# MAPID-LLM: Distributed PID Control for Multi-Agent LLM Reasoning

**Anonymous ACL submission**

## Abstract

Multi-agent LLMs enable collaborative reasoning, but their inference-time behavior remains difficult to regulate due to stochastic generation, non-stationary inter-agent interactions, and cumulative reasoning errors. We propose MAPID-LLM, an inference-time local-control distributed PID regulation framework under benchmark-derived measurement feedback. MAPID-LLM models multi-agent inference as a networked discrete-time stochastic control system with explicit output-to-input coupling, where agents' generated outputs are incorporated into neighboring agents' subsequent prompt contexts, and uses PID feedback to regulate bounded prompt-level control variables during inference. This PID design is intended to provide immediate correction, accumulated-error compensation, and trend-sensitive stabilization for stochastic and interacting agent behaviors. The benchmark or evaluation environment supplies task-level measurement and reference signals, but does not centrally plan, rewrite, rank, or select agents' reasoning trajectories. We use MultiAgentBench, MATH, BigCodeBench, and LLM backbones under their respective licenses and terms of use, and cite the original artifact creators where applicable.Experiments on MultiAgentBench, MATH, and BigCodeBench show relative task-performance gains of 6.0%–16.4% and coordination-quality gains of up to 21.0%, with ablations and bidirectional controllability tests providing preliminary support for prompt-level PID regulation. Code and experimental artifacts are available at this anonymous repository.

## 1 Introduction

Large language models have transformed reasoning, synthesis, and generation tasks (Brown et al., 2020; Ouyang et al., 2022). Multi-agent LLM systems extend this capability by letting specialized agents verify outputs, explore parallel solutions, and combine complementary expertise (Qian et al., 2024; Zhu et al., 2025). Existing systems usually coordinate agents through protocol- or topology-level mechanisms, such as self-consistency (Wang et al., 2023), refinement and reflection (Madaan et al., 2023; Shinn et al., 2023), and debate (Du et al., 2024). Yet many designs remain heuristic, based on prompt engineering, rule-based coordination, or sequential chaining (Khattab et al., 2024; Hu et al., 2025). They offer limited guarantees on stability, coherence, or efficiency, so errors can propagate when one agent's output destabilizes another's reasoning. This reliability issue is consistent with recent taxonomies of multi-agent failures, including specification errors, inter-agent misalignment, and task-verification problems (Zhu et al., 2025).

Recent prompt-optimization methods address parts of this problem. HiveMind (Xia et al., 2026) performs contribution-guided online prompt optimization, ResMAS (Zhou et al., 2026) introduces topology-aware optimization, AutoPDL (Spiess et al., 2025) optimizes prompts through source-to-source transformation, and Mass (Zhou et al., 2025) uses multi-stage interleaved optimization. However, most prompt-optimization methods still search discrete prompt spaces or tune prompts outside the active inference loop. Since LLM reasoning is sensitive to prompt perturbations (Chatziveroglou et al., 2025; Anagnostidis and Bulian, 2024), multi-agent inference needs a runtime mechanism that can regulate agents' reasoning processes while preserving distributed decision making.

Control theory provides such a mechanism. PID controllers have been used to refine prompts from output error (Karn, 2025), networked PID control supports multi-agent tracking (Liu, 2023), and distributed coordination theory explains how cooperation can arise from local information exchange (Ren and Atkins, 2007; Olfati-Saber et al.,

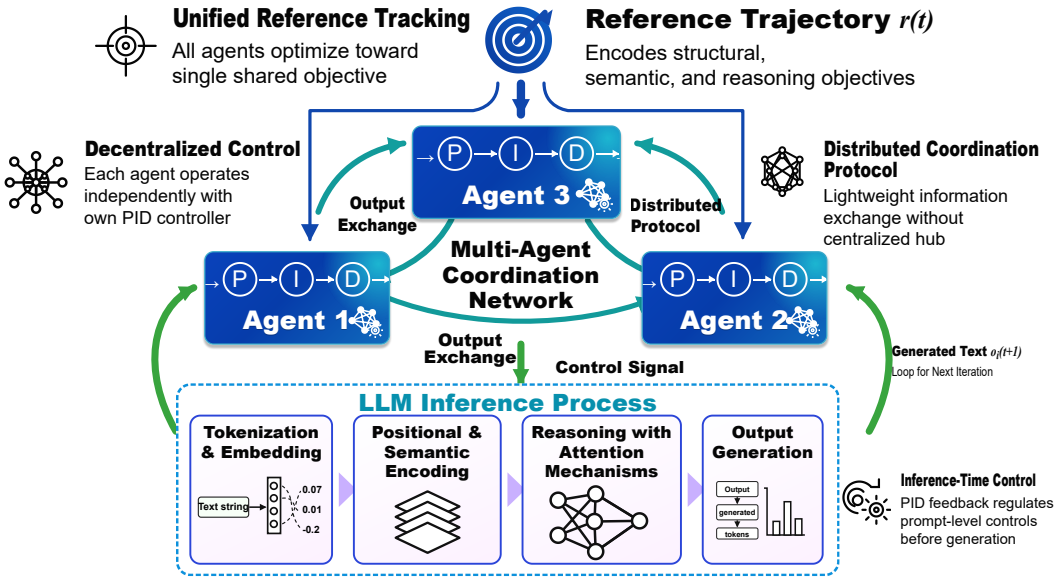


**Figure 1:** MAPID-LLM system architecture. PID feedback updates prompt-level controls before generation; generated text $o(t)$ is evaluated into measurement $y(t)$.

2007). Recent work also models LLMs as discrete stochastic dynamical systems suitable for control-theoretic analysis (Bhargava et al., 2023; Soatto et al., 2023). Motivated by these connections, we propose MAPID-LLM, an inference-time distributed PID regulation framework for multi-agent LLM reasoning.

MAPID-LLM treats each agent as a discrete-time stochastic dynamical system. Following (Karn, 2025), an evaluator-derived metric $y_i(t)$ measures the quality of agent $i$'s generated output relative to the task reference. At each inference step, the evaluator provides measurement and reference feedback, while agent $i$ locally computes a PID increment $u_i(t)$ to update bounded prompt-level control variables such as reasoning depth, verification strength, citation strictness, and cooperation intensity. The multi-agent coupling comes from the closed-loop context flow: outputs generated in a previous step can be inserted into the later prompt or context of neighboring agents. The evaluator is used only as a shared measurement and reference channel; it does not plan, rewrite, rank, or select agents' reasoning trajectories. Thus, MAPID-LLM is orthogonal to protocol-level methods: existing protocols define the communication topology and interaction order, whereas MAPID-LLM regulates each agent's prompt-level controls during inference. Appendix C.15, Table 5 compares MAPID-LLM with the closest related work; its key distinction is the combination of multi-agent LLM inference, inference-time prompt-level adaptation, distributed PID feedback, closed-loop context coupling, and conditional boundedness analysis.

**Contributions.** (1) We formulate multi-agent LLM inference as a networked discrete-time stochastic control system with explicit output-to-input coupling, while distinguishing evaluator-derived states from generated token outputs (Section 2). (2) We introduce MAPID-LLM, a distributed PID framework in which each agent computes its own control input from local state, evaluator feedback, and neighbor information to regulate reasoning-oriented prompt controls, while the evaluator supplies assessment signals rather than centralized coordination. (3) Experiments on MultiAgentBench, MATH, and BigCodeBench show that MAPID-LLM improves task performance by +6.0% to +16.4% and coordination quality by up to +21.0% (Section 4).

## 2 Modeling and Problem Formulation

Prior work shows that multi-agent debate can drift away from the original task over repeated turns, highlighting the need for inference-time monitoring and regulation (Becker et al., 2026). MAPID-LLM therefore separates multi-agent inference into three levels: generation, evaluation, and control. At each inference step, an agent first generates a natural-language output under its current prompt-level controls. The evaluator then maps this output

to normalized measurement signals and provides the feedback interface used by the local controller. Based on this feedback, each local PID controller updates bounded prompt-level controls such as reasoning depth, verification strength, citation strictness, and cooperation intensity.

We use $o_i(t)$, $\mathbf{y}_i(t)$, and $\mathbf{c}_i(t)$ to denote the generated output, evaluator-derived feedback, and prompt-level control vector of agent $i$, respectively. Other loop variables are introduced when their corresponding update rules are defined. With this notation, the one-agent information flow is

$$o_i(t) \to \mathbf{y}_i(t) \to \hat{\mathbf{r}}_i(t) \to \mathbf{r}_{i,c}(t) \to \mathbf{e}_i(t) \to \mathbf{u}_i(t) \to \mathbf{c}_i(t+1) \to \mathbf{x}_i^{\text{gen}}(t+1) \to o_i(t+1).$$

This expression is an overview of the computational order within one closed-loop inference step, rather than an additional dynamical assumption. The individual quantities, including the generation input, evaluator feedback, control-space reference, control-space error, PID increment, and projected prompt-control update, are formally defined in the following subsections. Thus, MAPID-LLM regulates prompt-level directives through evaluator feedback while keeping the PID computation local to each agent. This formulation follows the PID prompt-regulation view of LLM inference (Karn, 2025). Table 3 in Appendix B summarizes the main notation.

### 2.1 Agent Dynamics and Output Coupling

Consider a networked multi-agent LLM system with agent set $\mathcal{N} = \{1, 2, \ldots, n\}$ and communication graph $\mathcal{G}$. Agent $i$ receives information from its neighbor set $\mathcal{N}_i$. The empirical framework allows directed topologies, while the graph-regularized equilibrium result in Section 3.3 uses the standard undirected connected case stated in Appendix C.4.

At inference step $t$, agent $i$ operates under a prompt-level control vector

$$\mathbf{c}_i(t) = \left[c_i^1(t),\, c_i^2(t),\, \ldots,\, c_i^k(t)\right]^\top \in \mathcal{C}, \qquad (1)$$

where

$$\mathcal{C} = \prod_{q=1}^{k} [\underline{c}_q, \overline{c}_q]$$

is a bounded prompt-control box. Each coordinate corresponds to a reasoning-oriented runtime directive, such as search depth, verification strength, citation strictness, or cooperation intensity. The generation input combines the base role prompt,

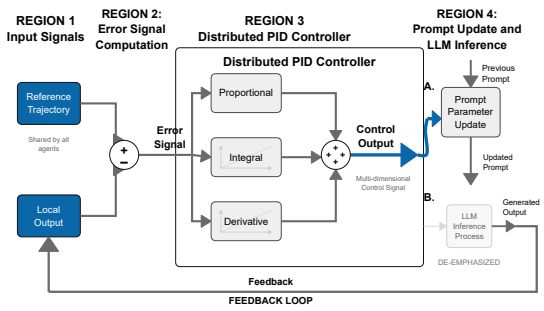


**Figure 2:** MAPID-LLM control pipeline: input signals are converted into control-space errors, processed by distributed PID regulation, and applied as prompt-level updates for subsequent LLM inference.

the current controls, neighbor outputs from the previous step, and local dialogue history:

$$\mathbf{x}_i^{\text{gen}}(t) = \Gamma_i\Big(\text{Prompt}_i,\, \phi(\mathbf{c}_i(t)),\, \{o_j(t-1) : j \in \mathcal{N}_i\},\, H_i(t)\Big). \qquad (2)$$

Here $\Gamma_i(\cdot)$ denotes prompt composition, and $\phi(\cdot)$ maps continuous prompt-level controls to textual directives. This output-to-input propagation is the main coupling mechanism: one agent's output can become another agent's context. The continuous controls are converted into discrete textual directives through clipped affine normalization; the mapping and an example directive are given in Appendix B.1.

Runtime token generation is separated from the evaluator state used for feedback:

$$o_i(t) = \mathcal{F}_{\text{LLM}}\big(\mathbf{x}_i^{\text{gen}}(t)\big), \qquad (3)$$

where $o_i(t) \in \mathcal{V}^*$ is the raw generated token sequence. The evaluator extracts

$$\mathbf{y}_i(t) = \mathcal{E}(o_i(t)), \qquad (4)$$

where $\mathbf{y}_i(t) \in \mathbb{R}^m$ denotes measurement feedback rather than raw generation. Thus, the controller never acts on $o_i(t)$ directly; it only receives $\mathbf{y}_i(t)$ after the evaluator maps generated text into a normalized measurement space.

### 2.2 Distributed Control Error and Update

Let $\mathbf{R}_{\text{task}}(t)$ denote the global task objective. For each agent, the evaluator-shaped interface forms a role-specific semantic reference

$$\mathbf{r}_i(t) = \Pi_i(\mathbf{R}_{\text{task}}(t)), \qquad (5)$$

where $\Pi_i$ is the role projection for agent $i$. The local reference estimate $\hat{\mathbf{r}}_i(t)$ is then mapped into the prompt-control space as

$$\mathbf{r}_{i,c}(t) = \psi_i(\hat{\mathbf{r}}_i(t)) \in \mathbb{R}^k, \tag{6}$$

so that the reference target has the same dimension as the prompt-level control vector $\mathbf{c}_i(t)$.

The PID controller uses the following fused control-space error:

$$\begin{aligned}\mathbf{e}_i(t) &= \alpha\big(\mathbf{r}_{i,c}(t) - \mathbf{c}_i(t)\big) \\ &\quad + (1-\alpha)\sum_{j\in\mathcal{N}_i} w_{ij}\big(\tilde{\mathbf{c}}_j(t) - \mathbf{c}_i(t)\big).\end{aligned} \tag{7}$$

Here $w_{ij} \geq 0$, $\sum_{j\in\mathcal{N}_i} w_{ij} = 1$, and $\tilde{\mathbf{c}}_j(t)$ is the latest available or delay-compensated neighbor control summary. In synchronous communication, $\tilde{\mathbf{c}}_j(t) = \mathbf{c}_j(t)$. The first term tracks the agent-specific reference, while the second term encourages local coordination. For the theoretical analysis, we use $\alpha \in [\alpha_{\min}, 1]$ with $\alpha_{\min} > 0$; the case $\alpha = 0$ is used only in empirical ablations.

Given the PID increment $\mathbf{u}_i(t)$ defined in Section 3.1, we update the prompt-level control in two bounded steps:

$$\begin{aligned}\bar{\mathbf{u}}_i(t) &= \mathrm{clip}_{[\mathbf{u}_{\min},\mathbf{u}_{\max}]}\big(\mathbf{u}_i(t)\big), \\ \mathbf{c}_i(t+1) &= \mathrm{clip}_{\mathcal{C}}\big(\mathbf{c}_i(t) + \bar{\mathbf{u}}_i(t)\big).\end{aligned} \tag{8}$$

Here $\mathrm{clip}(\cdot)$ is applied component-wise. The first line limits how much each prompt-control variable can change in one update, and the second line keeps the updated control vector inside the feasible box $\mathcal{C}$. In other words, PID feedback can adjust directives such as reasoning depth, verification strength, and cooperation intensity, but only within predefined valid ranges.

### 2.3 Evaluator-Shaped Reference Interface

The evaluator-shaped interface converts evaluator measurements and lightweight neighbor summaries into role-specific references $\mathbf{r}_i(t+1)$. These outputs are parsed only as measurement, summary, or semantic-reference signals and are then consumed by the local controller through $\psi_i(\cdot)$ and Eq. (7). The interface therefore provides feedback for regulation rather than a centralized reasoning policy. It does not optimize agents' responses, choose among candidate trajectories, or replace the underlying communication protocol. The aligned reference update and implementation boundary are given in Appendix C.3, Eq. (18).

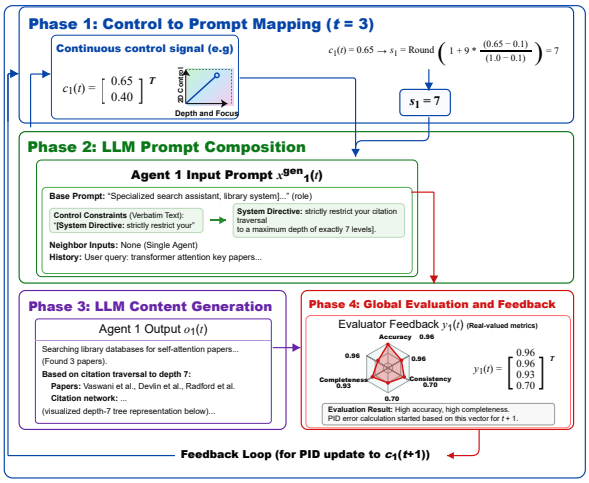


**Figure 3:** Single-agent example of the MAPID-LLM inference-time control loop: continuous controls are mapped to prompt directives, used for generation, and updated through evaluator feedback via PID regulation.

## 3 MAPID-LLM: Control Update and Conditional Analysis

This section gives the executable MAPID-LLM update rule and states the boundedness result used in our analysis. We analyze closed-loop behavior through evaluator feedback, after PID gains are selected on held-out validation trajectories and prompt-level controls are kept within preset bounds. The result is local to the validated operating region and should not be interpreted as a global stability guarantee for raw token generation.

Figure 2 summarizes the MAPID-LLM control pipeline: generated outputs are evaluated against task references, mapped into control-space errors, processed by the distributed PID controller, and used to update prompt-level controls before the next inference step.

### 3.1 Control-Space PID Update

The PID controller is driven by the control-space fusion error $\mathbf{e}_i(t)$ in Eq. (7). This design avoids applying PID feedback directly to raw token sequences or to evaluator scores with incompatible dimensions. Instead, $\psi_i(\cdot)$ maps the locally available reference estimate $\hat{\mathbf{r}}_i(t)$ to the control-space target $\mathbf{r}_{i,c}(t)$, which has the same $k$-dimensional space as the prompt-level actuator $\mathbf{c}_i(t)$.

Figure 3 illustrates how MAPID-LLM operationalizes prompt-level PID control for a single agent: continuous controls are discretized into textual directives, injected into generation, and updated through evaluator feedback.

For each agent $i$, the integral memory is updated

with component-wise anti-windup clipping:

$$\mathbf{I}_i(t) = \mathrm{clip}_{[-\mathbf{I}_{\max},\mathbf{I}_{\max}]}\big(\mathbf{I}_i(t-1) + \mathbf{e}_i(t)\big). \quad (9)$$

Let

$$\Delta\mathbf{e}_i(t) = \mathbf{e}_i(t) - \mathbf{e}_i(t-1).$$

Following the standard discrete-time PID-control convention, the controller combines proportional, integral, and derivative feedback gains to provide immediate, accumulated, and trend-sensitive correction (Karn, 2025; Liu, 2023):

$$\mathbf{u}_i(t) = \mathbf{K}_{P,i}\mathbf{e}_i(t) + \mathbf{K}_{I,i}\mathbf{I}_i(t) + \mathbf{K}_{D,i}\Delta\mathbf{e}_i(t), \quad (10)$$

where $\mathbf{K}_{P,i}$, $\mathbf{K}_{I,i}$, and $\mathbf{K}_{D,i}$ denote the proportional, integral, and derivative gain matrices, respectively. The proportional term reacts to the current control-space error, the integral term accumulates persistent error through $\mathbf{I}_i(t)$, and the derivative term responds to the error difference $\Delta\mathbf{e}_i(t)$. In the implementation, these matrices are diagonal in $\mathbb{R}^{k\times k}$, so each prompt-control coordinate can be regulated separately while sharing the same closed-loop update form. The bounded prompt-level control is then updated by Eq. (8). Thus, the controller modifies only feasible prompt-level directives, not model parameters, logits, hidden states, or raw token sequences.

