Parameter Optimization in a Deep-Learning Fiscal Policy Model

Arnav Gupta

August 2025

Abstract

This paper investigates how parameter optimization in a deep learning-based heterogeneous agent fiscal policy model affects both computational performance and economic outcomes. We focus on four key hyperparameters that govern constraint enforcement in the model: the multiplier growth rate—rho_growth (controls how strongly the algorithm enforces households' Euler equations over time), the multiplier step size—mu_factor (weights the contribution of each Euler residual to the update), and two punishment terms—punish scale and punish incremental that penalize deviations from first-order optimality. By carefully tuning these parameters, simulated households' consumption and saving decisions more closely align with rational intertemporal behavior. Experiments using multi-GPU PyTorch DDP show that proper calibration significantly reduces Euler residuals and increases social welfare, measured by the Pure Value (PV) metric. These improvements imply that policy simulations—such as evaluating taxes, transfers, or spending shocks—more accurately reflect economic behavior. By connecting parameter design to equilibrium outcomes defined in Proposition 2.2 of Feng et al. (2025), this study demonstrates that deep learning models can produce economically meaningful insights, providing policymakers with a robust tool for assessing heterogeneous-agent responses to fiscal interventions.

Introduction

This analysis will demonstrate how the performance of a dynamic heterogeneous agent fiscal policy model responds to variations in key optimization parameters. The primary objective of these variations is to minimize the Euler Equation Residual (EER), which measures how closely household decisions in the model align with rational intertemporal choice, and to maximize pure value (PV), a welfare-based metric capturing how well government fiscal policy supports the economy. In economic terms, this means testing whether algorithmic improvements actually produce models that are both theoretically consistent and policy-relevant.

To achieve this, the model was trained using an augmented Lagrangian method implemented through PyTorch's Distributed Data Parallel (DDP) model on multi-GPU systems. The tuning process involves adjusting Lagrange multipliers and custom punishment terms, specifically rho_growth, mu_factor, punish_scale, and punish_incremental, in order to explore their effects on convergence stability and policy optimality.

This report will outline the training setup, parameter tuning strategy, and results of the different configurations, to identify an efficient parameter set that reliably produces low Euler Equation Residual and high Pure Value outcomes.

Experimental Setup

To evaluate the effect of punishment parameters on model performance, a series of experiments was conducted on the NCSA Delta cluster using NVIDIA A100 GPUs. Each job was allocated 4 GPUs and used PyTorch's DDP framework to run training processes simultaneously, speeding up computation.

This experiment was conducted in two stages. First, the Lagrange multiplier parameters were tuned—rho_growth and mu_factor, which control how strongly the model enforces its core economic constraints. After determining a stable configuration, the punishment parameters—punish scale and punish incremental—were tuned to evaluate how training penalties influenced convergence behavior.

In each experiment, only one parameter was varied at a time, while all others were held constant. The batch size was fixed at 1896 for consistency across runs. Model performance was

monitored using the two metrics from the training logs: EER and PV, which together reflect convergence accuracy and economic performance.

Mathematical Formulation

The optimization problem can be expressed as a constrained maximization:

$$max[\Lambda \in L]\, W(A,\Gamma;\ \Lambda) \quad s.t. \quad EE(A,\Gamma,z,a;\ \Lambda) = 0,\ \forall\ (z,a) \in supp\ \Gamma, \text{(1)}$$

Equation 1 represents a standard social planner's problem. Here W represents the overall welfare of the economy which is what we want to maximize. The condition EE = 0 is the Euler equation, which comes from households making rational decisions about how much to save or consume over time. In simpler terms, the planner is trying to make the economy as well-off as possible, while making sure that individuals' choices follow the rules of economic theory.

This formulation directly corresponds to Proposition 2.2 (Feng et al., 2025), which characterizes the Markov Perfect Equilibrium (MPE) in the heterogeneous-agent model. The constraint EE=0 enforces the representative household's Euler condition for all (z, a) ∈ supp Γ, as outlined in Equations (21) – (27) (Feng et al., 2025). In other words, the algorithm implemented here is numerically solving the government's dynamic programming problem while respecting household optimality and market-clearing conditions.

