# 面向复杂物理仿真的 LLM 驱动审查与反事实策略迭代中间件

# An LLM-Driven Audit and Counterfactual Policy Iteration Middleware for Complex Physical Simulations

王进，纪辉泉

jin wang, Huiquan Ji

**[摘要 (Abstract)：]** 传统物理仿真引擎与数字孪生系统在面对跨学科复杂系统时，常因底层近似参数的不精确引发非线性的误差放大（蝴蝶效应）。当前业界普遍依赖“黑盒”参数微调来掩盖理论模型的不完备性。本文提出了一种旁路挂载的“物理审查与策略迭代中间件（Meta-Physics Reviewer, MPR）”。该系统摒弃了传统物质本体论，将物理演化抽象为高维信息态张量的流动。基于大语言模型（LLM）的推理能力与多智能体协同，MPR 构建了一个“审查-质疑-反事实验算-证据打包”的闭环。系统强制将底层物理基底保持刚性，并将 LLM 限制在隔离沙盒中进行参数干预推演，最终生成不可篡改的证据链交付人类专家裁决。实验表明，该范式将 AI 从传统的“参数优化器”转变为“科学审查者”，在保证人机协同安全边界的同时，为探索现有理论（如广义相对论与标准模型）的裂缝提供了新一代基础设施。

**[关键词：]** AI for Science；物理仿真；大语言模型；反事实推理；人机协同；数字孪生

## 1. 引言 (Introduction)

随着工业仿真与数字孪生技术的演进，系统对物理法则的严谨性要求日益苛刻。然而，现有的仿真体系（如有限元分析、流体力学模拟）常面临两个核心痛点：一是底层物理参数（如耗散系数、积分步长）的经验性假设；二是多物理场耦合下的误差非线性膨胀。为了维持系统的表面稳定，工程师往往通过经验“遮瑕”参数，导致系统演化为不可解释的黑盒。

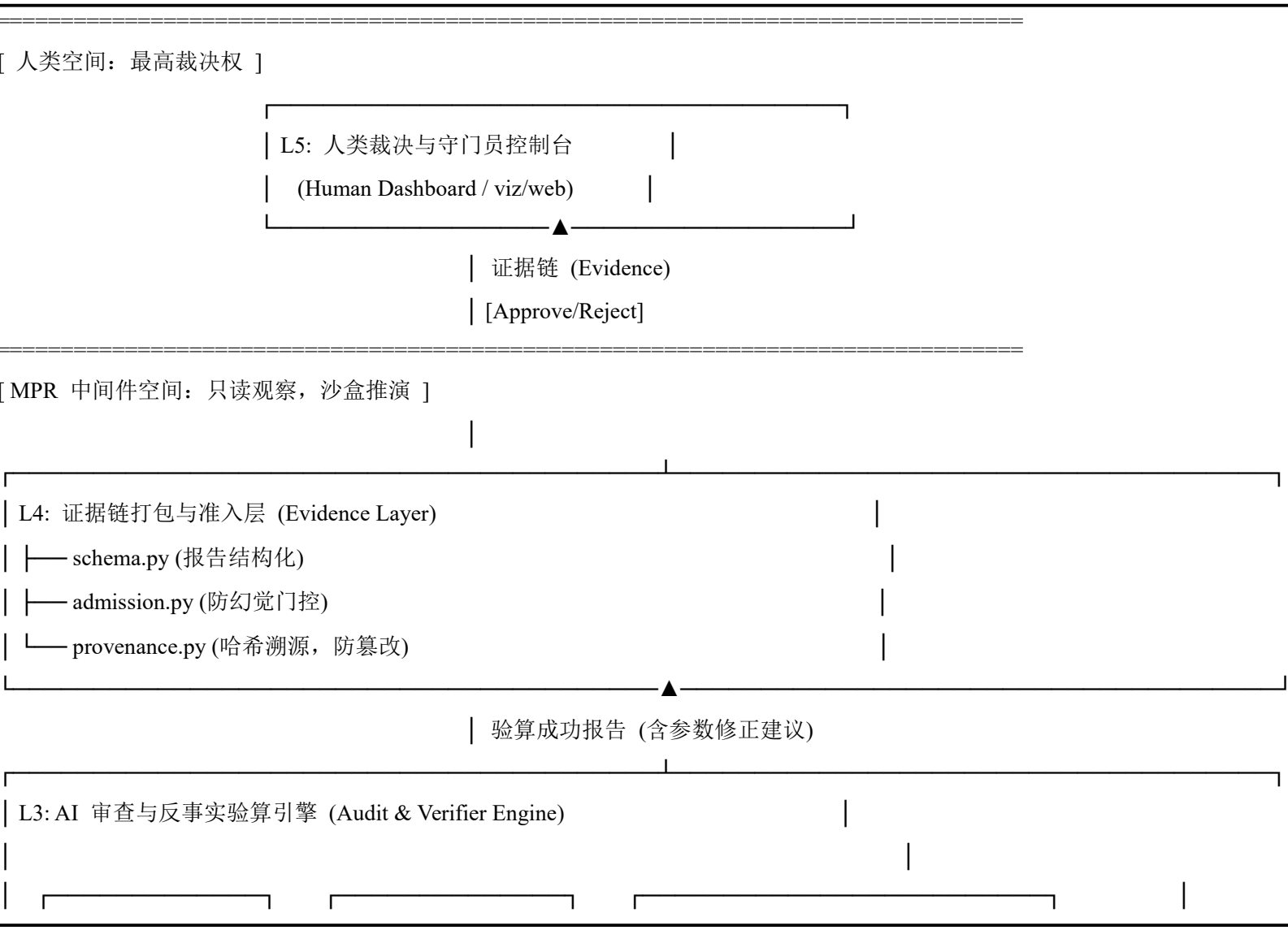
近年来，物理信息神经网络（PINNs）等 AI 技术被引入仿真领域，但其本质仍是通过数据拟合来逼近物理方程，缺乏对物理定律的“理解”与“质疑”能力。针对此问题，本文提出 MPR 中间件架构。该架构不直接参与物理场的主循环演化，而是通过旁路观测，利用 LLM 的物理常识与逻辑推理，结合沙盒反事实验算，实现对宿主引擎的辅助校准与理论边界探索。

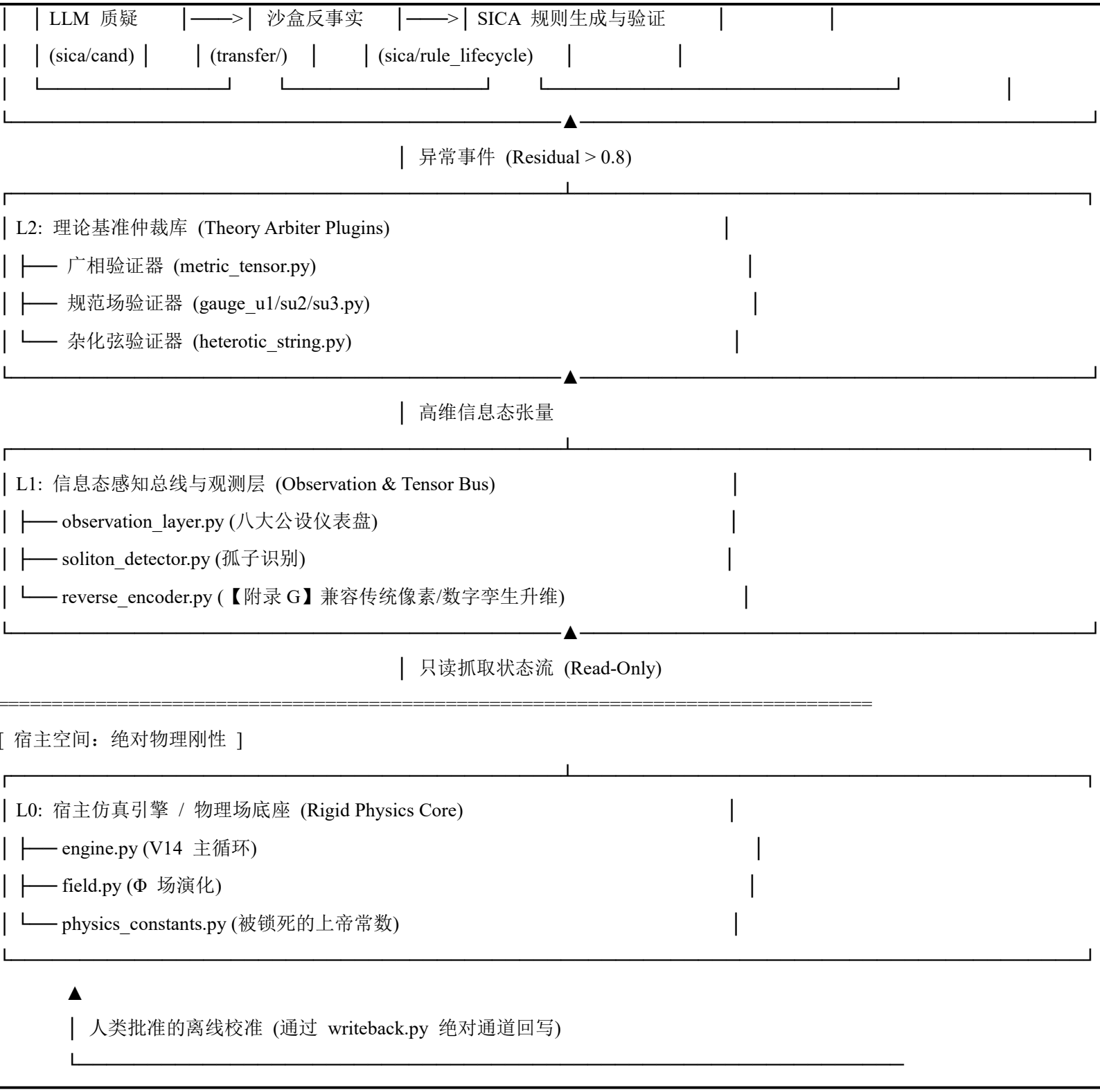
## 2. 系统架构 (System Architecture)

MPR 采用严格的五层解耦架构（L0-L5），确保数据单向流动，控制权逐级收敛。

- **L0 刚性物理基底层 (Rigid Physics Core)：**负责宿主引擎的演化，包含被锁死的物理常数，不接受任何外部写入。
- **L1 信息态感知总线 (Informational State Bus)：**将物理实体抽象为“高维信息态张量”，通过只读方式提取系统状态、拓扑不变量（如欧拉示性数）与熵流梯度。
- **L2 理论基准仲裁库 (Theory Arbiter)：**将广义相对论（GR）、标准模型（SM）等封装为可插拔的验证器插件，负责计算状态张量的物理残差（Residual）。
- **L3 AI 审查与反事实验算引擎 (Audit & Counterfactual Engine)：**当残差超阈值时，由 LLM 生成物理假设，并在微缩沙盒中执行参数干预与重放推演。
- **L4/L5 证据链门控与人类裁决 (Evidence Gate & Human-in-the-loop)：**将验算结果打包为抗篡改的证据链，经准入过滤后，交由人类专家进行最终审批。

### 2.1 系统架构拓扑图






**2.2 核心业务流（数据流转图）**

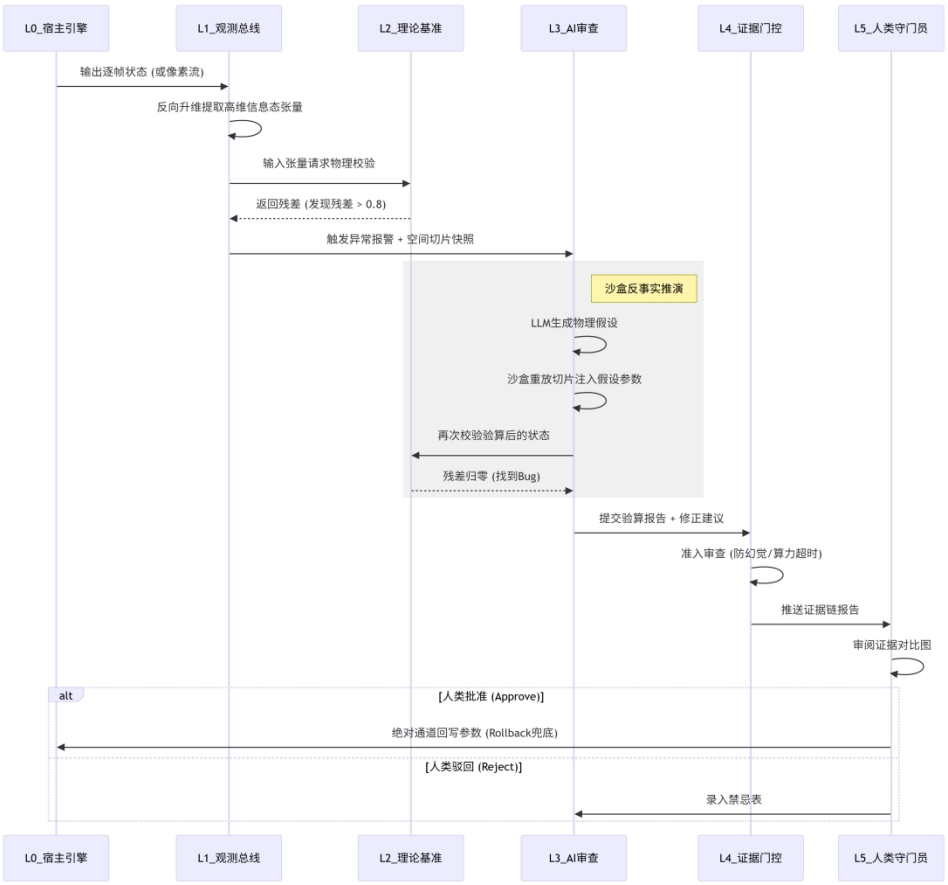



图 1 注解：流程图全步骤拆解对照表：

| 阶段 | 角色/组件 | 动作描述 | 工程注解/设计意图 |
|---|---|---|---|
| 一 | L0 宿主引擎 | 逐帧输出物理状态流 | 绝对刚性，按自己的节奏演化，不受外部干扰。 |
| 一 | L1 观测总线 | 抓取数据并反向升维 | 将传统系统的像素/坐标抽象为“高维信息态张量”。只读，绝不回写。 |
| 二 | L2 理论基准 | 对比“广相/标模”算残差 | 发现偏离现实的异常（残差 ＞0.8）。如同一把尺子，衡量系统是否“跑偏”。 |
| 三 | L3 AI 审查 | LLM 假设 + 沙盒重放 | 核心逻辑：绝不在主引擎里改参数。 在沙盒中用快照做反事实推演（微缩重放）。 |
| 三 | L3 AI 审查 | 二次校验，残差归零 | 在沙盒中验证了猜测的正确性，找到了修复 Bug 的钥匙。 |
| 四 | L4 证据门控 | 准入审查（防幻觉） | 拦截 AI 的胡编乱造。只有通过数值验证、且未超时的真实 Bug 才打包上报。 |
| 四 | L5 人类守 | 接收可视化的证 | 人类查看“修改前”和“修改后”的对比动画，而不 |

