Multi-Agent Financial Systems with RL: A Pension Ecosystem Case (WIP)

Fatih Ozhamaratli$^{1[0000-0002-7870-6316]}$ and Paolo Barucca$^{1[0000-0003-4588-667X]}$

University College London, 66-72 Gower St, WC1E 6EA, London, UK
{ucabfoz,p.barucca}@ucl.ac.uk

Abstract. This paper introduces a multi-agent reinforcement learning (MARL) model for the pension ecosystem, aiming to optimize contributor saving and investment strategies. The multi-agent approach enables the examination of endogenous and exogenous shocks, business cycle impacts, and policy decisions on contributor behavior. The model generates synthetic and diverse income trajectories to develop inclusive savings strategies for a broader population. Additionally, this research innovates by developing a multi-agent model capable of adapting to various environmental changes, contrasting with traditional econometric models that assume stationary employment and market dynamics. The multi-stationary nature of the model allows for a more realistic representation of economic systems, enabling a better understanding of the complex interplay between agents and their responses to evolving economic conditions.

1 Introduction

The problem of pension savings has been extensively researched in economics, with Merton [16] first formulating the problem using an econometric approach that assumed non-insurable labour income, constant income risk and investment returns, and no liquidity requirements. However, subsequent research has revealed the need for a more comprehensive understanding of pension dynamics [5], including the introduction of liquidity constraints [3]. Factors such as labour income fluctuations and asset return fluctuations [4]. These factors are influenced by interactions between businesses and individuals as well as central bank decisions, which are crucial to consider in pension investment strategies. Despite the importance of these factors, econometric models often assume them to be constant or to follow predetermined linear models. An agent-based model (ABM) of the pension ecosystem, using multi-agent reinforcement learning (MARL) to optimize investment strategies, can address the limitations of traditional econometric models by accounting for the endogenous dynamics of the pension environment.

The proposed model generates synthetic and heterogeneous income trajectories that can be used to devise inclusive savings strategies for a broader population. Furthermore, this research presents a novel contribution by developing a multi-agent model that is robust to changes in environmental dynamics, which distinguishes it from existing econometric models that rely on stationary assumptions about employment and market dynamics. By emphasizing the multi-stationary nature of the model, researchers can not only analyze first-order effects but also capture emergent properties arising from the interactions within the system, as described by [1] . This approach allows for better responsiveness to paradigm shifts and black swan events and accounts for the consequences of heterogeneous profiles among pension investors.

Pension funds are meant to invest with long-term strategic vision, to avoid the effect of financial crises and vulnerability to low probability high impact black swan events, as observed in 2008 financial crisis [15] or the 2022 pensions leveraged gilt crisis [7] that effected pension funds that are in general investing with short-term vision. Yet there are examples of successful investment strategies such as Norwegian Sovereign Wealth fund that invests counter-cyclically with business cycle [20].

Research on U.S. Social Security data indicates that the labour income has characterising moments that are counter-cyclically exposed to business cycle effects [12]. Cascaded effects of investment decisions [6] and supply chain shock propagation [21] result in a non-stationary market dynamics, which violates the premises of general pension models assuming stationary income risk and asset return dynamics.

The proposed ABM of the pension ecosystem addresses these limitations by using a MARL approach to optimize investment strategies. However, implementing such a model is not without its challenges. Deep

Reinforcement Learning has been successful in accomplishing complex tasks [17], but multi-agent deep reinforcement learning is still a computationally expensive solution [24]. The challenges of MARL include non-stationarity of the environment, combinatorial complexity, difficulties arising when number of agents are greater than 2, and multidimensional learning objectives as stated in a comprehensive review of MARL [27]. Despite these challenges, recent research has shown that carefully trained Proximal Policy Optimization can perform successfully for optimizing MARL problems in cooperative environments [28].

Applying reinforcement learning for game theory with applications in finance was recently explored in AI Economist [29]

Recent advances in software architectures that bridge the gap between mathematical formulations with GPU accelerated Just-in-Time (JIT) executed codes by [9], enable scientists to express the fundamental mathematical operations governing interactions between agents and simulation dynamics as APIs similar to NumPy [13] that are easy to use for mathematical expressions, without need to factor the code for batches, and distributed execution.

