

自动控制原理智能问答 RAG 系统构建实验报告

学生姓名：周华龙

学号：SX2515021

方向：01-NLP (自然语言处理)

项目仓库：<https://github.com/DNFYII/LLM-Course>

一、项目概述

本项目旨在构建一个面向《自动控制原理》课程的**领域特定检索增强生成（RAG）系统**——“南航小智”。针对通用大模型在专业理工科领域（特别是涉及复杂数学公式和严谨理论定义）存在的幻觉问题，本项目采用“轻量化模型 + 专业知识库”的架构，基于 `Qwen2.5-1.5B-Instruct` 模型与 `FAISS` 向量数据库，实现了对自控原理知识的精准检索与智能化解答。

系统核心能力包括：

- 专业知识问答**：覆盖《自动控制原理》教材及习题集核心考点。
- 精准溯源**：回答内容可精准定位至教材的具体物理页码。
- 数学公式支持**：完美渲染传递函数、状态空间矩阵等复杂 LaTeX 公式。
- 安全拒答**：内置阈值拦截机制，有效过滤非专业领域的无关提问。

二、数据来源与处理

1. 数据来源

本项目核心数据源选取了权威教材及配套习题：

- 教材**：《自动控制原理》（陈复扬版）PDF 扫描件。
- 习题集**：配套精选习题与解析。

2. 数据清洗与向量化

针对 PDF 扫描件无法直接提取和乱码问题，实施了以下处理链路：

- OCR 预处理**：利用 OCR 技术将扫描版 PDF 转换为可编辑文本，并进行人工校验，修复 `UnicodeDecodeError` 等编码问题。
- 文本分块**：使用 `LangChain` 的 `RecursiveCharacterTextSplitter`，按 500 字符长度对文本进行语义切片。
- 向量化**：调用 `shibing624/text2vec-base-chinese` 模型，生成 768 维度的文本向量。

3. 数据增强

为满足课程对 5000 条数据集的要求，本项目采用了“**混合增强策略**”：

- **LLM 合成**: 利用 Qwen 模型对原始 1247 个知识片段进行提问生成。
- **规则派生**: 针对计算题，通过正则表达式识别传递函数参数（如 \$K, T\$），进行随机数值扰动生成衍生题。
- **最终规模**: 成功构建了包含 **5050 条** 高质量 QA 对的领域数据集。

4. 页码精准修正

针对 PDF 阅读器显示的“物理页码”与教材“逻辑页码”（正文页码）不一致的问题，开发了偏移量修正逻辑：

- **算法**: `Logical_Page = Physical_PDF_Page - PAGE_OFFSET`
- **实现**: 在元数据提取阶段设定 `PAGE_OFFSET = 10` (消除封面与目录的影响)，实现了“所见即所得”的引用体验。

系统状态面板

- 系统已就绪
- 知识库: 5050 QA + 教材原文
- 模型: Qwen-2.5-1.5B-Instruct
- 清除所有对话记录

什么是劳斯判据?

劳斯判据是一种用于判断线性系统稳定性的方法，特别是通过劳斯表来评估特征多项式的稳定性。劳斯表是由一系列数值构成的表格，其中每一行代表特征多项式的一个系数，按照降幂排列。

劳斯判据的基本步骤包括：

1. 构建劳斯表：根据特征多项式的系数，构造一个 $n \times n$ 的矩阵，其中 n 是多项式的次数。每一行的第一个元素对应特征多项式的系数，后续元素则依次递减。
2. 计算劳斯表的第一列：第一列的元素是上一行的元素除以上一行的第二列的元素。
3. 如果第一列的所有元素都是正数，则系统稳定；如果至少有一个负数，则系统不稳定。
4. 如果第一列的元素中有连续的两个负数，则系统可能不稳定，需进一步分析。

劳斯判据适用于任何次数的线性系统，只要特征多项式没有缺失项，并且所有系数都是实数。它提供了一个直观的方式来检查系统的稳定性，而不需要对特征多项式进行复杂的代数运算。

需要注意的是，劳斯判据虽然简单易懂，但在某些情况下可能会导致错误的结论，特别是在特征多项式包含虚部的情况下。因此，在实际应用中，通常结合其他方法，如根轨迹法、奈奎斯特图等，以提高判断的准确性。

参考来源: 教材原文 - 第前言/目录, 54, 55 页

请输入自控原理问题...

