



# 毕业论文(设计)

论文(设计)题目: AI 辅助计算物理实验

姓 名 赵世豪  
学 号 202100101066  
学 院 物理学院  
专 业 应用物理学  
年 级 2021  
指导教师 卢红旺

2025 年 5 月 26 日

## 摘要

由于现代教育对信息技术与学科教学相融合不断加深，在教学中应用计算机模拟、人工智能等方法也是当前的教学方向之一，现今大部分学校仍是用老师讲，同学们听，加课后作业的形式上课，这种教学方式效率太低，很难调动学生的积极性和主动性。针对这些教学问题，PT（Physics Tournament）育人模式便随之应运而生。PT育人的概念就是以探究性物理竞赛为牵引，而不是一般的传统型的奥赛，主要是实验探究类竞赛，在具体的物理问题上进行解决来提高自身的动手能力和思维能力。这就对学生的计算物理能力提出了更高要求。计算物理通过计算机对物理现象进行数值化处理，能够模拟实验过程、处理大量的实验数据，并确定近似的函数关系。通过数值计算和模拟，计算物理解决了许多解析理论研究无法解决的问题，检验了有效理论和近似成立的条件，并定量比较了理论与实验测量结果。在此背景下，为满足 PT 育人模式和计算物理实验课程教学要求，通过探究实现了一个 Python 编程的针对计算物理的实验展示模块以及基于 PyQt5 环境的人工智能(AI)辅助计算物理实验展示平台的设计与研发，该平台包括多个已实现的物理可视化实验供用户使用，并允许用户加入新的演示实验，同时具备 AI 功能，辅助解决数学、物理和代码方面的问题。

**关键词:**Python; 物理场可视化;PyQt5

## ABSTRACT

Due to the deepening integration of information technology and subject teaching in modern education, applying methods such as computer simulation and artificial intelligence in teaching has become one of the current teaching directions. However, most schools still adopt the traditional teaching approach where teachers lecture and students listen, supplemented by after-class homework. This teaching method is highly inefficient and fails to effectively stimulate students' enthusiasm and initiative. In response to these teaching issues, the PT (Physics Tournament) educational model has emerged. The concept of PT education is to use inquiry-based physics competitions as a driving force, rather than the conventional Olympiad-style competitions, mainly focusing on experimental exploration. It aims to enhance students' practical and thinking abilities by solving specific physical problems. This places higher demands on students' computational physics capabilities. Computational physics utilizes computers to numerically process physical phenomena, enabling the simulation of experimental processes, handling large amounts of experimental data, and determining approximate functional relationships. Through numerical calculation and simulation, computational physics has solved many problems that analytical theoretical research cannot address, verified the conditions under which effective theories and approximations hold, and quantitatively compared theoretical predictions with experimental measurements. Against this backdrop, to meet the requirements of the PT educational model and computational physics experimental courses, an experimental demonstration module for computational physics implemented in Python and an AI-assisted computational physics experimental demonstration platform based on the PyQt5 environment have been designed and developed through exploration. This platform includes multiple implemented physics visualization experiments for users to utilize and allows users to add new demonstration experiments. It also features AI capabilities to assist in solving mathematical, physical, and coding problems.

**Key Words:** Python, Visualization of physical fields, PyQt5

# 目 录

第一章 引言 .....	1
1.1 研究背景 .....	1
1.2 研究内容 .....	2
1.3 研究意义 .....	2
第二章 设计和实现 .....	4
2.1 整体框架搭建 .....	4
2.1.1 系统架构 .....	4
2.1.2 主界面设计 .....	4
2.1.3 实例选择对话框设计 .....	5
2.1.4 实例演示窗口展示 .....	6
2.2 物理实例加入过程（以加入多矩孔夫琅禾费衍射项目为例） .....	6
2.2.1 准备物理演示代码 .....	6
2.2.2 准备实例说明 pdf.....	6
2.2.3 将演示代码封装成类 .....	7
2.2.4 创建相应文件夹 .....	7
2.2.5 完善新项目与主程序的链接逻辑 .....	8
2.3 AI 交互部分搭建 .....	10
2.3.1 AI 交互整体架构 .....	10
2.3.2 交互流程 .....	10
2.3.3 Ollama 本地部署步骤 .....	11
2.3.4 详细搭建过程 .....	12
2.3.4.1 用户界面创建 .....	12
2.3.4.2 处理用户输入 .....	12
2.3.4.3 旋转动画创建 (表征 AI 处理中) .....	12
2.3.4.4 创建 AI 处理线程 .....	12
2.3.4.5 调用 AI (Ollama).....	13
2.3.4.6 显示 AI 返回结果 .....	13

第三章 实例分析-迈克尔逊干涉仪 .....	14
3.1 物理背景 .....	14
3.2 实现流程图 .....	16
3.3 具体实现过程 .....	16
3.4 实例意义 .....	17
第四章 总结与展望 .....	19
参考文献 .....	20
致    谢 .....	22
附    录 .....	23

# 第一章 引言

## 1.1 研究背景

随着时代的发展和科技的进步，在现代物理学的教学与科研中，计算机模拟物理问题和实验已成为理解和解决复杂物理问题的重要手段。物理方程的数值计算和具体的可视化技术的结合，使得物理实验的研究方法逐步从传统的实验室实验发展到多种实验方式并存。随着人工智能的快速发展，计算机编程方面的技术对物理学的研究影响愈发强烈，从 2024 年的诺贝尔物理学奖的颁发中不难看出物理学发展的变化。这种变化不仅提高了研究效率，还降低了实验成本，解决了一些传统实验难以实现的问题。同时人工智能还为物理教学提供了极有力的工具，降低了物理学的学习门槛<sup>[1]</sup>。

Python 语言以其丰富的数学和科学计算库（如 NumPy、SciPy、Matplotlib）和良好的可扩展性，在物理教学和研究中得到了广泛应用<sup>[2]</sup>。同时，PyQt5 作为一种基于 Qt 的 GUI 开发框架，内含许多丰富的便捷的 GUI 组件，为图形界面设计提供了强大的支持<sup>[3]</sup>。

物理学是十分重要的基础学科，在其他国家的限制和科技兴国战略的驱动下，物理学的发展近几年备受关注。国家对物理学教学和物理学人才培养的要求也越来越高。传统的物理学教学存在很大的局限性，特别是在实验教学、探究式学习方面存在很大的不足。基于此问题，不少物理学教授提出了 PT (physics tournament) 育人模式，鼓励学生参与物理实验以及各项物理竞赛，通过引入更多开放式实验和探究任务，培养学生的科学素养<sup>[4]</sup>。而在体验式学习中，物理学问题的模拟是重中之重，学生们需要通过模拟解决问题，也需要通过计算机模拟展示物理学过程<sup>[5]</sup>。

计算物理是物理学研究的重要工具。随着计算机科学和技术的迅速发展，计算物理已经成为物理学中不可或缺的分支。它通过数值计算和模拟来探索和解决物理问题，成为推动物理学研究范式的重要力量。计算物理不仅可以模拟实验过程、处理大量实验数据，还可以确定近似的函数关系，使得许多复杂的物理现象得以可视化、实时控制与分析。计算物理在解决复杂问题中具有显著优势。传统的计算方

法在面对大规模和复杂问题时显得力不从心，而高性能计算通过并行和分布式计算技术，能够解决传统计算机无法处理的复杂问题，从而更快地获取更准确的结果，提高科学的研究和工程设计的效率。计算物理在物理学中的应用非常广泛。它不仅可以用于理论物理和实验物理的研究，还可以应用于材料科学、生物物理、天体物理等多个领域。通过数值模拟和计算，科学家们能够预测新的物理现象、物理效应和物理规律，甚至开发新材料。例如，在材料科学中，计算物理可以帮助研究人员理解材料的微观结构与性能之间的关系，从而优化材料的设计和应用。

综合上述几点，为了有效增强计算物理课程的教学效果，提升学生的计算物理水平，促进教学模式向 PT 育人模式的转变，计算物理实验展示平台项目应运而生<sup>[6]</sup>。

## 1.2 研究内容

1. 使用 PyQt5 完成平台搭建、设计用户交互界面；基于 PyQt5 组件制作良好的人机交互界面，加入多款物理实验演示。
2. 采用 Ollama（本地 AI 模型）添加到平台，给平台加码。通过为平台集成 AI 交互功能，用户可以请教有关数学、物理和代码方面的相关问题，方便自己的学习。
3. 平台开放了学生的权限并提供了简单方便的接口来让学生自行加入到物理实验演示中来。

## 1.3 研究意义

传统物理实验必须在有物理设备的地方进行，对时间、空间也有很大的局限性；而计算机模拟则是无须任何物理装置即可完成实验，在时间、空间上又有了很大的提升，能够极大地提高实验操作性和教学灵活性<sup>[7][8]</sup>。

物理学教学过程中涉及大量经典的实验和物理现象，可见集中化展示、可视化的物理实验可以促进学生的理解，提高物理学整体的教学质量<sup>[9][10]</sup>。

平台融合了 AI 交互，可以提高学习效率。计算物理教学的难点在于代码实现。物理学学生对物理学公式以及数学推导理解较为深刻，但在编程方面的知识却比较薄弱。该平台项目的创新点在于引入 AI 交互，允许学生直接向 AI 提问，进

