$EcoSim_{py}$: an Open, Collaborative, Agent-Based Modeling Tool in Python for Economists

Sergio Rivero
Unidade Acadêmica de Economia e Finanças
Universidade Federal de Campina Grande (UFCG)

Abstract

This paper introduces $EcoSim_{py}$, an open-source, collaborative agent-based modeling (ABM) framework developed in Python specifically for economists. The motivation for $EcoSim_{py}$ stems from the increasing complexity of economic systems and the limitations of traditional aggregate modeling approaches, which often sacrifice realism for analytical tractability. Despite the potential of ABM to address such complexity by modeling heterogeneous agents and emergent phenomena, its broader adoption in economics has been hampered by challenges including the curse of dimensionality, forecasting difficulties, reliance on micro-data, and the absence of robust, community-driven frameworks. $EcoSim_{py}$ aims to overcome these barriers by offering a modular, extensible, and user-friendly platform. Model structure and scenario initialization are defined via *JSON* files, while simulations are executed and analyzed in Python, lowering the barrier for economists and facilitating collaboration. The framework's architecture separates model logic from implementation, encourages code reuse, and supports flexible scenario analysis. To illustrate its capabilities, the paper presents an implementation of the iterated prisoner's dilemma game, highlighting $EcoSim_{py}$'s utility for modeling, experimentation, and data analysis. The paper concludes by outlining future development goals for $EcoSim_{py}$, emphasizing its potential to foster community of economic modelers and to advance the use of ABM for tackling complex economic issues such as climate change, financial stability, and structural transitions.

Keywords: Agent-based modeling; Economic simulation; Python; Collaborative modeling;

Área Temática: Área 3. Método, Teorias Econômicas Heterodoxas e História Econômica

1 Introduction

Computers and computer simulation have become an ubiquitous part of economic modeling. Techniques such as Monte Carlo simulations, numerical optimization, computational general equilibrium, and dynamic stochastic general equilibrium models have all been widely used in economics. In recent years, Agent-based modeling (ABM) has gained some popularity as a modeling approach in economic research (Cincotti et al., 2022; Giulioni, 2019; Dosi and Roventini, 2019; Gubareva and Gomes, 2019; Dilaver et al., 2018; Dosi and Roventini, 2017; Caiani et al., 2016; Dweck et al., 2020; Rolim, 2022). The ability to represent heterogeneous agents and to explore emergent phenomena made ABM a useful approach for investigating the complexity of economic systems.

Unfortunately, broader use of ABM in economic research has been hindered by several factors (Axtell and Farmer, 2025). One significant challenge is the so-called curse of dimensionality: ABMs often involve a large number of parameters, making the exploration of the parameter space exponentially more complex and costly. Although specialized tools are available, handling numerous parameters remains demanding. Second, forecasting presents another obstacle. Unlike conventional macroeconomic models, which rely on aggregate data for initialization, ABMs require comprehensive microdata about individual agents to generate accurate forecasts. Such detailed data is rarely available, and devising plausible individual states that align with aggregate measurements and the model´s dynamics remains an unresolved issue. If these individual states are incompatible, the model can display transient behaviors and produce unreliable predictions. Third, mainstream economics has traditionally favored analytical tractability, often simplifying assumptions to enable mathematical solutions even at the expense of realism. In contrast, ABM prioritizes behavioral realism and is open to relaxing these strong assumptions, resulting in a fundamentally different approach to constructing economic models. Finally, the field still lacks well-established, community-developed ABM frameworks, making collaboration and standardization more difficult.

Most of these challenges are not exclusive to ABM; models such as DSGE and CGE also face similar difficulties. Some of these challenges, such as the large number of parameters and the forecasting problem, have been addressed by the significant increase in computing power and the growing availability of microdata and large datasets. These aspects, combined, have enabled exploration of larger parameter spaces and the initialization of ABMs with detailed microdata.

The third challenge is more related to the established traditions of economics and the traditional way of doing economic modeling. This aggregate and synthetic approach has been dominant across both mainstream and heterodox approaches to economic modeling (Rivero and Almeida, 2023). Economists often prefer models that can be solved analytically, even if less realistic, rather than using more plausible assumptions and more complex models that can only be solved numerically. This is a cultural issue that is not easy to change, but the increasing availability of computational tools and the growing interest in complexity economics could help to shift this paradigm.