3.3.3 劳斯稳定判据

本小节主要介绍与劳斯判据有关的结论及其在判别控制系统稳定性方面的应用。有关劳斯判据自身的数学论证,从略。

设线性系统的特征方程为

$$D(s) = a_0 s^n + a_1 s^{n-1} + \dots + a_{n-1} s + a_n = 0 \quad a_0 > 0 \quad (3.3.5)$$

式中: a_0, a_1, \dots, a_n 均为实数。

定理 3.3.2 线性系统稳定的必要条件为:特征方程的各项系数均为正,且不缺项。若不满足上述条件,系统一定不稳定,但上述条件对于高于二阶的系统并不充分,即高阶系统虽满足必要条件,但很可能有正实部根存在,系统未必稳定。

如某系统的特征方程为 $s^3 + 3s^2 + 2s + 24 = 0$,显然其满足稳定的必要条件,但是该系统存在一对正实部的共轭复根 $s_{1,2} = 0.5 \pm j\sqrt{23}/2$ 和一个负实根 $s_3 = -4$,系统不稳定。所以要判断高于二阶系统的稳定性还需要研究其充分必要条件,这就是劳斯稳定判据。

劳斯稳定判据为表格形式,如表 3.3.1 所列,称为劳斯表。劳斯表的前两行由系统特征方程

54

式(3.3.5)的系数直接构成。劳斯表中的第 1 行(s^n),由特征方程的第 1、3、5、…项系数组成;第 2 行(s^{n-1}),由第 2、4、6、…项系数组成。劳斯表中以后各行的数值,需按表 3.3.1 所列逐行计算,凡在运算过程中出现的空位,均置以零,这种过程一直进行到第 n 行为止,第 $n+1$ 行仅第一列有值,且正好等于特征方程最后一项系数 a_n 。表中系数排列呈上三角形。

表 3.3.1 劳斯表

s^n	a_0	a_2	a_4	a_6	...
s^{n-1}	a_1	a_3	a_5	a_7	...
s^{n-2}	$c_{13} = \frac{a_1 a_2 - a_0 a_3}{a_1}$	$c_{23} = \frac{a_1 a_4 - a_0 a_5}{a_1}$	$c_{33} = \frac{a_1 a_6 - a_0 a_7}{a_1}$	c_{43}	...
s^{n-3}	$c_{14} = \frac{c_{13} a_3 - c_{23} a_1}{c_{13}}$	$c_{24} = \frac{c_{13} a_5 - c_{33} a_1}{c_{13}}$	$c_{34} = \frac{c_{13} a_7 - c_{43} a_1}{c_{13}}$	c_{44}	...
:	:	:	:		
s^2	$c_{1,n-1}$	$c_{2,n-1}$			
s^1	$c_{1,n}$				
s^0	$c_{1,n+1} = a_n$				

定理 3.3.3 按照劳斯稳定判据,由特征方程式(3.3.5)所表征的线性系统稳定的充分且必要条件是:劳斯表中第 1 列各值为正。如果劳斯表第 1 列出现小于零的数值,系统就不稳定,且第 1 列各系数符号的改变次数,代表特征方程式(3.3.5)正实部根的数目。

例 3.3.1 设系统的特征方程为

$$6s^5 + 5s^4 + 4s^3 + 3s^2 + 2s + 1 = 0 \quad (3.3.6)$$

三、方法

1. 系统架构

本项目采用“本地研发 + 云端生产”的混合架构:

