The GNN as a Low-Pass Filter: A Spectral Perspective on Achieving Stability in Neural PDE Solvers

Peilin Rao*[1]

[1]Department of Economics, University of California, Los Angeles, US

Sept 4, 2025

Abstract

The choice of architecture in Graph Machine Learning (GML) presents a trade-off between expressive power and implicit regularization. We investigate this using the challenge of solving high-dimensional Hamilton-Jacobi-Bellman (HJB) partial differential equations. Our experiments show that while flexible networks succeed on problems with smooth solutions, they fail catastrophically on non-smooth problems where the architectural bias of a Graph Neural Network (GNN) is essential. We explain the GNN's success through its role as a spectral low-pass filter, which provides implicit Lipschitz regularization crucial for achieving stable and generalizable solutions. This work establishes a principled framework for selecting GML architectures based on a problem's mathematical structure.

*jackrao@g.ucla.edu

1 Introduction

A central challenge in Graph Machine Learning (GML) is selecting an architecture with the appropriate inductive bias. The spectrum of choices ranges from unstructured, universal approximators like Feed-Forward Networks (FFNs) (Hornik et al., 1989) to highly structured models like Graph Neural Networks (GNNs). This choice dictates a fundamental trade-off between expressive power and implicit regularization, directly impacting a model's ability to generalize. When should an architectural constraint be viewed as a helpful regularizer versus a detrimental "straitjacket"?

To investigate this question, we turn to the challenging scientific domain of solving high-dimensional partial differential equations (PDEs), specifically the Hamilton-Jacobi-Bellman (HJB) equation, which arises in optimal control. Solving the HJB equation is hampered by the "curse of dimensionality" (Bellman, 1957), and neural network solvers represent a significant breakthrough (Han et al., 2018). This context is an ideal testbed for GML, as HJB solutions can range from smooth and global to highly localized and non-smooth.

This paper demonstrates that the optimal architecture for this task is predictably determined by the mathematical structure of the PDE's solution. We develop our argument through a "three-act" empirical narrative, showing that while an FFN is superior for smooth problems, it fails catastrophically on non-smooth problems where the structural bias of a GNN is essential for learning stable, physically-plausible solutions.

We provide a formal theoretical foundation for the GNN's success by leveraging its established understanding as a spectral low-pass filter (Kipf and Welling, 2017; Wu et al., 2019). Our contribution is to explicitly link this spectral property to the implicit Lipschitz regularization required for solving HJB equations, thereby explaining the mechanism behind the observed numerical stability. Section 2 formulates the problem, Section 3 presents our empirical results, Section 4 details the GNN's spectral bias, and Section 5 synthesizes these findings into a framework.

2 Problem Formulation

2.1 The Hamilton-Jacobi-Bellman Equation for Multi-Agent Systems

We consider a system of N interacting agents whose joint state $\mathbf{x}_t \in \mathbb{R}^N$ evolves according to a stochastic differential equation (SDE):

$$d\mathbf{x}_t = \boldsymbol{\mu}(\mathbf{x}_t, \boldsymbol{\alpha}_t)dt + \boldsymbol{\Sigma}(\mathbf{x}_t)d\mathbf{W}_t, \tag{1}$$

where $\boldsymbol{\mu}$ is the drift, $\boldsymbol{\Sigma}$ is the volatility, $\boldsymbol{\alpha}_t \in \mathbb{R}^N$ is the control, and $\mathbf{W}_t$ is a Wiener process. The solution to the optimal control problem is the value function, $V(t, \mathbf{x})$, which must satisfy a Hamilton-Jacobi-Bellman (HJB) PDE. The specific form depends on whether the problem is cooperative (a single PDE) or non-cooperative (a system of coupled PDEs), as detailed in Appendix B. Our goal is to train a neural network $V_{\boldsymbol{\theta}}$ to parameterize the value function by minimizing the HJB equation's mean squared residual.

2.2 Solver Architectures and Inductive Biases

We investigate three neural architectures for parameterizing $V_{\boldsymbol{\theta}}$:

- **Feed-Forward Network (FFN):** A universal approximator with no structural priors. It treats the input $\mathbf{x}$ as a flat vector, relying on the implicit spectral bias of gradient descent for regularization (Rahaman et al., 2019).
- **Symmetric MLP (SymmetricMLP):** Imposes a strong feature bias via permutation equivariance (Zaheer et al., 2017). It processes per-agent features based on low-dimensional, hand-crafted statistics (e.g., agent state $\mathbf{x}_i$ and global mean $\bar{\mathbf{x}}$) using a shared MLP.

- **Graph Neural Network (GNN):** Imposes a structural bias by constraining information flow to a specified graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We use a Graph Convolutional Network (GCN) (Kipf and Welling, 2017), with layer-wise updates $\boldsymbol{H}^{(l+1)} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)})$, where $\hat{\boldsymbol{A}}$ is the symmetrically normalized adjacency matrix.

3 The Empirical Investigation: A Tale of Three Games

All experiments were conducted on a single NVIDIA 4070 GPU. We design a sequence of three experiments to deconstruct the trade-offs between these architectures. Full mathematical specifications for each are in Appendix B.

