UNDERSTANDING ROLLOUT ERROR IN GRAPH WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models are increasingly used for planning by predicting future states under candidate actions. Many planning environments, however, are naturally graph-structured, with agents, tools, skills, routes, and dependencies connected through fixed or evolving relations. Understanding how prediction errors accumulate in such graph-structured rollouts is therefore critical for reliable long-horizon planning.In this work, we systematically study how rollout errors propagate in Graph World Models (GWMs). We analyze both fixed- and dynamic-edge regimes and derive topology-aware error bounds that characterize the roles of graph structure, learned dynamics, and evolving edges. We then validate these theoretical findings across synthetic graph topologies and heterogeneous agent-graph testbeds, showing that rollout error and planning regret increase with horizon and that dynamic-edge training is important when graph structure evolves. Together, these results characterize when GWMs remain reliable under long-horizon planning and when graph structure amplifies rollout errors. Motivated by these findings, we propose **Error-Aware GWM**, which combines spectral regularization, rollout consistency, and critical-node weighting, and improves long-horizon stability without sacrificing one-step accuracy. Experiments show that Error-Aware GWM consistently improves long-horizon rollout stability without sacrificing prediction accuracy.

1 INTRODUCTION

World models have become an important approach to planning by predicting future states under candidate actions (Ha & Schmidhuber, 2018; Hafner et al., 2018; 2020; 2024; Chua et al., 2018; Schrittwieser et al., 2020; Wu et al., 2024; Bruce et al., 2024). Many planning environments, however, are naturally graph-structured: knowledge-graph memories support agent planning (Anokhin et al., 2025), external APIs form tool-use spaces (Li et al., 2023; Qin et al., 2024; Liu et al., 2024b; Trivedi et al., 2024), skill graphs organize reusable capabilities (Huang & Huang, 2025; Huang et al., 2025; Wang et al., 2025a; Xia et al., 2026), and communication networks connect interacting entities. In these environments, the world state includes not only node states but also the relations through which actions and information propagate. Reliable planning therefore depends critically on controlling rollout error, since prediction errors can accumulate over time and affect downstream decisions (Asadi et al., 2018; Janner et al., 2019; Kakade & Langford, 2002).

Graph World Models (GWMs) provide a natural framework for such planning by autoregressively predicting graph states $G_t = (V, E_t, X_t, A_t)$ (Feng et al., 2025). Given a learned transition model $\mathcal{F}_\theta(\hat{G}_k, a_k)$, a GWM predicts $\hat{G}_1, \ldots, \hat{G}_H$ to evaluate candidate action sequences. Yet how rollout error propagates in GWMs remains largely unexplored. Prior work has focused mainly on static graph tasks (Liu et al., 2024a; Chen et al., 2024), graph generation and dynamic graph representation learning (You et al., 2018; Vignac et al., 2023; Rossi et al., 2020; You et al., 2022), and domain-specific graph planning (Zhang et al., 2021; Ammanabrolu & Riedl, 2021). Feng et al. (Feng et al., 2025) introduced GWMs with action nodes, but did not study long-horizon error propagation under coupled node-edge prediction. This leaves an important gap for both long-horizon error propagation and dynamic-edge (DE) settings, where predicted edges become part of the state used for subsequent predictions.

Graph structure makes compounding error harder to analyze than in vector-state world models. Standard model-based RL analyses typically summarize rollout error with a scalar Lipschitz constant over a fixed-dimensional state (Asadi et al., 2018), but graph rollouts introduce three additional challenges. First, error propagation depends explicitly on topology: the same local perturbation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

may remain localized on a chain but spread widely in highly connected graphs, consistent with prior studies of graph propagation, dynamics, stability, and rewiring (Battaglia et al., 2016; 2018; Sanchez-Gonzalez et al., 2018; Oono & Suzuki, 2020; Chamberlain et al., 2021; Tori et al., 2024; Yu et al., 2025). For a message-passing GWM, this dependence separates into a topology term $\rho(A)$ and a model term $\prod_\ell \|W_\ell\|_2$. Second, when edges are predicted, node and edge errors become coupled: node-feature error $e_k^X = \|\hat{X}_k - X_k\|_F$ affects future edge prediction, while edge error $e_k^A = \|\hat{A}_k - A_k\|_F$ changes subsequent message passing. Third, these rollout errors ultimately matter because they affect downstream planning decisions over horizon H. These observations motivate our graph-specific analysis: we define $\widehat{\mathrm{GEAF}} = \rho(A) \prod_\ell \|W_\ell\|_2$ for fixed-edge error amplification, introduce a joint operator B for the recursion of (e_k^X, e_k^A) in dynamic-edge GWMs, and connect both forms of rollout error to planning regret.

We analyze accumulative rollout error in GWMs theoretically and empirically. Our graph error amplification factor $\widehat{\mathrm{GEAF}} = \rho(A) \prod_\ell \|W_\ell\|_2$ separates topology-dependent amplification from the model-dependent spectral norm product, while a joint node-edge operator B captures node-to-node, edge-to-node, node-to-edge, and edge-to-edge error propagation. This distinguishes fixed-edge (FE) models, where $M_X = 0$ and error propagation is node-only, from dynamic-edge (DE) models, where $M_X > 0$ activates node-edge cross-coupling. We evaluate these findings on seven synthetic graph topologies using GCN (Kipf & Welling, 2017), MPNN (Gilmer et al., 2017), GPS (Rampášek et al., 2022), and ActionNode GWM (Feng et al., 2025), with both FE and DE simulators and heterogeneous agent-graph testbeds inspired by tool/API and agent benchmarks (Li et al., 2023; Qin et al., 2024; Liu et al., 2024b; Trivedi et al., 2024). The results show that long rollouts amplify prediction errors into larger planning errors, while evolving graphs require DE training and exhibit the node-edge coupling predicted by B. Overall, GWM stability depends jointly on topology, horizon, and rollout regime rather than architecture alone.

We further propose **Error-Aware GWM** to improve rollout stability on high-spectral-radius graphs. The method controls topology-induced error amplification through spectral regularization, with rollout consistency and critical-node weighting further preserving predictive accuracy. Empirically, it prevents the divergence observed in vanilla GCN rollouts while maintaining low long-horizon error.

Our contributions are summarized as follows:

- **Systematic study of rollout error in GWMs (Section 2).** We study how errors accumulate in Graph World Models under both fixed-edge and dynamic-edge rollouts, establishing a unified framework for graph-structured planning.
- **Topology-aware error theory and analysis (Section 3).** We derive graph-valued rollout-error bounds that characterize topology-dependent amplification, node-edge coupling, and their effects on long-horizon planning.
- **Error-Aware GWM (Section 5).** Motivated by our theoretical and empirical findings, we propose Error-Aware GWM to control error amplification and improve long-horizon rollout stability.
- **Extensive evaluation (Section 6).** We validate the theory and method across diverse graph topologies, FE/DE rollout regimes, and heterogeneous agent-graph testbeds.
- **Real-world benchmark evaluation (Section H).** We further evaluate GWMs on real-world graph benchmarks to examine their applicability beyond controlled environments.

2 Graph World Model Framework

2.1 Preliminaries: Agent Graph World

A **Graph World Model** (GWM) represents an agent environment as a graph and learns its state transition for planning. At time t, the graph state is

$$G_t = (V, E_t, X_t, A_t). \tag{2.1}$$

Here, $V = \{v_1, \ldots, v_N\}$ is a fixed set of N nodes representing agents, tools, skills, or tasks; $E_t \subseteq V \times V$ is the edge set; $X_t \in \mathbb{R}^{N \times d}$ contains the d-dimensional node features; and $A_t \in \{0,1\}^{N \times N}$ is the adjacency matrix associated with E_t. Node features describe quantities such as agent status, tool outputs, skill availability, task progress, and failure signals. Edges describe calling, dependency, routing, or communication relations.

At each step, the planner selects an action $a_t \in \mathcal{A}$, where $\mathcal{A}$ denotes the action space. The GWM is a learned transition function

$$\mathcal{F}_\theta : \mathcal{G} \times \mathcal{A} \to \mathcal{G}, \qquad \hat{G}_{t+1} = \mathcal{F}_\theta(G_t, a_t), \tag{2.2}$$

where $\mathcal{G}$ is the graph-state space and θ denotes the model parameters. Given an initial state $\hat{G}_0 = G_t$ and a candidate action sequence $(a_0, \ldots, a_{H-1})$, an H-step rollout is defined recursively by

$$\hat{G}_{k+1} = \mathcal{F}_\theta(\hat{G}_k, a_k), \qquad k = 0, \ldots, H-1. \tag{2.3}$$

The resulting trajectory $(\hat{G}_1, \ldots, \hat{G}_H)$ is used to evaluate the candidate action sequence.

2.2 GWM Rollout Mechanism

Following Feng et al. (Feng et al., 2025), each action a_t is represented by an action node whose feature vector encodes the selected action and whose connections specify its targets in the current graph. A **node-level action** changes the state of one or more nodes, an **edge-level action** changes one or more relations, and a **graph-level action** changes a graph-wide property. We consider two settings according to whether the adjacency matrix is fixed or predicted during rollout.

Fixed-Edge GWM (FE-GWM). In FE-GWM, the edge set and adjacency matrix remain fixed during rollout, so $E_{k+1} = E_k = E$ and $A_{k+1} = A_k = A$, while the model predicts only node features. Let D be the degree matrix of A and $\tilde{A} = D^{-1/2} A D^{-1/2}$. A representative two-layer GCN transition is

$$\hat{X}_{k+1} = \tanh(\tilde{A}\hat{X}_k W_1 + \mathbf{1}(U a_k)^\top) W_2 + \xi_k, \tag{2.4}$$

where $\hat{X}_k \in \mathbb{R}^{N \times d}$ is the predicted node-feature matrix, $a_k \in \mathbb{R}^{d_a}$ is the action embedding, $U \in \mathbb{R}^{d_h \times d_a}$ maps the action to the hidden space, W_1 and W_2 are learnable weights, and $\xi_k \in \mathbb{R}^{N \times d}$ denotes one-step prediction error or rollout noise. If an action targets only part of the graph, its representation is injected only into the connected nodes.

Dynamic-Edge GWM (DE-GWM). In DE-GWM, both node features and graph structure are predicted:

$$\hat{X}_{k+1} = F_\theta^X(\hat{X}_k, \hat{A}_k, a_k), \qquad \hat{A}_{k+1} = F_\theta^A(\hat{X}_k, \hat{A}_k, a_k), \tag{2.5}$$

where F_θ^X and F_θ^A denote the node- and edge-transition functions, respectively. The node transition uses the current predicted adjacency $\hat{A}_k$ for message passing. For edge prediction, let $h_{k,i} \in \mathbb{R}^{d_h}$ denote the hidden representation of node v_i at rollout step k. A representative edge predictor is

$$\hat{A}_{k+1,ij} = \sigma(h_{k,i}^\top Q h_{k,j}), \tag{2.6}$$

where $Q \in \mathbb{R}^{d_h \times d_h}$ is learnable and σ is the sigmoid function, so $\hat{A}_{k+1,ij}$ is the predicted strength or probability of edge (v_i, v_j). Given ground-truth next-step states (X_{k+1}, A_{k+1}), the model can be trained with

$$\mathcal{L} = \|F_\theta^X(X_k, A_k, a_k) - X_{k+1}\|_F^2 + \lambda_e \|F_\theta^A(X_k, A_k, a_k) - A_{k+1}\|_F^2 + \lambda_w \|\theta\|_2^2, \tag{2.7}$$

where $\lambda_e \geq 0$ controls the edge-prediction loss and $\lambda_w \geq 0$ controls weight regularization.

3 Error Amplification Theory

Our analysis follows the assumptions stated in Section B. Figure 1 summarizes the resulting framework, from topology-dependent error amplification and autoregressive rollout to the joint node-edge error operator B and planning regret, while distinguishing the FE and DE regimes.

3.1 Problem Formulation

We study the accumulation of prediction error over an H-step GWM rollout defined in Equations (2.1)–(2.3). Let (X_k, A_k) be the true graph state at step k and $(\hat{X}_k, \hat{A}_k)$ its prediction. We define the node and edge rollout errors as

$$u_k = (e_k^X, e_k^A)^\top, \quad e_k^X = \|\hat{X}_k - X_k\|_F, \quad e_k^A = \|\hat{A}_k - A_k\|_F, \tag{3.1}$$

Our goal is to characterize how u_k evolves with rollout horizon and graph structure under the FE and DE settings defined in Section 2.2. Specifically, we seek bounds of the form $u_{k+1} \preceq B u_k + \epsilon$, where B describes graph-dependent error propagation and ϵ captures one-step prediction error. We then study when this recursion remains stable and how the resulting rollout error affects planning regret.

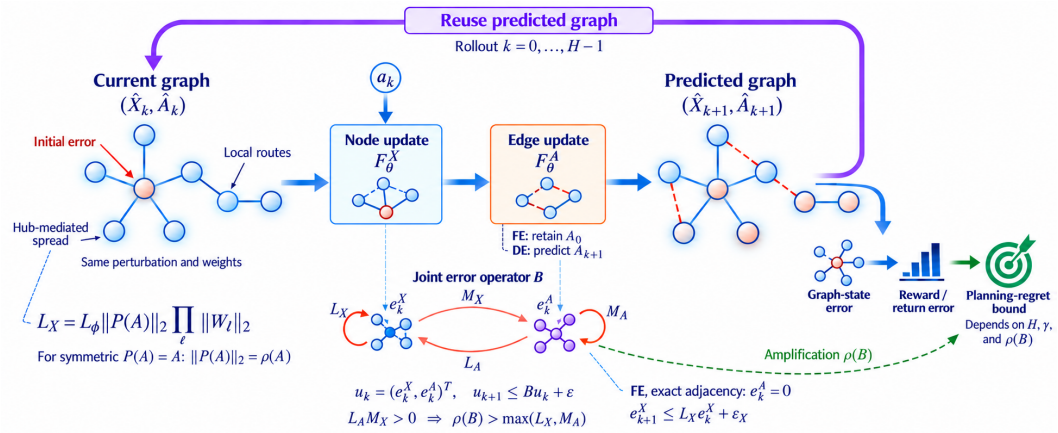


Figure 1: **Topology-dependent rollout error in GWMs.** Autoregressive rollouts propagate node and edge errors through graph structure. The joint operator B describes node-edge error propagation and its effect on planning regret; FE models reduce to node-only propagation, whereas DE models introduce coupled node-edge errors.

3.2 Fixed-Edge Rollout Error

We first analyze the fixed-edge (FE) regime, where topology is held fixed and only node features are predicted. Rollout error then propagates through a fixed message-passing operator. The bound separates the topology-dependent part of amplification from the model-dependent part.

Lemma 1 (Fixed-edge node error bound)**.** *Under the FE-GWM defined in Section 2.2 and Assumptions 2–3, let $A_t \equiv A$ and let the activation have Lipschitz constant L_σ. Then, for any rollout step $k \ge 0$,*

$$e_k^X \le L_X^k e_0^X + \epsilon_X \frac{L_X^k - 1}{L_X - 1}, \quad L_X = L_\sigma \underbrace{\rho(A)}_{\text{topology}} \underbrace{\prod_\ell \|W_\ell\|_2}_{\text{model}}. \tag{3.2}$$

Lemma 1 separates FE-GWM error amplification into a topology factor $\rho(A)$ and a model factor $\prod_\ell \|W_\ell\|_2$. The former captures error propagation through graph structure, while the latter captures amplification by the learned transition. We therefore define the **Graph Error Amplification Factor (GEAF)** as

$$\widehat{\mathrm{GEAF}}(G) = \rho(A) \prod_\ell \|W_\ell\|_2. \tag{3.3}$$

For fixed model weights, $\widehat{\mathrm{GEAF}}$ shows how graph topology alone can change long-horizon error growth. The proof of Lemma 1 is given in Appendix J.1.

3.3 Dynamic-Edge Rollout Error

We next analyze the dynamic-edge (DE) regime, where the GWM predicts both node states and graph structure. Here, rollout errors are coupled: node-feature errors can affect future edge prediction, and edge errors can change future message passing. We capture this interaction with a joint node-edge error operator.