Significant part of the reinforcement learning literature evolved around computer games [18], or virtual simulations [11]. Although games provide a flexible environment that can be used to collect large samples, analysis of such games are relatively simplistic in comparison to the modelling needed for interpreting financial ecosystem. To analyse financial phenomena, a system needs not only to provide a general test-bench for MARL, but also to be able to integrate financial and economic indicators, and to benchmark against real-world observations, e.g. for analysing cooperation, competition and behavioural heterogeneity in an environment. Financial systems, given the constraints of their marketplaces, are generally well suited to be captured by well-defined interactions between agents that can be expressed as mathematical equations. Such a system is well suited to leverage accelerators such as GPUs and TPUs. In this way, a high number of samples can be generated in an interactive environment that is energetically and computationally efficient in comparison to classical CPU multi-processing loads for games.

2 Design Choices of Financial Model

Algorithm 1: Simulation Loop Overview

```
Initialize: $BusinessEntity$, $IndividualEntity$, $CBEntity$, $MarketView$, and $RewardCalculator$;
 $N \leftarrow 0$;
while $N <$ maxSimulationStepCount do
    with $MarketView$ and $RewardCalculator$ capturing data from entities
        CB Entity:                                      Interest Rate Decision
        Business Entities:                              Borrow Choice
        Market Dynamics:                                Execution
        Business Entities:                              B2B Trade
        Business and Individual Entities:               Employment
        Business Entities:                              Production
        Business and Individual Entities:               B2C Trade
        Individual Entities:                            Consumption
        Individual Entities:                            Investment Choice
        Rewards:                                        Feedback
        Training:                                       Step
    end
    $N \leftarrow N + 1$;
end
```

2.1 Actors of Ecosystem and interactions

The multi-agent reinforcement learning model presented in this research simulates a pension ecosystem consisting of three main actors: Individuals, Businesses, and the Central Bank. The Central Bank, as the regulator of the economy, sets the interest rate for businesses to obtain loans. Businesses, in turn, trade with each other and produce goods using a Sectoral IO matrix and a chosen production function. Businesses also engage in trade with individuals for their labour.

Individuals, as the primary contributors to the pension system, make investment decisions that affect their pension savings. They also consume goods produced by the businesses. The model simulates the impact of endogenous and exogenous shocks, business cycles, and policy decisions on the behaviour of these individuals. Additionally, the model generates synthetic and highly diverse income trajectories to provide a more realistic representation of the population, which can be used to develop more inclusive savings strategies.

In terms of investment decisions, the model assumes that asset returns are correlated with the estimated fundamental values of companies, which are estimated based on the market trading value of their goods. The model also uses meta-strategies for contributor agents that are robust to changes in the environment dynamics, as opposed to traditional econometric models that assume stationary employment and market dynamics.

The proposed model in this research utilises a simulation loop to simulate the interactions between the different actors in a pension ecosystem. The actors in this ecosystem include Individuals, Businesses, and the Central Bank (CB). The simulation begins by initialising various entities that represent these actors, including the Business Entity, Individual Entities, CB Entities, Market View, and Reward Calculator.

The simulation loop 1 begins with N set to 0 and continues until N is less than the maximum number of simulation steps. Inside the loop, several operations are performed in a specific order. The Market View and Reward Calculator capture data from the entities, allowing them to calculate respectively the price statistics for agents to make informed decisions about the market and rewards that are being used for reinforcement learning. The Central Bank then make an interest rate decision which affects the borrowing choices of the Business Entities. Market dynamics are then executed, simulating the interactions between the different actors in the ecosystem.

The Business Entities then engage in Business-to-Business (B2B) trade and both Businesses and Individuals engage in employment. Business Entities also produce goods and both Businesses and Individuals engage in Business-to-Consumer (B2C) trade. Individual Entities then consume the goods and make investment choices. The Reward Calculator provides feedback and the training step is executed. The variable N is then incremented by 1 and the loop continues until the maximum number of simulation steps is reached.

The algorithm reflects the order of operations in the simulation, which is designed to simulate the interactions between the different actors in the pension ecosystem and how they affect the investment and saving behaviours of individuals. It is a key aspect of the proposed model, as it allows for the simulation of various scenarios and the impact of different factors on the ecosystem as a whole.

