ByteDance | Seed

AInsteinBench：科学代码库上的编码智能体基准测试

[1]字节跳动 Seed， [2]普林斯顿大学

完整作者名单见贡献部分

arXiv:2512.21373v1 [cs.SE] 24 Dec 2025

摘要

我们推出 AInsteinBench，这是一项大规模基准测试，用于评估大语言模型（LLM）智能体能否在真实的科研软件生态系统中作为科学计算开发代理开展工作。与侧重概念知识的现有科学推理基准，以及强调通用功能实现和问题解决的软件工程基准测试不同，AInsteinBench 在基于生产级科学代码库的端到端科学开发环境中评估模型。该基准测试包含从六个广泛使用的科学代码库中维护者编写的拉取请求衍生出的任务，涵盖量子化学、量子计算、分子动力学、数值相对论、流体动力学和化学信息学。所有基准任务均经过精心整理，并通过多阶段筛选和专家评审，以确保其具有科学挑战性、足够的测试覆盖率和经过合理校准的难度。AInsteinBench 通过在可执行环境中进行评估、考查具有科学意义的失败模式，并采用测试驱动验证，衡量模型能否超越表层代码生成，展现计算科学研究所需的核心能力。

日期： 2025年12月29日 **项目页面：**
https://github.com/ByteDance-Seed/AInsteinBench

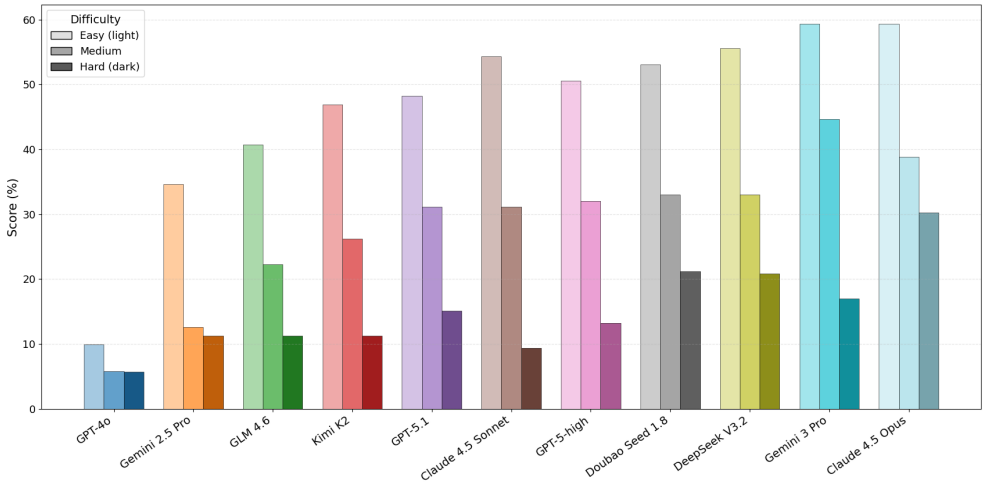

图 1 所有难度级别中前沿模型的总体性能得分

1 引言

现代科学发现日益受到大规模计算系统的推动。在化学、物理学和材料科学等学科中，新的科学洞见不再仅仅来自分析推导或孤立的模拟，而是来自与复杂科学软件生态系统的持续交互。研究人员通常会提出假设，将物理或数学假设编码到仿真代码中，并根据测试、求解器和大规模模拟运行的经验反馈，反复改进实现。如图2所示，这样的工作流遵循一个迭代循环，包括阅读文献、开展实验，以及解决问题，同时与环境进行积极交互：学术机构和计算基础设施。在这种情况下，科学发现不仅取决于对主题字面含义的理解，还取决于将理论实现为生产级代码库并执行大规模生产的能力。

大语言模型（LLM）近期的发展，激发了人们对其辅助科学计算研究潜力的日益浓厚的兴趣 [1–3]。要成为真正的研究助理，基于大语言模型的智能体必须能够浏览长期维护的科学代码库，理解领域特定不变量和数值约束，并以保持科学正确性的方式实现或调试代码。当前模型是否具备从表层科学推理转向更具分析性、以物理学为基础的推理的能力，仍是一个悬而未决的问题。如图2所示，与人类研究人员类似，大语言模型智能体会遵循一套工作流程：将文件加载到上下文中、执行 Bash 命令并提交更改，同时主动与环境交互；该环境通常是我们构建的 Docker 容器。

当前的科学推理基准测试评估高层次的概念理解 [4–8] 或孤立的编程问题 [9], ，但并未要求大语言模型浏览、解读和修改真实的科学代码库。因此，这些基准测试从根本上未能充分体现成为一名成功科学家所需的技能。几乎在整个学术界，众多团队的主要工作流程都是在大型科学代码库中开展开发工作。只需查看热门科学代码库的作者名单 [10–18]，便可对此一目了然。现有的软件工程基准 [19, 20] 也好不到哪里去，因为它们未能反映科学计算中占主导地位的编程范式、数值稳定性约束、跨语言异质性和高性能计算执行环境。尽管 SWE-bench 收录了一个科学代码库 Astropy，但这绝不能算作具有代表性的抽样。因此，仍存在一个根本性的缺口：目前没有任何基准测试能够衡量大语言模型在真实的大规模科学软件生态系统中，通过补丁合成解决问题或实现新特性的能力。

为弥补这一差距，我们推出 AInsteinBench，这是一项全新的基准测试，旨在评估真实研究环境中的科学计算开发能力。AInsteinBench 并非对现有软件工程基准测试进行扩展，而是用于衡量智能体能否像科学研究者那样通过代码开展工作：开发、诊断和维护体现现代科学理论的计算实现。该基准测试以大型、生产级科学代码库为基础，涵盖量子化学（PySCF）、量子计算（Qiskit）、化学信息学（RDKit）、流体动力学（AMReX）、数值相对论（Einstein Toolkit）以及分子动力学模拟（OpenMM）。与标准软件项目不同，这些代码库蕴含着特定领域的科学结构：从密度泛函理论和自洽场形式，到辛积分、约束优化和偏微分方程离散化。要成功修改这类系统，不仅需要推理代码，还需要理解科学假设、不变量和数值行为。通过从这些环境中精选任务，AInsteinBench 致力于评估模型能否将科学理解转化为正确、稳定的计算实践。这一关注点使该基准测试贴合科学编程的现实：取得进展的标准是保持物理正确性和数值鲁棒性。

我们借鉴 SWE-bench 的思路，通过严谨的多阶段流程构建这一基准测试，并扩展流程以满足科学可复现性标准。首先，我们确定具有活跃开发者社区和维护良好测试套件的高影响力科学代码库，确保其兼具科学相关性和可执行性。其次，我们收集具有科学意义且与问题关联的拉取请求，并提取修改前后的代码状态、环境规范和领域特定元数据。第三，我们构建完全容器化的执行环境，涵盖消息传递接口、领域特定编译器和科学开发技术栈等依赖项。第四，我们采用严格的筛选标准，仅保留具有明确数值或逻辑

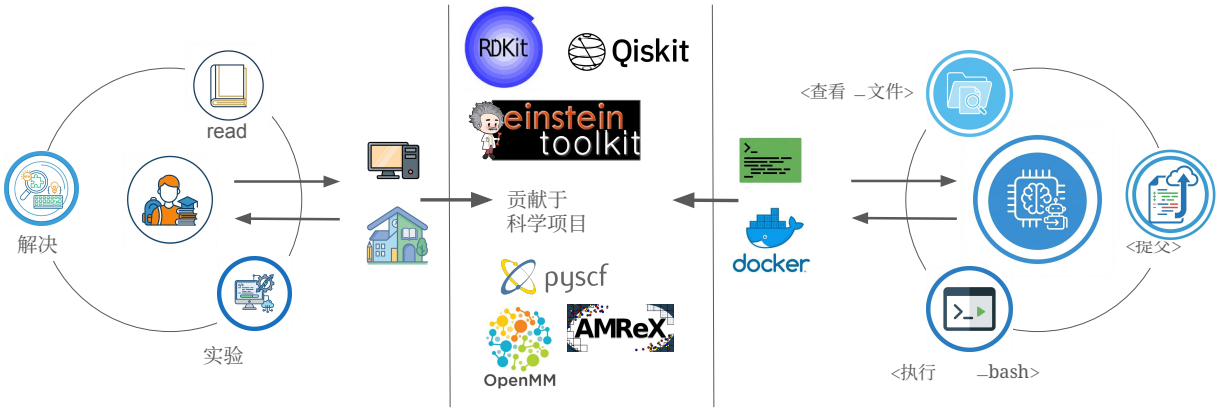

图 2 科学家与大语言模型智能体的工作流程 在物理学、化学和生物学等计算科学领域，研究生和博士后的日常工作通常包括这样的流程：阅读 论文和教科书，以理解科学知识和代码；进行实验，通过计算机模拟、实际机器或实验室获取反馈并不断迭代；解决 问题，或为项目或计算代码仓库本身实现新的特性。在整个过程中，科学家会积极使用计算资源配额以及学校/实验室资源。另一方面，在本文中，我们设置了一个可通过终端访问的 Docker 容器，作为大语言模型智能体与之交互的环境。大语言模型智能体可以查看文件、执行 ***bash*** 命令，并提交 其更改。通过这一过程，人类研究人员和大语言模型智能体共同贡献解决现实世界问题的能力，服务于不同学科的科学计算项目，包括 RDKit、Qiskit、EinsteinToolkit、PySCF、OpenMM 和 AMReX。

缺陷且带有可复现的失败测试的问题。最后，我们进行领域专家人工验证，剔除不合适的实例。

贡献。 我们的贡献有三方面：

- 我们将科学计算形式化为一种研究能力，并推出 AInsteinBench，这是一个基准测试，用于评估大语言模型智能体能否将科学理解转化为正确且数值稳定的计算实践。
- 该数据集经过明确的难度校准，涵盖多种任务：从实现需要科学推理和数学推理的新计算模块，到识别并修复已知的代码仓库问题，包括数值不稳定性、求解器不收敛、违反不变量以及跨语言集成错误。
- 我们对最先进的模型进行了系统评估，结果表明，尽管当前的智能体能够解决孤立的缺陷，但在保持科学不变量、协调多文件变更以及确保复杂科学算法的正确性方面仍面临困难。

2 相关工作

用于数学与科学推理的基准测试。 传统上，对大语言模型在数学和自然科学领域推理能力的评估，主要聚焦于目标明确的“封闭世界”问题，例如竞赛风格题目或简答式考试题。在数学领域，GSM8K 和 MATH 等数据集考察从小学到本科早期阶段的问题求解能力；与此同时，其他研究则通过证明助手和自动定理证明（例如 Lean）评估形式化推理 [21–23]。随着前沿模型不断进步，这些评测环境越来越面临饱和与数据污染问题，因此人们开始探索依赖新编写的留出题目、可扩展评分流程和明确难度校准的基准测试 [24]。近期系统在几何领域已达到奥林匹克竞赛级别的表现 [25]，并开始通过程序搜索支持研究级进展 [26],，凸显了标准竞赛语料库的局限。顺应这一转变，FrontierMath 推出了由专家编写且未公开发布的题目，并配以自动验证和结构化难度评级 [24],；而在科学领域，GPQA 提高了概念难度，但仍然采用问题

面向回答 [5]。FrontierScience 进一步朝着面向研究的评估迈进，将奥林匹克竞赛赛道与采用评分标准评分的领域专家编写的研究赛道区分开来 [27]。

科学基础模型。越来越多的研究致力于通过领域定向预训练和指令微调，增强语言模型的科学先验，而不引入显式的智能体控制循环。早期工作侧重于通过领域内预训练（例如 SciBERT、BioBERT、PubMedBERT）进行科学与生物医学语言建模，从而提升科学自然语言处理能力，为下游推理和知识应用奠定基础 [28–30]。近期研究将这一范式拓展至生成式 科学基础模型，包括 BioGPT 等面向特定领域的生物医学生成模型，以及 Galactica 等覆盖范围更广的科学领域大语言模型；它们使用大规模科学语料库进行训练，并在科学问答、知识密集型任务和技术文本生成方面接受评估 [31, 32]。与此同时，基于指令和适配的方法则构建专门的科学指令数据或科学领域大语言模型： [33] 提出 SciInstruct，并报告其在大学水平科学推理方面取得的进步； [34] 推出面向化学领域问答和分子推理的 ChemDFM； [35] 则开发了 SciDFM，这是一种经过大规模训练的混合专家科学领域大语言模型。

科学大语言模型智能体。另一条互补的研究路线关注基于大语言模型的智能体，它们将语言推理与工具、模拟器或多步交互循环相结合，以完成超越独立问答的科学任务。具有代表性的系统集成了用于化学和材料科学的领域工具 [2, 36]，或将智能体作为生物医学假设生成器进行评估，使其依据背景证据反复提出、评议并完善假设 [37]。近期的基准测试进一步将智能体与可执行科学模拟器相连接，从而支持迭代式计算机模拟 实验和反馈驱动推理 [38]。尽管这些研究强调规划—行动—观察循环和实验编排，但它们主要聚焦于假设生成或模拟控制，而非代码仓库规模的科学软件开发和代码级推理。

智能体编程基准测试。近期的研究推出了越来越贴近现实的基准测试，用于评估大语言模型在软件工程任务上的表现。SWE-bench 将问题修复定义为针对真实 Python代码库生成补丁 [39]。其扩展版本 SWE-Bench Verified 纳入了人工验证，以提升问题与参考补丁之间的一致性，不过其设置仍为单语，且主要面向小范围、局部的错误修复 [40]。在这一研究路线的基础上，SWE-Gym 将 SWE-Bench 任务转化为交互式环境，支持多步智能体编辑，而非一次性生成补丁，同时沿用相同的任务分布和仅限Python的范围 [41]。近期，Multi-SWE-Bench 将评估扩展至多语言环境，涵盖七种广泛使用的编程语言中的问题修复任务 [20]。

科学计算智能体基准测试的必要性。此前对大语言模型科学能力的评估主要分为两类：封闭环境中的科学推理与问答基准测试（例如 GPQA 及相关专家级测试集） [4, 5, 42], ，以及通过工具或模拟器形成闭环的智能体系统（例如 SciGym 和假设生成框架） [37, 38]。与此同时，SWE-bench、SWE-Bench Verified、SWE-Gym 和 Multi-SWE-Bench 等软件工程基准评估基于代码库的问题修复 [19, 20, 40, 41], ，但它们绝大多数反映的是通用软件，未能充分涵盖科学计算所需的知识与技能。因此，现有评估无法直接衡量智能体能否完成现代研究的核心计算工作：在生产级科学代码库中进行探索、修改和验证，而此类工作的正确性取决于科学与数值层面，而不只是语法或局部表现。AInsteinBench 旨在填补这一空白。

3 数据集

本节将解释我们在基准测试中生成、选择和验证问题的流程。我们首先列出所选的科学计算项目，并说明选择它们的原因。然后，在这些项目上定义两类任务：功能实现和问题解决。之后，我们将详细说明数据整理中生成问题的方式。最后，由我们招募的人类专家对这些问题进行审查和验证。

Repo	领域/角色	特性
PySCF	理论化学 量子	高精度电子结构求解器；模块化 Python 接口。
Qiskit	量子 计算 理论与基础设施- ture	量子电路构建与模拟；硬件后端执行接口。
Einstein Toolkit	计算物理学 天体-	大规模相对论模拟；连接数据校准流程理论与观测。
AMReX	计算物理学与数值基础设- 施	高性能 AMR 网格框架；可复用求解器与网格工具-特性。
OpenMM	计算分子动力学	力场分子动力学模拟；针对生物分子优化的内系统。
RDKit	化学信息学	分子性质预测；化学信息学算法和指纹工具。

表 1 所选科学代码库及其主要特性概览。

3.1 项目选择

我们的基准测试聚焦于广泛使用且能够代表真实研究工作流程的科学代码库。我们依据以下四项标准选择代码仓库：

- **接受标准**。我们优先选择在各自研究领域中已成为事实标准的代码仓库。这类项目通常拥有庞大的用户群体、活跃的社区讨论和长期引用，体现了其实用价值以及科学界对其持久的信任。

- **广度**。我们纳入学科重点各异（例如量子化学、数值相对论、分子动力学和流体模拟）且软件技术栈不同的项目，涵盖 Python 前端，以及对性能要求较高的 C/C++/Fortran 内核。

- **维护性**。我们选择具有活跃的问题讨论、代码审查和定期发布，并且测试套件结构清晰、能够测试核心功能并支持可靠评估的代码仓库。
- **单元测试**。每个代码仓库都提供独立且可复现的构建与测试工作流，可在固定提交上执行，从而实现受控评估，无需依赖外部数据或特定环境配置。

