目录

1. 实验任务

本实验的**目的**是实现对连续搅拌反应罐系统（CSTR）的控制器设计。在基于gPROMS 数值建模仿真的虚拟对象中，将不同比例的原材料甲醇（CH_3OH)和乙酸（CH_3COOH）从流入口进入反应罐后充分搅拌，生成乙酸甲酯（CH_3COOCH_3）和水。本次实验中，反应器温度、成分含量、液位高度等可以根据需要在 gPROMS 中配置后在 MATLAB 中采集到，入口温度、反应器外部加热量和入口流速作为操纵量均可在 MATLAB 控制器计算后传回 gPROMS 实施对应控制。

本课程设计**基本任务**是完成对基于gPROMS的CSTR系统的控制器设计，大致主要可以分四个部分，数据采集与模型辨识、控制系统设计与仿真、实物仿真与参数调整以及实验报告的撰写：

1. **数据采集与模型辨识**
   熟悉装置，使用计算机事先准备好的程序进行数据采集，在课后完成实验系统的模型辨识。
2. **控制系统设计与仿真**
   基于上一环节辨识模型，自行完成建模和辨识环节。设计控制器，并在此基础上完成仿真工作。
3. **实物仿真与参数调整**
   按照事先确定的时间整组来实验室，基于设计的控制器进行实物控制，在线调整系统结构和参数。
4. **完成实验报告的撰写**

图 1 电院 2-100 实验环境

实验在电院 2-100 进行，利用已经搭建好 gPROMS 以及 MATLAB 平台的仿真环境完成。其中，gPROMS 完成对 CSTR 对象的模拟设计，MATLAB 完成实验交互、控制器算法设计等实验要求。

2. 实验基本对象

2.1. CSTR系统介绍

连续搅拌反应釜(Continuously Stirred Tank Reactor，CSTR)是化工生产中一种极为常见的反应容器，在化工领域有着广泛的使用，其中，在三大合成材料的生产过程中，其应用数量可以达到90%以上，其控制系统的稳定性和性能至关重要。CSTR系统具有一些较难攻克的难点，其系统自身具有很强的时滞特性与非线性，导致其控制系统大都算法复杂且鲁棒性、实时性相对来说较差[1]。

CSTR作为一个典型的非线性系统，具有高非线性、大时滞性、不确定性、高危险性等特点，不利于常规算法控制。

对于CSTR控制系统来说，对产品质量和产量影响最大的参数是温度、压力和浓度[2]，通常情况下浓度的直接测量比较困难，而对压力变量的控制又不容易实现。因此，在CSTR控制系统的设计过程中，习惯取温度为控制器的输出。在实际工业操作中，通过向反应器内加入冷却剂的方式，对整个反应器进行溢出热量的去除，来确保反应温度保持在工艺要求的特定温度附近。

2.2. 对象基本组成

CSTR的具体结构图如图2所示，CSTR主要由搅拌容器与搅拌机两个部分组成，而搅拌容器分为筒体、夹套换热元件和内构件三个部分，搅拌机分为搅拌轴、密封装置、搅拌器和传动装置几个部分。

反应罐有一个入口和一个出口，其出口安置在高于反应罐底部hp处。反应罐外部包裹一层传热装置，可额外给设备提供热量。实验控制变量对象为反应罐温度h_{out}，可操作控制对象为入口温度h_{in}与夹套传热功率 Q。

