

An Agent-Based Model of Information Diffusion

Final Presentation

Neza Vodopivec

Applied Math and Scientific Computation Program

nvodopiv@math.umd.edu

Advisor: Dr. Jeffrey Herrmann


Department of Mechanical Engineering

jwh2@umd.edu

Outline

1. Overview of the Model and the Project
2. Implementing and Parallelizing the Simulation
3. Testing: Comparing the Model to Real Twitter Data
4. Validation: Computing Intervals within Which Simulation Points Are Likely to Fall
5. Validation: Convergence of Agent-Based Model to Analytical Bass Model
6. Validation: Comparing Parameters p and q in the Analytical Model to those in the Agent-Based Model
7. Project Summary and Deliverables

Outline

1. Overview of the Model and the Project 
2. Implementing and Parallelizing the Simulation
3. Testing: Comparing the Model to Real Twitter Data
4. Validation: Computing Intervals within Which Simulation Points Are Likely to Fall
5. Validation: Convergence of Agent-Based Model to Analytical Bass Model
6. Validation: Comparing Parameters p and q in the Analytical Model to those in the Agent-Based Model
7. Project Summary and Deliverables

Information Diffusion Simulation: The Bass Model

The Bass model (Bass, 1969), which was originally developed to model the diffusion of new products in marketing, can be applied to the diffusion of information. The model is based on the assumption that people get their information from two sources:



Bass Model Formulation

The Bass model describes the change in the fraction of a population that has become aware of a piece of information:

$$\frac{F'(t)}{1-F(t)} = p + qF(t)$$

$$F(0) = 0,$$

where $F(t)$ is the aware fraction of the population, p is the advertising coefficient, and q is the word-of-mouth coefficient.

An Agent-Based Bass Model

We can formulate an agent-based model inspired by the classical Bass model.

We discretize the problem and make the following modifications:

1. Instead of taking a deterministic time aggregate, we update probabilistically.
2. Instead of allowing each agent to be influenced by the entire population, it is influenced only by its neighbors.

How Information Spreads through the Network

- The agent-based Bass model is a discrete-time model in which each agent has one of two states at each time step t : (1) unaware or (2) aware.
- At time $t=0$, all agents are unaware.
- At each time step, an unaware agent has an opportunity to become aware. Its state changes with P , the probability that it becomes aware due to advertising or due to word of mouth.
- The probability of that an agent becomes aware due to word of mouth increases as a function of the fraction of its neighbors who became aware in previous time steps.
- Once an agent becomes aware, it remains aware for the rest of the simulation.

Probability an Agent Becomes Aware

At each time step, the probability that an unaware agent i becomes aware is:

$$P_i(t) = p \Delta t + q \Delta t [a_i(t) / n_i] - (p q \Delta t^2 [a_i(t) / n_i])$$

Probability that agent becomes aware due to advertising.

Probability that agent becomes aware due to WOM.


Probability that agent becomes aware due to both advertising and WOM.

- n_i is the number of neighbors of agent i .
- $a_i(t)$ is the number of neighbors of agent i that became aware before time t .
- p and q are parameters which indicate the effectiveness of advertising and WOM per unit of time, respectively.

Project Goals

1. Create a fast, memory-efficient implementation of the agent-based Bass model.
2. Validate code to ensure model is correctly implemented.
3. Test model against observed Twitter data. Optimize parameters so the model best fits this data.
4. Analyze how agent-based Bass model behaves when compared to analytical ODE-based Bass model.

Outline

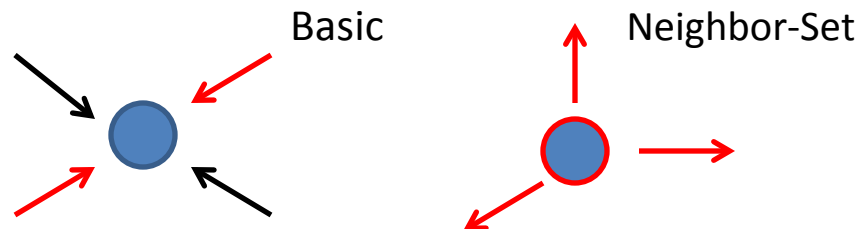
1. Overview of the Model and the Project
2. Implementing and Parallelizing the Simulation 
3. Testing: Comparing the Model to Real Twitter Data
4. Validation: Computing Intervals within Which Simulation Points Are Likely to Fall
5. Validation: Convergence of Agent-Based Model to Analytical Bass Model
6. Validation: Comparing Parameters p and q in the Analytical Model to those in the Agent-Based Model
7. Project Summary and Deliverables

Implementation

- The codebase for the agent-based Bass model was written in MATLAB.
- First, a basic implementation was coded as a reference.
- Then a faster, more memory efficient neighbor set implementation was developed. The neighbor set implementation is based on a more efficient updating rule and takes advantage of sparse data structures.
- These implementations were compared to an existing NetLogo implementation.
- The network structures used for all simulations were obtained from real Twitter follower data.

Neighbor Set Implementation: Using A More Efficient Updating Rule

- In order to decide whether to change the status of an unaware node, the node's number of unaware upstream nodes (its "awareness number") must be computed. The basic implementation effectively recomputes each node's awareness number from scratch at every time step.
- But changes in the awareness number are entirely due to nodes which have just become aware.

