东北大学秦皇岛分校控制工程学院

**《过程控制系统》课程设计报告**

**锅炉蒸汽压力控制系统设计**

|  |  |
| --- | --- |
| **学 生：** | **周禄也** |
| **专 业：** | **测控技术与仪器** |
| **班级学号：** | **202213630** |
| **指导教师：** | **宋爱娟 马淑华** |
| **设计时间：** | **2025.7.7-2025.7.18** |

目录

[一、设计题目与设计任务 4](#_Toc203293472)

[1. 设计题目： 4](#_Toc203293473)

[2. 设计任务细化： 4](#_Toc203293474)

[二、绪论 5](#_Toc203293475)

[三、主体设计部分 7](#_Toc203293476)

[1. 系统硬件设计 7](#_Toc203293477)

[2. PROTEUS仿真51单片机软件设计 10](#_Toc203293478)

[2.1 程序流程图 10](#_Toc203293479)

[2.2 软件任务书与核心代码分析 11](#_Toc203293480)

[3. MATLAB仿真建模部分 14](#_Toc203293481)

[3.1 MATLAB仿真建模部分设计背景与意义 14](#_Toc203293482)

[3.2 控制系统主要结构 15](#_Toc203293483)

[3.3 系统建模 15](#_Toc203293484)

[3.4 控制器设计 16](#_Toc203293485)

[3.5 系统结构仿真与分析 17](#_Toc203293486)

[3.6数字控制器PID参数整定 19](#_Toc203293487)

[3.7系统辨识简单小助手开发 24](#_Toc203293488)

[3.8 PID参数快速验证工具 28](#_Toc203293489)

[四、结束语 29](#_Toc203293490)

[五、参考文献 30](#_Toc203293491)

[六、附录 31](#_Toc203293492)

[1. MATLAB小助手源码 31](#_Toc203293493)

[2 . 51单片机程序源码 36](#_Toc203293494)

[3. PID参数快速验证工具 53](#_Toc203293495)

摘要

本报告围绕锅炉蒸汽压力控制系统的建模、仿真与嵌入式实现展开，系统介绍了基于PID算法的控制器设计、MATLAB仿真分析及单片机自动控制系统实现过程。通过软硬件协同与系统辨识工具开发，验证了控制策略的有效性和工程可行性。设计内容涵盖理论分析、仿真优化与工程实践，体现了过程控制系统的综合应用能力。

# 一、设计题目与设计任务

## 1. 设计题目：

锅炉蒸汽压力控制系统设计

## 2. 设计任务细化：

建立锅炉蒸汽压力控制系统的数学模型，分析其动态特性、主要影响因素及建模假设，结合机理法与实验法提升模型的准确性。设计基于PID算法的数字控制器，完成参数整定与优化，分析不同参数对系统响应速度、超调量、稳态误差等性能指标的影响。

利用MATLAB/Simulink平台进行系统建模与仿真，开展多组工况下的仿真实验，验证控制策略的有效性和鲁棒性。基于Proteus和51单片机，搭建硬件仿真平台，实现控制算法的嵌入式移植与联调，考察软硬件协同效果。开发系统辨识辅助工具，实现一阶系统参数的快速拟合与可视化，提升辨识效率和工程实用性。

# 二、绪论

锅炉蒸汽压力控制系统作为工业自动化领域的典型应用之一，广泛应用于电力、化工、冶金、食品等行业。其核心任务是让锅炉蒸汽压力始终保持在设定值附近，确保设备安全、稳定运行。随着工业自动化水平的不断提升，锅炉压力控制系统的智能化、精细化和高可靠性成为行业发展的重要方向。

设计高效、可靠的控制系统，不仅能提升生产效率，还能降低能耗、保障设备安全。传统的人工调节方式已难以满足现代工业对高效、精准和智能化控制的需求，因此，研究和设计高性能的锅炉蒸汽压力自动控制系统具有重要的理论价值和现实意义。

目前，PID控制器因其原理简单、性能稳定，在工业控制中应用极为广泛，尤其适用于锅炉蒸汽压力控制系统。在硬件条件一定的情况下，PID参数的整定往往决定了闭环系统的性能优劣。随着嵌入式技术和仿真工具的发展，软硬件协同设计成为提升系统性能和工程实践能力的重要手段。

本设计综合运用了MATLAB建模仿真和Proteus 51单片机仿真两大工具。首先，利用MATLAB对锅炉蒸汽压力控制系统进行建模与仿真，分析系统的动态特性和控制效果。随后，结合Proteus平台和51单片机，搭建硬件仿真环境，实现对控制算法的嵌入式验证和系统联调。通过软硬件协同仿真，进一步提升了设计的完整性和工程实践能力。

**主要设计内容包括：**

**系统建模：**建立锅炉蒸汽压力控制系统的数学模型，分析其动态特性。

控制器设计：设计合适的PID控制器并进行参数整定，提升系统的响应速度和稳定性。

**系统仿真：**用MATLAB对设计好的控制系统进行仿真实验，验证控制策略的有效性。

**嵌入式仿真：**基于Proteus和51单片机实现控制系统的硬件仿真与验证，考察算法的实际可行性。

**结果分析：**分析仿真和实验结果，评价系统的稳定性、响应速度和精度，并与理论分析对比，提出优化建议。

此外，本设计还注重理论与实践结合，强调系统辨识、参数优化、工程实现等环节的完整性，力求为后续相关课题研究和实际工程应用提供参考。

# 三、主体设计部分

## 1. 系统硬件设计

本系统硬件部分由多个功能模块组成，各模块协同工作，实现对锅炉蒸汽压力的自动检测、信号采集、数据处理、控制输出和人机交互。整体结构设计注重模块化、可扩展性和抗干扰能力，便于后续维护和升级。具体包括：

**系统硬件结构图:**

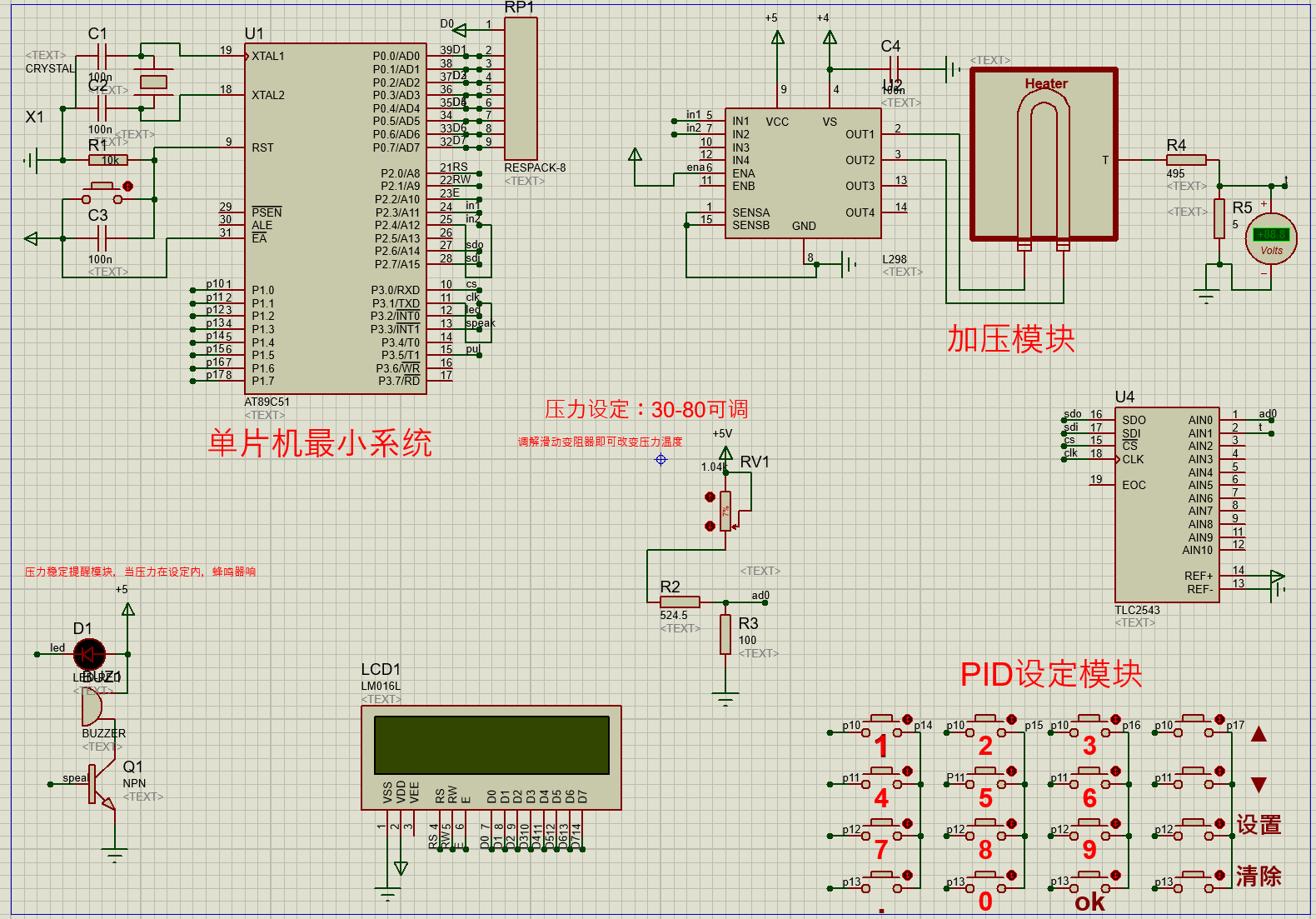


图3.1

**滑动变阻器 + TLC2542 AD转换模块：**

滑动变阻器用于手动设定目标压力值，模拟量信号通过TLC2542模数转换芯片转为数字信号，输入单片机，实现设定值的数字化采集。该模块具有高分辨率、低漂移等优点，保证设定值输入的准确性和稳定性。

**51单片机最小系统：**

作为系统的核心控制单元，负责采集各类传感器信号、执行控制算法、输出控制指令，并协调各功能模块的工作。单片机选型注重处理速度、I/O资源和可靠性，确保系统实时性和稳定性。系统采用模块化设计，便于功能扩展和后期维护。

**LED与蜂鸣器驱动报警电路：**

用于实时显示系统运行状态。当检测到压力异常或故障时，LED灯点亮并蜂鸣器报警，提醒操作人员及时处理。该模块设计有多级报警逻辑，提升系统安全性。报警阈值可根据实际需求灵活设定。

**LCD屏幕显示电路：**

实时显示锅炉压力、设定值、系统状态等关键信息，方便用户监控和操作。LCD显示界面友好，支持多页面切换和参数实时刷新。显示内容可根据用户需求自定义，提升人机交互体验。

**矩阵键盘输入电路：**

提供人机交互接口，用户可通过矩阵键盘输入参数、设定目标值或进行菜单操作，提高系统的灵活性和可操作性。键盘扫描算法优化，支持防抖和多功能扩展。可实现参数设置、模式切换等多种功能。

**L298N + KSD12V-40W驱动加热控制电路：**

L298N作为功率驱动芯片，控制KSD12V-40W陶瓷加热片的通断，实现对锅炉加热功率的调节。单片机输出控制信号，经L298N放大后驱动加热元件，具备过流保护和状态反馈功能。驱动电路设计充分考虑了安全性和可靠性。

**ICS1220压力检测变送电路：**

ICS1220压力传感器将锅炉内的蒸汽压力转换为标准电信号，经变送电路处理后送入单片机，实现压力的实时检测和反馈。传感器选型注重精度、线性度和抗干扰能力，保障系统测量的可靠性。信号调理电路设计合理，有效抑制噪声干扰。