Lemma 2 (Joint node-edge error bound)**.** *Under the DE-GWM defined in Section 2.2 and Assumptions 2 and 4, let $u_k = (e_k^X, e_k^A)^\top$. Then the joint rollout error satisfies*

$$u_{k+1} \preceq Bu_k + \epsilon, \qquad B = \begin{pmatrix} L_X & L_A \\ M_X & M_A \end{pmatrix}, \qquad \epsilon = (\epsilon_X, \epsilon_A)^\top. \tag{3.4}$$

The spectral radius of the joint error operator B is

$$\rho(B) = \frac{1}{2}\left[(L_X + M_A) + \sqrt{(L_X - M_A)^2 + 4L_A M_X}\right]. \tag{3.5}$$

If $L_A M_X > 0$, then $\rho(B) > \max(L_X, M_A)$.

so node-edge coupling increases the error amplification rate beyond either uncoupled component. Lemma 2 shows that DE-GWM rollout error is governed by a coupled operator rather than a scalar

node-error recursion. The off-diagonal terms L_A and M_X encode edge-to-node and node-to-edge error propagation. In FE models, there is no edge-prediction head, so $M_X = 0$ and the recursion reduces to node-only prediction. In DE models, $M_X > 0$ activates node-edge feedback. The proof is provided in Appendix J.2.

3.4 GEAF as a Proxy for the Joint Operator

The joint operator B characterizes node-edge error amplification, but its entries may not be directly available. We therefore study $\widehat{\mathrm{GEAF}}$ as a topology-aware proxy for its dominant amplification scale.

Lemma 3 (GEAF proxy bound)**.** *Under Assumptions 1–4, suppose $\|A\|_2 \geq R_X$, where R_X bounds the non-topological contribution to the first-row error term. Then*

$$\rho(B) \leq \widehat{\mathrm{GEAF}}(G)\left(1 + \frac{R_X}{\|A\|_2}\right) \leq 2\widehat{\mathrm{GEAF}}(G). \tag{3.6}$$

When the model weights are fixed across graph topologies, the ordering induced by $\widehat{\mathrm{GEAF}}(G)$ follows that of $\rho(A)$.

Lemma 3 bounds the joint amplification rate $\rho(B)$ by $\widehat{\mathrm{GEAF}}$ up to a constant factor in this regime. For FE-GWM, the edge-error terms vanish and $\rho(B) = L_X = \widehat{\mathrm{GEAF}}$ by construction. For DE-GWM, node-edge coupling breaks this identity, making the relation between $\rho(B)$ and $\widehat{\mathrm{GEAF}}$ nontrivial. The proof is given in Appendix J.3.

3.5 Planning Regret Under Graph Rollout Error

We next connect the joint rollout error in Equation (3.4) to planning performance under Assumptions 5–7. Recall that $u_k = (e_k^X, e_k^A)^\top$ is defined in Equation (3.1) and B is the joint error operator defined in Equation (3.4).

Lemma 4 (Planning regret bound)**.** *Under Assumptions 5–7, let $c = (\alpha, \beta)^\top$, $\tilde{\epsilon} = \|(\epsilon_X, \epsilon_A)^\top\|_2$, and let $\kappa(B)$ satisfy $\|B^j\|_2 \leq \kappa(B)\rho(B)^j$. Let $\hat{\pi}_\theta$ and π^* be the policies selected under the learned and true dynamics, respectively. Then*

$$J^*(\pi^*) - J^*(\hat{\pi}_\theta) \leq 2L_R\|c\|_2\kappa(B)\tilde{\epsilon}\,\Phi_H(\gamma, \rho(B)) + 2\epsilon_R\frac{1-\gamma^H}{1-\gamma}, \tag{3.7}$$

where $L_R, \epsilon_R, \gamma, H$ are defined in Assumptions 5–6, $\rho(B)$ is defined in Equation (3.5)*, and*

$$\Phi_H(\gamma, \rho) = \frac{1}{\rho - 1}\left[\frac{1-(\gamma\rho)^H}{1-\gamma\rho} - \frac{1-\gamma^H}{1-\gamma}\right]. \tag{3.8}$$

If $\gamma\rho(B) > 1$, the rollout-error term grows super-linearly with H. Lemma 4 shows that small one-step errors can produce large planning regret when the joint error operator is expansive. The spectral radius $\rho(B)$ controls error growth across rollout steps, while γ controls how strongly future errors affect the planning objective. The proof is given in Appendix J.4.

Corollary 5 (Error level versus growth rate)**.** *Under the FE recursion in Lemma 1, if the rollout error is not dominated by the one-step error floor, then*

$$\frac{\partial \log e_k^X}{\partial k} \to \log L_X. \tag{3.9}$$

Thus, the log-error growth rate can reveal topology-dependent amplification even when the final error approaches the floor $\epsilon_X/(1 - L_X)$. We therefore report GrowthSlope as an exploratory diagnostic. The proof is given in Appendix J.5.

4 Dataset and Evaluation Protocol

4.1 Graph Topologies

We evaluate GWMs on seven canonical graph families with distinct structural and spectral properties: chain, tree, grid, small-world, scale-free, star, and complete graphs. Each graph contains $N = 50$ nodes, with three outer seeds per topology. Together, these families span adjacency spectral radii from $\rho(A) = 2.0$ to 49.0 and cover sparse, clustered, hub-dominated, and dense connectivity regimes. Table 1 summarizes their topology statistics and rollout behavior.

Table 1: Topology statistics for the seven synthetic graph families at $N = 50$. $\rho(A)$ is the adjacency spectral radius, $\widehat{\mathrm{GEAF}}$ is the mean estimated GEAF over successful baseline runs, NodeMSE@32 is the median 32-step node prediction error, GrowthSlope is the median log-error slope from horizons 4 to 32, and Diverged counts rollout divergence across three seeds.

Topology	N	$\rho(A)$	$\widehat{\mathrm{GEAF}}$	NodeMSE@32	GrowthSlope	Diverged
Chain	50	2.00	8.7	5.0×10^{-4}	-0.0075	0/3
Tree	50	2.53	11.8	4.0×10^{-4}	-0.0042	0/3
Grid	50	3.65	16.0	2.7×10^{-4}	-0.0071	1/3
Small-World	50	4.16 ± 0.05	19.2	4.5×10^{-4}	$+0.0008$	1/3
Scale-Free	50	5.94 ± 0.15	24.3	2.9×10^{-4}	$+0.0005$	3/3
Star	50	7.00	29.4	1.9×10^{-4}	-0.0039	0/3
Complete	50	49.00	224.9	1.7×10^{-4}	0.0000	0/3

Scale-free graphs follow preferential attachment (Barabási & Albert, 1999), while small-world graphs follow the Watts–Strogatz model (Watts & Strogatz, 1998); the remaining families are deterministic for fixed N. Each graph is initialized with node features $X_0 \in \mathbb{R}^{N \times 8}$. Full generation parameters are provided in Appendix C.

4.2 Evaluation Environments

We evaluate both rollout regimes studied in Section 3. In the **FE simulator**, A remains fixed and node states evolve according to

$$X_t = f_{\mathrm{true}}(X_{t-1}, A) + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma^2 I), \tag{4.1}$$

with $\sigma = 0.01$, rollout length $T = 50$, and evaluation horizons $H \in \{1, 2, 4, 8, 16, 32\}$. In the **DE simulator**, both node states and edges evolve, activating the coupled error dynamics in Equation (3.4). The edge predictor follows Equation (2.6), and rollouts use $T = 32$.

Beyond canonical graphs, we evaluate two heterogeneous agent-graph environments. The **agent calling-tree testbed**, motivated by tool/API-based agent systems (Li et al., 2023; Qin et al., 2024; Liu et al., 2024b; Trivedi et al., 2024), represents multi-stage task execution as a typed directed acyclic graph, with sink-node success rate $\mathrm{sr}_{\mathrm{sink}}$ as the primary metric. The **platform skill-graph testbed**, motivated by graph-structured skill systems (Huang & Huang, 2025; Huang et al., 2025; Xia et al., 2026), models dependencies among skills, tools, validators, artifacts, and recovery components, with skill success rate $\mathrm{sr}_{\mathrm{skill}}$ as the primary metric. Detailed graph construction, node and edge types, simulator parameters, and calibration results are provided in Appendix C.

5 Error-Aware GWM

Motivated by the error amplification analysis in Section 3, we propose **Error-Aware GWM**, a training framework for improving long-horizon GWM stability. It applies to the FE and DE models defined in Section 2.2 without changing their basic transition architecture. The key idea is to retain the standard one-step prediction objective while explicitly controlling the sources of rollout error identified by our theory.

Specifically, Error-Aware GWM optimizes

$$\mathcal{L}_{\mathrm{EA}} = \mathcal{L}_{\mathrm{pred}} + \lambda_{\mathrm{spec}} R_{\mathrm{spec}} + \lambda_{\mathrm{roll}} R_{\mathrm{roll}} + \lambda_{\mathrm{crit}} R_{\mathrm{crit}}, \tag{5.1}$$

where $\mathcal{L}_{\mathrm{pred}}$ is the standard one-step prediction loss. The three additional terms target complementary sources of long-horizon error:

$$R_{\mathrm{spec}} = \sum_{\ell} \|W_\ell\|_2^2, \quad R_{\mathrm{roll}} = \sum_{h \in \mathcal{H}} \|\hat{X}_{t+h}^{(h)} - X_{t+h}\|_F^2, \quad R_{\mathrm{crit}} = \sum_{v \in V} w_v \|\hat{X}_{t+1,v} - X_{t+1,v}\|_2^2. \tag{5.2}$$

Here, $\mathcal{H}$ denotes the training rollout horizons, $\hat{X}_{t+h}^{(h)}$ the h-step autoregressive prediction, and w_v the normalized structural importance of node v. The three terms respectively control the model factor $\prod_\ell \|W_\ell\|_2$ in $\widehat{\mathrm{GEAF}}$, penalize accumulated rollout drift, and emphasize errors on structurally important nodes. These designs follow spectral control for Lipschitz stability (Miyato et al., 2018) and multi-step training under predicted trajectories (Bengio et al., 2015; Janner et al., 2019).

For DE-GWM, we further include edge prediction and regularize node-edge interactions associated with the joint operator B in Equation (3.4). Thus, Error-Aware GWM addresses both topology-dependent amplification and DE node-edge coupling. Full details are provided in Appendix D.

6 EXPERIMENTAL SETUP

Baselines. We compare six world-model baselines spanning non-graph dynamics, message passing, graph transformers, and action-node GWMs. The MLP world model (MLP-WM) ignores graph structure. The vanilla GCN world model is based on graph convolution (Kipf & Welling, 2017). The MPNN world model is based on neural message passing (Gilmer et al., 2017). We also include a GPS-style graph transformer (Rampášek et al., 2022). The ActionNode GWM is based on the fixed-edge architecture of Feng et al. (Feng et al., 2025). Our Error-Aware GWM adds spectral control, rollout consistency, and critical-node weighting to the same rollout interface.

Topologies. We evaluate all baselines on seven synthetic graph families: chain, tree, grid, small-world, scale-free, star, and complete graphs. Each graph has $N = 50$ nodes. Scale-free graphs follow the preferential-attachment model (Barabási & Albert, 1999), and small-world graphs follow the Watts–Strogatz model (Watts & Strogatz, 1998). These topologies span a broad spectral range, from $\rho(A) = 2.0$ for chain graphs to $\rho(A) = 49.0$ for complete graphs, as summarized in Table 1.

Fixed-edge training. In the fixed-edge setting, the graph topology is fixed during rollout and each model predicts node states only. We train all six baselines on the FE simulator described in Section 4.2. Evaluation uses rollout horizons $H \in \{1, 2, 4, 8, 16, 32\}$, allowing us to measure both short-horizon prediction quality and long-horizon error accumulation, following the standard concern of compounding error in model-based rollout (Asadi et al., 2018; Janner et al., 2019).

Dynamic-edge training. In the dynamic-edge setting, each baseline is extended with the bilinear edge-prediction head in Equation (C.2). The model predicts both node features and edges, and is trained with the node loss, edge BCE loss, and the graph regularizer in Equation (D.6). This setting is related to dynamic graph representation learning and temporal graph networks (Rossi et al., 2020; You et al., 2022), but differs in that the predicted graph is fed back as an autoregressive world-model state for rollout prediction. All six baselines are retrained on dynamic-edge rollouts across the six DE topologies and three seeds, yielding $6 \times 6 \times 3 = 108$ runs.

Heterogeneous agent testbeds. For the agent calling-tree and platform skill-graph testbeds, we evaluate six R-GCN variants: small, medium, deep, wide, action-node, and critical-node variants. These models handle typed nodes and typed edges in heterogeneous graphs (Schlichtkrull et al., 2018), following the general message-passing view of graph networks (Battaglia et al., 2018). The testbeds are motivated by API/tool-use and interactive-agent benchmarks (Li et al., 2023; Qin et al., 2024; Liu et al., 2024b; Trivedi et al., 2024), but evaluate a different question: whether a graph world model can roll out the evolving execution graph. Each model is trained for 50 epochs with T-batched forward passes.

7 RQ1: DOES TOPOLOGY PREDICT ROLLOUT-ERROR GROWTH?

We first test whether topology predicts the *rate* of rollout-error growth. Our theory suggests that $\rho(A)$ and $\widehat{\mathrm{GEAF}}$ should be reflected more directly in temporal growth than in terminal error, which can be affected by optimization and error floors. We therefore examine the relation between $\widehat{\mathrm{GEAF}}$ and GrowthSlope and distinguish this effect from standard GNN over-smoothing.

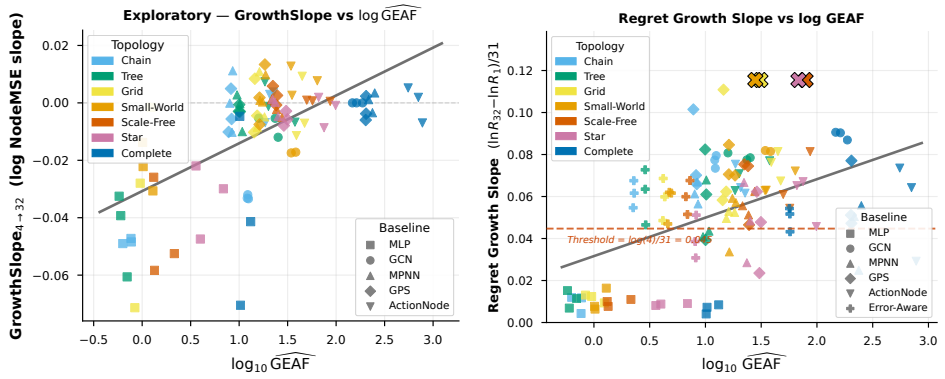
7.1 GEAF TRACKS ERROR-GROWTH SLOPE

Figure 2a relates $\log \widehat{\mathrm{GEAF}}$ to $\mathrm{GrowthSlope}_{4\to32}$, the slope of log-NodeMSE from horizon 4 to 32. Larger $\log \widehat{\mathrm{GEAF}}$ is associated with faster error growth, consistent with Corollary 5. Thus, $\widehat{\mathrm{GEAF}}$ does not necessarily determine terminal error, but captures the topology-dependent rate of long-horizon error amplification.

7.2 TOPOLOGY EFFECTS STRENGTHEN WITH ROLLOUT HORIZON AND EDGE DENSITY

We study how rollout error changes with horizon. The experiment evaluates $H \in \{1, 2, 4, 8, 16, 32, 48\}$ across six baselines, seven topologies, and three seeds. Table 2 reports NodeMSE@H as a function of rollout horizon.

Low-risk topologies such as chain and tree remain stable over long horizons. High-risk topologies such as scale-free, star, and grid show sharp error growth at larger horizons. This complements the



(a) Rollout-error growth.

(b) Planning-regret growth.