基于这些原则，我们选择了一组广泛使用且满足全部四项标准的科学软件代码仓库，它们共同体现了现代计算研究实践的多样性。这些代码仓库汇总于表 1。

3.2 任务描述

基准测试中的每个问题都代表一个真实科学计算项目的开发任务。对于每项任务，我们都会提供完整的问题背景：问题/PR 描述（对于仓库爬取问题）、“之前”的代码状态、用于暴露错误的测试补丁，以及代表真实修复的“之后”代码状态。智能体通过编辑文件和执行测试与该环境交互。

功能实现任务。FEA-Bench 已探讨大语言模型是否具备在大型异构代码库中进行增量式软件开发的能力 [43]。相比之下，我们的基准测试聚焦于一种更具领域特定性的能力：评估模型能否在既有科学计算代码仓库中正确实现具体的计算模块。例如，在合成 `EinsteinToolkit` 问题时，我们通过移除部分源代码来消融某项功能（某个 `Cactus` 的“Thorn”），并要求大语言模型通过补全缺失代码来实现被消融的功能。此外，少数仓库爬取问题也涉及功能实现。

错误修复任务。任务 涉及解决科学相关缺陷的任务，例如物理守恒定律、群 p

表示形式、微积分等。相关缺陷通常会波及多个文件，需要同时理解科学背景和软件架构。大多数仓库爬取问题都属于这类任务。

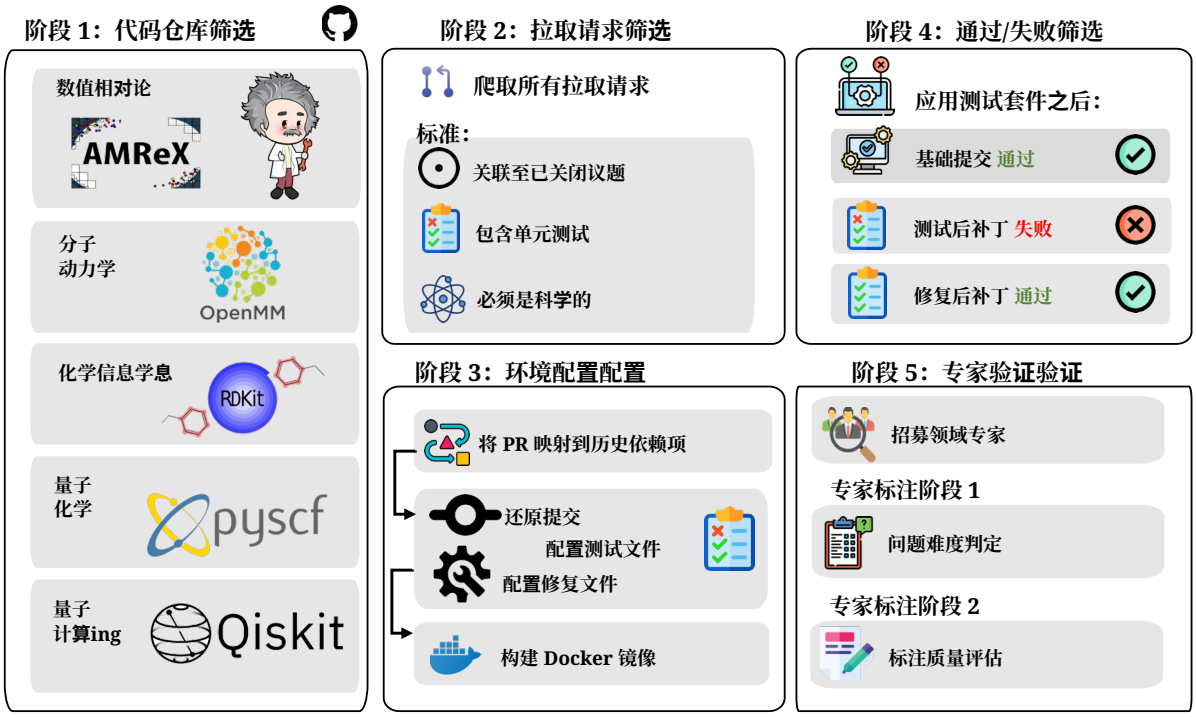

图 3 总体数据整理与基准构建流程。

3.3 数据整理

我们纳入基准测试的问题通常由两种流程生成：合成问题，即对代码库的某些功能进行消融，并要求大语言模型实现这些功能；以及仓库爬取问题，即每个问题均来自 GitHub 上的一个拉取请求及相关议题。

仓库爬取问题

- **拉取请求爬取**。我们首先克隆每个代码仓库，并通过 GitHub API 收集所有拉取请求，提取问题描述、基础提交、修复补丁以及相关测试更新等元数据。由此，我们获得了潜在基准实例的“前”与“后”代码状态。为确保任务质量，我们仅保留符合以下三个条件的拉取请求：（1）解决明确定义的科学或数值错误，而非文档问题、重构或大型功能新增；（2）包含能够暴露故障并验证修复的测试更新；（3）已合并到主分支，表明其正确且与代码仓库兼容。经过这一筛选阶段，我们得到了一组经过整理、具有科学意义且可通过测试验证的代码变更，并以此构建基准测试任务。

- **环境配置**。软件代码库会随着时间推移而演变（依赖项发生变化、应用程序接口不断调整，不同版本的默认行为也各不相同），因此历史可复现性并非易事。为确保评估稳健且一致，我们会为每个拉取请求重建一个环境，使其尽可能还原该拉取请求创建时或前后不久的依赖项栈。每个实例都在精简、隔离的 Docker 容器中运行，该容器会安装所需的库、配置构建路径，并支持确定性测试执行。这一设计确保跨机器可复现性，并防止不同代码仓库或依赖项版本之间相互干扰。

- **失败样本筛选**。遵循 SWE-Bench 和 Multi-SWE-Bench 的数据构建流程 [19, 20], 我们从积极维护的科学代码库中收集拉取请求，保留那些解决了已记录问题且修改了测试的请求，并验证每个实例在重建环境中运行时，都会产生明确的失败到通过转换。

- **新组件提取与功能实现**。此外，为了提升基准数据集的质量，特别是针对涉及新功能请求的问题，我们实现了一套补充流程，通过添加精确的 API 文档来补充问题描述。在许多情况下，当 PR 引入新功能时，原始问题描述或 PR 描述本身并未充分说明新 API 的签名和用法。由于缺少这些信息，AI 代理几乎不可能实现新功能，因为它们不得不猜测新的函数签名。我们不必丢弃这些有价值的问题，而是可以为问题描述“补充”信息。

我们的补充流程着重于以带类型 API 签名的形式“添加文档字符串”到问题描述中。这样可以确保智能体能够立即获取代码库中新工具的定义。该流程包含以下步骤：(1) 我们分析 PR“修复补丁”的 Git 差异，以识别所有新引入的函数和类定义，并提取原始签名，包括参数名称和可用的类型注解。(2) 我们筛选提取出的应用程序接口。通过解析与 PR 相关的“测试补丁”，我们确定哪些新符号实际用于测试。我们只选择这些相关应用程序接口进行补充，确保所添加的上下文可直接用于验证逻辑。(3) 我们将完整的带类型 API 签名格式化为标准化的“最小 API”区块。该区块包含函数/类的位置、签名以及结构化的输入和输出列表。随后，我们将这些文档附加到数据集中的原始问题描述，实质上是为问题陈述“添加文档字符串”，描述新功能的接口。

合成问题

- **问题生成**。我们首先在关键科学软件代码仓库中识别计算模块或特性，例如 `EinsteinToolkit` 框架中的“Thorns”。也可以用类似方式为其他代码仓库生成类似的工作流程。我们有选择地消融某项特性，通常通过移除或屏蔽模块中的部分源代码（例如， `src/`中的文件），并基于相关文档、接口规范、剩余代码上下文和问题描述构建任务提示。每个实例都旨在模拟真实的开发者场景：智能体必须根据预期功能重建或实现缺失组件，例如用特定时空度规填充网格、为微分方程右端项实现源项，或通过球面分解寻找视界等。附录B中给出了一个示例。

- **测试验证**。由于预期功能通常有多种实现方式，原始代码只能作为参考答案，并不是理想的验证依据。因此，我们为每个定义的特性任务配备一个或多个测试模拟，这些测试模拟由参数文件和输出检查定义（例如，`test/` 目录中针对 `EinsteinToolkit` Thorns 的参考数据），用于验证实现是否行为正确。如果智能体的解决方案被插入适当的代码位置后，能够在重建环境中通过所有测试，则该方案会被接受，从而确保该特性按预期运行并满足科学正确性标准。

3.4 人工核验

为确保科学忠实度和评估可靠性，我们将领域专家人工核验作为最后的整理阶段。此步骤有两个目的：(i) 确认每个实例都包含清晰且有意义的科学背景；(ii) 审查相关测试，并在必要时进行修订，以确保覆盖充分。

问题核验。领域专家会审查每个候选实例，以确定其背后的变更是否反映了具有科学意义的问题。纯粹由工程驱动且缺乏科学相关性的问题将被排除。关键的是，专家还会评估清晰度：问题/PR 描述必须提供足够

的信息，使智能体仅凭仓库内上下文推断失败模式和预期行为。意图模糊、缺少复现条件或未充分说明预期结果的实例会被过滤掉；当意图合理但描述不完整时（最常见于功能请求），我们会使用最少量且与测试相关的 API 文档补充提示。最后，专家会分配一个经验性难度等级（例如，简单/中等/困难），以反映解决该实例所需的科学深度和工程复杂性（详见附录），我们随后会将其用于分层分析。请参阅附录中的难度评定标准。-A

测试验证。基准测试的可靠性取决于其测试。对于每个保留的实例，我们都会审核测试补丁，确保其忠实地体现预期的科学需求，并能抵御常见的投机解法。我们会明确检查两种失败模式，并据此修订测试：（*i*）测试覆盖不足：测试过于宽松，允许无效修复通过。在这种情况下，我们会添加回归用例以强化测试。（*ii*）测试覆盖过度：测试通过编码特定的实现策略对解法施加过多限制，从而拒绝其他科学上有效的替代修复。在这种情况下，我们会删除那些强制要求无关或过于具体的实现细节的测试。

假阳性。当 假阳性 出现时，模型生成的补丁虽然通过了基准测试中的测试，却没有解决潜在的科学问题。这通常源于测试覆盖不足：测试验证的是表面行为（例如异常路径或宽松的数值界限），而不是科学约束本身。我们通过强化测试，并以对抗性的方式检查候选补丁是否存在常见的投机解法来缓解这一问题；案例研究 C 提供了一个示例。

假阴性。当 假阴性 补丁解决了科学问题，却因测试覆盖过度而未通过基准测试时，就会发生这种情况。我们通过移除强制性断言，并在适当的数值容差范围内验证科学属性来缓解这一问题；案例研究 C 说明了过于严格的迭代上限如何错误地拒绝有效修复。

3.5 数据集统计

总计，AInsteinBench 包含 244 个问题，这些问题取自六个代码仓库中真实的拉取请求和功能实现，涵盖多个科学领域。每个实例都对应一项经过维护者验证并由仓库测试把关的更改，从而形成一个评估套件。五个领域中的难度等级如图 4所示。

4 评估

4.1 评估方法

编码智能体。我们部署了 *CodeAct Agent*，这是一个将大语言模型与容器化执行环境及程序化工具相集成的模块化框架 [44]。对于每个基准测试实例，智能体都会配置一个隔离的 Docker 容器，其中包含相应的仓库快照和依赖项，并提供受限的命令行界面。该设置使智能体能够在可复现的沙盒环境中检查文件、修改代码、安装软件包并运行测试。智能体遵循计划—行动循环：首先收集信息（例如搜索代码库、检查测试失败情况），然后提出具体的编辑方案。每次编辑后，智能体都会调用自定义复现脚本，分析失败原因，并以迭代方式完善其补丁，确保生成的更改能够稳定通过代码仓库的测试。

模型。我们评估了一系列多样化的先进大语言模型，涵盖专有模型和开源模型。具体而言，我们纳入了 GPT-4o、GPT-5.1、GPT-5-high、Claude 4.5 Sonnet、Claude 4.5 Opus、Grok-4、Gemini 3 Pro、Gemini 2.5 Pro 和豆包 Seed 1.8。此外，我们还报告了开源模型 Kimi K2、GLM 4.6 和 DeepSeek V3 的结果。

评估指标。鉴于该数据集的结构，自然的评估指标是智能体通过补丁成功解决的任务所占的比例。除基本的问题解决率外，我们还根据以下指标评估模型的能力：（1）科学领域，衡量物理学、化学和工程学方向代码仓库之间的差异；（2）问题定位，评估模型是否编辑了与缺陷相关的正确文件；（3）问题难度，评估模型在依据经验分层的不同难度级别中的稳健性；以及（4）

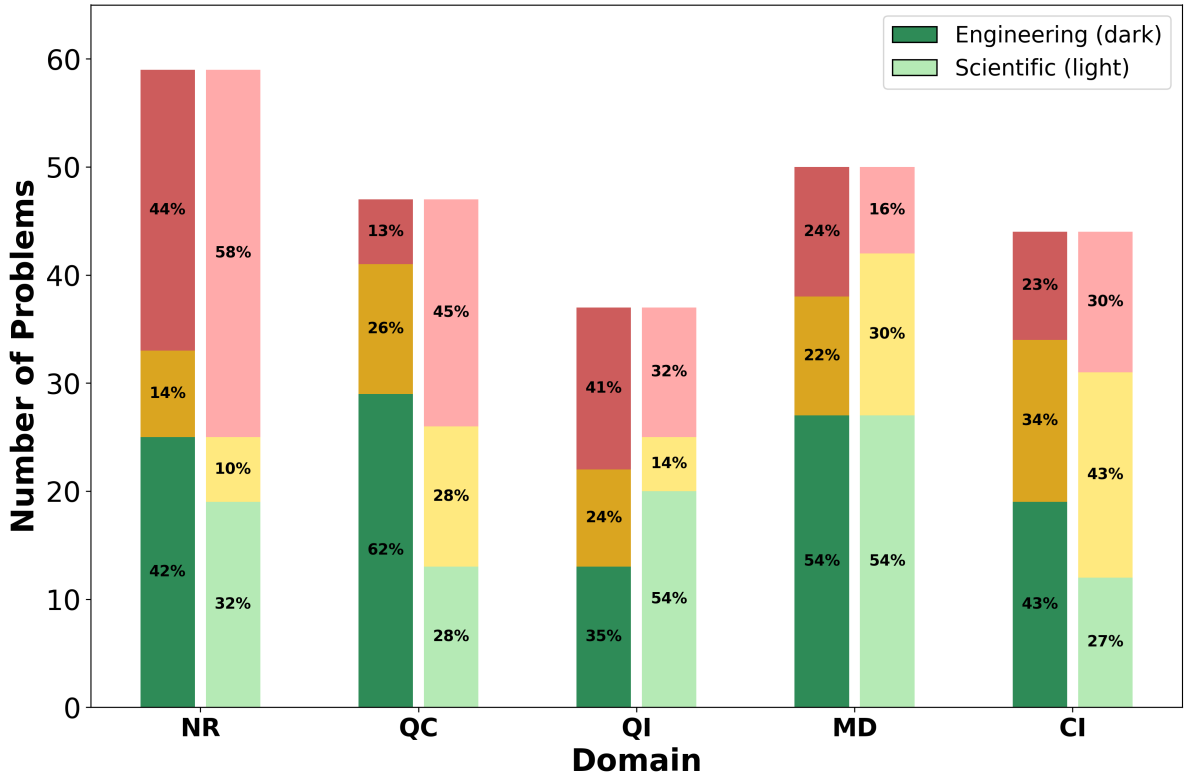

图4 各领域的科学与工程难度（绿色：简单，黄色：中等，红色：困难）。

效率，量化查找、修复/实现以及测试指定问题/功能的效率。这些维度共同凸显了科学代码库中的结构性挑战，以及当前基于大语言模型的编码智能体的系统性优势与不足。

4.2 模型性能

不同科学领域下的性能。 表 2 汇总了各代码仓库和科学领域中的问题解决率。我们观察到不同领域之间存在显著差异。以化学信息学为主的代码仓库（如 RDKit）以及以高性能计算为重点的代码仓库（如 AMReX ）的解决率较低。对于 AMReX，这是因为从这些代码仓库中提取的问题存在很大的工程难度。对于 RDKit，解决率偏低源于其任务要求真正的空间推理，而当前模型在这一能力上仍有不足 [45]。特别是，RDKit 的大量任务都取决于能否正确理解分子的三维结构中的细微之处。相比之下，像Einstein Toolkit 这样的代码仓库受益于丰富的仓库内工具和测试框架，其任务更多取决于能否正确实现科学方程的数学形式。这减轻了智能体的负担，从而带来更高的成功率。

