A Methodology for Limit Cycle Detection in Simulation Models

Francesco Bertolotti[1][0000−0003−1274−9628] and Luca Mari[1][0000−0002−7128−3453]

LIUC - Università Cattaneo, Castellanza (VA) 21053, Italy
fbertolotti@liuc.it

Abstract. The exploration of model behavior is often a necessary step to validate and generate information from them, and it permits modelers and users to identify critical parameters. This is even more crucial when simulation techniques, such as agent-based models, are employed. The paper contributes to the discipline by proposing a novel exploration methodology to detect parameter configurations that generate limit cycles. It differs from pre-existing methodologies since it automatically explores the parameter space with a strategy that, once detected, investigates all the neighboring space until the limit cycle region ends.

Keywords: Model exploration · Limit cycles · Simulation · Agent-based model.

1 Introduction

The use of mathematical models to study the behavior of dynamic systems has a long history, particularly in some disciplines such as economics, biology and engineering [5]. Two notable examples are Adam Smith's model of economic growth [40], where it was posited that the sustained growth of the economy could result from the accumulation of capital and the division of labor, and Isaac Newton's model of universal gravitation [32], which described the motion of celestial bodies in terms of their masses and distances. At the beginning of the 20th century, one important development was the generalization of the mathematical meta-structure of models, particularly those based on the physical theories related to the tradition of Newtonian mechanics. This meta-structure typically involved systems of differential equations that described the evolution over time of state variables such as position, velocity, and acceleration. By developing more general mathematical frameworks for these models, researchers were able to apply them to a wider range of contexts.

The practice of mathematical modeling involves the creation of representations of real-world systems, that can be employed to understand and predict their behavior [15]. However, no model can perfectly capture the complexity and variability of a real-world system and no perfect knowledge can be grasp in advance regarding how the model works [10, 45]. It is then crucial to study the behavior of models as the parameters that govern their behavior are varied. It is then a way of assessing the validity of the model, reliability of the results, and to generates insight regarding the real-world model, which is the main goal even for abstract models [31, 27]. Moreover, this parameter variation could be used to detect emergent behaviours, which are typically hard to find, as well as explain how they happen [9].

In recent years, the development of large-scale models and computer-implemented simulation models has made the exploration of model behavior as parameters vary even more important [35]. Since these models can involve thousands or even millions of parameters, exploring the behavior of the model across this high-dimensional parameter space is a daunting task [21]. Fortunately, many techniques have been developed to do this task, including sensitivity analysis, optimization, and uncertainty quantification [11, 39, 18]. These techniques can help researchers and practitioners to identify the most important parameters, estimate their values more accurately, and assess the reliability of the model's predictions [44, 48].

When two time series results of represent on a 2-dimensional state space, one of the emerging features that can appear during a model exploration are limit cycles a feature of dynamical systems that have long captivated the attention of scientists and mathematicians alike [34, 25, 46]. These behavioral patterns arise when a system is driven to oscillate between two or more states over time, without settling into a stable equilibrium. They are an indicator of qualitative changes in the periodic behavior of dynamical systems when a distinguished parameter is varied. In some cases, the presence of these bifurcations may signal the emergence of chaos, leading to the sudden disappearance of periodic behavior [28, 36]. So, the emergence of oscillatory behavior provides essential information about the interactions within the underlying system. Despite their apparent simplicity, limit cycles can be hard to predict or understand. As a result, the study of limit cycles remains an active area of research across many different fields, from physics and chemistry to biology and economics [12, 37, 29]. Extensive research has been conducted in the field of bifurcations, examining a diverse range of dynamical systems that encompass the Hopf bifurcation. Typically, investigations of these phenomena have employed either maps or invariant measures of mathematical variables as the analytical tool.

In this paper, we propose a novel exploration methodology that permits the detection of parameter configurations that generate limit cycles in simulation models. This follows an existing need for new analytical tools for exploring the dynamic behavior of agent-based models (ABMs) [14]. While this methodology can potentially be employed for different simulation models that require the investigation of a parameter space to find non-punctual equilibria and are analytically intractable [13], traditional techniques already exist for equation-based models to investigate the presence of limit cycles. To the best of our knowledge, nothing similar exists for agent-based models and other non-formalized simulation models. In many cases, it could be relevant to discover more about the dynamics of an agent-based system, even when, as is often the case, the global behavior is too complex to be formally depicted by a set of differential equations. Moreover, in comparison to pre-existing techniques such as Lyapunov analysis

[26], Poincaré maps [34], or the Bendixson's Criterion [3], the main advantage of this methodology is its ability to enable multi-dimensional parameter space exploration. This eliminates the need for a person to run the model, evaluate the simulation output, and adjust the parameters to get a better result. This is especially relevant for complex non-equation-based models with unknown behavior.

