

Introduction to Hexagonal Lattices in CompuCell3D

Maciej Swat

*Biocomplexity Institute and Department of Physics, Indiana University, 727 East 3rd Street,
Bloomington IN, 47405-7105, USA*

Hexagonal Lattices

CompuCell3D supports Cartesian (square) and regular hexagonal lattices (later referred to as hexagonal lattice). Most users are familiar with Cartesian lattice and their intuition is usually based on the “Cartesian world”. Hexagonal lattice is, in many respects, a lot like Cartesian lattice except pixels in hex lattice are hexagons (rhombic dodecahedrons in 3D) instead of squares (cubes in 3D) as it is the case for Cartesian lattice. In this section we will show in great detail how to construct hexagonal lattice, how to derive transformation formulas converting coordinates of pixels on a square lattice into corresponding pixel coordinates on a hex lattice. We will also show how to discretize diffusion equation on a hex lattice.

Constructing hexagonal lattice

Although, on first sight, hexagonal lattice has little to do with Cartesian lattice it in fact can be shown that the two lattices are isomorphic i.e. each pixel on the Cartesian lattice has unique corresponding pixel on the hex lattice (and vice versa). Let’s look at the picture below

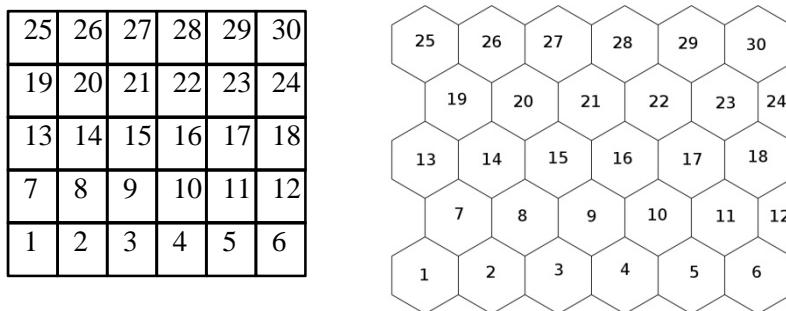


Fig 1. Two isomorphic, 2D , 6x5 lattices: Cartesian and hexagonal. Each pixel of a Cartesian lattice has a corresponding pixel on a hex lattice and vice versa. For illustration purposes indexed pixels using consecutive integers. Pixel with index 1 corresponds to x, y coordinates (0,0) whereas pixel with index 30 corresponds to x,y coordinates (6,5).

Based on Fig 1 we can see that indeed we can construct 1 to 1 mapping between pixels of hex and Cartesian lattices. However those lattices do look different and they have indeed different properties for example, square lattice pixel will have 4 nearest neighbors whereas pixel on hexagonal lattice will have 6 nearest neighbors. This automatically brings question of distances between lattice points – they are different on hexagonal and square lattices. For example, let’s pick pixel number 9 – on a square lattice pixels number 3,8,10,15 (1st order nearest neighbors) are 1 unit apart from pixel number 9. Pixels

2,4,14,16 (second nearest neighbors) are $\sqrt{1^2 + 1^2} = \sqrt{2}$ units apart and in similar way we can calculate the Euclidian distance between for 3rd,4th and higher order nearest neighbors.

On the hexagonal lattice things look quite different because pixel 9 has 6 nearest neighbors (3,4,8,10,15,16) which are 1 unit apart.

From computational stand point main question we ask then is how to map Cartesian lattice coordinates into hex lattice coordinates. The reason we want to do it is because in our code we usually want to work with rectangular arrays where one array element (inside computer memory) corresponds to one lattice pixel. Indexing rectangular lattice in any code is essentially built into modern computer architecture. Hexagonal lattices unfortunately are not. But this is not a problem as we will see in a second.

To derive transformation formula between Cartesian lattice and hex lattice we need a little bit of advanced mathematics. The main theorem we will use is one due to Pythagoras.

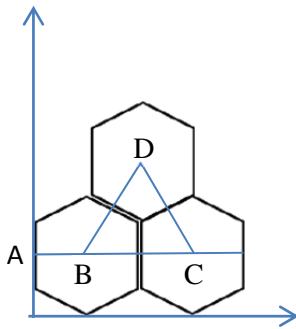


Fig 2. Deriving transformation formulas between Cartesian and hex lattices

Fig 2. shows 3 hexagonal pixels (say, 1, 2 and 7 – see Fig 1) and their centers marked as points B, C, D. Since our lattice is regular $BC=BD=CD$ and triangle BCD is equilateral. Now let us assume that distance between hex lattice pixels is L (i.e. length of segment $BC=L$). Similarly we assume that distance between pixels on a square lattice is also 1. If so then for the bottom row of pixels of both lattices we can write:

$$x_{hex} = x_{cart}L$$

where x_{cart} denotes x coordinate of pixel on Cartesian lattice and x_{hex} denotes x coordinate of pixel on hex lattice.

For the next row of pixels (7-12 in figure 2) we could write:

$$x_{hex} = \left(x_{cart} + \frac{1}{2} \right) L$$

The $\frac{1}{2}L$ factor comes from the fact that rows with odd y coordinate on hex lattice are shifted towards right by exactly $\frac{1}{2}L$ (as $AB=1/2*BC$ – see Fig. 2)

Now, all we need to do is to figure out transformations along the y axis. The distance between B and D pixels along y axis is equal to the height of the triangle BCD which is

$$\Delta y = \sqrt{|BD|^2 - \left(\frac{|BD|}{2}\right)^2} = \frac{\sqrt{3}}{2}|BD| = \frac{\sqrt{3}}{2}L$$

Therefore we can write transformation law for the y coordinates as follows:

$$y_{hex} = \frac{\sqrt{3}}{2} y_{cart} L$$

which is effectively saying that on hexagonal lattice separation between layers of pixels along y axis is $\frac{\sqrt{3}}{2}L$.

So putting all that together we get:

$$(x_{hex}, y_{hex}) = \left(x_{cart} L, \frac{\sqrt{3}}{2} y_{cart} L \right) \quad \text{for } y \text{ even}$$

$$(x_{hex}, y_{hex}) = \left(\left(x_{cart} + \frac{1}{2} \right) L, \frac{\sqrt{3}}{2} y_{cart} L \right) \quad \text{for } y \text{ odd}$$

In 3D situation looks slightly more complicated due to the fact that we have to deal with rhombic dodecahedrons which is a regular polyhedron with 12 faces which can pack space. Just like in 2D y-odd and y-even layers of pixels are interlaced (shifted by $\frac{1}{2}L$) in 3D we will have similar shifts taking place except we will deal with interlaced 2D “planes” of rhombic dodecahedrons.

Before we proceed with derivations of 3D it is useful to familiarize yourself with rhombic dodecahedron. It is not very commonly encountered polyhedron but fortunately for us its properties are far less scarier than its name suggests. The easiest way to think of a rhombic dodecahedron is to imagine a cube with 6 other cubes stuck to its faces (see Fig 3). We draw lines connecting the centers of the outer cubes with vertices of the inner cube (e.g. segments NE, KB, NG, NH). This constructs rhombic dodecahedron which has this nice property that it fills the packs the space (regular tetrahedrons do not).

