A Simulation and a Study of Random Walks

Mohamed Hamada
Language Processing Lab.
The University of Aizu
Aizu Wakamatsu, Japan
E-mail: hamada@u-aizu.ac.jp

Abstract: - Random walk model has many useful applications in the study of stochastic (probabilistic) processes including: modeling the transport of molecules in physics, modeling the locomotion of organisms in biology, and modeling the behavior over time of financial markets in economy. In this paper we introduce a visualized simulation of a random walk in d-dimensional lattice. We also introduce a numerical analysis of the d-dimensional lattice walk. A two- and three-dimension off-lattice walks are also discussed. In addition to simulating self-avoiding walks.

Key-Words: - On-lattice random walk, self-avoiding walk, simulation, visualization.

1 Introduction

In this paper we are concerned about off-lattice and on-lattice random walks. The off-lattice random walk goes back to Karl Pearson who published a query in *Nature* entitled *The Problem of the Random Walker* [7], asking for the probability of a random walker, with equal distance walks in any direction, (after n steps) to be at a distance, from the starting point, bounded by certain values. We will study and implement this model for two and three dimension. We also consider on-lattice random walk (in the sequel referred as lattice random walk). We study the general form of this model, that is called d-dimensional lattice walk. For simplicity we will assume that the length of steps are fixed and equal to the unity. Now let a point walk around at random on this lattice. Upon reaching any vertex of the graph, the probability of choosing any one of the $2d$ edges leading out of that vertex is $1/2d$. When $d = 1$, i.e. one-dimensional random walk, the lattice is just an infinite line divided into segments of length one. When $d = 2$, i.e. two-dimensional random walk, the lattice looks like an infinite network of streets and avenues. When $d = 3$, i.e. three-dimensional random walk, the lattice looks like an infinite "jungle gym." It is worth noting that when $d = 3$, the wandering of our walking point could be regarded as an approximate representation of the random path of a molecule diffusing in an infinite cubical crystal. We also consider another model of random walks, that is, *self-avoiding walk* (SAW). SAW differs from other models of random walks in such a way that it never visits a previously visited point.

In spite of the simplicity of the notion of the random walk it is remarkable that the general idea, as well as ramifications and specific implementations of the general idea, play such a crucial role in so many scientific disciplines.

The paper is organized as follows. In section 2 we introduce the implementation of two- and three-dimensional off-lattice walk. The implementation of d-dimensional lattice walk will be given in section 3. In section 4 we discus the numerical analysis of both ran-

dom walks and introduce implementation to compute related values such as *mean square end to end distance* and *radius of gyration*. In section 5 a visualization of several walks is introduced. The simulation of SAW is given in section 6. Finally, in section 7 we conclude our paper and show other related works. In this paper we use the powerful system Mathematica version 4 in our implementation. So we will assume some familiarity with Mathematica. A comprehensive guide can be found in [8]. For recent work and applications on randomness we refer to [9].

2 Off-Lattice Random Walk

In this section we discuss the off-lattice random walk. In this model the walker starts at any point and walk in any direction for a unit distance. Then he turns for any angle whatever and walk for another unit distance. He repeats this process n times. We will study this model in two and three dimensions.

2.1 Off-lattice walk in 2-dimension

The off-lattice random walk model in two-dimension (it is also called Pearson's random walk) can be described as follows. A walker starts from a point p and walks one unit in a straight line, he then turns through any angle (between 0 and 360 degrees (2π)) whatever and walks another unit in a second straight line. He repeats this process n times. To implement this model we need to use polar coordinates instead of the Cartesian one. A point (x, y) in Cartesian coordinates can be transformed into polar coordinates as follows: $x = r.cos(\phi)$ and $y = r.sin(\phi)$, where r is the distance walked and ϕ is the angle of motion. Since we use the unit distance, $r = 1$. Hence the Cartesian coordinates point (x, y) will be represented by the polar coordinates $(cos(\phi), sin(\phi))$.

The corresponding Mathematica program can be given as follows.