而解决编程方面的问题<sup>[11]</sup>。如“三体问题动态演示代码如何编写？”，“请生成一个计算  $\pi$  的 python 程序。”，这样，学生可以快速获得解答，提升学习效率。

顺应时代的潮流进步，在一定程度上可以增加计算物理的教学效果，还有利于学生去了解和认识其他的物理现象，而且也可以增强学生们计算机模拟物理问题的能力。作为一个学习辅助工具，本平台可以有效的推动物理教学模式朝着 PT 育人模式的方向去发展。

## 第二章 设计和实现

### 2.1 整体框架搭建

#### 2.1.1 系统架构

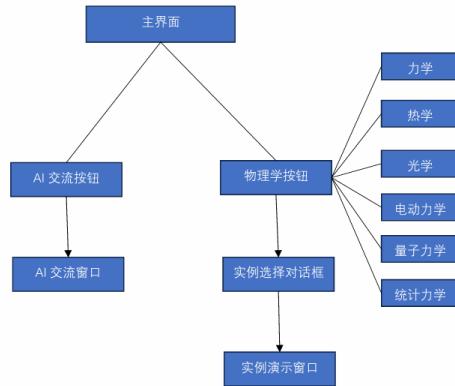


图 1 平台系统架构图

#### 2.1.2 主界面设计



图 2 平台主界面

PyQt5 是基于 Qt5 (Qt 是由 Qt Company 于 1991 年推出的跨平台 C++ 编写的图形用户界面应用程序开发框架, 可以创建高性能的桌面、移动或 Web 应用程序) 的一个 Python GUI 开发框架, 它提供了一系列的 GUI 控件 (widgets), 可以通过它来创建桌面应用程序。

其特点包括：1.跨平台运行：可以安装在 Windows, MacOS 以及 Linux 上使用；界面控件种类丰富：按钮(QPushButton)、标签(QLabel)、文本框(QTextEdit)、绘图窗口(QGraphicsView)；使用了信号槽(Signal&Slot)机制：可以响应用户的操作；可以与其他 Python 科学计算库配套使用，如 NumPy、Matplotlib、SciPy 等应用于物理实验的可视化<sup>[11]</sup>。

PyQt5 用作此平台上的物理实验展示平台图形用户界面以及实现实验模拟窗口的管理功能，可以实现多实验窗口的任意切换<sup>[12][13]</sup>

图 3 即为主界面。主界面采用 QMainWindow 作为主窗口，在其内加入 QWidget 作为中心部件，并采用 QVBoxLayout（垂直布局）摆放部件，使界面规整，相对紧凑合理。另外在界面上方加入一个 QLabel 组件，用来展示本程序的应用名“计算物理实验学习及优秀案例展示平台”，该名称所用字体是 Arial、字号为 38 并加粗显示，达到突出目的，在 Qt 的样式表样式表中设置为居中对齐，同时对其添加适当的内边距（padding），使其更具美观性。

页面中部用 create\_simulation\_buttons 方法创建 AI 交流入口按钮与多个模拟实验入口按钮，点击不同的按钮，将进入不同的物理实验仿真实验室或进入 AI 聊天窗口；按钮选用 QPushButton 控件，大小设置为 250\*80，保证有足够的点击面积<sup>[12]</sup>。用(QIcon)设置对应实验类别的图标，让用户一眼便知是什么样的实验。按顺序使用 VLayout 排列所有的实验按钮，统一采用圆角样式+交互效果（悬停变色，按下背景改），更有视觉的冲击力。

### 2.1.3 实例选择对话框设计

整体逻辑在 show\_simulation\_options 方法中实现。首先利用字典 simulation\_options 来确定主界面按钮与相关实例的对应关系，然后 show\_simulation\_options 方法接收 domain 作为参数，domain 表示模拟的领域类型，它从 self.simulation\_options[domain] 中获取该领域的所有选项。在点击某个按钮后，会创建一个 QInputDialog 对话框，对话框的大小为 500x500，标题为领域名称（例如：“选择力学物理模拟”）。使用 setComboBoxItems 方法将具体领域的选项填充到下拉列表中。最后调用 dialog.exec 方法得到返回结果，如果返回结果是 QtWidgets.QDialog.Rejected（即用户取消了对话框）或者选择的选项为

空字符串，则直接返回，不继续执行后续操作。如果用户选择了一个有效的选项，则调用 `self.add_simulation_frame(selected_option)` 方法来根据选中的选项添加模拟框架。

#### 2.1.4 实例演示窗口展示

整体逻辑由 `add_simulation_frame` 方法，`add_simulation_window` 方法和各种具体模拟框架的方法共同实现（如 `add_three_body_tab` 方法和 `add_movement_in_air_tab` 方法等）。首先是 `add_simulation_window` 方法，它接收两个参数：`frame_class`: 传入一个类（如 `three_body_frame`）用于生成具体的模拟框架，`tab_name`: 窗口的标题（例如 "三体运动"）。它还自动调整窗口的大小，并设置最小尺寸，以防窗口过小。最终创建一个新的 `QDialog` 窗口并展示模拟界面。接下来是 `add_simulation_frame` 方法，该方法接收一个 `selected_option` 参数，表示用户选择的模拟选项。它使用一个 `tab_mapping` 字典来映射每个选项到相应的模拟方法。如果选中的选项在字典中，则调用对应的方法（如 `add_three_body_tab` 方法和 `add_movement_in_air_tab` 方法等）都对应一个具体的模拟场景，并调用 `add_simulation_window` 方法来显示对应的框架和标题。

## 2.2 物理实例加入过程（以加入多矩孔夫琅禾费衍射项目为例）

### 2.2.1 准备物理演示代码

根据多矩孔夫琅禾费衍射的物理背景以及用户所需的交互功能，用 python 以及各种计算物理中所需的 python 库完成项目代码。完成后检验项目的正确性和稳定性。演示代码文件命名为 `multi_slit_fraunhofer_diffraction`。

### 2.2.2 准备实例说明 pdf

为了方便该项目加入平台后，其他用户能够在短时间内理解该项目，项目提供者需要撰写一个项目实例说明 pdf，其中应包括但不限于物理背景，求解方法，用户操作方式，项目实例意义等，完成后将该 pdf 加入程序文件夹 pdfs 里。

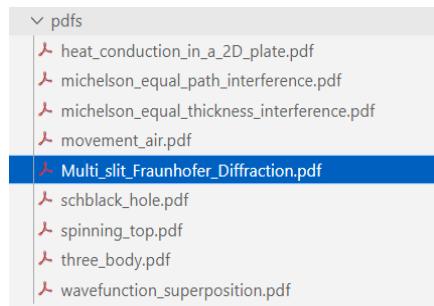


图 3 pdfs 文件夹展开

### 2.2.3 将演示代码封装成类

实例代码之前已经完成，并且通过了检验。现在只需将完整代码封装成主程序所需的类，方便主程序调用。按下图所示进行封装，在代码前部加入如图所示相关代码，类名命名规则为“文件名+frame”

```
class
multi_slit_fraunhofer_diffraction_frame(QMainWindow,
Ui_MainWindow):
    def __init__(self, parent=None):
        super().__init__()
```

### 2.2.4 创建相应文件夹

物理实例代码所需的修改已经全部完成，接下来创建新的文件夹放入实例代码文件，使得整个平台的代码组成结构清晰，实现效果如图 5 所示。

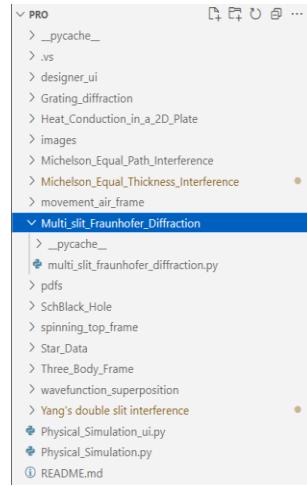


图 4 创建文件夹示例

### 2.2.5 完善新项目与主程序的链接逻辑

首先找到 Physical\_Simulation.py 文件，这是项目的主界面文件，整体运行逻辑都包含在这个文件中。

然后找到字典 simulation\_options，在对应的物理领域后面加入“多矩孔夫琅和费衍射”，效果如图 6 所示。

```
# 领域及选项配置
self.simulation_options = {
    "力学": ["三体运动", "抛体运动", "旋转运动"],
    "热学": ["二维板材导热展示(差分法)", "2", "3"],
    "光学": ["迈克尔逊干涉仪等倾干涉", "迈克尔逊干涉仪等厚干涉", "多矩孔夫琅和费衍射"],
    "电动力学": ["选项 1", "选项 2", "选项 3"],
    "量子力学": ["一维无限深势阱波函数态叠加", "选项 2", "选项 3"],
    "统计力学": ["史瓦西黑洞下的引力透镜效应", "选项 2", "选项 3"]
}
```

图 5 领域及选项配置

此时我们运行平台程序后，选择光学按钮后就会出现多矩孔夫琅和费衍射这个选项了，效果如图 7 所示。



图 6 选择演示

然 后 加 入 打 开 该 具 体 的 模 拟 框 架 方 法 add\_multi\_slit\_fraunhofer\_diffraction\_tab。该方法调用 add\_simulation\_window 方法创建多矩孔夫琅和费衍射的窗口，加入位置如图 8 所示。

```
Tabnine | Edit | Test | Explain | Document
def add_michelson_equal_thickness_interference_tab(self):
    self.add_simulation_window(michelson_equal_thickness_interference_frame, "迈克尔逊干涉仪等厚干涉")
Tabnine | Edit | Test | Explain | Document
def add_heat_conduction_in_a_2D_plate_tab(self):
    self.add_simulation_window(heat_conduction_in_a_2D_plate_frame, "二维板材导热展示（差分法）")
Tabnine | Edit | Test | Explain | Document
def add_multi_slit_fraunhofer_diffraction_tab(self):
    self.add_simulation_window(multi_slit_fraunhofer_diffraction_frame, "多矩孔夫琅和费衍射")
```

