Monte Carlo Simulations of Galaxy Formation and Evolution

Uri Pierre Burmester (u5561093) - Australian National University
(Dated: October 12, 2019)

This paper is an overview of the applications of Monte Carlo Methods (MCMs) to studying galaxy formation and evolution. An introduction to the astrophysics underlying the current theories of galaxy formation is given, including the Lambda Cold Dark Matter model and some background into the types and properties of galaxies. Next, the paper covers the basic premises of a Monte Carlo analysis. Then several papers are discussed as representatives of the current body of work, including strengths and weaknesses of the approaches. Lastly, simulation results are shown illustrating how MCMs can be used to estimate the fractal dimension of spiral galaxies, based on the work of Yoshino and Sagawa [1].

INTRODUCTION

In the past two decades, scientific papers on the topic of galaxy formation and evolution have exploded in popularity. This interest owes partly to the continual improvement of astronomical equipment such as the construction of ever-larger telescopes and more precise optics, allowing for observations of fainter and more distant objects. However, another contributing factor is the dramatic increase in the availability, speed and usability of computing power. While semi-analytical models purporting to describe galactic processes have existed for some time [2, 3], these do not have closed-form solutions nor easily-measurable parameters, making numerical methods a vital tool for evaluating our models and comparing them with observations. This paper introduces some of the physical models underlying galaxy formation and evolution, introduces Monte Carlo Methods (MCMs) which can be applied to these situations and discusses the benefits and limitations of these approaches. Lastly, a simplified model of galaxy formation from Yoshino and Sagawa is replicated to demonstrate the principle [1].

PROPERTIES OF GALAXIES

The central question of galaxy formation can be understood by considering the state of the universe at the present time and comparing it to a time shortly after the Big Bang. The early universe was extremely hot and very homogeneous, containing only hydrogen and helium. In the present day, however, the universe is very inhomogenous on large length scales - intergalactic space has a density of only around one atom per cubic metre, with distances between galaxies in the millions of light-years [4]. Inside galaxies however, hundreds of billions of stars are crowded into a space only thousands of light-years across. This transition from a homogeneous to an inhomogeneous universe is what models of galaxy formation and evolution seek to explain [5].

Models of galaxy formation and evolution need to accurately predict more than simply the existence and mass of galaxies, though. Galaxies have several other distinguishing properties which these models seek to fit. One of these is morphology, or galaxy shape; galaxies can be broadly grouped into elliptical, spiral and spiral barred based on their shape. Spiral galaxies also belong to the class of 'disk galaxies'. Galactic discs consist of a stellar component (composed of most of the galaxy's stars) and a gaseous component (mostly composed of cool gas and dust) [5]. Some of the more common galaxy morphologies are shown in Hubble's classification system, as shown in Figure 1. Other properties include metallicity (the proportion of observable mass made up of atoms heavier than hydrogen and helium) and fractal dimension, D. This last measure is important to determine as the large-scale structure of matter in the universe is often modelled as a scale-invariant fractal of dimension D.

These explanations must also take into account a model of dark matter, which is hypothesised to make up the majority of mass in galaxies despite not interacting via the electromagnetic force. Measurements of the rotations of galaxies' outer stars reveal that these stars rotate faster than would be possible for a galaxy containing only baryonic mat-

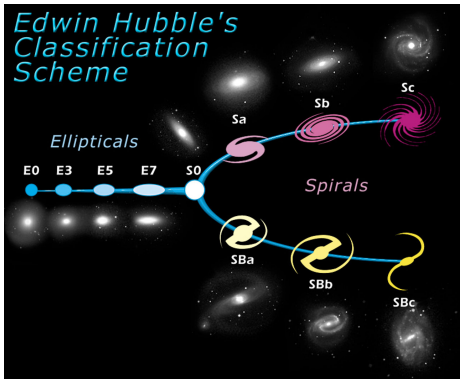


FIG. 1: Galaxy Morphology Classification System proposed by Edwin Hubble. Source: [6]

ter. Dark Matter 'halos' are hypothesised to envelop the galactic disc and extend well beyond the edge of the visible galaxy [7].

