Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

REGRESSION-BASED SOCIAL INFLUENCE NETWORKS AND THE LINEARITY OF AGGREGATED BELIEF

Michael J. Garee
Hong Wan

School of Industrial Engineering
Purdue University
315 North Grant St.
West Lafayette, IN, 47907, USA

Wai Kin Victor Chan

Environmental Science and New Energy
Technology Engineering Laboratory
Tsinghua-Berkeley Shenzhen Institute
Shenzhen 518055, P. R. CHINA

ABSTRACT

Consider an agent-based social influence (belief adoption) network where agents share beliefs with neighbors using a linear regression model. One relevant question is: can aggregated, system-level belief also be fit by a linear regression model? Earlier work demonstrated several scenarios where system-level linearity of belief holds. This paper extends that research, varying model and simulation factors through experimental design. When linearity does not hold, we isolate the responsible factors. Finally, we investigate whether system-level linearity is as an absorbing state, that is, when system-level linearity is present at some time *t*, it continues to hold for all later times.

1 INTRODUCTION

Agent-based simulation is a useful tool for building and analyzing social influence networks—systems in which agents exchange information with their neighbors and influence each other's levels of belief over time. In this study, agents update their beliefs via a linear model, based on a weighted sum of their neighbors' beliefs and modified by internal bias and white noise. Similar update functions are used elsewhere in social network analysis (Jackson 2010) and sensor network consensus modeling (Touri and Nedic 2009). We build system-level measures of influence by aggregating values from each agent. One reasonable hypothesis is that when agents interact in a linear way, the system-level measures may also respond in a linear fashion. Chan (2017) finds that for a particular network configuration, this idea is valid. To examine this hypothesis more completely, we build upon the previous work by varying model and simulation elements through experimental design, identifying factors that impact the linearity of system-level belief, and exploring whether individual observations of system-level linear behavior make good predictors of steady-state activity.

In this research, we choose to focus only on linear models and behaviors. Linear systems are widely used in literature and in practice, so we seek to explore their validity as social influence network models. However, we do not make any claims to the importance of linear systems to this topic, nor wish to imply that linear systems are more or less valid than non-linear ones.

We identify ten factors that affect the structure and properties of the network, the schedule used for agent updates, and the settings that control the agent's initial states. We then build a Nearly Orthogonal Latin Hypercube design to systematically study the behavior of the system under different factor settings. The key constant across all trials is the agents' use of a linear model to update their beliefs. We find that linear agent interactions in most cases do not generate linear system-level responses. Given the ten experimental factors in our design, three factors can significantly hinder whether we observe linear responses, and five factors have little to no effect for the range of levels used. However, no single factor on its own is observed to absolutely prevent system-level linear responses. Also, we observe that the degree to which aggregated belief can be fit by a linear model can vary over time (i.e., linearity is not an absolute absorbing state in general), so measuring linear behavior for some single time t does not make for a perfect predictor of future performance.

The rest of this paper is organized as follows. We provide a brief review of the literature related to our topic in Section 2. In Section 3, we develop the model. In Section 4, we describe the experimental design for the study. We present our analysis methods and results in Section 5, and we provide conclusions and discuss future work in Section 6.

2 BACKGROUND

2.1 Social Influence and Learning Networks

The terms contagion and social influence are often used interchangeably to describe the process of altering behavior or belief due to communication and comparison among actors in a social system (Leenders 2002). Social learning is the process of "learning through observation or interaction with other individuals" (Rendell et al. 2010). In social network analysis, learning tends to be used when agents in influence networks seek an optimal behavior or true belief.

Social learning in network analysis is broadly divided into diffusion models and information aggregation models (Banerjee et al. 2016). Diffusion looks at the spread of information through a population; information aggregation focuses on convergence of opinions. The Bass model is a straightforward model that describes binary adoption of a belief or behavior without explicitly using the network structure (Jackson 2010). DeGroot (1974) developed a simple linear updating model to describe information aggregation. There, agents begin with initial estimated beliefs, and all agents update simultaneously, replacing their current level of belief with a weighted average of their neighbors' belief levels and their own. Agents will converge to a consensus value if the network structure meets certain conditions of aperiodicity and communication (DeGroot 1974; Golub and Jackson 2010).

A second way of dividing social learning is into Bayesian models and DeGroot models. Bayesian models focus on learning by observing the actions of neighbors and the payoffs they receive, while DeGroot models learn myopically from communicating and processing only the current system state (Acemoglu et al. 2011). The DeGroot model remains a seminal model of information transmission and social learning analysis (Golub and Sadler 2016; Banerjee et al. 2016; Chandrasekhar et al. 2015).

2.2 Regression Analysis

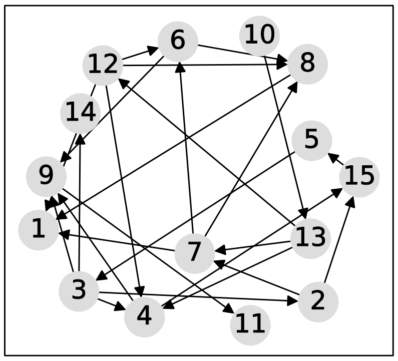
Regression analysis is popular in social influence studies. Here, we comment on several recent examples. Mavrodiev et al. (2013) studied indirect social influence in a sequential decision making experiment with humans. Participants had access to the mean of all previous decisions, but did not interact directly with other individuals. The authors found a statistically significant fit for a linear regression model relating the amount individuals changed their decision over time and the distance between their previous decision and the current mean. Similarly, Cheng et al. (2015) used logistic regression on opinion data from a Taiwanese online bulletin board. Their results showed that users are more likely to post comments that match the sentiment (approval or disapproval) in recent posts, while users are indifferent to the average sentiment of the entire history of comments. Finally, Chan (2017) modeled a social influence network where agents interact using linear regression equations. For the particular network configurations used, he found that the aggregated system-level belief could be well-described using a linear regression model. Those findings are a key motivation for the present paper.