The paper is divided as follows. In the next section, we review and discuss the main research regarding simulation model exploration, providing a background for this work. In the method section, we present both the detection algorithm (which is the main focus of this research) and the case study models on which we tested it. Then, we present the results of applying the algorithm to the case study models, and draw some conclusions.

2 Background

The idea of limit cycles originated from the well-known works of Poincaré [34] and was subsequently popularized by the 23 mathematical problems presented by David Hilbert at the Second International Congress of Mathematicians [22]. System Theory has deeply investigated the topic, especially finding the set of differential equations and the parameter settings that could make them happen [28]. Later, the need for detecting limit cycle-like behavior emerged in various application fields, even when the unit of analysis was not aggregate (and then represented with an equation-based model) but individual (so developed with an agent-based modeling), from the detection of fashion cycles [1] to understanding how environmental parameters affect prey-predator systems [?]. Moreover, evolutionary game theory (which can be considered, from a structural perspective, a specific configuration of agent-based modeling) has a long tradition of detecting limit cycles [20, 16].

While the methodology proposed in this work is original, it is far from new to study agent-based models by means of parameter space explorations [43], as an alternative to manually altering the parameters during long simulations to detect dynamical shifts [49]. Typically, these techniques consider the model from a black-box perspective, so what is "inside" the box (i.e., the model) is mostly ignored, and only the inputs U and the outputs Y are observed. Using this simplification, it is possible to observe two classes of simulation model exploration techniques. The first class is composed of methodologies that adjust the input and observe the subsequent effect on the output. The most famous of these models is called sensitivity analysis [38], which consists of evaluating the dependency of the model output on the model input (usually defining sensitivity as $\frac{\Delta y}{\Delta u}$) and investigating the role of each model input in the determination of the output to understand the principal factors [30, 38, 23]. Sensitivity analysis implicitly implies computing a measure (the sensitivity), but one could also be interested in exploring the effect on the output of specific combinations or studying the conditional variations between variables. In this case, the model can be simulated by sampling the target parameters from defined ranges of values and observing the outputs, and the goal would be to find a function f that connects U to Y (or two of their subsets). The simplest strategy for exploring a parameter space is called "sample grid sampling", which consists of exploring every combination of points of the parameter space [24, 8]. While this methodology permits not taking any preliminary stance regarding the potential outcome, and so it is especially suitable when there is no knowledge nor expectation towards the result of the simulation model, the computational time needed to complete it grows with the number of possible combinations, making it often unfeasible [14]. So, several techniques for enhancing sampling in numerical experiments under limited computational resources have been proposed, such as Sobol sequences or Latin hypercube sampling. The Sobol sequence is the sequence that minimizes the discrepancy for a set of parameters, where no fitting functions are used to decompose the output variance [41]. It only takes into account the average impact of parameters across the entire parameter space and does not examine various patterns within this space [11]. Sensitivity analysis can be used to address uncertainty regarding a single parameter, but it is not adequate to address uncertainty related to multiple parameters. To sample variations in multiple parameters without considering every possible permutation, one can use Latin hypercube sampling, which consists of taking one point in each dimension of a hypercube whose number of dimensions is equal to the number of parameters involved in the sampling [19, ?]. Other exploration techniques of this type include the quantification of tipping points where transitions between behavioral regimes occur due to external forcing, typical of closed biological systems [2].

The second class of exploration techniques regards the ones that fix a desired output and adjust the input to find the right combination to get the target behavior, calibrating the behavior of the model on a target. This is especially relevant for models of real-world systems, whose target is to be used for understanding or predicting their behavior [47].

For agent-based models of real-world systems with available data, the calibration of the behavior could be made by means of dynamic data assimilation, for example, using the ensemble Kalman filter (EnKF) to address the typical nonlinearity of ABM and their computational elevated cost [47]. Moreover, different studies highlight how to use genetic algorithms in agent-based modeling, either evaluating the possible methodologies [42] or applying to fit real-world behavior [7]. A comparison of economic agent-based model calibration methods also finds that Bayesian estimation consistently outperforms different other calibration methods in many contexts [33]. Moreover, there is a way to reduce the computational costs of both behavior and parameter explorations by learning a surrogate and approximate sub-model of the original model with a lower number of training points [50].

Finally, a methodology with an analog finality to the one presented in this paper was already tested on a prey-predator model [12], using Monte Carlo singular spectrum analysis to find statistically significant oscillatory patterns in the behavior of a prey-predator agent-based model. Nevertheless, the methodology is very computationally expensive and tailor-made for a specific setting whose analytical equations are known. In this paper, we are proposing a faster technique that can be generically applied to each analytical intractable simulation model with two outputs.

3 Methods

The code used to implement and test the methodogy depicted in this section is written in Python 3.9. For the sake of transparency and replicability, all the code used and the experiments employed can be found at the following link: https://github.com/francescobertolotti/limitcycledetection/.

