Υπολογιστική Φυσική Στερεάς Κατάστασης
Set 1

Σεϊτανίδου Δήμητρα

30 Μαρτίου 2020

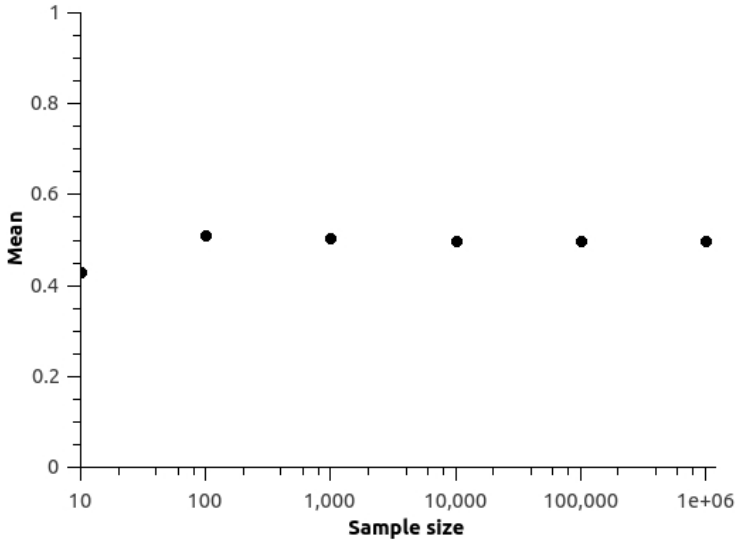
Άσκηση 1

Καλούμαστε να υπολογίσουμε τον μέσο όρο ενός πλήθους τυχαίων αριθμών από την ομοιόμορφη κατανομή (0,1). Ο μέσος όρος υπολογίζεται για δείγματα μεγέθους $N = 100$ μέχρι 10^6. Το πρόγραμμα εκτελεί ένα βρόχο για N επαναλήψεις, αθροίζει τους τυχαίους αριθμούς και στο τέλος βρίσκει τον μέσο όρο από την σχέση:

$$\bar{x} = \frac{1}{N}\sum_{i=0}^{N} x_i \tag{1}$$

Η γραφική παράσταση του μέσου όρου ως προς το N δίνεται παρακάτω.

Σχήμα 1: Μέσος όρος ομοιόμορφης κατανομής.



Από το διάγραμμα παρατηρούμε ότι όσο μεγαλώνει το δείγμα μας, τόσο πιο κοντά είμαστε στην θεωρητική τιμή του μέσου όρου, που είναι $\mu = 0.5$. Ο κώδικας δίνεται παρακάτω.

```cpp
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;

int main(int argc, char const *argv[]) {
  int N[] = {10,100,1000,10000,100000,1000000};
  int i,j;
  double sum;

  srand(4372); // seed for random numbers
  ofstream file("ex1.txt");

  for(i=0;i<6;i++){
    sum = 0;
    for(j=0;j<N[i];j++){
      sum += ((double) rand() / (RAND_MAX));
    }
    file << N[i] << "\t" <<sum/N[i] << endl;
  }

  return 0;
}
```

Ex1.cpp

Άσκηση 2

Πραγματοποιούμε τυχαίο περίπατο για 1000 βήματα. Ο τρόπος που διαλέγουμε τι βήμα θα κάνουμε είναι ο εξής. Στην μια διάσταση έχουμε δύο πιθανές περιπτώσεις, να μετακινηθούμε ένα βήμα μπροστά ή ένα βήμα πίσω. Για κάθε βήμα που θέλουμε να κάνουμε παίρνουμε ένα τυχαίο αριθμό από την ομοιόμορφη κατανομή (0,1). Αν ο αριθμός αυτός είναι ανάμεσα στο 0 και το 0.5, τότε κάνουμε ένα βήμα μπροστά, αλλιώς ένα βήμα πίσω. Στο τέλος υπολογίζουμε το τετράγωνο της μετατόπισης R^2. Η σχέση που χρησιμοποιούμε είναι η:

$$R^2 = (N_1 - N_2)^2 \tag{2}$$

όπου N_1 είναι τα βήματα που κάνουμε προς τα εμπρός και N_2 τα βήματα που κάνουμε προς τα πίσω.

Αντίστοιχα για δύο διαστάσεις έχουμε τέσσερις περιπτώσεις, να κινηθούμε μπρος, δεξιά, πίσω και αριστερά. Έτσι αν ο τυχαίος αριθμός είναι ανάμεσα στο 0 και 0.25 κάνουμε ένα βήμα μπροστά, αν είναι ανάμεσα στο 0.25 και 0.5 ένα βήμα δεξιά, αν είναι στο 0.5 και 0.75 ένα βήμα πίσω και τέλος αν είναι ανάμεσα στο 0.75 και 1 κάνουμε ένα βήμα αριστερά. Οι φορές που κινηθήκαμε προς κάθε κατεύθυνση, με τη σειρά που τις αναφέραμε παραπάνω, είναι N_1, N_2, N_3, N_4 αντίστοιχα και η μετατόπιση δίνεται από τη σχέση:

$$R^2 = (N_1 - N_3)^2 + (N_2 - N_4)^2 \tag{3}$$

Εκτελούμε τα παραπάνω βήματα για 10000 επαναλήψεις και βρίσκουμε τον μέσο όρο του R^2.