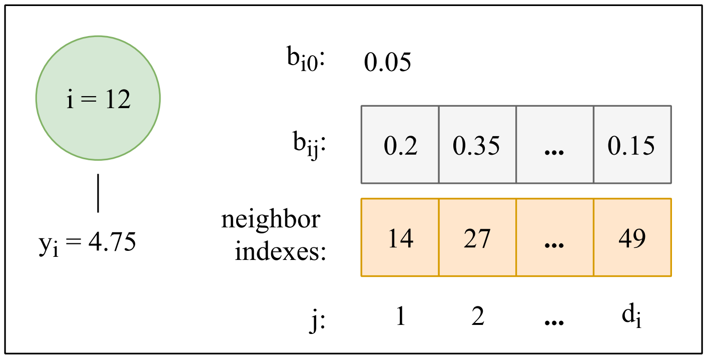
3 MODEL

3.1 Networks and Agents

We use agent-based simulation to model a social influence network with N agents (nodes). Each agent is connected to one or more other agents using directed edges; self-loops are not permitted. The degree of each agent and the distribution of degree across the network depend on the network structure model family (e.g. scale-free, random, etc.) for a given trial. We use only static network structures for this study, so the set of agents and their edges do not change during a run of the simulation (Figure 1a). Particular network instances are a function of the structure family, network parameters, and randomness, and we use an assortment of network instances in our experimental design (Section 4).

Agents are indexed $i = 1, 2, \ldots, N$. Agent i's neighbors are the agents that receive out-edges from i, so neighbor relationships are not reciprocal. Each agent keeps a list of its neighbor indexes, sorted in ascending order. The list of neighbors is indexed by $j = 1, 2, \ldots d_i$, where d_i is the out-degree of agent i (Figure 1b). Agents also track their own current level of belief y_i, internal bias b_{i0}, and multipliers b_{ij} for neighbor j's belief, which can be thought of as weights for the network edges. As a practical example, y_i may be a person's current opinion of a political topic, b_{i0} is their intrinsic or baseline opinion that cannot be changed by others, and b_{ij} is the weight the person places on the opinions held by their friends. When we need to explicitly compare these values between different time steps, we use superscript (t) to index them by time step t, as in $y^{(t)}$.





(a) (b)

Figure 1: (a) Example of directed Erdős-Rényi random graph with 15 nodes. Arrows point from an agent to an agent's neighbor. This structure model allows for a different number of neighbors for each agent. (b) Data structures inside an agent. This example shows the agent with index $i = 12$.

3.2 Simulation Procedure

First we explain how updates are scheduled from the system-level perspective, then we describe interactions at the agent level, and finally we provide pseudocode for the simulation algorithm. Code for this study is written in Python 3; noteworthy packages include the agent-based simulation framework Mesa (Masad and Kazil 2015), the NetworkX package for graph structures and algorithms (Hagberg et al. 2008), and Statsmodels for regression analysis (Seabold and Perktold 2010). Complete code and other supporting files for this study are available in our online appendix (Garee 2018).

Our simulation uses a scheduling approach we call batched simultaneous update. We divide the population into batches of equal size. At each time step, batches are updated one at a time in either a fixed or random sequence. When a batch is updated, all agents in the batch update simultaneously. Simultaneous update takes two steps: first, all agents in an updating batch compute their new belief value y_i but store it in a temporary variable, then all agents in the batch update their y_i with the value of their temporary variable. This two-step approach ensures that the update sequence within a batch does not matter, though the sequence in which batches are visited may affect the outcome. Once all agents in the population update, their belief values may be normalized by dividing them by the sum of all y_i values, if the trial settings call for normalization of y_i. This batched update mechanism mimics a real-world situation in which people first exchange ideas within a small group of acquaintances and then extend the idea exchange to other groups.

Agents are assigned to batches $1, 2, \dots, n_b$ uniformly at random, independent of the network structure. If $n_b = 1$, all agents are in the same batch, so the full population updates simultaneously (as in the original DeGroot model). If $n_b = N$, all agents are in different batches (of size 1), so the full population updates sequentially.

Some trials create *uninformed agents* that have zero initial belief. If an agent is uninformed, it updates its belief value only if it has one or more informed neighbors, at which time it becomes informed permanently and updates normally. This concept can appear in information diffusion models.

Agents interact only with their neighbors. Agent i computes its level of belief y_i using the linear regression model

$$y_i^{(t)} = b_{i0} + \sum_{j=1}^{d_i} b_{ij} x_{ij}^{(t)} + \varepsilon_{\mathrm{i}}^{(t)}, \tag{1}$$

where $\varepsilon_i^{(t)}$ is a random error term generated each time step and $x_{ij}^{(t)}$ is the current belief value for agent i's jth neighbor (i.e. if the jth neighbor of agent i has index k, then $x_{ij} = y_k$), and the other terms are as defined previously in Section 3.1. The simulation procedure for a single trial is described in Table 1.