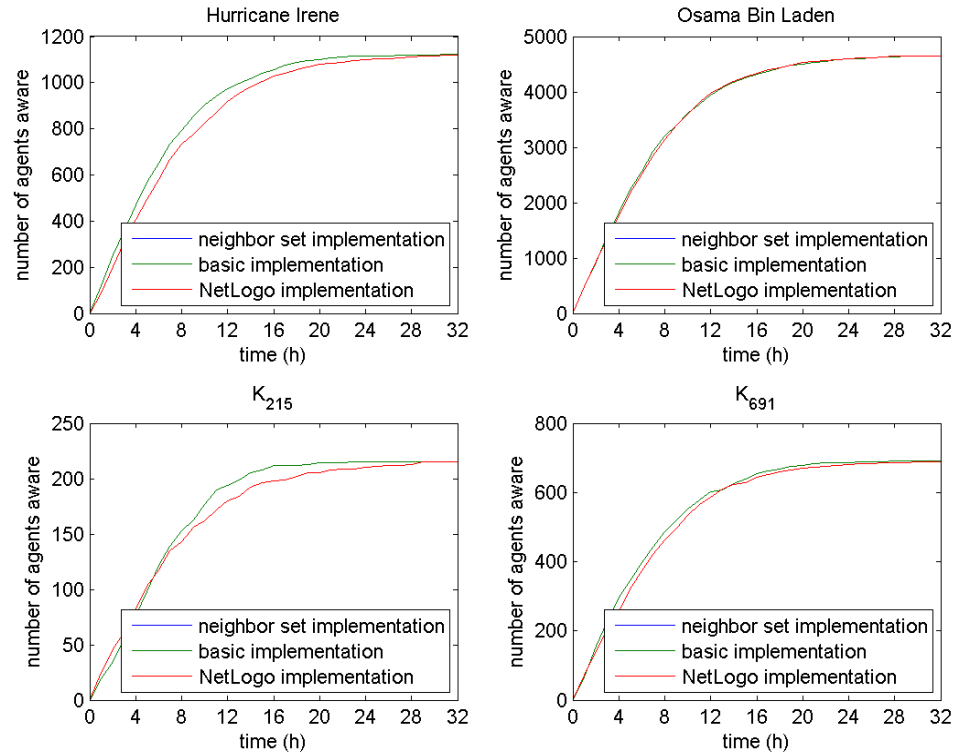


- A possible improvement: a preliminary pass through just the newly-aware nodes which updates just their downstream nodes. After this preliminary step, we can proceed as in the basic implementation, but without needing to recompute awareness numbers.

Neighbor Set Implementation: Representing Adjacency Efficiently

- Our new updating procedure suggests a further possible improvement: replacing the network's adjacency matrix with a sparse data structure which reflects the structure of the updating rule.
- Information about adjacency can be stored by rewriting the adjacency relation as a function $f: V \rightarrow 2^V$ which returns a node's downstream nodes.
- Concretely, this function is most naturally implemented as a vector of length $|D|$ concatenating the output sets of together with a list of pointers marking the start of each set.

Simulation Results: Comparing the Three Programs



- Basic and neighbor set curves show the number of agents aware at each time step based on a single simulation. The plots show the results when the two implementations are seeded with the same random numbers.
- The NetLogo results were also obtained by a single execution of the simulation.

Parallelization


- Information diffusion is modeled by running the simulation numerous times. The results are then analyzed by computing the mean and 95% confidence intervals at each time step.
- The neighbor set implementation was updated so that the simulations run in parallel.
- The parallelization was implemented using MATLAB's 'parfor' command.
- The code was executed with two simulations running in parallel.

Comparison of Time Efficiency

Bin Laden Network	1 Simulation	100 Simulations	500 Simulations	1,000 Simulations
NetLogo	~ 3 min.	--	--	--
Basic	13.8 sec.	22.8 min.	1.8 hrs.	3.6 hrs.
Neighbor Set	0.8 sec.	1.3 min.	6.3 min.	12.5 min.
NS Parallel	10.2 sec.	1.0 min.	4.5 min.	8.6 min.

Irene Network	1 Simulation	100 Simulations	500 Simulations	1,000 Simulations
NetLogo	~ 50 sec	--	--	--
Basic	3.74 sec.	6.1 min.	29.9 min.	58.2 min.
Neighbor Set	0.27 sec.	23.0 sec.	1.8 min.	3.6 min.
NS Parallel	9.46 sec.	28.7 sec.	1.4 min.	2.6 min.

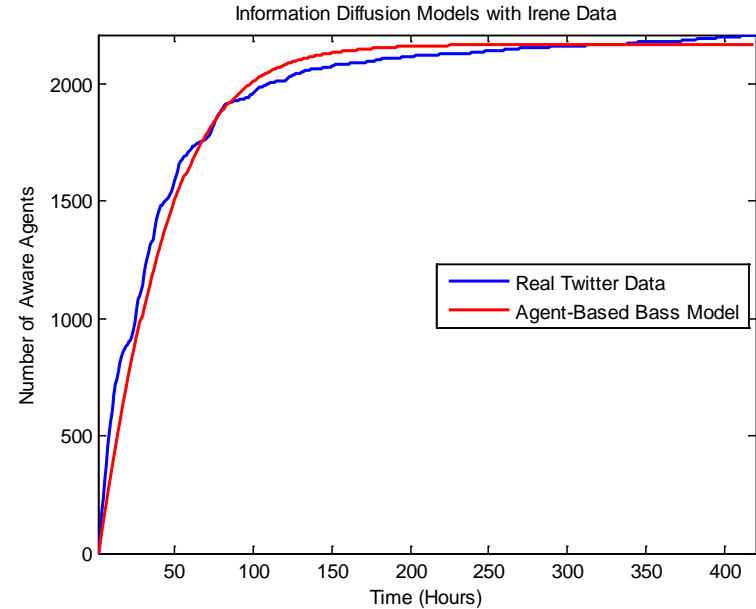
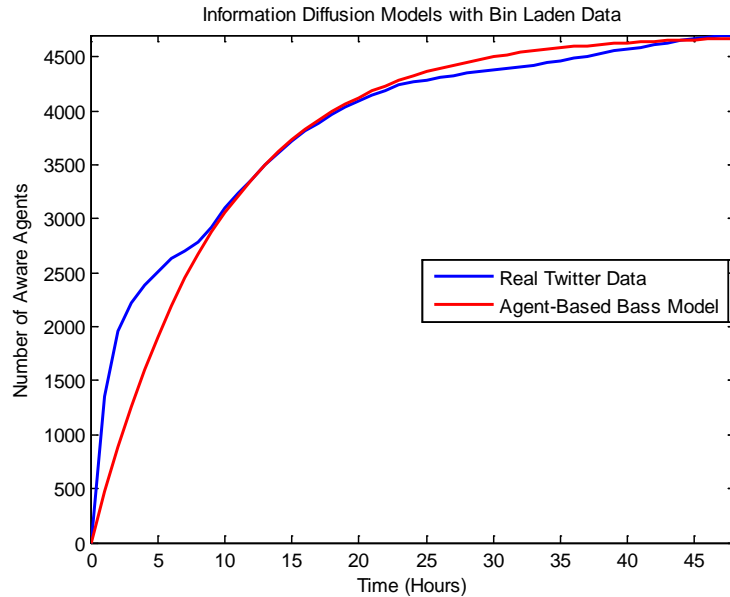
Outline