**小结：**

各模块之间通过信号线和电源线连接，形成完整的闭环控制系统。系统能够自动采集压力信号，根据设定值和实际值的偏差，自动调节加热功率，保障锅炉压力稳定在安全范围内。整体硬件布局合理，便于调试和维护。

**主要硬件参数表**

模块名称 主要参数及说明

**滑动变阻器**：阻值范围：1kΩ100kΩ，功率：0.5W2W，用于设定目标压力值

**TLC2542 AD转换芯片**：分辨率：12位，输入电压：0~5V，速度：100ksps，8路多路复用。

**51单片机最小系统**：主频：12MHz，ROM：8K，RAM：256B，I/O口：32个。

**LED与蜂鸣器**： LED电压：23V/1020mA，蜂鸣器电压：5V，电流<30mA。

**LCD显示屏**： 常用型号：1602，工作电压：5V，显示压力和状态。

**矩阵键盘4×4**： 工作电压：5V，参数输入与菜单操作。

**L298N电机驱动模块 电压**： 5~35V，最大输出电流：2A/通道，驱动加热片。

**KSD12V-40W陶瓷加热片**： 额定电压12V，功率40W，锅炉加热元件。

**ICS1220压力传感器** 测量范围： 0~5MPa，输出：0~50mV，精度：±0.1%。

## 2. PROTEUS仿真51单片机软件设计

本系统软件部分采用模块化设计思想，主要包括主程序流程、参数初始化、数据采集、PID控制算法、外设驱动、人机交互等功能模块。通过合理的软件架构设计，提升系统的可维护性和扩展性。

### 2.1 程序流程图

**程序流程图**

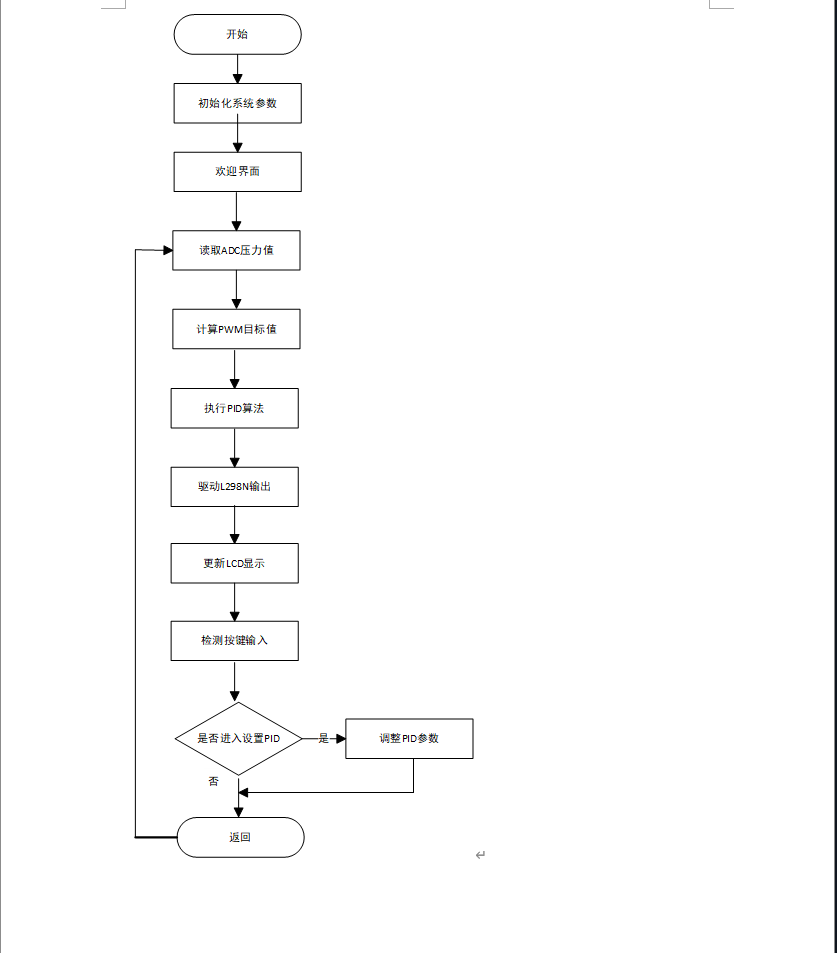


图3.2

程序流程图清晰展示了系统的主要运行逻辑，包括系统初始化、参数设置、数据采集、PID调节、报警处理、数据显示与人机交互等环节。各模块之间协同工作，实现对锅炉压力的闭环自动控制。

### 2.2 软件任务书与核心代码分析

**引脚定义与硬件接口:**

代码开头通过sbit定义了L298N驱动、AD采集、LCD显示、LED、蜂鸣器等外设的引脚，明确了单片机与外围电路的连接关系，便于后续维护和移植。

**变量与参数声明:**

设定了PID参数（KP、KI、KD）、PWM相关变量、状态标志等，为后续控制和界面交互做准备。参数声明规范，便于后续调试和优化。

**主循环与流程:**

main()函数初始化硬件，显示欢迎界面，进入主循环：

采集设定值和压力值，进行PID计算，输出PWM控制信号。判断压力是否达标，控制LED和蜂鸣器报警。根据setflag和pageflag切换不同的参数设置和显示界面。主循环结构清晰，便于功能扩展和异常处理。

**PID控制算法:**

mypid()函数实现了标准的PID闭环调节，按设定值与实际压力的偏差动态调整加热输出，保证压力稳定。算法实现简洁高效，便于参数整定和移植。

**人机交互:**

包含LCD显示、AD采集、矩阵键盘扫描等函数，支持人机交互和数据可视化。各外设驱动模块独立，便于后续升级和维护。

**异常处理与安全保护:**

软件设计中充分考虑了异常工况的检测与处理，如压力超限、传感器故障等，提升系统的安全性和鲁棒性。整体软件架构层次分明，功能模块划分合理，既满足了系统的实时性要求，又便于后期维护和功能扩展。

**核心代码简析**

**PID控制核心**

void mypid(float Kp,float Ki,float Kd,uint count,uint point)

{

static float Uk;

ek=point-count;

Uk=Kp\*(ek-ek1)+Ki\*ek+Kd\*(ek-2\*ek1+ek2);

pwm=Uk;

if(pwm>freq)pwm=freq;

if(pwm<=0){pwm=0;in1=0;in2=1;}

if(pwm>0) {in1=1;in2=0;}

ek2=ek1;

ek1=ek;

}

该函数根据设定值和实际压力的偏差，动态调整PWM输出，实现闭环自动控制。

**主循环结构**

while(1)

{

pwm\_temp=500\*(float)(adnum0)/4095;

mypid(KP,KI,KD,adnum1,pwm\_temp);

scantemp=keyscan();

// ...界面切换与参数设置逻辑...

}

主循环不断采集数据、执行PID调节，并响应用户输入，实现系统的自动运行和参数整定。

## 3. MATLAB仿真建模部分

### 3.1 MATLAB仿真建模部分设计背景与意义

锅炉蒸汽压力控制系统是工业自动化领域的重要组成部分，其核心任务是让锅炉蒸汽压力始终保持在设定值附近，确保设备安全、稳定运行。设计高效、可靠的控制系统，不仅能提升生产效率，还能降低能耗、保障设备安全。

目前，PID控制器因其原理简单、性能稳定，在工业控制中应用极为广泛，尤其适用于锅炉蒸汽压力控制系统。在硬件条件一定的情况下，PID参数的整定往往决定了闭环系统的性能优劣。

本设计将利用MATLAB对锅炉蒸汽压力控制系统进行建模与仿真，主要包括以下几个方面：

**系统建模**：建立锅炉蒸汽压力控制系统的数学模型，分析其动态特性和主要影响因素。

**控制器设计**：设计合适的PID控制器并进行参数整定，分析不同参数对系统性能的影响。

**系统仿真**：用MATLAB对设计好的控制系统进行仿真实验，考察系统的响应速度、超调量、稳态误差等指标。

**结果分析**：分析仿真结果，评价系统的稳定性、响应速度和精度，并与理论分析对比，提出优化建议。

通过理论建模与仿真分析，为后续硬件实现和参数优化提供理论依据。

### 3.2 控制系统主要结构

本系统采用典型的单闭环结构，主要包括：

**控制器**：单片机（实现数字PID算法）

**执行器**：L298N+KSD12V-40W陶瓷加热片（驱动电路+执行元件）

**被控对象**：锅炉（近似为一阶惯性对象，具体开环传递函数通过辨识或查阅文献获得）

**变送器**：ICS1220压力传感器（实现压力信号的采集与变送）

### 3.3 系统建模

在设计锅炉蒸汽压力控制系统前，需对系统进行详细建模。根据锅炉的工作原理和控制要求，建立数学模型。常用方法有机理法和实验法。。

**机理法**

机理法基于锅炉系统的物理和化学原理，通过分析结构、能量和物质流动过程，建立数学模型。主要考虑：

质量守恒：建立水和蒸汽的质量平衡方程，描述锅炉内水和蒸汽的变化。

能量守恒：分析锅炉吸收热量、蒸发过程中的能量转换，建立能量平衡方程。

动量守恒：考虑流体在锅炉内的流动特性，建立动量平衡方程。

实际锅炉系统较为复杂，通常需做如下简化：

忽略部分较小的时间常数和影响因素；

对非线性环节进行线性化处理；

采用集中参数法简化分布参数系统。

通过上述分析和假设，可推导出锅炉蒸汽压力控制系统的传递函数或状态空间 模型，为后续控制器设计和仿真提供理论基础。

**实验法**

实验法通过对实际系统进行试验，采集输入输出数据，建立系统模型。常用方法有：

**阶跃响应法：**在系统稳态下施加阶跃输入，记录输出响应曲线。根据曲线特征参数（如上升时间、超调量、稳态值等）估算系统参数。适用于一阶、二阶等简单系统。

**频率响应法：**对系统施加不同频率的正弦信号，测量输出的幅值和相位，绘制幅频和相频特性曲线（波特图），据此拟合系统传递函数。

**系统辨识法：**采集系统在不同工况下的输入输出数据，利用最小二乘法等数学工具进行参数估计和模型拟合，适用于结构复杂或难以直接建模的系统。

实际工程中，常将机理法和实验法结合使用：

先用机理法建立系统的基本结构模型，保证物理合理性；再通过实验法对模型参数进行修正和优化，提高准确性；最后通过仿真和实际运行数据对模型进行验证和完善。这种结合方法既能反映系统的物理本质，又能兼顾实际运行的动态特性。