3.1 Detection algorithm

The exploration methodology proposed in this paper presents many steps, depicted in Figure 1.

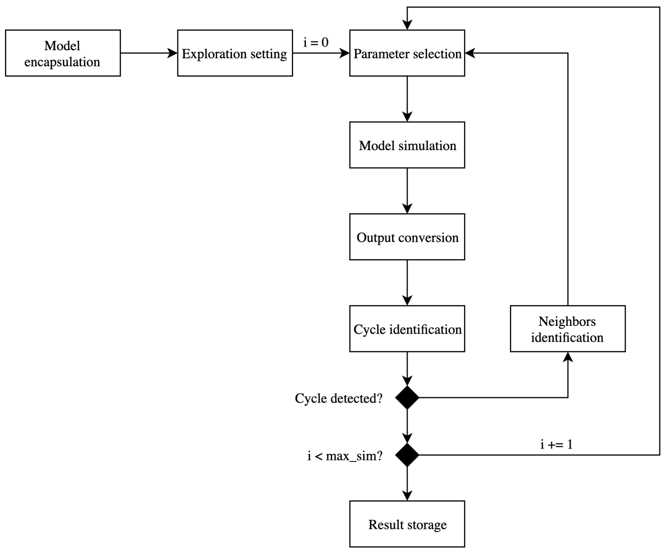

Fig. 1. The process of model exploration for limit cycles

Description Firstly, the target simulation model is encapsulated into a function, so that the underlying structure is ignored and it can be treated as a black box. This encapsulation permit to get a given output from a specific combination of parameters.

Next, the encapsulated model reads a combination of parameters as input to generate an output. These parameters can be generated in any fashion. In the case of numeric parameters distributed on a single order of magnitude, sampling from a uniform distribution is sufficient. However, if the range of available values of a parameter exceeds a single order of magnitude, the values of the parameter could be alternatively sampled from a loguniform distribution (the discussion about which sampling method is better exceeds the scope of this work). In this phase, it is crucial to clearly pinpoint a division of the parameter space so that the overall number of cases can be computed, and it is even possible to identify the size of the parameter space addressed at each exploration. Additionally, this specification later permits the recognition of the neighbors of a point in the parameter space that generates a limit cycle once the model is run. At the end of this step, a sampling strategy is defined, from which a set of parameters can be randomly generated at each time and fed to the model.

Subsequently, a loop starts where the number of iterations is a parameter of the exploration meta-model. In the loop, a set of parameters is randomly generated (using the previously defined strategy) and provided as input to the simulation model. From this simulation, a two-dimensional time-series output is generated. The output is then converted into a boolean matrix of size m (Figure 2 presents an example of the transformation process), where m is a parameter of the exploration algorithm discussed at the end of this section. The compression algorithm consists of two phases. First, it takes the original two-dimensional trajectory and makes it denser so that for each consecutive point, a fixed number of intermediate points is created. This operation is required so that even if the trajectory makes a long jump between two consecutive points, the path is completely fulfilled. Secondly, the graphical representation of the line in two dimensions and the matrix are superimposed in such a manner that the upper and lower boundaries of the line coincide with the extremes of the matrix. At this point, each point of the line is reported inside the matrix, marking each cell of the matrix where the line is passing with a 1, and a 0 otherwise. Figure 2 depicts this output conversion process.

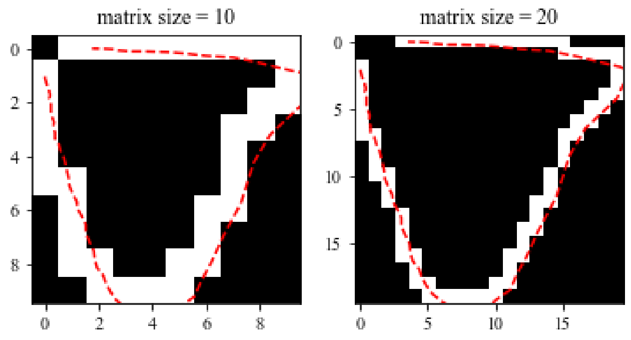

Fig. 2. The Figure presents the comparison between the two-dimensional behaviors of a single run of the Lotka-Volterra case study model (depicted by the red dotted line) and its conversion two matrices with different sizes. When the size of the matrix is equal to 10, it is possible to observe how a long trajectory is actually confused with a limit cycle. However, this does not happen in the case of a higher definition.

After transforming the two-dimensional trajectory of the model's behavior into a matrix, an algorithm is used to evaluate the size of the cycle. The cycle dimension, denoted as C, can be defined as the number of cells in the largest area of cells with a value of 0 around which there is a continuous path of cells with a value of 1. The size of the cycle is then compared to the overall area of the matrix, obtaining a value c between 0 and 1.