3.1 Act I: The Primacy of Feature Bias in a Mean-Field Game

Setup: We implement a linear-quadratic (LQ) mean-field game where interactions are all-to-all (a complete graph). The analytical solution for each agent's value function is a simple quadratic form of its deviation from the global mean (see Appendix B.2). This provides a clear test for the feature-engineered SymmetricMLP.

Result: As shown in Table 1, while the GNN achieves the lowest HJB loss, the most critical metric—True L2 Error against the known solution—shows the SymmetricMLP dramatically outperforming both alternatives by nearly an order of magnitude.

Insight: When a problem's structure can be distilled into low-dimensional sufficient statistics, explicit feature engineering is the most effective approach. The SymmetricMLP's direct access to the mean feature was a more powerful bias than the GNN's general structural constraint. The GNN's low HJB loss but high L2 error indicates it found a function that locally satisfied the PDE but missed the correct global structure.

3.2 Act II: The Peril of Mismatched Bias in a Sparse LQ Game

Setup: We pivot to a social planner LQR problem on a sparse ring graph. Agent dynamics depend only on their immediate neighbors, a structure seemingly aligned with the GNN's bias. However, the true value function $V(\mathbf{x}, t) = \frac{1}{2}\mathbf{x}^T \boldsymbol{P}_t \mathbf{x}$ is governed by a matrix Riccati equation whose solution, $\boldsymbol{P}_t$, is a dense, globally-structured matrix (see Appendix B.3).

Result: Against expectations, the GNN fails catastrophically. The winner, as shown in Table 1 and Figure 1, is the unstructured FFN, which learns a near-perfect approximation of the true quadratic value function.

Insight: The GNN's architectural bias, while aligned with the system's local dynamics, acted as a detrimental "straitjacket," preventing it from approximating the globally-structured, dense quadratic solution. The FFN, free from structural priors, easily discovered the smooth function. This reveals a critical lesson: the architectural bias must match the functional form of the solution, not just the system's one-step dynamics.

Table 1: Consolidated results for Act I and Act II. The primary success metric (True L2 Error) is bolded.

Experiment	Model	Parameters	Final HJB Loss	Final True L2 Error
Act I: Mean-Field Game	**SymmetricMLP**	4,481	0.0719	**0.0040**
	FFN	18,053	0.3079	0.0481
	GNN	3,417	*0.0248*	0.1529
Act II: Sparse LQ Game	**FFN**	**18,177**	**0.2016**	**0.0263**
	SymmetricMLP	21,793	0.6101	0.1424
	GNN	1,185	3.2144	3.7266

Note: In Act I, the GNN's low HJB loss is misleading, as the L2 error reveals it failed to find the true solution.

3.3 Act III: The Vindicating Power of Regularization

Setup: We introduce complexity by adding a localized, non-linear cost term, $\lambda(\mathbf{x}^0)^4$, to a single agent in the ring graph (the "Clog in the Ring" game, specified in Appendix B.4).