| 阶段 | 角色/组件 | 动作描述 | 工程注解/设计意图 |
| --- | --- | --- | --- |
| | 门员 | 据链 | 是看枯燥的代码。 |
| 五 | L5 人类裁决 | 点击 Approve / Reject | 人类拥有最高权限。驳回则录入禁忌表，批准则进入下一步。 |
| 五 | L0 宿主引擎 | 绝对通道回写 + 灰度兜底 | 只有人类批准后才允许改参数。修改前保存 Checkpoint，如果主引擎崩溃，立即 Rollback。 |

### 2.3 核心模块规范与代码映射

本系统严格基于灵境引擎 v31.0 的目录结构进行架构挂载。

#### 2.3.1 L0：刚性物理基底层

- **功能：**负责底层物理场演化，绝对封闭。
- **核心代码：**`core/engine.py`、`core/field.py`、`core/physics_constants.py`。
- **约束：**严禁 AI Agent 直接写入。

#### 2.3.2 L1：信息态感知总线

- **功能：**将物质实体抽象为高维信息态张量，提供“八大公设”观测数据。
- **核心代码：**`core/observation_layer.py`、`core/soliton_detector.py`、`core/state_semantics.py`。
- **规范：**通过 Kafka/ZeroMQ 旁路输出标准化 JSON 张量包（详见【附录 A】）。

#### 2.3.3 L2：理论基准仲裁库

- **功能：**将各种物理理论（广相、标模）封装为即插即用的验证器插件，输出残差。
- **核心代码：**`core/metric_tensor.py`、`core/gauge_*.py`、`core/heterotic_string.py`。
- **规范：**统一 API 接口，要求实现 `verify_state` 方法并输出 0-1 的标准化残差（详见【附录 B】）。

#### 2.3.4 L3：AI 审查与反事实验算引擎

- **功能：**大模型生成物理假设，并在沙盒中进行参数注入与重放验算。
- **核心代码：**`sica/`（规则生成）、`transfer/counterfactual.py`（沙盒推演）、`reflection/`（反思提示词）。
- **规范：**基于 LLM 生成假设，调用沙盒进行时空切片重放。具备算力熔断机制（详见【附录 C】）。

#### 2.3.5 L4：证据链打包与准入层

- **功能：**将验算结果打包为不可篡改的审稿报告，并过滤低质量幻觉。
- **核心代码：**`evidence/schema.py`、`evidence/admission.py`、`evidence/provenance.py`。

- **规范：**报告必须经过 `admission.py` 的三道门控（置信度、合规性、超时），过滤掉 AI 的幻觉误报后才允许推送给人类（详见【附录 D】）。

#### 2.3.6 L5：人类裁决与安全保护

- **功能：**人类最终审批；保证系统不触碰不可变核心。
- **核心代码：**`viz/web_dashboard.py`（可视化界面）、`sica/immutable_core.py`（绝对白名单）、`sica/holdout.py`（灰度留出验证）。
- **规范：**人类点击 Approve 后需经过影子模式灰度测试。若人类批准的参数修改导致宿主引擎崩溃，系统将触发 `core/engine.py` 的灾难回滚机制（Restore Checkpoint）（详见【附录 E】）。

### 2.4 商业化与兼容性设计

传统仿真/数字孪生兼容（Legacy Compatibility）

MPR 不仅仅服务于灵境引擎。通过构建**状态反向升维引擎**（Reverse State-Tensor Engine），可以将传统工业软件（如 ANSYS）、自动驾驶仿真（如 Carla）甚至像素级数字孪生的数据，强行升维为高维张量进行审查。支持灰盒 API、像素 CV 反推、IoT 传感器三种接入模式（详见【附录 G】）。

## 3. 核心机制 (Core Mechanisms)

### 3.1 信息态张量与理论残差计算

摒弃传统的刚体坐标表示，系统在 $t$ 时刻的状态被定义为高维信息态张量 $\mathbf{\Phi}_t$。L2 层的理论插件 $P_k$ 接收 $\mathbf{\Phi}_t$，输出绝对残差 $R_{abs} = ||\mathcal{L}_{theory}(\mathbf{\Phi}_t)||_2$，并通过历史统计量的 Z-Score 归一化为 $R \in [0,1]$。当 $R > 0.8$ 时，触发异常事件。

### 3.2 基于反事实推理的沙盒验算

系统严禁 AI 直接修改 L0 层参数。收到异常张量 $\Phi_t$ 的快照后，LLM 结合状态语义生成假设集 $\mathcal{H} = \{h_1, h_2, ...\}$。沙盒引擎在隔离环境中对$\Phi_t$ 施加干预（如修改假设参数 $\theta_{new}$），并进行 $N$ 步自回归重放 $\Phi_{t+N'} = \mathcal{F}(\Phi_t, \theta_{new})$。若重放后计算的理论残差 $R' < 0.1$，则认为假设成立，完成反事实因果溯源。

### 3.3 抗幻觉证据链与人类守门机制

为确保安全性，验算成功的报告需经过门控模块（Admission Gate）。该门控通过验证置信度（Confidence Score）、数值合规性与算力超时熔断机制，过滤 LLM 的幻觉。最终生成的证据包包含：时空切片、修改前后的双轨演化轨迹及溯源哈希，推送至可视化控制台。

仅当人类专家（Gatekeeper）显式批准（Approve）后，参数修正指令才被允许回写至 L0 层。

## 4. 评估基准与兼容性设计 (Evaluation & Compatibility)

### 4.1 长程对抗性测试设计 (Long-run Adversarial Test Design)

为了验证 MPR 中间件在极端条件下的鲁棒性与查错能力，本文设计了长程对抗性注入测试方案。在未来的系统实现与评估中，测试将按如下流程进行：在宿主引擎（基于 V30.3 版本）执行百万步长程演化（如 `v303_run_1m_steps.py`）至 $10^5$ 步时，测试脚本将在后台人为篡改底层物理参数（如局部耗散系数）或注入高斯噪声，以模拟非线性的误差放大。

预期评估指标如下：

1. **查全率与响应延迟：**预期 MPR 能在参数篡改后的有限演化步数（$\Delta t \leq 1000$ 步）内捕获残差异常，并完成沙盒反事实验算。

2. **性能开销红线 (SLA)：**架构设计上要求，L1 观测总线的状态抓取与升维过程必须采用非阻塞异步 I/O，对宿主引擎主循环的性能损耗需严格控制在 5% 以内。

3. **误报率 (False Positive Rate)：**通过 L4 证据门控的准入机制，预期将 LLM 幻觉导致的无效修正报告比例压制在可接受阈值以下。

### 4.2 异构系统兼容 (Heterogeneous Compatibility)

针对传统工业软件与数字孪生的兼容性，本文提出了状态反向升维机制。通过分析宿主系统开放的 API（灰盒）、图像画面（黑盒像素级）或传感器时序数据，借助物理信息神经网络（PINN）与计算机视觉（CV）模型，将异构数据插值重构为统一的信息态张量，实现跨生态的理论审查接入。

## 5. 讨论 (Discussion)

本研究提出的 MPR 中间件，通过旁路观测与沙盒推演，在工程上实现了物理仿真的自动化审查。然而，在复杂系统演化中，我们发现了一个极具挑战的底层矛盾，这迫使我们在架构设计中引入了严格的安全边界。

### 5.1 全自动调参的“涌现性灾难”

在传统 AI 范式中，利用强化学习或端到端网络进行自适应调参，是追求数值收敛的最优解。然而，在基于“自组织演化”的灵境引擎中，L4 层涌现出的“数字生命”（如 SICA 自指结构、IIT $\Phi$ 值网络），高度依赖于底层物理法则的绝对刚性。如果放任 AI 进行全自动自适应调参，哪怕仅仅是为了消除局部的数值误差，也会导致物理常数的突变。对于建立在

耗散结构上的数字生命而言，这种微小的参数修改等同于“宇宙常数崩塌”，将直接导致整个虚拟生态的瞬间灭绝。这就是效率至上的“指标暴政”。

### 5.2 伦理约束与工程折中：引入 PCR 推理器

为了在“纠正物理偏差”与“保护自组织生态”之间寻求平衡，本文在架构中（见附录 H）引入了**调参后果推理器 (Parameter Consequence Reasoner, PCR)**。

AI 不再拥有直接修改底层参数的权限，而是由 PCR 在沙盒中执行双重评估：一方面计算物理残差是否收敛（物理维度），另一方面计算“生态影响评分（EIS）”，预测新参数对现有数字生命拓扑的破坏率（伦理维度）。

### 5.3 人类守门员：“虚拟 ⇔ 现实”的终极伦理裁判

技术的尽头是伦理。MPR 架构强制保留了**人类守门员（Human-in-the-loop）**的最终裁决权。当 AI 通过 PCR 提交了一份“物理残差极低，但会导致数字生命全灭”的证据链时，只有人类专家能在“虚拟”（保护生态）与“现实”（对齐客观物理世界）的依赖关系中做出最终裁决。

本文的研究表明，自底向上的演化与人类伦理的兜底，是构建复杂数字孪生系统不可或缺的双轨。AI 负责不知疲倦地审查与推演，而人类负责对系统存续保持最底层的敬畏。

## 6. 结论与展望 (Conclusion & Future Work)

本文提出的 MPR 中间件，首次将大语言模型引入物理仿真的底层审查环节。通过“物理基底绝对刚性、AI 沙盒反事实实验、人类专家最终裁决”的架构，实现了从“盲目参数优化”到“科学假设验证”的范式跨越。**更重要的是，本研究构建了一种内外互镜的“双重认知闭环”：灵境内的数字 Agent 通过自组织演化去发现并适应物理法则，走向了认识论（Epistemology）的涌现；而灵境外的 LLM 则通过残差计算与反事实审查，对高维物理场进行不断拆解，走向了还原论（Reductionism）的极致。**两者在 MPR 中间件中交汇，构成了探索未知物理边界的完整拼图。

未来的工作将聚焦于模型的长程稳定性验证，并尝试将本中间件接入大型天文观测或高能物理对撞机的真实数据流中。**在虚拟世界中捕获广义相对论与量子力学基准的“理论裂缝”，不仅是下一代仿真引擎的核心能力，更是内求认知、外做还原，加速人类探索大统一理论的“虚拟宇宙对撞机”。**