### 3.4 控制器设计

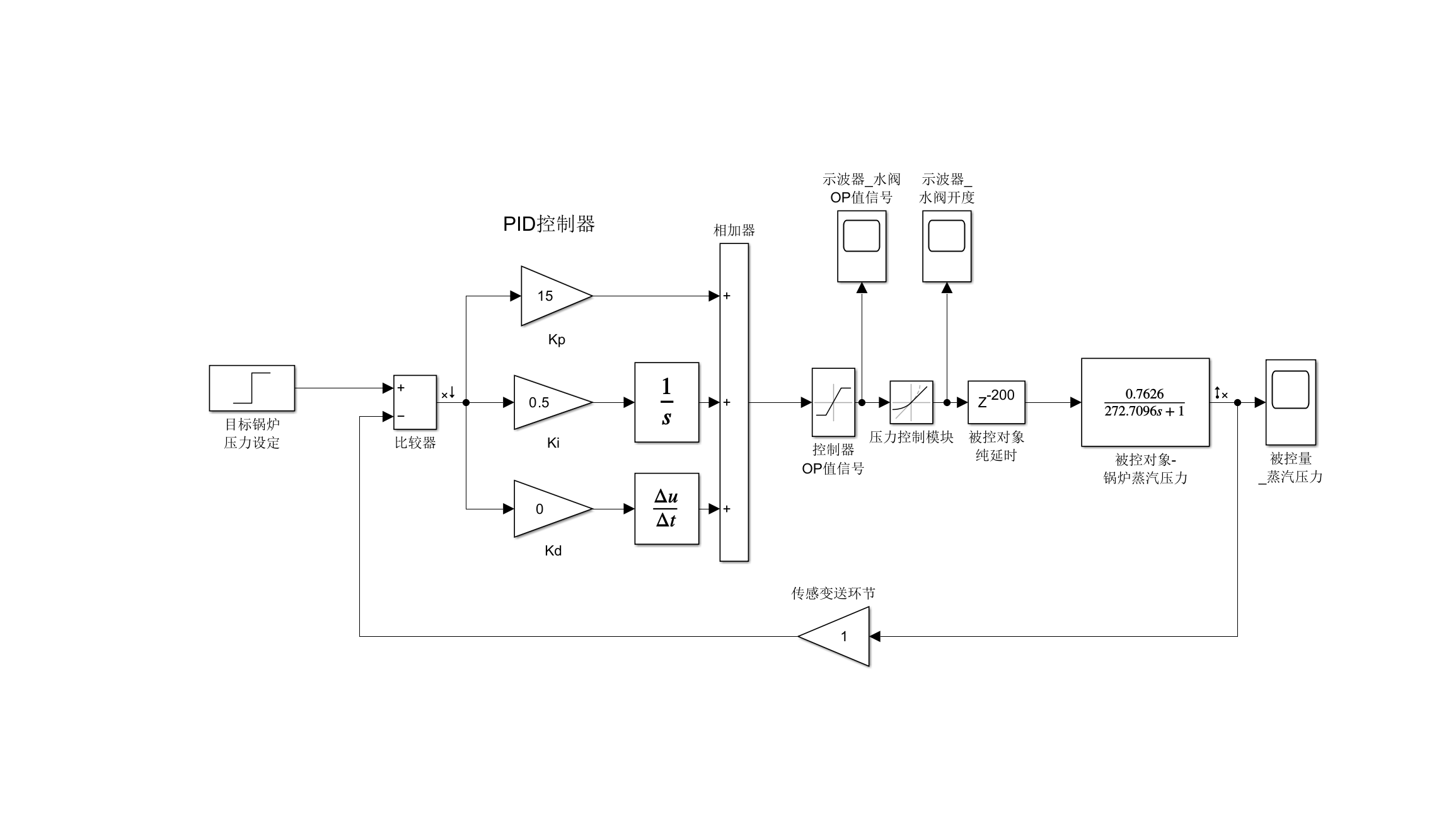
本次课程设计选用简单实用的PID数字控制器。PID控制器结构清晰，参数整定灵活，能够满足锅炉蒸汽压力控制系统对稳定性和响应速度的要求。设计过程中，重点分析比例、积分、微分参数对系统超调量、响应速度、稳态误差等性能指标的影响，结合仿真结果进行参数优化。

### 3.5 系统结构仿真与分析

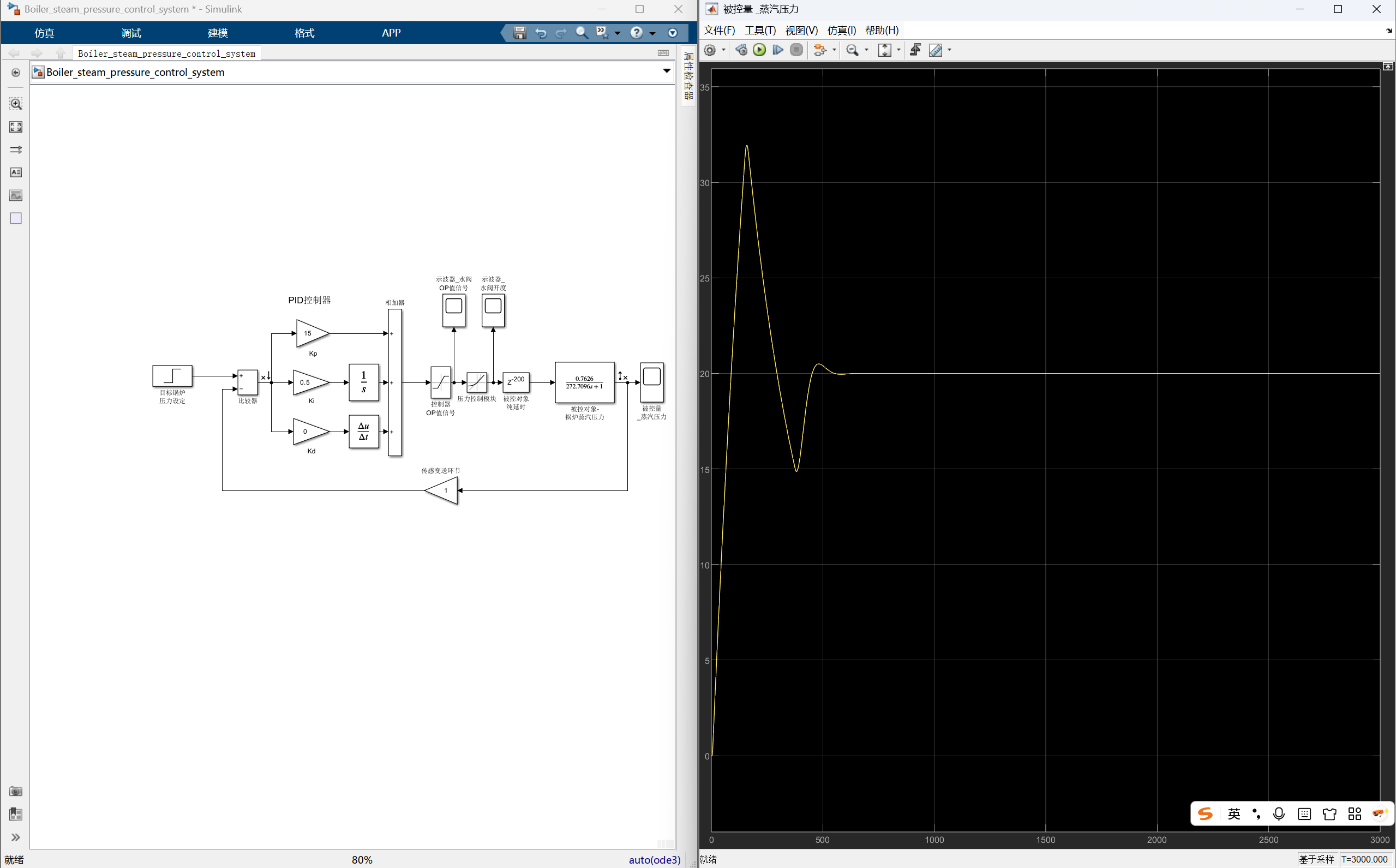
**系统总体结构**

本系统采用单闭环控制结构，主要包括：比价器、PID数字控制器、执行器、被控对象、传感变送器。各环节协同工作，实现对锅炉压力的自动调节和实时反馈。

锅炉蒸汽压力控制系统的总体结构如下图所示：  
在本次MATLAB仿真中 ，结合实际情况，被控对象，锅炉压力的开环传递函数是：  
((1.079)(z^-1000))/(101.470s+1) , (纯延时1000ms ) 。

图3.3

**系统仿真**

在MATLAB中，采用Simulink进行系统建模和仿真。通过搭建系统仿真模型，考察不同PID参数、扰动工况下系统的动态响应特性。下图为锅炉蒸汽压力控制系统的Simulink模型示例及仿真效果：  
  
 图3.4

### 3.6数字控制器PID参数整定

合理的参数整定能够有效提升系统的响应速度、减小超调量、降低稳态误差，并增强系统的鲁棒性。

针对锅炉蒸汽压力控制系统，常用的PID参数整定方法主要包括：经验法与临界比例法通过逐步调整比例（Kp）、积分（Ki）、微分（Kd）参数，观察系统响应，直观判断参数对超调、响应速度和稳态误差的影响。

采用临界比例法（Ziegler-Nichols法），先将Ki、Kd设为零，逐步增大Kp至系统出现持续振荡，记录此时的Kp和振荡周期，再按经验公式计算Ki、Kd。仿真优化法利用MATLAB/Simulink仿真平台，建立系统模型，采用阶跃响应、扰动响应等典型工况，批量仿真不同参数组合，选取最优性能指标（如调节时间、超调量、稳态误差等）对应的参数。  
 下面是采用临界比例度法对系统PID参数进行整定的**步骤**：  
 （1）找到临界震荡

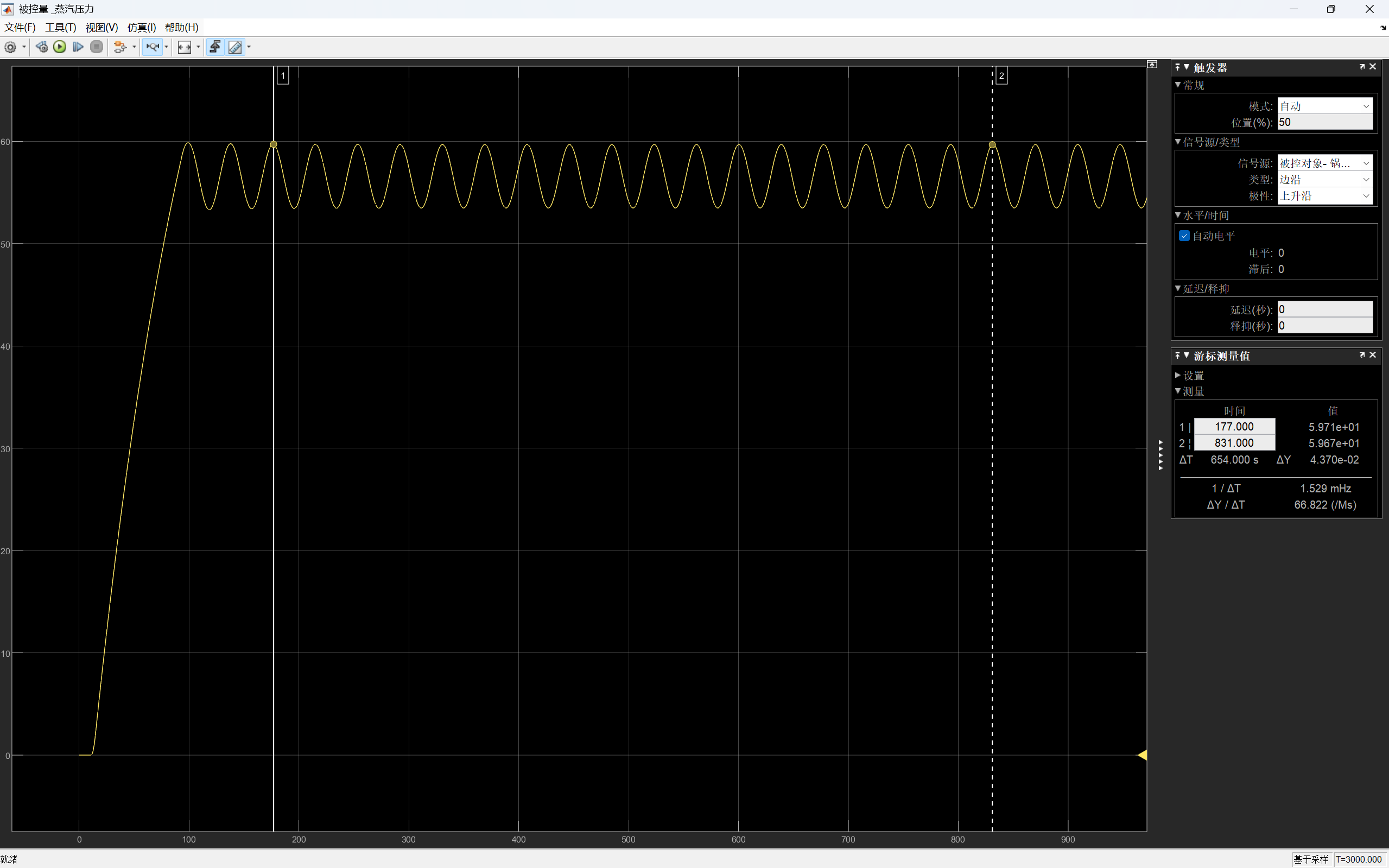


图3.5

经试验得到 Ku = 15.4  
计算得到Tu = (654.00s)/17 = 38s   
（2）PID控制器参数整定

|  |  |  |  |
| --- | --- | --- | --- |
| 控制类型 | Kp | Ti | Td |
| P | 0.5⋅Ku | \_ | \_ |
| PI | 0.45⋅Ku | Tu/1.2 | \_ |
| PID | 0.6⋅Ku | Tu/2 | Tu/8 |