Τα αποτελέσματα που βρήκαμε είναι τα εξής:

1. Για μία διάσταση:$< \mathbf{R^2} >= \mathbf{984.694}$
2. Για δύο διαστάσεις: $< \mathbf{R^2} >= \mathbf{981.097}$

Τα αποτελέσματα αυτά είναι κοντά στο 1000, που είναι αυτό που περιμέναμε από την θεωρία. Οι κώδικες δίνονται παρακάτω.

```cpp
#include <iostream>
#include <cstdlib>
#include <cmath>
using namespace std;

// 1D random walk for n steps for m runs

int main(int argc, char const *argv[]) {
  int m=10000;
  int n=1000;
  int i,j;
  int N1,N2;
  double x,sum,R[m];

  srand(4372);

  for(j=0;j<m;j++){
    N1=N2=0;
    for(i=0;i<n;i++){
      x = ((double) rand() / (RAND_MAX)); //random number in range (0,1)
      if(x<0.5){
        N1 += 1; //steps foreward
      } else{
        N2 += 1; //steps backward
      }
    }
    R[j] = pow(N1-N2,2);
  }

  // mean calculation
  sum = 0;
  for(i=0;i<m;i++){
    sum +=R[i];
  }

  cout << sum/m << endl;

  return 0;
}
```

Ex2_1D.cpp

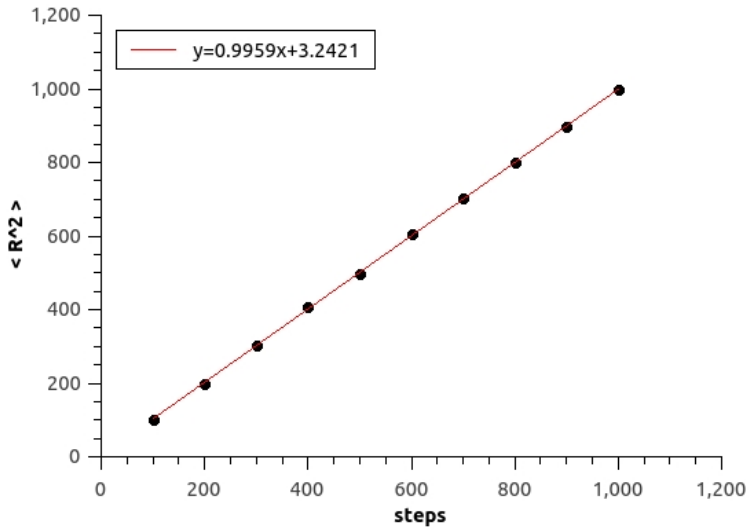
```cpp
#include <iostream>
#include <cstdlib>
#include <cmath>
using namespace std;

// 2D random walk for n steps for m runs

int main(int argc, char const *argv[]) {
  int m=10000;
  int n=1000;
  int i,j;
  int N1,N2,N3,N4;
  double x,sum,R[m];

  srand(4372);

  for(j=0;j<m;j++){
    N1=N2=N3=N4=0;
    for(i=0;i<n;i++){
      x = ((double) rand() / (RAND_MAX));
      if(x<0.25){
        N1 += 1; //steps foreward
      } else if(x<0.5){
        N2 += 1; //steps right
      } else if(x<0.75){
        N3 +=1; //steps backward
      } else {
        N4 +=1; //steps left
      }
    }
      R[j] = pow(N1-N3,2) + pow(N2-N4,2);
  }

  // mean calculation
  sum = 0;
  for(i=0;i<m;i++){
    sum +=R[i];
  }

  cout << sum/m << endl;

  return 0;
}
```

