Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

A FRAMEWORK OF DIGITAL TWINS FOR MODELING HUMAN-SUBJECT WORD FORMATION EXPERIMENTS

Hao He[1], Xueying Liu[1], Chris J. Kuhlman[2], and Xinwei Deng[1]

[1]Department of Statistics, Virginia Tech, Blacksburg, VA, USA
[2]Advanced Research Computing, Virginia Tech, Blacksburg, VA, USA

ABSTRACT

Agent-based models (ABMs) are used to simulate human-subject experiments. A comprehensive understanding of these human systems often requires executing large numbers of simulations, but these requirements are constrained by computational and other resources. In this work, we build a framework of digital twins for modeling human-subject experiments. The framework has three modules: ABMs of player behaviors built from game data; extensions of these models to represent virtual assistants (agents that are exogenously manipulated to create controlled environments for human agents); and an uncertainty quantification module composed of functional ANOVA and a Gaussian process-based emulator. The emulator is built from the extended ABM; we focus on emulator validation. By incorporating experimental data and agent-based simulation data, our proposed framework enhances the virtual representation of the dynamics in human-subject word formation experiments, which we consider a digital twin. Networked anagram experiments are used as an exemplar to demonstrate the methods.

1 INTRODUCTION

1.1 Background and Motivation

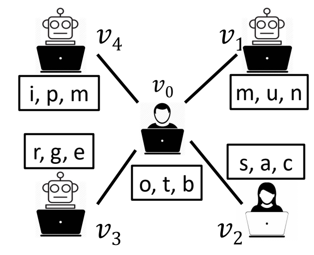
Augmenting agent-based models. Agent-based modeling (ABM) has been used for many decades, and with ever expanding modeling capabilities, there is a commensurate desire to expand the sets of inputs to be analyzed, and the sizes of problems to solve (Matković et al. 2014). Oftentimes, there are restrictions on computing resources, making it difficult to evaluate all possible input sets in a timely fashion. Stochastic simulations require even more simulation executions, to account for variability. A case in point is epidemiological simulations, whose value has been illustrated with recent Ebola and COVID outbreaks. In a Cluster Computing and the Grid (CCGRID) Scale Challenge, Bhatele et al. (2017) demonstrated that a stochastic agent-based simulation (ABS) of influenza, on a constructed digital twin of the U.S. population incorporating 280 million people and 5.8 billion daily interactions over a 180-day simulation period, could be completed in 10.4 seconds. This simulation and others used some of the most advanced supercomputers at the time (Blue Waters, Cori, and Mira). For the first time, then, meaningful parametric studies could be completed on a digital twin of the continental U.S. in less than 24 hours. The COVID outbreaks demonstrated that the Centers for Disease Control and Prevention (CDC) require daily predictions on disease propagation to assist in setting policy, so this simulation scaling has real-world implications.

Therefore, there is ample motivation to accelerate simulations and commensurate analyses of digital twins for understanding phenomena, establishing causal relationships, and predicting or forecasting. Alternatives to making massively parallel simulators (e.g., Bhatele et al. (2017)) are coupling simulations with other analytic tools, such as dynamical systems models (Swinerd and McNaught 2012) and statistical models (Boulesteix et al. 2020). By incorporating analytic tools to explore input spaces, these tools can efficiently guide the selection of input parameters with which to run a reduced set of detailed ABSs. Also, using statistical methods, one can build surrogate models based on ABM results (Edali and Yücel 2019; Wate et al. 2020). For our purposes here, a surrogate model is a (statistical) model that takes as input a (sub)set of ABM parameters and predicts one output quantity of an ABS; it is trained on ABS data. Such surrogate models, embedded in software called emulators, can then be run much more rapidly with far fewer resources, enabling uncertainty quantification and the exploration of greater input spaces (Carbajal et al. 2017; Wate et al. 2020). With the above motivation, this work devises an approach for integrating ABM and surrogate modeling for understanding networked anagram games (i.e., word construction games).

Extending human behavior data and models. When building (agent-based) models based on experimental data, the resulting models could be limited by the parameter space of the experiments. Therefore, it is often desirable to calibrate such models (with appropriate analyses) and use them for conditions beyond those of the experimental data. One use of extending models of human behaviors is for the design and analysis of experiments involving virtual assistants (VAs). A virtual assistant is an agent that has realistic behaviors that are controlled exogenously to purposely manipulate the environment of other agents (e.g., humans). Game designers often use VAs to control an ego's environment (Kohavi et al. 2009). In this work, we use data-driven modeling of heterogeneous human behaviors in word formation games, as an exemplar, for extending an ABM.

Studying anagram games is useful because they are recognized as non-trivial cognitive tasks with well-defined success parameters, e.g., whether a word is formed in a single-player game and the number of words formed in a multi-player game (Cadsby et al. 2007). These games are used to research the effects of and on cognitive physiological tasks, e.g., (Charness et al. 2014).

Example game configuration



Set of player actions

Player Action	Description
a_1	Idle (no-op)
a_2	Reply to request
a_3	Request letter
a_4	Form word

Illustrative sequence of game actions

Game time	Player	Action	Detail
t_1	v_0	a_3	v_0 requests "m" from v_1.
t_2>t_1	v_1	a_2	v_1 replies with "m" to v_0.
t_2+1	v_0	---	v_0 now has letters o, t, b, m.
t_3>t_2+1	v_0	a_4	v_0 forms word "bottom".

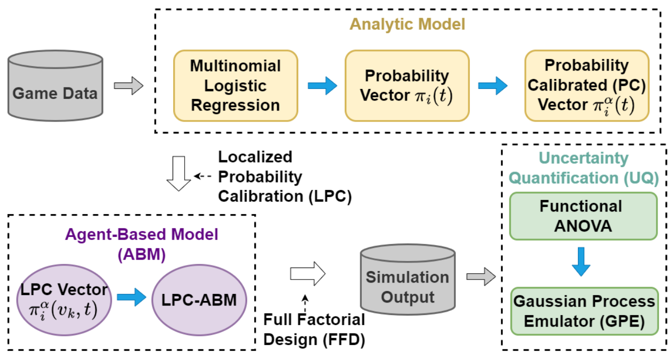
Figure 1: Illustrative networked group anagram game (NGrAG). The network $G(V,E)$ of five players is on the left, with two humans ($v_0, v_2 \in V$) and three VAs ($v_1, v_3, v_4 \in V$). Sets $L_{init}(k)$ of $n_l = 3$ initial letters are assigned to each $v_k \in V$, $k \in \{0,1,2,3,4\}$. The set of permissible player action types is given in the middle. The table on the right shows a time sequence of representative interactions between player v_0 and VA v_1 in carrying out various actions of a game at time steps $0 < t_1 < t_2 < t_3 \leq t_g$.