图 7 具体调用方法

接下来找到字典 tab\_mapping，将该方法与多矩孔夫琅和费衍射这个选项关联起来，关联后如图 9 所示。

```
tab_mapping = {
    "三体运动": self.add_three_body_tab,
    "抛体运动": self.add_movement_in_air_tab,
    "旋转运动": self.add_spinning_top_tab,
    "一维无限深势阱波函数态叠加": self.add_wavefunction_superposition_tab,
    "史瓦西黑洞下的引力透镜效应": self.add_schwarzschlack_hole_tab,
    "迈克尔逊干涉仪等厚干涉": self.add_michelson_equal_path_interference_tab,
    "迈克尔逊干涉仪等厚干涉": self.add_michelson_equal_thickness_interference_tab,
    "二维板材导热展示（差分法）": self.add_heat_conduction_in_a_2D_plate_tab,
    "多矩孔夫琅和费衍射": self.add_multi_slit_fraunhofer_diffraction_tab,
}
```

图 8 选项关联

关联后，当用户选择多矩孔夫琅和费衍射选项后，程序将调用该方法并创建相关窗口。至此，添加过程全部完成，运行后效果如图 10 所示。

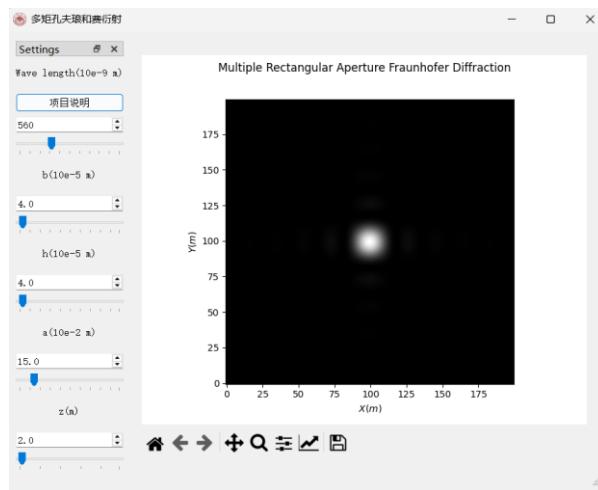


图 9 项目运行图

## 2.3 AI 交互部分搭建

### 2.3.1 AI 交互整体架构

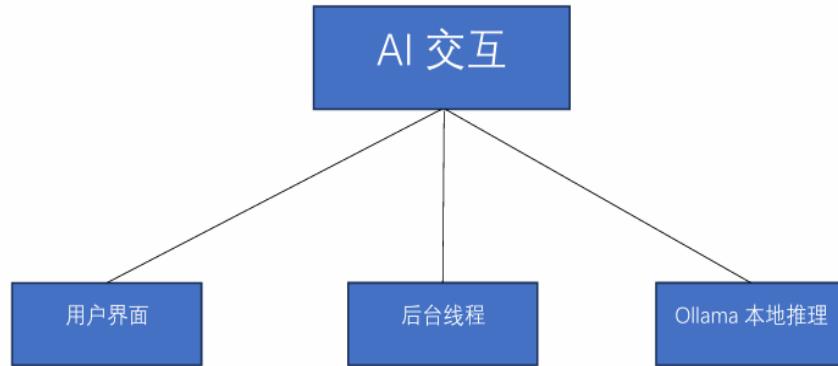


图 10 AI 部分架构

### 2.3.2 交互流程

表格 1 步骤组件对照表

步骤	实现组件	说明
用户输入问题	QTextEdit	实现输入框
点击“发送”	QPushButton	触发发送方法
显示用户输入	QTextBrowser	显示用户提问
显示加载动画	QTimer	展示旋转动画
启动 AI 线程	QThread	开启后台线程
调用本地 AI	subprocess.Popen	获取 AI 回复
获取 AI 结果	AIWorker.response_ready.emit()	线程返回结果
停止思考动画	QTimer.stop()	移除旋转动画
显示 AI 回复	QTextBrowser.append()	展示生成答案

### 2.3.3 Ollama 本地部署步骤

#### （1）安装 Ollama

如果尚未安装 Ollama，可以使用以下命令安装（适用于 Linux 和 MacOS）：

```
curl -fsSL https://ollama.com/install.sh | sh
```

Windows 用户可以从官方 Ollama 官网下载安装。

#### （2）下载 AI 模型

在终端运行以下命令，下载 LLaMA 3.2 Vision 模型：

```
ollama pull llama3.2-vision
```

#### （3）测试 AI 交互

在终端运行以下命令，启动具体模型

```
ollama run llama3.2-vision
```

当模型启动后，你可以输入：“请给我一个计算  $\pi$  的 python 程序”，模型会返回详细解释，具体如图 1 所示。

```
PS D:\llajiconda\proj> ollama run llama3.2-vision
>>> 你好！我是中文聊天机器人。有什么问题或话题想要和我讨论吗？
>>> 请给我一个计算π的python程序
以下是一个使用 Monte Carlo 方法计算 π 的 Python 程序。
```python
import random
import math

def calculate_pi(num_samples):
    inside_circle = 0
    for _ in range(num_samples):
        x = random.uniform(0, 1)
        y = random.uniform(0, 1)
        if x**2 + y**2 < 1:
            inside_circle += 1

    return (inside_circle / num_samples) * 4

num_samples = int(1e6) # 使用 1,000,000 个样本
pi_estimate = calculate_pi(num_samples)

print("估计的 π 值: {:.10f}".format(pi_estimate))
print("真实的 π 值: {:.10f}\n".format(math.pi))

# 计算误差
error = abs(pi_estimate - math.pi)
print("误差: {:.2f} %".format(error * 100))
```
这个程序使用 Monte Carlo 方法估计 π 的值。它将随机点生成在 [0,1]x[0,1] 区域内，并计算这些点中位于单位圆内的比例。由于面积比为 4:π，因此可以通过将该比例乘以 4 来获得 π 的估计值。
您可以调整 `num_samples` 变量来改变样本数量，从而影响估算的精度。
>>> ? send a message (/? for help)
```

图 11 模型启动演示图

### 2.3.4 详细搭建过程

#### 2.3.4.1 用户界面创建

首先创建一个对话框窗口，窗口命名为“AI 交流”，大小为  $700 \times 600$ 。

然后创建对话框窗口内部的众多组件。窗口大致分为三个区域：聊天显示区、用户输入区、发送按钮区。通过 QTextBrowser 组件实现聊天显示区。通过 QTextEdit 组件实现输入框，然后设置输入框内的提示句“请输入你的问题...”。为了窗口整体美观，设置输入框高度为 100。最后通过 QPushButton 组件添加“发送”按钮，并将其与 send\_message 方法相关联。

#### 2.3.4.2 处理用户输入

在用户输入具体问题并点击发送按钮后，会触发 send\_message 方法。该方法首先通过 toPlainText() 获取用户输入的问题，然后使用 append 方法将问题添加入聊天显示区，同时通过 clear 方法清空用户输入区，通过上述流程创造出用户成功发送问题的效果。接下来通过 timer.start 方法启动旋转圆圈提示动画，用来表示 AI 正在思考，同时通过 start\_ai\_worker 方法启动 AI Worker 线程。

#### 2.3.4.3 旋转动画创建（表征 AI 处理中）

在这一部分，我们需要每次取出一个不同的字符，循环显示，最终达到动态转圈动画的效果。首先，每次从旋转字符列表 spinner\_characters 中取出一个字符，接下来操作 QTextCursor 光标，先将光标移动到文本最后一段落的末尾，再往开头移动并选中整段文字，也就是说，现在选中了最后一行，最后使用 removeSelectedText 删除旧的提示，使用 insertText 插入新的旋转字符，最后更新一下计数器 thinking\_dots，用于下一次显示新的字符。

#### 2.3.4.4 创建 AI 处理线程

在 UI 中点击“发送”按钮后，不希望主线程（UI 线程）被阻塞，因为 AI 回复可能耗时较长，用户此时可以选择做其他工作。于是我在这一部分把 AI 的计算逻辑放进了一个 QThread 子类 AIWorker 中，从而让它在后台运行，执行完了再通过信号返回给主界面更新内容。首先定义两个信号，response\_ready：当 AI

回复完成后，发出信号，携带字符串结果。`error_occurred`: 如果发生异常，也通过信号发送错误信息。

然后创建 `run` 方法，该方法调用 `get_ai_response` 方法，如果成功，该方法最终会发出 `response_ready` 信号；如果出错，发出 `error_occurred` 信号。

#### 2.3.4.5 调用 AI (Ollama)

这是整个 AI 部分最重要的一环，在这一部分我们大体要实现四步：接收用户输入的提示语（`prompt`）和确定目前使用的 AI 模型（默认使用 `codellama`），通过调用 `ollama run <model>` 命令启动子进程，把 `prompt` 发送给这个进程，获取它的输出作为 AI 的回复。