Table 1: Simulation procedure for a single trial.

```
Trial Initialization:  Construct network from structure model family and parameters

For each replication of trial:
    Replication Initialization:
        Generate $b_{i0}$, $b_{ij}$, and $\varepsilon_i^{(0)}$ for each agent
        Set $y_i^{(0)} = b_{i0} + \varepsilon_i^{(0)}$ as initial level of belief

    For each time step $t$:
        For each batch:
            For each agent $i$ in batch:
                Generate $\varepsilon_i^{(t)}$
                Set temporary variable for new belief using Eq. 1

            For each agent $i$ in batch:
                Set $y_i^{(t)}$ equal to temporary variable

        Set $Y^{(t)} = \sum_{i=1}^{N} y_i^{(t)}$
        If using $y_i$ normalization, set $y_i^{(t)} = y_i^{(t)} / Y^{(t)}$ for each agent $i$

        Collect time-step level data
    Collect system-level data
```

3.3 Response Variables

Our focus in this study is to explore linearity in the system-level belief, achieved with a multiple linear regression model of the form

$$Y^{(t)} = B_0^{(t)} + B_1^{(t)}\mathrm{X}_1^{(\mathrm{t})} + B_2^{(t)}\mathrm{X}_2^{(\mathrm{t})} + \cdots + B_{d^*}^{(t)}X_{d^*}^{(t)} + E^{(t)}$$

where $Y^{(t)}$ is the system-level belief, $X_j^{(t)}$ represents the aggregated belief of neighbor j across the network, for $j = 1, 2, \ldots, d^*$; d^* is the maximum out-degree over all agents in the network; B_0 is the system's internal bias; B_j is the multiplier for the aggregation of neighbor j's beliefs; and $E^{(t)}$ is the random error term. These values are captured each time step, so they are indexed by time. These interpretations are based on (Chan 2017). The aggregated values $Y^{(t)}$ and $X_j^{(t)}$ are the dependent and independent terms, respectively, in our regression model and are our key response variables obtained from the simulation:

$$Y^{(t)} = \sum_{i=1}^{N} y_i^{(t)} \qquad \text{and} \qquad X_j^{(t)} = \sum_{i=1}^{N} x_{ij}^{(t)}\,[j \le d_i].$$

(Since d_i can vary between agents, $x_{ij}^{(t)}$ may not be defined for all values of j for some agents. The Iverson bracket $[j \le d_i]$ resolves this.) The set of all $X_j^{(t)}$ terms for a single time step form the vector $\mathbf{X}^{(t)}$. Together, $Y^{(t)}$ and $\mathbf{X}^{(t)}$ make up a single observation of system-level responses for the time step.

We validated our simulation framework with two tests. First, we built a DeGroot model (1974) and observed that the aggregated system-level belief converged as expected. Then, we replicated Chan's (2017) model and obtained qualitative agreement with his system-level regression results.

4 EXPERIMENTAL DESIGN

In this paper, a *factor* is an input variable that may have an impact on the responses, *levels* are values a factor may be assigned, a *trial* is a combination of levels for each experimental factor (one row from the design matrix, also known as a design point), and a *replication* is one repetition of a trial using different initial randomization settings. We run a replication for 500 time steps and replicate each trial 100 times. The ten experimental factors we use for this study affect network characteristics, update scheduling, and agent interaction. These factors and their associated levels are:

1. Number of agents N. 100, 500, or 1000.
2. Network structure instance. For each level of N, we create 14 structure instances, discussed below.
3. Number of batches n_b. 1, 5, $N/4$, or N. For our chosen levels of N, we ensure $N \bmod n_b = 0$.
4. Update sequence. Batches are updated each time step in either a fixed or random sequence.
5. Distribution of b_{ij} coefficients. Uniform(0, 1), Uniform(-1, 1), or Normal(0, 1).
6. b_{i0} coefficients. Initialize using the distribution for b_{ij} or set to zero to remove internal agent bias.
7. Normalize b_{i0} and b_{ij} coefficients. Within each agent, across the population, or do not normalize.
8. Variance of error terms. Error terms ε_i are sampled from the normal distribution with mean zero and variance σ^2 of 0.5, 1.0, or 2.0.
9. Normalize y_i each time step. Yes or No.
10. Fraction of uninformed agents. 0, 0.05, or 0.25. This fraction of agents have y_i set to zero and are flagged as *uninformed* at the start of the run.

Each factor affects one or more high-level features of the simulation that may influence the linearity of system-level responses. Factors 1 and 2 affect the size and shape of the network, while Factors 3 and 4 control agent update scheduling. Factors 5 to 8 govern the regression terms for the update equation (1) within each agent. Factor 9 alters the scale of belief values, and Factor 10 lets us see how diffusion of initial belief affects the results.

A network structure instance is a particular realization of a network model for a given structure family, input parameters for that family (including network size N), and a randomization seed, when applicable. The network structure families we use are scale-free, directed random tree, Erdős-Rényi random graph, and random k-out. Two to four instances from each family are made using several sets of input parameters and randomization seeds. We select inputs that produce a satisfactory range of values for the standard network measures of mean out-degree centrality, assortativity, reciprocity, and efficiency.

A full factorial crossed design for this experiment is written as $2^3 \times 3^5 \times 4^1 \times 14^1$ and requires a costly 108K trials. Instead, we adopt a data farming view and use a nearly orthogonal Latin Hypercube (NOLH) design; this choice is motivated by Sanchez and Wan (2015). The NOLH design tool we use (Sanchez 2011) reduces our experiment to 255 trials. The final experimental design matrix uses all ten factors and their associated levels defined earlier and is available with our online appendix (Garee 2018).