1. Overview of the Model and the Project
2. Implementing and Parallelizing the Simulation
3. **Testing: Comparing the Model to Real Twitter Data** 
4. Validation: Computing Intervals within Which Simulation Points Are Likely to Fall
5. Validation: Convergence of Agent-Based Model to Analytical Bass Model
6. Validation: Comparing Parameters p and q in the Analytical Model to those in the Agent-Based Model
7. Project Summary and Deliverables

Comparing the Model to Real Twitter Data

- The model was tested to see how well it predicts the actual spread of information through a Twitter network.
- The two real-world cases used to assess the model measure the diffusion of the following information, respectively, through Twitter networks:
 1. The attack that killed Osama bin Laden
 2. News of Hurricane Irene.
- A grid search was performed to determine the parameters p and q for the model which produced the best fit to real Twitter data.

Real Twitter Data vs. Model



The simulation was run 100 times for each data set and the mean number of aware agents was computed at each time step. The red curve shows this mean. The blue curve gives the actual spread of information obtained from real Twitter data.

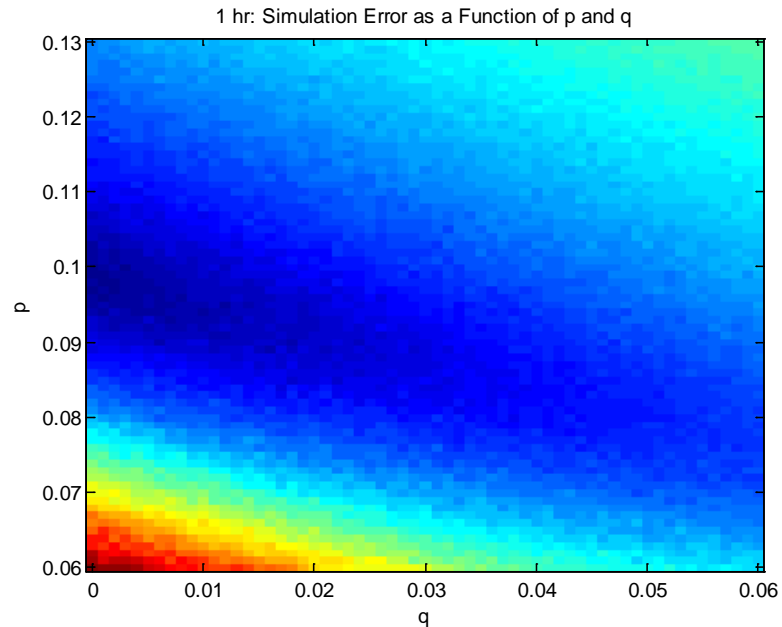
Determining Parameters for the Best Fit to Twitter Data

For each value of p and q , we run the simulation 10 times and compute the mean number of aware agents at each time step.

We then use a grid search to find values of p and q that minimize the error between the simulation means and the real Twitter data. The area between the two curves was used as the error metric.

A color map shows the error as a function of p and q when $\Delta t = 1$ hour.

Optimal Parameters: $p = 0.099$, $q = 0.001$.




What happens as Δt changes?

Δt	optimal $p\Delta t$	optimal $q\Delta t$	relative error
1 hour	0.099	0.0010	1.000
45 minutes	0.076	0.0008	1.008
30 minutes	0.051	0.0006	1.000
15 minutes	0.026	0.0002	1.013

The simulation was run with different time steps Δt . Each time, the results were compared to Twitter data evaluated at the same times. Grid searches were then used to determine the optimal p and q for each value of Δt . Finally, to determine the error between the simulation and the actual Twitter curves as a function of Δt , the area between the curves was computed each time using the corresponding optimal values of p and q . The relative error was obtained by normalizing all errors with respect to the error at $\Delta t = 1$.

Outline

1. Overview of the Model and the Project
2. Implementing and Parallelizing the Simulation
3. Testing: Comparing the Model to Real Twitter Data
- 4. Validation: Computing Intervals within Which Simulation Points Are Likely to Fall** 
5. Validation: Convergence of Agent-Based Model to Analytical Bass Model
6. Validation: Comparing Parameters p and q in the Analytical Model to those in the Agent-Based Model
7. Project Summary and Deliverables

Probability an Agent Becomes Aware at Each Time Step

One way to validate the simulation code is to compare the distributions of the number of aware agents at each time step with those predicted theoretically.

The probability that an unaware agent i becomes aware at each time step:

$$P_i(t) = p \Delta t + q \Delta t [a_i(t) / n_i] - (p q \Delta t^2 [a_i(t) / n_i])$$

- n_i is the number of neighbors of agent i .
- $a_i(t)$ is the number of neighbors of agent i that became aware before time t .
- p and q are parameters which indicate the effectiveness of advertising and WOM per unit of time, respectively.

Computing Probability Distributions at Each Time Step of the Simulation

We make a simplifying assumption: all agents are connected. As a result, local network structure disappears.

Because each agent's neighbor set includes every single agent (even itself), for each i , $a_i(t)/n_i$ is simply $A(t)/N$, the aware fraction of the network.

If we let $p' = p \Delta t$ and $q' = q \Delta t$, then at each time step the probability that an unaware agent becomes aware is

$$P = p' + \frac{A}{N} q' - p' q' \frac{A}{N} = p' + q' \frac{A}{N} (1 - p').$$

Agents' States Depend on One Another

At every time step, each agent is in one of two states: aware or unaware. Thus, there are 2^N possible states for the system.

We cannot assume that the number of aware agents is binomially distributed at each time step. The state of each agent is not independent of the states of other agents – in fact, it is very much influenced by them!

Nevertheless, because its current state depends only on its previous state, the system forms a Markov process.

Reducing the State Space

The map which counts the number of aware agents takes the system's state space to a reduced space with only $N+1$ states.

Because of the very special structure of the system, the counting map respects the original Markov process, giving rise to a compatible Markov process on the reduced state space.