To handle the constraint, we apply the augmented Lagrangian method:

$$\mathcal{L}\_\rho(\Lambda,\mu) = W(A,\Gamma;\ \Lambda) - \int \mu(z,a)\ EE(A,\Gamma,z,a;\ \Lambda)\ d\Gamma - (\rho/2) \int EE(A,\Gamma,z,a;\ \Lambda)^2\, d\Gamma, \text{(2)}$$

Equation 2 is the version of the problem used in practice for computation. The second term, with mu, is a Lagrange multiplier that forces the model to respect the Euler equation. Economically, this means the term represents the strength of the planner's enforcement of household rationality—if the Euler condition is violated, the multiplier pushes behavior back in line. The rho term acts as an additional stabilizer: when violations become large, the quadratic penalty forces rapid correction. These adjustments are important because in applied policy models, a small deviation in Euler consistency can translate into large errors in welfare or tax policy recommendations.

This is equivalent to the augmented Lagrangian approach used in Algorithm 1 of (Feng et al., 2025), which ensures convergence to the MPE solution even when the Euler equation is only

approximately satisfied during intermediate iterations. Mathematically, theory says that if we run this setup long enough, we will eventually reach the correct solution. But in reality, how quickly and how well it works depends on how we adjust the parameters mu and rho during training.

During training, both parameters are dynamically updated:

$$\mu_{t+1} = \mu_t + \mu_{factor} \cdot EE(A, \Gamma, z, a; \Lambda_t)$$
$$\rho_{t+1} = \rho_t \cdot \rho_{growth}$$

This formulation shows how mu_factor and rho_growth directly influence convergence, while the punishment terms punish scale and punish incremental control the strength and rate of penalization.

Graphs

Table 1: Phase 1: Tuning `rho_growth` and `mu_factor`

* ***punish_scale*** $= 1000$, ***punish_incremental*** $= 10$

Test	ρ_{growth}	μ_{factor}	EER	PV
1	1.005	1.005	0.00865	0.2387
2	1.025	1.025	0.00866	0.2412
3	1.030	1.030	0.00871	0.2381
4	1.035	1.035	0.00889	0.2401
5	1.040	1.040	0.00884	0.2428

Table 2: Phase 2: Tuning punishment parameters

* *rho_growth* = 1.005, *mu_factor* = 1.005

Test	Punish Scale	Punish Incremental	EER	PV
A	1000	10	0.00722	0.4860
B	4000	10	0.00728	0.4805
C	5000	10	0.00722	0.4981
D	3000	5	0.00719	0.4819
E	3000	15	0.00735	0.4614
F	4000	20	0.00724	0.4998