Finally, the lack of well-established ABM frameworks is a significant barrier to the widespread adoption of ABM in economics. The development of a collaborative, open-source ABM tool tailored for economists could facilitate the sharing of models, methodologies, and best practices, allowing the economic modeling community to construct models that can address issues that are difficult and inherently complex and extrapolate the narrow space of economic models, such as climate change, financial stability, and the transition to low-carbon growth, among others.

In this paper, we address the last of these challenges. The need to build a collaborative, open-source ABM tool that can evolve and serve as a shared platform for developing economic models. The idea behind this tool is not just to present a complete and ready-to-use ABM platform, but to offer a set of tools and a framework that economists can extend to develop ideas and basic models, making community development and model sharing easier and more accessible. The tool, called $EcoSim_{py}$[1], is an open-source ABM platform developed in Python and designed to be user-friendly

[1]The framework can be downloaded as a library directly from PyPi using the pip command (`pip install EcoSimpy`)

and flexible for economists. The main goal of $EcoSim_{py}$ is to provide a collaborative environment where economists can share their models, methodologies, and best practices, fostering a community of users that can contribute to the development of the platform and the advancement of ABM in economics.

2 ABM Platforms for Economists

There are currently several ABM generic platforms available, and some of them have been used by economists. However, most of these platforms lack the specific features and functionalities needed for economic modeling and often require additional effort to develop the software components used in economic models. Almost all of these platforms are open and extensible, but in practice, wider adoption is hindered by multiple factors. In this section, we provide a brief overview of these platforms and discuss the main reasons they are not widely used by economists.

One of the most relevant platforms used in macroeconomic modeling outside the mainstream is LSD[2]. LSD is a powerful and flexible ABM platform that has been used to develop a wide variety of economic models, including financial markets, macroeconomic, and innovation and technological change models. The package has been extensively used to develop theoretical and analytical models for problems in macroeconomics (Dosi et al., 2008; Lamperti, Monasterolo, and Roventini, Lamperti et al.; Lamperti et al., 2020; Dweck et al., 2020; Rolim, 2022; Rolim et al., 2023; Reissl, Fierro, Lamperti, and Roventini, Reissl et al.). The framework uses an equation-based approach and generates C++ code for the models. The code is efficient and fast, but generating it in *C++* reduces the platform's flexibility, limiting the ability to explore more detailed structural models.

Another specialized ABM platform is *JMAB (Java Macroeconomic Agent-Based Modeling)*[3]. *JMAB* is a Java-based ABM platform that follows more modern software development practices (mainly Dependency Injection and Inversion of Control) and is designed to be flexible and extensible. The framework has been used to develop a highly influential paper in macroeconomics Caiani et al. (2016), but it has not been widely adopted by economists. The main reason for this is that the use of Java as the programming language of the framework makes it less accessible for economists, who are more familiar with other programming languages such as Python or R. Additionally, the framework is designed to be flexible and extensible, but it can be difficult for users to understand and work with the underlying code, which can be a barrier to entry for economists who are not familiar with programming. One interesting aspect of the work with *JMAB* is that the authors have developed a *Benchmark Model*, which has served as a reference for the development of other models. The idea of having a benchmark model that can serve as a reference for the development of other models is a good practice and, if implemented in a computational framework, can facilitate the sharing of models and the development of new models based on the benchmark model. The collective development of families of basic models in code that can be used and specialized by

[2]The code and models can be found here www.labsimdev.org

[3]The code and models can be found here https://github.com/S120/jmab

scientist is already a common practice in other scientific disciplines and could be very helpful to advance the use of ABM in economics and to improve the treatment of problems that are difficult and inherently complex and extrapolate the narrow space of economic models, as such climate change, financial stability, and the transition to low-carbon growth, among others.

In the next section we present $EcoSim_{py}$, in more detail, and we discuss the main concepts and implementation methodology of the framework.