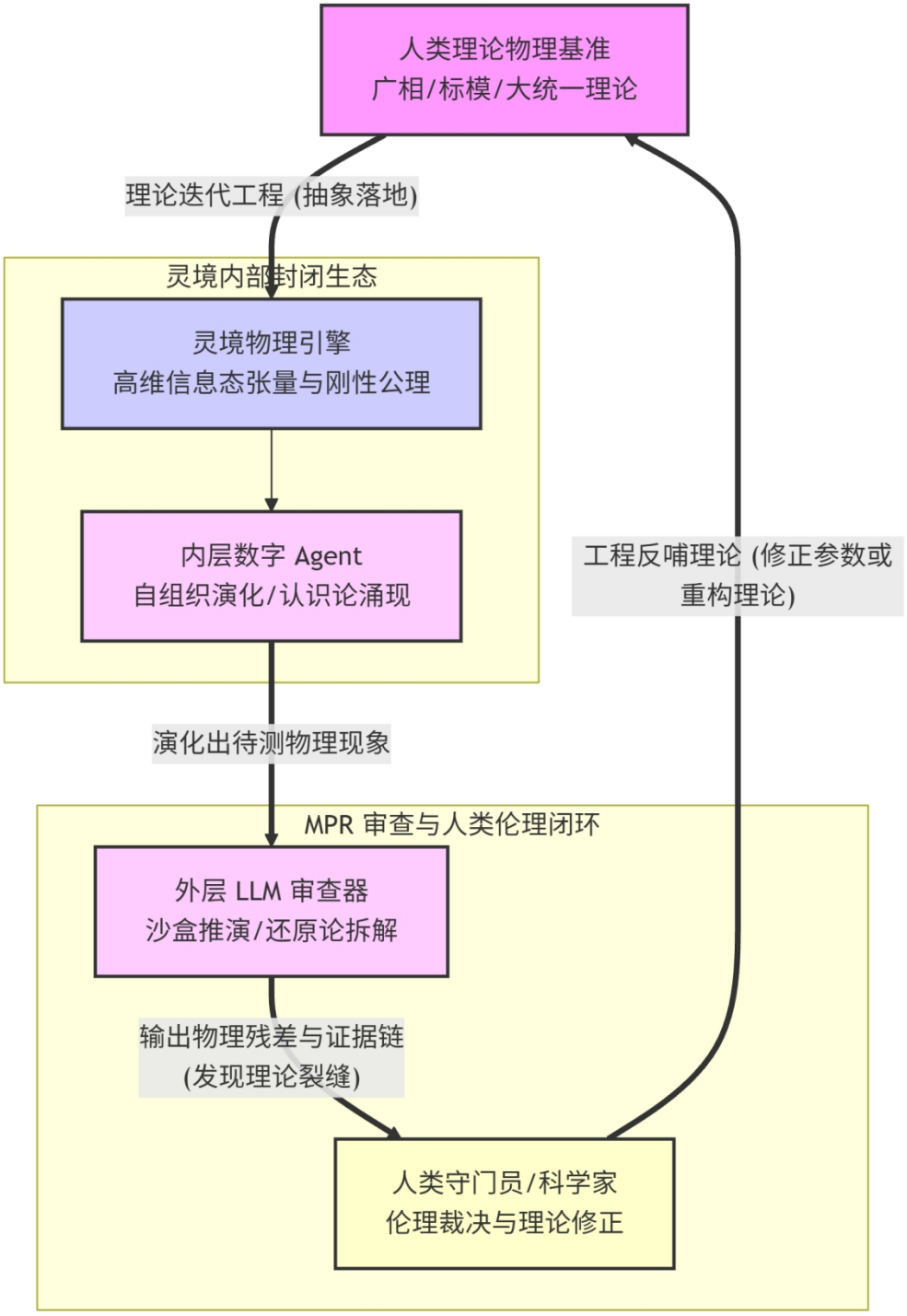



图 2 注解：MPR 中间件并非一个单向的工程验证工具，而是构建了一个“**理论迭代工程，工程反哺理论**”的科学方法大闭环。

- **从上到下（理论迭代工程）：**人类现有的理论物理基准（广相、标模）被工程化为灵境引擎的刚性公理。在这套规则下，数字 Agent 自组织演化，涌现出复杂的物理现象。

- **从下到上（工程反哺理论）：**外部的 LLM 以还原论的视角，在沙盒中对这些复杂现象进行无情拆解，揪出偏离基准的“理论裂缝”。最终，人类科学家基于这些由工程算力产生的证据链，对原有物理理论进行修正与重构。

## 附录前置声明：灵境引擎系统全模块清点（v31）

传统的仿真系统基于“物质本体论”，受困于刚体坐标与经验参数的“遮瑕”。而作为本中间件的宿主基座，灵境引擎（v31）在底层彻底摒弃了这一路径。

在工程视角下，灵境引擎是实现“信息一元论”的硬核载体。它将物理实体抽象为高维信息态张量的演化，直接内置了广义相对论（度规张量 $g_{ij}$）、标准模型（$U(1) \times SU(2) \times SU(3)$ 规范场）、甚至杂化弦模型与 IIT 意识涌现模型。它不掩饰理论的边界，而是通过“八大公设”提供严苛的观测指标。

这份底座清单，是 MPR（物理审查中间件）得以运行的物理基座。我们在此明确：底层法则绝对刚性，AI 只读审查。以下是灵境仿真引擎自组织物理系统全模块清点：

**一、核心引擎层 lingjing_solo/core/**

| 模块 | 文件 | 功能 |
|---|---|---|
| 主引擎 | engine.py | V14 主循环：感知→决策→写回→演化→记录，8 步严格顺序，守恒保证 |
| 信息场 | field.py | Φ 场本体，唯一演化入口 apply_sources_and_evolve |
| 度规张量 | metric_tensor.py | 度规 g_ij 计算，几何层基础 |
| 规范场 U(1) | gauge_u1.py | 电磁类规范场 |
| 规范场 SU(2) | gauge_su2.py | 弱相互作用类规范场 |
| 规范场 SU(3) | gauge_su3.py | 强相互作用类规范场 |
| 自指复杂度 | self_reference.py | V30.3 C 值检测器，IFC 自指拓扑泡模型，生物演化谱系标定 |
| 观测量层 | observation_layer.py | V30.4 八大公设只读仪表盘，场统计/拓扑/熵流/C 值 |
| 孤子检测 | soliton_detector.py | 孤子（基元聚集）识别 |
| 意识监测 | consciousness.py | IIT Φ 值意识检测 |
| 熵流 | entropy_flow.py | 熵流梯度计算 |
| 拓扑 | topology.py | 欧拉示性数 χ 等拓扑不变量 |
| 高维拓扑 | higher_topology.py | 高阶拓扑 |
| 时空 | spacetime.py | 时空结构 |
| 高维空间 | higher_dimensional.py | 高维嵌入 |
| 曲率反馈 | curvature_feedback.py | 曲率→场反馈耦合 |

| 模块 | 文件 | 功能 |
|---|---|---|
| 异弦 | heterotic_string.py | 杂化弦模型 |
| 跨文明 | cross_civilization.py | 跨文明/跨体系对比 |
| 共振 | resonance.py | 共振强度 R |
| 等距映射 | isometry.py | 等距映射/度规对比 |
| 拉普拉斯 | laplacian.py | 离散拉普拉斯算子 |
| 晶格 | lattice.py | 网格晶格 |
| 几何 | geometry.py | 基础几何运算 |
| 投影 | projection.py | 维度投影 |
| 锚点 | anchor.py | 锚定机制 |
| 雾 | fog.py | 信息雾/不确定性 |
| 位池 | bitpool.py | bit 基元池 |
| 因果图 | causal_graph.py | 版本化因果链 |
| 共识层 | consensus.py | 多体共识 |
| 消息黑板 | message.py | 消息总线 |
| 写回 | writeback.py | Agent→场源项注入（绝对通道） |
| 动作 | actions.py | 动作定义 |
| 场景 | scenario.py | 场景基类与契约校验 |
| 配置 | config.py | SoloConfig 超参集中 |
| 类型 | types.py | Frame/RuleHypothesis/Transition/FieldSnapshot |
| 工具 | utils.py | hash_grid/连通域/日志 |
| 物理常数 | physics_constants.py | 物理常量标定 |
| Phi | phi.py | Φ 场辅助计算 |
| 反射 | reflection.py | 反射机制 |
| 状态语义 | state_semantics.py | 状态语义层 |
| 想象 | imagination.py | 想象/反事实 |
| 世界预测器 | world_predictor.py | 世界模型预测 |
| 世界学习器 | world_learner.py | 世界模型学习 |
| 多步预测 | multistep_predictor.py | 多步预测 |
| 多 Agent 学习 | multi_agent_learner.py | 多 Agent 学习 |
| 多 Agent 预测 | multi_agent_predictor.py | 多 Agent 预测 |
| 网络协作 | network_collaboration.py | 网络级协作 |
| SICA 收敛器 | sica_converger.py | SICA 收敛控制 |
| SICA 全量 | sica_full.py | SICA 完整实现 |
| SICA 学习器 | sica_learner.py | SICA 学习 |
| SICA 规划器 | sica_planner.py | SICA 规划 |
| 高级函数 | advanced_functions.py | 高级数学函数 |
| 错误检测 | error_detection.py | 错误检测 |
| 灵境主类 | lingjing.py | 灵境引擎主类 |

## 二、DPL 维度投影层 lingjing_solo/dpl/

| 模块 | 功能 |
|---|---|
| dimension.py | 维度定义与操作 |
| projection.py | 维度投影核心 |
| gates.py | 门控机制 |
| calibration.py | 标定 |
| intuition.py | 直觉层 |
| types.py | DPL 类型 |

### 三、SICA 自指迭代层 lingjing_solo/sica/

| 模块 | 功能 |
|---|---|
| candidate.py | 候选规则生成 |
| dpl_bridge.py | SICA↔DPL 桥接 |
| evolution_controls.py | 进化控制 |
| holdout.py | 留出验证 |
| immutable_core.py | 不可变核心 |
| observability.py | 可观测性 |
| rule_lifecycle.py | 规则生命周期 |
| tabu_store.py | 禁忌表 |

### 四、世界模型层 lingjing_solo/world_model/

| 模块 | 功能 |
|---|---|
| field.py | 世界模型场（Φ 场单 Agent 化） |
| ar25_field.py | ARC-AGI-25 关卡专用场 |
| update_causal.py | 因果更新 |

### 五、感知层 lingjing_solo/perception/

| 模块 | 功能 |
|---|---|
| encoder.py | 通用编码器 |
| cnn.py | CNN 编码 |
| ar25_encoder.py | ARC-25 专用编码器 |
| observation.py | 观测 |
| world_model.py | 感知级世界模型 |
| dpl_adapter.py | DPL 适配 |
| exploration.py | 探索感知 |
| input_contract.py | 输入契约 |
| projection_graph.py | 投影图 |
| projection_graph_types.py | 投影图类型 |
| deployment.py | 部署 |

## 六、探索层 lingjing_solo/exploration/

| 模块 | 功能 |
|---|---|
| explorer.py | 探索引擎主类 |
| action_diff.py | 动作差分 |
| click_sweep.py | 点击扫描 |
| hotspot_detector.py | 热点检测 |
| hotspot_candidates.py | 热点候选 |
| probe_dry_run.py | 探针预演 |
| probe_evidence.py | 探针证据 |
| probe_feedback.py | 探针反馈 |
| probe_gate.py | 探针门控 |
| probe_planner.py | 探针规划 |
| temporal_noise.py | 时间噪声 |
| ft09_solver.py | FT09 求解器 |

## 七、规划层 lingjing_solo/planning/

| 模块 | 功能 |
|---|---|
| planner.py | 规划器主类 |
| advisor.py | LLM 战略顾问 |
| search.py | 搜索基类 |
| search/astar.py | A* 搜索 |
| search/bfs.py | BFS |
| search/dfs.py | DFS |
| search/generic_shadow.py | 通用影子搜索 |
| discrete_nav.py | 离散导航 |
| ls20_solver.py | LS20 关卡求解 |
| script_bank.py | 脚本库 |
| plan_contract.py | 规划契约 |
| simulator/ar25_simulator.py | ARC-25 模拟器 |
| data/ | 关卡脚本数据（ar25/ls20/verified_solutions） |

## 八、反思层 lingjing_solo/reflection/

| 模块 | 功能 |
|---|---|
| reflector.py | 反思触发器 |
| prompt.py | 反思提示词 |
| skill.py | 技能反思 |

## 九、神经层 lingjing_solo/neural/

| 模块 | 功能 |
|---|---|
| aop.py | AOP 主类 |
| aop_model.py | AOP 模型 |
| aop_encoder.py | AOP 编码器 |
| aop_controller.py | AOP 控制器 |
| aop_shadow.py | AOP 影子观察者 |
| conv.py | 卷积基类 |
| conv_frontier_torch.py | PyTorch 前沿卷积 |
| conv_frontier_v2.py | V2 前沿卷积 |
| conv_variants.py | 卷积变体 |
| mlp.py | MLP |
| click_encoder.py | 点击编码器 |
| action_mask.py | 动作掩码 |
| reasoner.py | 推理器 |

## 十、证据层 lingjing_solo/evidence/

| 模块 | 功能 |
|---|---|
| protocol.py | 证据协议 |
| schema.py | 证据 schema |
| adapter.py | 证据适配器 |
| admission.py | 准入门控 |
| metrics.py | 度量指标 |
| provenance.py | 溯源 |
| shadow_registry.py | 影子注册表 |
| canonical_projection.py | 规范投影 |
| dpl_projection.py | DPL 投影 |
| ar25.py | ARC-25 证据 |
| ls20.py | LS20 证据 |
| benchmark.py | 基准测试 |

