universität wien

MASTERARBEIT

Titel der Masterarbeit:

"Discordian Adaptive Networks: on the Fragility of the Mean Field in Opinion Formation Models"

Verfasser:

Matan Bendix Shenhav BSc

angestrebter akademischer Grad

Master of Science (MSc)

Wien, November 2013

Studienkennzahl lt. Studienblatt:	A0948694
Studienrichtung lt. Studienblatt:	Mathematik
Betreuer:	Prof. Joachim Hermisson

Abstract

Es wurden vier Modelle, durch Betrachtung der Molekularfeldgleichungen, der Meinungsbildung auf adaptive Netzwerke erörtert. Das Makroskopische Verhalten von verschiedenen mikroskopischen Imitations- und Trennungs-Mechanismen wurden verglichen und zeigen signifikante Differenzen im Molekularfeld auf, welches sich nicht immer in vergleichbaren stochastischen Simulationen widerspiegelt. Auch im Falle des Molekularfeldes, besagen die Modelle eine Fragilität zur Perturbation durch einen Rauschparameter, der unabhängige Innovationen darstellt. Verschiedene Probleme der Molekularfeldannäherung sind erfasst, vor allem zu beachten sind die qualitativen Differenzen der Stabilitätseigenschaften und Konvergenzzeiten sowie topologische abweichende Bifurkations Diagramme. Daraus wird der Schluss gezogen, dass für ein umfassendes Verständnis des stochastischen adaptiven Netzwerkmodells, Betrachtungen jenseits des Molekularfeldes stattfinden müssen.

Abstract

Four models of opinion formation on adaptive networks are investigated by means of moment closed mean field equations. The macroscopic behavior of different microscopic imitation and segregation mechanism is compared, revealing significant differences in the mean field which do not always seem reflected in the corresponding stochastic simulations. The models also prove fragile to perturbation by a noise parameter representing independent innovation, even in the mean field case. Various problems with the mean field approximation are cataloged, most notably qualitatively different stability properties and convergence times, and topologically differing bifurcation diagrams. It is concluded that comprehensive understanding of stochastic adaptive network models must look beyond the mean field.

Contents

Dedicated to my Dear Mother,
whose continuous love and support
got me to where I am.

1 Introduction

Since the early 20th Century, the question of how the *macroscopic behavior* of a complex system emerges from its *microscopic properties* has been posed. In the context of statistical mechanics, the particles in a physical system are typically modeled as nodes on a graph (often a regular lattice), with edges representing the adjacency of a pair of particles. On the microscopic level each node typically has a finite number of states; one of the classic models of statistical mechanics - the Ising Model of ferromagnetism - allows each particle one of two spin states: up or down. Meanwhile on the macroscopic level the model typically has a large or infinite number of states; indeed if we consider the Ising model on N particles the macroscopic system has 2^N states (known as *configurations*). The state transition probabilities of any particular node may depend only on the states of nodes in the (local or extended) neighborhood and on parameters governing the microscopic update mechanism. In the zero temperature (noise free) Ising model for example, the selected particle flips to a different spin state with a probability proportional to the fraction of neighbors of the opposite state; this noise free case is sometimes called the Voter model. Adding noise allows us to increase the rate at which particles switch state independently of their neighbors. Depending on the particular graph structure and the update mechanism, various types of phase transitions may occur in the macroscopic properties of these models as parameter values governing microscopic processes are changed.

Recent decades have seen this class of models exapted for the modeling of ecological, evolutionary, epidemiological and sociocultural contact processes [5]. In these contexts each node represents a spatial site or an individual organism or agent, and each link on such a graph represents a social or physical proximity or contact of some sort. In addition to regular lattices (utilized primarily in spatial description) various graph topologies have been proposed to describe other social or biological structures. In these contexts the microscopic update mechanism may be asymmetric with respect to substitution of state; indeed it would be strange if in an epidemiological model the healthy (uninfected) state would also be transmittable to a neighboring individual. In statistical mechanics a variety of analytical results have been established through use of the thermodynamic limit ($N \to \infty$). In contrast, solutions for models on finite and irregular networks have typically relied on numerical simulations and mean field models to analyze such dynamics. Mean field models use so-called master equations to focus on the first moment of the macroscopic probability distribution to simplify the analysis of the dynamics. In order to formulate mean field models, so-called moment closure approximations need to be used to make the number of equations tractable.

1.1 Contact Processes on Adaptive Networks

Most recently, work has been undertaken to study contact processes on adaptive networks [7]. An adaptive network is a network in which in addition to transitions of node states (dynamics *on* the network) one also considers the dynamics of graph topology (dynamics *of* the network). Both these aspects have been investigated

separately in the past decades, but models of the coevolution of network state *and* topology - known as *adaptive networks* - have only been investigated in recent years. The edge rewiring mechanism - like the node update mechanism - can be implemented in a wide variety of ways. The mechanism may reflect some postulated aspect of the physical system to be modeled or may simply be chosen for technically favorable properties; however, as it turns out, different microscopic mechanisms can induce radically divergent macroscopic behavior.

We are interested in investigating adaptive network models of opinion formation, so we will use the corresponding terminology. We will speak of three classes of events that occur in the updating of our networks; first we have *imitation*, in which opinion is transfered from an individual to its neighbor. Second, we consider a process of social segregation as network rewiring; this is the tendency of individuals to break connections with neighbors of a different state and replace them with a new contact. We will refer to this as *segregation* or sometimes simply as rewiring. Finally we have our noise parameter, in which an individual changes state independently of its neighbors. We will refer to this as *innovation*.

We model an adaptive network of N individuals using a graph $G(t) = \{X(t), A(t)\}$ to represent the configuration of our network at time t. We will denote by $B = \{-1, +1\}$ the state space of individual nodes. The random binary vector $X(t) = (x_1, x_2, \ldots x_N) \in \{-1, +1\}^N$ will represent the node configuration; if $x_i(t) = \alpha$ then node i is in state α at time t. Meanwhile $A(t)$ is a symmetric binary random $N \times N$ adjacency matrix (with the diagonal set to 0) describing the edge configuration and hence recording the graph's topology; a_{ij} is 1 if and only if nodes i and j are in contact. Together, $X(t)$ and $A(t)$ allow us to record the state of an adaptive network contact process. In order to focus the investigation, we narrow down on a class of models that fulfill the following five conditions:lfil the following five conditions:

I: The transition probabilities for each node depend only on the configuration of its immediate neighborhood. This means the contact process only occurs between directly adjacent nodes.

II: The processes are modeled by a continuous time Markov process with transition rate matrix Q with non-zero entries only for transitions that change the state of at most one node or edge at a time.

III: We assume conservation of links - i.e. in topological evolution only rewiring is allowed. This is equivalent to keeping the mean degree (the mean number of neighbors) of the graph constant, and is assumed for technical convenience.

IV: When performing numerical simulations updating is asynchronous - meaning only one node or edge is updated at each time. This is believed to yield the optimal approximation of the corresponding continuous time system [5].

V: Binary node state space, with dynamics symmetric under substitution of node state. This assumption is reasonable for modeling a neutral opinion formation model where neither opinion propagates more effectively than the other. Note that this assumption excludes the possibility of modeling epidemics since in epidemiology only the infected state may spread.

It is important to note that most adaptive network models violate some of these properties. Do and Gross [7] characterize a class of models with properties I-IV, but some of them violate condition V. While many models have finite - often binary - opinion sets (often called Voter-like Models), some have a continuous range of opinion states. Some models do not conserve the number of links in the graph, opting instead to eliminate links without replacing them. In this thesis we will be dealing strictly with binary Voter-like models which fulfill conditions I-V, and we will refer to these as *Discordian Adaptive Networks*.

Adaptive network models may differ in the precise details of the updating procedure. While we have restricted our range of models, the update mechanisms remain to be defined. We can view the different possible outcomes of each time step as a possibility tree. Since we are implementing an asynchronous update procedure, each branch must result in either an opinion update, the rewiring of an edge, or no change. The branches springing from the root of the tree (algorithmically the first step) are distinguish the type of event that occurs, with the subsequent steps representing the possible outcomes of the event.

We will distinguish between node events - in which the first step is to randomly pick a node from the network - and edge events - in which the first step to randomly select an edge. This is one of the critical determinants of the structure of the event's branch. Following the selection of a node, one of its neighbors may be selected. If this is the case we will call the first node selected the *ego* and the selected neighbor will be named the *alter*.

The processes of imitation, segregation and innovation may be modeled in different ways. Conceptually innovation is the simplest; this is clearly best modeled a node event. However - to the best of this author's knowledge - existing models of opinion formation have not implemented such a noise parameter. Gross *et al* [2] have investigated the so-called Adaptive SIS model; this is an epidemic model, and in this context the innovation event is a recovery event and therefore has asymmetry with respect to substitution of state. Since we wish to investigate models in which we have symmetry under substitution of state, we must allow for the event to occur in both directions.

With regard to segregation (rewiring) both edge based and node based approaches are conceivable; indeed while Gross *et al* [2] implements segregation events as edge events, Nardini *et al* [1] and Mandrà *et al* [3] both implement rewiring as node events, although they take different approaches in implementation. Meanwhile Schmittmann *et al* [4] opt to implement rewiring as a synchronous procedure and simultaneously rewire all edges.

Given a segregation event implemented as a node event - we may distinguish between two possible implementations. In *egocentric segregation* where the ego keeps the edge while the alter's end is broken and rewired. In *altercentric segregation* on the other hand, the alter is the one maintaining the edge and the rewiring reassigns the ego end of the edge to another node. Both Nardini *et al* and Mandrà *et al* implement egocentric segregation.

Another difference in the segregation mechanism is whether the newly formed connection is always with a node of the same opinion or whether it instead rewires to a randomly selected node. The former is dubbed *optimistic rewiring* while the

latter is called *pessimistic rewiring*. Gross *et al* [2] and Mandrà *et al* [3] implement optimistic rewiring while Nardini *et al* [1] implements a pessimistic scheme.

Finally there are a variety of ways to implement imitation. The comparison of these update mechanisms will be one of our foci in this thesis. Gross *et al* use edge events to model a contact process; a random heterogeneous infected-susceptible edge is chosen to turn into a homogeneous edge infected-infected edge. In that case we have asymmetry with respect to state substitution, but if we would attempt to model state-symmetric imitation as an edge event, its effects will not be visible in the mean field model. All the other opinion formation models previously mentioned use node events to model imitation. Nardini *et al* [1] present two distinct update mechanisms: the direct Voter-like model in which a randomly selected node adopts the opinion of a randomly selected neighbor, and the reverse Voter-like model in which a randomly selected node convinces a randomly selected neighbor of its own opinion. We will refer to the former imitation process as *learning* and latter as *teaching*. Meanwhile - in what might be termed a neighborhood event - Mandrà *et al* implement a so called threshold procedure in which a randomly selected node flips its opinion if and only if the fraction of its neighbors with a different opinion exceeds a certain threshold parameter. Schmittmann *et al* implement a similar procedure called the majority rule, in which the threshold is fixed to half.

1.2 Outline

In this thesis, we will evaluate the performance of mean field description of two types of segregation (egocentric and altercentric) and two types of imitation (learning and teaching) in Discordian Adaptive Networks. In order to do so we will create a set of four models, consisting of one imitation and one segregation process in addition to the innovation process. All of these processes will be implemented as node events. Renormalization of process rates permits us to have two parameters per model without loss of generality.

Following Nardini *et al*, each model will use either learning or teaching. In the models of Nardini *et al*, a random individual (the ego) is chosen to either learn from (direct Voter-like Model) or teach (reverse Voter-like Model) a randomly chosen neighbor. Nardini *et al* use egocentric segregation in their models, and we will investigate this as well. But in addition we will also investigate analogous models which use altercentric segregation instead. Both processes will be implemented as pessimistic rewiring - we reason here that individuals do not know the opinion of the individuals they will connect to *a priori*. With two options for imitation and two options for segregation, we get as mentioned above four distinct models. Since we wish to evaluate the robustness of these models to noise - something not done by Nardini *et al* - we add to all models the innovation process by which nodes randomly and independently switch opinion.

It follows that our egocentric models will be a generalization of Nardini *et al*'s models, with an additional noise parameter. Setting the innovation rate to zero - the model with egocentric segregation and learning will be equivalent to the reverse Voter-like Model when innovation is set to zero. Meanwhile, the model with egocentric segregation and teaching becomes equivalent to the direct Voter-like Model

when innovation is omitted.

In section 2 we will introduce the models of Nardini *et al* as examples of previously published discordian adaptive networks, and derive from master equations a set of ordinary differential equations describing the models at the level of pairs. In section 3 we will introduce our moment closure approximation. In section 4 we will derive our models using master equations and close the equations using a moment closure approximation. In section 5 we will perform numerical bifurcation analysis of our models. In section 6 we will present stochastic simulation results, discussing and comparing the models and simulations in section 7. Our conclusion will be presented in section 8.

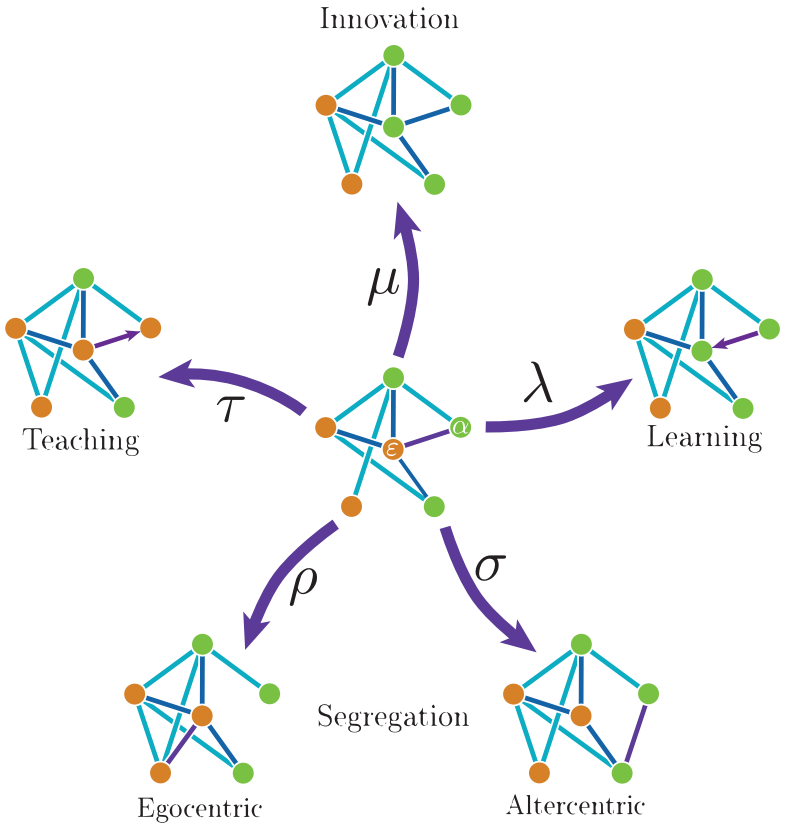


Figure 1: The five different processes that occur in our models. In the center is the configuration before updating, with the node marked with ε being the ego and the node marked with α being the alter.

1.3 Additional Notation

We will follow notation similar to Nardini *et al* [1]. We consider a population $V = \{1, \ldots, N\}$ of N individuals as nodes in a simple graph $G(t) = (X(t), A(t))$ with

$1 \leq L \leq N(N-1)/2$ undirected edges indexed in set $E = \{1, \ldots, L\}$ representing social interactions. Individuals may hold opinions from the set $B = \{-, +\} = \{-1, +1\}$ and we will denote by $n_+(t)$ and $n_-(t)$ the frequency of individuals with opinion $+$ and $-$ respectively. We will use α, β and γ to denote arbitrary opinions (i.e. elements of B), and $\bar{\alpha}$ to denote the opinion opposite to α. By h,i and $j \in V$ we will denote selected individuals, and by $e \in E$ we will denote a selected edge. Given an edge e connecting α and β nodes, e_α and e_β will denote respectively the node with state α and the node with state β. We denote by $x_i = \alpha$ the state of node i and the state of edge e as $y_e = \alpha\beta$. Note that here and elsewhere we suppress the time dependence for aesthetic purposes.

By an $\alpha\beta\gamma$ triplet we mean connected chain of three distinct individuals with the central individual having state β while the marginal nodes have states α and γ. It is critical to be consistent about how we count edges and triplets. While edges are undirected, in counting we distinguish between an $\alpha\beta$ edge and a $\beta\alpha$ edge, and between an $\alpha\beta\gamma$ triplet and a $\gamma\beta\alpha$ triplet. This means that we will often count $\alpha\alpha$ pairs twice. We will denote by $l_{\alpha\beta}(t) = l_{\beta\alpha}(t)$ the per capita frequencies of edges connecting pairs of individuals with α and β opinions. Finally by $m_{\alpha\beta\gamma}(t)$ we will denote the per capita frequency of $\alpha\beta\gamma$ triplets; thus three individuals connected in a triangle will be counted as six distinct triplets. We denote by $k = \frac{2L}{N}$ the mean degree of the graph, which will remain constant due to property III. We will refer to links between nodes of differing opinions as *discordant links* and similarly links between nodes in agreement will be called *concordant links*.

We will use $K(i)$ to denote the degree of node i and $K_\alpha(i)$ to denote the number of neighbors of individual i which are in state α. Let k_α represent the mean degree of α nodes, $k_{\alpha|\beta}$ represent the average number of α neighbors of a β node and by $k_{\alpha|\beta\gamma}$ denote the expected number of α-individuals neighboring the β in a $\beta\gamma$ pair. Then we have, for $n_\alpha > 0$, $n_\beta > 0$ and $l_{\alpha\beta} > 0$ respectively:

$$k_\alpha = \frac{1}{n_\alpha} \sum_{i:x_i=\alpha} K(i) = \frac{l_{\alpha\bar{\alpha}} + 2l_{\alpha\alpha}}{n_\alpha} \tag{1}$$

$$k_{\alpha|\beta} = \frac{1}{n_\beta} \sum_{i:x_i=\beta} K_\alpha(i) = \frac{(1+\delta_{\alpha\beta})\, l_{\alpha\beta}}{n_\beta} \tag{2}$$

$$k_{\alpha|\beta\gamma} = \frac{1}{l_{\beta\gamma}} \sum_{e:y_e=\beta\gamma} K_\alpha(e_\beta) = \frac{m_{\alpha\beta\gamma}}{l_{\beta\gamma}} + \delta_{\alpha\gamma} \tag{3}$$

The $\delta_{\alpha\gamma}$ is a Dirac delta and counts the γ in the focal $\beta\gamma$ pair if it is equal to α. We must count this γ separately; this is not accounted for in the first term since a triplet must consist of three distinct individuals. For $n_\alpha = 0$, $n_\beta = 0$ and $l_{\alpha\beta} = 0$ we set $k_\alpha = 0$, $k_{\alpha|\beta} = 0$ and $k_{\alpha|\beta\gamma} = 0$ respectively. We will denote by $k_{\beta\gamma} = \sum_\alpha k_{\alpha|\beta\gamma}$ the expected number of neighbors for a β individual in a $\beta\gamma$ pair and by $q_{\alpha|\beta} = k_{\alpha|\beta}/k_\beta$ the expected fraction of α neighbors of a β node; for convenience this will also be equated to 0 when $k_\beta = 0$.

By μ we will denote the rate of innovation, by λ the rate of learning, by τ the rate of teaching, by ρ the rate of egocentric segregation and by σ the rate of altercentric segregation. To denote various events on nodes and edges we will use a

superscripted arrow notation: $(\alpha) \xrightarrow{\mu} (\bar{\alpha})$, $(\alpha\beta) \xrightarrow{\lambda} (\beta\beta)$, $(\alpha\beta) \xrightarrow{\tau} (\alpha\alpha)$, $(\alpha\bar{\alpha}) \xrightarrow{\rho} (\alpha\beta)$ and $(\bar{\alpha}\alpha) \xrightarrow{\sigma} (\beta\alpha)$ will represent innovation, learning, teaching, egocentric-, and altercentric-segregation events respectively. Note how we use to first element of a pair $(\alpha\beta)$ to represent the ego, and the second to represent the alter; in the learning event the ego changes opinion while in the teaching event the alter changes opinion.

Since we also know that $n_+ + n_- = 1$ and $\frac{k}{2} = l_{++} + l_{+-} + l_{--}$, it suffices to consider the variables n_+, l_{+-} and l_{++} to provide a complete description of the system on the level of pairs. As we will see, the differential equations for the last two variables will depend on triplet variables, which is why a moment closure approximation is necessary to close the system of equations.

2 Existing models

In this section we will explore several relevant recently published models which fulfill properties I-V. As mentioned previously, these models display some qualitative similarities in their microscopic update mechanism, but several key statistical differences between them will reveal qualitatively different macroscopic behavior.

2.1 The Direct and Reverse Voter-like Models

Nardini *et al* [1] proposed a pair of models of opinion formation, in which only imitation (λ or τ) and egocentric segregation (ρ) events are possible, and no innovation events are considered ($\mu = 0$). The models implement the following update procedure:

1. A random individual i - the ego - is selected.
2. One of its neighbors j - the alter - is selected randomly.
3. With probability ϕ we get a ρ-event: if $x_i \neq x_j$, individual i breaks its connection with j and forms a new connection with a randomly selected individual, with double and self edges not allowed.
4. With probability $1 - \phi$ an imitation event takes place. This is where the two models diverge:
   a. In the direct Voter-like Model (dVM) we get a learning event: agent i adopts the opinion of agent j.
   b. Meanwhile in the reverse Voter-Model (rVM), we get a teaching event: agent i convinces agent j of their opinion.

2.1.1 Derivation of Master Equations for the dVM

Supposing we investigate an adaptive network with configuration $G(t)$, and we wish to study the aggregate effect of microscopic events on a macroscopic variable that is a function $f(G)$ of the configuration. Assuming events occur asynchronously, we may use a so-called master equation to describe the dynamics [11]:

$$\frac{df}{dt} = \sum_{\text{events } \varepsilon} r_G(\varepsilon)\Delta f(\varepsilon) \tag{4}$$

where $r_\sigma(\varepsilon)$ is the rate at which event ε occurs under configuration G and $\Delta f(\varepsilon)$ is the impact of event ε on function f. The sum therefore adds the impact of all events on the function, weighted according to their respective rates.