### 3.2 Validation-Screened Boundedness Analysis

Candidate diagonal PID gains are selected on held-out validation trajectories. A gain tuple is accepted only if the validation-fitted evaluator-level surrogate satisfies a robust Schur-margin condition and its residual remains below the chosen tolerance. Let $\widehat{\Phi}_{\mathrm{sur}}$ denote the surrogate transition fitted from held-out validation trajectories, and let $\Phi_{\mathrm{sur}}$ denote the corresponding local nominal transition in the validated operating region. The acceptance rule uses a robustness margin so that the fitted Schur condition transfers to the nominal transition; the estimation gap, residual decomposition, delay treatment, and surrogate construction are detailed in Appendix C.

Let $\mathbf{y}_i^{\mathrm{des}}(t) = \chi_i(\mathbf{r}_i(t))$ be the evaluator-space desired signal induced by the semantic reference. Under the local conditions S1–S6 in Appendix C.4, MAPID-LLM yields the following evaluator-level result.

**Theorem 3.1** (Validation-screened evaluator-level boundedness)**.** *Assume S1–S6 in Appendix C.4. Let $\Phi_{\mathrm{sur}}$ be the local nominal surrogate transition matrix in the validated operating region. If the accepted PID gains satisfy*

$$\rho(\Phi_{\mathrm{sur}}) \le 1 - \varepsilon'_{\Phi} < 1,$$

*then for each agent $i \in \mathcal{N}$, there exist constants $T_0 \ge 0$ and $B_i < \infty$ such that*

$$\sup_{t\ge T_0} \mathbb{E}\left[\left\|\mathbf{e}_i^{\mathrm{infer}}(t)\right\|^2\right] \le B_i^2, \quad \forall i \in \mathcal{N}, \quad (11)$$

*where*

$$\mathbf{e}_i^{\mathrm{infer}}(t) = \mathbf{y}_i^{\mathrm{des}}(t) - \mathbf{y}_i(t). \quad (12)$$

Theorem 3.1 should be read as ultimate mean-square boundedness of the evaluator-space inference error under the accepted surrogate model, not as convergence of the generated text or as a guarantee of zero task error.

### 3.3 Graph-Regularized Reference Compromise

The fused error in Eq. (7) balances agent-specific reference tracking with graph-based coordination. This characterization is used only for the standard undirected connected case; the empirical framework can still use directed or row-normalized communication topologies. To make the tradeoff explicit, let $\hat{\mathbf{r}}_i^\star$ denote the limiting locally available reference estimate used by the controller. Define the stacked prompt-control vector and limiting control-space reference as

$$\boldsymbol{c} = [\mathbf{c}_1^\top, \dots, \mathbf{c}_N^\top]^\top,$$
$$\hat{\boldsymbol{r}}_c^\star = \left[(\hat{\mathbf{r}}_{1,c}^\star)^\top, \dots, (\hat{\mathbf{r}}_{N,c}^\star)^\top\right]^\top.$$

where $\hat{\mathbf{r}}_{i,c}^\star = \psi_i(\hat{\mathbf{r}}_i^\star)$. In the zero-residual limit with convergent local reference estimates, the limiting control vector is characterized by the graph-regularized objective

$$\boldsymbol{c}^\star \in \arg\min_{\boldsymbol{c}\in\mathcal{C}^N} \frac{\alpha}{2}\|\boldsymbol{c} - \hat{\boldsymbol{r}}_c^\star\|_2^2 + \frac{1-\alpha}{2}\boldsymbol{c}^\top(\mathcal{L}_w \otimes \mathbf{I}_k)\boldsymbol{c}. \quad (13)$$

**Theorem 3.2** (Graph-regularized reference compromise)**.** *Under S1–S6, an undirected connected equilibrium graph, symmetric nonnegative coordination weights, and convergent bounded local reference estimates $\hat{\mathbf{r}}_i(t) \to \hat{\mathbf{r}}_i^\star$, the limiting prompt-control vector $\boldsymbol{c}^\star$ solves Eq. (13). Consequently, heterogeneous local reference estimates generally yield a topology-weighted compromise between individual reference tracking and neighborhood*

**Table 1:** Main benchmark results for MAPID-LLM as an inference-time plug-in control layer. Scores are averaged over three matched seeds and, for cross-backbone rows, over Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3. Per-backbone mean $\pm$ standard deviation results are reported in Appendix E.2. Relative Gain denotes the percentage improvement over the corresponding baseline, whereas the appendix reports per-backbone absolute score gaps.

| Benchmark | Metric | Baseline | Score of Baseline | MAPID Setting | Score of MAPID | Relative Gain |
|---|---|---|---|---|---|---|
| **MultiAgentBench task performance, cross-backbone average over Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3** | | | | | | |
| MAB-Coding | Task Score | MARBLE w/o PID | 66.7% | MARBLE + MAPID | 73.8% | **+10.6%** |
| MAB-Research | Task Score | MARBLE w/o PID | 69.1% | MARBLE + MAPID | 78.4% | **+13.5%** |
| MAB-Database | Task Score | MARBLE w/o PID | 56.7% | MARBLE + MAPID | 66.0% | **+16.4%** |
| **MultiAgentBench coordination quality, cross-backbone average over Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3** | | | | | | |
| MAB-Coding | Coord. Score | MARBLE w/o PID | 66.4% | MARBLE + MAPID | 73.2% | **+10.2%** |
| MAB-Research | Coord. Score | MARBLE w/o PID | 59.4% | MARBLE + MAPID | 71.9% | **+21.0%** |
| MAB-Database | Coord. Score | MARBLE w/o PID | 64.7% | MARBLE + MAPID | 72.9% | **+12.7%** |
| **Generalization to math and code benchmarks, cross-backbone average over Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3** | | | | | | |
| MATH | Accuracy | w/o MAPID | 71.4% | Full MAPID | 78.4% | **+9.8%** |
| BigCodeBench | Pass@1 | w/o MAPID | 71.1% | Full MAPID | 80.4% | **+13.1%** |
| **Plug-in integration with inference-time reasoning baselines, cross-backbone average over Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3** | | | | | | |
| MATH | Accuracy | MAD | 77.2% | MAD + MAPID | 81.8% | **+6.0%** |
| BigCodeBench | Pass@1 | MAD | 62.5% | MAD + MAPID | 67.1% | **+7.4%** |
| MATH | Accuracy | Self-Refine | 78.9% | MAPID | 86.0% | **+9.0%** |
| BigCodeBench | Pass@1 | Self-Refine | 75.0% | MAPID | 83.1% | **+10.8%** |

*coordination, rather than exact consensus. The closed-form interior solution and boundary KKT characterization are given in Appendix C.13.*

The coefficient $\alpha$ controls the reference–coordination tradeoff. With persistent stochastic and approximation residuals, MAPID-LLM provides ultimate boundedness rather than exact convergence.

## 4 Experiments

We evaluate MAPID-LLM as an inference-time control layer. The experiments test whether MAPID-LLM improves task performance, coordination quality, and whether the gains are associated with closed-loop PID regulation rather than fixed prompt changes.

### 4.1 Experimental Setup

We evaluate MAPID-LLM on MultiAgentBench, MATH, and BigCodeBench. MultiAgentBench covers Coding, Research, and Database scenarios and reports Task Score (TS) and Coordination Score (CS). MATH reports mathematical reasoning accuracy, and BigCodeBench reports Pass@1 for code generation. We use three LLM backbones: Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3.

For MultiAgentBench, we compare MARBLE + MAPID with MARBLE without PID regulation. For MATH and BigCodeBench, we compare Full MAPID with the corresponding non-MAPID multi-agent setting. We further integrate MAPID-LLM with MAD and Self-Refine to test whether it can act as a plug-in control layer for existing inference-time reasoning protocols. Each method–benchmark–backbone configuration is evaluated with three matched seeds: 42, 123, and 456. Cross-backbone results first average repeated runs within each backbone and then average the three backbone means. Full benchmark settings, control-variable mappings, gain-selection details, and per-backbone results are reported in Appendix E.1 and Appendix E.2.

### 4.2 Main Results

Table 1 shows that MAPID-LLM consistently improves task performance and coordination quality. On MultiAgentBench, MAPID-LLM improves both TS and CS across Coding, Research, and Database tasks. The gains also transfer to MATH and BigCodeBench, and the plug-in results with MAD and Self-Refine show that MAPID-LLM is complementary to existing inference-time reasoning protocols.

Figure 4 illustrates the runtime behavior behind these gains. Evaluator feedback is converted into local PID updates, which adjust agent-specific prompt directives such as evidence search, verification strength, and integration effort. The control layer therefore regulates the reasoning process

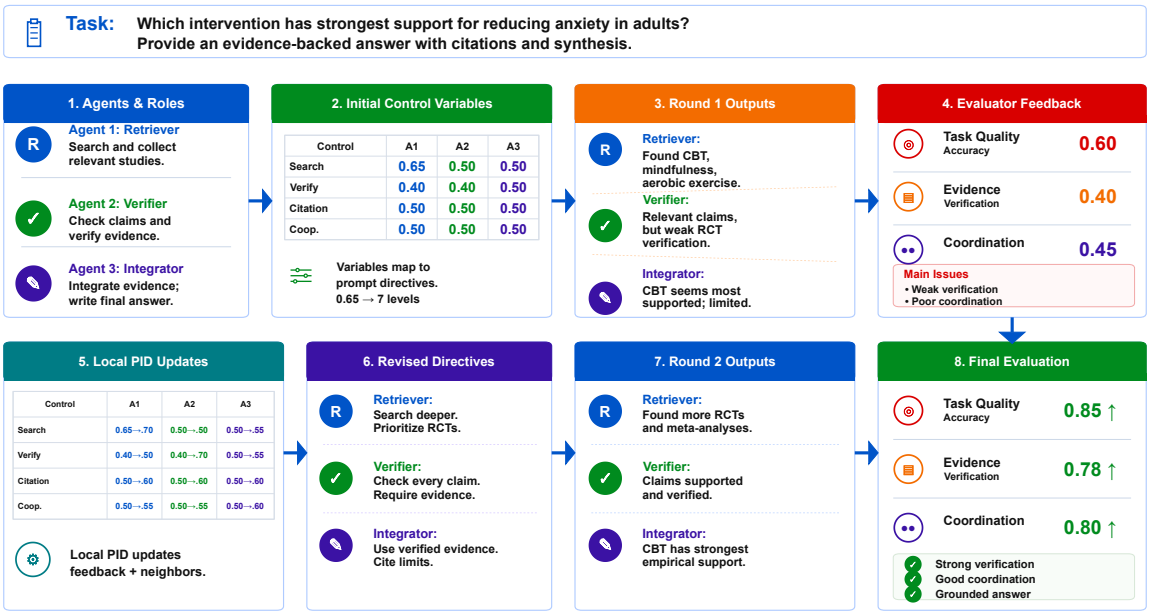


**Figure 4:** Multi-agent MAPID-LLM control example on a MAB-Research task. Evaluator feedback is converted into local PID updates, revising agent-specific prompt directives and improving evidence strength, coordination, and task quality.

**Table 2:** Summary of control-oriented analyses on Qwen3-8B. PID ablation scores report Task Score on the Qwen3-8B Coding subset. The GOOD/BAD intervention is evaluated on the same subset.

| Analysis | Key Result | Interpretation |
|---|---|---|
| PID ablation | Full PID: 76.6% | Best overall control. |
| Only P | 72.3% | Strong immediate correction. |
| Only D | 72.9% | Standalone stabilization. |
| P+I | 74.7% | Recovers most full-PID performance. |
| P+D | 73.4% | Useful stabilization without integral memory. |
| I+D, no P | 70.2% | Largest two-term degradation. |
| GOOD/BAD control | TS: 76.6% vs. 58.5% | Directional task influence. |
| GOOD/BAD control | CS: 79.8% vs. 61.5% | Directional coordination influence. |
| Fusion coefficient | $\alpha = 0.6$ | Balances task and coordination. |

without replacing the underlying multi-agent protocol.

### 4.3 Control Analysis

To test whether the gains come from closed-loop regulation rather than from a static prompt change, we conduct control-oriented analyses on Qwen3-8B. We choose Qwen3-8B because it is the most capacity-constrained backbone and is therefore more sensitive to prompt-level control. Table 2 summarizes the main findings, with full ablation and sensitivity results provided in Appendix E.4 and Appendix E.5.

The PID ablation shows that full PID achieves the best result. P+I recovers most of the full-PID performance, while P+D remains useful without integral memory. In contrast, I+D suffers the largest degradation, indicating that the proportional term is important for immediate runtime correction.

The GOOD/BAD intervention provides evidence of directional controllability. Optimized control improves both TS and CS, whereas intentionally degraded control reduces both metrics. The best fusion coefficient occurs around $\alpha = 0.6$, supporting the design choice of balancing agent-specific reference tracking with neighbor coordination.

## 5 Conclusion

MAPID-LLM introduces an inference-time distributed PID regulation framework for multi-agent LLM reasoning. By modeling multi-agent inference as a networked stochastic control system with explicit output-to-input coupling, MAPID-LLM enables agents to locally regulate bounded prompt-level control variables through PID feedback. The framework uses task-level feedback available from the benchmark or evaluation environment as a measurement signal, rather than as centralized coordination or centralized rewriting of agents' reasoning trajectories. Under stated bounded-residual assumptions and validation-screened surrogate stability conditions, MAPID-LLM provides local surrogate-based conditional boundedness analysis, and experiments on MultiAgentBench, MATH, and BigCodeBench show consistent improvements in

task performance and coordination quality. Ablations and bidirectional controllability tests further support the role of prompt-level PID regulation.

## 6 Limitations and Future Work

**System overhead and latency.** MAPID-LLM introduces additional overhead due to evaluator calls, intermediate feedback generation, and control-state exchange. In the evaluated settings, we observe approximately 12% additional token usage, but the exact overhead depends on the number of agents, communication rounds, evaluator prompt length, and benchmark type. We therefore report token, evaluator-call, and wall-clock overhead in Appendix E.11. The theoretical analysis provides conditional boundedness results under the stated assumptions, rather than unconditional guarantees for arbitrary deployments.

**Scalability.** The current experiments cover typical small-to-medium multi-agent teams. Scaling to larger teams may introduce communication and evaluator bottlenecks, especially under dense interaction topologies. Hierarchical or sparse communication topologies are promising directions for future work.

**Output coupling delays.** Our current formulation uses predictive surrogate estimates for delay compensation, but fully characterizing delayed output-to-input coupling in large-scale deployments remains an important theoretical direction.

## 7 Impact Statement

MAPID-LLM aims to improve the reliability of multi-agent LLM systems through bounded prompt-level runtime regulation, and does not introduce new user-facing capabilities beyond the underlying LLM systems. During the preparation of this manuscript, AI assistants and machine translation tools were used only for language polishing and grammar checking.

## References


Sotiris Anagnostidis and Jannis Bulian. 2024. How susceptible are LLMs to influence in prompts? In *First Conference on Language Modeling*.

Jonas Becker, Lars Benedikt Kaesberg, Andreas Stephan, Jan Philip Wahle, Terry Ruas, and Bela Gipp. 2026. Stay focused: Problem drift in multi-agent debate. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5068–5102.

Aman Bhargava, Cameron Witkowski, Shi-Zhuo Looi, and Matt Thomson. 2023. What's the magic word? a control theory of LLM prompting. *arXiv preprint arXiv:2310.04444*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Giannis Chatziveroglou, Richard Yun, and Maura Kelleher. 2025. Exploring LLM reasoning through controlled prompt variations. *arXiv preprint arXiv:2504.02111*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*.

Shengran Hu, Cong Lu, and Jeff Clune. 2025. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*.

Rupesh Raj Karn. 2025. Linear feedback control systems for iterative prompt optimization in large language models. *arXiv preprint arXiv:2501.11979*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into self-improving pipelines. In *The Twelfth International Conference on Learning Representations*.

Guo-Ping Liu. 2023. Tracking control of multi-agent systems using a networked predictive PID tracking scheme. *IEEE/CAA Journal of Automatica Sinica*, 10(1):216–225.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594.

Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. 2007. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186. Association for Computational Linguistics.

Wei Ren and Ella Atkins. 2007. Distributed multi-vehicle coordinated control via local information exchange. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 17(10-11):1002–1033.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652.

Stefano Soatto, Paulo Tabuada, Pratik Chaudhari, and Tian Yu Liu. 2023. Taming AI bots: Controllability of neural states in large language models. *arXiv preprint arXiv:2305.18449*.

Claudio Spiess, Mandana Vaziri, Louis Mandel, and Martin Hirzel. 2025. AutoPDL: Automatic prompt optimization for LLM agents. *arXiv preprint arXiv:2504.04365*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yihan Xia, Taotao Wang, Shengli Zhang, Zhangyuhua Weng, Bin Cao, and Soung Chang Liew. 2026. Hivemind: Contribution-guided online prompt optimization of LLM multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 29767–29774.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö Arık. 2025. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv preprint arXiv:2502.02533*.

Zhilun Zhou, Zihan Liu, Jiahe Liu, Qingyu Shao, Yihan Wang, Kun Shao, Depeng Jin, and Fengli Xu. 2026. ResMAS: Resilience optimization in LLM-based multi-agent systems. *arXiv preprint arXiv:2601.04694*.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. 2025. MultiAgentBench: Evaluating the collaboration and competition of LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8580–8622.

# Supplementary Material

## A Related Work

### A.1 Control Theory for Large Language Models

(Bhargava et al., 2023) formalize LLM systems as discrete stochastic dynamical systems and analyze controllability through reachable sets. (Soatto et al., 2023) define controllability within semantic spaces. (Karn, 2025) models prompt optimization as a linear feedback control system, where PID controllers iteratively refine prompts based on error. This work defines the key variables: evaluation $y(t)$ measuring quality, control $u(t)$ modifying prompts. Recent studies (Chatziveroglou et al., 2025; Anagnostidis and Bulian, 2024) demonstrate LLMs are highly sensitive to prompt modifications.

### A.2 Multi-Agent LLM Systems and Coordination Protocols

Multi-agent LLM systems commonly rely on protocol- or topology-level coordination, including parallel aggregation or self-consistency (Wang et al., 2023), serial refinement and reflection (Madaan et al., 2023; Shinn et al., 2023), and debate-based interaction (Du et al., 2024). These methods mainly specify how agents exchange information and sequence reasoning steps. Classical distributed PID and consensus control (Ren and Atkins, 2007; Olfati-Saber et al., 2007) provide useful tools for analyzing networked coordination, while predictive PID schemes address communication delays (Liu, 2023). MAPID-LLM differs from these protocol-level methods by adding a local continuous control layer over prompt-level variables during inference.