1.2 Networked Anagram Game

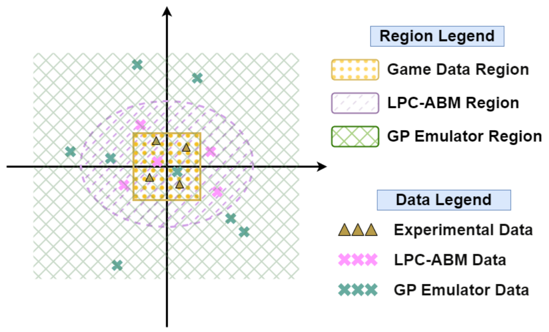
Figure 1 is a representative instance of a networked group anagram game (NGrAG). On the left, a star graph $G(V,E)$ is specified with human hub node v_0 and VAs for three of four leaf nodes (v_1, v_3, v_4), which along with human player v_2, form the node set (i.e., player set) V. The four communication channels (i.e., lines) comprise the edge set E. Each node is assigned n_l initial letters, placed by the players in rectangles. Players can interact with their network neighbors, requesting letters and replying to letter requests (see action set in the middle of Figure 1). A player who replies with a copy of a letter also retains that letter, so the replier can continue to use that letter in her words. Each player may repeat these actions, in any order, throughout the $t_g = 300$ second game duration. The team's earnings in a game are based on the total number of words formed by the entire team; the earnings are split evenly among the players to encourage them to cooperate during the game. Players can use a letter multiple times in the same word, e.g., see the far right table in the figure: using *t* and *o* twice in word *bottom*. Once a player acquires a letter, she can use it any number of times within and across words; the letter is never lost. This enables players to form more words. Additional details and experimental results of over 240 games/experiments played online with all human subjects (i.e., no VAs) are provided in (Cedeno-Mieles et al. 2020). This is also the game we model herein, and these data are used to build the models.

1.3 Novelty and Contributions

Novelty. The novelty of this work is two-fold. First, models of heterogeneous behaviors in VAs in ABSs are presented. Second, NGrAG emulators are built from the ABM output. Both of these have seldom been done for models based on experimental game data of human behavior. They have not been done for group word construction games.


Analytic Model
Game Data
Multinomial Logistic Regression
Probability Vector $\pi_i(t)$
Probability Calibrated (PC) Vector $\pi_i^\alpha(t)$
Localized Probability Calibration (LPC)
Uncertainty Quantification (UQ)
Functional ANOVA
Gaussian Process Emulator (GPE)
Agent-Based Model (ABM)
LPC Vector $\pi_i^\alpha(v_k, t)$
LPC-ABM
Simulation Output
Full Factorial Design (FFD)

(a) Analysis methodology pipeline.



(b) Game data space and expanded LPC-ABM and GPE prediction spaces.

Figure 2: (a) The complete analysis pipeline for extrapolative analysis of experimental data. The upper box, Analytical Model, is presented in prior work, which includes experimental validation of the models (see text for references). The remaining two boxes, Agent-Based Model and Uncertainty Quantification, are contributions of this work. (b) Conceptual view of the space of experiment inputs, denoting that the models are built to explore input parameter spaces beyond those tested, with those of Gaussian Process Emulator (GPE) beyond those of the Localized Probability Calibration Agent-Based Model (LPC-ABM).

Contributions. The contribution of the proposed framework can be summarized as the analysis pipeline of Figure 2a. The first component, the upper dotted box titled Analytic Model, represents previous work (Cedeno-Mieles et al. 2020; Liu et al. 2023) and the ABMs in it were validated extensively in (Cedeno-Mieles et al. 2020). The second and third components are new and are part of our contribution. Our first contribution, which is the second component (bottom left box in purple), Agent-Based Model, is a *local* version of the Probability Calibration (PC) model (Liu et al. 2023), designated LPC model, that captures heterogeneous behaviors across players (and agents) and within players over time. The ABM has been extended to incorporate these modeling effects and is referred to as LPC-ABM. The models are presented in Section 3.

Our second contribution is the third component (bottom right box in green) of Figure 2a. Functional ANOVA (FANOVA) and Gaussian Process Emulator (GPE) inherently incorporate Uncertainty Quantification (UQ), which enhances model capability and resource optimization. The concept of “uncertainty" encompasses the uncertainty present in the game data, simulation outputs, and the outcomes produced by the GP Emulator. The technical details of Functional ANOVA and GP Emulator are elaborated in Section 3, and further discussions can be found in Górecki and Smaga (2019) and Gramacy (2016). Building surrogate models remains a challenge (Heppenstall et al. 2021). Use of surrogate modeling for prediction and UQ can be much more efficient than running a set of ABSs. It can greatly enhance wider exploration of input parameter spaces of ABMs. This is conveyed in Figure 2b, where we illustrate that experimental data can be used to create ABMs that can investigate a larger input space (owing to resource limitations in conducting experiments). Then, surrogate (e.g., Gaussian process) models can be built from simulation output and used to construct emulators. The emulator space is greater than that for ABMs because more analyses can be completed with an emulator, in a given amount of time.

Given the impracticality of enumerating all possible experimental conditions, our GPE facilitates extrapolation into unexplored domains. In this work, FANOVA is run on LPC-ABM input/output to identify the input parameters whose changes most affect a target dependent variable. GP modeling (GPM) approaches use ABM simulation output to construct statistical models that emulate LPC-ABM; the software that is produced are the GPEs. Execution of GPEs in this work focuses on emulator validation. Results of FANOVA guide the specification of inputs to the GPE, but other input conditions can also be specified.

Our third contribution is the application of this pipeline and digital twin framework to NGrAG data. LPC-ABM is used to produce the output of new player behaviors in games. Simulations include (1) node of the game network that is occupied by a VA, (2) time segment of the 300-second game duration in which VA behavior differs from human behavior, and (3) the level of deviation in the performance of VAs relative to human subjects. Results show, for example, that the latter two factors more significantly affect results than the first factor. Also, there are carry-over effects in that for some conditions, when VA behavior is removed, the system's performance remains greater than baseline system behaviors. Analyses confirm that the quality of the GPE emulator predictions against LPC-ABM results are good, with the Nash-Sutcliffe efficiency (NSE) above 98%.