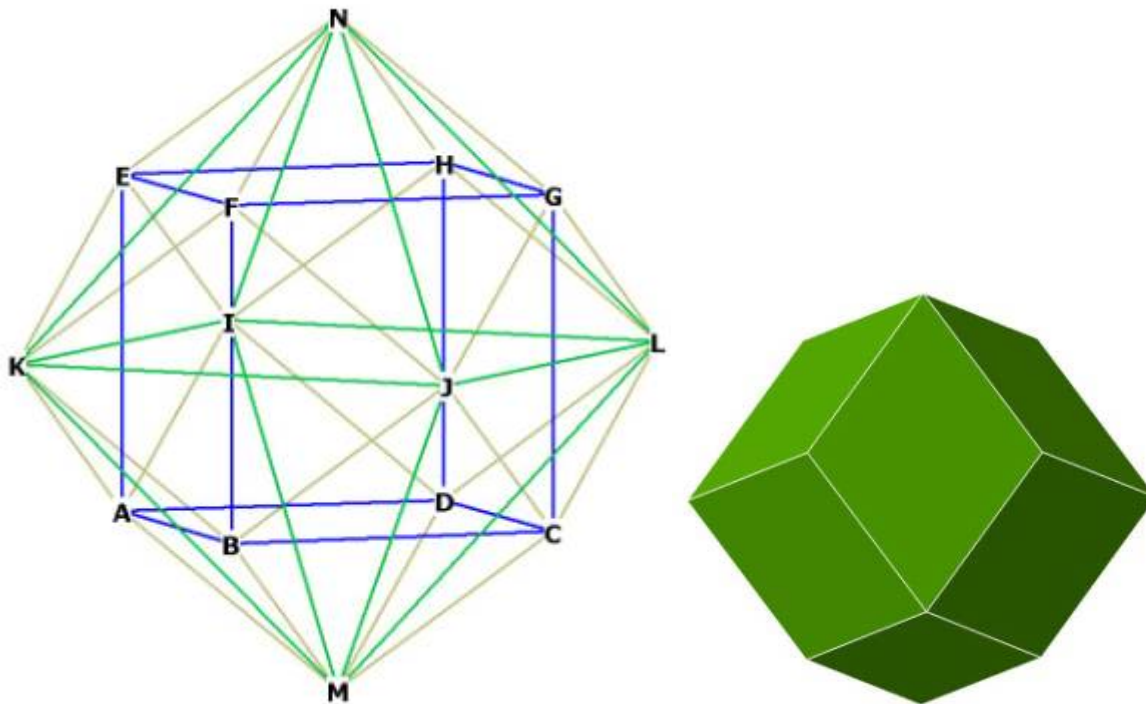
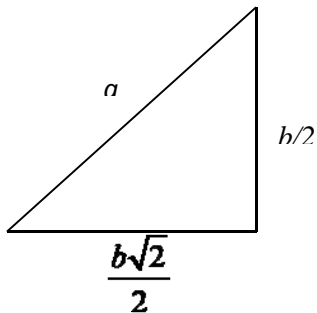


Figure 3. Construction of rhombic dodecahedron ascribed on a cube of edge length 'b' (image source <http://www.kjmaclean.com/Geometry/rhombicdodeca.html>) . Yellowish lines indicate the edges of the rhombic dodecahedron.

First lets calculate some basic properties of rhombic dodecahedron such as surface area, volume and distance between two adjacent dodecahedra (also known as voxel separation distance of the dodecahedral lattice i.e. version of hex lattice in 3D).

First let us assume that cube has edge of length b i.e $|EF| = b$ and edge of a dodecahedron $|NE| = a$.



From the Pythagorean theorem we can easily express a in terms of b :

$$a^2 = \frac{b^2}{4} + \frac{2b^2}{4} = \frac{3}{4}b^2 \Rightarrow a = \frac{\sqrt{3}}{2}b \Rightarrow b = \frac{2}{\sqrt{3}}a$$

Once we have this relation it will be easy to calculate area of the single rhombus (one hedron) and then

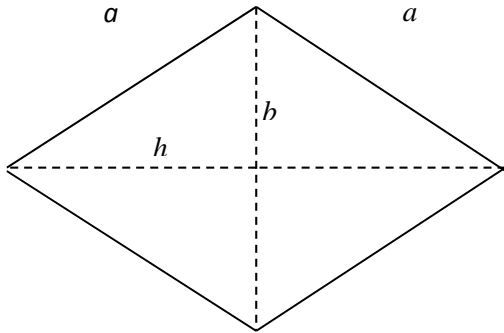


Fig 4. Single face of the rhombic dodecahedron.

surface area of the entire tetrahedron. To calculate area of the rhombus is enough to multiply its diagonals. One of the diagonals has length b . The length of the other one (call it h) we can calculate using Pythagorean theorem:

$$\frac{h^2}{4} = a^2 - \left(\frac{b}{2}\right)^2 = a^2 - \frac{1}{3}a^2 = \frac{2}{3}a^2 \Rightarrow h = 2\sqrt{\frac{2}{3}}a$$

where we used the fact that $b = \frac{2}{\sqrt{3}}a$.

Consequently we get the area of the rhombus as:

$$S_r = \frac{1}{2}b * h = \frac{1}{2} * 2 * \frac{2}{\sqrt{3}} * \sqrt{\frac{2}{3}}a^2 = \frac{2\sqrt{2}}{3}a^2$$

and total area of the dodecahedron is

$$S_d = 12S_r = 8\sqrt{2}a^2$$

Now let us calculate volume of the rhombic dodecahedron which can be expressed as volume of the inner cube on which dodecahedron is ascribed (the one with edge length b) plus the sum of six pyramid shaped caps (one per each cube) face. Let's start with volume of the pyramid:

$$V_p = \frac{1}{3} b^2 * \frac{b}{2} = \frac{1}{3} \frac{4}{3} a^2 \frac{1}{\sqrt{3}} a = \frac{4}{9\sqrt{3}} a^3$$

where we multiplied surface area of the base of the pyramid - b^2 by its height $\frac{b}{2}$ and $\frac{1}{3}$ factor.

Now the total volume of the dodecahedron can be calculated as:

$$V = 6V_p + b^3 = 6 \frac{4}{9\sqrt{3}} a^3 + \left(\frac{2}{\sqrt{3}} a \right)^3 = \frac{16\sqrt{3}}{9} a^3$$

The only thing remaining now is to calculate separation distance between centers of two adjacent dodecahedra (i.e they have one common face). Looking at Fig 3 we can notice that this distance (call it L) is equal to twice the distance between middle of the inner cube (ABCDEFGH) and the middle of one of its edges (say EF). Again using Pythagorean theorem we get:

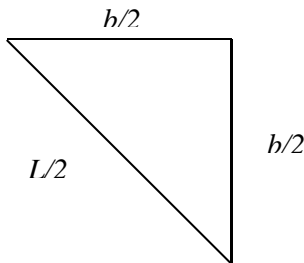
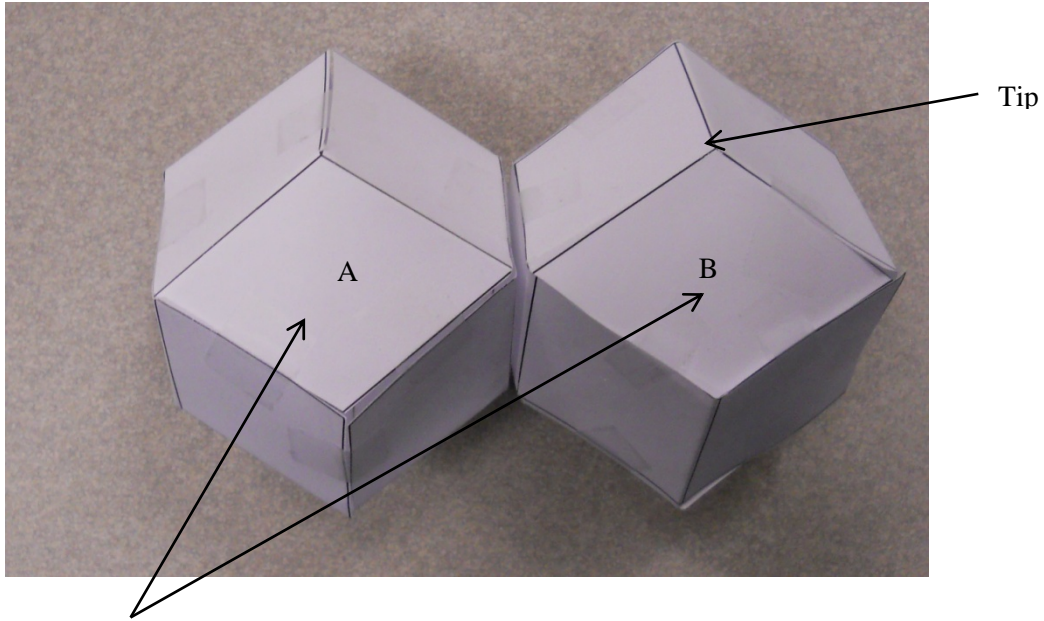