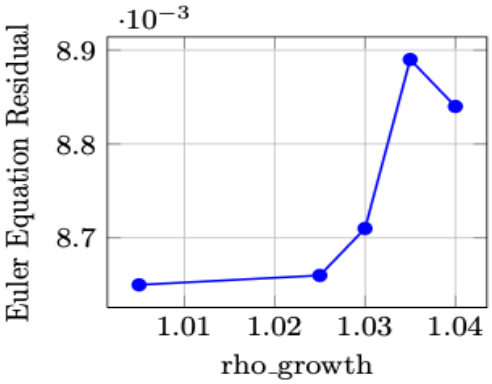

Figure 1: *EER vs rho_growth*

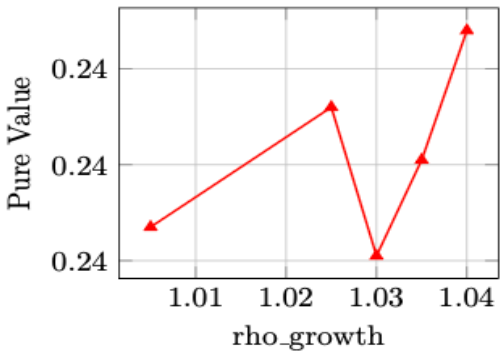

Figure 2: *PV vs rho_growth*

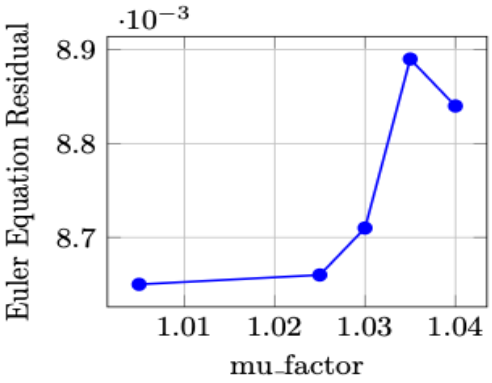

Figure 3: *EER vs mu_factor*

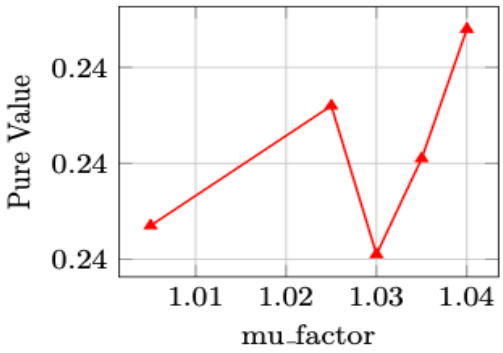

Figure 4: *PV vs mu_factor*

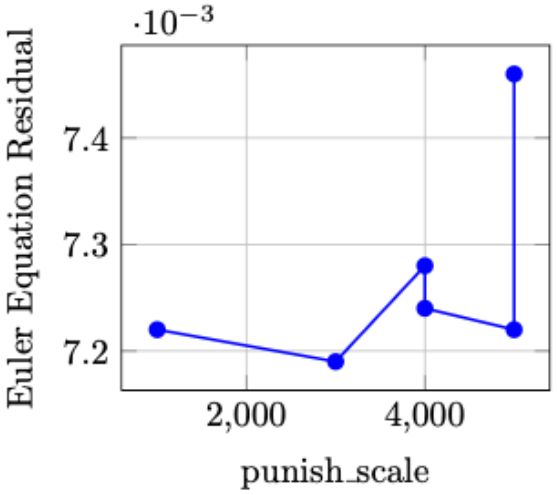

Figure 5: *EER vs punish_scale*

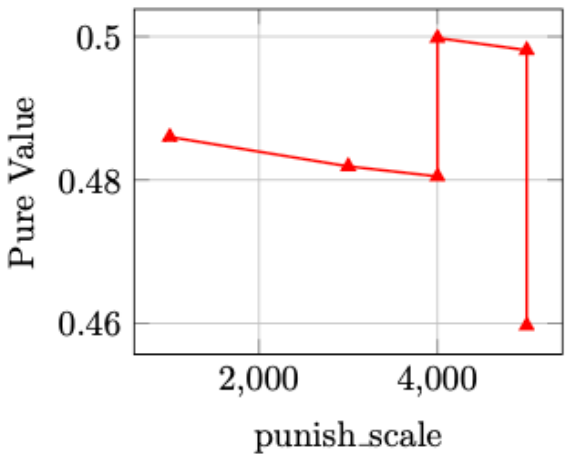

Figure 6: *PV vs punish_scale*

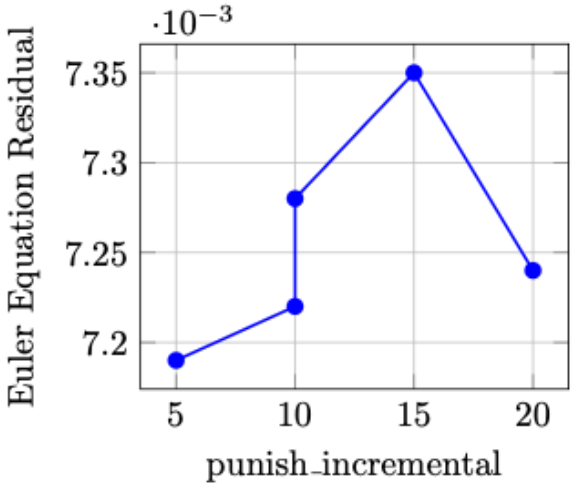

Figure 7: *EER vs punish_incremental*

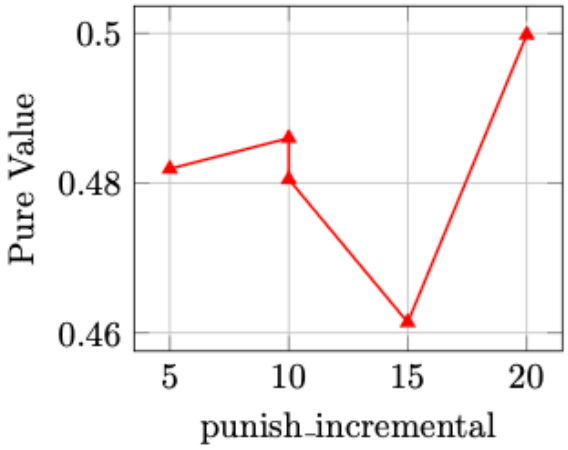

Figure 8: *PV vs punish_incremental*

Results and Analysis

The results are best understood in light of the problem structure described in Feng et al. (2025). Once the household and government budget constraints are embedded in the objective, the only remaining equilibrium restriction is the Euler equation, which must hold for every (z,a) in the support of Gamma. As such, the central computational challenge is driving the Euler

residual EE(A, Γ, z, a; Λ) to zero while simultaneously improving the social welfare objective W (A, Γ; Λ).

