# Robust Policy Design in Agent-Based Simulators using Adversarial Reinforcement Learning

**Akash Agrawal**[1], **Joel Dyer**[1], **Aldo Glielmo**[2], **Michael Wooldridge**[1]
[1]University of Oxford, Oxford, UK
[2]Applied Research Team, Directorate General for IT, Bank of Italy, Rome, Italy*
agrawal.akash9702@gmail.com, joel.dyer@cs.ox.ac.uk, aldo.glielmo@bancaditalia.it, mjw@cs.ox.ac.uk

## Abstract

Agent-based models (ABMs) of complex socioeconomic systems provide policymakers with highly detailed synthetic environments in which policy interventions can be designed and optimised, for example through reinforcement learning (RL). Although they model systems at a fine level of granularity, ABMs will in general be imperfectly specified, such that there exists non-negligible distributional shifts between agent-based simulators and the complex socioeconomic systems they attempt to model. Policies that are tested and optimised in these simulated environments may therefore exhibit brittleness in the face of such distributional shifts, presenting a challenge to the use of ABMs in real-world policy design. In this work, we investigate the robustness of policies designed on ABMs using RL-based methods. We show that the naive application of RL to discover optimal policies can result in policies that are vulnerable to simulated distributional shifts, where small changes in the model lead to noticeable drops in policy effectiveness. To address this, we explore multiple types of adversarial training methods aimed at improving policy robustness. We show that these methods not only make policies resilient to the specific simulation-to-reality shifts they were trained on, but also to unforeseen types of shifts. In experiments, we demonstrate these effects in a complex macroeconomic ABM with multiple types of agents, including households, firms, a central bank, and government. Our work highlights the importance of robustness in RL-based policy optimisation using ABMs, and showcases the potential of adversarial training as a robustification strategy.

## Introduction

Agent-based models (ABMs) have gained considerable attention for their capacity to simulate complex socioeconomic phenomena by modeling the interactions of diverse, heterogeneous agents. By directly simulating the behavior of various entities, such as individuals, organizations, and institutions, ABMs provide a flexible framework for analyzing emergent phenomena, and offer insights into how micro-level interactions give rise to macro-level patterns (Axtell and Farmer 2022). In economics, for instance, ABMs may facilitate a more granular exploration of disequilibrium dynamics and emergent behaviors, such as financial crises or business cycles (Bookstaber 2017; Gatti et al. 2008; Raberto, Teglio, and Cincotti 2012; Wiese et al. 2024).

This gives agent-based modelling the potential to be a very valuable tool for policymakers, who can use ABMs to simulate the effects of potential interventions before implementing them in the real world (Deissenberg, Van Der Hoog, and Dawid 2008). In particular, ABMs may provide policymakers with highly detailed synthetic environments in which policies and interventions can be designed, evaluated, and optimised by simulating various scenarios and observing their potential outcomes (Lempert 2002; Dawid, Fagiolo et al. 2008). Indeed, some recent work has investigated the use of reinforcement learning (RL) in such environments to optimize policies, allowing decision-making agents to adapt and learn strategies through trial and error (Mi et al. 2023; Brusatin et al. 2024; Yao et al. 2024). Such RL-based approaches are particularly effective in complex environments where traditional analytical methods fall short, such as in the problems of designing fiscal policies or managing inflation in a simulated economy (Olmez et al. 2024; Dong, Dwarakanath, and Vyetrenko 2024).

## Policy Optimization in Misspecified ABMs

Despite their advantages as highly detailed testbeds for policy experimentation and optimization, ABMs share fundamental limitations common to all models. A primary issue is the *simulation-to-reality gap*: models are imperfectly specified representations of real-world systems. Socioeconomic systems in particular are highly complex; modeling them often entails sacrifices in granularity, and the implemented behavioural rules will generally fail to fully capture the complex decision-making processes of real-world agents. This means that models generally do not fully capture the true dynamics of the system, such that there can be discrepancies between ABMs and the real-world environments they emulate.

As a result of these distributional shifts, the real-world utility of policies optimised in ABMs via RL methods is threatened: policies that perform well in the simulated environment may fail to generalize effectively to real-world conditions, leading to diminished performance or, even more dangerously, unintended or unsafe consequences when ap-

*The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of Banca d'Italia.

plied in practice. For this reason, strategies for accounting for non-negligible simulation-to-reality gaps are an extremely valuable component to simulation-based policy optimisation via RL in ABMs. Unfortunately, to date, such techniques have been neglected in the literature on policy optimisation in ABMs via RL (Mi et al. 2023; Olmez et al. 2024).

### Our Contribution

In light of this problem, we investigate in this paper techniques for improving the robustness of policies discovered and optimised in agent-based simulators with respect to distributional shifts. In particular, we investigate the use of robust training methods to train RL polices such that they perform well across a range of simulated distributional shifts. Our work makes the following contributions:

- We demonstrate that RL based policies perform worse on environments with simulated distributional shifts in parameters governing behavioural dynamics in both a simple epidemic ABM and a complex macroeconomic ABM.
- We identify and evaluate suitable training methods in the literature on robust RL.
- We show, using both the aforementioned small-scale epidemic agent-based simulator and the macroeconomic ABM, that such robust RL methods can train policies that exhibit considerably improved performance in simulated distributional shifts, in comparison to policies trained with standard RL methods.
- Through the above, we highlight the importance of robust policy design in ABMs, and signal the need for future research on this topic in the field of agent-based modelling.

## Related Work

RL has seen significant advancements in its application to socioeconomic models, including ABMs, with studies exploring its potential to design more adaptive and effective policies. Typically, the RL agent represents a central planning authority like the government and learns optimal decisions on issues such as monetary policies. Examples of the application of RL to design policies in non-agent-based socioeconomic simulators include Atashbar and Shi (2023), who discuss the application of deep RL within a Dynamic Stochastic General Equilibrium (DSGE) model, showing that agents converge to close-to-optimal policies. A further example is Koster et al. (2022), who demonstrate in a DSGE model that RL can learn directly from human feedback to design socially accepted and optimal economic policies. Here, the authors train an RL agent to design policies about redistribution of shared resources that were favoured by a majority of human participants during voting. Chen et al. (2021) additionally showed that deep RL algorithms were promising tools for learning optimal monetary policy when the agent represents the central bank.

In the context of agent-based socioeconomic simulators, Olmez et al. (2024) use RL in an ABM of the UK housing market to explore the mitigation of market shocks through policy interventions. The agent, which represents the central bank, learns to adjust interest rates to maintain market stability. Further, Zheng et al. (2022) simulate a small society with RL agents representing households, who play a "gather-and-build" game, along with a central planner (the government), who decide the marginal taxation policy at various income levels. They show that the tax policy found by the RL agent outperforms traditional tax policies like the Saez formula or the US federal tax policy in improving overall productivity and economic equality. Brusatin et al. (2024) use MARL in a macroeconomic ABM to make policies about price-quantity strategies for consumer goods firms. The paper shows that RL agents adapt strategies based on competition, improving output but sometimes increasing instability. Despite the growing interest in and use of RL to design optimal policies using agent-based simulators as training environments, the use of RL training procedures that encourage robustness to, and account for, distributional shifts arising from simulation-to-reality gaps is lacking.

## Method

Our goal is to demonstrate how agent-based simulators of complex socioeconomic systems can be used to design and optimise social policies, while accounting for the fact that the simulator will generally not perfectly capture the true data-generating process. To this end, we investigate the use of *robust* RL procedures as a means to optimise policies in simulated complex systems.

### Formalization

We formalize the problem of optimizing various kinds of policy decisions from the government's perspective as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, S, O, A, T, T_{\mathcal{S}}, R, \gamma, \phi, \phi_0)$. This MDP is built on an agent-based simulator, $\mathcal{S}$. $S$ is the state space representing the full system state of the simulator. $O$ is the observation space for the government agent. $A$: Action space, containing all possible actions the agent can take (independent of observation space). $T : S \times A \to \Delta(S)$ is the MDP's transition function governing state transitions, which subsumes the simulator's evolution function, $T_{\mathcal{S}} : S \to \Delta(S)$. Note that the agent-based simulator in itself does not need an action to evolve – it evolves due to various stochastic and deterministic processes on its own. $R : S \times A \times S \to \mathbb{R}$ is the reward function, providing feedback based on state-action transitions. $\gamma$ is the discount factor. $\phi$ is the set of parameters governing the simulator's evolution dynamics and $\phi_0$ is the initial parameters (e.g. number of agents), required for setting up the system.