Figure 2: **GEAF predicts long-horizon error and planning-regret growth.** (a) $\log \widehat{\text{GEAF}}$ is positively associated with GrowthSlope$_{4\to32}$ of NodeMSE. (b) Higher $\log_{10} \widehat{\text{GEAF}}$ is also associated with faster planning-regret growth from horizon 4 to 32.

regret analysis: long-horizon behavior is strongly topology-dependent, and short-horizon performance does not fully characterize rollout stability.

Table 2: **Rollout-length scaling across methods.** Median NodeMSE over seven graph topologies and three random seeds at each rollout horizon. Lower is better.

Method	$H=1$	$H=2$	$H=4$	$H=8$	$H=16$	$H=32$	$H=48$
MLP	1.299×10^{-1}	2.077×10^{-2}	1.062×10^{-3}	2.694×10^{-4}	2.598×10^{-4}	2.991×10^{-4}	2.959×10^{-4}
GCN	5.815×10^{-1}	5.103×10^{-1}	5.294×10^{-1}	7.491×10^{-1}	9.948×10^{-1}	9.948×10^{-1}	9.945×10^{-1}
MPNN	1.815×10^{-3}	5.883×10^{-4}	$\mathbf{2.118\times10^{-4}}$	$\mathbf{1.985\times10^{-4}}$	$\mathbf{1.920\times10^{-4}}$	$\mathbf{2.152\times10^{-4}}$	$\mathbf{2.652\times10^{-4}}$
GPS	3.111×10^{-3}	7.845×10^{-4}	3.608×10^{-4}	2.788×10^{-4}	3.158×10^{-4}	3.118×10^{-4}	3.731×10^{-4}
ActionNode GWM	2.286×10^{-3}	6.851×10^{-4}	2.819×10^{-4}	2.588×10^{-4}	2.649×10^{-4}	2.976×10^{-4}	4.022×10^{-4}
Error-Aware GWM	$\mathbf{1.665\times10^{-3}}$	$\mathbf{5.063\times10^{-4}}$	2.209×10^{-4}	2.030×10^{-4}	2.206×10^{-4}	2.495×10^{-4}	3.291×10^{-4}

7.3 Rollout Error Affects Planning Regret?

We next test whether prediction error affects downstream planning. Because GWMs select actions through imagined graph rollouts, accumulated prediction errors become harmful when they distort predicted returns. Figure 2b shows that runs with larger $\widehat{\text{GEAF}}$ tend to exhibit faster regret growth from horizon 4 to 32, consistent with Lemma 4: stronger error amplification makes imagined trajectories less reliable as the horizon increases. Planning regret at horizon 32 is also substantially larger than one-step regret across all topologies, confirming that the observed planning failure is primarily a long-horizon rollout effect rather than a one-step artifact.

8 RQ2: When Does Dynamic-Edge Coupling Matter?

We study when node-edge coupling matters. In FE models, edges are fixed and coupling is absent, whereas in DE models, predicted edges affect future node states and node states affect future edge predictions. We test whether DE training improves DE rollout accuracy and whether trained models exhibit the coupling predicted by Lemma 2.

8.1 DE Training Improves Rollout Accuracy and Activates Node-Edge Coupling

We compare FE-trained and DE-trained models on DE rollouts using $n = 36$ paired cells: ActionNode GWM and Error-Aware GWM across six topologies and three seeds. Table 3 shows that DE training improves NodeMSE@32 for both architectures, with every paired comparison favoring DE training. The combined improvement is $11.0\times$, supporting Lemma 2 that DE rollouts require learning both node dynamics and node-edge interactions.

We further test whether trained DE models exhibit the predicted coupling. Because the edge predictor gives $M_X > 0$ and message passing gives $L_A > 0$, Lemma 2 predicts

$$\rho(B) > \max(L_X, M_A) \tag{8.1}$$

when $L_A M_X > 0$. This condition holds in all 108 DE-trained cells, with $p < 10^{-30}$ and mean excess ≈ 155 (Table 3). The large margin shows that the learned models have strong node-edge coupling rather than behaving like nearly fixed-edge models.

Table 3: Dynamic-edge training and node-edge coupling. The FE-trained / DE-trained NodeMSE ratio measures the improvement from DE training, with larger values indicating greater improvement.

Test / Architecture	n	FE-trained / DE-trained NodeMSE	Wilcoxon p
DE-trained vs. FE-trained on DE rollouts			
ActionNode GWM	18	$9.85\times$	3.8×10^{-6}
Error-Aware GWM	18	$12.29\times$	3.8×10^{-6}
Combined	**36**	$\mathbf{11.0}\times$ [9.4, 12.8]	$\mathbf{1.5 \times 10^{-11}}$
Node-edge coupling in DE-trained models			
All DE-trained models	108	$\rho(B)/\max(L_X, M_A) > 1$ in 108/108; mean excess ≈ 155	$< 10^{-30}$

8.2 FE Models Show Dormant Coupling and Perturbation Contraction

In the FE regime, the edge-prediction head is absent, so $M_X = 0$ and the joint operator reduces to node-only dynamics. The coupling condition in Equation (8.1) therefore cannot be activated. As shown in Figure 3, injected perturbations rapidly decay to the numerical floor, indicating contractive FE dynamics. Error-Aware GWM also achieves substantially lower GEAF than ActionNode GWM across topologies, consistent with spectral regularization reducing rollout-error amplification.

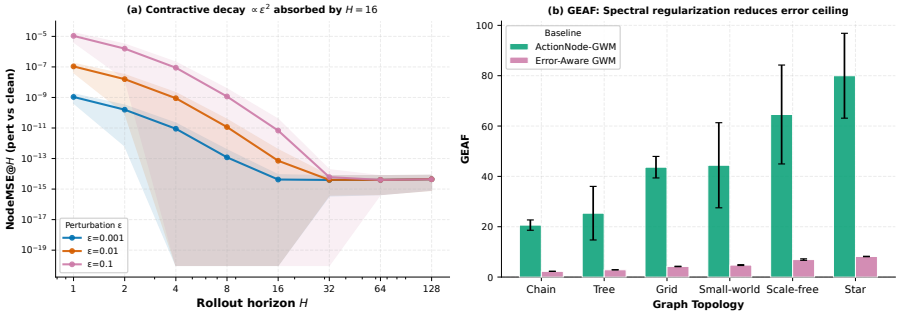


Figure 3: **FE-regime contraction and GEAF reduction.** Left: injected perturbations rapidly decay to the numerical floor. Right: Error-Aware GWM achieves lower GEAF than ActionNode GWM across topologies.

8.3 FE Perturbation Decomposition

Figure 4 further separates FE behavior under node-only, edge-only, and joint perturbations. Node-only perturbations have consistently negative GrowthSlope, showing that their effects decay over time. Edge-only perturbations remain near zero growth, while joint perturbations closely follow the edge-only case. This behavior is consistent with the FE condition $M_X = 0$, under which node-edge cross-coupling is inactive.

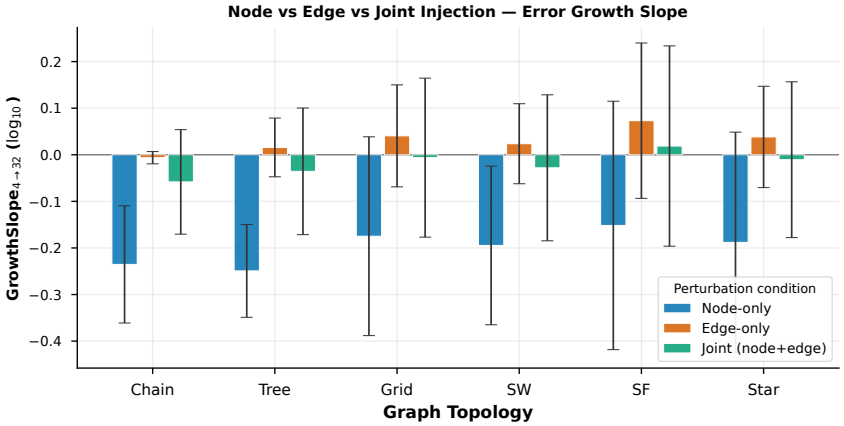


Figure 4: **Perturbation decomposition in the FE regime.**

9 Conclusion

We systematically studied rollout error in Graph World Models under fixed- and dynamic-edge settings, showing how topology, learned dynamics, and node-edge coupling govern long-horizon error propagation and planning regret. Our theoretical and empirical results characterize when graph-structured rollouts remain stable and when errors accumulate over long horizons. Motivated by these findings, we proposed Error-Aware GWM, which improves rollout stability across diverse graph topologies and heterogeneous agent-graph environments.

AI USE STATEMENT

AI tools were used for editorial polishing, including improvements to grammar, sentence structure, clarity, concision, and terminological consistency.

REFERENCES

Prithviraj Ammanabrolu and Mark O. Riedl. Learning knowledge graph-based world models of textual environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.

Kavosh Asadi, Dipendra Kumar Misra, and Michael L. Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 2024.

Ben Chamberlain, James Rowbottom, Maria I. Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: Graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1407–1418. PMLR, 2021.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. LLaGA: Large language and graph assistant. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 2024.

Taiye Chen, Xun Hu, Zihan Ding, and Chi Jin. Learning world models for interactive video generation. *arXiv preprint arXiv:2505.21996*, 2025.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Tao Feng, Yexin Wu, Guanyu Lin, and Jiaxuan You. Graph world model. In *International Conference on Machine Learning (ICML)*, 2025.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017.

Nico Gürtler and Georg Martius. Long-horizon planning with predictable skills. *Reinforcement Learning Journal*, 6:1247–1272, 2025.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Ken Huang and Jerry Huang. Audited skill-graph self-improvement for agentic llms via verifiable rewards, experience synthesis, and continual memory. *arXiv preprint arXiv:2512.23760*, 2025.

Xu Huang, Junwu Chen, Yuxing Fei, Zhuohan Li, Philippe Schwaller, and Gerbrand Ceder. Cascade: Cumulative agentic skill creation through autonomous development and evolution, 2025. URL https://arxiv.org/abs/2512.23880.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2002.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *International Conference on Learning Representations (ICLR)*, 2020.

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

Srijan Kumar, Francesca Spezzano, V. S. Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *IEEE International Conference on Data Mining (ICDM)*, pp. 221–230, 2016.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1269–1278, 2019.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3102–3116, 2023.

Mingwei Li, Xiaoyuan Zhang, Chengwei Yang, Zilong Zheng, and Yaodong Yang. Prismatic world model: Learning compositional dynamics for planning in hybrid systems. *arXiv preprint arXiv:2512.08411*, 2025a.

Xinyi Li, Zaishuo Xia, Weyl Lu, Chenjie Hao, and Yubei Chen. Smallworlds: Assessing dynamics understanding of world models in isolated environments. *arXiv preprint arXiv:2511.23465*, 2025b.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*, 2024a.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024b.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Tianwei Ni, Esther Derman, Vineet Jain, Vincent Taboga, Siamak Ravanbakhsh, and Pierre-Luc Bacon. Long-horizon model-based offline reinforcement learning without conservatism. *arXiv preprint arXiv:2512.04341*, 2025.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *International Conference on Learning Representations (ICLR)*, 2020.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14501–14515, 2022.

Luke Rivard, Sun Sun, Hongyu Guo, Wenhu Chen, and Yuntian Deng. Neuralos: Towards simulating operating systems via neural generative models. *arXiv preprint arXiv:2507.08800*, 2025.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4470–4479. PMLR, 2018.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference (ESWC)*, pp. 593–607, 2018.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2020.

Floriano Tori, Vincent Holst, and Vincent Ginis. The effectiveness of curvature-based rewiring and the role of hyperparameters in gnns revisited, 2024. URL `https://arxiv.org/abs/2407.09381`.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 16022–16076, 2024.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations (ICLR)*, 2023.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library, 2025a. URL https://arxiv.org/abs/2512.17102.

Zizhao Wang, Kaixin Wang, Li Zhao, Peter Stone, and Jiang Bian. Dyn-o: Building structured world models with object-centric representations, 2025b. URL https://arxiv.org/abs/2507.03298.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Interactive videogpts are scalable world models, 2024. URL https://arxiv.org/abs/2405.15223.

Tianle Xia, Lingxiang Hu, Yiding Sun, Ming Xu, Lan Xu, Siying Wang, Wei Xu, and Jie Jiang. Grasp: Graph-structured skill compositions for llm agents, 2026. URL https://arxiv.org/abs/2604.17870.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning (ICML)*, 2018.

Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: Graph learning framework for dynamic graphs. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

Youn-Yeol Yu, Jeongwhan Choi, Jaehyeon Park, Kookjin Lee, and Noseong Park. Piorf: Physics-informed ollivier-ricci flow for long-range interactions in mesh graph neural networks, 2025. URL https://arxiv.org/abs/2504.04052.

Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In *International Conference on Machine Learning (ICML)*, 2021.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, and Nicolas Ballas. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026.

Deyao Zhu, Li Erran Li, and Mohamed Elhoseiny. Value memory graph: A graph-structured world model for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

A Related Work

World models for planning. World models have been widely studied in model-based reinforcement learning, where a learned dynamics model is used to simulate future trajectories for planning or policy learning. A central challenge is compounding error: small one-step model errors can accumulate over long rollouts and degrade decision quality. Asadi et al. (Asadi et al., 2018) analyze this effect through Lipschitz continuity of learned transition models. PETS uses probabilistic ensembles and trajectory sampling to propagate uncertainty during model-based planning (Chua et al., 2018). Janner et al. (Janner et al., 2019) study model-based policy optimization with short branched rollouts to reduce error accumulation. Kakade and Langford (Kakade & Langford, 2002) provide performance-difference tools for policy improvement. Related sequence-modeling work addresses exposure bias by training models on their own generated histories (Bengio et al., 2015). Recent world-model systems reduce rollout drift through retrieval-augmented context (Chen et al., 2025), hierarchical latent planning (Zhang et al., 2026), Bayesian long-horizon planning (Ni et al., 2025), modular dynamics experts (Li et al., 2025a), skill-level abstraction (Gürtler & Martius, 2025), operating-system simulation (Rivard et al., 2025), and controlled long-horizon evaluation benchmarks (Li et al., 2025b). These works motivate rollout-error analysis, but they do not model graph topology, message passing, or coupled node-edge prediction.

Graph neural networks and graph-structured prediction. Graph neural networks provide a standard framework for learning over graph-structured data by propagating information along edges. Interaction Networks and Graph Networks introduced object- and relation-centric neural dynamics for physical reasoning (Battaglia et al., 2016; 2018), and graph-network physics engines showed that such models can support inference and control in dynamical systems (Sanchez-Gonzalez et al., 2018). Graph convolutional networks aggregate normalized neighborhood information (Kipf & Welling, 2017). Message-passing neural networks generalize local edge-based aggregation (Gilmer et al., 2017). Graph transformers such as GPS combine message passing with global attention (Rampášek et al., 2022; Ying et al., 2021). Dynamic graph models and temporal graph networks learn over time-varying structure (Rossi et al., 2020; You et al., 2022), while graph generation methods model distributions over discrete graph structure (You et al., 2018; Vignac et al., 2023). Prior theory shows that graph propagation is strongly affected by spectral properties of the adjacency or normalized adjacency matrix (Oono & Suzuki, 2020), and diffusion-based GNNs connect graph learning to stability of discretized diffusion dynamics (Chamberlain et al., 2021). Related work on graph rewiring studies how modifying topology can improve information flow and reduce harmful propagation effects (Tori et al., 2024; Yu et al., 2025). These results show that topology shapes prediction behavior, but they mainly study representation learning, graph generation, or one-step dynamics rather than autoregressive graph rollouts where prediction errors become future inputs.

Graph world models. Graph world models extend world modeling to graph-structured states. Feng et al. (Feng et al., 2025) introduce a GWM framework that represents multimodal observations, graph states, action nodes, and target nodes within a unified graph interface. Zhang et al. (Zhang et al., 2021) learn latent landmarks as a graph-structured abstraction for planning. Value Memory Graph constructs graph-structured memory for offline reinforcement learning (Zhu et al., 2023). Knowledge-graph world models represent textual environments through structured relational memory (Ammanabrolu & Riedl, 2021). Contrastive structured world models learn object-centric relational dynamics (Kipf et al., 2020). Dyn-O builds structured world models with object-centric representations (Wang et al., 2025b). These works mainly focus on representation, architecture, and task performance. Our work instead studies long-horizon error behavior: how topology affects rollout error, how fixed-edge and dynamic-edge regimes differ, and how node and edge prediction errors interact through a joint operator.

