Abstract I show how to design the value of the diffusion constant $D$ for the random walks of Squares and Triangles over their respective regular lattice in two-dimensions. By allowing movements to grid locations other than nearest neighbors, I can design the value of the diffusion constant $D$ to a value larger that unity (the default) or to a value less than unity.

Keywords: random walk, diffusion constant, geometrical form

1 Introduction

It is well known that the random walk over vertexes of unit spaced regular lattices in two-dimensions exhibits diffusion effects. The average distance traveled, $r$, after $t$ steps is well described by

$$< r^2 > = D t, \quad (1)$$

where $D$ is the diffusion constant. If each time stepped move is required to move to a nearest neighbor a unit distance away, then for all regular lattices in two-dimensions we find statistically that $D = 1$ when Equation (1) is averaged over a large number of independent trials for the same value of $t$.

In this paper I consider the random walk of polygons whose center of mass orient on vertexes of a regular lattice. For these random walks of geometrical forms in two-dimensions, the Square walks over a square lattice, the Triangle walks over a honeycomb lattice, and the Hexagon walks over a triangular lattice. (Note that the random walk of a Hexagon offers nothing different than the random walk of points on the triangular lattice - consisting of only (what I will call) Edge moves.)

For random walks of the Square, and the Triangle, I do not restrict each time stepped move to only the traditional nearest neighbors, but rather I do allow moves to some next-nearest-neighbor(s) under a biased probability. A move to a next-nearest-neighbor by a polygon can be visualized as a pivot about a vertex of the polygon; while a standard move can be viewed as a pivot or rotation about an edge of the polygon. The inclusion of possible next-nearest-neighbor moves allows for the design of diffusion constants with values greater than unity.

For the random walk scenarios considered herein, I allow for a Null move - again under a biased probability. A Null move causes the polygon to remain at the current location. The inclusion of possible Null moves allows for the design of diffusion constants with values below unity. The usefulness of the techniques contained herein could be in simulations which require a spatially-inhomogeneous diffusion constant.

2 Random Walk of the Square

I describe how to design the diffusion constant for the random walk of a Square on a two-dimensional Cartesian grid. I introduce two bias parameters in order to design the diffusion constant, $D$, such that $0 < D \leq 2$.

2.1 Methodology Overview

The random walk starts with the Square placed at the origin on a two-dimensional Cartesian grid. During each step of the random walk, the Square can in general move in one of the following ways:

4 types of Edge moves:
+*x, -x, +y, -y.

4 types of Vertex moves:
(+x+y), (-x+y), (+x-y), (-x-y).
A Null move.
The possible moves of the Square during each step is also shown in Figure 1.

After \( t \) moves the location of the Square can be described by a vector

\[
\mathbf{r} = \sum_{i=1}^{t} \mathbf{r}_i,
\]

where each \( \mathbf{r}_i \) is either an Edge move, a Vertex move, or a Null move. The average distance squared after \( t \) steps, \( < r^2 > \), for walks is two dimensions is then given as

\[
<r^2> = \left( \sum_{i=1}^{t} \mathbf{r}_i \right) \cdot \left( \sum_{j=1}^{t} \mathbf{r}_j \right) = \sum_{i=1}^{t} r_i^2 + 2 \sum_{i=1}^{t-1} \sum_{j=i+1}^{t} \mathbf{r}_i \cdot \mathbf{r}_j.
\]

For a large number of independent trials at the same value of \( t \), the second term on the right-hand side of Equation (2) typically vanishes. Thus, \( < r^2 > \) is determined by the lengths of the individual steps, \( r_i^2 \), and the relative frequency thereof. I can combine Equation (1) and Equation (2) to obtain

\[
<r^2> = \sum_{i=1}^{t} r_i^2 = D t,
\]

which is valid in the averaged limit of a large number of trials at \( t \) steps.