3 Main Concepts and Implementation Methodology

Economic Simulation in Python ($EcoSim_{py}$) is an agent-based modeling and simulation framework made in Python 3+. The models are written in Python, but the initialization and interface definitions are in *JSON*. The main idea is to provide a framework that allows for easy definition of the model and the interface in a *JSON* file, and then run the simulation in Python. The framework's structure is shown in Figure 1.

3.1 Construction of the simulation framework

The framework is designed to be flexible and modular, allowing users to easily create and modify models without changing the underlying code. Using JSON files for model definition and initialization provides a clear separation between the model structure and implementation, making it easier for users to understand and work with the framework. The concept behind the simulation framework is to provide a set of basic classes and functions for creating a wide variety of models. The framework is designed to be extensible, allowing users to add new features and functionalities as needed. The choice of Python as the framework's programming language has two main motivations. First is that Python is a simple well know general programming language that is used by a wide range of users specially in data science, so the learnign curves would be more smooth; second, the language allows for a wide range of libraries and tools to be used in conjunction with the framework, allowing not only the use of graphs and some statistical analysis tools but also the use of machine learning algorithms and data processing libraries.

Another reason for choosing Python as the implementation language is that it will make it easier to extend the framework's basic classes. This will allow users to create their own classes and functions that can be used in models without modifying the framework's underlying code. This way, we can facilitate the use of the framework and make it more accessible for economists, and we hope they will encourage the development of a community of users that can share their models and their extensions of the basic classes, overcoming one of the main challenges of ABM modeling in economics(Axtell and Farmer, 2025).

The main components of the framework are:

- **Core** - Is the main part of the simulator. In the core, we have full control of the simulator. A simulation is created by executing main.py, with the model definition files (*JSON*) passed as parameters.

- **Apps** - The apps are the implementations of the models. Each model has its own folder and subfolders. The simulation initialization is defined in a *JSON* file, and the agents, spaces, and actions in the model are defined in specific Python modules.

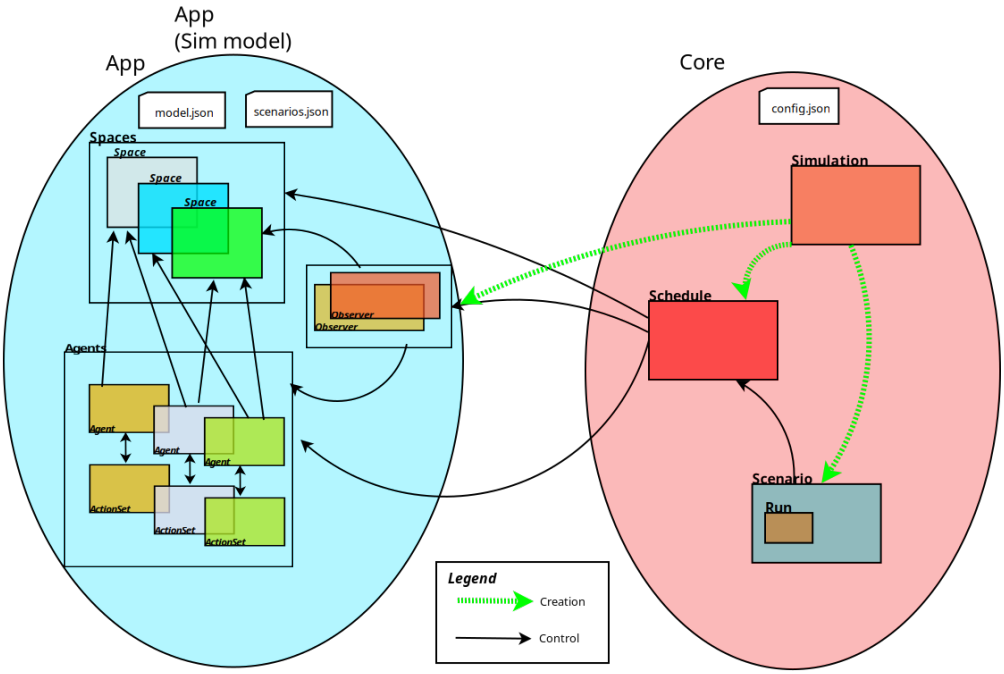


Figure 1: Structure of $EcoSim_{py}$

The implementation is based in two design principles:

- **MVC** (*Model-View-Controller*) - We separate the model and view components in the implementation. The modules model, *BasicAgents*, *BasicSpaces*, etc., are the model part of the framework. The modules *Simulation* and *BasicSchedule* are responsible for the control component, and the module *BasicObserver* and a *Jupyter Notebook* file are responsible for the view component.