Agent graphs and skill-graph systems. Multi-agent and tool-using systems increasingly organize computation as graphs, including agent calling-trees, knowledge-graph memories, tool graphs, and skill graphs. API-Bank, ToolLLM/ToolBench, AgentBench, and AppWorld evaluate tool retrieval, API calling, multi-turn decision-making, and interactive digital tasks (Li et al., 2023; Qin et al., 2024; Liu et al., 2024b; Trivedi et al., 2024). AriGraph uses graph memory for LLM agents (Anokhin et al., 2025). Audited Skill-Graph Self-Improvement studies verifier-backed skill-graph maintenance (Huang & Huang, 2025). CASCADE studies cumulative agentic skill creation through

autonomous development and evolution (Huang et al., 2025). Reinforcement learning for self-improving agents with skill libraries studies policy learning over reusable skills (Wang et al., 2025a). GraSP compiles retrieved skills into a typed skill graph so failed subgraphs can be repaired locally rather than replanning the full trajectory (Xia et al., 2026). These works show the practical value of graph-structured agent execution. Our work is complementary: we analyze how prediction errors accumulate when such graph states are rolled out over time, and we propose Error-Aware GWM to improve long-horizon stability in graph-based planning.

B ASSUMPTIONS

We use standard assumptions for finite-horizon model-based planning and graph neural networks. They require bounded graph states and one-step model errors, Lipschitz node and edge transitions, and bounded reward sensitivity. These conditions hold in our settings, where node identities remain fixed during each rollout.

Assumption 1 (Bounded graph states)**.** *For the graph state G_t defined in Equation* (2.1)*, $|V| = N$, $X_t \in \mathbb{R}^{N\times D}$, and $A_t \in [0,1]^{N\times N}$. There exist $R_X, R_A < \infty$ such that*

$$\|X_t\|_F \le R_X, \qquad \|A_t\|_F \le R_A \tag{B.1}$$

for all rollout steps $t \in [0, T]$.

This holds in our experiments because node identities are fixed, node features are bounded, and adjacency entries lie in $[0, 1]$. The analysis extends to growing graphs using a time-dependent operator B_t with $\sup_t \rho(B_t) < \infty$.

Assumption 2 (Bounded one-step model error)**.** *Let $F^\star = (F^{X\star}, F^{A\star})$ denote the true transition and $F_\theta = (F_\theta^X, F_\theta^A)$ the learned transition from Equation* (2.2)*. There exist $\epsilon_X, \epsilon_A \ge 0$ such that*

$$\begin{aligned} \|F_\theta^X(X, A, a) - F^{X\star}(X, A, a)\|_F &\le \epsilon_X, \\ \|F_\theta^A(X, A, a) - F^{A\star}(X, A, a)\|_F &\le \epsilon_A. \end{aligned} \tag{B.2}$$

Here, ϵ_X and ϵ_A bound one-step node and edge prediction errors. The same analysis applies in expectation when only expected errors are available.

Assumption 3 (Lipschitz message passing)**.** *For the FE transition in Equation* (2.4)*, the activation σ is coordinate-wise L_σ-Lipschitz with $L_\sigma \le 1$, and the transition weights have finite spectral norms.*

For multiple message-passing layers, the model dependence is captured by $\prod_\ell \|W_\ell\|_2$. This condition covers the GNN-based transition models considered in our analysis.

Assumption 4 (Lipschitz dynamic edge update)**.** *For the DE transition in Equation* (2.5)*, the node and edge errors defined in Equation* (3.1) *satisfy*

$$\begin{aligned} e_{k+1}^X &\le L_X e_k^X + L_A e_k^A + \epsilon_X, \\ e_{k+1}^A &\le M_X e_k^X + M_A e_k^A + \epsilon_A, \end{aligned} \tag{B.3}$$

for finite constants $L_X, L_A, M_X, M_A \ge 0$.

This condition covers differentiable edge predictors such as the bilinear predictor in Equation (2.6). In FE-GWM, edge prediction is absent and the recursion reduces to node error alone.

Assumption 5 (Lipschitz reward)**.** *For any graph states G, G' and action a, the true reward satisfies*

$$|R^\star(G, a) - R^\star(G', a)| \le L_R d_G(G, G'), \tag{B.4}$$

and the learned reward satisfies

$$|R_\theta(G, a) - R^\star(G, a)| \le \epsilon_R, \tag{B.5}$$

for constants $L_R, \epsilon_R \ge 0$.

This assumption is required only for the planning-regret analysis in Lemma 4.

Assumption 6 (Finite-horizon return)**.** *For rollout horizon H, planning uses*

$$J^\star = \sum_{k=0}^{H-1} \gamma^k r_{t+k}, \qquad \gamma \in [0, 1]. \tag{B.6}$$

The case $\gamma = 1$ is allowed for finite H.

Assumption 7 (Graph distance)**.** *Using the node and edge errors in Equation* (3.1)*, the graph distance satisfies*

$$d_G(G_k, \hat{G}_k) \le \alpha e_k^X + \beta e_k^A, \tag{B.7}$$

where $\alpha, \beta \ge 0$.

This condition is natural for fixed node identities and also permits alternative graph distances that can be bounded by node and edge errors.

Together, Assumptions 1–7 support the FE and DE analyses. Lemma 1 analyzes FE error amplification, Lemma 2 extends it to coupled node-edge errors, Lemma 3 relates $\widehat{\mathrm{GEAF}}$ to the joint operator, and Lemma 4 connects rollout error to planning regret.

C Dataset and Simulator Details

C.1 Synthetic Graph Generation

All synthetic graphs contain $N = 50$ nodes and are evaluated using three outer seeds per topology. Scale-free graphs are generated by preferential attachment with $m = 2$ edges per new node (Barabási & Albert, 1999). Small-world graphs use the Watts–Strogatz model with neighborhood size $k = 4$ and rewiring probability $p = 0.3$ (Watts & Strogatz, 1998). Chain, tree, grid, star, and complete graphs are deterministic for fixed N. Each graph instance stores its adjacency matrix A, initial node features $X_0 \in \mathbb{R}^{N\times 8}$, and rollout trajectory $\{X_t, A_t\}_{t=1}^T$.

The seven families cover three broad structural regimes: low-connectivity or tree-like graphs (chain and tree), lattice or clustered graphs (grid and small-world), and hub-dominated or dense graphs (scale-free, star, and complete). Their spectral range, summarized in Table 1, provides controlled variation in the topology factor $\rho(A)$ appearing in $\widehat{\mathrm{GEAF}}$.

C.2 Fixed-Edge Simulator

The FE simulator follows the fixed-edge setting in Section 3.2. The adjacency matrix remains unchanged throughout each trajectory, while node features evolve as

$$X_t = f_{\mathrm{true}}(X_{t-1}, A) + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma^2 I), \tag{C.1}$$

with $\sigma = 0.01$. Node states $X_t \in \mathbb{R}^{N\times 8}$ represent continuous state variables such as congestion or utilization. Each trajectory has length $T = 50$, and prediction error is evaluated at horizons $H \in \{1, 2, 4, 8, 16, 32\}$ using the autoregressive rollout in Equation (2.3).

C.3 Dynamic-Edge Simulator

The DE simulator follows the dynamic-edge setting in Section 3.3, with both X_t and A_t evolving during rollout. The edge-prediction head is

$$\hat{A}_t = \sigma(H_t Q H_t^\top), \tag{C.2}$$

where H_t contains node representations, Q is the learnable bilinear edge matrix, and σ maps edge scores to $[0, 1]$. We use $\|Q\|_2 = 1.0$ by default and rollout length $T = 32$.

Unlike FE rollouts, DE rollouts activate both directions of node-edge coupling in the joint operator B from Equation (3.4). Across our DE experiments, the estimated spectral radius lies in the range $\rho(B) = 132$–368, compared with 2–50 in the corresponding FE setting. This provides a substantially stronger amplification regime for testing the joint-error analysis in Lemma 2.

C.4 Agent Calling-Tree Testbed

The agent calling-tree testbed represents heterogeneous task execution as a directed acyclic graph with approximately 22–30 nodes per instance. It contains nine node types: *planner*, *validator*, *executor*, *checker*, *aggregator*, *reporter*, *logger*, *error_handler*, and *final_answer*. Six relation types are used: *calls*, *validates*, *reports*, *errors*, *triggers*, and *logs*. Each node is represented by an 8-dimensional execution-status vector.

The primary evaluation metric is sink-node success rate, $\mathrm{sr}_{\mathrm{sink}}$, which measures whether the *final_answer* node reaches a successful terminal state. A random policy obtains $\mathrm{sr}_{\mathrm{sink}} = 0.262$, while an oracle policy exceeds 0.40, providing lower and upper calibration references for planning performance.

C.5 Platform Skill-Graph Testbed

The platform skill-graph testbed represents a heterogeneous skill-management environment with $N = 40$ nodes and eight node types: *skill*, *validator*, *adapter*, *artifact*, *failure_case*, *patch*, *task*, and *tool*. The graph captures dependencies among skills, validation components, adapters, artifacts, tools, and failure-recovery mechanisms, following the broader setting of tool/API and skill-based agent systems (Li et al., 2023; Qin et al., 2024; Liu et al., 2024b; Trivedi et al., 2024; Huang & Huang, 2025; Huang et al., 2025; Xia et al., 2026).

The primary metric is skill success rate, $\mathrm{sr}_{\mathrm{skill}}$. Under a no-op policy, $\mathrm{sr}_{\mathrm{skill}}$ decreases from 1.0 initially to approximately 0.317 at $T = 10$, confirming that skill quality degrades without maintenance actions. As an additional calibration, an R-GCN predictor (Schlichtkrull et al., 2018) evaluated against the ground-truth simulator obtains Pearson $r = 0.231$ and MAE $= 0.029$.

D Error-Aware GWM Details

This section gives the full formulation of Error-Aware GWM introduced in Section 5. The method augments the standard GWM training objective with spectral regularization, rollout consistency, and critical-node weighting. For dynamic-edge models, it additionally regularizes the interaction between node and edge transitions.

D.1 Base GWM Objective

We use the GWM transition and rollout definitions in Equations (2.2)–(2.3). In the FE setting, the adjacency matrix remains fixed and the model predicts only node states. In the DE setting, both node states and edges are predicted as defined in Equation (2.5). We denote the standard supervised prediction objective by $\mathcal{L}_{\mathrm{pred}}$; it contains the node-state prediction loss in FE-GWM and both node- and edge-prediction losses in DE-GWM.

Error-Aware GWM augments this objective as

$$\mathcal{L}_{\mathrm{EA}} = \mathcal{L}_{\mathrm{pred}} + \lambda_{\mathrm{spec}} R_{\mathrm{spec}} + \lambda_{\mathrm{roll}} R_{\mathrm{roll}} + \lambda_{\mathrm{crit}} R_{\mathrm{crit}}, \tag{D.1}$$

where $\lambda_{\mathrm{spec}}, \lambda_{\mathrm{roll}}, \lambda_{\mathrm{crit}} \geq 0$ control the three regularization terms.

D.2 Spectral and Rollout Regularization

Lemma 1 shows that FE rollout amplification depends on the product of transition-layer spectral norms through $\widehat{\mathrm{GEAF}}(G) = \rho(A) \prod_{\ell} \|W_\ell\|_2$. We therefore regularize these norms using

$$R_{\mathrm{spec}} = \sum_{\ell} \|W_\ell\|_2^2. \tag{D.2}$$

This reduces the model-dependent component of $\widehat{\mathrm{GEAF}}$ without modifying the graph topology. The motivation follows standard spectral control of neural-network Lipschitz constants (Miyato et al., 2018). Its effect becomes particularly important when $\rho(A)$ is large, because topology and model amplification enter multiplicatively in the FE bound.

Spectral control limits amplification but does not directly optimize predictions under repeated model application. We therefore additionally train on autoregressive trajectories. For a set of rollout horizons $\mathcal{H}$,

$$R_{\text{roll}} = \sum_{h \in \mathcal{H}} \|\hat{X}^{(h)}_{t+h} - X_{t+h}\|_F^2, \tag{D.3}$$

where $\hat{X}^{(h)}_{t+h}$ is obtained by applying the learned transition recursively for h steps from the state at time t. Unlike the one-step objective, intermediate predictions are fed back into subsequent transitions. The model is therefore optimized under the prediction distribution encountered during rollout, reducing the mismatch between one-step training and long-horizon inference (Bengio et al., 2015; Janner et al., 2019).

D.3 Critical-Node Weighting

Graph topology also makes the effect of prediction error nonuniform across nodes. An error at a structurally influential node can affect more downstream states through subsequent message passing than an error at a peripheral node. We account for this asymmetry with

$$R_{\text{crit}} = \sum_{v \in V} w_v \|\hat{X}_{t+1,v} - X_{t+1,v}\|_2^2, \tag{D.4}$$

where $w_v \geq 0$ is the normalized structural-importance weight of node v, with $\sum_{v \in V} w_v = 1$. The weights can be computed from degree, PageRank, or betweenness centrality and are fixed during training. This term changes the allocation of prediction accuracy across the graph rather than introducing a new transition mechanism: errors on nodes with greater potential to propagate through the graph receive larger penalties.

D.4 Dynamic-Edge Extension

For DE-GWM, both node and edge errors contribute to rollout instability through the joint operator B in Equation (3.4). We therefore apply the same node-state regularizers while including the edge-prediction loss in $\mathcal{L}_{\text{pred}}$. The edge transition uses the bilinear predictor defined in Equation (2.6),

$$\hat{A}_{k+1,ij} = \sigma(h_{k,i}^{\top} Q h_{k,j}), \tag{D.5}$$

where Q is the learnable edge-prediction matrix.

To control the interaction between node and edge transitions, we additionally use

$$R_g = \frac{1}{2}\|W\|_2 \|Q\|_2, \tag{D.6}$$

where W denotes the node-transition weight matrix and Q the edge-prediction weight matrix. The complete DE objective is therefore

$$\mathcal{L}^{\text{DE}}_{\text{EA}} = \mathcal{L}^{\text{DE}}_{\text{pred}} + \lambda_{\text{spec}} R_{\text{spec}} + \lambda_{\text{roll}} R_{\text{roll}} + \lambda_{\text{crit}} R_{\text{crit}} + \lambda_g R_g, \tag{D.7}$$

where $\lambda_g \geq 0$ controls the DE-specific graph regularizer. This term targets the cross-coupling responsible for the off-diagonal components of B, complementing spectral regularization of the node transition.

Together, these components address three distinct mechanisms identified by our analysis: R_{spec} limits transition amplification, R_{roll} reduces error accumulation under autoregressive prediction, and R_{crit} concentrates predictive accuracy on structurally influential nodes. The DE extension further controls node-edge coupling when the graph structure itself evolves.

E RQ1: Does Topology Predict Rollout-Error Growth?

We further investigate **RQ1** through the following additional analyses.

E.1 Rollout Amplification vs. Over-Smoothing

We distinguish GWM rollout amplification from classical GNN over-smoothing. Over-smoothing results from repeated normalized propagation that drives node representations toward a low-variation regime, whereas GWM rollout error is governed by the autoregressive amplification factor $\widehat{\mathrm{GEAF}}(\ell) = \rho(A)^\ell c^\ell$. Figure 5 shows that the normalized propagation operator remains bounded while $\widehat{\mathrm{GEAF}}(\ell)$ increases with message-passing depth and graph spectral radius. Thus, the two capture distinct mechanisms: representation homogenization and autoregressive error amplification.