In the case of the direct Voter-like Model described by Nardini *et al* [1] we wish to derive master equations for the mean-field variables n_+, l_{+-} and l_{++}. We must consider 4 types of events:

ε	$r_G(\varepsilon)$	$\Delta n_+(\varepsilon)$	$\Delta l_{+-}(\varepsilon)$	$\Delta l_{++}(\varepsilon)$
$(-+) \xrightarrow{\lambda} (++)$	$(1-\phi)\cdot n_- \cdot q_{+\|-}$	$+\frac{1}{N}$	$\frac{k_{-\|-+}-k_{+\|-+}}{N}$	$\frac{+k_{+\|-+}}{N}$
$(+-) \xrightarrow{\lambda} (--)$	$(1-\phi)\cdot n_+ \cdot q_{-\|+}$	$-\frac{1}{N}$	$\frac{k_{+\|+-}-k_{-\|+-}}{N}$	$\frac{-k_{+\|+-}}{N}$
$(-+) \xrightarrow{\rho} (--)$	$\phi \cdot n_- \cdot q_{+\|-} \cdot n_-$	0	$-\frac{1}{N}$	0
$(+-) \xrightarrow{\rho} (++)$	$\phi \cdot n_+ \cdot q_{-\|+} \cdot n_+$	0	$-\frac{1}{N}$	$\frac{+1}{N}$

Hence we can derive three master equations, noting that $n_\alpha q_{\bar{\alpha}|\alpha} = l_{\alpha\bar{\alpha}}/k_\alpha$:

$$\dot{n}_+ = (1-\phi)\frac{l_{+-}}{N}\left[\frac{1}{k_-} - \frac{1}{k_+}\right] \tag{5}$$

$$\begin{aligned}\dot{l}_{+-} =& (1-\phi)\frac{1}{N}\left[n_+ q_{-|+}\cdot\left(k_{+|+-}-k_{-|+-}\right) + n_- q_{+|-}\left(k_{-|-+}-k_{+|-+}\right)\right] \\ & - \phi\cdot\frac{1}{N}\left[n_- q_{+|-}\cdot n_- + n_+ q_{-|+}\cdot n_+\right] \\ =& (1-\phi)\frac{l_{+-}}{N}\left[\frac{k_{+|+-}-k_{-|+-}}{k_+} - \frac{k_{+|-+}-k_{-|-+}}{k_-}\right] - \phi\frac{l_{+-}}{N}\left[\frac{n_-}{k_-}+\frac{n_+}{k_+}\right]\end{aligned} \tag{6}$$

$$\dot{l}_{++} = (1-\phi)\frac{l_{+-}}{N}\left[\frac{k_{+|-+}}{k_-} - \frac{k_{+|+-}}{k_+}\right] + \phi\frac{l_{+-}}{N}\frac{n_+}{k_+} \tag{7}$$

The astute reader will have noticed the occurrence of triplet terms $k_{h|ij}$ in these equations. These depend on the (time dependent) number of triplets m_{hij}; indeed the equations for each network moment depend on a higher moment, with the exception of the final moment. In order to circumvent this issue we will be using a so-called moment closure approximation (MCA), where the triplet terms will be approximated in terms of lower order moments. Indeed we will be comparing several such moment closures to see which better approximate numerical simulations of the exact stochastic model. We will elaborate on moment closure approximations in the next chapter.

2.1.2 Derivation of Master Equations for the rVM

The situation for the Reverse Voter-like Model is almost identical to the Direct Voter-like Model, except the rates for the two learning events are swapped:

ε	$r_G(\varepsilon)$	$\Delta n_+(\varepsilon)$	$\Delta l_{+-}(\varepsilon)$	$\Delta l_{++}(\varepsilon)$
$(-+) \xrightarrow{\tau} (--)$	$(1-\phi)\cdot n_- \cdot q_{+\|-}$	$-\frac{1}{N}$	$\frac{k_{+\|+-}-k_{-\|+-}}{N}$	$-\frac{k_{+\|+-}}{N}$
$(+-) \xrightarrow{\tau} (++)$	$(1-\phi)\cdot n_+ \cdot q_{-\|+}$	$+\frac{1}{N}$	$\frac{k_{-\|-+}-k_{+\|-+}}{N}$	$+\frac{k_{+\|-+}}{N}$
$(-+) \xrightarrow{\rho} (--)$	$\phi \cdot n_- \cdot q_{+\|-} \cdot n_-$	0	$-\frac{1}{N}$	0
$(+-) \xrightarrow{\rho} (++)$	$\phi \cdot n_+ \cdot q_{-\|+} \cdot n_+$	0	$-\frac{1}{N}$	$+\frac{1}{N}$

Its quite clear that the equation for the first network moment will simply be identical to that of the dVM but with the rates of the two asocial learning events swapped. For the first moment equation this results in a complete reversal of the dynamics:

$$\dot{n}_+ = (1-\phi) l_{+-} \left[\frac{1}{k_+} - \frac{1}{k_-} \right] \tag{8}$$

Similarly we swap the rates of the two imitation events in the second moment equations; in this case however the occurrence of rewiring terms means this is not a simple reversal of the dVM dynamics:

$$\dot{l}_{+-} = (1-\phi) l_{+-} \left[\frac{k_{+|+-} - k_{-|+-}}{k_-} - \frac{k_{+|-+} - k_{-|-+}}{k_+} \right] - \phi \cdot l_{+-} \left[\frac{n_-}{k_-} + \frac{n_+}{k_+} \right] \tag{9}$$

$$\dot{l}_{++} = (1-\phi) \frac{l_{+-}}{N} \left[\frac{k_{+|-+}}{k_+} - \frac{k_{+|+-}}{k_-} \right] + \phi \frac{l_{+-}}{N} \frac{n_+}{k_+} \tag{10}$$

3 Moment Closure Approximations

Unfortunately, the equations describing dynamics at the level of pairs depends on triplet terms ($k_{\alpha|\beta\gamma}$ depend on $m_{\alpha\beta\gamma}$). Attempting to remedy the situation by formulating equations for the triplet level is futile; it is known that the equations describing dynamics of each network moment depend on the next network moment. Hence we are forced to approximate the triplet terms in terms of singlets and pairs. This approximation procedure is known as *moment closure* and allows us to close the system of differential equations and subsequently solve them. A moment closure approximation on the level of pairs is also known as a pair approximation.

3.1 Poisson Pair Approximation

Rand [11] mentions two well known pair approximations; the first - utilizing Poisson statistics - is in good approximation when one assumes $K_\alpha(i)$ is Poisson distributed over β sites with mean $k_{\alpha|\beta}$:

$$m_{\alpha\beta\gamma} \approx \hat{m}_{\alpha\beta\gamma} = \frac{l_{\alpha\beta} l_{\beta\gamma}}{n_\beta} \tag{11}$$

From this we derive:

$$\hat{k}_{\alpha|\beta\gamma} = \frac{\hat{m}_{\alpha\beta\gamma}}{l_{\beta\gamma}} + \delta_{\alpha\gamma} = \frac{l_{\alpha\beta}}{n_\beta} + \delta_{\alpha\gamma} \tag{12}$$

$$\hat{k}_{\beta\gamma} = \sum_\alpha \hat{k}_{\alpha|\beta\gamma} = 1 + \sum_\alpha \frac{l_{\alpha\beta}}{n_\beta} = 1 + k_\beta - \tfrac{1}{2}k_{\beta|\beta} \tag{13}$$

Rand has shown that the Poisson pair approximation functions well in some adaptive networks; this has also been confirmed by Gross *et al* [2] for the Adaptive SIS model.

3.2 Bernoulli Pair Approximation

The second approximation functions well under Bernoulli trial statistics; the number of α neighbors of a β node at i is chosen from k independent repeated trials with success probability $k_{\alpha|\beta}/k$, generating a similar Pair Approximation:

$$m_{\alpha\beta\gamma} \approx \check{m}_{\alpha\beta\gamma} = \kappa \frac{l_{\alpha\beta} l_{\beta\gamma}}{n_\beta} \tag{14}$$

where $\kappa = (k-1)/k$, a constant for the models considered here. We note that $\lim_{k\to\infty} \check{m}_{\alpha\beta\gamma} = \hat{m}_{\alpha\beta\gamma}$; for networks of high degree the Bernoulli and Poisson Pair Approximations are similar.

Rand has shown that the Bernoulli pair approximation seems to function well for a range of (static) regular lattices and random networks with a globally two-dimensional structure (typically created by randomly connecting nodes to neighbors within a certain distance).

4 Discordian Adaptive Networks

In this section we will be deriving a class of models incorporating the processes of innovation, imitation and segregation. We will begin by deriving the mean field equations with all five processes, and after moment closure we will reduce them later to specific models with two parameters (and three processes) each.

4.1 Mean-Field Equations

ε	$r_G(\varepsilon)$	$\Delta n_+(\varepsilon)$	$\Delta l_{+-}(\varepsilon)$	$\Delta l_{++}(\varepsilon)$
$(-) \xrightarrow{\mu} (+)$	μn_-	$+\frac{1}{N}$	$\frac{1}{N}\left(k_{-\|-} - k_{+\|-}\right)$	$+\frac{k_{+\|-}}{N}$
$(+) \xrightarrow{\mu} (-)$	μn_+	$-\frac{1}{N}$	$\frac{1}{N}\left(k_{+\|+} - k_{-\|+}\right)$	$-\frac{k_{+\|+}}{N}$
$(-+) \xrightarrow{\lambda} (++)$	$\lambda \cdot n_- \cdot q_{+\|-}$	$+\frac{1}{N}$	$\frac{1}{N}\left(k_{-\|-+} - k_{+\|-+}\right)$	$+\frac{k_{+\|-+}}{N}$
$(+-) \xrightarrow{\lambda} (--)$	$\lambda \cdot n_+ \cdot q_{-\|+}$	$-\frac{1}{N}$	$\frac{1}{N}\left(k_{+\|+-} - k_{-\|+-}\right)$	$-\frac{k_{+\|+-}}{N}$
$(-+) \xrightarrow{\tau} (--)$	$\tau \cdot n_- \cdot q_{+\|-}$	$-\frac{1}{N}$	$\frac{1}{N}\left(k_{+\|+-} - k_{-\|+-}\right)$	$-\frac{k_{+\|+-}}{N}$
$(+-) \xrightarrow{\tau} (++)$	$\tau \cdot n_+ \cdot q_{-\|+}$	$+\frac{1}{N}$	$\frac{1}{N}\left(k_{-\|-+} - k_{+\|-+}\right)$	$+\frac{k_{+\|-+}}{N}$
$(-+) \xrightarrow{\rho} (--)$	$\rho \cdot n_- \cdot q_{+\|-} \cdot n_-$	0	$-\frac{1}{N}$	0
$(+-) \xrightarrow{\rho} (++)$	$\rho \cdot n_+ \cdot q_{-\|+} \cdot n_+$	0	$-\frac{1}{N}$	$+\frac{1}{N}$
$(-+) \xrightarrow{\sigma} (++)$	$\sigma \cdot n_- \cdot q_{+\|-} \cdot n_+$	0	$-\frac{1}{N}$	$+\frac{1}{N}$
$(+-) \xrightarrow{\sigma} (--)$	$\sigma \cdot n_+ \cdot q_{-\|+} \cdot n_-$	0	$-\frac{1}{N}$	0

For $0 < n_+ < 1$ we can use $n_\alpha q_{\bar{\alpha}|\alpha} k_\alpha = l_{+-}$ to derive the Master Equations:

$$\dot{n}_+ = (\lambda - \tau)\frac{l_{+-}}{N}\left\{\frac{1}{k_-} - \frac{1}{k_+}\right\} + \mu\frac{1}{N}\left\{n_- - n_+\right\} \tag{15}$$

$$\begin{aligned}\dot{l}_{+-} =& \lambda\frac{l_{+-}}{N}\left\{\frac{k_{+|+-} - k_{-|+-}}{k_+} - \frac{k_{+|-+} - k_{-|-+}}{k_-}\right\} - \rho\frac{l_{+-}}{N}\left\{\frac{n_-}{k_-} + \frac{n_+}{k_+}\right\} \\ &+\tau\frac{l_{+-}}{N}\left\{\frac{k_{+|+-} - k_{-|+-}}{k_-} - \frac{k_{+|-+} - k_{-|-+}}{k_+}\right\} - \sigma\frac{l_{+-}}{N}\left\{\frac{n_+}{k_-} + \frac{n_-}{k_+}\right\} \\ &+\mu\frac{1}{N}\left\{n_-\left(k_{-|-} - k_{+|-}\right) + n_+\left(k_{+|+} - k_{-|+}\right)\right\}\end{aligned} \tag{16}$$

$$\begin{aligned}\dot{l}_{++} =& \lambda\frac{l_{+-}}{N}\left\{\frac{k_{+|-+}}{k_-} - \frac{k_{+|+-}}{k_+}\right\} + \rho\frac{l_{+-}}{N}\frac{n_+}{k_+} + \mu\left\{k_{+|-}n_- - k_{+|+}n_+\right\} \\ &+\tau\frac{l_{+-}}{N}\left\{\frac{k_{+|-+}}{k_+} - \frac{k_{+|+-}}{k_-}\right\} + \sigma\frac{l_{+-}}{N}\frac{n_+}{k_-}\end{aligned} \tag{17}$$

For $n_+ = 0$ or $n_+ = 1$ on the other hand - all the non-μ terms are set to zero.

4.2 Moment Closure

Here we derive the closed mean-field equations for the generalized DAN model. We will be using the Poisson pair approximation outlined in equations 12 and 13. This is chosen because it is simple and has been demonstrated to work for a similar adaptive network model by Gross *et al* [2]. In order to simplify notation and distinguish the variables in the approximated system from those in exact system, we introduce the following notation: the approximated version of n_+ will be denoted by z, the approximated version of l_{+-} will be denoted by d and the approximated version of l_{++} will be denoted by c. We take this chance to translate other variables using our

definitions from section 1.3:

$$\begin{aligned} n_- =& 1-z \\ l_{--} =& \tfrac{k}{2}-d-c \\ k_- =& (k-d-2c)/(1-z) \\ k_+ =& (d+2c)/z \end{aligned}$$

Thus are derived the moment closed equations for $0 < z < 1$:

$$\dot{z} = (\lambda-\tau)\frac{d}{N}\left\{\frac{1-z}{k-d-2c}-\frac{z}{d+2c}\right\}+\mu\frac{1}{N}\{1-2z\} \tag{18}$$

$$\begin{aligned} \dot{d} =& \lambda\frac{d}{N}\left\{\frac{c-d-z}{d+2c}-\frac{4d-k+2c+2(1-z)}{2(k-d-2c)}\right\} \\ &+\tau\frac{d}{N}\left\{\frac{[c-d-z](1-z)}{z(k-d-2c)}-\frac{[4d-k+2c+2(1-z)]z}{2(1-z)(d+2c)}\right\} \\ &-\rho\frac{d}{N}\left\{\frac{(1-z)^2}{k-d-2c}+\frac{z^2}{d+2c}\right\}-\sigma\frac{d}{N}\left\{\frac{z(1-z)}{k-d-2c}+\frac{z(1-z)}{d+2c}\right\}+\mu\frac{1}{N}\{k-4d\} \end{aligned} \tag{19}$$

$$\begin{aligned} \dot{c} =& \lambda\frac{d}{N}\left\{\frac{d+1-z}{k-d-2c}-\frac{c}{d+2c}\right\}+\tau\frac{d}{N}\left\{\frac{z(d+1-z)}{(1-z)(d+2c)}-\frac{(1-z)c}{z(k-d-2c)}\right\} \\ &+\rho\frac{d}{N}\cdot\frac{z^2}{d+2c} \quad +\sigma\frac{d}{N}\cdot\frac{z(1-z)}{k-d-2c} \quad +\mu\frac{1}{N}\{d-2c\} \end{aligned} \tag{20}$$

with the non-μ terms once again set to zero for $z = 0$ and $z = 1$.

4.3 Two-Parameter Models

The parameter space of the equations above is rather expansive, so we now wish to reduce the number of parameters. Normalization allows us to lose one parameter for free, but we wish to have only two parameters per model models. Each model will have a single imitation process (either learning or teaching) and a single segregation process (either altercentric or egocentric), leading us to the following generalized parameter scheme:

I. A random individual is selected. With probability μ an innovation event occurs.

II. Otherwise (with probability $1-\mu$), one of its neighbors is selected randomly and:

   1. With probability ϕ we get get a segregation event.
   2. With probability $1-\phi$ we get an imitation event.

Thus - for picking one imitation process and one segregation process, four models emerge, which we name based on the imitation and segregation process they implement:

	Learning (λ)	Teaching (τ)
Egocentric Segregation (ρ)	$\lambda\rho$-DAN	$\tau\rho$-DAN
Altercentric Segregation (σ)	$\lambda\sigma$-DAN	$\tau\sigma$-DAN

4.3.1 $\lambda\rho$-DAN

The $\lambda\rho$-DAN model is just a version of the dVM equations with an added noise parameter. The equations for $0 < z < 1$ read:

$$\dot{z} = (1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{1-z}{k-d-2c} - \frac{z}{d+2c}\right\} + \mu\frac{1}{N}\{1-2z\} \tag{21}$$

$$\begin{aligned}\dot{d} = &(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{c-d-z}{d+2c} - \frac{4d-k+2c+2(1-z)}{2(k-d-2c)}\right\} \\ &-(1-\mu)\phi\frac{d}{N}\left\{\frac{(1-z)^2}{k-d-2c} + \frac{z^2}{d+2c}\right\} + \mu\frac{1}{N}\{k-4d\}\end{aligned} \tag{22}$$

$$\begin{aligned}\dot{c} = &(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{d+1-z}{k-d-2c} - \frac{c}{d+2c}\right\} \\ &+(1-\mu)\phi\frac{d}{N}\cdot\frac{z^2}{d+2c} \quad +\mu\frac{1}{N}\{d-2c\}\end{aligned} \tag{23}$$

4.3.2 $\tau\rho$-DAN

Likewise, the $\tau\rho$-DAN model is a version of the rVM equations with a noise parameter. The equations for $0 < z < 1$ read:

$$\dot{z} = -(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{1-z}{k-d-2c} - \frac{z}{d+2c}\right\} + \mu\frac{1}{N}\{1-2z\} \tag{24}$$

$$\begin{aligned}\dot{d} = &(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{[c-d-z](1-z)}{z(k-d-2c)} - \frac{[4d-k+2c+2(1-z)]z}{2(1-z)(d+2c)}\right\} \\ &-(1-\mu)\phi\frac{d}{N}\left\{\frac{(1-z)^2}{k-d-2c} + \frac{z^2}{d+2c}\right\} + \mu\frac{1}{N}\{k-4d\}\end{aligned}$$

$$\begin{aligned}\dot{c} = &(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{z(d+1-z)}{(1-z)(d+2c)} - \frac{(1-z)c}{z(k-d-2c)}\right\} \\ &+(1-\mu)\phi\frac{d}{N}\cdot\frac{z^2}{d+2c} \quad +\mu\frac{1}{N}\{d-2c\}\end{aligned} \tag{25}$$

4.3.3 $\lambda\sigma$-DAN

Meanwhile, the $\lambda\sigma$-DAN model exhibits different rewiring terms. The equations for $0 < z < 1$ read:

$$\dot{z} = (1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{1-z}{k-d-2c} - \frac{z}{d+2c}\right\} + \mu\frac{1}{N}\{1-2z\} \tag{26}$$

$$\begin{aligned}\dot{d} = &(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{c-d-z}{d+2c} - \frac{4d-k+2c+2(1-z)}{2(k-d-2c)}\right\} \\ &-(1-\mu)\phi\frac{d}{N}\left\{\frac{z(1-z)}{k-d-2c} + \frac{z(1-z)}{d+2c}\right\} + \mu\frac{1}{N}\{k-4d\}\end{aligned} \tag{27}$$

$$\begin{aligned}\dot{c} = &(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{d+1-z}{k-d-2c} - \frac{c}{d+2c}\right\} \\ &+(1-\mu)\phi\frac{d}{N}\cdot\frac{z(1-z)}{k-d-2c} + \mu\frac{1}{N}\{d-2c\}\end{aligned} \tag{28}$$

4.3.4 $\tau\sigma$-DAN

Similarly for the $\tau\sigma$-DAN model, the equations for $0 < z < 1$ read:

$$\dot{z} = -(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{1-z}{k-d-2c} - \frac{z}{d+2c}\right\} + \mu\frac{1}{N}\{1-2z\} \tag{29}$$

$$\begin{aligned}\dot{d} = &+(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{[c-d-z](1-z)}{z(k-d-2c)} - \frac{[4d-k+2c+2(1-z)]z}{2(1-z)(d+2c)}\right\} \\ &-(1-\mu)\phi\frac{d}{N}\left\{\frac{z(1-z)}{k-d-2c} + \frac{z(1-z)}{d+2c}\right\} + \mu\frac{1}{N}\{k-4d\}\end{aligned}$$

$$\begin{aligned}\dot{c} = &(1-\mu)(1-\phi)\frac{d}{N}\left\{\frac{z(d+1-z)}{(1-z)(d+2c)} - \frac{(1-z)c}{z(k-d-2c)}\right\} \\ &+(1-\mu)\phi\frac{d}{N}\cdot\frac{z(1-z)}{k-d-2c} + \mu\frac{1}{N}\{d-2c\}\end{aligned} \tag{30}$$

4.4 Combinatorial Constraints

4.4.1 Initial conditions

While theoretically a wide range of initial conditions are possible for the mean field model, we will constrain our initial conditions to be consistent with values expected from an initial Erdős-Rényi network configuration (sometimes denoted in the literature as $G(n, M)$ with n the number of nodes and M the number of edges). The network is configured as follows: given an initial fraction of + individuals z_0, the L edges in our network are assigned randomly. While we may investigate different values of z_0, this particular process of assigning edges constrains d_0 and c_0 - the initial values of d and c; the fraction of $(+-)$ or $(++)$ links in the initial network will be

proportional to the fraction we would find in the complete graph with a fraction z_0 of its nodes in a + state, hence:

$$d_0 = \frac{L}{N} \cdot \frac{Nz_0 \cdot N(1-z_0)}{N(N-1)/2} = \frac{Nz_0(1-z_0)}{N-1} \cdot k \tag{31}$$

$$c_0 = \frac{L}{N} \cdot \frac{Nz_0 \cdot (Nz_0-1)/2}{N(N-1)/2} = \frac{z_0(Nz_0-1)}{2(N-1)} \cdot k \tag{32}$$

4.4.2 Bounds on pair terms

The structure of the discrete stochastic network model leads us to expect certain boundaries, which may or may not be respected by the corresponding mean-field equations. There are several constraints to consider, but since it is not immediately obvious which of these trump which, we will list them all. First, denoting l_{--} with h, we have $d + c + h = \frac{k}{2}$, which implies the constraints $0 \leq d, c \leq \frac{k}{2} = \frac{L}{N}$. Second, for given values of z, the number of possible links of a certain type is combinatorially constrained:

$$d \leq \hat{d}_c = \frac{NzN(1-z)}{N} = Nz(1-z) \tag{33}$$

$$c \leq \hat{c}_c = \frac{Nz(Nz-1)}{N} = z(Nz-1) \tag{34}$$

$$h \leq \hat{h}_c = \frac{N(1-z)(N(1-z)-1)}{N} = (1-z)(N(1-z)-1) \tag{35}$$

Finally, it could be that the network is so saturated with edges that there exists a minimum bound on c, d and h. This occurs when L exceeds the maximum constraints outlined above for the other two types of edges:

$$d \geq \tilde{d}_c = \max\{0, \frac{L}{N} - \hat{c}_c - \hat{h}_c\} \tag{36}$$

$$c \geq \tilde{c}_c = \max\{0, \frac{L}{N} - \hat{d}_c - \hat{h}_c\} \tag{37}$$

$$h \geq \tilde{h}_c = \max\{0, \frac{L}{N} - \hat{c}_c - \hat{d}_c\} \tag{38}$$

5 Analysis of the Mean-Field Equations

In this section we will present the bifurcation analysis of our models. Only a few things can be said analytically about our equations, so for the most part the analysis is done numerically using *AUTO 07p* [9] - a software package written in FORTRAN for continuation and bifurcation problems in ordinary differential equations. To simplify the analysis we will investigate only the case of $k = 5$ and use initial conditions consistent with equations 31-32 (and thus be consistent with the initial configuration of the corresponding stochastic simulations).