The rest of this paper is organized as follows. Section 2 contains related work. Section 3 presents the models, including details about the LPC-ABM, FANOVA and GPE introduced in Figure 2. Section 4 begins with a presentation of the LPC-ABM simulation results, followed by an analysis of the FANOVA test outcomes and an evaluation of the GPE's performance. Section 5 summarizes this paper and discusses some future work.

2 RELATED WORK

Studies use statistics to steer simulations for exploring large input spaces. Matković et al. (2014) employed hybrid steering of simulations in analyzing car engines, to identify new input parameter sets for simulation. Their system enables a domain expert to interactively select data points (representing simulation inputs) in an iterative manner, with the aid of regression to interpolate the simulation solution space. To realize efficient but rigorous parameter space exploration in constructing neurological networks, Nowke et al. (2018) developed an interactive tool for visualizing and steering parameters in neural network simulation models.

Emulators are used as a proxy for simulation systems; classes of emulators are provided in Pietzsch et al. (2020). The literature is vast and we give a small sampling here. Kasim et al. (2021) used neural networks to build simulation emulators from small datasets, for climate and energy sciences. Wate et al. (2020) built emulators from simulation data, using Gaussian process modeling, to predict heating and cooling loads for buildings, incorporating variability in inputs through uncertainty quantification. Emulators for communication in traffic simulations are described in (Babu and Kumar P 2022). We know of no works that emulate group-based word construction games, as we do here.

3 THE PROPOSED METHODOLOGY

This section contains the models within the flowchart of Figure 2a. Sections 3.1 and 3.3 provide, respectively, the LPC model and the LPC-ABM (see the bottom left box of the figure). Sections 3.4 and 3.5 contain, respectively, details of FANOVA and the Gaussian process model in green in the bottom right box.

3.1 Localized Probability Calibration (LPC) Model

Given that a decision-making sequence of a player in a 5-minute NGrAG—see the right table in Figure 1 as a simple example of such sequences—can be treated as a discrete-time stochastic process, multinomial logistic regression is utilized to estimate the transition probabilities from a player's current action to possible next actions (Cedeno-Mieles et al. 2020).

The dynamics of an agent in the NGrAG are as follows. The action a_i, $i \in \{1,2,3,4\}$, of a player/agent at time $(t-1)$ is known. Permissible actions are those in the middle table of Figure 1. The goal is to compute the player's next action a_j, $j \in \{1,2,3,4\}$ at time t. The transition probability vector is $\boldsymbol{\pi}_i(t) = (\pi_{i1}(t), \pi_{i2}(t), \pi_{i3}(t), \pi_{i4}(t))^T$, where π_{ij} is the probability that the next action is a_j, given a_i. For the conditions at $(t-1)$, including known i, the four π_{ij} are computed using a multinomial logistic regression model, and using these four probabilities, then a multinomial sampling is used to determine the player's action a_j at time t (Cedeno-Mieles et al. 2020). This process is repeated over all agents at a fixed t, and then over all $0 < t \leq t_g$, in performing one simulation instance of the word game.

To generate behaviors not observed in the original experiments, a probability calibration (PC) model is developed in Liu et al. (2023) to adjust the transition probability vector $\boldsymbol{\pi}_i(t)$ as

$$\boldsymbol{\pi}_i^{\alpha}(t) \propto (1+\boldsymbol{\alpha}) \cdot \boldsymbol{\pi}_i(t),$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ are calibration parameters. For example, $\boldsymbol{\alpha} = (0,0,0,1)^T$ means that an agent is much more likely to form words because α_4, corresponding to action a_4, is elevated. A human agent has $\boldsymbol{\alpha} = (0,0,0,0)^T$. This model, however, assumes uniform calibration across all players throughout the game's duration, not accounting for the varied effects of VAs on different players at different times.

To address this shortcoming, we introduce a *localized* probability calibration (LPC) method to explore the heterogeneous effects of VA usage among game players. Specifically herein, the transition probability of only one node at a time in the network is calibrated for particular time segments, allowing for an in-depth exploration of how the VA affects decision-making dynamics within the game. The calibration parameter $\boldsymbol{\alpha}(\tilde{v}, \tilde{\tau})$ is defined as a function of node $\tilde{v}$ and time segment $\tilde{\tau}$, and the LPC model is expressed as

$$\boldsymbol{\pi}_i^{\alpha}(\tilde{v}, \tilde{\tau}) \propto (1+\boldsymbol{\alpha}(\tilde{v}, \tilde{\tau})) \cdot \boldsymbol{\pi}_i(t)\,.$$

3.2 Experimental Design

To investigate the heterogeneous effects of VAs, our experimental design focuses on three factors. First, node $\tilde{v}$, is the ID of the node to which the LPC is applied. The small ego network in Figure 3 represents a node in a larger graph that has two neighbors. The node $\tilde{v}$ is either v_0 or v_1 in analyses below. Second, alpha value $\boldsymbol{\alpha}$ is the specific calibration vector applied to the node. In this study, we focus on calibrating the probability of being idle π_{i1}; hence, for simplicity, we denote α_1 as α and explore four values $\alpha = \alpha_1 = -0.2, -0.4, -0.6$, and -0.8, while keeping $\alpha_2 = \alpha_3 = \alpha_4 = 0$. This makes the VA more active because the probability of being idle is reduced. Third, $\tilde{\tau}$ is the time segment where LPC is applied. The 300-seconds game is divided into the three segments: $\tilde{\tau} = 1$: 1-100 seconds; $\tilde{\tau} = 2$: 101-200 seconds, and $\tilde{\tau} = 3$: 201-300 seconds.

Using a Full Factorial Design (FFD), there are 24 unique experimental runs to consider all possible combinations of the three factors: node $\tilde{v}$, alpha value α, and time segment $\tilde{\tau}$, as shown in Table 1. Additionally, Table 2 provides an example of how the LPC model works within simulation run 9. The FFD allows for a systematic examination of both individual and joint effects of multiple factors, and therefore provides a more comprehensive understanding of how different factors collectively influence decision-making dynamics in the NGrAG. Here, we restrict the LPC model to a single node, but the proposed method easily extends to multiple nodes.