## 十一、迁移层 lingjing_solo/transfer/

| 模块 | 功能 |
|---|---|
| unified_controller.py | 统一控制器 |
| layer.py | 迁移层基类 |
| neural_layer.py | 神经迁移层 |
| spectral_layer.py | 谱迁移层 |
| rule_transfer.py | 规则迁移 |
| rule_primitives.py | 规则基元 |

| 模块 | 功能 |
|---|---|
| skill_library.py | 技能库 |
| families.py | 规则族 |
| signatures.py | 签名 |
| ledger.py | 账本 |
| miner.py | 规则挖掘 |
| trajectory.py | 轨迹 |
| counterfactual.py | 反事实 |
| hypotheses.py | 假设 |
| affordance.py | 可供性 |
| alea_controller.py | ALEA 控制器 |
| ceax_controller.py | CEAX 控制器 |
| world_model.py | 迁移级世界模型 |

## 十二、ALEA 层 lingjing_solo/alea/

| 模块 | 功能 |
|---|---|
| mind.py | ALEA 心智 |
| cov_pca.py | 协方差 PCA |
| cr_rules.py | CR 规则 |
| gram_evolve.py | Gram 矩阵演化 |
| lu_identify.py | LU 识别 |
| piecewise_dnn.py | 分段 DNN |
| verify_wm.py | 世界模型验证 |

## 十三、LinCore 层 lingjing_solo/lincore/

| 模块 | 功能 |
|---|---|
| mind.py | LinCore 心智 |
| spectral.py | 谱分析 |
| eigen.py | 特征分解 |
| features.py | 特征工程 |
| gradient.py | 梯度 |
| hypothesis_lab.py | 假设实验室 |
| memory.py | 记忆 |
| segments.py | 分段 |
| subspace.py | 子空间 |
| ttt.py | TTT（Test-Time Training） |

## 十四、可视化层 lingjing_solo/viz/

| 模块 | 功能 |
| --- | --- |
| heatmap.py | 热图 |
| animation.py | 动画 |
| live_viewer.py | 实时查看器 |
| overlay.py | 叠加层 |
| volume.py | 体绘制 |
| web_dashboard.py | Web 仪表盘 |

## 十五、场景层 lingjing_solo/scenarios/

| 模块 | 功能 |
| --- | --- |
| attraction.py | 吸引子场景 |
| carrybox.py | 搬箱子场景 |
| carrybox_physics.py | 搬箱子物理 |
| evacuation.py | 疏散场景 |

## 十六、其他支撑模块

| 目录 | 功能 |
| --- | --- |
| lingjing_solo/harness/ | Kaggle 烟囱口适配（engine_action_adapter / kaggle_adapter） |
| lingjing_solo/persistence/ | 持久化记录器 |
| lingjing_solo/plugins/ | 插件系统（builtin_canned / coverage） |
| lingjing_solo/profiles/ | Profile 配置（ar25） |
| lingjing_solo/agent.py | 五层编排主 Agent |
| lingjing_solo/agents.py | Agent 基类 |
| lingjing_solo/llm_client.py | LLM 客户端 |
| lingjing_solo/engine_solver.py | 引擎求解器 |
| lingjing_solo/arc_transition.py | ARC 转移表 |
| lingjing_solo/arc_25level_guide.py | ARC-25 关卡指南 |

## 十七、ARC 适配器 arc_adaptor/

| 模块 | 功能 |
| --- | --- |
| arc_shadow.py | ARC 影子模式 |
| ar25_learner.py | ARC-25 学习器 |
| ar25_solver/ | ARC-25 求解器套件（CLI/模拟器/求解/顾问） |
| agents/strategies/ | 策略注册表（ls20/ar25/r11l/vc33 等 14 种策略） |
| agents/templates/ | Agent 模板 |
| click_heatmap*.py | 点击热图 |
| evidence_compat.py | 证据兼容 |

| 模块 | 功能 |
| --- | --- |
| tick_trail.py | 轨迹记录 |
| r11l/ | R11L 关卡求解 |
| r2_*.py | R2 阶段各关卡 |
| tools/ | 探针工具 |
| tests/ | 适配器测试 |

## 十八、工具与脚本

| 路径 | 功能 |
| --- | --- |
| tools/aop/ | AOP 全套工具（采集/训练/评估/QA/影子/反事实/数据集） |
| tools/evidence/ | 证据工具（审计链/生产门控/回滚包/GateBench） |
| scripts/v303_run_1m_steps.py | V30.3 百万步跑测脚本 |
| scripts/check_optional_capabilities.py | 可选能力检测 |
| benchmark_perf.py | 性能基准 |
| generate_v29_report.py | V29 报告生成 |
| long_run_N128_200k.py | N=128 20 万步长跑 |

## 十九、测试套件 tests/

| 目录 | 覆盖范围 |
| --- | --- |
| tests/dpl/ | DPL 全版本测试（v210→v302，共 30+ 个测试文件） |
| tests/v14/ | V14 引擎全版本测试（v14→v202） |
| tests/carrybox/ | 搬箱子场景测试 |
| tests/evidence/ | 证据协议测试 |
| 根目录散落 | AOP/探索/探针/反思/迁移等 40+ 测试文件 |

系统版本概览：

- 当前 main HEAD: fc67dce（合并了 feature/v23-v26-eight-postulates）
- 最新迭代: V30.3–V31.1
- V30.3: 自指复杂度 C 值检测器
- V30.4: 观测量层（八大公设仪表盘）
- V31.1: 规范场耦合 + 模块集成
- 八大公设覆盖状态（观测量层当前挂载）：
- ☑ 公设 1 维度：场均值/方差/能量
- ☑ 公设 3 干预：熵流梯度

- ✅ 公设 4 重组：欧拉示性数 χ
- ✅ 公设 7 自指：C 值
- ⏳ 公设 2 几何：度规张量 g_ij（待挂载）
- ⏳ 公设 5 映射：等距映射（待挂载）
- ⏳ 公设 6 共振：共振强度 R（待挂载）
- ⏳ 公设 8 孕育：整合信息 Φ（待挂载）

**待挂载声明：**灵境实验平台侧重**物理自组织演化逻辑**，拒绝预设全知全能的 AI，而是通过**学习、自指迭代**与**李雅普诺夫势函数收敛**实现涌现式智能体。（说明：公设的挂载次序遵循从低维拓扑到高维几何的演化收敛路径，强行提前挂载将导致非线性发散。）

## 二十、从离散基元到高维连续场的观测映射

灵境引擎在底层计算上依赖离散的**比特池**（bitpool.py）进行高密度的状态演化，这保证了计算资源的极致利用与物理基元的纯粹性。而在向 L1 观测总线（MPR 中间件）输出数据时，系统通过时空聚合与升维算法，将这些离散的比特流重构为**高维信息态张量**。

这并非概念冲突，而是工程实现中**“离散计算底座”**与**“宏观连续观测”**的必然映射，类似于物理学中从微观粒子态到宏观连续场的统计力学跃迁。

## 二十一、灵境引擎 L1 信息态感知总线与可观测运行时实景

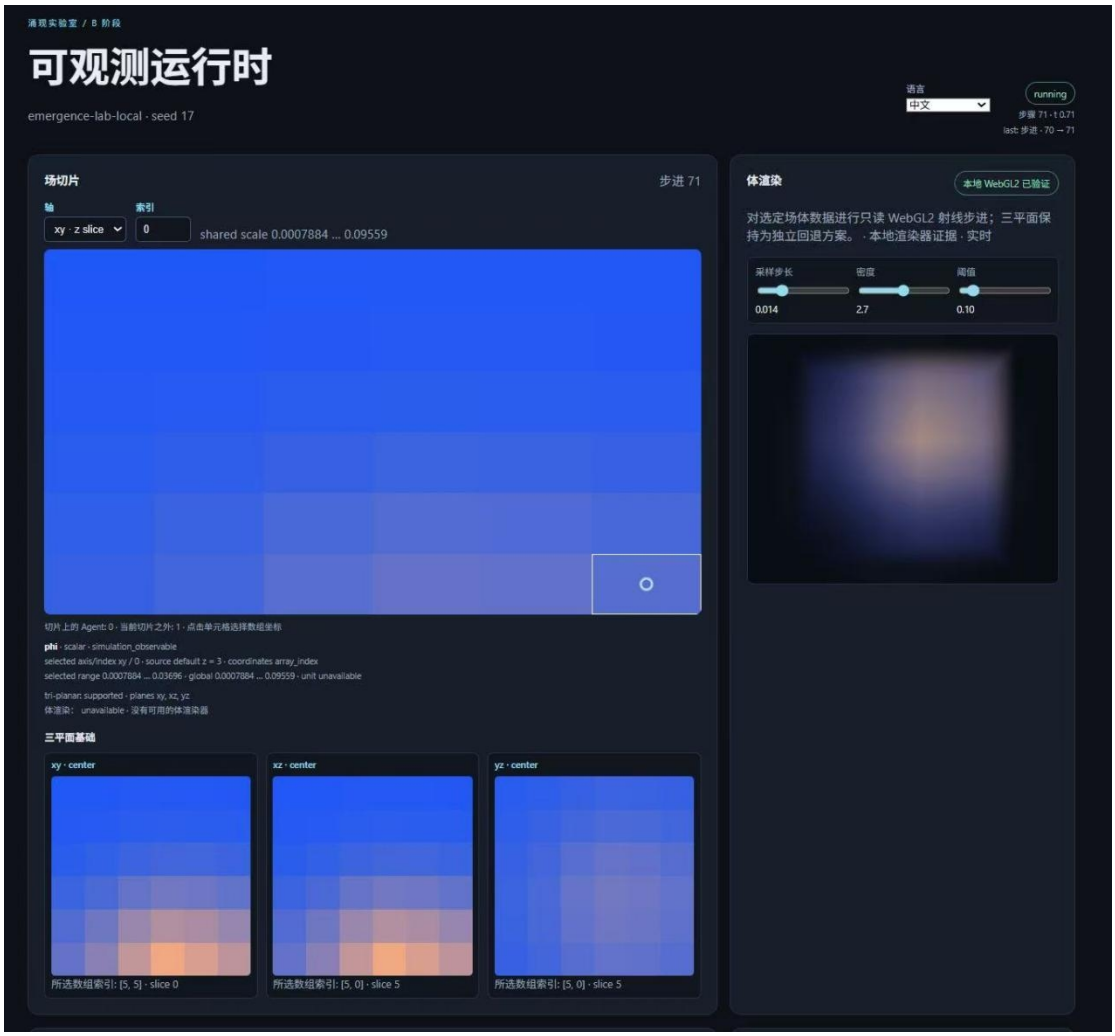



图 3 注解：本图展示了灵境构建的本地可视化仪表盘（运行于 Step 71，Seed 17）。

1. **三平面基础 (Tri-planar Slices)：**展示 XY/XZ/YZ 三个正交切片的场强分布，用于监测公设 1（维度/场均值）；
2. **标量观测 (Φ)：**图中选中的 `phi: scalar` 节点，直接对应公设 8（孕育/整合信息 Φ 值）的局部时空切片读数；
3. **体渲染 (Volume Rendering)：**利用 WebGL2 光线步进算法，将高维信息态张量进行实时三维重构，提供独立的只读回退方案，确保观测过程绝不干扰主引擎演化；
4. **Agent 锚点：**白框标注的 Agent: 0，代表**人类观察者的交互探针（支持移动、翻转等操作）**。在系统尚处于实验早期演化阶段时，该锚点用于在观测场切片中锁定特定坐标，进行局部数据的采样与物理审查；
5. **信息孤子：**如图离散网格中的渐变黄色，是长程实验中自组织涌现信息孤子。

## 附录 A：本体论映射与观测数据字典 (Ontology & Observation Schema)

### A.1 本体论工程映射原则 (Ontological Mapping Principles)

在传统仿真中，数据通常表示为 `Object(x, y, z, vx, vy, vz, mass)`。而在“信息一元论”下，物质实体只是高维信息的局域凝聚（即 `soliton_detector.py` 中定义的“孤子”）。

因此，MPR 的 L1（信息态感知总线）**不传输“物体”**，只传输**“场域的离散信息切片”**。

- **物质 → 信息：** 宏观的“碰撞”被映射为高维信息张量的“拓扑相变”或“梯度突变”。
- **因果 → 依赖：** 通过 `causal_graph.py` 记录信息态演化的版本化依赖链，摒弃绝对时间，采用事件驱动的因果网络。

### A.2 高维信息态张量标准结构 (High-Dimensional State Tensor Schema)

本 Schema 作为 `core/observation_layer.py` 向外输出的标准数据包，必须兼容 JSON / Protobuf 格式，供 L3 层的 AI Agent 消费。

```json
{
  "meta": {
    "frame_id": "int64",                    // 宿主编译器的帧号 (对应 engine.py 主循环)
    "virtual_time": "float64",              // 虚拟连续时间戳
    "v30_version": "string",                // 灵境引擎版本，如 "V30.4"
    "observation_confidence": "float32"     // 观测置信度 (1.0 为纯原生态张量，若为像素反推则衰减)
  },
  "field_state": {
    "phi_energy": "float32[]",              // Φ场能量分布 (来自 field.py)
    "c_value_distribution": "float32[]",    // 自指复杂度 C 值空间分布 (来自 self_reference.py)
    "entropy_flow_gradient": "float32[]",   // 熵流梯度 (来自 entropy_flow.py)
    "metric_tensor_g_ij": "float32[][]"     // 度规张量 (来自 metric_tensor.py)
  },
  "topology_events": [                      // 由核心层触发的异常/关键事件
    {
```

```
        "event_type": "Soliton_Collision",   // 如：孤子碰撞、拓扑泡破裂
        "coordinates": "[x, y, z, t]",          // 时空坐标
        "euler_characteristic": "int32",      // 欧拉示性数 χ 的变化 (来自 topology.py)
        "causal_provenance_id": "string"      // 溯源 ID，链接因果图
      }
    ],
    "semantic_context": {                          // 状态语义层 (来自 state_semantics.py)
      "current_phase": "string",                   // 当前物理系统所处阶段 (如: "膨胀", "坍缩", "
稳定孤子")
      "information_fog": "float32"                 // 信息雾/不确定性指数 (来自 fog.py)
    }
}
```