5.1 Analytical Results

All four models take a similar form:

$$\dot{z} = (1-\mu)(1-\phi)\frac{d}{N}I_1(k,z,d,c) + \mu\frac{1}{N}\{1-2z\} \tag{39}$$

$$\dot{d} = (1-\mu)(1-\phi)\frac{d}{N}I_2(k,z,d,c) - (1-\mu)\phi\frac{d}{N}S_2(k,z,d,c) + \mu\frac{1}{N}\{k-4d\} \tag{40}$$

$$\dot{c} = (1-\mu)(1-\phi)\frac{d}{N}I_3(k,z,d,c) + (1-\mu)\phi\frac{d}{N}S_3(k,z,d,c) + \mu\frac{1}{N}\{d-2c\} \tag{41}$$

Where I_j are imitation terms, and S_j are segregation terms. We note that all terms divide by N, and that this is the only place where N occurs; thus changing N is equivalent to rescaling time, and N cannot change any qualitative features of the field in the limit or perturb equilibria.

When we set $\mu = 0$ it is clear that since d is a coefficient of all non-μ terms, that $d = 0$ is an equilibrium for all values of z and c. This is clearly also the case in the stochastic model, where all processes except innovation require a discordant $(+-)$ edge to occur. Thus the manifold defined by $d = 0$ consists of degenerate equilibria (and hence not Lyapunov stable). We will omit this invariant manifold from all our diagrams. On another extreme, we have the pure noise case with $\mu = 1$. All models become identical at this parameter value, and present $z = \frac{1}{2}$, $d = \frac{k}{4}$ and $c = \frac{k}{8}$ as the only stable equilibrium for all ϕ values.

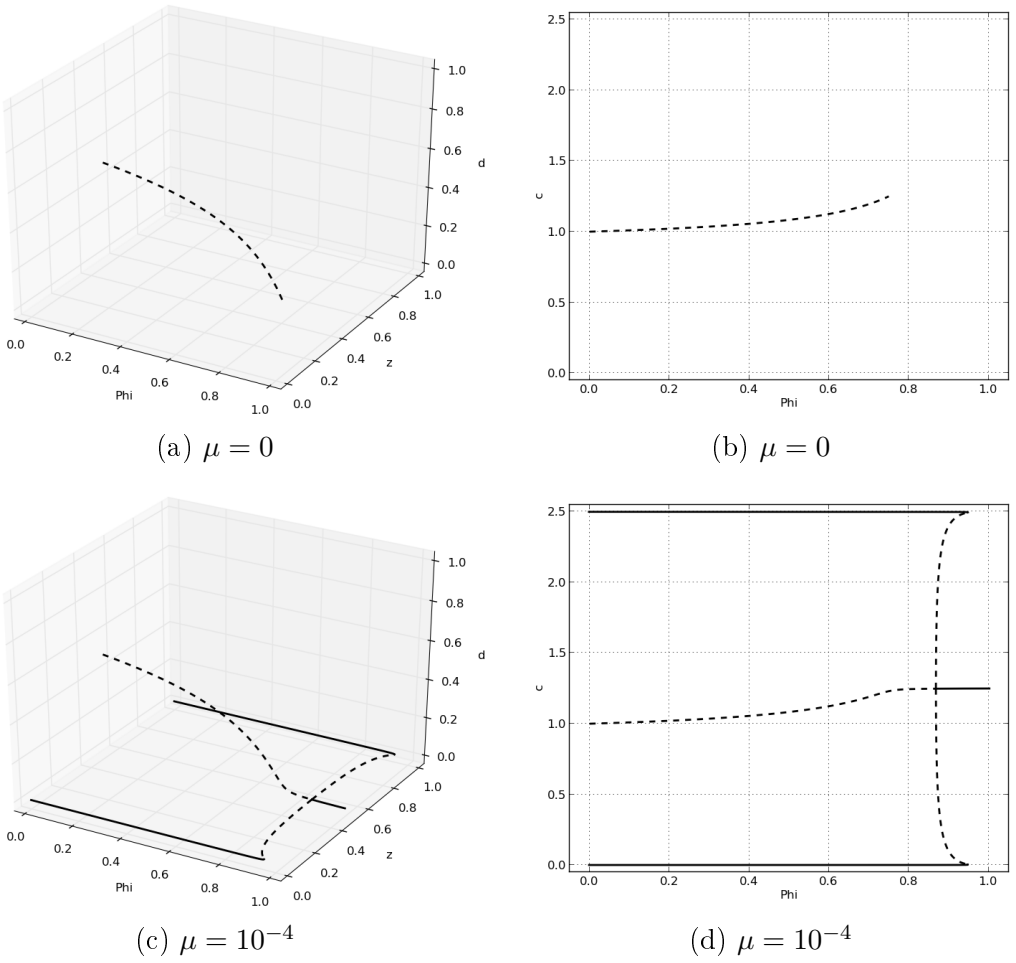
5.2 Numerical Results

5.2.1 $\lambda\rho$-DAN

In the $\lambda\rho$-DAN model consists of the learning, egocentric segregation, and innovation processes. We recall that egocentric segregation only directly acts on the second network moments, and does not affect a direct field on the first network moment z. Meanwhile - learning is the only process affect a z-field pushing away from $z = 0.5$, while innovation affects a field pushing towards it or average.

Prima facie - from an understanding of its stochastic version - we would expect the $\lambda\rho$-DAN model to exhibit stable outer equilibria at $\{z = 1, d = 0, c = 2.5\}$ and $\{z = 0, d = 0, c = 0\}$ when we have no innovation ($\mu = 0$) and low ϕ. However - since the mean field model exhibits a degenerate invariant manifold at $d = 0$, these equilibria are embedded in this manifold and become destabilized, and only a central saddle point at positive d remains, as we see in figures 2c and 2d. For low ϕ the invariant manifold at $d = 0$ is repulsive, with trajectories close to the manifold generally converging to the outer equilibria. As ϕ is increased, the increased intensity of segregation pushes the unstable equilibrium down into the manifold, and at a critical $\phi_c = 0.75$ the saddle point is absorbed into the manifold which subsequently becomes attractive. This corresponds to a phase transition in the stochastic model from a consensus phase where learning dominates and one opinion takes over the

whole network, and a so-called frozen state where egocentric segregation dominates and the network fragments into components which are piece-wise in consensus [1].



(a) $\mu = 0$ (b) $\mu = 0$

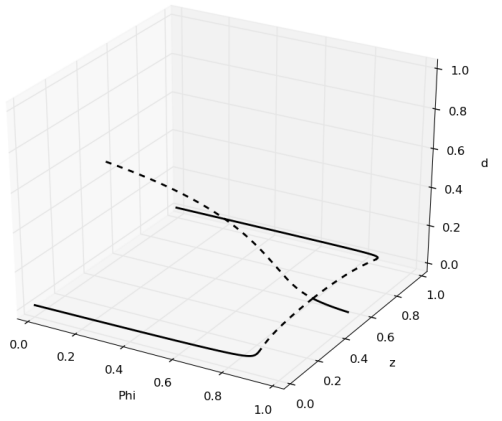
(c) $\mu = 10^{-4}$ (d) $\mu = 10^{-4}$

Figure 2: Bifurcation diagrams for the $\lambda\rho$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria.

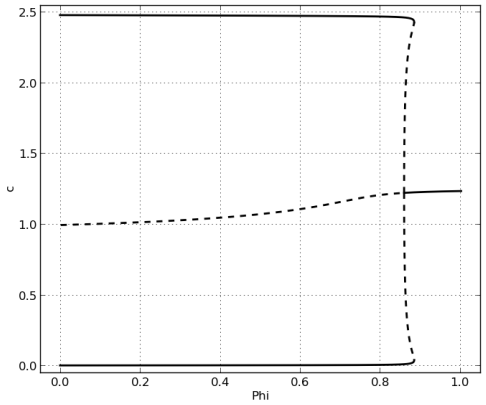
In figures 2c and 2d we can see that as we increase μ somewhat to 10^{-4}, the outer equilibria are perturbed above the manifold (and slightly inwards in the z-axis) and thus become stable, forming a bistable region before the critical point. ϕ_c is perturbed to a high value $\phi_c^*(\mu)$ and becomes a pitchfork bifurcation, with the saddle now a stable equilibrium at $z = 0.5$ and low d. Thus is formed a narrow region beyond ϕ_c^* value where three stable equilibria coexist with two new unstable equilibria. Finally we find that the two unstable equilibria swiftly grow towards the outer equilibria until they undergo saddle-node bifurcations, leaving the end of the parameter space with a single stable equilibrium at $z = 0.5$. Note how the innovation perturbs the attractive $d = 0$ manifold down so it is now outside the phase space, sending the trajectories starting close to $d = 0$ to the new central stable equilibrium instead.

As μ is further increased, the innovation process becomes increasingly strong relative to the opposing learning process. This pushes all bifurcation points to lower

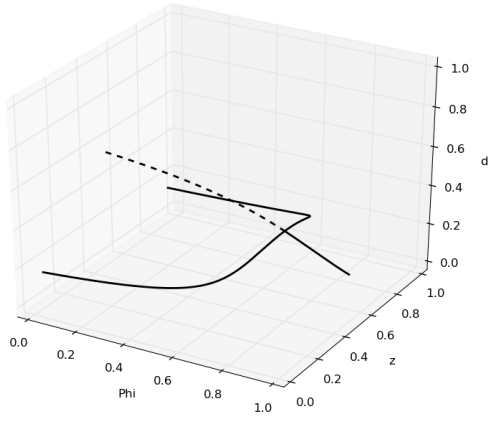
and lower ϕ values, as we see in diagrams 3a and 3b. The saddle-node bifurcations are more effected since innovation is stronger on the boundaries, leading these bifurcations to overtake the pitchfork bifurcation $\phi_c^*(\mu)$, as we observe in diagram 3c and 3d. We also note how this increased noise increases the d value of all equilibria, as one would expect. As μ continues to be increased, $\phi_c^*(\mu)$ shrinks until it leaves the phase space at $\phi = 0$; noise now predominates the whole parameter space and the central stable equilibrium is the the only attractor.



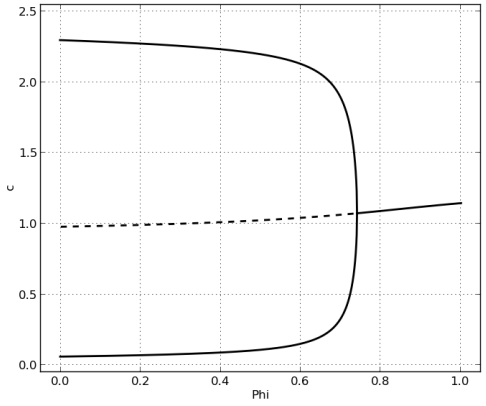
(a) $\mu = 10^{-3}$



(b) $\mu = 10^{-3}$

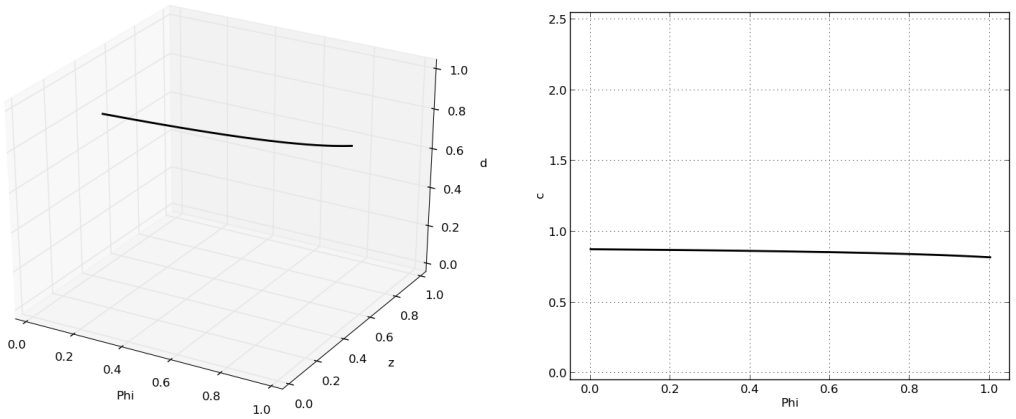


(c) $\mu = 10^{-2}$



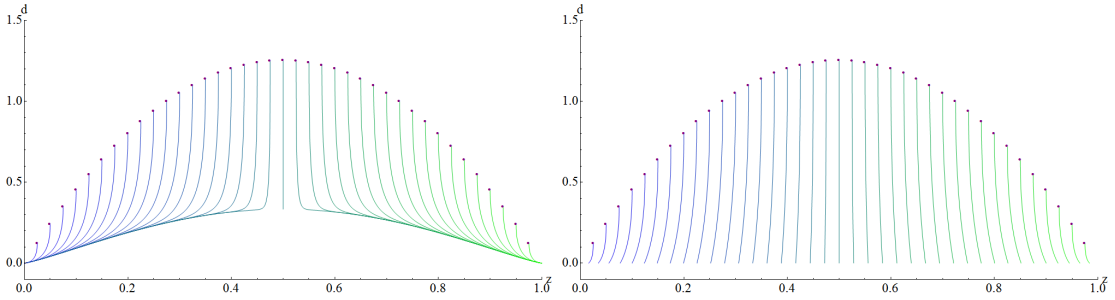
(d) $\mu = 10^{-2}$

Figure 3: Bifurcation diagrams for the $\lambda\rho$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria.



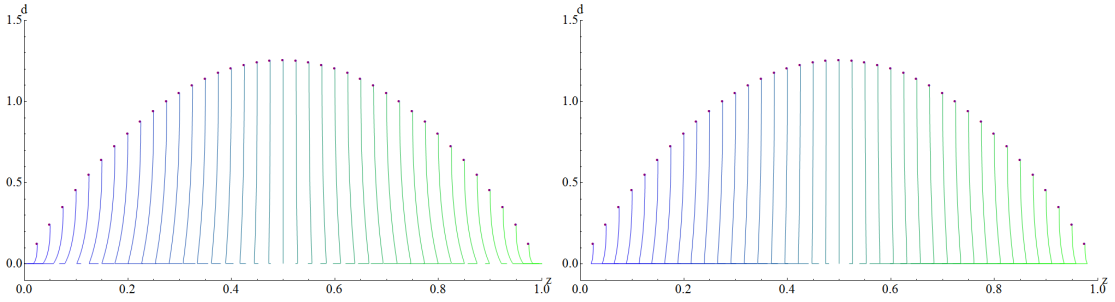
(a) $\mu = 10^{-1}$ (b) $\mu = 10^{-1}$

Figure 4: Bifurcation diagrams for the $\lambda\rho$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria.



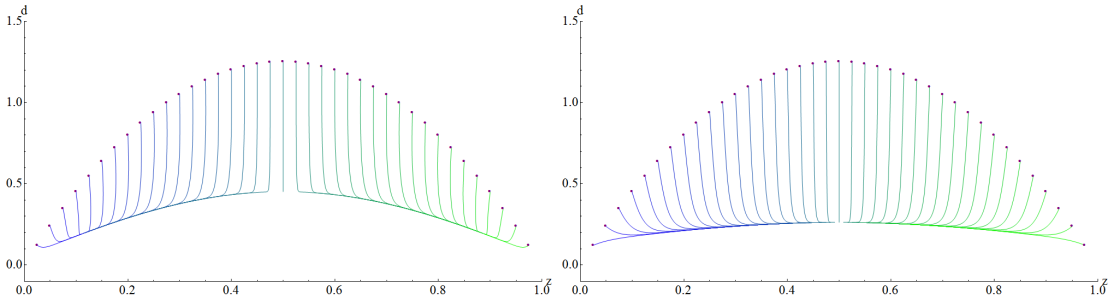
(a) $\phi = 0.5$ (b) $\phi = 0.9$

Figure 5: Orbits for the $\lambda\rho$-DAN model with $\mu = 0$ at $t = 10^5$. The orbits start at the purple dots - located at $z = 0.025, 0.05, \ldots 0.975$ with corresponding d and c values computed using equations 31 and 32 to be consistent with a random initial network configuration.



(a) $\phi = 0.9$ (b) $\phi = 0.95$

Figure 6: Orbits for the $\lambda\rho$-DAN model with $\mu = 10^{-4}$ at $t = 2 \cdot 10^6$. The orbits start at the purple dots - located at $z = 0.025, 0.05, \ldots 0.975$ with corresponding d and c computed using equations 31 and 32 to be consistent with a random initial network configuration.

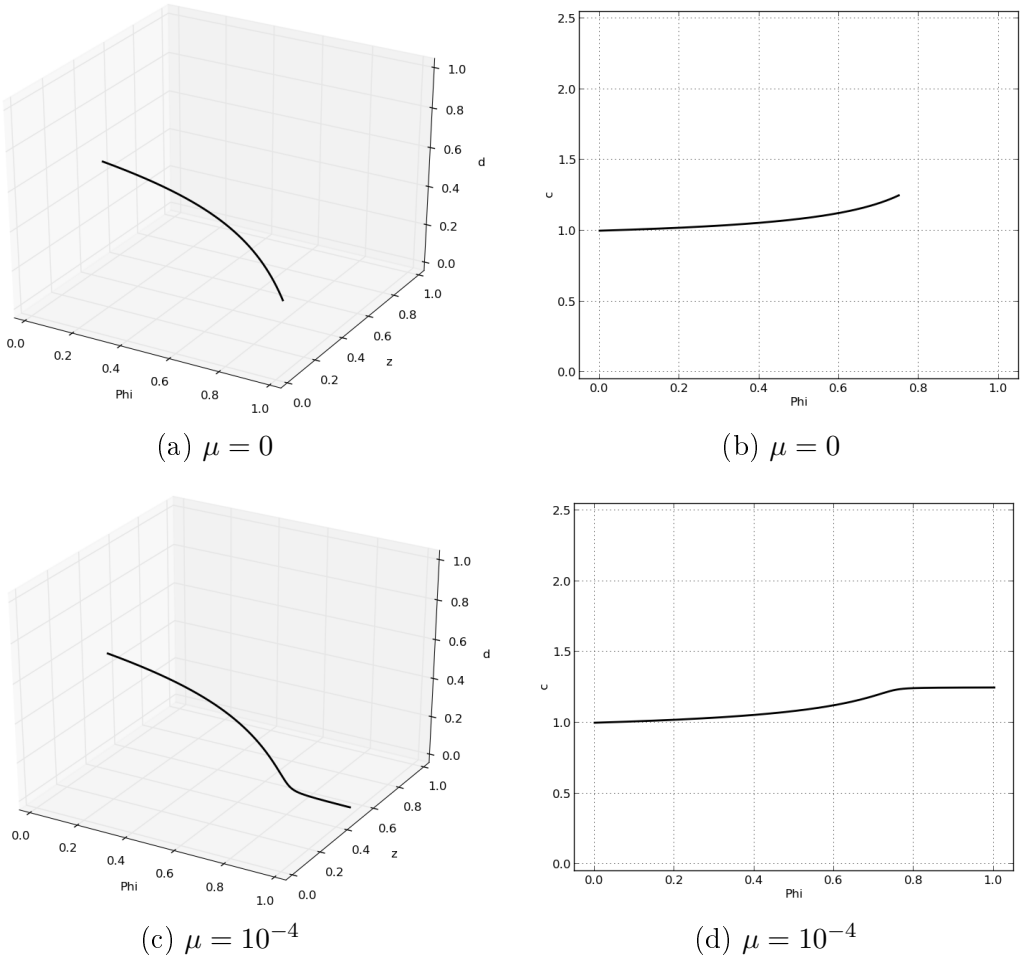


(a) $\phi = 0.5$ (b) $\phi = 0.9$

Figure 7: Orbits for the $\lambda\rho$-DAN model with $\mu = 10^{-2}$ at $t = 10^5$. The orbits start at the purple dots - located at $z = 0.025, 0.05, \ldots 0.975$ with corresponding d and c computed using equations 31 and 32 to be consistent with a random initial network configuration.