Fig 6. Calculating distance between centers of two adjacent dodecahedra

$$\left(\frac{L}{2} \right)^2 = \left(\frac{b}{2} \right)^2 + \left(\frac{b}{2} \right)^2 = \frac{b^2}{4} \Rightarrow L^2 = 2b^2 = 2 \frac{4}{3} \frac{3}{3} a^2 \Rightarrow L = \frac{2\sqrt{6}}{3} a$$

Equipped with all those handy formulas we can now attempt to derive transformation formulas between pixel coordinates on a Cartesian lattice and pixel location on a dodecahedron-based lattice. One of the tricks when constructing dodecahedral lattice in 3D is to choose convenient orientation of dodecahedrons (by that we mean positioning them with respect to a e.g xy plane). If you look at the two dodecahedra you can arrange them on a flat table as shown in Fig 6



Faces (A and B) parallel to the table

Fig 6 . Two dodecahedra on a flat table resting on two faces parallel to faces A and B.

The arrangement shown on figure 6 is not the most convenient because when we look from the top i.e. along z-axis perpendicular to faces A and B we can see that in such projection dodecahedrons look like hexagons but now equilateral ones and therefore we will not be able to form honeycomb pattern in the xy plane out of dodecahedrons oriented like that way with respect to the xy plane – see Fig 7

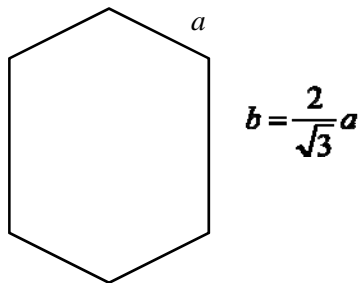


Fig 7. Orientation of dodecahedron as shown on Fig 6 when projected on to xy plane forms an irregular hexagon with the shorter side being equal to the length of a rhombus edge (a) and two longer side being equal to the length of shorter rhombus diagonal $b = \frac{2}{\sqrt{3}}a$

Looking on Fig 6 we can however find alternative arrangement of the dodecahedrons, namely we can try placing them on a tip and then if we look from the top (along the z axis) it is easy to see that projection of a dodecahedron on to xy plane is a regular hexagon as shown on Fig 8

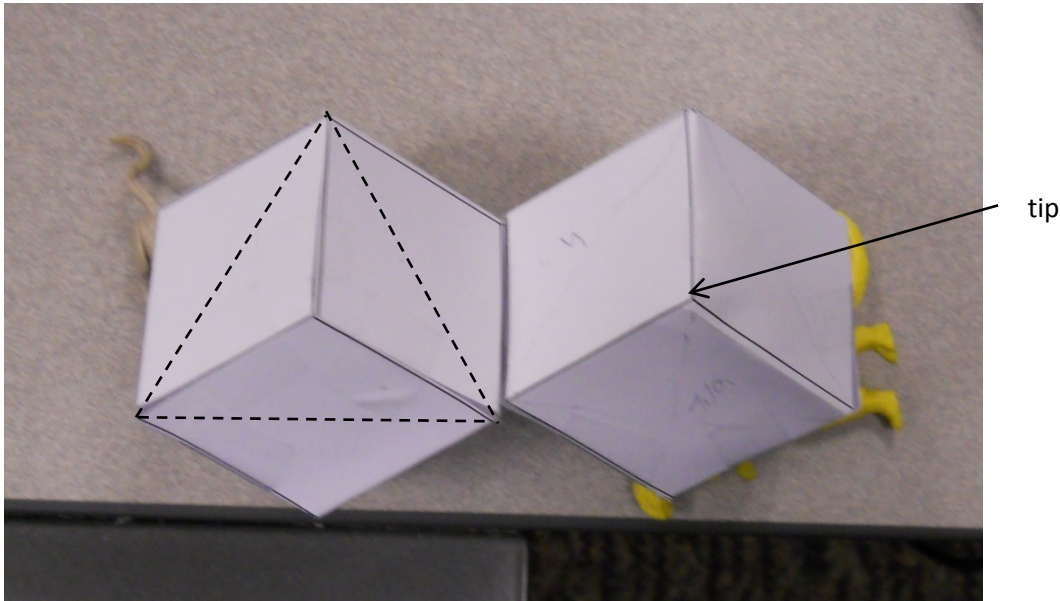


Fig 8. Arranging dodecahedrons in a way that they rest on their 'tips' on the xy-plane ensures that the projection of a dodecahedron on to the xy plane is a regular hexagons. The dashed lines form an equilateral triangle which is parallel to the xy plane and has side equal to the longer diagonal of the dodecahedral rhombus $h = 2\sqrt{\frac{2}{3}}a$ where a denotes length of the dodecahedral edge.

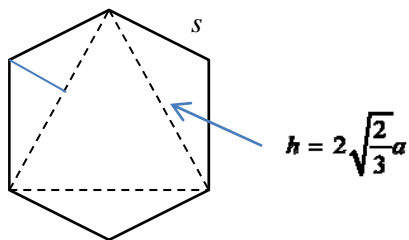


Fig 9. Projection of the dodecahedron onto the xy-plane for orientation as shown in Fig 8.

Using basic properties of a regular hexagon we can calculate

$$\frac{h/2}{s} = \cos 30^\circ = \frac{\sqrt{3}}{2} \Rightarrow s = \frac{h}{\sqrt{3}}$$

We also remember from previous derivations (see fig 6 and compare expression for L with expression for h from Fig 4) that $h = L$ where L denotes distance between centers of adjacent dodecahedrons. From now on we will express all the distances in terms of L , so

$$s = \frac{L}{\sqrt{3}}$$

$$h = L$$

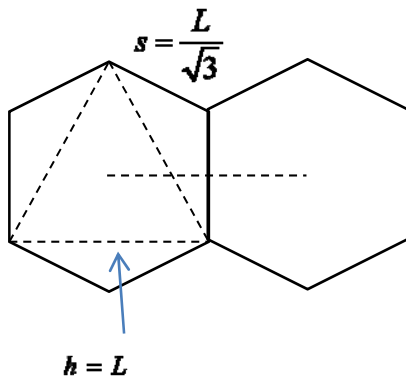


Fig 9. Projection of two dodecahedra onto the xy -plane for orientation as shown in Fig 8. From basic properties of hexagons we can figure out that distance between centers of hexagons is L – same as distance between centers of dodecahedrons

Based on the fact that distance between centers of hexagons is L – see Fig 9 we can immediately write that for the xy plane with $z=0$ we have transformation formulas same as in the 2D case:

$$(x_{hex}, y_{hex}, 0) = \left(x_{cart}L, \frac{\sqrt{3}}{2} y_{cart}L, 0 \right) \quad \text{for } y \text{ even}$$

$$(x_{hex}, y_{hex}, 0) = \left(\left(x_{cart} + \frac{1}{2} \right) L, \frac{\sqrt{3}}{2} y_{cart}L, 0 \right) \quad \text{for } y \text{ odd}$$

where now 'hex' subscript refers to rhombic dodecahedron lattice and x_{hex} denotes x position of a pixel on a rhombic dodecahedron lattice. Similar convention holds for other coordinates.