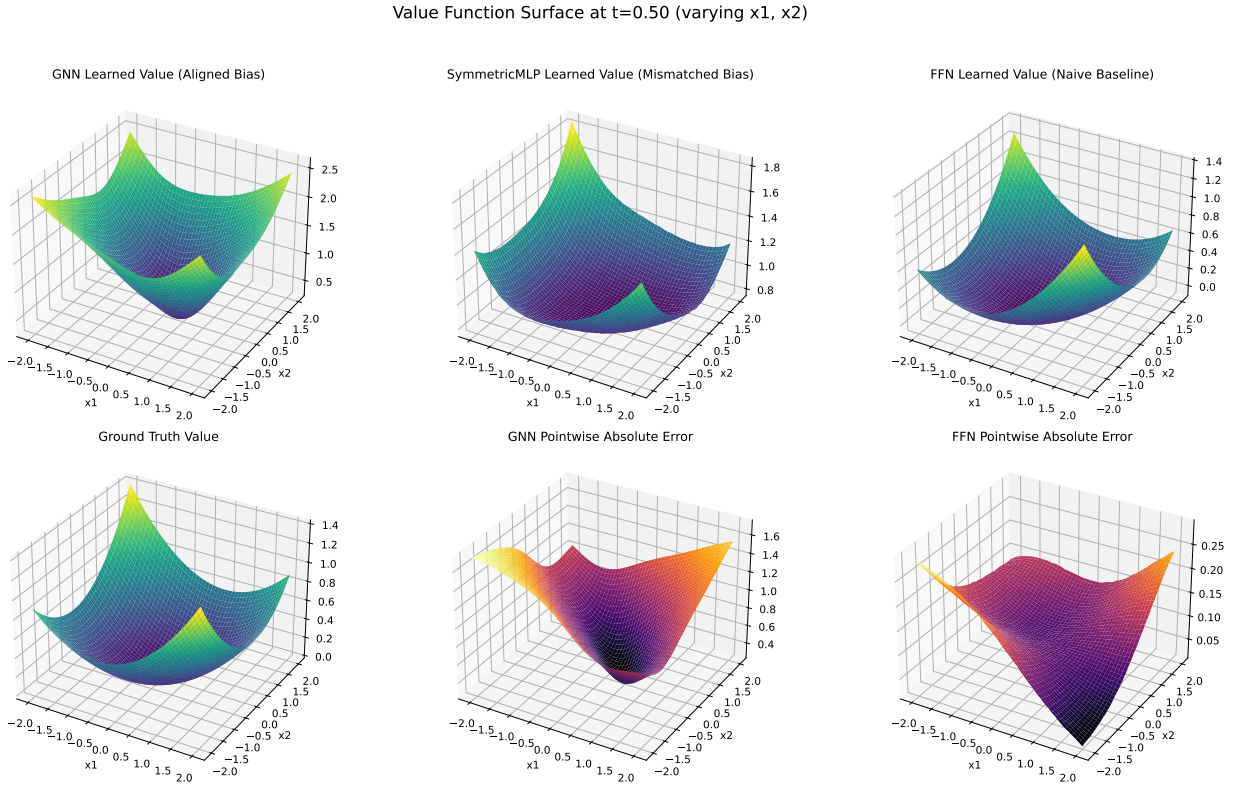


Figure 1: Learned value function surfaces for the Sparse LQ Game (Act II). The FFN (top right) learns a smooth surface that closely matches the Ground Truth (bottom left), while the GNN (top left) learns an incorrect function, visually demonstrating the failure of its mismatched bias.

This modification makes the true value function non-quadratic and non-smooth, with sharp local gradients. No analytic solution exists, so we evaluate models on test-set HJB residual, Monte Carlo rollout cost, and critically, the stability of the learned policy.

Result: The roles dramatically reverse. The FFN, previously the victor, now learns an erratic policy (Figure 2b) leading to catastrophic performance across all metrics (Table 2). More tellingly, a conflict emerges between the remaining models: both the Monte Carlo cost and the HJB residual suggest the SymmetricMLP is the superior model, as it achieves lower values than the GNN. This sets the stage for a deeper analysis to uncover the true nature of the learned solutions.

Insight: This experiment reveals the potential for standard metrics to be misleading and confirms our central thesis: for problems with complex, non-smooth solutions, implicit

architectural regularization is paramount. To demonstrate this, we perform a local stability analysis by computing the eigenvalues of the learned closed-loop system's Jacobian at the $\mathbf{x} = \mathbf{0}$ equilibrium (see Appendix B.4 for the formulation).

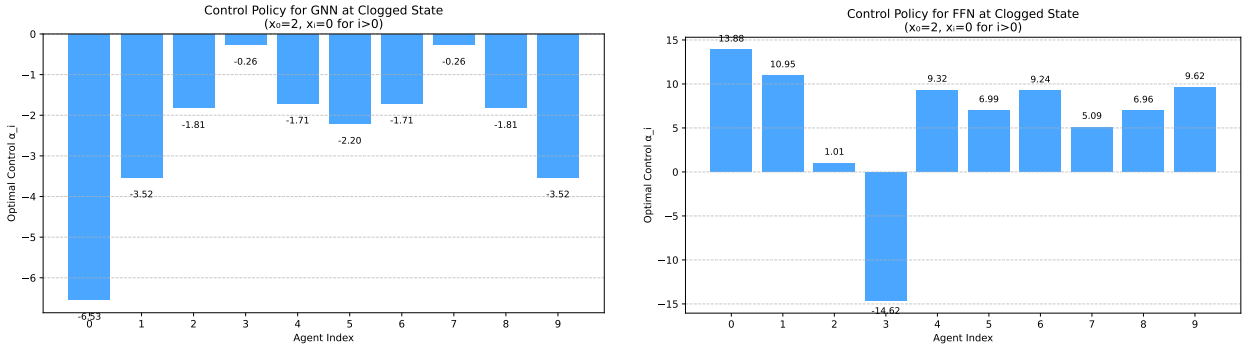
The results, summarized in Table 2, are definitive.

- The **FFN** learns a catastrophically unstable policy, with a maximum real eigenvalue of $+6.63$. Its failure is corroborated by its massive HJB residual and rollout cost. This instability is visualized in the bottom row of Figure 2, where the norm of the value function's gradient explodes.

- The **SymmetricMLP** appears optimal based on cost and HJB residual. However, stability analysis reveals its max real eigenvalue is $+3.50 \times 10^{-8}$. It achieves its low cost by reverting to a trivial, passive, and ultimately unstable policy that fails to actively control the non-linear dynamics.

- Only the **GNN** leverages its structural bias to learn a provably stable policy, with a max real eigenvalue of -1.66. It finds a valid, active control strategy, even at the expense of slightly higher cost and residual metrics.

The GNN's architectural constraint acts as an essential regularizer, preventing the instabilities that plague the FFN and guiding the solver to a physically-meaningful solution that the other, more misleading metrics would have missed. For complex dynamic systems, stability is the decisive measure of success, and the GNN's implicit regularization is the key to achieving it.