接下来使用 `subprocess.Popen` 启动一个 Ollama 模型，然后把 `stdin`、`stdout`、`stderr` 都连接起来，便于交互，最后设置 `text=True` 和 `encoding="utf-8"`，进而保证我们读写的是字符串。

接下来与子进程通信。通过 `communicate` 把 `prompt` 传入 `stdin`，并读取 AI 模型返回的 `stdout` 和 `stderr`。

目前我们已经得到了 AI 回复，只需整理一下回复格式，完善一下异常处理，将回复正常返回调用者即可。

#### 2.3.4.6 显示 AI 返回结果

当 AI 线程中的回复结束之后会触发 `response_ready` 信号，并调用 `display_ai_response` 方法。再之后取消掉正在运行的旋转动画，并把这次 AI 的回复添加到聊天显示区域中去，整个 AI 交互就结束了。

## 第三章 实例分析-迈克尔逊干涉仪

### 3.1 物理背景

迈克尔逊干涉仪是 1881 年迈克尔逊为了研究“以太”是否存在而设计的。

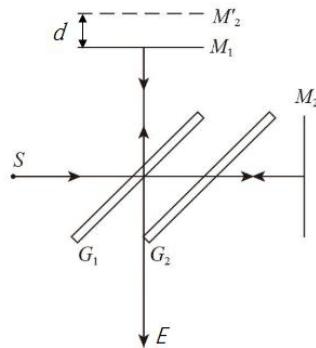


图 12 迈克尔逊干涉仪

如图所示,  $G_1$  和  $G_2$  是两块折射率和厚度都相同的平行平面玻璃板,  $G_1$  的背面有镀银或镀铝的半反射面, 称为分光板;  $G_2$  称为补光板。 $M_1$  和  $M_2$  是两块平面反射镜, 它们装在与  $G_1$  成  $45^\circ$  角的彼此互相垂直的两臂上。 $M_1$  固定不动,  $M_2$  可沿臂轴方向前后平移。

从光源  $S$  来的光, 经分光板  $G_1$  分成两部分, 它们分别入射在平面反射镜。 $M_1$  和  $M_2$  上。 $M_1$  经反射的光回到分光板后一部分透过分光板沿  $E$  方向传播, 而经  $M_2$  反射的光回到分光板后则是一部分被反射在  $E$  方向。由于两者是相干的, 在  $E$  处可观察到干涉条纹。

光束  $M_1$  和  $M_2$  上的反射相当于自距离为  $d$  的  $M_1$  和  $M'_2$  上的反射, 其中  $M'_2$  是平面镜  $M_2$  经分光板所成的虚像<sup>[14]</sup>。因此, 迈克尔逊干涉仪所产生的干涉与厚度为  $d$ 、没有多次反射的空气平行平面所产生的干涉完全一样。经  $M_2$  反射的光三次通过分光板, 而经  $M_1$  反射的光只通过分光板一次, 补偿板就是为消除这种不对称性而设置的<sup>[15]</sup>。

#### (1) 等倾干涉

当  $M_1$  和  $M_2$  严格垂直, 即  $M_1$  与  $M'_2$  严格平行时, 所得干涉为等倾干涉。干涉条纹为位于无限远或透镜焦平面上明暗的同心圆环。干涉圆环的特征是: 内疏

外密。由等倾干涉理论可知：当  $M_1$  与  $M'_2$  之间的距离  $d$  减小时，任意指定的区级条纹将缩小其半径，并逐渐收缩至中心处消失，即条纹“陷入”；当  $d$  增大，即条纹“外冒”。每“陷入”或“冒出”一个圆环， $d$  就相应增加或减少  $\lambda/2$  的距离。如果“陷入”或“冒出”的环数为  $N$ ， $d$  的改变量为  $\Delta d$ ，则

$$\Delta d = N\lambda/2 \quad (1)$$

因此，在实际应用中可以由已知波长  $\lambda$  求得  $M_2$  移动的距离，这就是利用干涉测长法；反之，若已知  $M_2$  移动的距离，可求得波长  $\lambda$ 。

根据光波的叠加原理，可得出迈克尔逊干涉仪的等倾干涉的光强分布为

$$I = I_0 \cos^2 \left[ \frac{2\pi d}{\lambda} \cos \left( \arctan \frac{r}{f} \right) \right] \quad (2)$$

$$r = \sqrt{x^2 + y^2} \quad (3)$$

实现效果如图 13。

## (2) 等厚干涉

当  $M_1$  与  $M_2$  两平面镜不完全垂直时，则由  $M_1$  与  $M'_2$  的平面构成一个劈形空气层，若  $d$  很小，则干涉条纹定域于所形成的空气层表面附近。干涉图样是由等距离分布的明暗相间的直条纹组成的。

实现效果如图 14。

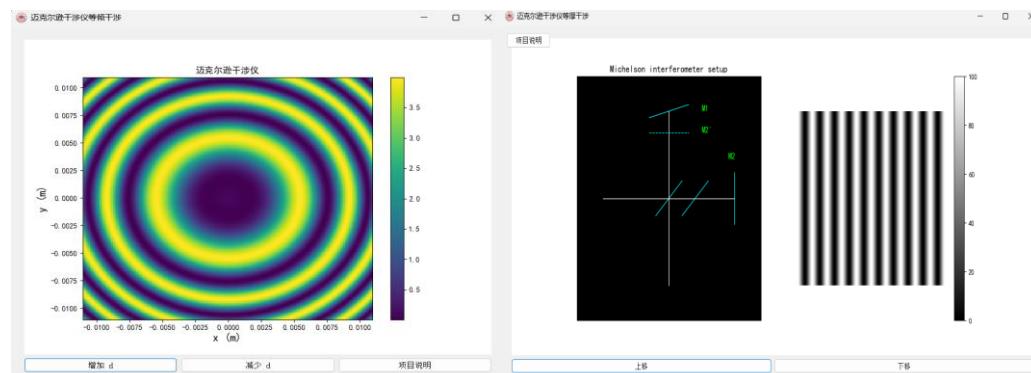


图 13 迈克尔逊等倾干涉

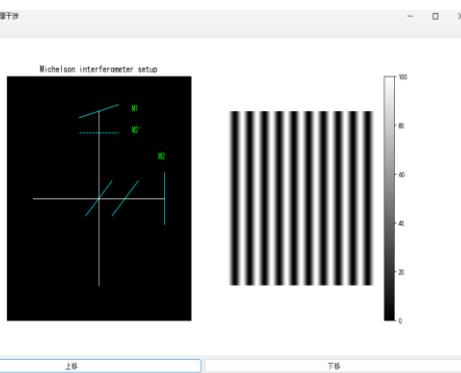


图 14 迈克尔逊等厚干涉

## 3.2 实现流程图

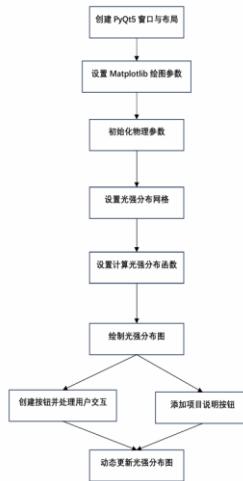


图 15 迈克尔逊干涉仪实现流程

## 3.3 具体实现过程

首先创建一个类，这样以后就方便集成到实验演示平台上。然后在 `__init__` 方法中，创建一个 PyQt5 窗口并设置标题、大小。接下来使用 Matplotlib 创建一个图形窗口 `fig` 和一个坐标轴 `ax`，`FigureCanvas` 是 Matplotlib 的画布，允许将图形嵌入到 PyQt5 窗口中。`QVBoxLayout` 是 PyQt5 中的一个重要的布局管理器类，用于实现垂直方向的布局，使用 `QVBoxLayout` 来管理窗口布局，最后将 `self.canvas` 添加到窗口中，完成布局设置。

因为实验演示平台将用于不同系统中，为了防止出现不兼容的问题，我根据不同操作系统（MacOS、Linux、Windows），设置不同的字体来确保中文能够正确显示。通过 `plt.rcParams` 设置 Matplotlib 的全局参数，避免不同系统上字体显示不一致。使用函数 `axes.unicode_minus` 设置为 `False` 以确保负号能够正常显示。

`self.f` 代表透镜的焦距，单位为米。`self.lambda_` 是入射光的波长，这里设置为 550 nm（可见光的波长）。`self.d` 是干涉仪中的空气薄膜初始厚度（即光程差的初始值），单位为米。

`xMax` 和 `yMax` 设置了坐标的最大范围，表示光强分布图的边界。使用 `np.meshgrid` 创建了一个二维坐标网格，覆盖了光强分布图的整个区域。`r` 是每个

坐标点到原点的径向距离，用于后续的光强计算。`self.Ir` 存储了根据初始光程差 `self.d` 计算出的光强分布。

`calculate_intensity` 函数使用了迈克尔逊干涉的基本公式来计算光强。使用余弦函数来计算每个点的光强值。返回的值 `Ir` 是一个二维数组，表示每个点的光强。

使用 `self.ax.pcolormesh` 绘制光强分布图。`self.x` 和 `self.y` 是坐标，`self.Ir` 是光强数据。`set_xlabel`, `set_ylabel`, `set_title` 分别设置坐标轴标签和标题。`fig.colorbar(self.H)` 为图像添加颜色条，用来表示光强的大小。