When we now stack another layer of dodecahedrons on top of bottom $z=0$ layer of dodecahedrons we will get the following:

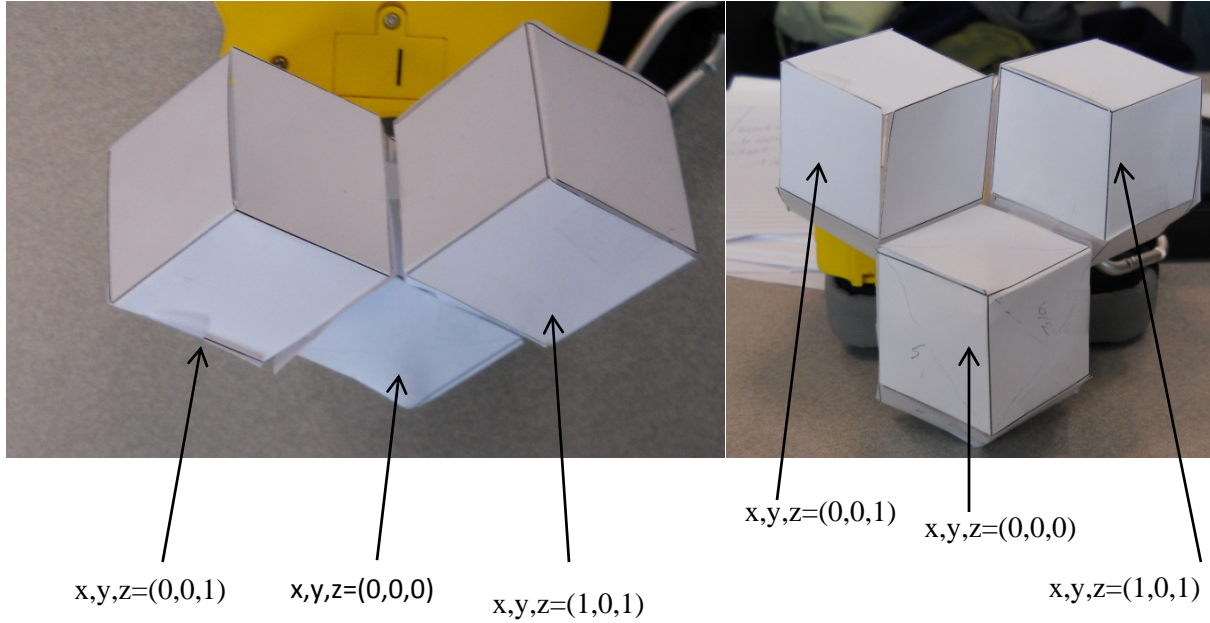


Fig 10. Vertical stacking of dodecahedrons (looking down the z axis). x,y,z denote equivalent Cartesian lattice indices. Left photo - view from the top – along z axis. Right photo view from the side – along y axis

Notice moving up in z direction requires shifts in 'y' and 'z' directions. When we project arrangement shown in Fig. 10 onto xy plane we would get the following picture:

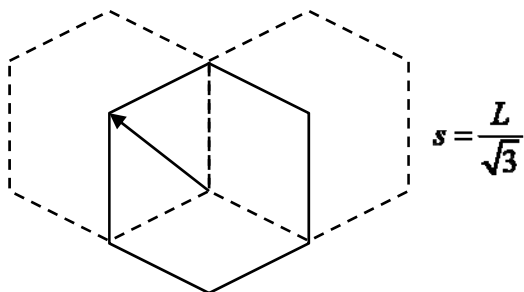


Figure 11. When moving up along z-axis we need to shift center of dodecahedrons in y and z directions.. The projection of the shift vector onto xy plane is shown as an arrow. Dashed line hexagons are projections of dodecahedrons with coordinate (0,0,1) and (1,0,1).

Calculating the length of a shift is straightforward, it is simply equal to the length of the side of the

hexagon $s = \frac{L}{\sqrt{3}} = \frac{\sqrt{3}}{3} L$

Decomposing this vector along x and y axis we get:

$$\vec{v} = \left(-\frac{L}{2}, \frac{\sqrt{3}L}{6} \right)$$

Now we have to calculate shift along the z axis for the pixel (x,y,z)=(0,0,1). Before we do this let us calculate the distance between opposite tips of the dodecahedron. If look at Fig 8 this is distance between point marker as tip and point of contact on dodecahedron with the table (not shown)

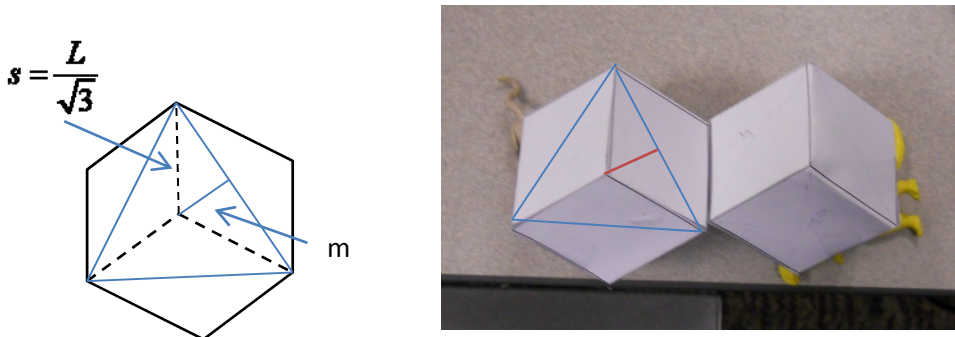


Fig 12. Calculating the height of the pyramid. Base of the pyramid is marked as a blue triangle with its 2D projection. The segment m lays in a plane of the big blue triangle.

From Fig 12 we get:

$$\frac{m}{s} = \cos 60^\circ \Rightarrow m = \frac{s}{2} = \frac{\sqrt{3}}{6} L$$

The height of the pyramid will be found from the triangle made up of m , x (height of the pyramid) and half of the shorter diagonal of a dodecahedron rhombus (marked as red line on Fig 12).

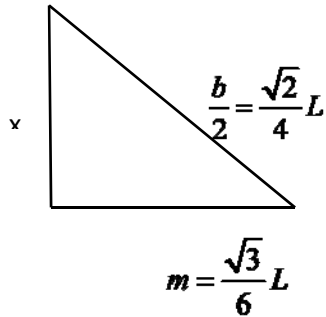


Fig 13. Triangle used to calculate height of the pyramid shown on Fig 12.

Based on Fig 13 we can write:

$$x^2 = \frac{L^2}{8} - \frac{3}{36}L^2 = \frac{3}{72}L^2$$

$$x = \frac{\sqrt{6}}{12}L$$

With this we can calculate distance from tip to tip by noticing that it is equal to

$$D = 2x + a + x = 3\frac{\sqrt{6}}{12}L + \frac{\sqrt{6}}{4}L = \frac{\sqrt{6}}{2}L$$

where a denotes the length of the dodecahedron and we used the fact that $a = \frac{\sqrt{3}}{\sqrt{22}}L = \frac{\sqrt{6}}{4}L$. Figure

14 explains why this is so:

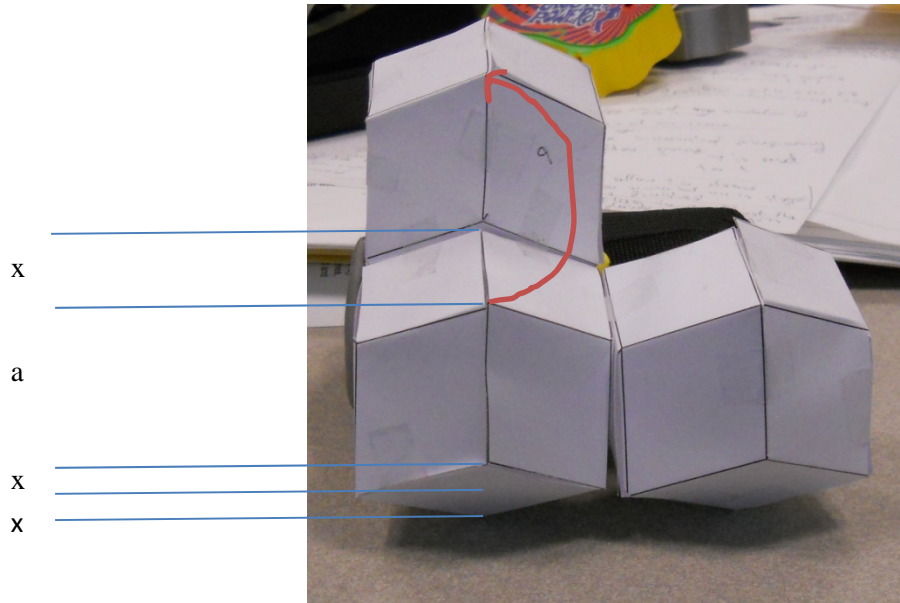


Fig 14. The distance between opposite tips of the dodecahedron is $3x + a$. The top segment appears larger on a photograph than it is in reality. The reader may convince him/herself by making a 3D model of dodecahedron to see that this is indeed the case. The red arrow indicates displacement along the z axis that is necessary to stack dodecahedrons on top of each other. As we can see this displacement is

$$x + a = \frac{\sqrt{6}}{12}L + \frac{\sqrt{6}}{4} = \frac{\sqrt{6}}{3}L$$

Consequently full shift vector i.e. the difference between $(0,0,1)$ and $(0,0,0)$ is :

$$\vec{v} = \left(-\frac{1}{2}L, \frac{\sqrt{3}}{6}L, \frac{\sqrt{6}}{3}L \right)$$