(a) GNN Policy

(b) FFN Policy

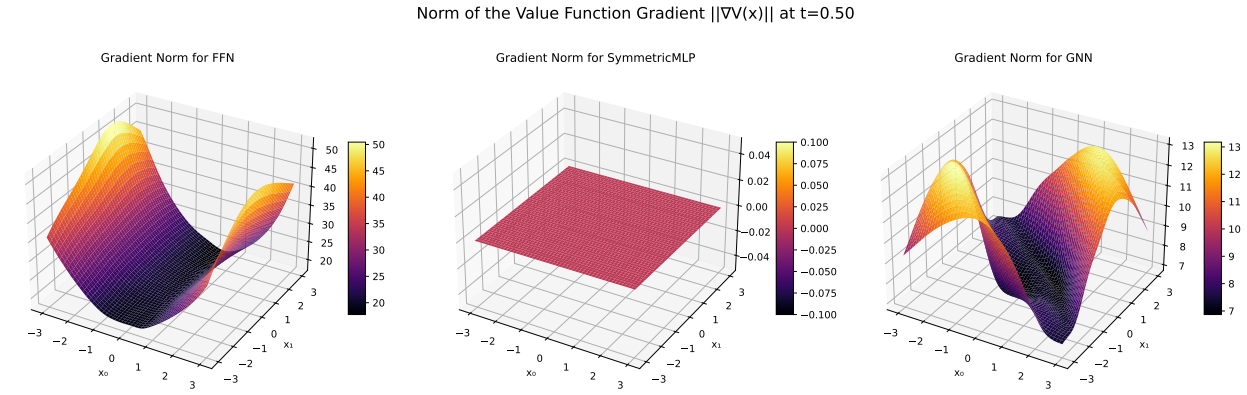


Figure 2: Analysis for the "Clog" game. **(Top Row)** Learned policies $\hat{\boldsymbol{\alpha}} = -\nabla_{\mathbf{x}} V$ at a test state. The GNN (a) learns a plausible policy while the FFN's (b) is nonsensical. **(Bottom Row)** Norm of the value function gradient, $||\nabla_{\mathbf{x}} V||_2$, across a 2D slice of the state space. The FFN's gradient exhibits exploding magnitudes, visually explaining its instability, while the GNN's gradient is smooth and controlled as a direct consequence of its implicit regularization.

Table 2: Final evaluation for the "Clog in the Ring" Game (Act III).

Model	Final Test HJB Residual	MC Mean Cost	MC Std Error	Max Real Eigenvalue	Stability
GNN	12,079.68	15.29	0.22	**-1.66**	**Stable**
SymmetricMLP	*11,020.24*	*13.08**	0.79	+3.50e-08	Unstable
FFN	40,400.03	18,242.22	289.26	+6.63	Unstable

Note: *The SymmetricMLP's low cost and HJB residual are misleading; they result from a passive policy that fails to actively control the system's non-linearities. Only the GNN learns an active and provably stable control strategy.

4 Theoretical Foundation: GNN as a Low-Pass Filter

We now formalize why the GNN succeeds in Act III, proving that the GCN layer functions as a spectral low-pass filter. This filtering promotes smoother functions, effectively regularizing the solution's Lipschitz constant and preventing the instabilities seen in the FFN. We use Graph Fourier Analysis, where eigenvectors $\boldsymbol{U}$ of the normalized adjacency matrix $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ form an orthonormal basis representing graph frequencies (Chung, 1997).

Theorem 1 (GCN as a Low-Pass Filter). *Let a GCN layer be defined by* $\boldsymbol{H}_{out} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}_{in}\boldsymbol{W})$. *The graph convolution operator* $\hat{\boldsymbol{A}}$ *decomposes any input signal into low-frequency and high-frequency components and has the following effect:*

1. *The low-frequency component (in the eigenspace of* $\lambda_1 = 1$*) is preserved.*
2. *The norm of any high-frequency component is strictly contracted by a factor of at least* $\lambda_{\max}^{\perp} = \max(|\lambda_2|, |\lambda_N|) < 1$.

Since the gain on the lowest frequency is 1 while the gain on all higher frequencies is strictly less than 1, the operator acts as a low-pass filter.

Proof Sketch. The proof follows by linearity. The low-frequency component is an eigenvector with eigenvalue 1 and is unchanged. Any high-frequency component is a linear combination of other eigenvectors, and its norm contracts because all other eigenvalues have magnitude strictly less than 1. A 1-Lipschitz activation preserves this attenuation. ■.

A full proof, including necessary preliminaries on graph spectral theory, is provided in Appendix C.

From Spectral Bias to Implicit Lipschitz Regularization

Theorem 1 explains the GNN's stability. The Lipschitz constant of a multi-layer network is bounded by the product of its layers' Lipschitz constants. For an FFN, this product can grow, permitting solutions with massive gradients, as seen in Act III. For a GNN, the graph

convolution operator $\hat{\boldsymbol{A}}$ is not merely non-expansive (Lipschitz constant of 1); it is an *active low-pass filter*. By systematically contracting high-frequency components at every layer, the GNN architecture penalizes high-frequency oscillations. This spectral bias regularizes the solution space, inherently limiting the Lipschitz constant of both $V_{\boldsymbol{\theta}}$ and its gradient $\nabla_{\mathbf{x}} V$, providing the implicit regularization that ensures a stable learned policy.

5 A Principled Framework for Solver Selection

Our results culminate in a framework for selecting a neural HJB solver based on the expected characteristics of the problem's solution, summarized in Table 3.

Table 3: A Principled Framework for Neural HJB Solver Selection

Problem Property	**Example Solution Character**	**Optimal Solver**	**Governing Principle**	**Primary Validation Criterion**
Mean-Field Interaction	Global, simple aggregates	**SymmetricMLP**	Feature Bias	True L2 Error vs. Analytic
Local, Simple Cost	Global, smooth, quadratic	**FFN**	Approximation Power	True L2 Error vs. Riccati
Local, Complex Cost	Non-smooth, sharp gradients	**GNN**	Structural Regularization	**Stable Closed-Loop Dynamics** (via max real eigenvalue), which serves as the decisive criterion over misleadingly low MC cost and HJB residuals.