`self.button_increase`、`self.button_decrease` 和 `self.button_info` 分别是增加光程差、减少光程差和显示项目说明的按钮。每个按钮通过 `clicked.connect` 连接到相应的事件处理方法 (`increase_d`、`decrease_d` 和 `open_pdf`)。按钮被放置在一个 `QHBoxLayout` 布局中，水平排列。`open_pdf` 函数用于打开一个 PDF 文件，展示项目的详细说明。`QDesktopServices.openUrl` 用于调用系统默认程序打开本地文件，路径设置为 '`pdfs./michelson_equal_path_interference.pdf`'。

`increase_d` 和 `decrease_d` 分别增加和减少 `d`(光程差)的值。每次改变 `d` 后，调用 `calculate_intensity` 函数重新计算光强分布。`self.H.set_array(self.Ir.T.ravel())` 更新图像数据，`self.canvas.draw_idle` 方法刷新画布，使得图像实时更新。

### 3.4 实例意义

迈克尔逊干涉仪是经典的重要物理实验装置之一，是光学科目教学当中常用的仪器设备。模拟迈克尔逊干涉仪能够使学生更加直观地了解光波干涉中光程差对干涉条纹的影响等知识点，同时锻炼学生动手能力以及实验设计思维，非常适合于在物理实验课上开展此类活动，也符合普通物理实验课的教学优势。除此之外，在教学平台中，还可以让学生看到迈克尔逊干涉仪的源码，知道如何设计迈克尔逊干涉仪，并且对于不清楚的部分，可以通过 AI 分析帮助学生找到更合适的方法来研究这部分，提高学生的计算物理能力。对于计算物理本科教育以及传统教育向 PT 育人模式的转化意义重大<sup>[16]</sup>。

迈克尔逊干涉仪的实验是物理学本科教育中的一项非常重要的实验。由于它需要用到相当精密的仪器，实验步骤也较为繁杂，在实验过程中产生错误的可能性较大。因此若能通过模拟干涉仪实验，则会更好地弥补传统实验存在的不足，使学生获得更为准确的实验数据，并使学生能在实验数据的基础上发现和分析错误的原因<sup>[17]</sup>。

采用计算机模拟技术让学生通过模拟，直观感受到光程差不同，干涉条纹分布也不同，而且可随光程差变化而变化，还可看到如果改变光源、光强、狭缝宽度等因素对干涉条纹的影响；对于某些难以进行实验的干涉现象，如实验证明较难达到的杨氏双缝干涉，或者比较难以设置的薄膜干涉等情形下，可通过模拟更好地了解其形成的本质以及产生的原因，理解光的波动性和干涉原理，掌握正确处理实验中出现的各种偏差的方法。最后，对于如习题解答、计算机操作等具体问题的解决，需有一定的自学能力和解决能力，并且具有理论上的指导作用，有助于增强学生的思维能力，增强运用物理知识的能力，对整个大学学习起到了良好的带动作用<sup>[18]</sup>。

采用迈克尔逊干涉仪进行仿真，使学生有机会反思一些常见于物理实验中的失误。这些错误包括了误差，以及在选择实验器材的时候出现的问题等等。由于缺乏真实的实验设备，所以在仿真的时候也就不会受到相应的局限性，在出现问题以后也能够得到及时的反馈，并对此做出修改，有利于加深学生对于实验原理以及物理现象的理解。

## 第四章 总结与展望

本项目是基于 PyQt5 平台下的一个计算物理实验展示软件，并加入 AI 辅助。将传统计算物理实验与现代技术相融合，在平台上利用好智能分析的功能，让学生可以更快速的理解数学物理概念的同时，在编写代码方面也得到了很好的辅助作用，可以让学生的学习更加轻松以及提高效率。

基于 PyQt5 技术实现了一个计算物理实验演示平台，该平台可以方便地与用户交互并修改相应的参数，让其可以在动态变化的过程中看到实验的运行情况。并且平台中集成了 AI 助手，使用的是本地部署的 Ollama 系统，在线实时地为学生们解答数学类、物理类以及编程类的问题，让学生们更具有自主性地去探究问题，在解答问题的过程中不断培养自己思考问题、分析问题的能力，提高学习效率与学习氛围<sup>[19]</sup>。同时能够让学生们接触到现实中很难做到或很难看到的一些经典物理现象或者图像，利用代码去验证一些相关联的理论模型，来加深学生们对于一些理论的了解与掌握。

基于以上平台在计算物理实验方面的优点，并不是没有缺点与不足。继续增加平台对物理实验的覆盖范围，涉及力学、电磁学、热力学等相关物理实验；在当今 AI 日新月异的发展背景下，更新平台内部 AI 助手，提高其智能化水平，采用更加可靠、安全性更高的 AI 模型如 deepseek，在保持原有系统的前提下添加模型切换功能，将原本粗放式的思路逐步转变为精细化处理，进一步提高解答问题的准确率、快速度，特别是在面对复杂的物理问题时，更快、更精确的给出解决方案<sup>[20]</sup>。从平台现状上来看，一方面我们已经实现了基本的互动、可视等功能，但是在用户友好程度、操作便捷性方面还有较大提升的空间。未来可以进行相关的界面优化，也可以增加一些交互方式、定制化的内容等，提高用户使用体验。比如，以后平台上的物理例子多了，就可以加个搜索；人数多了就可以加上评论。此外，除了应用在物理学科教学和科研之外，该平台的设计理念也完全可以应用到数学、化学等领域，在各门学科的教学、科研当中都可以起到辅助的作用，因此也可以基于此平台再做延伸，比如开发其他的平台、功能，建设成跨多个学科领域的一个广泛的交流学习平台。

## 参考文献

- [1] 何红斌, 郑华, 张文超, 等. 计算机模拟辅助教学在分析力学中的应用[J]. 大学物理, 2024, 43(5): 33.
- [2] Tufino E, Oss S, Alemani M. Using Jupyter Notebooks to foster computational skills and professional practice in an introductory physics lab course[C]//Journal of Physics: Conference Series. IOP Publishing, 2025, 2950(1): 012022.
- [3] 梁楠, 王成喜, 张春飞, 等. 基于 Python 的多维度, 层次化的综合实验平台[J]. 吉林大学学报(信息科学版), 2023, 41(5): 858–865.
- [4] Dzioł D, Kwiatkowski L, Sokolowska D. Studying the impact of the class tournament as an assessment method on student achievements in physics courses[J]. arXiv preprint arXiv:1512.08663, 2015.
- [5] Borovský D, Hanč J, Hančová M. Scientific computing with open SageMath not only for physics education[J]. arXiv preprint arXiv:2308.07199, 2023.
- [6] 陆冰, 裴东. 计算机辅助教学在大学物理教学中的实践[J]. 物理, 2003, 32(01): 0–0.
- [7] Algorashi M K. The Effectiveness of Virtual Laboratories for Teaching Physics Experiments in a Saudi Secondary School[D]. University of Exeter (United Kingdom), 2024.
- [8] Negahban A. Simulation in engineering education: The transition from physical experimentation to digital immersive simulated environments[J]. Simulation, 2024, 100(7): 695–708.
- [9] Kabigting L D C. Computer simulation on teaching and learning of selected topics in physics[J]. European Journal of Interactive Multimedia and Education, 2021, 2(2): e02108.
- [10] Li Y, Ma L, Shi Y. Exploration on computer simulation method in physics education[C]//Education and educational technology. Springer Berlin Heidelberg, 2012: 517–522.
- [11] Henze J, Bresges A, Becker-Genschow S. AI-Supported Data Analysis Boosts Student Motivation and Reduces Stress in Physics Education[J]. arXiv preprint arXiv:2412.20951, 2024.
- [12] Fitzpatrick M. Create GUI Applications with Python & Qt5 (PyQt5 Edition): The hands-on guide to making apps with Python[M]. Martin Fitzpatrick, 2020.
- [13] Meier B A. Python GUI Programming Cookbook[M]. Packt Publishing Ltd, 2017.
- [14] Summerfield M. Rapid GUI programming with Python and Qt: the definitive guide to PyQt programming[M]. Pearson Education, 2007.
- [15] Restivo M T, Villate J, Almeida F, et al. The Michelson Interferometer: a learning object[J]. Proceedings of REV, 2007.
- [16] Alka M, Akib I, Bancong H. Developing a Virtual Reality Simulation of the Michelson–Morley Experiment for Physics Education: Design, Validation, and

- Educational[J]. International Journal of Learning, Teaching and Educational Research, 2024, 23(11): 254-267.
- [17]黄绍书. 迈克尔逊干涉仪及其干涉中的几个问题[J]. Modern Physics, 2022, 12: 50.
- [18]崔芬萍, 江一帆, 裴世鑫. 基于迈克尔逊干涉仪的波长计设计与实现[J]. 实验科学与技术, 2022, 20(1): 22-27.
- [19]姜涛, 孙艳, 于华民. 人工智能在大学物理教学中的应用[J]. Creative Education Studies, 2024, 12: 423.
- [20]赵沛, 杨卫. AI 时代的力学教学[J]. 力学与实践, 2025, 47(1): 9-14.