I now introduce the use of two bias parameters: \( p \) and \( b \), and construct a set of moves, \( t \), based on the probability for each type of move. For the total \( t \) moves, let \( p \cdot b \) represent the number of Null moves, let \( 4 \cdot p \) be the number of Edge moves, and let \( 4 \) be the number of Vertex moves. The total number of moves is then

\[
t = 4 + 4 \cdot p + p \cdot b.
\]

The distance moved by a Square in a Vertex move is \( r^2 = 2 \), and the distance moved by a Square in an Edge move is \( r^2 = 1 \). From Equation (3) I obtain

\[
D = \frac{\sum_{i=1}^{t} r_i^2}{t} = \frac{2 \cdot 4 + 1 \cdot 4 \cdot p + 0 \cdot p \cdot b}{4 + 4 \cdot p + p \cdot b} = \frac{8 + 4 \cdot p}{4 + p \cdot (4 + b)},
\]

which is an equation for the diffusion constant for the random walk of a Square in terms of the bias parameters \( p \) and \( b \).

Consider now a random walk where I set \( b = 2 \) and \( p = 1 \). If I take each move according to the algorithm in Figure 2, then 40% of the time I expect that the Square takes an Edge move, 40% of
the time I expect that the Square takes a **Vertex** move, and 20% of the time I expect that the Square takes a **Null** move. For this example, from Equation (4), \( D = 1.2 \). Figure 3 shows a sample run of a random walk code for a Square where 5000 independent walkers\(^1\) are averaged at each value of \( t \) (=Terms in the figure) using \( b = 2 \) and \( p = 1 \).

![Figure 3: Sample random walk statistics for a Square using \( b = 2 \) and \( p = 1 \).](image)

### 2.2 On the Design of a Diffusion Constant Value

The equation for the diffusion constant, \( D = D(b, p) \), in Equation (4), is here written explicitly as a function of the bias parameters. To design for a specific diffusion constant value, I convert Equation (4) to give \( p = p(b, D) \):

\[
p = \frac{8 - 4 \times D}{(4 + b) \times D - 4}.
\]

Using Equation (5), once I fix a value of \( b \), I can design the value of \( p \) that will give a specific diffusion constant \( D \). Table 3 shows this design process for various values of the bias parameter \( b \). Figure 4 shows plots of \( D(p) \) for the same set of \( b \) values shown in Table 3. The key effects of the parameter \( b \) is to set the limit for \( D \) as \( p \to \infty \), and to allow for \( D < 1 \).

\(^1\)Here each walker is independent in the sense that each walker begins by re-seeding the random number generator using the value of microseconds from the system clock.

---

**Table 3**

<table>
<thead>
<tr>
<th>Terms</th>
<th>( &lt; r \times r &gt; ) diffusion</th>
<th>Diffusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.395</td>
<td>1.197</td>
</tr>
<tr>
<td>4</td>
<td>4.931</td>
<td>1.233</td>
</tr>
<tr>
<td>6</td>
<td>7.225</td>
<td>1.204</td>
</tr>
<tr>
<td>8</td>
<td>9.811</td>
<td>1.226</td>
</tr>
<tr>
<td>10</td>
<td>11.793</td>
<td>1.179</td>
</tr>
<tr>
<td>12</td>
<td>14.198</td>
<td>1.183</td>
</tr>
<tr>
<td>14</td>
<td>16.994</td>
<td>1.210</td>
</tr>
<tr>
<td>16</td>
<td>18.963</td>
<td>1.186</td>
</tr>
<tr>
<td>18</td>
<td>21.251</td>
<td>1.181</td>
</tr>
<tr>
<td>20</td>
<td>24.085</td>
<td>1.204</td>
</tr>
</tbody>
</table>

---

Figure 4: Diffusion constant \( D \) versus \( p \) for a Square Walk with \( b = 4 \), \( b = 2 \), and \( b = 1 \).

### 3 Random Walk of the Triangle

The random walk starts with the Triangle placed at the origin of a honeycomb lattice. Figure 5 shows the type of **Edge** moves and **Vertex** moves available to the Triangle. Figure 5 shows both the **Down** node and the **Up** node of the honeycomb lattice. Discounting **Null** moves, moves from a **Down** node go to an **Up** node, and moves from an **Up** node go to a **Down** node.

For definitiveness set the origin at a **Down** node. Using a Cartesian grid with \( \Delta x = \sqrt{3}/2 \) and \( \Delta y = 1/2 \), the vertexes of a unit spaced honeycomb grid will overlap a subset of the vertexes of this underlying Cartesian grid. During each step of the random walk, the Triangle can move in one of the following ways:

**From a Down node**

3 types of **Edge** moves: \(+x+y), (-x+y), (-2y)\).