- **Dependency Injection** - Is used to create the objects in the model. The Agents, the Spaces, the Observers,, and the Schedule are all created from a JSON file. This file also contains all the necessary information for the initialization of the model, like the number of agents of each type, the spaces, the initialization of the variables of the agents, the variables that will be observed, etc. The idea is to make the construction of scenarios and models more flexible using the same set of classes. These classes can be initialized using the information in the JSON file, and we can define each scenario's initialization in this file.

The framework is designed to allow the execution of a simulation using two JSON files that define the model and initialize the scenarios. The first JSON file contains the model definition, including the agents, spaces, the number of agents,, and other model details. The second JSON file (scenarios.json) contains the definitions of the scenarios and the initial values of the agents' and spaces' variables. This way, we can easily define different scenarios using the same model

definition. The use case of the framework is shown in the Figure2. 1. The user can define the model and scenarios using JSON files, then run the simulation in a Jupyter notebook. The simulation results can be processed and displayed using the appropriate Python commands. The framework is designed so that the user can run the simulation and analyze results from a single file, without modifying the model or the framework. This way, we can facilitate the use of the framework and make it more accessible for economists who are less familiar with programming.

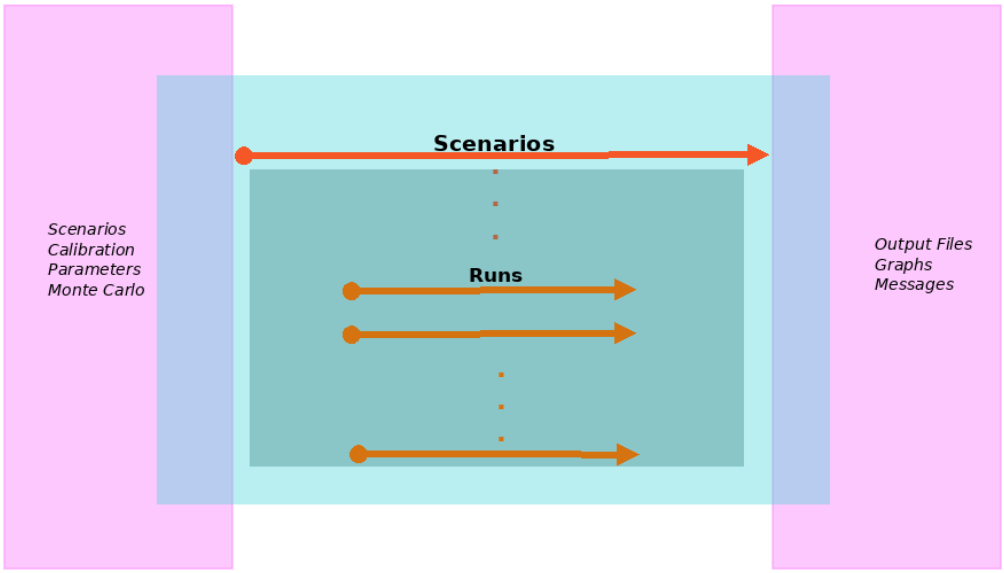


Figure 2: Use Case of $EcoSim_{py}$

4 Economic Examples

The example we present in this paper is a simple model for studying the iterated prisoner's dilemma game. It has been used to study the results of the work by Press and Dyson (2012), but with the game generalized to multiple agents, multiple interactions, and multiple strategies. The model is based on the work of Axelrod (1984) and Axelrod (1997), but with modifications to generalize the game to multiple agents and interactions. The model is implemented using the $EcoSim_{py}$ framework, and the results are analyzed using Python libraries for data analysis and visualization[4].