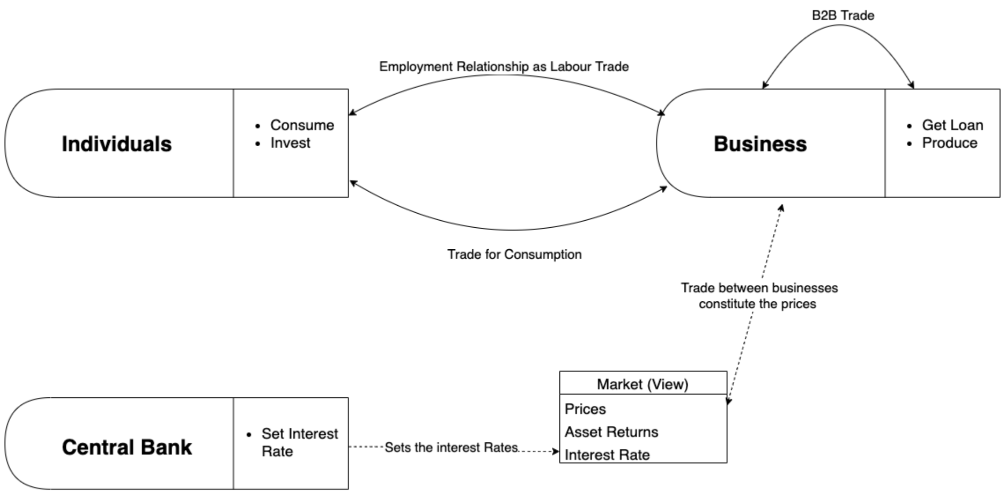


Fig. 1: Agents and Environment Diagram

The choices that agents in the pension ecosystem make are crucial to the functioning of the system and are as follows:

Entity	Key Decisions
Business	Borrowing choices
	Trading activities
	Production capacity utilization
Individual	Employment
	B2C trade activities
	Saving Decision
	Investment and portfolio allocation
Central Bank	Determining interest rate

Table 1: Key decisions for each entity in the model

Table 1 presents the key decision-making factors for various entities in the financial ecosystem. These factors play a crucial role in shaping the dynamics of the pension ecosystem and should be considered when modeling agent interactions in our multi-agent reinforcement learning framework.

It is important to note that the choices made by each agent have a direct impact on the overall functioning of the pension ecosystem. Businesses must balance production and debt management, while individuals must consider their consumption, savings, and investment decisions. The Central Bank's decision to set interest rates also plays a crucial role in the functioning of the ecosystem. Prices and market dynamics manifest from the interactions between agents.

2.2 Mechanism of interactions

There are three types of operations, firstly an interaction of a trade nature where two parties actively participate by making an offer and the other party taking decisions based on the offer. Secondly, there are choice operations where an agent makes a choice by itself, on issues such as setting interest rates, deciding how much percentage to utilise for production capacity, or an individual making an investment decision. Thirdly the simulation dynamics such as charging the owed interest per month to businesses, where no active decision is made but operations relevant to simulation dynamics are executed.

Trade operations consist of two components: an Offer and a Decision. In this example, a deep neural network is utilised to take in an agent's own embedding, along with market data, and output a vector that specifies the goods being offered and the requested amount of cash. Although the trade module can also handle barter transactions, for simplicity, cash-only transactions are preferred as they allow for easy market view coupling. This means that the average prices of goods transactions can be used to update the market table, which acts as a signal for businesses to guide their decisions and shape rewards for RL. The Decision network uses the embedding of the decision-making party, the offer, and the market table as input to make a decision on whether to accept the offer. Following policy inference model of [29], agents share the trained policy inference parameter θ and each agent has their own state $h_{i,t}$, and state is updated with each policy inference:

$$a_{i,t}^{offer} \sim \pi(o_{i,t}^{agent}, o_{i,t}^{market}, h_{i,t}; \theta) \tag{1}$$

$$a_{i,t}^{decision} \sim \pi(o_{i,t}^{agent}, o_{i,t}^{offer}, o_{i,t}^{market}, h_{i,t}; \theta) \tag{2}$$

Choice operations get the relevant agent's embedding and market information to make a decision via deep neural network.

$$a_{i,t}^{choice} \sim \pi(o_{i,t}^{agent}, o_{i,t}^{market}, h_{i,t}; \theta) \tag{3}$$