|  |  |  |  |
| --- | --- | --- | --- |
| PID参数计算 | Kp | Ti | Td |
| P | 7.7 | \_ | \_ |
| PI | 6.93 | 31.67 | \_ |
| PID | 9.24 | 19 | 4.75 |

|  |  |  |  |
| --- | --- | --- | --- |
| PID参数计算 | Kp | Ki | Kd |
| P | 7.7 | \_ | \_ |
| PI | 6.93 | 0.219 | \_ |
| PID | 9.24 | 0.486 | 43.80 |

P控制实验图像：  
  
 图3.6

控制效果不错，超调量不大，响应速度快，但是有稳态误差 。

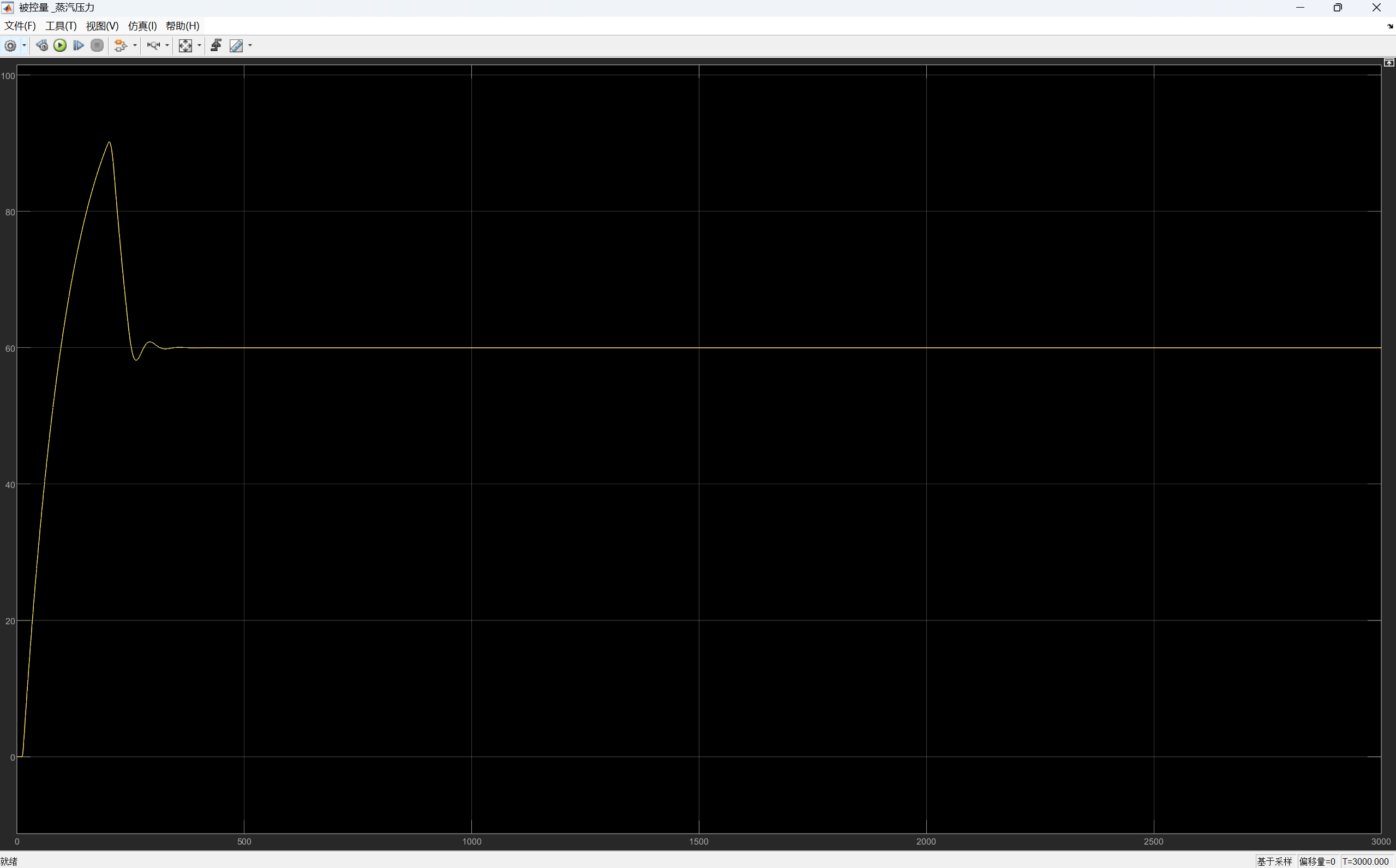
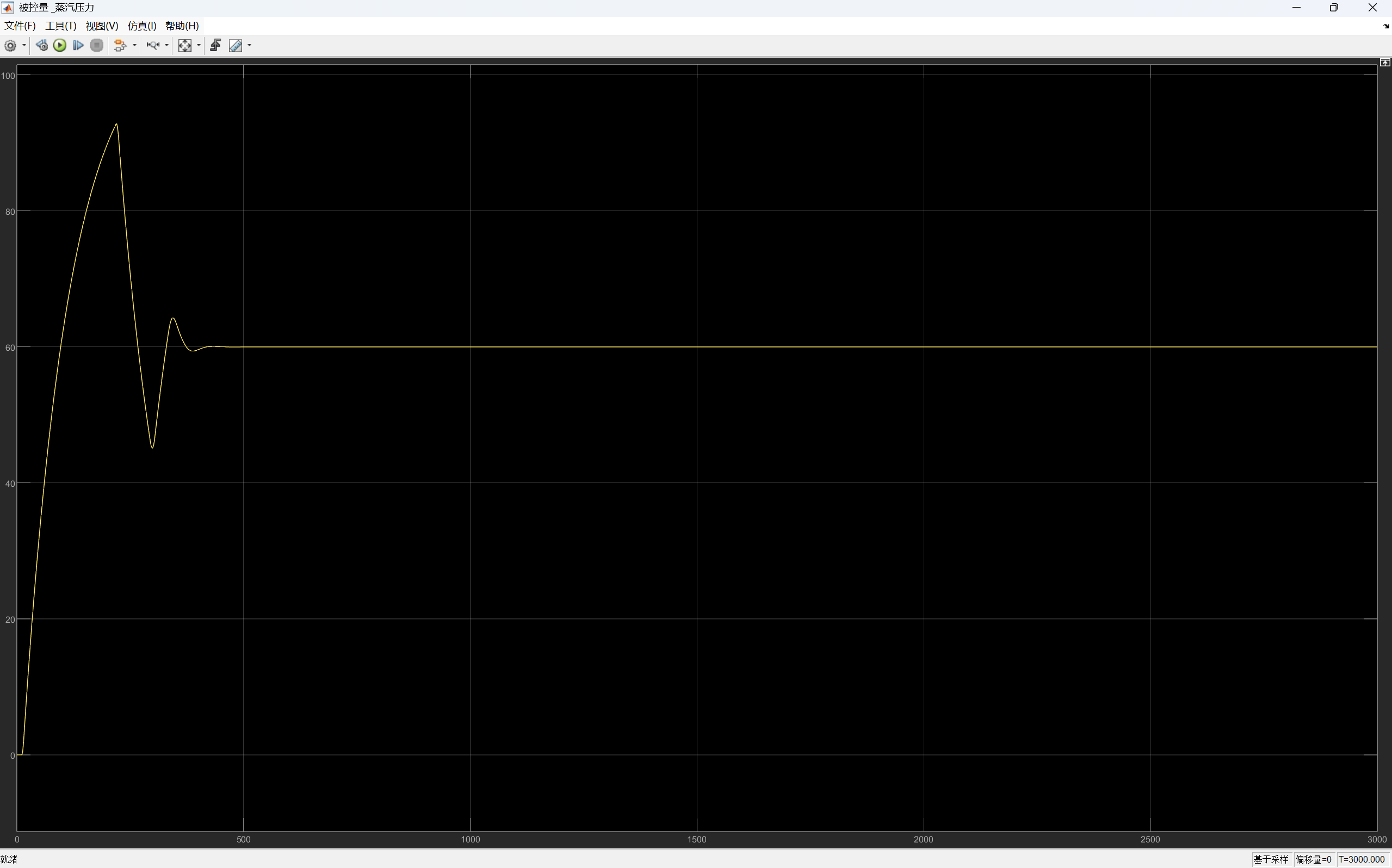
PI控制实验图像：  


