10000
20000
30000
40000
50000
60000
70000
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N
N

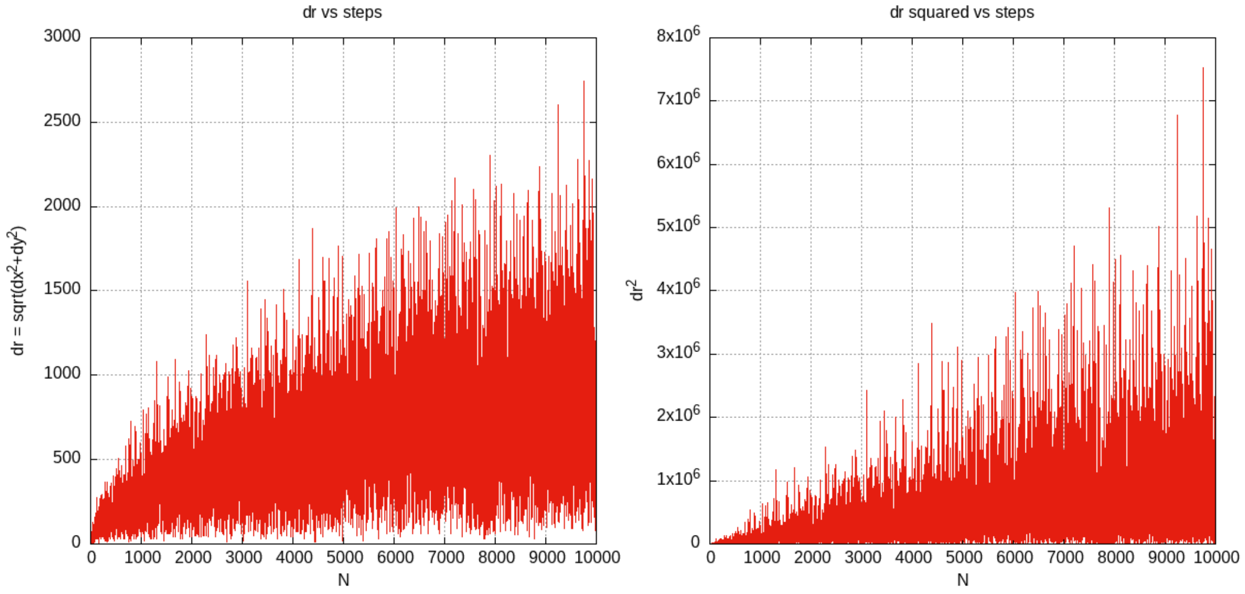

2-D Random walk (part c for h,k=0.1)
dr vs steps
dr squared vs steps
dr = sqrt(dx^2+dy^2)
dr^2
0
5
10
15
20
25
30
0
100
200
300
400
500
600
700
800
900
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N
N


2-D Random walk (part c for h,k=1.0)
dr vs steps
dr squared vs steps
dr = sqrt(dx^2+dy^2)
dr^2
0
50
100
150
200
250
300
0
10000
20000
30000
40000
50000
60000
70000
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N
N