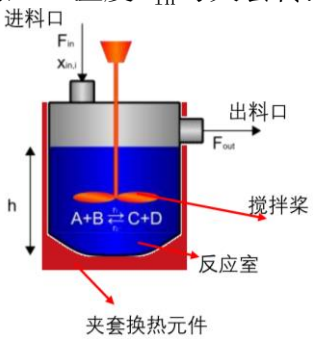

图 2　连续反应搅拌罐

本次实验以基本的酯化反应为例进行控制器的设计，酯化反应方程式为：

$$CH_3OH + CH_3COOH \rightleftharpoons CH_3COOCH_3 + H_2O \qquad (2-1)$$

2.3. 基本假设

实验中为了方便建模与仿真，假设设备为完全理想化，使反应可以充分发生，基于此建模过程中做如下假设：

假设2.3.1. 反应釜搅拌为完全理想，物料可完全混合接触，反应可充分发生；

假设2.3.2. 反应物料与冷却剂的一般物理特性为常数，不随温度改变而改变；

假设2.3.3. 反应釜外环境理想，无热量交换；

假设 2.3.4. 反应进程中，进料和出料的流量相同。

3. 系统辨识

3.1 对象分析

基于对 CSTR 系统物理模型的理解，我们可以简单列出系统所满足的微分方程，即从质量守恒以及能量守恒进行机理建模。

列出质量守恒方程如式(3-1)、(3-2)所示：

$$\frac{\partial N_i}{\partial t} = F_{in}x_{in,i} - F_{\text{out}}\ x_i + N_T \sum_{j \in REAC} v_{ij} r_j, i \in \text{COMP} \qquad (3-1)$$

$$r_j = k_{0j} e^{\frac{-E_{A_j}}{RT}} \prod_{i \in \text{COMP}} x_i^{a_{ij}}, 1 \le j \le N_r \qquad (3-2)$$

能量守恒方程如式(3-3)所示：

$$\frac{\partial H}{\partial t} = F_{in}(h_{in}) - F_{\text{out}}\ h_{\text{out}}\ + N_T \sum_{j=1}^{N_r} r_j\left(-\Delta h_{r,j}\right) + Q \qquad (3-3)$$