At this point, whether or not the cycle is detected depends on two elements. The first element is the sensitivity of the detection, denoted by s, which is the threshold above which c is considered sufficiently high to indicate the presence of a limit cycle. In the case studies presented later in this paper, two different values of s are proposed. Additionally, if the model is stochastic, more than one simulation may be needed to accurately assess the value of c, in order to avoid missing configurations that return limit cycles only a fraction of the time. In this case, the trigger for further exploration of the surroundings of the detected point is not $c > s$, but rather $E[c] > s$, where $E[c]$ is the expected value of c. To avoid being misled by noise resulting from stochasticity and equifinality, which is the case where different rules and parameter settings lead to the same model output [17], it is necessary to repeat the experiment a statistically valid number of times for each parameter combination to ensure the robustness of the results.

So, if the condition is not satisfied, the cycle loop is exited and another parameter is randomly selected. Otherwise, the neighbors of the current point in the parameter space are identified based on the division of the parameter space defined in the second step of the process. Once the neighbors are identified, they are stored in a list and the exploration process is repeated for each neighbor, following the same cycle as before. This process continues as long as there are points in the list of neighbors that have not been explored yet. If a limit cycle is detected in any of the neighbors of the previously explored point, new points are added to the exploration list. This ensures that once a single point in the parameter space that generates a limit cycle in the encapsulated model is detected, the entire n-dimensional area around it is explored, and this exploration continues until no new points with limit cycles are found.

Finally, whenever the number of explorations i gets equal to the maximum number of simulations max_sim, the exploration stops and the results are stored in a dataframe. In this sense, this exploration meta-model can be seen as a black-box, which read in input an encapsulated model and ranges of a set of allowable values for each parameter, along with the distribution from which to sample them, and returns a table which columns are the parameters and the correspondent c (or $E[c]$, if the model is stochastic). Figure 3 presents this interpretation.

Limitations The proposed methodology has some limitations and requirements for its application. Firstly, it can address both deterministic and stochastic models, since most simulation models have some degree of stochasticity, especially those that are adherent to reality. Nevertheless, the whole process is considerably less expensive in terms of computation required to explore the space when

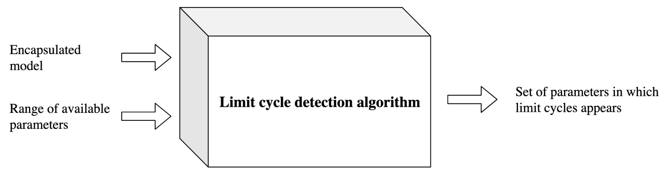
Encapsulated model
Range of available parameters
Limit cycle detection algorithm
Set of parameters in which limit cycles appears

Fig. 3. A representation of the black box interpretation of the limit cycle exploration methodology

used with deterministic models. Whenever the model studied is deterministic or stochastic, the high sensitivity to initial conditions typically affects the result of a single run [6, 4]. Even more problematic in terms of computational cost is addressing models with high dimensionality, due to the so-called "curse of dimensionality." This can be addressed by using a coarser subdivision of the parameter space.

Secondly, the methodology only considers two numeric outputs, allowing them to be represented in a two-dimensional state diagram. This limitation permits the graphical depiction of the behavior and identification of the hole.

Moreover, appropriate setting of the cycle detection threshold and the matrix granularity is required. A value that is too large or too small could compromise the achievement of the final result. Regarding granularity, a value that is too large could make trajectories resemble cycles even when they have not closed, while a value that is too small could create gaps between points, breaking the cycle and preventing the algorithm from identifying it later. On the other hand, a threshold for identification that is too large does not allow any limit cycle to be identified (trivially, $c < 1$ in any case since the area of the circle bounded by the curve is necessarily lower than that of the square because there will be matrix cells that delimit the perimeter of the figure), while a threshold that is too low captures configurations of parameters that create gaps in the state space between various lines (typically, at least pseudo-chaotic behavior), rendering the automatic recognition procedure unusable (which is, in fact, the true strength of this methodology).

3.2 Case study models

These are two models whose characteristics make them good candidates for applying the new methodology proposed in this paper. First, their behavior could not be derived analytically. Second, they are well-known in the scientific literature, so there is no need to discuss the results of the exploration, as the purpose of the case study exploration is solely to show that the procedure presented in the previous section works. Third, they are structurally different, as the scheduling strategy and the manner in which agents interact with each other vary. Fourth, they can be deeply simplified, so that a primitive version could be implemented for the mere scope to test the algorithm on it.