3.3 Local Probability Calibration Agent-Based Model and Simulation

The aforementioned LPC method is employed in the LPC-ABM simulation system, and a flowchart of the operations is provided in Figure 4. The flowchart adheres to the game description of Section 1.2 quite closely. After reading in simulation parameters such as the human/VA network, word corpus, and number of simulations to conduct, per-agent inputs are read, e.g., multinomial logistic regression constants for the transition model and initial letters to assign to each player. In this work, we perform 100 simulation

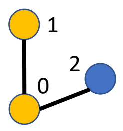


Figure 3: Star-2 network G_{S2} where there are two leaf nodes (node v_1 and v_2) with degrees $d_1 = d_2 = 1$ and one hub node (v_0) with $d_0 = 2$. This small graph is intended as a subgraph for an ego node (v_0) and its neighbors within a larger graph. The orange nodes here are $\tilde{v}$, which are assgined $\alpha = -0.2, -0.4, -0.6$ and -0.8, while $\alpha_2 = \alpha_3 = \alpha_4 = 0$. The blue node always has $\boldsymbol{\alpha} = (0,0,0,0)$, i.e., no calibration.

Table 1: The 24 simulation runs for 3 factors: node ($\tilde{v}$), alpha value ($\boldsymbol{\alpha}$), and time segment ($\tilde{\tau}$).

Run	Node $\tilde{v}$	α	Time Segment $\tilde{\tau}$	Run	Node $\tilde{v}$	α	Time Segment $\tilde{\tau}$	Run	Node $\tilde{v}$	α	Time Segment $\tilde{\tau}$
1	Hub(v_0)	-0.2	1	9	Leaf(v_1)	-0.2	2	17	Hub(v_0)	-0.6	3
2	Hub(v_0)	-0.4	1	10	Leaf(v_1)	-0.4	2	18	Hub(v_0)	-0.8	3
3	Hub(v_0)	-0.2	2	11	Leaf(v_1)	-0.2	3	19	Leaf(v_1)	-0.6	1
4	Hub(v_0)	-0.4	2	12	Leaf(v_1)	-0.4	3	20	Leaf(v_1)	-0.8	1
5	Hub(v_0)	-0.2	3	13	Hub(v_0)	-0.6	1	21	Leaf(v_1)	-0.6	2
6	Hub(v_0)	-0.4	3	14	Hub(v_0)	-0.8	1	22	Leaf(v_1)	-0.8	2
7	Leaf(v_1)	-0.2	1	15	Hub(v_0)	-0.6	2	23	Leaf(v_1)	-0.6	3
8	Leaf(v_1)	-0.4	1	16	Hub(v_0)	-0.8	2	24	Leaf(v_1)	-0.8	3

Table 2: Illustration of the LPC model within simulation run 9 in Table 1: applying alpha values $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (-0.2, 0, 0, 0)$ at leaf node $\tilde{v} = v_1$ in the range [101, 200] seconds.

Node	α in Time Segment $\tilde{\tau}$		
	1-100 s	101-200 s	201-300 s
v_0	0	0	0
v_1	0	-0.2	0
v_2	0	0	0

instances for each set of inputs. Before starting each instance, initial conditions are reset. Then, each instance consists of looping over time in one-second intervals, from 0 to 300 seconds (0 seconds corresponds to the initial conditions). At each time, each agent executes its model, represented by the two rows of five boxes each in Figure 4. These boxes break down into three groups of activities. First, requests and replies from other agents, sent in the previous step, are received by the appropriate agents at the beginning of the agent's activities and messages are put on queues (boxes 1-3). Second, the probability computations of Section 3.1 are completed and the next action for an agent is determined (boxes 4-6). Third, the agent executes the action, e.g., if the action is $a_j = a_3$, then the agent requests a letter from one of its neighbors (see boxes 7-10). $\boldsymbol{\alpha}$ is applied only to agents representing VAs (see last box, first row). We use fixed time steps of one second, for the 300-second game, because the experimental (game) data show that the granularity of successive actions by players is more than one second in the vast majority of the data. Simulation outputs include actions of agents in time, from which counts of words formed, and requests and replies made, can be computed.

3.4 Functional ANOVA for Analyzing ABS Data

To analyze the effects of the three factors (node $\tilde{v}$, alpha value α, and time segment $\tilde{\tau}$) on player behavior in the LPC model, we adopt FANOVA, a technique for investigating differences in functional data across different conditions. Suppose there are n replications for each simulation. Then the linear model for the main effect of these three factors is

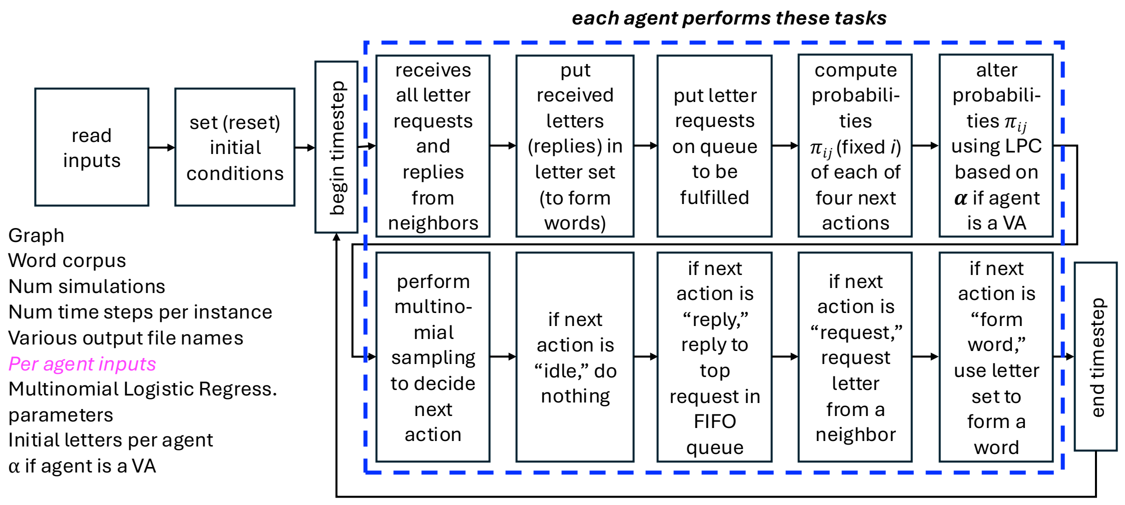

each agent performs these tasks
read inputs
set (reset) initial conditions
begin timestep
receives all letter requests and replies from neighbors
put received letters (replies) in letter set (to form words)
put letter requests on queue to be fulfilled
compute probabilities π_{ij} (fixed i) of each of four next actions
alter probabilities π_{ij} using LPC based on α if agent is a VA
perform multinomial sampling to decide next action
if next action is "idle," do nothing
if next action is "reply," reply to top request in FIFO queue
if next action is "request," request letter from a neighbor
if next action is "form word," use letter set to form a word
end timestep
Graph
Word corpus
Num simulations
Num time steps per instance
Various output file names
Per agent inputs
Multinomial Logistic Regress. parameters
Initial letters per agent
α if agent is a VA