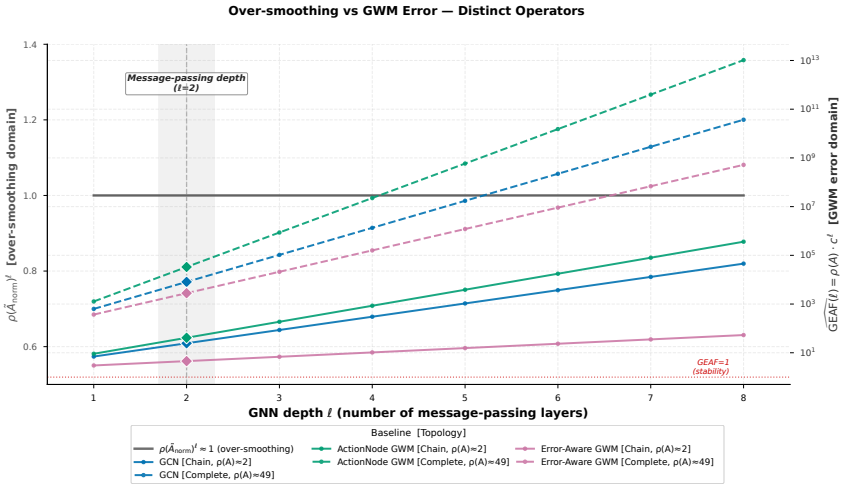


Figure 5: **Rollout amplification vs. over-smoothing.** The normalized propagation operator remains bounded, whereas $\widehat{\mathrm{GEAF}}(\ell)$ increases with message-passing depth and graph spectral radius, distinguishing representation homogenization from autoregressive error amplification.

E.2 Global Spectral Amplification vs. Local Hub Failure

We distinguish global spectral amplification from local hub-mediated failure. Figure 6 shows that complete graphs have the largest $\widehat{\mathrm{GEAF}}$, whereas star graphs exhibit the largest affected-node fraction under hub injection despite having lower $\widehat{\mathrm{GEAF}}$. Thus, $\rho(A)$ captures global spectral amplification, while hub concentration governs the spread of localized perturbations.

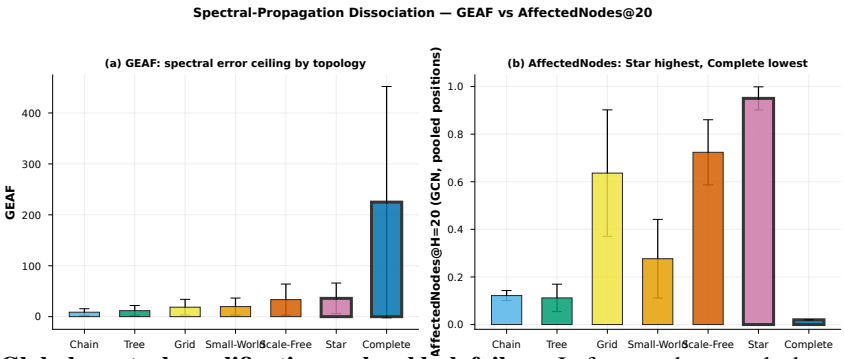


Figure 6: **Global spectral amplification vs. local hub failure.** Left: complete graphs have the largest $\widehat{\mathrm{GEAF}}$. Right: star graphs exhibit the largest affected-node fraction under hub injection, showing that global spectral amplification and local hub-mediated spread are distinct.

E.3 Edge-Density Sweep

We vary graph density using Erdős–Rényi graphs with edge probability $p \in \{0.02, 0.05, 0.10, 0.20\}$ and compare the vanilla GCN world model, ActionNode GWM, and Error-Aware GWM over three

seeds. Figure 7 shows that increasing density raises $\rho(A)$ and substantially increases NodeMSE@32 for vanilla GCN, while ActionNode GWM and Error-Aware GWM remain near the low-error floor. This controlled sweep further supports the role of topology-dependent spectral amplification in rollout stability.

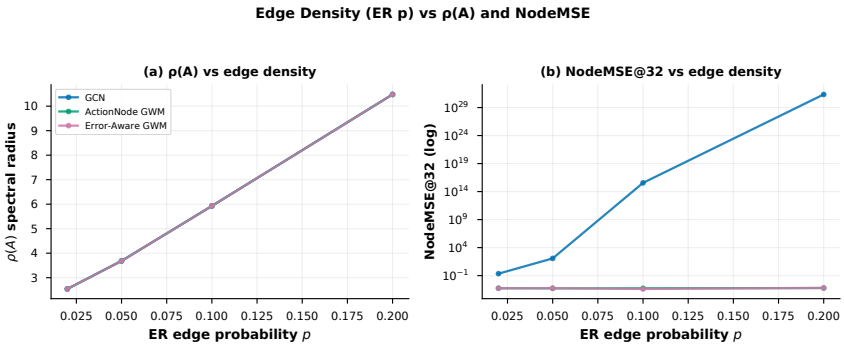


Figure 7: **Edge-density sweep.** Left: adjacency spectral radius $\rho(A)$ versus Erdős–Rényi edge probability p. Right: NodeMSE@32 versus p. Increasing density raises $\rho(A)$ and destabilizes vanilla GCN rollouts, while ActionNode GWM and Error-Aware GWM remain stable. Results are averaged over three seeds.

F RQ3: Does Error-Aware GWM Improve Long-Horizon Stability?

We evaluate whether Error-Aware GWM stabilizes long-horizon rollouts without sacrificing accuracy. This RQ compares the full objective with vanilla GCN rollouts and simpler stabilization methods such as gradient clipping, weight decay, and spectral-norm projection. We also test robustness under graph rewiring, noisy observations, and graph-size scaling.

F.1 Error-Aware GWM Prevents Divergence While Preserving Accuracy

Figure 8 presents the main FE comparison. On high-risk scale-free and star topologies, the vanilla GCN world model shows rapidly increasing error as the horizon grows, while Error-Aware GWM stays near the low-error floor. Across all paired topology-seed cells, the full objective eliminates the divergences observed in the vanilla GCN setting. This result supports the design of Error-Aware GWM: spectral control is most effective when combined with rollout consistency and critical-node weighting, allowing the model to remain both stable and accurate.

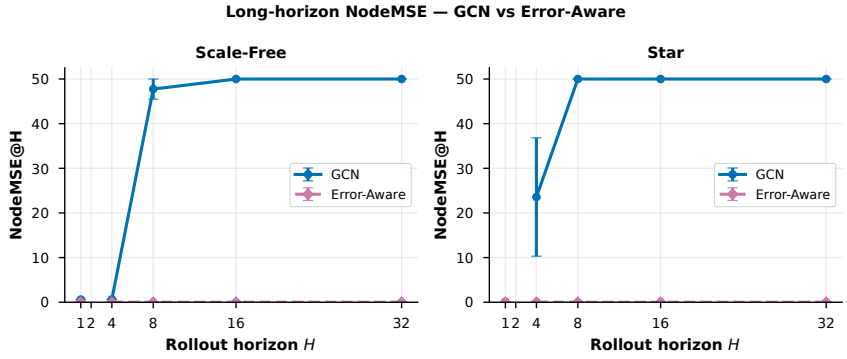


Figure 8: **Error-Aware GWM prevents long-horizon divergence.** NodeMSE@H is shown for the vanilla GCN world model and Error-Aware GWM on scale-free and star topologies. Error-Aware GWM stays near the low-error floor across horizons, while the vanilla model shows rapidly increasing error as the horizon grows.

F.2 Ablation: Stability Alone Is Not Enough

Table 4 shows that simple stabilization methods reduce divergence, but they neither eliminate it nor maintain prediction accuracy. Weight decay and spectral-norm projection reduce divergence to 4/21 cells, yet their median NodeMSE@32 remains orders of magnitude larger than that of Error-Aware GWM. Spectral-norm projection has $1690\times$ higher error than Error-Aware GWM, while weight decay approaches a nearly constant predictor. The advantage of Error-Aware GWM is therefore not only lower divergence, but the combination of stability and prediction accuracy.

Table 4: Regularization ablation over 21 FE cells. Simple stabilization methods reduce divergence but leave large long-horizon prediction errors; Error-Aware GWM is the only variant that combines zero divergence with low NodeMSE@32, showing that stability must be maintained together with prediction accuracy.

Variant	Regularizer	Diverged	Median NodeMSE@32	Ratio vs. Error-Aware GWM
Vanilla GCN world model	none	10/21	0.995	$6934\times$
GCN + gradient clipping	gradient clipping	8/21	0.987	$5790\times$
GCN + weight decay	weight decay	4/21	1.091	$4172\times$
GCN + spectral-norm projection	spectral norm projection	4/21	0.364	$1690\times$
Error-Aware GWM	$R_{\text{spec}} + R_{\text{roll}} + R_{\text{crit}}$	**0/21**	$\mathbf{2.67 \times 10^{-4}}$	$\mathbf{1}\times$

F.3 Robustness to Topology Perturbation and Graph Size

Graph Rewiring Robustness. We test whether rollout stability is robust to changes in graph topology by applying six rewiring operations to scale-free graphs: identity, PageRank-based edge removal, random edge addition, random edge flipping, sparsification, and densification. We compare the vanilla GCN world model, ActionNode GWM, and Error-Aware GWM over three seeds. Figure 9 shows that vanilla GCN remains in a high-error regime across all rewiring operations. ActionNode GWM is stable under most changes but has higher error under densification and random edge addition. In contrast, Error-Aware GWM remains stable across all rewiring operations, with NodeMSE@$H = 20$ between approximately 2.0×10^{-4} and 5.0×10^{-4}. These results show that Error-Aware GWM remains stable under topology perturbations, consistent with the role of spectral regularization in controlling topology-dependent error amplification.

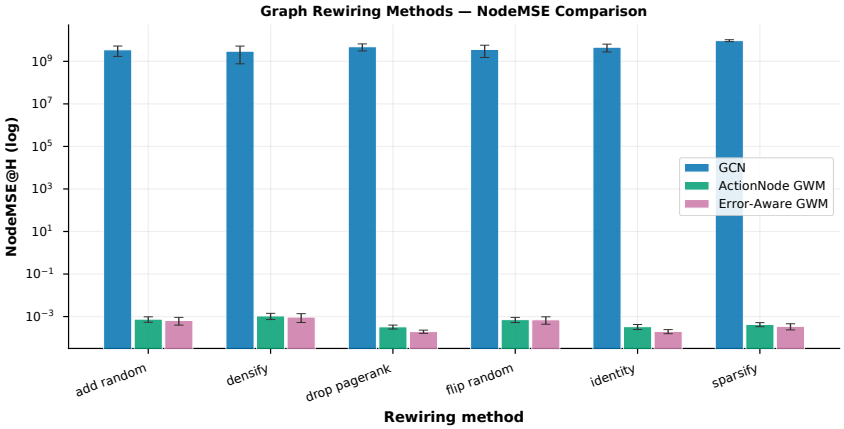


Figure 9: **Robustness to graph rewiring.** NodeMSE@$H = 20$ under six rewiring operations on scale-free graphs across three world-model variants and three seeds. Error-Aware GWM remains in the low-error regime across all rewiring operations, while vanilla GCN remains unstable and ActionNode GWM is more sensitive to increased connectivity.

Graph Size Scaling. We next test whether the observed stability patterns persist as graph size increases. We evaluate $N \in \{20, 50, 100, 200, 500\}$ across five topologies, six baselines, and three seeds. Figure 10 reports $\widehat{\text{GEAF}}$ and NodeMSE@32 as functions of N. ActionNode GWM, GPS, and Error-Aware GWM remain near the low-error regime across graph sizes, whereas vanilla GCN continues to produce large rollout errors in high-risk settings. MPNN encounters out-of-memory

failures at larger graph sizes because of its dense message tensors. Thus, the stability patterns observed at $N = 50$ continue across a substantially wider range of graph sizes.

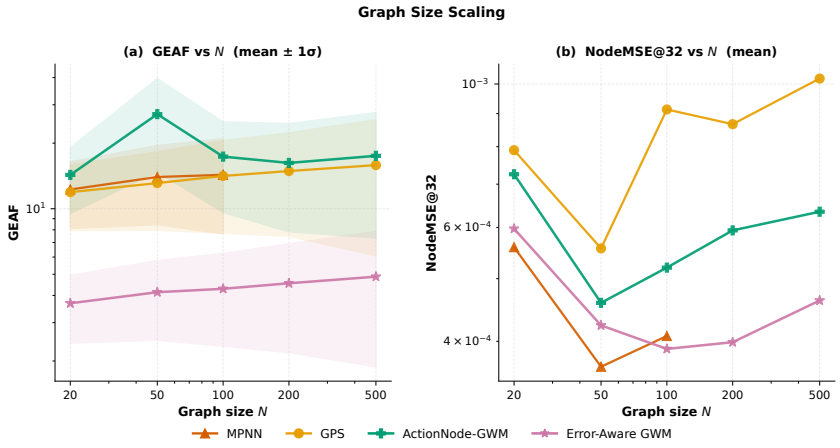


Figure 10: **Graph size scaling.** Left: $\widehat{\mathrm{GEAF}}$ as a function of graph size N. Right: NodeMSE@32 as a function of N. Curves show means across seeds, with shaded regions showing variation across seeds or runs. ActionNode GWM, GPS, and Error-Aware GWM remain in the low-error regime as graph size increases, while vanilla GCN remains unstable in high-risk settings.

G RQ4: How Do GWMs Behave on Heterogeneous Agent Graphs?

We evaluate GWMs on heterogeneous agent and skill graphs, where nodes represent different roles such as planners, executors, validators, tools, and failure handlers. We study whether R-GCN rollouts match the ground-truth simulators, whether rollout failures depend on node roles, whether targeted correction can reduce errors, and how task success changes with rollout horizon.

G.1 Calibration Against Heterogeneous Simulators

We first compare R-GCN rollout predictions with the ground-truth simulators on the agent calling-tree and platform skill-graph testbeds. As shown in Figure 11, on the agent calling-tree testbed, the R-GCN mean is 0.442 versus a ground-truth mean of 0.437, with Pearson $r = 0.311$ and MAE $= 0.051$. On the skill-graph testbed, the corresponding means are 0.387 and 0.406, with Pearson $r = 0.231$ and MAE $= 0.029$. The predictions are close to the simulator outputs in mean and absolute error, but the instance-level correlations remain limited. We therefore use these R-GCN models to study rollout behavior on heterogeneous graphs rather than as precise replacements for the ground-truth simulators.

G.2 Role-Dependent Failure Sensitivity

We next test whether different functional roles contribute differently to rollout quality. We mask role-specific subgraphs in the agent calling-tree testbed and measure the resulting success-rate drop and trajectory-level NodeMSE. Figure 12 shows that masking the full graph causes the largest degradation. Among partial masks, removing hub- and validator-related components has a larger effect than removing leaf nodes, while leaf masking has little effect. This shows that structurally or functionally important nodes contribute more strongly to rollout quality.

We further study how errors propagate when they originate from different roles. We inject perturbations into planner, validator, hub, bridge, and random nodes across 20 instances. Figure 13 reports failure propagation depth (FPD) and the number of error nodes at the final timestep. Planner and validator injections have a median FPD of about 3, while bridge and hub injections have smaller median FPD. Random injection has the largest median FPD of 4. Final error-node counts are similar on average, but their distributions differ across roles. These results show that error propagation depends on functional role rather than graph position alone.

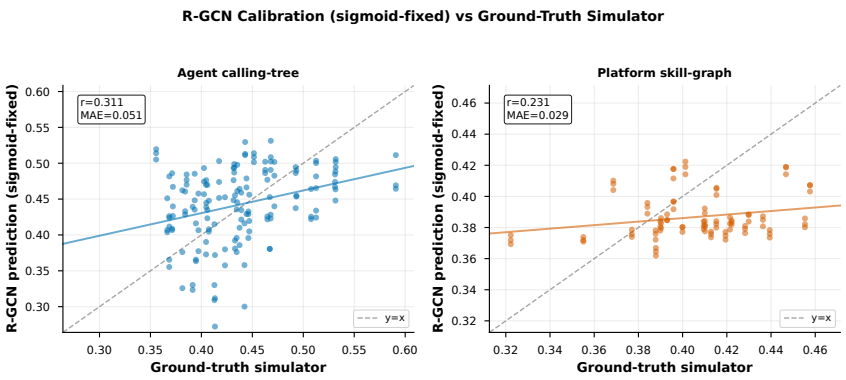


Figure 11: **R-GCN calibration against heterogeneous simulators.** Left: agent calling-tree testbed. Right: platform skill-graph testbed. R-GCN predictions are close to the simulator outputs in mean and absolute error, while instance-level correlations remain limited.