When we get to $z=2$, however, we observe (and this is easy to be overlooked) that projection of dodecahedron at point $(0,0,2)$ does not overlap with projection of dodecahedron at $(0,0,0)$:

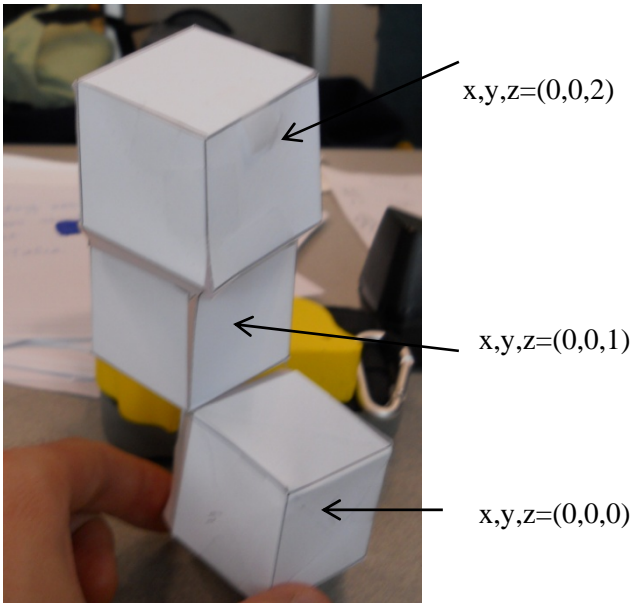


Fig 15. Stacking 3 dodecahedrons on top of each other ($x=0$ and $y=0$ for the three dodecahedrons). Observe that projection of top dodecahedron does not overlap with projection of the bottom dodecahedron.

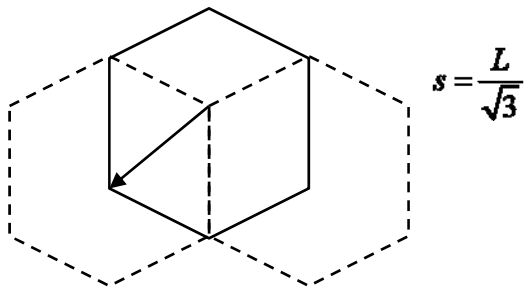


Figure 16. When moving up along z -axis we need to shift center of dodecahedrons in y and z directions.. The projection of the shift vector onto xy plane is shown as an arrow. Dashed line hexagons are projections of dodecahedrons with coordinate $(0,0,2)$ and $(1,0,2)$. Solid line is a projection of dodecahedron at position $(0,0,0)$.

Following similar reasoning as for $z=1$ we can write shift vector between $(0,0,0)$ and $(0,0,2)$ as:

$$\vec{v} = \left(-\frac{1}{2}L, -\frac{\sqrt{3}}{6}L, \frac{2\sqrt{6}}{3}L \right)$$

Now that we know the displacements associated with stacking dodecahedrons on top of each other (we have stacks of three alternating planes in the z direction) we can write full transformation formulas for the dodecahedral lattice:

$$[x_{hex}, y_{hex}, z_{hex}] = \left[\left(x_{cart} + \frac{1}{2} \right) L, \frac{\sqrt{3}}{2} y_{cart} L, \frac{\sqrt{6}}{3} z_{cart} L \right] \quad \text{for } y\%2 = 0 \text{ and } z\%3 = 0$$

$$[x_{hex}, y_{hex}, z_{hex}] = \left[x_{cart} L, \frac{\sqrt{3}}{2} y_{cart} L, \frac{\sqrt{6}}{3} z_{cart} L \right] \quad \text{for } y\%2 = 1 \text{ and } z\%3 = 0$$

$$[x_{hex}, y_{hex}, z_{hex}] = \left[x_{cart} L, \left(\frac{\sqrt{3}}{2} y_{cart} + \frac{\sqrt{3}}{6} \right) L, \frac{\sqrt{6}}{3} z_{cart} L \right] \quad \text{for } y\%2 = 0 \text{ and } z\%3 = 1$$

$$[x_{hex}, y_{hex}, z_{hex}] = \left[\left(x_{cart} + \frac{1}{2} \right) L, \left(\frac{\sqrt{3}}{2} y_{cart} + \frac{\sqrt{3}}{6} \right) L, \frac{\sqrt{6}}{3} z_{cart} L \right] \quad \text{for } y\%2 = 1 \text{ and } z\%3 = 1$$

$$[x_{hex}, y_{hex}, z_{hex}] = \left[x_{cart} L, \left(\frac{\sqrt{3}}{2} y_{cart} - \frac{\sqrt{3}}{6} \right) L, \frac{\sqrt{6}}{3} z_{cart} L \right] \quad \text{for } y\%2 = 0 \text{ and } z\%3 = 2$$

$$[x_{hex}, y_{hex}, z_{hex}] = \left[\left(x_{cart} + \frac{1}{2} \right) L, \left(\frac{\sqrt{3}}{2} y_{cart} - \frac{\sqrt{3}}{6} \right) L, \frac{\sqrt{6}}{3} z_{cart} L \right] \quad \text{for } y\%2 = 1 \text{ and } z\%3 = 2$$

The first 2 formulas are 2D transformation formulas we had before but we added scaling along z axis. We, however, switched y odd and even formulas to be consistent with the way it is implemented in CompuCell3D. 4 remaining formulas are obtained by applying shift vectors between pixels $(0,0,0)$, $(0,0,1)$ and $(0,0,2)$. Notice that as we go up along z -axis the xy projection of the centroid of the dodecahedron moves along 'triangular spiral':

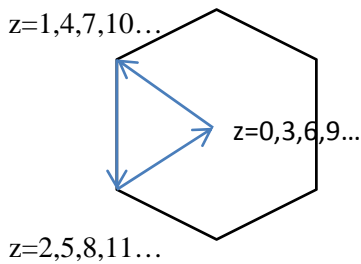


Figure 17. xy projections of centroid for different value of z.

