Modelling the Snowflake with a Cellular Automata System

Liam Blake
a1742080

May 18, 2018

Report submitted for **ADVANCED MATHEMATICAL PERSPECTIVES I** at the School of Mathematical Sciences University of Adelaide.

Project Area: **PATTERN FORMATION**
Project Supervisor: **MATTHEW ROUGHAN**

Abstract

This report follows the development of a cellular automata model for the growth of snow crystals in the atmosphere. It interprets snowflake structure as a finite tessellation of discrete regular hexagonal cells.

We will investigate the capabilities, reasonableness and limitations of using such a model to simulate snowflake formation. It was found to be very effective at demonstrating several types of snowflake growth observed in nature. The model had some significant limitations and assumptions, which meant it was descriptive, rather than explanatory of the natural process of snowflake growth.

1 Introduction

One of the best examples of patterns formed in nature is the snowflake. The intricate and complex structures exhibited and their almost perfect hexagonal symmetry leads to snowflakes being considered one of nature's most extraordinary feats.

This report will present the development and application of a mathematical model for modelling the formation of a snowflake. The computing environment MATLAB will be used to visualise and test the models. These results will then be compared and related to real-life examples of snow crystal growth. The primary aims of this report are

- to develop a simple mathematical model of snowflake growth using cellular automata and diffusion;
- to use this model with different starting parameters to generate a variety of patterns; and
- to compare the results of this model with real-life examples and other models previously developed.

The model will be primarily developed using the concept of cellular automata, where a system of connecting cells evolve through a number of discrete time steps. We shall assume a snowflake is composed of a finite number of discrete hexagonal cells. A discrete interpretation of the diffusion equation, which can be used to model the movement of particles in a medium, will also be implemented as part of the model.

The model was found to produce patterns which exhibited different types of snowflake formations observed in nature, in particular dendrite and plate growth. It could generate patterns which were classifiable as different types of snow crystal shapes. However, there were some significant assumptions and limitations which restricted the usefulness of the model, especially when compared to natural processes. Finally, almost all the generated patterns exhibited six-fold rotational symmetry, which provided some insight into this distinct aspect of naturally observed snowflakes.

Figure 1: Simple examples of snowflakes formed with (from left to right) plates, dendrites and columns [3].

2 Background

Before beginning to develop a model, we must first understand the formation process of snowflakes. In general, ice crystals are formed when atmospheric water vapour condenses directly into ice. The crystal pattern is developed as water vapour accumulates. The behaviour of the water vapour, and hence the structure of the crystal formed, depend on the properties of the air and clouds in which the snowflakes are formed [2]. Snowflakes form from a tiny nucleus of ice, which grows into a hexagonal prism, owing to the molecular structure of ice. From this, more complex and sophisticated structures grow. There are several different types of ice crystal growth observed in nature, but these can be generalised into 3 groups:

- plates and dendrites,
- columns and needles,
- anomalies and variants,

with some simple examples shown in Figure 1.

The molecular structure of ice crystals, shown in Figure 2, explains several of the distinct features of snow crystals. Since the H_2O molecules arrange themselves in a hexagonal lattice, the resulting snowflake has its distinct 6-fold rotational symmetry.

This structure justifies the use of a discrete hexagonal lattice to model snowflake structure. Since water molecules arrange themselves in a hexagonal lattice in ice, a snowflake can be approximated as a finite number of hexagonal cells.

The main modelling technique used in this report is cellular automata. Cellular automata is a discrete model consisting of a grid of cells, each of

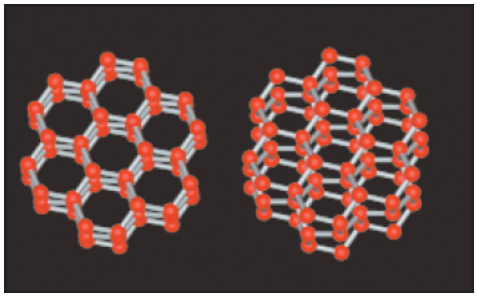

Figure 2: The molecular structure of H_2O in the normal form of ice in 3 dimensions. The red spheres represent oxygen atoms, while each gray bar is a hydrogen atom. Notice the flat, hexagonal structure [2].

which is in one of a set of finite states. The systems develops over a set of discrete time steps, with the states of the cells changing according to one or more fixed rules. Given a starting system, the system is iterated through each time step [7].

This report will apply cellular automata on a hexagonal lattice of cells, as described above. Each cell will be in one of two states, either frozen or not, which will determine the structure of the snowflake formed.