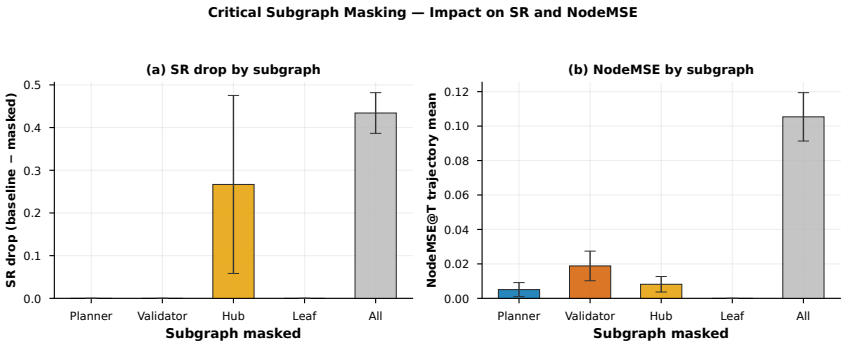


Figure 12: **Role-specific subgraph masking on the agent calling-tree testbed.** Left: success-rate drop after masking different subgraphs. Right: NodeMSE@T under the same masking conditions. Full-graph masking causes the largest degradation, while leaf masking has little effect.

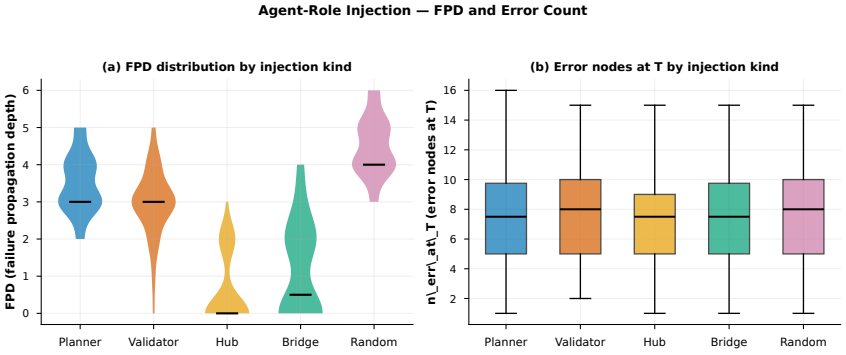


Figure 13: **Error injection across agent roles.** Left: failure propagation depth by injection role. Right: number of error nodes at the final timestep. Different roles produce different propagation patterns in the calling tree.

G.3 Correction Policies

We next test whether a local GEAF score can guide node-level correction. In the FE setting,

$$\widehat{\mathrm{GEAF}}_{\mathrm{local}}(v) = \mathrm{degree}(v) \prod_{\ell} \|W_\ell\|_2. \tag{G.1}$$

Because $\prod_\ell \|W_\ell\|_2$ is constant across nodes for a fixed model, $\widehat{\mathrm{GEAF}}_{\mathrm{local}}(v)$ gives the same node ranking as degree centrality. Figure 15a confirms this result on the agent calling-tree testbed: GEAF-proxy and degree correction both achieve a mean error reduction of 12.9%, outperforming random correction but remaining below the oracle policy at 22.4%. Thus, local GEAF provides a useful structural correction rule, but in the FE setting its ranking is equivalent to degree centrality.

We also compare correction policies more broadly across six baselines, six topologies, and three seeds under the same 10% correction budget. Figure 14 shows that oracle correction provides the largest NodeMSE reduction. Among practical policies, uncertainty-based correction gives the largest average improvement over random correction, while GEAF-proxy correction improves over random but matches degree-based correction. This is consistent with Equation (G.1): for fixed model weights, local GEAF and degree induce the same ranking.

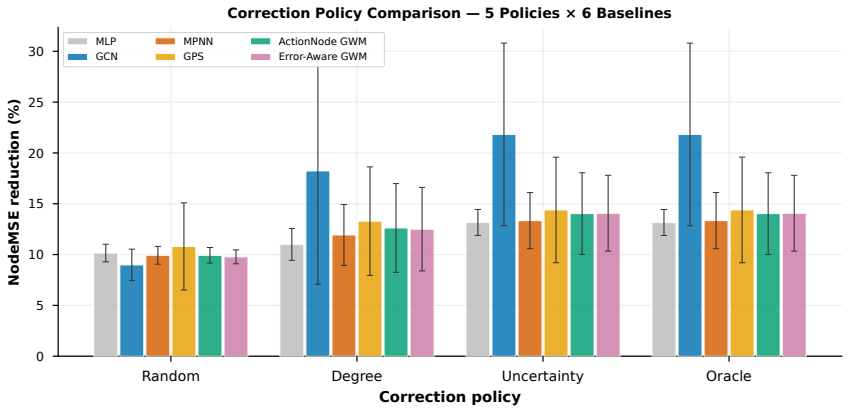
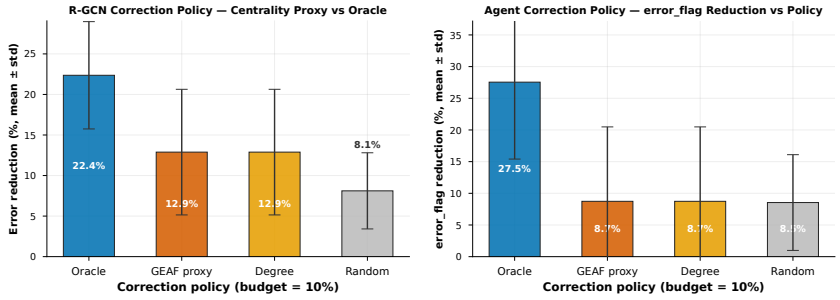


Figure 14: **Node correction policy comparison.** NodeMSE reduction under a 10% correction budget across six baselines, six topologies, and three seeds. Oracle correction gives the largest reduction, uncertainty is the strongest practical policy, and GEAF-proxy matches degree-based correction in the FE setting.

Finally, we evaluate the same correction idea directly on the agent calling-tree testbed. Figure 15b compares random, degree, GEAF-proxy, and oracle correction under a 10% node-correction budget. Oracle correction gives the largest reduction in the mean error flag, while GEAF-proxy and degree remain tied because they select the same nodes. Together, these experiments show that local GEAF provides a useful non-random correction rule in FE agent graphs, but does not add information beyond degree centrality in this setting.



(a) Local GEAF correction.

(b) Agent-level correction.

Figure 15: **Correction policies on heterogeneous agent graphs.** (a) Local GEAF and degree-based correction give the same ranking in the FE setting because the local GEAF score is degree multiplied by a model-wide constant. (b) Under a 10% node-correction budget, oracle correction gives the largest reduction in mean error flag, while GEAF-proxy and degree correction remain tied because they induce the same FE node ranking.

G.4 Long-Horizon Agent Task Completion

We finally evaluate task completion on the agent calling-tree and platform skill-graph testbeds. We pool 160 instances per testbed and report task success rate (SR) at horizons $H \in \{1, 5, 10, 20\}$. Results are averaged over six testbed-seed settings for each method and horizon. As shown in Table 5, all methods achieve SR around 0.48–0.49 at $H = 1$, which decreases to approximately 0.41–0.42 at longer horizons. Error-Aware GWM remains competitive but does not remove this decline, showing that long-horizon task completion remains difficult on heterogeneous agent graphs.

Table 5: Agent task completion across rollout horizons. All methods decline from approximately 0.49 at $H = 1$ to approximately 0.42 for $H \geq 5$, showing the difficulty of long-horizon planning on heterogeneous agent graphs.

Baseline	SR@1	SR@5	SR@10	SR@20
MLP world model	0.477 ± 0.039	0.420 ± 0.029	0.422 ± 0.033	0.423 ± 0.033
GCN world model	0.487 ± 0.069	0.415 ± 0.025	0.415 ± 0.025	0.416 ± 0.025
MPNN world model	0.492 ± 0.084	0.411 ± 0.028	0.411 ± 0.026	0.413 ± 0.023
Error-Aware GWM (ours)	0.491 ± 0.080	0.420 ± 0.030	0.417 ± 0.028	0.417 ± 0.025

H RQ5: What is the Real-World Scope of GWMs

Finally, we evaluate GWMs on standard real-world graph benchmarks to examine their practical scope relative to task-specific graph models. We consider static node classification and temporal link prediction, where GWMs are not specifically designed to replace specialized architectures.

H.1 Static Node Classification

We evaluate Cora and Citeseer (Yang et al., 2016) using their standard public splits. Cora contains 2,708 nodes, 5,278 edges, and 7 classes, while Citeseer contains 3,327 nodes, 4,732 edges, and 6 classes. We compare FE-GWM and Error-Aware GWM with GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017). All models are trained for 200 epochs over three seeds.

Table 6: Test accuracy (%) on Cora and Citeseer, reported as mean $\pm$ standard deviation over three seeds.

Method	Cora	Citeseer
GCN (Kipf & Welling, 2017)	$\mathbf{80.6 \pm 1.0}$	65.6 ± 2.4
GAT (Veličković et al., 2018)	79.3 ± 0.4	$\mathbf{68.1 \pm 0.8}$
GraphSAGE (Hamilton et al., 2017)	78.8 ± 0.6	66.5 ± 1.1
FE-GWM (ours)	73.5 ± 1.6	66.2 ± 2.7
Error-Aware GWM (ours)	75.0 ± 1.6	66.9 ± 1.8

Table 6 shows that specialized classifiers remain stronger on Cora, whereas the gap is smaller on Citeseer. Error-Aware GWM improves over FE-GWM on both datasets, indicating a modest benefit from error-aware regularization even in the static setting.

H.2 Temporal Link Prediction

We evaluate Bitcoin-Alpha (Kumar et al., 2016), a temporal trust network with 3,783 nodes and 24,186 edges, using 60-day windows. We compare GWM variants with a JODIE-style temporal model (Kumar et al., 2019) and Static GCN (Kipf & Welling, 2017), reporting AP, AUC, and MRR over three seeds.

Table 7 shows a mixed result: FE-GWM achieves the highest AP, Static GCN the highest AUC, and JODIE-style the highest MRR. DE-GWM does not consistently improve over FE-GWM, indicating that evolving-edge modeling is most useful when edge dynamics provide sufficiently predictive rollout signal.

Table 7: Temporal link prediction on Bitcoin-Alpha, reported as mean ± standard deviation over three seeds.

Method	AP	AUC	MRR
JODIE-style (Kumar et al., 2019)	0.772 ± 0.009	0.646 ± 0.009	**0.261 ± 0.104**
Static GCN (Kipf & Welling, 2017)	0.815 ± 0.014	**0.774 ± 0.030**	0.109 ± 0.034
FE-GWM (ours)	**0.820 ± 0.016**	0.740 ± 0.030	0.166 ± 0.056
DE-GWM (ours)	0.805 ± 0.004	0.717 ± 0.009	0.122 ± 0.037
Error-Aware DE-GWM (ours)	0.805 ± 0.006	0.713 ± 0.009	0.134 ± 0.041

I Additional Experiments

This section provides additional analyses of seed stability, multi-hop error propagation, message-passing architectures, and temporal memory.

I.1 Seed Stability of Error Metrics

We first examine whether the observed rollout-error patterns depend strongly on random seeds. Figure 16 reports seed-level distributions of NodeMSE@32 and GrowthSlope across the seven topology families. The distributions remain stable across seeds, with particularly small variation on deterministic topologies. This supports the three-seed protocol used in the main experiments and indicates that the observed topology-dependent trends are not driven by isolated graph instances.

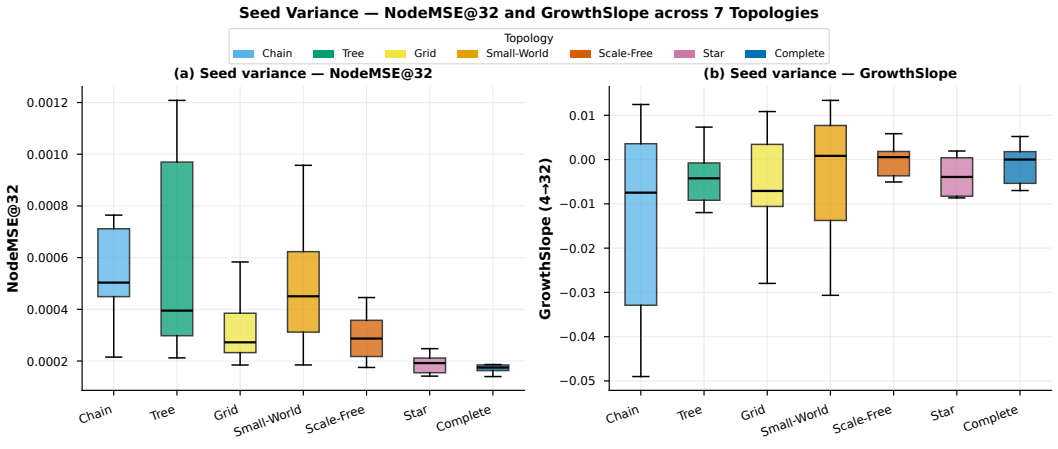


Figure 16: **Seed stability across topologies.** Left: seed-level distributions of NodeMSE@32. Right: seed-level distributions of GrowthSlope. The distributions remain stable across seeds, supporting the three-seed protocol used in the main experiments.

I.2 Multi-Hop Error Propagation Under FE Dynamics

We test whether local perturbations propagate across multiple graph hops by injecting perturbations at depths $\{1, 2, 4, 6, 8, 10\}$ on chain and scale-free graphs across six baselines and three seeds. Figure 17(a) shows that propagation remains limited for non-diverged models. Chain graphs remain below NodeMSE@$H = 20$ of 10^{-12} across injection depths, while non-diverged scale-free runs remain below 10^{-13}. Severe propagation appears mainly in already-diverged GCN runs, supporting the contractive interpretation of the FE regime.

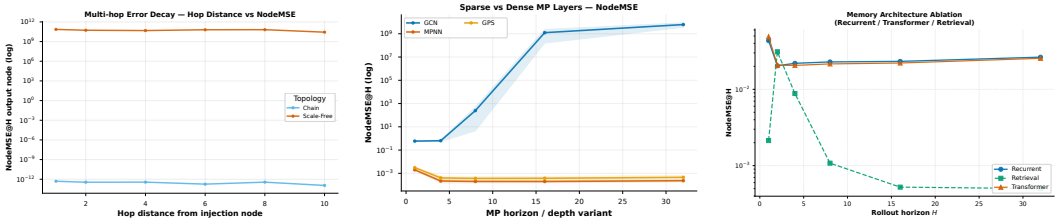
I.3 Sparse and Dense Message-Passing Layers

We compare the vanilla GCN world model, MPNN world model, and GPS-style graph transformer on the scale-free topology over horizons $H \in \{1, 4, 8, 16, 32\}$ and three seeds. Figure 17(b) shows that GCN diverges with horizon, reaching a median NodeMSE@32 of 6.5×10^{9}, whereas MPNN and GPS remain near the low-error floor at 2.2×10^{-4} and 4.0×10^{-4}, respectively. Thus, the observed

instability is not explained by message-passing density alone, but depends on the interaction between the transition operator and graph topology.

I.4 Temporal Memory Variants

We compare Markov, last-2, and last-4 history variants, together with recurrent, transformer, and retrieval-augmented memory architectures. Figure 17(c) shows that additional temporal context reduces long-horizon error, with most of the gain obtained from the last-2 context. Among the extended variants, retrieval-augmented memory achieves the lowest long-horizon error, indicating that temporal context can further improve rollout stability.