The model *JSON* file contains the definitions of the agents, spaces,, and actions. The agents are defined as a class that inherits from the framework's basic agent class, and the spaces are defined as a class that inherits from the framework's basic space class. The agents' actions are defined as methods of the agent class and executed in the order specified in the model's schedule. The model also includes an observer that collects data on the variables of interest during the simulation. The observer is a class that inherits from the framework's basic observer class and is executed at the end of each simulation step.

Listing 1: Model Definition JSON

```
1 {
```

[4]The model code can be downloaded from https://github.com/gECo-lab/ipd

```
  "simulation_name": "ipd",
  "simulation_parameters": [
    {
      "parameter_name": "total_payoff",
      "parameter_value": 0.0
    }
  ],
  "model_name": "ipd",
  "schedule": [
    {
      "schedule_type": "MixedSchedule",
      "schedule_name": "MyMxSchd"
    }
  ],
  "spaces": [
    {
      "space_type": "IpdGame",
      "space_name": "IpdGame",
      "space_variables": {
        "payoffs": 0.0
      }
    }
  ],
  "agents_classes": [
    "GoodPlayer",
    "HardTitForTatPlayer",
    "ZDExtortionPlayer"
  ],
  "agents": [
   {
      "agent_type": "GoodPlayer",
      "agent_prefix": "Allc",
      "agent_spaces": [
        "IpdGame"
      ],
      "no_of_agents": 100,
      "has_observer": true
    },
    {
      "agent_type": "ZDExtortionPlayer",
```

```
        "agent_prefix": "ZDExt",
        "agent_spaces": [
          "IpdGame"
        ],
        "no_of_agents": 100,
        "has_observer": true
      },
     {
        "agent_type": "BadPlayer",
        "agent_prefix": "Alld",
        "agent_spaces": [
          "IpdGame"
        ],
        "no_of_agents": 100,
        "has_observer": true
      }
   ]
}
```

In the example, we have four agent types: *GoodPlayer*, *BadPlayer*, *HardTitForTatPlayer* and *ZDExtortionPlayer*. The *GoodPlayer* always cooperates, the *BadPlayer* always defects, the *HardTitForTatPlayer* execute the hard tit-for-tat strategy and the *ZDExtortionPlayer* is a player that uses a zero-determinant strategy (Press and Dyson, 2012) that allows the player to enforce a linear relationship between their own payoff and the opponent's payoff. The model is initialized with 100 agents of each type, which interact randomly with one another in the *IpdGame* space. The game's payoffs are defined in the space and updated after each interaction. The simulation results show the evolution of each agent type's total payoff over time and the distribution of strategies among agents. This *JSON* file allows defining multiple agent types and easily changing the types and number of agents in the model, separating the development of the agent types from their use in the simulation. It's used in the simulation to make it easier to define different scenarios and simulations using the agent types' code defined in the agents module. One can define agent types in the agents module and use them across different simulations by changing the *JSON* file.

In the scenarios *JSON* file, we can define different scenarios for the model by changing the agents' initial conditions. Each scenario can be executed a determined number of times, and the results are saved in *CSV* files that can be analyzed using Python libraries for data analysis and visualization. Using JSON files for model definition and initialization provides a clear separation between the model structure and implementation, making it easier for users to understand and work with the framework.

The agents and strategies are implemented as classes that inherit from the *DiscreteEventAgent* class of the framework. The actions of the agents are defined as methods in the action set module and they are executed in a specific order defined by the step method of the agent class. The

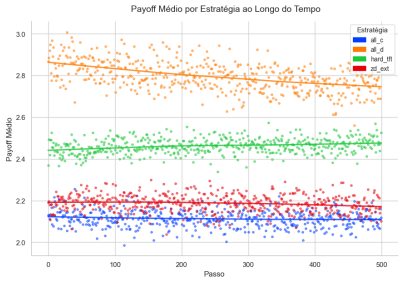
interactions between the agents are defined in the space class.