In addition, each cell has an associated value, which can be interpreted as the amount of water vapour present in the cell, which changes over time and depends on the values of the cells surrounding it. Each time step, it loses a certain amount of vapour to its nearest neighbouring cells, in addition to gaining some from these cells. This is analogous to diffusion, which dictates the spread of particles or energy through a system. Assuming that every cell has the same rate of vapour loss, and "absorbs" vapour at the same rate, the amount of vapour that a cell loses in every step is evenly spread about its 6 neighbours. So, if $l(t, \mathbf{p})$ is the amount of vapour lost in the step from t to $t+1$, the amount of water in a cell is

$$u(t+1, \mathbf{p}) = u(t, \mathbf{p}) - l(t, \mathbf{p}) + \frac{1}{6} \sum_{\mathbf{n} \in N_{\mathbf{p}}} l(t, \mathbf{n}),$$

where $N_{\mathbf{p}}$ is the set of 6 cells directly neighbouring $\mathbf{p}$. Let us assume that the rate which a cell loses its vapour is directly proportional to the amount of vapour present in it. That is

$$l(t, \mathbf{p}) = au(t, \mathbf{p}),$$

$$\begin{array}{ccccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{array} \longrightarrow \begin{array}{ccccc} a_{11} & & a_{13} & & a_{15} \\ & a_{12} & & a_{14} & \\ a_{21} & & a_{23} & & a_{25} \\ & a_{22} & & a_{24} & \\ a_{31} & & a_{33} & & a_{35} \\ & a_{32} & & a_{34} & \\ a_{41} & & a_{43} & & a_{45} \\ & a_{42} & & a_{44} & \\ a_{51} & & a_{53} & & a_{55} \\ & a_{52} & & a_{52} & \end{array}$$

Figure 3: A simple example of the convention used to represent a hexagonal lattice in a 2-dimensional array.

where a is some constant. Hence, the entire equation can be written as

$$u(t+1, \mathbf{p}) = u(t, \mathbf{p}) + a\left(-u(t, \mathbf{p}) + \frac{1}{6}\sum_{\mathbf{n}\in N_{\mathbf{p}}} u(t, \mathbf{n})\right).$$

This equation is a discrete representation of the diffusion equation

$$\frac{\partial \mathbf{u}}{\partial t} = \alpha \nabla^2 \mathbf{u},$$

which is used to model continuous systems [1]. A similar equation was used in Reiter's model [6]. Throughout this report, the discrete representation of this equation will be used to dictate the movement of water vapour between cells.

This report will deal with regular hexagonal lattices, owing to the hexagonal structure of ice molecules, which can be interpreted visually as a tessellation of regular hexagons. However, MATLAB works with 2 dimensional arrays which cannot be used trivially to represent a hexagonal lattice. Hence, a system for representing said lattices in a 2 dimensional, $m \times n$ array will be used throughout this report. The lattices will be encoded into an array such that every second (even-numbered) column is translated downwards, as shown in Figure 3. The set $N_{\mathbf{p}}$ in the array is hence different for the shifted and non-shifted rows.

3 Model

We shall begin with ideas presented in a Boolean model created and popularised by Norman Packard and Stephen Wolfram [5] and further developed by Clifford Reiter [6]. The model evolves on a hexagonal lattice, with each cell having a state of either being frozen or not. With each time step, any already frozen cell remains frozen. Each cell has a value, on which the state of the cell depends. The evolution of the system is governed by diffusion. This model can exhibit behaviour very similar to the actual plate growth of snowflakes, so we will use this idea as a starting point for our model.

Since, at a basic level, snowflakes are formed by the accumulation of water vapour, it is justifiable to attempt to model its growth with a cellular automata. Each cell accumulates water vapour, until it reaches a sufficient amount to freeze. We start with a small nucleus (i.e. a single frozen cell) and allow the system to develop over a finite number of discrete time steps.

While temperature is a function of time, this model shall assume that whether a cell is frozen or not depends solely on the amount of water vapour present in it. This is a significant assumption, but is more appropriate for a simple model, as a snowflake is formed by the accumulation of vapour.