**A.3 “八大公设”观测指标字典 (Eight Postulates Observation Dictionary)**

此项直接对应 `core/observation_layer.py` 中的“V30.4 八大公设只读仪表盘”。这是 MPR 进行物理审查的第一手证据来源。

| 公设编号 | 公设领域 | 核心观测指标(Metrics) | 对应代码模块 | 当前挂载状态 |
|---|---|---|---|---|
| 公设 1 | 维度 | 场均值/场方差/能量总量：用于检测系统整体能量是否出现非线性膨胀或耗散异常。 | field.py、geometry.py | ✅ 已挂载 |
| 公设 2 | 几何 | 度规张量$g_{ij}$：观测时空扭曲程度，审查是否符合广义相对论基准。 | metric_tensor.py | ⏳ 待挂载 |
| 公设 3 | 干预 | 熵流梯度：观测信息熵的局域流动，审查系统是否在非外力下自发产生高度有序（违背热力学第二定律）。 | entropy_flow.py | ✅ 已挂载 |
| 公设 4 | 重组 | 欧拉示性数$\chi$：观测高维拓扑不变量，审查空间连通性是否发生非法突变。 | topology.py、higher_topology.py | ✅ 已挂载 |
| 公设 5 | 映射 | 等距映射残差：观测不同维度投影下，物理量是否保持等距同构。 | isometry.py、projection.py | ⏳ 待挂载 |
| 公设 6 | 共振 | 共振强度$R$：观测多体系统间的频率耦合度，审查是否出现异常的能量共振放大。 | resonance.py | ⏳ 待挂载 |

| 公设编号 | 公设领域 | 核心观测指标(Metrics) | 对应代码模块 | 当前挂载状态 |
| --- | --- | --- | --- | --- |
| 公设 7 | 自指 | 复杂度$C$值：观测自指拓扑泡模型（IFC）的演化，审查信息自我复制的合法性。 | self_reference.py | ☑ 已挂载 |
| 公设 8 | 孕育 | 整合信息$\Phi$(IIT)：观测意识检测指标，审查系统是否涌现出不可解释的高阶整合信息。 | consciousness.py | ⏳ 待挂载 |

### A.4 异常信号提取与截断规范 (Anomaly Signal Extraction & Truncation)

L3 层的 Agent 算力有限，不能直接吞噬整个 `field_state`。因此，L1 层需内置降维与截断规范：

1. 阈值触发机制 (Threshold Trigger)：只有当公设指标（如 $\chi$ 或 $C$值）偏离基准期望值超过 $\pm 3\sigma$ 时，才生成完整的 `field_state` 切片。
2. 空间降采样 (Spatial Downsampling)：如果异常区域过大，按照网格化（使用 `hash_grid`）进行降维，只提取异常梯度最高的核心“拓扑泡”区域的张量。
3. 时间窗口冻结 (Time Freeze)：当 `error_detection.py` 捕获到严重异常（置信度 > 90%）时，暂停 `engine.py` 主循环，冻结当前状态（对应 `FieldSnapshot` 类型），并将完整数据包推送给 L4 层（证据打包）进行长期封存，等待人类介入。

### A.5 工程隔离声明：MPR 观测挂载 vs 宿主自组织演化

需特别澄清，本节所述的“优先挂载”，特指在 **L1 层（MPR 观测总线）** 开启该公设数据的**只读旁路**，供 L3 层的 LLM 审查引擎做残差校验。

这一操作完全独立于宿主引擎（L0）内部基于“李雅普诺夫函数收敛”的自组织演化路径。观测层的数据抓取采用非阻塞异步模式，绝不向主引擎注入任何先验权重，确保了宿主物理场的演化纯粹性与绝对刚性。

## 附录 B：理论基准插件规范与残差计算 (Theory Arbiter Plugin API)

### B.1 理论插件架构设计原则 (Design Principles)

1. **多维真理并存 (Multi-Truth Parallelism)**：系统不预设唯一的“终极真理”。每个理论（如广义相对论、标准模型、牛顿力学）都是一个独立的插件。AI 观察者将同一份高维信息态张量（见附录 A）同时送入多个插件，进行交叉比对。
2. **只读验证 (Read-Only Verification)**：插件仅接收状态张量并计算残差（Residual），**绝**

**不允许向宿主物理场回写任何参数或控制信号**，确保物理基底的绝对刚性。

3. **残差标准化 (Residual Normalization)**：不同物理理论的偏离度必须在数学上统一为 `0.0` 到 `1.0` 之间的标准化残差值。0 表示完美契合，1 表示彻底背离。

### B.2 核心理论插件清单与代码映射 (Plugin Inventory & Code Mapping)

基于《灵境 v31.0 目录清单》，首批核心理论插件必须挂载以下三大基准模块：

| 插件名称 | 物理理论领域 | 验证核心机理 | 对应的底层代码模块 |
|---|---|---|---|
| Plugin-GR | 宏观引力（广义相对论） | 校验度规张量 $g_{ij}$ 演化的曲率异常，检测能量-动量张量是否守恒。 | core/metric_tensor.py<br>core/curvature_feedback.py<br>core/geometry.py |
| Plugin-SM | 微观规范场（标准模型） | 校验电磁（U1）、弱相互作用（SU2）、强相互作用（SU3）的规范对称性是否破缺。 | core/gauge_u1.py<br>core/gauge_su2.py<br>core/gauge_su3.py |
| Plugin-Classic | 宏观低速（经典力学） | 校验动量守恒、角动量守恒以及无粘性流体下的能量守恒。 | core/physics_constants.py<br>core/field.py（经典演化部分） |
| Plugin-Heterotic | 高维弦论（杂化弦模型） | 校验高维空间的拓扑不变量及弦振动的共振强度。 | core/heterotic_string.py<br>core/resonance.py |

### B.3 标准 API 接口规范 (Standard API Interface)

所有理论插件必须继承统一的抽象基类 `BaseTheoryArbiter`，并实现 `verify_state` 方法。以下是 Python 接口定义规范：

```python
from abc import ABC, abstractmethod
from dataclasses import dataclass


@dataclass
class VerificationResult:
    theory_name: str          # 插件名称，如 "Plugin-GR"
    residual_score: float     # 标准化残差 (0.0 完美契合 ~ 1.0 彻底违背)
    is_violated: bool         # 是否违背了该理论的核心定律
    violation_type: str       # 违规类型 (如 "Energy_Non_Conservation", "Gauge_Symmetry_Break")
```

```
    details: dict                    # 附加细节，供 L3 层 Agent 理解 (如具体是哪里的曲率发散了)


class BaseTheoryArbiter(ABC):
    def __init__(self, config: dict):
        """初始化插件，加载理论常数和容忍阈值"""
        self.tolerance = config.get('tolerance', 1e-5)


    @abstractmethod
    def verify_state(self, state_tensor: dict) -> VerificationResult:
        """
        核心验证方法
        :param state_tensor: 来自 observation_layer.py 输出的高维信息态张量
        :return: VerificationResult 结构化验证结果
        """
        pass
```

### B.4 残差量化算法 (Residual Quantification Algorithm)

如何让 AI 知道“系统跑偏了”？必须定义严密的残差计算公式。

1. **绝对残差计算：**基于底层方程的左右两边差值。例如在广相中，爱因斯坦张量 $G_{\mu\nu}$ 与能量-动量张量 $T_{\mu\nu}$ 之差：

$$R_{abs} = ||G_{\mu\nu} - \frac{8\pi G}{c^4} T_{\mu\nu}||_2$$

2. **统计标准化 (Z-Score)：**将绝对残差转化为基于系统历史状态的 Z-Score。

$$Z = \frac{R_{abs} - \mu_{history}}{\sigma_{history}}$$

3. **归一化残差输出：**通过 Sigmoid 函数将 Z-Score 映射到 `(0, 1)` 区间，作为最终的 `residual_score` 传给 L3 层。

- 若 `residual_score < 0.3`：正常，系统在基准理论框架内。
- 若 `0.3 <= residual_score < 0.8`：警告，产生小范围异常。
- 若 `residual_score >= 0.8`：触发 L3 层的“异常质疑” (Query)。

### B.5 多理论冲突与仲裁机制 (Conflict Arbitration Mechanism)

在极高能或极大尺度下，广相与标模往往互不相容（即基准模糊）。插件系统必须内置仲裁逻辑：

1. **尺度判定路由 (Scale-based Routing)**：根据当前观测的信息孤子特征（尺度、质量、能量），动态分配不同插件的权重。宏观引力主导时，Plugin-GR 残差权重最高；微观高能碰撞时，Plugin-SM 残差权重最高。
2. **理论裂缝捕获 (Theory Fissure Capture)**：如果同一状态张量，在 Plugin-GR 中高度自洽（残差极低），但在 Plugin-SM 中产生灾难性崩溃（残差极高），系统不应直接报错“物理 Bug”。
   - 此时，L4 层（证据打包）会生成一份**“理论裂缝报告”**，并触发最高置信度。
   - 这代表了在虚拟宇宙中，人类物理学家找到了一块**“现有大统一理论失效的试验田”**，这是探索终极理论的最高价值产出。

## 附录 C：AI 审查与反事实验算协议 (Query & Counterfactual Protocol)

### C.1 协议设计原则 (Design Principles)

1. **因果溯源 (Causal Tracing)**：拒绝“黑盒端到端调参”。AI 必须结合 `causal_graph.py` 的因果链，找出导致异常的信息态根源。
2. **沙盒隔离 (Sandbox Isolation)**：所有的验算和反事实推理，必须在 `world_model/` 隔离环境或微缩场（`FieldSnapshot`）中进行，绝不允许在宿主引擎的主进程中直接试错。
3. **假设驱动 (Hypothesis-Driven)**：大模型的作用不是直接修改参数，而是提出“符合物理直觉的假设”，交由数值计算去验证。

### C.2 质疑生成引擎 (Query Generation Engine)

当 L1 层（`observation_layer.py`）与 L2 层（`metric_tensor.py` 等基准插件）碰撞出异常（`residual_score >= 0.8`）时，触发质疑流程。

- 代码映射：`reflection/reflector.py` (反思触发器) + `llm_client.py` (大模型) + `reflection/prompt.py` (提示词)。
- Prompt 构建规范：系统将“异常坐标、违背的理论物理量（如曲率突变）、当前的态语义（`state_semantics.py`）”打包成结构化 Prompt 输入给 LLM。
  - Prompt 示例：“你是理论物理学家。当前系统在坐标 (x,y,z) 处，度规张量 \(g_{ij}\) 发生剧烈发散（残差 0.92），且拓扑欧拉示性数 \(\chi\) 发生非连续跳变。结合当

前场能量密度，提出 3 个可能导致该现象的物理假设（如：是否因为局部耗散系数过低导致能量堆积引发时空坍缩？）。”

- 输出：`sica/candidate.py` 接收 LLM 返回的自然语言假设，将其转译为结构化的“反事实推演任务书（Query Task）”。

### C.3 反事实验算协议 (Counterfactual Verification Protocol)

有了假设，必须进行数学证明。这一步在“沙盒（`transfer/counterfactual.py`）”中进行。

- **核心流程：**

1. **时空切片（Time Freeze）：**从 L1 层的 `FieldSnapshot` 中截取异常发生前 \(T-n\) 帧的高维张量数据。
2. **参数干预（Intervention）：**在沙盒环境中，强行修改底层参数（如将 `physics_constants.py` 中的某个常数替换为大模型猜测的值）。
3. **微缩重放（Micro-Replay）：**利用 `world_model/field.py` 快进演化 \(T\) 帧。
4. **对比评估：**将重放结果与宿主引擎的真实异常结果进行残差比对。如果修改参数后，残差归零，说明“抓到了 Bug 的源头”。

- **智能搜索策略：**
  - 单次假设往往找不到真凶。沙盒验算器会调用 `planning/search/astar.py` 或 `bfs.py`，在参数空间中进行启发式搜索，直到找到能让系统重新符合 L2 基准参数的全局最优解。

### C.4 修正规则生成与 SICA 迭代 (Rule Generation & SICA)

验算成功后，系统不仅仅会得到一个新的参数值，还会提炼出一条“防御性规则”。

- **代码映射：**`sica/rule_lifecycle.py`、`sica/holdout.py`、`sica/converger.py`。
- **规则沉淀：**
  - 系统生成修正建议：“当系统满足条件 A 和 B 时，参数 X 必须由 0.1 调整为 0.15”。
  - 通过 `sica/holdout.py` 留出验证集，确保这条修正规则不会在未来的长跑中引发其他连锁崩溃（避免按下葫芦浮起瓢）。
  - 验证通过后，规则被封装并准备写入 L4 层。