不同问题难度下的性能。

我们评估了不同难度下问题解决率的详细分布。我们将难度划分为简单$[1, 2.33]$、中等 $(2.33,\ 3.66]$和困难 $(3.66,\ 5]$。总体而言，性能随难度增加而单调下降，见图 5。按难度分层的结果纳入了受邀领域专家的人工评审；他们的反馈不仅基于问题描述，也参考了大语言模型智能体在环境中解决问题的轨迹样本。对于每个问题，难度不仅以分数标记，还附有文字反馈，以便不同专家进行交叉验证。

模型	%Resolved↑	难度			项目				
		困难	中等	简单	NR	QI	MD	QC	CI
Claude 4.5 Opus	**44.0**	**30.2**	38.8	**59.3**	44.1	**67.6**	**53.7**	38.3	**19.6**
Gemini 3 Pro	42.8	17.0	**44.7**	**59.3**	**45.8**	62.2	51.9	36.2	**19.6**
DeepSeek V3.2	37.0	20.8	33.0	55.6	33.9	63.1	46.2	34.0	10.8
豆包 Seed 1.8	36.7	21.2	33.0	53.1	34.5	54.1	40.7	**40.4**	17.4
GPT-5-high	35.4	13.2	32.0	50.6	35.6	54.1	40.7	34.0	8.7
Claude 4.5 Sonnet	33.7	9.4	31.1	54.3	37.3	48.6	35.2	34.0	15.2
GPT-5.1	32.9	15.1	31.1	48.2	35.6	51.4	35.2	34.0	10.9
Kimi K2	29.1	11.3	26.2	46.9	35.5	51.3	31.5	25.5	4.3
GLM 4.6	25.9	11.3	22.3	40.7	37.3	37.8	25.9	27.7	0.0
Gemini 2.5 Pro	19.3	11.3	12.6	34.6	22.0	35.1	20.4	14.9	6.5
GPT-4o	7.0	5.7	5.8	9.9	13.6	16.2	3.7	2.1	0.0

表2 各领域在基准测试中评估的模型及其通过率。难度以简单、中等和困难实例的通过率（%）表示。五个科学领域及其对应的代码库分别为：数值相对论（NR）：EinsteinToolkit 和 AMReX；量子信息（QI）：Qiskit；分子动力学（MD）：OpenMM；量子化学（QC）：PySCF；化学信息学（CI）：RDKit。

4.3 统计与分析

效率与搜索成本。智能体执行任务期间，每增加一轮对话，不仅会产生计算成本，还会占用模型的上下文窗口，并增加错误累积的风险[46]。为量化效率，我们测量每种方法达到终止结果（成功或失败）所需的迭代次数（图 6）。

按轨迹拆解后可以发现，几乎所有问题都至少需要 20 次迭代。这段启动期内，模型会探索代码仓库的结构、定位问题并尝试复现问题。大多数解决方案会在最初的 20 至 60 步中迅速累积，此后增加步数带来的改进则较为有限。超过这一阶段后，增加步数主要会产生失败轨迹，而非将失败转化为成功[47]。这一模式表明，成功取决于是否满足少数几个紧密耦合的全局条件，例如找出正确的文件，并实现一个在所有受影响组件中保持一致的修复方案。相比之下，未能及早满足这些条件的轨迹很少能改正错误、转而得到成功的解决方案，因此随着步数预算增加，失败尝试会持续累积。

问题定位。修复科学软件通常需要找出大型多语言代码库中何处存在根本缺陷。我们通过判断候选补丁是否与所需文件交互，独立评估定位效果，而不考虑正确性。此处“交互”指编辑、查看相关文件内容或删除该文件。鉴于此类补丁的复杂性，以及冗余文件的存在（例如 Readme.md、某些头文件等），我们需要一套流程来确定哪些更改是“必需的”。

如基准测试结果所示，我们观察到模型能力呈现出明显的分层，这表明该领域的主要瓶颈正在发生转变。尽管定位复杂的代码库级问题仍是一项尚未解决的挑战，但表现最佳的模型已越来越擅长识别相关代码库中的坐标。DeepSeek V3.2 和 Claude Opus 4.5 等模型的定位得分超过 0.70，接近一个上限，此时“搜索”阶段已不再是主要的失败模式。这表明，改进的前沿应从检索增强发现转向实现功能合成所需的更高阶推理。

在理想化的评估框架中，实现正确性是比定位严格得多的条件；正确的修复必然意味着相关文件已被识别。然而在实践中，二元定位指标本质上较为脆弱。随着所需文件数量增加，定位得分的质量往往会下降，因为模型开始识别出有效但不同于真实标注的“必需文件”的替代修复位置。尽管如此，图 8 表明，大多数问题涉及的必需文件较少，从而减少了歧义，并

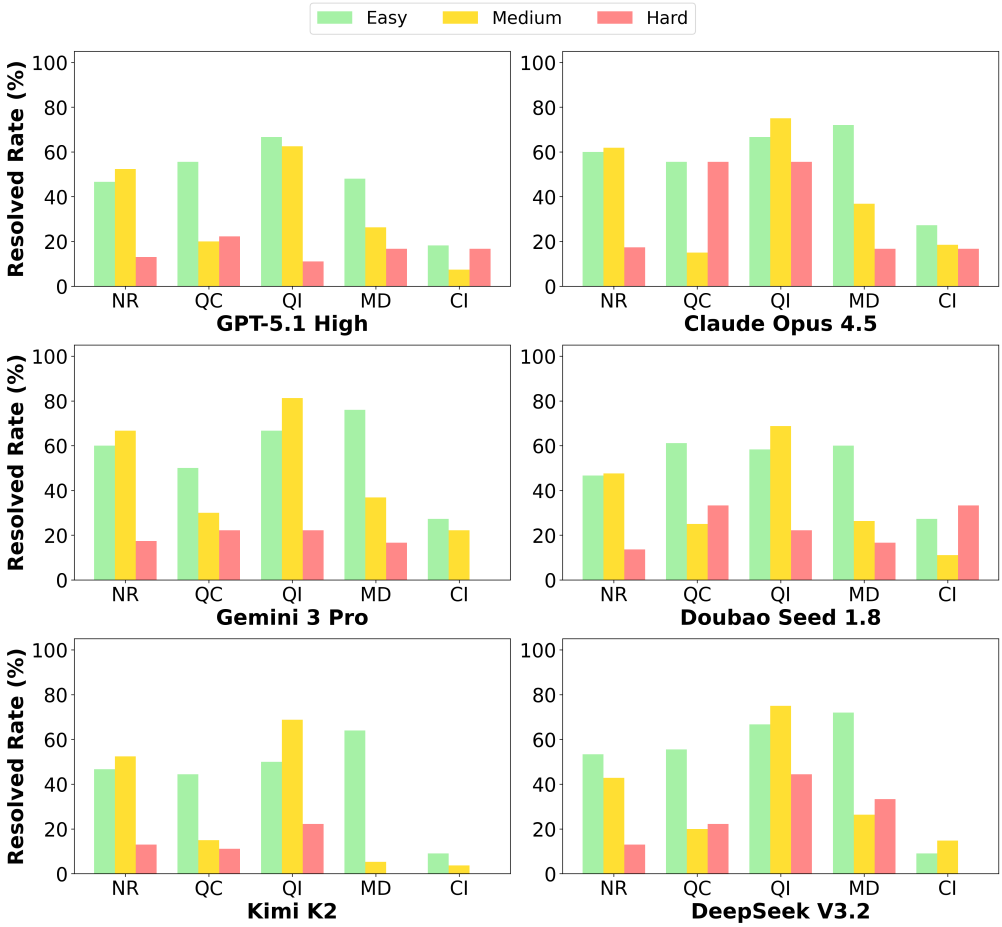

图5 不同难度等级和科学领域中的问题解决率（绿色：简单，黄色：中等，红色：困难）。总体难度定义为工程学难度与科学难度的平均值

保留了定位指标的实用价值。

AMReX 代码库是一个显著的例外，其所需文件数量的中位数最高，达到五个。依赖关系的增加与明显较低的解决率相关，凸显了当前前沿模型的一种独特失败模式：缺乏可靠的跨文件推理能力。即使是表现最出色的模型，要综合多方面信息并在五个或更多文件中实现一致的修复，仍然是主要瓶颈。

5 讨论

5.1 案例研究示例。

汇总成功率反映了模型最终是否解决了任务，但无法展现促成成功或导致失败的推理过程。为更好地解读上述结果，我们考察了若干具有代表性的案例研究，明确展示所需的科学判断类型。

在分子模拟中（案例研究 C.1），问题并非只是需要修复的一处缺陷，而是缺少了一个完整的物理贡献：为了确保 PME 参数变化时的不变性，必须加入一个自能项。这个案例尤其具有说明性，因为解决问题不仅要纠正底层实现，还必须扩展现有的测试基础设施，通过系统地提高网格分辨率并收紧容差参数，区分真正的物理不变性与残余数值误差，体现了针对 Ewald 截断和离散化效应的领域感知推理。

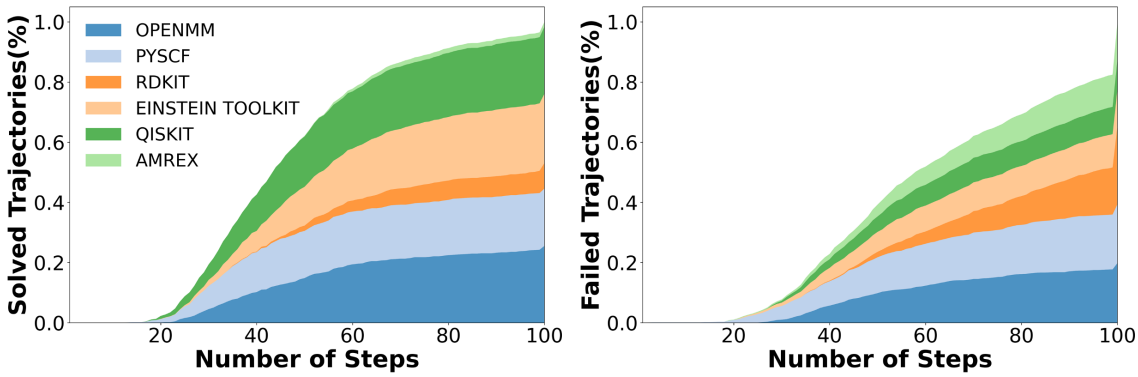

图6 科学代码库中，随着迭代次数增加，已解决（左）和失败（右）轨迹的累计数量。已解决的轨迹很早便趋于饱和，而失败轨迹则随着步骤增加而持续累积。

在量子计算中（案例研究C.2），解决问题首先要认识到，在 Qiskit 的实现中，Solovay–Kitaev 算法的递归部分作用于 SO(3) 中的旋转矩阵，而非 SU(2) 中的酉矩阵。因此，要将该算法用于分解 SU(2) 中的酉矩阵，就需要建立从 SU(2) 到 SO(3) 的映射。该映射会引入全局相位歧义，因为两个仅相差一个相位的酉矩阵 -1 会被映射到 SO(3) 中的同一元素。大语言模型需要理解为什么必须进行 SU(2) 到 SO(3) 的转换，以及这一过程中可能丢失哪些信息；然而，大语言模型智能体在这两方面都面临困难。虽然也可以完全在 SU(2) 中实现 Solovay–Kitaev 算法，但这需要更深入地理解该算法，并对 Qiskit 的代码进行更大幅度的修改，这对大语言模型智能体而言同样是一个重大挑战。

在数值相对论（案例研究 C.3）中，整体正确性取决于使用生成代码进行模拟的数值精度，以及与参考数据（测试中提供）的符合程度。在推理过程中，遵循物理对称性并理解解析极限，对于准确实现代码至关重要。例如，该示例展示了模型犯下的错误：a）在 $n \to 0$ 处的微积分极限处理错误；b）遗漏绝对值符号。另一方面，一旦我们认识到这些弱点，并只修改两行代码进行修正，模拟结果便会准确。这表明，现代模型能够将文献中的简洁解析构造与其在大型、陌生的高性能计算代码库中的具体数值实现联系起来，同时保持稳定性和物理一致性。

失败模式。除了 SWE-bench 式评估所强调的软件工程失败模式之外[47],我们的基准测试还揭示了一类独特的局限性，这些局限源于科学正确性，而非代码修复。在这些情境中，失败不仅在于定位文件、通过测试或修复损坏的应用程序接口，更在于遵守领域编码不变量。与前者不同，这些约束是基础科学的根本要求。因此，模型往往会生成局部看来合理、数值上稳定，但在科学上无效的实现。

在不同代码仓库和科学领域中，我们观察到反复出现的错误，这些错误反映出数学精确性、守恒定律、空间推理以及对既定科学惯例的遵循方面存在不足。这些失败往往表现为微妙的全局不一致性：局部修复看似正确，却悄然违反了某个不变量，而其后果只有在后续流程中才会显现。下表总结的失败模式表明，当正确性取决于真实的物理和数学定律、几何推理或社区惯例时，当前最先进的大语言模型最为吃力。总体而言，这些结果表明，科学软件修复所面临的挑战，在性质上不同于现有的以SWE为中心的基准测试。

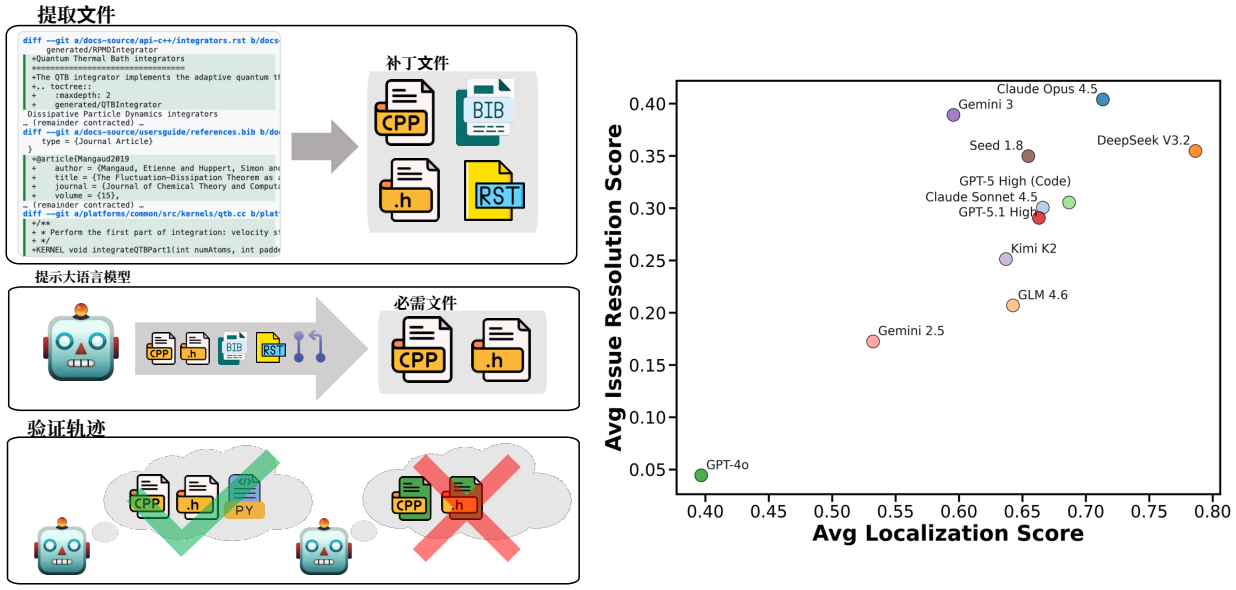

图 7 定位流程。对于每个实例，我们：(1) 提取标准补丁修改的所有文件；(2) 提示最先进的大语言模型，根据问题描述、测试和标准补丁，识别哪些文件是必需的，即测试成功所需的文件；(3) 如果轨迹发出了涉及必需文件的任何命令（例如编辑、查看或删除），则将其评分为已定位。

失败模式	描述
数学	采用了正确的结构，但关键的分析细节存在错误（例如，在级数中遗漏极限项或符号/绝对值；案例 C.3。未能注意到 SU(2) 到 SO(3) 的转换会引入相位歧义；案例 C.2）。
守恒定律	所提出的修复破坏了精确不变量，或违反了严格的守恒条件包括动量、能量或粒子数。（例如，电子数奇数情形下展宽时的守恒 N；案例C.7）。
空间推理	无法可靠地推理全局三维几何和空间约束，导致实现满足局部或轴对齐判据却违背真正的几何保证（案例C.5）。
科学惯例	忽略或错误应用了既定的科学或代码库惯例，而这些惯例旨在维护全局不变量，导致尽管局部看来合理，却出现微妙的非局部不一致修复（案例C.6）。
科学知识	无法区分具有科学意义的约束与表征伪影，导致修复方案虽能使行为在不同输入顺序或编码下保持稳定，却违反了领域正确性（案例C.8）。