Consider a hexagonal lattice, each hexagon having a radius of 1 arbitrary unit and representing a single cell. Each cell has an amount of water vapour u, as a function of the position vector of its center $\mathbf{p}$ and time t. The amount of water vapour in each cell evolves over time according to the equation

$$u(t+1,\mathbf{p}) = u(t,\mathbf{p}) + a\left(-u(t,\mathbf{p}) + \frac{1}{6}\sum_{\mathbf{n}\in N_{\mathbf{p}}} u(t,\mathbf{n})\right).$$

where $N_{\mathbf{p}}$ is the set of the position vectors of the six cells neighbouring $\mathbf{p}$. When the amount of water vapour in a cell reaches a certain threshold, which in this case will be 1, the cell freezes. Hence, each cell is one of two states, which is given by the function

$$s(t,\mathbf{p}) = \begin{cases} 1, & u(t,\mathbf{p}) \geq 1, \\ 0, & u(t,\mathbf{p}) < 1, \end{cases}$$

where $s(\mathbf{p}) = 1$ means the cell is frozen, while $s(\mathbf{p}) = 0$ means the cell is not.

Any additional growth to a snowflake should depend on the already formed ice crystals. That is, a cell is more likely to freeze if it is a direct

neighbour to an already frozen one. This introduces a new state to our model; one where a cell is more likely to freeze in some way. Reiter's model defines a new state of cells to describe this feature; a receptive state, which occurs when a cell is either frozen or directly neighbouring a frozen one. In this model, we assume that these cells do not lose any vapour. However, to promote the growth of the crystal, with each time step a constant amount of vapour will be added to these cells. Hence, the state of the cell with position $\mathbf{p}$ is now

$$s(\mathbf{p}) = \begin{cases} 1, & u(t,\mathbf{p}) \geq 1 \text{ or } u(t,\mathbf{n}) \geq 1, \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathbf{n} \in N_{\mathbf{p}}$ as before. A cell is considered to be receptive if $s(\mathbf{p}) = 1$, in which case, it is assumed to retain all vapour which enters it. This effectively simulates molecular growth, as bonds are much more likely to form and persist rather than break. Adding a constant with each time step will lead to the continued growth of the pattern, as more water is being added to the system. These receptive cells will not contribute to the addition of water to neighbouring cells. Hence, the entire system can be described with

$$u(t+1,\mathbf{p}) = \begin{cases} u(t,\mathbf{p}) + \gamma, & s(\mathbf{p}) = 1, \\ u(t,\mathbf{p}) + a\left(-u(t,\mathbf{p}) + \frac{1}{6}\sum_{\mathbf{n}\in N_{\mathbf{p}}} u(t,\mathbf{n})\right), & s(\mathbf{p}) = 0, \end{cases}$$

where the set $N_{\mathbf{p}}$ now denotes the *non-receptive cells* neighbouring $\mathbf{p}$.

Another factor to consider is the behaviour of the system at the boundaries. We are assuming that this system is isolated from external effects, so we want the system to be bordered by cells which do not add any vapour to the system. We will make the amount of water vapour in these outer cells will remain constant at 0, so the cells forming the boundaries of the system will lose vapour to these outer cells, but not gain any from them.

To prevent the resulting snowflake from taking up the entire system and being a flat plate, we will introduce a circular boundary around the center. If the frozen structure reaches this boundary, the iterative process stops. That is, the process stops if there is a frozen cell $\mathbf{f}$ where $s(\mathbf{f}) = 1$ which satisfies

$$\left|\left|\mathbf{f} - \left(\frac{n}{2}, \frac{n}{2}\right)\right|\right| > \left(\frac{n}{2}\right).$$

This is the model we will attempt to use to generate snowflake formations. The starting parameters are

- a, the rate of vapour loss in a cell. This is kept constant throughout the iterative process.
- k, the initial level of water vapour in every cell, except the central one. This must be a value such that $0 \leq u_0 < 1$ for the model to produce any useful results.
- γ, the amount of water vapour being added to receptive cells with each time step. This also remains constant.
- k_n, the initial temperature of the central cell of the system. In most cases, we will use $k_n = 1$ to simulate a frozen nucleus.
- n, the size of the system. In MATLAB, this will generate an $n \times n$ array using the convention shown in Figure 3.
- T, the number of iterations to run the system through.

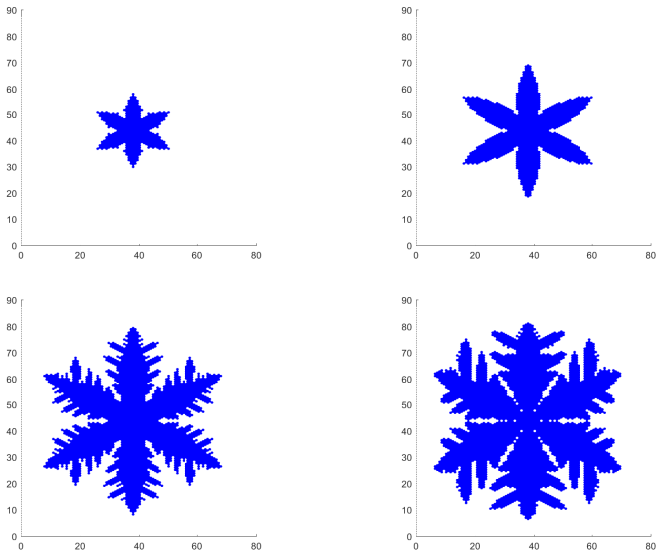

Figure 4: The patterns produced by varying just the parameter a. k, γ, n and T are kept constant at 0.01, 0.5, 101 and 500 respectively. Starting at the top left, the values of a are 0.1, 0.25, 0.5 and 1.