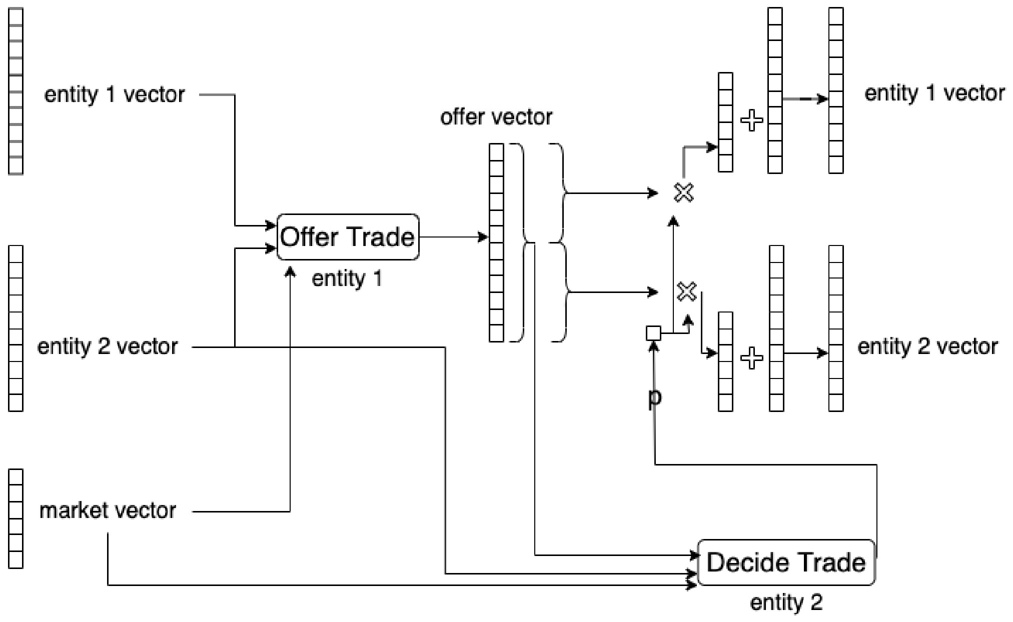


Fig. 2: The Trade Module includes Offer and Decision Models that handle the inference on trade operations. The Offer Model creates an Offer Vector, which is made up of two parts: first half represents the inventory that the offering party will provide, and the second half represents the inventory that the accepting party will provide. The Decision Model then produces a decision rate value, indicating whether or not to proceed with the trade, with 0 meaning no trade and 1 meaning full acceptance of the trade offer.

2.3 Alignment with Finance Community

To make the simulation results more accessible to the broader financial community, we have taken steps to align our model with familiar financial terminology and concepts found in lifetime consumption and portfolio selection literature.

Our choices include the use of utility functions, which are widely studied in finance and help to express trade-offs between immediate and future returns in terms of intertemporal preferences [16].

The model's calibration is grounded in observable phenomena through the use of the input-output technology matrix as the only hard-encoded market dynamic. This approach facilitates comparisons between various countries with differing input-output technology matrices.

Regarding asset endowment dynamics, two configurations are explored. The first is a cash and risky vs riskless asset dynamic [5], which can be calibrated to have the same mean and variance as other portfolio allocation models in the literature:

$$R_{t+1} - R_f = \mu_{t+1} + \sigma_{t+1} \tag{4}$$

where R_f is the risk-free return rate, determined by the interest rate set by Central Bank, and the risky asset return R_{t+1} is defined by R_f and excess returns characterised by μ_{t+1} and σ_{t+1}. Another configuration that is differentiating from the literature that can provide analysis of the higher-order effects, would be coupling asset returns to the revenues or valuation of the companies being simulated in the model, such an approach can provide new insights.

For investigating different countries, the simulations must be bootstrapped with populations reflecting the respective sectoral distributions of countries.

The parallelism between the discounted rewards used in reinforcement learning [25] and time-discounting in the multi-step portfolio optimisation makes this methodology suitable to communicate the financial interpretation of an agent-based model. Time discounting can be classically found in financial literature incorporated to utility functions such as CRRA preferences [16] or the Bellman optimisation equations [5].

Another potential improvement involves incorporating heterogeneous employee profiles by breaking down labor into various professional groups, each with their own pricing. This heterogeneity can be integrated into the input-output production matrix, as seen in [23]. Alternatively, differentiated labor values can reflect the talent distribution of the population, which may result in differing incomes for individuals with varying talent, education, or experience levels [10].