2.1.1 The program

```
OffLatticeWalk2D[steps_] :=
      Map[Flatten,
          FoldList[Plus, {0.0, 0.0},
          Map[{Cos[#], Sin[#]}&,
              Table[{Random[Real,
                    {0, N[2Pi]}]},
                    {steps}]]
              ]
         ]
```

A typical run of this program for 10 steps is:

```
OffLatticeWalk2D[10]
```

which produce the following list of steps:

```
{{0., 0.},      {-0.919224, 0.393734},
 {-1.68058, 1.0421}, {-1.5205, 0.054},
 {-0.5231, 0.1257},  {-0.8532, 1.069},
 {-0.9396, 2.0659}, {-1.2547, 3.0149},
 {-1.75684, 2.1501}, {-2.024, 3.1138},
 {-1.03713, 2.95233}}
```

Here the first point in the list (i.e.,`{0., 0.}`) represents the starting point (i.e. the original). The other 10 points represent the random 10 steps in the walk.

2.2 Off-lattice walk in 3-dimension

The off-lattice random walk model in three-dimension can be described as follows. A walker starts from a point p and walks one unit in a straight line in the space, he then turns through any angle (between 0 and 360 degrees (2π)) whatever and walks another unit in a second straight line. He repeats this process n times. To implement this model we need to use spherical coordinates instead of the Cartesian one. A point (x, y, z) in Cartesian coordinates can be transformed

into spherical coordinates as follows (see figure 1 below): $x = r.sin(b).cos(a)$, $y = r.sin(b).sin(a)$, and $z = r.sin(b)$, where r is the distance walked, $0 \leq a \leq 2\pi$ is the angle between the positive x-axis and the projection onto the (x, y) plane (also called *geographical longitude*), and $0 \leq b \leq \pi$ is the angle between the polar axis (positive z-axis) and the latitude coordinate. Since we use the unit distance, $r = 1$. Hence the Cartesian coordinates point (x, y, z) will be represented in the spherical coordinates by the point $(sin(b).cos(a), sin(b).sin(a), cos(b))$.

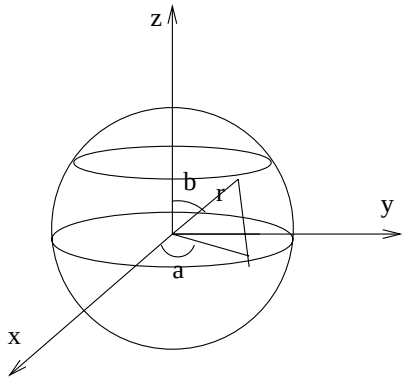

Fig. 1 Spherical vs. Cartesian coordinates

The corresponding program can be written in Mathematica as follows.

2.2.1 The program

```
OffLatticeWalk3D[steps_] :=
 FoldList[Plus, {0.0, 0.0, 0.0},
         Apply[{Sin[#]Cos[#],
               Sin[#]Sin[#], Cos[#]}&,
               Table[{Random[Real,
                         {0, N[2Pi]}],
                      Random[Real,
                         {0, N[Pi]}] },
                    {steps}],
                {1}]]
```

A typical run of this program for 10 steps is:

```
OffLatticeWalk3D[10]
```

which produces the following list of steps:

```
{{0., 0., 0.}, {0.2979, 0.901, 0.3137},
 {0.191, 1.8909, 0.421},
 {0.32076, 1.90709, -0.570369},
 {-0.162251, 2.27786, 0.222871},
 {0.178655, 2.41209, -0.707593},
 {0.525812, 3.27193, -0.3338},
 {0.331571, 3.3112, 0.64696},
 {0.310621, 4.31076, 0.667914},
 {0.64044, 5.18655, 1.02035},
 {0.345477, 5.28282, 0.0696997}}
```

3 d-Dimensional Lattice Walk

In this section we introduce the implementation of the d-dimensional lattice walk. We will consider lattice walks on the d-dimension. Following [1], the d-dimensional lattice walk can be constructed as follows. To construct a d-dimensional lattice, we take as vertices those points $(x_1, \cdots, x_d)$ of $\mathbf{R}^d$ all of whose coordinates are integers, and we join each vertex by an undirected line segment to each of its $2d$ nearest neighbors. These connecting segments, which represent the edges of our graph, each have unit length and run parallel to one of the coordinate axes of $\mathbf{R}^d$. This walk consists of steps of uniform length. At any lattice point, the walk is randomly taken in one of the possible $2d$ directions. The list of possible increments for a step in this walk are then given by the following list: $\{\{1, 0, \cdots, 0\}, \{0, 1, \cdots, 0\}, \{0, 0, \cdots, 1\}, \cdots \{\{-1, 0, \cdots, 0\}, \{0, -1, \cdots, 0\}, \{0, 0, \cdots, -1\}\}$. To implement this in Mathematica we introduce the following program.