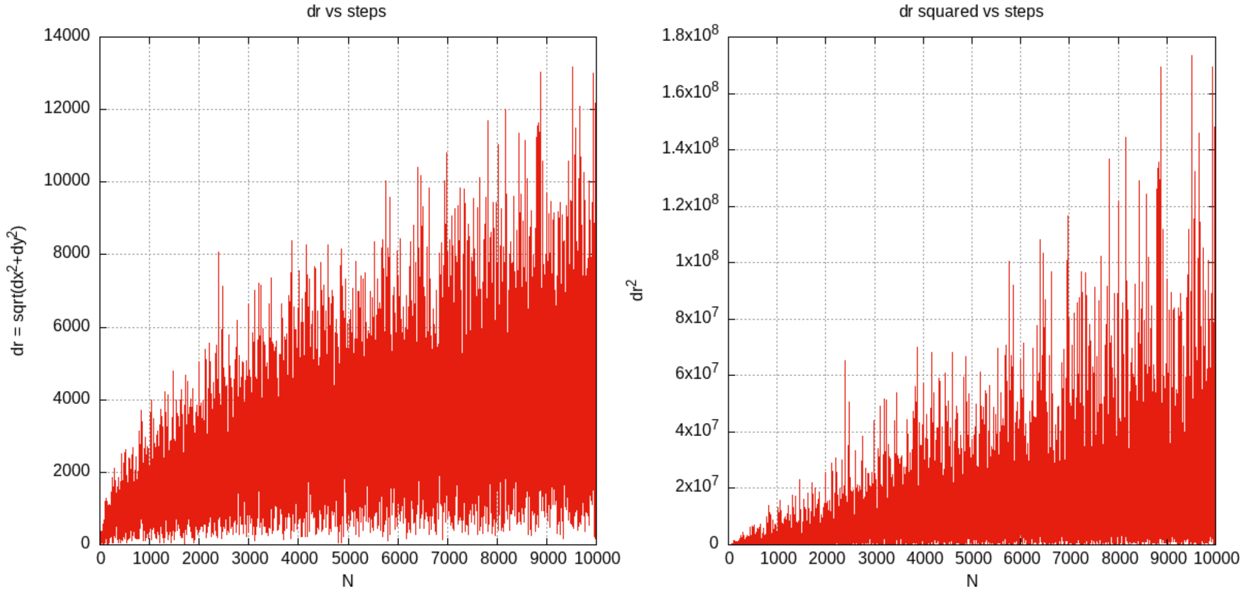
2-D Random walk (part c for h,k=2.0)



2-D Random walk (part c for h,k=10.0)



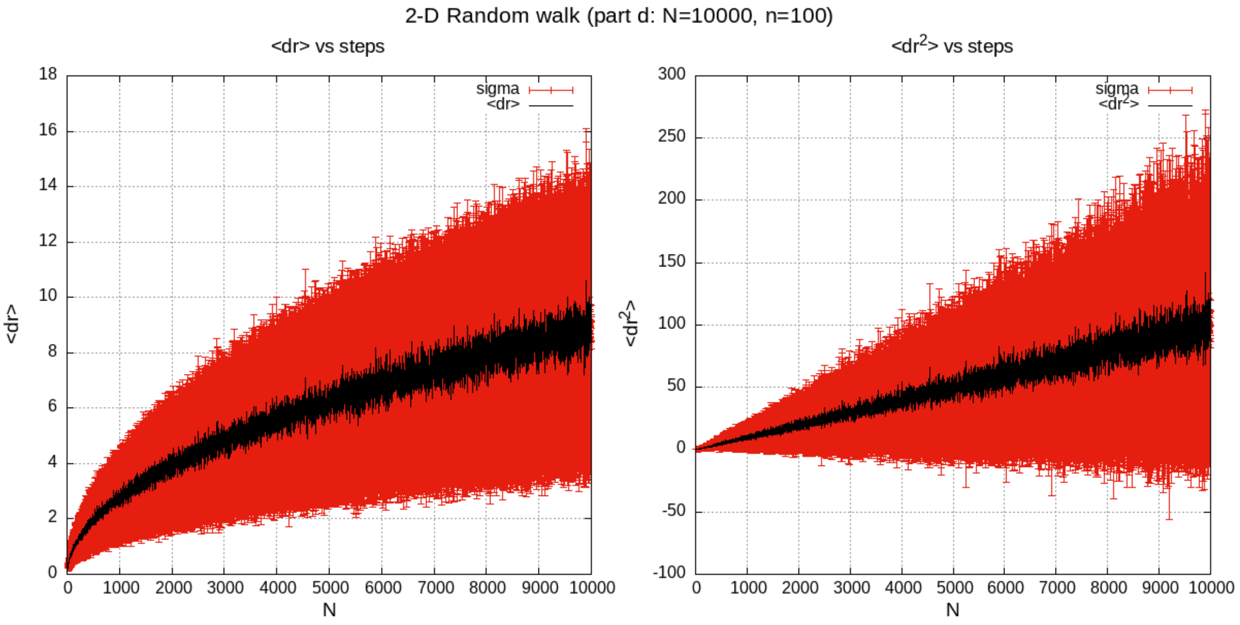
2-D Random walk (part c for h,k=50.0)