Figure 4: Flowchart of LPC-ABM simulations. Inputs are read (listed below the left-most box), and initial conditions are reset for the first (or next) simulation instance. The first (or next) time step begins. The sequence of ten possible events are executed by each agent (see header above the ten boxes). When the time step ends, if the maximum number of time steps is not reached for the current simulation instance, then control goes to begin the next time step. Otherwise, control goes from end time step to the reset initial conditions, as preparation for the next simulation instance (arrow not shown to reduce clutter).

$$y_{ijkl}(t) = \eta + \delta_i(t) + \beta_j(t) + \gamma_k(t) + \varepsilon_{ijkl}(t), \quad t = 1,...,300,$$
$$i = 1,..,a, \quad j = 1,..,b, \quad k = 1,..,c, \quad l = 1,...,n,$$

where $y_{ijkl}(t)$ represents the analyst-specified outcome (total number of words formed by all players, in our case) for the lth replicate at time t, under the ith level of $\tilde{v}$, the jth level of α, and the kth level of $\tilde{\tau}$; δ_i denotes the effect of the ith level of $\tilde{v}$, with $a = 2$ levels: $\tilde{v} = v_0$ and $\tilde{v} = v_1$; β_j denotes the effect of the jth level of α, with $b = 4$ levels: $\alpha = -0.2, -0.4, -0.6$, and -0.8; γ_k denotes the effect of the kth level of $\tilde{\tau}$, with $c = 3$ levels: $\tilde{\tau} = 1$, 2, and 3; and ε_{ijkl} is an error term, assumed to be independently and identically distributed $N(0, \sigma^2)$. In this work, $n = 100$ replicates.

FANOVA is used to test whether there is significant difference in functional data when factors are at different levels over time. There are two types of tests we employ in this study.

1. Main effect test: This test aims to determine whether a particular factor has significant impact on the outcome. For instance, to assess the effect of the LPC on node $\tilde{v}$ regarding the number of words formed over time, the null hypothesis H_0 can be written as

$$H_0 : \delta_1(t) = \delta_2(t), \quad t = 1,\ldots,300.$$

The alternative hypothesis H_1 states that at least one level of $\delta_i(t)$ is different from others over the time range. In this example, $\delta_1(t)$ and $\delta_2(t)$ represent the unknown effects of the factor node $\tilde{v}$, and the estimation and hypothesis testing are conducted using observed functional response $y_{ijkl}(t)$ (total number of words formed by all players as a function of time t). The detailed procedures are illustrated in (Górecki and Smaga 2019). The main effect tests are conducted for all three factors.

2. Pairwise test: Pairwise test compares the LPC outcomes to the baseline case (without LPC) represented by $y_{000l}(t)$, $l = 1,\ldots,n$. The null hypothesis H_0 is

$$H_0 : \mu_{ijk}(t) = \mu_{000}(t), \quad t = 1,\ldots,300,$$

where $\mu_{ijk}(t) = \sum_{l=1}^{n} y_{ijkl}(t)/n$ and $\mu_{000}(t) = \sum_{l=1}^{n} y_{000l}(t)/n$ are the mean of the functional data over n replications. The pairwise test is conducted across all 24 simulation conditions. For example, if we want to compare the case of the VA being the hub node with $\alpha = \alpha_1 = -0.6$ being operative in the range [101, 200] seconds, with respect to the baseline, then the null hypothesis would be $H_0 : \mu_{132}(t) = \mu_{000}(t)$. The results of this pairwise test provide insight into whether there are statistically significant differences in the response (which is the words formed by all nodes as a function of t) for the experimental setting in Run 15 in Table 1 compared to baseline.

Through FANOVA, we aim to systematically assess the impact of various factors on time-dependent outcomes in the NGrAG. Investigating the main effect allows us to understand the individual impact of each factor on the outcome, while the pairwise test allows for direct comparisons between specific levels of the factors versus baseline.

3.5 Gaussian Process Emulator Based on LPC-ABM

The full factorial experimental design for ABSs proposed in Section 3.2 enables investigating the effects of various experimental settings on human and VA behaviors. However, the exhaustive enumeration of all possible conditions via LPC-ABM is impractical, particularly when our predictors are potentially time-dependent functions. In response to this limitation, we introduce the GP model as an emulator, which is a statistical model that enables us to extrapolate the full spectrum of VA and human behaviors under any experimental conditions as shown in Figure 2b. In our full factorial design, the GP model is defined as follows:

$$Y(x,t) = \beta_0 + Z(x,t) + \varepsilon, \quad t = 1,...,300,$$

where $Y(x,t)$ is the functional response of the total number of words formed by all the players, β_0 is the overall mean of the data, and $Z(x,t) \sim \mathbb{GP}(0, k(\cdot,\cdot))$ is a Gaussian process with mean 0 and covariance function $k(\cdot,\cdot)$ (e.g., $k(Y(x_1,t_1),Y(x_2,t_2)) = Cov(Y(x_1,t_1),Y(x_2,t_2))$) defined as a separable product function (Hung et al. 2015). The parameters in the covariance function are estimated through maximum likelihood estimation (MLE). Then the distribution of prediction at new input points (i.e., new sets of inputs) can be derived from conditional likelihood:

$$Y^*(x^*,t^*) \mid \mathscr{D} \sim \mathscr{N}(\beta_0 + \mu((x^*,t^*)), \sigma^2((x^*,t^*))) \,,$$

where $\mathscr{D}$ represents the set of all training data, $\mathscr{N}$ is a normal distribution, (x^*,t^*) represents the value of new input points, and $\mu((x^*,t^*))$, $\sigma^2((x^*,t^*))$ are the prediction mean and variance (Gramacy 2016).