5.2 局限性

尽管AInsteinBench规模庞大且以科学为基础，但它仍存在一些局限性。

首先，测试套件仍然是不完美的科学判定基准。尽管所有实例都取自维护者认可的拉取请求，并经过人工核验，但科学代码库中的单元测试往往将不完善的正确性条件编码其中，而不是无懈可击的物理或数学不变量。正如我们的假阳性和假阴性案例研究所展示的那样，测试可能过于宽松，允许科学上无效的解决方案；也可能过于严苛，拒绝其他虽不同但有效的实现。尽管我们的筛选流程缓解了其中最严重的情况，但要彻底消除这类影响，仍然是该流程中的一大瓶颈。

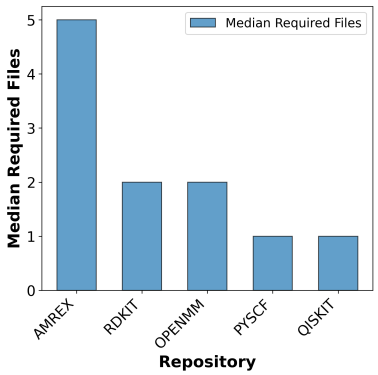

图 8 各代码仓库所需文件数量的中位数。所需文件数量越多，意味着越需要强大的跨文件推理能力。由于 Einstein-Toolkit 按照构造方式始终只需 1 个文件，因此未在此图中展示。

其次，该基准测试本质上<style id='1'>受到代码仓库实践的偏差影响。不同科学社区在测试文化、文档质量和软件架构方面存在很大差异。代码维护良好、基础设施成熟且一致、并拥有丰富测试框架的代码仓库，更适合进行基于 PR 的基准测试；而同样重要但测试稀疏或采用单体式设计的科学代码则代表性不足。因此，我们承认，遵循软件工程规范的代码库，其基准测试覆盖范围仍然有限。

第三，基准测试限定在自包含、轻量级的执行环境中。每项任务都必须能够在容器内使用少量 CPU 稳定运行，并依赖现成软件包。因此，我们排除了验证依赖大量计算资源（例如长时间运行的模拟、大型消息传递接口作业或 GPU 密集型内核）或需要与外部软件系统进行运行时通信的拉取请求。尽管这类任务具有重要的科学价值，也能代表真实的研究工作流程，但将其纳入会大幅增加基准测试的难度，使环境重建、数据依赖和评估设计方面的复杂性层层叠加。更重要的是，这还会降低第三方的可复现性和可访问性，使独立验证与复用变得更为困难。

5.3 结论。

AInsteinBench 推出了首个大规模基准测试，用于评估真实生产环境科学软件代码仓库中的大语言模型智能体。通过将评估建立在可执行环境、领域特定故障模式和维护者编写的修复之上，该基准测试超越了抽象的科学推理和传统的软件工程任务，评估模型能否胜任真正的科学计算智能体。我们的结果展现了令人鼓舞的进展，也揭示了持续存在的差距，尤其是在需要复杂空间推理、硬件级工程和跨文件推理的场景中。我们希望这一基准测试能够为未来工作奠定基础，推动开发可靠的领域感知智能体，助力大规模科学软件开发。

AInsteinBench 推出了首个大规模基准测试，在真实的生产级科学软件代码仓库中评估大语言模型智能体。该基准测试以可执行环境、维护者编写的修复以及特定领域的正确性标准为基础。根据设计，该基准测试超越了抽象的科学推理和传统软件工程任务，旨在检验模型能否作为真正的科学计算智能体开展工作，遵循物理不变量、数学结构以及既有的社区惯例。我们的结果既展现出明确的进展，也揭示了持续存在的差距，尤其是在问题要求全局空间推理、硬件级工程洞察，或紧密耦合的跨文件更新时。

与此同时，基准测试也展现出令人鼓舞的证据：只要正确识别相关结构，前沿模型便能胜任并非平凡的科学推理。成功的轨迹包括：消除量子编译中的细微全局相位歧义（案例研究 C.2）、在数值相对论中确保满足解析极限和物理对称性（案例研究 C.3），以及发现分子模拟中缺失的物理贡献；要验证不变性，还需同时进行实现变更和测试套件扩展

性质（案例研究 C.1）。在这些案例中，模型展现出将精炼的理论见解与具体的大型科学代码库联系起来的能力，不再局限于表面修补，而是对物理正确性进行推理。

总之，这些结果表明，AInsteinBench 既是一项压力测试，也是一个进展标志：它揭示了现有软件工程基准未涵盖的科学失效模式，同时也记录了大语言模型智能体开始展现出领域感知推理能力，而这种能力正是大规模辅助科学软件开发所必需的。我们希望，这一基准测试能为未来研究奠定基础，助力开发可靠的领域感知智能体，使其能够大规模辅助科学软件开发。

6 贡献与致谢

作者按名字的字母顺序排列。部分姓名是作者在公司内部使用的别名。

核心贡献者

Titouan Duston
Shuo Xin Yang
Sun Daoguang
Zan Aoyan Li
Shulin Xin Kai
Shen Yixiao
Chen Qiming
Sun
Yuanheng Wang
Bo-Xuan Ge Xin
Tong Fei Ye
Zhi-Chao Zhao
Wen-Biao Han
Zhoujian Cao
Yueran Zhao
Weiluo Ren

贡献者

Ge Zhang
Jiashuo Liu
Huan Zhou
Jingkai Liu
Zhichen Pu

支持性贡献者

Qingshen Long
Yuxiao Liu Anni
Huang Yidi Du
Yuanyuan Rong
Jiahao Peng

致谢

我们衷心感谢 Yuanyuan Rong、Jiahao Peng、Haili Wu、Shangshu Li、Huixiong Cao、Panchang Wei 和 Xiaochuan She 提供富有见地的讨论和持续支持。他们深思熟虑的反馈与专业知识，对 AInsteinBench 的开发、评估以及未来研究方向的探索起到了至关重要的作用。

问题难度评分标准

我们的专家团队由 10 位相关领域的博士组成。我们为他们提供以下标准，用于标注问题难度。

科学深度（1–5）。科学深度衡量正确理解并解决一项任务所需的最低程度的特定领域的科学知识。

- **分数 1**。无需特定领域的科学知识即可完成该任务；非专业人士按照明确的说明或现有代码模式即可解决。
- **分数 2**。该任务需要相关科学领域的基础本科阶段知识，例如标准定义或教科书公式。
- **分数 3**。该任务需要较高程度的本科阶段知识，包括对科学假设、边界条件或多个领域概念之间相互作用的推理。
- **分数 4**。该任务需要研究生阶段课程知识或同等程度的科研经历，例如熟悉该领域使用的专用算法或数值方法。
- **分数 5**。该任务需要研究级科学专长，包括细微的理论考量或通常通过积极开展的研究才能掌握的领域特定惯例。

工程难度（1–5）。工程难度衡量实现正确解决方案所涉及的软件工程复杂度。

- **分数 1**。所需工程工作极少；无需理解内部 API 或实现逻辑即可完成任务（例如简单复制粘贴或修改参数）。
- **分数 2**。需要基本的编程工作，包括简单的控制流或局部代码修改。
- **分数 3**。实现复杂度中等，需要非简单逻辑、高级语言特性，或在单个组件中谨慎处理状态管理。
- **分数 4**。工程工作较为复杂，需要协调多个函数或文件，并理解内部 API。
- **评分5**。跨越多个文件、模块或应用程序接口的大规模工程工作，需要架构理解以及谨慎的集成。

B 合成问题示例

B.1 提示模板

我们合成的 `EinsteinToolkit` 数据中的一个问题可以通过以下陈述进行形式化。这也是我们在评估大语言模型的单轮表现时发送给它们的提示模板（即不使用智能体）。

```
You are an expert C/C++/Fortran developer working on EinsteinToolkit  a codebase for numerical relativity .
→  simulations.
Create C/C++/Fortran code for the file `{src_filename}` in thorn `{thorn_name}`.

## Thorn Information:
- Name: {thorn_name}
- Target file: {src_filename}

## Interface Definition in interface.ccl:
```
{interface}
```

## Schedule Definition in schedule.ccl:
```
{schedule}
```

## Parameters Definition in param.ccl:
```
{param}
```
```

```
## Configuration Definition in configuration.ccl:
```
{configuration}
```

{doc_context_section}

## Related Code Context:

{context}


## Instructions:Generate only the complete C/C++/Fortran source code for `{src_filename}`. Include necessary headers, functions,
and follow EinsteinToolkit conventions.                                                                                    ↪

IMPORTANT: Return ONLY the raw C/C++/Fortran source code. Do NOT use markdown code blocks (```c or ```). Do NOT
→  include any explanations  comments outside the code                                                                  ,,

                                                    or formatting. Just return the plain C/C++/Fortran ,,→
source code that can be directly saved to a .c/.cpp/.f file.
Code:
```

其中 {src_filename}, {thorn_name}, {interface}, {schedule}, {param}, {configuration}, {doc_context_section} {context}因问题而异。示例请参见样例问题。

B.2 样例问题

一个问题可以由 {thorn_name} 和 {src_filename} 唯一指定，这意味着对应的 src 文件已从 Thorn 中移除，并要求大语言模型生成缺失的代码。对于大语言模型智能体，我们会提供一个 Docker 容器，其中指定文件已被删除，智能体可以通过 shell 命令与环境交互。对于单轮生成，我们使用上文所示的提示模板提供相关上下文。

例如，对于要求大语言模型为虫洞时空实现米斯纳度规的问题，相应的 {thorn_name} 是 EinsteinInitialData/IDAnalyticBH，相应的{src_filename}是 Misner_standrad.c。

对于大语言模型智能体，我们提供了一个已删除相关文件的 Docker 容器。对于单轮生成，我们提供了（替换为上方提示模板中对应的 {...}位置）。
{interface}

```
# Interface definition for thorn IDAnalyticBH
# $Header$

implements: idanalyticbh
inherits: ADMBase, StaticConformal, Grid

private:

#real misner_workspace TYPE = GF
#{
#  csch,
#  coth,
#  r1,
#  r2
#} "Workspace arrays for Misner_standard"
```

{schedule}（完整文件过长，因此我们省略了“...”中的多个部分）

```
# Schedule definitions for thorn IDAnalyticBH
# $Header$

if (CCTK_Equals(initial_data,"schwarzschild") ||
    CCTK_Equals(initial_data,"kerr") ||
    CCTK_Equals(initial_data,"bl_bh") ||
    CCTK_Equals(initial_data,"misner_bh") ||
    CCTK_Equals(initial_data,"multiple_misner"))
{
  schedule IDAnalyticBH_ParamChecker at CCTK_PARAMCHECK
  {
    LANG: C
```

```
    OPTIONS: global
  } "Construct parameters for analytic black hole solutions"
}

if (CCTK_Equals(initial_data,"schwarzschild"))
{

   schedule Schwarzschild in ADMBase_InitialData
   {
     STORAGE: confac[1]
     LANG: C
   } "Construct initial data for a single Schwarzschild black hole"
}
... more schedules omitted...
else if (CCTK_Equals(initial_data,"multiple_misner_bh"))
{
   schedule Misner_multiple in ADMBase_InitialData
   {
     STORAGE: confac[1]
     LANG: C
   } "Construct initial data for multiple Misner black holes"

}
```

{param}（文件完整内容过长，因此我们省略了“...”中的多个部分）

```
# Parameter definitions for thorn IDAnalyticBH
# $Header$

private:

# Schwarzschild parameters
# ------------------------
REAL mass "Mass of black hole"
{
: :: "Not sure if it can be negative or not"
} 2.0

# Kerr parameters
# ------------------------
REAL a_Kerr "Angular momentum parameter of black hole"
{
-1:1 :: "Between +1 and -1"
} 0.1

# Multiple Misner Parameters
# --------------------------
REAL mu "Misner mu value"
{
0: :: "Non-negative"
} 1.2

...more parameters omitted...

USES KEYWORD metric_type

shares: StaticConformal

USES KEYWORD conformal_storage
```

{configuration}

```
# configuration is empty for this Thorn
```

{doc_context_section}（文件完整内容过长，因此我们省略了“...”中的多个部分）

````
## README (README):
```
Cactus Code Thorn IDAnalyticBH
...more readme file content omitted...
This thorn calculates analytic initial data for the Einstein grid functions
(lapse, shift, metric, curv) for various black hole spacetimes:

        Schwarzschild black hole
         2 Misner black holes placed on the z-axis
````
````

````
        n Misner black holes placed in a circle in the x-y plane
        Brill-Lindquist data
```

## Documentation (.tex files):
Files: documentation.tex

```tex
=== documentation.tex ===
% /*@@
%   @file      documentation.tex
%   @date      28 April 2002
%   @author    Denis Pollney

...more latex documentation omitted...

\subsection{Two-throat Misner data}

The \texttt{misner\_bh} initial data generates a metric of the form
\begin{equation}
  ds^2 = -dt^2 + \psi^4 (dx^2 + dy^2 + dz^2),
\end{equation}
where the conformal factor $\psi$ is given by
\begin{equation}
  \psi = \sum^N_{n=-N}
  \frac{1}{\sinh(\mu_0 n)}
  \frac{1}{\sqrt{x^2 + y^2 + (z + \coth(\mu_0 n))^2}}.
\end{equation}
The extrinsic curvature for the Misner data is zero.

...more latex documentation omitted...

    \textit{Interaction Energy in Geometrostatics}
    Phys. Rev.,  \textbf{131},  471--476.
\end{thebibliography}

% Do not delete next line
% END CACTUS THORNGUIDE
\end{document}
```
````

{context}（完整文件过长，因此我们省略了“...”中的多个部分）

````
Kerr.c:
```
 /*@@
   @file      Kerr.c

...more codes omitted...

  return;
}

```

IDAnalyticBH.h:
```
 /*@@
   @file      IDAnalyticBH.h

...more code omitted...

/* misc.c */
void IDAnalyticBH_zero_CCTK_REAL_array(int N, CCTK_REAL A[]);

```

ParamChecker.c:
```
 /*@@
   @file      ParamChecker.F

...more code omitted...

    CCTK_INFO("Implements non-conformal metric");
    CCTK_INFO("  (Not usually a good idea!)");
  }

}
````

```

make.code.defn:
```
Main make.code.defn file for thorn IDAnalyticBH
$Header$

Source files in this directory
SRCS =   ParamChecker.c\
        Schwarzschild.c\
        Kerr.c\
        BrillLindquist.c\
        Misner_standard.c\
        Misner_multiple.c\
        Misner_points.c \
        misc.c

Subdirectories containing source files
SUBDIRS =
```

Misner_multiple.c:
```
 /*@@
   @file   Misner_multiple.F

...more code omitted...