Prey-predator agent-based model In the model, there are two types of entities positioned on a toroidal surface: preys and predators. Each agent is defined by a two-dimensional spatial location. Both agents move at every time step, and the size and direction of the movement are generated from a random uniform distribution $U \sim Unif(-1, 1)$. Finally, both agents can asexually reproduce, so that each prey and predator have a given probability, at every time step, to generate another prey or another predator agent. There are two differences between predator and prey agents. Firstly, predator agents can prey on prey agents. More specifically, whenever a predator has preys at a distance lower than a given threshold, it chooses one of them and eats it. While doing so, it acquires a fixed amount of energy. Furthermore, predators have energy levels, which decrease at a constant rate at each time step. When the level gets to 0, predators die. Whenever a predator breeds, half of its energy is given to the bred agent, so that the overall amount of energy in the system does not increase with a reproduction event. The pseudo-code in Algorithm 1 describes the scheduling of the model.

Algorithm 1 Main Loop of the Simulation

```
1: for $t$ in $[1...t_{max}]$ do
2:     shuffle PREDATORSLIST
3:     shuffle PREYSLIST
4:     for PREY in PREYSLIST do
5:         PREY move randomly
6:     for PREDATOR in PREDATORSLIST do
7:         PREDATOR move randomly
8:         if any? PREY in PREYSLIST at distance $< 2$ then
9:             energy of PREDATOR $+=$ energy per hunt
10:            set PREYTARGET one of preys in PREYSLIST at distance $< 2$
11:            die PREYTARGET
12:        energy of PREDATOR -= energy consumption
13:        if energy of PREDATOR $<= 0$ then
14:            die PREDATOR
       new preys = int(length(PREYSLIST) $\times$ growth rate prey)
15:    for $i$ in $[i...newpreys]$ do
16:        create NEWPREY
17:        append NEWPREY to PREYSLIST
       new predators = int(length(PREDATORSLIST) $\times$ growth rate predators)
18:    for $i$ in $[i...newpreys]$ do
19:        set PREDATORFATHER one of predators in PREDATORSLIST
20:        create NEWPREDATORS
21:        energy of NEWPREDATORS = energy of PREDATORFATHER $\times$ 0.5
22:        energy of PREDATORFATHER = energy of PREDATORFATHER $\times$ 0.5
23:        append NEWPREDATORS to PREDATORSLIST
```

Table 1. Parameters used in the exploration of the prey-predator model

Parameter	Min value	Max Value	Interval between values
$e_p y$	0	10	1
pd_g	0	0.1	0.01
py_c	0.2	0.8	0.1
py_g	0	0.2	0.01

Prisoner's dilemma on a network The code defines an Agent-Based Model where agents play the Prisoner's Dilemma, a classic game theory model of strategic interaction. In the Prisoner's Dilemma, two individuals are arrested for a crime, and the police offer each of them a plea deal to testify against the other. If both individuals stay silent, they will be convicted for a lesser crime, but if one confesses and the other stays silent, the confessor will go free, and the silent one will be punished harshly. If both confess, they will receive a moderate sentence. The prisoner's dilemma is valid with different settings (as long as $T > R > P > S$). Table 2 depicts the payoff configuration employed in this implementation, where the specific values are input parameters of the model.

Table 2. Payoff Matrix for the Prisoner's Dilemma

	Cooperate	Defect
Cooperate	(R, R)	(S, T)
Defect	(T, S)	(P, P)

The model consists of n agents connected to each other by means of a complete network, which is a network in which every node is connected to every other node. At every round, two random agents are picked to play the prisoner's dilemma against each other. As in the classic prisoner's dilemma configuration, agents are endowed with a strategy that can be cooperative or defective. An agent can start the simulation being cooperative or defective with a probability defined at the beginning of the game, so that the expected share of agents starting with a cooperative strategy is given by the input parameter c. The strategy of any agent can change over time, since each agent has a memory of previous rounds. During each round, two agents are randomly selected to play the game, and their payoffs are updated based on their chosen strategy (as in Table 2). If both agents cooperate, they receive a reward (R); if one cooperates and the other defects, the defector receives the highest reward (T), and the cooperator receives the lowest reward (S); if both defect, they receive a punishment (P). After each round, the agents adjust their strategies based on their average payoffs. Specifically, if an agent's payoff is below the average payoff of all agents, the agent has a probability of switching to the other strategy. The model continues for a given number of rounds T (a parameter of the model). Algorithm 2 presents the pseudo-code of this implementation of the prisoner's dilemma.