5.2.2 $\tau\rho$-DAN

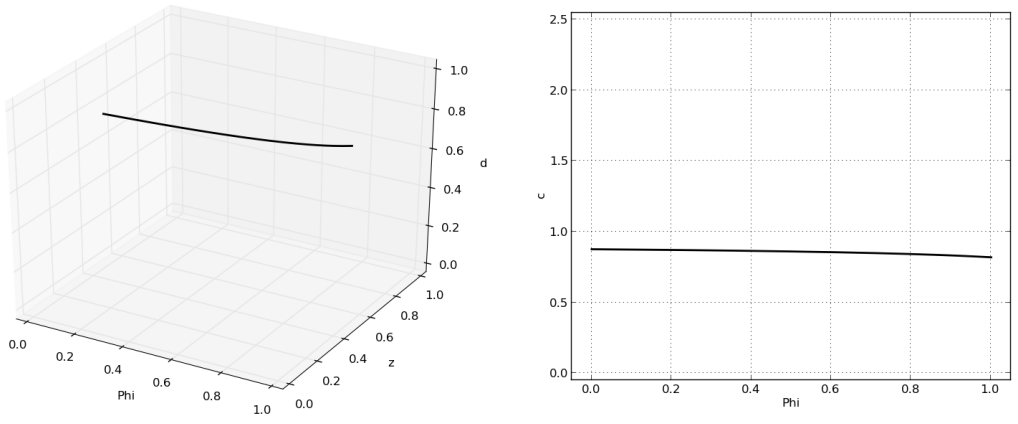
The $\tau\rho$-DAN model consists of the teaching, egocentric segregation, and innovation processes. The teaching term in the z-field is precisely equal and opposite to its corresponding learning term, and thus acts to stabilize the central equilibrium. This means that for $\mu = 0$ the flow is exactly reversed in the z-direction. For this reason, this model contrasts the previous one in having this central equilibrium as the sole attractor of all interior orbits for ϕ values lower than ϕ_c. For values above ϕ_c however, this equilibrium disappears into the manifold $d = 0$, corresponding once again to the degenerate frozen state (figure 8a and 8b). Once again - the invariant manifold transform from being repulsive for $\phi < \phi_c$ to being attractive for $\phi > \phi_c$. This is again a fragile state, destabilizing as soon as μ is increased with the central equilibrium rising to higher d as the sole stable equilibrium for all ϕ values. Once more - increasing μ raises the equilibria to higher d values, but otherwise this model exhibits rather trivial bifurcation behavior with no other phase changes observed.



(a) $\mu = 0$ (b) $\mu = 0$

(c) $\mu = 10^{-4}$ (d) $\mu = 10^{-4}$

Figure 8: Bifurcation diagrams for the $\tau\rho$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria.



(a) $\mu = 10^{-3}$ (b) $\mu = 10^{-3}$

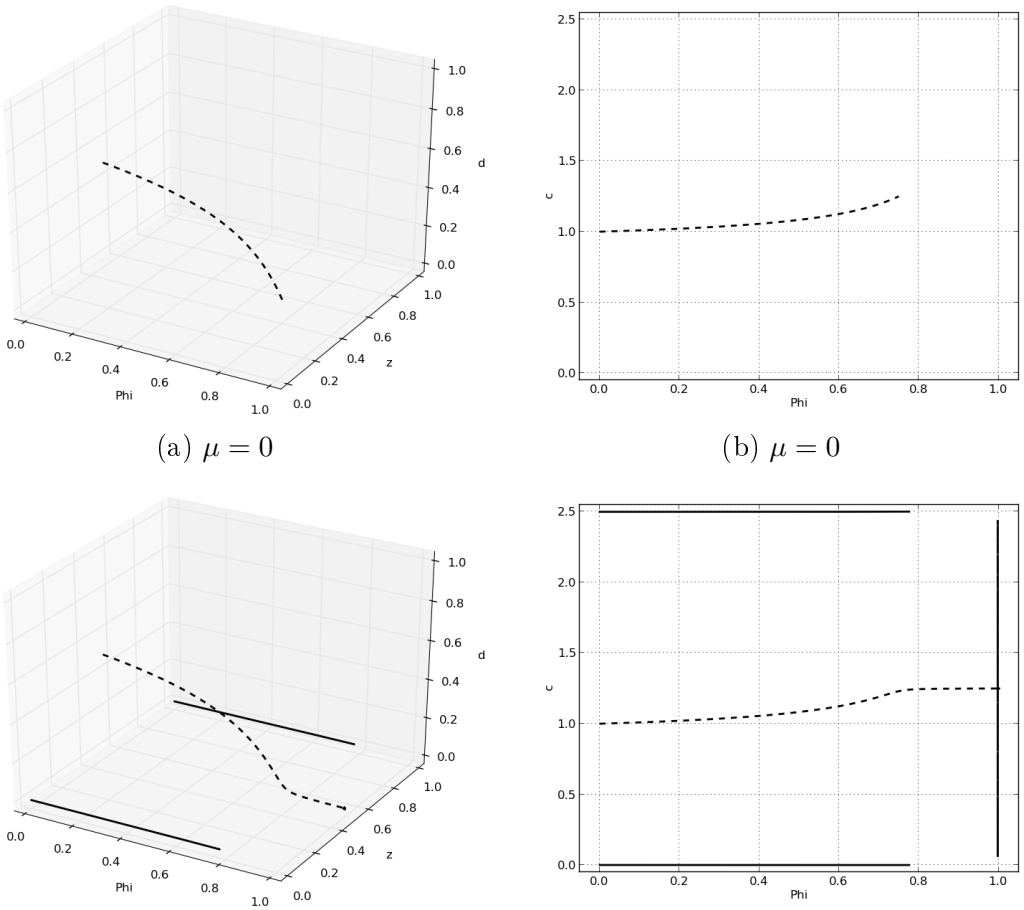
Figure 9: Bifurcation diagrams for the $\tau\rho$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria.

5.2.3 $\lambda\sigma$-DAN

Now we switch from egocentric to altercentric segregation, and while the distinction does not directly affect changes in the mean z-field, we shall see that it does indeed qualitatively change the dynamics of the discordian adaptive network. Both segregation processes - in beginning by selecting a random individual for an ego - tend to effect the most predominant node type in the network on average. In the selection of the ego's random neighbor (the alter), both processes only take effect when a discordant link is chosen. The key difference occurs in the rewiring that follows this initial step; for egocentric segregation proceeds by maintaining the edge's connection to the ego while altercentric segregation maintains the end of the link connected to the alter.

While both processes may at this point rewire the discordant link to another discordant link - given an identical network configuration, egocentric segregation would tend to convert the remaining discordant links to concordant links of the *majority* opinion, while altercentric segregation would tend to convert them to concordant links of the *minority* opinion. This implies that altercentric segregation affects an opposite c field than egocentric segregation.

The $\lambda\sigma$-DAN model incorporates the learning, altercentric segregation and innovation processes. Naturally, when $\phi = 0$ the model behaves identically to the $\lambda\rho$-DAN model. As ϕ is increased however, we notice the models diverge in behavior. As ϕ is increased, more and more of the rewired links are passed to the minority opinion. This increased number of concordant links increases the mean degree of the minority and reduces the mean degree of the majority. This bestows the minority with an advantage against invading opinions, and the majority gets a corresponding disadvantage. Thus altercentric segregation indirectly produces a z-field pushing away from $z = 0.5$, but this field should grow weaker for z values further from $z = 0.5$ as more and more of the discordant links are converted to other discordant links. This combination of differences in fields explains why we see that for equal values of μ, the $\lambda\sigma$-DAN model has a higher ϕ_c^* than the $\lambda\rho$-DAN model.

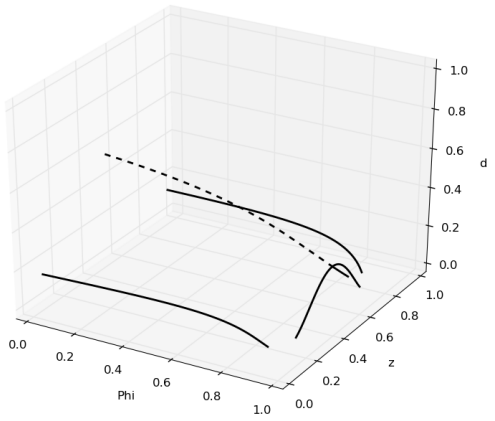


(a) $\mu = 0$ (b) $\mu = 0$

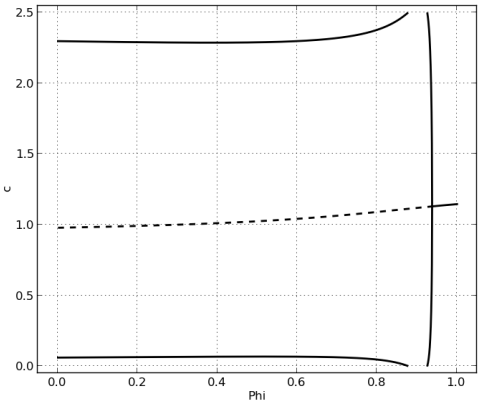
(c) $\mu = 10^{-4}$ (d) $\mu = 10^{-4}$

Figure 10: Bifurcation diagrams for the $\lambda\sigma$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria. Note that the 'dot' in figure 10c is in fact a little arch of equilibria, which grows as we increase μ (see figures 11a and 11c).

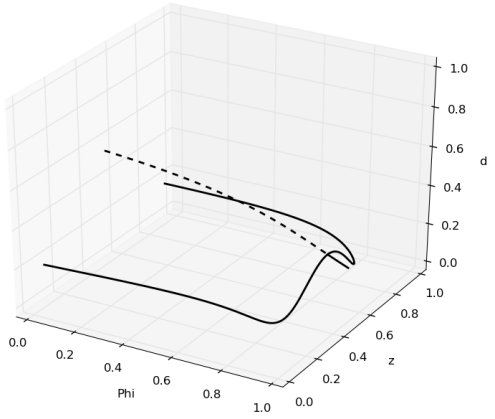
As μ is increased, we notice a similar progression to the one observed in the $\lambda\rho$-DAN model; equilibria have growing d values and the pitchfork bifurcation moves to lower and lower ϕ values. As noted before - altercentric segregation affects different z- and c-fields than egocentric segregation; for this reason we see a dip in the equilibria for high ϕ and extreme z values.



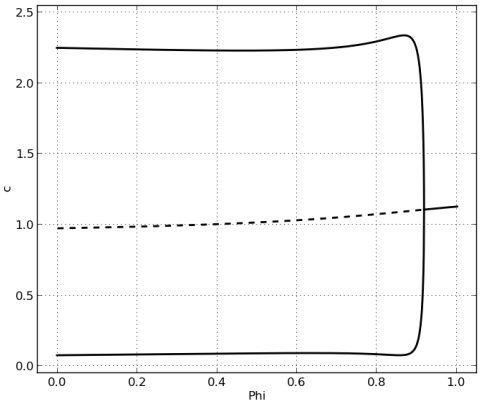
(a) $\mu = 10^{-2}$



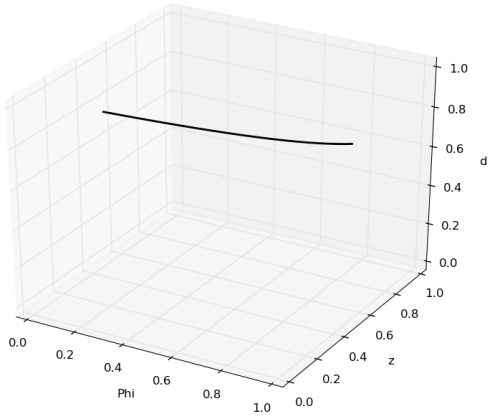
(b) $\mu = 10^{-2}$



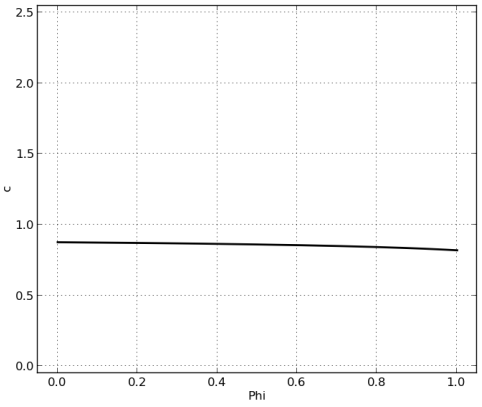
(c) $\mu = 1.2 \cdot 10^{-2}$



(d) $\mu = 1.2 \cdot 10^{-2}$

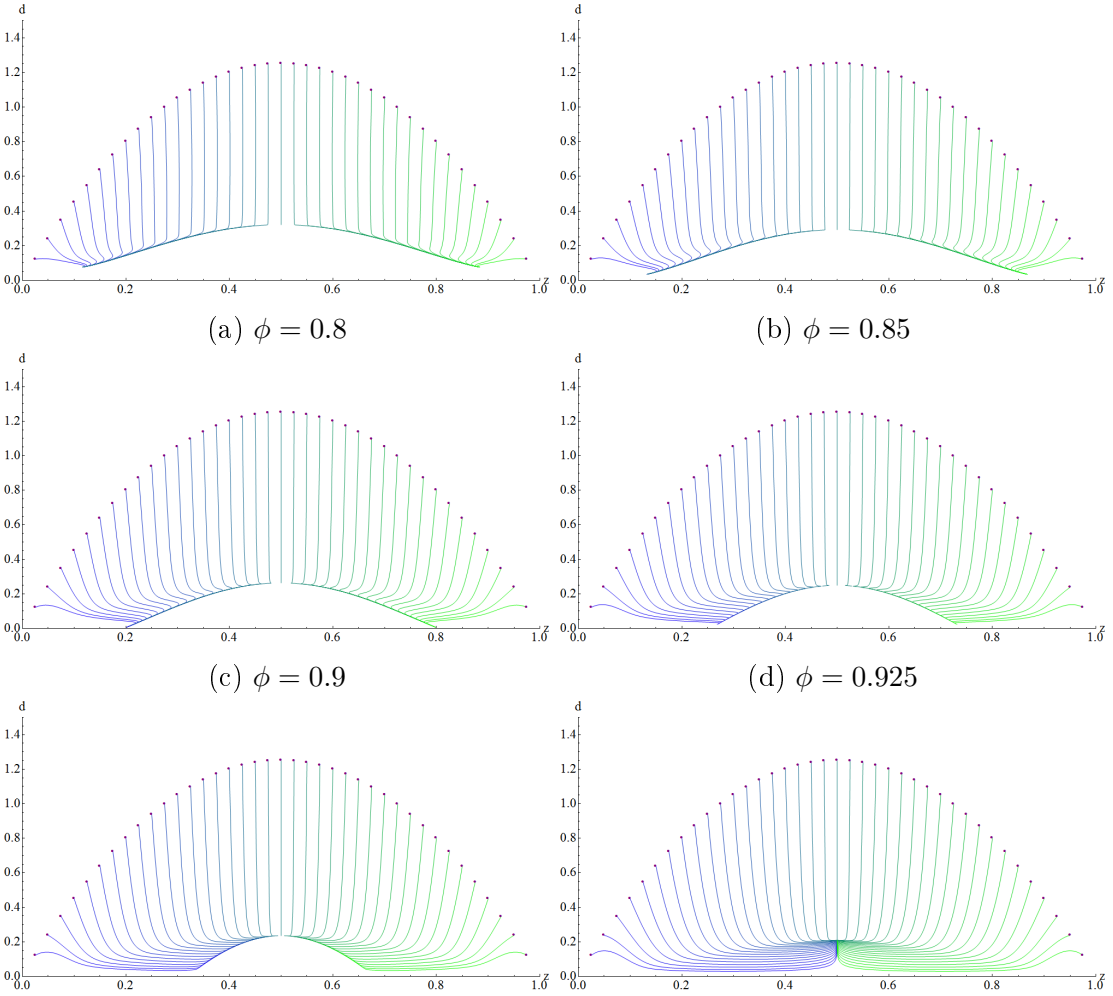


(e) $\mu = 10^{-1}$



(f) $\mu = 10^{-1}$

Figure 11: Bifurcation diagrams for the $\lambda\sigma$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria. Note that the arch in high ϕ values of figures 11a and 11c spans different ϕ values, with the outer ends having slightly lower ϕ values and lower d values than the central point - where the pitchfork bifurcation occurs.



(a) $\phi = 0.8$ (b) $\phi = 0.85$

(c) $\phi = 0.9$ (d) $\phi = 0.925$

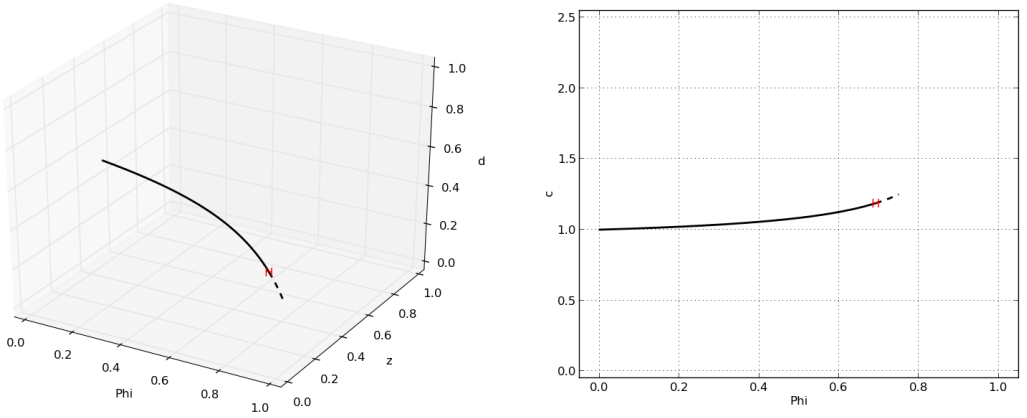
(e) $\phi = 0.95$ (f) $\phi = 1.0$

Figure 12: Orbits for the $\lambda\sigma$-DAN model with $\mu = 10^{-2}$ at $t = 10^5$. The orbits start at the purple dots - located at $z = 0.025, 0.05, \ldots 0.975$ with d and c computed using equations 31 and 32 to be consistent with a random initial network configuration.

5.2.4 $\tau\sigma$-DAN

The $\tau\sigma$-DAN model incorporates the teaching, altercentric segregation and innovation processes. Just like its egocentric counterpart, the model starts out with a single stable equilibrium at $z = 0.5$ for low ϕ and μ, when teaching predominates. However - as previously mentioned - as ϕ is increased rewired links are increasingly passed to the minority opinion, giving it an advantage against invading opinions, while the majority gets a corresponding disadvantage. The minority opinion thus grows, but the edge it gains with its higher mean degree lags behind this growth and allows it to grow to become the majority - at which points the dynamics reverse. This explains the Hopf bifurcation and subsequent oscillations observed.

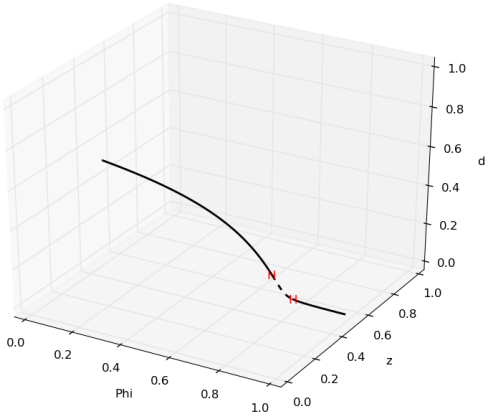
When ϕ is further increased however, discord is dropped too fast and the network reaches a frozen state where the oscillations are no longer sustainable. When μ is increased, we observe the same raised discord in equilibria we saw in the other models. The increased rate of innovation also acts to increase the field strength towards $z = 0.5$ until the oscillations are suppressed.



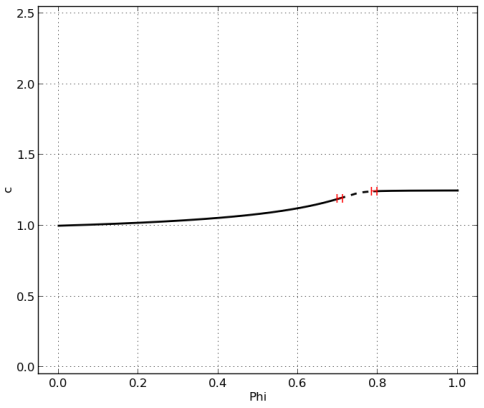
(a) $\mu = 0$ (b) $\mu = 0$

Figure 13: Bifurcation diagrams for the $\tau\sigma$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria and the red H marks the Hopf bifurcation point.

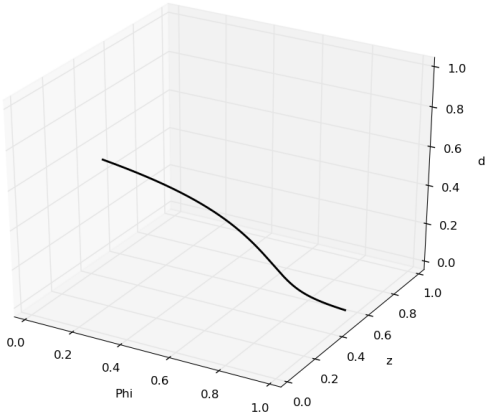
The period of the oscillations grows with ϕ until the second Hopf bifurcation is reached, as we can see in figure 16d and 17d.