5 ANALYSIS & RESULTS

5.1 Data Processing

The first stage of analysis is to convert simulation outputs into a suitable analysis database. Figure 2 gives a schematic overview of our data processing activity. To process a single trial, we group the simulation outputs $\left(Y^{(t)}, \mathbf{X}^{(t)}\right)$ by time step from across all replications of that trial. Within each time step, data cleaning removes outliers (e.g. due to floating point error or very extreme values) based on the standard 1.5-IQR (interquartile range) rule and drops observations with null Y values (e.g. caused by Y growing to infinity). Using the remaining observations, we fit a main-effects only linear regression model of the form described in Section 3.3, record the model's adjusted R^2 $\left(R^2_{adj}\right)$, and assess R^2_{adj} as *Significant* if the model's p-value is below 0.10 or as *Not Significant* otherwise. This is done for each time step of the trial.

R^2_{adj} and the significance rating can move unpredictably between time steps due to randomness, so we apply to both parameters a smoothing function and the prefix "MA" for moving average. For R^2_{adj}, we use a five-time step simple moving average on R^2_{adj} (the arithmetic mean of the R^2_{adj} values for the last five time steps) and call the result MA-R^2_{adj}. For the smoothed significance rating, which we call MA-Sig., we assess the rating at each time step as the rating of the most recent sequence of length three or greater: Over time, as we observe three Significant time steps in a row (based on R^2_{adj}), we begin assigning MA-Sig. as Significant, until we observe three Not Significant time steps in a row and switch to assigning MA-Sig. as Not Significant, and so on, switching back and forth as needed.

MA-R^2_{adj} and MA-Sig. drive a function that classifies each time step as *Linear*, *Not Linear*, or *Invalid*. The Invalid classification is applied if either MA-R^2_{adj} or MA-Sig. is undefined, which seems to occur when Y or elements of $\mathbf{X}$ grow too large or have infinite variance, or when the residual degrees of freedom for the model is too low. (The maximum degree of the network structure governs the number of elements in $\mathbf{X}$ and number of factors in the regression model, so trials with large networks and highly-connected agents require more replications to potentially be valid.) We also classify cases where MA-R^2_{adj} is exactly equal to unity as Invalid; rather than indicating a perfect fit for the regression model on the simulation output, this occurs for some trials with very extreme Y values, many outliers, or otherwise problematic data. A valid time step is Linear if MA-$R^2_{adj} \geq 0.50$ and MA-Sig. is Significant. Otherwise, it is classified as Not Linear. (Note: Linear and Not Linear are defined only with respect to our goal of fitting time step data with a model having the same functional form as the update equation used by the agents. We acknowledge that there may exist linear models that strongly fit data that we classify as Not Linear, but they must contain explanatory factors outside the current scope.) The values for MA-R^2_{adj}, MA-Sig., and Classification for each time step, and for each trial, populate the analysis database, which we use in concert with the experimental design matrix for all subsequent analysis.