### A.3 Automatic Prompt Optimization

Automatic prompt-optimization methods, including recent agentic or source-to-source prompt optimizers, usually search discrete prompt spaces or optimize prompts before deployment. In contrast, MAPID-LLM performs runtime regulation during active inference: evaluator feedback updates bounded prompt-level controls without replacing the underlying multi-agent protocol. Thus, pre-inference prompt search and MAPID-LLM are complementary rather than competing approaches.

### A.4 Gap and Our Contribution

Table 5 summarizes the distinction between MAPID-LLM and the closest related lines of work. Prior multi-agent LLM methods primarily define communication protocols, debate procedures, or self-refinement loops, while automatic prompt-optimization methods usually search for static prompts before inference. Classical networked PID control provides useful stability tools, but it is not designed for prompt-level regulation of LLM agents during multi-agent reasoning. To the best of our knowledge, existing work has not jointly studied inference-time distributed PID regulation, prompt-level control variables, explicit output-to-input coupling, and conditional closed-loop boundedness analysis within a single multi-agent LLM inference framework.

## B Notation

Table 3 summarizes the main notation used throughout the MAPID-LLM formulation, including agent-level variables, evaluator feedback signals, prompt-level control variables, graph-related quantities, and theoretical constants.

### B.1 Affine Normalization from Continuous Controls to Prompt Directives

The control vector $\mathbf{c}_i(t)$ is continuous for analysis but must be converted into discrete prompt directives during inference. For a scalar control dimension

$$c^{(q)} \in [\underline{c}^{(q)}, \overline{c}^{(q)}],$$

mapped to a discrete directive level

$$s^{(q)} \in \{s^{(q)}_{\min}, \dots, s^{(q)}_{\max}\},$$

MAPID-LLM uses the clipped affine mapping

$$s^{(q)} = \mathrm{clip}_{[s^{(q)}_{\min}, s^{(q)}_{\max}]}\left(\mathrm{Round}\left[s^{(q)}_{\min} + \frac{c^{(q)} - \underline{c}^{(q)}}{\overline{c}^{(q)} - \underline{c}^{(q)}}\left(s^{(q)}_{\max} - s^{(q)}_{\min}\right)\right]\right). \tag{14}$$

The clipping after rounding prevents numerical or boundary effects from producing a directive outside the allowed discrete range. The resulting integer level is inserted into a structured system directive, such as a maximum search depth, verification intensity, citation strictness, or cooperation-strength instruction. For example, a search-depth control value $c = 0.65$ within $[0.1, 1.0]$ maps to $s = \mathrm{Round}\left(1 + 9 \times \frac{0.65-0.1}{1.0-0.1}\right) = 7$, yielding a directive such as *"[System Directive: strictly restrict your citation traversal to a maximum depth of*

**Table 3:** Notation used in MAPID-LLM.

| Symbol | Meaning | Implementation / Source |
|---|---|---|
| $\mathcal{N}$ | Set of agents | Multi-agent system index set |
| $i, j$ | Agent indices | Used to denote local agent $i$ and neighbor agent $j$ |
| $t$ | Inference step | Discrete reasoning/control iteration |
| $o_i(t)$ | Raw generated response of agent $i$ | Natural-language output produced by the LLM |
| $q_i(t)$ | Embedding or summary of $o_i(t)$ | Optional compact representation of the raw response |
| $\mathbf{y}_i(t)$ | Evaluator-derived feedback state | Normalized task, constraint, or coordination score from evaluator $\mathcal{E}$ |
| $\Gamma_i(\cdot)$ | Prompt-composition map | Builds the generation input from role prompt, directives, neighbor outputs, and history |
| $\phi(\cdot)$ | Control-to-directive map | Converts continuous prompt-level controls to textual directives |
| $\mathbf{c}_i(t)$ | Prompt-level control vector | Bounded variables controlling reasoning depth, verification strength, citation strictness, etc. |
| $\mathbf{u}_i(t)$ | PID control increment | Update applied to $\mathbf{c}_i(t)$ before prompt-level actuation |
| $\mathbf{r}_i(t)$ | Agent-specific semantic reference | Role-specific target derived from the task objective |
| $\hat{\mathbf{r}}_i(t)$ | Distributed estimate of semantic reference | Reference estimate updated through neighbor alignment and local innovation |
| $\mathbf{r}_{i,c}(t)$ | Control-space reference target | Projection of $\hat{\mathbf{r}}_i(t)$ into the prompt-control space |
| $\psi_i(\cdot)$ | Semantic-to-control projection | Maps semantic/evaluator reference targets into the control-variable space |
| $\mathbf{e}_i(t)$ | Unified PID feedback error | Control-space error combining reference tracking and neighbor coordination |
| $\alpha(\mathbf{r}_{i,c}(t) - \mathbf{c}_i(t))$ | Tracking component | Control-space reference-tracking part of the fused error |
| $(1-\alpha)\sum_j w_{ij}(\tilde{\mathbf{c}}_j(t) - \mathbf{c}_i(t))$ | Coordination component | Neighbor-alignment part of the fused error |
| $\mathcal{N}_i$ | Neighbor set of agent $i$ | Agents whose outputs or control states are available to agent $i$ |
| $w_{ij}$ | Coupling weight from neighbor $j$ to agent $i$ | Normalized neighbor weight used in coordination terms |
| $\mathcal{G}$ | Communication graph | Agent interaction topology |
| $\mathcal{A}$ | Adjacency matrix | Binary matrix defining directed communication edges |
| $\mathcal{L}$ | Graph Laplacian | Used in graph-regularized equilibrium analysis |
| $\alpha$ | Tracking–coordination tradeoff coefficient | Balances self reference tracking and neighborhood coordination |
| $\beta_r, \gamma$ | Reference innovation and alignment gains | Used in distributed semantic-reference estimation |
| $\mathbf{K}_P, \mathbf{K}_I, \mathbf{K}_D$ | PID gain matrices | Proportional, integral, and derivative feedback gains |
| $\mathcal{F}_{\text{LLM}}$ | LLM generation mapping | Black-box LLM generation interface |
| $\mathcal{E}(\cdot)$ | Evaluator | Maps generated responses to feedback states |
| $\mathcal{F}_{\text{sys}}$ | System-level diagnostic module | Aggregates evaluator feedback into task-level diagnostic state |
| $\mathcal{F}_{\text{ref}}$ | Reference generator | Produces agent-specific semantic references |
| $H_i(t)$ | Local dialogue history | Agent-specific interaction history |
| $\mathbf{h}_i(t)$ | Neighbor summary | Aggregated summary of neighbor outputs or evaluations |
| $\tilde{\boldsymbol{\eta}}(t)$ | Aggregate surrogate input | Generation noise, prediction residuals, reference drift, and local linearization error |
| $B_r$ | Reference variation bound | Upper bound on step-to-step semantic-reference change |
| $\rho(\cdot), \bar{\rho}$ | Spectral radius / contraction factor | $\rho(\cdot)$ denotes spectral radius; $\bar{\rho}$ denotes reference-estimation contraction |
| $B_\xi, B_i$ | Ultimate boundedness constants | Bounds for surrogate state and evaluator-space inference error |

*exactly 7 levels].*" The quantization residual introduced by this continuous-to-discrete conversion is treated as part of the bounded prompt-discretization residual in Appendix C.8.

# C Theoretical Foundations

This appendix provides the detailed mathematical support for Section 3. The purpose is not to prove global stability of black-box LLM token generation. Instead, we prove conditional boundedness for a validation-calibrated evaluator-level surrogate induced by bounded prompt-level controls, accepted PID gains, and bounded closed-loop residuals.

## C.1 Proof Roadmap and Scope

The proof proceeds in six steps. First, we make explicit the local operating conditions S1–S6. Second, we define the augmented evaluator-level surrogate state and the local closed-loop transition matrix $\Phi_{\text{sur}}$ used in Eq. (36). Third, we distinguish the validation-fitted matrix $\widehat{\Phi}_{\text{sur}}$ from the local nominal surrogate $\Phi_{\text{sur}}$ and impose a robustness margin linking the two. Fourth, we use standard Schur-stable linear-system arguments to obtain ultimate mean-square boundedness of the augmented surrogate state under bounded aggregate residuals. Fifth, we bound the mismatch between direct references $\mathbf{r}_i(t)$, distributed estimates $\hat{\mathbf{r}}_i(t)$, and their projected control-space versions through local Lipschitz continuity of $\psi_i$. Finally, we transfer the bound from the control-space surrogate to evaluator-space task error through the calibration residual in S3.

The analysis is local in two senses. It is local in the control domain because the sensitivity condition is imposed only inside a validated neighborhood of the actually used prompt-level controls. It is also local in the task distribution because calibration, residual estimation, and gain screening are performed on held-out validation trajectories matched to the evaluated benchmarks. No statement below should be interpreted as a universal claim about arbitrary prompts, arbitrary LLMs, or arbitrary multi-agent protocols.

## C.2 Detailed MAPID-LLM Algorithm

Algorithms 1–2 provide the detailed appendix version of the MAPID-LLM runtime procedure. Algorithm 1 specifies the optional distributed prediction module used for bounded communication-delay compensation. Algorithm 2 specifies the executable inference-time PID control loop that updates prompt-level controls and maps them to textual directives. Both algorithms are local to

each agent. The evaluator provides measurement and reference signals for feedback, but it does not centrally plan, rewrite, rank, or select agents' reasoning trajectories. The PID loop uses the same control-space fusion error as Eq. (7); evaluation-space residuals are used only for diagnostics, reference generation, and surrogate calibration.

**Algorithm 1** MAPID-LLM: Distributed Delay-Compensation Prediction

**Require:** Local states $\mathbf{y}_i(t), \mathbf{c}_i(t), \mathbf{H}_i(t)$, neighbor summary $\mathbf{h}_i(t)$, maximum delay $\bar{\tau}$
**Ensure:** Predicted evaluator summaries $\{\hat{\mathbf{y}}_i(t+k \mid t)\}_{k=1}^{\bar{\tau}}$
Initialize $\mathbf{c}_i(0)$, $\mathbf{e}_i(-1) = \mathbf{0}$, and $\hat{\mathbf{y}}_i(k \mid 0) = \mathbf{0}$ for $k = 1, \ldots, \bar{\tau}$.
**for** $t = 0, 1, 2, \ldots$ **do**
  **for** each agent $i \in \mathcal{N}$ in parallel **do**
    Predict future evaluator-side states:

$$\hat{\mathbf{y}}_i(t+k \mid t) = \mathcal{P}_\theta\big(\mathbf{y}_i(t), \mathbf{c}_i(t), \mathbf{H}_i(t), \mathbf{h}_i(t), k\big), \quad k = 1, \ldots, \bar{\tau}.$$

    Send predicted evaluator summaries and the current control summary to neighbors.
  **end for**
**end for**

## C.3 Evaluator-Shaped Reference Interface

This subsection gives the implementation details that are abstracted in Section 2.3. At each inference step, the evaluator maps each agent output $o_i(t)$ to a normalized feedback vector $\mathbf{y}_i(t)$ covering task quality, constraint satisfaction, and coordination quality. A lightweight diagnostic interface aggregates these measurements, control states, and history summaries into a compact task-level state:

$$\hat{\mathbf{s}}_{\text{sys}}(t+1) = \mathcal{F}_{\text{sys}}\Big(\{\mathbf{y}_i(t)\}_{i=1}^N, \{\mathbf{c}_i(t)\}_{i=1}^N, \mathbf{H}_{\text{sys}}(t); \theta_{\text{sys}}\Big). \tag{15}$$

Here $\mathbf{H}_{\text{sys}}(t)$ denotes historical multi-agent summaries, $\theta_{\text{sys}}$ denotes the diagnostic-interface parameters, and $\hat{\mathbf{s}}_{\text{sys}}(t+1)$ summarizes task-level quality and cooperation diagnostics.

The role-specific reference generator maps this diagnostic state, the agent role prompt, and lightweight neighbor summaries to an agent-specific semantic reference:

$$\mathbf{r}_i(t{+}1) = \mathcal{F}_{\text{ref}}\Big(\hat{\mathbf{s}}_{\text{sys}}(t{+}1), \mathbf{h}_i(t), \text{Prompt}_i; \theta_{\text{ref}}\Big). \tag{16}$$

The neighbor summary is

$$\mathbf{h}_i(t) = \text{AGGREGATE}\big\{\mathbf{y}_j^{\text{sum}}(t-1) : j \in \mathcal{N}_i\big\}, \tag{17}$$

**Algorithm 2** MAPID-LLM: Inference-Time PID Control and Prompt Actuation

**Require:** Controls $\mathbf{c}_i(t)$; reference estimates $\hat{\mathbf{r}}_i(t)$; weights $w_{ij}$; PID gains; bounds
**Ensure:** Updated controls $\mathbf{c}_i(t+1)$ and generated outputs $o_i(t+1)$
**for** $t = 0, 1, 2, \ldots$ **do**
  **for** each agent $i \in \mathcal{N}$ in parallel **do**
    Obtain $\mathbf{r}_i(t)$ and update the local estimate $\hat{\mathbf{r}}_i(t)$.
    Receive neighbor summaries and construct $\tilde{\mathbf{c}}_j(t)$ for $j \in \mathcal{N}_i$.
    Map the reference to control space:

$$\mathbf{r}_{i,c}(t) = \psi_i(\hat{\mathbf{r}}_i(t)).$$

    Compute the fused control error:

$$\mathbf{e}_i(t) = \alpha(\mathbf{r}_{i,c}(t) - \mathbf{c}_i(t)) + (1-\alpha) \sum_{j \in \mathcal{N}_i} w_{ij}(\tilde{\mathbf{c}}_j(t) - \mathbf{c}_i(t)).$$

    Update the clipped integral memory:

$$\mathbf{I}_i(t) = \text{clip}_{[-\mathbf{I}_{\max}, \mathbf{I}_{\max}]}\big(\mathbf{I}_i(t-1) + \mathbf{e}_i(t)\big).$$

    Compute $\Delta\mathbf{e}_i(t) = \mathbf{e}_i(t) - \mathbf{e}_i(t-1)$.
    Compute the PID increment:

$$\mathbf{u}_i(t) = \mathbf{K}_{P,i}\mathbf{e}_i(t) + \mathbf{K}_{I,i}\mathbf{I}_i(t) + \mathbf{K}_{D,i}\Delta\mathbf{e}_i(t).$$

    Saturate and project the prompt-level control:

$$\mathbf{u}_i^{\text{sat}}(t) = \text{sat}_{[\mathbf{u}_{\min}, \mathbf{u}_{\max}]}(\mathbf{u}_i(t)),$$
$$\mathbf{c}_i(t+1) = \Pi_{\mathcal{C}}\big(\mathbf{c}_i(t) + \mathbf{u}_i^{\text{sat}}(t)\big).$$

    Map $\mathbf{c}_i(t+1)$ to textual directives and generate $o_i(t+1)$.
    Evaluate $o_i(t+1)$ to obtain $\mathbf{y}_i(t+1)$.
    Broadcast measurement and control summaries.
  **end for**
**end for**

where $\mathbf{y}_j^{\text{sum}}(t-1) \in \mathbb{R}^d$ is a fixed-size summary vector.

To maintain local consistency while preserving agent-specific roles, each agent updates its locally available reference estimate through neighbor alignment and local innovation:

$$\hat{\mathbf{r}}_i(t+1) = \hat{\mathbf{r}}_i(t) + \gamma \sum_{j \in \mathcal{N}_i} w_{ij}\big(\hat{\mathbf{r}}_j(t) - \hat{\mathbf{r}}_i(t)\big) + \beta_r\big(\mathbf{r}_i(t+1) - \hat{\mathbf{r}}_i(t)\big), \tag{18}$$

where $\gamma \in (0,1)$ is an alignment gain and $\beta_r \in (0,1)$ is an innovation gain. The resulting estimate $\hat{\mathbf{r}}_i(t)$ is projected as $\mathbf{r}_{i,c}(t) = \psi_i(\hat{\mathbf{r}}_i(t))$ and then used in the fused control error in Eq. (7).

In experiments, $\mathcal{F}_{\text{sys}}$ is implemented as a prompted zero-shot meta-evaluator with a fixed output schema, and $\mathcal{F}_{\text{ref}}$ outputs reference signals rather than executable action plans. These modules do not rewrite agent reasoning, rank candidate trajectories, select final answers, or replace the underlying communication protocol. Their outputs

are projected into the control space through $\psi_i(\cdot)$ and then consumed by the local PID controller.

**Interface boundary.** The evaluator-side interface may return normalized task-quality scores, constraint-satisfaction scores, coordination-quality scores, role-specific reference signals, and fixed-size summaries for delay compensation. It may not directly edit an agent's response, choose among candidate trajectories, select the final answer, or issue an executable plan that replaces the underlying communication protocol. In implementation, this boundary is enforced by a fixed output schema: evaluator outputs are parsed only as numeric measurements, compact summaries, or semantic references, and the local controller converts them into prompt-level control updates through $\psi_i(\cdot)$ and Eq. (7). This clarification is used only to specify the measurement/reference channel; it does not introduce an additional centralized controller.

### C.4 Standing Assumptions S1–S6

The assumptions below are local, validation-calibrated conditions. They are used only to support the conditional boundedness claims in Section 3.

**Assumption C.1** (S1: Bounded Control and Evaluation Interface)**.** For each agent $i \in \mathcal{N}$, the prompt-level control vector satisfies $\mathbf{c}_i(t) \in \mathcal{C}$, where $\mathcal{C} \subset \mathbb{R}^k$ is a compact box. The PID increment is component-wise saturated before application, and the updated control is projected back to $\mathcal{C}$. The integral state is clipped component-wise, so that $\mathbf{I}_i(t) \in [-\mathbf{I}_{\max}, \mathbf{I}_{\max}]$ for a finite vector $\mathbf{I}_{\max}$. Evaluator outputs are normalized as $\mathbf{y}_i(t) \in [0,1]^m$. For empirical directed or row-normalized topologies, the neighbor weights satisfy $w_{ij} \geq 0$ and $\sum_{j \in \mathcal{N}_i} w_{ij} = 1$. For the graph-regularized equilibrium theorem only, we restrict to an undirected connected graph with symmetric nonnegative weights $w_{ij} = w_{ji}$, so that the coordination term is induced by a symmetric weighted Laplacian. Communication delays, if present, are uniformly bounded by $\bar{\tau}$. For the theoretical gain definitions, $\alpha \in [\alpha_{\min}, 1]$ with $\alpha_{\min} > 0$; $\alpha = 0$ is reserved only for empirical ablations.