When system interactions are governed by simple, low-dimensional statistics (Act I), an explicit feature bias (SymmetricMLP) is most effective. When interactions are local but the value function is globally smooth (Act II), the unconstrained approximation power of an FFN is paramount, as a GNN's structural bias can be overly restrictive. When complex, localized phenomena induce non-smoothness or sharp gradients (Act III), an FFN's flexibility becomes a liability. Here, the GNN's architectural bias is essential. As explained by our low-pass filter

theorem, this structural regularization induces a provably stable closed-loop system, guiding the solver to a physically-meaningful solution where more flexible models fail.

6 Conclusion

We established a principled framework for architectural selection in GML, grounded in the trade-off between approximation power and implicit regularization. Using HJB equations as a testbed, we showed that the optimal architecture depends on the mathematical properties of the target function. Our central contribution is to explain and apply the GNN's role as an implicit regularizer in a new scientific domain. By connecting the theory of GCNs as spectral low-pass filters to the practical challenge of solving dynamic systems, we established a concrete mechanism by which architectural bias enforces smoothness and confers stability. This insight is key to understanding GNN success on complex scientific problems where more flexible models fail and suggests a GML paradigm that deliberately selects architectures whose spectral biases align with the expected structure of a problem's solution.

References

Bellman, Richard E., *Dynamic Programming*, Princeton University Press, 1957.

Carmona, Rene A., Jean-Pierre Fouque, and Li-Hsien Sun, "Mean field games and systemic risk," *Communications in Mathematical Sciences*, 2015, *13* (4), 911–933.

Chung, Fan R. K., *Spectral Graph Theory*, Vol. 92 of *CBMS Regional Conference Series in Mathematics*, American Mathematical Society, 1997.

Han, Jiequn, Arnulf Jentzen, and Weinan E, "Solving high-dimensional partial differential equations using deep learning," *Proceedings of the National Academy of Sciences*, 2018, *115* (34), 8505–8510.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, 1989, *2* (5), 359–366.

Kipf, Thomas N. and Max Welling, "Semi-supervised classification with graph convolutional networks," in "International Conference on Learning Representations" 2017.

Rahaman, Nasim Ullah, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron C. Courville, "On the Spectral Bias of Neural Networks," in "Proceedings of the 36th International Conference on Machine Learning" PMLR 2019, pp. 5301–5310.

Wu, Felix, Tianyi Zhang, Amauri Holanda de Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger, "Simplifying Graph Convolutional Networks," in "Proceedings of the 36th International Conference on Machine Learning" PMLR 2019, pp. 6861–6871.

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R. Salakhutdinov, and Alexander J. Smola, "Deep Sets," in "Advances in Neural Information Processing Systems 30" 2017.

7 Appendix: Supplementary Materials

A Hyperparameters and Implementation Details

This section provides the hyperparameters used for training the models in each of the three empirical acts. All models were implemented in JAX and Flax.

Table 4: Model and Training Hyperparameters

Parameter	**Act I**	**Act II**	**Act III**
Learning Rate	1×10^{-4}	3×10^{-4}	1×10^{-4}
Training Steps	20,000	50,000	60,000
Batch Size	256	256	256
Terminal Weight	100.0	100.0	100.0
Optimizer	AdamW	AdamW	AdamW
FFN Hidden Dims			
	[128, 128]	[128, 128]	[128, 128]
SymmetricMLP Hidden Dims			
	[64, 64]	Ag:[32,32], Hd:[64]	Ag:[32,32], Hd:[64]
GNN Hidden Dim			
	56	32	32

B Detailed Problem Formulations

B.1 The General Hamilton-Jacobi-Bellman Equation

We consider a system of N interacting agents whose joint state $\mathbf{x}_t \in \mathbb{R}^N$ evolves according to the stochastic differential equation (SDE) in Equation 1. The specific form of the HJB partial differential equation (PDE) that the value function must solve depends on the nature of the multi-agent interaction.

- **Social Planner (Cooperative) Problem:** A central planner seeks to minimize a single, joint cost. This yields a single, global value function $V(t, \mathbf{x})$ that solves the HJB equation:

$$-\frac{\partial V}{\partial t} = \inf_{\boldsymbol{\alpha} \in \mathbb{R}^N} \left\{ (\nabla_{\mathbf{x}} V)^T \boldsymbol{\mu}(\mathbf{x}, \boldsymbol{\alpha}) + L(\mathbf{x}, V) + C(\mathbf{x}, \boldsymbol{\alpha}) \right\} \tag{2}$$

  where $L(\mathbf{x}, V) = \frac{1}{2}\text{Tr}\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T \text{Hess}_{\mathbf{x}}(V)\right)$ is the second-order diffusion term and $C(\cdot)$ is the joint running cost.