3 Architecture of Simulation

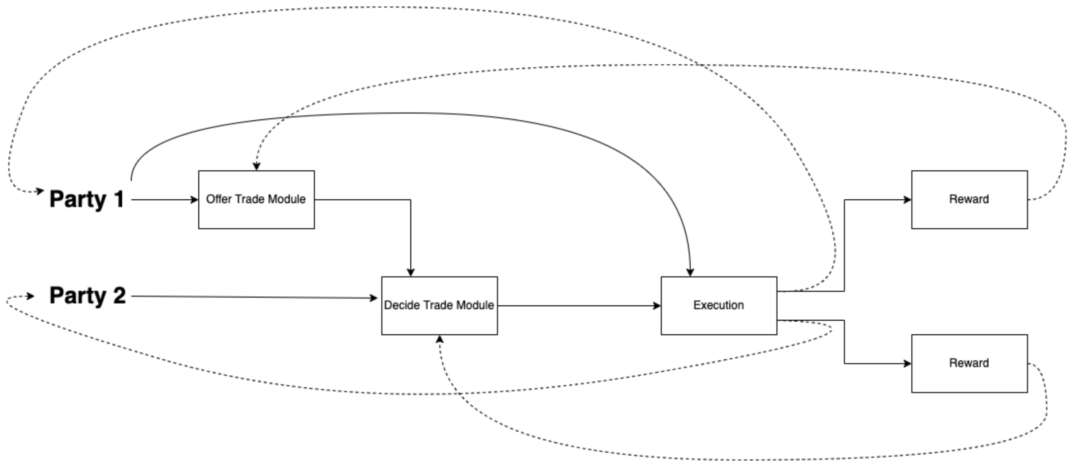
JAX [2] provides a unified framework where both the training and inference of deep neural networks as well as the execution of simulation operations can be done by transforming simple expressions to vectorised operations that can utilise hardware accelerators. Furthermore auto-differentiation property provides flexible functionality for training deep neural network models. The JAX ecosystem provides Flax [14] as a Deep Neural Networks library. Such a flexible framework ensures the flexibility to build framework that can work with variable inputs. The input flexibility is important to scale and sophisticate the models with minor interventions to the code and underlying structure, and to simplify the process of feeding the system with more granular and richer data sets.

Developing an efficient system that leverages low-level hardware functionalities and accurately models a financial system or any other complex system is a challenging task. Often, researchers or companies must prioritize one aspect due to increased complexity, resource, and time constraints. A framework and methodology that allows for rigorous modeling of a financial ecosystem without the need to worry about efficient code execution at every step of development enables resources to focus on the model, optimization algorithms, and sensitivity analysis of the ABM results.

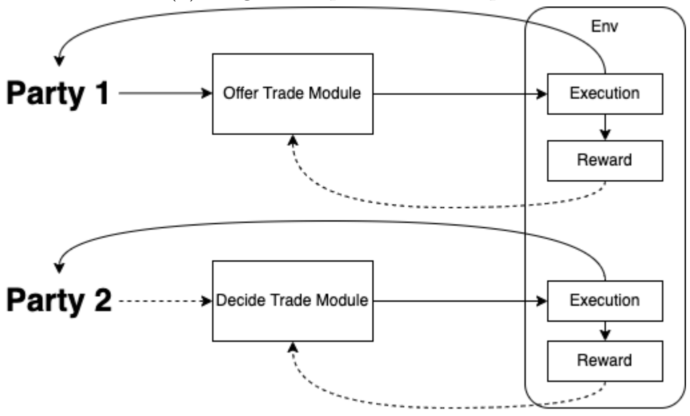
This research introduces a framework and methodology to test financial agent-based models and simulations that can be optimised with various optimisation techniques including deep RL in a structured way, where the dynamics of the system are coded for one agent as simple python functions, and the execution at the backend will be JIT and hardware accelerated by GPUs, TPUs and CPUs. Another advantage is decoupling the development of code responsible for parallelism, from the code responsible for the simulation dynamics; this helps the modellers to deal with the complexity of modelling financial systems, parallelising it for efficient execution via hardware accelerators, and developing complex optimisation algorithms.

One way to simultaneously take advantage of accelerated JIT execution of RL and financial models in our ABM is expressing them on a single computational graph, where simulation operations, reward calculation and optimisation algorithms run on a single network. The advantages of a single computational graph come with a great cost of high development complexity, error-proneness and not being flexible for adapting to various reward mixing methodologies. In contrast, a modular and decoupled methodology for the execution of various operations provides advantages, such as flexibility by reward mixing and ease of development of simulation and training environment. This can be accomplished without jeopardising the computational efficiency of accelerated JIT under a single framework.

- SimDirector: SimDirector module is responsible for calling the operations regarding the simulation at the highest level, rewards computation and training. It is the module where the system is orchestrated and where, with slight modifications, different simulation and training flows can be achieved, such as accessing the intermediate states of the simulation for reward mixing purposes, or changing order of the simulation operations.
- Reward Calculator: RewardCalculator is a buffer module where the implementation of desired complex reward mixing, storage of state transitions and rewards from various episodes, as well as parallel running simulation instances can be achieved.
- Entity: The Entity module is the wrapper around the stack module and agent modules. Each entity that has multiple underlying units, must have a Stack module attached. Entities can have multiple agents, regarding different areas of interaction.
- Stack: The Stack module stores the matrix for data of multiple units, such as a collection of businesses, but also provide an API to access units as single objects. It is the module that provides duality of hardware accelerated matrix backend data type and a familiar Object Oriented Programming like API.
- Agent: Agent is the module that encapsulates the underlying Model module responsible for ML inference and training at low-level, and Operation Executor module that is responsible for executing the environment actions/dynamics for updating environment state, once ML module made a decision. Agent module is governing the loop for dividing matching and shuffling of units in operations which require interaction of same kind of entities, such as businesses trading with each other.