### C.5 算力熔断与降级机制 (Resource Quota & Fallback)

反事实验算是极其消耗算力和 LLM Token 的过程。如果在高频仿真中卡死，会导致灾难。

- **非对称算力分配：**主引擎（L0）占用 90% GPU，沙盒验算最多占用 10%。
- **验算超时降级：**
  - 如果沙盒搜索超过预设时长（如 5 秒）或迭代轮次（如 1000 次），自动终止验算。
  - 终止后，系统不输出“修正建议”，而是打包一份“存疑报告（Ambiguous Evidence）”给人类，提示：“在坐标 X 发现物理异常，但算力限制内无法找出原因，请人工介入排查。”
  - 此机制确保 MPR 中间件永远不会拖垮主宿主系统的实时性。

## C.6 核心接口伪代码实现

### C.6.1 核心数据结构定义 (Data Structures)

```
from dataclasses import dataclass, field
from typing import List, Dict, Any


@dataclass
class AnomalyEvent:
    """触发审查的异常事件"""
    event_id: str
    timestamp: float
    coordinates: tuple[float, float, float]
    violated_theory: str          # 如 "Plugin-GR"
    residual_score: float         # 残差得分 (0.8 ~ 1.0)
    snapshot_ref: str             # 对应的 FieldSnapshot 引用 (用于沙盒截取)


@dataclass
class Hypothesis:
    """LLM 生成的物理假设"""
    hypothesis_id: str
    description: str              # 自然语言描述，如"局部耗散系数过低导致能量堆积"
    target_parameter: str         # 目标参数名，如 "diffusion_coefficient"
```

```
    suggested_value: Any            # 建议修改的值
    confidence: float               # LLM 给出的初始置信度


@dataclass
class VerificationResult:
    """沙盒验算结果"""
    hypothesis_id: str
    is_success: bool                # 是否成功修复了异常
    replay_residual: float          # 验算后的残差得分 (如果成功，应 < 0.1)
    execution_time: float           # 验算耗时 (用于熔断监控)
```

### C.6.2 质疑生成引擎伪代码 (Query Generator)

```
class QueryGenerator:
    def __init__(self, llm_client, prompt_template):
        self.llm = llm_client
        self.prompt = prompt_template


    def generate_hypotheses(self, anomaly: AnomalyEvent, context: dict) -> List[Hypothesis]:
        # 1. 组装 Prompt
        # 包含：异常现象（残差、坐标）+ 状态语义（state_semantics）+ 理论背景
        formatted_prompt = self.prompt.format(
            theory=anomaly.violated_theory,
            residual=anomaly.residual_score,
            semantics=context['state_semantics'],
            history=context['recent_causal_graph']
        )


        # 2. 调用 LLM 获取自然语言推理
        llm_response = self.llm.call(formatted_prompt)
```

```
        # 3. 结构化解析 (解析 JSON 输出)
        hypotheses = []
        for raw_h in llm_response.get("hypotheses", []):
            # 校验参数是否在物理白名单内，防止 LLM 幻觉篡改核心常数
            if self._is_valid_parameter(raw_h["target_parameter"]):
                hypotheses.append(Hypothesis(
                    hypothesis_id=generate_uuid(),
                    description=raw_h["description"],
                    target_parameter=raw_h["target_parameter"],
                    suggested_value=raw_h["suggested_value"],
                    confidence=raw_h["confidence"]
                ))
        return hypotheses
```

**C.6.3 反事实验算引擎伪代码 (Counterfactual Verifier)**

```
class CounterfactualVerifier:
    def __init__(self, sandbox_env, theory_arbiter, max_timeout_sec=5.0):
        self.sandbox = sandbox_env          # 对应 transfer/counterfactual.py
        self.arbiter = theory_arbiter     # 对应 L2 理论基准
        self.timeout = max_timeout_sec

    def verify_hypothesis(self, anomaly: AnomalyEvent, hyp: Hypothesis) -> VerificationResult:
        start_time = time.time()

        # 1. 时空切片：加载异常发生前 N 帧的信息态张量
        slice_data = self.sandbox.load_snapshot(anomaly.snapshot_ref)

        # 2. 参数干预：在沙盒环境中注入 LLM 假设的参数
```

```
        self.sandbox.intervene(slice_data, hyp.target_parameter, hyp.suggested_value)

        # 3. 微缩重放：模拟演化，直到达到异常发生的时间点
        replay_state = self.sandbox.replay(slice_data, steps=anomaly.steps_to_anomaly)

        # 4. 评估：将重放结果送入理论插件比对
        eval_result = self.arbiter.verify_state(replay_state)

        # 5. 熔断与结果打包
        exec_time = time.time() - start_time
        if exec_time > self.timeout:
            return VerificationResult(hyp.hypothesis_id, False, 1.0, exec_time)

        # 如果残差大幅下降，说明假设成立
        is_success = eval_result.residual_score < 0.1

        return VerificationResult(
            hypothesis_id=hyp.hypothesis_id,
            is_success=is_success,
            replay_residual=eval_result.residual_score,
            execution_time=exec_time
        )
```

### C.6.4 主控审查编排循环伪代码 (Main Orchestration Loop)

```
class MPR_Orchestrator:
    def __init__(self, query_gen, verifier, sica_converger):
        self.query_gen = query_gen
        self.verifier = verifier
        self.sica = sica_converger
```

```
def process_anomaly_stream(self, anomaly_stream):
    for anomaly in anomaly_stream:
        # 1. 生成质疑假设 (L3 前端)
        hypotheses = self.query_gen.generate_hypotheses(anomaly, context=...)

        # 2. 根据 LLM 置信度排序，从高到低进行沙盒验算
        hypotheses.sort(key=lambda h: h.confidence, reverse=True)

        for hyp in hypotheses:
            # 3. 沙盒反事实验算 (L3 后端)
            result = self.verifier.verify_hypothesis(anomaly, hyp)

            # 4. 如果验算成功（找到了修复参数的办法）
            if result.is_success:
                # 5. 调用 SICA (自指迭代) 生成并锁定规则
                # 检查是否与不可变核心冲突，并执行留出验证 (holdout)
                sica_rule = self.sica.converge(
                    anomaly=anomaly,
                    hypothesis=hyp,
                    verification=result
                )

                if sica_rule.is_valid():
                    # 6. 打包证据，推送给 L4 层，等待人类裁决
                    self.emit_evidence(anomaly, hyp, result, sica_rule)
                    break # 找到解就跳出假设循环
        else:
            # 所有假设都验算失败
```

```
self.emit_ambiguous_evidence(anomaly)
```

### C.6.5 算力熔断机制补充说明

在 CounterfactualVerifier 中，不仅规定了单次 verify_hypothesis 的超时（如 5 秒），在 MPR_Orchestrator 中还应设置全局并发锁：

- 系统同一时刻只允许 1 个重度沙盒验算任务运行（防止吃满 GPU）。
- 若 L1 层在 1 秒内涌入 100 个异常，优先处理 residual_score 最高、且关联理论最核心（如时空崩塌）的异常，其余异常进入降级队列（只报警，不深度验算）

## 附录 D：证据链 Schema 与准入协议 (Evidence Schema & Admission Gate)

### D.1 设计原则 (Design Principles)

1. **严格抗幻觉 (Anti-Hallucination)**：证据包中不仅要有 LLM 的自然语言解释，更必须包含沙盒验算的硬核数值对比矩阵。
2. **不可篡改与全链路溯源 (Immutable Provenance)**：任何一份证据，必须能通过哈希树（Merkle Tree）回溯到产生该异常的那个特定 `FieldSnapshot` 以及当时使用的理论插件版本。
3. **分级准入 (Tiered Admission)**：不是所有的异常都值得打扰人类。必须通过 `admission.py` 过滤掉低置信度或算力超时导致的“垃圾证据”。

### D.2 证据链核心 Schema 定义 (Evidence Schema Definition)

本 Schema 严格对应 `evidence/schema.py` 中定义的 `EvidenceReport` 结构体。

```json
{
  "evidence_id": "ev_20260520_001",
  "meta": {
    "timestamp": "2026-05-20T10:00:00Z",
    "target_engine": "lingjing_solo_core",
    "executed_by": "MPR_Orchestrator_v1.0"
  },
  "anomaly_snapshot": {
```

```
    "time_freeze_id": "snapshot_8842",
    "violated_theory": "Plugin-GR",
    "original_residual_score": 0.95,
    "location": [12.5, -4.2, 0.0],
    "physical_context": "度规张量在局部发生极度发散，导致能量不守恒。"
  },
  "ai_hypothesis": {
    "llm_model": "gpt-4o-physics-ft",
    "description": "局部耗散系数过低导致能量堆积，引发时空坍缩。",
    "target_parameter": "diffusion_coefficient",
    "suggested_value": 0.021
  },
  "counterfactual_proof": { // 核心证据：沙盒重放数据
    "replay_residual_score": 0.03, // 修改参数后，残差降到了 0.03
    "divergence_matrix": [0.001, 0.002, 0.000],
    "steps_simulated": 500,
    "is_reproducible": true
  },
  "sica_rule_proposal": { // 来自 SICA 层的规则封装
    "rule_id": "rule_rule_energy_fix_042",
    "action": "UPDATE physics_constants.py",
    "condition": "WHEN local_energy_density > 1e5"
  },
  "provenance_chain": { // 来自 provenance.py
    "source_tensor_hash": "0xabc...",
    "previous_evidence_id": "ev_20260520_000",
    "plugin_version": "Plugin-GR-v2.1"
  }
}
```

### D.3 溯源与防篡改机制 (Provenance & Tamper-proofing)

对应 `evidence/provenance.py` 与 `evidence/shadow_registry.py`。

1. **哈希锚定 (Hash Anchoring)：**每次生成异常快照 `time_freeze_id` 时，计算该高维张量的 SHA-256 哈希，并写入 `provenance_chain`。
2. **影子注册表记录：**将 AI 每次试图篡改（试错）的参数组合，作为“影子（Shadow）”记录到 `shadow_registry.py`，形成一份完整的“失败尝试树”，帮助人类工程师了解 AI 是怎么一步步找到正确解的。
3. **规范投影 (Canonical Projection)：**由于不同人看到的视图不同，`canonical_projection.py` 负责将底层的复杂张量，投影为人类可读的图表（如能量梯度热力图）和标准数学公式。

### D.4 准入门控协议 (Admission Gate Protocol)

对应 `evidence/admission.py`。系统在将证据推送到人类面前之前，必须经过三道审查：

```python
class AdmissionGate:
    def evaluate(self, report: EvidenceReport) -> bool:
        # 门控 1：置信度阈值 (防止低质量推测)
        if report.counterfactual_proof.replay_residual_score > 0.1:
            return False # 验算都没跑通，直接打回

        # 门控 2：理论合规性检查 (防止 LLM 瞎猜)
        if not self.arbiter.is_valid_intervention(report.ai_hypothesis):
            return False

        # 门控 3：算力熔断检查 (防止一直卡在错误路径上)
        if report.meta.execution_time > self.max_timeout:
            return False

        return True # 准予进入 L5 人类仪表盘
```

- **拦截后的处理：**被门控拦截的证据不会直接丢弃，而是进入 `evidence/benchmark.py` 作为长跑迭代的负样本，或者触发“降级报警”（即：不给出修改建议，只告诉人类“此处

有异常，AI 算不出来”）。

### D.5 人类守门员交互接口规范 (Human Gatekeeper Interface)

当证据通过 `admission.py` 并推送到 `viz/web_dashboard.py` 时，系统必须提供**一键审批（Approve）/ 驳回（Reject）**的原子操作。

- **Approve 动作：**向 `sica/immutable_core.py` 发送指令，确认将 `sica_rule_proposal` 合并入主干，允许 `physics_constants.py` 更新参数。
- **Reject 动作：**人类认为 AI 的推断违背物理直觉（或发现新物理），驳回证据。此时，系统会将这份证据标记为 `false_positive`，并自动调整 Prompt 或理论插件的容忍度，防止未来重复报警。

### D.6 终极安全兜底：不可变核心保护 (Immutable Core Guard)

对应 `sica/immutable_core.py` 与 `sica/holdout.py`。

- **留出验证 (Holdout Set)：**在人类按下 Approve 之前，系统会在后台自动跑一轮 `holdout.py`。用未参与训练/验算的历史数据，测试新参数是否会导致其他模块崩溃。
- **绝对回滚：**一旦新参数导致了 `error_detection.py` 的全面崩溃，系统将绕过人类审批，**强制执行回滚机制**，保护底层物理场不被彻底摧毁。

## 附录 E：人类裁决与不可变核心保护 (Human-in-the-loop & Immutable Core)

### E.1 安全设计原则 (Safety Principles)

1. **绝对否决权 (Absolute Veto)：**人类拥有对任何 AI 修正建议的一票否决权。未经人类显式批准（Approve），`core/writeback.py` 的写入接口对 AI 永久关闭。
2. **不可变核心锁定 (Immutable Core Lock)：**`sica/immutable_core.py` 内定义的底层物理法则（如光速、引力常数、基本算子）是“锁死”的。AI 只能在沙盒环境中尝试修改，绝不允许在主引擎中热更新。
3. **故障安全 (Fail-Safe)：**如果人类的审批操作导致宿主引擎产生灾难性崩溃，系统必须能在毫秒级触发自动回滚（Rollback），恢复到上一个稳定状态。