Rather than describe the first Markov process and then trace through the counting map to obtain a description of the second Markov process, it is clearer to describe the second Markov process directly.

Updating Awareness as a Markov Chain

Take the case of a fully-connected network with N agents.

Let X_k be a random variable giving the number of agents aware at time step k .

Since the number of agents aware at a given time step depends only on the number aware at the previous time step, the sequence X_1, X_2, \dots, X_m forms a Markov chain.

We can represent its transition probability distribution with a matrix.

Transition Matrix

current state j

$T(i,j) = P(X_{k+1} = i | X_k = j)$

probability of transitioning
from state j to state i
at time step k+1

future
state i

The system is in state j if exactly j agents are aware.

Because the state space consists of the N+1 elements 0, 1, ... , N, it is convenient to index the rows and columns of the transition matrix T beginning with 0 instead of 1.

Entries of Transition Matrix

Choose the number of newly-aware agents from available candidates to result in future state i .

Probability that an agent becomes aware given current state j .

Probability that an agent remains unaware given current state j .

$$T(i, j) = \begin{cases} \binom{N-j}{i-j} \left[p' + q' \frac{j}{N} (1 - p') \right]^{i-j} \left[1 - \left(p' + q' \frac{j}{N} (1 - p') \right) \right]^{N-i} & i \geq j \\ 0, & i < j \end{cases}$$

The number of aware agents cannot decrease in a future state.

An Example with a Small Network

$$\begin{array}{c}
 \left[\begin{array}{ccc}
 \binom{3}{0} 0.1^0 0.9^3 & 0 & 0 \\
 \binom{3}{1} 0.1^1 0.9^2 & \binom{2}{0} 0.11^0 0.89^2 & 0 \\
 \binom{3}{2} 0.1^2 0.9^1 & \binom{2}{1} 0.11^1 0.89^1 & \binom{1}{0} 0.12^0 0.88^1 \\
 \binom{3}{3} 0.1^3 0.9^0 & \binom{2}{2} 0.11^2 0.89^0 & \binom{1}{1} 0.12^1 0.88^0
 \end{array} \right]
 \end{array}
 \overset{m}{\begin{array}{c} \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \end{array} \right]
 \begin{array}{c} \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \end{array} \right] \end{array}$$

$$T(3,1) = P(X_{k+1} = 3 | X_k = 1) = \binom{2}{2} \left[0.1 + \left(\frac{1}{3} (0.03 - 0.1 * 0.03) \right) \right]^2 \left[1 - \left(0.1 + \left(\frac{1}{3} (0.03 - 0.1 * 0.03) \right) \right) \right]^0$$

$$\begin{aligned}
 p' &= 0.100 \\
 q' &= 0.033
 \end{aligned}$$

A transition from 1 to 3 aware agents requires that 2 agents become aware.

* Entries are indexed from 0.

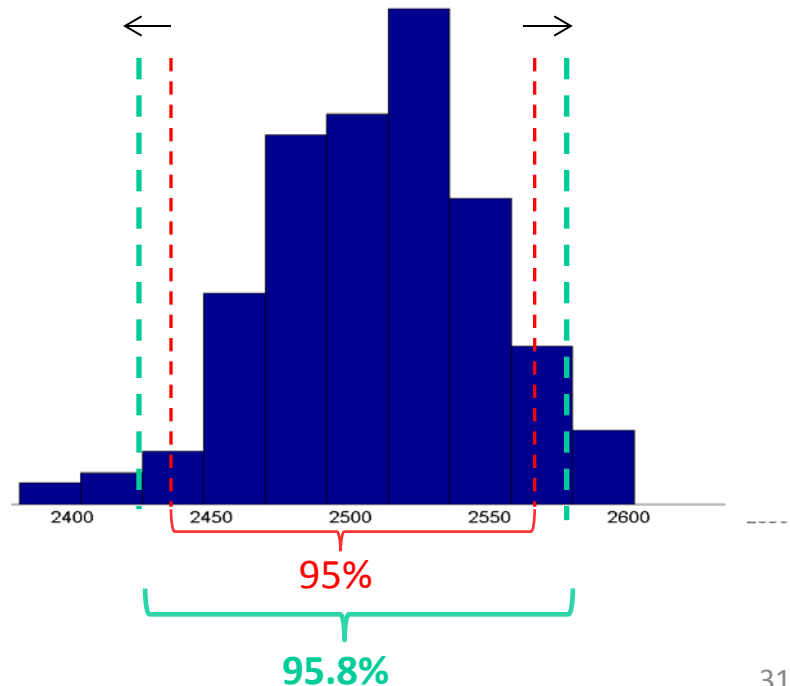
Computing Theoretical Intervals for the Distribution

We can use the transition matrix to compute the distributions of the random variables X_k .

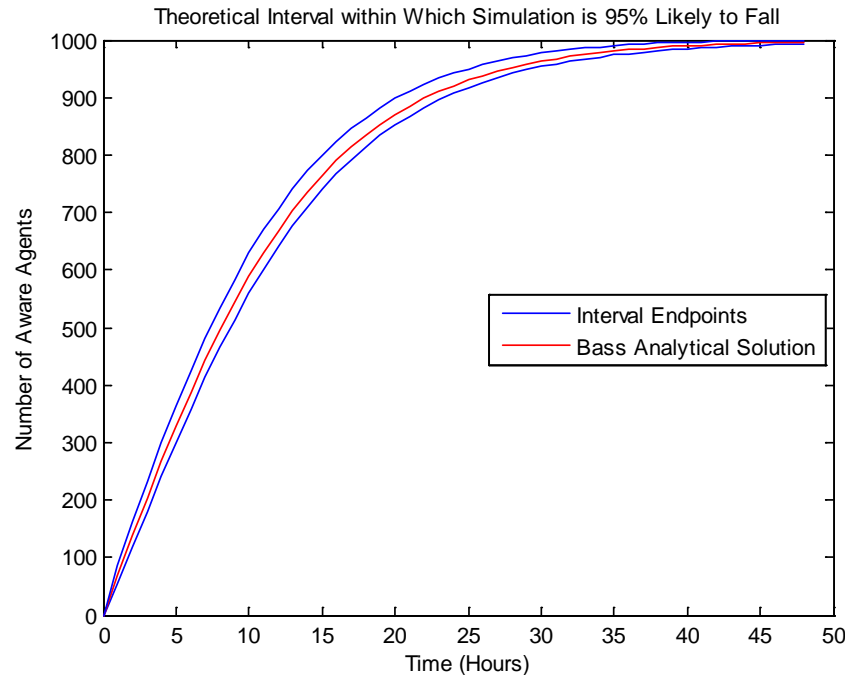
At each time step k , we wish to determine an interval $[a_k, b_k]$ such that $P(a_k \leq X_k \leq b_k) = 0.95$.