图3.7

PID控制实验图像：

  
图3.8

按照公式计算 ，PI 和 PID的响应效果不是很好 ，所以下一步应在上面实验的基础上调整参数。

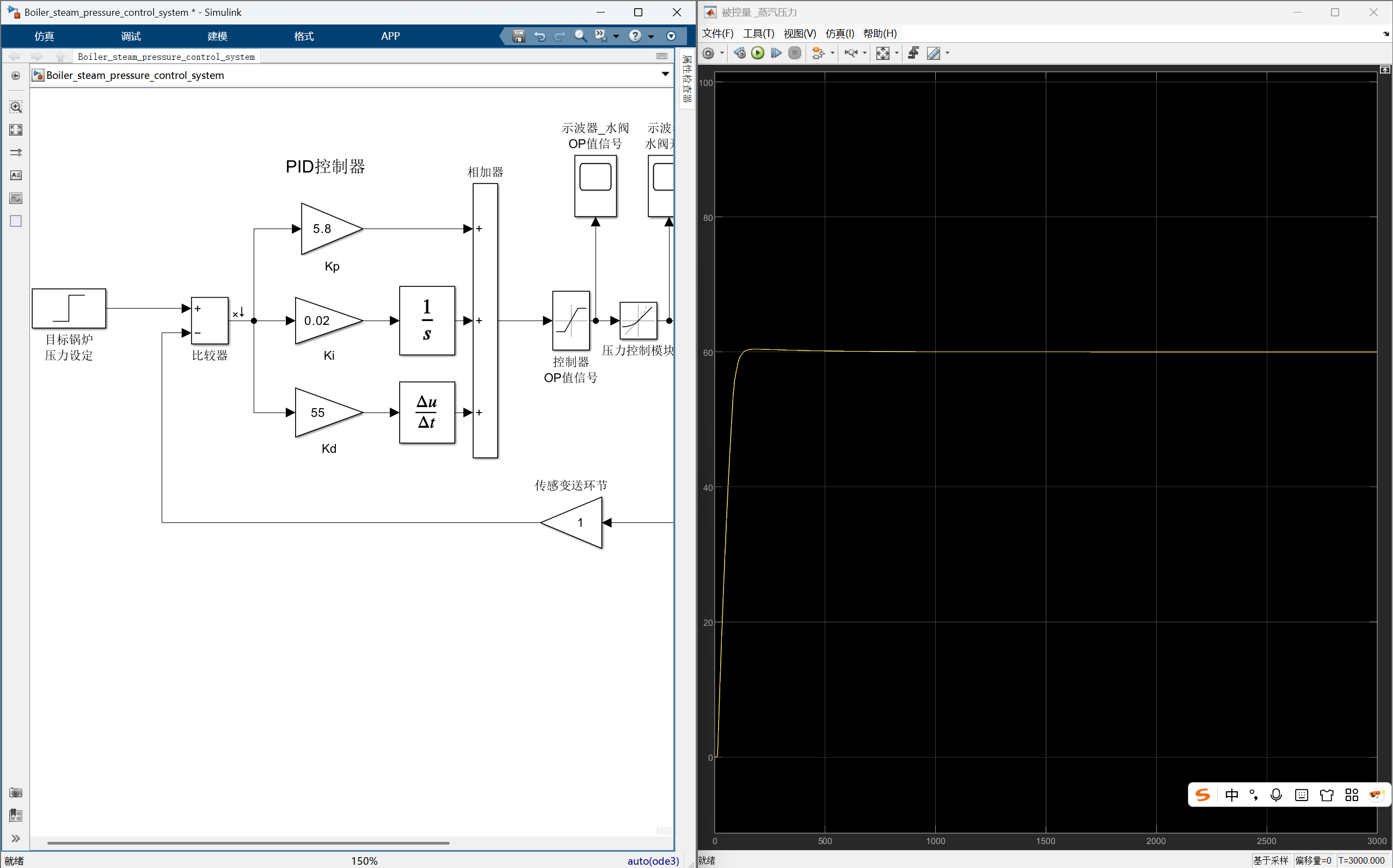
参数试凑法试验后得到的PID控制器：Kp: 5.8 Ki: 0.02 Kd: 55  
实验图像：  


图3.9

工程实践建议初步整定后，需在实际系统或硬件仿真平台上进行微调，兼顾系统的响应速度与稳定性，防止过度超调或振荡。对于锅炉等大惯性、时滞对象，建议适当降低Kp、增大Ki，Kd可根据实际响应微调，避免高频噪声放大。

通过多次仿真和实验，最终确定最优参数。通过上述整定流程，结合仿真与实验结果，最终获得满足锅炉蒸汽压力控制要求的PID参数，实现系统的高效、稳定运行。

### 3.7系统辨识简单小助手开发

本模块开发了一个基于MATLAB GUI的“系统辨识简单小助手”，用于辅助一阶系统参数辨识与传递函数拟合，提升辨识效率和可视化水平，便于教学和工程初步建模。

**使用方法：** 依次输入从实验室得到的被控对象开环传递函数（一阶），该MATLAB脚本能够自动拟合曲线，并且计算出来被控对象的开环传递函数，使用边界，人机交互友好，自动化程度高

**功能展示**

下图展示了“系统辨识简单小助手”的主要界面和功能布局：

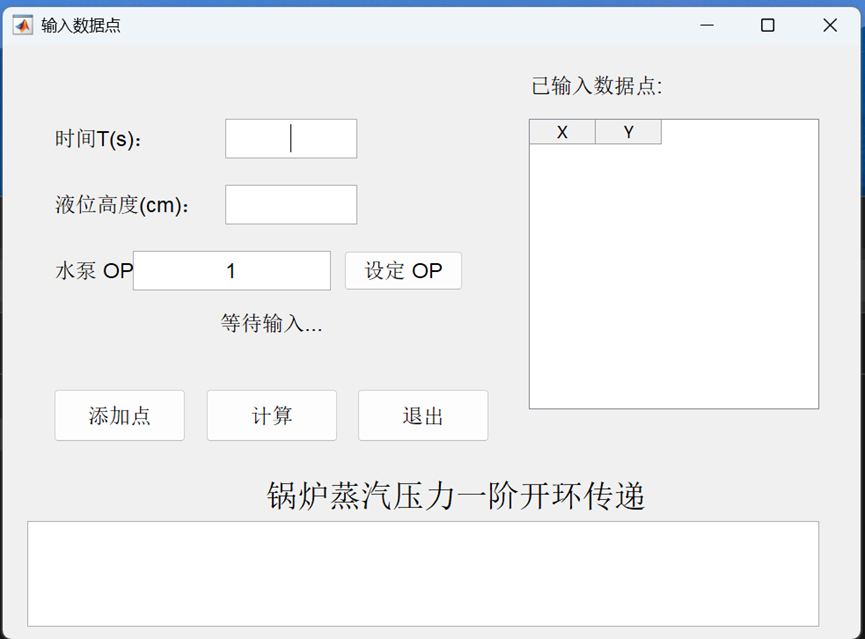


图3.10

界面设计简洁直观，包含数据输入、参数设定、结果显示、曲线绘制等功能区，便于用户操作和结果分析。

**设计目标**

本模块旨在为用户提供一个便捷的系统辨识工具，支持通过实验数据点快速完成一阶系统的参数辨识与传递函数拟合。界面直观，操作简便，适合教学和工程初步建模。

**功能需求**

数据输入功能：

支持用户逐点输入实验数据（时间T、液位高度Y），并实时显示在数据表格中。支持通过按钮或回车快捷键添加数据点，提升输入效率。

参数设定功能

支持用户输入和设定水泵OP（操作点）参数，参数可随时修改，界面有提示反馈。

数据管理功能

可清空输入框，便于连续输入。支持退出程序，操作便捷。

系统辨识与拟合功能

至少输入3组数据后，支持一阶系统的参数拟合。自动拟合一阶惯性环节模型：y = K\*(1-exp(-t/τ))。输出拟合得到的K、τ参数及传递函数表达式。

在界面和命令行输出结果，并绘制原始数据与拟合曲线对比图。

用户交互与提示

对无效输入、数据不足等情况有友好提示。操作成功后有反馈信息。

功能展示，如图所示：

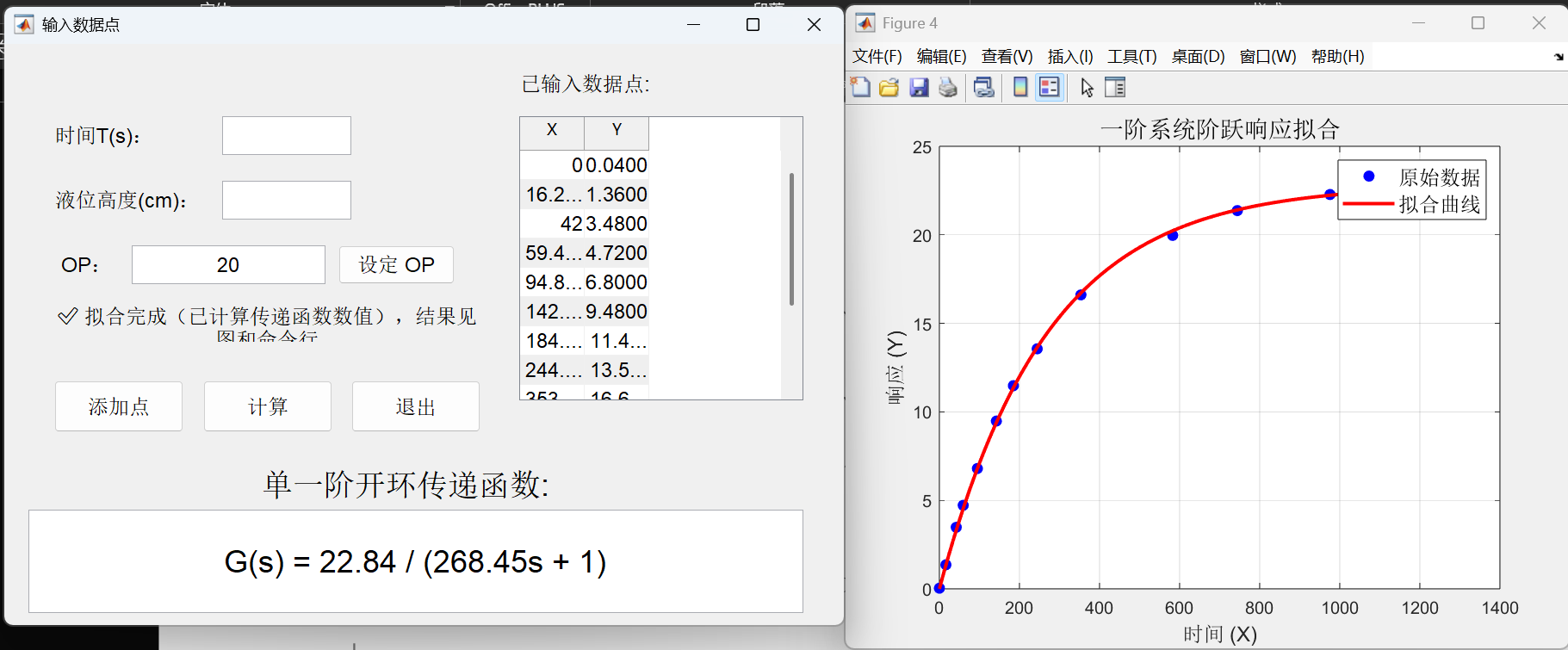


图3.11

**输入实验点之后，一阶对象阶跃响应图像拟合程度好，开环传递函数计算比较准确，有较强的实用价值。**

**界面设计**

主窗口：包含输入区、参数区、数据表格区、结果显示区和操作按钮区。

输入区：时间T、液位高度Y的输入框，便于逐点录入实验数据。

参数区：OP输入框及设定按钮，支持参数实时调整。

数据表格区：显示已输入的所有数据点，便于数据核查。

结果显示区：显示拟合得到的传递函数和参数。

操作按钮区：添加点、计算、清空、退出等按钮，提升操作便捷性。

**主要模块与流程**

初始化

初始化数据存储数组（X、Y）、OP参数、回车计数器。创建GUI各控件并布局，界面美观实用。

数据输入与管理

用户输入T、Y后，点击“添加点”或按回车，数据自动存入数组并刷新表格。

输入框自动切换焦点，提升输入效率。

OP参数设定

用户输入OP值，点击“设定OP”按钮，系统校验输入有效性并保存。

系统辨识与拟合

用户点击“计算”按钮，若数据点数≥3，则进行一阶系统拟合。

使用MATLAB fit函数，拟合模型y = K\*(1-exp(-t/τ))。

拟合结果（K、τ、传递函数）显示在界面和命令行。

自动绘制原始数据与拟合曲线对比图，便于直观分析。

用户提示与异常处理

对无效输入、数据不足等情况，界面有醒目提示。

操作成功后有确认信息。

**关键代码说明**

数据输入与表格刷新

X(end+1) = xVal;

Y(end+1) = yVal;

set(dataTable, 'Data', [X(:), Y(:)]);

OP参数设定

OP = str2double(get(opEdit, 'String'));