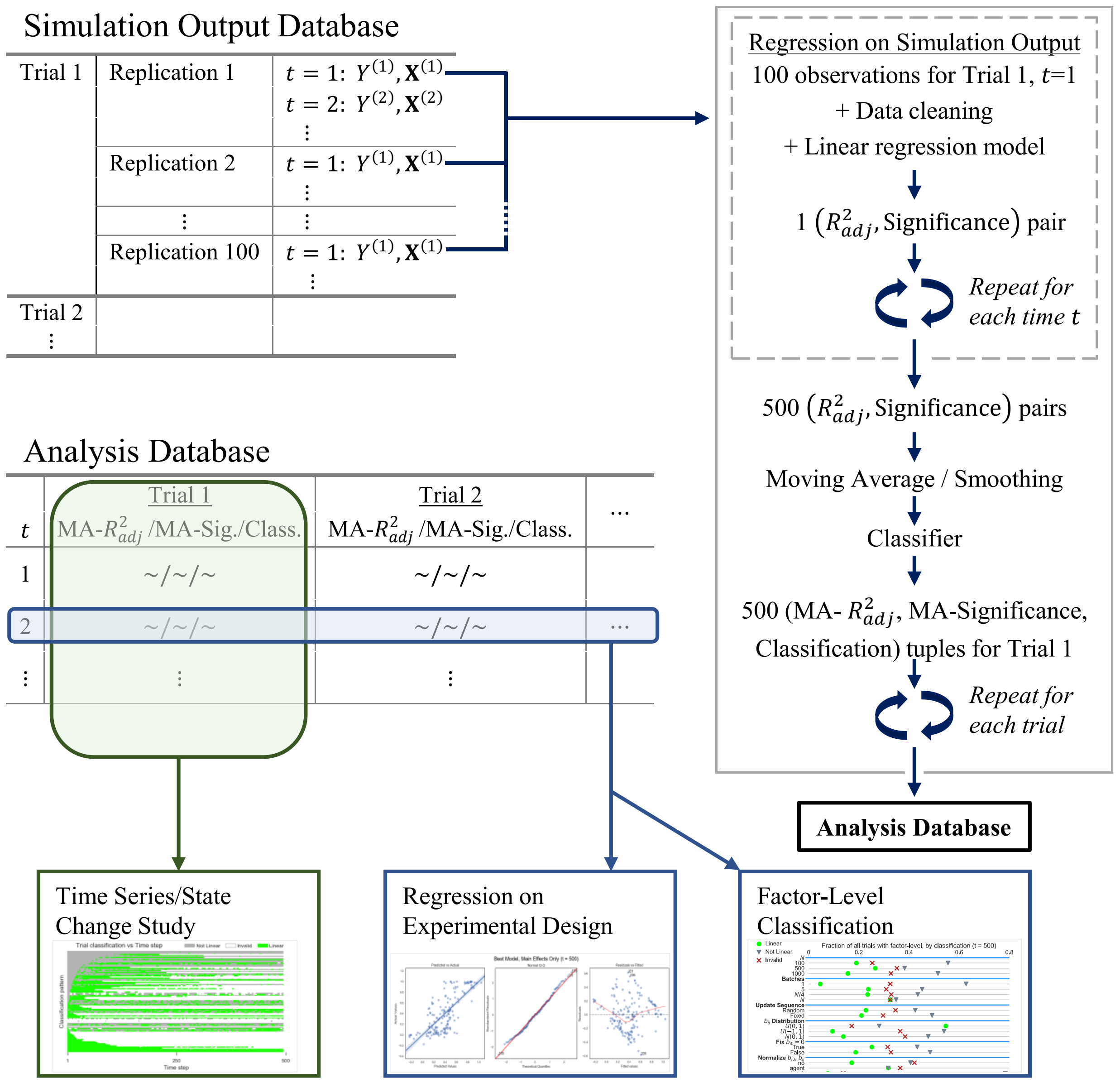

Simulation Output Database
Trial 1
Replication 1
t = 1: Y(1), X(1)
t = 2: Y(2), X(2)
Replication 2
t = 1: Y(1), X(1)
Replication 100
t = 1: Y(1), X(1)
Trial 2
Regression on Simulation Output
100 observations for Trial 1, t=1
+ Data cleaning
+ Linear regression model
1 (R²adj, Significance) pair
Repeat for each time t
500 (R²adj, Significance) pairs
Moving Average / Smoothing
Classifier
500 (MA- R²adj, MA-Significance, Classification) tuples for Trial 1
Repeat for each trial
Analysis Database
Analysis Database
t
Trial 1
MA-R²adj /MA-Sig./Class.
Trial 2
MA-R²adj /MA-Sig./Class.
1
2
~/~/~
Time Series/State Change Study
Regression on Experimental Design
Factor-Level Classification

Figure 2: Schematic for data processing and analysis. Each independent replication of a trial yields one observation of the system $\left(Y^{(t)}, \mathbf{X}^{(t)}\right)$ for every time step t. For each time step, the set of observations are cleaned and put into a linear regression model of the form defined in Section 3.3 to produce an R^2_{adj} value and Significance rating based on the p-value. Due to randomness, these values can move erratically between time steps, so we apply a smoothing function and the prefix "MA" for moving average. The smoothed data classifies each time step as Linear, Not Linear, or Invalid, ending data processing and creating the Analysis Database. This processed data is sliced per trial for time series studies, or per time step to support regression analysis on our experimental design and finding classification rates for each factor-level in the design matrix.

5.2 Analysis

Our first two analysis products, factor-level classification and experimental design regression, consider the system at a single time step (t = 500), while the third is a time series study. First, for each factor-level in the experimental design, we filter the analysis database by trials containing that factor-level and calculate the fraction of trials with each of the three classifications (Linear, Not Linear, and Invalid). This provides an intuitive way of making qualitative assessments of how the factors affect system-level linearity. A dot plot of this data (Figure 3) lets us to make several observations:

- Updating the population in a single batch is significantly worse for linearity; N batches are best.
- Randomizing the update sequence of batches has little effect on linearity.
- Generating b_{ij} values from the Uniform(0, 1) distribution produces linear results significantly more often than when we use Uniform(-1, 1) or Normal(0, 1), which both have mean of zero.
- Normalizing all b_{i0} and b_{ij} values per agent yields the most linear trials.
- Normalizing y_i may be beneficial because it reduces the chance of a trial being classified as invalid.

We do not include the network structure in this part of the analysis, because each structure instance is a direct function of N and also has a very low sample size (10–20).

Next, we build regression models at t = 500 on the experimental design using MA-R^2_{adj} as the dependent variable and omitting any Invalid data points. (The R^2_{adj} of these models is only indirectly related to the R^2_{adj} from the raw simulation output that underpins MA-R^2_{adj}.) The initial main effects model with the ten original design factors performs adequately (R^2_{adj} = 0.513). We slightly improve on this baseline model by proxying the network structure factor, a categorical variable, with the maximum degree d_i of the network, a discrete variable. Surprisingly, if we replace the categorical levels 1, 5, $N/4$, and N for number of batches with the discrete number of batches n_b, we find a significantly worse model fit. The model we select as best contains only five factors: the categorical batch quantity level, the distribution function for b_{ij}, whether we normalize y_i, the way we normalize b_{i0} and b_{ij}, and the maximum degree d_i of the network (a proxy factor for network structure instance). This model's R^2_{adj} is 0.532 and is statistically significant. All factors are significant and have low variance inflation factors. Table 2 summarizes the model evolution process, and Figure 4 contains diagnostic plots of the best regression model. Full model results are available in the online appendix, but we provide a brief interpretation here:

- Increasing the number of batches causes MA-R^2_{adj} to monotonically increase, but the rate of increase falls off rapidly above $n_b = 5$.
- Selecting Uniform(0, 1) for the b_{i0} and b_{ij} distribution is associated with higher values of MA-R^2_{adj}.
- Normalizing b_{i0} and b_{ij} values per agent instead of population-wide or not at all is linked to higher MA-R^2_{adj}.
- Normalizing y_i and the maximum network degree are statistically significant but have very minor coefficients. Removing these factors from the model lowers R^2_{adj} to 0.507, so we instead retain them.

These observations agree strongly with our earlier comments on factor-level classification percentages. We finish our regression analysis by constructing a model with main effects and two-way interaction terms. One of the highest-quality interaction models we found uses a slightly different set of design factors, created from the main effects model by removing maximum degree d_i and adding whether the update sequence is randomized, the variance of the error distribution, and whether b_{i0} is set equal to zero. Backward elimination is used to selectively remove low p-value interaction terms, yielding a statistically significant model with R^2_{adj} = 0.754. Diagnostic plots of this model are similar to Figure 4 but with tighter distributions.