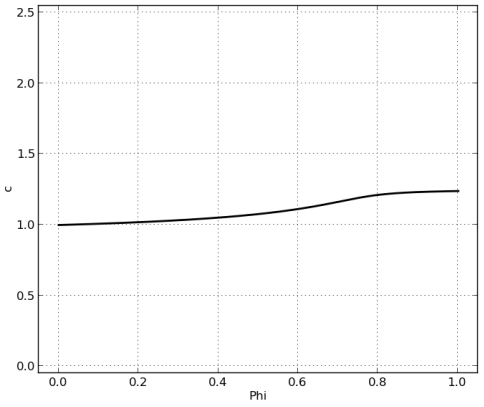
(a) $\mu = 10^{-4}$



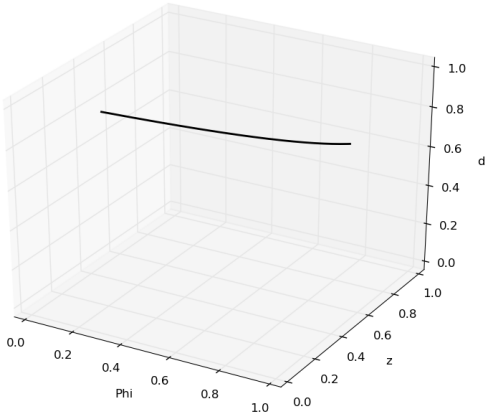
(b) $\mu = 10^{-4}$



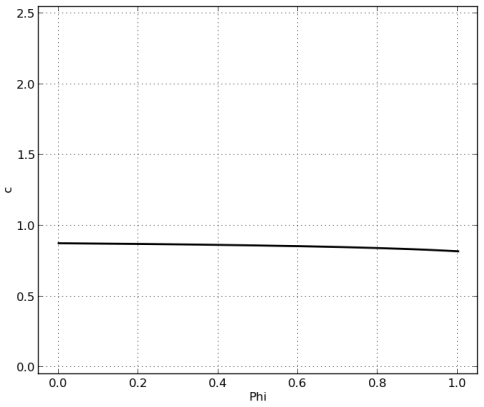
(c) $\mu = 10^{-3}$



(d) $\mu = 10^{-3}$



(e) $\mu = 10^{-1}$



(f) $\mu = 10^{-1}$

Figure 14: Bifurcation diagrams for the $\tau\sigma$-DAN model. Solid lines show stable equilibria, dashed lines are unstable equilibria and the red H marks the Hopf bifurcation points.

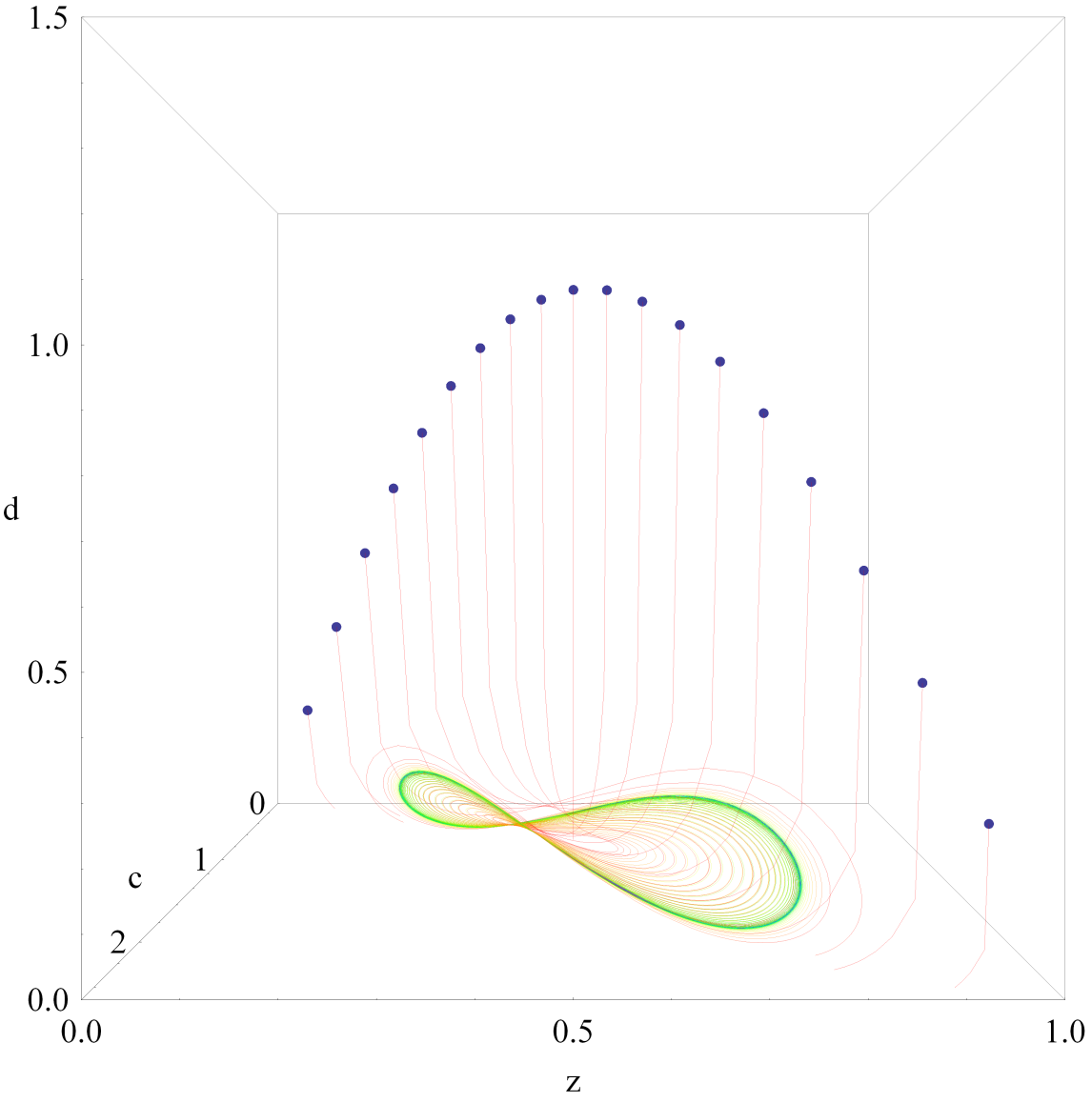
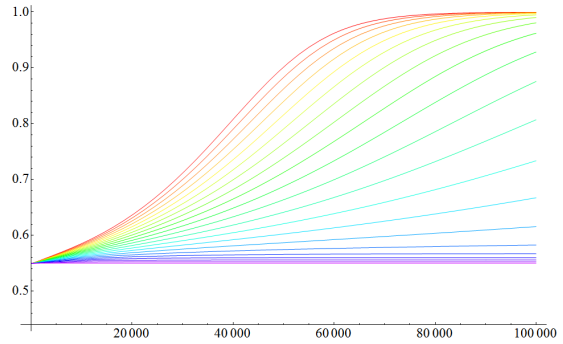


Figure 15: Dynamics of the $\tau\sigma$-DAN model with $\phi = 0.72$ converging to a limit cycle. The hue of the trajectory is proportioned to time - with red at $t = 0$ and violet at $t = 2 \cdot 10^7$.

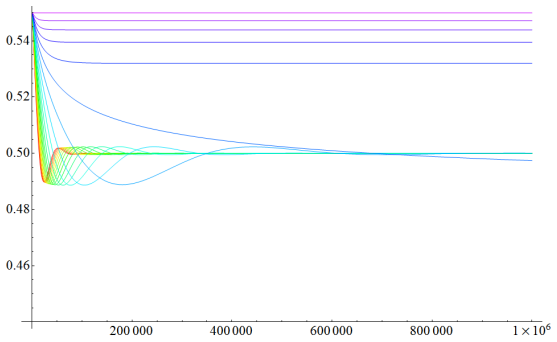
5.2.5 Convergence Times

Another interesting aspect of our four models is their convergence times. For $\mu = 0$ we observe that convergence in z value is quickest for $\phi = 1$ (see figure 16); the trajectories here simply begin in the equilibrium z value. As ϕ is lowered, convergence time slowly grows to the order of magnitude of orders 10^3 to 10^4, until it suddenly jumps in magnitudes of the order 10^5 to 10^6 at ϕ_c. As ϕ is further lowered, convergence time drops gradually, reaching the orders of magnitude ranging from 10^4 to 10^5.

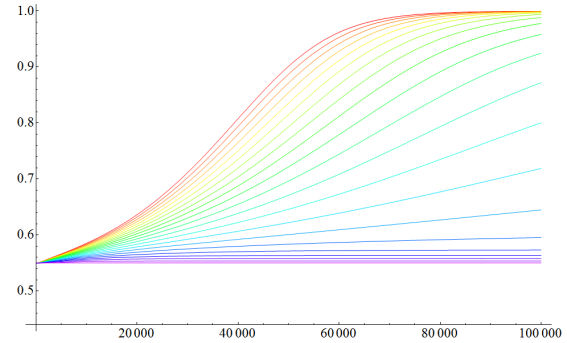
For $\mu > 0$ the phase beyond ϕ_c is perturbed, and thus convergence in this region is now drastically longer, as we see in figures 17 and 18. $\phi = 0$ now consistently is the quickest to converge, although the slowest convergence here is often found for some intermediate ϕ value close to ϕ_c.



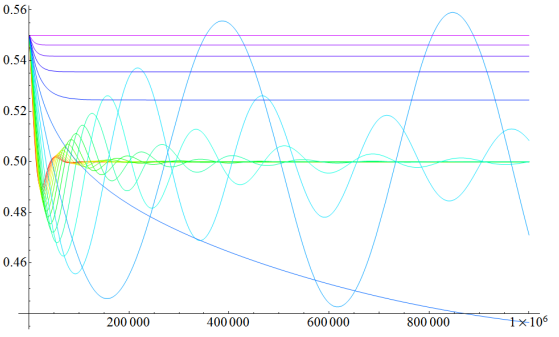
(a) $\lambda\rho$-Model



(b) $\tau\rho$-DAN model

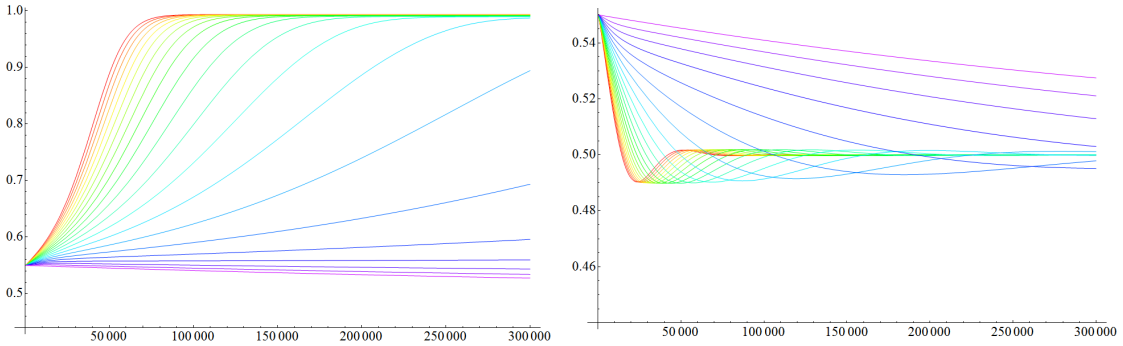


(c) $\lambda\sigma$-DAN model



(d) $\tau\sigma$-DAN model

Figure 16: Time series of z for the four discordian adaptive network mean field models at $\mu = 0$, with initial $z_0 = 0.23$. ϕ values $0, 0.05, \ldots, 1.0$ are shown, with hue proportional to ϕ ($\phi = 0$ is red and $\phi = 1$ is violet).



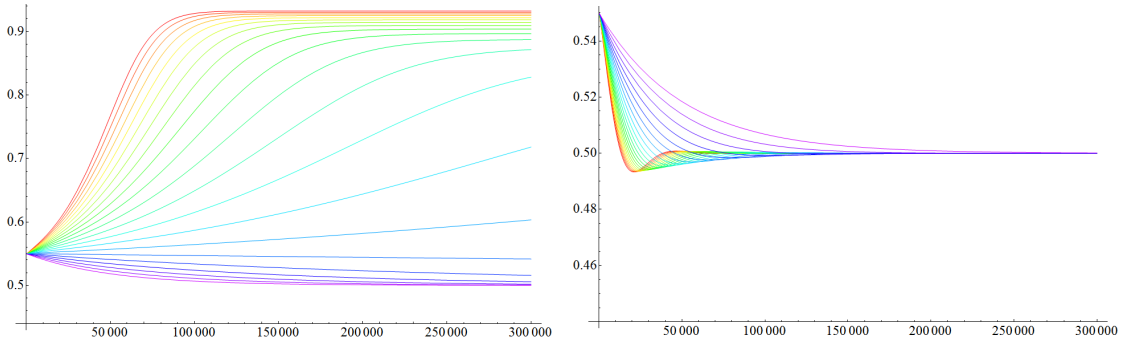
(a) $\lambda\rho$-DAN model

(b) $\tau\rho$-Model

(c) $\lambda\sigma$-DAN model

(d) $\tau\sigma$-DAN model

Figure 17: Time series of z for the four discordian adaptive network mean field models at $\mu = 10^{-3}$, with initial $z_0 = 0.23$. ϕ values $0, 0.05, \ldots, 1.0$ are shown, with hue proportional to ϕ ($\phi = 0$ is red and $\phi = 1$ is violet).



(a) $\lambda\rho$-DAN model

(b) $\tau\rho$-DAN model

(c) $\lambda\sigma$-DAN model

(d) $\tau\sigma$-DAN model

Figure 18: Time series of z for the four discordian adaptive network mean field models at $\mu = 10^{-2}$, with initial $z_0 = 0.23$. ϕ values $0, 0.05, \ldots, 1.0$ are shown, with hue proportional to ϕ ($\phi = 0$ is red and $\phi = 1$ is violet).

6 Individual-based Stochastic Simulations

In this section we present the results of stochastic individual-based simulations performed using the C++ library *Largenet2* [8]. The code written for this purpose is available in Appendix B. The simulations were performed for $N = 1000$ and $k = 5$, with an initial fraction $z_0 = 0.55$ of nodes set to the + state and edges randomly distributed to form an Erdős-Rényi random graph. Each simulation run lasted $t = 1,000,000$ steps and was repeated 100 times for each set of parameter values. Parameter values of ϕ ranged from 0 to 1 with increments of 0.05 and μ values 0, 10^{-4}, 10^{-3}, 10^{-2} and 10^{-1} we investigated.

We can reason a few boundary conditions that the stochastic model should have. In the noise-free case without innovation ($\mu = 0$) we know that if $\phi = 0$ we only have an imitation process and every component in the network drifts to consensus. The striking importance of the initial network configuration here is notable: when we have no edges the largest connected component (LCC) is size 1, and when we have the maximum saturation of $N(N-1)/2$ edges, the LCC is of size N. So on the one extreme the initial z value z_0 must be the equilibrium value as well, and on the other we expect the network to eventually drift consensus ($z = 0$ or $z = 1$). In the case of learning this is expected to happen swiftly, but for teaching - with the mean field pushing to $z = 0.5$ - this can take an exceedingly long time.

In our case networks are configured by random assignment of $L = 2500$ edges to all possible (unconnected) pairs chose from the $N = 1000$. In this case we expect the LCC will contain 99% of the nodes, and thus we can expect that the stochastic models usually converge to a concensus (or close to one) for $\mu = 0$ and $\phi = 0$.

Since our simulations primarily use $z = 0.55$ as an initial condition, in the bistable case we see a narrow majority of simulations attracted to $z = 1$. We note for all four models that when ϕ and μ are close to zero (i.e. almost all events are imitation), variability is higher. It is in these cases that we see the networks occasionally converge to $z = 1$ (Learning models) and even $z = 0$ in the teaching models. In the stochastic model, Imitation events of both kinds occur and the network - being largely fixed for these parameter values - drifts hither and thither. This effect is not accounted for in the Mean-Field models, which only observe the expected change. Indeed we note this an extreme discrepancy, for in the Mean-Field Models with low μ we observe that the lowest ϕ values converge the fastest, while in the stochastic model they converge the slowest.

On the other end of the spectrum - if $\phi = 1$ - we have only rewiring and the system is degenerate and every initial z value is also its own equilibrium. The question is then to which equilibrium values do d and c converge to. d should generally drop - being reduced by all processes except innovation - to either to zero or to some positive equilibrium value depending on the network's initial conditions. In our case the number of edges is sufficiently low for it to drop to zero, and we note indeed that for low μ and high ϕ values the stochastic models all tend to converge to the initial value (see figure 6.1.1) - in contrast to the mean field which predicts for small $\mu > 0$ that these trajectories would converge to $z = 0.5$.

The phase transition ϕ_c which is observed in the Mean-Field models seems smoothed in the stochastic models. In particular the $\lambda\sigma$-DAN model exhibits a

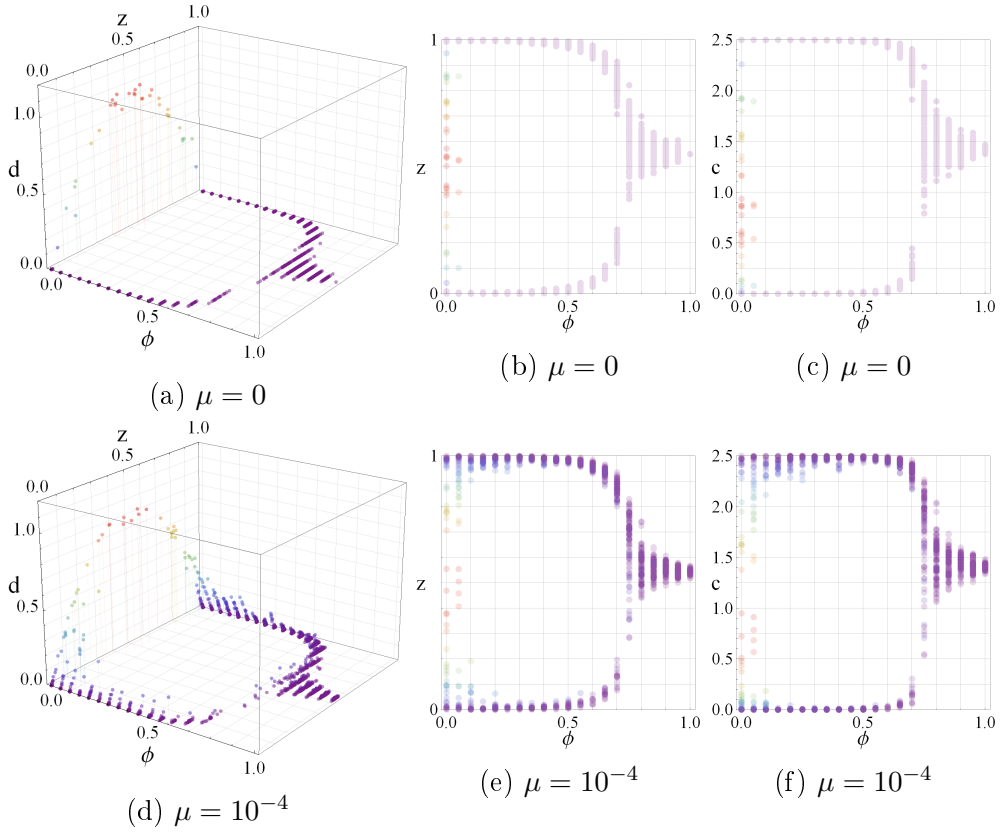
much more gradual shift in equilibrium values from $z = 0$ and $z = 1$ to $z = 0.5$ as ϕ increases. Thus some of the more nuanced differences in models that we observe in the mean field models do not seem to appear in the stochastic models; we do not seem to observe a tristable region in the $\lambda\rho$-DAN model for instance (see figure 6.1.1), nor do we seem to see cycles in the $\tau\sigma$-DAN model (see figure 6.1.4).

Notable for $\mu = 10^{-2}$ in the learning models is that just as in the mean field model intermediate ϕ values have the slowest convergence rate. In this case, different processes are acting against one another to slow convergence. In contrast, the discrepancy we observed for low μ and ϕ values is cause by *the same process* acting against itself, which is why this effect was not reflected in the mean field.

Surprisingly perhaps - considering the dramatic differences in the mean field models - egocentric and altercentric segregation seem to differ very little from one another in stochastic case (see figures 19 to 28). We do note however that for the learning case, the altercentric model exhibits a more gradual inwards perturbation of the outer equilibria with rising ϕ, unlike the egocentric model and in contrast to its mean field analog. This means that the indirect z-field we deduced in the Mean-Field is indeed observed in the stochastic model.

6.1 Simulation Results

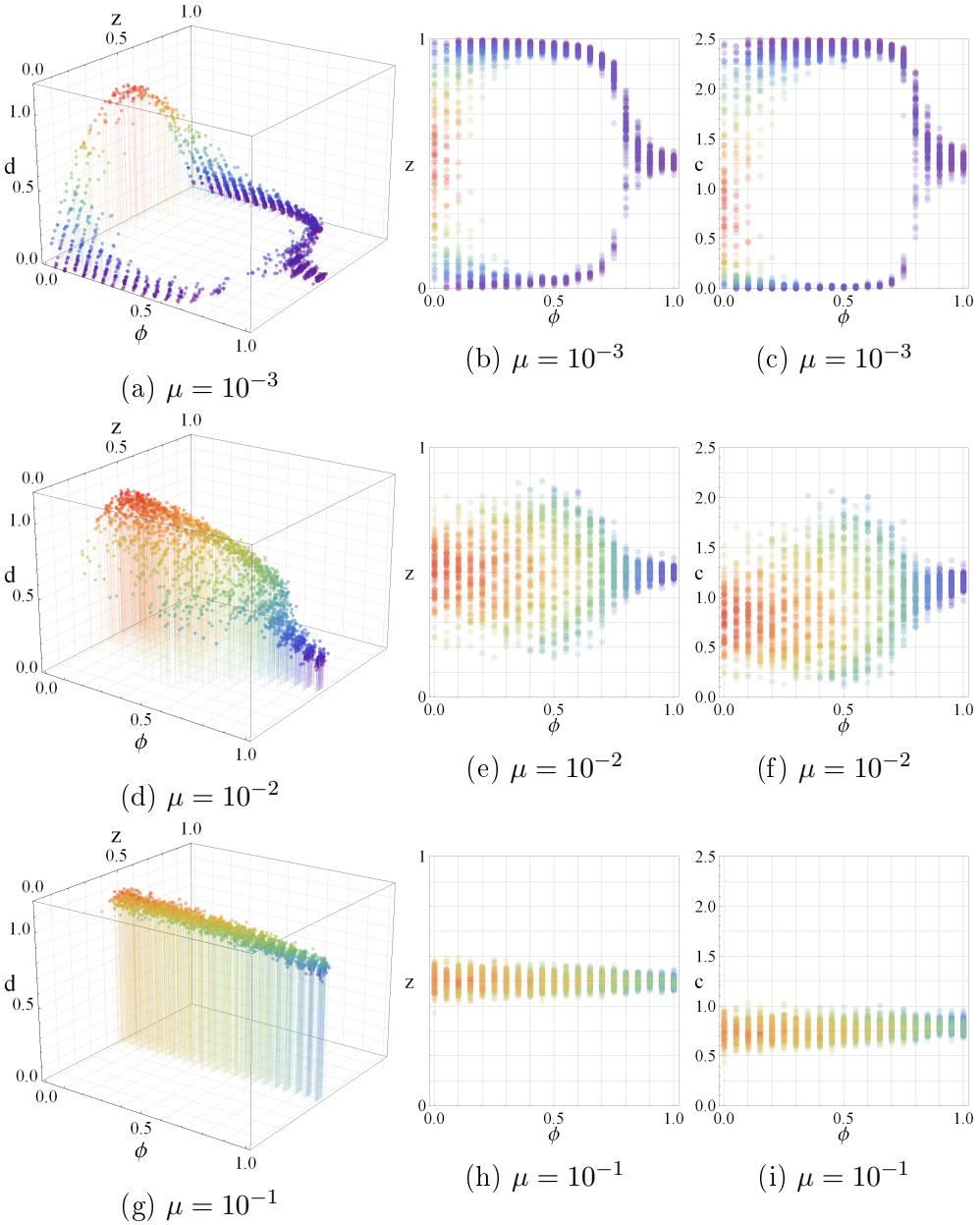
6.1.1 the $\lambda\rho$-DAN model



(a) $\mu = 0$ (b) $\mu = 0$ (c) $\mu = 0$

(d) $\mu = 10^{-4}$ (e) $\mu = 10^{-4}$ (f) $\mu = 10^{-4}$

Figure 19: Scatter plots of the largenet2 simulations for the $\lambda\rho$-DAN model. Color reflects d-value, renormalized to the range of values in each plot.