3.1 The program

```
WalkNdimension[dimension_, steps_]:=
 FoldList[Plus,Table[0,{dimension}],
    Union[IdentityMatrix[dimension],
         -IdentityMatrix[dimension]]
```

```
    [[Table[Random[Integer,
     {1, 2*dimension}], {steps}]]]
            ]
```

A typical run of this program for 10 steps in 2-, 3-, and 4-dimension respectively is:

```
WalkNdimension[2, 10]
WalkNdimension[3, 10]
WalkNdimension[4, 10]
```

which, respectively, produce the following lists of steps:

```
{{0, 0},  {-1, 0}, {-1, 1}, {-1, 2},
 {-1, 1}, {-1, 0}, {-1, 1}, {-1, 2},
 {-2, 2}, {-1, 2}, {-1, 1}}

{{0, 0, 0}, {0, 1, 0},  {0, 1, 1},
 {-1, 1, 1},{-1, 0, 1}, {-1, 0, 0},
 {-1, -1,0},{-1, -2, 0},{-1, -2, -1},
 {-1, -3, -1}, {0, -3, -1}}

{{0, 0, 0, 0},    {0, 0, -1, 0},
 {1, 0, -1, 0},   {0, 0, -1, 0},
 {0, 0, -1, -1},  {0, 0, -1, -2},
 {0, 1, -1, -2} , {0, 1, -2, -2},
 {-1, 1, -2, -2}, {-1, 1, -1, -2},
 {-2, 1, -1, -2}}
```

4 Numerical Analysis of Random Walks

In this section we introduce an implementation to compute some useful values related to random walks, namely: mean square end-to-end distance and radius of gyration.

4.1 Mean square end-to-end distance

The square end-to-end distance, denoted by r^2, of a d-dimensional lattice walk is given by $(x_1 - f_1)^2 + (x_2 - f_2)^2 + \cdots + (x_m - f_m)^2$ where $(x_1, x_2, \cdots x_m)$ is the starting point (the origin) of the walk and $(f_1, f_2, \cdots f_m)$ is the final point of the walk. Choosing the origin to be $(0, 0, \cdots 0)$ as the starting point of the walk simplifies the formula to $f_1^2 + f_2^2 + \cdots + f_m^2$.

4.1.1 The program

A program for computing the mean square end-to-end distance, denoted by $< r^2 >$, for `numberOfWalks` trails of `steps`-steps in a `dimension`-dimensional lattice can be written as follows.

```
MeanSquarEndToEndDistance[dimension_,
           steps_, numberOfWalks_]:=
  Module[{WalkNdimension},
          WalkNdimension[dimension1_,
          steps1_] :=
    FoldList[Plus,
           Table[0, {dimension1}],
           Union[
            IdentityMatrix[dimension1],
           -IdentityMatrix[dimension1]]
           [[Table[Random[Integer,
            {1, 2*dimension1}],
            {steps1}]]]];
          N[Sum[Apply[Plus,
           Last[WalkNdimension
               [dimension, steps]]^2],
     {numberOfWalks}]/ numberOfWalks]]
```

A typical run of this program for for 25 trails of 10-step walks in 2-, 3-, and 5-dimension respectively, is given as follows.

```
  MeanSquarEndToEndDistance[2,10,25]
8.8

  MeanSquarEndToEndDistance[3,10,25]
11.36

  MeanSquarEndToEndDistance[5,10,25]
10.32
```

For the off-lattice walk, the mean square end-to-end distance is given, in [4], as the simple formula $< r^2 >= na^2$, where n is the number of steps and a is the length of the step. In our case here we consider walks with unit length (i.e., $a = 1$) so this formula becomes trivial one.

4.2 Mean square radius of gyration

The mean square radius of gyration of a d-dimensional lattice walk, denoted by $< G^2 >$, is the sum of the squares of the distances of the step locations from the center of mass, divided by the number of step locations. The center of mass is the sum of the step locations divided by the number of step locations. This can be expressed by the following Mathematica program.