The agent interacts with the environment by observing a partial observation of the state $s$, $o(s) \in O$, taking actions $a \in A$, and receiving rewards $R(s, a, s')$. Transitions depend on both the agent's actions and the simulator dynamics – in each step, the agent's actions change some parameters governing the simulator's dynamics, and the simulator proceeds with a simulation step that evolves its state, which is the next environmental state.

## Problem Setting

To assess robustness, we simulate distributional shifts that reflect the modeller's uncertainty regarding the true data-generating process, which will in general not match that of the simulator exactly. These shifts are modeled as variations in key parameters, which reflect the assumptions made by the ABM designer about agent behaviors. Since real-world agent behaviors often differ from these assumptions, modifying parameters allows us to simulate more unpredictable, realistic scenarios, providing a better test of policy robustness.

We define robustness using two evaluation criteria: fixed parameter-shifts and adversarial parameter-shifts. Absolute robustness is not defined due to the unknown nature of an optimal policy. The definitions of these two kinds of robustness are dependent on the expected cumulative reward of a policy $\mu$ in environment $\mathcal{E}$ characterised by an MDP $M$:

$$\mathcal{R}(\mu) \quad = \quad \mathbb{E}_{\mu}\left[\sum_{t=1}^{T} r_t\right] \qquad (1)$$

where $T$ is the number of steps in the episode and $r_t$ is the reward in the $t^{\text{th}}$ step.

### Fixed-Parameter Shift Robustness

**Definition 1** (Fixed Parameter Shift)**.** *A fixed-parameter shift on an MDP $M$ is the a change in the specification of its simulator, specifically the parameters governing the simulator's dynamics. Hence, it can be represented by an MDP $M'$, which has the pair $(\phi', T'_{\mathcal{S}})$ instead of $(\phi, T_{\mathcal{S}})$. $\phi'$ are the new parameters of the simulator and $T_{\mathcal{S}'}$ is the updated evolution function.*

We say that $\mu_1$ is more robust to the $M \to M'$ parameter shift than $\mu_2$ if the expected reward of $\mu_1$ on $M$ is better than or similar to that of $\mu_2$ i.e. $\mathcal{R}(\mu_1) \gtrapprox \mathcal{R}(\mu_2)$, and if there is an increase in the expected reward on $M'$ i.e. $\mathcal{R}'(\mu_1) > \mathcal{R}'(\mu_2)$.

### Adversarial Parameter Shift Robustness

**Definition 2** (Adversarial Parameter Shift)**.** *An adversarial parameter shift on an MDP $M$ involves an adversary operating with a policy $\nu$ that dynamically alters the parameters governing the simulator's dynamics, $\phi$, at each step. Hence, the MDP changes at each step. This results in a non-stationary environment, which has parameters $(\phi_t, T_{\mathcal{S},t})$ at time step t such that $(\phi_0, T_{\mathcal{S},0}) = (\phi, T_{\mathcal{S}})$.*

We modify the definition of the reward over the episode for a government policy $\mu$:

$$\mathcal{R}_{\nu}(\mu) \quad = \quad E_{\mu,\nu}\left[\sum_{t=1}^{T} R_t(s_t, a_{1,t}, a_{2,t}, s_{t+1})\right], \quad (2)$$

where $R_t$ is the reward function of the MDP $M_t$.

We again say that $\mu_1$ is more robust than $\mu_2$ to the $(M, \nu)$ adversarial parameter shift if the expected reward of $\mu_1$ on $M$ is better than or similar to that of $\mu_2$ i.e. $\mathcal{R}(\mu_1) \gtrapprox \mathcal{R}(\mu_2)$, and if there is an increase in the expected reward on $(M, \nu)$ i.e. $\mathcal{R}_{\nu}(\mu_1) > \mathcal{R}_{\nu}(\mu_2)$.

## Adversarial Training Methods

In this section, we describe *adversarial* training methods for learning robust policies via RL. In particular, we consider two related flavours of adversarial RL – Stepwise Adversarial RL and Episode-wise Adversarial RL – in which an additional agent, called the adversary, imposes distributional shifts to the agent-based simulator in an attempt to reduce the reward observed by the policy agent. We assume throughout that the distributional shifts imposed by the Adversary are parameterised distributional shifts, such that the action space of the Adversary is a space of parameter values. To begin, we describe the standard, non-adversarial ("Vanilla") RL training method.

**Vanilla** The vanilla method is the simplest RL approach used to train the government agent to optimize a given reward function in the environment. It follows a standard RL training loop, where the policy is updated $N_{\text{iter}}$ times. After each update, a rollout is generated over $k$ episodes, each of length $T$. A sufficiently large $N_{\text{iter}}$ is selected to ensure policy convergence. The Vanilla training method is described in Algorithm 1.

```
Algorithm 1: Vanilla Training
    **Input:** Environment $\mathcal{E}$, stochastic government policy $\mu$,
    length of episode $T$, episodes per rollout $k$
    **Initialize:** Learnable parameters $\theta_0^{\mu}$
    **for** $i = 1, 2, \ldots, N_{\text{iter}}$ **do**
        $\{(s_t^i, a_t^i, r_t^i)\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta_{i-1}^{\mu}}, k \cdot T)$
        $\theta_i^{\mu} \leftarrow \text{update\_policy}\left(\{(s_t^i, a_t^i, r_t^i)\}, \mu, \theta_{i-1}^{\mu}\right)$
    **end for**
```

**Stepwise Adversarial (Step-Adv)** This approach is based on Robust Adversarial Reinforcement Learning from Pinto et al. (2017). In stepwise adversarial training, during each step, both the government and the adversary act, rather than just the government. The goal of the adversary is to minimize the government's reward through its actions.

The adversary's observation space is the same as that of the government. The adversary has a distinct action space, through which it can alter the environment's parameters (i.e. it can change $\phi$). The government's reward at time step $t$ is $r_t^1$ – which is the same ($r_t$) as before – and the adversary's reward is $r_t^2 = -r_t = -r_t^1$.

Both agents are trained in an alternating manner: for $k$ episodes, the adversary's policy is kept constant while the government's policy is updated; and then, for the next $k$ episodes, the government's policy is kept constant while the adversary's policy is updated. In essence, this method tries to model unknown train-test distribution shifts as adversarial changes in the environment's configuration during training. We describe this method formally in Algorithm 2.

**Episodic Adversarial (Ep-Adv)** In episodic adversarial training, the adversary acts once before each episode begins, altering the environment for the entire episode to reduce the government's reward. Both the government and adversary are trained alternately: for some episodes, the government's

Algorithm 2: Stepwise Adversarial Training

```
Input: Environment $\mathcal{E}$, stochastic government and adversary policies $\mu$ and $\nu$, length of episode $T$
Initialize: Learnable parameters $\theta_0^\mu$ and $\theta_0^\nu$
for $i = 1, 2, \ldots, N_{\text{iter}}$ do
    $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta_{i-1}^\mu}, \nu_{\theta_{i-1}^\nu}, 2 \cdot T)$
    $\theta_i^\mu \leftarrow \text{update\_policy}\left(\{(s_t^i, a_t^{1i}, r_t^{1i})\}, \mu, \theta_{i-1}^\mu\right)$
    $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \text{roll}(\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_{i-1}^\nu}, 2 \cdot T)$
    $\theta_i^\nu \leftarrow \text{update\_policy}\left(\{(s_t^i, a_t^{2i}, r_t^{2i})\}, \nu, \theta_{i-1}^\nu\right)$
end for
```

policy is updated while the adversary's remains constant, and vice versa.

Unlike in stepwise training, here, it does not make sense for the adversary to have a proper observation space, since it always observes the same initial configuration of the environment. Hence, we set the observation space of the adversary nominally to be a singlet, $\{0\}$. This makes it a Bandit problem (Robbins 1952), as the adversary selects actions without observing the environment. The action space of the adversary is defined similarly as in the section on stepwise adversarial training. The reward function of the adversary is the negative of the reward the government receives over the course of the episode:

$$r_{\text{adv}} = -\sum_{t=1}^{T} r_t. \tag{3}$$

We describe the episodic adversarial training method in Algorithm 3.