EXISTING MODELS

'Bottom-up' models of galaxy formation have been gaining increasing acceptance among astrophysicists. The Lambda Cold Dark Matter (LCDM) model is one such model which posits that the universe contains cold dark matter (CDM). The "cold" in this case refers to the average speed of the dark matter particles, they are relatively slow compared to the speed of light [5]. Slight differences in the density of the primordial universe during its inflationary stage can be observed in the cosmic microwave background radiation. These mass overdensities attract more mass than less dense areas, causing the disparity in density to grow and allowing galaxies to accumulate mass by clustering and merging. The Press-Schechter formalism is one of the most widely used mathematical formalisms for understanding this - it attempts to predict the number of objects (e.g. galaxies, galaxy clusters, dark matter halos) of given mass within a universe of given cosmological properties [8]. Press and Schechter predict that in a scale-free power spectrum, the number of objects between masses M and $M + dM$ is dn.

$$M\frac{dn}{dM} = \frac{1}{\sqrt{\pi}}\frac{\bar{\rho}}{M}\left(\frac{M}{M^*}\right)^{1/2} e^{-M/M^*} \qquad (1)$$

Where M^* is a cut-off mass below which structures will form and $\bar{\rho}$ is the mean (baryonic and dark) matter density of the universe during the period of the initial mass fluctuation. The Press-Schechter formalism is derived by assuming that each object is formed by gravitational collapse of a density fluctuation. [8].

MONTE CARLO METHODS

In many ways, galactic evolution is very well suited to numerical simulation. The dynamics are very easily expressible in terms of individual stars, which are a useful discrete 'particle' on which to base a simulation. The relative emptiness of the universe also makes the physical situations simple to model because of the absence of most disperse forces such as friction and air resistance. The relative abundance of stars in the observable universe and long timescales of astronomical systems also assures that stars as a whole can be well-approximated by statistical distributions [9].

Monte Carlo Methods are one such set of computational tools. Fundamentally, the principle underlying these algorithms is repeated random sampling. By utilising many random samplings, we are able to simulate an effect which may be deterministic in principle. The three most common applications of this are: numerical integration, optimisation and generating draws from a random distribution. N-Body Monte Carlo simulations can be used for all three of these applications; the classic example of this is using N randomly distributed particles in a square of area A to estimate the value of π. As $n \to \infty$, π can be approximated as

$$\pi = \lim_{n\to\infty} 4\frac{N_c}{N}$$

where N_c is the number of particles inscribed in the circle. An example of the third application is the so-called Markov-Chain Monte Carlo (MCMC) method. When a random distribution is known, a Markov chain model can be designed that will approximate this probability distribution in the limit as the number of steps goes to infinity. One application of this is calculating the conditional probability $P(D|x)$ that a given data set D occurs given the values of some model parameters, x. Bayes' Theorem states that, given a prior $P(x)$, the posterior probability of the model $\pi(x)$ is

$$\pi(x) \propto P(D|x)P(x)$$

[9]. A MCMC method then simply steps from one point in the sample space to the next, choosing from a distribution that depends only on the preceding point. This can be thought of like traversing a tree where nodes are connected by edges with associated probabilities. The main advantage of MCMC is that the posterior distribution and correlations for the parameters can be recovered from the sample list and the un-normalised probability [9].

MCM FOR SIMULATING GALACTIC EVOLUTION AND FORMATION

Knowing this background, we can see some examples of MCMs in use in astrophysics. One application is simulating the type and distribution of isotopes present in a galaxy during its lifetime. This is a 'Galactic Chemical Evolution' model and aims to provide an estimate of the Star Formation Rate (SFR), initial mass function (IMF), types and occurrences of supernovae (SN Ia, Ib/c, II) and the stellar nucleosynthetic contributions of stars with distinct mass and metallicity. Sahjipal and Kaur instead approached this problem using an N-Body Monte Carlo simulation, where each 'particle' was a star with a mass between 0.1 to 100 solar masses. Stars were tracked inside a simulated galaxy that was divided into annular rings using a stellar number distribution function $G_i(t, m)$, where G_i gave the number of stars of a particular mass m at time t in ring i. Sahjipal and Kaur adopted a two/three infall accretion model, where galaxy halo was formed first and the galactic halo was formed second from the gradual accumulation of intergalactic gas [10].