4.2.1 The program

```
MeanSquarRadiusGyration[dimension_,
        steps_, numberOfWalks_]:=
 Module[{squareRadiusGyration},
   squareRadiusGyration[steps1_]:=
    Module[{locations,centerOfmass,
      choices=Union[
         IdentityMatrix[dimension],
        -IdentityMatrix[dimension]]},
     locations=FoldList[ Plus,
               Table[0, {dimension}],
                 choices[[Table[
                 Random[Integer,
                   {1, 2*dimension}],
                        {steps}]]]];
     centerOfmass =
       N[Apply
       [Plus,locations]/ (steps1+1)];
       Apply[Plus,Flatten
             [(Transpose[locations]-
               centerOfmass)^2]]
               /(steps1+1)];
     N[Sum[squareRadiusGyration
    [ numberOfWalks],{steps}]/steps]]
                                    ]
```

A typical run of this program for for 50 trails of 15-step walks in 2-, 3-, and 10-dimension respectively, is given as follows.

```
  MeanSquarRadiusGyration[2,15,50]
1.23

  MeanSquarRadiusGyration[3,15,50]
1.642

  MeanSquarRadiusGyration[10,15,50]
1.93
```

For the off-lattice walk, the mean square radius of gyration is given, in [4], by the formula $< G^2 >= \frac{1}{6}na^2(1 - \frac{1}{n^2})$, where n is the number of steps and a is the length of the step. In our case here we consider walks with unit length (i.e., $a = 1$) so the formula can be simplified to $< G^2 >= \frac{1}{6}n(1 - \frac{1}{n^2})$ which can simply implemented in Mathematica.

5 Visualization of the random walk

5.1 Graphic visualization

The lattice and off-lattice random walks can be visualized graphically by the following program. Note that, in practice, walks with dimension greater than 3 can not be visualized. In such case the program will output the message: `Sorry, I can't show dimensions other than 2 and 3.`

5.1.1 The program

```
ShowWalk[walkProgram_,options___]:=
 Switch[Length[First[walkProgram]],
  2, Show[Graphics[Line[walkProgram],
  options, AspectRatio ->Automatic]],
3, Show[Graphics3D[Line[walkProgram],
   options, AspectRatio ->Automatic]],
  _, Print[``Sorry,
```

```
I can't show dimensions
other than 2 and 3]]
```

5.1.2 Simulation

A simulation of 3-dimension lattice walk with 250 steps is shown in figure 2 below.

```
ShowWalk[WalkNdimension[3,250]]
```

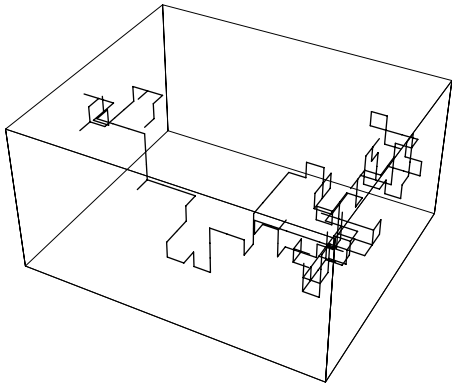

Fig.2 A simulation of 3-dimension lattice walk with 250 steps.

A simulation of 3-dimension off-lattice walk with 50 steps and 2-dimension off-lattice walk with 2500 steps are shown in figure 3 below.

```
ShowWalk[OffLatticeWalk3D[50]]
ShowWalk[OffLatticeWalk2D[2500]]
```

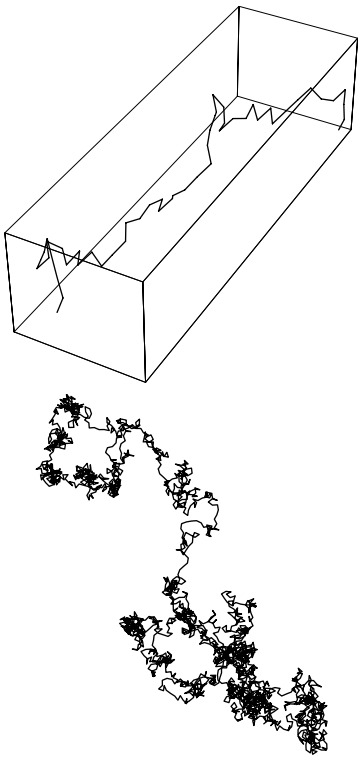

Fig.3 Simulation for a 3-dimension off-lattice walk with 50 steps (up) and 2-dimension off-lattice walk with 2500 steps (down).

6 Self Avoiding Walk

Self Avoiding Walk (SAW for short) is another model of random walks in which a walker, at any point, moves at random (for one unit) in any direction such that it never visits a previously visited site. One of the most important applications of SAW model is in polymer science where linear polymer molecules in chemical physics can be modeled as a SAW [6], one simple example is *polyethylene*. There is two algorithms explain the behavior of SAW: the *Slithering Snake* algorithm and the *Pivot* algorithm.