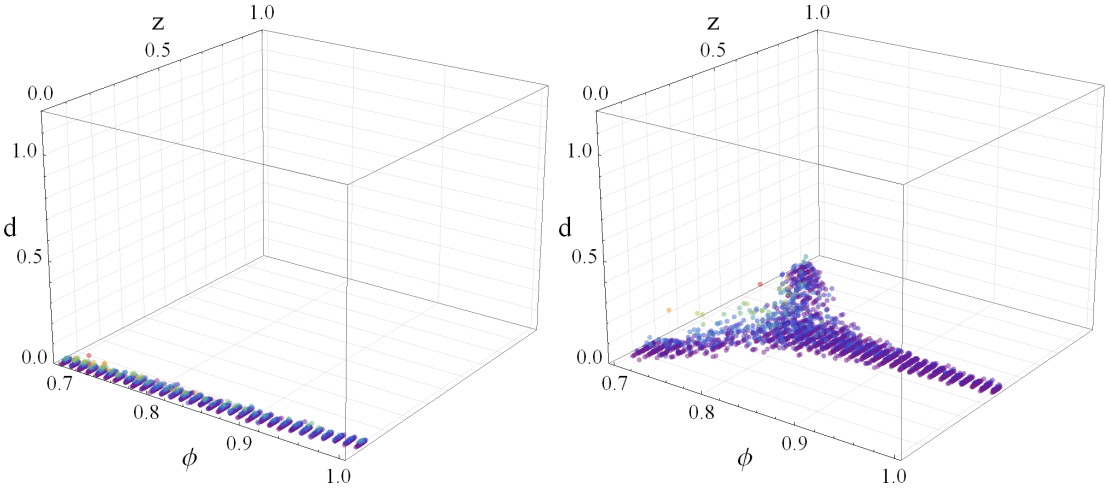


(a) $\mu = 10^{-3}$ (b) $\mu = 10^{-3}$ (c) $\mu = 10^{-3}$

(d) $\mu = 10^{-2}$ (e) $\mu = 10^{-2}$ (f) $\mu = 10^{-2}$

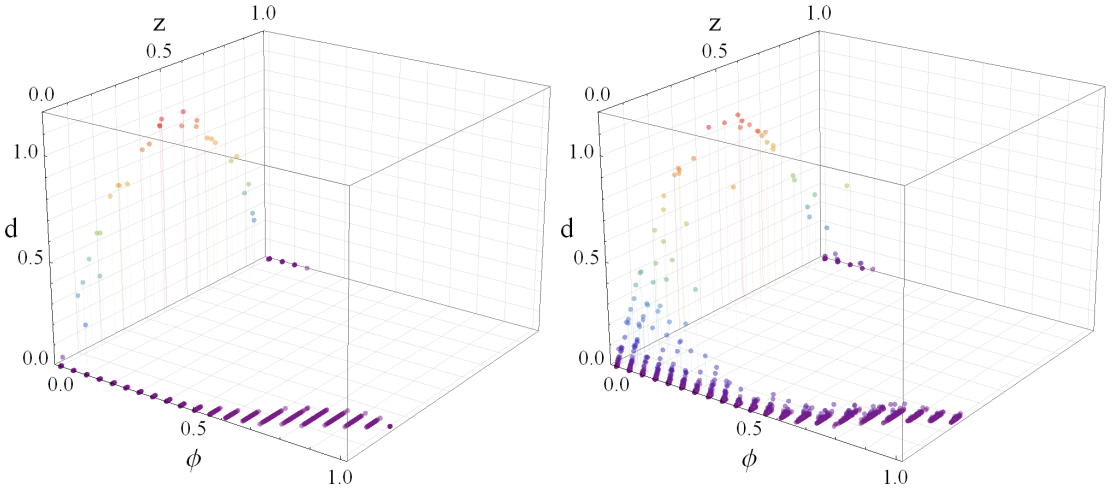
(g) $\mu = 10^{-1}$ (h) $\mu = 10^{-1}$ (i) $\mu = 10^{-1}$

Figure 20: Scatter plots of the largenet2 simulations for the $\lambda\rho$-DAN model. Color reflects d-value, renormalized to the range of values in each plot.



(a) $z_0 = 0.01$ (b) $z_0 = 0.49$

Figure 21: Scatter plots of the largenet2 simulations for the $\lambda\rho$-DAN model with $\mu = 10^{-4}$, starting under different initial conditions. Color reflects d-value, renormalized to the range of values in each plot. In this region of the parameter space the Mean-Field Model displays tristability, which we do not observe in the stochastic model.

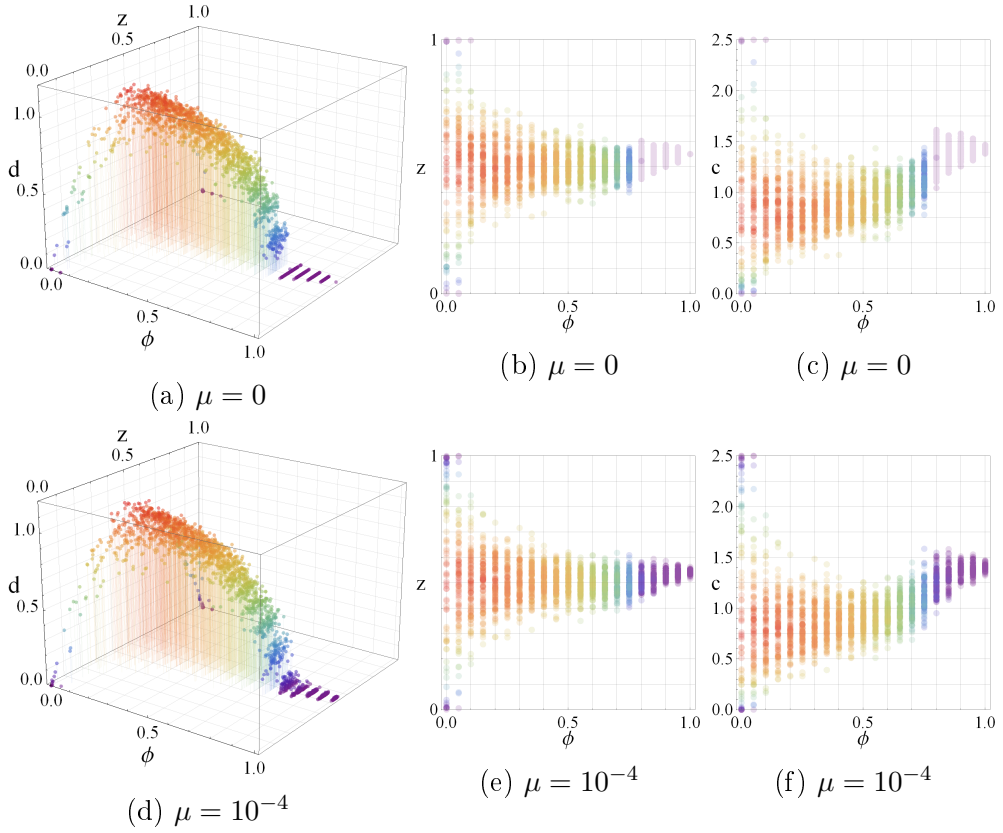


(a) $\mu = 0$ (b) $\mu = 10^{-4}$

(c) $\mu = 10^{-3}$ (d) $\mu = 10^{-2}$

Figure 22: Scatter plots of the largenet2 simulations for the $\lambda\rho$-DAN model with $z_0 = 0.23$. Color reflects d-value, renormalized to the range of values in each plot. We note the proximity of the simulations to the initial conditions for high ϕ and low μ; the Mean-Field model predicted a stable equilibrium at $z = 0.5$; it is likely that this will eventually converge to this value regardless, but the Mean-Field model predicts convergence at this point.

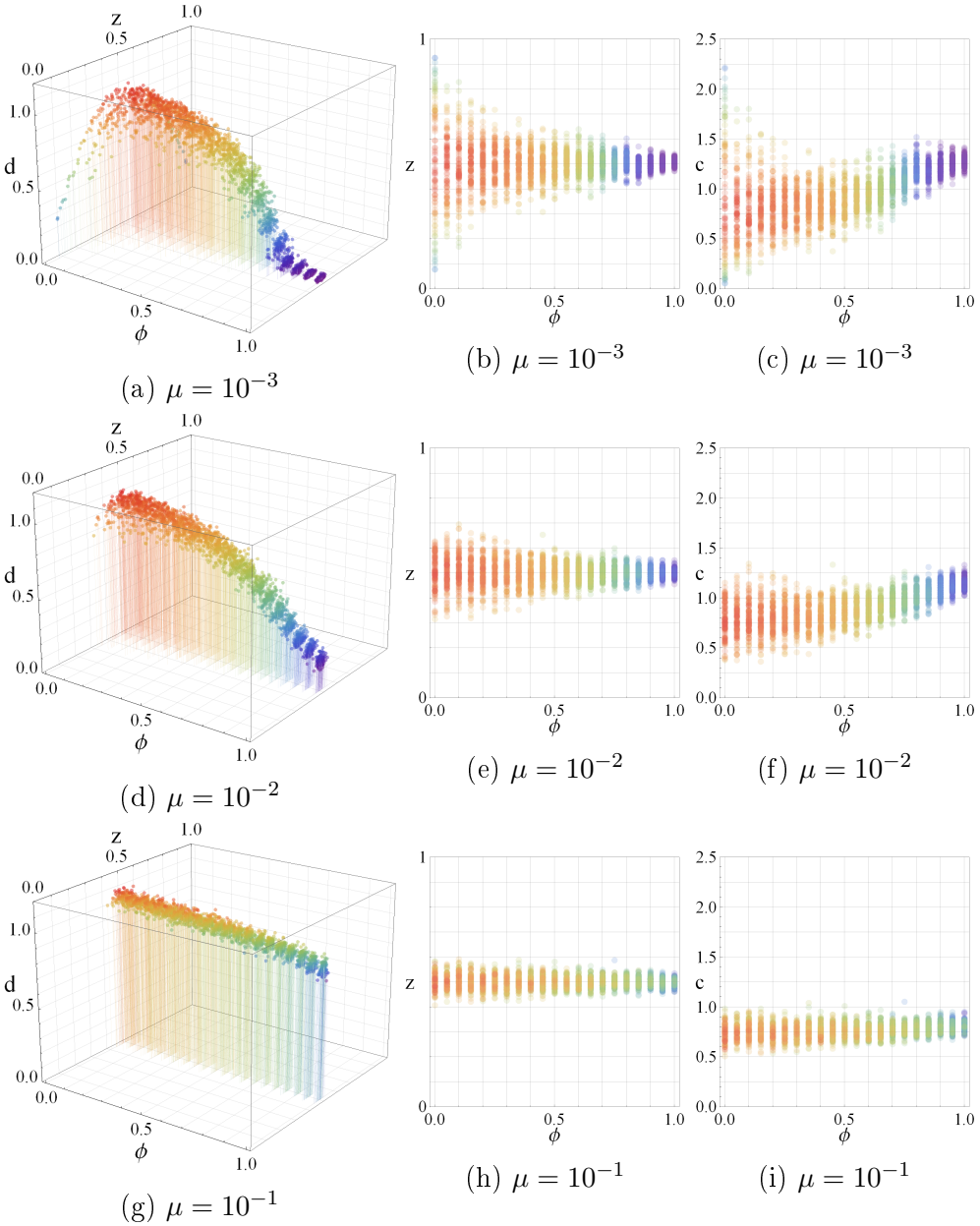
6.1.2 the $\tau\rho$-DAN model



(a) $\mu = 0$ (b) $\mu = 0$ (c) $\mu = 0$

(d) $\mu = 10^{-4}$ (e) $\mu = 10^{-4}$ (f) $\mu = 10^{-4}$

Figure 23: Scatter plots of the largenet2 simulations for the $\tau\rho$-DAN model. Color reflects d-value, renormalized to the range of values in each plot.



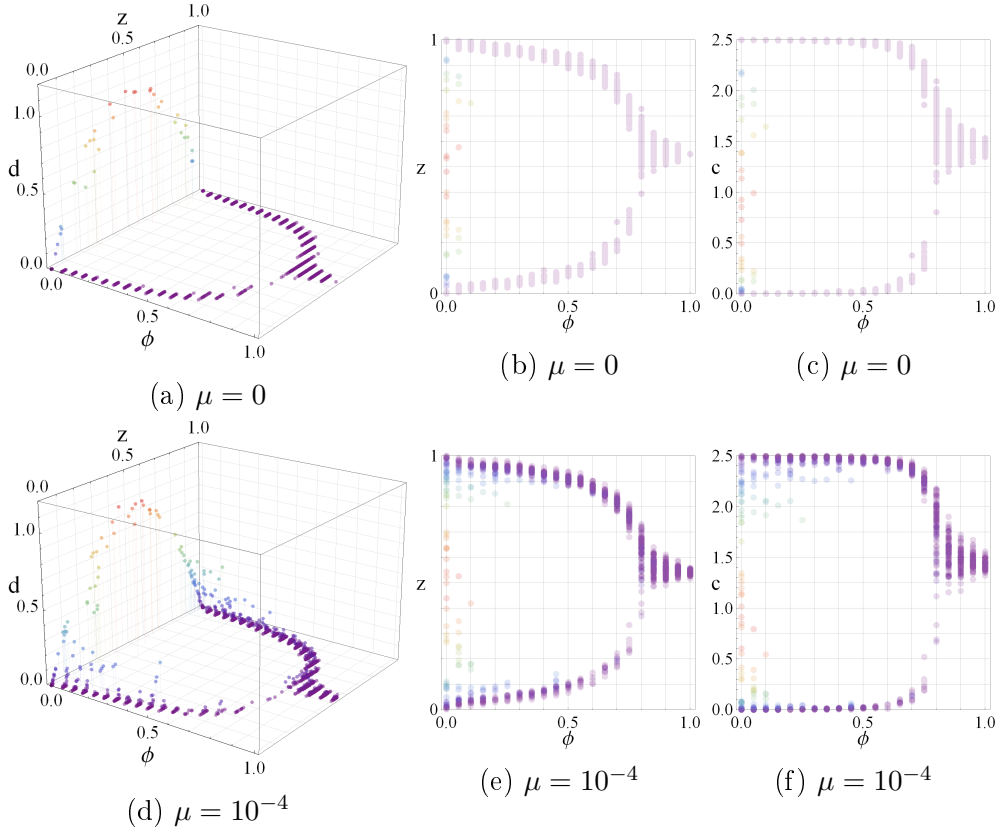
(a) $\mu = 10^{-3}$ (b) $\mu = 10^{-3}$ (c) $\mu = 10^{-3}$

(d) $\mu = 10^{-2}$ (e) $\mu = 10^{-2}$ (f) $\mu = 10^{-2}$

(g) $\mu = 10^{-1}$ (h) $\mu = 10^{-1}$ (i) $\mu = 10^{-1}$

Figure 24: Scatter plots of the largenet2 simulations for the $\tau\rho$-DAN model. Color reflects d-value, renormalized to the range of values in each plot.

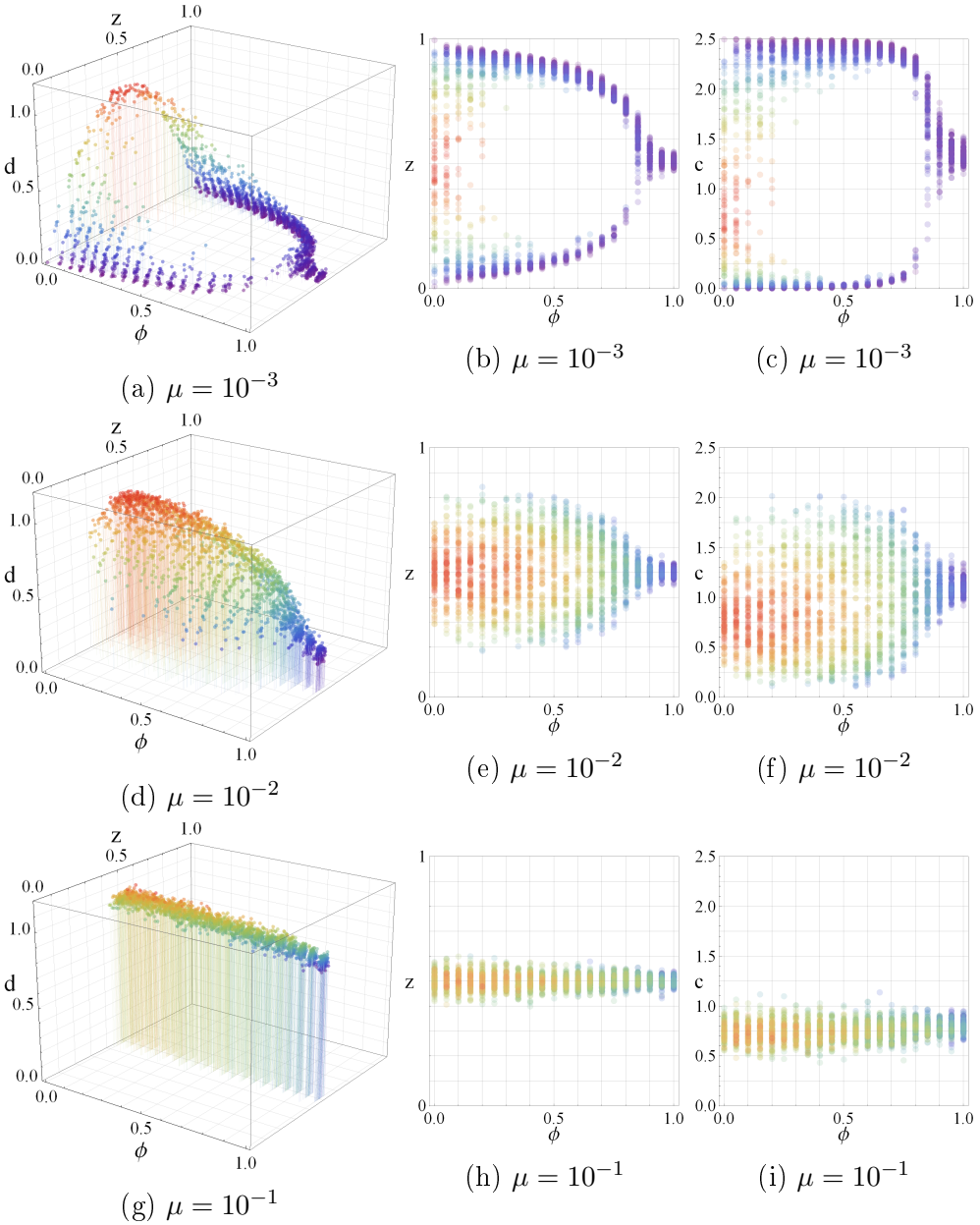
6.1.3 the $\lambda\sigma$-DAN model



(a) $\mu = 0$ (b) $\mu = 0$ (c) $\mu = 0$

(d) $\mu = 10^{-4}$ (e) $\mu = 10^{-4}$ (f) $\mu = 10^{-4}$

Figure 25: Scatter plots of the largenet2 simulations for the $\lambda\sigma$-DAN model. Color reflects d-value, renormalized to the range of values in each plot.



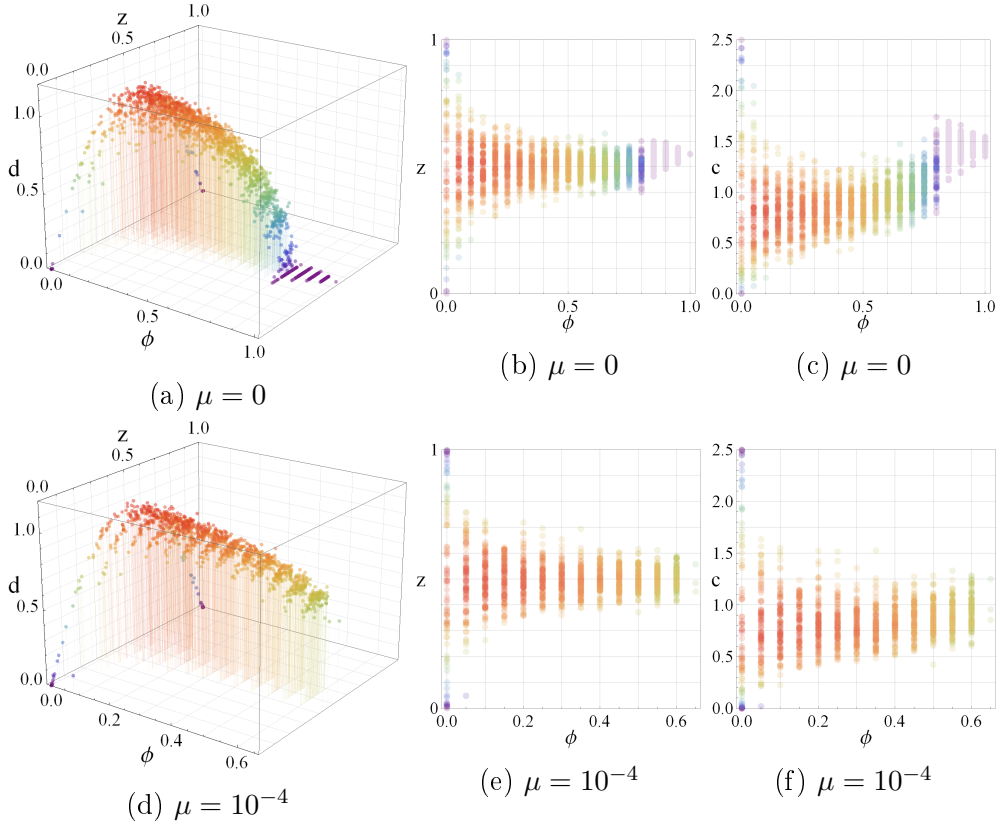
(a) $\mu = 10^{-3}$ (b) $\mu = 10^{-3}$ (c) $\mu = 10^{-3}$

(d) $\mu = 10^{-2}$ (e) $\mu = 10^{-2}$ (f) $\mu = 10^{-2}$

(g) $\mu = 10^{-1}$ (h) $\mu = 10^{-1}$ (i) $\mu = 10^{-1}$

Figure 26: Scatter plots of the largenet2 simulations for the $\lambda\sigma$-DAN model. Color reflects d-value, renormalized to the range of values in each plot.