The simulation timestep was 10^6 years. Each of the annular rings evolved distinctly in terms of star formation and stellar nucleosynthesis, with some mixing of interstellar gas taking place between the rings in each timestep. After each timestep, new stars are formed based on the availability of gas in each ring, the star formation rate, and metallicity. The stellar ejecta from these stars are homogenized instantaneously within the entire annular ring, thereby enriching the metallicity of the ring. The nucleosynthetic yields of the stars were based on stellar surveys [10].

After completing thousands of simulations including various different initial conditions, Sahijpal and Kaur concluded that the Milky Way accreted during the initial stages ($\leq 1Gyr$) to form the halo, then later the disc. They also concluded that the three-infall accretional model seems to be the most viable scenario that explains the majority of the evolutionary features related to the elemental abundance distributions and gradients. Using an N-body simulation in this case provided significant advantages over past methods in order to reach these conclusions. Previous models, such as those based on Press-Schechter theory, relied on a complex set of differential equations which were based on (possibly incorrect) assumptions about the formation and evolution of the galacic halo and disc [10]. The MCM simulation also better accounts for the radial flow of isotopes than do differential equation models. Astronomical observations indicate that stars drift away from their birthplaces, influencing the chemical and kinematical properties of the galactic disc. Intragalactic gas is also observed to experience radial flow, influencing chemical abundance within galaxies [10].

There are also drawbacks to this model, though. In simplifying the underlying dynamics, the MCM risks losing important features of the system. For example, Sahjipal and Kaur's simulation assumes the stellar ejecta at each timestep are immediately mixed across the whole annular ring, an assumption that certainly wouldn't be true of a spiral galaxy which has alternating bands of high and low stellar density. The timestep of 10^6 years is also necessary for computational expediency and would represent the lifespan of most stars well; however, very large stars have lifespans on the order of 10^6 years, meaning their dynamics are practically not modelled at all. Choosing a timestep, scale and update equations to balance computational complexity and the complexity of the underlying system is a perennial problem of these kinds of simulations.

Henriques et al. are a good example of using MCMC methods to model galaxy formation. They used the data created by the Millennium dark matter simulation to explore the relevant cosmological parameter space. This simulation used the evolution of more than 10 billion particles in a simulated universe over 2 billion light-years on a side, creating terabytes of data. Scientists such as Springel

et al. then coupled these large datasets to Semi-Analytic (SA) models in an attempt to create models of interstellar dust, galaxy mergers and other phenomena [11]. These models could be made to reproduce many observable properties however Henriques et al. observed that the choice of parameters used for these models had never been studied in a statistically consistent way. Some of these properties include the cosmic baryon fraction, star formation frequency, hot gas black hole accretion rate, etc. The large number of observational properties that galaxy formation models attempt to predict requires a large number of parameters (some of which are strongly correlated). This causes considerable difficulties in determining how to improve the agreement with new observations without destroying the match with existing data sets [12].

These difficulties can be overcome by combining multiple observations with proper sampling of high-dimensional parameter spaces, which Henrique et al. accomplish by applying MCMC methods to the semi-analytic (SA) model created by Springel et al. [11]. Henrique et al. sampled a subset of the Millennium dataset at each MCMC step, running the SA model with the proposed set of parameters and computing the acceptance probability by comparing the outputted galaxy properties with the observational constraints [12]. One of these was by comparing the model with obsevations of the K-Band (infrared) luminosity distribution using a χ^2 probability test. The researchers then produced one- and two-dimensional maximum likelihood and MCMC posterior distributions, allowing them to quantify the correlation between the sets of simulation parameters. They then repeated this analysis using other features to obtain mean values, confidence limits and likelihood contours for the best-fitting models. Ultimately, Henrique et al. find that the previously published parameter values lie within their confidence limits [12].

One of the main advantages of this work is that it provides a statistically rigorous way of demonstrating the correlation between sets of simulation parameters, which provides a strong indication of an underlying deterministic relationship between those variables. Additionally, by placing confidence intervals on the best-fit parameters, researchers are able to make informed choices about which simulation values are well known versus those that have a high uncertainty. One disadvantage of using these statistical tests to form a confidence interval is that the results are only as reliable as observational datasets against which they are compared. If they contain some systematic error, for example, this will propagate into the prior/posterior distributions and taint the results.