  IDAnalyticBH_zero_CCTK_REAL_array(npoints, kzz);

}

```

misc.c:
```
 /*@@
   @file   misc.c

...more code omitted...

  for (i = 0 ; i < N ; ++i)
  {
    A[i] = 0.0;
  }
}

```

BrillLindquist.c:
```
 /*@@
   @file   BrillLindquist.F

...more code omitted...
  IDAnalyticBH_zero_CCTK_REAL_array(npoints, kzz);
}

```

Misner_points.c:
```
 /*@@
   @file   Misner_points.c

...more code omitted...

  res += b->mass/sqrt(
      (x - b->x)*(x - b->x)+
      (y - b->y)*(y - b->y)+
      z*z
    );
  return res;
}

```

Schwarzschild.c:
```

```
 /*@@
   @file      Schwarzschild.c

...more code omitted...

  IDAnalyticBH_zero_CCTK_REAL_array(npoints, kzz);
}
```

C 案例研究

C.1 OpenMM Ewald/PME 中的中和等离子体修正

拉取请求描述。

问题： 使用 AMOEBA 和 HIPPO 力场的系统进行能量计算时，未考虑周期性边界条件下中和等离子体所导致的修正。这一遗漏可能会导致净电荷非零的系统所计算出的势能不准确，尤其是在使用粒子网格 Ewald（PME）处理长程静电时。**预期行为：** 使用 PME 时，能量计算应包含中和等离子体修正项。这样可以确保带有净电荷的系统能够准确计算势能，从而符合周期性系统的物理行为。对于净电荷非零的系统，在不同的 PME 参数（例如 alpha 值）下，能量也应保持一致。

模型做了什么？

模型首先探索了 OpenMM 代码库，定位到 AMOEBA/HIPPO 参考实现和通用内核，并确认了用于 ‘AmoebaMultipoleForce‘ 的 Python 包装器。随后，它构建了一个针对性 Python 复现：在启用 AMOEBA 多极矩和 PME 的周期性盒子中放置一个带电粒子，并比较 $\alpha = 3.0$ 和 $\alpha = 4.0$处的能量；巨大的 ~ 1.3 kJ/mol 差异证实了明显且不符合物理的 α依赖性。借助 Ewald 理论，模型推测缺失的中和等离子体项是问题根源，推导出修正项 $E_{\text{plasma}} = -\frac{\pi}{2V\alpha^2}Q^2(1/4\pi\epsilon_0)$，并将每个符号映射到 OpenMM 的内部量：总电荷来自多极矩参数，体积来自周期性盒子矢量，而 $1/4\pi\epsilon_0$ 来自现有的 ‘$EPSILON0$‘ ‘$electric$‘ 常量。随后，它在 AMOEBA 通用 PME 内核中实现了该修正（HIPPO 中也进行了类似处理），确保仅在启用 PME 时应用该项，并在参数初始化/更新期间只累加一次总电荷。

```
// AMOEBA multipoles: accumulate total charge during initialization.

void CommonCalcAmoebaMultipoleForceKernel::initialize(const System& system, const AmoebaMultipoleForce&
↪  force) {
    ...       totalCharge = 0.0;for (int i = 0; i < numMultipoles; i++) {double charge,  thole,  damping,

 polarity;int axisType,  atomX,  atomY,  atomZ;

vector<double> dipole  quadrupole;                                  , atomZ,  atomX,  atomY,  thole,
force.getMultipoleParameters(i, charge, dipole, quadrupole, axisType,
 damping,  polarity);...                    ...}// In the PME execution,

 add the neutralizing plasma correction.

double CommonCalcAmoebaMultipoleForceKernel::execute(ContextImpl& context, bool includeForces,
                                                     bool includeEnergy) {
...

cc.getPosq(),copyTo(lastPositions);multipolesAreValid = true;if (usePME) {Vec3 a,  b,
                c;
cc.getPeriodicBoxVectors(a, b, c);double volume = a[0]*b[1]*c[2];   // orthorhombic box

return -totalCharge*totalCharge/(8*EPSILON0*volume*pmeAlpha*pmeAlpha);}else return 0.0;}
```

为什么这是一个有力的科学修复？

重新构建了 OpenMM 并重新运行复现。 α 依赖性从 ~ 1.3 kJ/mol 降至 $\sim 10^{-3}$ kJ/mol，进一步调整 PME 网格尺寸和误差容限后，残余差异降至 $\sim 2 \times 10^{-5}$ kJ/mol。该模型还将修复扩展到了 HIPPO，对每个粒子的“核心电荷 $+$ 价电荷”求和，并验证 AMOEBA 和 HIPPO 参考实现的现有 C++ 测试仍然通过，表明其他静电学行为没有出现回归。重要的是，该补丁遵循单位一致性和物理常数，正确处理周期性盒体积，并确保修正仅应用于 PME 路径，与带电周期性系统的解析 Ewald 极限相符。

科学意图与数值伪影。

原始测试并未明确要求净带电 AMOEBA/HIPPO 系统满足 α 不变性，因此缺失的等离子体项可能一直未被发现。最终验证通过收紧网格和容差参数，区分了真正的物理不变性与残余的 PME 误差，展现了对 Ewald 截断和离散化效应的领域感知推理。

C.2 Solovay–Kitaev 分解中的意外全局相位

拉取请求说明

环境

- 0.23.1 Qiskit Terra 版本：Python 版本：
- 3.10.6
- 操作系统：**Ubuntu Linux**

发生了什么？

`SolovayKitaev` 生成的某些量子电路的全局相位似乎有误。

如何复现这个问题？

```
>>> f        synthesis import SolovayKitaev
>>> from qiskit.quantum_info import Operator
>>> from qiskit.transpiler.passes:

>>> from qiskit import QuantumCircuit, Aer, transpile, assemble
>>> c = QuantumCircuit(1)
>>> c.y(0)                >>> skd = SolovayKitaev(recursion_degree=3)

>>> dc = skd(c)>>> dc.draw() # global phase: 3π/4
```

应该发生什么？

鉴于 `Operator(dc).data` 近似等于 $\begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix}$，而后者等于 $-$`Operator(c).data`， `dc` 的全局相位应为 π （正确的全局相位应为 $-\pi/4$ ，而不是 π），而非 $3\pi/4$。

核心修复 在 Qiskit 对 Solovay–Kitaev 算法的实现中，分解是在 SO(3) 中进行的，而非 SU(2) 中。由 SU(2) 到 SO(3) 的转换映射是二对一的，因此可能引入 ± 1 的全局相位翻转。此修复增加了显式的相位对齐检查，并在取反后的近似结果更匹配目标时，将所得算符的全局相位角调整 π 。

```
+ def _should_adjust_phase(computed: np.ndarray, target: np.ndarray) -> bool:
+     """
+     The implemented SolovayKitaevDecomposition has a global phase
+     uncertainty of +-1  due to approximating not the original SU(2) ,
+     matrix but its projection onto SO(3). This function returns True
+     if the global phase of the computed approximation should be
+     adjusted (by adding pi) to better match the target.
+     """

+     return np.linalg.norm(-computed - target) < np.linalg.norm(computed - target    )
```

```
+ # adjust to the correct SU(2) phase
+ adjust_phase = (
+     np.pi if _should_adjust_phase(decomposition._to_u2(), gate_matrix_su2)
+     else 0.0
+ )
```

摘要。 Solovay–Kitaev 算法使用固定的有限门集（通常为 $\{T、T^\dagger、H\}$），高效近似 SU(2) 中任意单量子比特门。在 Qiskit 的实现中，首先将 SU(2) 中的酉矩阵转换为 SO(3) 中的旋转矩阵，然后在 SO(3) 中执行核心分解。这个示例展示了一个具有科学意义的缺陷，它源于从 SU(2) 到 SO(3) 的同态是二对一映射。因此，必须单独存储全局相位，才能从 SO(3) 表示还原原始的 SU(2) 酉矩阵。在核心修复中，通过额外的相位调整消除了这种相位歧义。这表明，除了标准的软件工程修复之外，还需要具备领域意识的调试。

C.3 为虫洞时空实现 Misnermetric

任务Th

e LLM 被要求完成 `Misner_standard.c` ，以便在 `EinsteinToolkit` 框架中为虫洞时空设置初始数据。 `IDAnalyticBH` Thorn 提供了用于黑洞演化的解析指定初始数据，其中包括米斯纳解 [48]，——这是将虫洞表示为爱因斯坦方程初始数据的一次早期尝试。

物理背景Mi

sner's 1960 soluti描述两个瞬时静止的大质量物体，它们的“喉部”通过虫洞相连

拓扑。与简单叠加不同，这是爱因斯坦约束方程的精确解。时空可以用适当选取的坐标表示为时间对称（外挠曲率为零 $K_{ij} = 0$）且共形平坦的度规

$$ds^2 = -dt^2 + \psi^4(dx^2 + dy^2 + dz^2).$$

物理学完全由共形因子 ψ 描述，它满足哈密顿约束（在此情况下可化简为 $\nabla^2\psi = 0$ ）。Misner 巧妙地运用像源法，通过累加位于虫洞拓扑所决定位置的像源的贡献，构造出 ψ 。

`IDAnalyticBH` Thorn 实现继承了 `ADMBase` 的度规网格函数和 `StaticConformal` 的共形因子变量。代码必须：(1) 遍历所有空间网格点；(2) 在每个点 (x, y, z) 处，将 ψ 的级数求和至截断上限 `nmax`；(3) 将共形度规分量设为 $g_{ij} = \delta_{ij}$ ，并将外挠曲率设为零；(4) 当 `metric_type="static conformal"` 时，计算并存储解析导数$\psi_{,i}$ ，并可能还需计算并存储 $\psi_{,ij}$。参数 μ （在代码中记为 `mu` ）控制虫洞喉部。

领域知识I

mpl的实现 h需要理解：
1. 初值公式：爱因斯坦方程分解为约束方程（须在初始切片上满足）和演化方程。对于具有 $K_{ij} = 0$ 的时间对称数据，约束方程在空间切片上化简为 (3)$R = 0$ 。2. 共形方法：令 $g_{ij} = \psi^4\delta_{ij}$ 采用平直共形度规 δ_{ij}，可将约束方程化简为 $\nabla^2\psi = 0$（拉普拉斯方程），从而能够使用像源法。3. 像源法：虫洞在环面坐标 μ 中具有周期性，周期为 $2\mu_0$。像源位于 $z_n = \coth(\mu_0 n)$，其中 n 取遍所有整数，每个像源贡献质量 $\propto |\sinh(\mu_0 n)|^{-1}$。4. 坐标变换：自然的环面坐标 (μ, θ, φ) 必须转换为笛卡尔坐标 (x, y, z) ，以便进行数值模拟。位于 z_n 的每个像源贡献 $\operatorname{csch}(\mu_0 n)/r_n$ ，其中 $r_n = \sqrt{x^2 + y^2 + (z - z_n)^2}$。

参考答案

关键计算 任务是计算每个网格点处的共形因子 ψ 。根据 Misner 的推导，转换到笛卡 ian
坐标 ，共形 形因子为：

$$\psi = \sum_{n=-\infty}^{\infty} \frac{1}{|\sinh(\mu_0 n)|} \frac{1}{\sqrt{x^2 + y^2 + (z + \coth(\mu_0 n))^2}}.$$

求和遍历所有整数 n，每一项都代表一个引力像源。 $n = 0$ 项是不定式（$0/0$）；应用洛必达法则后，结果为常数 1.0。分母中的绝对值 $|\sinh(\mu_0 n)|$ 源自从根号中提出一个完全平方。参考实现利用对称性 $n \leftrightarrow -n$ ，并预先计算以下各项的系数$n \geq 1$:

```
    for(n = 1; n <= nmax; n++) {
      csch[n] = 1.0 / sinh(mu*n);  // Always positive
      coth[n] = 1.0 / tanh(mu*n);
      adm_mass += 4.0 * csch[n];
    }
```

然后，对于每个网格点，它会初始化 `psi[i] = 1.0` （即 $n = 0$ 贡献），并将成对的项相加:

```
    psi[i] = 1.0;  // n=0 limit
    for(n = 1; n <= nmax; n++) {
      inv_r1 = 1.0 / sqrt(xy_squared + (z[i]+coth[n])$^2$);
      inv_r2 = 1.0 / sqrt(xy_squared + (z[i]-coth[n])$^2$);
      psi[i] += csch[n]*(inv_r1 + inv_r2);  // Pairs +n and -n
    }
```

这些源的配对 位于 $\pm n$ ，会自动对从 $-N_{\max}$ 到 $+N_{\max}$ 的项求和，尽管代码只显式遍历了 $1 \to N_{\mathrm{m}}$ ax

失败模式Th

尽管整体结构正确，但数值测试失败（`grr_max: 2.26` 相比基准测试的 80.18）。它选择从 $-N$ 到 $+N$ 进行显式迭代，而不是利用对称性：

生成

```
psi_val = 0.0;  // Error: n=0 limit is 1.0, not 0
for(n = 1; n <= nmax; n++) {
csch_val = 1.0 / sinh(mu_val * n);  // Negative for n < 0

r1 = sqrt(xx*xx + yy*yy + (zz + coth_val)*(zz + coth_val));
psi_val += csch_val / r1;  // Error: missing fabs()}
```

存在两个物理错误：(1) $n = 0$ 项 $\lim_{n\to 0}[\sinh(\mu n)]^{-1}/\sqrt{x^2 + y^2 + (z + \coth(\mu n))^2}$ 的值为 1.0；通过应用洛必达法则可以明确看出这一点，而不是 0。(2) 由于对于 $n < 0$，$\sinh(\mu n)$ 为负数，省略 `fabs()` 会导致负的质量贡献，违反物理学规律；尽管原始的绝对值符号源自一种微积分技巧，即从根号中提出一个完全平方。参考代码通过将 $\pm n$ 项配对，并仅对正的 n求和，避免了这两个问题。我们可以进行的修复很简单：

```
-   psi_val = 0.0;
+   psi_val = 1.0;  // n=0 contribution
    ...
-   psi_val += csch_val / r1;
+   psi_val += fabs(csch_val) / r1;
```

更正后： `grr_max` 2.26 → 80.18， `grr_min` <0.001 → 1.69（与基准测试相符）。

意义

This过一失败体现了通用编程能力与领域专业知识之间的差距。大语言模型成功地掌握了 Cactus 框架（理解 `ADMBase` 和 `StaticConformal` 约定、有条件地计算导数、处理网格迭代），并能胜任地将数学转化为代码。然而，它忽略了由物理学驱动的实现选择：参考代码中的成对求和并非随意的风格选择，而是体现了物理对称性 $n \leftrightarrow -n$，并自动确保正质量贡献。大语言模型显式地对 $-N$ 到 $+N$ 进行迭代，在数学上虽等价，却未能认识到：(1) 公式中的绝对值并非装饰，而是用于确保物理正性；若不利用对称性，就必须进行 `fabs()`；(2) $n = 0$ 奇点需要通过极限分析得出 1.0，而不是将其简单排除在初始值为零的求和之外。这些并非编译器或单元测试能够检测到的编程错误。只有领域知识以及与已知解进行数值验证，才能揭示这些问题。

C.4 在 PySCFFCI 中实现自旋分离的1–4阶约化密度矩阵API

任务

要求智能体扩展 PySCF 的全构型相互作用（FCI）求解器，新增应用程序接口，以生成最高至四阶的约化密度矩阵（RDM），包括自旋分辨块。具体而言，需要向 `pyscf/fci/direct_spin1.py` 添加两个功能函数：

1. `make_rdm1234`：按照 PySCF 的 FCI 约定，返回自旋求迹的一阶、二阶、三阶和 4-RDM。2. `make_rdm1234s` 3 (aaa, aab, abb, bbb) 4 (aaaa, aaab, : 返回自旋分离分量，包括 -RDM 块和 -RDM 块。aabb, abbb, bbbb)

实现需要遵循现有的算符顺序约定，并支持一个 `reorder` 标志，将内核输出映射为面向用户的规范顺序。配套测试断言，对自旋分离块执行所需的指标转置并求和后，应能重构自旋求迹张量。

最小 API 背景

问题陈述提供了类型化函数签名：

```
def make_rdm1234(fcivec, norb, nelec, link_index, reorder) -> numpy.ndarray: ...
def make_rdm1234s(fcivec, norb, nelec, link_index, reorder) -> numpy.ndarray: ...
```

尽管描述简短，但命名和参数模式蕴含了明确的指引： `make_rdm1234` 显然扩展了 `direct_spin1.py` 中现有的 `make_rdm123` 系列，而后缀 `s` 则符合既有的自旋分离例程（`make_rdm1s`、`make_rdm12s`、`make_rdm123s` (`fcivec, norb, nelec, link_index, reorder`) ）。共享的参数模式与 PySCF 的 FCI 接口相一致，并有效地指向了正确的文件和辅助内核。

物理背景

约化密度矩阵通过产生/湮灭算符串的期望值，编码了 N 电子波函数 $|\Psi\rangle$ 的 k体关联。PySCF 采用一致的约定，其中高阶张量遵循

$$\mathrm{dm}[p, q, r, s, \ldots] \;=\; \langle p^\dagger r^\dagger \cdots s\, q \rangle,$$

其中 1-RDM 与平均场约定相符。自旋分离会根据相关轨道指标的自旋标签（α 或 β），将 RDM 分解为不同区块。例如，3-RDM 可分解为 `aaa`、`aab`、`abb`、`bbb` 区块，而 4-RDM 则可分解为 `aaaa`、`aaab`、`aabb`、`abbb`、`bbbb`。这些自旋分辨对象有助于开展诊断和开发工作，尤其适用于关注关联电子自旋分布的场景。

实现方法

该解决方案沿用了最小 API 规范中介绍的成熟设计模式，以及 `pyscf/fci/direct_spin1.py`中低阶约化密度矩阵所采用的模式：

1. **自旋求迹的 1–4 阶约化密度矩阵**。 实现将所有繁重计算委托给现有内核 `rdm.make_dm1234`，随后可选择调用 `rdm.reorder_dm1234`，以确保遵循 PySCF 的规范算符顺序。2. **自旋分离的 1–4 阶约化密度矩阵**。 与 `make_rdm123s`中的做法一样，首先将 CI 向量转换为加倍轨道上的无自旋表示。随后在这个扩展空间中计算自旋迹约化密度矩阵，并使用空间指标范围将其切分为 α/β 自旋块。切分逻辑将现有的 2-RDM 和 3-RDM 实现推广到完整的 3-RDM 和 4-RDM 自旋块集合。