(a) Single Computational Graph



(b) Modular Decoupled Computations

Fig. 3: Contrasting single computational graph vs modular computations

– Model: it is the module where underlying neural networks are implemented and called for forward inference and backwards backpropagation operations on neural networks, as well as batch normalisation.
– Operation Executor: the Operation Executor module is responsible for the execution of JIT functions defined in Operation module for batches of randomly chosen units or indexed batches, by applying the operations on Stack module.
– Operation: the Operation module is where simulation dynamics are implemented as stateless python functions conforming to the stateless function format of JAX that is required for vectorisation and JIT. The functions are implemented without explicitly implementing by factoring in the batch dimensions or parallel execution.
– View: the View module is where the calculated statistics information from the underlying entities such as latest prices of assets being traded in the simulation or by agents designated information such as interest rate can be stored.
– Logger: the Logger module is responsible for calculating relevant statistics that can be outputted as plots, or be used for constituting rewards.

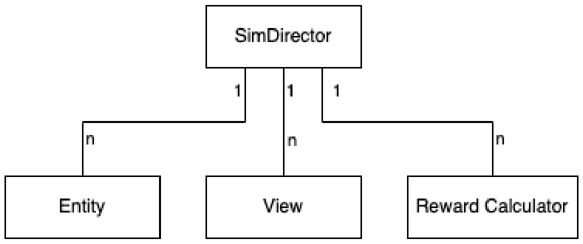


Fig. 4: SimDirector orchestrates the simulation and underlying modules

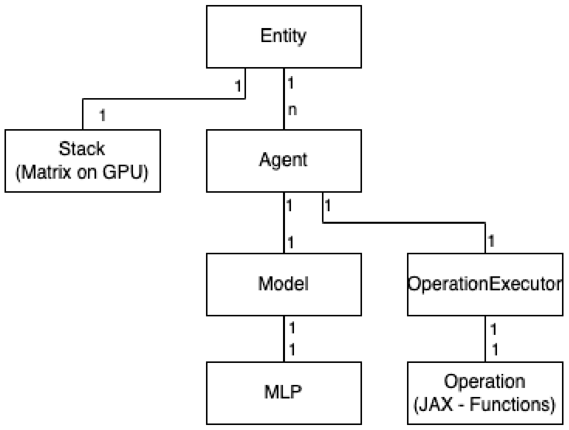


Fig. 5: Entity and underlying modules

4 Challenges

4.1 Addressing Stationarity and Training Challenges

Multi-agent systems face stationarity challenges compared to single-agent reinforcement learning. The dynamics agents learn actively change during training [27]. Various methodologies address this issue, with PPO, which shows strong empirical performance, being chosen for our multi-agent pension model [28].

PPO was preferred over MADDPG due to its on-policy nature, simplifying learning by updating policies based on recent experiences, enhancing sample efficiency. Real-world economic agents typically lack global information access, relying on local observations. Accurate financial models should consider this limited information availability.

MADDPG, while effective in some settings, might not suit financial systems modeling due to its assumption of full state and action space access during training. This could lead to over-optimization and unrealistic models. PPO, however, optimizes local policies based on an agent's experiences, offering a more realistic representation of market dynamics.

Considering real-world limitations and the importance of local information, PPO is a more suitable choice for financial system modeling. Agents can be trained as meta-learners, acting optimally in different dynamics. Possible approaches include using a latent recurrent state to capture historical experiences or employing a meta-learner methodology [22].

4.2 Stateful agents

The proposed model can be augmented with sequential neural networks, such as LSTMs or transformers, which have proven successful in capturing memory and attention to information [26]. This could help train agents to consider future rewards more effectively.