But since our distributions are discrete, such an interval may not exist. We therefore choose a_k and b_k so that $P(X_k \leq a_k) \approx 0.025$ and $P(X_k \leq b_k) \approx 0.975$.

Probability Distribution of the Number of Aware Agents at Time Step 7



Intervals within Which the Distribution is Approximately 95% Likely to Fall at Each Time Step



The plot shows curves for a_k (bottom) and b_k (top) at each time step k , such that $P(a_k \leq X_k \leq b_k) \approx 0.95$. Between the curves lies the analytical solution to the Bass ODE.

Validating Code Implementation by Comparing the Simulation to Theoretical Results

Does our simulation produce a similar distribution of the number of aware agents at each time step as the theoretical distribution does?

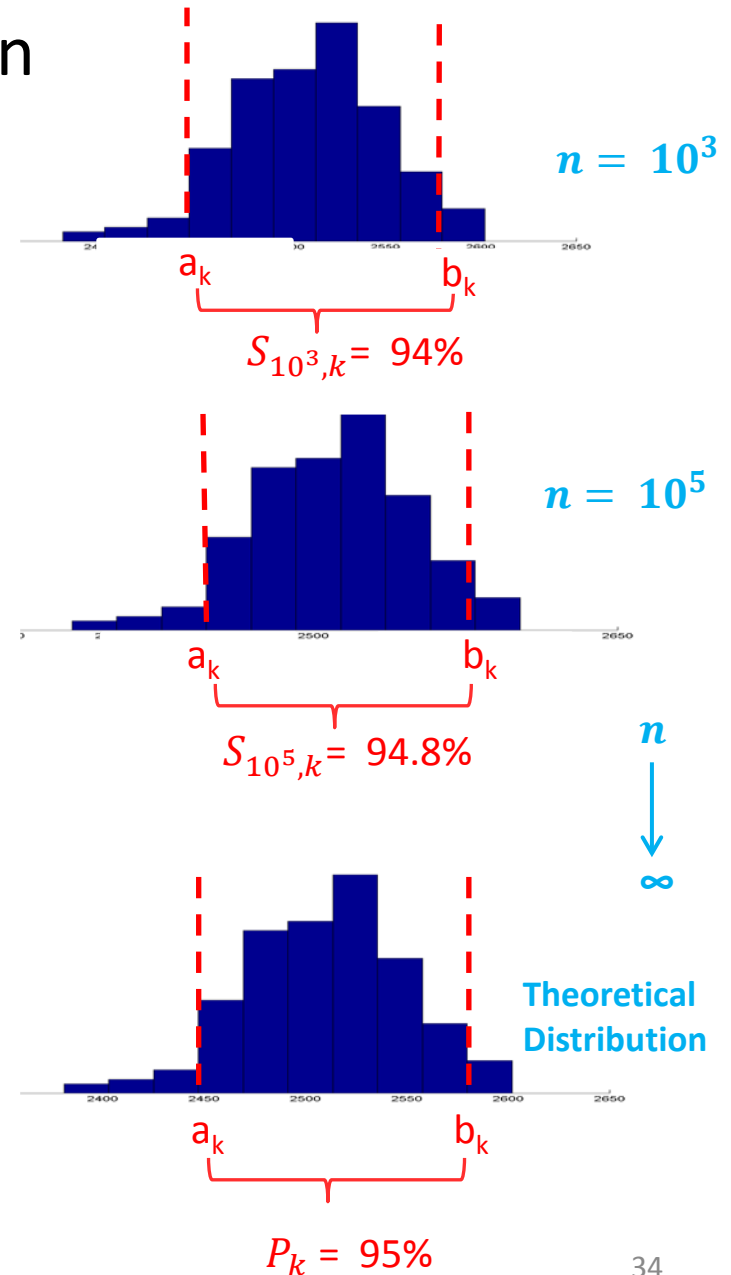
One way to compare the two distributions is to run the simulation numerous times and determine how frequently the simulation results fall within the theoretically computed $\sim 95\%$ interval at each time step.

How Frequently Does the Simulation Fall within Theoretical Intervals?

At each time step k , we fix each interval $[a_k, b_k]$ and theoretically determine an exact value for P_k , the $\sim 95\%$ probability that the number of agents will fall within the interval.

We then run the simulation n times and compute an empirical percent $S_{n,k}$ of times that that the simulation points fall within the interval.