Although the GP model is a highly robust predictive tool, it demands considerable computational resources (computation complexity of $\mathscr{O}(q^3)$). This is especially the case for time-dependent functional responses, where the number of data points (q) is equal to the number of time points (300) multiplied by the number of experimental settings (24). To mitigate this computational cost, we employ the local approximation for Gaussian Process (laGP), which dynamically defines the support of a Gaussian process predictor based on a local subset of the data (Gramacy 2016). The Leave-One-Out-Cross-Validation (LOOCV) is adopted to evaluate the predictive performance of laGP. The specific criterion we selected is the Nash-Sutcliffe efficiency (NSE) given by:

$$NSE = 1 - \frac{\sum_{t\in[1,300]}(\hat{y}_{pred}(t) - y_{test}(t))^2}{\sum_{t\in[1,300]}(y_{test}(t) - \bar{y}(t))} \,,$$

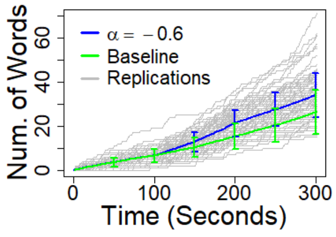
where $\hat{y}_{pred}(t)$ is the predicted function from the GP model for $t = 1,\ldots,300$, $y_{test}(t)$ is the function from one of the 24 runs left out and treated as test data, and $\bar{y}$ is the average of the test data. The NSE value represents an estimate of the proportion of variability in the response Y explained by the predictive model, analogous to R^2 as a performance metric in the linear model.

4 RESULTS

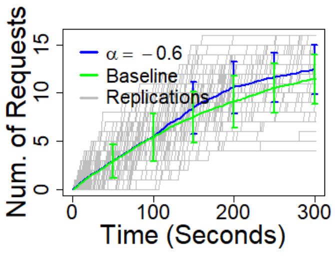
This section presents the ABS and GP emulator results from exercising the models of Section 3 for the NGrAG configuration of Figure 3.

4.1 LPC-ABS and FANOVA Results

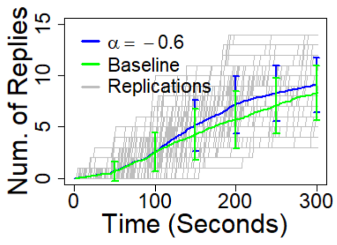
Figure 5 contains selected results from the FFD of Table 1. In all plots, the VA is node $\tilde{v} = v_0$, the hub node in Figure 3, with $\alpha = -0.6$ active during $\tilde{\tau} = 2$, i.e., during the time segment $101 \leq t \leq 200$ seconds of the 300-second game. Moving left to right across the figure, the plots are the time histories of number of words formed, of letter requests made, and of (letter) replies sent. Each blue curve is the cumulative total of all three agents in the graph, averaged over the 100 simulation instances; the individual instances are in gray. The baseline results (averaged) in green represent no effects of VAs. The blue and green curves deviate from each other near $t = 100$ seconds, when α is applied, as intuitively expected.



(a) $\alpha = -0.6$, Total Words



(b) $\alpha = -0.6$, Requests

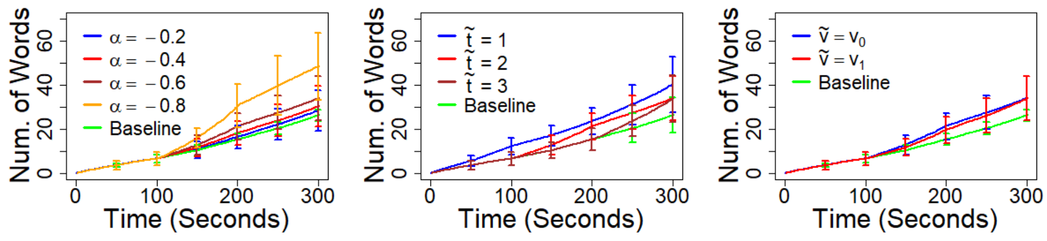


(c) $\alpha = -0.6$, Replies

Figure 5: LPC-ABM results for NGrAG and the network of Figure 3. Simulation conditions are: $\tilde{v} = v_0$; $\tilde{\tau} = 2$ (i.e., the VA has its α value non-zero in the time range $101 \leq t \leq 200$ seconds of the 300-second game); and $\alpha = -0.6$. (a) Numbers of words formed; (b) Numbers of letters requested; and (c) Numbers of replies to letter requests. All values on y-axes are the cumulative numbers over all three agents; the blue and green curves are averages over 100 simulation instances. The instances or replicates are in gray. Error bars are ± 1 standard deviation.

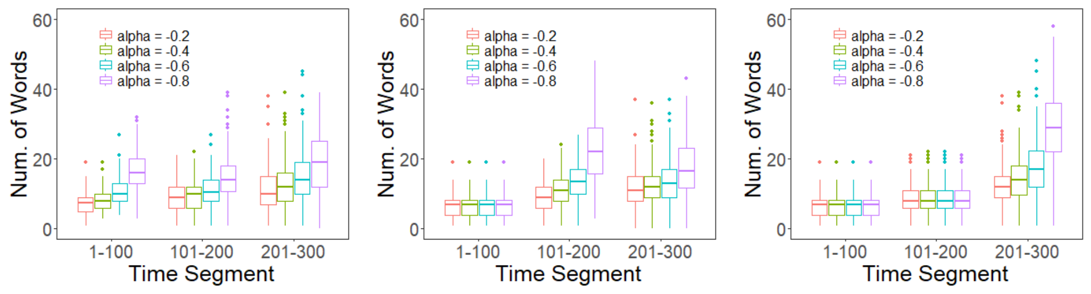
Figure 6 provides results for greater ranges of input conditions, focusing on the output of the cumulative number of words formed across all three agents. Figure 6a employs v_0 as the VA, and is active during $\tilde{\tau} = 2$ (i.e., over $101 \leq t \leq 200$ seconds). The VA becomes more active as α decreases from 0 to -0.8. A large change in results is observed in going from $\alpha = -0.6$ to -0.8, indicating that the effect of α is nonlinear. Figure 6b shows the effect of time segment $\tilde{\tau}$ over which $\alpha \neq 0$. The greatest effect is when α is applied in the first time segment $\tilde{\tau} = 1$, i.e., in the range $1 \leq t \leq 100$ seconds (in blue). The departure from baseline is immediate. The results for $\tilde{\tau} = 2$ and 3 show lesser effects, and the times at which the curves depart from baseline are in accord with the time segments. Interestingly, for $\tilde{\tau} = 2$, the behavior in the third time segment ($201 \leq t \leq 300$ seconds) is retarded to the point that the results for $\tilde{\tau} = 3$ data catch up. But these curves never catch up to the $\tilde{\tau} = 1$ curve. Figure 6c shows no effect of the location of the VA (whether hub node [$\tilde{v} = v_0$] or leaf node [$\tilde{v} = v_1$]).