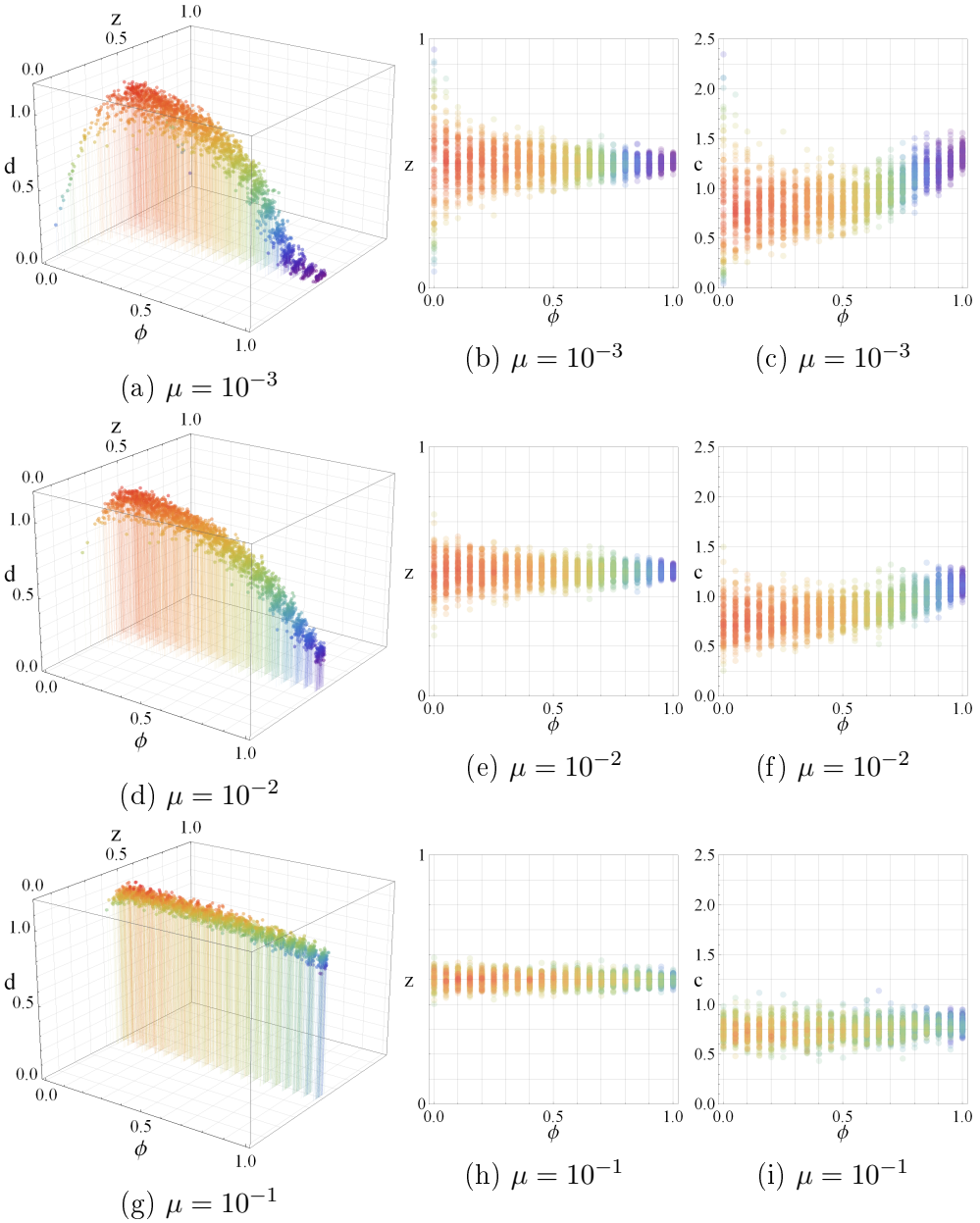
6.1.4 the $\tau\sigma$-DAN model



(a) $\mu = 0$ (b) $\mu = 0$ (c) $\mu = 0$

(d) $\mu = 10^{-4}$ (e) $\mu = 10^{-4}$ (f) $\mu = 10^{-4}$

Figure 27: Scatter plots of the largenet2 simulations for the $\tau\sigma$-DAN model. Color reflects d-value, renormalized to the range of values in each plot.

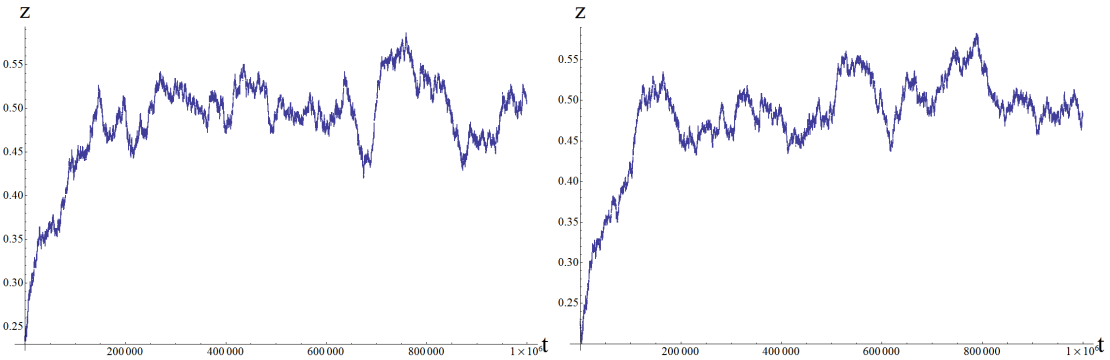


(a) $\mu = 10^{-3}$ (b) $\mu = 10^{-3}$ (c) $\mu = 10^{-3}$

(d) $\mu = 10^{-2}$ (e) $\mu = 10^{-2}$ (f) $\mu = 10^{-2}$

(g) $\mu = 10^{-1}$ (h) $\mu = 10^{-1}$ (i) $\mu = 10^{-1}$

Figure 28: Scatter plots of the largenet2 simulations for the $\tau\sigma$-DAN model. Color reflects d-value, renormalized to the range of values in each plot.



(a) $\tau\sigma$-DAN Model (b) $\tau\rho$-DAN Model

Figure 29: Time series for the stochastic simulations of the two Teaching models for $\phi = 0.72$ and $\mu = 0$. Note the fluctuations in the $\tau\sigma$-DAN model are not significantly more extreme than the $\tau\rho$-DAN model, indicating the limit cycle observed in the mean field model is not revealed in the stochastic version.

7 Discussion

Previous work on adaptive network opinion formation models have focused on the balance between imitation and segregation processes, but has generally ignored noise. In this thesis, we expanded two models previously published by Nardini *et al* [1] as the direct and reverse Voter Models with a noise parameter (μ) which corresponds to independent opinion change (innovation). We also investigated two addition and analogous models that utilize a different and novel rewiring mechanism. We corroborated past work on the difference between different types of imitation (here called learning and teaching); these were found - as in previous work - to form fields of opposite directions in the variable that measures opinion density in the network.

We discovered that changing the rewiring mechanism also dramatically changes the dynamics of the mean field. The previously investigated rewiring mechanism - the egocentric segregation mechanism - tends to rewire more discordant links to concordant links of the majority opinion than the minority. Meanwhile our new variant - altercentric segregation - tends to rewire more discordant links to the minority opinion than the majority. These additional links indirectly affect an advantage in the z-field, which we have seen modifies the bifurcation diagrams' topology. As has been previously shown with teaching and learning processes, we see the dramatic difference which slight differences in microscopic mechanism can confer on macroscopic dynamics of the mean field. However, unlike the differences between the learning and teaching mechanism however, this difference is largely not revealed in the stochastic simulations, which for the most part appear qualitatively similar under substitution of rewiring mechanism.

Unsurprisingly perhaps, all the distinguishing features of the four models under investigations we fragile to the noise parameter μ. With innovation of the order 10^{-1} we saw all four models qualitatively indistinguishable; all non-trivial attractors disappear and the models all exhibit $z = 0.5$ as the sole attractor of the dynamics, in both the mean field and the stochastic models. In Sociocultural Evolution, μ corresponds to mutation; it is not entirely inconceivable that in many cases, a fraction $\frac{1}{10}$ of opinion formation processes are independent of neighbors.

As has been noted in previous work [7], adaptivity seems to somewhat improve the fit of mean field models of network dynamics. High dependence on initial network structure for low rewiring rates compounds with a tendency of imitation processes to act against themselves in a way unaccounted for in the mean field model. Together, these make dynamics at low segregation rates converge the slowest in the stochastic model, while in the mean field these are some of the quickest to converge. It is clear that this is due to random walk like behavior that is not accounted for in the mean field, and brings to question the drawing of conclusions about convergence time in stochastic network models based on their mean field.

Another notable discrepancy between the mean field and the stochastic model was elucidated by the perturbation of μ. $\{z = 0, d = 0, c = 0\}$ and $\{z = 1, d = 0, c = k/2\}$ are expected to be stable attractors in the learning models for low ϕ in the stochastic model; however in the mean field models these equilibria are hidden inside an invariant manifold and thus are not Lyapunov stable. This seems to be due to an inability of the master equation to capture the *combinatorial* constraints

prevalent in the discrete stochastic model. This brings into question the validity of naïve analysis of stability and phase change in Markov random fields using mean field approximations.

8 Conclusion

In comparing different combinations of imitation and segregation processes in this class of discordian adaptive network models of opinion formation, we have seen that seemingly trivial decisions about mechanism can make a qualitative difference of the dynamics. In some cases - such as the difference between teaching and learning - this difference is found in both the mean field and the stochastic model. In other cases - such as the difference between egocentric and altercentric segregation - these differences are most pronounced in the mean field but seem to make little to no difference in the stochastic models. We have also seen how adding a noise parameter quickly obscures all the differences between the models even in the mean field model. Additionally we have noted several points in which the mean field models fail to capture critical elements of their stochastic equivalents: attractive equilibria in the stochastic version may not be Lyapunov stable in the mean field, and convergence times can diverge when we make a mean field approximation.

These points brings into question the naïve application of mean field theory to adaptive networks, and demands that comprehensive analysis of Markov random fields should incorporate an understanding of the evolution of the whole probability distribution and not just the mean field. Finally we see that this class of opinion formation models seems exceedingly fragile to perturbations. Future investigations might wish to seek more robust models, and placing an emphasis on the complete description of the stochastic model of adaptive networks seems desirable.

References

[1] C. Nardini, B. Kozma, A. Barrat (2008). Who's Talking First? Consensus or Lack Thereof in Coevolving Opinion Formation Models. *Phys. Review Letters* **100**, 158701

[2] T. Gross, C.J. Dommar D'Lima, B. Blasius (2006). Epidemic Dynamics on an Adaptive Network. *Phys. Review Letters* **96**, 208701

[3] S. Mandrà, S. Fortunato, C. Castellano (2009) Coevolution of Glauber-like Ising dynamics and topology. *Physical Review E* **60**, 056105

[4] B. Schmittmann, A. Mukhopadhyay (2010). Opinion formation on adaptive networks with intensive average degree. *Physical Review E* **82**, 066104

[5] R. Durrett, S.A. Levin (1994) Stochastic Spatial Models: A User's Guide to Ecological Applications. *Phil. Trans.: Biological Sciences* **343**, 1305

[6] T. Gross, B. Blasius (2007). Adaptive coevolutionary networks: a review. *J. R. Soc. Interface* **5**, 259-271

[7] A.-L. Do, T. Gross (2009). Contact Processes and Moment Closure on Adaptive Networks. *Understanding Complex Systems* **2009**, 191-208

[8] G. Zschaler, T. Gross (2013). Largenet2: an object-oriented programming library for simulating large adaptive networks. *Bioinformatics*, **29(2)**. 277-278.

[9] E.J. Doedel, B.E. Oldeman (2011) AUTO-07P: Continuation and bifurcation software for ordinary differential equations *(software documentation)* http://indy.cs.concordia.ca/auto/

[10] M. Taylor, P.L. Simon, D.M. Green, T. House, I.Z. Kiss (2011). From Markovian to pairwise epidemic models and the performance of moment closure approximations. *J. Math. Biol.*

[11] D.A. Rand (1999). Correlation Equations and Pair Approximations for Spatial Ecologies. *CWI Quarterly* **12**, 329

[12] T. House, M.J. Keeling (2010) The impact of contact tracing in clustered populations. *PLoS Comput. Biol.* **6(3)**, e1000721

[13] M.J. Keeling (1999). The effects of local spatial structure on epidemiological invasions. *Proc. R. Soc. Lond. B* **266**, 859-867

Acknowledgments

I would like to thank the following individuals, without whom this thesis would not have been possible: My supervisor - *Prof. Joachim Hermisson* - for his persistent patience and assistance despite my idiosyncratic choices in this thesis. *Dr. Gerd Zschaler* for his extensive support with his Largenet2 simulation library. *Prof. Sebius Doedel* for his extensive support with his AUTO 07p continuation software. *Dr. Thilo Gross* for his advice, in particular for pointing me to the Largenet2 library he coauthored. Pun-dit *Piter '000' Pasma* for his incredible help and advice in debugging and cleaning up the Largenet2 simulation code. *Dr. Thijs Ruijgrok* - my Bachelor's Thesis supervisor - for having nudged me in the direction of network theory in the first place. *Mario Ebenwalder* for his help in translating the abstract into German. *Roger Gómez Ortelles*, *Frederik Amann*, *Fabian Burkes*, and *Mario Ebenwalder* for their warm and welcoming hospitality during my vagabond days in Wien. Last but not least, my dear mother *Nicolette Van Zuiden* for her love and extensive support during my studies in general, and in the last year in particular.

I would also like to thank: my family, for their love and support throughout my entire education. *Jürgen Harreiter* for his support, wise words and escapades; and for leading my wandering path to Wien. *Fabian Burkes*, *Joscha Krauss*, *Roger Gómez Ortelles* and *Helene Weigang* for all their love and support - and for all the wild adventures we shared in Wien. Finally I would like to thank the spags at *#discord* for all the (in)sanity and the fnords they provided.

A AUTO 07p Code

A.1 The Equations File `xdan.f90`

```
!---------------------------------------------------------------------
!---------------------------------------------------------------------
!  Discordian Adaptive Network Equations File for AUTO 07p
!---------------------------------------------------------------------
!---------------------------------------------------------------------

      SUBROUTINE FUNC(NDIM,U,ICP,PAR,IJAC,F,DFDU,DFDP)
!     ---------- ----

      IMPLICIT NONE
      INTEGER, INTENT(IN) :: NDIM, ICP(*), IJAC
      DOUBLE PRECISION, INTENT(IN) :: U(NDIM), PAR(*)
      DOUBLE PRECISION, INTENT(OUT) :: F(NDIM)
      DOUBLE PRECISION, INTENT(INOUT) :: DFDU(NDIM,NDIM), DFDP(NDIM,*)

      DOUBLE PRECISION Z, D, H, Mu, Phi, Psi, Nu, N, k

      Z=U(1)
      D=U(2)
      H=U(3)

      Mu=PAR(1)
      Phi=PAR(2)
      Psi=PAR(3)
      Nu=PAR(4)
      N=PAR(5)
      k=PAR(6)

!       The Differential Equations

      F(1)= (1-Mu)*(1-Phi)*(2*Psi-1)*(D/N)*((1-Z)/(K-D-2*H)-Z/(D+2*H))+(Mu/N)*(1-2*Z)

      F(2)= (1-Mu)*( (1-Phi)*Psi*(D/N)*((H-D-Z)/(D+2*H)-0.5*(4*D-K+2*H+2*(1-Z))/(K-D-2*H)) &
       +(1-Phi)*(1-Psi)*(D/N)*((H-D-Z)*(1-Z)/(Z*(K-D-2*H))-0.5*(4*D-K+2*H+2*(1-Z))*Z/((1-Z)*(D+2*H))) &
       -Phi*Nu*(D/N)*( (1-Z)**2/(K-D-2*H) + Z**2/(D+2*H) )  &
       -Phi*(1-Nu)*(D/N)*Z*(1-Z)*(1/(K-D-2*H) + 1/(D+2*H)) ) + (Mu/N)*(K-4*D)

      F(3)= (1-Mu)*( (1-Phi)*Psi*(D/N)*((D+1-Z)/(K-D-2*H)-H/(D+2*H))  &
       +(1-Phi)*(1-Psi)*(D/N)*(Z*(D+1-Z)/((1-Z)*(D+2*H))-(1-Z)*H/(Z*(K-D-2*H)))  &
       +Phi*Nu*(D/N)*Z**2/(D+2*H) + Phi*(1-Nu)*(D/N)*Z*(1-Z)/(K-D-2*H) ) +(Mu/N)*(D-2*H)

      END SUBROUTINE FUNC

      SUBROUTINE STPNT(NDIM,U,PAR,T)
!       Initial Parameter Values

      IMPLICIT NONE
      INTEGER, INTENT(IN) :: NDIM
      DOUBLE PRECISION, INTENT(INOUT) :: U(NDIM),PAR(*)
      DOUBLE PRECISION, INTENT(IN) :: T

       PAR(1)= 1.0D-2   ! Mu
       PAR(2)= 0.0      ! Phi
       PAR(3)= 1.0      ! Psi   >> Imitation:   1=Learning      0=Teaching
       PAR(4)= 0.0      ! Nu    >> Rewiring:    1=Egocentric    0=Altercentric
       PAR(5)= 1000.    ! N
       PAR(6)= 5.       ! K

       U(1)=0.95
       U(2)=PAR(5)*PAR(6)*U(1)*(1-U(1))/(PAR(5)-1)
       U(3)=U(1)*PAR(6)*(PAR(5)*U(1) -1)/(2*(PAR(5)-1))

      END SUBROUTINE STPNT

      SUBROUTINE BCND
      END SUBROUTINE BCND

      SUBROUTINE ICND
      END SUBROUTINE ICND

      SUBROUTINE FOPT
      END SUBROUTINE FOPT
```

```
      SUBROUTINE PVLS(NDIM,U,PAR)
!       These Control Params record Z, D and C and permit the diagram
!       to stop at the boundries. See UZSTOP in the Constants File.

      IMPLICIT NONE
      INTEGER, INTENT(IN) :: NDIM
      DOUBLE PRECISION, INTENT(IN) :: U(NDIM)
      DOUBLE PRECISION, INTENT(INOUT) :: PAR(*)
      DOUBLE PRECISION, EXTERNAL :: GETP

      PAR(7)=GETP('MIN',1,U)            ! Minimum of Z
      PAR(8)=GETP('MAX',1,U)            ! Maximum of Z
      PAR(9)=GETP('MIN',2,U)/PAR(6)     ! Minimum of D
      PAR(10)=GETP('MAX',2,U)/PAR(6)    ! Maximum of D
      PAR(11)=GETP('MIN',3,U)/PAR(6)    ! Minimum of C
      PAR(12)=GETP('MAX',3,U)/PAR(6)    ! Maximum of C

      END SUBROUTINE PVLS
```

A.2 The Constants File `c.xdan`

```
########################################################
# Discordian Adaptive Network - AUTO 07p Constants File #
########################################################


#NAMES#
parnames = {1: 'Mu', 2: 'Phi', 3: 'Psi', 4: 'Nu', 5: 'N', 6: 'k', 7:'Zn', 8:'Zx', 9:'Dn', 10:'Dx',
11:'Cn',12:'Cx'}
unames = {1: 'z', 2: 'd', 3: 'c'}

#OUTPUT#
ICP =  ['Phi'], sv = 'D0'

#LIMITS#
RL0 = -0.001, RL1 = 1.001
UZSTOP = {1: [-0.005, 1.005], 7: -0.005, 8: 1.005, 9: -1.500, 10: 0.5005,  11: -0.0005, 12: 0.5005}


#PROBLEM CONSTANTS#
IPS =   1, IRS =   0, ILP =   0

NPAR = 36, JAC= 0, NDIM=   3, ISW = -1

#CONTINUATION#
DS  =   0.0001, DSMIN=  0.0001, DSMAX=   0.001, IADS=   1


#OTHER#

NTST=  35, NCOL=   4, IAD =   3, ISP =   2, IPLT= 0, NBC= 0, NINT= 0

NMX=   10000, NPR=  5000, IID =   2, ITMX= 8, ITNW= 7, NWTN= 3

EPSL= 1e-07, EPSU = 1e-07, EPSS = 1e-05

THL =  {}, THU =  {}
```