In the slithering snake algorithm, the walker acts like a snake. The snake slithers randomly and swaps its head and tail if he finds its body in front. It has a problem in that sometimes the snake is not able to go anywhere.

The pivot algorithm has no walker. It just acts like a bent wire. if the bended wire touches itself, the bending is canceled. In this algorithm the wire length does not change.

In this section we study the pivot algorithm.

6.1 The Pivot Algorithm

The pivot algorithm is an efficient dynamic algorithm for generating d-dimensional SAWs in a canonical ensemble. It based on randomly selecting one of the $2^d d!$ symmetry operations on a d-dimensional lattice and applying the operation to the section of the SAW beyond a randomly selected step location. In 2-dimension it is sufficient to consider just three (rotation of -90, +90, and 180 degrees on X-Y plane) of the eight ($2^2.2!$) symmetry operations. The implementation of the 2-dimensional case was given in [3]. In this section we extend this implementation to the 3-dimensional case.

6.1.1 The Program

The Mathematica program for the pivot algorithm in 3-dimensional is given as follows.

```
Pivot3DSAW[n_Integer, m_Integer] :=
 Module[{squaredist,twistAndShout},
    squaredist = n^2;
    twistAndShout =
      (Module[{ball, fixsec, movesec,
        rotchoice,newsec, newconfig,
        rot = {
        {{0,1,0},{-1,0,0},{0,0,1}},
        {{0,-1,0},{1,0,0},{0,0,1}},
        {{-1,0,0},{0,-1,0},{0,0,1}},
        {{0,0,1},{0,1,0},{-1,0,0}},
        {{0,0,-1},{0,1,0},{1,0,0}}}},
    ball = Random[Integer, {1, n-1}];
       fixsec = Take[#, ball];
       movesec =
        Take[#, ball - (n + 1)];
         rotchoice =
          rot[[Random[Integer,{1,5}]]];
        newsec =
         Map[Function[y,#[[ball]] +
            (y-#[[ball]]).rotchoice],
               movesec];
        If[Intersection
           [newsec,fixsec]\[Equal]{},
            newconfig =
              Join[fixsec, newsec];
             squaredist +=
              Apply[Plus,
                Last[newconfig]^2];
             newconfig,
             squaredist +=
              Apply[Plus, Last[#]^2];
             #]])&;
Nest[twistAndShout,
     Table[{j,0,0},{j, 0, n}], m];
N[squaredist/(m + 1)]]
```

7 Conclusion

In this paper we introduced a visual simulation of two types of random walks, that is, the on-lattice random walk in d-dimensional space and the off-lattice walk in 2- and 3-dimensions. We also give an implementation of two numerical values related to these random walks, namely, the mean square end-to-end distance and the mean square radius of gyration. An implementation of SAW was also introduced. This work is a generalization of the work given in [3] in which random walks in 1- and 2-dimensions are considered. A similar work was also given in [2, 5] for 2-dimensions random walk so our work subsumes these work.

In this work we considered only single walker, for future work, we plan to consider several walkers with a possibility of interacting with each other.

References

[1] P. G. Doyle and J. L. Snell, Random Walks and Electric Networks. The Mathematical Association of America, Inc., Third printing, 1999.

[2] R. J. Gaylord and K. Nishidate, Modeling Nature: Cellular Automata Simulations with Mathematica. Springer-Verlag New York, 1996.

[3] R. J. Gaylord and P. R. Wellin, Computer Simulations with Mathematica: Explorations in Complex Physical and Biological Systems. Springer-Verlag New York, 1995.

[4] B. D. Hughes, Random Walks and Random Environments, Vol.1: Random Walks. Oxford University Press, 1995.

[5] M. Hamada, A Visual Simulation and Analysis of Random Walks in d-Dimensional Lattice. Proceedings of the 4th St. Petersburg Conference on Simulation, pp. 245-252, St. Petersburg State University, 2001.

[6] N. Madras and G. Slade, The Self-Avoiding Walk. Birkhauser Boston Press, 1996.

[7] K. Pearson, The Problem of the Random Walker. *Nature*, **72**, 294, 1905.

[8] S. Wolfram, *The Mathematica Book*, WolframMedia Inc. Champaign, Illinois, USA, and Cambridge University Press 2000.

[9] S. Wolfram, *A new kind of Science* WolframMedia Inc. Champaign, Illinois, USA, and Cambridge, 2002.