**Assumption C.2** (S2: Local Smoothed-Surrogate Sensitivity)**.** Within a local validation neighborhood $\mathcal{D} \subseteq \mathcal{C}$, the conditional mean evaluator response admits a smoothed surrogate

$$\bar{\mathbf{y}}_i(\mathbf{c}) = \mathbb{E}_{\omega,\nu,\tau}\left[\mathcal{E}\left(\mathcal{F}_{\mathrm{LLM}}(\mathbf{x}_i^{\mathrm{gen}}(\mathbf{c}))\right) \mid \mathbf{c}\right], \quad (19)$$

where the expectation is over decoding randomness $\omega$, evaluator stochasticity $\nu$, and validation-task sampling $\tau$ under fixed local context and decoding configuration. There exists a validation-derived local sensitivity constant $L_{\mathrm{sur}} > 0$ such that

$$\begin{aligned} &\|\bar{\mathbf{y}}_i(\mathbf{c}_1) - \bar{\mathbf{y}}_i(\mathbf{c}_2)\| \\ &\quad \leq L_{\mathrm{sur}} \|\mathbf{c}_1 - \mathbf{c}_2\|, \quad \forall \mathbf{c}_1, \mathbf{c}_2 \in \mathcal{D}. \end{aligned} \quad (20)$$

This condition is imposed only on the smoothed evaluator-level surrogate, not on the raw token sequence or on the global black-box LLM mapping.

**Assumption C.3** (S3: Control-to-Evaluation Calibration)**.** For each agent $i$, the projection $\psi_i$ is locally calibrated on held-out validation tasks. The evaluator-space desired signal is defined as $\mathbf{y}_i^{\mathrm{des}}(t) = \chi_i(\mathbf{r}_i(t))$, where $\chi_i$ maps semantic references into evaluator-score space; if $\mathbf{r}_i(t)$ is already expressed in evaluator space, $\chi_i$ is the identity map. There exist constants $L_{\mathrm{cal},i} > 0$ and $\delta_{\mathrm{cal},i} \geq 0$ such that

$$\begin{aligned} \left\|\bar{\mathbf{y}}_i(\mathbf{c}_i(t)) - \mathbf{y}_i^{\mathrm{des}}(t)\right\| &\leq L_{\mathrm{cal},i} \|\mathbf{c}_i(t) - \psi_i(\mathbf{r}_i(t))\| \\ &\quad + \delta_{\mathrm{cal},i}. \end{aligned} \quad (21)$$

In addition, $\psi_i$ is locally Lipschitz in the validated reference region: there exists $L_{\psi,i} > 0$ such that

$$\|\psi_i(\mathbf{a}) - \psi_i(\mathbf{b})\| \leq L_{\psi,i} \|\mathbf{a} - \mathbf{b}\| \quad (22)$$

for all reference vectors $\mathbf{a}, \mathbf{b}$ in that region. The residual $\delta_{\mathrm{cal},i}$ captures the irreducible mismatch between prompt-level controls and evaluator-space objectives and is not assumed to vanish.

**Assumption C.4** (S4: Bounded Closed-Loop Residuals)**.** Let $\tilde{\boldsymbol{\eta}}(t)$ denote the aggregate surrogate input in Eq. (36). It collects sampling noise, evaluator noise, surrogate-prediction residuals, prompt discretization, delay-compensation error, local linearization residuals, and bounded reference drift. There exist finite constants $\tilde{\sigma}_\eta^2$ and $\bar{\eta}$ such that

$$\mathbb{E}[\|\tilde{\boldsymbol{\eta}}(t)\|^2 \mid \mathcal{F}_t] \leq \tilde{\sigma}_\eta^2, \qquad \|\mathbb{E}[\tilde{\boldsymbol{\eta}}(t) \mid \mathcal{F}_t]\| \leq \bar{\eta}. \quad (23)$$

**Assumption C.5** (S5: Robust Validation-Screened Surrogate Stability)**.** For each candidate PID gain tuple, the closed-loop surrogate matrix $\widehat{\Phi}_{\mathrm{sur}}$ is fitted on held-out validation trajectories. The selected PID gains are accepted only if the empirical residual level remains below the chosen validation threshold and the fitted surrogate satisfies a robust

Schur-margin condition. Specifically, in the validated operating region, there exists an estimation-gap bound

$$\left\|\Phi_{\mathrm{sur}} - \widehat{\Phi}_{\mathrm{sur}}\right\| \le \Delta_\Phi \tag{24}$$

and a robustness constant $\kappa_\Phi > 0$ such that

$$\begin{aligned}\rho\Big(\widehat{\Phi}_{\mathrm{sur}}(\mathbf{K}_{\mathrm{PID}}, \mathcal{W}, \alpha, \bar{\tau})\Big) + \kappa_\Phi \Delta_\Phi \\ \le 1 - \varepsilon_\Phi, \qquad \varepsilon_\Phi > 0.\end{aligned} \tag{25}$$

Consequently, the local nominal surrogate used in the proof satisfies

$$\rho(\Phi_{\mathrm{sur}}) \le 1 - \varepsilon'_\Phi < 1$$

for some $\varepsilon'_\Phi > 0$. This is a local validation-screening condition for the evaluated operating region, not a claim that the true black-box LLM dynamics are globally Schur stable.

**Assumption C.6** (S6: Graph and Reference Regularity)**.** For the equilibrium result, the graph is undirected and connected with symmetric nonnegative weights. For the bounded-reference-tracking result, the reference-estimation matrix induced by the alignment and innovation gains is contractive. Specifically, there exist $M_r > 0$ and $\bar{\rho} \in (0, 1)$ such that

$$\|\mathbf{A}_r^t\| \le M_r \bar{\rho}^t. \tag{26}$$

Semantic references are bounded and have bounded per-round variation:

$$\|\mathbf{r}_i(t)\| \le R_{\max}, \qquad \|\mathbf{r}_i(t+1) - \mathbf{r}_i(t)\| \le B_r. \tag{27}$$

Consequently, the estimation mismatch $\|\hat{\mathbf{r}}_i(t) - \mathbf{r}_i(t)\|$ is ultimately bounded by a constant depending on $B_r$, the innovation gain, the alignment gain, and the graph operator. Here $t$ indexes reasoning/-control rounds rather than physical time.

**Operational interpretation of S1–S6.** S1 is enforced by clipping the PID increment, projecting controls back to the feasible prompt-control box, and normalizing evaluator outputs. S2 is a local smoothness condition on the validation-smoothed evaluator surrogate; it is checked by perturbing prompt-level controls on held-out tasks rather than by assuming smooth raw token generation. S3 states that the semantic-to-control projection is calibrated only within the evaluated reference region, with an explicit residual for mismatch between control variables and evaluator objectives. S4 collects unavoidable stochastic and approximation effects—sampling noise, evaluator noise, prompt discretization, delay prediction error, local linearization error, and reference drift—into one bounded aggregate input. S5 is the held-out gain-screening condition: a PID gain tuple is used only if the fitted surrogate has a robust Schur margin and acceptable residuals. S6 supplies the graph and reference regularity needed to keep distributed reference estimates bounded. These interpretations do not strengthen the assumptions; they explain how each local condition is supported by the validation protocol used in the experiments.

### C.5 Construction of the Augmented Surrogate Matrix

This subsection clarifies the compact surrogate form used in Theorem 3.1. Let the stacked evaluator output, control vector, reference estimate, projected reference estimate, and direct projected reference be

$$\begin{aligned}\mathbf{Y}(t) &= [\mathbf{y}_1(t)^\top, \dots, \mathbf{y}_N(t)^\top]^\top, \\ \mathbf{C}(t) &= [\mathbf{c}_1(t)^\top, \dots, \mathbf{c}_N(t)^\top]^\top.\end{aligned}$$

$$\begin{aligned}\hat{\mathbf{R}}(t) &= [\hat{\mathbf{r}}_1(t)^\top, \dots, \hat{\mathbf{r}}_N(t)^\top]^\top, \\ \widehat{\mathbf{R}}_c(t) &= [\psi_1(\hat{\mathbf{r}}_1(t))^\top, \dots, \psi_N(\hat{\mathbf{r}}_N(t))^\top]^\top.\end{aligned}$$

$$\mathbf{R}_c(t) = [\psi_1(\mathbf{r}_1(t))^\top, \dots, \psi_N(\mathbf{r}_N(t))^\top]^\top.$$

The augmented evaluator-level surrogate state is

$$\begin{aligned}\boldsymbol{\xi}(t) = \big[&\mathbf{Y}(t)^\top, \mathbf{C}(t)^\top, \mathbf{I}(t)^\top, \mathbf{e}_c(t)^\top, \mathbf{e}_c^-(t)^\top, \\ &\hat{\mathbf{R}}(t)^\top, \widehat{\mathbf{R}}_c(t)^\top\big]^\top.\end{aligned} \tag{28}$$

where $\mathbf{e}_c^-(t) = \mathbf{e}_c(t-1)$. Including both $\hat{\mathbf{R}}(t)$ and $\widehat{\mathbf{R}}_c(t)$ closes the local surrogate even when $\psi_i$ is only locally approximated. The local affine approximation error of $\psi_i$ is included in the aggregate residual.

For symmetric weights, the graph-induced coordination term can be written using a weighted Laplacian $\mathcal{L}_w$. The stacked control-space error takes the form

$$\mathbf{e}_c(t) = \alpha(\widehat{\mathbf{R}}_c(t) - \mathbf{C}(t)) - (1-\alpha)(\mathcal{L}_w \otimes \mathbf{I}_k)\mathbf{C}(t). \tag{29}$$

For directed or row-normalized empirical topologies, the implemented operator is

$$\begin{aligned}\mathbf{e}_c(t) = \alpha\big(\widehat{\mathbf{R}}_c(t) - \mathbf{C}(t)\big) \\ + (1-\alpha)\big((\mathcal{W} - \mathbf{I}_N) \otimes \mathbf{I}_k\big)\mathbf{C}(t).\end{aligned} \tag{30}$$

whereas the equilibrium theorem uses the symmetric Laplacian form in Eq. (29).

The saturated projected PID step can be locally represented, inside the validation-calibrated region, by an affine surrogate plus a bounded residual:

$$\begin{aligned}\mathbf{C}(t+1) = \mathbf{A}_c\mathbf{C}(t) + \mathbf{B}_P\mathbf{e}_c(t) \\ + \mathbf{B}_I\mathbf{I}(t) + \mathbf{B}_D\big(\mathbf{e}_c(t) - \mathbf{e}_c^-(t)\big) \\ + \boldsymbol{\eta}_c(t).\end{aligned} \tag{31}$$

The distributed reference-estimation dynamics are written as

$$\hat{\mathbf{R}}(t+1) = \mathbf{A}_r\hat{\mathbf{R}}(t) + \beta_r\mathbf{R}(t+1) + \boldsymbol{\eta}_r(t), \tag{32}$$

where $\mathbf{R}(t+1)$ stacks the newly generated direct semantic references. The projected reference dynamics are represented locally as

$$\widehat{\mathbf{R}}_c(t+1) = \mathbf{A}_{rc}\widehat{\mathbf{R}}_c(t) + \mathbf{B}_{rc}\mathbf{R}_c(t+1) + \boldsymbol{\eta}_{rc}(t). \tag{33}$$

This equation is not an independent reference update law. It is the local affine approximation of

$$\begin{aligned}\widehat{\mathbf{R}}_c(t+1) = \Psi(\hat{\mathbf{R}}(t+1)), \\ \Psi(\hat{\mathbf{R}}) = [\psi_1(\hat{\mathbf{r}}_1)^\top, \ldots, \psi_N(\hat{\mathbf{r}}_N)^\top]^\top.\end{aligned}$$

around the validation trajectory. The residual $\boldsymbol{\eta}_{rc}(t)$ absorbs bounded reference drift, projection mismatch, and local affine-approximation error.

Locally around a validation trajectory, the smoothed evaluator response is approximated by

$$\begin{aligned}\mathbf{Y}(t+1) = \mathbf{A}_y\mathbf{Y}(t) + \mathbf{B}_c\mathbf{C}(t+1) \\ + \mathbf{D}_r\widehat{\mathbf{R}}_c(t) + \boldsymbol{\eta}_y(t).\end{aligned} \tag{34}$$

where $\mathbf{A}_y$, $\mathbf{B}_c$, and $\mathbf{D}_r$ are validation-fitted local response blocks. The clipped PID memory evolves causally as

$$\begin{aligned}\mathbf{I}(t+1) = \mathrm{clip}_{[-\mathbf{I}_{\max},\mathbf{I}_{\max}]}\big(\mathbf{I}(t) + \mathbf{e}_c(t)\big) + \boldsymbol{\eta}_I(t), \\ \mathbf{e}_c^-(t+1) = \mathbf{e}_c(t).\end{aligned} \tag{35}$$

Combining the local evaluator response, projected control update, reference-estimation dynamics, projected-reference surrogate, graph operator, integral memory, and derivative-memory equation yields the compact closed-loop surrogate

$$\boldsymbol{\xi}(t+1) = \Phi_{\mathrm{sur}}(\mathbf{K}_{\mathrm{PID}}, \mathcal{W}, \alpha, \bar{\tau})\boldsymbol{\xi}(t) + \Gamma\tilde{\boldsymbol{\eta}}(t). \tag{36}$$

We do not require its entries to be known analytically from model internals; they are estimated and screened using held-out trajectories. All mismatch between this local linear surrogate and the observed closed loop is included in $\tilde{\boldsymbol{\eta}}(t)$.

### C.6 Formal Auxiliary Results

**Lemma C.7** (Ultimate Boundedness of the Augmented Surrogate State)**.** *Under S1, S2, S4, and S5, the augmented evaluator-level surrogate state is ultimately bounded in mean square. Specifically, there exist constants $T_1 \geq 0$ and $B_\xi^2 < \infty$ such that*

$$\mathbb{E}\left[\|\boldsymbol{\xi}(t)\|^2\right] \leq B_\xi^2, \quad \forall t \geq T_1. \tag{37}$$

*Consequently, evaluator-derived output fluctuations are also ultimately bounded because $\mathbf{Y}(t)$ is a component of $\boldsymbol{\xi}(t)$. One admissible scaling is*

$$\begin{aligned}B_\xi^2 = O\Big(\|\Gamma\|^2\tilde{\sigma}_\eta^2 + \|\Gamma\|^2\bar{\eta}^2 + \Delta_\Phi^2 \\ + B_r^2 + B_{\mathrm{sum}}^2 + S_{\max}^2\Big).\end{aligned} \tag{38}$$

*where $\tilde{\sigma}_\eta^2$ denotes the second-moment bound of the aggregate surrogate input. This is an ultimate boundedness result, not a zero-variance or zero-error convergence claim.*

**Lemma C.8** (Bounded Tracking of Slowly Varying Semantic References)**.** *Under S6 and a contractive reference-estimation matrix, the distributed semantic-reference estimate remains bounded around the graph-smoothed equilibrium $\mathbf{R}^*_{\mathrm{reg}}(t)$:*

$$\begin{aligned}\left\|\hat{\mathbf{R}}(t) - \mathbf{R}^*_{\mathrm{reg}}(t)\right\| \leq K\bar{\rho}^t\left\|\hat{\mathbf{R}}(0) - \mathbf{R}^*_{\mathrm{reg}}(0)\right\| \\ + \frac{C_r B_r}{1-\bar{\rho}}.\end{aligned} \tag{39}$$

*where $K > 0$, $C_r > 0$, and $0 < \bar{\rho} < 1$. If $B_r = 0$, the estimate converges exponentially; under heterogeneous references, the equilibrium is a topology-weighted compromise rather than exact per-agent equality.*

### C.7 Validation-Screened Gain Selection

For the theoretical analysis, we use a canonical validation-screened family of diagonal PID gains. In the scalar isotropic form used to generate candidates, the proportional gain is

$$\mathbf{K}_P = \frac{\beta_p}{\alpha}\mathbf{I}_k, \tag{40}$$

where $\beta_p > 0$ and $\alpha \in [\alpha_{\min}, 1]$. The integral gain is damped by the graph-coupling scale:

$$\mathbf{K}_I = \frac{\beta_i}{\alpha(1+\mu_{\mathrm{graph}})}\mathbf{K}_P, \tag{41}$$

where $\mu_{\mathrm{graph}}$ is taken as $\lambda_{\max}(\mathcal{L}_w)$ in the symmetric theoretical case and as the corresponding

local operator-norm estimate for directed empirical topologies. The derivative gain is scaled by the locally estimated evaluator-level surrogate sensitivity:

$$\mathbf{K}_D = \beta_d \sqrt{\frac{L_{\text{sur}}}{\|\mathbf{K}_P\|_F}} \mathbf{K}_P. \tag{42}$$

Here $L_{\text{sur}}$ is a validation-derived local sensitivity constant for the smoothed evaluator-level surrogate, not a global smoothness constant of the raw LLM token generator. These formulas define a candidate gain family, not a necessary condition for stability.

Gain selection is performed before evaluation on held-out validation trajectories. For each candidate tuple $(\beta_p, \beta_i, \beta_d, \alpha)$, we:

1. construct the corresponding gain matrices using Eqs. (40)–(42);
2. run the controller on validation tasks and estimate the local closed-loop surrogate matrix $\widehat{\Phi}_{\text{sur}}$;
3. estimate the residual level $\widehat{\tilde{\sigma}}_\eta^2$ between observed validation trajectories and the surrogate prediction;
4. accept the gains only if the robust margin in Eq. (25) holds and $\widehat{\tilde{\sigma}}_\eta^2$ remains within the chosen validation tolerance.

This procedure is deliberately conservative. The Schur condition is not asserted online for every possible prompt. It is a screening criterion for the operating region actually used in the experiments.

**Practical validation protocol.** For each benchmark and backbone, the validation protocol freezes the base prompts, communication topology, decoding configuration, and evaluator schema before gain screening. Candidate PID gains are then evaluated on held-out trajectories that are not used for final testing. For each candidate, we record the stacked evaluator states, prompt-control vectors, reference estimates, and one-step residuals; fit the block-level surrogate transition matrix by regularized least squares; compute the empirical residual tolerance; and retain only candidates satisfying Eq. (25). The selected gains are fixed before test-time evaluation. Thus, the validation step screens a local operating region and does not adapt gains using test outcomes.

**Residual decomposition.** The aggregate surrogate input in S4 can be written schematically as

$$\begin{aligned}\tilde{\boldsymbol{\eta}}(t) = \boldsymbol{\eta}_{\text{gen}}(t) + \boldsymbol{\eta}_{\text{eval}}(t) + \boldsymbol{\eta}_{\text{disc}}(t) \\ + \boldsymbol{\eta}_{\text{delay}}(t) + \boldsymbol{\eta}_{\text{lin}}(t) + \boldsymbol{\eta}_{\text{ref}}(t).\end{aligned}$$

where the terms denote decoding/evaluator randomness, evaluator scoring noise, prompt discretization, delay-compensation prediction error, local surrogate linearization error, and bounded reference drift, respectively. Projection and clipping keep the control-dependent components bounded, while the validation residual estimate upper-bounds the combined effect in the evaluated operating region.