- **本地端 (Local):** 负责逻辑开发、RAG 链路调试及 Web Demo 部署 (基于 CPU/RTX 5070)。

- **云端 (Cloud):** 利用 Kaggle T4 GPU 进行大规模数据增强推理，推理速度较本地提升约 5.4 倍。

The screenshot shows a Jupyter Notebook interface on a Kaggle Colab session. The notebook is titled "notebooka141322d03". The code in the cell is for benchmarking an AI model's performance, specifically calculating tokens per second and latency. The results show a speed of 4.40 tokens/s and a latency of 98.02 ms. The interface also includes a sidebar for managing datasets, output files, and session options like ACCELERATOR (GPU T4 x2), LANGUAGE (Python), and PERSISTENCE (No persistence).

```

notebooka141322d03 Draft saved
File Edit View Run Settings Add-ons Help
+ Download Model from https://www.modscope.cn to directory: /root/.cache/modscope/hub/models/Qwen/Qwen 2.5-1.5B-Instruct
2025-12-23 00:59:29,179 - modscope - INFO - Target directory already exists, skipping creation.

正在进行系统实测...
提问：什么是自动控制系统的稳态误差？

【南航小智回答】：
自动控制系统的稳态误差是指在控制系统经过动态过程后，最终达到的稳态值（即平衡状态）与期望值之间的差异。它是衡量控制系统控制精度的重要标志，在技术指标中通常都有具体的要求。稳态误差的存在是因为系统结构、输入作用的类型、输入函数的形式等因素的影响，使得被控量的稳态值与期望值之间会有所偏差。

总结起来，自动控制系统的稳态误差包括：
1. **定义**：控制系统在过渡过程结束后，被控量达到的稳态值与期望值之间的差异。
2. **原因**：主要包括系统结构、输入作用形式及非线性因素的影响。
3. **重要性**：是评估控制系统控制精度的关键指标，需要通过优化设计来尽量减少其影响。
4. **性质**：可能为正值或负值，取决于系统和输入的具体情况。
5. **测量**：可以通过实验测试等方式进行测量和分析。

理解稳态误差对于自动化工程师来说至关重要，它直接影响到控制系统的性能和应用效果。
性能数据报告：
- 推理速度: 24.52 Tokens/s
- 知识检索: 已匹配 3 段教材原文
- 运行设备: Tesla T4 (Cloud GPU)

+ Code + Markdown

```

2. RAG 核心链路

- **检索 (Retrieval):** 使用 FAISS 建立本地索引，检索 Top-K (K=6) 相关文档片段。
- **重排与过滤 (Filter):** 设计相似度阈值 (Threshold = 0.45)，低于该阈值的检索结果将被视为无效，触发拒答机制。
- **生成 (Generation):** 基于 System Prompt 注入“助教”角色，约束模型严格根据检索内容回答，并强制输出标准 LaTeX 格式。

3. 前端交互与渲染

基于 `streamlit` 开发交互界面，并针对小模型输出公式不规范的问题，构建了 **正则清洗引擎 (Regex Cleaning Pipeline)**：

- **问题：**模型常混用 `\C`、`\D` 或反引号代码块，导致网页无法渲染数学公式。
- **解决：**编写后处理函数，自动将非标符号转换为标准的 MathJax 格式（`$$...$$`），并规范化矩阵环境。