Table 3. Parameters used in the exploration of the Prisoner Dilemma model

Parameter	Min value	Max Value	Interval between values
m	1	10	1
R	0	3	1
S	0	3	1
T	-3	0	1
P	3	5	1

Algorithm 2 Main Loop of the Simulation

```
1: for $t$ in $[1...t_{max}]$ do
2:     set PLAYER1 one of agents in AGENTSLIST
3:     set PLAYER2 one of agents in AGENTSLIST
4:     payoff of PLAYER1 = f(strategy of PLAYER1, strategy of PLAYER2, payoff matrix)
5:     payoff of PLAYER2 = f(strategy of PLAYER1, strategy of PLAYER2, payoff matrix)
6:     append payoff of PLAYER1 to payofflist of PLAYER1
7:     append payoff of PLAYER2 to payofflist of PLAYER2
8:     for AGENT in AGENTSLIST do
9:         meanpayoff of AGENT = mean(payofflist of AGENT)
       mean payoffs = mean(meanpayoff of AGENTS in AGENTSLIST)
10:    if meanpayoff of PLAYER1 $<$ mean payoffs then
11:        set strategy of PLAYER1 one of possible strategies chosen randomly
12:    if meanpayoff of PLAYER2 $<$ mean payoffs then
13:        set strategy of PLAYER2 one of possible strategies chosen randomly
14:    remove last element to payofflist of PLAYER1
15:    remove last element to payofflist of PLAYER2
```

4 Case studies results

This section presents the results from the application of the proposed methodology to the case study simulation models. The results in this section are not discussed, as it is not within the scope of the paper to study the behavior of the models or derive insights from them. Instead, we aim to demonstrate the applicability of the methodology and highlight any differences encountered during the exploration.

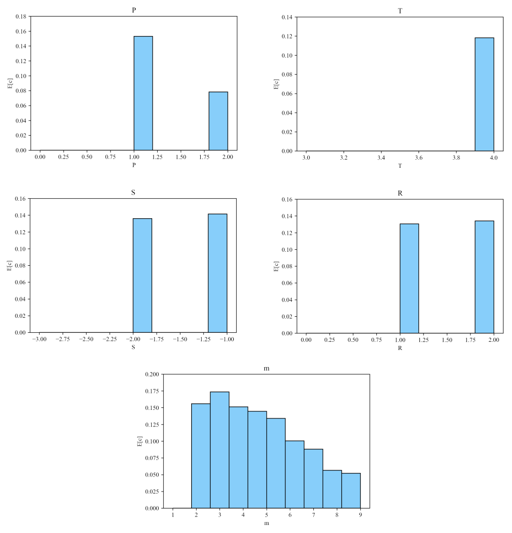

Fig. 4. Results of the limit cycles exploration for the prey predator model

The results generated from the proposed methodology are presented in tabular format, collecting the results from a 2000 simulation limit cycle exploration. Figure 4 and Figure 5 illustrate the aggregation of the results, presenting the parameter space points where limit cycles appear. The methodology was ap-

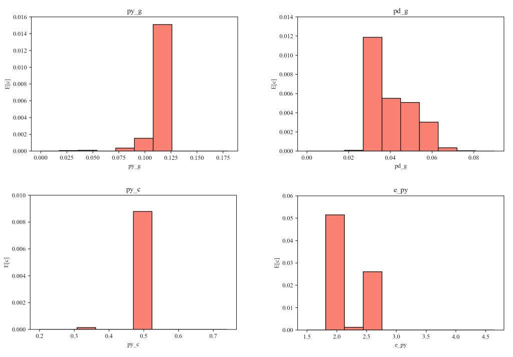

Fig. 5. Results of the limit cycles exploration for the prisoner dilemma model

plied to two different models to demonstrate its applicability, and the results are structurally similar. Specifically, both models exhibit the emergence of limit cycles, and parameters differently affect this phenomenon. The distribution of the results, such as for the variable m in the Prisoner's Dilemma model, indicates that the appearance of limit cycles is less sensitive to the presence of a given parameter. However, for some parameters, such as P in the Prisoner's Dilemma model or py_c in the Prey-Predator model, limit cycles are only detectable when the parameter has a specific value. This insight sheds light on the elements that generate an orbit in the state space, and is affected by the granularity of the parameter space.

5 Conclusions

After briefly reviewing the existing literature on simulation model exploration, this paper presented a method to detect limit cycles in non-formalized simulation models. The proposed methodology was described in detail, and its application to two case study models was presented. This methodology has great potential for enhancing the exploration of simulation models, particularly agent-based or complex equation-based models without formalization.

However, the proposed methodology has three main limitations. Firstly, it can only consider two output variables per experiment. Secondly, it can be computationally expensive, especially for high-dimensional models that require exploration with high granularity. Thirdly, it does not guarantee finding a result unless the entire parameter space is explored. Nonetheless, it represents an advance in the state-of-the-art, particularly for low-dimensional models.

Further developments could involve testing the methodology on simulation models with a higher number of parameters, to assess its feasibility and performance. Additionally, a new version of the algorithm could be implemented to compute the exploration direction following a gradient descent.