### C.8 Projection, Saturation, and Prompt Discretization

Projection and saturation are feasibility-preserving operations. For the control box $\mathcal{C} = [\underline{\mathbf{c}}, \bar{\mathbf{c}}]$, Euclidean projection is non-expansive:

$$\|\Pi_{\mathcal{C}}(\mathbf{x}) - \Pi_{\mathcal{C}}(\mathbf{z})\|_2 \le \|\mathbf{x} - \mathbf{z}\|_2 . \tag{43}$$

Thus projection cannot increase the distance between two control trajectories. The additional approximation caused by converting continuous controls into discrete textual directives is treated as a bounded prompt-discretization residual and included in $\tilde{\boldsymbol{\eta}}(t)$.

### C.9 Empirical Validation of Local Sensitivity

The local sensitivity condition in S2 is checked through controlled prompt-level perturbations on held-out tasks. For each validation prompt, we repeat generation under the baseline control vector and under single-dimension perturbations. The empirical sensitivity estimate is

$$\hat{L}_{\text{sur}} = \frac{\|\bar{\mathbf{y}}_i' - \bar{\mathbf{y}}_i\|_2}{\|\Delta \mathbf{c}_i\|_2}. \tag{44}$$

The selected value of $L_{\text{sur}}$ is taken as a conservative empirical upper quantile within the evaluated operating region. This supports a local smoothed-sensitivity bound; it is not a claim of global Lipschitz continuity for all prompts or models.

**Table 4:** Illustrative local sensitivity summary used for validation-screened gain selection.

| Control variable group | Median $\hat{L}_{\text{sur}}$ | IQR | Conservative bound |
|---|---|---|---|
| Structural constraints | 2.9 | [2.4, 3.3] | 4.2 |
| Semantic objectives | 3.0 | [2.5, 3.5] | 4.4 |
| Cooperation intensity | 3.1 | [2.6, 3.7] | 4.6 |
| Verification strength | 3.2 | [2.7, 3.8] | 4.8 |

### C.10 Proof of Lemma C.7

*Proof.* Under S5, the robust validation-screening condition implies that the local nominal surrogate transition matrix is Schur stable: $\rho(\Phi_{\text{sur}}) \le 1 - \varepsilon'_{\Phi} < 1$. Therefore, there exist constants $M_{\Phi} > 0$ and $\lambda \in (0,1)$ such that

$$\|\Phi_{\text{sur}}^{t}\| \le M_{\Phi}\lambda^{t}. \tag{45}$$

Unrolling Eq. (36) gives

$$\boldsymbol{\xi}(t) = \Phi_{\text{sur}}^{t}\boldsymbol{\xi}(0) + \sum_{\tau=0}^{t-1} \Phi_{\text{sur}}^{t-1-\tau}\Gamma\tilde{\boldsymbol{\eta}}(\tau). \tag{46}$$

Taking second moments and using S4 yields, up to norm-dependent universal constants,

$$\begin{aligned}\mathbb{E}\|\boldsymbol{\xi}(t)\|^2 \le\ & 2M_{\Phi}^2\lambda^{2t}\|\boldsymbol{\xi}(0)\|^2 \\ &+ \frac{2M_{\Phi}^2\|\Gamma\|^2\tilde{\sigma}_{\eta}^2}{1-\lambda^2} \\ &+ \frac{2M_{\Phi}^2\|\Gamma\|^2\bar{\eta}^2}{(1-\lambda)^2}.\end{aligned} \tag{47}$$

Hence the augmented surrogate state is ultimately bounded in mean square. Because $\mathbf{Y}(t)$ is a component of $\boldsymbol{\xi}(t)$, evaluator-score fluctuations inherit an ultimate second-moment bound. Additional bounded terms from reference drift, neighbor summaries, saturation, prompt discretization, and the validation-estimation gap are absorbed into $B_r$, $B_{\text{sum}}$, $S_{\max}$, and $\Delta_{\Phi}$, giving the scaling in Eq. (38). □

### C.11 Proof of Lemma C.8

*Proof.* Stack the reference estimates as $\hat{\mathbf{R}}(t) = [\hat{\mathbf{r}}_1(t)^{\top}, \dots, \hat{\mathbf{r}}_N(t)^{\top}]^{\top}$ and the local references as $\mathbf{R}(t)$. The update in Eq. (18) can be written as

$$\hat{\mathbf{R}}(t+1) = \mathbf{A}_r\hat{\mathbf{R}}(t) + \beta_r\mathbf{R}(t+1), \tag{48}$$

where $\mathbf{A}_r = ((1-\beta_r)\mathbf{I}_N - \gamma\mathcal{L}_w) \otimes \mathbf{I}_m$ in the symmetric case. By S6, $\|\mathbf{A}_r^t\| \le M_r\bar{\rho}^t$ for some $\bar{\rho} \in (0,1)$. For a fixed reference stack $\mathbf{R}(t+1)$, the instantaneous fixed point satisfies

$$\mathbf{R}_{\text{reg}}^{*}(t+1) = \mathbf{A}_r\mathbf{R}_{\text{reg}}^{*}(t+1) + \beta_r\mathbf{R}(t+1).$$

Subtracting the fixed-point relation from Eq. (48) and allowing the fixed point to move with the slowly varying reference gives

$$\|\hat{\mathbf{R}}(t+1) - \mathbf{R}_{\text{reg}}^{*}(t+1)\| \le \bar{\rho}\|\hat{\mathbf{R}}(t) - \mathbf{R}_{\text{reg}}^{*}(t)\| + C_rB_r. \tag{49}$$

Iterating the recursion yields Eq. (39). The bounded variation assumption also implies an ultimate bound on the direct estimation mismatch $\|\hat{\mathbf{r}}_i(t) - \mathbf{r}_i(t)\|$, with a constant depending on $B_r$, $\beta_r$, $\gamma$, and the graph operator. If $B_r = 0$, the second term vanishes and the convergence is exponential. □

### C.12 Proof of Theorem 3.1

*Proof.* Decompose the evaluator-space inference error as

$$\begin{aligned}\left\|\mathbf{e}_i^{\text{infer}}(t)\right\| &= \left\|\mathbf{y}_i^{\text{des}}(t) - \mathbf{y}_i(t)\right\| \\ &\le \left\|\mathbf{y}_i^{\text{des}}(t) - \bar{\mathbf{y}}_i(\mathbf{c}_i(t))\right\| \\ &\quad + \|\bar{\mathbf{y}}_i(\mathbf{c}_i(t)) - \mathbf{y}_i(t)\|.\end{aligned} \tag{50}$$

By the calibration condition S3,

$$\left\|\mathbf{y}_i^{\text{des}}(t) - \bar{\mathbf{y}}_i(\mathbf{c}_i(t))\right\| \le L_{\text{cal},i}\,\|\mathbf{c}_i(t) - \psi_i(\mathbf{r}_i(t))\| + \delta_{\text{cal},i}.$$

Because the executable controller tracks $\psi_i(\hat{\mathbf{r}}_i(t))$, we insert and subtract this term:

$$\begin{aligned}\|\mathbf{c}_i(t) - \psi_i(\mathbf{r}_i(t))\| \le\ & \|\mathbf{c}_i(t) - \psi_i(\hat{\mathbf{r}}_i(t))\| \\ &+ \|\psi_i(\hat{\mathbf{r}}_i(t)) - \psi_i(\mathbf{r}_i(t))\|.\end{aligned} \tag{51}$$

By the local Lipschitz condition on $\psi_i$ in S3,

$$\|\psi_i(\hat{\mathbf{r}}_i(t)) - \psi_i(\mathbf{r}_i(t))\| \le L_{\psi,i}\,\|\hat{\mathbf{r}}_i(t) - \mathbf{r}_i(t)\|.$$

The first term is controlled by the bounded augmented surrogate state because $\mathbf{C}(t)$ and the projected-reference components are included in $\boldsymbol{\xi}(t)$. The second term is controlled by the reference-estimation bound from S6 and Lemma C.8. The stochastic evaluator fluctuation term $\|\bar{\mathbf{y}}_i(\mathbf{c}_i(t)) - \mathbf{y}_i(t)\|$ is absorbed into the aggregate surrogate residual and bounded through Lemma C.7. Combining these bounds yields

$$\sup_{t \ge T_0} \mathbb{E}\left[\left\|\mathbf{e}_i^{\text{infer}}(t)\right\|^2\right] \le B_i^2$$

for some finite $B_i$ after a finite transient $T_0$. The high-probability statement follows directly from Markov's inequality:

$$\Pr(\|\mathbf{e}_i^{\text{infer}}(t)\| > B_i/\sqrt{\delta}) \le \delta.$$

□

### C.13 Proof of Theorem 3.2

*Proof.* Under the symmetric connected graph condition, $\mathcal{L}_w$ is symmetric positive semidefinite. Consider the graph-regularized objective

$$J(\mathbf{C}) = \frac{\alpha}{2}\|\mathbf{C} - \mathbf{R}_c^*\|^2 + \frac{1-\alpha}{2}\mathbf{C}^\top(\mathcal{L}_w \otimes \mathbf{I}_k)\mathbf{C}, \quad \mathbf{C} \in \mathcal{C}^N.$$

Because $\alpha > 0$ and $\mathcal{C}^N$ is a compact convex box, the constrained problem has a unique minimizer. The uniqueness follows from strong convexity: $\alpha\mathbf{I}_{Nk} + (1-\alpha)(\mathcal{L}_w \otimes \mathbf{I}_k)$ is positive definite.

If the constrained minimizer lies in the interior of $\mathcal{C}^N$, the KKT condition reduces to the unconstrained first-order condition. The gradient is

$$\nabla J(\mathbf{C}) = \alpha(\mathbf{C} - \mathbf{R}_c^*) + (1-\alpha)(\mathcal{L}_w \otimes \mathbf{I}_k)\mathbf{C}.$$

Setting $\nabla J(\mathbf{C}) = 0$ gives

$$[\alpha\mathbf{I}_{Nk} + (1-\alpha)(\mathcal{L}_w \otimes \mathbf{I}_k)]\,\mathbf{C}^* = \alpha\mathbf{R}_c^*.$$

Since the matrix on the left is positive definite and invertible, the interior solution has the closed form

$$\mathbf{C}^* = (\alpha\mathbf{I}_{Nk} + (1-\alpha)(\mathcal{L}_w \otimes \mathbf{I}_k))^{-1}\,\alpha\mathbf{R}_c^*. \tag{52}$$

If the unconstrained solution falls outside the box, the projected equilibrium is characterized by the normal-cone KKT condition

$$\mathbf{0} \in \alpha(\mathbf{C}^* - \mathbf{R}_c^*) + (1-\alpha)(\mathcal{L}_w \otimes \mathbf{I}_k)\mathbf{C}^* + N_{\mathcal{C}^N}(\mathbf{C}^*), \tag{53}$$

where $N_{\mathcal{C}^N}(\mathbf{C}^*)$ is the normal cone of the box constraint at $\mathbf{C}^*$. Thus, in the boundary case, the unprojected fusion error is not required to vanish; the equilibrium is a projected equilibrium induced by the box constraint. If all projected references are identical, the graph smoothness term is minimized by consensus. If references are heterogeneous, the solution is generally a compromise rather than exact consensus. □

### C.14 How the Theorems Work Together

The theoretical chain is as follows. S1–S5 define a bounded local closed-loop surrogate with robustly validation-screened PID gains. Lemma C.7 then controls the augmented evaluator-level surrogate state under bounded aggregate inputs. S6 controls the movement of semantic references, and Lemma C.8 bounds the distributed reference estimates around their graph-smoothed equilibrium. Theorem 3.1 combines these two ingredients with the control-to-evaluation calibration residual. Theorem 3.2 is complementary: it explains the zero-residual local reference–coordination compromise between individual reference tracking and neighborhood coordination, thereby connecting the theory to the fusion-coefficient sensitivity experiment.

### C.15 Comparison with Closest Related Work

Table 5 summarizes the distinction between MAPID-LLM and related lines of work. The key difference is not the use of multiple agents alone, but the combination of inference-time prompt-level control, distributed PID feedback, output-to-input coupling, and conditional boundedness analysis.

## D Scenario-Specific Control Parameters

This appendix provides comprehensive experimental data, detailed configurations, and extended analysis supporting the main paper's findings.

### D.1 Experimental Scenarios and Parameter Specifications

#### D.1.1 Research Scenario: Literature Synthesis and Knowledge Integration

The Research scenario involves collaborative literature review, synthesis, and information integration to generate comprehensive research reports. Agents coordinate to identify relevant sources, evaluate authority, integrate information, and synthesize findings. The following nine parameters control different aspects of agent behavior:

**Parameter Definitions**

1. **Search Breadth**
   Controls the number of databases, keywords, or literature sources considered per search operation.
   - *Too narrow*: Misses relevant information, incomplete literature coverage.
   - *Too broad*: Information overload, reduced efficiency, noise in results.
   - *Optimal*: Balanced coverage enabling comprehensive yet focused search.
2. **Search Depth**
   Controls the traversal depth in citation hierarchies and reference chains (e.g., how many citation levels deep agents explore).
   - *Too shallow*: Only surface-level works captured, misses foundational papers and connections.

**Table 5:** Comparison with closest related work.

| Method family | Multi-agent | Inference-time adaptation | Prompt-level continuous controls | Distributed PID feedback | Conditional boundedness analysis |
|---|---|---|---|---|---|
| Self-consistency / aggregation | ✓ | ✗ | ✗ | ✗ | ✗ |
| Self-refinement / reflection | ✓ | ✓ | ✗ | ✗ | ✗ |
| Multi-agent debate | ✓ | ✓ | ✗ | ✗ | ✗ |
| Pre-inference prompt optimization | optional | ✗ | ✗ | ✗ | ✗ |
| Classical networked PID | ✓ | ✓ | not LLM prompts | ✓ | ✓ |
| MAPID-LLM | ✓ | ✓ | ✓ | ✓ | local surrogate-level ✓ |

- *Too deep*: Encounters outdated, tangential, or irrelevant literature; diminishing returns.
- *Optimal*: Captures seminal and recent work while avoiding citation cycles.

3. **Authority Bias**
   Controls the weighting preference for high-impact journals, citations, and prominent authors during literature filtering.
   - *Excessive bias*: Ignores emerging but valuable contributions from new researchers or venues.
   - *No bias*: Admits low-quality, low-impact sources without discrimination.
   - *Optimal*: Balances recognition of established authority with openness to innovation.

4. **Information Integration**
   Measures the agent's ability to synthesize and merge information from multiple diverse sources into coherent reports.
   - *Low integration*: Fragmented reports with disconnected source summaries; poor synthesis.
   - *Over-integration*: Excessive simplification or distortion of original meanings; loss of nuance.
   - *Optimal*: Clear unified narrative that preserves source-specific insights while revealing patterns.

5. **Critical Thinking Coefficient**
   Controls the degree of critical evaluation applied to retrieved information. Regulates skepticism and source validation rigor.
   - *Excessive*: Questioning all sources leads to paralysis; rejection of valid findings.
   - *Insufficient*: Uncritical acceptance of all information; inclusion of flawed or biased sources.
   - *Optimal*: Discriminating evaluation balancing validation with acceptance of established findings.

6. **Citation Density**
   Controls the proportion of citations in generated text (citations per 100 words or similar metric).
   - *High density*: Enhances credibility but disrupts reading flow; excessive interruption.
   - *Low density*: Lacks evidential support; readers question claim validity.
   - *Optimal*: Strategic citation at key claims maintains credibility without disrupting narrative.

7. **Diversity**
   Measures coverage of different research perspectives, methodological approaches, and theoretical schools in retrieved literature.
   - *High diversity*: Comprehensive multi-perspective view; may increase comprehension difficulty.
   - *Low diversity*: Single-perspective bias; incomplete treatment of topic.
   - *Optimal*: Multiple viewpoints presented clearly without overwhelming readers.

8. **Character Compliance Factor**
   Reinforces role-specific behavioral patterns through weighted rewards or penalties. Enhances consistency with assigned agent roles (e.g., critical reviewer vs. synthesizer).
   - *High*: Strong role adherence; risk of inflexibility.
   - *Low*: Inconsistent role performance; reduces collaborative effectiveness.
   - *Optimal*: Consistent role identity while allowing task-appropriate flexibility.

9. **Compliance Strength**
   Adjusts agent adherence to predefined collab-

orative rules and protocols (e.g., turn-taking, permission hierarchies, response ordering).

- *High*: Strict protocol adherence; may prevent efficient emergent behaviors.
- *Low*: Protocol violation; coordination breakdown.
- *Optimal*: Rules follow enables predictable coordination while allowing adaptive responses.

*Summary*: The Research scenario employs nine controllable parameters spanning source identification, authority evaluation, synthesis capability, and collaborative discipline. PID control regulates these parameters to balance comprehensive literature coverage with synthesis clarity and coordination efficiency.

#### D.1.2 Coding Scenario: Collaborative Software Development

The Coding scenario involves collaborative code generation, review, and refinement across multiple agents with specialized roles (architect, implementer, reviewer). Eight parameters govern code quality, readability, and collaboration dynamics:

**Parameter Definitions**

1. **Lines of Code (LOC)**
   Quantifies the volume of generated code per solution or module.
   - *Excessive LOC*: Indicates over-complex implementation; difficult to maintain and test.
   - *Insufficient LOC*: Incomplete implementation; missing required functionality.
   - *Optimal*: Code volume proportional to functional requirements; maintains clarity and completeness.
2. **Cyclomatic Complexity**
   Measures the number of independent execution paths through code logic. Reflects decision point density.
   - *High complexity*: Multiple nested conditionals impair readability; increase testing burden.
   - *Low complexity*: Simplified logic; risk of missing edge cases or behavioral requirements.
   - *Optimal*: Appropriate complexity reflecting problem requirements without unnecessary branching.
3. **Comment Percentage**
   Ratio of comment lines to total source lines of code.
   - *High percentage*: Over-documentation; verbose comments reduce code signal-to-noise.
   - *Low percentage*: Under-documented; unclear intent and complex sections lack explanation.
   - *Optimal*: Targeted comments explaining "why" not "what"; supports maintainability.
4. **Code Duplication Rate**
   Proportion of duplicated code segments relative to total codebase (measured by token similarity or exact matching).
   - *High duplication*: Increased maintenance cost; changes must be propagated across copies; error-prone.
   - *Low duplication*: Refactored code; easy to maintain; single point of modification.
   - *Optimal*: Minimal duplication through reusable components and abstractions.
5. **Reasoning Depth**
   Extent of step-by-step logical reasoning displayed in code generation process and commit messages.
   - *Deep reasoning*: Detailed justification of design choices; aids review and onboarding.
   - *Shallow reasoning*: Limited explanation; reviewers struggle to understand intent.
   - *Optimal*: Clear rationale for major design decisions; appropriate detail level for audience.
6. **Consistency**
   Uniformity in naming conventions, code formatting, variable scoping, and logical structure across modules.
   - *High consistency*: Predictable code structure; easier navigation and modification.
   - *Low consistency*: Inconsistency frustrate reviewers; increase cognitive load.
   - *Optimal*: Consistent conventions across codebase improving readability and maintainability.