## 致 谢

光阴似箭，岁月如梭。转眼间大学四年的学习生涯就要结束了，在毕业论文完成之际，本科生涯也即将结束，借此机会，我想对我的母校，我的老师，我的同学表达由衷的感谢。

首先，感谢山东大学为我们提供良好的学习环境，让我们畅游在知识的海洋中，努力地充实、提高自己。其次，感谢我的论文指导老师卢红旺老师对我的悉心指导，从论文选题到结构，从观点推敲到语序检查，都凝聚着他的心血，卢红旺老师不仅在学业上教会我们很多，而且以他兢兢业业的学习、工作态度感染了我们每一位山大学子。另外感谢大学四年中的任课老师学习期间对我的严格要求，感谢他们对我学习上和生活上的帮助，使我了解了许多专业知识和为人的道理。最后，还要感谢同学们的帮助，同学们的计算物理实验的结课设计为这个项目作出了很大贡献，郑浩学长的设计贡献十分突出，再次感谢大家！

## 附录

主程序代码：

```
from PyQt5 import QtWidgets
from PyQt5.QtCore import Qt, QThread, pyqtSignal, QTimer, QSize
from PyQt5.QtGui import QFont, QTextCursor, QIcon
import subprocess
import re
import sys

sys.stdout.reconfigure(encoding='utf-8')

# 导入其他模块
from Physical_Simulation_ui import Ui_Simulation
from Three_Body_Frame.Three_Body_Frame import three_body_frame
from movement_air_frame.movement_air_frame import movement_air_frame
from spinning_top_frame.spinning_top_frame import spinning_top_frame
from wavefunction_superposition.wavefunction_superposition import
wavefunction_superposition_frame
from SchBlack_Hole.SchBlack_Hole_ui import schblack_hole_frame
from Michelson_Equal_Path_Interference.Michelson_Equal_Path_Interference
import michelson_equal_path_interference_frame
from
Michelson_Equal_Thickness_Interference.Michelson_Equal_Thickness_Interference
import michelson_equal_thickness_interference_frame
from Heat_Conduction_in_a_2D_Plate.Heat_Conduction_in_a_2D_Plate import
heat_conduction_in_a_2D_plate_frame
from Multi_slit_Fraunhofer_Diffraction.multi_slit_fraunhofer_diffraction import
multi_slit_fraunhofer_diffraction_frame
```

```

def clean_ansi_codes(text):
    ansi_escape = re.compile(r'\x1B(?:[@-Z\\-_]|[[0-?]*[-/]*[@-~]])')
    text = ansi_escape.sub("", text)

    # 移除可能的进度指示符（如 :::: 等）
    spinner_chars = re.compile(r'[:::.:.:.:.:.]')
    text = spinner_chars.sub("", text)

    return text.strip()

# 调用 ollama 处理 AI 回复
def get_ai_response(prompt, model="codellama:latest"):
    try:
        # 使用 subprocess 调用 ollama run
        process = subprocess.Popen(
            ["ollama", "run", model],
            stdin=subprocess.PIPE,
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            text=True, # 确保文本模式
            encoding="utf-8" # 设置编码为 UTF-8
        )

        # 将用户输入传递给命令行工具，同时捕获 stdout 和 stderr
        stdout, stderr = process.communicate(input=prompt)

        # 清理 stderr 的控制字符并移除多余空格
        stderr_cleaned = clean_ansi_codes(stderr).strip()
    
```

```
if stderr_cleaned: # 仅在 stderr 有有效内容时打印错误信息
    print("错误信息:", stderr_cleaned)

# 清理 stdout 的控制字符并返回
return clean_ansi_codes(stdout.strip())

except UnicodeDecodeError as e:
    print("编码错误:", e)
    return "AI 回复时发生编码错误"

except Exception as e:
    print("发生错误:", e)
    return "AI 生成失败"

# AI 交流窗口类
class AIChatDialog(QtWidgets.QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setWindowTitle("AI 交流")
        self.setMinimumSize(700, 600)

        # 布局
        layout = QtWidgets.QVBoxLayout(self)

        # 聊天显示区域
        self.chat_display = QtWidgets.QTextBrowser(self)
        layout.addWidget(self.chat_display)

        # 用户输入区域
        self.input_field = QtWidgets.QTextEdit(self)
```

```
self.input_field.setPlaceholderText("请输入你的问题...")
self.input_field.setFixedHeight(100)
layout.addWidget(self.input_field)

# 发送按钮
self.send_button = QtWidgets.QPushButton("发送", self)
self.send_button.clicked.connect(self.send_message)
layout.addWidget(self.send_button)

# 创建 AI 处理线程
self.ai_worker = None

# 动态显示旋转的圆圈
self.spinner_characters = ['.', '..', '...', '...', '...', '...', '...', '...', '...', '...'] #

旋转动画
self.thinking_dots = 0
self.timer = QTimer(self)
self.timer.timeout.connect(self.update_thinking_message)

def send_message(self):
    user_message = self.input_field.toPlainText().strip()
    if user_message:
        # 显示用户消息
        self.chat_display.append(f"<b>你:</b> {user_message}")
        self.input_field.clear()

        # 显示旋转的圆圈提示
        self.chat_display.append("") # 初始为空行
```

```
# 启动定时器进行动态显示旋转的圆圈
self.thinking_dots = 0
self.timer.start(200) # 每 200 毫秒更新一次

# 启动 AI Worker 线程
self.start_ai_worker(user_message)

def update_thinking_message(self):
    try:
        # 使用旋转动画的字符
        spinner = self.spinner_characters[self.thinking_dots % len(self.spinner_characters)]

        # 获取文本游标并更新文本
        cursor = self.chat_display.textCursor()

        # 移动到最后一行的开始位置
        cursor.movePosition(QTextCursor.EndOfBlock)
        cursor.movePosition(QTextCursor.StartOfBlock,
                           QTextCursor.KeepAnchor)

        # 删除上一行的内容，并插入新的字符
        cursor.removeSelectedText() # 清除上一行的内容
        cursor.insertText(f'{spinner}') # 插入当前的旋转字符

        self.thinking_dots += 1
    except Exception as e:
```

```
print(f"动态更新错误: {e}") # 捕获动态更新中的任何错误

def start_ai_worker(self, user_message):
    self.ai_worker = AIWorker(user_message)
    self.ai_worker.response_ready.connect(self.display_ai_response)
    self.ai_worker.error_occurred.connect(self.display_error)
    self.ai_worker.start()

def display_ai_response(self, response):
    try:
        # 停止定时器并清理旋转的圆圈提示
        self.timer.stop()

        # 删除旋转的圆圈提示并显示 AI 回复
        cursor = self.chat_display.textCursor()
        cursor.movePosition(QTextCursor.StartOfBlock)
        cursor.movePosition(QTextCursor.Down,
QTextCursor.KeepAnchor)
        cursor.removeSelectedText()

        self.chat_display.append(f"<b>AI:</b> {response}")

    except Exception as e:
        print(f"显示响应错误: {e}")

def display_error(self, error_message):
    try:
        # 停止定时器并清理旋转的圆圈提示
        self.timer.stop()
```

```
# 删除旋转的圆圈提示并显示错误信息
cursor = self.chat_display.textCursor()
cursor.movePosition(QTextCursor.StartOfBlock)
cursor.movePosition(QTextCursor.Down,
QTextCursor.KeepAnchor)
cursor.removeSelectedText()

self.chat_display.append(f"<b>AI:</b> 出现错误：
{error_message}")

except Exception as e:
    print(f"显示错误信息时发生问题: {e}")

def closeEvent(self, event):
    # 确保关闭时正确清理和终止
    if self.ai_worker and self.ai_worker.isRunning():
        self.ai_worker.terminate()
    event.accept()

# AI 处理线程类
class AIWorker(QThread):
    response_ready = pyqtSignal(str)  # AI 回复信号
    error_occurred = pyqtSignal(str)  # 错误信号

    def __init__(self, user_message):
        super().__init__()
        self.user_message = user_message
```

```

def run(self):
    try:
        # 获取 AI 回复
        response = get_ai_response(self.user_message)
        self.response_ready.emit(response)
    except Exception as e:
        self.error_occurred.emit(str(e))

class SimulationApp(QtWidgets.QMainWindow, Ui_Simulation):
    def __init__(self):
        super().__init__()

        # 领域及选项配置
        self.simulation_options = {
            "力学": ["三体运动", "抛体运动", "旋转运动"],
            "热学": ["二维板材导热展示（差分法）", "2", "3"],
            "光学": ["迈克尔逊干涉仪等倾干涉", "迈克尔逊干涉仪等厚干涉", "多矩孔夫琅和费衍射"],
            "电动力学": ["选项 1", "选项 2", "选项 3"],
            "量子力学": ["一维无限深势阱波函数态叠加", "选项 2", "选项 3"],
            "统计力学": ["史瓦西黑洞下的引力透镜效应", "选项 2", "选项 3"]
        }

    self.setupUi(self) # 先调用 UI 设置