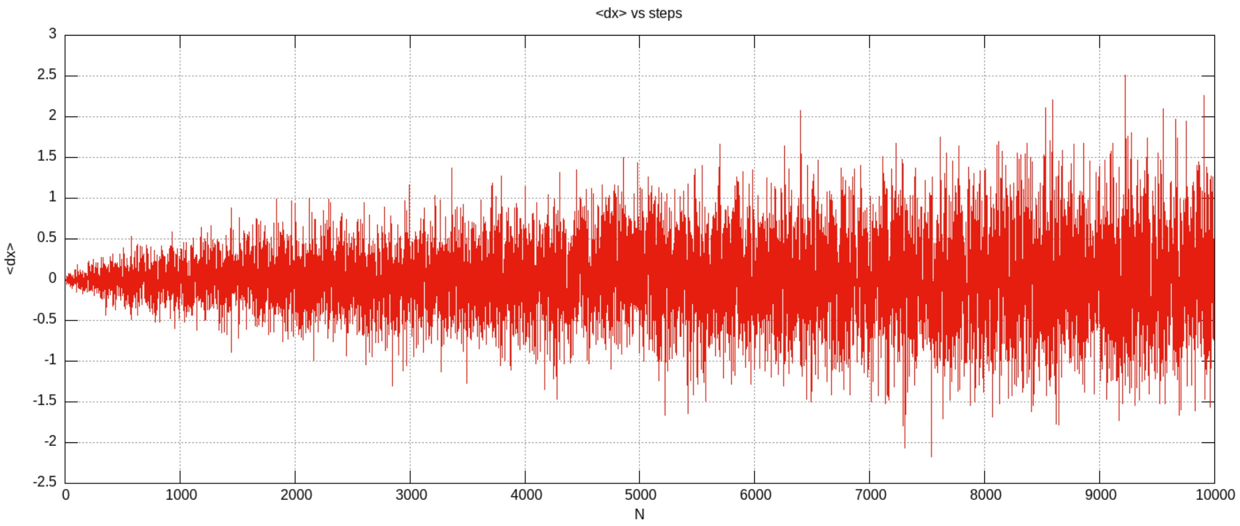


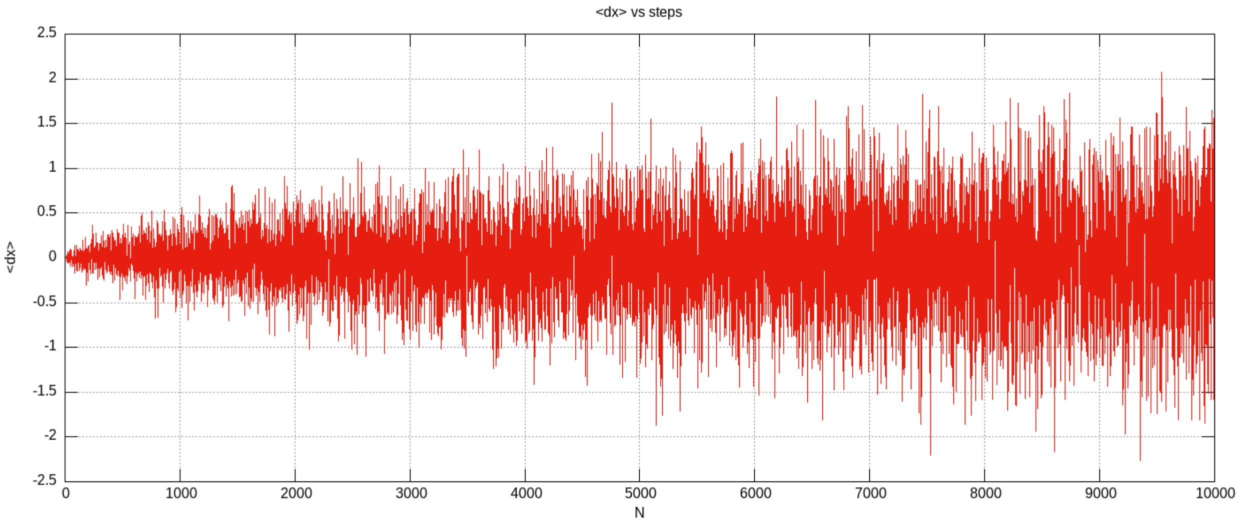
PROBLEM 5 : (Part d)