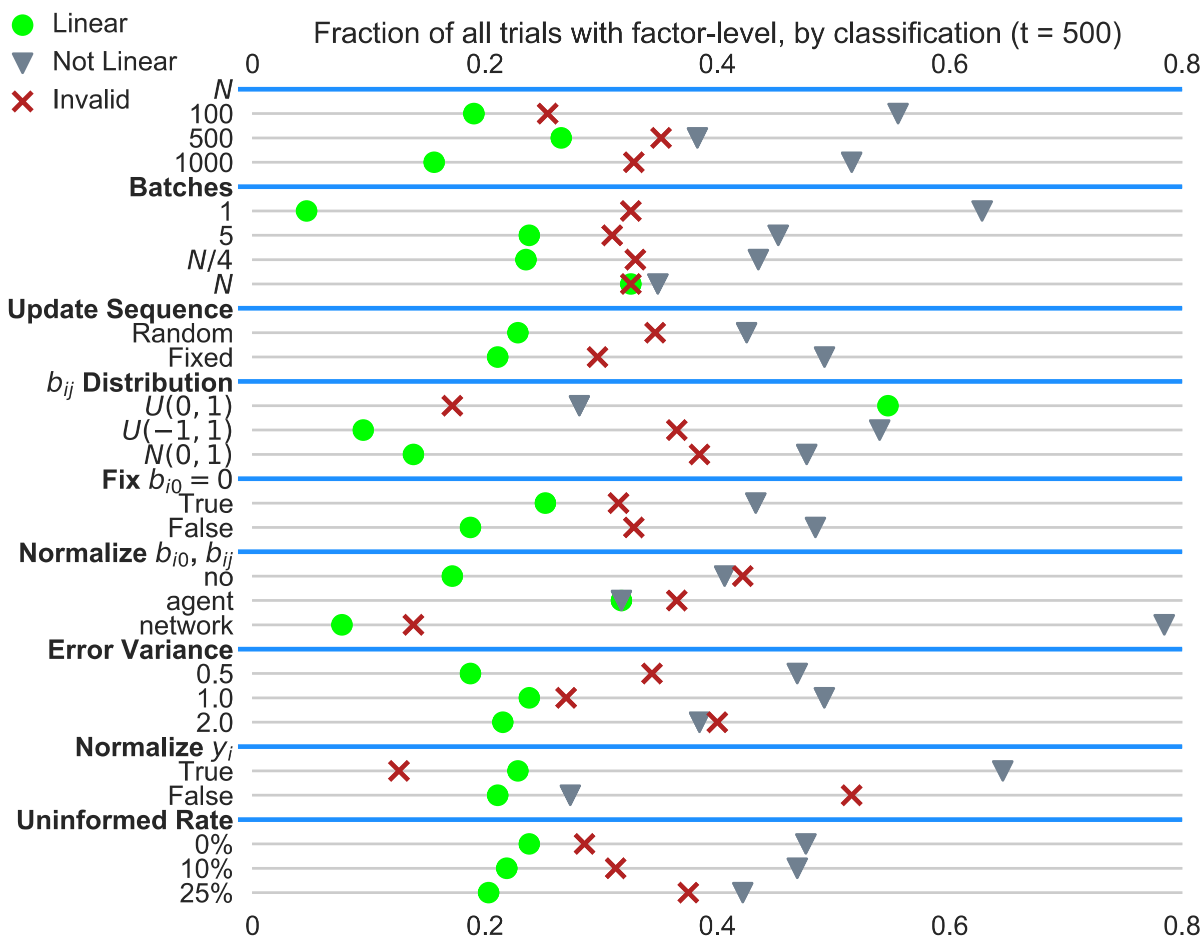


Figure 3: Dot plot of fraction of trials with each classification per level for each factor at $t = 500$. Some observations from this include: the update sequence setting has little effect on linearity, but moderate effect on trial validity; using only one batch is detrimental to linearity; and normalizing y_i leads to more valid—but not linear—trials. The network structure factor is omitted from this plot.

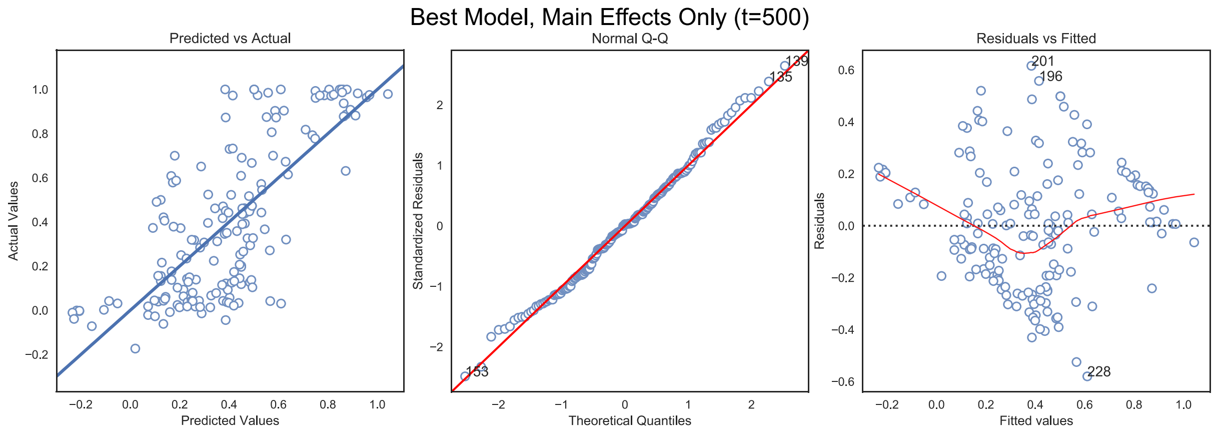


Figure 4: Diagnostic plots of main effects only model of experimental design regressed on MA-R^2_{adj}. This model uses five of the ten experimental factors and achieves $R^2_{adj} = 0.532$.