4 Results

Of the three primary parameters (a, k, and γ), a has the least impact on the patterned formed. Figure 4 shows that the structure is very similar for different values of a, with the parameter instead having more impact on the rate of growth on the snowflake.

The initial amount of background water vapour, k, has more of an effect on the pattern, illustrated in Figure 5. Similar to a, the parameter has an effect on the rate at which the snowflake grows. However, it also appears to have some influence on the intricacy and structure of the snowflake, particularly when its value is high. An example of this is the last example in Figure 5.

γ, on the other hand, has a significant effect on the type of growth exhibited by the model. As this value is increased, the frozen cells become more dense, resulting in plate patterns. Figure 6 shows that by just varying γ, the type of growth can range from dendrite to plate, or a combination of both.

Since a appears to have the least impact on the type of growth exhibited, it would be useful to compare the patterns formed as both k and γ are varied. Figure 7 shows that the value of k has an impact on the intricacy of the pattern, especially in the amount of dendrites formed, while

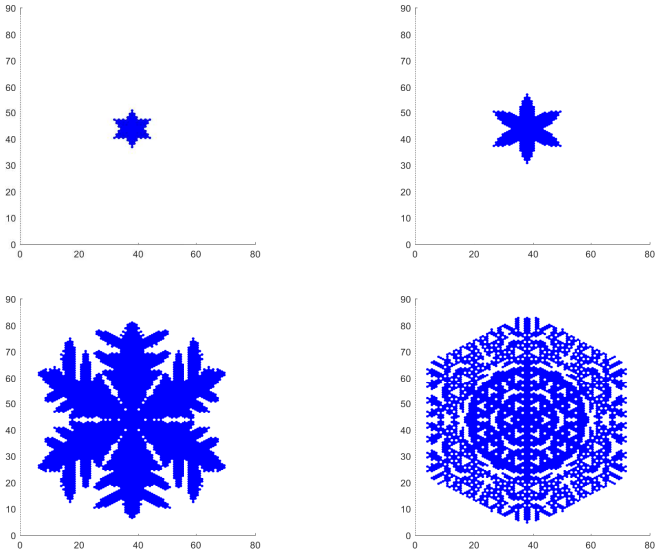

Figure 5: The patterns produced by varying just the parameter k. a, γ, n and T are kept constant at 1, 0.01, 101 and 500 respectively. Starting at the top left, the values of b are 0.1, 0.2, 0.5 and 0.9.

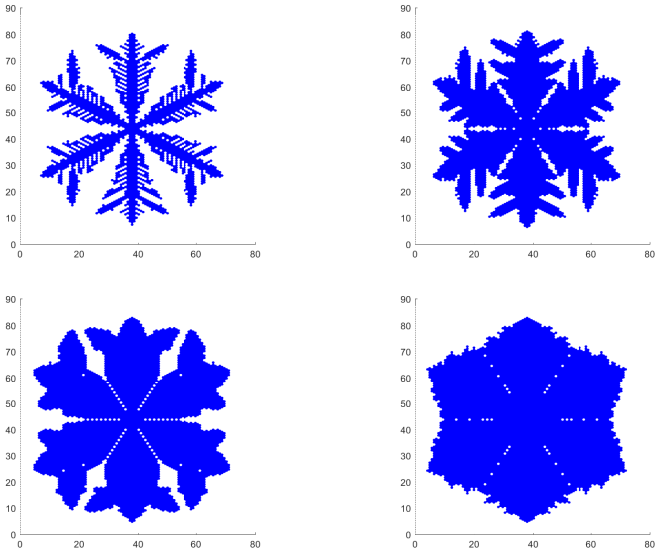

Figure 6: The patterns produced by varying just the parameter γ. a, k, n and T are kept constant at 1, 0.5, 101 and 500 respectively. Starting at the top left, the values of γ are 0.001, 0.005, 0.01 and 0.05.