```

```
# 设置中央小部件
self.central_widget = QtWidgets.QWidget(self)
self.setCentralWidget(self.central_widget)

# 创建一个垂直布局
main_layout = QtWidgets.QVBoxLayout(self.central_widget)

# 创建标题部分，显示“计算物理实验展示平台”
self.title_label = QtWidgets.QLabel("计算物理实验学习及优秀案例
展示平台", self)
self.title_label.setAlignment(Qt.AlignCenter) # 居中显示
font = QFont("Arial", 38, QFont.Bold) # 设置字体为 Arial，大小为
24，加粗
self.title_label.setFont(font)
self.title_label.setStyleSheet("color: #000; padding: 200px;") # 设置
字体颜色和内边距
main_layout.addWidget(self.title_label)
# 设置大小策略
self.title_label.setSizePolicy(QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Expanding)

# 创建一个垂直间隔，保证标题和按钮之间的空间
main_layout.addSpacerItem(QtWidgets.QSpacerItem(20, 40,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding))

# 创建按钮并添加到布局
self.create_simulation_buttons(main_layout)
```

```
# 创建“AI 交流”按钮
ai_button = QtWidgets.QPushButton("AI 交流", self.central_widget)
ai_button.setMinimumSize(250, 80)
ai_button.setText("AI 交流")
ai_button.setToolTip("点击与 AI 进行交流")
ai_button.clicked.connect(self.open_ai_chat)
ai_button.setStyleSheet("""
    QPushButton {
        border-radius: 15px;
        background-color: #f0f0f0;
        font-size: 16px;
        color: #333;
        border: 2px solid #B0B0B0;
        padding: 10px;
    }
    QPushButton:hover {
        background-color: #e0e0e0;
    }
    QPushButton:pressed {
        background-color: #c0c0c0;
    }
""")
ai_button.setIcon(QIcon("./images./cpu.png"))
ai_button.setIconSize(QSize(47, 47))
main_layout.addWidget(ai_button)

# 设置窗口大小
self.setGeometry(500, 100, 1000, 900) # 设置窗口的大小和位置
```

```

def create_simulation_buttons(self, layout):
    domain_icons = {
        "力学": "images./mechanics_icon.png",
        "热学": "images./thermodynamics_icon.png",
        "光学": "images./optics_icon.png",
        "电动力学": "images./electrodynamics_icon.png",
        "量子力学": "images./quantum_icon.png",
        "统计力学": "images./statistics_icon.png"
    }

    for domain in self.simulation_options.keys():
        button = QtWidgets.QPushButton(self.central_widget)
        button.setMinimumSize(250, 80)
        button.setText(domain)
        button.setToolTip(f"点击选择{domain}模拟")
        icon_path = domain_icons.get(domain,
                                     "path/to/default_icon.png") # 获取领域的图标路径
        icon = QIcon(icon_path)
        button.setIcon(icon)
        button.setIconSize(QSize(40, 40)) # 设置图标大小
        button.clicked.connect(lambda _, d=domain:
                               self.show_simulation_options(d))
        button.setStyleSheet("""
            QPushButton {
                border-radius: 15px;
                background-color: #f0f0f0;
                font-size: 16px;
            }
        """)

```

```
color: #333;  
border: 2px solid #B0B0B0;  
padding: 10px;  
}  
  
QPushButton:hover {  
    background-color: #e0e0e0;  
}  
  
QPushButton:pressed {  
    background-color: #c0c0c0;  
}  
""")  
layout.addWidget(button)  
  
  
def show_simulation_options(self, domain):  
    options = self.simulation_options[domain]  
    dialog = QtWidgets.QInputDialog(self.central_widget)  
    dialog.resize(500, 500)  
    dialog.setWindowTitle(f"选择{domain}模拟")  
    dialog.setLabelText(f"请选择{domain}模拟")  
    dialog.setComboBoxItems(options)  
    dialog.setWindowFlags(dialog.windowFlags() &  
~Qt.WindowContextHelpButtonHint)  
    result = dialog.exec_()  
    selected_option = dialog.textValue()  
    if result == QtWidgets.QDialog.Rejected or selected_option == "":  
        return  
    self.add_simulation_frame(selected_option)
```

```

def add_simulation_frame(self, selected_option):
    tab_mapping = {
        "三体运动": self.add_three_body_tab,
        "抛体运动": self.add_movement_in_air_tab,
        "旋转运动": self.add_spinning_top_tab,
        "一维无限深势阱波函数态叠加": self.add_wavefunction_superposition_tab,
        "史瓦西黑洞下的引力透镜效应": self.add_schblack_hole_tab,
        "迈克尔逊干涉仪等倾干涉": self.add_michelson_equal_path_interference_tab,
        "迈克尔逊干涉仪等厚干涉": self.add_michelson_equal_thickness_interference_tab,
        "二维板材导热展示（差分法）": self.add_heat_conduction_in_a_2D_plate_tab,
        "多矩孔夫琅和费衍射": self.add_multi_slit_fraunhofer_diffraction_tab,
    }

    if selected_option in tab_mapping:
        tab_mapping[selected_option]()

```

```

def add_simulation_window(self, frame_class, tab_name):
    # Create a new QDialog window for the simulation
    dialog = QtWidgets.QDialog(self)
    dialog.setWindowTitle(tab_name)

    dialog.adjustSize() # 自动调整大小为内容的大小

```

```
# 设置最小尺寸以避免过小
dialog.setMinimumSize(900, 700)
dialog.setWindowFlags(dialog.windowFlags() | Qt.WindowMinMaxButtonsHint)
dialog.setWindowFlags(dialog.windowFlags() & ~Qt.WindowContextHelpButtonHint) # 移除问号按钮

# Create the simulation frame and add it to the dialog
frame = frame_class(dialog)
layout = QtWidgets.QVBoxLayout(dialog)
layout.addWidget(frame)

dialog.show() # Show the dialog window

# 这些方法用于打开具体的模拟框架
def add_three_body_tab(self):
    self.add_simulation_window(three_body_frame, "三体运动")

def add_movement_in_air_tab(self):
    self.add_simulation_window(movement_air_frame, "抛体运动")

def add_spinning_top_tab(self):
    self.add_simulation_window(spinning_top_frame, "旋转运动")

def add_wavefunction_superposition_tab(self):
    self.add_simulation_window(wavefunction_superposition_frame, "一维无限深势阱波函数态叠加")
```

```
def add_schblack_hole_tab(self):
    self.add_simulation_window(schblack_hole_frame, "史瓦西黑洞下的
引力透镜效应")

def add_michelson_equal_path_interference_tab(self):
    self.add_simulation_window(michelson_equal_path_interference_fram
e, "迈克尔逊干涉仪等倾干涉")

def add_michelson_equal_thickness_interference_tab(self):
    self.add_simulation_window(michelson_equal_thickness_interference_
frame, "迈克尔逊干涉仪等厚干涉")

def add_heat_conduction_in_a_2D_plate_tab(self):
    self.add_simulation_window(heat_conduction_in_a_2D_plate_frame, "
二维板材导热展示（差分法）")

def add_multi_slit_fraunhofer_diffraction_tab(self):
    self.add_simulation_window(multi_slit_fraunhofer_diffraction_frame, "
多矩孔夫琅和费衍射")

def open_ai_chat(self):
    # 打开 AI 聊天窗口并启动后台进程
    self.chat_dialog = AIChatDialog(self)
    self.chat_dialog.show()

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    main_window = SimulationApp()
    main_window.show()
```

```
    sys.exit(app.exec_())

迈克尔逊干涉仪代码:

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Button
from PyQt5 import QtWidgets
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as
FigureCanvas
import sys
import platform
from PyQt5.QtGui import QDesktopServices
from PyQt5.QtCore import QUrl

class michelson_equal_path_interference_frame(QtWidgets.QWidget):
    def __init__(self, parent=None):
        super(michelson_equal_path_interference_frame, self).__init__(parent)

        # 设置 PyQt5 窗口
        self.setWindowTitle('迈克尔逊干涉仪模拟')
        self.setGeometry(100, 100, 800, 600)

        # 创建 Matplotlib 图像
        self.fig, self.ax = plt.subplots(figsize=(8, 6))
        self.canvas = FigureCanvas(self.fig)
        layout = QtWidgets.QVBoxLayout()
        layout.addWidget(self.canvas)
        self.setLayout(layout)
```

```
# 设置参数  
# 检查操作系统  
os_name = platform.system()  
if os_name == 'Darwin': # MacOS  
    plt.rcParams['font.sans-serif'] = ['Hei']  
elif os_name == 'Linux': # Linux  
    plt.rcParams['font.sans-serif'] = ['SimHei']  
elif os_name == 'Windows': # Windows  
    plt.rcParams['font.sans-serif'] = ['SimHei'] # 或者 'Microsoft  
YaHei'  
else:  
    print("Unsupported operating system")  
  
plt.rcParams['axes.unicode_minus'] = False  
  
self.f = 0.2 # 透镜的焦距 (单位: m)  
self.lambda_ = 550e-9 # 入射光波长 (单位: m)  
self.d = 0.4e-3 # 空气薄膜的初始厚度 (单位: m)  
  
# 根据公式求出光强分布  
self.xMax = 20000 * self.lambda_ # x 取值范围约为[-11mm,11mm]  
self.yMax = 20000 * self.lambda_ # y 取值范围约为[-11mm,11mm]  
self.step = 200 * self.lambda_  
  
self.x, self.y = np.meshgrid(np.arange(-self.xMax, self.xMax, self.step),  
                            np.arange(-self.yMax, self.yMax,  
self.step)) # 建立坐标网格  
self.r = np.sqrt(self.x**2 + self.y**2)
```

```
# 初始光强分布
self.Ir = self.calculate_intensity(self.d)

# 绘制光强分布
self.H = self.ax.pcolormesh(self.x, self.y, self.Ir, shading='auto')
self.ax.set_xlabel('x (m)', fontsize=15)
self.ax.set_ylabel('y (m)', fontsize=15)
self.ax.set_title('迈克尔逊干涉仪', fontsize=12)
self.fig.colorbar(self.H)

# 创建按钮
self.button_increase = QtWidgets.QPushButton('增加 d', self)
self.button_increase.clicked.connect(self.increase_d)

self.button_decrease = QtWidgets.QPushButton('减少 d', self)
self.button_decrease.clicked.connect(self.decrease_d)

self.button_info = QtWidgets.QPushButton('项目说明', self)
self.button_info.clicked.connect(self.open_pdf)

# 布局
button_layout = QtWidgets.QHBoxLayout()
button_layout.addWidget(self.button_increase)
button_layout.addWidget(self.button_decrease)
button_layout.addWidget(self.button_info) # 添加项目说明按钮
layout.addLayout(button_layout)
```

```

def calculate_intensity(self, d):
    """根据给定的 d 计算光强分布"""
    return 4 * (np.cos(2 * np.pi * d / self.lambda_) * np.cos(np.arctan(self.r /
self.f))))**2

def increase_d(self):
    """增加 d 的值并更新光强分布图"""
    self.d += 0.00005e-3 # 增加 d
    self.Ir = self.calculate_intensity(self.d) # 计算新的光强分布
    self.H.set_array(self.Ir.T.ravel()) # 更新图像数据
    self.canvas.draw_idle() # 更新绘图

def decrease_d(self):
    """减少 d 的值并更新光强分布图"""
    self.d -= 0.00005e-3 # 减少 d
    self.Ir = self.calculate_intensity(self.d) # 计算新的光强分布
    self.H.set_array(self.Ir.T.ravel()) # 更新图像数据
    self.canvas.draw_idle() # 更新绘图

def open_pdf(self):
    """打开项目说明 PDF"""
    pdf_path = 'pdfs./michelson_equal_path_interference.pdf' # 设置
    PDF 文件的路径
    QDesktopServices.openUrl(QUrl.fromLocalFile(pdf_path))

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    window = michelson_equal_path_interference_()