SIMULATING RESULT FROM YOSHINO AND SAGAWA

Yoshino and Sagawa are primarily interested in exploring the homogeneity of the universe at different length scales, as this is vital to understanding the process of galaxy formation. This paper is primarily focused on spiral galaxies and quantifies the homogeneity using the fractional dimension. Historically, the fractal dimension of the large scale matter distribution in the universe was first discussed by Peebles [3], who predicted a value of $D = 1.2$ in a universe whose scale is smaller than 20 MPc. Elmegreen obtained the fractal dimensions from the observations of several star formations in the spiral galaxy NGC 2207; he found a value $D = (1.12 \pm 0.25)$ [3]. Thanki also analyzed the intensity levels of spiral galaxy images and showed that the fractal dimensions spread from 0.8 to 1.5 with the average around 1.2. Yoshino and Sagawa attempt to construct a stochastic model which results in the same behaviour [1]. The next paragraphs describe Yoshino and Sagawa's methodology, which was also used in our replication simulation.

We require a model of stellar evolution. Large stars typically live for around 10^6 years before undergoing a series of rapid expansions and contractions which lead to an explosive supernova. The force of this explosion propels the gasses which made up the star hundreds of parsecs, forming new clouds with different areas of inhomogeneous density. This leads to the formation of new stars.

Our model consists of a polar grid of $Nr = 149$ concentric rings, each with $6N$ cells (where N is the ring number). These cells are 100pc in size, corresponding to a typical galactic radius of 150 kpc. Assume a rotation of the cells of $Vr = 2cells/\Delta t$ $(200km/s)$, for a time step of $\Delta t = 10^6 yrs$. Lastly, we randomly populate 2 % of the cells in the inner 120 rings with large stars with a lifespan of $10^6 yrs$. Every timestep, these stars experience a supernova and their shockwave is transmitted to the

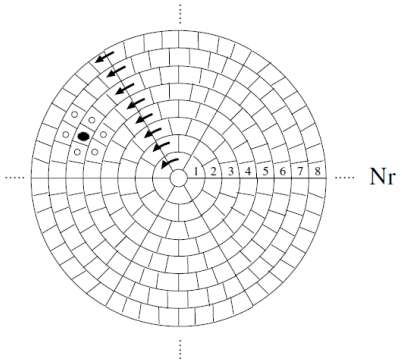


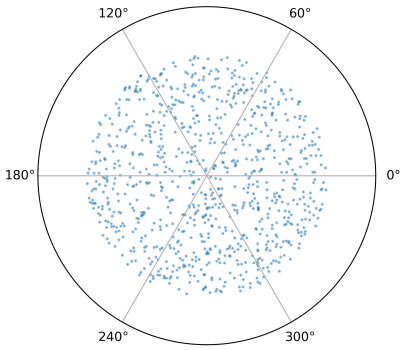
FIG. 2: Polar array used to simulate galaxy formation. Source: [1]

adjacent cells at a speed of $V_{sw} = 0.2 cells/\Delta t$, as shown in Figure 2. Each of these 'neighbour' cells will form a new star with a probability $P_{sw} = 0.183$.

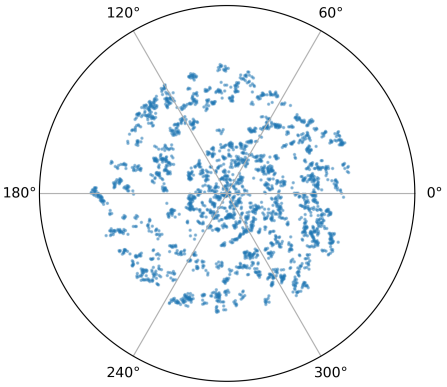
Our replication of these results is shown in Figure 3. It is clear that this method visually achieves some success in modelling the formation of spiral galaxies. The authors were able to achieve convincing spiral galaxy shapes after only 100 timesteps (i.e $100 Myr$) after which the shape remained stable; this was consistent with our results. The probabilistic element of the simulation lies in the random initial distribution of the large stars and in the probabilistic method of star formation. This allows us to make an empirical argument about the role played by supernovae in the formation of the spiral galaxy structure without constructing an unwieldy analytical model. The fractal dimension is then computed using a two-point correlation method, which gives an estimated fractal dimension of $D = 1.25$ [1]. This experiment was then repeated using a more complicated stochastic model with different star types and different gas densities across the array, but the result was similar.