We are almost done here except we have to be aware that in the transformation formulas for 2D and 3D lattices we have factor L. Surfaces and volumes of all the shapes discussed above can be expressed in terms of L. Therefore if we set L to be one then neither surface, nor volume of the dodecahedron or surface /perimeter of hexagon will be an integer multiple of 1. However in CompuCell3D cells are being destroyed once their volume reaches 0. Therefore it will be convenient to constrain the surface of the 2D of the pixel/ volume of the 3D voxel to be 1 irrespective of the lattice. Consequently on hex/dodecahedral lattices $L \neq 1$

In 2D, area of the hexagon is:

$$V_{hex} = \frac{\sqrt{3}}{2} L^2 = \frac{3\sqrt{3}}{2} a^2$$

and

$$a = \frac{L}{\sqrt{3}}$$

where a denotes length of the side of the hexagon. We denote it by V_{hex} to be in agreement with CompuCell3D convention where in 2D area of the pixel has the meaning of pixel volume and perimeter of a pixel has a meaning of surface pixel. You can understand this better if you assume that 2D pixels are in fact 3D shapes with height 1. Constraining V_{hex} to be 1 gives:

$$V_{hex} = \frac{\sqrt{3}}{2} L^2 = 1 \Rightarrow L = L_{hex-unit} = \sqrt{\frac{2}{\sqrt{3}}} \approx 1.075$$

and length of the side of the hexagon becomes

$$S_{hex-unit} = \sqrt{\frac{2}{3\sqrt{3}}} \approx 0.6204$$

In 3D we have:

$$V_{hex} = \frac{16\sqrt{3}}{9} a^3 \Rightarrow a = \sqrt[3]{\frac{9V_{hex}}{16\sqrt{3}}}$$

so if we set $V_{hex} = 1$ we get $a = \sqrt[3]{\frac{9}{16\sqrt{3}}}$ and using our previous result that $L = \frac{2\sqrt{6}}{3} a$ we obtain

$$L_{hex} = \frac{2\sqrt{6}}{3} \sqrt[3]{\frac{9V_{hex}}{16\sqrt{3}}}$$

Where when we set $V_{hex} = 1$ we will get

$$L_{hex-unit} = \frac{2\sqrt{6}}{3} \sqrt[3]{\frac{9}{16\sqrt{3}}} \approx 1.122$$

The area of a single side of the dodecahedron is given by (see earlier derivations)

$$S_{hex} = \frac{2\sqrt{2}}{3} a^2 = \frac{2\sqrt{2}}{3} \left(\sqrt[3]{\frac{9V_{hex}}{16\sqrt{3}}} \right)^2 = \frac{2\sqrt{2}}{3} \sqrt[3]{\left(\frac{9V_{hex}}{16\sqrt{3}} \right)^2}$$

For $V_{hex} = 1$ we get

$$S_{hex-unit} = \frac{2\sqrt{2}}{3} \sqrt[3]{\left(\frac{9}{16\sqrt{3}} \right)^2} \approx 0.445$$

On the Cartesian lattice unit surface, and unit length have all numeric values of 1. On hex lattice they do not as expressions for $L_{hex-unit}$ and $S_{hex-unit}$ show. For example when calculating numerical value of a cell surface in CompuCell3D we keep adding and subtracting multiples of $S_{hex-unit}$. The distances are on such lattice are calculated based on transformation formulas derived earlier.

Discretizing the Diffusion Equation

In CC3D we use explicit finite difference solvers also known as Forward Euler (FE) solvers. We will skip derivation of discretization of the diffusion equation with isotropic diffusion constant on a Cartesian lattice as it is presented in many books. Instead let us derive discretization for the case when diffusion constant is a position of x,y, and z

$$\begin{aligned} \frac{\partial c}{\partial t} &= \text{div}(D\vec{\nabla}c) = \frac{\partial}{\partial x} \left(D \frac{\partial c}{\partial x} \right) + \frac{\partial}{\partial y} \left(D \frac{\partial c}{\partial y} \right) + \frac{\partial}{\partial z} \left(D \frac{\partial c}{\partial z} \right) = \\ &= D \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2} \right) + \frac{\partial D}{\partial x} \frac{\partial c}{\partial x} + \frac{\partial D}{\partial y} \frac{\partial c}{\partial y} + \frac{\partial D}{\partial z} \frac{\partial c}{\partial z} \end{aligned}$$

Discretizing it on Cartesian lattice is straightforward now and we use forward differences to express all derivatives:

$$\frac{c(x, y, z, t+T) - c(x, y, z, t)}{T} = D(x, y, z) \left(\frac{c(x+L, y, z, t) - 2c(x, y, z, t) + c(x-L, y, z, t)}{L^2} + y, z \text{ terms} \right) + \left(\frac{(D(x+L, y, z, t) - D(x, y, z, t))(c(x+L, y, z, t) - c(x, y, z, t))}{L^2} + y, z \text{ terms} \right)$$

Which gives discretization of the diffusion equation with space dependent diffusion coefficient.

$$c(x, y, z, t+T) = TD(x, y, z) \left(\frac{c(x+L, y, z, t) - 2c(x, y, z, t) + c(x-L, y, z, t)}{L^2} + y, z \text{ terms} \right) + T \left(\frac{(D(x+L, y, z, t) - D(x, y, z, t))(c(x+L, y, z, t) - c(x, y, z, t))}{L^2} + y, z \text{ terms} \right) + c(x, y, z, t)$$

On a square lattice the method of calculating LHS is quite simple – for a given pixel we visit neighboring pixels (nearest neighbors) and compute terms making up the LHS. If this is true then, perhaps on a hex lattice we can simply manipulate above formulas and take into account that pixel on a hex lattice has more neighbors. Alas, this would not really work. If we did naïve extension of the above formula to be applied on a hex lattice we would miss important scaling factors. The correct way is to start from first principles and discretize divergence term by using its integral-based definition which states that divergence is equal to flux of vector \vec{v} through surface (perimeter in 2D) of the small element of volume δV divided by volume of the element (surface in 2D):

$$\text{div}(\vec{v}) = \lim_{\delta V \rightarrow 0} \frac{\oiint \vec{v} \cdot \vec{n} dS}{\delta V}$$

To begin we will now show how to discretize on a 2D Cartesian lattice $\text{div}(D\vec{v}c)$ term using above definition of the divergence. The derivations for hexagonal lattice in 2D and 3D will look very similar.

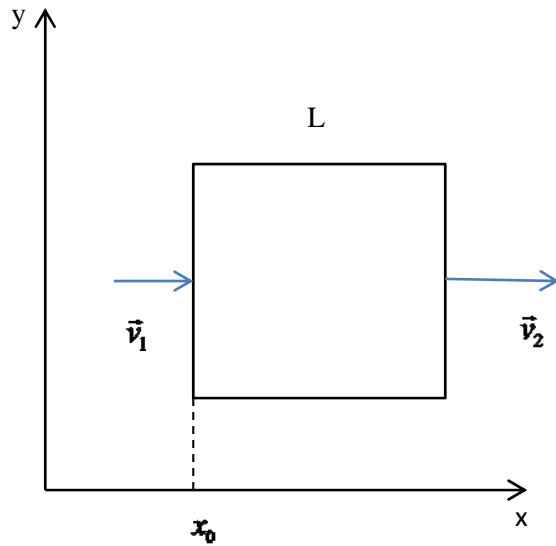


Fig 18. Flux balance along the x axis for an Infinitesimal square pixel of side length L.

When we do flux balance along the x axis the expression expressing in- and outflow of flux along the x axis is given below. Notice that we assume that we have to multiple vectors by the area/length (in our case it is length as we are in 2D) of the wall through which flux enters the element. In our case this is L.

$$\oiint_{\delta S} \vec{v} \cdot \vec{n} dS(x\text{-axis}) = L(\vec{v}_2 - \vec{v}_1)$$

\vec{v} 's are gradients of a scalar quantity (concentration c in our case) along the direction perpendicular to the wall of the element and they are given by:

$$\vec{v}_1 = D(x_0, y) \frac{c(x_0, y) - c(x_0 - L, y)}{L} \cdot \vec{n}_x$$

$$\vec{v}_2 = D(x_0 + L, y) \frac{c(x_0 + L, y) - c(x_0, y)}{L} \cdot \vec{n}_x$$

where \vec{n}_x denotes unit vector along the x-axis (perpendicular to the element's wall)

$$\begin{aligned}
\oiint_{\delta S} \vec{v} \cdot \vec{n} dS (x\text{-axis}) &= L(\vec{v}_2 - \vec{v}_1) = L \left[D(x+L, y) \frac{c(x+L, y) - c(x, y)}{L} - D(x, y) \frac{c(x, y) - c(x-L, y)}{L} \right] \\
&= L \left[(D(x+L, y) - D(x, y) + D(x, y)) \frac{c(x+L, y) - c(x, y)}{L} - D(x, y) \frac{c(x, y) - c(x-L, y)}{L} \right] \\
&= L \left[D(x, y) \frac{c(x+L, y) + c(x-L, y) - 2c(x, y)}{L} + \frac{(D(x+L, y) - D(x, y))(c(x+L, y) - c(x, y))}{L} \right]
\end{aligned}$$

where we dropped subscript from in the x_0 and, on purpose, we did leave out factors of L- we will use them later for derivation of expression for hex lattice