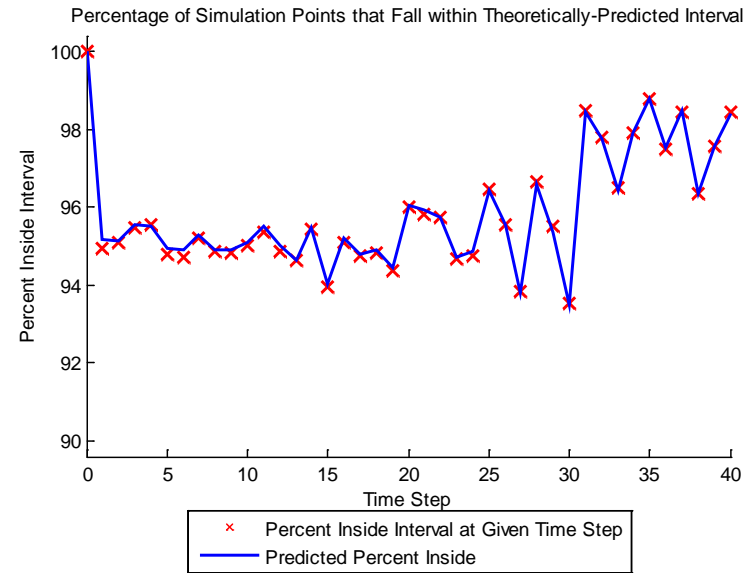
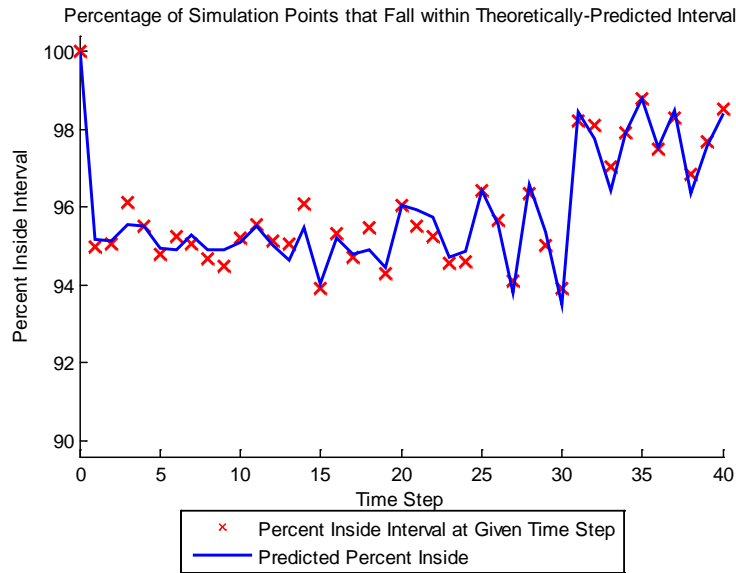
We would like to know how well $S_{n,k}$ compares to P_k as n increases.



Percentage of Simulation Points that Fall within Theoretically Predicted Intervals

$n = 5,000$

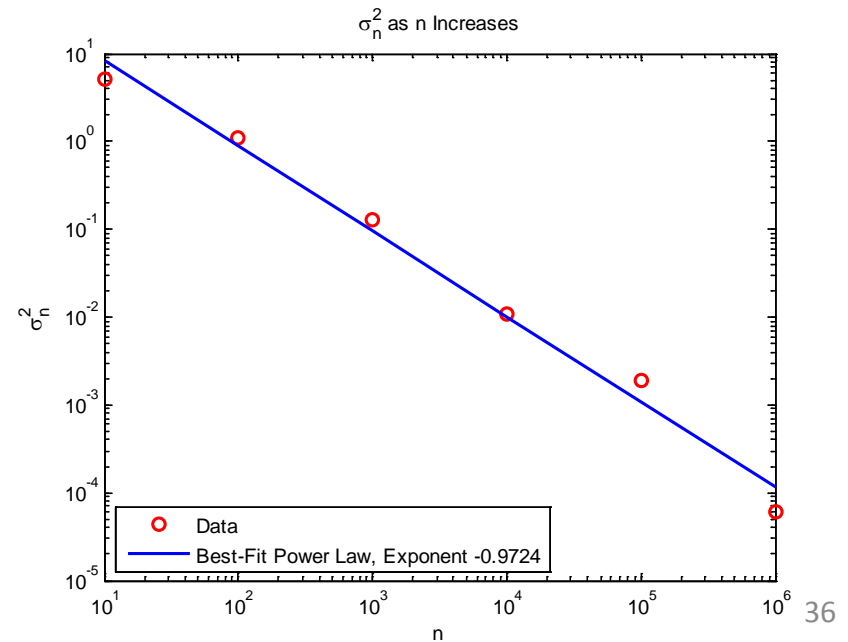
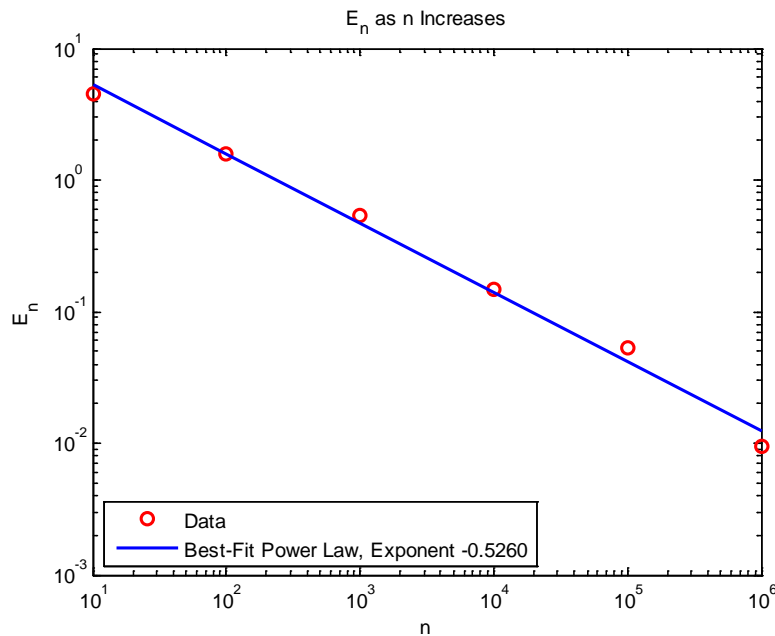
$n = 50,000$




What happens as we increase the number of simulations?

For each n and k , we can compute $E_{n,k} = |P_k - S_{n,k}|$, the discrepancy between the predicted percent of times that X_k should fall in the interval $[a_k, b_k]$ and the actual percent of times that it does. To summarize the behavior of the $E_{n,k}$ at a given n , we can take their mean \bar{E}_n as k varies. In addition to computing the first moment across k , we also compute the second moment σ_n^2 .