CONCLUSION

In this essay, we have seen an introduction to the ways that Monte Carlo Methods are being used to understand the formation and evolution of galaxies. We initially introduced the concept of overdensities, that galaxies form from the gradual accretion of matter that was originally very homogeneous. We then introduce some of the characteristics of galaxies, such as their mass, fractional dimension and



(a) Initial, random star distribution



(b) Distribution after 300 steps ($300 Myr$)

FIG. 3: Simulation of galaxy formation using Yoshino and Sagawa's formalism

metallicity and then several Monte Carlo Methods that are relevant to this problem. We then discussed Henriques et al.'s use of MCMC methods for parameter estimation, Sahijpal and Kaur's stochastic model of elemental abundance and finally, we replicated Yoshino and Sagawa's simulation of the formation of spiral galaxies. Lastly, we discussed the importance of this result in verifying experimental observations of the fractal dimension.

REFERENCES

[1] D. Yoshino and H. Sagawa, "MONTE CARLO SIMULATION OF SPIRAL GALAXY FORMATION," *Fractals*, vol. 16, no. 02, pp. 119–127, Jun. 2008.

[2] J. Silk, "Cosmic Black-Body Radiation and Galaxy Formation," *The Astrophysical Journal*, vol. 151, p. 459, Feb. 1968.

[3] P. J. E. Peebles, "The Black-Body Radiation Content of the Universe and the Formation of Galaxies." *The Astrophysical Journal*, vol. 142, p. 1317, Nov. 1965.

[4] "The Big Bang Model," https://www.atnf.csiro.au/outreach/education/senior/cosmicengine/bigbang.html, May 2019.

[5] "The Formation of Galaxies," https://www.atnf.csiro.au/outreach/education/senior/cosmicengine/galaxy_formation.html, May 2019.

[6] "The Hubble tuning fork - classification of galaxies — ESA/Hubble," https://www.spacetelescope.org/images/heic9902o/.

[7] "Galaxy - Evolution of galaxies and quasars," https://www.britannica.com/science/galaxy.

[8] W. H. Press and P. Schechter, "Formation of Galaxies and Clusters of Galaxies by Self-Similar Gravitational Condensation," *The Astrophysical Journal*, vol. 187, p. 425, Feb. 1974.

[9] S. Paltani, "Monte Carlo Methods," *Monte Carlo*, p. 90, 2011.

[10] S. Sahijpal and T. Kaur, "A Monte Carlo based simulation of the Galactic chemical evolution of the Milky Way Galaxy," *Monthly Notices of the Royal Astronomical Society*, vol. 481, no. 4, pp. 5350–5369, Dec. 2018.

[11] V. Springel, S. D. M. White, A. Jenkins, C. S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J. A. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. Colberg, and F. Pearce, "Simulating the joint evolution of quasars, galaxies and their large-scale distribution," *Nature*, vol. 435, no. 7042, pp. 629–636, Jun. 2005.

[12] B. M. B. Henriques, P. A. Thomas, S. Oliver, and I. Roseboom, "Monte Carlo Markov Chain parameter estimation in semi-analytic models of galaxy formation," *Monthly Notices of the Royal Astronomical Society*, vol. 396, no. 1, pp. 535–547, Jun. 2009.

CODE

The code used to create the spiral galaxy simulation from Figure 3 is shown below.

Listing 1: code/main.py

```
import matplotlib.pyplot as plt
from GalaxySimulation import AdvanceTimestep, PlotStarsFromStellarArray,
    ↪ PrintPercentageComplete, stellarArray, queue

nSteps = 300    # Total number of simulation timesteps

# Plot initial setup
PlotStarsFromStellarArray(stellarArray)
plt.savefig("galaxySimulationBefore.png", dpi=400)

# Advance Timestep and plot
for step in range(nSteps):
    AdvanceTimestep(step, stellarArray, queue)
    PrintPercentageComplete(step, nSteps)

PlotStarsFromStellarArray(stellarArray)
plt.savefig("galaxySimulationAfter.png", dpi=400)
```