Algorithm 3: Episodic Adversarial Training

```
Input: Environment $\mathcal{E}$, stochastic government and adversary policies $\mu$ and $\nu$, length of episode $T$, number of episodes per rollout $k$
Initialize: Learnable parameters $\theta_0^\mu$ and $\theta_0^\nu$
for $i = 1, 2, \ldots, N_{\text{iter}}$ do
    $\{(s_t^i, a_t^{1i}, r_t^{1i})\} \leftarrow \text{roll}_1(\mathcal{E}, \mu_{\theta_{i-1}^\mu}, \nu_{\theta_{i-1}^\nu}, k \cdot T)$
    $\theta_i^\mu \leftarrow \text{update\_policy}\left(\{(s_t^i, a_t^{1i}, r_t^{1i})\}, \mu, \theta_{i-1}^\mu\right)$
    $\{(s_t^i, a_t^{2i}, r_t^{2i})\} \leftarrow \text{roll}_2(\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_{i-1}^\nu}, k)$
    $\theta_i^\nu \leftarrow \text{update\_policy}\left(\{(s_t^i, a_t^{2i}, r_t^{2i})\}, \nu, \theta_{i-1}^\nu\right)$
end for
```

Unlike stepwise training, the adversary acts only once before each episode, treating the episode as a single step. Thus, we use different rollout functions: $\text{roll}_2$ for the adversary, with $k$ tuples, and $\text{roll}_1$ for the government, with $k \cdot T$ tuples. The adversary's reward $r_t^{2i}$ applies to the entire episode, while the government's $r_t^{1i}$ applies to each step within the episode.

## Experimental setup

In this section, we discuss two case studies that demonstrate the utility of robust RL methods when designing optimal policies using socioeconomic agent-based simulators.

### Agent-based simulators tested

**A SIRS Epidemic Simulator** We employ a Susceptible-Infected-Recovered-Susceptible (SIRS) model (Kermack and McKendrick 1927) to simulate disease spread within a fixed population. The SIRS model captures the cyclical nature of infection, recovery, and re-susceptibility. Starting with a population size $N$, a fraction $i_0$ is initially infected, with the rest being susceptible. The model's dynamics are defined by three parameters: $\alpha$ (infection rate), $\beta$ (recovery rate), and $\gamma$ (re-susceptibility rate). At each time step, probabilistic transitions between susceptible (S), infected (I), and recovered (R) states occur based on these parameters. The transitions are defined as follows:

$$N_{S\to I}^{t+1} = \text{Binomial}\left(N_S^t, 1 - \exp\left(-\frac{\alpha \cdot N_I^t}{N}\right)\right), \tag{4}$$

$$N_{I\to R}^{t+1} = \text{Binomial}\left(N_I^t, 1 - \exp(-\beta)\right), \tag{5}$$

$$N_{R\to S}^{t+1} = \text{Binomial}\left(N_R^t, 1 - \exp(-\gamma)\right). \tag{6}$$

The simulation runs for a fixed time of $T$ time steps. In the SIRS model, the government's role is to minimize infections by imposing lockdowns. A lockdown reduces the infection rate to zero, preventing new infections. However, lockdowns incur a cost, so the government must balance the timing of lockdowns with periods of interaction, which may increase infections.

The observation that the government receives is $(N_S^t, N_I^t, N_R^t)$ at time $t$. The government takes a binary action – whether to impose a lockdown or not. The adversary's action consists of setting the value of only 1 parameter – but we train separate policies adversarially for each of the 3 transition parameters $(\alpha, \beta, \gamma)$ in the description of the SIRS model above. The government receives the following reward at each time step:

$$r_t = -N_I^t - C. \tag{7}$$

Here, $C$ represents our belief about the cost of a lockdown imposed at any time step.

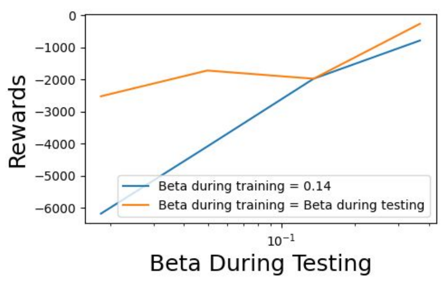


Figure 1: Rewards observed for Vanilla RL agents tested across different recovery rates ($\beta$) in the SIRS model. The blue line represents the Fixed-Parameter Performance, where a policy trained on fixed parameters ($\beta = 0.14$) is tested on varying $\beta$ values. The orange line represents the Matched-Parameter Performance, where policies were trained and tested on matching $\beta$ values ($\beta$ ranging from 0.02 to 0.37).

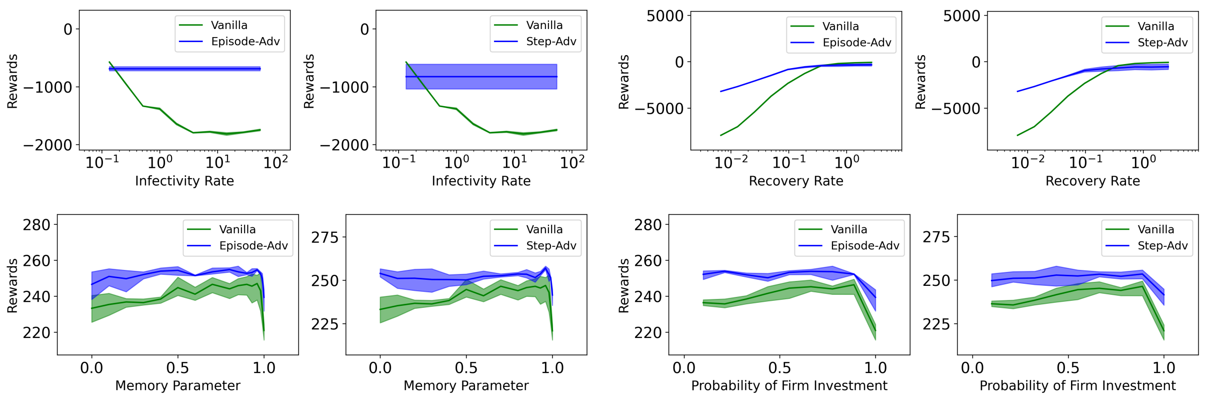


Figure 2: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the SIRS (top row) and M-ABM (bottom row) models. The shaded region indicates the maximum and minimum rewards obtained by various policies with different seeds/initialisations, but trained in the same manner; the line plot indicates the mean of policies' rewards. Adversarially trained RL agents exhibit enhanced robustness to single-parameter shifts, achieving better worst-case rewards.

**M-ABM** The M-ABM model (Assenza, Delli Gatti, and Grazzini 2015) is an agent-based macroeconomic model that simulates interactions among various economic agents. The model aims to capture realistic macroeconomic dynamics through the actions of different agents in interconnected markets. In its design, the model includes four main types of agents: households, consumption-good producing firms (C-firms), capital-producing firms (K-firms), and banks. These agents interact across several markets, such as labor, goods, and credit markets, to represent a complex economic system.