References

1. Apriasz, R., Krueger, T., Marcjasz, G., Sznajd-Weron, K.: The hunt opinion model-an agent based approach to recurring fashion cycles. PLoS ONE (2016). https://doi.org/10.1371/journal.pone.0166323
2. Ashwin, P., Wieczorek, S., Vitolo, R., Cox, P.: Tipping points in open systems: Bifurcation, noise-induced and rate-dependent examples in the climate system. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences (2012). https://doi.org/10.1098/rsta.2011.0306
3. Bendixson, I.: Sur les courbes définies par des équations différentielles. Acta Mathematica **24**(1), 1 (1901). https://doi.org/10.1007/BF02403068, https://doi.org/10.1007/BF02403068
4. Berceanu, C., Patrascu, M.: Initial Conditions Sensitivity Analysis of a Two-Species Butterfly-Effect Agent-Based Model. In: Baumeister, D., Rothe, J. (eds.) Multi-Agent Systems. pp. 60–78. Springer International Publishing, Cham (2022)
5. von Bertalanffy, L.: General system theory: Foundations, development, applications. G. Braziller (1968)
6. Bertolotti, F., Locoro, A., Mari, L.: Sensitivity to Initial Conditions in Agent-Based Models. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2020). https://doi.org/10.1007/978-3-030-66412-1_32
7. Bertolotti, F., Roman, S.: Risk sensitive scheduling strategies of production studios on the US movie market: An agent-based simulation. Intelligenza Artificiale **16**, 81–92 (2022). https://doi.org/10.3233/IA-210123
8. Bertolotti, F., Roman, S.: The Evolution of Risk Sensitivity in a Sustainability Game: an Agent-based Model (2022)
9. Bodine, E.N., Panoff, R.M., Voit, E.O., Weisstein, A.E.: Agent-Based Modeling and Simulation in Mathematics and Biology Education. Bulletin of Mathematical Biology (2020). https://doi.org/10.1007/s11538-020-00778-z
10. Box, G.E.P.: Science and statistics. Journal of the American Statistical Association **71**(356), 791–799 (1976)
11. ten Broeke, G., van Voorn, G., Ligtenberg, A.: Which sensitivity analysis method should i use for my agent-based model? JASSS (2016). https://doi.org/10.18564/jasss.2857
12. Colon, C., Claessen, D., Ghil, M.: Bifurcation analysis of an agent-based model for predator-prey interactions. Ecological Modelling (2015). https://doi.org/10.1016/j.ecolmodel.2015.09.004
13. Cranmer, K., Brehmer, J., Louppe, G.: The frontier of simulation-based inference. Proceedings of the National Academy of Sciences of the United States of America (2020). https://doi.org/10.1073/pnas.1912789117
14. Daly, A.J., De Visscher, L., Baetens, J.M., De Baets, B.: Quo vadis, agent-based modelling tools? Environmental Modelling & Software p. 105514 (2022)
15. Epstein, J.M.: Why model? Journal of artificial societies and social simulation **11**(4), 12 (2008)
16. Ficici, S.G., Melnik, O., Pollack, J.B.: A game-theoretic and dynamical-systems analysis of selection methods in coevolution. IEEE Transactions on Evolutionary Computation **9**(6), 580 – 602 (2005). https://doi.org/10.1109/TEVC.2005.856203
17. Gräbner, C.: How to relate models to reality? An epistemological framework for the validation and verification of computational models. JASSS (2018). https://doi.org/10.18564/jasss.3772