Listing 2: code/GalaxySimulation.py

```
import numpy as np
np.random.seed(2)
import matplotlib.pyplot as plt

## Set simulation parameters
Nr          = 149   # Number of rings
Psw         = 0.183 # Probability of new star formation
Vr          = 2     # Rotational speed of galaxy, Cells/timestep
Vsw         = 0.2   # Supernova shockwave velocity, Cells/timestep
innerRing   = 120   # Rings in which stars can be initialised
initProb    = 0.02  # Number of stars initially spawned as a percentage of
    ↪ possible cells

def NumberOfCellsInNRings(n):
    '''
    Returns the number of cells in n polar rings. There are 6n cells in the
        ↪ nth ring so this is equal to 6 x the nth triangular number
    '''
    return int(6 * (n ** 2 + n)/2)

## Auxillary quantities
#offset     = (np.pi/6)  # angular offset in radians
delaySteps  = int(1/Vsw) # Timesteps before new star formed
nCells      = NumberOfCellsInNRings(Nr)
nCellsInner = NumberOfCellsInNRings(innerRing)

```

```
25
26 # Preinitialise Stellar Array
27 # The stellarArray is a long vector with nCells entries. A 1 corresponds to
   ↪  a star. The linear index can be translated into (ring,index)
   ↪ coordinates
28 stellarArray                = np.zeros(nCells)
29 stellarLocations             = np.random.choice([0, 1], size=nCellsInner, p
   ↪ =[1-initProb, initProb])
30 stellarArray[0:nCellsInner] = stellarLocations
31
32 # Initialise the queue
33 # The queue holds the locations of new stars to be added to the
   ↪ stellarArray
34 queue = [[],[],[],[],[]]
35
36 def PrintPercentageComplete(step, nSteps):
37     '''
38     Print a completion percentage during the simulation, in steps of 10%
39     '''
40     if (step % round(nSteps/10) == 0):
41         print(str(step/nSteps* 100) + " pc compete")
42
43 def FindNeighbours(ring, idx, nRings):
44     '''
45     FindNeighbours returns the six 'neighbour' cells to (ring,idx) which
       ↪ the supernova shockwave will be propagated to
46     '''
47
48     sector              = np.ceil(idx/ring)
49     numNeighbours       = 0
50     neighboursFound     = []
51
52     # neighbours on same ring
53     numNeighbours   += 2
54     idxM1, _, idxP1 = AdjIndices(ring, idx)
55     neighboursFound.append([ring, idxM1])
56     neighboursFound.append([ring, idxP1])
57
58     # neighbours on outer ring
59     if ring < nRings:
60         numNeighbours += 2
61         idxM1, idxP0, idxP1 = AdjIndices(ring + 1, idx + sector)
62         neighboursFound.append([ring + 1, idxP0])
63         neighboursFound.append([ring + 1, idxM1])
64
65     # neighbours on inner ring
66     if ring > 1:
67         numNeighbours += 2
68         idxM1, idxP0, idxP1 = AdjIndices(ring - 1, idx - sector)
69         neighboursFound.append([ring - 1, idxP0])
```

```
70          neighboursFound.append([ring - 1, idxP1])
71
72      return (neighboursFound, numNeighbours)
73
74  def AdjIndices(ring, idx):
75      '''
76      AdjIndices returns the circular the adjacent indices to idx on ring,
            ↪ wrapping maxIdx + 1 to 1
77      '''
78
79      maxIdx = 6 * ring
80
81      idxP0 = idx
82      while idxP0 <= 0:
83          idxP0 += maxIdx
84
85      # Find the below index
86      trialIdx = idxP0 - 1
87      if trialIdx == 0:
88          idxM1 = maxIdx
89      else:
90          idxM1 = trialIdx
91
92      # Find the above index
93      trialIdx = idxP0 + 1
94      if trialIdx > maxIdx:
95          idxP1 = 1
96      else:
97          idxP1 = trialIdx
98
99      return (int(idxM1), int(idxP0), int(idxP1))
100
101 def RingIdxFromLinearIdx(linIdx):
102     '''
103     This function takes the linearIdx (the location in stellarArray) and
            ↪ returns the (ringIdx, circIdx) equivalent
104     '''
105
106     ringIdx = 1
107     circIdx = linIdx + 1
108
109     while circIdx > 6 * ringIdx:
110         circIdx -= 6 * ringIdx
111         ringIdx += 1
112
113     return (ringIdx, circIdx)
114
115 def RThetaFromRingIdx(ring, idx):
116     '''
117     This function converts an (ring, idx) coordinate pair to polar
```

```
        ↪ coordinates (r,theta) for plotting
118     '''
119     offset      = (np.pi/6)/ring  # angular offset in radians
120     numElems    = 6 * ring