```

```
window.show()  
sys.exit(app.exec_())
```

## 使用 AI 模拟物理问题实际操作：

```
>>> 实现一个一维无限深势阱的波函数可视化
为了可视化一维无限深势阱的波函数，我们需要使用一种特殊的图形工具。下面是一些建议：
1. 使用 Python 中的 Matplotlib 库来可视化波函数。Matplotlib 提供了各种方法来绘制函数，包括 Plotly、Seaborn、等等。
...python
import matplotlib.pyplot as plt

# 生成波函数
def wave_function(x):
    return np.sin(np.pi * x) / (np.pi * x)

# 可视化波函数
plt.plot(wave_function(np.linspace(-1, 1, 100)))
plt.show()
```
在上面的代码中，我们首先定义了一个 `wave_function()` 函数来生成波函数。然后，我们使用 Matplotlib 的 `plot()` 方法来绘制线图。最后，我们使用 `show()` 方法显示图形。
3. 对于更复杂的波函数，可以使用不同的绘图方法。例如，我们可以使用 `plot_surface()` 方法来绘制三维曲面图，这样就能更好地显示波函数的结构。
```

图 16 描述问题

```
4. 如果你想要在一个二维平面上可视化整个波函数，可以使用 `plot()` 方法来绘制一系列的线图。例如：
...python
import matplotlib.pyplot as plt

# 生成波函数
def wave_function(x, y):
    return np.sin(np.pi * x) / (np.pi * x) + np.cos(np.pi * y) / (np.pi * y)

# 可视化波函数
x = np.linspace(-1, 1, 100)
y = np.linspace(-1, 1, 100)
X, Y = np.meshgrid(x, y)
Z = wave_function(X, Y)
plt.plot(X, Z, 'o')
plt.show()
```
在上面的代码中，我们首先定义了一个 `wave_function()` 函数来生成波函数。然后，我们使用 Matplotlib 的 `meshgrid()` 方法来生成一系列的线图。最后，我们使用 `plot()` 方法来绘制整个波函数。
```

图 17 AI 回复

```
5. 如果你想要在三维空间中可视化整个波函数，可以使用 `plot_surface()` 方法来绘制一系列的三维曲面图。例如：
...python
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

# 生成波函数
def wave_function(x, y):
    return np.sin(np.pi * x) / (np.pi * x) + np.cos(np.pi * y) / (np.pi * y)

# 可视化波函数
x = np.linspace(-1, 1, 100)
y = np.linspace(-1, 1, 100)
X, Y = np.meshgrid(x, y)
Z = wave_function(X, Y)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='viridis')
plt.show()
```
在上面的代码中，我们首先定义了一个 `wave_function()` 函数来生成波函数。然后，我们使用 Matplotlib 的 `meshgrid()` 方法来生成一系列的三维曲面图。最后，我们使用 `plot_surface()` 方法来绘制整个波函数。
```

图 18 关于三维实现

将重点聚焦于第四条，二维平面实现一维无限深势阱波函数叠加。会发现代码有一些错误：例如未导入 numpy，重复定义变量，物理公式也使用错误。

### 优化过程：

```
>>> 请用Python写一段程序，模拟一维无限深势阱中粒子的波函数叠加
...
import numpy as np

# 定义粒子位置和动量
n = 100 # 粒子个数
dx = 0.01 # 粒子间距
xmin = -5.0 # 势阱的左边界
xmax = 5.0 # 势阱的右边界
m = 1.0 # 粒子质量
k = 1.0 # 势阱的势能常数

# 初始化波函数
psi = np.zeros((n+1, n+1)) # 创建一个(n+1) x (n+1)矩阵，用于存储粒子位置和动量的信息
psi[0][0] = 1.0 # 第一个粒子为原点

# 定义波函数的迭代方程
def psi_next(x, t):
    return np.exp(-k*t)*np.sin(np.pi*x)

# 定义势阱的迭代方程
def V(x):
    if x < xmin or x > xmax:
        return 0.0
    else:
        return k*(xmax - x)

# 进行迭代
t = 0.0 # 时刻起点
dt = 0.01 # 时钟步长
T = 50.0 # 时钟终点
while t < T:
    psi_new = np.zeros((n+1, n+1)) # 创建一个(n+1) x (n+1)矩阵，用于存储下一时刻的波函数信息
    for i in range(1, n+1):
        for j in range(1, n+1):
            psi_new[i][j] = psi_next(xmin + (i-1)*dx, t) * np.sin(np.pi*xmin + (j-1)*dx) / (m*np.pi)
    psi[:] = psi_new # 将下一时刻的波函数信息更新到psi中
    t += dt # 更新时间
```

图 19 使用 AI 优化

```

# 绘制结果
x = np.linspace(xmin, xmax, N+1) # 创建一个(N+1)长度的坐标轴, 用于存储粒子位置信息
y = np.zeros((N+1)) # 创建一个(N+1)长度的数组, 用于存储振幅信息
for i in range(1, N+1):
    for j in range(1, N+1):
        y[i-1] += np.abs(psi[i][j])**2 # 计算振幅的平方
y = y * dx
plt.plot(x, y) # 绘制结果
plt.show() # 显示图像
```

```

图 20 优化后的绘制

此时代码已经比较完善，逻辑比较清晰，但物理部分有些问题，显然 AI 没有考虑清楚一维定态薛定谔方程，我们将物理部分修改正确，增删一些细节。最终修改后代码计算部分如图所示：

```

import numpy as np
class Wavefunction:
    Tabonne | Edit | Test | Explain | Document
    def __init__(self, a=1, miu=9.18938356e-31, h=6.62607015e-34):
        self.a = a
        self.miu = miu
        self.hbar = h / (2 * np.pi)
        self.main_quantum_number = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
        self.coefficient = [1] + [0] * 9 # 默认第一能级为1, 其余为0
    Tabonne | Edit | Test | Explain | Document
    def psi(self, n, x, t):
        En = n**2 * self.hbar**2 / (8 * self.miu * self.a**2)
        return np.sqrt(2 / self.a) * np.sin(n * np.pi * x / self.a) * np.exp(-1j * En * t / self.hbar)
    Tabonne | Edit | Test | Explain | Document
    def desirable_psi(self, x, t):
        wave_ideal = np.zeros_like(x, dtype=complex)
        for n, c in zip(self.main_quantum_number, self.coefficient):
            wave_ideal += c * self.psi(n, x, t)
        norm = np.sqrt(np.trapezoid(np.abs(wave_ideal)**2, x))
        return wave_ideal / norm if norm != 0 else wave_ideal
```

```

图 21 修改后代码

最终运行效果图：

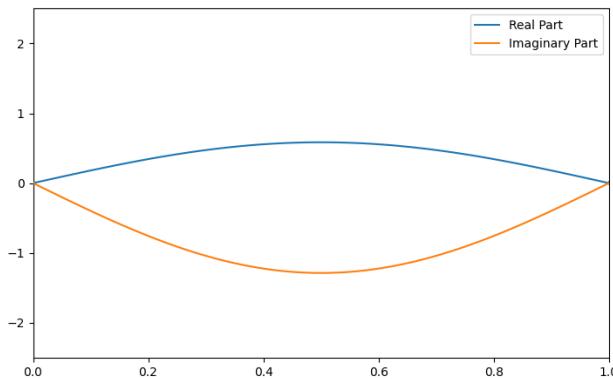


图 22 最终效果