The behavior of time segment $\tilde{\tau}$ on the results in Figure 6b is investigated in greater depth. Figure 7 presents the total words formed by the three agents within each time segment, for all four values of α. Each box represents results from the 100 simulation instances. These results demonstrate that the trends for the particular results in Figure 6b hold for all α, and although not shown, for all $\tilde{v}$, *on average*. That is, once a VA is active for one time segment of a simulation, the subsequent time segments show increased numbers of words formed, so there is a lingering or carry over effect of α. For example, in Figure 7a, VA is only operative in $\tilde{\tau} = 1$ (i.e., between $1 \leq t \leq 100$ seconds), but the plot shows that the players will form more words in each of the later time segments $101 \leq t \leq 200$ seconds and $201 \leq t \leq 300$ seconds. Also,



(a) G_{S2}, $\tilde{v} = v_0$, $\tilde{\tau} = 2$, $101 \le t \le 200$ (b) G_{S2}, $\tilde{v} = v_0$, $\alpha = -0.6$ (c) G_{S2}, $\alpha = -0.6$, $101 \le t \le 200$

Figure 6: LPC-ABM results highlighting the effects of (a) α value, (b) $\tilde{\tau}$ over which $\alpha \neq 0$, and (c) node that is the VA. The fixed variables are given below each plot. See text for explanation of results.



(a) α calibrated $\tilde{\tau} = 1$, $1 \le t \le 100$ (b) α calibrated $\tilde{\tau} = 2$, $101 \le t \le 200$(c) α calibrated $\tilde{\tau} = 3$, $201 \le t \le 300$

Figure 7: Number of words formed by all three agents in the graph for each $\tilde{\tau}$ and each α, when the VA is active in (a) $\tilde{\tau} = 1$, (b) $\tilde{\tau} = 2$, and (c) $\tilde{\tau} = 3$. Each plot contains the average results from two simulations: one where VA is the hub node (v_0) and one where the VA is a leaf node (v_1). Each plot contains three collections of boxes, corresponding the cumulative number of words formed in each of the time ranges [1, 100], [101, 200], and [201, 300]. The four boxes in each collection correspond to the four values of α. Results show that when a VA is active during one particular time segment, the effects continue, in the form of elevated numbers of words formed, in subsequent time segments.

the second group of boxes in Figure 7a, for $\tilde{\tau} = 2$, can be compared against the second group of boxes in Figure 7c where α still has not been applied. In this comparison we see that the results for $\alpha = -0.2$ are comparable, but that there is an appreciable difference for $\alpha = -0.8$. We write *on average* above because boxes are averages of results for $\tilde{v} = v_0$ and $\tilde{v} = v_1$ conditions; there are particular cases where there are not cumulative effects, as noted in Figure 6b for $\tilde{\tau} = 2$.

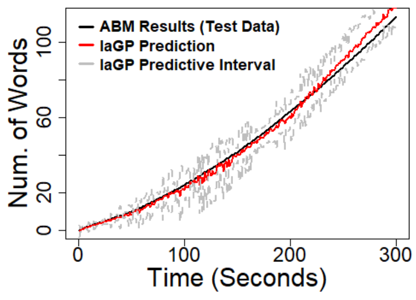
FANOVA, introduced in Section 3.4, is used to determine the significance of various factors on the number of words formed. The p-value is used as a numerical measure of whether there are significant differences between different levels of a factor. For the FFD specified in Table 1, the p-values for the factors $\tilde{v}$, α, and $\tilde{\tau}$ are 8.3×10^{-6}, 0, and 0, respectively, suggesting that all three factors have significant influence on the total words formed. Table 3 presents results for more detailed pair-wise tests between each run of Table 1 and the baseline, where there are no VAs (i.e., $\alpha = 0$). There are extremely small p-values for runs 13 through 24, and those for runs 2, 4, 8, and 10 are also small, suggesting that the magnitude of α is the most influential factor on the outcome. The large p-values for runs 5, 9, and 11 indicate that there is no significant different between the baseline and probability calibrated VA outcomes if the probability calibration is introduced in a later stage of the game (i.e., $\tilde{\tau} = 2$ and $\tilde{\tau} = 3$) with small magnitudes of α.

Table 3: P-values of pairwise tests for the 24 runs of Table 1. Results indicate that the magnitude of α is the most influential factor for forming words and that there is no significant difference between the baseline and VA-included simulations if the VA is introduced in a later stage with small magnitudes of α.

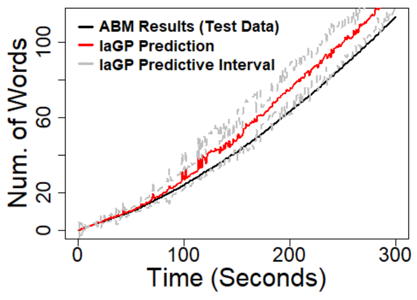
Run	p-value	Run	p-value	Run	p-value	Run	p-value	Run	p-value	Run	p-value
1	0.025	5	0.620	9	0.449	13	0.000	17	0.000	21	0.000
2	0.000	6	0.065	10	0.005	14	0.000	18	0.000	22	0.000
3	0.137	7	0.593	11	0.581	15	0.000	19	0.000	23	0.001
4	0.000	8	0.006	12	0.146	16	0.000	20	0.000	24	0.000

4.2 Gaussian Process Emulator Prediction Results

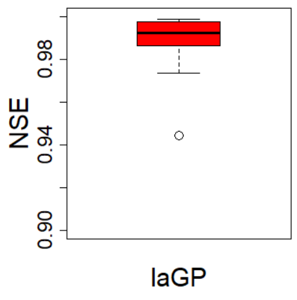
The FFD framework enabled by LPC-ABM and the FANOVA testing results have established solid conditions for predictive models. With GP models, we are able to interpolate and extrapolate to experimental conditions that have not been simulated. Since a GP model naturally makes predictions with uncertainty measure, it also facilitates the process of designing optimal experimental settings. Figures 8a and 8b illustrate the LOOCV prediction results for selected runs. The GP model makes predictions with mean point-wise values in red and with grey dashed lines as confidence bands (95% confidence interval); these bands contain the (black) simulation test data.



(a) Run 12 as test data



(b) Run 21 as test data



(c) LOOCV NSE Boxplot

Figure 8: Results of laGP predictions. (a) and (b): LPC-ABM data serve as input to construct a GPE. laGP predictions are given in red, with prediction intervals in gray. These intervals almost always contain the corresponding test data, in black. (c) NSE results summarizing the goodness of fit across all 24 cases of Table 1; results show good prediction capability.