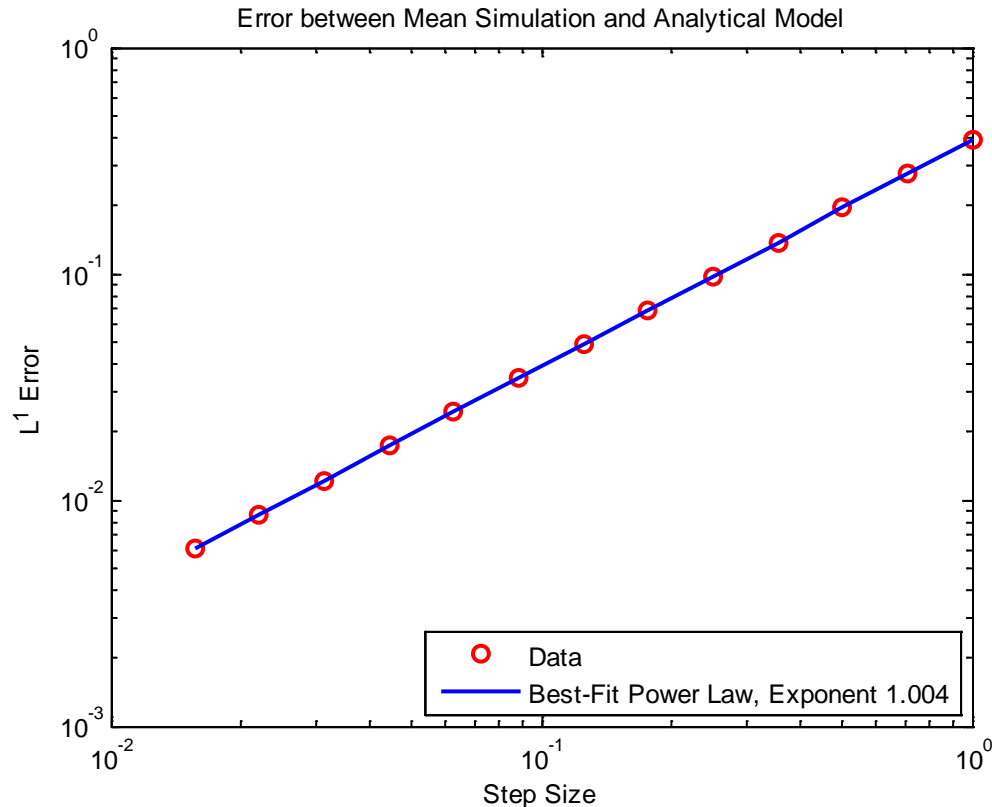
The plots below show the behavior of \bar{E}_n and σ_n^2 as n increases.



Outline


1. Overview of the Model and the Project
2. Implementing and Parallelizing the Simulation
3. Testing: Comparing the Model to Real Twitter Data
4. Validation: Computing Intervals within Which Simulation Points Are Likely to Fall
5. **Validation: Convergence of Agent-Based Model to Analytical Bass Model** 
6. Validation: Comparing Parameters p and q in the Analytical Model to those in the Agent-Based Model
7. Project Summary and Deliverables

Agent-Based Model Converges to Analytical Curve as Δt Decreases



We ran the simulation 100 times on a fully-connected network for each value of Δt and computed the mean number of aware agents at each time step. We then compared this mean at each time step to the number of aware agents given by the analytical model.

Outline

1. Overview of the Model and the Project
2. Implementing and Parallelizing the Simulation
3. Testing: Comparing the Model to Real Twitter Data
4. Validation: Computing Intervals within Which Simulation Points Are Likely to Fall
5. Validation: Convergence of Agent-Based Model to Analytical Bass Model
6. Validation: Comparing Parameters p and q in the Analytical Model to those in the Agent-Based Model 
7. Project Summary and Deliverables

Analytical Bass Model Formulation

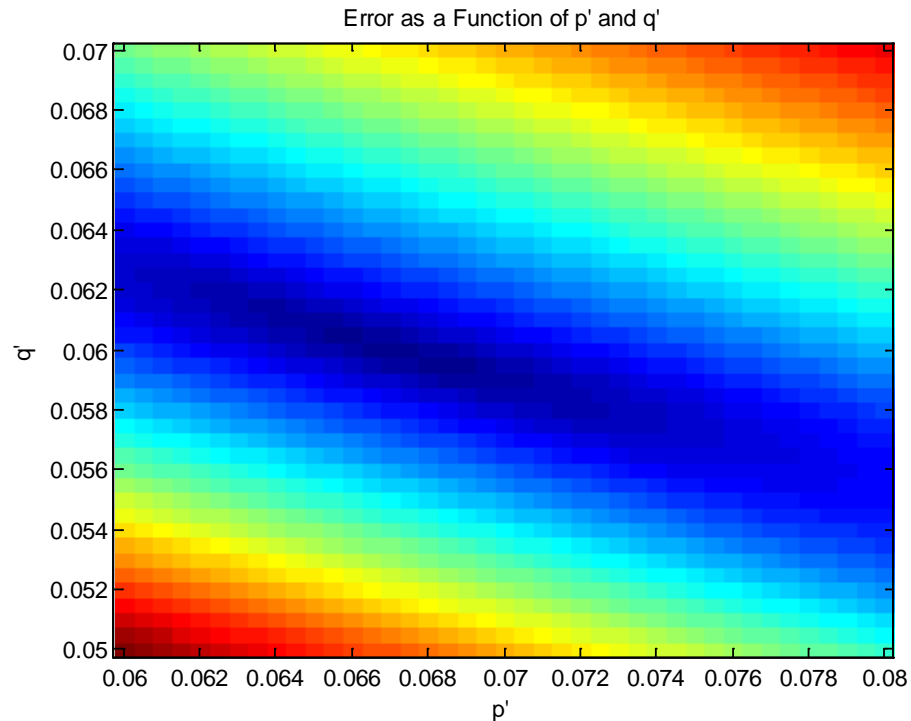
The Bass model describes the change in the fraction of a population that has become aware of a piece of information:

$$\frac{F'(t)}{1-F(t)} = p + qF(t)$$

$$F(0) = 0,$$

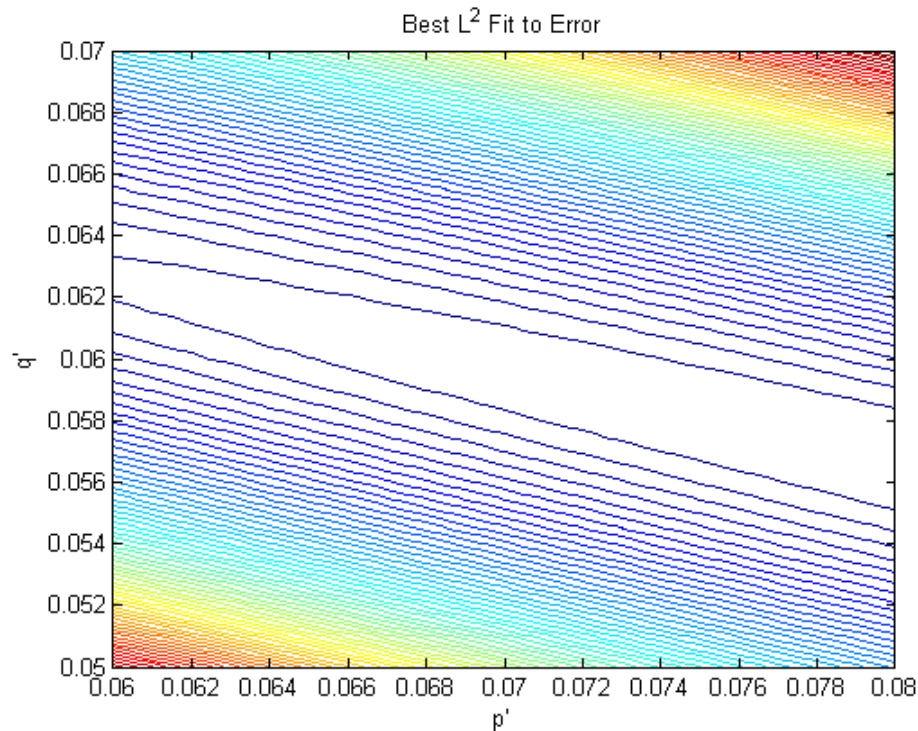
where $F(t)$ is the aware fraction of the population, p is the advertising coefficient, and q is the word-of-mouth coefficient.

What is the relationship between the parameters in the analytical model and those in the agent-based model?



Let p and q be parameters for the analytical Bass model. Let p' and q' be parameters for the agent-based model. Fix $(p, q) = (0.07, 0.06)$. The color map gives the error between the analytical and agent-based models as a function of p' and q' . For each p' and q' the mean of the agent-based model taken over 10 runs was used. The minimum error occurred at $(p', q') = (0.0690, 0.0595)$.


Fitting a Quadratic Approximation to the Error



$$E = 9871[0.2832(p' - 0.09041) + 0.9591(q' - 0.05368)]^2 + 22.67[-0.9591(p' - 0.09041) + 0.2832(q' - 0.05368)]^2 + 0.2438$$

After computing the best-fit quadratic approximation to the error function, we plot its contours above. The conic center is quite different from the computed minimum at $(p', q') = (0.0690, 0.0595)$.

Outline

1. Overview of the Model and the Project
2. Implementing and Parallelizing the Simulation
3. Testing: Comparing the Model to Real Twitter Data
4. Validation: Computing Intervals within Which Simulation Points Are Likely to Fall
5. Validation: Convergence of Agent-Based Model to Analytical Bass Model
6. Validation: Comparing Parameters p and q in the Analytical Model to those in the Agent-Based Model
7. **Project Summary and Deliverables** 

Project Summary

- An agent-based Bass information diffusion model was implemented in MATLAB and then analyzed.
- Using a more efficient updating rule and taking advantage of sparse data structures produced a more time- and memory-efficient code.
- The current implementation is faster and runs more reliably than a previous NetLogo implementation, allowing simulations on to be performed on larger, more fully connected networks in future research.
- With the correct parameters, the agent-based Bass model provides a reasonable description of real-world Twitter information diffusion. The model parameters that produce the best fit are specific to each data set. Varying the length of time steps has little effect on the fit between the model and real data.
- As the length of time steps decrease, the agent-based Bass model converges to the analytical Bass model. Parameters p and q of the analytical model correspond well to parameters p' and q' of the agent-based model.

Milestones

- October Developed basic simulation code. Wrote a more efficient neighbor set implementation of simulation.
- November Validated code against analytic model. Developed code for statistical analysis of results. Analyzed confidence intervals.
- December Validated simulation against existing NetLogo implementation. Prepared mid-year presentation and report.

Milestones

- January Tested model against empirical Twitter data. Performed parameter searches to determine best fit. Tested best fit as a function of Δt .
- February Parallelized code. Analyzed convergence of agent-based model to analytic model. Computed probability distributions using Markov chains.
- March Computed and tested 95% intervals. Compared correspondence of parameters in agent-based model to those in analytic model. Analyzed the error.
- April Revised and re-tested $\sim 95\%$ intervals. Fitted quadratic function to error. Wrote final project report and prepared presentation.

Proposed Deliverables

- Simulation code.
- Code for statistical analysis.
- A graph with the following three curves based on data collected from numerous runs of the simulation: mean and both ends of a 95 percent confidence interval at each time step.
- A detailed comparison of my code's running time against that of the existing NetLogo implementation.
- Side by side, the graphs of simulation results compared with the real-world observed Twitter data.

Additional Deliverables

- Faster neighbor set implementation of the simulation, parallelized.
- Code for grid search to determine best-fit parameters.
- Code to compute theoretical distributions for fully-connected networks using Markov chains.

References

- Bass, Frank (1969). “A new product growth model for consumer durables”. *Management Science* 15 (5): p. 215–227.
- Chandrasekaran, Deepa and Tellis, Gerard J. (2007). “A Critical Review of Marketing Research on Diffusion of New Products”. *Review of Marketing Research*, p. 39-80; Marshall School of Business Working Paper No. MKT 01-08.
- Devore, J.L.. Probability and statistics for engineering and science. Brooks/Cole, 1991.
- Dodds, P.S. and Watts, D.J. (2004). “Universal behavior in a generalized model of contagion”. *Phys. Rev. Lett.* 92, 218701.
- Karlin, S. and Taylor, H.M. A first course in stochastic processes. Academic Press, 1968.
- Mahajan, Vijay; Muller, Eitan and Bass, Frank (1995). “Diffusion of new products: Empirical generalizations and managerial uses”. *Marketing Science* 14 (3): G79–G88.
- Rand, William M. and Rust, Roland T. (2011). “Agent-Based Modeling in Marketing: Guidelines for Rigor (June 10, 2011)”. *International Journal of Research in Marketing*; Robert H. Smith School Research Paper No. RHS 06-132.