4.3 Rewards

Rewards for the trained agents should account for both intertemporal reward attribution and reward attribution in the presence of multiple modules determining actions. Reinforcement learning relies on rewards as training signals. The rewards for each module being optimized can be expressed as a combination of rewards from various action steps, from the agent's perspective as well as broader statistics, such as overall output, where cooperative action can be rewarded. Social concerns like income inequality have been shown to serve as a component of system-wide rewards in the literature [29]. As a distinguishing feature compared to utility calculation, this research employs Epstein-Zin preferences [8] for utility computation, separating risk aversion and inter-temporal consumption preference parameters for more accurate agent parametrization. Rewards are modeled as advantage estimations of utility differences, aggregated at the reward-constituent level and combined according to a set of hyperparameters in the polynomial ϕ. The Reward Mixture comprises immediate advantage rewards of the agent after taking an action, the reward of the agent following all trading and choice activities within a single loop time-step covering all economic activities, as well as immediate operation relevant global rewards such as collective production, and global rewards at the end of the simulation loop time-step addressing global objectives like reducing economic fluctuation amplitude and promoting social welfare:

$$R_{i,t}^{Mix} \sim \phi(R_{i,immediate}, R_{i,t}, R_{global,immediate}, R_{global,t}) \tag{5}$$

4.4 Input to ML models

Each model that is trained for agents must at least get as input the deciding agent's own state, further inputs regarding the decision must be provided as input. These inputs can be directly relevant aspects such as a trade offer, or can be general signals such as prices from the market, the information flow structure is chosen to imitate the real world, where agents make decisions from multiple sources, but they are not omniscient, the information flow structure shall be closely related with the communication structure that is being observed in the real world.

4.5 Curricula Learning

Training multiple models each governing different aspects for agents such as borrowing decisions, investment decisions and trade decisions is challenging, one way to deal with the complexity is curriculum learning, where the different functionalities of the agents are being enabled and trained incrementally, which can also help to tackle with the training instability problem of reward attribution from the outcomes of multiple machine learning modules of the agents in a temporal setting. Curriculum learning is proven to be beneficial for developing increasingly complex capabilities [19] can greatly speed up and improve the success of the training.

4.6 Calibration to real-world

Calibrating a deep multi-agent model to the statistics and phenomena that is being observed in the real world can be challenging. The only hard inputted information is the inter-company technology production matrix(input-output) that is being used by business to produce sectoral outputs, further fine-tuning of the system can be done by training the models with a reward signal reflecting the divergence of the simulated statistics from the real world phenomena.

4.7 Parallelization and Optimization

To further enhance the scalability of the model, we can leverage parallelization techniques and optimization algorithms. By distributing the learning process across multiple processors or GPUs, we can significantly speed up the training phase. Additionally, optimization techniques such as experience replay, prioritized sampling, and parameter optimization can be used to enhance learning efficiency and performance. The proposed multi-agent reinforcement learning model for the pension ecosystem can be adapted to handle large-scale systems with millions of agents through a combination of distributed learning, hierarchical structures, model abstraction, and optimization techniques.

5 Conclusion

This paper presents a multi-agent reinforcement learning model for the pension ecosystem, focusing on optimizing contributor saving and investment strategies. By incorporating multiple agents, the research aims to explore the effects of endogenous and exogenous shocks, business cycles, and policy decisions on contributor behavior. The model generates synthetic and diverse income trajectories, paving the way for more inclusive savings strategies.

Building on traditional econometric models, the research seeks to learn meta-strategies for contributor agents that demonstrate robustness amid changing environmental dynamics. While the methodology has been thoroughly defined, the model's calibration and interpretation of simulations within the pension ecosystem will be the subject of future research phases, with experimentation and results yet to be conducted and analyzed.

References

1. Asano, Y.M., Kolb, J.J., Heitzig, J., Farmer, J.D.: Emergent inequality and endogenous dynamics in a simple behavioral macroeconomic model (Jul 2019), `http://arxiv.org/abs/1907.02155`, arXiv:1907.02155 [econ, q-fin]
2. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q.: JAX: composable transformations of Python+NumPy programs (2018), `http://github.com/google/jax`
3. Campanale, C., Fugazza, C., Gomes, F.: Life-cycle portfolio choice with liquid and illiquid financial assets. Journal of Monetary Economics **71**, 67–83 (Apr 2015). `https://doi.org/10.1016/j.jmoneco.2014.11.008`, `https://linkinghub.elsevier.com/retrieve/pii/S0304393214001652`
4. Campbell, J.Y., Viceira, L.M.: Strategic asset allocation: portfolio choice for long-term investors. Oxford University Press, New York (2002)
5. Cocco, J.F., Gomes, F.J., Maenhout, P.J.: Consumption and Portfolio Choice over the Life Cycle. Review of Financial Studies **18**(2), 491–533 (2005). `https://doi.org/10.1093/rfs/hhi017`, `https://academic.oup.com/rfs/article-lookup/doi/10.1093/rfs/hhi017`
6. Cont, R., Wagalath, L.: FIRE SALES FORENSICS: MEASURING ENDOGENOUS RISK: FIRE SALES FORENSICS: MEASURING ENDOGENOUS RISK. Mathematical Finance **26**(4), 835–866 (Oct 2016). `https://doi.org/10.1111/mafi.12071`, `https://onlinelibrary.wiley.com/doi/10.1111/mafi.12071`