- **Nash Equilibrium (Non-Cooperative) Problem:** Each agent i seeks to minimize its own cost, taking the policies of other agents as given. This results in a system of N coupled PDEs, one for each agent's value function $V^i(t, \mathbf{x})$:

$$-\frac{\partial V^i}{\partial t} = \inf_{\alpha_i \in \mathbb{R}} \left\{ \mathcal{L}^{\hat{\boldsymbol{\alpha}}_{-i}} V^i(\mathbf{x}) + C_i(\mathbf{x}, \alpha_i) \right\} \tag{3}$$

  where $\mathcal{L}^{\hat{\boldsymbol{\alpha}}_{-i}}$ is the infinitesimal generator of the SDE under the optimal policies of all other agents, $\hat{\boldsymbol{\alpha}}_{-i}$, and C_i is the running cost for agent i.

B.2 Act I: Mean-Field LQ Game

This model is based on the canonical mean-field game formulation for systemic risk presented in Carmona et al. (2015). It is a linear-quadratic game of N agents on a complete graph.

B.2.1 System Dynamics and Cost

The state of agent i, $\mathbf{x}_t^i$, evolves according to:

$$d\mathbf{x}_t^i = \left[a(\bar{\mathbf{x}}_t - \mathbf{x}_t^i) + \alpha_t^i\right] dt + \text{noise}_t^i \tag{4}$$

where $\bar{\mathbf{x}}_t = \frac{1}{N}\sum_j \mathbf{x}_t^j$. In matrix form, the drift is $\boldsymbol{\mu}(\mathbf{x}_t, \boldsymbol{\alpha}_t) = \boldsymbol{F}\mathbf{x}_t + \boldsymbol{B}\boldsymbol{\alpha}_t$, with:

- **Dynamics Matrix:** $\boldsymbol{F} = a(\frac{1}{N}\boldsymbol{J} - \boldsymbol{I}) = -\frac{a}{N}\boldsymbol{L}_{\text{complete}}$, where $\boldsymbol{J}$ is the matrix of ones and $\boldsymbol{L}_{\text{complete}}$ is the standard graph Laplacian for the complete graph.
- **Control Matrix:** $\boldsymbol{B} = \boldsymbol{I}$.

The noise has common and idiosyncratic components, with covariance $(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T)_{jk} = \sigma^2(\rho^2 + \delta_{jk}(1-\rho^2))$. Each agent i seeks to minimize a cost functional $J_i(\boldsymbol{\alpha})$ with running cost f_i and terminal cost g_i:

$$f_i(\mathbf{x}, \alpha_i) = \frac{1}{2}(\alpha_i)^2 - q\alpha_i(\bar{x} - x_i) + \frac{\epsilon}{2}(\bar{x} - x_i)^2 \tag{5}$$

$$g_i(\mathbf{x}) = \frac{c}{2}(\bar{x} - x_i)^2 \tag{6}$$

This defines a Nash Equilibrium problem where each agent's value function V^i must satisfy a coupled HJB equation.

B.2.2 Analytical Solution

The true value function for agent i is quadratic:

$$V^i(t, \mathbf{x}) = \frac{\eta_t}{2}(\bar{x} - x_i)^2 + \mu_t \tag{7}$$

This can be written as $V^i(t, \mathbf{x}) = \frac{1}{2}\mathbf{x}^T \boldsymbol{P}_t^i \mathbf{x} + \mu_t$, where the state-dependent part is determined by a time-dependent, rank-one matrix:

$$\boldsymbol{P}_t^i = \eta_t \left(\frac{1}{N}\mathbf{1} - \mathbf{e}_i\right)\left(\frac{1}{N}\mathbf{1} - \mathbf{e}_i\right)^T \tag{8}$$

Here, $\mathbf{e}_i$ is the i-th standard basis vector and the scalar function η_t solves the Riccati ODE:

$$\dot{\eta}_t = 2(a+q)\eta_t + \left(1 - \frac{1}{N^2}\right)\eta_t^2 - (\epsilon - q^2), \quad \eta_T = c \tag{9}$$

B.3 Act II: Sparse LQ Social Planner

This model is a cooperative linear-quadratic regulator problem on a ring graph.

B.3.1 System Dynamics and Cost

The state of agent i evolves according to local diffusion dynamics:

$$d\mathbf{x}_t^i = \left[a(\mathbf{x}_t^{i-1} - 2\mathbf{x}_t^i + \mathbf{x}_t^{i+1}) + \alpha_t^i\right] dt + \text{noise}_t^i \tag{10}$$

In matrix form, $\boldsymbol{\mu}(\mathbf{x}_t, \boldsymbol{\alpha}_t) = \boldsymbol{F}\mathbf{x}_t + \boldsymbol{B}\boldsymbol{\alpha}_t$, with:

- **Dynamics Matrix:** $\boldsymbol{F} = a\boldsymbol{L}_{\text{ring}}$, where $\boldsymbol{L}_{\text{ring}}$ is the sparse graph Laplacian for the ring graph.
- **Control Matrix:** $\boldsymbol{B} = \boldsymbol{I}$.

The social planner minimizes a single cost functional $J(\boldsymbol{\alpha})$ with quadratic running cost f and terminal cost g:

$$f(\mathbf{x}, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} + \frac{1}{2}\mathbf{x}^T\boldsymbol{Q}\mathbf{x}, \quad \text{where} \quad \boldsymbol{Q} = \epsilon(-\boldsymbol{L}_{\text{ring}}) \tag{11}$$

$$g(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\boldsymbol{Q}_T\mathbf{x}, \quad \text{where} \quad \boldsymbol{Q}_T = c(-\boldsymbol{L}_{\text{ring}}) \tag{12}$$

This defines a social planner problem with a single value function $V(t, \mathbf{x})$ that must solve the corresponding HJB PDE.