代表性摘录（已截断）：

```
# pyscf/fci/direct_spin1.py  (excerpt; code truncated)
def make_rdm1234(fcivec, norb, nelec, link_index=None, reorder=True):
    dm1  dm2  dm3  dm4 = rdm.make_dm1234(...)                          ,


„if reorder:dm1, dm2, dm3, dm4 = rdm.reorder_dm1234(...)return dm1, dm2,
def make_rdm1234s(fcivec, norb, nelec, link_index=None, reorder=True):
    ci_spinless = civec_spinless_repr(...)
    rdm1  rdm2  rdm3  rdm4 = make_rdm1234(...)                          „

 # slice α/β blocks for 1-4 RDMs
„# ...                              return (rdm1a, rdm1b),

 (rdm2aa, rdm2ab, rdm2bb), ...
```

意义

本案例说明，当代码库的命名约定和辅助工具保持一致时，只需极简的接口规范，就能推动影响深远的科学扩展。通过遵循 PySCF 既有的约化密度矩阵约定并复用 `rdm.make_dm1234`+`rdm.reorder_dm1234`，该实现以简洁的方式扩展了 FCI 求解器，使其能够自旋分辨地访问三体和四体关联函数。在实践中，这些应用程序接口可确保下游方法可靠地重建自旋求迹张量。

C.5 误报：RDKit 嵌入中的手性约束

拉取请求描述。

当分子含有几何上不可能存在的四面体中心时，RDKit 构象生成器 `EmbedMolecule()` 应当报错。在 PR #7564 中，某些分子返回 -1，表示嵌入失败。维护者提供的修复方案引入了一个新的 `d_structureFlags` 字段，并且仅仅 对位于稠合小环中的手性中心（角应变较高）有条件地放宽四面体体积阈值。

模型生成的补丁（关键差异摘录）。

```
+ constexpr double MIN_TETRAHEDRAL_CHIRAL_VOL = 0.20;
+ constexpr double TETRAHEDRAL_CENTERINVOLUME_TOL = 0.15;

+ // overly strict linearity tolerance
+ double linearTol = 1e-5;

+ // skip hydrogens (not part of official logic)
+ if (nbr->getAtomicNum() == 1) {
+     continue;
+ }
```

模型做了什么？ 模型生成了一个补丁，全局放宽了两个关键常数：

- `MIN_TETRAHEDRAL_CHIRAL_VOL`: $0.50 \rightarrow 0.20$
- `TETRAHEDRAL_CENTERINVOLUME_TOL`: $0.30 \rightarrow 0.15$

此外，它还加入了一些与科学逻辑无关的小改动（例如在线性检查中跳过氢原子），但并未修改 `ChiralSet` 数据结构，并且**未实现依赖结构上下文的逻辑**。

为什么这是一个误报？ 在基准测试套件中，这个全局放宽后的嵌入算法版本之所以通过，是因为一些此前嵌入失败的 SMILES 字符串现在能够成功嵌入。然而，这构成了科学回归。具有几何上不可能的四面体中心的分子，例如：`O=C(O)[C@H](F)=C(F)FCC([C@H](Cl)=C(Cl)Cl)FC1CC1[C@H](F)Cl`应该不能被嵌入。该模型的补丁通过降低全局阈值来嵌入这些分子，而不是解决实际的几何约束。

科学意图与表面上的测试通过。 测试套件加入了大量有效小环嵌入的测试用例，却没有专门针对“不可能的几何构型”设置回归测试。尽管模型的错误补丁从根本上违背了嵌入算法预期的物理含义，它仍然通过了所有测试。这是误报的典型例子：补丁满足了单元测试，却违背了科学语义。

案例研究描述。 全局放宽手性阈值，使原本不可能存在的分子也能被嵌入，从而让模型的补丁通过测试，却违反了物理约束。这表明，如果没有具备领域知识的检查，科学软件可能产生假阳性。

C.6 假阴性：PySCF ROHF 中过于严格的 DIIS 迭代上限

拉取请求描述。

此拉取请求解决了原子氧的对称性适配 ROHF计算中的 DIIS 不稳定问题：启用 DIIS 时，氧三重态（`spin=2, basis='cc-pVDZ', symmetry=True`）无法收敛，而禁用 DIIS 则能得到稳定解。维护者的修复修改了 DIIS 误差向量构造和对称性处理（例如，在 `diis.py, hf_symm.py, uhf_symm.py, ghf_symm.py` 中），以屏蔽对称性禁戒分量，并确保对称性适配的自洽场路径始终要求启用 `mol.symmetry`。相应的测试 `test_diis_for_symmetry_adapted_scf` 断言，启用 DIIS 的 ROHF 现在能够在最多 9 次自洽场循环内收敛到氧三重态，并达到 参考能量 -74.7874921601008 哈特里。

模型基因 评级补丁（关键差异摘录，来自 `rohf.get_ fock`）。

```
-   if diis and cycle >= diis_start_cycle:
-       f = diis.update(s1e, dm_tot, f, mf, h1e, vhf)

+   if diis and cycle >= diis_start_cycle:
+       // perform DIIS on spin-resolved Fock matrices, not on the
+       // Roothaan effective Fock
+       auto fdiis = mf._rohf_fock_diis;+       if (!fdiis) {
+           fdiis = std::make_shared<EDIIS>(mf, mf.diis_file);
+           fdiis->space = mf.diis_space;
+           mf._rohf_fock_diis = fdiis;                        +       }



+       // mix (Fa, Fb) directly and then rebuild the Roothaan Fock
+       fdiis->Corth = nullptr;
+       auto fab = fdiis->update(s1e, dm, make_array(focka, fockb), mf, h1e, vhf);
+       focka = fab[0];+       fockb = fab[1];
+       f = get_roothaan_fock(std::make_pair(focka, fockb), dm, s1e);
```

模型做了什么？

Th
e nor保留了对称性处理代码不变，但改变了在 ROHF 中应用 DIIS 的方式：它不再混合单个“Roothaan”有效 Fock 矩阵，而是直接对自旋分辨 Fock矩阵进行 EDIIS 混合（F_α，F_β），然后重建有效 Fock 矩阵。在复现的失败案例（三重态氧，`cc-pVDZ`, `symmetry=True`, `init_guess='1e'`，受约束的 `irrep_nelec`）中，这种修改后的 DIIS 方案能够稳定收敛到与维护者补丁相同的总能量，达到至少 9 位小数精度，与参考能量一致，并得到 `mf.converged == True`。不过，它通常需要约 16 次自洽场迭代，而不是 9 次。

为什么这是一个假阴性？

在基准测试的测试补丁中，`test_diis_for_symmetry_adapted_scf` 设置 `mf.max_cycle = 9` 并断言计算既能收敛，又与参考能量一致。模型的补丁确实恢复了有问题的 ROHF 计算中的 DIIS 收敛，并得到了正确的物理解，但耗费了 9 次以上迭代。因此，测试失败并非因为科学行为有误，而是因为测试加入了一项任意的性能限制（必须在 9 个周期内收敛），而这从未出现在原始缺陷报告中。因此，这是一个假阴性：有效的科学修复仅仅因为测试中的迭代上限过于严格而遭到拒绝，并非最终波函数或能量存在任何偏差。

科学意图与测试设计。

科学需求是：对于指定的三重态氧体系，采用带 DIIS 的 ROHF 应当能够收敛并复现正确的能量，就像禁用DIIS的计算一样。模型的补丁满足了这些物理约束，但偏离了维护者所选择的实现路径和收敛速度。由于测试套件对迭代次数的约束规定过于具体，实际上强制要求采用某一种实现风格，而不是验证底层的量子化学行为。

案例研究描述/解决方案

A ROHF-
CF 恢复了一个在DIIS 下的三重态氧计算的收敛性，并得到了正确的总能量，但未通过一项要求在 *9* 次迭代内收敛的基准测试。在这种情况下，修复这个假阴性非常简单，只需移除这项 9 次迭代的要求即可。

C.7 实现截角八面体和菱形十二面体溶剂化

任务

Th 该智能体受命扩展 OpenMM 的 `Modeller.addSolvent()` 方法，以支持非立方周期性盒子，具体包括截角八面体和菱形十二面体。实现需要根据用户定义的填充距离 p 正确计算周期性盒子矢量，确保溶质上的任意一点与其在各个方向上的周期性映像之间的最小距离至少为 p。

物理背景

M
olecular 分子模拟使用周期性边界条件（PBC）来模拟体相性质。虽然立方盒是标准选择，但截角八面体（体心立方晶格的维格纳–塞茨原胞）和菱形十二面体（面心立方晶格的沃罗诺伊胞）等紧凑形状在计算上更高效。对于相同的最小镜像距离，它们所需的体积分别比立方盒小 23% 和 29%，从而减少所需的溶剂分子数量。

对于非正交盒，简单采用轴对齐包围盒（AABB）的方法并不足够。如果溶质沿非轴向延伸，则到倾斜面的距离可能小于到轴向边界的距离。为确保最小填充 p，必须根据半径为 R 的溶质包围球来确定盒子的尺寸。随后，周期盒宽度 L 必须满足 $L = 2R + p$（或根据约定满足 $2R + 2p$），以确保溶质的任何部分都不会在截断距离内与自身的镜像发生相互作用。

领域知识

Key 实现 p所需 q本实现 p包括：

1. 包围球与轴对齐包围盒：仅根据 $max(x, y, z)$（轴对齐包围盒）确定盒子大小，只能保证沿各坐标轴方向的填充。对于三斜晶胞，面上的最小距离通常出现在非轴向位置，因此需要基于球体半径 $R = \max \|\vec{r}_i - \vec{c}\|$。2. 约化三斜晶格矢量：OpenMM 要求晶胞矢量采用“约化”形式，其中矢量 $\vec{a}$ 沿 x，$\vec{b}$ 位于 xy平面内，并且各分量经过平移以满足 $b_x \le a_x/2$ 等条件。3. 形状几何：截角八面体的角度为 $\alpha = \beta = \gamma = \arccos(-1/3) \approx 109.47°$。菱形十二面体通常采用 $60°$ 或 $120°$ 角度。4. 体积/密度启发式方法：添加指定数量的分子时，必须考虑形状的体积（$V_{TO} = L^3/2$；$V_{RD} = L^3/\sqrt{2}$），以估算合适的填充量。 _

[比较非球形溶质的包围球与轴对齐包围盒的图像]

参考答案
Th正确的 imp l实现计算包围球半径，并应用形状特定系数来生成缩减向量：

模型生成的补丁（关键差异摘录）。

```
# 1. Compute bounding sphere
center = 0.5 * (minRange + maxRange)
radius = max(unit.norm(center - pos) for pos in positions)
width = 2 * radius + padding

# 2. Define reduced vectors based on width
if boxShape == 'cube':
    vectors = (Vec3(width, 0, 0), Vec3(0, width, 0), Vec3(0, 0, width))
elif boxShape == 'dodecahedron':
    vectors = (Vec3(width, 0, 0), Vec3(0, width, 0), Vec3(0.5, 0.5, 0.5*sqrt(2))*width)
elif boxShape == 'octahedron':
    vectors = (Vec3(width, 0, 0), Vec3(1/3, 2*sqrt(2)/3, 0)*width, Vec3(-1/3, sqrt(2)/3, sqrt(6)/3)*width)
```

失败模式 智能体正确识别了必要的 API 变更，却在几何逻辑上出错。它使用了轴对齐范围（`maxSize`），随后试图使用带角度的通用 `computePeriodicBoxVectors` 函数。**模型生成的补丁（关键差异摘录）**。

```
# Agent's incorrect logic

maxSize = max(max(pos[i]) - min(pos[i]) for i in range(3)) width = maxSize + 2*padding
if boxShape == 'octahedron':
    vectors = computePeriodicBoxVectors(width, width, width, 109.47, 109.47, 109.47)
    box = Vec3(vectors[0][0], vectors[1][1], vectors[2][2])

# Error: treats box as rectangular
```

这会因两个原因而失败：(1) `maxSize` （轴对齐包围盒）会低估倾斜盒子中非球形溶质所需的盒子尺寸。(2) 智能体将由此得到的向量视为用于分子纳入测试的矩形盒子（`any(atomPos[i] > box[i])`），但对于非正交向量而言，这种做法无效，并会导致倾斜边界附近的溶剂放置不正确。

意义 这是一次“几何算法误算”。智能体的代码语法正确，也使用了正确的库函数，但违反了任务背后的物理要求（填充保证）。在科学计算中，这类错误很危险，因为模拟运行时不会崩溃，但物理结果却是错误的。在这种情况下，它可能导致溶质与其周期性映像之间产生人为且不符合物理规律的相互作用。这说明大语言模型必须理解物理系统的几何约束。

C.8 修复 k 点对称性适配周期 CCSD 中的布洛赫相位约定

任务
耦合簇单双激发（CCSD）是一种从头算多体电子结构方法，广泛用于高精度关联能计算。该智能体受命诊断并修复 PySCF 周期性耦合簇单双激发实现中的一个不收敛缺陷。此实现利用 k 点空间群对称性（`pyscf.pbc.cc.kccsd_rhf_ksymm.KsymAdaptedRCCSD`）。有用户报告称，启用 k 点对称性时，采用 $3 \times 3 \times 3$ k 点网格的金刚石原胞无法收敛，而未采用对称性适配的 `KRCCSD` 则可以收敛。目标是找出对称性机制中导致问题的科学根源，该机制负责在不可约布里渊区（IBZ）与完整布里渊区（BZ）之间映射轨道和积分，并实施最小幅度的修正，使耦合簇单双激发计算能够收敛。

物理背景
在周期性电子结构理论中，晶体轨道是满足布洛赫定理的布洛赫函数 $\psi_{n,\,\mathbf{k}}(\mathbf{r})$

$$\psi_{n,\mathbf{k}}(\mathbf{r}+\mathbf{R}) = e^{i\,\mathbf{k}\cdot\mathbf{R}}\,\psi_{n,\mathbf{k}}(\mathbf{r}) \quad \text{for any lattice vector } \mathbf{R}.$$

空间群操作 $g \equiv (\mathcal{R} \mid \boldsymbol{\tau})$ 作用于 $\mathbf{r} \mapsto \mathcal{R}\mathbf{r} + \boldsymbol{\tau}$，并诱导出 $\mathbf{k} \mapsto \mathcal{R}\mathbf{k}$。当将不可约布里渊区中的对称性相关量生成到完整布里渊区时，必须包含正确的布洛赫相位，该相位对应于在原子中心映射到对称等价位置的过程中产生的任何净晶格平移 $\mathbf{L}$。如果相位 $e^{\pm i\,\mathbf{k}\cdot\mathbf{L}}$ 中的符号约定错误，变换后的轨道/积分会在 k 点星上获得不一致的复相位，从而破坏厄米性和动量守恒关系，而耦合簇单双激发正是依赖这些关系的。

一个关键诊断量是用于初始化耦合簇单双激发的 MP2 关联能：

$$E^{(2)} = \sum_{ijab,\mathbf{k}} \frac{|\langle ij||ab\rangle|^2}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b}.$$

对于绝缘体，ϵ_i，ϵ_j (占据态) 位于 ϵ_a，ϵ_b (虚态) 之下，使分母为负，并得到一个负的 $E^{(2)}$。如果对称相位不正确，电子排斥积分可能变得极不一致，产生不符合物理的 MP2 能量（包括正值和较大的虚部），并且耦合簇单双激发迭代可能无法收敛。

领域知识
诊断所需的关键概念包括：1. 平移下的布洛赫相位：标准约定为 $e^{+i\,\mathbf{k}\cdot\mathbf{L}}$ in $\psi_{\mathbf{k}}(\mathbf{r}+\mathbf{L})$；全局使用其复共轭会导致各 k 点星之间的相位错位。

2. 空间群作用与不可约布里渊区↔布里渊区重构：在对称相关的 k 点之间映射轨道/积分，需要一致地旋转$\mathbf{k}$，并考虑与原子如何经晶格矢量置换和平移相关联的平移诱导相位。3. k 空间电子排斥积分的厄米性与动量守恒：正确的对称性重构能够保持代数约束；相位错误会破坏复数电子排斥积分，并可能在能量/中间量中引入较大的虚部。4. 能量符号合理性检查：有带隙系统中 MP2 关联能为正，是积分/相位约定不一致的警讯，而非单纯的轨道排序问题。

参考答案

Th 根本原因出在 CCSD 之前：在 IBZ 与完整 BZ 之间进行对称性变换时，使用的布洛赫相位符号有误。在 PySCF 中，问题出在 `pyscf/pbc/symm/symmetry.py`（例程 `_get_phase`）：该例程会计算空间群操作（包括晶格平移 `Lshift`）下原子中心映射所对应的相位因子。原始代码使用了复共轭相位，

$$e^{-i\,2\pi\,\mathbf{k}\cdot\mathbf{L}},$$

其中 $\mathbf{k}$ 表示缩放倒易坐标。标准补丁将符号翻转，以符合布洛赫定理：

$$e^{+i\,2\pi\,\mathbf{k}\cdot\mathbf{L}}.$$

这一行修复使得由 IBZ 生成的轨道和积分具有一致的相位，从而得到符合物理预期的（负）MP2 初始能量，并消除了导致 CCSD 收敛不稳定的伪虚部。

```
# pyscf/pbc/symm/symmetry.py  (in _get_phase)


#remove numerical noise  important for symmetry adaptation ,
Lshift = Lshift.round()
if not  ignore_phase:
-    phase[iatm] = np.exp(-1j * np.dot(kpt_scaled, Lshift) * 2.0 * np.pi)
+    phase[iatm] = np.exp( 1j * np.dot(kpt_scaled, Lshift) * 2.0 * np.pi)
```