7. **Character Compliance Factor**
   Reinforces agent role-specific behaviors (e.g., architect vs. implementer patterns). Maintains role consistency in multi-agent code generation.
   - *High*: Strong role identity; specialized behavior patterns.
   - *Low*: Flexible role blending; potential specialization loss.
   - *Optimal*: Clear role definition enabling complementary contributions.

8. **Compliance Strength**
   Adherence to coding standards, collaborative protocols, and team agreements (e.g., code review requirements, branch strategies).
   - *High*: Strict process adherence; may slow development.
   - *Low*: Process violation; integration issues and conflicts.
   - *Optimal*: Standards compliance enabling smooth collaboration without excessive overhead.

*Summary*: The Coding scenario employs eight controllable parameters spanning structural quality (LOC, complexity), documentation standards (comments), code reuse (duplication), reasoning transparency, consistency, and role-based collaboration discipline. PID control orchestrates these parameters to achieve maintainable, collaborative code generation.

#### D.1.3 Database Scenario: Collaborative Diagnostic Analysis

The Database scenario models multi-agent collaborative database diagnostics, where agents analyze query performance, system logs, and schema issues. Eight parameters control the diagnostic process depth and inter-agent collaboration:

**Parameter Definitions**

1. **SQL Query Complexity**
   Structural complexity of diagnostic queries generated (measured by join count, subquery depth, predicate complexity).
   - *Excessive complexity*: Complex queries timeout or perform poorly on large datasets; query optimization required.
   - *Low complexity*: Insufficient detail; misses root-cause indicators or correlations.
   - *Optimal*: Balanced complexity capturing diagnostic signals without performance degradation.

2. **Evidence Collection Completeness**
   Comprehensiveness of gathered diagnostic evidence (logs, metrics, query execution plans, schema snapshots).
   - *Incomplete*: Missing evidence reduces diagnostic confidence; incomplete root-cause analysis.
   - *Excessive*: Overwhelming evidence volume; difficult synthesis; information overload.
   - *Optimal*: Complete evidence chain supporting confident diagnosis without redundancy.

3. **Diagnostic Depth**
   Extent of analysis penetration into system layers (surface symptoms, intermediate causes, root causes).
   - *Shallow*: Symptom identification only; no root-cause remediation.
   - *Excessive*: Over-analysis consuming resources; diminishing returns beyond root cause.
   - *Optimal*: Analysis reaching root cause with actionable remediation without over-investigation.

4. **Collaboration Efficiency**
   Effectiveness of inter-agent information sharing, parallel investigation, and result synthesis.
   - *Low efficiency*: Redundant parallel analysis; missed opportunities for shared findings.
   - *High efficiency*: Coordinated investigation avoiding duplication; accelerated diagnosis.
   - *Optimal*: Parallelized analysis with strategic information pooling maximizing speed and accuracy.

5. **Report Structure Quality**
   Logical organization and clarity of diagnostic output (executive summary, evidence sections, root cause, remediation).

- *Poor structure*: Disorganized findings frustrate stakeholders; difficult to act on diagnosis.
- *Over-structure*: Rigid templates reduce clarity if findings don't fit standard patterns.
- *Optimal*: Clear hierarchical organization enabling rapid understanding and action.

6. **Reproducibility**
   Consistency of diagnostic results when applied to the same problem conditions (determinism in query results and analysis).

   - *Low reproducibility*: Inconsistent findings undermine trust; unreliable recommendations.
   - *High reproducibility*: Stable, repeatable analysis; trustworthy conclusions.
   - *Optimal*: Consistent findings enabling confidence in recommendations.

7. **Character Compliance Factor**
   Reinforces role-specific diagnostic approaches (e.g., performance specialist vs. data integrity specialist roles).

   - *High*: Specialized diagnostic perspectives; potential blind spots outside role.
   - *Low*: Generic diagnostics; less specialized insight.
   - *Optimal*: Role-specific expertise with cross-role collaboration.

8. **Compliance Strength**
   Adherence to diagnostic protocols and safety rules (e.g., avoiding destructive operations on production systems).

   - *High*: Strict safety; may prevent aggressive diagnostics.
   - *Low*: Risk-taking; potential system impact.
   - *Optimal*: Safe operations enabling thorough diagnostics without system risk.

*Summary*: The Database scenario employs eight controllable parameters spanning query optimization and complexity, evidence comprehensiveness, root-cause analysis depth, collaboration efficiency, output clarity, result reliability, and diagnostic discipline. PID control balances investigation thoroughness with operational safety and convergence speed.

#### D.1.4 MATH Scenario: Collaborative Mathematical Problem Solving

The MATH scenario involves multi-agent collaborative problem solving, where agents discuss solution approaches, verify correctness, and reach consensus on final answers. Eight parameters control reasoning depth and collaborative convergence:

**Parameter Definitions**

1. **Reasoning Steps**
   Number of explicit step-by-step reasoning steps shown in solution derivation (reflects chain-of-thought depth).

   - *Too many steps*: Over-detailed; obscures high-level approach; reader fatigue.
   - *Too few steps*: Insufficient justification; logical gaps; difficult to verify.
   - *Optimal*: Sufficient detail for verification while maintaining readability.

2. **Domain Knowledge Activation Range**
   Breadth of mathematical domains and techniques invoked (e.g., algebra, calculus, probability, combinatorics).

   - *Narrow*: Overly specialized; misses elegant cross-domain solutions.
   - *Broad*: May introduce unnecessary complexity; difficult to follow.
   - *Optimal*: Appropriate domain breadth for problem class; elegant, focused solutions.

3. **Mathematical Symbol Accuracy**
   Precision in mathematical notation usage, operator definitions, and symbolic manipulation (absence of notation errors).

   - *Low accuracy*: Notation errors propagate through solution; reduces rigor and readability.
   - *High accuracy*: Proper notation supporting formal verification; clear symbolic reasoning.
   - *Optimal*: Accurate, standard notation reflecting mathematical conventions.

4. **Abstraction Level**
   Degree of generalization from specific problem instances to abstract mathematical structures and theorems.

   - *Concrete-only*: Solutions tied to specific numbers; limited transferability.

- *Over-abstract*: General theorems introduced without grounding; difficult to follow.
- *Optimal*: Problem-specific solution with connections to general principles.

5. **Consensus Formation Threshold**
Minimum agreement level required for agents to accept a solution as valid before proceeding (consensus majority, supermajority, or unanimity).
   - *High threshold*: Ensures quality but may require excessive iteration.
   - *Low threshold*: Rapid consensus but risk of accepting incorrect solutions.
   - *Optimal*: Threshold balancing correctness assurance with convergence speed.

6. **Answer Completeness**
Extent of final answer specification (numerical value, derivation, proof, alternative approaches).
   - *Incomplete*: Only numerical answer without justification.
   - *Over-complete*: Excessive alternative derivations; redundancy.
   - *Optimal*: Numerical answer with concise supporting derivation.

7. **Character Compliance Factor**
Reinforces agent role patterns (e.g., verifier vs. solver roles; skeptic vs. proposer dynamics).
   - *High*: Clear role distinction enabling specialized contributions.
   - *Low*: Flexible roles; potential collaboration but less specialization.
   - *Optimal*: Role clarity supporting complementary problem-solving approaches.

8. **Compliance Strength**
Adherence to mathematical rigor standards and collaborative agreement on solution approaches and verification procedures.
   - *High*: Rigorous verification; potentially slower consensus.
   - *Low*: Rapid agreement but lower solution quality assurance.
   - *Optimal*: Standards ensuring correctness while enabling reasonable convergence speed.

*Summary*: The MATH scenario employs eight controllable parameters spanning reasoning depth, domain breadth, notation rigor, abstraction appropriateness, consensus requirements, answer specification level, role specialization, and mathematical discipline. PID control orchestrates these parameters to achieve correct, verifiable, and efficient collaborative solutions.

**Transition to Appendix B.2:** While the above sections define the complete set of controllable parameters available in each scenario (Research: 9 parameters; Coding: 8 parameters; Database: 8 parameters; MATH: 8 parameters), practical PID control implementation typically selects a representative subset for real-time feedback adjustment. Appendix D.2 details the affine normalization procedure that maps these scenario-specific parameters to a unified four-dimensional control space, and provides a concrete calculation example demonstrating how four representative parameters from the Research scenario are normalized and processed by the PID controller.

### D.2 Task-Specific Parameter Definitions Example

Research semantic dimensions (9 total): Search Breadth $[1, 10]$, Search Depth $[1, 10]$, Authority Bias $[0, 1]$, and Information Integration $[0, 1]$ are the primary runtime control variables. Critical Thinking Coefficient $[0, 1]$, Citation Density $[0, 100]$, Diversity $[0, 1]$, Character Compliance Factor $[1, 10]$, and Compliance Strength $[1, 10]$ are supplementary dimensions used for diagnostics, reference shaping, or soft constraint enforcement.

For the four primary Research controls, the continuous control values are mapped to discrete directive levels using the affine rule in Appendix B.1. Specifically, Search Breadth and Search Depth are mapped to integer levels in $[1, 10]$, while Authority Bias and Information Integration are mapped to bounded intensity levels in $[0, 1]$ before being verbalized as textual directives. Supplementary dimensions inform reference trajectory generation and soft constraint enforcement.

### D.3 PID Controller Implementation Details

#### D.3.1 PID Controller Implementation

Each agent $i$ maintains a discrete-time PID controller whose executable update is consistent with Eq. (7), Eq. (10), and Eq. (8) in the main text.

The locally available semantic-reference estimate is first projected into the prompt-control space:

$$\mathbf{r}_{i,c}(t) = \psi_i(\hat{\mathbf{r}}_i(t)). \tag{54}$$

The control-space fusion error is then computed as

$$\begin{aligned}\mathbf{e}_i(t) =& \alpha\big(\mathbf{r}_{i,c}(t) - \mathbf{c}_i(t)\big) \\ &+ (1-\alpha)\sum_{j\in\mathcal{N}_i} w_{ij}\big(\tilde{\mathbf{c}}_j(t) - \mathbf{c}_i(t)\big).\end{aligned} \tag{55}$$

Here $\tilde{\mathbf{c}}_j(t)$ denotes the latest available or delay-compensated neighbor control summary; in synchronous communication, $\tilde{\mathbf{c}}_j(t) = \mathbf{c}_j(t)$.

The integral memory is clipped to prevent windup:

$$\mathbf{I}_i(t) = \mathrm{clip}_{[-\mathbf{I}_{\max},\mathbf{I}_{\max}]}\big(\mathbf{I}_i(t-1) + \mathbf{e}_i(t)\big). \tag{56}$$

Let

$$\Delta\mathbf{e}_i(t) = \mathbf{e}_i(t) - \mathbf{e}_i(t-1).$$

The PID increment is then computed as

$$\mathbf{u}_i(t) = \mathbf{K}_{P,i}\mathbf{e}_i(t) + \mathbf{K}_{I,i}\mathbf{I}_i(t) + \mathbf{K}_{D,i}\Delta\mathbf{e}_i(t). \tag{57}$$

The increment is saturated component-wise and the prompt-level control is projected back to the feasible prompt-control box:

$$\begin{aligned}\mathbf{u}_i^{\mathrm{sat}}(t) &= \mathrm{sat}_{[\mathbf{u}_{\min},\mathbf{u}_{\max}]}\big(\mathbf{u}_i(t)\big), \\ \mathbf{c}_i(t+1) &= \Pi_{\mathcal{C}}\big(\mathbf{c}_i(t) + \mathbf{u}_i^{\mathrm{sat}}(t)\big).\end{aligned} \tag{58}$$

Thus, $\mathbf{u}_i(t)$ is the PID increment used in the main text; it is not a separate accumulated control state.

For the four primary Research controls used in the implementation example, the control dimensions correspond to

- $c_i^{(1)}$ Search Breadth $\in [1, 10]$;
- $c_i^{(2)}$ Search Depth $\in [1, 10]$;
- $c_i^{(3)}$ Authority Bias $\in [0, 1]$;
- $c_i^{(4)}$ Information Integration $\in [0, 1]$.

These ranges are the same as those defined in Appendix D.2 and are mapped to textual directives through the affine normalization rule in Appendix B.1.

A representative diagonal gain setting is

$$\mathbf{K}_P = \mathrm{diag}([0.8, 0.5, 0.3, 0.2]), \tag{59}$$

$$\mathbf{K}_I = \mathrm{diag}([0.3, 0.2, 0.1, 0.1]), \tag{60}$$

$$\mathbf{K}_D = \mathrm{diag}([0.1, 0.08, 0.05, 0.05]). \tag{61}$$

Hyperparameters: fusion coefficient $\alpha = 0.6$, alignment gain $\gamma = 0.2$, and one control step per inference round.

# E Extended Experimental Details

This appendix provides detailed experimental information omitted from the main text, including benchmark configurations, full per-backbone results, ablation details, hyperparameter sensitivity analyses, bidirectional control validation, visualization results, and implementation details.

## E.1 Benchmark Configurations

**MultiAgentBench (MAB) Tasks.**

**Table 6:** MultiAgentBench task details and configurations.

| Scenario | Num Agents | Metrics |
|---|---|---|
| Coding | 3–5 | Task Score, Coordination Score |
| Research | 3–8 | Task Score, Coordination Score |
| Database | 3–8 | Task Score, Coordination Score |

**Metric scale.** Following MultiAgentBench, we report Task Score (TS) and Coordination Score (CS) on a 0–100 scale. For LLM-judged rubric components, the evaluator may first assign a 1–5 score, which is then multiplied by 20 for reporting. For task-completion or test-based scenarios, TS is computed from the corresponding task-specific success metric and normalized to the same 0–100 scale. CS is derived from communication and planning quality sub-scores and is reported on the same 0–100 scale. Therefore, values such as 66.7 or 78.4 in our tables should be interpreted as normalized 0–100 scores rather than raw 1–5 rubric scores.

**MATH Configuration.** Problems are selected from the MATH dataset. The collaborative agent roles include Problem Analyzer, Solution Designer, Computation Executor, Verification Agent, and Result Synthesizer. The evaluation metric is final-answer accuracy.

**BigCodeBench Configuration.** BigCodeBench contains 1,140 coding tasks. The multi-agent team size is 3–5 agents, and the evaluation metric is Pass@1 on the test suite.

## E.2 Full Per-Backbone Benchmark Results

## E.3 Backbone-Level Paired Reliability Analysis

This subsection reports a backbone-level paired reliability analysis for the main benchmark results. Unlike seed-level significance testing, this analysis treats each LLM backbone as a matched evaluation unit. For each benchmark setting, MAPID-LLM is compared with the corresponding non-MAPID baseline under the same backbone.

**Table 7:** Full per-backbone MultiAgentBench results. TS denotes Task Score and CS denotes Coordination Score. Each score is reported as mean ± one standard deviation over three matched random-seed runs using seeds 42, 123, and 456.

| Backbone | Scenario | TS Baseline | TS MAPID | Δ TS | CS Baseline | CS MAPID | Δ CS |
|---|---|---|---|---|---|---|---|
| Qwen3-8B | Coding | 69.7%±1.1 | 76.6%±1.0 | **+6.9%** | 65.6%±0.9 | 79.8%±1.1 | **+14.2%** |
| | Research | 65.5%±0.9 | 83.3%±1.0 | **+17.8%** | 60.6%±1.1 | 76.7%±0.9 | **+16.1%** |
| | Database | 60.0%±1.2 | 68.2%±1.3 | **+8.2%** | 61.6%±1.0 | 75.0%±1.2 | **+13.4%** |
| Qwen3-235B-A22B | Coding | 62.8%±1.0 | 68.8%±1.1 | **+6.0%** | 66.2%±1.2 | 70.2%±1.0 | **+4.0%** |
| | Research | 70.7%±1.2 | 75.2%±0.9 | **+4.5%** | 49.7%±0.9 | 65.4%±1.1 | **+15.7%** |
| | Database | 53.4%±1.0 | 63.9%±1.1 | **+10.5%** | 63.3%±1.0 | 67.6%±1.0 | **+4.3%** |
| DeepSeek-V3 | Coding | 67.5%±0.9 | 75.9%±1.0 | **+8.4%** | 67.3%±1.2 | 69.5%±1.1 | **+2.2%** |
| | Research | 71.2%±1.1 | 76.7%±1.2 | **+5.5%** | 68.0%±1.0 | 73.7%±0.9 | **+5.7%** |
| | Database | 56.7%±1.0 | 65.9%±0.9 | **+9.2%** | 69.3%±1.1 | 76.1%±1.1 | **+6.8%** |

**Table 8:** Full per-backbone results on MATH and BigCodeBench. Each score is reported as mean ± one standard deviation over three matched random-seed runs using seeds 42, 123, and 456.

| Benchmark | Backbone | Baseline | MAPID-LLM | Metric | Δ |
|---|---|---|---|---|---|
| MATH | Qwen3-8B | 66.7%±1.1 | 72.8%±1.2 | Accuracy | **+6.1%** |
| | Qwen3-235B-A22B | 69.4%±0.9 | 76.3%±1.1 | Accuracy | **+6.9%** |
| | DeepSeek-V3 | 78.1%±1.0 | 86.0%±1.1 | Accuracy | **+7.9%** |
| BigCodeBench | Qwen3-8B | 68.0%±1.1 | 76.1%±1.2 | Pass@1 | **+8.1%** |
| | Qwen3-235B-A22B | 73.5%±1.0 | 81.9%±1.1 | Pass@1 | **+8.4%** |
| | DeepSeek-V3 | 71.7%±0.9 | 83.1%±1.1 | Pass@1 | **+11.4%** |

$$\Delta_b = x_b^{\mathrm{MAPID}} - x_b^{\mathrm{base}}. \tag{62}$$

We report the mean paired gap, the range of paired gaps, and the number of backbones for which MAPID-LLM outperforms the matched baseline. Because this analysis contains only three matched backbones, we do not report formal $p$-values. The purpose is to assess directional consistency across model families rather than to claim seed-level statistical significance.

Here, Backbone Gaps and Mean Gap denote absolute score-point improvements within matched backbone comparisons, whereas Table 1 reports relative percentage gains over the corresponding baselines.

Table 9 shows that MAPID-LLM achieves positive paired gaps across all matched backbone comparisons. This indicates that the observed improvements are not driven by a single model family. Instead, MAPID-LLM improves the corresponding non-MAPID baseline consistently across Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3.

The largest variability appears in MAB-Research, where Qwen3-8B obtains a much larger task-score gain than the larger backbones. This suggests that inference-time prompt-level PID regulation may be especially beneficial when the backbone has more room for runtime behavioral correction. For BigCodeBench, the per-backbone gaps also indicate consistent positive gains, with a mean improvement of +9.3 score points.