基于能量守恒和物料守恒进行推导，可以明显看出 CSTR 系统为典型的非线性系统。

3.2 数据采集

在 MATLAB 中，我们设计脚本文件对 gPROMS 的黑箱模型进行了批量阶跃测试，并将采集到的数据存储在 txt 文件中。

我们进行了 u1 以 2 为间隔、u2 以 5 为间隔的打表式采样方法，依次进行阶跃，共计采集数据 80 组。

部分阶跃测试图如下所示

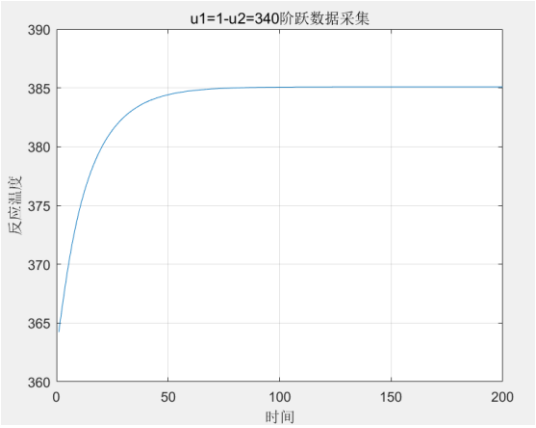

图 3　u1=1-u2=340 阶跃测试图

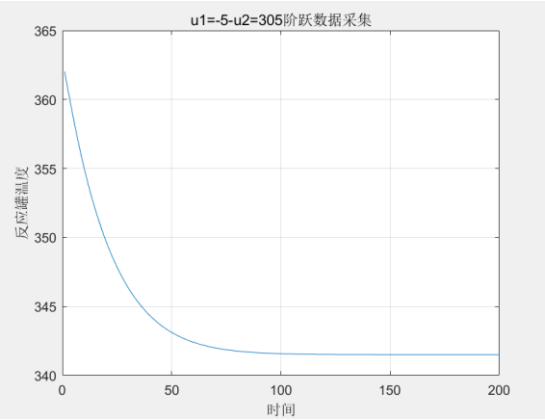

图 4　u1=-5-u2=305 阶跃测试图

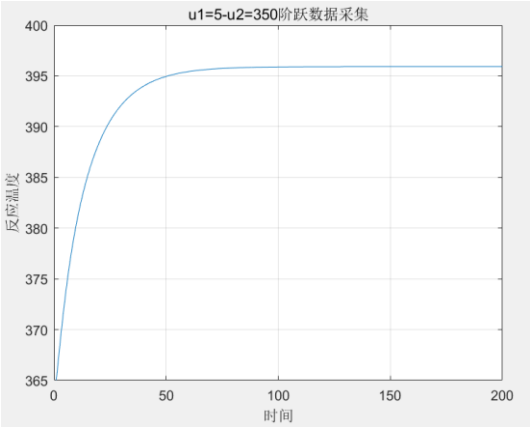

图 5　u1=5-u2=350 阶跃测试图

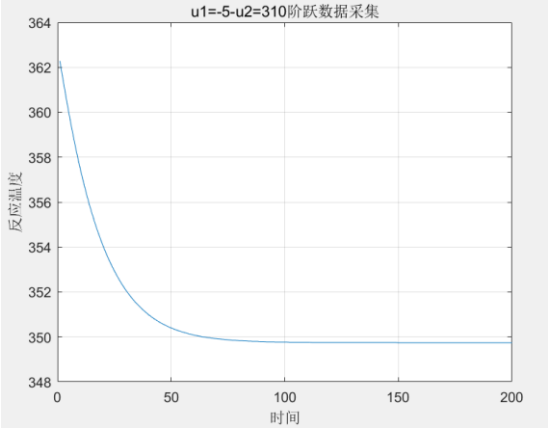

图 6　u1=-5-u2=310 阶跃测试图

完整采集得到的 txt 存储如下所示：

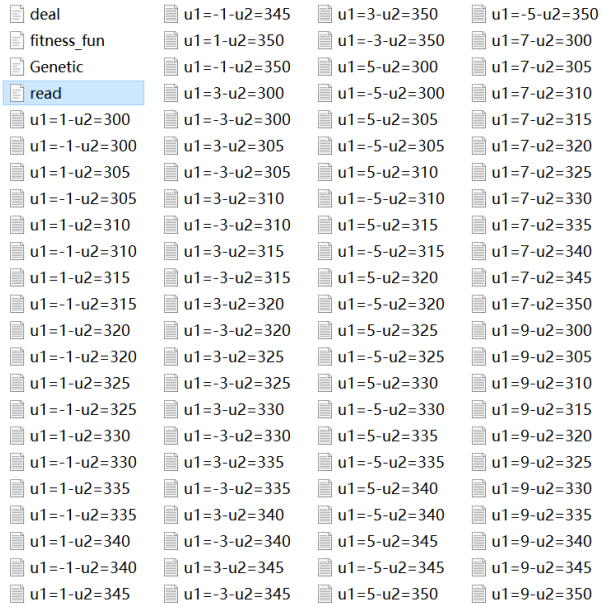

图 7　采集的 txt 目录

3.3 非线性辨识

线性系统模型辨识常用的方法为机理建模法与实验建模法两种，迁移到 CSTR 系统作为典型的非线性系统，我们认为也有两种基本的模型辨识方法。

其一是基于上文的能量守恒与质量守恒方程进行机理建模，之后进行参数的无量纲化求解。这一方法需要辨识的模型参数较多，且需要采用多参数逼近的方法进行求解，较难求解。

另一种方法采用局部线性化的基本思路，基于黑箱思想与阶跃测试，对采样点处的模型进行打表辨识，得到系统在不同输入参数下的模型参数。考虑《过程控制系统》这一课程的所学知识，我们采用了这一方法，并基于最小二乘法进行实现。