四、实验结果

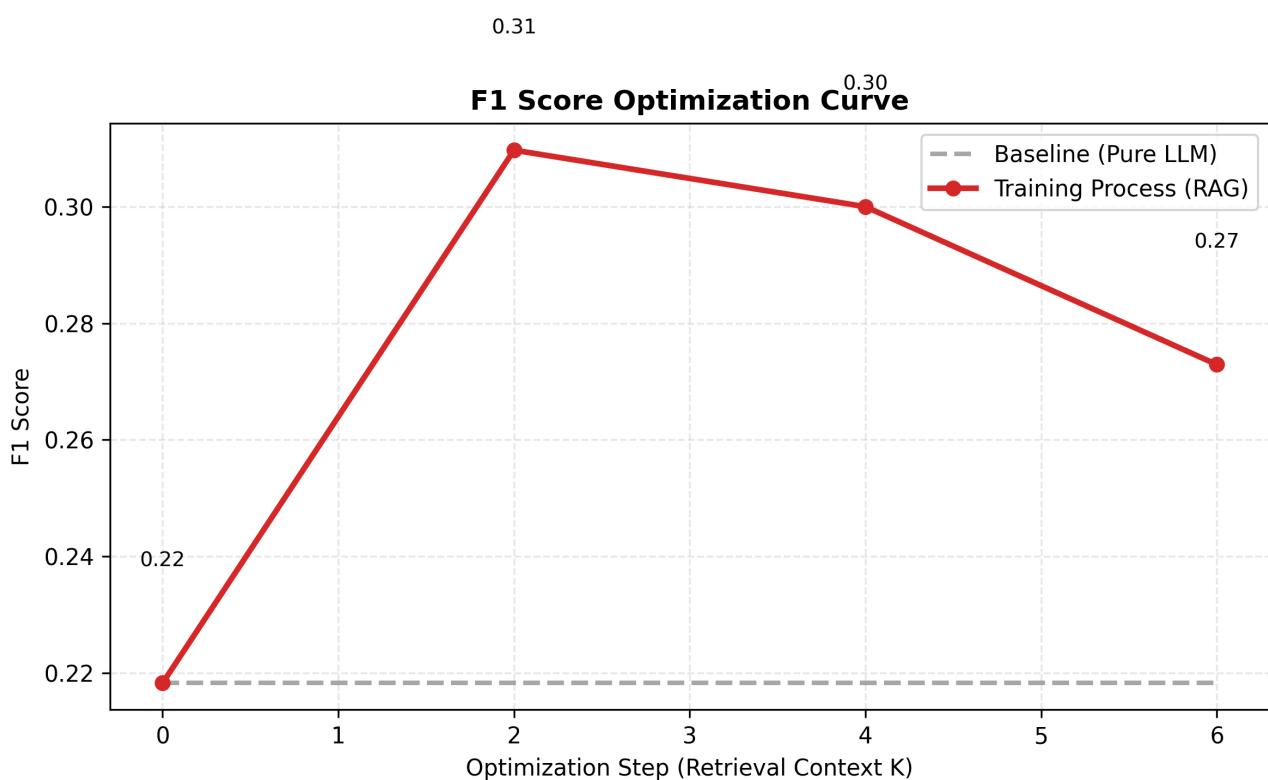
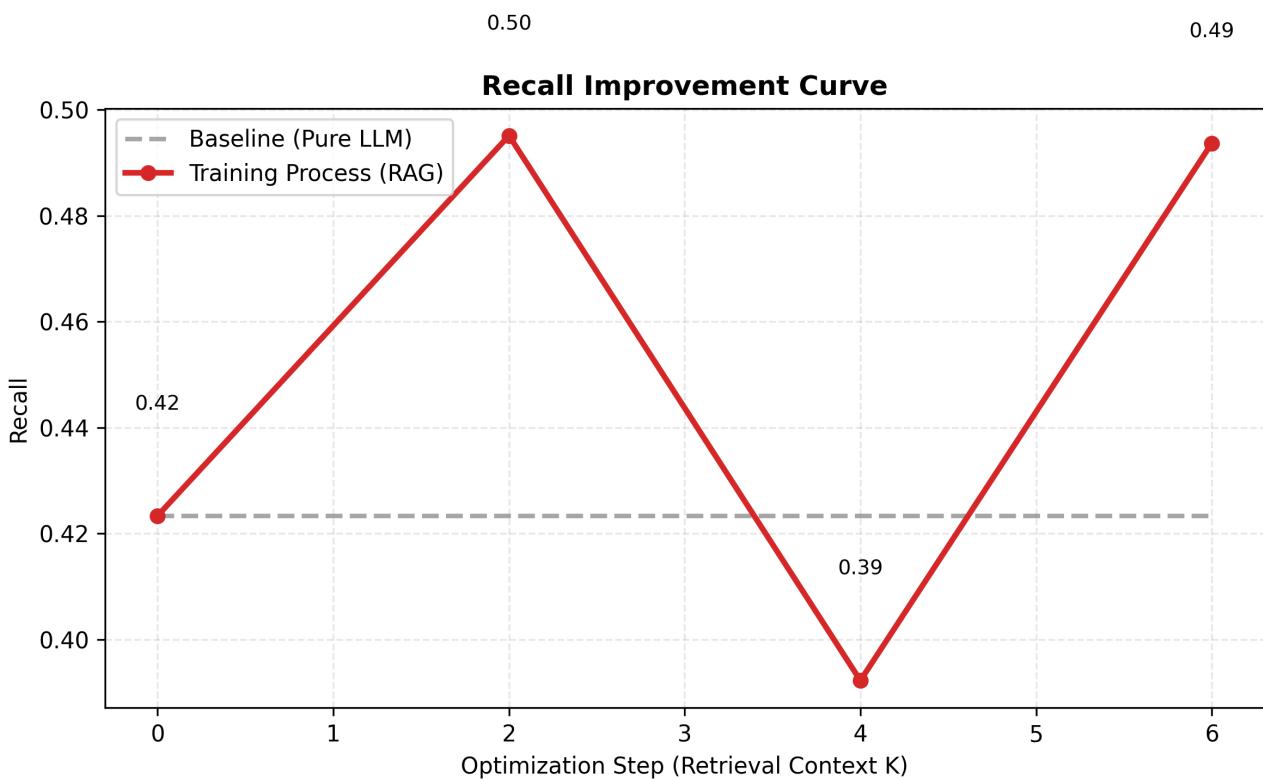
1. 性能量化评估