18. Hales, D., Rouchier, J., Edmonds, B.: Model-to-model analysis. JASSS (2003)
19. Hamis, S., Stratiev, S., Powathil, G.G.: Uncertainty and Sensitivity Analyses Methods for Agent-Based Mathematical Models: An Introductory Review, chap. Chapter 1, pp. 1–37 (2020). https://doi.org/10.1142/9789811223495_0001, https://www.worldscientific.com/doi/abs/10.1142/9789811223495_0001
20. Hauert, C., Wakano, J.Y., Doebeli, M.: Ecological public goods games: Cooperation and bifurcation. Theoretical Population Biology **73**(2), 257 – 263 (2008). https://doi.org/10.1016/j.tpb.2007.11.007
21. Heppenstall, A.J., Evans, A.J., Birkin, M.H.: Genetic algorithm optimisation of an agent-based model for simulating a retail market. Environment and Planning B: Planning and Design (2007). https://doi.org/10.1068/b32068
22. Hilbert, D.: Sur les problèmes futurs des Mathématiques. In: 1900 International Congress of Mathematicians. Paris (1900)
23. Iooss, B., Saltelli, A.: Introduction to Sensitivity Analysis, pp. 1103–1122. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-12385-1_31, https://doi.org/10.1007/978-3-319-12385-1_31
24. Klabunde, A.: Computational economic modeling of migration. In: The Oxford Handbook of Computational Economics and Finance (2018). https://doi.org/10.1093/oxfordhb/9780199844371.013.41
25. Lotka, A.J., Others: Elements of physical biology (1925)
26. Lyapunov, A.M.: The general problem of the stability of motion. International Journal of Control **55**(3), 531–534 (1992). https://doi.org/10.1080/00207179208934253, https://doi.org/10.1080/00207179208934253
27. Matsiuk, V., Galan, O., Prokhorchenko, A., Tverdomed, V.: An Agent-Based Simulation for Optimizing the Parameters of a Railway Transport System. In: ICTERI. pp. 121–128 (2021)
28. May, R.M.: Simple mathematical models with very complicated dynamics. Nature (1976). https://doi.org/10.1038/261459a0
29. Mittal, S., Mukhopadhyay, A., Chakraborty, S.: Evolutionary dynamics of the delayed replicator-mutator equation: Limit cycle and cooperation. Physical Review E **101**(4), 42410 (2020)
30. Morris, M.D.: Factorial sampling plans for preliminary computational experiments. Technometrics (1991). https://doi.org/10.1080/00401706.1991.10484804
31. Murase, Y., Jo, H.H., Török, J., Kertész, J., Kaski, K.: Deep Learning Exploration of Agent-Based Social Network Model Parameters. Frontiers in big Data **4**, 739081 (2021)
32. Newton, I.: Philosophiae naturalis principia mathematica, vol. 1. G. Brookman (1833)
33. Platt, D.: A comparison of economic agent-based model calibration methods. Journal of Economic Dynamics and Control (2020). https://doi.org/10.1016/j.jedc.2020.103859
34. Poincaré, H.: Mémoire sur les courbes définies par une équation différentielle (I). Journal de Mathématiques Pures et Appliquées **7**, 375–422 (1881), http://eudml.org/doc/235914
35. Raimbault, J., Cottineau, C., Le Texier, M., Le Néchet, F., Reuillon, R.: Space matters: Extending sensitivity analysis to initial spatial conditions in geosimulation models. JASSS (2019). https://doi.org/10.18564/jasss.4136
36. Robbio, F.I., Alonso, D.M., Moiola, J.L.: Detection of limit cycle bifurcations using harmonic balance methods. International Journal of

Bifurcation and Chaos in Applied Sciences and Engineering (2004). https://doi.org/10.1142/S0218127404011491
37. Roman, S., Bullock, S., Brede, M.: Coupled Societies are More Robust Against Collapse: A Hypothetical Look at Easter Island. Ecological Economics (2017). https://doi.org/10.1016/j.ecolecon.2016.11.003
38. Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S.: Global Sensitivity Analysis. The Primer (2008). https://doi.org/10.1002/9780470725184
39. Schouten, M., Verwaart, T., Heijman, W.: Comparing two sensitivity analysis approaches for two scenarios with a spatially explicit rural agent-based model. Environmental Modelling and Software (2014). https://doi.org/10.1016/j.envsoft.2014.01.003
40. Smith, A.: The wealth of nations. na (1776)
41. Sobol, I.M.: Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. Mathematics and Computers in Simulation (2001). https://doi.org/10.1016/S0378-4754(00)00270-6
42. Stonedahl, F.J.: Proposal - Genetic Algorithms for the Exploration of Parameter Spaces in Agent-Based Models. ProQuest Dissertations and Theses (2011)
43. Terano, T.: Exploring the vast parameter space of multi-agent based simulation. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2007). https://doi.org/10.1007/978-3-540-76539-4_1
44. Troost, C., Huber, R., Bell, A.R., van Delden, H., Filatova, T., Le, Q.B., Lippe, M., Niamir, L., Polhill, J.G., Sun, Z., Others: How to keep it adequate: A protocol for ensuring validity in agent-based simulation. Environmental Modelling & Software **159**, 105559 (2023)
45. Vermeer, W.H., Smith, J.D., Wilensky, U., Brown, C.H.: High-fidelity agent-based modeling to support prevention decision-making: An open science approach. Prevention Science pp. 1–12 (2021)
46. Volterra, V.: Variazioni e fluttuazioni del numero d'individui in specie animali conviventi, vol. 2. Societá anonima tipografica" Leonardo da Vinci" (1926)
47. Ward, J.A., Evans, A.J., Malleson, N.S.: Dynamic calibration of agent-based models using data assimilation. Royal Society Open Science (2016). https://doi.org/10.1098/rsos.150703
48. Wikstrom, K., Nelson, H.T.: Spatial Validation of Agent-Based Models. Sustainability **14**(24), 16623 (2022)
49. Woods, J., Perilli, A., Barkmann, W.: Stability and predictability of a virtual plankton ecosystem created with an individual-based model. Progress in Oceanography (2005). https://doi.org/10.1016/j.pocean.2005.04.004
50. Zhang, Y., Li, Z., Zhang, Y.: Validation and Calibration of an Agent-Based Model: A Surrogate Approach. Discrete Dynamics in Nature and Society (2020). https://doi.org/10.1155/2020/6946370