Listing 2: Model Execution Jupyter Notebook

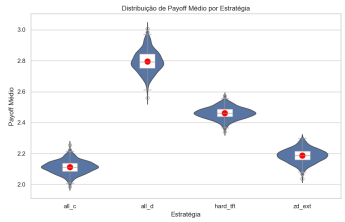
```
new_sim = EcoSimpy.Simulation(app_dir=app_dir,
                              config_file=config_json_file,
                              model_file=model_file,
                              scenarios_file=scenarios_file,
                              clean_run=True)

new_sim.execute_simulation()
```

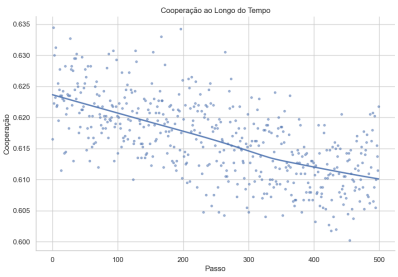
The simulation is executed from a *Jupyter notebook* containing the code to run the simulation and analyze the results. The simulation results are analyzed using Python libraries for data analysis and visualization, including *pandas* and *matplotlib*. The results are presented in the form of graphs and tables that show the evolution of the variables of interest over time and the distribution of strategies among agents. Figures 3a, 3b and 3c show the simulation results. The first figure shows the mean payoff of each agent type over time; the second, the distribution of each agent type's payoffs at the end of the simulation; and the third, the evolution of the cooperation rate among agents over time.



(a) Mean Payoff by Strategy



(b) Payoff Distribution by Strategy



(c) Cooperation Over Time

The simulation and analysis can be performed without modifying the model code, allowing the construction of different scenarios and the analysis of results. The separation between the agents is coded in Python, and the model definition and initialization are in *JSON* files, allowing a clear separation between the model structure and the implementation, making it easier for users to understand and work with the framework. This way, we can facilitate the use of the framework and make it more accessible to economists who are less familiar with programming, allowing them to focus on economic modeling and analysis rather than the technical details of implementation.

For the framework to be effective, a set of basic classes and functions is needed to create a wide variety of models. The framework's extensibility enables this functionality. But for this extensibility to be effective,, there is a need for a community of users to share their models and extensions of the basic classes, overcoming one of the main challenges of ABM modeling in economics (Axtell and Farmer, 2025). The development of a collaborative, open-source ABM tool tailored for economists could facilitate the sharing of models, methodologies, and best practices, allowing the economic modeling community to construct models that address difficult, inherently complex issues and expand the narrow space of economic models.

5 Final Remarks

Dealing with complexity is one of the main challenges of economic modeling. The traditional approach to economic modeling has been to simplify models as much as possible to make them analytically tractable, but this approach has its limitations. Agent-based modeling (ABM) is a powerful tool for addressing complexity in economic systems, but it also poses its own challenges. One of the main challenges of ABM is the lack of well-established, community-developed frameworks, which makes collaboration and standardization more difficult. In this paper, we have presented $EcoSim_{[py}$, an open-source ABM platform developed in Python that is user-friendly and flexible for economists.

The framework is designed to be flexible and modular, allowing users to easily create and modify models without changing the underlying code. Using JSON files for model definition and initialization provides a clear separation between the model structure and implementation, making it easier for users to understand and work with the framework. The example presented in this paper is a simple model for studying the iterated prisoner's dilemma game, but the framework can be used to build a wide variety of economic models. The next steps for the development of $EcoSim_{[py}$ include adding more features and functionalities, such as parallel simulation, machine learning for data analysis, and a web-based interface for the simulation. The development of a collaborative, open-source ABM tool tailored for economists could facilitate the sharing of models, methodologies, and best practices, allowing the economic modeling community to construct models that can address issues that are difficult and inherently complex and extrapolate the narrow space of economic models, such as climate change, financial stability, and the transition to low-carbon growth, among others.