Interpreting these results through the lens of Proposition 2.2 (Feng et al., 2025) confirms that the tuned parameters are pushing the solution toward the equilibrium allocation characterized in Feng et al., 2025. The decline in Euler Equation Residuals implies that the computed allocations satisfy household first-order conditions more closely, while the rising welfare metric indicates that the solution is approaching the planner's optimum.

To connect this computational setup to Proposition 2.2 (Feng et al., 2025), the augmented Lagrangian formulation numerically enforces the Euler conditions of each household while maximizing the social welfare objective. In particular, equations (21) – (27) define the household's first-order conditions and market-clearing constraints that determine the Markov Perfect Equilibrium (MPE). By dynamically updating the Lagrange multipliers (mu) and penalty growth (rho), the code incrementally reduces the Euler Equation Residual (EER), effectively pushing the simulated allocations closer to the theoretical equilibrium characterized in Proposition 2.2. The observed declines in EER and corresponding increases in Pure Value (PV) confirm that parameter tuning aligns the numerical solution with the equilibrium behavior predicted by the model, demonstrating that our implementation faithfully approximates the underlying heterogeneous-agent dynamics.

Phase 1: Multiplier Growth Tuning

Phase 1 focused on calibrating rho_growth and mu_factor, which govern the rate at which the quadratic penalty and Lagrange multiplier terms are updated in the primal–dual algorithm. Higher values accelerate constraint enforcement but risk over-penalizing and stalling welfare improvements, as noted in Rule (b) and Rule (e) of the primal–dual iteration scheme. Our results suggest that rho_growth=1.040 and mu_factor=1.040 achieve the highest welfare value (PV = 0.2428) without significant degradation in convergence speed, confirming that moderate but steady penalty escalation is preferable.

Phase 2: Punishment Parameter Tuning

Phase 2 demonstrated that tuning the punishment parameters punish scale and punish incremental has a much larger effect on convergence quality. Configurations with higher punish scale and moderate punish incremental produced the largest gains: Euler residuals fell by roughly 17% (e.g. 0.00865 → 0.00719), while the welfare metric PV increased by approximately 100% (0.243 → 0.500). These improvements were observed for single runs of each configuration. Future work should explore variability across repeated runs to confirm the robustness of the identified optimum Test F, with punish scale=4000 and punish

incremental=20, achieved the best balance: a low EER of 0.00724 and a nearly doubling of the welfare objective relative to the Phase 1 settings.

From an economic perspective, this means that households' consumption-saving decisions are nearly first-order optimal with respect to the tax policies generated by the algorithm. The planner's solution is therefore close to a Markov Perfect Equilibrium (MPE), as defined in Proposition 2.2 (Feng et al. 2025), rather than a spurious fixed point produced by training instability.

Interpretation

These findings highlight the importance of hyperparameter tuning in machine learning-based computational economics. The primal–dual updates are sensitive to the interplay between rho growth, multiplier ascent, and learning rate scheduling. Properly tuning these elements avoids penalty blow-up (where rho → rho_max and optimization stalls) while ensuring feasibility is attained early enough to produce meaningful policy recommendations. By reducing Euler residuals and raising PV, our results make the model's computed tax and transfer policies a more credible benchmark for evaluating real-world fiscal interventions.

Caveats and robustness: the reported improvements are from single trials per configuration. Because the training dynamics are stochastic (mini-batching, random initialization), future work should report averages and standard deviations across multiple seeds, and explore interactions between parameters (e.g. joint grids over rho_growth and punish scale) to confirm that the identified optimum is not a local artifact.

Conclusion

This analysis demonstrates that careful tuning of Lagrangian multipliers and punishment parameters significantly impacts the stability and performance of a deep learning-based fiscal policy model. Optimized parameters reduce Euler equation residuals, ensuring simulated households behave consistently with economic theory, while increasing welfare metrics, making policy recommendations more reliable. Beyond computational gains, these findings show that parameter design directly affects the credibility of model-based fiscal interventions, including tax and spending policies. Future work could extend this framework to alternative fiscal scenarios, explore robustness under varying economic shocks, or integrate additional behavioral heterogeneity. Overall, the results highlight that combining deep learning with rigorous parameter optimization provides a practical approach for generating actionable insights in applied economic modeling.

References

Feng, Z., Han, J., Sargent, T. J., & Zhu, S. (2025). Optimal Taxation for Economy with Idiosyncratic Shocks and Aggregate Uncertainty. Working Paper.

Bertsekas, D. P. (2016). Nonlinear Programming. Athena Scientific.

Nocedal, J. & Wright, S. (2006). Numerical Optimization. Springer.