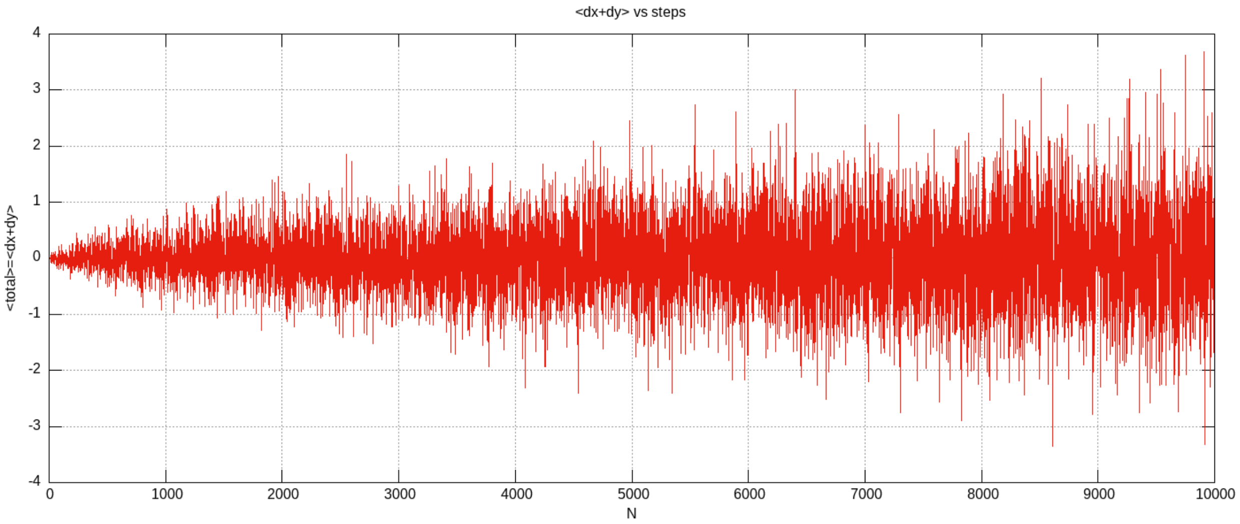
In [ ]:

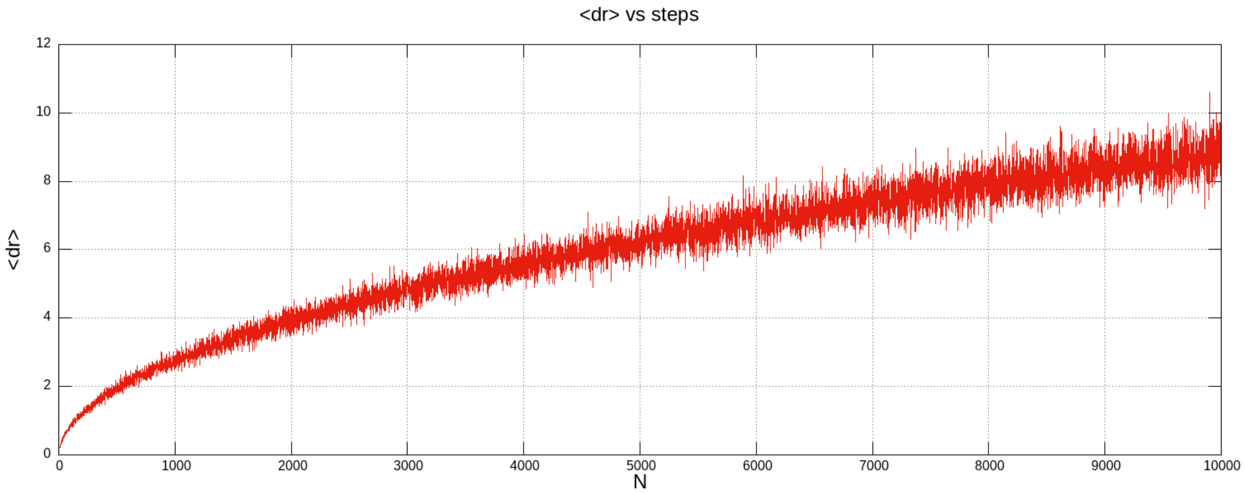
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