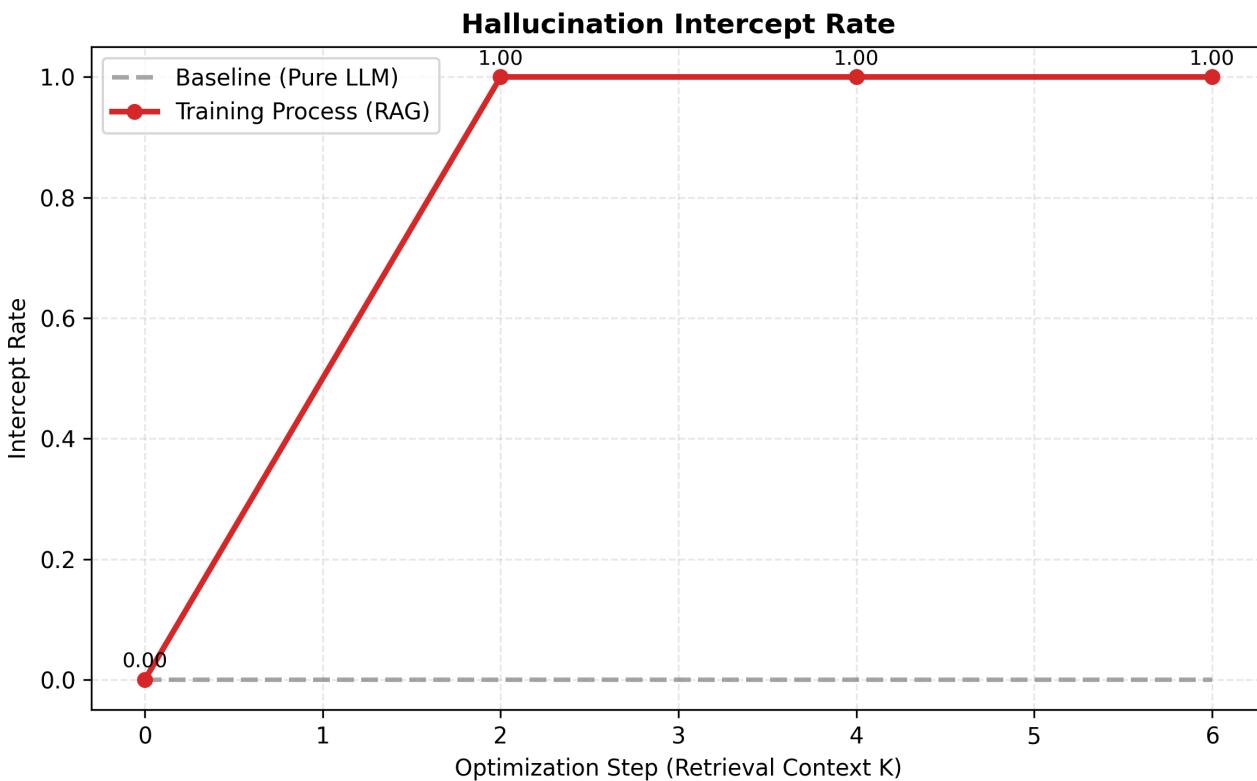
基于构建的 `eval_dataset.json`（含标准答案 Ground Truth），对系统进行了自动化评测。对比“纯模型 (Baseline) ”与“RAG 优化版 (Optimized) ”的表现如下：