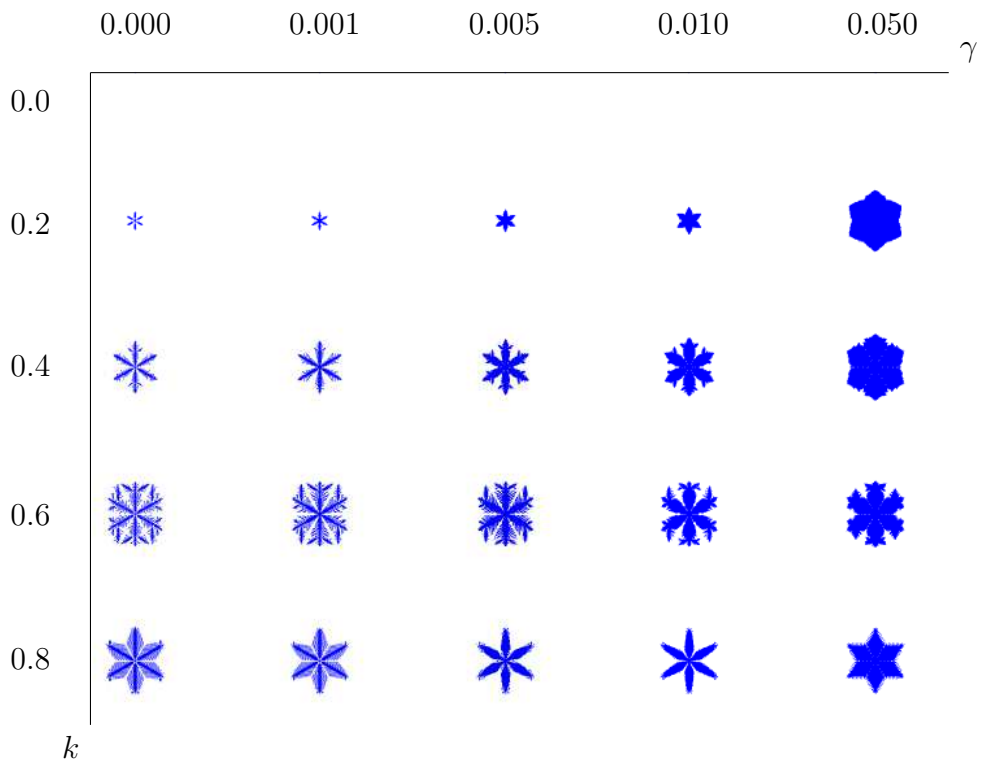

Figure 7: The different patterns formed by varying γ and k. a, n and T are kept constant at 1, 101 and 500 respectively.

γ affects the density, as shown before. Combining the two parameters results in a range of different crystal structures being formed, allowing the model to exhibit several different types of snow crystal growth.

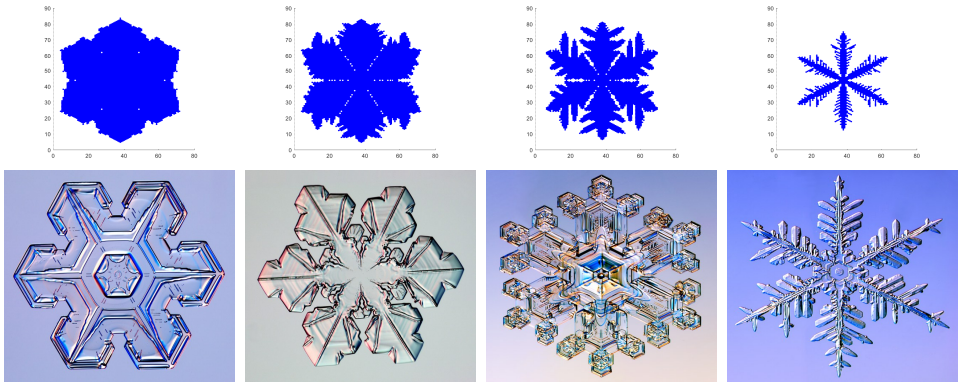

Figure 8: Patterns generated by the model (above), and real-life examples of the type of growth being exhibited. From left to right, the types of snowflakes are stellar plates, sectored plates, stellar dendrites and fernlike dendrites. [3]

5 Discussion

5.1 Comparison of Results to Nature

The results have shown that the model is capable of generating a wide range of patterns. With certain parameters, it can exhibit types of snow crystal growth commonly observed in nature. In particular, plate and dendrite growth is prominent in the resulting patterns. This is the most common type of snowflake growth observed in nature, so the model can model a wide range of snowflake types.