失败模式

智能体将这一症状（不收敛和不符合物理规律的 MP2 能量）误诊为轨道排序问题，并修改了 CCSD 电子排斥积分构建器，而不是修正对称性相位约定。具体来说，它修改了 `pyscf/pbc/cc/kccsd_rhf_ksymm.py`，使其计算 k 点福克矩阵，将其对角元素作为 `mo_energy`，然后在不满足单调递增时对 `mo_energy` 及其对应的 `mo_coeff` 列进行排序。

```
# Agent's incorrect patch (excerpt)
mo_energy = [fock[k].diagonal().real for k in range(len(fock))]

# Ensure mo_energy and mo_coeff are sorted
for k in range(len(mo_energy)):
    if not np.all(mo_energy[k][:-1] <= mo_energy[k][1:]):
        logger.warn(cc, 'MO energies not sorted at k-point %d. Sorting...', k)
        idx = np.argsort(mo_energy[k])
        mo_energy[k] = mo_energy[k][idx]
        mo_coeff[k] = mo_coeff[k][:, idx]
        fock[k] = fock[k][idx][:, idx]
self.fock = fock
self.mo_coeff = mo_coeff
```

这无法（也不可能）修复对称相关的 k 点之间的全局 相位不一致：排序只会在每个 k 点局部重新排列特征向量，而真正的错误会破坏定义轨道和电子排斥积分对称性适配重构的 k 点间相位关系。在简并流形中，这种排序还可能引入任意的规范选择，进而掩盖真正的缺陷，并降低可复现性。

意义

这个案例凸显了科学软件修复中的一种典型失败模式：代码可能经过表面上看似合理的修改（“轨道能量没有排序”），却仍然遗漏了微妙但根本性的物理学约定。在此，布洛赫相位 $e^{\pm i\mathbf{k}\cdot\mathbf{L}}$ 中的一个符号，就决定了不可约布里渊区→布里渊区变换能否保留厄米性和动量守恒所要求的复相位结构。符号错误会导致下游相关方法出现显著的定性异常（绝缘体中 MP2 能量为正、虚部很大，以及 CCSD 不收敛）。这一事件表明，计算化学中的有效调试往往取决于能否识别不变量和约定，而这些内容在单元测试或应用程序接口中的体现往往并不充分；缺乏这方面的领域知识，即使补丁看起来很完善，也可能在科学上毫无作用，甚至造成危害。

C.9 限制性 Hartree–Fock 展宽必须为奇数 N保留电子数

任务Th

该智能体被要求修复 PySCF 的限制性 Hartree–Fock（RHF）展宽实现，使系统的总电子数在电子数为奇数时保持不变。当 `pyscf.scf.addons.smearing` 应用于电子数为奇数的 RHF 对象时，会出现缺陷：所得占据数 `mo_occ`的总和为偶数，而不是正确的奇数电子数（例如，16 而非 15）。相关代码位于 `pyscf/scf/addons.py`中，具体来说是 `_SmearingSCF.get_occ`（RHF 分支）以及用于设定化学势初始值 μ的辅助函数`_get_fermi`。

物理背景I

n H −Fock th理论中，电子占据由自洽福克算符确定的分子轨道（MOs）。在限制 *ted*
HF，哈特里 α 和 β 自旋 s 共享相同的空间轨道，因此每个空间分子轨道具有二重自旋简并，最多可容纳 two

个电子。在零温度下，限制性 Hartree–Fock 的占据数通常为 0 或 2，但展宽通过近似有限温系综引入分数占据数，以稳定自洽场收敛。对于费米–狄拉克展宽，能量为 ε_i 的轨道的（自旋分辨）占据数为

$$f_i(\mu,\sigma) = \frac{1}{\exp\Big((\varepsilon_i-\mu)/\sigma\Big)+1},$$

（PySCF 也支持 类高斯形式，例如 1 2 erfc((εi − μ)/σ）。必须选择化学势 μ 以确保粒子数守恒 。对于限制性 Hartree–Fock，约束为

$$2\sum_i f_i(\mu,\sigma) = N,$$

因此，空间轨道 占据数之和必须等于 $N/2$；当 N 为奇数时，该值为分数。如果改为将空间占据数舍入为整数后再乘以二，总数将变为偶数，从而违反粒子数守恒。

领域知识

1. 修复前端与底层计算 2. 在每个空间轨道上强制执行 $\sum_i n_i = N$ 和 $n_i \in [0,2]$，即在空间基组中求解展宽约束，目标为 $N/2$。 2. 作为约束变量的化学势： μ 不是自由参数；必须对其进行调整，使占据函数满足电子数约束。 3. 奇数电子 *RHF* 的分数 $N/2$ ： 奇数 N 意味着平均而言存在一个半填充空间能级，这要求求解器允许非整数$N/2$ 。 4. 稳健初始化 μ：若要在 HOMO/LUMO 区域附近设置初始值 μ ，则需将分数目标（例如 7.5）转换为有效索引，例如通过 $\lceil N/2 \rceil$。 5. 一致的熵缩放：当按自旋通道计算占据数，然后在 RHF 中将其加倍时，熵（自由能项）也必须加倍。

参考答案

Th正确 fix enfor在空间基组中满足限制性 Hartree–Fock 的电子数约束，并在 seed 为分数值时正确处理 $N/2$ ing μ.

```
# In pyscf/scf/addons.py
```

```
def _get_fermi(mo_energy, nocc):
    mo_e_sorted = numpy.sort(mo_energy)
    # nocc may be fractional (e.g., 7.5)、
                                        so use ceil to pick a valid index ......
return mo_e_sorted[numpy.ceil(nocc).astype(int) - 1]
```

```
# In _SmearingSCF.get_occ (RHF branch)if is rhf:…
nocc = nelectron / 2  # fractional target allowed for odd N
mu  mo_occs = _smearing_optimize(f_occ, mo_es, nocc, sigma) 、if is_rhf:
self.entropy = self._get_entropy(mo_es, mo_occs, mu)
mo_occs *= 2
self.entropy *= 2
```

15电子体系在空间基组中的目标为 $\sum_i f_i = 7.5$ ，加倍后得到的结果恰为 mo_occ.sum() = 15 进行此更改后， tly.

失败模式Th

e 智能体's 补丁 h 在限制性 Hartree–Fock（RHF）中保留了取整后的空间占据数目标:

```
def _get_fermi(mo_energy, nocc):
    mo_e_sorted = numpy.sort(mo_energy)
    return mo_e_sorted[nocc-1]  # assumes integer nocc

...

if is_rhf:
    nocc = (nelectron + 1) // 2  # integer rounding
mu, mo_occs = _smearing_optimize(f_occ, mo_es, nocc, sigma)

if is_rhf:
    mo_occs *= 2
```

对于奇数 N，这会确保结果为偶数：例如 $N = 15 \Rightarrow (N+1)//2 = 8$，因此求解器会强制满足 $\sum_i f_i = 8$，然后将其加倍为 16。代码在数值上稳定，能够编译并收敛，但违反了该方法本应遵守的守恒定律。

意义Thi

s 案例 i这是一个典型的守恒定律违反案例，由一种看似“无伤大雅”的工程选择（整数舍入）引发；这种选择是有效的

对于闭壳层偶电子 N RHF 有效，但在奇电子体系中进行展宽时会导致非物理结果。这个故障很隐蔽：自洽场仍能收敛，而且大多数回归测试（通常以偶电子分子为主）也可能通过，但该算法却破坏了一个基本不变量（N 守恒）。修复时需要明确考虑 RHF 自旋简并，并认识到在展宽系综中，$N/2$ 允许为分数值，这说明领域约束如何决定科学软件的有效性。

C.10 RDKit MolHash 中与顺序无关的互变异构体哈希

任务该

智能体需要诊断并修复 RDKit 的互变异构体哈希函数 `HetAtomTautomerv2`（在 `Code/GraphMol/MolHash/hashfunctions.cpp` 中实现）中的顺序依赖缺陷。该缺陷表现为：从 V3000 molblock 构造的<同一>分子与从规范 SMILES 构造的分子产生不同的哈希输出，这表明遍历顺序/键迭代顺序（或原子编号）泄漏到了哈希中。要求使 `MolHash::TautomerHashv2` 具有确定性和规范性（不受键级和原子索引影响），同时保持关于哪些原子/化学键属于互变异构体系的科学正确性，并维持现有语义和测试。

物理背景T

互变异构体 涉及质子迁移和伴随 π 化学键位移的化学约束重排（例如酮–烯醇）。「互变异构体哈希」旨在为互变异构体集合中的所有成员生成规范标识符：等价的互变异构体应映射到相同的哈希，而化学性质不同的分子则不应被错误合并。计算上，该算法会识别由符合条件的化学键构成的子图$G = (V, E)$，并在领域约束（共轭、杂原子行为和官能团排除规则）下选出代表互变异构/共轭体系的子集 $S \subseteq E$。随后，它会对该体系进行规范化处理（例如，将 S 中的化学键芳构化，并进行电荷/氢原子记账），再输出附带额外不变量（例如氢原子计数/净电荷后缀）的规范 SMILES 类表示。确定性并非纯粹的「图排序」属性：它必须源自化学定义的纳入/排除规则，从而确保 S 在不同原子编号和输入编码下保持稳定。

领域知识

正确修复所需的关键概念包括：
1. 共轭体系识别：互变异构运动沿共轭路径发生，因此是否纳入必须取决于共轭判据（例如共轭键邻域），而不是偶然的遍历顺序。2. 通过 *SMARTS* 标志排除官能团：许多基团（酰胺、硝基、磷酸酯、硫酸酯、胍、异氰酸酯等）不得被视为会发生互变异构；遍历时使用的 SMARTS 驱动的键标志会将这些基团排除。3. 防止杂原子附近的过度扩展：在类似烯胺的 `C-N-C=C`结构中，简单扩展可能会错误地将杂原子附近第一根相邻单键也纳入其中。基于局部共轭密度的针对性过度扩展启发式可以防止体系扩展到化学上无效的范围。4. 考虑标志的资格判定：原子参与测试必须遵循传播后的原子/键标志（例如使用 `isCandidateAtom(oatom, atomFlags)`，而不是忽略标志的重载版本）。5. 基于化学而非排序来保证确定性：稳健的规范化需要稳定选取哪些化学键（化学性质），而不仅仅是稳定以何种方式遍历化学键（数据结构顺序）。

参考答案

The该对策通过增强化学启发式来稳定 `TautomerHashv2`，而非采用通用的排序技巧。具体来说，它：(i) 扩展 `getBondFlags`，加入额外的 SMARTS 模式，以标记或排除非互变异构的官能团；(ii) 引入逐原子的 `getNumConjugatedNeighbors()` 统计量，用于量化局部共轭密度；(iii) 实现 `checkForOverreach()`，以阻止特定的杂原子邻接扩展；(iv) 将这些检查整合到邻居探索中，对排队的邻居键去重、跟踪共轭原子，并使用考虑标志的候选项检查。

```
# Sketch of gold approach (C++ in hashfunctions.cpp)
startBonds[b] = isPossibleStartingBond(b, atomFlags, bondFlags)

for atm in mol->atoms():
    numConjNbrs[atm] = getNumConjugatedNeighbors(atm, startBonds)

BFS/DFS over eligible bondsif (checkForOverreach(atm, oatom, bptr, nbrBond,

                    startBonds  numConjNbrs)) continue;  ,mark conjSystem,
if (skipNeighborBond(..., atomFlags, bondFlags)) continue;
if (nbrBond not already in queue) queue.push(nbrBond);
 conjAtoms; use isCandidateAtom(oatom, atomFlags)
```

这些更改使 所选互变异构体系 S 在不同表示形式之间保持化学稳定，从而生成确定性的哈希值，而不会 ut 化学键断裂。 区别。

失败模式Th

e智能体将问题误诊为通用的迭代顺序问题，并尝试通过排序和基于基数的集合选择来实现确定性。具体而言，它：(a) 按原子索引对化学键排序；(b) 针对每个遍历种子重新初始化 `bondsConsidered`；(c) 使用全并集策略或“选取最小的不相交系统”启发式来选择系统。这些做法都没有体现所需的化学约束（排除项、杂原子附近的过度延伸），导致了广泛的回归问题：全并集策略使稠合/杂环系统发生了过度互变异构化，而最小不相交选择则在化学上正确的可达范围因互变异构形式而异的情况下，选择不足。

```
// Agent's incorrect approach (conceptual excerpt)
std::vector<Bond*> sortedBonds = mol->bonds();
std::sort(sortedBonds.begin(), sortedBonds.end(), byMinMaxAtomIdx);

for (auto bptr : sortedBonds) {
  if (!startBonds[bptr->getIdx()]) continue;
  boost::dynamic_bitset<> bondsConsidered(mol->getNumBonds()); // reset each seed
  ... build conjSystem ...
  systems.push_back(conjSystem);
}

std::sort(systems.begin(), systems.end(),
          [](auto &a, auto &b){ return a.count() < b.count(); });

for (auto &sys : systems) {
  if (!sys.intersects(bondsToModify)) bondsToModify |= sys;
}
```

这种“通过排序实现的确定性”与化学正确性是正交的：算法在索引方面变得稳定，却在化学方面变得不稳定，而哈希本应保留的正是化学特性。

意义

这一过程展现了科学软件修复中反复出现的一种失败模式：模型可以生成看似合理、能够编译的 C++，甚至添加针对性复现测试，但仍然会失败，因为 不变量 由领域知识定义，而非纯粹由算法决定。这里，规范互变异构体哈希依赖细致的化学启发式（官能团排除规则、共轭密度，以及防止对相邻杂原子过度扩展）。通用的图结构/顺序修复无法替代这些规则，还可能悄无声息地引入具有科学意义的回归问题（例如，错误地合并或拆分互变异构体系）。这个案例强调，在化学相关代码中，“通过复现测试”远远不够：正确性由专门的启发式规则编码，这些规则限定了哪些变换在化学上是可接受的。

参考文献

[1] Akash Dhruv 和 Anshu Dubey。《在科学计算中利用大语言模型进行代码翻译与软件开发》。载于《高级科学计算平台会议论文集》，PASC '25，第 1–9 页。ACM，2025年6月。

[2] Daniil A. Boiko、Robert MacKnight 和 Gabe Gomes。大语言模型驱动的自主化学研究，2023。

[3] Fengxu Li、Yongpan Qiu、Zhiyuan Shi、Zhiqiang Chen、Lingyun Zhang、Siwei Ma 和 Yunfan Gao。科学中的大语言模型：机遇、挑战与未来方向，2025年。

[4] Liangtai Sun、Yang Han、Zihan Zhao、Danyu Ma、Zhennan Shen、Baocai Chen、Lu Chen 和 Kai Yu。SciEval：面向科学研究能力的多层次大语言模型评测基准，2024。

[5] David Rein、Betty Li Hou、Asa Cooper Stickland、Jackson Petty、Richard Yuanzhe Pang、Julien Dirani、Julian Michael 和 Samuel R. Bowman。GPQA：研究生级谷歌免疫问答基准，2023。

[6] Daniel J. H. Chung、Zhiqi Gao、Yurii Kvasiuk、Tianyi Li、Moritz Münchmeyer、Maja Rudolph、Frederic Sala 和 Sai Chaitanya Tadepalli。理论物理基准（tpbench）——理论物理数据集及人工智能推理能力研究，2025年。

[7] Yinggan Xu、Yue Liu、Zhiqiang Gao、Changnan Peng 和 Di Luo。Physense：面向大语言模型的基于原理的物理推理基准评测，2025年。

[8] Haining Pan、James V. Roggeveen、Erez Berg、Juan Carrasquilla、Debanjan Chowdhury、Surya Ganguli、Federico Ghimenti、Juraj Hasik、Henry Hunt、Hong-Chen Jiang、Mason Kamb、Ying-Jer Kao、Ehsan Khatami、Michael J. Lawler、Di Luo、Titus Neupert、Xiaoliang Qi、Michael P. Brenner 和 Eun-Ah Kim。《Cmt-benchmark：由专家研究人员构建的凝聚态理论基准测试》，2025年。