Ex2_2D.cpp

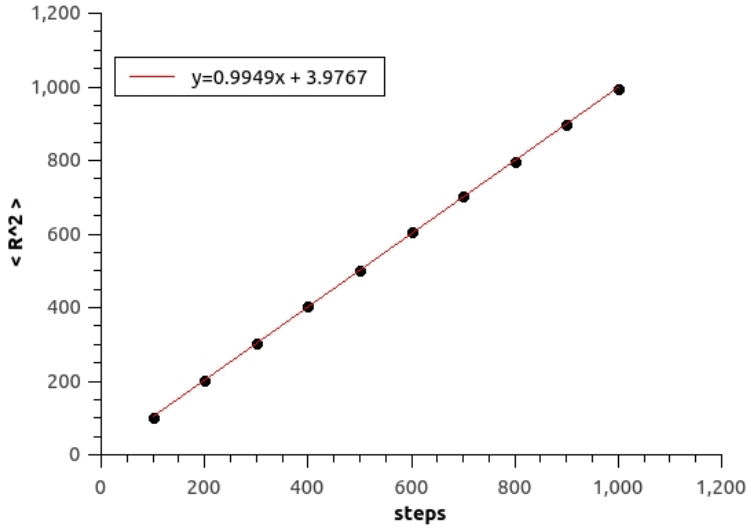
Άσκηση 3

Σε αυτή την άσκηση επαναλαμβάνουμε όσα κάναμε στην άσκηση 2, μόνο που αυτή τη φορά αντί να υπολογίζουμε τον μέσο όρο του R^2 για κάθε run, υπολογίζουμε τον μέσο όρο κάθε 100 βήματα για όλα τα runs. Θα έχουμε δηλαδή ένα $< R^2 >$ για 100 βήματα, ένα για 200 κ.ο.κ. Για να το κάνουμε αυτό κρατάμε σε ένα δισδιάστατο πίνακα τις τιμές του R^2 για κάθε 100 βήματα, για 10000 επαναλήψεις, έτσι ώστε η κάθε γραμμή του πίνακα να αντιστοιχεί στα 100,200,... βήματα. Τέλος υπολογίσουμε τον μέσο όρο. Οι τιμές που βρήκαμε στην μία και στις δύο διαστάσεις δίνονται στις παρακάτω γραφικές παραστάσεις.

Σχήμα 2: Μέσος όρος R^2 για τυχαίο περίπατο στην μια διάσταση.



Σχήμα 3: Μέσος όρος R^2 για τυχαίο περίπατο στις δύο διαστάσεις.



Μαζί με τα σημεία στην γραφική παράσταση δίνεται και η ευθεία ελαχίστων τετραγώνων που αντιστοιχεί στα σημεία. Βλέπουμε ότι και στην μια και στις δύο διαστάσεις η κλίση της

ευθείας είναι περίπου ίση με 1, κάτι που αναμέναμε να δούμε, αφού το $< R^2 >$ ισούται με τον αριθμό των βημάτων. Οι κώδικες δίνονται παρακάτω.

```cpp
#include <fstream>
#include <cstdlib>
#include <cmath>
using namespace std;

int main(int argc, char const *argv[]) {  // 1D random walk
  int i,j,N1,N2;
  int n=1000;
  int m=10000;
  double list[11][m];  // matrix to keep R^2
  double x,R,sum;

  srand(4372);
  ofstream f("ex3_1d.txt");

  for(i=0;i<m;i++){
    N1 = N2 = 0;

    for(j=0;j<n+1;j++){
      x = ((double) rand() / (RAND_MAX));
      if(x<0.5){
        N1 += 1;  //steps foreward
      }else {
        N2 += 1; //steps backward
      }
      if(j%100==0){
        R = pow(N1-N2,2);
        list[j/100][i] = R; // keep R^2 for every 100 steps
      }
    }
  }

// mean calculation
for(j=1;j<11;j++){
    sum = 0;
    for(i=0;i<m;i++){
      sum += list[j][i];
    }
    f << j*100 << "\t" << sum/m << endl;
}
  return 0;
}
```

Ex3_1D.cpp

```cpp
#include <fstream>
#include <cstdlib>
#include <cmath>
using namespace std;

int main(int argc, char const *argv[]) {  // 2D random walk
  int i,j,N1,N2,N3,N4;
  int n=1000;
  int m=10000;
  double list[11][m];  // matrix to keep R^2
  double x,R,sum;

  srand(4372);
  ofstream f("ex3_2d.txt");

  for(i=0;i<m;i++){
    N4 = N1 = N2 = N3 = 0;

    for(j=0;j<n+1;j++){
      x = ((double) rand() / (RAND_MAX));
      if(x<0.25){
        N1 += 1; //steps foreward
      }else if(x<0.5){
        N2 += 1; //steps right
      }else if(x<0.75){
        N3 += 1; //steps backward
      }else {
        N4 += 1; //steps left
      }
      if(j%100==0){
        R = pow(N1-N3,2) + pow(N2-N4,2);
        list[j/100][i] = R; // keep R^2 for every 100 steps
      }
    }
  }

// mean calculation
for(j=1;j<11;j++){
    sum = 0;
    for(i=0;i<m;i++){
      sum += list[j][i];
    }
    f << j*100 << "\t" << sum/m << endl;
}
  return 0;
}
```