In particular, the results show examples of simple prism growth, stellar and sectored plate growth, stellar dendrites, fernlike stellar dendrites, and simple stars [3]. Some examples of these results compared to real-life examples of these types of growth are shown in Figure 8.

Although not perfect, almost every pattern produced by the model demonstrates some six-fold rotational symmetry. This is due to the system being composed of hexagonal cells, and the growth process being symmetrical. Since this is analogous to natural snowflakes, this feature exemplifies the appropriateness of using hexagonal cells to approximate a snow crystal. This observation also provides some insight into the rotational symmetry of natural snowflakes, as it shows that a hexagonal arrangement of cells (such as that of water molecules in ice) at a small scale can result in rotational symmetry being retained at a larger scale.

In general, the movement of water vapour in air depends on a range of conditions, such as humidity and temperature. The parameter a could be interpreted as a representation of these conditions. Similarly, the initial

amount of background vapour k is analogous to the humidity of air. This would also not be uniform or constant, like the model assumes, in any natural environment. Finally, γ could be considered a measure of the influence of frozen vapour on unfrozen vapour around it. That is, how much more likely vapour is to freeze if it is in close proximity to already frozen structure. The size of the system n and number of iterations T are arbitrary in a real-life system. However, since this model aims to be descriptive, rather than explanatory, it is not necessary to attempt to relate these parameters to nature.

The model is completely deterministic, with no randomness involved. The natural process of snowflake formation, however, can be considered stochastic.

From a qualitative perspective, most of the patterns which the model is capable of producing are convincing snowflakes. They are easily recognisable as common types of snowflakes, and the model is capable of producing intricate patterns often seen in nature. This fact further exemplifies the effectiveness of the model at describing the structure of snow crystals.

5.2 Assumptions and Limitations

The model makes many significant assumptions about the growth of snowflakes, namely;

- The structure of a snowflake consists of many small, hexagonal cells.
- The conditions in which the snowflake forms remain constant throughout its growth.
- The amount of water vapour present in a cell determines whether that cell forms part of the snowflake. The temperature is irrelevant and not considered to have an effect on this.
- The growth of a snowflake is contained within its system. That is, no external factors impact the growth.
- Water vapour moves at the same rate throughout this system.
- The 3-dimensional structure of a snowflake is insignificant enough for the entire flake to be considered 2-dimensional.

Many of these assumptions are not true in nature, but given the amount of conditions in a real-life system, it is difficult to encapsulate all of these. Studies have shown that the unique structures exhibited in snowflakes are the result of the crystal passing through many different conditions as it forms [4]. The model assumes that the conditions remain constant throughout the growth of the snowflake. Likewise, the movement of water vapour in the air would depend on the conditions (such as temperature), and would thus change as the snowflake moved through the atmosphere. Furthermore, while approximating a snowflake to a two dimensional shape is not realistic, the model could be extended to three-dimensions by considering the same process over a 3-dimensional lattice of hexagonal prisms.

It would make more intuitive sense to have each cell contain a temperature, which changes over time according to diffusion. However, this would be difficult to produce meaningful results from, as this process would not accurately simulate the actual growth of snowflakes. Instead of simply developing when water freezes into ice, snowflakes are formed by the accumulation of water vapour on a frozen nucleus. Thus, it is more appropriate to model this behaviour with the movement of particles, rather than diffusion. A combination of both could potentially be an effective model, but would require a much more complicated mathematical framework. Many more parameters would be required, but this suggests the possibility of a broader range of patterns which could be generated.

Having been derived in a similar way, using the same initial assumption, this model is very similar to Reiter's model [6], and produces similar results. This model, however, is more simplified, as its parameters and derivation are more straightforward.

Furthermore, by isolating the growth of the snowflake to a self-contained system, the resulting patterns lack the inherent randomness seen in natural snowflakes. Many snowflakes formed in nature have irregularities, the result of external factors such as other particles affecting the growth process [2]. The model assumes that external factors have no impact on the system, and so the formed patterns are always regular.

Since the model makes these significant assumptions, it does not accurately recreate the natural process in which snowflakes are formed. The assumption that a cell freezing depends on the amount of water vapour present, while producing effective results, is arguably not realistic. It assumes a constant temperature in the system, which is also not a realistic assumption. However, as a simple mathematical formulation describing the pattern formed, this model is very effective. It can produce a wide range of patterns easily recognisable and classifiable as snow crystals.

While its assumptions make it difficult to relate the model directly to nature, it works effectively as a descriptive model.

6 Conclusion

Throughout this report, we have developed a simple model for the formation of snowflakes. The model used cellular automata to simulate the accumulation of water vapour in the formation of snow crystals.