一阶系统拟合

fit\_type = fittype('K \* (1 - exp(-x / tau))', ...);

[fit\_result, gof] = fit(X(:), Y(:), fit\_type, 'StartPoint', initial\_guess);

结果输出与绘图

tfStr = sprintf('G(s) = %.2f / (%.2fs + 1)', K, tau);

set(tfOutput, 'String', tfStr);

plot(X, Y, 'bo', ...); plot(x\_fit, y\_fit, 'r-', ...);

本模块极大提升了系统辨识的效率和可视化水平，为后续控制系统建模与参数优化提供了有力工具。

### 3.8 PID参数快速验证工具

**使用方法**：输入被控对象的开环传递函数（1阶），输入PID参数 ，自动计算阶跃响应图像

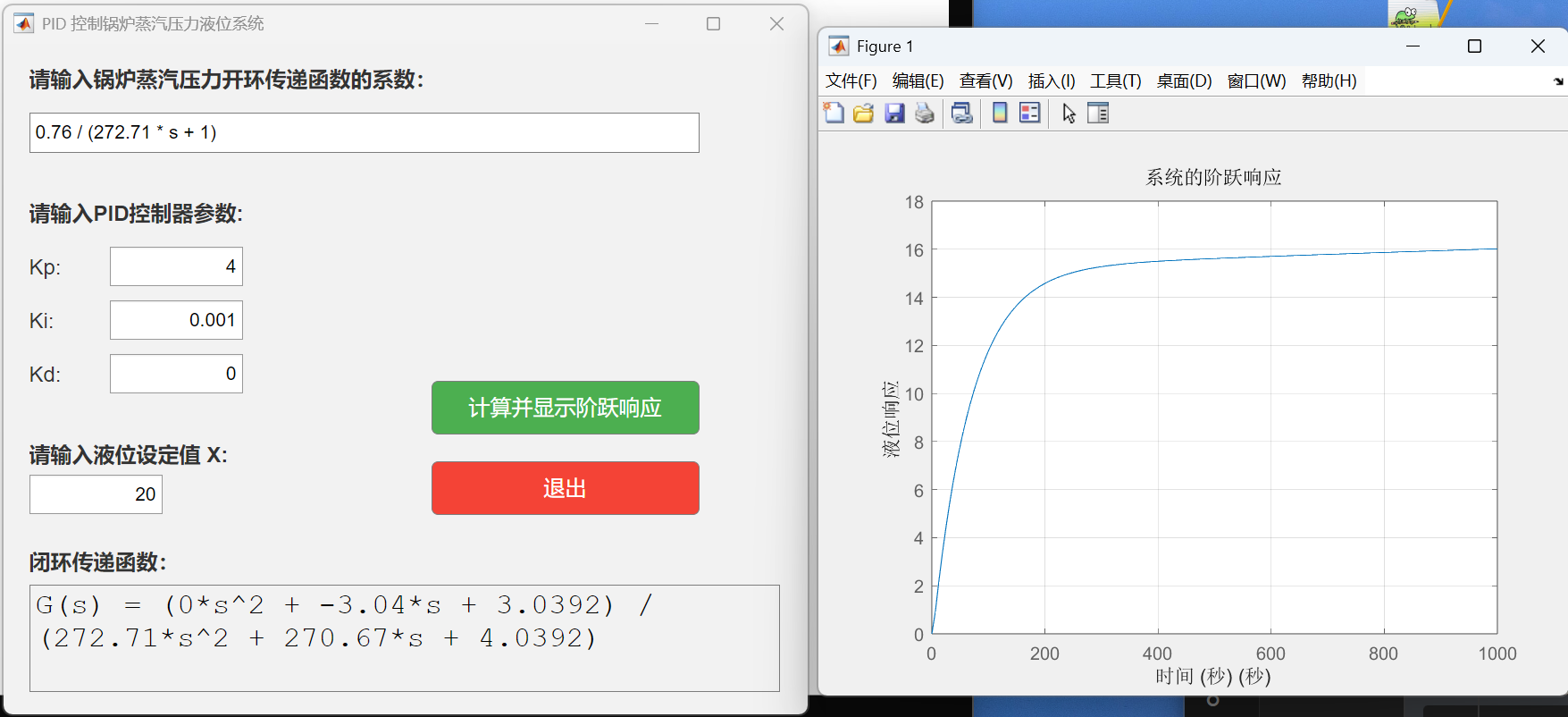
自动控制工程师 ，往往会遇到这样的情况，已知被控对象的开环传递函数，或者依照经验，知道大致的开环传奇函数。仿真精度要求不太高的，并且又要快速得进行PID的参数整定和验证 ，这时候就可以快速地打开我编写的这款MATLAB小程序，不需要搭建复杂的SIMULINK仿真程序，直接输入参数 ，就可以使用MATLAB进行数学运算，直接输出系统的阶跃响应图像：  
 程序编写主要依赖与MATLAB的库函数 ，相关源代码见附录。  


图3.12

与此同时， 该款小程序 ，也可以用于教学，课堂展示 ，实验课的软件补充，或者开放到开源网站，供自动控制爱好者学习，体验。

# 四、结束语

本次《过程控制系统》课程设计以锅炉蒸汽压力控制系统为对象，系统性地完成了从理论建模、控制器设计、仿真分析到嵌入式实现的全过程。通过查阅文献、分析工艺流程、建立数学模型，深入理解了锅炉压力控制的工程背景和技术难点。在MATLAB平台上完成了系统建模与仿真，掌握了PID参数对系统性能的影响规律，并通过多组工况的仿真实验，验证了控制策略的有效性和适应性。

在硬件实现方面，结合Proteus和51单片机平台，完成了控制算法的嵌入式移植与联调，提升了软硬件协同开发能力。通过开发系统辨识小助手工具，进一步加深了对一阶系统辨识与参数拟合的理解，增强了工程实践能力和创新意识。

本设计不仅巩固了过程控制、自动控制原理、嵌入式系统等多学科知识，还锻炼了独立分析问题和解决实际工程问题的能力。设计过程中也体会到理论与实践结合、团队协作和持续优化的重要性。

总之，本次课程设计为后续深入学习和工程实践打下了坚实基础，也为今后从事过程控制自动化系统开发积累了宝贵经验。

# 五、参考文献

[1] 任彦硕, 赵一丁, 张家生. 自动控制系统[M]. 北京: 北京邮电大学出版社, 2007.

六、附录

1. MATLAB小助手源码  
% (0.00, 0.04)

% (16.20, 1.36)

% (42.00, 3.48)

% (59.40, 4.72)

% (94.80, 6.80)

% (142.20, 9.48)

% (184.80, 11.48)

% (244.20, 13.56)

% (297.00, 15.07)

% (353.40, 16.61)

% (436.80, 18.09)

% (527.40, 19.29)

% (582.60, 19.97)

% (670.20, 20.80)

% (744.00, 21.37)

% (813.60, 21.64)

% (895.20, 22.01)

% (975.60, 22.28)

% (1054.80, 22.43)

% (1135.20, 22.69)

% (1207.80, 22.73)

% (1240.80, 22.81)

% 作者：周禄也 2025 04 30

function simple\_point\_input\_gui()

    % 初始化坐标数组

    X = [];

    Y = [];

    OP = 1;  % 初始 OP 值为 1

    enterCount = 0; % 回车次数计数器

    % 创建 GUI 窗口

    f = figure('Name', '输入数据点', 'Position', [500, 300, 650, 450], ...

               'NumberTitle', 'off', 'MenuBar', 'none', 'Resize', 'on');

    % 添加 X 输入框

    uicontrol('Style', 'text', 'Position', [40, 360, 200, 30], ...

              'String', '时间T(s)：', 'FontSize', 12, 'HorizontalAlignment', 'left');

    xEdit = uicontrol('Style', 'edit', 'Position', [170, 365, 100, 30], ...

                      'FontSize', 12, 'Callback', @editCallback);

    % 添加 Y 输入框

    uicontrol('Style', 'text', 'Position', [40, 310, 200, 30], ...

              'String', '液位高度(cm)：', 'FontSize', 12, 'HorizontalAlignment', 'left');

    yEdit = uicontrol('Style', 'edit', 'Position', [170, 315, 100, 30], ...

                      'FontSize', 12, 'Callback', @editCallback);

    % 添加 OP 输入框

    uicontrol('Style', 'text', 'Position', [40, 260, 100, 30], ...

              'String', '水泵 OP：', 'FontSize', 12, 'HorizontalAlignment', 'left');

    opEdit = uicontrol('Style', 'edit', 'Position', [100, 265, 150, 30], ...

                       'FontSize', 12, 'String', '1');

    uicontrol('Style', 'pushbutton', 'String', '设定 OP', ...

              'Position', [260, 265, 90, 30], 'FontSize', 12, ...

              'Callback', @setOP);

    % 信息提示

    infoText = uicontrol('Style', 'text', 'Position', [40, 220, 330, 30], ...

                         'String', '等待输入...', 'FontSize', 12, ...

                         'HorizontalAlignment', 'center');

    % 添加按钮：添加点

    uicontrol('Style', 'pushbutton', 'String', '添加点', ...

              'Position', [40, 150, 100, 40], 'FontSize', 12, ...

              'Callback', @addPoint);

    % 添加按钮：计算

    uicontrol('Style', 'pushbutton', 'String', '计算', ...

              'Position', [155, 150, 100, 40], 'FontSize', 12, ...

              'Callback', @calculateFit);

    % 添加按钮：退出

    uicontrol('Style', 'pushbutton', 'String', '退出', ...

              'Position', [270, 150, 100, 40], 'FontSize', 12, ...

              'Callback', @(src, event) close(f));

    % 显示输入的数据点的表格区域

    uicontrol('Style', 'text', 'Position', [400, 400, 200, 30], ...

              'String', '已输入数据点:', 'FontSize', 12, 'HorizontalAlignment', 'left');

    dataTable = uitable('Position', [400, 175, 220, 220], ...

                        'ColumnName', {'X', 'Y'}, ...

                        'RowName', [], 'FontSize', 12);

    % 显示传递函数输出的文本框

    uicontrol('Style', 'text', 'Position', [200, 95, 300, 30], ...

              'String', '一阶开环传递函数:', 'FontSize', 18, 'HorizontalAlignment', 'left');

    tfOutput = uicontrol('Style', 'edit', 'Position', [20, 10, 600, 80], ...

                         'FontSize', 18, 'HorizontalAlignment', 'center', 'Enable', 'inactive');

    % 编辑框回调：处理回车事件

    function editCallback(~, ~)

        enterCount = enterCount + 1;

        if enterCount == 1

            uicontrol(yEdit); % 第一次按回车，聚焦到 Y 输入框

        elseif enterCount == 2

            addPoint(); % 第二次按回车，执行添加点操作

            enterCount = 0; % 重置回车计数器

        end

    end

    % 添加点的回调

    function addPoint(~, ~)

        xStr = get(xEdit, 'String');

        yStr = get(yEdit, 'String');

        xVal = str2double(xStr);

        yVal = str2double(yStr);

        if isnan(xVal) || isnan(yVal)

            set(infoText, 'String', '⚠️ 无效输入，请输入数字');

            return;

        end

        X(end+1) = xVal;

        Y(end+1) = yVal;

        % 更新数据点表格

        set(dataTable, 'Data', [X(:), Y(:)]);

        % 清空输入框并回到X输入框

        set(xEdit, 'String', '');

        set(yEdit, 'String', '');

        uicontrol(xEdit);  % 将焦点设回X框

        set(infoText, 'String', sprintf('✅ 已添加点 (%.2f, %.2f)，共 %d 个点', xVal, yVal, length(X)));

    end

    % 设置 OP 的回调

    function setOP(~, ~)

        val = str2double(get(opEdit, 'String'));

        if isnan(val) || val <= 0

            set(infoText, 'String', '⚠️ OP值无效，请输入正数');

        else

            OP = val;

            set(infoText, 'String', sprintf('✅ 当前 OP = %.2f', OP));

        end

    end

    % 计算拟合

    function calculateFit(~, ~)

        if length(X) < 3

            set(infoText, 'String', '⚠️ 至少需要 3 个点才能拟合');

            return;

        end

        % 拟合一阶系统模型

        fit\_type = fittype('K \* (1 - exp(-x / tau))', ...