B.3.2 Analytical Solution

The true value function has a global quadratic structure:

$$V(t, \mathbf{x}) = \frac{1}{2}\mathbf{x}^T \boldsymbol{P}_t \mathbf{x} + \mu_t \tag{13}$$

The matrix $\boldsymbol{P}_t \in \mathbb{R}^{N \times N}$ solves the continuous-time matrix Differential Riccati Equation (DRE):

$$-\dot{\boldsymbol{P}}_t = \boldsymbol{Q} + \boldsymbol{F}^T \boldsymbol{P}_t + \boldsymbol{P}_t \boldsymbol{F} - \boldsymbol{P}_t \boldsymbol{B}\boldsymbol{B}^T \boldsymbol{P}_t \tag{14}$$

With the specific matrices for this model, this becomes:

$$-\dot{\boldsymbol{P}}_t = \epsilon(-\boldsymbol{L}_{\text{ring}}) + a(\boldsymbol{L}_{\text{ring}} \boldsymbol{P}_t + \boldsymbol{P}_t \boldsymbol{L}_{\text{ring}}) - \boldsymbol{P}_t^2 \tag{15}$$

solved backwards from the terminal condition $\boldsymbol{P}_T = c(-\boldsymbol{L}_{\text{ring}})$. Although the system matrices $\boldsymbol{F}$ and $\boldsymbol{Q}$ are sparse, the solution $\boldsymbol{P}_t$ to the Riccati equation is a dense, non-local matrix.

B.4 Act III: The "Clog in the Ring" Game

This model modifies the sparse LQ problem of Act II to include a localized, non-linear cost.

B.4.1 System Dynamics and Cost

The system dynamics and noise structure are identical to those in Act II, governed by $\boldsymbol{F} = a\boldsymbol{L}_{\text{ring}}$. The social planner's cost functional is modified with a quartic penalty on the

state of agent 0:

$$f(\mathbf{x}, \boldsymbol{\alpha}) = \underbrace{\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} + \frac{1}{2}\mathbf{x}^T\boldsymbol{Q}\mathbf{x}}_{\text{Original LQ Cost}} + \underbrace{\lambda(\mathbf{x}^0)^4}_{\text{The Clog}} \tag{16}$$

$$g(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\boldsymbol{Q}_T\mathbf{x} + \lambda(\mathbf{x}^0)^4 \tag{17}$$

The closed-loop HJB equation for the value function $V(t, \mathbf{x})$ is:

$$\begin{aligned} -\frac{\partial V}{\partial t} = (\nabla_{\mathbf{x}} V)^T \boldsymbol{F}\mathbf{x} - \frac{1}{2}||\nabla_{\mathbf{x}} V||_2^2 \\ + \left(\frac{1}{2}\mathbf{x}^T\boldsymbol{Q}\mathbf{x} + \lambda(\mathbf{x}^0)^4\right) \\ + \frac{1}{2}\mathrm{Tr}\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T \mathrm{Hess}_{\mathbf{x}}(V)\right) \end{aligned} \tag{18}$$

Due to the non-quadratic $(\mathbf{x}^0)^4$ term, this HJB equation does not admit an analytical solution.

B.4.2 Stability Analysis Formulation

The stability of the learned policy is assessed by linearizing the learned closed-loop dynamics around the equilibrium point $\mathbf{x} = \mathbf{0}$. The Jacobian matrix of the closed-loop system is given by:

$$\boldsymbol{A}_{\mathrm{cl}}(\mathbf{x}) = \boldsymbol{F} - \boldsymbol{B}\boldsymbol{B}^T \mathrm{Hess}_{\mathbf{x}}(V_{\boldsymbol{\theta}}(\mathbf{x})) \tag{19}$$

A stable policy requires that all eigenvalues of $\boldsymbol{A}_{\mathrm{cl}}(\mathbf{0})$ have negative real parts.

C Full Theoretical Proofs

C.1 Preliminaries: Graph Fourier Analysis

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected, non-bipartite graph. The symmetrically normalized adjacency matrix $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$ is real and symmetric, admitting an eigendecomposition $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$.

- The columns of the orthogonal matrix $\boldsymbol{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ form the **Graph Fourier Basis**.

- The diagonal matrix $\boldsymbol{\Lambda}$ contains the real eigenvalues, ordered $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_N \geq -1$. The eigenvector $\mathbf{u}_1$ is the constant vector, representing the lowest frequency (DC component), while subsequent eigenvectors correspond to increasingly higher frequencies of oscillation on the graph.

This eigendecomposition serves as the basis for graph Fourier analysis, where the eigenvectors of the graph operator are interpreted as frequency components (Chung, 1997).

Any signal $\mathbf{h} \in \mathbb{R}^N$ on the graph can be decomposed into two orthogonal subspaces relative to this basis:

- **Low-Frequency Subspace:** $\mathcal{H}_{\|} = \text{span}(\mathbf{u}_1)$.

- **High-Frequency Subspace:** $\mathcal{H}_{\perp} = \text{span}(\mathbf{u}_2, \ldots, \mathbf{u}_N)$.