int c,c1,c2;
double dx,dy,dr;
//calculating distance dN = x[N]-x[0]
double dN(int N,double arr[N]) {
    return (arr[N-1]-arr[0]);
}
// generating uniform random numbers b/w [0:1] N times
void rwalk(int N,double x[N],double y[N])
{
    int i,j=1,k=1;
    double u[N],h=0.1,K=0.1;
    c1=0,c2=0; // initial value count
    x[0]=0.0;y[0]=0.0;
    for (int i=0;i<N;++i) {
        u[i]=((double)rand()/(double)RAND_MAX);
        //printf("u[%d]:\t%lf\n",i,u[i]);
        if(u[i]<=0.25) {
            x[j]=x[j-1]-h;
            c1++;j++;
        }
        else if (u[i]>0.25 && u[i]<=0.5) {
            x[j]=x[j-1]+h;
            c1++;j++;
        }
        else if(u[i]>0.5 && u[i]<0.75) {
            y[k]=y[k-1]+K;
            c2++;k++;
        }
        else {
            y[k]=y[k-1]-K;
            c2++;k++;
        }
    }
    // rejecting extra value in either array
    if (c1>c2)
        c=c2;
    else
        c=c1;

    dx=dN(c,x);
    dy=dN(c,y);
    dr=sqrt(dx*dx+dy*dy);
}
// calculating mean of an array
double mean(int N,double arr[N]) {
    double sum=0;
    for (int i=0;i<N;++i)
        sum += arr[i];
    return sum/N;
}
// calculating dispersion relation of an array
double sigma(int N,double arr[N]) {
    double avg,std=0;
    avg=mean(N,arr);
    for (int i=0;i<N;++i)
        std += pow((arr[i]-avg),2);
    return sqrt(std/(N-1));
}
int main()
{
    int i,j,N=10000,n=100;
    double x[N],y[N];
    srand(time(0));