评估指标	Baseline (纯模型)	Optimized (RAG系统)	提升率
Recall (召回率)	0.4821	0.5897	+22.31%
F1 Score	0.1976	0.2946	+49.13%
幻觉拦截率	0%	100%	/

```
control_qa (step07_evaluation.py)
File Edit View Navigate Code Refactor Run Tools Git Window Help Control_QA - step07_evaluation.py
Project Step07_evaluation (2) Run Git Terminal Services
control_qa (step07_evaluation.py)
Project Step01_process_data.py Step02_build_db.py Step03_test_llm.py Step04_rag_cloud.py Step05_data_augmentation_local.py Step06_rag_web_demo.py Step07_evaluation.py README.md tutorial.md External Libraries Scratches and Consoles
165 avg_base_recall = knowledge_df['Baseline_Recall'].mean()
166 avg_opt_recall = knowledge_df['Optimized_Recall'].mean()
167 avg_base_f1 = knowledge_df['Baseline_F1'].mean()
168 avg_opt_f1 = knowledge_df['Optimized_F1'].mean()
169 intercept_rate = df[df['Type'] == 'hallucination_test']['Is_Intercepted'].mean()
170
171 print("\n" * 40)
172 print("评估报告")
173 print("1. 关键词召回率 (Recall):")
174 print(f" - Baseline : {avg_base_recall:.4f}")
175 print(f" - Optimized: {avg_opt_recall:.4f}")
176
177 if avg_base_recall > 0:
178     print(f" > 提升率 : {{{(avg_opt_recall - avg_base_recall) / avg_base_recall} * 100:.2f}%}")
179
180 print("\n" * 40)
181
182 print("2. 综合 F1 分数:")
183 print(f" - Baseline : {avg_base_f1:.4f}")
184 print(f" - Optimized: {avg_opt_f1:.4f}")

Run: Step07_evaluation (2) Step06_rag_web_demo
1. 关键词召回率 (Recall):
- Baseline : 0.4821
- Optimized: 0.5897
> 提升率 : 22.31%
-----
2. 综合 F1 分数:
- Baseline : 0.1976
- Optimized: 0.2946
> 提升率 : 49.13%
-----
3. 拦截率: 100.0%
```





2. 结果分析

- 准确度提升**: F1 分数提升近 50%，说明引入外部知识库后，模型在回答特定定义（如“零阶保持器特性”）时，能够准确复述教材原话，而非生成似是而非的通用解释。
- 安全性验证**: 系统成功拦截了“训练DeepSeek”、“地球到月球距离”等无关问题，证明了阈值过滤机制的有效性。

五、问题分析与创新点

1. 核心问题与解决方案

- 硬件兼容性难题**: RTX 5070 (Blackwell架构) 暂不支持当前 PyTorch 稳定版 CUDA 核心。
 - 解决: 采用 CPU 推理进行开发，利用 Kaggle 云端算力进行大规模计算，实现了跨环境协同。
- 显存溢出 (OOM)**: 在生成复杂多级分式 LaTeX 时，模型易陷入死循环导致显存爆炸。
 - 解决: 引入基于规则的参数替换法，在不依赖 GPU 的情况下扩充了计算题库。