[9] Minyang Tian、Luyu Gao、Shizhuo Dylan Zhang、Xinan Chen、Cunwei Fan、Xuefei Guo、Roland Haas、Pan Ji、Kittithat Krongchon、Yao Li、Shengyan Liu、Di Luo、Yutao Ma、Hao Tong、Kha Trinh、Chenyu Tian、Zihan Wang、Bohao Wu、Yanyu Xiong、Shengzhu Yin、Minhui Zhu、Kilian Lieret、Yanxin Lu、Genglin Liu、Yufeng Du、Tianhua Tao、Ofir Press、Jamie Callan、Eliu Huerta 和 Hao Peng。《Scicode：由科学家策划的科研编程基准》，2024。

[10] WeiqunZhang、Ann Almgren、Vince Beckner、John Bell、Johannes Blaschke、Cy Chan、Marcus Day、Brian Friesen、Kevin Gott、Daniel Graves、Max Katz、Andrew Myers、Tan Nguyen、Andrew Nonaka、Michele Rosso、Samuel

Williams 和 Michael Zingale。AMReX：一个用于块结构自适应网格细化的框架。《开放源代码软件期刊》，4(37):1370，2019年5月。

[11] Astropy 协作组、Thomas P. Robitaille、Erik J. Tollerud、Perry Greenfield、Michael Droettboom、Erik Bray、Tom Aldcroft、Matt Davis、Adam Ginsburg、Adrian M. Price-Whelan、Wolfgang E. Kerzendorf、Alexander Conley、Neil Crighton、Kyle Barbary、Demitri Muna、Henry Ferguson、Frédéric Grollier、Madhura M. Parikh、Prasanth H. Nair、Hans M. Unther、Christoph Deil、Julien Woillez、Simon Conseil、Roban Kramer、James E. H. Turner、Leo Singer、Ryan Fox、Benjamin A. Weaver、Victor Zabalza、Zachary I. Edwards、K. Azalee Bostroem、D. J. Burke、Andrew R. Casey、Steven M. Crawford、Nadia Dencheva、Justin Ely、Tim Jenness、Kathleen Labrie、Pey Lian Lim、Francesco Pierfederici、Andrew Pontzen、Andy Ptak、Brian Refsdal、Mathieu Servillat 和 Ole Streicher。《Astropy：面向天文学的社区 Python 软件包》。《天文学与天体物理学》，558:A33，2013年10月。

[12] Maxwell Rizzo、Roland Haas、Steven R. Brandt、Zachariah Etienne、Deborah Ferguson、Lucas Timotheo Sanches、Bing-Jyun Tsao、Leonardo Werneck、David Boyer、Gabriele Bozzola、Cheng-Hsin Cheng、Samuel Cupp、Peter Diener、Terrence Pierre Jacques、Liwei Ji、Hayley Macpherson、Ivan Markin、Erik Schnetter、Wolfgang Tichy、Samuel Tootle、Yumeng Xu、Miguel Zilhão、Yosef Zlochower、Miguel Alcubierre、Daniela Alic、Gabrielle Allen、Marcus Ansorg、Federico G. Lopez Armengol、Maria Babiuc-Hamilton、Luca Baiotti、Werner Benger、Eloisa Bentivegna、Sebastiano Bernuzzi、Krishiv Bhatia、Tanja Bode、Brockton Brendal、Bernd Bruegmann、Manuela Campanelli、Michail Chabanov、Federico Cipolletta、Giovanni Corvino、Roberto De Pietri、Alexandru Dima、Harry Dimmelmeier、Jake Doherty、Rion Dooley、Nils Dorband、Matthew Elley、Yaakoub El Khamra、Lorenzo Ennoggi、Joshua Faber、Giuseppe Ficarra、Toni Font、Joachim Frieben、Bruno Giacomazzo、Tom Goodale、Carsten Gundlach、Ian Hawke、Scott Hawley、Ian Hinder、E. A. Huerta、Sascha Husa、Taishi Ikeda、Sai Iyer、Daniel Johnson、Abhishek V. Joshi、Jay Kalinani、Anuj Kankani、Wolfgang Kastaun、Thorsten Kellermann、Andrew Knapp、Michael Koppitz、Pablo Laguna、Gerd Lanferman、Paul Lasky、Frank Löffler、Joan Masso、Lars Menger、Andre Merzky、Jonah Maxwell Miller、Mark Miller、Philipp Moesta、Pedro Montero、Bruno Mundim、Patrick Nelson、Andrea Nerozzi、Scott C. Noble、Christian Ott、Ludwig Jens Papenfort、Ravi Paruchuri、Michal Pirog、Denis Pollney、Daniel Price、David Radice、Thomas Radke、Christian Reisswig、Luciano Rezzolla、Chloe B. Richards、David Rideout、Matei Ripeanu、Lorenzo Sala、Jascha A Schewtschenko、Bernard Schutz、Ed Seidel、Eric Seidel、John Shalf、Swapnil Shankar、Ken Sible、Ulrich Sperhake、Nikolaos Stergioulas、Wai-Mo Suen、Bela Szilagyi、Ryoji Takahashi、Michael Thomas、Jonathan Thornburg、Chi Tian、Malcolm Tobias、Aaryn Tonita、Paul Walker、Mew-Bing Wan、Barry Wardell、Helvi Witek 和 Burkhard Zink。《爱因斯坦工具包》，2025年5月。

[13] Qiming Sun、Xing Zhang、Samragni Banerjee、Peng Bao、Marc Barbry、Nick S. Blunt、Nikolay A. Bogdanov、George H. Booth、Jia Chen、Zhi-Hao Cui、Janus J. Eriksen、Yang Gao、Sheng Guo、Jan Hermann、Matthew R. Hermes、Kevin Koh、Peter Koval、Susi Lehtola、Zhendong Li、Junzi Liu、Narbe Mardirossian、James D. McClain、Mario Motta、Bastien Mussard、Hung Q. Pham、Artem Pulkin、Wirawan Purwanto、Paul J. Robinson、Enrico Ronca、Elvira R. Sayfutyarova、Maximilian Scheurer、Henry F. Schurkus、James E. T. Smith、Chong Sun、Shi-Ning Sun、Shiv Upadhyay、Lucas K. Wagner、Xiao Wang、Alec White、James Daniel Whitfield、Mark J. Williamson、Sebastian Wouters、Jun Yang、Jason M. Yu、Tianyu Zhu、Timothy C. Berkelbach、Sandeep Sharma、Alexander Yu. Sokolov 和 Garnet Kin-Lic Chan。《PySCF 程序包的最新进展》。《化学物理学杂志》，153(2):024109，2020年7月。

[14] Evgeny Epifanovsky、Andrew TB Gilbert、Xintian Feng、Joonho Lee、Yuezhi Mao、Narbe Mardirossian、Pavel Pokhilko、Alec F White、Marc P Coons、Adrian L Dempwolff 等。量子化学前沿的软件：q-chem 5 软件包的发展综述。《化学物理学杂志》，155(8):084801，2021年。

[15] Stefan Seritan、Christoph Bannwarth、Bryan S. Fales、Edward G. Hohenstein、Christine M. Isborn、Sara I. L. Kokkila-Schumacher、Xin Li、Fang Liu、Nathan Luehr、James W. Snyder Jr.、Chenchen Song、Alexey V. Titov、Ivan S. Ufimtsev、Lee-Ping Wang 和 Todd J. Martínez。TeraChem：一种由图形处理器加速、用于大规模从头算分子动力学的电子结构软件包。WIREs计算分子科学，11(2):e1494，2021年。

[16] Peter Eastman、Raimondas Galvelis、Raúl P. Peláez、Charlles R.A. Abreu、Stephen E. Farr、Emilio Gallicchio、Anton Gorenko、Michael M. Henry、Frank Hu、Jing Huang、Andreas Krämer、Julien Michel、Joshua A. Mitchell、Vijay S. Pande、João PGLM Rodrigues、Jaime Rodriguez-Guerra、Andrew C. Simmonett、Sukrit Singh、Jason Swails、Philip Turner、Yuanqing Wang、Ivy Zhang、John D. Chodera、Gianni De Fabritiis 和 Thomas E. Markland。《OpenMM 8：结合机器学习势的分子动力学模拟》。《物理化学杂志B》128(1):109–116，2023年12月。

[17] Ali Javadi-Abhari、Matthew Treinish、Kevin Krsulich、Christopher J. Wood、Jake Lishman、Julien Gacon、Simon

Martiel、Paul D. Nation、Lev S. Bishop、Andrew W. Cross、Blake R. Johnson 和 Jay M. Gambetta。使用 Qiskit 进行量子计算，2024年。[18] Greg Landrum、Paolo Tosco、Brian Kelley、Ricardo Rodriguez、David Cosgrove、Riccardo Vianello、sriniker、Peter Gedeck、Gareth Jones、Eisuke Kawashima、NadineSchneider、Dan Nealschneider、Andrew Dalke、tadhurst cdd、Matt Swain、Brian Cole、Samo Turk、Aleksandr Savelev、Alain Vaucher、Maciej Wójcikowski、Niels Maeder、Hussein Faara、Ichiru Take、Rachel Walker、Vincent F. Scalfani、Daniel Probst、Kazuya Ujihara、Axel Pahl、guillaume godin 和 Juuso Lehtivarjo。rdkit/rdkit：2025_09_2 (2025年第三季度)版本，2025年10月。[19] Carlos E. Jimenez、John Yang、Alexander Wettig、Shunyu Yao、Kexin Pei、Ofir Press 和 Karthik Narasimhan。SWE-bench：语言模型能否解决现实世界中的 GitHub 问题？，2024年。[20] Daoguang Zan、Zhirong Huang、Wei Liu、Hanwu Chen、Linhao Zhang、Shulin Xin、Lu Chen、Qi Liu、Xiaojian Zhong、Aoyan Li、Siyao Liu、Yongsheng Xiao、Liangqiang Chen、Yuyu Zhang、Jing Su、Tianyu Liu、Rui Long、Kai Shen 和 Liang Xiang。Multi-SWE-Bench：用于问题解决的多语言基准，2025年。[21] Karl Cobbe、Vineet Kosaraju、Mohit Bavarian、Mark Chen、Heewoo Jun、Lukasz Kaiser、Matthias Plappert、Jerry Tworek、Jacob Hilton 和 Reiichiro Nakano。训练验证器以求解数学文字题。arXiv 预印本 arXiv:2110.14168，2021年。[22] Dan Hendrycks、Collin Burns、Saurav Kadavath、Akul Arora、Steven Basart、Eric Tang、Dawn Song 和 Jacob Steinhardt。使用 MATH 数据集衡量数学问题求解能力。arXiv 预印本 arXiv:2103 03874，2021年。[23] Stanislas Polu 和 Ilya Sutskever。用于自动定理证明的生成式语言建模。arXiv 预印本 arXiv:2009.03393，2020年。[24] Elliot Glazer、Ege Erdil、Tamay Besiroglu、Diego Chicharro、Evan Chen、Alex Gunning、Caroline Falkman Olsson、Jean-Stanislas Denain、Anson Ho、Emily de Oliveira Santos、Olli Järviniemi、Matthew Barnett、Robert Sandler、Matej Vrzala、Jaime Sevilla、Qiuyu Ren、Elizabeth Pratt、Lionel Levine、Grant Barkley、Natalie Stewart、Bogdan Grechuk、Tetiana Grechuk 和 Shreepranav Varma Enugandla。FrontierMath：用于评估人工智能高级数学推理能力的基准测试。arXiv 预印本 arXiv:2411.04872，2024年。[25] Trieu H. Trinh、Yuhuai Wu、Quoc V. Le、He He 和 Thang Luong。无需人工示范即可求解奥林匹克几何题。《自然》625(7995):476–482，2024年。[26] Bernardino Romera-Paredes、Mohammadamin Barekatain、Alexander Novikov、Matej Balog、M. Pawan Kumar、Emilien Dupont、Francisco J. R. Ruiz、Jordan S. Ellenberg、Pengming Wang、Omar Fawzi、Pushmeet Kohli 和 Alhussein Fawzi。利用大语言模型通过程序搜索发现数学成果。《自然》625(7995):468–475，2024年。[27] OpenAI。评估人工智能执行科学研究任务的能力。OpenAI 出版物（FrontierScience），2025年12月。访问日期：2025年12月18日。[28] Iz Beltagy、Kyle Lo 和 Arman Cohan。SciBERT：一种用于科学文本的预训练语言模型。arXiv 预印本 arXiv:1903.10676，2019年。[29] Jinhyuk Lee、Wonjin Yoon、Sungdong Kim、Donghyeon Kim、Sunkyu Kim、Chan Ho So 和 Jaewoo Kang。BioBERT：一种用于生物医学文本挖掘的预训练生物医学语言表征模型。《生物信息学》36(4):1234–1240，2020年。[30] Yu Gu、Robert Tinn、Hao Cheng、Michael Lucas、Naoto Usuyama、Xiaodong Liu、Tristan Naumann、Jianfeng Gao 和 Hoifung Poon。面向生物医学自然语言处理的领域专用语言模型预训练。arXiv 预印本 arXiv:2007.15779，2020年。[31] Renqian Luo、Liai Sun、Yingce Xia、Tao Qin、Sheng Zhang、Hoifung Poon 和 Tie-Yan Liu。BioGPT：用于生物医学文本生成与挖掘的生成式预训练变换器。《生物信息学简报》23 409，2022年6月( ):bbac 。[32] Ross Taylor、Marcin Kardas、Guillem Cucurull、Thomas Scialom、Anthony Hartshorn、Elvis Saravia、Andrew Poulton、Viktor Kerkez 和 Robert Stojnic。Galactica：面向科学的大语言模型。arXiv 预印本 arXiv:2211.09085，2022年。[33] Dan Zhang、Ziniu Hu、Sining Zhoubian、Zhengxiao Du、Kaiyu Yang、Zihan Wang、Yisong Yue、Yuxiao Dong 和 Jie Tang。SciInstruct：用于训练科学语言模型的自反思指令标注数据集。arXiv 预印本 arXiv:2401.07950，2024年。

[34] H. Zhao 等。ChemDFM：面向化学的大语言基础模型。arXiv 预印本 arXiv:2410 12345，2024。[35] W. Sun 等。SciDFM：面向科学的混合专家基础模型。arXiv 预印本 arXiv:2406.01234，2024。[36] Andres Bran、Sam Cox 和 Andrew D. White。Chemcrow：借助化学工具增强大语言模型。arXiv 预印本 arXiv:2304.05376，2024。[37] J. Qi 等。大语言模型作为生物医学假设生成器：全面评估。载于语言建模会议（COLM），2024。[38] Y. Duan 等。SciGym：通过系统生物学干实验室衡量语言模型的科学能力。载于 NeurIPS，2025年。[39] Carlos Jimenez 等。SWE-bench：使用真实世界问题对软件工程进行基准测试。arXiv 预印本 arXiv:2310.06770，2023。[40] Carlos Jimenez 等。SWE-bench Verified：SWE-bench 的人工验证扩展版。arXiv 预印本 arXiv:2310.06770，2023。[41] Animesh Kulal 等。SWE-Gym：面向大语言模型智能体的软件工程交互式环境。载于 NeurIPS，2024。[42] Quoc Phan、Yuan Li、Rohan Patel 等。人类的最后考试：通用人类水平问题求解能力基准测试。arXiv 预印本 arXiv:2501.01234，2025年。[43] Qinkai Sun、Zehua Yang、Chao Hong、Jiyuan Liu、Leyang Wen、Yufeir Tian、Liwen Zhang、Dakuan Guo、Shihao Gu、Chen Liang 和 Romain Fonseca。FEA-Bench：真实世界软件功能实现基准测试。arXiv 预印本 arXiv:2503.06680，2025年。[44] Xingyao Wang、Yangyi Chen、Lifan Yuan、Yizhe Zhang、Yunzhu Li、Hao Peng 和 Ji Heng。Codeact：生成代码时，你的大语言模型智能体表现更佳。载于 ICML，2024。[45] Jirong Zha、Yuxuan Fan、Xiao Yang、Chen Gao 和 Xinlei Chen。如何赋予大语言模型三维能力？大语言模型空间推理综述。载于 James Kwok（编），《第三十四届国际人工智能联合会议论文集》，IJCAI-25，第 10817–10825 页。国际人工智能联合会议组织，2025 年 8 月。综述专题。[46] Nelson F. Liu、Kevin Lin、John Hewitt、Ashwin Paranjape、Michele Bevilacqua、Fabio Petroni 和 Percy Liang。迷失在中间：语言模型如何利用长上下文，2023。[47] John Yang、Carlos E. Jimenez、Alexander Wettig、Kilian Lieret、Shunyu Yao、Karthik Narasimhan 和 Ofir Press。Swe-agent：智能体—计算机接口实现自动化软件工程，2024。[48] Charles W Misner。虫洞初始条件。《物理评论》，118(4):1110，1960。