(a) Multi-hop propagation (b) Message passing (c) Temporal memory

Figure 17: **Additional rollout analyses.** (a) Multi-hop perturbations remain limited under contractive FE dynamics except in already-diverged runs. (b) GCN diverges on scale-free graphs while MPNN and GPS remain stable, showing that message-passing density alone does not explain instability. (c) Temporal context improves long-horizon prediction, with retrieval-augmented memory providing the strongest reduction among the evaluated memory variants.

J Proofs and Mathematical Background

J.1 Proof of Lemma 1

Proof. Since the graph is fixed in the FE regime, $A_t \equiv A$ for all rollout steps. Let $F_\theta^X(\cdot, A, a)$ and $F^{X\star}(\cdot, A, a)$ denote the learned and true node-transition maps, respectively. By Assumption 2,

$$\|F_\theta^X(X, A, a) - F^{X\star}(X, A, a)\|_F \le \epsilon_X \tag{J.1}$$

for every (X, A, a) in the compact state space.

Let $e_k^X := \|\hat{X}_k - X_k\|_F$. Adding and subtracting $F_\theta^X(X_k, A, a_k)$, applying the triangle inequality, and then using Equation (J.1) give

$$\begin{aligned} e_{k+1}^X &= \|F_\theta^X(\hat{X}_k, A, a_k) - F^{X\star}(X_k, A, a_k)\|_F \\ &= \big\|F_\theta^X(\hat{X}_k, A, a_k) - F_\theta^X(X_k, A, a_k) + F_\theta^X(X_k, A, a_k) - F^{X\star}(X_k, A, a_k)\big\|_F \\ &\le \|F_\theta^X(\hat{X}_k, A, a_k) - F_\theta^X(X_k, A, a_k)\|_F + \|F_\theta^X(X_k, A, a_k) - F^{X\star}(X_k, A, a_k)\|_F \\ &\le \|F_\theta^X(\hat{X}_k, A, a_k) - F_\theta^X(X_k, A, a_k)\|_F + \epsilon_X. \end{aligned} \tag{J.2}$$

Under Assumption 3, repeated application of the coordinate-wise L_σ-Lipschitz property of σ and submultiplicativity of the spectral and Frobenius norms yields

$$\begin{aligned} \|F_\theta^X(\hat{X}_k, A, a_k) - F_\theta^X(X_k, A, a_k)\|_F &\le L_\sigma \|A\|_2 \prod_\ell \|W_\ell\|_2 \|\hat{X}_k - X_k\|_F \\ &= L_\sigma \rho(A) \prod_\ell \|W_\ell\|_2 \, e_k^X \\ &=: L_X e_k^X, \end{aligned} \tag{J.3}$$

where the second equality uses $\|A\|_2 = \rho(A)$ for symmetric A, and

$$L_X := L_\sigma \rho(A) \prod_\ell \|W_\ell\|_2. \tag{J.4}$$

Substituting Equation (J.3) into Equation (J.2) gives

$$e_{k+1}^X \le L_X e_k^X + \epsilon_X. \tag{J.5}$$

Repeatedly applying Equation (J.5) gives

$$\begin{aligned} e_k^X &\le L_X e_{k-1}^X + \epsilon_X \\ &\le L_X\big(L_X e_{k-2}^X + \epsilon_X\big) + \epsilon_X \\ &\le L_X^2 e_{k-2}^X + \epsilon_X(1 + L_X) \\ &\vdots \\ &\le L_X^k e_0^X + \epsilon_X \sum_{i=0}^{k-1} L_X^i \\ &= \begin{cases} L_X^k e_0^X + \epsilon_X \dfrac{L_X^k - 1}{L_X - 1}, & L_X \neq 1, \\ e_0^X + k\epsilon_X, & L_X = 1. \end{cases} \end{aligned} \tag{J.6}$$

Equation (J.6) is precisely the claimed FE rollout-error bound, completing the proof. □

J.2 Proof of Lemma 2

Proof. By assumption,

$$\begin{aligned} e_{k+1}^X &\le L_X e_k^X + L_A e_k^A + \epsilon_X, \\ e_{k+1}^A &\le M_X e_k^X + M_A e_k^A + \epsilon_A. \end{aligned} \tag{J.7}$$

Define

$$u_k := \begin{pmatrix} e_k^X \\ e_k^A \end{pmatrix}, \qquad B := \begin{pmatrix} L_X & L_A \\ M_X & M_A \end{pmatrix}, \qquad \epsilon := \begin{pmatrix} \epsilon_X \\ \epsilon_A \end{pmatrix}. \tag{J.8}$$

Then

$$\begin{aligned} u_{k+1} &= \begin{pmatrix} e_{k+1}^X \\ e_{k+1}^A \end{pmatrix} \\ &\preceq \begin{pmatrix} L_X e_k^X + L_A e_k^A + \epsilon_X \\ M_X e_k^X + M_A e_k^A + \epsilon_A \end{pmatrix} \\ &= \begin{pmatrix} L_X & L_A \\ M_X & M_A \end{pmatrix} \begin{pmatrix} e_k^X \\ e_k^A \end{pmatrix} + \begin{pmatrix} \epsilon_X \\ \epsilon_A \end{pmatrix} \\ &= B u_k + \epsilon. \end{aligned} \tag{J.9}$$

Equation (J.9) proves Equation (3.4).

For the eigenvalues of B,

$$\begin{aligned} 0 &= \det(\lambda I - B) \\ &= \det \begin{pmatrix} \lambda - L_X & -L_A \\ -M_X & \lambda - M_A \end{pmatrix} \\ &= (\lambda - L_X)(\lambda - M_A) - L_A M_X \\ &= \lambda^2 - (L_X + M_A)\lambda + (L_X M_A - L_A M_X). \end{aligned} \tag{J.10}$$

Hence,

$$\begin{aligned} \lambda_\pm &= \frac{L_X + M_A \pm \sqrt{(L_X + M_A)^2 - 4(L_X M_A - L_A M_X)}}{2} \\ &= \frac{L_X + M_A \pm \sqrt{L_X^2 + 2L_X M_A + M_A^2 - 4L_X M_A + 4L_A M_X}}{2} \\ &= \frac{L_X + M_A \pm \sqrt{L_X^2 - 2L_X M_A + M_A^2 + 4L_A M_X}}{2} \\ &= \frac{1}{2}\left[L_X + M_A \pm \sqrt{(L_X - M_A)^2 + 4L_A M_X}\right]. \end{aligned} \tag{J.11}$$

Since B is nonnegative, Perron–Frobenius implies

$$\begin{aligned}\rho(B) &= \lambda_+ \\ &= \frac{1}{2}\left[L_X + M_A + \sqrt{(L_X - M_A)^2 + 4L_A M_X}\right].\end{aligned} \tag{J.12}$$

Equation (J.12) proves Equation (3.5).

If $L_A M_X > 0$, then

$$\begin{aligned}(L_X - M_A)^2 + 4L_A M_X &> (L_X - M_A)^2, \\ \sqrt{(L_X - M_A)^2 + 4L_A M_X} &> |L_X - M_A|.\end{aligned} \tag{J.13}$$

Using Equations (J.12) and (J.13),

$$\begin{aligned}\rho(B) &= \frac{1}{2}\left[L_X + M_A + \sqrt{(L_X - M_A)^2 + 4L_A M_X}\right] \\ &> \frac{1}{2}\left[L_X + M_A + |L_X - M_A|\right] \\ &= \max\{L_X, M_A\}.\end{aligned} \tag{J.14}$$

This proves Lemma 2. □

J.3 Proof of Lemma 3

Proof. Since B is nonnegative, the Gershgorin circle theorem gives

$$\rho(B) \le \max\{L_X + L_A, M_X + M_A\}. \tag{J.15}$$

In the signal-dominant regime, the first-row error term satisfies

$$\begin{aligned}L_X + L_A &\le \widehat{\mathrm{GEAF}}(G) + \widehat{\mathrm{GEAF}}(G)\frac{R_X}{\|A\|_2} \\ &= \widehat{\mathrm{GEAF}}(G)\left(1 + \frac{R_X}{\|A\|_2}\right).\end{aligned} \tag{J.16}$$

Under the same condition, $M_X + M_A$ is no larger than the bound in Equation (J.16). Therefore, using Equations (J.15) and (J.16) together with $\|A\|_2 \ge R_X$,

$$\begin{aligned}\rho(B) &\le \max\{L_X + L_A, M_X + M_A\} \\ &\le \widehat{\mathrm{GEAF}}(G)\left(1 + \frac{R_X}{\|A\|_2}\right) \\ &= 2\widehat{\mathrm{GEAF}}(G).\end{aligned} \tag{J.17}$$

Equation (J.17) proves Equation (3.6).

Finally, when the model weights are fixed across topologies, let $C_W := \prod_\ell \|W_\ell\|_2$, which is constant across graphs. Then, for any two topologies G_i and G_j with adjacency matrices A_i and A_j,

$$\begin{aligned}\widehat{\mathrm{GEAF}}(G_j) - \widehat{\mathrm{GEAF}}(G_i) &= \rho(A_j)C_W - \rho(A_i)C_W \\ &= C_W\big(\rho(A_j) - \rho(A_i)\big).\end{aligned} \tag{J.18}$$

Since $C_W \ge 0$, Equation (J.18) shows that the sign of $\widehat{\mathrm{GEAF}}(G_j) - \widehat{\mathrm{GEAF}}(G_i)$ is determined by the sign of $\rho(A_j) - \rho(A_i)$. Hence, $\widehat{\mathrm{GEAF}}(G)$ is monotone in $\rho(A)$ and induces the same topology ranking, completing the proof. □

J.4 Proof of Lemma 4

Proof. Let π^* be the policy selected by planning with the true dynamics and let $\hat{\pi}_\theta$ be the policy selected by planning with the learned GWM. By the simulation lemma,

$$J^*(\pi^*) - J^*(\hat{\pi}_\theta) \le 2\sum_{k=0}^{H-1} \gamma^k \left| R^\star(G_k, a_k) - R_\theta(\hat{G}_k, a_k) \right|. \tag{J.19}$$

Adding and subtracting $R^\star(\hat{G}_k, a_k)$ and applying Assumption 5,

$$\begin{aligned}\left|R^\star(G_k, a_k) - R_\theta(\hat{G}_k, a_k)\right| &\le \left|R^\star(G_k, a_k) - R^\star(\hat{G}_k, a_k)\right| + \left|R^\star(\hat{G}_k, a_k) - R_\theta(\hat{G}_k, a_k)\right| \\ &\le L_R d_G(G_k, \hat{G}_k) + \epsilon_R \\ &= L_R c^\top u_k + \epsilon_R \\ &\le L_R \|c\|_2 \|u_k\|_2 + \epsilon_R,\end{aligned} \tag{J.20}$$

From Lemma 2,

$$u_k \preceq B^k u_0 + \sum_{i=0}^{k-1} B^i \epsilon. \tag{J.21}$$

For rollouts initialized from the observed graph, $u_0 = 0$. Let $\tilde{\epsilon}$ be a uniform scale for the one-step error vector and let $\kappa(B)$ denote the matrix-dependent constant satisfying $\|B^i\|_2 \le \kappa(B)\rho(B)^i$. Equation (J.21) then gives

$$\begin{aligned}\|u_k\|_2 &\le \sum_{i=0}^{k-1} \|B^i \epsilon\|_2 \\ &\le \sum_{i=0}^{k-1} \|B^i\|_2 \|\epsilon\|_2 \\ &\le \kappa(B)\tilde{\epsilon} \sum_{i=0}^{k-1} \rho(B)^i.\end{aligned} \tag{J.22}$$

Substituting Equations (J.20) and (J.22) into Equation (J.19),

$$\begin{aligned}J^*(\pi^*) - J^*(\hat{\pi}_\theta) &\le 2 \sum_{k=0}^{H-1} \gamma^k \left(L_R \|c\|_2 \|u_k\|_2 + \epsilon_R\right) \\ &\le 2L_R \|c\|_2 \kappa(B)\tilde{\epsilon} \sum_{k=0}^{H-1} \gamma^k \sum_{i=0}^{k-1} \rho(B)^i + 2\epsilon_R \sum_{k=0}^{H-1} \gamma^k \\ &= 2L_R \|c\|_2 \kappa(B)\tilde{\epsilon} \sum_{k=0}^{H-1} \gamma^k \sum_{i=0}^{k-1} \rho(B)^i + 2\epsilon_R \frac{1-\gamma^H}{1-\gamma}.\end{aligned} \tag{J.23}$$

Writing $\rho = \rho(B)$, the rollout-error term in Equation (J.23) satisfies

$$\begin{aligned}\sum_{k=0}^{H-1} \gamma^k \sum_{i=0}^{k-1} \rho^i &= \sum_{k=0}^{H-1} \gamma^k \frac{\rho^k - 1}{\rho - 1} \\ &= \frac{1}{\rho - 1} \sum_{k=0}^{H-1} \left((\gamma\rho)^k - \gamma^k\right) \\ &= \frac{1}{\rho - 1} \left(\sum_{k=0}^{H-1} (\gamma\rho)^k - \sum_{k=0}^{H-1} \gamma^k\right) \\ &= \frac{1}{\rho - 1} \left(\frac{1 - (\gamma\rho)^H}{1 - \gamma\rho} - \frac{1-\gamma^H}{1-\gamma}\right) \\ &= \Phi_H(\gamma, \rho).\end{aligned} \tag{J.24}$$

Using Equation (J.24) with $\rho = \rho(B)$ in Equation (J.23),

$$J^*(\pi^*) - J^*(\hat{\pi}_\theta) \le 2L_R \|c\|_2 \kappa(B)\tilde{\epsilon}\Phi_H(\gamma, \rho(B)) + 2\epsilon_R \frac{1-\gamma^H}{1-\gamma}. \tag{J.25}$$

Finally, when $\gamma\rho(B) > 1$, Equation (J.24) contains the exponentially increasing factor $(\gamma\rho(B))^H$, so the rollout-error contribution grows faster than linearly with the planning horizon H. This proves Lemma 4. □

J.5 Proof of Corollary 5

Proof. From Lemma 1, the FE rollout error satisfies

$$\begin{aligned} e_k^X &\le L_X^k e_0^X + \epsilon_X \frac{L_X^k - 1}{L_X - 1} \\ &= L_X^k \left(e_0^X + \frac{\epsilon_X}{L_X - 1} \right) - \frac{\epsilon_X}{L_X - 1}. \end{aligned} \tag{J.26}$$

When the rollout is not dominated by the constant error floor, define $C := e_0^X + \epsilon_X/(L_X - 1)$. The leading k-dependence in Equation (J.26) is then

$$\begin{aligned} e_k^X &\approx C L_X^k \\ \log e_k^X &\approx \log \left(C L_X^k \right) \\ &= \log C + \log L_X^k \\ &= \log C + k \log L_X. \end{aligned} \tag{J.27}$$

Taking the slope with respect to the rollout step k in Equation (J.27) gives

$$\begin{aligned} \frac{\partial \log e_k^X}{\partial k} &\approx \frac{\partial}{\partial k} \left(\log C + k \log L_X \right) \\ &= \log L_X, \end{aligned} \tag{J.28}$$

and hence $\partial \log e_k^X / \partial k \to \log L_X$ in the amplification-dominated regime. This proves Corollary 5. □

K Auxiliary latent world-model bounds

K.1 Latent world-model setup

This appendix records the scalar latent-world-model bounds that underlie the graph-valued results in the main text. They are not a replacement for Lemmas 1–4; rather, they show how the familiar compounding-error argument for latent dynamics (Hafner et al., 2018; Chua et al., 2018; Janner et al., 2019) becomes the graph-specific operator analysis used in Section 3. We consider a sequential decision-making setting in which an agent interacts with a partially observed environment over discrete time steps. At each time step t, the agent receives an observation $o_t \in \mathcal{O}$, selects an action $a_t \in \mathcal{A}$, and receives a scalar reward $r_t \in \mathbb{R}$. Since the true environment state is generally not directly observable, the agent must infer a compact latent state from its interaction history. Let

$$h_t = (o_1, a_1, o_2, a_2, \ldots, o_t) \tag{K.1}$$

denote the observable history up to time t. A latent world model learns an internal state representation

$$z_t = e_\theta(h_t), \tag{K.2}$$

where $z_t \in \mathcal{Z}$ is a latent state and e_θ is an encoder parameterized by θ.