Ex3_2D.cpp

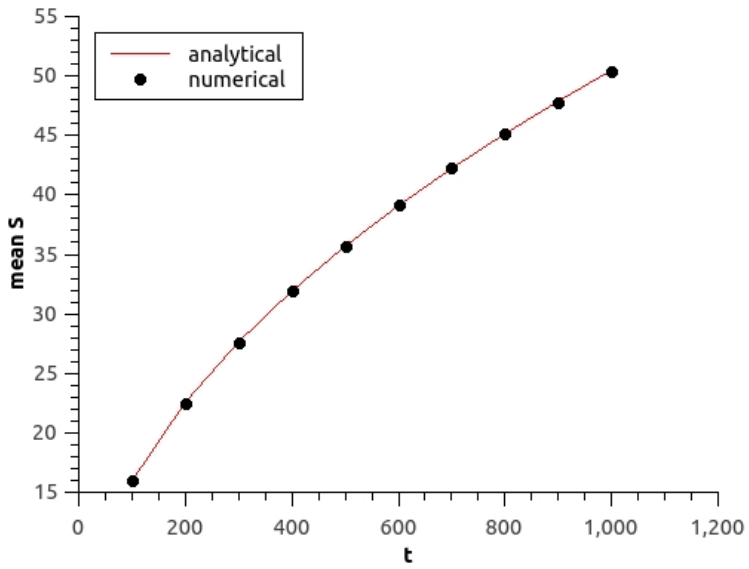
Άσκηση 5

Σε αυτή την άσκηση θέλουμε να καταγράψουμε από πόσες θέσεις έχει περάσει τουλάχιστον μια φορά ένα σωματίδιο που κάνει τυχαίο περίπατο. Για να το κάνουμε αυτό δημιουργούμε ένα μονοδιάστατο, δισδιάστατο ή τρισδιάστατο πλέγμα που αναπαριστά τον χώρο στον οποίο κινείται το σωματίδιο και ορίζουμε με 0 όλα τα σημεία από τα οποία δεν έχει περάσει ακόμα το σωματίδιο. Επιλέγουμε τι βήμα θα κάνει σύμφωνα με το όσα είπαμε στις προηγούμενες ασκήσεις. Αν το σωματίδιο πάει σε σημείο στο οποίο δεν έχει ξανά επισκεφτεί αλλάζουμε το 0 σε 1 και κρατάμε σε μία μεταβλητή πόσες φορές το κάναμε αυτό. Η μεταβλητή αυτή είναι το ζητούμενο S. Κρατάμε το S για κάθε 100 βήματα, όπως στην άσκηση 3, και τέλος υπολογίζουμε τον μέσο όρο. Οι γραφικές παραστάσεις του $< S >$ με τον χρόνο για μια, δυο και τρεις διαστάσεις δίνονται παρακάτω. Στις γραφικές παραστάσεις, πέρα από τον παραπάνω αριθμητικό υπολογισμό του $< S >$, δείχνουμε επίσης και την αναλυτική τιμή του, που υπολογίζετε σύμφωνα με τους παρακάτω τύπους:

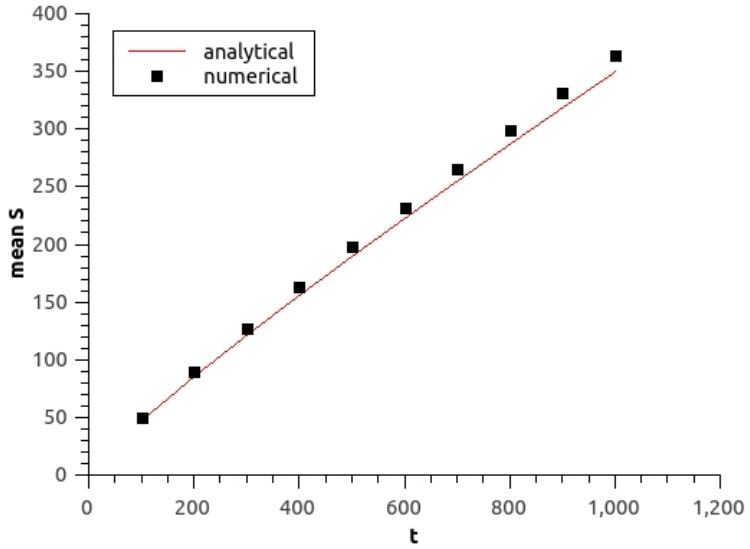
- $1D$: $S_N \cong (\frac{8N}{\pi})^{1/2}$
- $2D$: $S_N \cong \frac{\pi N}{ln(8N)}$
- $3D$: $S_N \cong 0.66N$

Στον τύπο για τις δύο διαστάσεις κάναμε μια μικρή διόρθωση (προσθέσαμε το 8 στον παρονομαστή) για να έχουμε καλύτερα αποτελέσματα. Να σημειωθεί ότι στην μια διάσταση χρησιμοποιήσαμε ένα χώρο με 1000 θέσεις, στις δυο διαστάσεις ένα χώρο με 1000×1000 θέσεις και στις τρεις διαστάσεις ένα χώρο με 250×250×250 θέσεις.