The behavior of each agent type is characterized by a set of rules that govern their decision-making processes. The basic behaviours of each agent as well as any specific behaviours important to our experiments are described as follows:

- **Households:** Households are the primary suppliers of labour and the main consumers of goods produced in the economy. For simplicity, they are either workers (who work at firms and supply labour) or capitalists (who own a firm and get returns on investments). Workers earn income through wages if employed and through government transfers if unemployed. Households' consumption decisions depend on their disposable income and overall wealth. At time step $t$, household $c$ has a notion of stable income $\bar{Y}_{c,t}$ and targets of particular amount of consumption $C_{c,t}$. Specifically,
$$\bar{Y}_{c,t} = \xi \cdot \bar{Y}_{c,t-1} + (1-\xi) \cdot Y_{c,t} \tag{8}$$
$$C_{c,t} = \bar{Y}_{c,t} + \chi \cdot D_{c,t} \tag{9}$$
where $\xi$ is called the *memory parameter* and $\chi$ is the *ratio of consumption to wealth*. Further, $Y_{c,t}$ is the income and $D_{c,t}$ are the deposits (total wealth) stored in the bank.
- **C-firms (Consumption-Good Producing Firms):** C-firms produce consumption goods demanded by households. They set prices for their goods and wages for labor. Their goal is to maximize profits, and they do so by adjusting the quantity of goods produced, denoted as $Y_{i,t}$, based on demand forecasts and production costs. At full capacity, the $i^{\text{th}}$ firm produces:
$$\hat{Y}_{i,t} = \min\left(\alpha N_{i,t}, \kappa K_{i,t}\right) \tag{10}$$
where $\alpha$ is the *productivity of labour* and $\kappa$, is the *productivity of capital* $N_{i,t}$ is the amount of labour and $K_{i,t}$ is the amount of capital the firm has at time $t$. The firm can decide to invest at each time step, which allows it to adjust its capital stock. The *probability of investing* at any time step is $\gamma$.
- **K-firms (Capital-Good Producing Firms):** K-firms are responsible for producing capital goods, which are required by C-firms for their production processes. These capital goods are typically long-term assets like machinery or equipment. For simplicity, they only need labour to produce capital-goods, and not any durable inputs.
- **Banks:** Banks facilitate the financial operations in the economy by providing credit to both C-firms and K-firms and taking deposits from consumers. Banks thereby play a key role in funding investments and enabling firms to finance their operations. For simplicity, the number of banks in the economy is set to 1.

We note that the 5 parameters defined above are chosen by the designers of the model, and are constants during the simulation i.e. are set beforehand and are not affected by the processed of the simulation.

The *Government* in the M-ABM model plays a critical role in stabilizing the economic system and ensuring welfare across agents (Assenza et al. 2018). The government's primary role is to collect taxes from both households and firms, based on their income and profits, respectively. This tax revenue is redistributed through public expenditures, such as unemployment subsidies. The net effect of taxes and subsidies influences disposable income, firm profitability, and, consequently, household consumption and firm investment decisions.

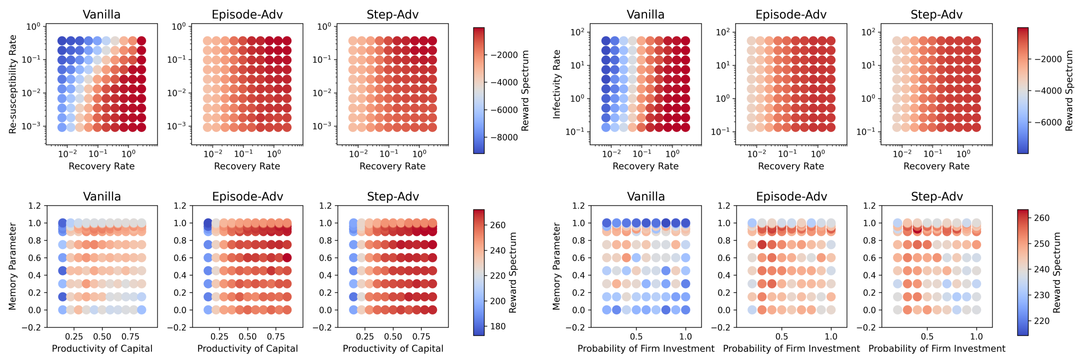


Figure 3: Rewards observed by testing Vanilla (left-most panels) and adversarially trained (middle and right-most panels) policy agents on different parameter settings in the SIRS (top row) and M-ABM (bottom row) models. Red (resp. blue) indicates higher (resp. lower) rewards. Adversarially trained RL agents exhibit enhanced robustness to double-parameter shifts, achieving better rewards across a range of distributional shifts.

**Observation Space** Given that M-ABM is a complex model with a high-dimensional state space, we leverage our prior knowledge of the model and the features most relevant to optimizing the reward function to design a partial observation space for the government agent. Specifically, we select a subset of features that should be visible to the agent. Additionally, in an economic setting, the temporal evolution of macroeconomic variables is crucial for policy decisions, so each observation includes data from two consecutive time steps. Thus, the observation is not strictly Markovian, as it depends on information from multiple time steps. However, we treat this as an approximation of an MDP for the purposes of formalization. The government agent receives observations of the Gross Domestic Product (GDP) and Bank Deposits (total money deposited in the bank) for both the current and previous time steps.

**Action Space** It can take actions to set the tax rate (income tax of households) and subsidy rate (benefits provided to unemployed workers) at each time step – both within $(0, 1)$. The adversary's action consists of setting the value of only $1$ parameter – but we train separate policies adversarially for each of the $5$ behaviour-governing parameters $(\xi, \chi, \gamma, \alpha, \kappa)$ in the description of the M-ABM model above.

**Reward Function** The reward function is intelligently designed to balance economic growth and stability, focusing on current GDP relative to potential GDP and penalizing fluctuations. The maximum potential GDP in the M-ABM model is:

$$\text{Max GDP} = \alpha \times W \times P_0 \tag{11}$$

where $W$ is the number of workers and $P_0$ is the initial price level. GDP crashes at time $t$ are measured as the drop from the previous period:

$$\text{GDP Crash}_t = \max(0, \text{GDP}_{t-1} - \text{GDP}_t). \tag{12}$$

The overall reward is:

$$r_t = \frac{\text{GDP}_t}{\text{Max GDP}} - B\left(\frac{\text{GDP Crash}_t}{\text{Max GDP}}\right)^2, \tag{13}$$

where $B$ balances growth and stability. This encourages sustainable growth by rewarding high, stable GDP and penalizing sharp drops.

## Experimental Results

We will begin by illustrating through a simple experiment in the SIRS model that policies trained using the Vanilla method degrade in performance even on simple simulated distributional shifts; hence, they are not robust to these shifts.

To combat this, we will show that robustness of policies can be improved through adversarial training. Specifically, we show positive results in three types of distributional shifts: (a) when one parameter is shifted, (b) when two parameters are simultaneously shifted, and (c) when parameters are shifted during episodes adversarially.

For each type of training – Vanilla, Episode-Adv, and Step-Adv – and for each type of adversary involved in training, we train multiple policies using different seeds/initializations to ensure that our results are not artifacts of specific initializations or random fluctuations. Further, for each testing-time environment, given that the environments themselves have stochasticity, we use multiple seeds to initlialize them, test the policies on each, and compute the average reward.

In the SIRS experiments, we train our policies for standard values of the infectivity rate $(\alpha)$ and recovery rate $(\beta)$. During training, the infectivity rate was $4.48$, which corresponds to a probability of infection of $20 - 50\%$, depending on the number of infected people at that time step, $N_{I,t}$. The recovery rate was $0.14$, corresponding to a probability of recovery of around $14\%$.

In the M-ABM experiments, we train our policies for standard values of the following parameters: probability of investing $(\gamma)$, memory parameter $(\xi)$, and productivity of capital $(\kappa)$. During training, the probability of investing was $0.25$, the memory parameter was $0.96$, and the productivity of capital was $0.33$.

For SIRS, the adversarially trained models are trained against an adversary that controls the re-susceptibility rate within $(0.00091, 0.37)$. For M-ABM, the adversarially trained models are trained against an adversary that controls the value of the memory parameter within $(0.5, 0.99)$.

Further details of training (libraries used, simulators used, etc.) can be found in Appendix 10.

## Robustness Issues in Vanilla Training

Using the SIRS model, we test policies across environments where the recovery rate $\beta$ varies between 0.02 and 0.37. We generate two line plots to illustrate this in Figure 1:

- Fixed-Parameter Performance: Shows the rewards of a Vanilla policy trained on fixed parameters ($\alpha = 4.48, \gamma = 0.02, \beta = 0.14$) and tested across different environments.
- Matched-Parameter Performance: Displays rewards from multiple Vanilla policies trained on ($\alpha = 4.48, \gamma = 0.02$), where the training and testing environments share the same $\beta$ value, with $\beta$ varying between 0.02 and 0.37 across different policies.

While it's expected that a policy performs best when tested on the parameters it was trained on, this comparison will highlight the poor robustness of Vanilla training across even slight shifts in $\beta$. We will later show how adversarial training addresses these issues and improves robustness.

As we can see in Figure 1, the relative performance of the Vanilla policy trained on $\beta = 0.14$ becomes increasingly worse when compared to Vanilla policies trained and tested on the same $\beta$ value, as we go further from $\beta = 0.14$.