探究非线性对象局部线性化的可行性，我们对不同的数据进行了拟合，可以看到，系统在一定时间内有着较好的线性，证明了这一思路的可行性，作出效果图如下所示。

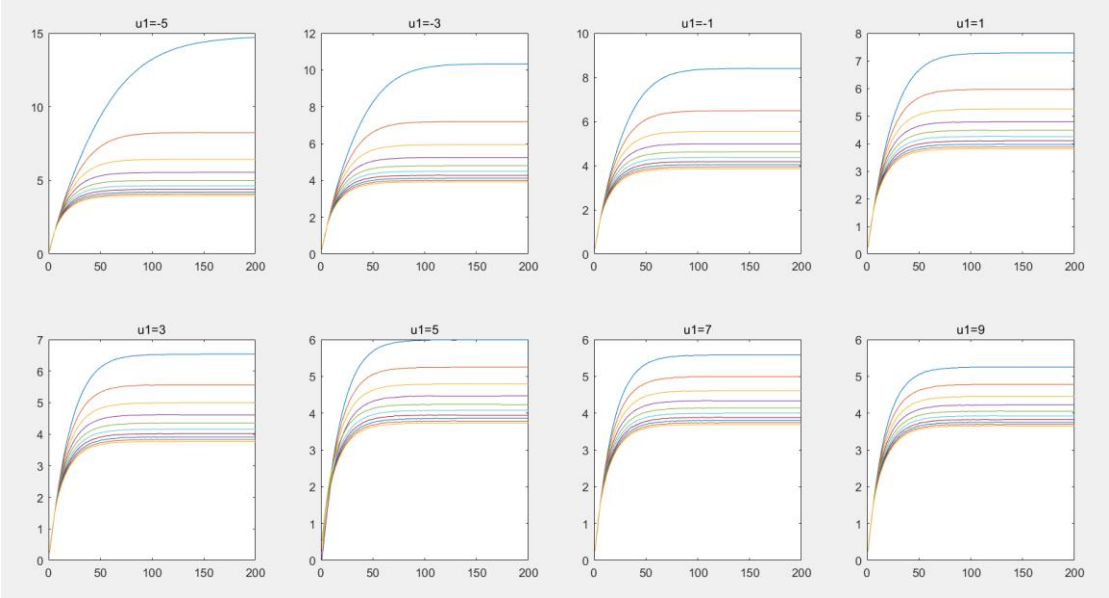

图 8 局部线性化多图

在助教老师的建议下，我们将 CSTR 输入输出进行了模型简化，即将系统看作一阶加纯滞后的模型进行辨识。系统基本传递函数如下所示：

$$G(s)=\frac{b}{a\mathbf{s}+1}e^{-Ls} \qquad (3-4)$$

最小二乘法基本原理如下，假设系统的传递函数为：

$$G(s)=\frac{b_1s^{n-1}+b_2s^{n-2}+\cdots+b_n}{s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_n}\mathrm{e}^{-Ls} \qquad (3-5)$$

在时域中，进行化简与推导得到：

$$y(t)=-a_1\int_{[0,t]}^{(1)}y(t)-a_2\int_{[0,t]}^{(2)}y(t)\cdots a_{n-1}\int_{[0,t]}^{(n-1)}y(t)-a_n\int_{[0,t]}^{(n)}y(t)$$
$$+\hbar b_1(t-L)+\frac{\hbar}{2}b_2(t-L)^2+\cdots+\frac{\hbar}{(n-1)!}b_{n-1}(t-L)^{n-1}+\frac{\hbar}{n!}b_n(t-L)^n \qquad (3-6)$$

则对于我们需要设计的一阶系统，有：

$$y(t)=-a_1\int_0^t y(\tau)d\tau-b_1L\hbar+b_1t\hbar \qquad (3-7)$$

考虑最小二乘法的思想，我们进行参数整理，并得到求解公式：

$$\theta=(\phi^T\phi)^{-1}\phi^TY \qquad (3-8)$$

其中：

$$\begin{cases}\gamma(t)=y(t),\\ \phi(t)^T=\left[-\int_0^t y(\tau)\mathrm{d}\tau-h\ th\right]\\ \theta=[a_1\quad b_1L\quad b_1]\end{cases} \qquad (3-9)$$