The model was found to be effective at modelling several types of snowflake growth, in particular dendrite and plate growth. By varying the parameters, different types of patterns could be generated, of which many could be compared to naturally formed snowflakes.

However, by making several significant assumptions, the model was ineffective at modelling the actual natural process of snowflake formation. Similarly, it was found to be difficult to relate the parameters of the model to natural measures. The model also had several limitations, which impacted its usefulness as a more general and robust model.

The results did, however, provide some insight into the symmetries found in natural snowflakes. The close-to-perfect six-fold rotational symmetry found in almost all the patterns generated suggested how a small-scale hexagonal structure can lead to larger scale symmetries.

A MATLAB Code

A.1 automataModel.m

```
function snowflakeModel(a,g,k0,n,t, fig)

%
%   snowflakeModel.m, Liam Blake, 2018
%
%   Generates a snowflake pattern using a cellular automata
%   model in 2 dimensions and plots the resulting image
%
%   Inputs: a = rate of water vapour movement
%           g = gamma constant
%           k0 = starting watre vapour level in cell
%           n = size of system (must be odd number)
%           t = number of iterations to run the system for
%           fig = figure number to print resulting ...
    pattern to
%

% starting parameters
kc = 1;   % initial temperature of nucleus
edge = 0; % temperature of edges

% define the system array
center = (n+1)/2;
sys = ones(n,n);
sys = sys*k0;
sys(center, center) = kc;

% apply the cellular automata model
sys = algorithmReceptive(sys,0.01,t,a,edge,g);

% plot the resulting snowflake
hexSnowPlot(sys,fig);

end
```

A.2 algorithmReceptive.m

```
function u_final = algorithmReceptive(u0, dt, n_t, alpha, ...
    edge, gamma)
%
%   algorithmReceptive.m, Liam Blake, 2018
%
%   ALGORITHMRECEPTIVE applies a cellular automata model ...
    to a discrete
%                       hexagonal lattice.
%
%   Input: u0 = the initial state of the system, a two ...
    dimensional array
%          dt = the time step to use
%          n_t = the total amount of time steps to run ...
    the system for
%          alpha = the constant of water vapour movement
%          edge = the value at the edges of the system
%          gamma = addition constant
%

% check the inputs are valid
assert(dt>0, 'dt should be > 0');
assert(alpha>0, 'alpha should be > 0');

% set up variables
n = size(u0);                   % the size of the system ...
    (rows, columns)
n_r = n(1);                     % the number of rows
n_c = n(2);                     % the number of columns
u = ones(n(2)+2,n(1)+2)*edge; % initialise system with zeroes
u(2:n(2)+1, 2:n(1)+1) = u0;    % insert initial system ...
    with padding of zeroes
T = (1:n_t)*dt;                 % time steps


for t = T
    % reinitialise variables
    u_old = u;
    receptive = zeros(size(u));
    normal = zeros(size(u));


    % split into receptive and non-receptive
    for r = 2:(n_r+1)
        for c = 2:(n_c+1)

            % check if boundary is frozen
            if ((c - (n_c+1)/2)^2 + (r - (n_r+1)/2)^2) >= ...
```

```
                ((n_r+1)/2)^2
42              if (u_old(r,c) >= 1)
43                  u_final = u(2:n(2)+1, 2:n(1)+1);
44                  return;
45              end
46          end
47
48          if mod(c,2) == 1        % column is shifted
49              neighbours = [u_old(r,c-1) u_old(r-1,c) ...
                    u_old(r,c+1) u_old(r+1,c+1) ...
                    u_old(r+1,c) u_old(r+1,c-1)];
50          else
51              neighbours = [u_old(r,c-1) u_old(r-1,c-1) ...
                    u_old(r-1,c) u_old(r-1,c+1) ...
                    u_old(r,c+1) u_old(r+1,c)];
52          end
53
54          if sum(neighbours >= 1) > 0 || u_old(r,c) >= 1
55              receptive(r,c) = u_old(r,c);
56          else
57              normal(r,c) = u_old(r,c);
58          end
59      end
60  end
61
62
63  % add constant to receptive cells
64  receptive = receptive + gamma*(receptive > 0);
65
66  % non-receptive cell process
67  normal_old = normal;
68  normal = zeros(size(normal));
69  for r = 2:(n_r+1)
70      for c = 2:(n_c+1)
71
72
73              % check if row shifted or not and find ...
                    neighbours
74              if mod(c,2) == 1        % column is shifted
75                  neighbours = [normal_old(r,c-1) ...
                        normal_old(r-1,c) ...
                        normal_old(r,c+1) ...
                        normal_old(r+1,c+1) ...
                        normal_old(r+1,c) ...
                        normal_old(r+1,c-1)];
76              else
77                  neighbours = [normal_old(r,c-1) ...
                        normal_old(r-1,c-1) ...
                        normal_old(r-1,c) ...
                        normal_old(r-1,c+1) ...
                        normal_old(r,c+1) normal_old(r+1,c)];
```

```
78                  end
79
80                  np = sum(neighbours);
81                  normal(r,c) = normal_old(r,c)+ ...
82                                      alpha*(-normal_old(r,c) ...
                                            + (1/6)*np);
83
84          end
85      end
86
87      % the resulting system
88      u = normal + receptive;
89
90  end
91
92  % return final system
93  u_final = u(2:n(2)+1, 2:n(1)+1);
94
95  end
```