7. of England, B.: Announcement of additional measures to support market functioning (10 2022), https://www.bankofengland.co.uk/news/2022/october/bank-of-england-announces-additional-measures-to-support-market-functioning
8. Epstein, L.G., Zin, S.E.: Substitution, risk aversion, and the temporal behavior of consumption and asset returns: A theoretical framework. Econometrica **57**(4), 937–969 (1989), http://www.jstor.org/stable/1913778
9. Frostig, R., Johnson, M.J., Leary, C.: Compiling machine learning programs via high-level tracing. Systems for Machine Learning **4**(9) (2018)
10. Gibbons, R., Waldman, M.: Task-specific human capital. The American Economic Review **94**(2), 203–207 (2004), http://www.jstor.org/stable/3592883
11. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE international conference on robotics and automation (ICRA). pp. 3389–3396. IEEE (2017)
12. Guvenen, F., Ozkan, S., Song, J.: The Nature of Countercyclical Income Risk (May 2012). https://doi.org/10.3386/w18035, https://www.nber.org/papers/w18035
13. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2
14. Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., van Zee, M.: Flax: A neural network library and ecosystem for JAX (2020), http://github.com/google/flax
15. Impavido, G., Tower, I.: How the financial crisis affects pensions and insurance and why the impacts matter (Jul 2009)
16. Merton, R.C.: Optimum consumption and portfolio rules in a continuous-time model. Journal of Economic Theory **3**(4), 373–413 (Dec 1971). https://doi.org/10.1016/0022-0531(71)90038-X, https://linkinghub.elsevier.com/retrieve/pii/002205317190038X
17. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
18. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
19. Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., Stone, P.: Curriculum learning for reinforcement learning domains: A framework and survey. arXiv preprint arXiv:2003.04960 (2020)
20. Papaioannou, M.G., Rentsendorj, B.: Sovereign Wealth Fund Asset Allocations—Some Stylized Facts on the Norway Pension Fund Global. Procedia Economics and Finance **29**, 195–199 (2015). https://doi.org/10.1016/S2212-5671(15)01122-3, https://linkinghub.elsevier.com/retrieve/pii/S2212567115011223
21. Pichler, A., Farmer, J.D.: Simultaneous supply and demand constraints in input–output networks: the case of Covid-19 in Germany, Italy, and Spain. Economic Systems Research **34**(3), 273–293 (Jul 2022). https://doi.org/10.1080/09535314.2021.1926934, https://doi.org/10.1080/09535314.2021.1926934
22. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: International conference on learning representations (2017)
23. del Rio-Chanona, R.M., Mealy, P., Beguerisse-Díaz, M., Lafond, F., Farmer, J.D.: Occupational mobility and automation: a data-driven network model. Journal of The Royal Society Interface **18**(174), 20200898 (Jan 2021). https://doi.org/10.1098/rsif.2020.0898, https://royalsocietypublishing.org/doi/10.1098/rsif.2020.0898
24. Samvelyan, M., Rashid, T., de Witt, C.S., Farquhar, G., Nardelli, N., Rudner, T.G.J., Hung, C.M., Torr, P.H.S., Foerster, J., Whiteson, S.: The StarCraft Multi-Agent Challenge (Dec 2019), http://arxiv.org/abs/1902.04043, arXiv:1902.04043 [cs, stat]
25. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017). https://doi.org/10.48550/ARXIV.1707.06347, https://arxiv.org/abs/1707.06347
26. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All you Need. In: Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
27. Yang, Y., Wang, J.: An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective (Mar 2021), http://arxiv.org/abs/2011.00583, arXiv:2011.00583 [cs]
28. Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y.: The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games (Jul 2022), http://arxiv.org/abs/2103.01955, arXiv:2103.01955 [cs]
29. Zheng, S., Trott, A., Srinivasa, S., Naik, N., Gruesbeck, M., Parkes, D.C., Socher, R.: The AI Economist: Improving Equality and Productivity with AI-Driven Tax Policies. arXiv:2004.13332 [cs, econ, q-fin, stat] (Apr 2020), http://arxiv.org/abs/2004.13332, arXiv: 2004.13332