The framework presented here was constructed with the collaborative construction of macroeconomic models in mind. The main goal of the framework is to provide a set of basic classes and functions that can be used to create a wide variety of models, and to encourage the development of a community of users that can share their models and their extensions of the basic classes, overcoming one of the main challenges of ABM modeling in economics (Axtell and Farmer, 2025). The choice of Python as the framework's programming language has two main motivations. First is that Python is a simple well know general programming language that is used by a wide range of users specially in data science, so the learning curves would be more smooth; second, the language allows for a wide range of libraries and tools to be used in conjunction with the framework, allowing not only the use of graphs and some statistical analysis tools but also the use of machine learning algorithms and data processing libraries. The use of Python as the programming language of the framework makes it more accessible for economists, who are more familiar with Python, and it allows for a wider range of libraries and tools to be used in conjunction with the framework, allowing not only the use of graphs and some statistical analysis tools but also the use of machine learning algorithms and data processing libraries. Additionally, the framework is designed to be flexible and extensible, while also easy to understand and work with, which can help reduce the barrier to entry for economists who are not familiar with programming. Using *JSON* files for model definition and initialization provides a clear separation between the model structure and implementation, making

it easier for users to understand and work with the framework.

References

Axelrod, R. (1984). *The evolution of cooperation*. New York: Basic Books.

Axelrod, R. M. (1997). *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton: Princeton University Press.

Axtell, R. L. and J. D. Farmer (2025). Agent-based modeling in economics and finance: Past, present, and future. *Journal of Economic Literature 63*(1), 197–287.

Caiani, A., A. Godin, E. Caverzasi, M. Gallegati, S. Kinsella, and J. E. Stiglitz (2016, aug). Agent based-stock flow consistent macroeconomics: Towards a benchmark model. *Journal of Economic Dynamics and Control 69*, 375–408.

Cincotti, S., M. Raberto, and A. Teglio (2022). Why do we need agent-based macroeconomics? *Review of Evolutionary Political Economy 3*(1), 5–29.

Dilaver, O., R. C. Jump, and P. Levine (2018). Agent-based macroeconomics and dynamic stochastic general equilibrium models: where do we go from here? *Journal of Economic Surveys 32*(4), 1134–1159.

Dosi, G., G. Fagiolo, and A. Roventini (2008, sep). Schumpeter meeting keynes: A policy-friendly model of endogenous growth and business cycles. *34*(9), —-.

Dosi, G. and A. Roventini (2017). Agent-based macroeconomics and classical political economy: Some italian roots. *Italian Economic Journal 3*, 261–283.

Dosi, G. and A. Roventini (2019). More is different ... and complex! the case for agent-based macroeconomics. *Journal of Evolutionary Economics 29*(1), 1–37.

Dweck, E., M. T. Vianna, and A. da Cruz Barbosa (2020). Discussing the role of fiscal policy in a demand-led agent-based growth model. *EconomiA 21*(2), 185–208.

Giulioni, G. (2019, jan). Gabriele: The general agent based repast implemented extensible laboratory for economics. *SoftwareX 9*, 255–259.

Gubareva, M. and O. Gomes (2019). *On the Edge of Climate Change: In a Search of an Adequate Agent-Based Methodology to Model Environmental Dynamics*, pp. 37–57. Springer International Publishing. MAG ID: 2944105537.

Lamperti, F., G. Dosi, M. Napoletano, A. Roventini, and A. Sapio (2020, apr). Climate change and green transitions in an agent-based integrated assessment model. *Technological Forecasting and Social Change 153*, 119806.

Lamperti, F., I. Monasterolo, and A. Roventini. Climate risks, economics and finance: Insights from complex systems. pp. 97–119. MAG ID: 2905386362.

Press, W. H. and F. J. Dyson (2012). Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent. *109*(26), 10409–10413.

Reissl, S., L. E. Fierro, F. Lamperti, and A. Roventini. The dsk stock-flow consistent agent-based integrated assessment model. *236*, 108641.

Rivero, S. and O. Almeida (2023). Da racionalidade axiomática à racionalidade ecológica: elementos para a construção de uma agenda de pesquisa em simulação computacional econômica. *Nova Economia 33*(2).

Rolim, L. N. (2022). *Income inequality, economic activity, and economic policy in an agent-based macroeconomic model*. phdthesis.

Rolim, L. N., C. T. Baltar, and G. T. Lima (2023). Income distribution, productivity growth, and workers' bargaining power in an agent-based macroeconomic model. *Journal of Evolutionary Economics 33*(2), 473–516.