B Largenet2 Code

B.1 Simulation Loop File `xdan-sim.cpp`

```
/**
 * xdan-sim.cpp
 * a largenet2 based simulation of Discordian Adaptive Networks
 *
 * This is a simulation of a class of simple models of sociocultural evolution.
 *
 * A network of @p N nodes, representing agents, and @p L undirected links, representing social
 * connections, is created. Each agent has a state representing an opinion, D = Down or U = Up.
 * The following parameters govern the probability rates for transitions per node :
 *
 * @p Mu:  Innovation Rate - prob. rate of node independently switches state
 * @p Phi: Rewiring Rate - relative prob. rate of Rewiring (Phi) to Imitation (1-Phi)
 * @p Psi: Learning Type - 1 = Learning         0 = Teaching
 * @p Nu:  Rewiring Type - 1 = Egocentric       0 = Altercentric
 *
 * Detailed mechanics of the processes are found in xdan.h
 *
 * @author Matan Bendix Shenhav <m.shenhav@gmail.com>
 *
 * Loosely based on sis.cpp - an example provided with largenet2 and written by Gerd Zschaler.
 */

#include "xdan.h"

#include "../lib/WELLEngine.cpp"
#include "../lib/WELLEngine.h"
#include "../lib/RandomVariates.h"

#include <largenet2.h>
#include <largenet2/StateConsistencyListener.h>
#include <largenet2/generators/generators.h>

#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <memory>
#include <limits>
#include <math.h>
#include <sys/timeb.h>


using namespace std;
using namespace largenet;


// Iterated parameter vector - these paramaters get iterated through a range of values in the
simulation.
class Params {
        public:
                double Phi;
                double Mu;

                Params(){}
                Params(double phi, double mu) { Phi=phi; Mu=mu;}

} ;

class NetParams {
        public:
                double Psi;
                double Nu;
                unsigned int N;
                unsigned int L;
                NetParams(){}
                NetParams(double psi, double nu, unsigned int n, unsigned int l) { Psi=psi; Nu=nu;
N=n; L=l; }

} ;

// Fixed parameter vector - these parameters are fixed throughout the simulation
class SimParams {
        public:
                unsigned int Runs;
```

```
                double RunTime;
                double OutInt;
                double InCond;
                double PhiMin;
                double PhiMax;
                double PhiStep;
                double MuMin;
                double MuMax;
                double MuStep;
                SimParams(){}
                SimParams(unsigned int runs, double runtime, double outint, double incond, double
phimin, double phimax, double phistep, double mumin, double mumax, double mustep) {Runs= runs;
RunTime= runtime; OutInt = outint; InCond= incond; PhiMin= phimin; PhiMax=phimax; PhiStep=phistep;
MuMin=mumin; MuMax=mumax; MuStep=mustep;}

} ;

// gives time is miliseconds, used to seed the RNG

int milli(){
        timeb tb;
        ftime(&tb);
        unsigned long int nCount = tb.millitm + (tb.time & 0xfffff) * 1000;
        return nCount;
}

// adds filename extention to title of simulation

char* lab(string title){
        stringstream spag;
        spag << title << ".raw.dat";
        const std::string fnord= spag.str();
        char *trip=new char[fnord.size()+1];
        trip[fnord.size()]=0;
        memcpy(trip,fnord.c_str(),fnord.size());
        return trip;
} ;



// a random number generator (WELL1024a)

typedef myrng::RandomVariates<myrng::WELLEngine> rng_t;



void simcore(string name, Params &sP, NetParams &sNP, SimParams &sSP, rng_t &R)
{

        // define our own model type for DAN using rng_t
        typedef DAN<rng_t> model_t;

        // Model parameters
        model_t::Params params = {sP.Phi, sP.Mu, sNP.Psi, sNP.Nu};

        //

        double initial_infected= sSP.InCond;

        // Create a Graph object with DAN::node_states possible node
        // states and DAN::link_states possible link states
        Graph net(model_t::node_states, model_t::link_states);

        // StateConsistencyListener object ensures Edge-States match Node-States.

        StateConsistencyListener<model_t::EdgeStateCalculator> scl(
                        auto_ptr<model_t::EdgeStateCalculator>(
                                        new model_t::EdgeStateCalculator));
                                        net.addGraphListener(&scl);

        // Connect the nodes randomly to form a random, undirected Erdős-Rényi
        // type network.
        generators::randomGnm(net, sNP.N, sNP.L, R, false);
```

```
        // Iterate over all nodes in the network
        Graph::NodeIteratorRange nodes = net.nodes();
        for (Graph::NodeIterator n = nodes.first; n != nodes.second; ++n)
                // Set node state to model_t::S for each node
                net.setNodeState(n->id(), model_t::D);

        const node_size_t initiallyInfectedNodes = static_cast<node_size_t>(round(
                        net.numberOfNodes() * initial_infected));

        // Set the first initiallyInfectedNodes' node states to SISModel::I
        // Note that this is fine, since the links among the nodes have been placed
        // randomly, i.e., disregarding the node order.
        Graph::NodeIterator n = nodes.first;
        while (net.numberOfNodes(model_t::U) < initiallyInfectedNodes)
        {
                net.setNodeState(n->id(), model_t::U);
                ++n;
        }

        // Now, all the edges in the network have been assigned appropriate states
        // by the StateConsistencyListener we supplied. Thus, an edge from an D-node
        // to an U-node will be in state UD, etc.

        // Create a DAN object that uses the Graph object, works with
        // the specified model parameters, and uses the provided random number
        // generator
        model_t model(net, params, R);

        // output file stream
        ofstream dataout(lab(name),ios::app);

        // Simulation loop
        // Perform model steps until simulation time exceeds maximum
        double t = 0, i= 0;
        double interval = sSP.OutInt; // output interval
        while (t <= sSP.RunTime)
        {
                // only write output at specified intervals
                if ( i == interval )
                {

                        dataout << t << "\t" << net.numberOfNodes(model_t::U)
                                     << "\t" << net.numberOfEdges(model_t::UD)
                                     << "\t" << net.numberOfEdges(model_t::UU)
                                     << "\n";

                        i =0;
                }

                // One simulation step
                model.step(R);
                t += 1;
                i += 1;
        }


        dataout.close();


        return;
}

int main(int argc, char **argv)
{

        /// Simulation Name ///

        string simname= "xdan.B.1.0";

        /// Network Parameters that define the model ///

        NetParams NP;

        /// Network Type ///
```

```
NP.Psi = 1;             ///< Psi: Learning Type:        0= Teaching     1= Learning
NP.Nu  = 0;             ///< Nu: Rewiring Type:         0= Altercentric 1= Egocentric

/// Network Size ///

NP.N = 1000;            ///< N: the number of nodes
NP.L = 2500;            ///< L: the number of edges  (note: mean degree k = 2L/N)

/// Simulation Parameters define the scope of the simulation ///

SimParams SP;

SP.InCond = 0.55;       ///< InCond: Initial fraction of individuals with + state

/// Temporal Scope ///

SP.Runs = 100;          ///< Runs: the number of times the sim is repeated per parameter value

SP.RunTime = 30000;     ///< RunTime: the number of timesteps for each simulation run
SP.OutInt  = 1000;      ///< OutInt: output interval; data is printed every OutInt steps

/// Parameter Space Scope ///

SP.PhiMin  = 0.0;       ///< PhiMin: lower bound on Phi values explored
SP.PhiMax  = 1.0;       ///< PhiMax: upper bound on Phi values explored
SP.PhiStep = 0.05;      ///< PhiStep: resolution on exploration of Phi values

SP.MuMin  = 0.0;        ///< MuMin: initial Mu value
SP.MuMax  = 4.0;        ///< MuMax: values will be explored from 1e-MuStep to 1e-MuMax
SP.MuStep = 1.0;        ///< MuStep: the exponent of Mu is raised by MuStep each round


/// Iterated Parameters ////

Params P(SP.PhiMin,SP.MuMin);  /// Initialize parameters

// Simulation Initialization

// Compute total number of Threads and Runs

int NPhi, NMu, TotRuns, TotThrds;
NPhi = (floor((SP.PhiMax - SP.PhiMin)/SP.PhiStep)+1);
NMu = (floor((SP.MuMax-SP.MuMin)/SP.MuStep)+1);
TotThrds = NPhi*NMu;
TotRuns = TotThrds*SP.Runs;

// print simulation details

cout << "Initiating XDAN Simulation, Sequence Codename: " << simname << "\n";
cout << "This simulation will have a total of " << TotThrds
     << " threads, and " << TotRuns << " Runs." << "\n";

remove(lab(simname)); // delete old file

// write output file header

ofstream data_out(lab(simname),ios::app);
data_out << " " << "\n";
data_out << "// Discordian Adaptive Network Simulation // Codename: " << simname << "\n";
data_out << "Psi:"  << "\t" << "Nu:" << "\t" << "Phi-Scope:" << "\t" << "Mu-Scope:" << "\t"
         << "Runs:" << "\t" << "Thrds:" << "\t" << "Time:" << "\t" << "Interval:" << "\n";
data_out << NP.Psi << "\t" << NP.Nu
         << "\t" << "{" << SP.PhiMin << "," << SP.PhiMax << "," << SP.PhiStep << "}"
         << "\t" << "{" << SP.MuMin  << "," << SP.MuMax  << "," << SP.MuStep  << "}"
         << "\t" << SP.Runs << "\t" << TotThrds << "\t" << SP.RunTime << "\t" << SP.OutInt
         << "\n";
data_out << " " << "\n";
data_out.close();


rng_t rng;              // declare RNG
rng.seed(milli());      // Seed RNG
```

```
        int Count=0;            // this is counts Threads
        double MuExp = 0;       // the exponent of Mu


        // Simulation Loop

        for(int m=1; m <= NMu; m++){                    // Mu Loop

        P.Phi = SP.PhiMin;

                for(int p=1; p <= NPhi; p++){           // Phi Loop

                        for(int i=1; i <= SP.Runs; ++i){                        // Run Loop


                                ofstream data_out2(lab(simname),ios::app);      // Print run header

                                data_out2 << " " << "\n";
                                data_out2 << "Phi:" << "\t" << "Mu:" << "\t" << "Run:" << "\n";
                                data_out2 << P.Phi << "\t" << P.Mu << "\t" << i << "\n";
                                data_out2 << " " << "\n";
                                data_out2 << "t" << "\t" << "z" << "\t" << "d" << "\t" << "c" << "\n";

                                data_out2.close();

                                simcore(simname,P,NP,SP,rng);                   // execute Run

                        };

                        P.Phi = SP.PhiMin+ p*SP.PhiStep;                        // increment Phi

                        Count++;
                        cout << "Thread " << Count << " of " << TotThrds << " complete." << "\n";
                };

        MuExp = -m*SP.MuStep;
        P.Mu = pow(10.0,MuExp);                                                 // increment Mu

        };

        ofstream data_out3(lab(simname),ios::app);
        data_out3 << " " << "\n" << " " << "\n" << " " << "\n" << " " << "\n" << " " << "\n";
        data_out3.close();


}
```

B.2 Simulation Header File `xdan.h`

```
/**
 * xdan.h
 * The XDAN class for the Discordian Adaptive Networks simulator
 * for more information, see xdan-sim.cpp
 *
 * @author Matan Bendix Shenhav <m.shenhav@gmail.com>
 * Based on SISModel.h - an example provided with largenet2 and written by Gerd Zschaler.
 */

#ifndef DAN_H_
#define DAN_H_

#include <largenet2.h>
#include <largenet2/sim/gillespie/DirectMethod.h>
#include <boost/bind.hpp>
#include "../lib/util.h"

template<class RandomGen>
class DAN
{
private:
        typedef DAN<RandomGen> self_type;
public:
        static const largenet::node_state_size_t node_states = 2; ///< number of node states
        static const largenet::edge_state_size_t link_states = 3; ///< number of edge states

        struct Params
        {
                double phi;     ///< Rewiring Rate:     1=All Rewiring  0=No Rewiring
                double mu;      ///< Innovation:        Mutation Prob.
                double psi;     ///< Learning Type:     1= Learning     0= Teaching
                double nu;      ///< Rewiring Type:     1= Egocentric   0= Altercentric
        };

        /// Names for node states
        enum NodeState
        {
                U, D
        };
        /**
         * Names for edge states
         *
         * Note that the edge states will be computed automatically by the library from the
         * states of the two nodes an edge connects by the largenet::StateConsistencyListener,
         * which makes use of the supplied DAN::EdgeStateCalculator.
         *
         * @see largenet2/StateConsistencyListener.h
         */
        enum LinkState
        {
                UU, UD, DD
        };

        /**
         * An edge state calculator
         *
         * This computes appropriate edge states from source and target node states.
         */
        class EdgeStateCalculator
        {
        public:
                EdgeStateCalculator()
                {
                }
                /**
                 * Compute edge state from given node states
                 * @param s source node state
                 * @param t target node state
                 * @return edge state
                 */
                largenet::edge_state_t operator()(const largenet::node_state_t s,
                                const largenet::node_state_t t) const
                {
                        if (s != t)
```

```
                        return UD;
                else if (s == U)
                        return UU;
                else
                        return DD;
        }
};


/**
 * Constructor
 * @param net Graph instance to use for the simulation
 * @param p DAN model parameters
 * @param rng Random number generator to use for the simulation
 */
DAN(largenet::Graph& net, Params p, RandomGen& rng) :
net_(net), par_(p), rng_(rng)
{
        // Register the different processes with the Gillespie stepper.
        // Each process is registered with a rate function which computes
        // the instantaneous reaction rate for the process and an "action"
        // function which performs the action of the process. Note how
        // the convenient use of boost::bind allows to use DAN member
        // functions for this.
        stepper.registerProcess(
                        boost::bind(&self_type::innovationRate, this),
                        boost::bind(&self_type::innovate, this)
                        );
        stepper.registerProcess(
                        boost::bind(&self_type::learningRate, this),
                        boost::bind(&self_type::learn, this)
                        );
        stepper.registerProcess(
                        boost::bind(&self_type::teachingRate, this),
                        boost::bind(&self_type::teach, this)
                        );
        stepper.registerProcess(
                        boost::bind(&self_type::ECrewireRate, this),
                        boost::bind(&self_type::ECrewire, this)
                        );
        stepper.registerProcess(
                        boost::bind(&self_type::ACrewireRate, this),
                        boost::bind(&self_type::ACrewire, this)
                        );
}

/**
 * Check whether the simulation has stopped
 *
 * Returns true when no further updates are possible (i.e. the absorbing
 * state of zero infectious nodes has been reached).
 */
bool stopped()
{
        if (par_.mu == 0.0)
                return net_.numberOfEdges(UD) == 0;
        else
                return false;
}

/**
 * Perform one simulation step
 * @param rng Random number generator, providing a method IntFromTo(int, int)
 * that returns a random integer between the specified boundaries.
 * @return time increment
 */
double step(RandomGen& rng)
{
        return stepper.step(rng);
}

private:

        double innovationRate() const
```

```
        {
                return par_.mu;
        }
        double learningRate() const
        {
                return (1.0-par_.mu)*(1.0-par_.phi)*par_.psi;
        }
        double teachingRate() const
        {
                return (1.0-par_.mu)*(1.0-par_.phi)*(1.0-par_.psi);
        }
        double ECrewireRate() const
        {
                return par_.nu*(1.0-par_.mu)*par_.phi;
        }
        double ACrewireRate() const
        {
                return (1.0-par_.nu)*(1.0-par_.mu)*par_.phi;;
        }


/// the innovation function
/// random node n1 flips its opinion

        void innovate()
        {
                largenet::Node* n1 = net_.randomNode(rng_);   // pick random node n1
                if (net_.nodeState(n1->id()) == U)
                        net_.setNodeState(n1->id(), D);       // change to the opposite state
                else
                        net_.setNodeState(n1->id(), U);
        }

/// the learning function
/// random node n1 adopts random neighbor n2's opinion

        void learn()
        {
                largenet::Node* n1 = net_.randomNode(rng_);   // pick random node n1

                if (0 == n1->undirectedDegree())              // do nothing if it has no neighbors
                        return ;
                else
                {                                             // pick a random neighbor n2
                        largenet::Node::edge_iterator ei;
                        ei = myrng::util::random_from(n1->undirectedEdges(), rng_);
                        largenet::Edge* e = *ei;
                        largenet::Node* n2 ;

                        if (e->source()->id()==n1->id())
                                n2 = e->target();
                        else
                                n2 = e->source();

                                                              // n1 adopts the state of n2
                        if (net_.nodeState(n1->id()) == net_.nodeState(n2->id()))
                                return ;
                        else
                                net_.setNodeState(n1->id(), net_.nodeState(n2->id()));
                }

        }

/// the teaching function
/// random node n1 convinces random neighbor n2 of its opinion

        void teach()
        {
                largenet::Node* n1 = net_.randomNode(rng_);   // pick random node n1

                if (0 == n1->undirectedDegree())              // do nothing if it has no neighbors
                        return ;
                else
                {                                             // pick a random neighbor n2
```

```
                        largenet::Node::edge_iterator ei;
                        ei = myrng::util::random_from(n1->undirectedEdges(), rng_);
                        largenet::Edge* e = *ei;
                        largenet::Node* n2 ;

                        if (e->source()->id()==n1->id())
                                n2 = e->target() ;
                        else
                                n2 = e->source() ;

                                                        // n2 adopts the state of n1
                        if (net_.nodeState(n1->id()) == net_.nodeState(n2->id()))
                                return ;
                        else
                                net_.setNodeState(n2->id(), net_.nodeState(n1->id()));
                }

        }

/// the egocentric rewiring function
/// random node n1 picks a random neighbor n2:
/// if n1 and n2 differ in opinion - n1 rewires its link with n2 to random node n3

        void ECrewire()
        {
                largenet::Node* n1 = net_.randomNode(rng_);

                if (0 == n1->undirectedDegree())
                        return ;
                else
                {
                        largenet::Node::edge_iterator ei;
                        ei = myrng::util::random_from(n1->undirectedEdges(), rng_);
                        largenet::Edge* e = *ei;
                        largenet::Node* n2 = (e->source()->id() == n1->id()) ?
                                         e->target() : e->source() ;
                                if (net_.nodeState(n1->id()) == net_.nodeState(n2->id()))
                                        return ;
                                else
                                {
                                        net_.removeEdge(e->id());
                                        largenet::edge_size_t old_edges = net_.numberOfEdges();
                                        unsigned int tries = 0;
                                        const unsigned int max_tries = net_.numberOfNodes();
                                        largenet::Node* n3 = net_.randomNode(rng_);
                                        while (tries < max_tries)
                                        {
                                                // disallow self-loops
                                                if (n1->id() == n3->id())
                                                {
                                                        n3 = net_.randomNode(rng_);
                                                        ++tries;
                                                        continue;
                                                }

                                                // add undirected edge
                                                net_.addEdge(n1->id(), n3->id(), false);

                                                // if this edge is already in the graph, try again
                                                if (net_.numberOfEdges() == old_edges)
                                                {
                                                        n3 = net_.randomNode(rng_);
                                                        ++tries;
                                                        continue;
                                                }
                                                else
                                                        break;
                                        }
                                }

                }
        }

/// the altercentric rewiring function
```

```
/// random node n1 picks a random neighbor n2:
/// if n1 and n2 differ in opinion - n2 rewires its link with n1 to random node n3

        void ACrewire()
        {
                largenet::Node* n1 = net_.randomNode(rng_);

                if (0 == n1->undirectedDegree())
                        return ;
                else
                {
                        largenet::Node::edge_iterator ei;
                        ei = myrng::util::random_from(n1->undirectedEdges(), rng_);
                        largenet::Edge* e = *ei;
                        largenet::Node* n2 = (e->source()->id() == n1->id()) ?
                                         e->target() : e->source() ;
                                if (net_.nodeState(n1->id()) == net_.nodeState(n2->id()))
                                        return ;
                                else
                                {
                                        net_.removeEdge(e->id());
                                        largenet::edge_size_t old_edges = net_.numberOfEdges();
                                        unsigned int tries = 0;
                                        const unsigned int max_tries = net_.numberOfNodes();
                                        largenet::Node* n3 = net_.randomNode(rng_);
                                        while (tries < max_tries)
                                        {
                                                // disallow self-loops
                                                if (n2->id() == n3->id())
                                                {
                                                        n3 = net_.randomNode(rng_);
                                                        ++tries;
                                                        continue;
                                                }

                                                 // add undirected edge
                                                net_.addEdge(n2->id(), n3->id(), false);

                                                // if this edge is already in the graph, try again
                                                if (net_.numberOfEdges() == old_edges)
                                                {
                                                        n3 = net_.randomNode(rng_);
                                                        ++tries;
                                                        continue;
                                                }
                                                else
                                                        break;
                                        }
                                }

                }
        }


private:
        largenet::Graph& net_;
        Params par_;
        sim::gillespie::DirectMethod stepper;
        RandomGen& rng_;
};

#endif /* DAN_H_ */
```

C Curriculum Vitae

Personalia

Name:	Matan Bendix Shenhav
Nationality:	Dutch / Israeli
Date of birth:	04-12-1987
Driving Liscence	European Driving License B
Address:	Kroosmeent 8, 1218BA Hilversum
Mobile number:	06-22.7676.98
E-mail:	m.shenhav@gmail.com

Education

2009-2013:	**MSc Mathematics, focus: Biomathematics** University of Vienna, Austria
2005-2009:	**BSc in Mathematics and Physics** University of Utrecht, the Netherlands
2003-2005:	**International Baccalaureate Diploma Program** International School Hilversum, Nederland

Scientific Background

(Bio)Mathematics:	Probability Theory Stochastic Processes Dynamical Systems Theory Network Theory and Mean-Field Theory Mathematical Ecology and Evolutionary Game Theory Mathematical Population Genetics and Coalescent Theory
Biology and Economics:	Theories of Sociocultural en Genetic Evolution Behavioral Biology and Economics Theorical Ecology and Biodiversity
Miscellaneous:	Philosophy of Science

Work experience

2010-2012:	Tutoring in mathematics, physics, biology and chemisty.
2004-2013:	Freelance graphic design *Illustrations, logos, print-work and small websites*

Additional skills

Language skills:	**English** - Excellent **Dutch** - Fluent **Hebrew** - Proficient **German** - Proficient
Computer skills:	**LaTeX** - Excellent **C++ and F90** - Basic **Mathematica** - Proficient **MATLAB** - Basic **SPSS** - Basic **Windows XP/7** - Excellent **Linux** - Basic **Adobe Illustrator** - Excellent **Adobe Photoshop** - Proficient **Adobe Indesign** - Proficient **HTML and CSS** - Proficient **MS Office** - Proficient

Overige

International background:	12 years in Israel, 5 months in the USA, 10 years in the Netherlands 1 year in Sweden and 3.5 years in Austria.
Hobbies:	Following developments in science and technology, (computer) drawing, reading, writing, conversation, cooking, travel, walking, dancing and music.