## Single Fixed-Parameter Shifts

For single-parameter fixed parameter shifts, we will use Max-Min-Mean plots to compare the performance of the Vanilla policy against that of an adversarially trained policy. In these plots, we evaluate policy performance across a range of values for each parameter using a line plot. The line represents the mean performance, while the shaded region indicates the maximum and minimum values. These statistics are calculated from the differently seeded policies trained for each configuration.

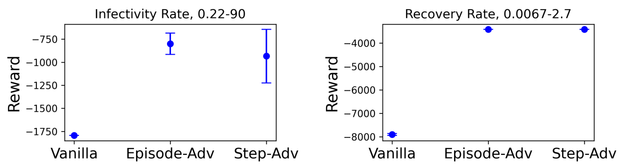


Figure 4: Rewards observed by testing Vanilla (left) and adversarially trained (middle and right) policy agents on a range of different continuously adversarially shifted parameter settings (mentioned above plots) in the SIRS model. The error bars indicate the maximum and minimum rewards and the plotted point is the mean. Adversarially trained RL agents exhibit enhanced robustness on adversarial parameter shifts, achieving better rewards across the board.

**SIRS** As we can see in the top row of Figure 2, for the SIRS experiments, there are certain values of "best-case" values of parameters that lead to good rewards for all policies. In these, Vanilla trained policies perform slightly better than adversarially trained policies. However, for the "worst-case" values, the adversarially trained policies perform significantly better than Vanilla trained policies. Further, this is seen not only when tested on shifts of the recovery rate (which they were adversarially trained against), but also on shifts of the infectivity rate. This suggests a level of generality in the robustness of adversarially trained policies. More examples of such generality (both when the training adversary is changed and when tested on different parameters' shifts) can be seen in Appendix 1.

**M-ABM** We plot the results of some of our experiments in the bottom row of Figure 2. We find that adversarially training leads to policy agents that perform much better than Vanilla trained policy agents even on shifts where the shifted parameter was not the one that the adversary controlled. This result is statistically significant, because the worst performing adversarially trained policy agents generally perform better than the best performing Vanilla agents, across seeds/initialisations. We provide further examples in Appendix 2.

## Two Simultaneous Fixed-Parameter Shifts

For dual-parameter fixed parameter shifts, we will shift two different parameters simultaneously during testing i.e. the simulator's specification would be different on two different parameters. We will use 2D heatmap plots to display the rewards of a given policy across a grid of two varying parameter values, when shifted simultaneously.

**SIRS** As we can see in the top row of Figure 3, adversarial training leads to policies that perform similarly or only slightly worse in the case of best-case shifts, as we saw in the single fixed-parameter shift experiments. Further, again, the adversarially trained policies perform significantly better on worst-case shifts – even more so than in the single shift case. These gains in robustness exist even when neither of the two simultaneously shifted parameters was the one that the adversary controlled during training, demonstrating the potential utility of such RL training procedures methods for introducing robustness to unseen distributional shifts. More examples of such improved performance can be seen in Appendix 3.

**M-ABM** We plot the results of a subset of experiments done with two simultaneous fixed parameters in Figure 3. As we can see, similar to the results we obtained for SIRS, even though the adversary, during training, only controlled the memory parameter, we see more robust performance even when the pairs of parameters shifted do not include it. Additional examples of such robust performance on other types of adversarial training are presented in Appendix 4. Further, we have analysed the differences in the types of robustness found in SIRS and M-ABM in Appendix 8.

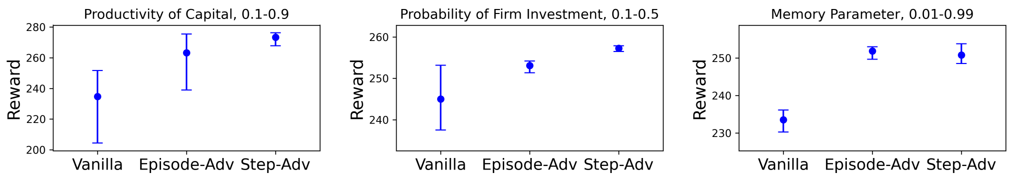


Figure 5: Rewards observed by testing Vanilla and adversarially trained policy agents on a range of different continuously adversarially shifted parameter settings (mentioned above plots) in the M-ABM model. The error bars indicate the maximum and minimum rewards and the plotted point is the mean. Adversarially trained RL agents exhibit enhanced robustness on adversarial parameter shifts, achieving better rewards across the board.

## Adversarial Parameter Shifts

For adversarial parameter shifts, we assess the performance of models in environments where an adversary is actively manipulating the value of a single parameter during each step of the episodes to minimize the reward of the government. For each adversary, we will use error-bar plots to compare the three types of training. Given that each we have trained multiple policies with different seeds, we once again show the max, min, and mean rewards. Adversarial parameter shifts bring a high amount of non-stationarity in the environment, since the adversary is changing the value of a key parameter governing its dynamics at each step.

**SIRS** We show results for two types of adversaries, which control:

- Infectivity Rate within $(0.22, 90)$
- Recovery Rate within $(0.0067, 2.7)$

As we can see in Figure 4, adversarially trained polices perform better than Vanilla trained policies not only when tested on shifts of the recovery rate, but also on shifts of the infectivity rate. This reinforces our claim about the generality in the robustness of adversarially trained policies. We observe that for all types of testing-time adversaries, the worst-performing adversarially trained policies perform better than the best-performing Vanilla trained policy. We provide more such figures in Appendix 5.

**M-ABM** We show results for three types of adversaries, which control:

- Productivity of Capital within $(0.1, 0.9)$
- Probability of Investing within $(0.1, 0.5)$
- Memory Parameter within $(0.01, 0.99)$

We plot the results of these experiments in Figure 5. As shown, adversarial training of both types leads to considerably better performance across the board – whether or not the testing-time adversarial shifts are of parameters controlled by the training-time adversary. As seen in the SIRS results, this is a stronger result than fixed shifts. Further, we note that the episodic adversarially trained agents perform well – even though they were tested against a stepwise adversary. However, the stepwise adversarially trained agents generally perform the best, as expected. We provide more such figures in Appendix 6.

## Further Results

We can further visualize the impact of adversarial training using the graphs in Appendix 9. In them, we can see that (1) Vanilla trained policies require more training episodes than adversarially trained policies to reach the optimal solution, (2) they are less stable once they're at the optimum, (3) they reach a slightly worse optimum, and (4) this difference in optimum performance increases when we shift the environments.

## Conclusions

Our work addresses the critical challenge of ensuring that policies optimized in complex socioeconomic simulation models perform reliably in real-world environments. To achieve this, we investigate robust reinforcement learning methods as a means of finding policies that are both performant and resilient to non-trivial changes in model behavior. Such distributional shifts mimic the simulation-to-reality gaps that arise when translating learned policies from simulation to the real world.

We are the first to explore robust reinforcement learning for designing policies in ABMs to bridge this gap. We demonstrate that policies trained in simulation can "break" under seemingly simple distributional shifts introduced within the agent-based simulator. We also show that adversarial training improves policy robustness to these shifts—not only to those encountered during training but also to other types of shifts. Notably, policies trained against an episodic adversary perform as well as those trained against a stepwise adversary. Furthermore, we find that the robustness of policies in simpler models like SIRS differs qualitatively from that in more complex models like M-ABM.

Our work highlights the pitfalls of non-robust policy optimization, the value of training for robustness alongside performance, and the need for greater emphasis on bridging the simulation-to-reality gap in agent-based modeling. Strengthening these methods will be essential for applying agent-based simulators to real-world policy design, enabling policymakers to maximize the benefits of imperfect simulation models of complex socioeconomic systems. This paper is part of a greater project in which we are currently exploring more statistically relevant ways of modeling distributional shifts across a wider range of ABMs.

## References

Assenza, T.; Colzani, P.; Delli Gatti, D.; and Grazzini, J. 2018. Does fiscal policy matter? Tax, transfer, and spend in a macro ABM with capital and credit. *Industrial and Corporate Change*, 27(6): 1069–1090.