                           'independent', 'x', 'dependent', 'y');

        initial\_guess = [max(Y), mean(X)];

        [fit\_result, gof] = fit(X(:), Y(:), fit\_type, 'StartPoint', initial\_guess);

        % 输出传递函数数值

        K = fit\_result.K;

        tau = fit\_result.tau;

        G = K / (tau + 1); % 计算传递函数数值

        % 输出传递函数

        tfStr = sprintf('G(s) = %.2f / (%.2fs + 1)', K, tau);

        set(tfOutput, 'String', tfStr);

        % 输出结果

        disp('—— 拟合结果 ——');

        disp(['原始拟合 K = ', num2str(fit\_result.K)]);

        disp(['τ = ', num2str(tau)]);

        disp(['传递函数 G(s) = ', num2str(K), ' / (', num2str(tau), 's + 1)']);

        disp(['传递函数数值 = ', num2str(G)]);

        % 绘图：不考虑 OP 值

        figure;

        plot(X, Y, 'bo', 'MarkerFaceColor', 'b');

        hold on;

        x\_fit = linspace(min(X), max(X), 1000);

        y\_fit = K \* (1 - exp(-x\_fit / tau));

        plot(x\_fit, y\_fit, 'r-', 'LineWidth', 2);

        title('一阶系统阶跃响应拟合', 'FontSize', 14);

        xlabel('时间 (X)', 'FontSize', 12);

        ylabel('响应 (Y)', 'FontSize', 12);

        legend({'原始数据', '拟合曲线'}, 'FontSize', 12);

        grid on;

        % 输出信息

        set(infoText, 'String', '✅ 拟合完成（已计算传递函数数值），结果见图和命令行');

    end

end

2 . 51单片机程序源码  
  
#include"reg51.h"

#define uchar unsigned char

#define uint unsigned int

#define ulong unsigned long

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*引脚定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//l298n 用来驱动输出加压

sbit output=P2^5;

sbit in1=P2^3;

sbit in2=P2^4;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//AD芯片 用来读取压力

sbit CLOCK=P3^1;

sbit D\_IN=P2^7;

sbit D\_OUT=P2^6;

sbit \_CS=P3^0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//lcd液晶 用来显示

sbit lcdrs=P2^0;

sbit lcdrw=P2^1;

sbit lcden=P2^2;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//灯

sbit led=P3^2;

//蜂鸣器

sbit speaker=P3^3;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*引脚定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*变量定义（申明）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int freq=500,pwm=0;

uchar setflag=0,pageflag=0,ok=0;

float pwm\_temp=0;

float KP=100,KI=10,KD=0;

uint pul\_count=0;

float ek=0,ek1=0,ek2=0;

float keytemp=0;

uchar code str[16]=" welcome! ";

uchar code str1[16]=" temperature PID";

uchar code str2[16]="set now " ;

uchar code str3[16]="P: I: D:" ;

uchar code str4[16]="set-P: " ;

uchar code str5[16]="set-I: " ;

uchar code str6[16]="set-D: " ;

void Init(void);

uint adcread(uchar port);

void delay(uint n);

void writecom(uchar com);

void writedata(uchar date);

void initlcd();

float keyscan(void);

uchar keyscans(void);

void lcdnumdisplay(uchar pos,double f);

void lcdnumdisplays(uchar pos,double f);

uint read\_pul();

void mypid(float Kp,float Ki,float Kd,uint count,uint point);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*变量定义（申明）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

uchar i;float scantemp;

float adnum0=0;

float adnum1=0;

bit init0,init1,init2,init3,init4;

Init();

initlcd();

writecom(0x80);

for(i=0;i<16;i++)writedata(str[i]); //huanyin

writecom(0x80+0x40);

for(i=0;i<16;i++)writedata(str1[i]); //shuming

delay(2000);

// output=0;

// in1=1;in2=0;

// while(1);

while(1)

{

pwm\_temp=500\*(float)(adnum0)/4095;

mypid(KP,KI,KD,adnum1,pwm\_temp);

scantemp=keyscan();

if(adnum1>=pwm\_temp-1.5&&adnum1<=pwm\_temp+1.5)

{

led=0;

speaker=1;

}

else{

led=1;

speaker=0;

}

if(setflag==0)

{

adnum0=adcread(0);

adnum1=adcread(1)/45;

if(init0==0)

{

initlcd();

writecom(0x80);

for(i=0;i<16;i++)

writedata(str2[i]); //shuming

init0=1;init1=0;init2=0;init3=0;init4=0;

}

lcdnumdisplays(0x80+0x40,(float)pwm\_temp); //pwm\_temp

lcdnumdisplays(0x80+0x4a,(float)adnum1);

}

if(setflag!=0&&pageflag==0)

{

if(init1==0)

{

initlcd();

writecom(0x80);

for(i=0;i<16;i++)writedata(str3[i]);

init0=0;init1=1;init2=0;init3=0;init4=0;

}

lcdnumdisplays(0x80+0x40,KP);

lcdnumdisplays(0x80+0x46,KI);

lcdnumdisplays(0x80+0x4D,KD);

}

if(setflag!=0&&pageflag==1)

{

if(init2==0)

{

initlcd();

writecom(0x80);

for(i=0;i<16;i++)writedata(str4[i]);

init0=0;init1=0;init2=1;init3=0;init4=0;

}

lcdnumdisplays(0x80+0x40,scantemp);

if(ok==1){KP=scantemp;ok=0;}

}

if(setflag!=0&&pageflag==2)

{

if(init3==0)

{

initlcd();

writecom(0x80);

for(i=0;i<16;i++)writedata(str5[i]);

init0=0;init1=0;init2=0;init3=1;init4=0;

}

lcdnumdisplays(0x80+0x40,scantemp);

if(ok==1){KI=scantemp;ok=0;}

}

if(setflag!=0&&pageflag==3)

{

if(init4==0)

{

initlcd();

writecom(0x80);

for(i=0;i<16;i++)writedata(str6[i]);

init0=0;init1=0;init2=0;init3=0;init4=1;

}

lcdnumdisplays(0x80+0x40,scantemp);

if(ok==1){KD=scantemp;ok=0;}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\* PID算法控制函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 参数说明：

Kp: 比例系数

Ki: 积分系数

Kd: 微分系数

count: 当前测量值

point: 目标设定值 \*/

void mypid(float Kp,float Ki,float Kd,uint count,uint point)

{

static float Uk;

ek=point-count;

// if(ek>=5&&ek<=-5) //积分分离

{Uk=Kp\*(ek-ek1)+Ki\*ek+Kd\*(ek-2\*ek1+ek2);}

// else

// Uk=Kp\*ek;

pwm=Uk;

//lcdnumdisplays(0x80+0x4a,(float)pwm);

if(pwm>freq)pwm=freq;

if(pwm<=0){pwm=0;in1=0;in2=1;}

if(pwm>0) {in1=1;in2=0;}

ek2=ek1;

ek1=ek;

}

uint read\_pul()

{

uint t1,th1,th2;

uint val;

while(1)

{

th1=TH1;

t1=TL1;

th2=TH1;

if(th1==th2)

break;

}

val=th1\*256+t1;

return val;

}

void delay(uint n)

{

uint i,j;

for(i=n;i>0;i--)

for(j=1;j>0;j--);

}

void Init(void)//初始化函数

{

TMOD=0x51;

TH0=(65536-10)/256;

TL0=(65536-10)%256;

EA=1;

ET0=1;

TR0=1;

TH1=0;

TL1=0;

TR1=1;

}

void Timer\_0(void) interrupt 1

{

static ulong t\_count=0;

static uint num\_count=0;

TR0=0;

TH0=(65536-10)/256;

TL0=(65536-10)%256;

TR0=1;

num\_count++;

if(num\_count>freq)num\_count=0; //1khz;

if(num\_count<pwm)output=1;

else output=0;

}

//lcd写命令

void writecom(uchar com)

{

lcdrs=0;

P0=com;

delay(1);

lcden=1;

delay(1);

lcden=0;

}

//lcd写数据

void writedata(uchar date)

//初始化lcd

{

lcdrs=1;

P0=date;

delay(1);

lcden=1;

delay(1);

lcden=0;

}

void initlcd()

{

lcdrw=0;

writecom(0x38);delay(1);

writecom(0x0c);delay(1);

writecom(0x06);delay(1);

writecom(0x01);delay(5);

}

/\*adc读,用来读取当前的压力数据，通过AD0口进行读取\*/

uint adcread(uchar port)

{

uint ad=0,i;

CLOCK=0;

\_CS=0;

port<<=4;

for(i=0;i<12;i++)

{

if(D\_OUT)

ad|=0x01;

D\_IN=(bit)(port&0x80);

CLOCK=1;

delay(1);

CLOCK=0;

delay(1);

port<<=1;

ad<<=1;

}

\_CS=1;

ad>>=1;

return(ad);

}

//lcd显示数据1

void lcdnumdisplays(uchar pos,float f) //(0.001-99999) 精度低 但方便数据更新

{

uchar i;

writecom(pos);

if(f>65535&&f<0.001) for(i=0;i<5;i++)writedata(0x23);//超出范围 显示#

else if(f==0){writedata(0x30);for(i=0;i<4;i++)writedata(0x20);}

else

{

if((uint)f/10000!=0)

{

writedata((uint)f/10000+0x30);

writedata((uint)f%10000/1000+0x30);

writedata((uint)f%1000/100+0x30);

writedata((uint)f%100/10+0x30);

writedata((uint)f%10+0x30);

}

else

{

if((uint)f/1000!=0)

{

writedata(0+0x30);

writedata((uint)f/1000+0x30);

writedata((uint)f%1000/100+0x30);

writedata((uint)f%100/10+0x30);

writedata((uint)f%10+0x30);

}

else

{

if((uint)f/100!=0)

{

writedata((uint)f/100+0x30);

writedata((uint)f%100/10+0x30);

writedata((uint)f%10+0x30);

writedata(0x2e);

writedata((uint)(f\*10)%10+0x30);

}

else

{

if((uint)f/10!=0)

{

writedata((uint)f/10+0x30);

writedata((uint)f%10+0x30);

writedata(0x2e);

writedata((uint)(f\*10)%10+0x30);

writedata((uint)(f\*100)%10+0x30);

}

else

{

writedata((uint)f%10+0x30);

writedata(0x2e);

writedata((uint)(f\*10)%10+0x30);

writedata((uint)(f\*100)%10+0x30);

writedata((uint)(f\*1000)%10+0x30);

}

}

}

}

}

}

//lcd显示数据2

void lcdnumdisplay(uchar pos,float f0) //(0.00001-99999.99999) 精度高 但数据需刷屏更新

{

uchar temp;

ulong f;

writecom(pos);

f=(ulong)f0;

temp=f/10000; //整数部分

if(temp!=0)