Table 2: Summary of regression models fit to experimental design with dependent variable MA-R^2_{adj} at time step $t = 500$. All results in the table are statistically significant (p-value < 0.001).

Regression Model on Experimental Design ($t = 500$)	R^2_{adj}
Main Effects Only	
Baseline: All 10 original factors	0.513
Proxy network structure (categorical) with max d_i (discrete)	0.533
Proxy batch quantity level (categorical) with batch quantity (discrete)	0.388
Best fit: baseline, omit N and randomize update sequence, use max d_i (8 factors)	0.538
Best model: batch quantity level, b_{ij} dist., normalize b_{i0} & b_{ij}, normalize y_i, max d_i (5 factors)	0.532
Main Effects and Two-Way Interactions	
Baseline: All 10 original factors & all interactions – input rank exceeds observation count	n/a
All 10 factors, proxy network structure (categorical) with max d_i (discrete)	0.739
Best fit with all interactions: baseline, omit N and uninformed rate, proxy network structure	0.761
Best model: 7 factors, proxy network structure, backward eliminate interaction terms	0.793

Finally, we consider how trial classification (Linear, Not Linear, or Invalid) changes over time, and whether linearity continues in the future once it appears. Some trials have identical classification patterns and can be merged, reducing the data from 255 trials to 94 patterns. This data reveals that the classifications of a small number of time steps do not help predict long-run activity (Figure 5). A single Linear time step can be part of a trial where the future is always Linear, where the system moves between classifications frequently, or as a random Linear time step as part of a mostly Not Linear trial.

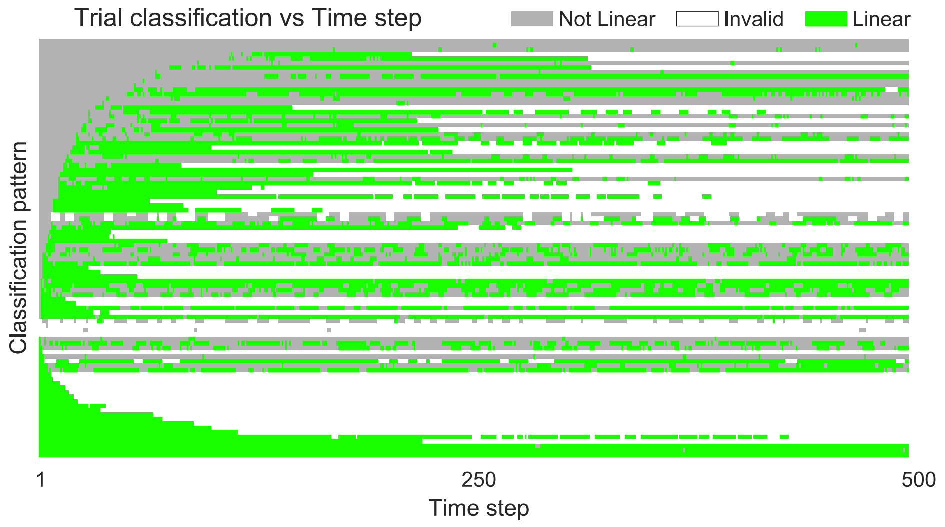


Figure 5: Trials are classified Linear, Not Linear, or Invalid at each time step based on MA-R^2_{adj} and MA-Sig., and trials with identical classification patterns are merged. This figure shows the 94 distinct classification patterns (rows), sorted left to right by time step (columns). The solid green bar at the bottom of the figure represents trials that are Linear for every time step (40 trials). Single time step observations do not predict future performance.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we use experimental design to build a collection of agent-based social influence networks populated by agents that exchange belief via a linear regression model. We then investigate the behavior of the aggregated, system-level belief using state transition probabilities, regression analysis, and a custom classifier of trial linearity. The hypothesis is that linear agent-level interactions would lead to linear system-level responses, but we identify several model features that challenge this assumption.

Features that negatively affect system-level linearity include updating the full agent population simultaneously as a single batch, generating agent's b_{ij} coefficients from a zero-mean probability distributions, and choosing not to normalize b_{i0} and b_{ij} values per agent. In real-world influence networks, people generally exchange ideas within small groups of acquaintances over time, so multiple update batches are more similar to reality. Also, the trust that real people have in their acquaintances (i.e. b_{ij}) is unlikely to be zero, so a zero-mean distribution may be inappropriate—if people trust their neighbors, the whole system may be more well-behaved. Therefore, our experimental results suggest that actual influence networks may tend more toward linearity.We observe no factor-level that completely prevents linearity on its own. Half of our experimental factors have no real effect on classification for the range of levels we selected, namely the population size, randomizing the batch update sequence, fixing (or not) agents' internal bias b_{i0} to zero, error variance, and the initial uninformed rate of agents. Testing these factors over a greater range of levels could justify omitting them from later experiments. Normalizing y_i each time step is valuable not for affecting linearity but rather for reducing trial invalidation. These findings may help analysts design more effective simulations in the future. Lastly, if we wish to predict long-run behavior with respect to classifying a system as linear, evaluating single time steps is inadequate. Further study is required to understand how long-run behavior relates to the experimental design factors.