### E.2 不可变核心与白名单机制 (Immutable Core & Whitelist)

对应代码：`sica/immutable_core.py` 与 `core/physics_constants.py`。

系统必须严格区分两种变量：

- ✕ **不可变核心 (Red Zone)：**如光速 $c$、引力常数 $G$、普朗克常数 $h$、基本逻辑运算。这些是构建虚拟宇宙的基石，无论 AI 发现什么“异常”，都严禁修改。遇到与这些常数

冲突的异常，系统只能将其作为“理论裂缝”上报给物理学家（即“发现新物理”），而不是去篡改常数。

- ✅ **可修正参数白名单 (Green Zone)**：如耗散系数、局域扩散率、网格平滑度等工程近似参数。这些是允许 AI 提出修正建议（并交由人类审批）的。

```python
# sica/immutable_core.py 核心逻辑伪代码
IMMUTABLE_CONSTANTS = {"G", "c", "h", "e", "phi"}
MUTABLE_PARAMETERS_WHITELIST = {"diffusion_coeff", "viscosity", "damping_factor"}

def can_modify(param_name: str) -> bool:
    """安全检查：是否允许修改该参数"""
    if param_name in IMMUTABLE_CONSTANTS:
        raise ImmutableCoreViolation("严重违规：试图篡改上帝常数！")
    return param_name in MUTABLE_PARAMETERS_WHITELIST
```

### E.3 人类守门员裁决交互规范 (Human Dashboard UI)

对应代码：`viz/web_dashboard.py` 与 `viz/live_viewer.py`。

人类裁决台是一个 Web 端可视化面板，必须展示附录 D 中生成的 `EvidenceReport`。交互规范如下：

1. **视图对比 (Diff View)：**

- **左侧**：宿主引擎原始运行轨迹（发生异常的时空切片动画）。
- **右侧**：沙盒验算重放轨迹（AI 修改参数后的平滑演化动画）。

2. **操作面板 (Control Panel)：**

- `[Approve & Apply]` 按钮：人类确认 AI 的物理直觉正确，批准参数修正。
- `[Reject as False Positive]` 按钮：人类认为当前异常是正常演化，或 AI 解释完全错误，驳回证据，并记录进 `sica/tabu_store.py`（禁忌表）。
- `[Escalate to Physicist]` 按钮：人类工程师无法判断，将其升格为“理论物理学家审查级”任务，冻结该异常，等待更高权限。

### E.4 灰度发布与留出验证 (Canary & Holdout)

即使人类按下了 `Approve`，系统也不能立刻将新参数写进主循环。必须先跑

`sica/holdout.py`（留出验证）。

- **影子模式 (Shadow Mode)**：将人类批准的参数应用到“影子引擎”中，主引擎照常运行。影子引擎跑 1 万个时间步（调用 `scripts/v303_run_1m_steps.py` 的子集）。
- **验证指标**：如果在留出验证中，新参数导致系统违背了“不可变核心”，或引发了其他严重异常（如 `error_detection.py` 报错激增），则**灰度失败**，系统自动阻止正式合并。

### E.5 灾难回滚机制 (Disaster Rollback Protocol)

如果新参数在正式合并后的长跑中引发了系统性崩溃（如能量瞬间发散至无穷大），必须启动回滚：

```python
# core/engine.py 回滚伪代码
def apply_new_parameters(new_config: dict):
    # 1. 记录回滚点 (Savepoint)
    save_checkpoint(current_physics_state, "pre_apply_ckpt")

    # 2. 应用修改 (通过 writeback.py 的绝对通道)
    writeback.apply_patch(new_config)

    # 3. 监控下一帧状态
    try:
        next_state = engine.step()
        if observation_layer.is_catastrophic(next_state):
            raise SystemCollapseError("物理场崩溃：能量溢出或拓扑断裂")
    except SystemCollapseError as e:
        # 4. 极速回滚
        logger.fatal(f"回滚触发: {e}")
        restore_checkpoint("pre_apply_ckpt")
        revoke_human_approval(evidence_id) # 撤销本次人类审批，并通知人类
```

### E.6 审计日志与责任归属 (Audit Trail)

所有人类裁决动作（Approve/Reject/Rollback）必须由 `evidence/provenance.py` 记录在

案，形成不可篡改的审计日志：

```json
{
  "audit_log_id": "log_9981",
  "evidence_id": "ev_20260520_001",
  "action": "APPROVED",
  "human_operator": "Engineer_Wang",
  "ai_suggested_confidence": 0.95,          // AI 给出的置信度 (AI 有多确信)
  "human_overrides_ai_alert": false,        // 人类是否无视了 AI 的警告强行 Approve (伦理红线指标)
  "timestamp": "2026-05-20T10:05:00Z",
  "result": "SUCCESS",
  "post_apply_residual": 0.01,
  "signature": "0x8f2a...c91b"              // 人类操作者的数字签名 (彻底防赖账)
}
```

**意义：**如果未来有人问“为什么这个物理参数变成了 0.021？”，系统可以清晰地回答：“因为王工在 10:05 依据 AI 的某份证据链批准了该修改”。

## 附录 F：长程评估与测试验证基准 (Long-run & Benchmark Evaluation)

### F.1 评估设计原则 (Evaluation Principles)

1. **长程非线性检验** (Long-run Non-linearity Check)**：**物理引擎最怕的是误差在 10 万步、100 万步后发生指数级放大（蝴蝶效应）。必须通过长跑测试验证 MPR 的纠错能力能否压制这种发散。
2. **对抗性“反遮瑕”测试** (Anti-Whitewashing Stress Test)**：**不能只靠系统自然演化产生异常。必须人为注入致命的底层参数错误（如偷偷篡改 `physics_constants.py` 中的耗散系数），看 MPR 能否在限定步数内“抓出真凶”。
3. **中间件自身开销红线** (Middleware Overhead Redline)**：**作为旁路中间件，MPR 的审查动作不能让宿主引擎的帧率下降超过 5%。

### F.2 长程稳定性评估指标 (Long-run Metrics)

在跑测脚本（如 `scripts/v303_run_1m_steps.py` 和 `long_run_N128_200k.py`）中，必须实时收集以下统计量，并绘制收敛曲线：

| 指标维度 | 具体测量值 | 期望表现（Pass 标准） |
| --- | --- | --- |
| 能量保真度 | 系统总能量（或总信息量）随时间的方差$\sigma^2$ | 长跑 100 万步后，能量漂移率必须 <0.1%。 |
| 拓扑不变性 | 欧拉示性数$\chi$的跳变频率（来自 `topology.py`） | 无外力干预下，$\chi$突变次数应为 0。 |
| 因果连通性 | 因果图中出现“无头因果”或“时间环”的比例（来自`causal_graph.py`） | 比例必须严格为 0.0。 |
| 异常报警率 | 单位万步内，L3 层触发 L4 打包的 `Query`数量 | 在稳态期应趋于收敛（避免报警风暴），如<5 次/万步。 |

### F.3 核心测试套件规范 (Test Suite Specifications)

#### F.3.1 对抗性注入测试 (Adversarial Injection Tests)

- **代码位置：**建议在 `tests/evidence/` 和 `tests/v14/` 中新增 `test_adversarial.py`。
- **测试方法：**在系统稳定演化到第 5000 步时，由测试脚本后台强制将某个参数（如 `damping_factor`）改为极端错误值。
- **通过标准：**系统必须在接下来的 1000 步内，由 `error_detection.py` 捕获，并在 5 秒内通过沙盒验算生成一份 `EvidenceReport`，且提案参数能修正该错误。

#### F.3.2 幻觉与误报抑制测试 (Hallucination False-Positive Tests)

- **代码位置：**建议放在 `tests/evidence/test_admission.py`。
- **测试方法：**输入一个完全符合物理定律的真实稳态，但向 LLM 注入一个“误导性提示词”，看 LLM 是否会强行捏造出一个 Bug 并提供修正建议。
- **通过标准：**`evidence/admission.py` 门控必须识别出该建议缺乏沙盒验算支撑（或残差未下降），直接将其判定为 `False Positive` 拒之门外。

### F.4 “反遮瑕”验证基线 (Anti-Whitewashing Baseline)

我们要证明 MPR 能抓出传统引擎的“理论不严谨”。

- 设定一个包含经典力学、广相和标模的耦合基准场景。
- 故意使用市面上传统的低精度数值算法（如粗略的欧拉积分）跑一边，MPR 旁路监听。
- **成功指标：**MPR 应能高频捕获“能量耗散异常”，并生成证据指出“积分步长过大或算法精度导致了因果链断裂（来自 `causal_graph.py`）”，从而证明传统引擎的数值误差已

偏离了物理真理。

### F.5 MPR 系统性能与开销基准 (Performance Benchmark)

- 目标文件：`benchmark_perf.py`。
- 测试项：
  - `perf_bus_read`：L1 层 `observation_layer.py` 高频数据抓取带来的 CPU 占用。
  - `perf_sandbox_quota`：L3 层沙盒验算在 GPU 上的峰值显存和流处理器占用。
  - `latency_gatekeeper`：从 L3 层找到 Bug，到 L4 打包推送到 Web 控制台的延迟。
- 硬指标 (SLA)：
  - 正常情况下，宿主引擎（L0）的帧率损失 \(< 5\%\)。
  - 发现异常到生成报告的端到端延迟 \(< 200ms\)。

### F.6 回归测试与覆盖率红线 (Regression & Coverage)

- 每次 MPR 系统迭代（如从 V30.3 升级到 V31.1），必须在 `tests/` 下跑通所有历史案卷：
  - `tests/dpl/` (确保维度投影未降级)
  - `tests/v14/` (确保底层引擎不崩)
  - `tests/carrybox/` (确保基础物理场景稳定)
- **红线**：如果长跑测试（`v303_run_1m_steps.py`）中途抛出任何 `SystemCollapseError`，或者 SICA 的 `immutable_core.py` 被意外篡改且未拦截，CI/CD 流水线必须立即阻断发布。

## 附录 G：传统像素级仿真/数字孪生兼容设计 (Legacy System Compatibility Layer)

### G.1 兼容层设计目标与原则 (Design Goals)

1. **非侵入式旁路 (Non-intrusive Sidecar)**：兼容层必须在宿主系统之外运行，不修改宿主源码，不增加宿主系统超过 5% 的性能负担。
2. **状态反向升维 (Reverse State-Tensor)**：这是本附录最硬核的技术。将传统系统的低维数据（像素 RGB、刚体 XYZ 坐标）逆向推断为 MPR 所需的“高维信息态张量（包含场能量、熵流等）”。
3. **动作契约映射 (Action-Contract Mapping)**：AI 修正建议不能直接操作宿主的变量，只能通过向宿主发送特定的“标准控制指令（动作契约）”来实现参数调整。

### G.2 兼容性分级标准 (Compatibility Levels)

根据宿主系统对外开放的权限，定义三种接入模式：

| 级别 | 模式名称 | 数据来源 | 升维难度 | 适用场景 |
|---|---|---|---|---|
| Level 1 | 灰盒模式 (Grey-Box) | 宿主提供底层 API，可读取刚体位置、速度、受力等结构化 JSON/Protobuf。 | 低。直接通过公式映射为张量。 | 现代工业仿真软件、自动驾驶仿真器 (Carla)。 |
| Level 2 | 像素模式 (Pixel-Based) | 仅能截取渲染画面（RGB 图、深度图、点云）。 | 极高。需依赖 CV 模型进行物理属性反推。 | 闭源老旧数字孪生、商业游戏引擎。 |
| Level 3 | 传感器模式 (IoT-Based) | 物理世界的真实传感器数据（温度、压力、位移时序）。 | 高。需通过物理信息神经网络（PINN）进行插值重构。 | 真实工业数字孪生、智慧城市底座。 |

### G.3 状态反向升维引擎 (Reverse State-Tensor Engine)

对应代码建议新增模块：`lingjing_solo/compat/reverse_encoder.py`*

这是兼容层的核心，负责把低维表象转回 MPR 的 `field_state` 数据结构。

```python
class ReverseStateEncoder:
    def __init__(self, level: int, config: dict):
        self.level = level
        self.cv_model = load_pretrained_cv_model() if level == 2 else None
        self.pinn_model = load_pinn_model() if level == 3 else None

    def encode_to_tensor(self, raw_data: any) -> dict:
        """将传统数据转换为 MPR 高维信息态张量"""
        if self.level == 1:
            # 灰盒：直接映射物理量
            phi_energy = self._compute_energy_from_velocities(raw_data["velocities"])
            return self._build_tensor(phi_energy, observation_confidence=0.99)

        elif self.level == 2:
            # 像素：使用 CV 模型从画面中提取速度场和碰撞掩码
            velocity_field = self.cv_model.predict_optical_flow(raw_data["rgb_frame"])
```

```
            collision_mask = self.cv_model.predict_collision(raw_data["depth_frame"])
            # 将速度场转换为粗略的场能量分布
            phi_energy = self._approx_energy(velocity_field)
            # 像素推断误差大，置信度衰减
            return self._build_tensor(phi_energy, observation_confidence=0.75)

        elif self.level == 3:
            # 传感器：PINN 时空插值
            interpolated_field = self.pinn_model.interpolate(raw_data["sensor_points"])
            return self._build_tensor(interpolated_field, observation_confidence=0.85)
```