Assenza, T.; Delli Gatti, D.; and Grazzini, J. 2015. Emergent dynamics of a macroeconomic agent based model with capital and credit. *Journal of Economic Dynamics and Control*, 50: 5–28.

Atashbar, T.; and Shi, R. A. 2023. *AI and macroeconomic modeling: Deep reinforcement learning in an RBC model*. International Monetary Fund.

Axtell, R. L.; and Farmer, J. D. 2022. Agent-based modeling in economics and finance: Past, present, and future. *Journal of Economic Literature*, 1–101.

Bookstaber, R. 2017. Agent-based models for financial crises. *Annual Review of Financial Economics*, 9(1): 85–100.

Brusatin, S.; Padoan, T.; Coletta, A.; Gatti, D. D.; and Glielmo, A. 2024. Simulating the economic impact of rationality through reinforcement learning and agent-based modelling. *arXiv preprint arXiv:2405.02161*.

Chen, M.; Joseph, A.; Kumhof, M.; Pan, X.; and Zhou, X. 2021. Deep reinforcement learning in a monetary model. *arXiv preprint arXiv:2104.09368*.

Dawid, H.; Fagiolo, G.; et al. 2008. Agent-based models for economic policy design: Introduction to the special issue. *Journal of Economic Behavior & Organization*, 67(2): 351–354.

Deissenberg, C.; Van Der Hoog, S.; and Dawid, H. 2008. EURACE: A massively parallel agent-based model of the European economy. *Applied mathematics and computation*, 204(2): 541–552.

Dong, J.; Dwarakanath, K.; and Vyetrenko, S. 2024. Tax Credits and Household Behavior: The Roles of Myopic Decision-Making and Liquidity in a Simulated Economy. *arXiv preprint arXiv:2408.10391*.

Gatti, D.; Gaffeo, E.; Gallegati, M.; Giulioni, G.; and Palestrini, A. 2008. *Emergent macroeconomics: an agent-based approach to business fluctuations*. Springer Science & Business Media.

Kermack, W. O.; and McKendrick, A. G. 1927. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772): 700–721.

Koster, R.; Balaguer, J.; Tacchetti, A.; Weinstein, A.; Zhu, T.; Hauser, O.; Williams, D.; Campbell-Gillingham, L.; Thacker, P.; Botvinick, M.; et al. 2022. Human-centred mechanism design with Democratic AI. *Nature Human Behaviour*, 6(10): 1398–1407.

Lempert, R. 2002. Agent-based modeling as organizational and public policy simulators. *Proceedings of the national academy of sciences*, 99(suppl_3): 7195–7196.

Mi, Q.; Xia, S.; Song, Y.; Zhang, H.; Zhu, S.; and Wang, J. 2023. Taxai: A dynamic economic simulator and benchmark for multi-agent reinforcement learning. *arXiv preprint arXiv:2309.16307*.

Olmez, S.; Heppenstall, A.; Ge, J.; Elsenbroich, C.; and Birks, D. 2024. Mitigating housing market shocks: an agent-based reinforcement learning approach with implications for real-time decision support. *Journal of Simulation*, 1–19.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *International conference on machine learning*, 2817–2826. PMLR.

Raberto, M.; Teglio, A.; and Cincotti, S. 2012. Debt, deleveraging and business cycles: An agent-based perspective. *Economics*, 6(1): 20120027.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268): 1–8.

Robbins, H. 1952. Some aspects of the sequential design of experiments.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Wiese, S.; Kaszowska-Mojsa, J.; Dyer, J.; Moran, J.; Pangallo, M.; Lafond, F.; Muellbauer, J.; Calinescu, A.; and Farmer, J. D. 2024. Forecasting Macroeconomic Dynamics using a Calibrated Data-Driven Agent-based Model. *arXiv preprint arXiv:2409.18760*.

Yao, Z.; Li, Z.; Thomas, M.; and Florescu, I. 2024. Reinforcement Learning in Agent-Based Market Simulation: Unveiling Realistic Stylized Facts and Behavior. *arXiv preprint arXiv:2403.19781*.

Zheng, S.; Trott, A.; Srinivasa, S.; Parkes, D. C.; and Socher, R. 2022. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science advances*, 8(18): eabk2607.

## Appendix

### 1. Further Plots for Single Fixed Parameter Shifts in SIRS

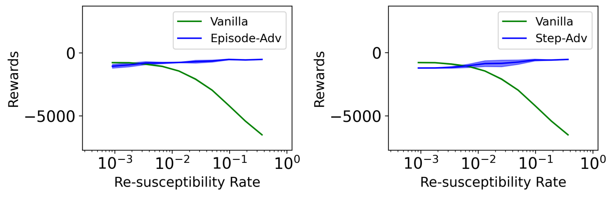


Figure 6: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the SIRS model. Adversarially trained RL agents exhibit enhanced robustness to single-parameter shifts, achieving better worst-case rewards. During training, adversary controlled $\gamma$ within $(0.00091, 0.37)$.

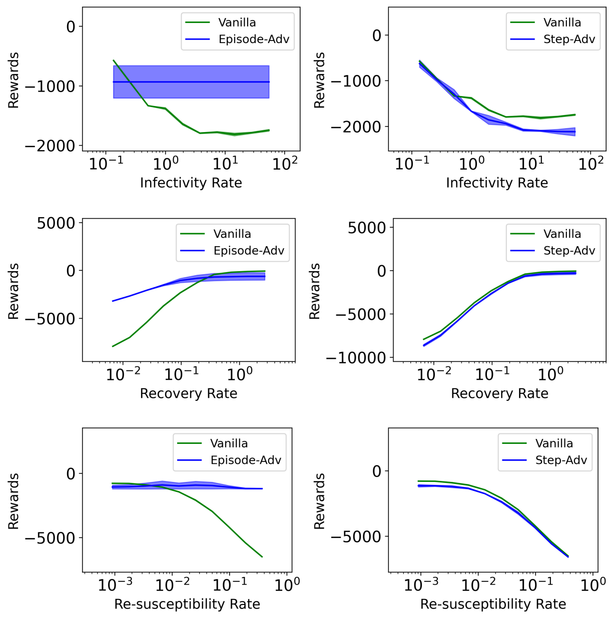


Figure 7: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the SIRS model. Adversarially trained RL agents exhibit enhanced robustness to single-parameter shifts, achieving better worst-case rewards. During training, adversary controlled $\alpha$ within $(0.22, 90)$.

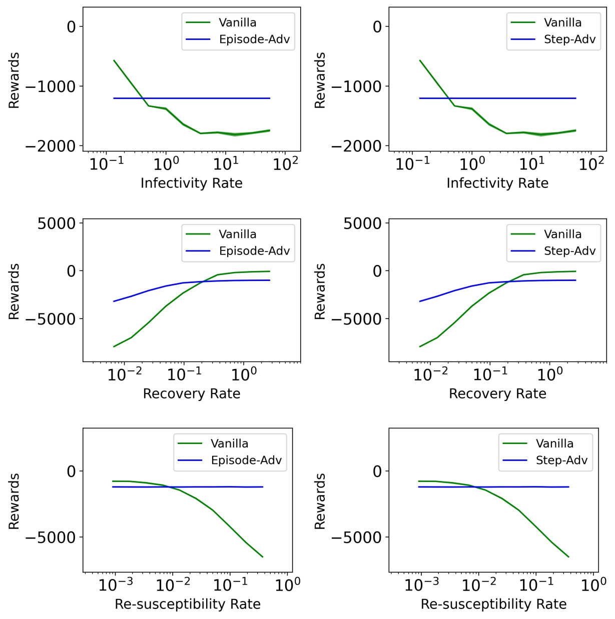


Figure 8: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the SIRS model. Adversarially trained RL agents exhibit enhanced robustness to single-parameter shifts, achieving better worst-case rewards. During training, adversary controlled $\beta$ within $(0.0067, 2.71)$.

### 2. Further Plots for Single Fixed Parameter Shifts in M-ABM

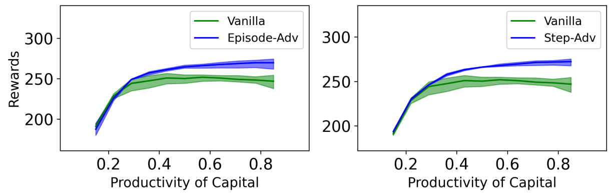


Figure 9: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the M-ABM model. Adversarially trained RL agents exhibit enhanced robustness to double-parameter shifts across the board. During training, adversary controlled Memory Parameter within $(0.5, 0.99)$.