A.3 hexSnowPlot.m

```
function hexSnowPlot(x,fig)
%
%   hexSnowPlot.m, Liam Blake, 2018
%
%HEXSNOWPLOT Given a logical array, draws a hexagonal ...
    tesselation of such
%
%   Inputs: x = the logical array of whether vectors. ...
    Must be a square
%               array
%           fig = the number figure to plot the image in. ...
    Must be a
%                 positive integer.
%
%   Function has no return values, instead creates a figure.
%

% confirm input is value
assert((fig > 0), 'Input fig must be a positive integer');

% set up variables
n = size(x);                                                % ...
    size of array
assert(n(1) == n(2), 'Input must be a square array');
n = n(1);

figure(fig);
hold on;
for r = 1:n         % row
    for c = 1:n     % column

        % determine vertice coordinates of hexagon
        if mod(c,2) == 0    % shifted column
            vert = hexCoords(0.5 + 0.75*(c-1), sqrt(3)/2 ...
                + ...
                                                    (sqrt(3)/2)*(r-1), ...
                                                       0.5);
            X = vert(:,1);
            Y = vert(:,2);
        else
            vert = hexCoords(0.5 + 0.75*(c-1), sqrt(3)/4 ...
                + ...
                                                    (sqrt(3)/2)*(r-1), ...
                                                       0.5);
            X = vert(:,1);
            Y = vert(:,2);
        end
```

```
40
41          % determine colour
42          if x(r,c) >= 1
43              C = [0 0 1];
44          else
45              C = [1 1 1];
46          end
47
48          % draw the hexagon
49          fill(X,Y,C,'edgealpha', 0.0);
50      end
51  end
52
53  axis square;
54
55  end
```

A.4 hexCoords.m

```
function [hex] = hexCoords(x,y,r)
%
%   hexCoords.m, Liam Blake, 2018
%
%   HEXCOORDS converts coordinates of the centre of a ...
    unit hexagon into
%
%   Inputs: x = x-coordinate of hexagon centre
%           y = y-coordinate of hexagon centre
%           r = radius of hexagon
%

stand = [-r 0; -r/2 r*sqrt(3)/2; r/2 r*sqrt(3)/2; r 0;
                                    r/2 -r*sqrt(3)/2; ...
                                       -r/2 -r*sqrt(3)/2];
hex = zeros(6,2);

for i=1:6
    hex(i,:) = [x y] + stand(i,:);
end

end
```

References

[1] Jessica Li. On the geometry and mathematical modelling of snowflakes and viruses. Fourth Annual MIT PRIMES Conference, May 2015.

[2] Kenneth Libbrecht. Snowflake science: A rich mix of mathematics, chemistry, and mystery. *American Educator*, 1:2–7, 2004.

[3] Kenneth Libbrecht. A guide to snowflakes. `http://www.its.caltech.edu/~atomic/snowcrystals/class/class-old.htm`, 2006. Accessed: 11-04-2018.

[4] Choji Mogona and Chung Woo Lee. Meteorological classification of natural snow crystals. *Journal of the Faculty of Science, Hokkaido University*, 1:321–355, 1966.

[5] Norman H. Packard. Lattice models for solidification and aggregation. In S. Ishizaka, editor, *Science on Form: Proceedings of the First International Symposium for Science on Form*, pages 95–101. Science of Form, Springer Netherlands, 1985.

[6] Clifford A. Reiter. A local cellular model for snow crystal growth. *Chaos, Solitons and Fractals*, 23:1111–1119, 2005.

[7] Eric Weisstein. Cellular automata. `http://mathworld.wolfram.com/CellularAutomaton.html`, April 2018. Accessed: 28-04-18.