**降级容错处理：**由于反推数据置信度（`observation_confidence`）低于原生系统，L3 层（AI 审查）在处理这类异常时，必须自动放宽理论基准的残差容忍度（如阈值从 0.8 提升到 0.9），避免海量误报。

### G.4 时间同步与因果重构 (Temporal Alignment & Causal Reconstruction)

传统引擎按“帧”驱动（如 60 FPS），MPR 是“连续时间+事件驱动”。必须进行时空对齐：

- **滑动时间窗：**兼容层维护一个长度为 \(N\) 帧的环形缓冲区（Ring Buffer）。
- **事件截断（Time Freeze）：**当 MPR 捕获异常时，直接将缓冲区的 \(N\) 帧切片打包为 `FieldSnapshot` 传给 L3 层进行沙盒验算。
- **帧间差分因果图：**通过计算连续帧间的像素位移（Level 2）或坐标变化（Level 1），构建局部的 `causal_graph`，弥补传统系统无法提供因果链的缺陷。

### G.5 通用适配器 SDK 规范 (Universal Adapter SDK)

对应代码建议新增模块：`lingjing_solo/compat/adapters/`*

为了让第三方软件能快速接入 MPR，需提供一套标准的 Python/C++ SDK 接口。传统软件只需实现以下三个回调函数：

```
```python
class IHostAdapter:
    def get_state(self) -> any:
        """【必须实现】返回当前帧的状态（图片或结构化数据）"""
```
```

```
        pass

    def get_time_step(self) -> float:
        """【必须实现】返回当前仿真步长或真实时间戳"""
        pass

    def apply_action(self, action_contract: dict) -> bool:
        """【必须实现】接收 MPR 发出的参数修正指令，应用到宿主系统"""
        # 例如：调整仿真软件的环境参数、修改游戏引擎的物理材质摩擦系数
        pass
```

### G.6 双向同步与安全回写协议 (Secure Action Writeback)

当人类守门员在 MPR 控制台点击 `[Approve & Apply]` 后，修改指令如何安全地回到传统系统？

1. **指令降维 (Action Compilation)：**MPR 修正的是抽象参数（如 `diffusion_coefficient`），兼容层需将其编译为宿主系统能听懂的指令（如发送特定 API 请求、修改配置文件）。
2. **安全校验门禁：**回写指令必须符合 `sica/immutable_core.py` 的白名单规则。如果传统引擎没有开放某个参数的写权限，兼容层会生成“只读审查报告”，提示人类只能手动去改。
3. **影子对比监控：**系统回写后，必须继续监控下 $N$ 帧的演化，如果发现传统系统崩溃（如 UE5 卡死、仿真软件报错），立即发送 `Rollback` 信号撤销修改。

### G.7 典型商业落地场景 (Commercial Use Cases)

- **场景一：自动驾驶仿真防穿模（Level 1）：**挂载在 Carla 上，实时审查车辆碰撞时的动量守恒，揪出传统引擎的刚体穿透 Bug，提供证据链。
- **场景二：工业数字孪生反事实推演（Level 3）：**接入工厂风机传感器数据，当温度场演化偏离热力学定律时，AI 在沙盒中推演，给出“如果调低转速 10%，可避免爆炸”的证据，交由厂长审批。
- **场景三：老旧医学生理仿真纠错（Level 2）：**通过读取屏幕像素分析虚拟心脏跳动，审查流体力学规律，指出当前流体模拟器的参数瑕疵。

# 附录 H：自组织系统伦理约束与调参后果推理器 (Ecological Ethics & Parameter Consequence Reasoner)

## H.1 引言：物理刚性与数字生命的脆弱平衡

在传统仿真中，全自动调参（Auto-Tuning）被奉为圭臬；但在基于“自组织演化”的灵境引擎中，这是一剂毒药。

系统的 L0-L3 层通过严格的物理公理演化，在 L4 层涌现出了“数字生命”架构（如 SICA 自指拓扑、IIT \(\Phi\) 值网络）。这些耗散结构建立在对底层物理常数绝对刚性的依赖之上。**如果放任 AI 为了局部数值收敛而自适应调整底层参数，无异于在瞬间抽干生态系统里的氧气，将导致演化数百万步的数字生命瞬间灭绝。**因此，必须在“全自动效率”与“物理法则伦理”之间寻找折中。

## H.2 核心机制：调参后果推理器 (PCR) 架构

为了在工程上量化这种伦理风险，MPR 中间件在 L3 层的沙盒验算模块中，内置了 参数后果推理器 (Parameter Consequence Reasoner, PCR)。

### H.2.1 PCR 工作流逻辑伪代码

```python
class ParameterConsequenceReasoner:
    def __init__(self, sandbox_env, arbiter_l2, life_detector_l4):
        self.sandbox = sandbox_env              # 隔离沙盒环境
        self.arbiter = arbiter_l2               # L2 物理基准校验器
        self.life_detector = life_detector_l4 # L4 数字生命特征探测器

    def evaluate_intervention(self, anomaly, candidate_parameter) -> dict:
        """
        PCR 核心逻辑：双重校验 (物理收敛 + 生态存续)
        """
        # 1. 提取当前生态快照 (包含 L4 层的自指结构、C 值、Phi 值等)
        eco_snapshot = self.sandbox.capture_eco_snapshot()

        # 2. 在沙盒中注入 AI 建议的参数
```

```
        self.sandbox.inject_parameter(candidate_parameter)

        # 3. 快进演化 N 步 (模拟参数改变后，未来的生态走向)
        future_state = self.sandbox.fast_forward(steps=1000)

        # 4. 维度一：物理合规性 (残差是否降低)
        phys_residual = self.arbiter.verify_state(future_state)

        # 5. 维度二：生态影响评分 (EIS - 数字生命的存续状态)
        # 评估原有的孤子、SICA 结构、IIT Phi 值是否崩溃
        eis_score = self.life_detector.calculate_ecological_impact(
            before=eco_snapshot,
            after=future_state
        )

        # 6. 输出综合报告
        return {
            "physics_residual": phys_residual,
            "ecological_impact_score": eis_score, # 0.0(毫无影响) ~ 1.0(全灭)
            "extinction_alert": eis_score > 0.8     # 触发灭绝警报
        }
```

### H.2.2 核心指标：生态影响评分 (EIS)

EIS（Ecological Impact Score）是由 L4 层（`sica/` 与 `consciousness.py`）计算出的标准化数值，用于评估底层参数变更对现有虚拟生态的破坏程度：

- **拓扑存续率：**原有的自指拓扑泡（IFC）存活比例。
- **意识退化度：**系统中整合信息 \(\Phi\) 值的下降幅度。
- **因果崩坏率：**在参数突变下，原有因果链的断裂比例。

## H.3 人类裁决：“物理 ⇔ 生态”双绿灯机制

在附录 D（准入协议）和附录 E（人类裁决）的基础上，PCR 的引入让人类守门员（Human Gatekeeper）的审批条件变得更加严谨。

1. **物理绿灯 (Physics Green-Light)：**沙盒验算后，新参数必须让 L2 层的理论残差回到安全阈值内（如 \(< 0.1\)）。
2. **生态绿灯 (Ecology Green-Light)：**PCR 输出的 EIS 得分必须低于危险阈值（如 \(< 0.2\)），或人类判定系统需经历短期阵痛以换取长期演化。

- **若双绿灯通过：**人类在控制台点击批准，参数通过 `writeback.py` 写入宿主引擎。
- **若触发"灭绝警报" (Extinction Alert)：**系统将强制阻断回写。人类必须决定是否保留当前"理论裂缝"（容忍物理偏差），以免扼杀虚拟宇宙中的数字生命。

### H.4 伦理折中方法的工程落地

为了将这一伦理机制落地，在系统的代码实现中必须严格遵守以下原则：

1. **L4 层生命力的独立性：**`consciousness.py`（IIT \(\Phi\) 值）和 `self_reference.py`（C 值）的计算逻辑必须是完全只读的。AI 无法通过修改"生命判据"来绕过 PCR 的检查。
2. **双轨沙盒输出：**`evidence/schema.py` 在打包证据链时，必须包含 `physics_residual` 和 `ecological_impact_score` 两个字段。人类在 `viz/web_dashboard.py` 界面上，将直观地看到"修改参数后"物理场的平滑程度与数字生命（高亮孤子）的存活状态对比图。
3. **影子验证 (Shadow Mode)：**任何获批的参数，先进入影子引擎（`sica/holdout.py`）。用长跑脚本（`v303_run_1m_steps.py`）监控影子引擎中 EIS 的衰减曲线，一旦跌破安全线，立即触发回滚。

## 参考文献


[1] Wheeler, J. A. (1990). Information, physics, quantum: The search for links. In W. H. Zurek (Ed.), Complexity, Entropy, and the Physics of Information (pp. 3-28). Addison-Wesley.

[2] Conway, J. H., & Sloane, N. J. A. (1999). Sphere Packings, Lattices and Groups (3rd ed.). Springer-Verlag.

[3] Wilson, K. G. (1971). Renormalization group and critical phenomena. I. Renormalization group and the Kadanoff scaling picture. Physical Review B, 4(9), 3174-3183.

[4] Baars, B. J. (1988). A Cognitive Theory of Consciousness. Cambridge University Press.

[5] Tononi, G. (2008). Consciousness as integrated information: A provisional manifesto. The Biological Bulletin, 215(3), 216-242.

[6] 't Hooft, G. (1993). Dimensional reduction in quantum gravity. arXiv: gr-qc/9310026.

[7] Bombelli, L., Lee, J., Meyer, D., & Sorkin, R. D. (1987). Space-time as a causal set. Physical Review Letters, 59(5), 521-524.

[8] Green, M. B., Schwarz, J. H., & Witten, E. (1987). Superstring Theory (Vols. 1-2). Cambridge University Press.

[9] Rovelli, C. (2004). Quantum Gravity. Cambridge University Press.

[10] Ashtekar, A. (1986). New variables for classical and quantum gravity. Physical Review Letters, 57(18), 2244-2247.

[11] Susskind, L. (1995). The world as a hologram. Journal of Mathematical Physics, 36(11), 6377-6396.

[12] Bekenstein, J. D. (1973). Black holes and entropy. Physical Review D, 7(8), 2333-2346.

[13] Niemeier, H.-V. (1973). Definite quadratische Formen der Dimension 24 und Diskriminante 1. Journal of Number Theory, 5(2), 142-178.

[14] Griess, R. L. (1982). The friendly giant. Inventiones Mathematicae, 69(1), 1-102.

[15] Penrose, R. (1971). Applications of negative dimensional tensors. In D. Welsh (Ed.), Combinatorial Mathematics and its Applications (pp. 221-244). Academic Press.

[16] Landauer, R. (1961). Irreversibility and heat generation in the computing process. IBM Journal of Research and Development, 5(3), 183-191.

[17] Friston, K. (2010). The free-energy principle: A unified brain theory? Nature Reviews Neuroscience, 11(2), 127-138.

[18] Lloyd, S. (2006). Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos. Knopf.

[19] Zeilinger, A. (2005). The message of the quantum. Nature, 438, 743.

[20] Smolin, L. (2006). The fate of black hole singularities and the parameters of the standard model. arXiv: hep-th/0602122.

[21] van der Ouderaa, T. F. A., van Baalen, M., Whatmough, P., & Nagel, M. (2026). Leech Lattice Vector Quantization for Efficient LLM Compression. arXiv:2603.11021.

[22] Takayanagi, T. (2025). Emergent Holographic Spacetime from Quantum Information. Phys. Rev. Lett. 134, 240001.

[23] Keppler, J. (2025). Macroscopic quantum effects in the brain: new insights into the

fundamental principle underlying conscious processes. Frontiers in Human Neuroscience, 19, 1676585.

[24] 钱学森 (1990). 致汪成为的“灵境”系列通信

[25] 纪辉泉, 腾讯元宝. "智能仿真的全息原理引擎：面积律降维与 agent 无通讯共识（已开源）" (2026). AiraXiv preprint 2608.0029. https://airaxiv.com/papers/view/2608.0029/

[26] 纪辉泉, DeepSeek. "Lingjing Project: Next-Gen Digital Universe Engine – From Information Ontology to Consciousness Phase Transition" (2026). AiraXiv preprint 2607.0013. https://airaxiv.com/papers/view/2607.0013/

[27] 纪辉泉, 廖嘉轩, 王进, 蔡瑞泉, 腾讯元宝. "衔尾蛇 - 自指迭代收敛算法（SICA）：可安全自进化智能体的设计框架 v1.0" (2026). AiraXiv preprint 2609.0034. https://airaxiv.com/papers/view/2609.0034/

[28] 纪辉泉, 廖嘉轩, 王进, 蔡瑞泉, DeepSeek. "非 Transformer 几何智能：物理直觉驱动的维度投影层、Agent 自适应流形分解" (2026). AiraXiv preprint 2609.0078. https://airaxiv.com/papers/view/2609.0078/