The purpose of the latent world model is not only to encode the current observation, but also to predict how the latent state evolves under actions. We define a latent world model as

$$\mathcal{W}_\theta = (e_\theta, p_\theta(z_{t+1} \mid z_t, a_t), p_\theta(o_t \mid z_t), p_\theta(r_t \mid z_t, a_t)), \tag{K.3}$$

where e_θ maps observation histories to latent states, $p_\theta(z_{t+1} \mid z_t, a_t)$ is the latent transition model, $p_\theta(o_t \mid z_t)$ is the observation reconstruction or prediction model, and $p_\theta(r_t \mid z_t, a_t)$ is the reward prediction model.

Given a trajectory

$$\tau = (o_1, a_1, r_1, o_2, a_2, r_2, \ldots, o_T), \tag{K.4}$$

the latent world model induces the following factorization:

$$p_\theta(o_{1:T}, r_{1:T}, z_{1:T} \mid a_{1:T}) = p_\theta(z_1 \mid o_1) \prod_{t=1}^{T} p_\theta(o_t \mid z_t) p_\theta(r_t \mid z_t, a_t) p_\theta(z_{t+1} \mid z_t, a_t). \tag{K.5}$$

This factorization shows that the model compresses the observable history into latent states while preserving the information needed to predict future observations and rewards.

The learning objective is to fit the latent world model to trajectories collected from the environment. A common training objective is the negative log-likelihood

$$\mathcal{L}_{\mathrm{WM}}(\theta) = -\sum_{t=1}^{T} \Big[\log p_\theta(o_t \mid z_t) + \log p_\theta(r_t \mid z_t, a_t) + \log p_\theta(z_{t+1} \mid z_t, a_t) \Big], \tag{K.6}$$

where $z_t = e_\theta(h_t)$. In variational latent world models, the latent transition term can be replaced by an evidence lower bound that includes an inference model $q_\theta(z_t \mid h_t)$ and a prior transition model $p_\theta(z_t \mid z_{t-1}, a_{t-1})$.

After training, the latent world model can be used for multi-step imagination or planning. Starting from an inferred latent state z_t, the model recursively predicts future latent states under a candidate action sequence $(a_t, a_{t+1}, \ldots, a_{t+H-1})$:

$$\hat{z}_{t+k+1} \sim p_\theta(\hat{z}_{t+k+1} \mid \hat{z}_{t+k}, a_{t+k}), \tag{K.7}$$

$k = 0, 1, \ldots, H-1$, with $\hat{z}_t = z_t$. The predicted return of this imagined trajectory is

$$\hat{J}_\theta(a_{t:t+H-1} \mid z_t) = \mathbb{E}_{\mathcal{W}_\theta} \left[\sum_{k=0}^{H-1} \gamma^k \hat{r}_{t+k} \right], \tag{K.8}$$

where

$$\hat{r}_{t+k} \sim p_\theta(\hat{r}_{t+k} \mid \hat{z}_{t+k}, a_{t+k}). \tag{K.9}$$

The planning problem can then be written as

$$a^\star_{t:t+H-1} = \arg\max_{a_{t:t+H-1}} \hat{J}_\theta(a_{t:t+H-1} \mid z_t). \tag{K.10}$$

Thus, a latent world model is an action-conditioned predictive representation of the environment. It learns a latent state space in which the agent can simulate future trajectories, estimate their rewards, and select actions according to predicted long-term outcomes.

K.2 Error Propagation in Latent World Models

We first study how a one-step latent transition error propagates through multi-step imagination. We consider the deterministic mean dynamics induced by the latent world model. Let

$$\hat{z}_{t+1} = F_\theta(\hat{z}_t, a_t) \tag{K.11}$$

denote the learned latent transition, and let

$$z_{t+1} = F^\star(z_t, a_t) \tag{K.12}$$

denote the latent dynamics induced by the true environment under the same latent representation. Here, $F^\star$ is not required to be a physical-state transition; it is the true transition map in the latent space.

Assumption 8 (Uniform one-step latent transition error)**.** *There exists a constant $\epsilon_F \geq 0$ such that*

$$\sup_{z \in \mathcal{Z}, a \in \mathcal{A}} \|F_\theta(z, a) - F^\star(z, a)\| \leq \epsilon_F. \tag{K.13}$$

Assumption 9 (Lipschitz continuity of the learned latent transition)**.** *There exists a constant $L_F \geq 0$ such that, for any $z, z' \in \mathcal{Z}$ and $a \in \mathcal{A}$,*

$$\|F_\theta(z, a) - F_\theta(z', a)\| \leq L_F \|z - z'\|. \tag{K.14}$$

Given an action sequence $a_{t:t+H-1}$, define the true latent rollout and the imagined latent rollout as

$$\begin{aligned} z_{t+k+1} &= F^\star(z_{t+k}, a_{t+k}), \\ \hat{z}_{t+k+1} &= F_\theta(\hat{z}_{t+k}, a_{t+k}), \end{aligned} \tag{K.15}$$

for $k = 0, 1, \ldots, H-1$. Define the k-step rollout error by

$$e_k = \|\hat{z}_{t+k} - z_{t+k}\|. \tag{K.16}$$

Theorem 6 (Multi-step latent transition error propagation)**.** *Suppose Assumptions 8 and 9 hold. Then, for any fixed action sequence $a_{t:t+H-1}$ and any $k \in \{0, 1, \dots, H\}$,*

$$e_k \le L_F^k e_0 + \epsilon_F \sum_{i=0}^{k-1} L_F^i. \tag{K.17}$$

In particular, if $e_0 = 0$,

$$e_k = \begin{cases} O(\epsilon_F), & 0 \le L_F < 1, \\ O(k\epsilon_F), & L_F = 1, \\ O(\epsilon_F L_F^k), & L_F > 1. \end{cases} \tag{K.18}$$

Proof. For each $k = 0, 1, \dots, H-1$, adding and subtracting $F_\theta(z_{t+k}, a_{t+k})$ and applying Assumptions 9 and 8 give

$$\begin{aligned} e_{k+1} &= \|F_\theta(\hat{z}_{t+k}, a_{t+k}) - F^\star(z_{t+k}, a_{t+k})\| \\ &\le \|F_\theta(\hat{z}_{t+k}, a_{t+k}) - F_\theta(z_{t+k}, a_{t+k})\| + \|F_\theta(z_{t+k}, a_{t+k}) - F^\star(z_{t+k}, a_{t+k})\| \\ &\le L_F \|\hat{z}_{t+k} - z_{t+k}\| + \epsilon_F \\ &= L_F e_k + \epsilon_F. \end{aligned} \tag{K.19}$$

Unrolling Equation (K.19) yields

$$\begin{aligned} e_k &\le L_F e_{k-1} + \epsilon_F \\ &\le L_F \left(L_F e_{k-2} + \epsilon_F\right) + \epsilon_F \\ &= L_F^2 e_{k-2} + \epsilon_F (1 + L_F) \\ &\le \cdots \\ &\le L_F^k e_0 + \epsilon_F \sum_{i=0}^{k-1} L_F^i. \end{aligned} \tag{K.20}$$

For $L_F \neq 1$, Equation (K.20) gives

$$\begin{aligned} e_k &\le L_F^k e_0 + \epsilon_F \sum_{i=0}^{k-1} L_F^i \\ &= L_F^k e_0 + \epsilon_F \frac{L_F^k - 1}{L_F - 1}. \end{aligned} \tag{K.21}$$

For $L_F = 1$, Equation (K.19) gives

$$\begin{aligned} e_k &\le e_{k-1} + \epsilon_F \\ &\le e_{k-2} + 2\epsilon_F \\ &\le \cdots \\ &\le e_0 + k\epsilon_F. \end{aligned} \tag{K.22}$$

Equations (K.20)–(K.22) establish the claimed bounds. □

Theorem 6 shows that even a uniformly bounded one-step transition error can increase across a multi-step imagined trajectory. When $L_F < 1$, the transition is contractive and the accumulated error remains controlled. When $L_F = 1$, the error grows at most linearly in the planning horizon. When $L_F > 1$, the error bound grows geometrically, which gives a formal explanation of compounding error in latent world models.

K.3 Reward Prediction and Return Error Bound

We next show that latent transition error induces reward prediction error and therefore biases the planning objective. Let the learned reward model be

$$\hat{r}_{t+k} = R_\theta(\hat{z}_{t+k}, a_{t+k}), \tag{K.23}$$

and let the true latent reward function be

$$r_{t+k} = R^\star(z_{t+k}, a_{t+k}). \tag{K.24}$$

For a fixed horizon H, define the learned imagined return and the true return under the same action sequence as

$$\hat{J}_\theta(a_{t:t+H-1} \mid \hat{z}_t) = \sum_{k=0}^{H-1} \gamma^k R_\theta(\hat{z}_{t+k}, a_{t+k}) \tag{K.25}$$

and

$$J^\star(a_{t:t+H-1} \mid z_t) = \sum_{k=0}^{H-1} \gamma^k R^\star(z_{t+k}, a_{t+k}). \tag{K.26}$$

Assumption 10 (Uniform one-step reward model error)**.** *There exists a constant $\epsilon_R \geq 0$ such that*

$$\sup_{z \in \mathcal{Z}, a \in \mathcal{A}} |R_\theta(z, a) - R^\star(z, a)| \leq \epsilon_R. \tag{K.27}$$

The previous theorem bounds the deviation of the imagined latent trajectory from the true latent trajectory. We now show that this latent rollout error directly induces reward prediction error, because the reward model is evaluated on the imagined latent states rather than on the true latent states.

Assumption 11 (Lipschitz continuity of the learned reward model)**.** *There exists a constant $L_R \geq 0$ such that, for any $z, z' \in \mathcal{Z}$ and $a \in \mathcal{A}$,*

$$|R_\theta(z, a) - R_\theta(z', a)| \leq L_R \|z - z'\|. \tag{K.28}$$

Proposition 7 (Reward prediction error bound)**.** *Suppose Assumptions 8, 9, 10, and 11 hold. For any fixed action sequence $a_{t:t+H-1}$ and any $k = 0, 1, \ldots, H-1$, define*

$$\Delta_k^R = |R_\theta(\hat{z}_{t+k}, a_{t+k}) - R^\star(z_{t+k}, a_{t+k})|. \tag{K.29}$$

Then

$$\Delta_k^R \leq L_R \left(L_F^k e_0 + \epsilon_F \sum_{i=0}^{k-1} L_F^i \right) + \epsilon_R. \tag{K.30}$$

In particular, if $e_0 = 0$, then

$$\Delta_k^R \leq L_R \epsilon_F \sum_{i=0}^{k-1} L_F^i + \epsilon_R. \tag{K.31}$$

Proposition 7 separates the reward prediction error into two sources. The first term comes from latent state mismatch and scales with the transition error accumulated through the rollout. The second term, ϵ_R, is the intrinsic one-step approximation error of the reward model. Therefore, even when the reward model is uniformly accurate at one step, the reward estimate can still degrade over a long imagined trajectory because the latent inputs themselves become increasingly biased.

Proof. For any $k = 0, 1, \ldots, H-1$, adding and subtracting $R_\theta(z_{t+k}, a_{t+k})$ and applying Assumptions 11 and 10 give

$$\begin{aligned} |R_\theta(\hat{z}_{t+k}, a_{t+k}) - R^\star(z_{t+k}, a_{t+k})| &\leq |R_\theta(\hat{z}_{t+k}, a_{t+k}) - R_\theta(z_{t+k}, a_{t+k})| \\ &\quad + |R_\theta(z_{t+k}, a_{t+k}) - R^\star(z_{t+k}, a_{t+k})| \\ &\leq L_R \|\hat{z}_{t+k} - z_{t+k}\| + \epsilon_R \\ &= L_R e_k + \epsilon_R. \end{aligned} \tag{K.32}$$

By Theorem 6,

$$\begin{aligned} |R_\theta(\hat{z}_{t+k}, a_{t+k}) - R^\star(z_{t+k}, a_{t+k})| &\leq L_R e_k + \epsilon_R \\ &\leq L_R \left(L_F^k e_0 + \epsilon_F \sum_{i=0}^{k-1} L_F^i \right) + \epsilon_R \\ &= L_R L_F^k e_0 + L_R \epsilon_F \sum_{i=0}^{k-1} L_F^i + \epsilon_R. \end{aligned} \tag{K.33}$$

Equation (K.33) establishes the claimed reward-error bound. □

Proposition 8 (Multi-step return error bound)**.** *Suppose Assumptions 8, 9, 10, and 11 hold. Then, for any fixed action sequence* $a_{t:t+H-1}$*,*

$$\left|\hat{J}_\theta(a_{t:t+H-1} \mid \hat{z}_t) - J^\star(a_{t:t+H-1} \mid z_t)\right| \le \sum_{k=0}^{H-1} \gamma^k \left[L_R \left(L_F^k e_0 + \epsilon_F \sum_{i=0}^{k-1} L_F^i\right) + \epsilon_R\right]. \quad \text{(K.34)}$$

In particular, when $e_0 = 0$*,* $L_F = 1$*, and* $\gamma = 1$*,*

$$\begin{aligned}\left|\hat{J}_\theta(a_{t:t+H-1} \mid z_t) - J^\star(a_{t:t+H-1} \mid z_t)\right| &\le \frac{L_R\epsilon_F H(H-1)}{2} + \epsilon_R H \\ &= O\big(L_R\epsilon_F H^2 + \epsilon_R H\big).\end{aligned} \quad \text{(K.35)}$$

Proposition 8 shows that transition error affects planning through the cumulative return objective. When $e_0 = 0$, $L_F = 1$, and $\gamma = 1$, the return error scales as $O(L_R\epsilon_F H^2 + \epsilon_R H)$. Thus, a one-step latent transition error of order ϵ_F can lead to a quadratic dependence on the planning horizon, while the reward approximation error accumulates linearly. This explains why a latent world model with small one-step prediction error may still produce biased long-horizon planning estimates.

Proof. By definition, the triangle inequality and Proposition 7 give

$$\begin{aligned}\left|\hat{J}_\theta(a_{t:t+H-1} \mid \hat{z}_t) - J^\star(a_{t:t+H-1} \mid z_t)\right| &= \left|\sum_{k=0}^{H-1} \gamma^k \left[R_\theta(\hat{z}_{t+k}, a_{t+k}) - R^\star(z_{t+k}, a_{t+k})\right]\right| \\ &\le \sum_{k=0}^{H-1} \gamma^k \left|R_\theta(\hat{z}_{t+k}, a_{t+k}) - R^\star(z_{t+k}, a_{t+k})\right| \quad \text{(K.36)} \\ &\le \sum_{k=0}^{H-1} \gamma^k \left[L_R \left(L_F^k e_0 + \epsilon_F \sum_{i=0}^{k-1} L_F^i\right) + \epsilon_R\right].\end{aligned}$$

For $e_0 = 0$, Equation (K.36) becomes

$$\left|\hat{J}_\theta(a_{t:t+H-1} \mid z_t) - J^\star(a_{t:t+H-1} \mid z_t)\right| \le \sum_{k=0}^{H-1} \gamma^k \left(L_R\epsilon_F \sum_{i=0}^{k-1} L_F^i + \epsilon_R\right). \quad \text{(K.37)}$$

When $L_F = 1$ and $\gamma = 1$, Equation (K.37) further gives

$$\begin{aligned}\left|\hat{J}_\theta(a_{t:t+H-1} \mid z_t) - J^\star(a_{t:t+H-1} \mid z_t)\right| &\le \sum_{k=0}^{H-1} (L_R\epsilon_F k + \epsilon_R) \\ &= L_R\epsilon_F \sum_{k=0}^{H-1} k + \epsilon_R H \\ &= \frac{L_R\epsilon_F H(H-1)}{2} + \epsilon_R H \\ &= O\big(L_R\epsilon_F H^2 + \epsilon_R H\big).\end{aligned} \quad \text{(K.38)}$$

Equation (K.38) proves the stated return-error bound and asymptotic rate. □