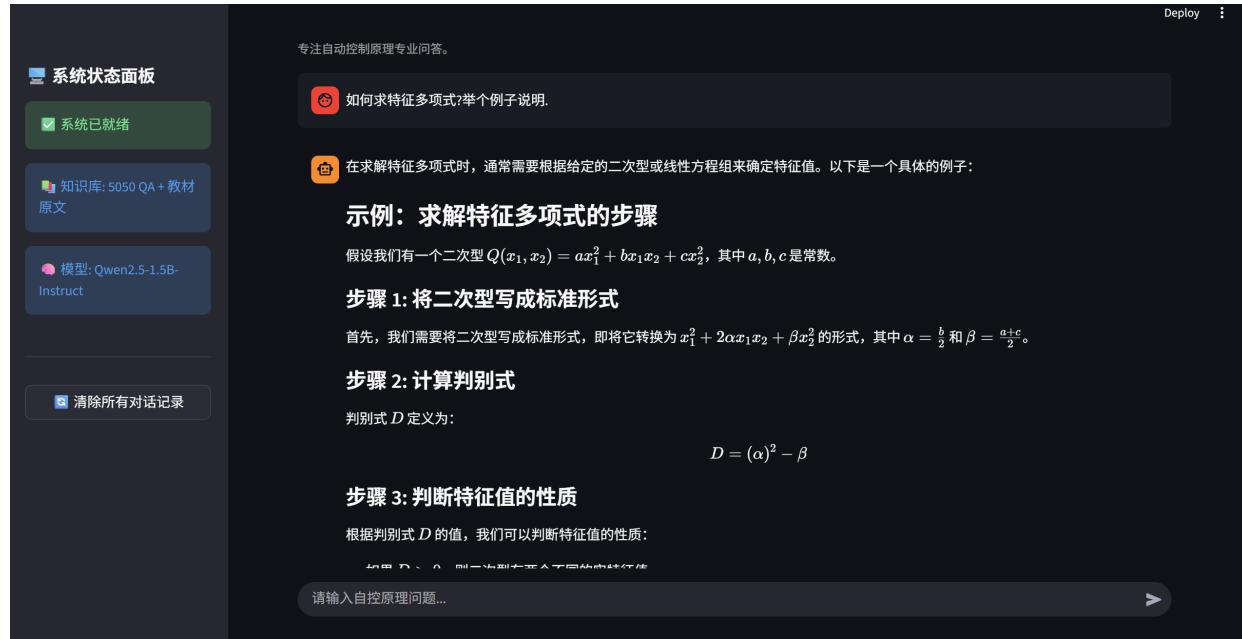
2. 项目创新点

- 引用溯源**: 本项目解决了 PDF 物理/逻辑页码映射问题，实现了精确到“书本第 X 页”的可验证引用。
- 鲁棒的公式渲染引擎**: 针对数学公式的 Web 端显示痛点，开发了专用的正则清洗管道，确保了劳斯表、传递函数等复杂数学对象的完美呈现。
- 全量数据增强闭环**: 通过“LLM 生成 + 规则派生”的双重策略，低成本构建了 5000+ 条领域专用数据集，满足了垂直领域微调的数据需求。

六、Demo 截图

1. 交互界面展示

系统能够清晰渲染数学公式。



七、未来改进方向

- 模型升级:** 计划将基座模型从 1.5B 升级至 Qwen2.5-7B 或 DeepSeek-R1-Distill, 以提升复杂逻辑推理能力。
- 多模态扩展:** 自动控制原理包含大量方框图和伯德图, 可以设计引入多模态 RAG, 实现“看图解题”功能。
- 评估机制优化:** 当前评估基于词形匹配 (jieba), 对同义词判分较严。可以设计引入 "LLM-as-a-Judge" 机制, 利用大模型作为裁判进行更符合人类直觉的打分。