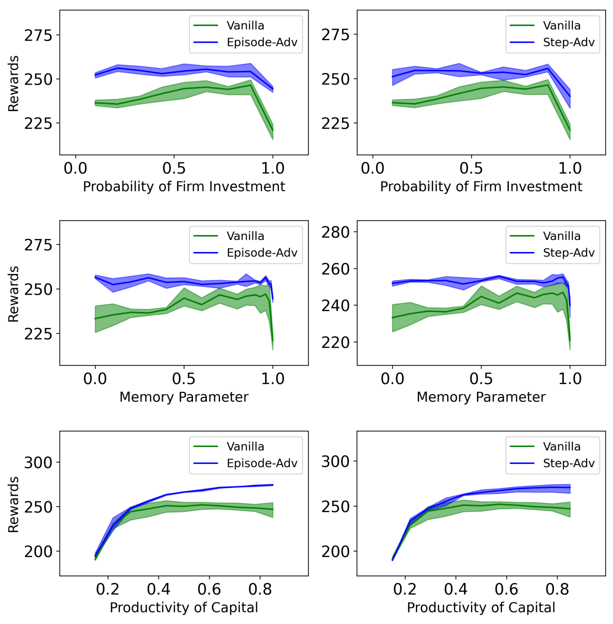


Figure 10: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the M-ABM model. Adversarially trained RL agents exhibit enhanced robustness to double-parameter shifts across the board. During training, adversary controlled Ratio of Consumption to Wealth within $(0.1, 0.9)$.

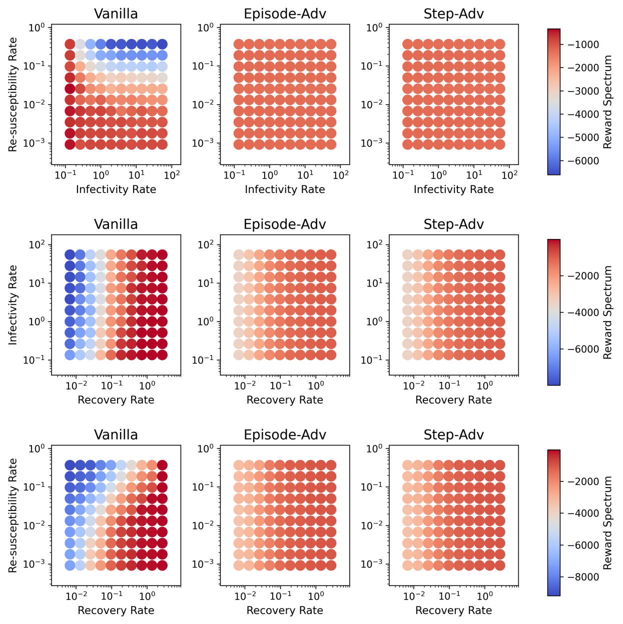


Figure 12: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the SIRS model. Adversarially trained RL agents exhibit enhanced robustness to double fixed-parameter shifts, achieving better worst-case rewards. During training, adversary controlled $\beta$ within $(0.0067, 2.71)$.

## 4. Further Plots for Two Simultaneous Fixed Parameter Shifts in M-ABM

## 3. Further Plots for Two Simultaneous Fixed Parameter Shifts in SIRS

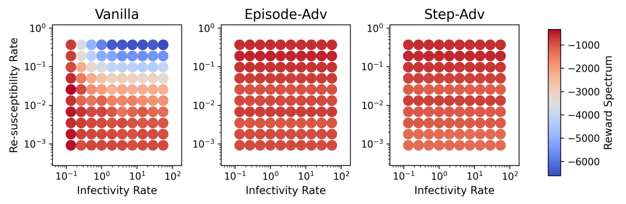


Figure 11: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the SIRS model. Adversarially trained RL agents exhibit enhanced robustness to double-parameter shifts, achieving better worst-case rewards. During training, adversary controlled $\gamma$ within $(0.00091, 0.37)$.

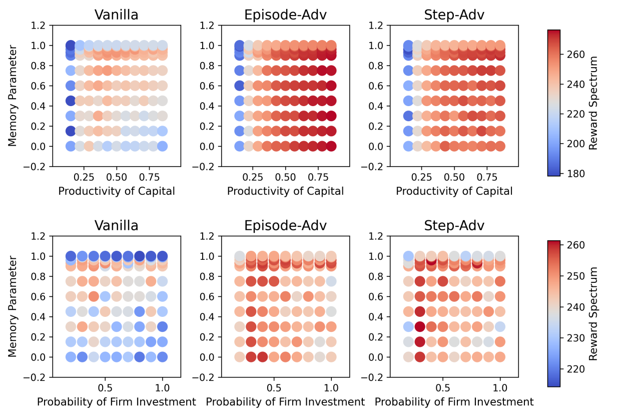
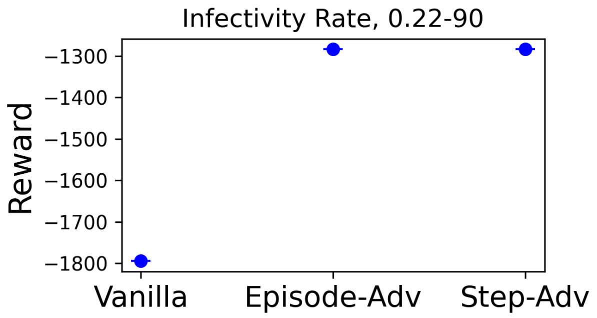


Figure 13: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the M-ABM model. Adversarially trained RL agents exhibit enhanced robustness to single-parameter shifts across the board. During training, adversary controlled Ratio of Consumption to Wealth within $(0.1, 0.9)$.

## 5. Further Plots for Adversarial Shifts in SIRS



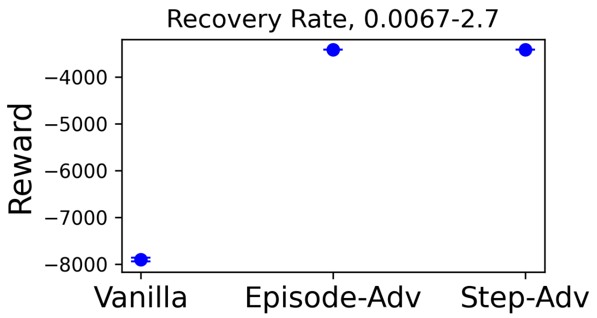
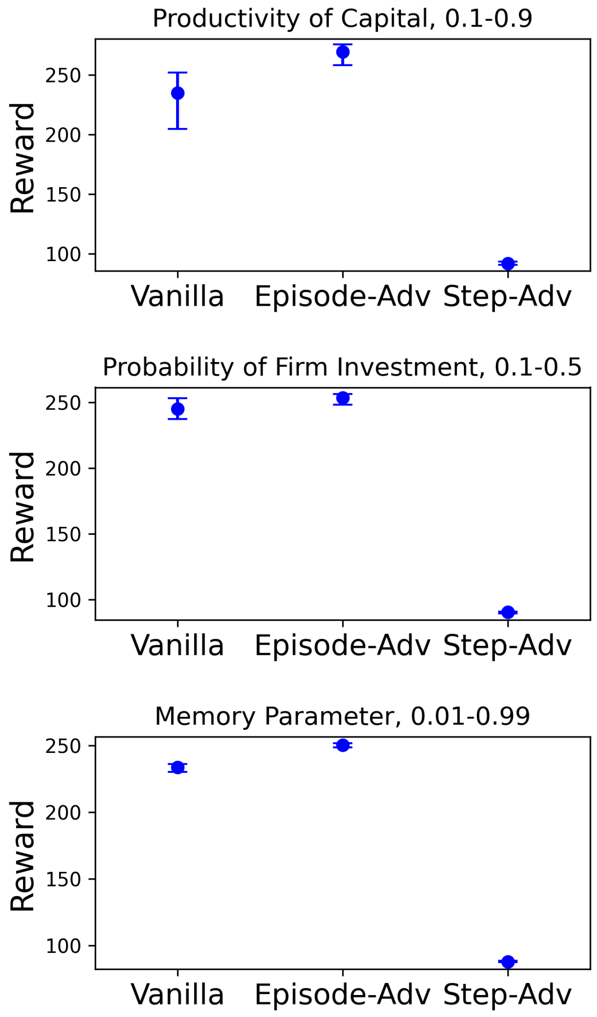


Figure 14: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different parameter settings in the SIRS model. Adversarially trained RL agents exhibit enhanced robustness to adversarial shifts, achieving better worst-case rewards. During training, adversary controlled $\beta$ within $(0.0067, 2.71)$.