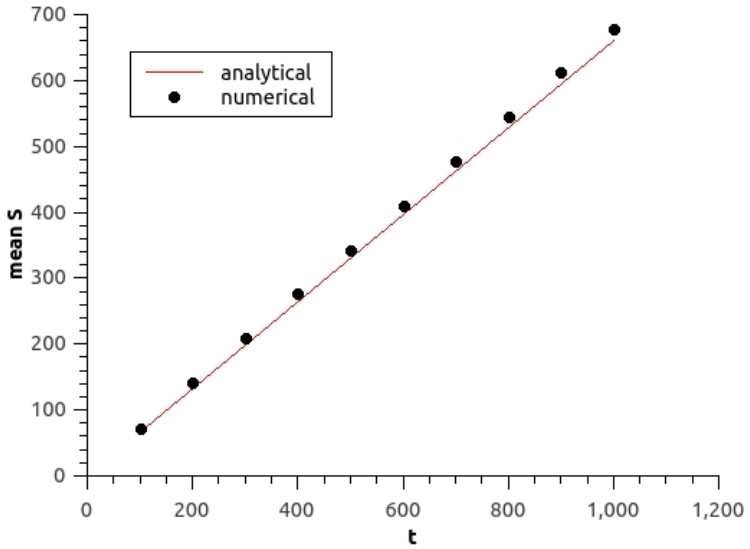
Σχήμα 4: Μέσος όρος S για τυχαίο περίπατο στην μία διάσταση.



Σχήμα 5: Μέσος όρος S για τυχαίο περίπατο στις δύο διαστάσεις.



Σχήμα 6: Μέσος όρος S για τυχαίο περίπατο στις τρεις διαστάσεις.



Από τα παραπάνω διαγράμματα παρατηρούμε ότι οι τιμές του $< S >$ αλλάζουν και όσο παίρνουμε περισσότερες διαστάσεις τόσο αυξάνεται το $< S >$. Αυτό το αποτέλεσμα είναι αναμενόμενο, αφού όσο περισσότερες διαστάσεις έχουμε τόσο πιο πολύ χώρο έχει να κινηθεί το σωματίδιο και περισσότερες θέσεις που μπορεί να επισκεφτεί. Επίσης οι αναλυτικές με τις αριθμητικές τιμές, βλέπουμε ότι έχουν αρκετά καλή συμφωνία. Οι κώδικες δίνονται παρακάτω.

```
#include <fstream>
#include <cstdlib>
#include <vector>
using namespace std;

int main(int argc, char const *argv[]) {
  int i,j,i0;
  int n=1000;
  int m=10000;
  double x,sum;
  double list[11][m];
  vector<int> tiles; // 1D grid

  srand(4372);
  ofstream file("ex5_1d.txt");

  for(j=0;j<m;j++){
    i0 = 500;
    tiles.assign(n,0); //intial values of grid = 0
    sum = 0;

    for(i=0;i<n+1;i++){
      x = ((double) rand() / (RAND_MAX));
      if(x<0.5){
        if(tiles[i0+1]==0){ //if tile hasn't been visited before change 0
    to 1
          tiles[i0+1] = 1;
          sum += 1;
        }
        i0 = i0+1; //particle's new position
      }else {
        if(tiles[i0-1]==0){
          tiles[i0-1] = 1;
          sum += 1;
        }
        i0 = i0-1;
      }
      if(i%100==0){
        list[i/100][j] = sum; //keep S for every 100 steps
      }
    }

// mean calculation
  for(i=1;i<11;i++){
      sum = 0;
      for(j=0;j<m;j++){
        sum += list[i][j];
      }
```

```
      file << i*100 << "\t" << sum/m << endl;
  }

  return 0;
}
```

Ex5_1D.cpp

```
#include <fstream>
#include <cstdlib>
#include <vector>
using namespace std;

int main(int argc, char const *argv[]) {
  int i,j,i0,j0;
  int n=1000;
  int m=10000;
  double x,sum;
  double list[11][m];
  vector<vector<int> > tiles(n,vector<int> (n)); //2D grid

  srand(4372);
  ofstream file("ex5_2d.txt");

  for(j=0;j<m;j++){
    i0 = 500;
    j0 = 500;
    tiles.assign(n, vector < int >(n, 0)); //intial values of grid = 0
    sum = 0;

    for(i=0;i<n+1;i++){
      x = ((double) rand() / (RAND_MAX));
      if(x<0.25){
        if(tiles[i0][j0+1]==0){ //if tile hasn't been visited before
          tiles[i0][j0+1] = 1;  //change 0 to 1
          sum += 1;
        }
        j0 = j0+1; //particle's new position
      }else if(x<0.5){
        if(tiles[i0+1][j0]==0){
          tiles[i0+1][j0] = 1;
          sum += 1;
        }
        i0 = i0+1;
      }else if(x<0.75){
        if(tiles[i0][j0-1]==0){
          tiles[i0][j0-1] = 1;
          sum += 1;
        }
        j0 = j0-1;
```

```
      }else {
        if(tiles[i0-1][j0]==0){
          tiles[i0-1][j0] = 1;
          sum += 1;
        }
        i0 = i0-1;
      }
      if(i%100==0){
        list[i/100][j] = sum; //keep S for every 100 steps
      }
    }
  }

// mean calculation
  for(i=1;i<11;i++){
      sum = 0;
      for(j=0;j<m;j++){
        sum += list[i][j];
      }
      file << i*100 << "\t" << sum/m << endl;
  }

  return 0;
}
```