3 types of **Vertex** moves: \(+2x-2y), (-2x-2y), (+4y)\).

A **Null** move.

**From an Up node**

3 types of **Edge** moves: \(-x-y), (+x-y), (+2y)\).

3 types of **Vertex** moves: \(-2x+2y), (+2x+2y), (-4y)\).

A **Null** move.
Table 1: Square Diffusion Constant Scenarios (Equation (4) and Equation (5)).

<table>
<thead>
<tr>
<th>$b$</th>
<th>$D$</th>
<th>$p \to \infty$</th>
<th>$p$</th>
<th>$D \to 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{8 + 4 \cdot p}{4 + 5 \cdot p}$</td>
<td>$D \to 4/5$</td>
<td>$\frac{8 - 4 \cdot D}{5 \cdot D - 4}$</td>
<td>$p \to 4$</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{8 + 4 \cdot p}{4 + 6 \cdot p}$</td>
<td>$D \to 2/3$</td>
<td>$\frac{8 - 4 \cdot D}{6 \cdot D - 4}$</td>
<td>$p \to 2$</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{8 + 4 \cdot p}{4 + 8 \cdot p}$</td>
<td>$D \to 1/2$</td>
<td>$\frac{8 - 4 \cdot D}{8 \cdot D - 4}$</td>
<td>$p \to 1$</td>
</tr>
</tbody>
</table>

Figure 5: The possible random walk moves for a Triangle (sans the Null move).
The distance of each type of move is then taken as
\[ r = \sqrt{\frac{3x^2}{4} + \frac{y^2}{4}} , \]
which calculates \( r = 1 \) for an Edge move and \( r = 2 \) for an Vertex move.

For a Triangle move I use the two bias parameters: \( p \) and \( b \), and construct a set of moves, \( t \), based on the probability for each type of move. Let \( p \times b \) represent the number of Null moves, let \( 3 \times p \) be the number of Edge moves, and let \( 3 \) be the number of Vertex moves. The total number of moves is then
\[ t = 3 + 3 \times p + p \times b . \]
The distance moved by a Triangle in a Vertex move is \( r^2 = 4 \), and for an Edge move, \( r^2 = 1 \). From Equation (3) I obtain
\[ D = \frac{12 + 3 \times p}{3 + p \times (3 + b)} , \] (6)
which is an equation for the diffusion constant for the random walk of the Triangle in terms of the bias parameters \( p \) and \( b \).

The algorithm to choose each move is similar to that of the Square, but I do need to keep track of what type of node the Triangle is on. Consider then a random walk where I set \( b = 3 \) and \( p = 2 \). Here 40% of the time I expect that the Triangle takes an Edge move, 20% of the time a Vertex move, and 40% of the time a Null move. For this example, from Equation (6), \( D = 1.2 \).

A sample run, using \( b = 3 \) and \( p = 2 \), of a random walk code for a Triangle is shown in Figure 6, where 5000 independent walkers are averaged at each value of \( t \).

4 Discussion

I have shown how to design diffusion constants for the random walks of a Square and a Triangle on their respective regular lattices in two-dimensions. To allow for the design of diffusion constants other than unity I introduced various bias parameters to control the probability that a polygon would take a move other than a unity length move.

I have begun work on the random walks of geometrical forms in three-dimensions (e.g., cubes within a Cartesian lattice, and tetrahedra and octahedra within face-centered cubic lattices). For motivation for the consideration of tetrahedra and octahedra, see, for example, [2, 3], which describes a field of computation I have termed computational cosmography. When we analyze the random walk of the tetrahedron, for example, I do find the same type of constrained movement like that available to the triangle on the honeycomb grid. As it is, within a face-centered cubic grid, the tetrahedron must walk around octahedral voids, and for octahedron walks, the octahedron must walk around tetrahedral voids.

As it is stated in the introduction, the usefulness of the techniques contained herein could be in particle simulations that constrain entities to vertexes of a regular lattice, and which may require a range of diffusion constant values. Also, for simulations which require a spatially-inhomogeneous diffusion constant, we could utilize the techniques described herein and vary the bias parameters throughout the simulation domain.

References