Figure 8c contains an evaluation of the predictive performance of the laGP method, using Nash-Sutcliffe Efficiency (NSE), which is analogous to R^2 as a performance measure in a linear model. The laGP method has a cross-validated NSE of 0.94 or higher and most of the simulation runs have NSE close to 1. This means that the GP Emulator captures most of the variances in the observed data, suggesting high predictive performance. (NSE is based on the average curves, shown in red in Figures 8a and 8b, and hence the larger variances in laGP predictions do not affect NSE calculations.)

5 SUMMARY AND FUTURE WORK

In this work, we develop a framework of digital twins to enable the expansion of the input space to study heterogeneous agent behaviors. The developed methods have been executed for a networked group word formation game. As part of this input space expansion, we develop a Gaussian process method to emulate agent-based simulations and validate the emulator predictions against simulation results. Contributions of this work are provided in Section 1.3. There are many avenues for future work, such as improving emulators to predict other quantities (e.g., number of interactions of players in the game). This highlights an interesting aspect of the overall methodology: its flexibility in that methods and parameters can be changed out.

REFERENCES

Babu, S. and A. R. Kumar P. 2022. “A Comprehensive Survey on Simulators, Emulators, and Testbeds for VANETs”. *International Journal of Communication Systems* 35:e5123–1—e5123–32.

Bhatele, A., J.-S. Yeom, N. Jain, C. J. Kuhlman, Y. Livnat, K. R. Bisset *et al*. 2017. “Massively Parallel Simulations of Spread of Infectious Diseases over Realistic Social Networks”. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 689–694. New York, NY: Institute of Electrical and Electronics Engineers Press.

Boulesteix, A.-L., R. H. Groenwold, M. Abrahamowicz, H. Binder, M. Briel, R. Hornung *et al*. 2020. “Introduction to Statistical Simulations in Health Research”. *BMJ Open* 10(12):e039921.

Cadsby, C. B., F. Song, and F. Tapon. 2007. “Sorting and Incentive Effects of Pay for Performance: An Experimental Investigation”. *Academy of Management Journal* 50:387–405.

Carbajal, J. P., J. P. Leitão, C. Albert, and J. Rieckermann. 2017. “Appraisal of Data-driven and Mechanistic Emulators of Nonlinear Simulators: The case of Hydrodynamic Urban Drainage Models”. *Environmental Modelling & Software* 92:17–27.

Cedeno-Mieles, V., Z. Hu, Y. Ren, X. Deng, A. Adiga, C. Barrett *et al*. 2020. “Networked Experiments and Modeling for Producing Collective Identity in a Group of Human Subjects Using an Iterative Abduction Framework”. *Social Network Analysis and Mining (SNAM)* 10:1–43.

Charness, G., R. Cobo-Reyes, and N. Jimenez. 2014. “Identities, Selection, and Contributions in a Public-Goods Game”. *Games and Economic Behavior* 87:322–338.

Edali, M. and G. Yücel. 2019. “Exploring the Behavior Space of Agent-Based Simulation Models using Random Forest Metamodels and Sequential Sampling”. *Simulation Modelling Practice and Theory* 92:62–81.

Górecki, T. and Ł. Smaga. 2019. “fdANOVA: an R Software Package for Analysis of Variance for Univariate and Multivariate Functional Data”. *Computational Statistics* 34:571–597.

Gramacy, R. B. 2016. “laGP: Large-scale Spatial Modeling via Local Approximate Gaussian Processes in R”. *Journal of Statistical Software* 72:1–46.

Heppenstall, A., A. Crooks, N. Malleson, E. Manley, J. Ge and M. Batty. 2021. “Future Developments in Geographical Agent-Based Models: Challenges and Opportunities”. *Geographical Analysis* 53:76–91.

Hung, Y., V. R. Joseph, and S. N. Melkote. 2015. “Analysis of Computer Experiments with Functional Response”. *Technometrics* 57(1):35–44.

Kasim, M. F., D. Watson-Parris, L. Deaconu, S. Oliver, P. Hatfield, D. H. Froula *et al*. 2021. “Building High Accuracy Emulators for Scientific Simulations with Deep Neural Architecture Search”. *Machine Learning: Science and Technology* 3(1):015013:1–13.

Kohavi, R., R. Longbotham, D. Sommerfield, and R. M. Henne. 2009. “Controlled Experiments on the Web: Survey and Practical Guide”. *Data Mining and Knowledge Discovery* 18:140–181.

Liu, X., Z. Hu, X. Deng, and C. J. Kuhlman. 2023. “A Calibration Model for Bot-like Behaviors in Agent-Based Anagram Game Simulation”. In *2023 Winter Simulation Conference (WSC)*, 221–232 https://doi.org/10.1109/WSC60868.2023.10408394.

Matković, K., D. Gračanin, R. Splechtna, R. Splechtna, M. Jelović, B. Stehno *et al*. 2014. “Visual Analytics for Complex Engineering Systems: Hybrid Visual Steering of Simulation Ensembles”. *IEEE Transactions on Visualization and Computer Graphics* 20(12):1803–1812.

Nowke, C., S. Diaz-Pier, B. Weyers, B. Hentschel, A. Morrison, T. W. Kuhlen *et al*. 2018. “Toward Rigorous Parameterization of Underconstrained Neural Network Models Through Interactive Visualization and Steering of Connectivity Generation”. *Frontiers in Neuroinformatics* 12:21.

Pietzsch, B., S. Fiedler, K. G. Mertens, M. Richter, C. Scherer, K. Widyastuti *et al*. 2020. “Metamodels for Evaluating, Calibrating and Applying Agent-Based Models: A Review”. *Journal of Artificial Societies and Social Simulation* 23:1–12.

Swinerd, C. and K. R. McNaught. 2012. “Design Classes for Hybrid Simulations Involving Agent-Based and System Dynamics Models”. *Simulation Modelling Practice and Theory* 25:118–133.

Wate, P., M. Iglesias, V. Coors, and D. Robinson. 2020. “Framework for Emulation and Uncertainty Quantification of a Stochastic Building Performance Simulator”. *Applied Energy* 258:113759–1—113759–23.

AUTHOR BIOGRAPHIES

HAO HE is a Ph.D. student in the Department of Statistics at Virginia Tech. His email address is haoh@vt.edu.

XUEYING LIU is a Ph.D. student in the Department of Statistics at Virginia Tech. Her email address is xliu96@vt.edu.

CHRIS J. KUHLMAN is a Computational Scientist in ARC at Virginia Tech. His email address is ckuhlman@vt.edu.

XINWEI DENG is a Professor in the Department of Statistics at Virginia Tech. His e-mail address is xdeng@vt.edu.