Ex5_2D.cpp

```
#include <fstream>
#include <cstdlib>
#include <vector>
using namespace std;

int main(int argc, char const *argv[]) {
  int i,j,i0,j0,k0;
  int n=1000;
  int m=10000;
  int v=250;
  double x,sum;
  double list[11][m];
  vector<vector<vector<int> >> tiles(v,vector<vector<int>> (v, vector<int
   > (v))); //3D grid

  srand(4372);
  ofstream file("ex5_3d.txt");

  for(j=0;j<m;j++){
    i0 = 125;
    j0 = 125;
    k0 = 125;
    tiles.assign(v,vector<vector<int>> (v, vector<int> (v,0)));//intial
```

```
  values of grid = 0
   sum = 0;

   for(i=0;i<n+1;i++){
     x = ((double) rand() / (RAND_MAX));
     if(x<1.0/6){
       if(tiles[i0][j0+1][k0]==0){ //if tile hasn't been visited before
         tiles[i0][j0+1][k0] = 1;  //change 0 to 1
         sum += 1;
       }
       j0 = j0+1; //particle's new position
     }else if(x<2.0/6){
       if(tiles[i0+1][j0][k0]==0){
         tiles[i0+1][j0][k0] = 1;
         sum += 1;
       }
       i0 = i0+1;
     }else if(x<3.0/6){
       if(tiles[i0][j0][k0+1]==0){
         tiles[i0][j0][k0+1] = 1;
         sum += 1;
       }
       k0 = k0+1;
     }else if(x<4.0/6){
       if(tiles[i0][j0-1][k0]==0){
         tiles[i0][j0-1][k0] = 1;
         sum += 1;
       }
       j0 = j0-1;
     }else if(x<5.0/6){
       if(tiles[i0-1][j0][k0]==0){
         tiles[i0-1][j0][k0] = 1;
         sum += 1;
       }
       i0 = i0-1;
     }else {
       if(tiles[i0][j0][k0-1]==0){
         tiles[i0][j0][k0-1] = 1;
         sum += 1;
       }
       k0 = k0-1;
     }
     if(i%100==0){
       list[i/100][j] = sum; //keep S for every 100 steps
     }
   }
 }
```

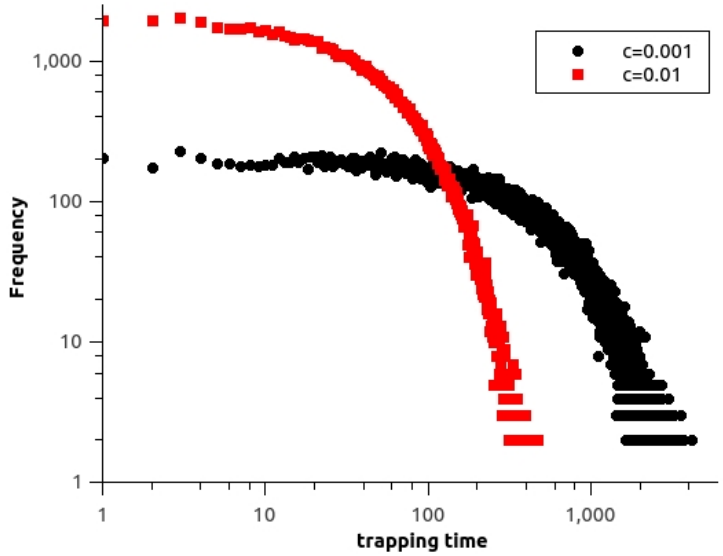
```
// mean calculation
  for(i=1;i<11;i++){
      sum = 0;
      for(j=0;j<m;j++){
        sum += list[i][j];
      }
      file << i*100 << "\t" << sum/m << endl;
  }

  return 0;
}
```