Many rich research areas remain that can build upon this work through small changes to the existing model, as the structure of our simulation allows us to easily test more intricate scenarios. We assess system-level linearity based strictly on regression models of the same functional form as the agent's interaction equation, but relaxing this definition and bringing new model features into the analysis may reveal new insights. Allowing self-loops in the network structure would see agents factoring their current belief value into the update process; applying an autoregressive-type model may be interesting. It could be worthwhile to explore the effect that linearity has on model behavior metrics such as speed of belief adoption. Instead of updating the full population each time step, agents could update stochastically or subject to conditions about their neighbors (e.g. homophily/thresholding, where agents ignore opinions too different from their own). A more substantial change to our simulation would be to have dynamic network structures, creating and destroying links over time. Finally, additional network metrics in the experimental design regression analysis could shed light on the influence of network structure on system-level linearity of belief.

REFERENCES

Acemoglu, D., M. A. Dahleh, I. Lobel, and A. Ozdaglar. 2011. "Bayesian Learning in Social Networks". *The Review of Economic Studies* 78(4):1201-1236.

Banerjee, A., E. Breza, A. G. Chandrasekhar, and M. Mobius. 2016. "Naïve Learning with Uninformed Agents". Working paper.

Chan, W. K. V. 2017. "Agent-Based and Regression Models of Social Influence". In *Proceedings of the 2017 Winter Simulation Conference* 1395-1406, edited by W.K.V.Chan et al., Piscataway, New Jersey: IEEE.

Chandrasekhar, A. G., H. Larreguy, and J. P. Xandri. 2015. "Testing Models of Social Learning on Networks: Evidence from a Lab Experiment in the Field". Working Paper 21468, National Bureau of Economic Research.

Cheng, S.L., W.H. Lin, F. K. H. Phoa, J.S. Hwang, and W.C. Liu. 2015. "Analysing the Unequal Effects of Positive and Negative Information on the Behaviour of Users of a Taiwanese On-Line Bulletin Board". *PLoS ONE* 10(9):e0137842.

DeGroot, M. H. 1974. "Reaching a Consensus". *Journal of the American Statistical Association* 69(345): 118-21.

Garee, M. 2018. "Online Appendix, Garee WSC 2018". Available via https://github.com/mgaree/wsc2018.

Golub, B. and M. O. Jackson. 2010. "Naïve Learning in Social Networks and the Wisdom of Crowds". *American Economic Journal: Microeconomics* 2(1):112-149.

Golub, B. and E. Sadler. 2016. "Learning in Social Networks". In *The Oxford Handbook of the Economics of Networks.* Oxford University Press.

Hagberg A. A., D. A. Schult, and P. J. Swart. 2008. "Exploring network structure, dynamics, and function using NetworkX". In *Proceedings of the 7th Python in Science Conference (SciPy2008)* 11-15. Pasadena, CA.

Jackson, M. O. 2010. *Social and Economic Networks*. Princeton, NJ: Princeton University Press.

Leenders, R. T. A. J. 2002. "Modeling Social Influence through Network Autocorrelation: Constructing the Weight Matrix". *Social Networks* 24(1):21-47.

Masad, D. and J. Kazil. 2015. "Mesa: An Agent-Based Modeling Framework". In *Proceedings of the 14th Python in Science Conference (SciPy 2015)* 53-60. Austin, TX.

Mavrodiev, P., C. J. Tessone, and F. Schweitzer. 2013. "Quantifying the Effects of Social Influence". *Scientific Reports* 3(2013):1360.

Rendell, L., R. Boyd, D. Cownden, M. Enquist, K. Eriksson, M. W. Feldman, L. Fogarty, S. Ghirlanda, T. Lillicrap, and K. N. Laland. 2010. "Why Copy Others? Insights from the Social Learning Strategies Tournament". *Science* 09 Apr 2010:208-213.

Sanchez, S. M. 2011. "NOLHdesigns spreadsheet". Accessed 17.04.2018. Available online via http://harvest.nps.edu/.

Sanchez, S. and H. Wan. 2015. "Work Smarter, Not Harder: A Tutorial on Designing and Conducting Simulation Experiments". In *Proceedings of the 2015 Winter Simulation Conference* 1795-1809, L.Yilmaz et al., Piscataway, New Jersey: IEEE.

Seabold, S. and J. Perktold. 2010. "Statsmodels: Econometric and Statistical Modeling with Python". In *Proceedings of the 9th Python in Science Conference (SciPy 2010)* 57-61. Austin, TX.

Touri, B. and A. Nedic. 2009. "Distributed Consensus over Network with Noisy Links". In *Proceedings of the 12th International Conference on Information Fusion* 146-154. Seattle, WA.

AUTHOR BIOGRAPHIES

MICHAEL J. GAREE is a Ph.D. student in the School of Industrial Engineering at Purdue University. He holds a M.S. in Operations Research from the Air Force Institute of Technology and a B.S. in Physics and Mathematics from Ohio Northern University. His research interests include agent-based simulation, social network simulation, and software engineering. He has been an Air Force officer since 2010. His email address is mgaree@purdue.edu.

WAI KIN (VICTOR) CHAN is Professor of the Tsinghua-Berkeley Shenzhen Institute (TBSI), Tsinghua University, China. He holds a Ph.D. in industrial engineering and operations research from University of California, Berkeley. His research interests include discrete-event simulation, agent-based simulation, and their applications in social networks, service systems, transportation, energy markets, and manufacturing. His e-mail address is chanw@sz.tsinghua.edu.cn.

HONG WAN is Associate Professor in the School of Industrial Engineering at Purdue University. Her research interests include design and analysis of simulation experiments, blockchain simulation and mechanism design, applied statistics, quality management, and healthcare engineering. She is a member of INFORMS and ASA. Her web page is http://web.ics.purdue.edu/hwan and her email address is hwan@purdue.edu.