## 6. Further Plots for Adversarial Parameter Shifts in M-ABM



b

Figure 15: Rewards observed by testing Vanilla (green) and adversarially trained (blue) policy agents on different adversarial settings in the M-ABM model. Adversarially trained RL agents exhibit enhanced robustness to adversarial parameter shifts across the board. During training, adversary controlled Ratio of Consumption to Wealth within $(0.1, 0.9)$.

## 7. Hyperparameters and Seeds

| | |
|---|---|
| Simulation Length (Episode Length) | $T = 100$ steps |
| Total Population Size | $N = 100$ |
| Initial Infected Fraction | $i_0 = 0.3$ |
| Infection Rate | $\alpha = 1.0$ |
| Recovery Rate | $\beta = -2.0$ |
| Re-susceptibility Rate | $\gamma = -4.0$ |
| Lockdown Cost | $C = 10$ |

Table 1: SIRS Simulation Hyperparameters

| | |
|---|---|
| Simulation Length (Episode Length) | $T = 300$ steps |
| Number of Workers | $W = 250$ |
| Number of Consumption Firms | $F = 30$ |
| Number of Capital Firms | $N = 6$ |
| Burn-in Time | $t_{\text{burn-in}} = 30$ steps |
| Previous Time Steps in Observations | $t_{\text{prev}} = 2$ |
| Reward Function Balance | $B = 100.0$ |

Table 2: M-ABM Simulation Hyperparameters

| | |
|---|---|
| Number of training updates | $N_{\text{iter}} = 4000$ iterations |
| Number of episodes per update | $k = 2$ episodes |
| Batch Size | 200 |
| Clip Range | 0.2 |
| Entropy Coefficient | 0.01 |

Table 3: SIRS Training Hyperparameters

| | |
|---|---|
| Number of training updates | $N_{\text{iter}} = 500$ iterations |
| Number of episodes per update | $k = 2$ episodes |
| Batch Size | 50 |
| Clip Range | 0.2 |
| Entropy Coefficient | 0.01 |

Table 4: M-ABM Training Hyperparameters

| | |
|---|---|
| Training | 42, 43 |
| Testing | $\{1, 2, \ldots, 10\}$ |

Table 5: SIRS Random Seeds

| | |
|---|---|
| Training | 42, 43, 44 |
| Testing | $\{1, 2, \ldots, 10\}$ |

Table 6: M-ABM Random Seeds

## 8. Model Complexity Impacts Policy Robustness

We observed distinct types of robustness in the SIRS and M-ABM models, highlighting the impact of model complexity on the outcomes of adversarial training. In the SIRS model, the adversarially trained policies demonstrated worst-case robustness, meaning that they were particularly well-suited to handle the most adverse conditions. In contrast, the M-ABM model exhibited improved policy performance more consistently across a wide range of scenarios, not just in the worst-case situations.

This difference in robustness can likely be attributed to the varying complexities of the two models. In the M-ABM model, the training process appears to benefit from adversarial training across the board, possibly because the adversary exposes the policy to scenarios that, while unlikely to occur in real life, serve to improve overall policy optimality. The complexity and non-linearity of the M-ABM system make it difficult for the adversary to consistently identify and exploit the worst-case shifts. As a result, the adversary tries different strategies during training, inadvertently helping the policy become more robust across a broader spectrum of situations, rather than just focusing on worst-case scenarios.

On the other hand, the SIRS model, with its simpler dynamics, allows the adversary to reliably identify the worst-case parameter shifts during training. This forces the government's policy to optimize specifically for those worst-case conditions. The adversary's ability to consistently find and exploit the worst-case scenarios means that the policy becomes specialized in responding to those particular shifts, resulting in a form of robustness that is more narrowly focused on worst-case performance, rather than general robustness.

## 9. RL Training Graphs

As we can see in Figure 16, adversarially trained policies take less time to reach the optimal policy and stay there in a more stable way, regardless of the algorithm used. Further, even for the non-shifted parameters (on which Vanilla policies are optimized), adversarially trained policies perform slightly better.

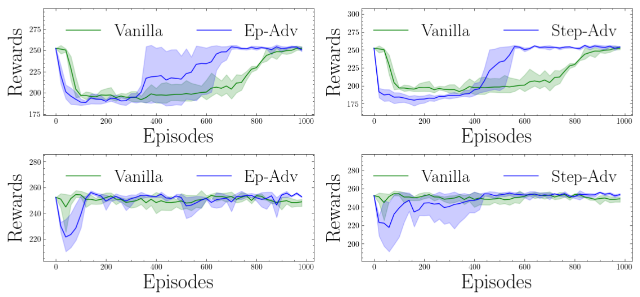


Figure 16: Rewards observed by testing Vanilla (green and adversarially trained (blue) policy agents during different levels of training. The upper two graphs show policy agents trained by the Advantage Actor Critic (A2C) algorithm and the lower two show agents trained by the Proximal Policy Optimization (PPO) algorithm.

If we consider how well the policies perform at different levels of training when we test them on shifted environments, we find that adversarially trained policies fare even better, relative to the Vanilla policies. This is regardless of whether the shifts are similar to the shifts the adversarially trained policies observed during training (in Figure 17) or different (in Figure 18).

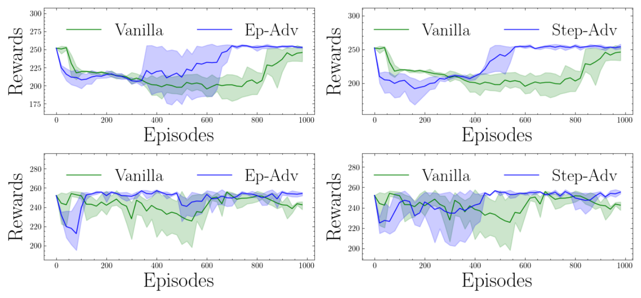


Figure 17: Rewards observed by testing Vanilla (green and adversarially trained (blue) policy agents during different levels of training on an environment where the memory parameter was set to 0.6 instead of 0.96. The upper two graphs show policy agents trained by the Advantage Actor Critic (A2C) algorithm and the lower two show agents trained by the Proximal Policy Optimization (PPO) algorithm.

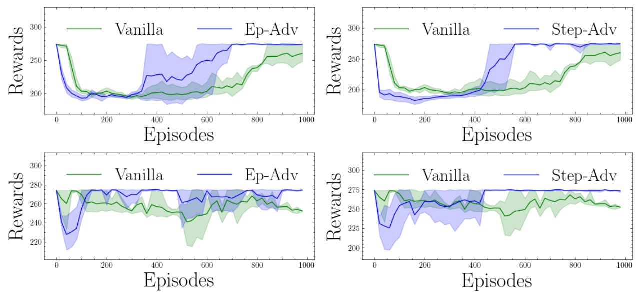


Figure 18: Rewards observed by testing Vanilla (green and adversarially trained (blue) policy agents during different levels of training on an environment where the productivity of capital was set to 0.85 instead of 0.33. The upper two graphs show policy agents trained by the Advantage Actor Critic (A2C) algorithm and the lower two show agents trained by the Proximal Policy Optimization (PPO) algorithm.

## 10. Training details

We use the Stable-Baselines3 (Raffin et al. 2021) implementation of the Proximal Policy Optimization algorithm (Schulman et al. 2017) to train the policies[1]. The episodes are of fixed length: in the SIRS model, they are of 100 steps each; and in the M-ABM model, of 300 steps. We provide additional hyperparameters about the simulation and training as well as the seeds/initializations used for training in Appendix 7.

[1]The code for the experiments is available at https://github.com/akash9702/robust-rl-policy-abm.