Ex5_3D.cpp

Άσκηση 6

Έχουμε ένα σωματίδιο που εκτελεί τυχαίο περίπατο σε ένα δισδιάστατο πλέγμα 500×500, το οποίο περιέχει παγίδες στις οποίες αν πέσει το σωματίδιο τερματίζεται ο περίπατος του. Το πόσα βήματα θα κάνει το σωματίδιο μέχρι να πέσει σε παγίδα ονομάζεται χρόνος παγίδευσης. Στο πρόγραμμα που τρέχουμε εκτελούμε αυτόν τον τυχαίο περίπατο, με την επιλογή των βημάτων όπως και στα προηγούμενα προγράμματα, 100000 φορές και καταγράφουμε τους χρόνους παγίδευσης. Κάθε φορά τοποθετούμε το σωματίδιο σε τυχαία θέση. Αν η τυχαία αυτή θέση έχει παγίδα τότε ο χρόνος παγίδευσης είναι μηδέν. Επιπλέον πρέπει να ελέγχουμε αν το σωματίδιο πάει να ξεφύγει από το πλέγμα. Αν το σωματίδιο βρίσκεται σε οριακή θέση, πχ στην 500 στήλη και πάει να κινηθεί δεξιά, το τοποθετούμε στην πρώτη στήλη στην αντίστοιχη σειρά. Για να υπολογίσουμε την κατανομή των χρόνων παγίδευσης, χρησιμοποιούμε μια δομή που ονομάζεται unordered map και αποθηκεύει δεδομένα που σχηματίζονται από ένα κλειδί και την αντίστοιχη τιμή του. Σαν κλειδί χρησιμοποιούμε τους χρόνους παγίδευσης και η αντίστοιχη τιμή είναι η συχνότητα εμφάνισης του. Το μόνο που μας μένει μετά είναι να κάνουμε τη γραφική παράσταση. Το διάγραμμα με την κατανομή των χρόνων παγίδευσης για δύο διαφορετικές συγκεντρώσεις παγίδων, $c = 10^{-2}$ και $c = 10^{-3}$, δίνεται παρακάτω.

Σχήμα 7: Κατανομή χρόνων παγίδευσης

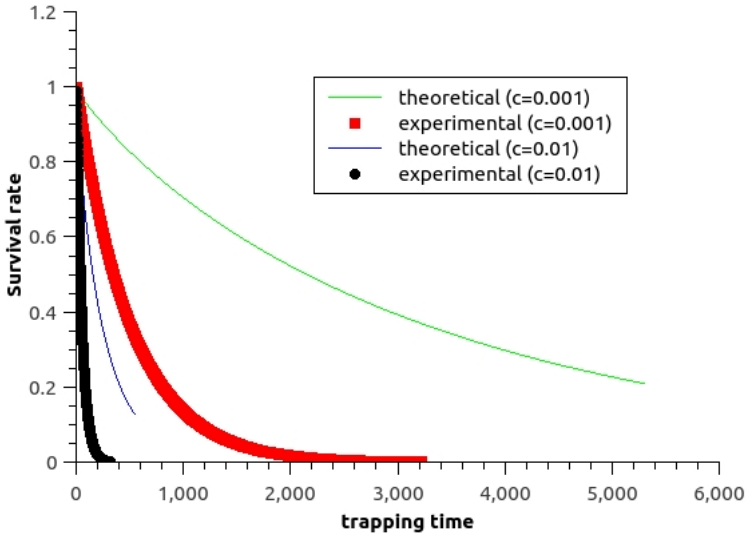


Από το σχήμα βλέπουμε ότι γενικά τα περισσότερα σωματίδια παγιδεύονται γρήγορα, με τη κατανομή να είναι μεγαλύτερη για μικρούς χρόνους παγίδευσης και μικρή για μεγάλους χρόνους παγίδευσης. Αν συγκρίνουμε τις δύο συγκεντρώσεις βλέπουμε ότι για την μεγαλύτερη συγκέντρωση παγίδων περισσότερα σωματίδια παγιδεύονται σε μικρούς χρόνους, αναμενόμενο αφού η πιθανότητα να συναντήσουν παγίδα είναι μεγαλύτερη. Επιπλέον οι χρόνοι παγίδευσης για $c = 10^{-2}$ δεν ξεπερνούν την τιμή 600, ενώ για $c = 10^{-3}$ οι χρόνοι παγίδευσης ξεπερνούν το 1000, που σημαίνει ότι τα σωματίδια κάνουν περισσότερο χρόνο για να συναντήσουν παγίδα, επίσης αναμενόμενο αφού μικρότερη συγκέντρωση σημαίνει μικρότερη πιθανότητα να πέσει σε παγίδα.
Εκτός από την κατανομή των χρόνων παγίδευσης, υπολογίσαμε και την πιθανότητα επιβίωσης, δηλαδή το ποσοστό των σωματιδίων που δεν έχουν πέσει σε παγίδα σε συνάρτηση με το χρόνο. Το ποσοστό αυτό είναι το υπολειπόμενο ποσοστό της σχετικής αθροιστικής συχνότητας των χρόνων παγίδευσης. Για να υπολογίσουμε τις αθροιστικές συχνότητες χρησιμοποιούμε το unordered map που φτιάξαμε για τους χρόνους παγίδευσης και τοποθετούμε τις τιμές του σε ένα set, που είναι μια δομή που τα στοιχεία της είναι μοναδικά και ταξινομημένα. Έπειτα αθροίζουμε την κάθε συχνότητα με τις προηγούμενες για να βρούμε τις αθροιστικές συχνότητες και τέλος την ζητούμενη πιθανότητα επιβίωσης. Στο παρακάτω διάγραμμα εμφανίζονται οι πιθανότητες επιβίωσης για τις δύο συγκεντρώσεις παγίδων ($c = 10^{-2}$ και $c = 10^{-3}$) καθώς και η θεωρητική τιμή της πιθανότητας επιβίωσης που δίνεται από την σχέση:

$$\Phi(t) = (1 - c)^{<S(t)>} \tag{4}$$

όπου $< S(t) >$ είναι ο μέσος όρος των πλεγματικών θέσεων που επισκέφτηκε το σωματίδιο τουλάχιστον μια φορά και δίνεται από την τύπο που αναφέραμε στην άσκηση 5, για την περίπτωση των δύο διαστάσεων.

Σχήμα 8: Πιθανότητα επιβίωσης



Συγκρίνοντας τις θεωρητικές με τις πειραματικές τιμές, δηλαδή τις τιμές που υπολογίσαμε από τον κώδικα, βλέπουμε ότι οι καμπύλες έχουν την ίδια τάση που μοιάζει με εκθετική μείωση, αν και δεν συμπίπτουν. Η τάση αυτή είναι ένα λογικό αποτέλεσμα γιατί όσο περνάει ο χρόνος, τόσο περισσότερα σωματίδια θα παγιδεύονται. Τώρα αν συγκρίνουμε τις συγκεντρώσεις των παγίδων βλέπουμε ότι η πιθανότητα επιβίωσης μηδενίζεται πολύ πιο γρήγορα για $c = 10^{-2}$ και φτάνει χρόνους που δεν ξεπερνούν το 500. Αντίθετα για $c = 10^{-3}$ η πιθανότητα επιβίωσης μηδενίζεται πολύ αργότερα, και με μικρότερο ρυθμό, και φτάνει χρόνους που ξεπερνούν το 3000. Όλα αυτά επίσης είναι λογικά αποτελέσματα, γιατί όπως είπαμε μικρότερη συγκέντρωση παγίδων σημαίνει μικρότερη πιθανότητα να παγιδευτεί το σωματίδιο αρά και μεγαλύτερη πιθανότητα επιβίωσης.
Ο κώδικας δίνεται παρακάτω.

```
#include <fstream>
#include <cstdlib>
#include <vector>
#include <bits/stdc++.h>
using namespace std;

int main(int argc, char const *argv[]) {
  int n=500;
  int m=100000;
  int i,l,iidx,jidx,i0,j0,sum;
  double x,c;
  double cumul = 0.0;
  vector <double> times(m);
  vector <vector<int> > grid(n,vector <int> (n));
  unordered_map<int, int> freq;

  grid.assign(n, vector < int >(n, 0));
  ofstream file1("ex6_freq.txt");
```

```
ofstream file2("ex6_cumul.txt");

srand(4372);
c = 0.001; // trap density
l = c*n*n; // number of traps

for(i=0;i<l;i++){ // set traps
  iidx = rand()%n;
  jidx = rand()%n;
  grid[iidx][jidx] = -1;
}

for(i=0;i<m;i++){
  sum = 0; // number of steps
  do {
    i0 = rand()%n;
    j0 = rand()%n;
    if(grid[i0][j0]==-1){ // check if particle is placed in trap
      break;
    }
    x = ((double) rand() / (RAND_MAX));
    if(x<0.25){
      if(j0==n-1){ // check if we are on the edge of the grid
        j0 = 0;
      } else{
        j0 = j0+1;
      }
    } else if(x<0.5){
      if(i0==n-1){
        i0 = 0;
      } else{
        i0 = i0+1;
      }
    } else if(x<0.75){
      if(j0==0){
        j0 = n-1;
      } else{
        j0 = j0-1;
      }
    } else {
      if(i0==0){
        i0 = n-1;
      } else{
        i0 = i0-1;
      }
    }
    sum += 1;
  } while(grid[i0][j0]!=-1);
```

```
    times[i] = sum; // trapping time
  }

  // trapping time frequencies
  for(i=0;i<m;i++)
    freq[times[i]]++;

  for(auto x : freq)
    file1 << x.first << "\t" << x.second << endl;

  // cumulative frequencies
  set<pair<int, int> > st;

  for(auto x : freq)
    st.insert({ x.first, x.second });

  for(auto x : st){
    cumul += x.second;
    file2 << x.first << "\t" << 1-cumul/m << endl;
  }
  return 0;
}
```

Ex6.cpp