Repeating the same process for y-axis gives us:

$$\begin{aligned}
\oiint_{\delta S} \vec{v} \cdot \vec{n} dS (y\text{-axis}) &= L(\vec{v}_2 - \vec{v}_1) = L \left[D(x, y+L) \frac{c(x, y+L) - c(x, y)}{L} - D(x, y) \frac{c(x, y) - c(x, y-L)}{L} \right] \\
&= L \left[(D(x, y+L) - D(x, y) + D(x, y)) \frac{c(x, y+L) - c(x, y)}{L} - D(x, y) \frac{c(x, y) - c(x, y-L)}{L} \right] \\
&= L \left[D(x, y) \frac{c(x, y+L) + c(x, y-L) - 2c(x, y)}{L} + \frac{(D(x, y+L) - D(x, y))(c(x, y+L) - c(x, y))}{L} \right]
\end{aligned}$$

Now adding all those things together we get:

$$\begin{aligned}
div(D\vec{\nabla}c) &= \frac{1}{L^2} L \left[D(x, y) \frac{c(x+L, y) + c(x-L, y) + c(x, y+L) + c(x, y-L) - 4c(x, y)}{L} \right] + \\
&+ \frac{1}{L^2} L \left[\frac{(D(x+L, y) - D(x, y))(c(x+L, y) - c(x, y)) + (D(x, y+L) - D(x, y))(c(x, y+L) - c(x, y))}{L} \right]
\end{aligned}$$

which has the same form as equation derived before by discretizing partial derivatives. So for completeness we can write fully discretized for of the diffusion equation $\frac{\partial c}{\partial t} = div(D\vec{\nabla}c)$ in 2D on Cartesian lattice:

$$c(x, y, t+T) = c(x, y, t) + \frac{T}{L^2} L \left[D(x, y) \frac{c(x+L, y, t) + c(x-L, y, t) + c(x, y+L, t) + c(x, y-L, t) - 4c(x, y, t)}{L} \right] +$$

$$+ \frac{T}{L^2} L \left[\frac{(D(x+L, y, t) - D(x, y))(c(x+L, y, t) - c(x, y, t)) + (D(x, y+L) - D(x, y))(c(x, y+L, t) - c(x, y, t))}{L} \right]$$

which is exactly as the one we got before.

Now, for the hex lattice in 2D all we have to do is to repeat the same flux balance analysis we did for the square pixel.

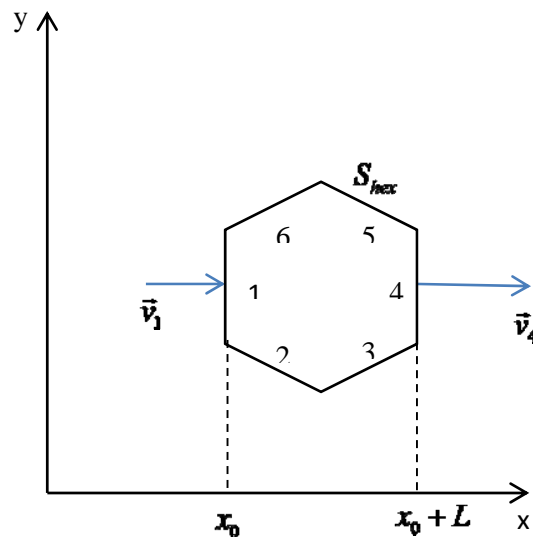


Fig. 19 Flux balance analysis for one pair of sides of the regular hexagon

Repeating for hexagonal pixel the same flux balance analysis we did for the square pixel we can write

$$\oiint_{\delta S} \vec{v} \cdot \vec{n} dS(x-axis) = S_{hex} (\vec{v}_4 - \vec{v}_1)$$

which implies

$$\begin{aligned}
\oiint_{\delta S} \vec{v} \cdot \vec{n} dS(x-axis) &= S_{hex} (\vec{v}_4 - \vec{v}_1) = S_{hex} \left[D(x+L, y) \frac{c(x+L, y) - c(x, y)}{L} - D(x, y) \frac{c(x, y) - c(x-L, y)}{L} \right] \\
&= S_{hex} \left[(D(x+L, y) - D(x, y) + D(x, y)) \frac{c(x+L, y) - c(x, y)}{L} - D(x, y) \frac{c(x, y) - c(x-L, y)}{L} \right] \\
&= S_{hex} \left[D(x, y) \frac{c(x+L, y) + c(x-L, y) - 2c(x, y)}{L} + \frac{(D(x+L, y) - D(x, y))(c(x+L, y) - c(x, y))}{L} \right]
\end{aligned}$$

Before we write expressions for subsequent pairs of parallel sides of the hexagon let us introduce the following shorthand notation:

$$\begin{aligned}
c(x, y, t) &= c(0, t) \\
D(x, y, t) &= D(0, t) \\
c(x-L, y, t) &= c(1, t) \\
D(x-L, y, t) &= D(1, t) \\
c(x+L, y, t) &= c(4, t) \\
D(x+L, y, t) &= D(4, t)
\end{aligned}$$

which means that we denote concentration/diffusion constant of a pixel adjacent to hexagon by prepending replacing coordinates of this pixel by the label of the side of the hexagon and 0 denotes x, y coordinates of the considered pixel.

With such convention our discretized flux balance along the x-axis for the hexagon becomes:

$$\begin{aligned}
\oiint_{\delta S} \vec{v} \cdot \vec{n} dS(\text{walls } 4-1) &= S_{hex} (\vec{v}_4 - \vec{v}_1) = \\
&= S_{hex} \left[D(0) \frac{c(4) + c(1) - 2c(0)}{L} + \frac{(D(4) - D(0))(c(4) - c(0))}{L} \right]
\end{aligned}$$

Using our indexing scheme it is easy to write expressions for the remaining pair of walls (2-5, and 3-6):

$$\begin{aligned}
\oiint_{\delta S} \vec{v} \cdot \vec{n} dS(\text{walls } 5-2) &= S_{hex} \left[D(0) \frac{c(5) + c(2) - 2c(0)}{L} + \frac{(D(5) - D(0))(c(5) - c(0))}{L} \right] \\
\oiint_{\delta S} \vec{v} \cdot \vec{n} dS(\text{walls } 6-3) &= S_{hex} \left[D(0) \frac{c(6) + c(3) - 2c(0)}{L} + \frac{(D(6) - D(0))(c(6) - c(0))}{L} \right]
\end{aligned}$$

and putting everything together we get fully discretized diffusion equation on a hex lattice

$$c(x, y, t+T) = c(x, y, t) + \frac{T}{V_{hex}} S_{hex} \left[D(0) \frac{\sum_{n=1}^6 c(n) - 6c(0, t)}{L} \right] + \frac{T}{V_{hex}} S_{hex} \left[\frac{(D(4) - D(0))(c(4, t) - c(0, t)) + (D(5) - D(0))(c(5, t) - c(0, t)) + (D(6) - D(0))(c(6, t) - c(0, t))}{L} \right]$$

where we using expressions for surface of the hexagon (V_{hex}), length of the hexagon side (S_{hex}) expressed in terms of L (pixel separation we get)

$$\frac{S_{hex}}{V_{hex} L} = \frac{\frac{L}{\sqrt{3}}}{\frac{\sqrt{3}}{2} L^3} = \frac{2}{3} \frac{1}{L^2}$$

so discretized diffusion equation becomes:

$$c(x, y, t+T) = c(x, y, t) + \frac{2}{3} \frac{T}{L^2} \left[D(0) \left(\sum_{n=1}^6 c(n) - 6c(0, t) \right) \right] + \frac{2}{3} \frac{T}{L^2} \left[(D(4) - D(0))(c(4, t) - c(0, t)) + (D(5) - D(0))(c(5, t) - c(0, t)) + (D(6) - D(0))(c(6, t) - c(0, t)) \right]$$