{

writedata(temp+0x30);

writedata(f%10000/1000+0x30);

writedata(f%1000/100+0x30);

writedata(f%100/10+0x30);

writedata(f%10+0x30);

}

else

{

temp=f%10000/1000;

if(temp!=0)

{

writedata(temp+0x30);

writedata(f%1000/100+0x30);

writedata(f%100/10+0x30);

writedata(f%10+0x30);

}

else

{

temp=f%1000/100;

if(temp!=0)

{

writedata(temp+0x30);

writedata(f%100/10+0x30);

writedata(f%10+0x30);

}

else

{

temp=f%100/10;

if(temp!=0)

{

writedata(temp+0x30);

writedata(f%10+0x30);

}

else

{

temp=f%10;

if(temp!=0)

{

writedata(temp+0x30);

}

else writedata(0+0x30);

}

}

}

}

if((ulong)(f0\*10)%10!=0||(ulong)(f0\*100)%10!=0||(ulong)(f0\*1000)%10!=0||(ulong)(f0\*10000)%10!=0)

{

writedata(0x2e);

temp=(ulong)(f0\*10000)%10;

if(temp!=0)

{

writedata((ulong)(f0\*10)%10+0x30);

writedata((ulong)(f0\*100)%10+0x30);

writedata((ulong)(f0\*1000)%10+0x30);

writedata(temp+0x30);

}

else

{

temp=(ulong)(f0\*1000)%10;

if(temp!=0)

{

writedata((ulong)(f0\*10)%10+0x30);

writedata((ulong)(f0\*100)%10+0x30);

writedata(temp+0x30);

}

else

{

temp=(ulong)(f0\*100)%10;

if(temp!=0)

{

writedata((ulong)(f0\*10)%10+0x30);

writedata(temp+0x30);

}

else

{

temp=(ulong)(f0\*10)%10;

if(temp!=0)

writedata(temp+0x30);

}

}

}

}

}

uchar keyscans(void)//矩阵键盘扫描

{

uchar i, j, temp, Buffer[4] = {0xef, 0xdf, 0xbf, 0x7f};

for(j = 0; j < 4; j++) //循环四次

{

P1 = Buffer[j]; //在P1高四位分别输出一个低电平

temp = 0x01; //计划先判断P1.0位

for(i = 0; i < 4; i++)

{

if(!(P1 & temp)) //从P1低四位，截取1位

return (i + j \* 4); //返回取得的按键值

temp <<= 1; //判断的位，左移一位

}

}

return 16;

}

float keyscan(void)//矩阵键盘扫描

{

static uchar dianflag=0;

static uchar diancount=0;

static uchar k=0;

uchar temp;

P1=0xfe;

temp=P1;

temp=temp&0xf0;

if(temp!=0xf0)

{

delay(100);

if(temp!=0xf0)

{

switch(temp)

{

case 0xe0: {k=1;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0xd0: {k=2;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0xb0: {k=3;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0x70: if(setflag==1) {pageflag++;dianflag=0;diancount=0;ok=0;keytemp=0;}

break;

}

while(temp!=0xf0)

{

temp=P1;

temp=temp&0xf0;

}

}

}

P1=0xfd;

temp=P1;

temp=temp&0xf0;

if(temp!=0xf0)

{

delay(100);

if(temp!=0xf0)

{

switch(temp)

{

case 0xe0: {k=4;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0xd0: {k=5;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0xb0: {k=6;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0x70: if(setflag==1) {pageflag--;dianflag=0;diancount=0;ok=0;keytemp=0;}

break;

}

while(temp!=0xf0)

{

temp=P1;

temp=temp&0xf0;

}

}

}

P1=0xfb;

temp=P1;

temp=temp&0xf0;

if(temp!=0xf0)

{

delay(100);

if(temp!=0xf0)

{

switch(temp)

{

case 0xe0: {k=7;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0xd0: {k=8;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0xb0: {k=9;if(dianflag==0)keytemp=keytemp\*10+k;if(dianflag==1){

diancount++;

if(diancount==1)keytemp=keytemp+k\*0.1;

if(diancount==2)keytemp=keytemp+k\*0.01;

if(diancount==3)keytemp=keytemp+k\*0.001;

if(diancount==4)keytemp=keytemp+k\*0.0001;

if(diancount==5)keytemp=keytemp+k\*0.00001;

}}

break;

case 0x70: {setflag=!setflag;pageflag=0;ok=0;}

break;

}

while(temp!=0xf0)

{

temp=P1;

temp=temp&0xf0;

}

}

}

P1=0xf7;

temp=P1;

temp=temp&0xf0;

if(temp!=0xf0)

{

delay(100);

if(temp!=0xf0)

{

switch(temp)

{

case 0xe0: dianflag=1;

break;

case 0xd0: {k=0;if(dianflag==0)keytemp=keytemp\*10+k;}

break;

case 0xb0: {dianflag=0;diancount=0;ok=1;}

break;

case 0x70: {dianflag=0;diancount=0;keytemp=0;ok=0;}

break;

}

}

while(temp!=0xf0)

{

temp=P1;

temp=temp&0xf0;

}

}

return keytemp;

}

3. PID参数快速验证工具

function open\_loop\_transfer\_close\_loop()

% 创建用户友好的界面

fig = uifigure('Name', 'PID 控制锅炉蒸汽压力液位系统', 'Position', [100, 100, 600, 500], 'Color', '#f2f2f2');

% 创建开环传递函数输入框

uilabel(fig, 'Position', [20, 460, 500, 30], 'Text', '请输入锅炉蒸汽压力开环传递函数的系数：', 'FontSize', 16, 'FontWeight', 'bold', 'FontColor', [0.2, 0.2, 0.2]);

G\_s\_input = uieditfield(fig, 'text', 'Position', [20, 420, 500, 30], 'FontSize', 14);

G\_s\_input.Value = '0.76 / (272.71 \* s + 1)'; % 默认值

% 创建PID控制器参数输入框

uilabel(fig, 'Position', [20, 360, 500, 30], 'Text', '请输入PID控制器参数:', 'FontSize', 16, 'FontWeight', 'bold', 'FontColor', [0.2, 0.2, 0.2]);

% Kp输入框

uilabel(fig, 'Position', [20, 320, 60, 30], 'Text', 'Kp:', 'FontSize', 16, 'FontColor', [0.2, 0.2, 0.2]);

Kp\_input = uieditfield(fig, 'numeric', 'Position', [80, 320, 100, 30], 'FontSize', 14);

Kp\_input.Value = 4; % 默认Kp

% Ki输入框

uilabel(fig, 'Position', [20, 280, 60, 30], 'Text', 'Ki:', 'FontSize', 16, 'FontColor', [0.2, 0.2, 0.2]);

Ki\_input = uieditfield(fig, 'numeric', 'Position', [80, 280, 100, 30], 'FontSize', 14);

Ki\_input.Value = 0.001; % 默认Ki

% Kd输入框

uilabel(fig, 'Position', [20, 240, 60, 30], 'Text', 'Kd:', 'FontSize', 16, 'FontColor', [0.2, 0.2, 0.2]);

Kd\_input = uieditfield(fig, 'numeric', 'Position', [80, 240, 100, 30], 'FontSize', 14);

Kd\_input.Value = 0; % 默认Kd

% 让用户输入液位设定值

uilabel(fig, 'Position', [20, 180, 500, 30], 'Text', '请输入液位设定值 X:', 'FontSize', 16, 'FontWeight', 'bold', 'FontColor', [0.2, 0.2, 0.2]);

x\_input = uieditfield(fig, 'numeric', 'Position', [20, 150, 100, 30], 'FontSize', 14);

x\_input.Value = 20; % 默认设定值

% 创建一个% 创建一个文本框来显示闭环传递函数

uilabel(fig, 'Position', [20, 100, 500, 30], 'Text', '闭环传递函数：', 'FontSize', 16, 'FontWeight', 'bold', 'FontColor', [0.2, 0.2, 0.2]);

output\_text = uitextarea(fig, 'Position', [20, 18, 560, 80], 'FontSize', 22, 'Editable', 'off', 'BackgroundColor', '#f2f2f2', 'FontName', 'Courier New');

output\_text.Value = {'等待计算结果...'}; % 默认值

% 将按钮移到右侧，保持空间更为合理

% 创建一个按钮来计算并显示阶跃响应

calc\_button = uibutton(fig, 'Position', [320, 210, 200, 40], 'Text', '计算并显示阶跃响应', 'FontSize', 16, 'BackgroundColor', '#4CAF50', 'FontColor', 'white', 'ButtonPushedFcn', @(btn, event) calculate\_response(G\_s\_input, Kp\_input, Ki\_input, Kd\_input, x\_input, fig));

% 创建一个退出按钮

exit\_button = uibutton(fig, 'Position', [320, 150, 200, 40], 'Text', '退出', 'FontSize', 16, 'BackgroundColor', '#f44336', 'FontColor', 'white', 'ButtonPushedFcn', @(btn, event) close(fig));

% 在回调函数中进行计算和显示

function calculate\_response(G\_s\_input, Kp\_input, Ki\_input, Kd\_input, x\_input, fig)

% 获取用户输入的开环传递函数

G\_s\_str = G\_s\_input.Value;

G\_s = str2func(['@(s) ' G\_s\_str]); % 将字符串转为函数

% 获取PID参数

Kp = Kp\_input.Value;

Ki = Ki\_input.Value;

Kd = Kd\_input.Value;

% 获取液位设定值

X = x\_input.Value;

% 2. 系统参数设定

delay\_time = 2; % 延迟时间 2秒

s = tf('s'); % 定义拉普拉斯变量 s

% 添加延迟项（使用Pade近似）

G\_delay = pade(exp(-s\*delay\_time), 1); % 1阶Pade近似表示延迟

% 输入信号设定（阶跃输入信号）

input\_value = X; % 目标液位为用户设定的液位

% 控制器的传递函数

G\_PID = @(s) Kp + Ki / s + Kd \* s;

% 计算系统的开环传递函数（加延迟）

G\_open = G\_s(s) \* G\_delay; % 锅炉蒸汽压力的开环传递函数 G(s) 乘上延迟项

% PID 控制器的传递函数

G\_pid\_tf = Kp + Ki / s + Kd \* s; % PID 控制器传递函数

% 计算闭环传递函数（加上延迟）

G\_closeloop = feedback(G\_pid\_tf \* G\_open, 1); % 闭环传递函数，使用 MATLAB 的 feedback 函数

% 输出闭环传递函数

numerator = G\_closeloop.Numerator{1}; % 分子

denominator = G\_closeloop.Denominator{1}; % 分母

disp('闭环传递函数 G(s) (系统):');

fprintf('G(s) = (%.3f\*s^2 + %.3f\*s + %.3f) / (%.3f\*s^2 + %.3f\*s + %.3f)\n', ...

numerator(1), numerator(2), numerator(3), denominator(1), denominator(2), denominator(3));

% 设置文本框内容

output\_text.Value = {['G(s) = (' num2str(numerator(1)) '\*s^2 + ' num2str(numerator(2)) '\*s + ' num2str(numerator(3)) ') / (' ...

num2str(denominator(1)) '\*s^2 + ' num2str(denominator(2)) '\*s + ' num2str(denominator(3)) ')']};

% 关闭之前打开的图形窗口

close all;

% 绘制阶跃响应

figure;

step(X\*G\_closeloop); % 绘制闭环传递函数的阶跃响应

title('系统的阶跃响应');

xlabel('时间 (秒)');

ylabel('液位响应');

grid on;

% 设置X轴范围为0到1000秒

xlim([0 1000]);

end

end