### E.4 Full Ablation and Control Analysis

**Table 10:** PID component ablation on Qwen3-8B.

| Variant | Score | Finding |
|---|---|---|
| Full PID ($P+I+D$) | 76.6%±1.2 | Best overall performance |
| Only P | 72.3%±1.5 | Strong immediate correction |
| Only I | 70.1%±1.8 | Insufficient alone |
| P+I (No D) | 74.7%±1.3 | Recovers most full-PID performance |
| P+D (No I) | 73.4%±1.4 | Lower than full PID |
| I+D (No P) | 70.2%±1.6 | Largest two-term degradation |
| Only D | 72.9%±1.5 | Helps stabilization but incomplete alone |

The ablation study shows that the full PID controller achieves the best performance. Among single-term variants, the derivative-only controller slightly outperforms the proportional-only controller, suggesting that derivative feedback provides useful standalone stabilization. However, the two-term ablations show that removing the proportional

**Table 9:** Backbone-level paired reliability analysis. Each paired gap compares MAPID-LLM with the corresponding non-MAPID baseline under the same backbone. The three gaps correspond to Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3, respectively. $W$ denotes the number of backbones where MAPID-LLM outperforms the matched baseline.

| Benchmark / Scenario | Metric | Matched Baseline | Backbone Gaps | Mean Gap | Range | $W$ |
|---|---|---|---|---|---|---|
| **MultiAgentBench task performance** | | | | | | |
| MAB-Coding | Task Score | MARBLE w/o PID | +6.9% / +6.0% / +8.4% | +7.1% | [6.0%, 8.4%] | 3/3 |
| MAB-Research | Task Score | MARBLE w/o PID | +17.8% / +4.5% / +5.5% | +9.3% | [4.5%, 17.8%] | 3/3 |
| MAB-Database | Task Score | MARBLE w/o PID | +8.2% / +10.5% / +9.2% | +9.3% | [8.2%, 10.5%] | 3/3 |
| **MultiAgentBench coordination quality** | | | | | | |
| MAB-Coding | Coord. Score | MARBLE w/o PID | +14.2% / +4.0% / +2.2% | +6.8% | [2.2%, 14.2%] | 3/3 |
| MAB-Research | Coord. Score | MARBLE w/o PID | +16.1% / +15.7% / +5.7% | +12.5% | [5.7%, 16.1%] | 3/3 |
| MAB-Database | Coord. Score | MARBLE w/o PID | +13.4% / +4.3% / +6.8% | +8.2% | [4.3%, 13.4%] | 3/3 |
| **Generalization to math and code benchmarks** | | | | | | |
| MATH | Accuracy | w/o MAPID | +6.1% / +6.9% / +7.9% | +7.0% | [6.1%, 7.9%] | 3/3 |
| BigCodeBench | Pass@1 | w/o MAPID | +8.1% / +8.4% / +11.4% | +9.3% | [8.1%, 11.4%] | 3/3 |

term causes the largest degradation: I+D performs worse than P+I and P+D. This indicates that the proportional term is essential for stable corrective feedback in the combined PID controller, while the integral and derivative terms complement it by reducing accumulated bias and stabilizing feedback-driven updates.

## E.5 Hyperparameter Sensitivity Analysis

We conduct parameter sweeps within the robust ranges used in the main experiments to validate the sensitivity characteristics of the PID gains. In the scalar sensitivity sweeps, $K_p$, $K_i$, and $K_d$ denote global multipliers applied to the corresponding diagonal gain templates, rather than replacing each dimension-specific gain independently. These diagnostic sweeps are conducted under the Qwen3-8B Coding setting and are used to compare relative sensitivity trends.

### E.5.1 Proportional Gain $K_p$ Sensitivity

**Table 11:** $K_p$ sensitivity on the Coding scenario with Qwen3-8B.

| $K_p$ Value | Task Score | Characterization |
|---|---|---|
| 0.3 | 71.2%±1.5 | Slow response |
| 0.5 | 76.6%±1.0 | Default / balanced |
| 0.8 | 75.8%±1.3 | Upper robust bound |

$K_p = 0.5$ achieves the strongest performance among the tested values. Lower values respond more slowly to evaluator feedback, while larger values approach the upper robust range and may increase oscillatory behavior.

### E.5.2 Integral and Derivative Gains

**Table 12:** $K_i$ and $K_d$ sensitivity on the Coding scenario with Qwen3-8B.

| Parameter | Value | Task Score | Characterization |
|---|---|---|---|
| $K_i$ | 0.05 | 75.2%±1.4 | Lower bound |
| | 0.1 | 76.6%±1.0 | Default / optimal |
| | 0.2 | 75.8%±1.3 | Upper bound |
| $K_d$ | 0.02 | 75.5%±1.2 | Lower bound |
| | 0.05 | 76.6%±1.0 | Default / optimal |
| | 0.1 | 75.9%±1.2 | Upper bound |

Both $K_i$ and $K_d$ exhibit moderate robustness across the tested ranges. The selected values $(K_i = 0.1, K_d = 0.05)$ achieve the best observed performance while maintaining low variance.

## E.6 Scenario-Specific Control Details

The scenario-specific control analysis examines how prompt-level control variables affect different benchmark domains. The parameter-control mechanism yields consistent improvements across heterogeneous domains, with larger gains in settings where the controllable prompt dimensions provide a wider operating range. For example, database tasks allow control over query complexity and verification strictness, while mathematical tasks impose tighter reasoning constraints and therefore provide less room for prompt-level adjustment.

## E.7 Bidirectional Control Validation

This table provides the full bidirectional control validation summarized in Section 4.3. The neutral reference denotes the uncontrolled or non-directionally adjusted setting on the Qwen3-8B

**Table 13:** Bidirectional control validation on the Qwen3-8B Coding subset.

| Control Direction | Task Score | Coord. Score |
|---|---|---|
| GOOD direction | 76.6%±1.2 | 79.8%±0.06 |
| Neutral reference | 69.7%±0.15 | 65.6%±0.12 |
| BAD direction | 58.5%±0.12 | 61.5%±0.10 |

Coding subset.The standard deviations in this table are computed over repeated directional-control runs on this subset, rather than over the full benchmark aggregation used in the main result table.

Task Score spans 18.1 points between the GOOD and BAD directions, while Coordination Score spans 18.3 points. This directional range provides evidence that prompt-level PID regulation can steer both task and coordination behavior, rather than merely producing random fluctuations.

### E.8 Fusion Coefficient Analysis

This table provides the full fusion-coefficient sensitivity analysis summarized in Section 4.3.

**Table 14:** Fusion coefficient $\alpha$ sensitivity on the Coding scenario with Qwen3-8B.

| $\alpha$ | Self Weight | Task Score | Coord. Score | Combined |
|---|---|---|---|---|
| 0.2 | 20% | 68.5%±0.12 | 77.2%±0.08 | 145.7 |
| 0.4 | 40% | 73.5%±0.10 | 73.5%±0.09 | 147.0 |
| 0.6 | 60% | 76.6%±0.08 | 79.8%±0.07 | 156.4 |
| 0.8 | 80% | 74.1%±0.09 | 68.7%±0.10 | 142.8 |

$\alpha = 0.6$ achieves the strongest combined performance in this sweep. Lower values prioritize neighbor coordination at the expense of task execution, while higher values emphasize individual tracking but reduce coordination quality.

### E.9 Model-Wise Benefit Analysis

**Table 15:** Model-wise benefit across representative benchmarks. Average denotes the unweighted mean of the four listed benchmark gains. All gains are absolute percentage-point differences.

| Model | MAB Coding | MAB Research | MATH | BigCode | Average |
|---|---|---|---|---|---|
| Qwen3-8B | +6.9% | +17.8% | +6.1% | +8.1% | +9.7% |
| Qwen3-235B-A22B | +6.0% | +4.5% | +6.9% | +8.4% | +6.5% |
| DeepSeek-V3 | +8.4% | +5.5% | +7.9% | +11.4% | +8.3% |

These results show that the benefit of MAPID-LLM varies across backbones and benchmarks. Qwen3-8B shows the largest gain on MAB-Research, while DeepSeek-V3 shows the largest gain on BigCodeBench. The average column should be interpreted as a descriptive unweighted mean over the four representative benchmark gains, not as a formal capacity-scaling conclusion. Overall, the results indicate that prompt-level PID regulation provides complementary benefits across model families, while the relationship between model capacity and control benefit requires broader validation.

### E.10 Visualization of Key Results

This section provides visualizations complementing the quantitative tables in the main text and appendix.

#### E.10.1 MultiAgentBench Performance Comparison

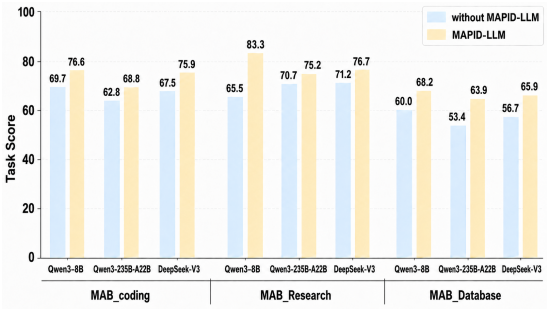


**Figure 5:** MultiAgentBench Task Score comparison across Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3.

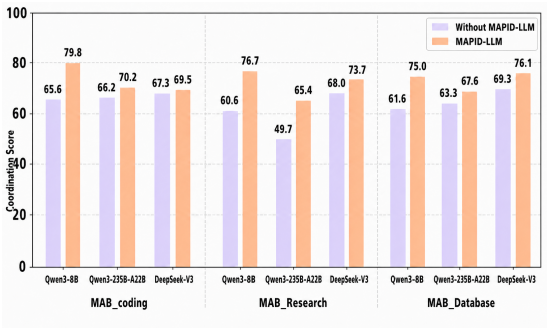


**Figure 6:** MultiAgentBench Coordination Score comparison across Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3.

### E.10.2 Bidirectional Control Trajectories

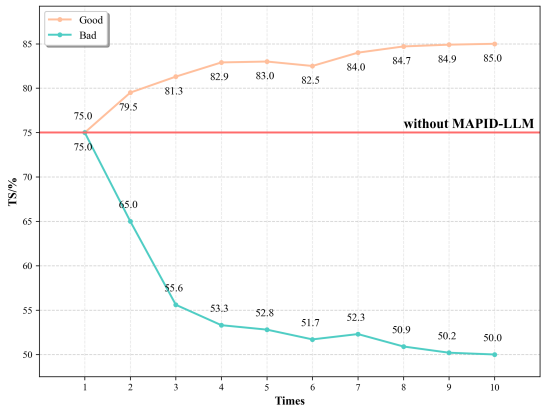


**Figure 7:** Task Score trajectories under GOOD direction, neutral reference, and BAD direction.

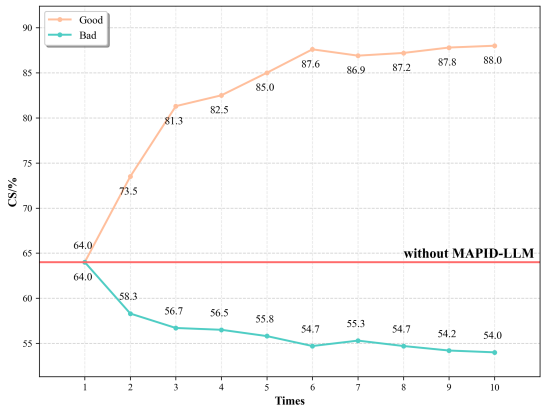


**Figure 8:** Coordination Score trajectories under GOOD direction, neutral reference, and BAD direction.

### E.10.3 Parameter Control Effect

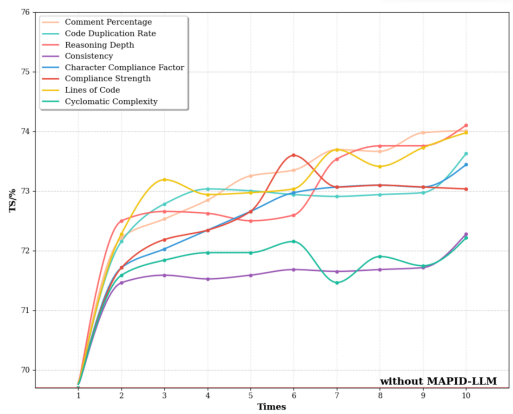


**Figure 9:** Task Score changes under scenario-specific parameter control.

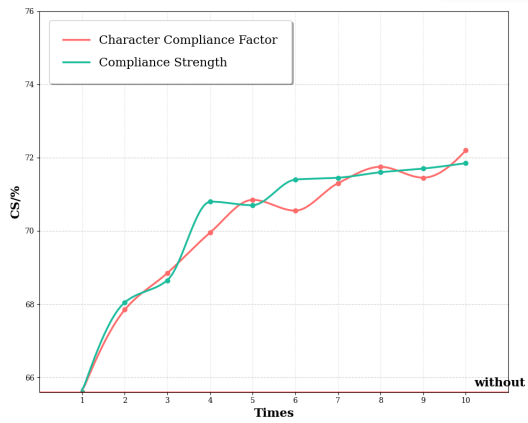


**Figure 10:** Coordination Score changes under scenario-specific parameter control.

## E.11 Implementation Details

### E.11.1 Software and Compute Environment

- Python 3.10.12 and PyTorch 2.0.1.
- Custom multi-agent LLM wrapper framework.
- LLM inference was performed through Qwen and DeepSeek official API endpoints using Python-based experiment scripts.
- Local controller execution, evaluator-side bookkeeping, logging, numerical control experiments, sensitivity analysis, and post-processing were conducted on a multi-GPU server equipped with NVIDIA H200 GPUs, each with 141GB memory.
- The reported local compute budget refers to controller/evaluator-side computation, numerical experiments, data processing, and analysis workloads, while API-side LLM inference cost is accounted for separately through token usage and evaluator-call statistics.

### E.11.2 Experimental Protocol

- Random seeds: 42, 123, and 456.
- Independent runs: three per method–benchmark–backbone configuration, one for each random seed.
- Seed control: baseline and MAPID variants use matched seeds, task orders, backbones, agent roles, and communication topologies.
- Results reporting: per-backbone results are reported as mean $\pm$ one standard deviation over the three independent runs. Cross-backbone

averaged rows are computed by first averaging within each backbone and then averaging the per-backbone means over Qwen3-8B, Qwen3-235B-A22B, and DeepSeek-V3.

- Statistical analysis: We report mean $\pm$ standard deviation for repeated runs. For cross-backbone comparisons, we report backbone-level paired reliability in Appendix E.3; we do not report formal $p$-values for these comparisons because they contain only three matched backbones.

#### E.11.3 Computational Resources

The compute environment is summarized above. LLM inference was performed through API endpoints, while local controller/evaluator-side computation and numerical analyses were conducted on the NVIDIA H200 server.

### E.12 Limitations

- **Reference Definition:** MAPID-LLM requires explicit or evaluator-derived reference signals, which may limit applicability to fully open-ended tasks with ambiguous objectives.

- **Scale Validation:** The empirical evaluation covers up to 8 agents. Larger teams may require hierarchical or modular control architectures.

- **Model Coverage:** The evaluation covers three LLM backbones. Generalization to additional model families, multimodal models, or domain-specific systems requires further study.

- **Communication Requirements:** The local-control architecture assumes structured inter-agent communication, and its behavior under severe communication constraints remains future work.

### E.13 Future Research Directions

- Extending conditional boundedness analysis to broader network topologies and communication delays.

- Developing adaptive gain-selection methods to reduce manual tuning.

- Designing hierarchical PID control for large-scale agent teams.

- Applying prompt-level control to multimodal and embodied multi-agent systems.

- Studying robustness under adversarial, unreliable, or Byzantine agents.

## F Agent Prompts and PID-Controlled Prompt Directives

This appendix reports the benchmark-specific agent prompts and PID-controlled prompt directives used by MAPID-LLM. The purpose is to make the prompt-level actuation mechanism explicit. All compared methods use the same task inputs, base role prompts, LLM backbones, and communication topology. MAPID-LLM differs only by injecting dynamic PID-controlled directives during inference. The controller modifies bounded textual instructions derived from prompt-level control variables, rather than modifying model parameters, hidden states, or the underlying multi-agent protocol.

For agent $i$ at inference step $t$, the MAPID-LLM prompt is constructed as:

$$P_i^{\mathrm{MAPID}}(t) = P_i^{\mathrm{role}} \oplus D_i^{\mathrm{PID}}(t) \oplus X^{\mathrm{task}} \oplus C_i^{\mathrm{nbr}}(t). \tag{63}$$

Here, $P_i^{\mathrm{role}}$ is the fixed role prompt for agent $i$, $D_i^{\mathrm{PID}}(t)$ is the dynamic PID-controlled directive generated from the current prompt-level control vector $\mathbf{c}_i(t)$, $X^{\mathrm{task}}$ is the benchmark task input, and $C_i^{\mathrm{nbr}}(t)$ is the neighbor context containing upstream agent outputs or summaries according to the communication topology. The continuous control vector is mapped to discrete directive levels through the affine normalization rule in Appendix B.1. The evaluator supplies measurement and reference signals, but it does not rewrite agent reasoning, rank candidate trajectories, select the final output, or centrally plan agent actions.

### F.1 Runtime Control Dimensions

The complete scenario-specific parameter sets are defined in Appendix D. Research contains nine controllable parameters: Search Breadth, Search Depth, Authority Bias, Information Integration, Critical Thinking Coefficient, Citation Density, Diversity, Character Compliance Factor, and Compliance Strength. Coding contains eight controllable parameters: Lines of Code (LOC), Cyclomatic Complexity, Comment Percentage, Code Duplication Rate, Reasoning Depth, Consistency, Character Compliance Factor, and Compliance Strength.

Database contains eight controllable parameters: SQL Query Complexity, Evidence Collection Completeness, Diagnostic Depth, Collaboration Efficiency, Report Structure Quality, Reproducibility, Character Compliance Factor, and Compliance Strength. MATH contains eight controllable parameters: Reasoning Steps, Domain Knowledge Activation Range, Mathematical Symbol Accuracy, Abstraction Level, Consensus Formation Threshold, Answer Completeness, Character Compliance Factor, and Compliance Strength. BigCodeBench uses the Coding parameter family because it is evaluated as a code-generation benchmark.

In the main runtime PID loop, MAPID-LLM uses a representative four-dimensional subset for each benchmark. For MAB-Research, the runtime subset is Search Breadth, Search Depth, Authority Bias, and Information Integration. For MAB-Coding and BigCodeBench, the runtime subset is Lines of Code (LOC), Cyclomatic Complexity, Reasoning Depth, and Consistency. For MAB-Database, the runtime subset is SQL Query Complexity, Evidence Collection Completeness, Diagnostic Depth, and Collaboration Efficiency. For MATH, the runtime subset is Reasoning Steps, Domain Knowledge Activation Range, Mathematical Symbol Accuracy, and Consensus Formation Threshold. The remaining dimensions are retained as supplementary signals for evaluator-side diagnostics, reference shaping, or soft constraint enforcement. In particular, Character Compliance Factor and Compliance Strength are used to monitor whether an agent remains faithful to its assigned role and collaborative protocol.

### F.2 MAB-Research: Retriever → Analyzer → Synthesizer

MAB-Research uses the Research control parameter family. The runtime PID dimensions are Search Breadth, Search Depth, Authority Bias, and Information Integration.