    FILE*fp=NULL;
    fp=fopen("5d1.txt","w");
    FILE*fp1=NULL;
    fp1=fopen("5d2.txt","w");
    double d1[N],d2[N],d3[N],d4[N],d5[N];
    double dxn[N],dyn[N],tot[N],drn[N],dr2n[N];
    double sig1[N],sig2[N];
    int k=0;
    for (i=10;i<N;i++) {
        for (j=0;j<n;j++) {
            rwalk(i,x,y);
            d1[j]=dx;
            d2[j]=dy;
            d3[j]=dx+dy;
            d4[j]=dr;
            d5[j]=d4[j]*d4[j];
        }
        dxn[k]=mean(n,d1);
        dyn[k]=mean(n,d2);
        tot[k]=mean(n,d3);
        drn[k]=mean(n,d4);
        dr2n[k]=mean(n,d5);
        sig1[k]=sigma(n,d4);
        sig2[k]=sigma(n,d5);
        fprintf(fp,"%d\t%.4lf\t%.4lf\t%.4lf\n",i,dxn[k],dyn[k],tot[k]);
        fprintf(fp1,"%d\t%lf\t%lf\t%lf\t%lf\t%lf\n",i,drn[k],sig1[k],dr2n[k],sig
        k++;
    }
}
```

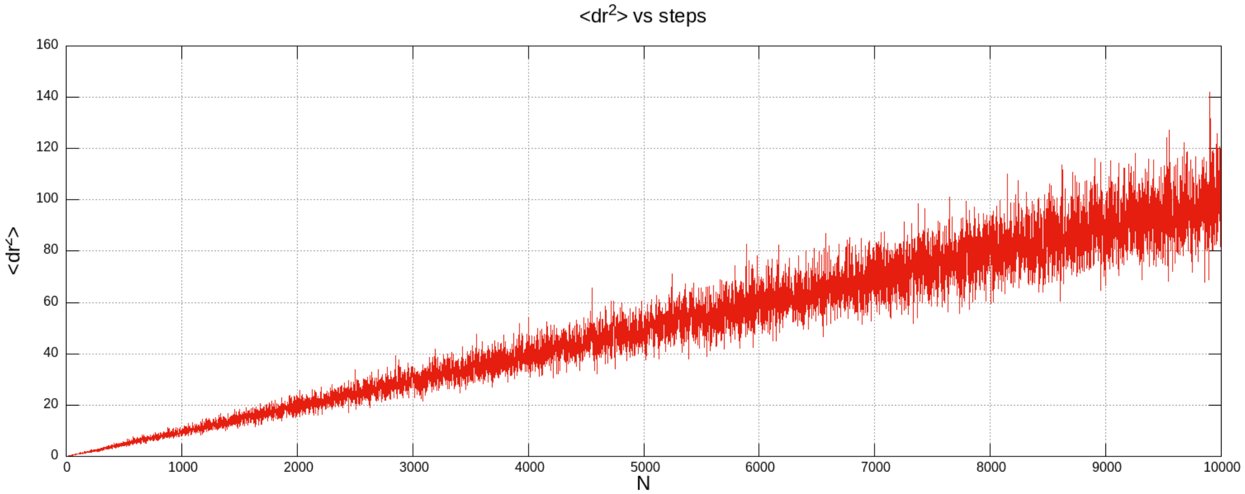


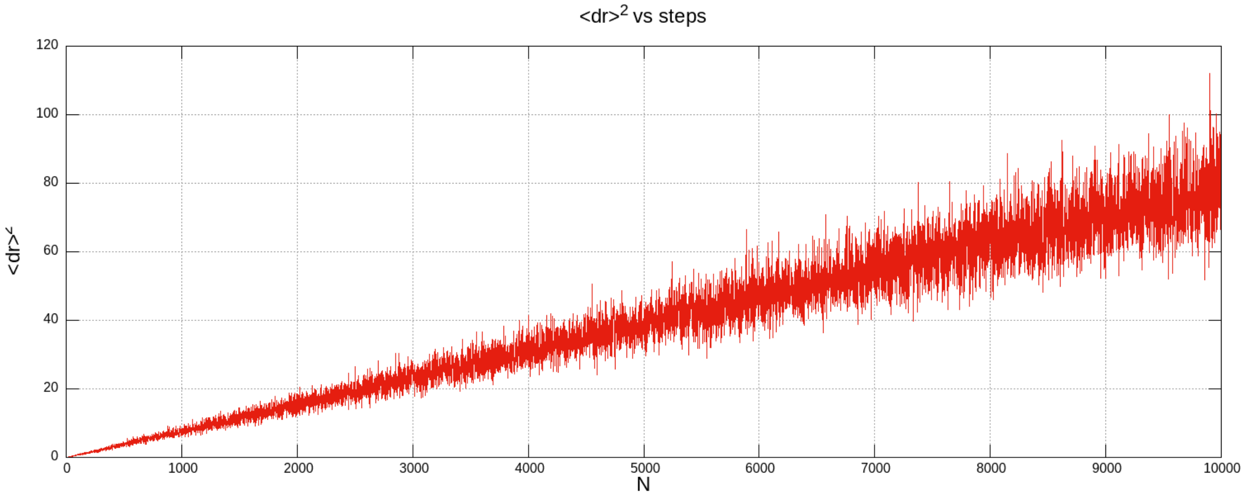

<dx> vs steps
<dx>
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N


<dx> vs steps
<dx>
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N


<dx+dy> vs steps
<total>=<dx+dy>
4
3
2
1
0
-1
-2
-3
-4
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N


<dr> vs steps
<dr>
0
2
4
6
8
10
12
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N


<dr^2> vs steps
<dr^2>
0
20
40
60
80
100
120
140
160
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N


<dr>^{2} vs steps
<dr>^{2}
0
20
40
60
80
100
120
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
N