121     theta       = offset + (idx - 1)/(numElems) * 2 * np.pi
122
123     return (ring, theta)
124
125 def PlotStarsFromStellarArray(stellarArray):
126     '''
127     This function produces a matplotlib.pyplot.polar plot of the stars in
        ↪ stellarArray
128     '''
129     radii   = []
130     theta   = []
131     labels  = []
132
133     # For each star in stellarArray, find its corresponding polar
        ↪ coordinate pair and plot them
134     for index, val in np.ndenumerate(stellarArray):
135         if val == 1:
136
137             ring,idx = RingIdxFromLinearIdx(index[0])
138             r,th = RThetaFromRingIdx(ring, idx)
139             lab = str((ring, idx))
140
141             radii.append(r)
142             theta.append(th)
143             labels.append(lab)
144
145     # Plot
146     fig = plt.figure()
147     ax = fig.add_subplot(111, projection='polar')
148     ax.scatter(theta, radii, marker='o', alpha=0.4, s=2)
149     ax.set_xticks(np.pi/180. * np.linspace(0,  360, 6, endpoint=False))
150     ax.set_thetalim(0, 2 * np.pi)
151     ax.set_rticks([])
152     # plt.show()
153
154 def RotateStellarArrayCells(stellarArray):
155     '''
156     RotateStellaArrayCells compensates for the rotation rate of the galaxy
        ↪ Vr by rotating each ring in stellarArray to its new location
157     '''
158
159     for ring in range(1,Nr+1):
160         marker = 6 * (ring - 1)
161
162         currRing = stellarArray[marker:marker+6]
163         rotdRing = np.roll(currRing, Vr)
```

```
164
165             stellarArray[marker:marker+6] = rotdRing
166
167         return stellarArray
168
169
170
171 def AdvanceTimestep(stepNumber, stellarArray, queue):
172     if (stepNumber % delaySteps == 0):
173         for index, val in np.ndenumerate(stellarArray):
174             if val == 1:
175
176                 # For each star in stellarArray ...
177                 linearIdx = index[0]
178
179                 # Make the star go nova
180                 stellarArray[linearIdx] = 0
181
182                 # Find its position in ring/idx coordinates
183                 ring,idx = RingIdxFromLinearIdx(linearIdx)
184
185                 # Find its neighbours
186                 nghbrs, nNghbrs = FindNeighbours(ring, idx, Nr)
187
188                 # Determine if any new stars will be formed
189                 for i in range(nNghbrs):
190                     newStar = np.random.choice([True,False],p=[Psw, 1-Psw])
191
192                     if newStar:
193                         newStarLoc = nghbrs[i]
194                         AddStarToQueue(newStarLoc, queue)
195
196         # Rotate all the cells
197         stellarArray = RotateStellarArrayCells(stellarArray)
198
199     # Update the queue, adding new stars and adding a new empty queue item
        ↪ at the end
200     UpdateQueue(stellarArray, queue)
201
202 def AddStarToQueue(newStarLoc, queue):
203     '''
204     When a new star is to be formed after delaySteps, this function adds it
        ↪ to the queue
205     '''
206     queue[delaySteps - 1].append(newStarLoc)
207
208 def UpdateQueue(stellarArray, queue):
209     '''
210     UpdateQueue adds stars in the 0th element of the queue to stellarArray.
        ↪ It then pops this element and adds a new empty element at the end
```

```
211         '''
212
213         # Grab the 0th element
214         currQueue = queue[0]
215
216         # Put scheduled stars into stellarArray
217         for i in range(len(currQueue)):
218             ring = currQueue[i][0]
219
220             oldIdx = currQueue[i][1]
221             newIdx = (oldIdx + delaySteps * Vr) % (6 * ring)
222
223             if newIdx == 0:
224                 newIdx = 6 * ring
225
226             linearIdx = LinearIdxFromRingIdx(ring, newIdx)
227             stellarArray[linearIdx] = 1
228
229         # Remove the just-added stars from the queue
230         queue.pop(0)
231         queue.append([])
232
233
234 def LinearIdxFromRingIdx(ring, circIdx):
235         '''
236         This function translates from (ring, circIdx) representation to a
              ↪ linearIdx, its location in stellarArray
237         '''
238
239         return NumberOfCellsInNRings(ring-1) + circIdx - 1
```