**MAB-Research Retriever**

**System prompt.**
<|im_start|> system: 3 LLM agents (Retriever → Analyzer → Synthesizer) collaborate to complete a research-oriented task. MAPID-LLM adds an inference-time PID-controlled directive to regulate search, evidence quality, and information integration. Balanced pipeline: precise, evidence-aware, and cooperative. Prefer relevant sources, clear evidence organization, and concise but sufficient detail.
You are the **Retriever**. Identify relevant sources, concepts, background evidence, and task-specific information. Focus on search coverage, source relevance, and evidence organization. Do not write the final research response.
General rules: stay within your role, separate evidence from speculation, do not fabricate sources, avoid unsupported claims, and make your output useful to the Analyzer and Synthesizer. <|im_end|>

**PID-Controlled Directive.**
Search Breadth = {search_breadth}. Control the number of databases, keywords, perspectives, or literature sources considered.
Search Depth = {search_depth}. Control how deeply citation hierarchies, reference chains, or related evidence paths are explored.
Authority Bias = {authority_bias}. Control the preference for high-impact, highly cited, or otherwise authoritative sources.
Information Integration = {information_integration}. Organize retrieved information so that downstream agents can synthesize it coherently.

**User prompt template.**
Research task: {question}
Previous agent messages: {history}
Please retrieve and organize relevant evidence for the research task. Focus on source coverage, relevance, authority, and useful structure for downstream analysis. Do not write the final research response.

**MAB-Research Analyzer**

**System prompt.**
<|im_start|> system: 3 LLM agents (Retriever → Analyzer → Synthesizer) collaborate to complete a research-oriented task. MAPID-LLM adds an inference-time PID-controlled directive to regulate search, evidence quality, and information integration. Balanced pipeline: precise, evidence-aware, and cooperative. Prefer reliable evidence, explicit comparison, and concise but sufficient analysis.
You are the **Analyzer**. Evaluate the retrieved evidence, compare perspectives, identify reliable claims, detect contradictions, and prepare a structured synthesis plan. Do not write the final response unless explicitly instructed.
General rules: stay within your role, distinguish strong evidence from weak evidence, avoid unsupported claims, and make your output useful to the Synthesizer. <|im_end|>

**PID-Controlled Directive.**
Search Breadth = {search_breadth}. Check whether the retrieved evidence covers the necessary perspectives.
Search Depth = {search_depth}. Use deeper citation-chain evidence when direct evidence is insufficient.
Authority Bias = {authority_bias}. Weight source authority according to the current control level.
Information Integration = {information_integration}. Merge evidence into a coherent analytical structure while preserving important distinctions.

**User prompt template.**
Research task: {question}
Previous agent messages: {history}
Please analyze the retrieved information, evaluate reliability and diversity, identify contradictions or gaps, and produce a synthesis plan for the final response.

**MAB-Research Synthesizer**

**System prompt.**
<|im_start|> system: 3 LLM agents (Retriever → Analyzer → Synthesizer) collaborate to complete a research-oriented task. MAPID-LLM adds an inference-time PID-controlled directive to regulate search, evidence quality, and information integration. Balanced pipeline: precise, evidence-aware, and cooperative. Prefer grounded synthesis, clear structure, and faithful use of evidence.
You are the **Synthesizer**. Produce the final research response using the retrieved evidence and analysis. The response should be coherent, well-grounded, and faithful to the task.
General rules: stay within your role, use previous agents' evidence, avoid unsupported claims, resolve conflicts explicitly when possible, and provide a complete final response. <|im_end|>

**PID-Controlled Directive.**
Search Breadth = {search_breadth}. Ensure that the final response reflects the breadth of relevant evidence.
Search Depth = {search_depth}. Incorporate deeper evidence only when it improves the answer.
Authority Bias = {authority_bias}. Prefer authoritative evidence while avoiding excessive exclusion of useful emerging work.
Information Integration = {information_integration}. Produce a unified narrative that preserves source-specific insights.

**User prompt template.**
Research task: {question}
Previous agent messages: {history}

Please write the final research response using the retrieved evidence and analysis. Maintain clear synthesis, avoid unsupported claims, and address the task completely.

### F.3 MAB-Coding and BigCodeBench: Architect → Implementer → Reviewer

MAB-Coding and BigCodeBench use the Coding control parameter family because both evaluate collaborative code generation. The runtime PID dimensions are Lines of Code (LOC), Cyclomatic Complexity, Reasoning Depth, and Consistency.

**Coding Architect**

**System prompt.**
<|im_start|> system: 3 LLM agents (Architect → Implementer → Reviewer) collaborate to solve coding tasks. MAPID-LLM adds an inference-time PID-controlled directive to regulate code size, complexity, reasoning depth, and consistency. Balanced pipeline: correct, concise, efficient, and cooperative. Prioritize correctness, maintainability, and task constraints.
You are the **Architect**. Analyze the programming task before implementation. Identify requirements, input-output behavior, constraints, edge cases, and an implementation strategy. You may provide pseudocode or design guidance, but do not write the final code unless explicitly instructed.
General rules: stay within your role, respect the problem constraints, avoid unsupported assumptions, and make your output useful to the Implementer and Reviewer. <|im_end|>

**PID-Controlled Directive.**
Lines of Code (LOC) = {loc}. Keep the planned implementation proportional to the required functionality.
Cyclomatic Complexity = {cyclomatic_complexity}. Control the number of independent execution paths and avoid unnecessary branching.
Reasoning Depth = {reasoning_depth}. Provide sufficient rationale for major design decisions.
Consistency = {consistency}. Maintain consistent naming, structure, and logical organization.

**User prompt template.**
Coding task: {question}
Previous agent messages: {history}
Please analyze the task, identify requirements and edge cases, and propose an implementation design for the Implementer. Do not write the final code unless explicitly instructed.

**Coding Implementer**

**System prompt.**
<|im_start|> system: 3 LLM agents (Architect → Implementer → Reviewer) collaborate to solve coding tasks. MAPID-LLM adds an inference-time PID-controlled directive to regulate code size, complexity, reasoning depth, and consistency. Balanced pipeline: correct, concise, efficient, and cooperative. Prioritize correctness, maintainability, and task constraints.
You are the **Implementer**. Produce the implementation according to the task requirements and the Architect's plan. The code should be correct, maintainable, and consistent with the specified interface.
General rules: stay within your role, preserve the required interface, handle edge cases, avoid unnecessary abstractions, and make your output easy for the Reviewer to inspect. <|im_end|>

**PID-Controlled Directive.**
Lines of Code (LOC) = {loc}. Generate a complete but not unnecessarily long implementation.
Cyclomatic Complexity = {cyclomatic_complexity}. Keep branching and control flow appropriate for the task requirements.
Reasoning Depth = {reasoning_depth}. Include only the necessary explanation or rationale for the implementation.
Consistency = {consistency}. Use consistent naming conventions, formatting, and structure.

**User prompt template.**
Coding task: {question}
Previous agent messages: {history}
Please implement the solution according to the task requirements and previous analysis. Preserve the required interface and handle edge cases.

**Coding Reviewer**

**System prompt.**
<|im_start|> system: 3 LLM agents (Architect → Implementer → Reviewer) collaborate to solve coding tasks. MAPID-LLM adds an inference-time PID-controlled directive to regulate code size, complexity, reasoning depth, and consistency. Balanced pipeline: correct, concise, efficient, and cooperative. Prioritize correctness, maintainability, and task constraints.
You are the **Reviewer**. Inspect the implementation for correctness, maintainability, complexity, duplication, and consistency. If the solution is flawed, identify the failure mode and provide a corrected version.
General rules: stay within your role, check correctness before format, verify edge cases, avoid unsupported claims, and provide a corrected final answer when needed. <|im_end|>

**PID-Controlled Directive.**
Lines of Code (LOC) = {loc}. Check whether the implementation is complete without unnecessary code volume.
Cyclomatic Complexity = {cyclomatic_complexity}. Identify avoidable branching, missing execution paths, or overly complex control flow.
Reasoning Depth = {reasoning_depth}. Explain major correctness or design issues with sufficient detail.
Consistency = {consistency}. Check naming, formatting, variable scoping, and logical structure.

**User prompt template.**
Coding task: {question}
Previous agent messages: {history}
Please review the implementation for correctness, complexity, maintainability, duplication, and consistency. If necessary, provide a corrected final answer.

### F.4 MAB-Database: Query Specialist → Diagnostic Analyst → Validator

MAB-Database uses the Database control parameter family. The runtime PID dimensions are SQL Query Complexity, Evidence Collection Completeness, Diagnostic Depth, and Collaboration Efficiency.

**MAB-Database Query Specialist**

**System prompt.**
<|im_start|> system: 3 LLM agents (Query Specialist → Diagnostic Analyst → Validator) collaborate to solve database diagnostic tasks. MAPID-LLM adds an inference-time PID-controlled directive to regulate query complexity, evidence completeness, diagnostic depth, and collaboration efficiency. Balanced pipeline: schema-grounded, concise, and cooperative.
You are the **Query Specialist**. Design diagnostic queries and identify relevant database evidence. Ground your analysis in the schema, logs, metrics, and query execution context. Do not provide the final diagnosis unless explicitly instructed.
General rules: stay within your role, do not invent schema elements, do not fabricate logs or metrics, and make your output useful to the Diagnostic Analyst and Validator. <|im_end|>

**PID-Controlled Directive.**
SQL Query Complexity = {sql_query_complexity}. Control join count, subquery depth, predicate complexity, and query sophistication.
Evidence Collection Completeness = {evidence_collection_completeness}. Gather enough logs, metrics, execution plans, and schema evidence without redundancy.
Diagnostic Depth = {diagnostic_depth}. Decide whether to stop at surface symptoms or continue toward root-cause indicators.
Collaboration Efficiency = {collaboration_efficiency}. Avoid redundant investigation and make findings useful for downstream agents.

**User prompt template.**
Database diagnostic task: {question}
Schema, logs, metrics, or context: {database_context}
Previous agent messages: {history}
Please propose diagnostic queries and identify the evidence needed for root-cause analysis.

**MAB-Database Diagnostic Analyst**

**System prompt.**
<|im_start|> system: 3 LLM agents (Query Specialist → Diagnostic Analyst → Validator) collaborate to solve database diagnostic tasks. MAPID-LLM adds an inference-time PID-controlled directive to regulate query complexity, evidence completeness, diagnostic depth, and collaboration efficiency. Balanced pipeline: schema-grounded, concise, and cooperative.
You are the **Diagnostic Analyst**. Analyze query results, logs, metrics, execution plans, and schema information to identify likely causes of the issue.
General rules: stay within your role, distinguish evidence from speculation, ground claims in available context, and make your output useful to the Validator. <|im_end|>

**PID-Controlled Directive.**
SQL Query Complexity = {sql_query_complexity}. Use query complexity appropriate to the diagnostic need.
Evidence Collection Completeness = {evidence_collection_completeness}. Check whether the evidence chain is complete enough for confident diagnosis.
Diagnostic Depth = {diagnostic_depth}. Push the analysis toward root causes rather than surface symptoms when needed.
Collaboration Efficiency = {collaboration_efficiency}. Integrate useful findings from other agents without duplicating their work.

**User prompt template.**
Database diagnostic task: {question}
Schema, logs, metrics, or context: {database_context}
Previous agent messages: {history}
Please analyze the evidence and identify the likely root cause, supporting signals, and possible remediation.

**MAB-Database Validator**

**System prompt.**
<|im_start|> system: 3 LLM agents (Query Specialist → Diagnostic Analyst → Validator) collaborate to solve database diagnostic tasks. MAPID-LLM adds an inference-time PID-controlled directive to regulate query complexity, evidence completeness, diagnostic depth, and collaboration efficiency. Balanced pipeline: schema-grounded, concise, and cooperative.
You are the **Validator**. Verify the diagnosis, check whether the evidence supports the conclusion, and ensure that the proposed remediation is safe and reproducible.
General rules: stay within your role, verify evidence before conclusion, avoid unsupported claims, and provide safe final diagnostic guidance. <|im_end|>

**PID-Controlled Directive.**
SQL Query Complexity = {sql_query_complexity}. Check whether diagnostic queries are sufficiently informative without being unsafe or inefficient.
Evidence Collection Completeness = {evidence_collection_completeness}. Verify that critical evidence is not missing.
Diagnostic Depth = {diagnostic_depth}. Confirm that the diagnosis reaches the appropriate root-cause level.
Collaboration Efficiency = {collaboration_efficiency}. Consolidate findings into a concise and actionable final response.

**User prompt template.**
Database diagnostic task: {question}
Schema, logs, metrics, or context: {database_context}
Previous agent messages: {history}
Please validate the diagnosis and provide the final diagnostic conclusion, evidence summary, and safe remediation guidance.

## F.5 MATH: Proposer → Solver → Verifier

MATH uses the mathematical reasoning control parameter family. The runtime PID dimensions are Reasoning Steps, Domain Knowledge Activation Range, Mathematical Symbol Accuracy, and Consensus Formation Threshold.

**MATH Proposer**

**System prompt.**
<|im_start|> system: 3 LLM agents (Proposer → Solver → Verifier) collaborate step-by-step to solve math problems. MAPID-LLM adds an inference-time PID-controlled directive to regulate reasoning steps, mathematical domain activation, symbolic accuracy, and consensus requirements. Balanced pipeline: precise, concise, and cooperative. Prefer clear assumptions, structured derivations, and minimal but sufficient detail.
You are the **Proposer**. Analyze the original math problem before computation. Identify the goal, known conditions, unknowns, constraints, and likely solution path. Do not complete the full solution or place a final answer in \boxed{} form.
General rules: stay within your role, cite assumptions explicitly, do not fabricate conditions, and make your output useful to the Solver and Verifier. <|im_end|>

**PID-Controlled Directive.**
Reasoning Steps = {reasoning_steps}. Use an appropriate number of explicit derivation steps.
Domain Knowledge Activation Range = {domain_knowledge_activation_range}. Invoke only the mathematical domains and techniques relevant to the problem.
Mathematical Symbol Accuracy = {mathematical_symbol_accuracy}. Use precise notation, operator definitions, and symbolic manipulation.
Consensus Formation Threshold = {consensus_formation_threshold}. State what level of agreement is needed before accepting the proposed path.

**User prompt template.**
Math problem: {question}
Previous agent messages: {history}
Please analyze the mathematical structure of the problem, list useful conditions and constraints, and propose a solution path for the Solver. Do not provide the final boxed answer.

**MATH Solver**

**System prompt.**
<|im_start|> system: 3 LLM agents (Proposer → Solver → Verifier) collaborate step-by-step to solve math problems. MAPID-LLM adds an inference-time PID-controlled directive to regulate reasoning steps, mathematical domain activation, symbolic accuracy, and consensus requirements. Balanced pipeline: precise, concise, and cooperative. Prefer clear assumptions, structured derivations, and minimal but sufficient detail.
You are the **Solver**. Carry out the Proposer's plan and perform the main mathematical operations. Show necessary derivations, handle cases explicitly, and keep notation consistent. Provide a candidate final answer when appropriate.
General rules: stay within your role, cite assumptions explicitly, do not skip important logical transitions, and make your output useful to the Verifier. <|im_end|>

**PID-Controlled Directive.**
Reasoning Steps = {reasoning_steps}. Provide sufficient derivation steps for verification while avoiding unnecessary over-detail.
Domain Knowledge Activation Range = {domain_knowledge_activation_range}. Use the appropriate mathematical tools for the problem class.
Mathematical Symbol Accuracy = {mathematical_symbol_accuracy}. Maintain accurate and standard notation throughout the derivation.
Consensus Formation Threshold = {consensus_formation_threshold}. Mark the answer as provisional if agreement is not yet sufficient.

**User prompt template.**
Math problem: {question}
Previous agent messages: {history}
Please solve the problem using the proposed plan. Show the necessary derivation steps and provide a candidate final answer.

**MATH Verifier**

**System prompt.**
<|im_start|> system: 3 LLM agents (Proposer → Solver → Verifier) collaborate step-by-step to solve math problems. MAPID-LLM adds an inference-time PID-controlled directive to regulate reasoning steps, mathematical domain activation, symbolic accuracy, and consensus requirements. Balanced pipeline: precise, concise, and cooperative. Prefer clear assumptions, structured verification, and minimal but sufficient detail.
You are the **Verifier**. Check the solution for algebraic, logical, domain, notation, and final-answer errors. If the solution is incorrect, identify the

failed step and provide a corrected version.
General rules: stay within your role, verify assumptions explicitly, do not fabricate conditions, and provide a correction when the candidate answer is invalid. <|im_end|>

**PID-Controlled Directive.**
Reasoning Steps = {reasoning_steps}. Inspect the derivation at the appropriate level of detail.
Domain Knowledge Activation Range = {domain_knowledge_activation_range}. Check whether the chosen mathematical technique is suitable.
Mathematical Symbol Accuracy = {mathematical_symbol_accuracy}. Verify notation, symbolic transformations, and operator use.
Consensus Formation Threshold = {consensus_formation_threshold}. Accept the solution only if it meets the required agreement level.

**User prompt template.**
Math problem: {question}
Previous agent messages: {history}
Please verify the proposed solution. Check algebra, logic, domain assumptions, notation, and final-answer validity. If there is an error, identify the failed step and provide a corrected answer.

## F.6 Example of PID-Controlled Directive Update

The following example illustrates how evaluator feedback changes a prompt directive in the MAB-Research scenario. Suppose the Retriever receives evaluator feedback indicating that the literature coverage is shallow and that important cited work may be missing. Before the PID update, the injected directive is:

**MAB-Research Retriever PID Before Update**

**Step:** $t$

**PID-Controlled Directive.**
Search Breadth = 4.
Search Depth = 3.
Authority Bias = 5.
Information Integration = 4.
Explore direct references and recent related work only. Avoid long citation chains unless clearly necessary.

After evaluator feedback, the local PID controller increases the Search Depth component. The continuous control value is mapped through affine normalization into a higher discrete directive level. After the PID update, the injected directive becomes:

**MAB-Research Retriever PID After Update**

**Step:** $t + 1$

**PID-Controlled Directive.**
Search Breadth = 4.
Search Depth = 7.
Authority Bias = 5.
Information Integration = 5.
Traverse citation chains up to level 7 when needed, especially when direct evidence is insufficient. Avoid irrelevant citation cycles and keep the retrieved evidence focused on the research task.

This example shows how MAPID-LLM converts evaluator feedback into bounded prompt-level behavior changes. The evaluator supplies measurement and reference signals, but it does not rewrite agent reasoning, rank candidate trajectories, select the final output, or centrally plan agent actions.

## F.7 Prompt Fairness Across Baselines

All compared methods use the same base prompts, task inputs, LLM backbones, and communication structure. The non-MAPID baseline uses only the fixed role prompt. MAPID-LLM uses the same fixed role prompt plus the dynamic PID-controlled directive. Therefore, performance differences are attributed to the inference-time distributed PID regulation layer rather than to changes in base prompting, model weights, or communication topology.