The following lemma, which establishes that $\hat{\boldsymbol{A}}$ is a contraction mapping on the high-frequency subspace, is central to our main result.

Lemma 1 (High-Frequency Contraction).

Let $\lambda_{\max}^{\perp} = \max(|\lambda_2|, |\lambda_N|) < 1$. *For any signal* $\mathbf{h}_{\perp} \in \mathcal{H}_{\perp}$, *the operator* $\hat{\boldsymbol{A}}$ *is a strict contraction:*

$$||\hat{\boldsymbol{A}}\mathbf{h}_{\perp}||_2 \leq \lambda_{\max}^{\perp} \cdot ||\mathbf{h}_{\perp}||_2$$

Proof. Any $\mathbf{h}_{\perp} \in \mathcal{H}_{\perp}$ is a linear combination $\mathbf{h}_{\perp} = \sum_{i=2}^{N} c_i \mathbf{u}_i$. Applying the operator yields $\hat{\boldsymbol{A}}\mathbf{h}_{\perp} = \sum_{i=2}^{N} c_i \lambda_i \mathbf{u}_i$. Due to the orthonormality of $\boldsymbol{U}$, the squared norm is $||\hat{\boldsymbol{A}}\mathbf{h}_{\perp}||_2^2 = \sum_{i=2}^{N} c_i^2 \lambda_i^2 \leq (\lambda_{\max}^{\perp})^2 \sum_{i=2}^{N} c_i^2 = (\lambda_{\max}^{\perp})^2 ||\mathbf{h}_{\perp}||_2^2$. Taking the square root completes the proof. ■

C.2 Full Proof of Theorem 1 (GCN as a Low-Pass Filter)

Theorem 1 (GCN as a Low-Pass Filter). *Let a single GCN layer be defined by the transformation $\boldsymbol{H}_{out} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}_{in}\boldsymbol{W})$, where σ is a 1-Lipschitz activation function with $\sigma(0) = 0$. Let $\boldsymbol{v} = \boldsymbol{H}_{in}\boldsymbol{w}_k$ be the aggregated feature vector for an output channel k, and decompose it into its orthogonal frequency components, $\boldsymbol{v} = \boldsymbol{v}_{\|} + \boldsymbol{v}_{\perp}$. The pre-activation vector $\tilde{\boldsymbol{h}}_k = \hat{\boldsymbol{A}}\boldsymbol{v}$ exhibits the following spectral properties:*

1. *The low-frequency component is preserved: $\hat{\boldsymbol{A}}\boldsymbol{v}_{\|} = \boldsymbol{v}_{\|}$.*
2. *The norm of the high-frequency component is strictly contracted: $||\hat{\boldsymbol{A}}\boldsymbol{v}_{\perp}||_2 \leq \lambda^{\perp}_{\max}||\boldsymbol{v}_{\perp}||_2$.*

Since the gain on the low-frequency component is 1 while the gain on all higher-frequency components is strictly less than 1, the graph convolution operator $\hat{\boldsymbol{A}}$ acts as a low-pass filter.

Proof. The aggregated feature vector for channel k is $\boldsymbol{v} = \boldsymbol{H}_{\text{in}}\boldsymbol{w}_k$. The pre-activation is $\tilde{\boldsymbol{h}}_k = \hat{\boldsymbol{A}}\boldsymbol{v}$. We analyze the action of $\hat{\boldsymbol{A}}$ on each orthogonal component of $\boldsymbol{v}$ by linearity: $\tilde{\boldsymbol{h}}_k = \hat{\boldsymbol{A}}(\boldsymbol{v}_{\|} + \boldsymbol{v}_{\perp}) = \hat{\boldsymbol{A}}\boldsymbol{v}_{\|} + \hat{\boldsymbol{A}}\boldsymbol{v}_{\perp}$.

1. **Effect on Low Frequencies:** By definition, $\boldsymbol{v}_{\|}$ lies in the eigenspace of $\hat{\boldsymbol{A}}$ corresponding to the eigenvalue $\lambda_1 = 1$. Therefore, $\hat{\boldsymbol{A}}\boldsymbol{v}_{\|} = \lambda_1\boldsymbol{v}_{\|} = \boldsymbol{v}_{\|}$. The low-frequency component passes through the filter unchanged, with its norm perfectly preserved.
2. **Effect on High Frequencies:** The component $\boldsymbol{v}_{\perp}$ lies entirely within the high-frequency subspace $\mathcal{H}_{\perp}$. Applying Lemma 1 directly gives $||\hat{\boldsymbol{A}}\boldsymbol{v}_{\perp}||_2 \leq \lambda^{\perp}_{\max}||\boldsymbol{v}_{\perp}||_2$. Since $\lambda^{\perp}_{\max} < 1$, the high-frequency component is strictly contracted.
3. **Impact of Activation:** The final output for the channel is $\boldsymbol{h}_{k,\text{out}} = \sigma(\tilde{\boldsymbol{h}}_k)$. As σ is 1-Lipschitz with $\sigma(0) = 0$, it is a non-expansive mapping: $||\sigma(\mathbf{z})||_2 \leq ||\mathbf{z}||_2$. This ensures the activation cannot reverse the relative attenuation of high-frequency components achieved in the linear step. ■