The offsets for sides 4,5,6 are given below:

$y \% 2 = 0$ (y is even)

offset[4]=(0,1,0);

offset[5]=(1,1,0);

offset[6]=(1,0,0);

$y\%2=1$ (y is odd)

$\text{offset}[4]=(-1,1,0);$

$\text{offset}[5]=(0,1,0);$

$\text{offset}[6]=(1,0,0);$

For 3D dodecahedral lattice the derivation will proceed exactly as for the 2D case of hex lattice except that we will have 6 pairs of parallel walls:

$$c(x, y, t + T) = c(x, y, t) + \frac{T}{V_{hex}} S_{hex} \left[\frac{D(0) \sum_{n=1}^{12} c(n) - 12c(0, t)}{L} \right] + \frac{T}{V_{hex}} S_{hex} \left[\frac{\sum_{n=7}^{12} (D(n) - D(0))(c(n, t) - c(0, t))}{L} \right]$$

Evaluating $\frac{S_{hex}}{V_{hex}L}$ for dodecahedron gives:

$$\frac{S_{hex}}{V_{hex}L} = \frac{\frac{8\sqrt{2}a^2}{12}}{\frac{16\sqrt{3}}{9}a^3 \frac{2\sqrt{6}}{3}a} = \frac{3}{16a^2} = \frac{3}{16 \frac{9}{24}L^2} = \frac{1}{2L^2}$$

Where we used the following relations

$$V_{hex} = \frac{16\sqrt{3}}{9}a^3$$

$$L = \frac{2\sqrt{6}}{3}a$$

$$S_{hex} = S_r = \frac{8\sqrt{2}a^2}{12}$$

So finally we get fully discretized for of the diffusion equation on a dodecahedral lattice in 3D

$$c(x, y, t+T) = c(x, y, t) + \frac{1}{2} \frac{T}{L^2} \left[D(0) \frac{\sum_{n=1}^{12} c(n) - 12c(0, t)}{L} \right] + \frac{1}{2} \frac{T}{L^2} \left[\frac{\sum_{n=7}^{12} (D(n) - D(0))(c(n, t) - c(0, t))}{L} \right]$$

When implementing equations care must be taken to make sure that neighbors 7-12 are on the opposite side of the dodecahedron as neighbors 1-6. In other words we want to make sure that all of the

neighbors listed in the term $\left[\frac{\sum_{n=7}^{12} (D(n) - D(0))(c(n, t) - c(0, t))}{L} \right]$ correspond to the walls through

which outflow of the flux occurs. Similar requirement holds for 2D hex lattice.

In 3D terms involving first derivatives are calculated through hedra with z offset 1 (i.e. 'upper cap' of the dodecahedron or hedra adjacent to 'tip' point from Fig 8 – they all have z offset coordinate equal to 1) and through hedra which have z offset coordinate equal to 0 and x,y offset coordinates correspond to those from 2D hexagonal lattice (3,5,6 from Fig. 19)

The offsets for hedra 7-12 are given below:

//y%2=0 and z%3=0

offset[0][0]=(0,1,0)

offset[0][1]=(1,1,0)

offset[0][2]=(1,0,0)

offset[0][3]=(0,-1,1)

offset[0][4]=(0,0,1)

offset[0][5]=(1,0,1)

//y%2=1 and z%3=0

offset[1][0]=(-1,1,0)

offset[1][1]=(0,1,0)

offset[1][2]=(1,0,0)
offset[1][3]=(-1,0,1)
offset[1][4]=(0,0,1)
offset[1][5]=(0,-1,1)

//y%2=0 and z%3=1

offset[2][0]=(-1,1,0)
offset[2][1]=(0,1,0)
offset[2][2]=(1,0,0)
offset[2][3]=(0,1,1)
offset[2][4]=(0,0,1)
offset[2][5]=(-1,1,1)

//y%2=1 and z%3=1

offset[3][0]=(0,1,0)
offset[3][1]=(1,1,0)
offset[3][2]=(1,0,0)
offset[3][3]=(0,1,1)
offset[3][4]=(0,0,1)
offset[3][5]=(1,1,1)

//y%2=0 and z%3=2

offset[4][0]=(-1,1,0)
offset[4][1]=(0,1,0)
offset[4][2]=(1,0,0)

offset[4][3]=(-1,0,1)

offset[4][4]=(0,0,1)

offset[4][5]=(0,-1,1)

//y%2=1 and z%3=2

offset[5][0]=(0,1,0)

offset[5][1]=(1,1,0)

offset[5][2]=(1,0,0)

offset[5][3]=(0,-1,1)

offset[5][4]=(0,0,1)

offset[5][5]=(1,0,1)

For completeness we also list nearest neighbors offsets for the dodecahedral lattice:

//y%2=0 and z%3=0

offset[0][0]=(1,0,0)

offset[0][1]=(-1,0,0)

offset[0][2]=(0,1,0)

offset[0][3]=(0,-1,0)

offset[0][4]=(0,0,1)

offset[0][5]=(0,0,-1)

offset[0][6]=(1,1,0)

offset[0][7]=(1,-1,0)

offset[0][8]=(1,0,1)

offset[0][9]=(1,0,-1)

offset[0][10]=(0,1,-1)

offset[0][11]=(0,-1,1)

//y%2=1 and z%3=0

offset[1][0]=(1,0,0)

offset[1][1]=(-1,0,0)

offset[1][2]=(0,1,0)

offset[1][3]=(0,-1,0)

offset[1][4]=(0,0,1)

offset[1][5]=(0,0,-1)

offset[1][6]=(-1,1,0)

offset[1][7]=(-1,-1,0)

offset[1][8]=(-1,0,1)

offset[1][9]=(-1,0,-1)

offset[1][10]=(0,1,-1)

offset[1][11]=(0,-1,1)

//y%2=0 and z%3=1

offset[2][0]=(1,0,0)

offset[2][1]=(-1,0,0)

offset[2][2]=(0,1,0)

offset[2][3]=(0,-1,0)

offset[2][4]=(0,0,1)

offset[2][5]=(0,0,-1)

offset[2][6]=(-1,1,0)

offset[2][7]=(-1,-1,0)

offset[2][8]=(-1,0,-1)
offset[2][9]=(0,1,1)
offset[2][10]=(0,1,-1)
offset[2][11]=(-1,1,1)

//y%2=1 and z%3=1

offset[3][0]=(1,0,0)
offset[3][1]=(-1,0,0)
offset[3][2]=(0,1,0)
offset[3][3]=(0,-1,0)
offset[3][4]=(0,0,1)
offset[3][5]=(0,0,-1)
offset[3][6]=(1,1,0)
offset[3][7]=(1,-1,0)
offset[3][8]=(1,0,-1)
offset[3][9]=(0,1,1)
offset[3][10]=(0,1,-1)
offset[3][11]=(1,1,1)

//y%2=0 and z%3=2

offset[4][0]=(1,0,0)
offset[4][1]=(-1,0,0)
offset[4][2]=(0,1,0)
offset[4][3]=(0,-1,0)
offset[4][4]=(0,0,1)

offset[4][5]=(0,0,-1)
offset[4][6]=(-1,1,0)
offset[4][7]=(-1,-1,0)
offset[4][8]=(-1,0,1)
offset[4][9]=(0,-1,1)
offset[4][10]=(0,-1,-1)
offset[4][11]=(-1,-1,-1)

//y%2=1 and z%3=2

offset[5][0]=(1,0,0)
offset[5][1]=(-1,0,0)
offset[5][2]=(0,1,0)
offset[5][3]=(0,-1,0)
offset[5][4]=(0,0,1)
offset[5][5]=(0,0,-1)
offset[5][6]=(1,1,0)
offset[5][7]=(1,-1,0)
offset[5][8]=(1,0,1)
offset[5][9]=(0,-1,1)
offset[5][10]=(0,-1,-1)
offset[5][11]=(1,-1,-1)