求解得到：

$$\begin{bmatrix}a_1\\b_1\\L\end{bmatrix}=\begin{bmatrix}\theta_1\\\theta_3\\\theta_2/\theta_3\end{bmatrix} \qquad (3-10)$$

部分辨识效果图如下所示：

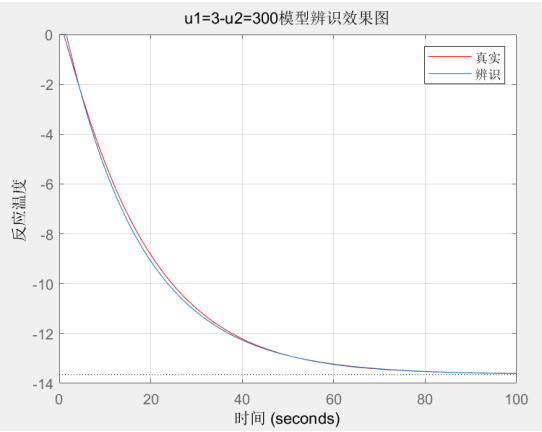

图 9　u1=3-u2=300 辨识图

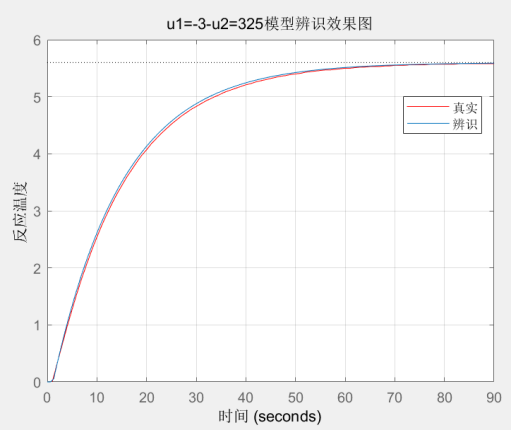

图 10　u1=3-u2=300 辨识图

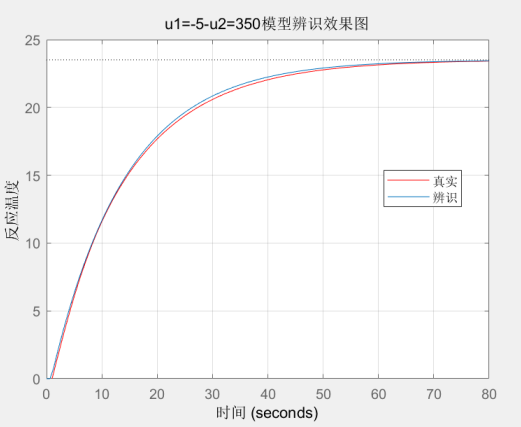

图 11　u1=-5-u2=350 辨识图

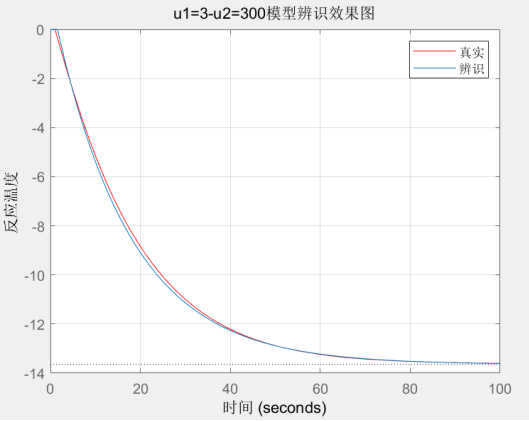

图 12　u1=3-u2=300 辨识图

从上图可以看到，一阶加纯滞后模型对与采集到的数据有非常好的辨识效果，实际数据曲线与辨识模型得到的曲线几乎重合。

我们编写了辨识脚本对所有数据依次打表处理，并将辨识得到的参数存储在同一个 xlsx 文件中便于查找，打表得到的参数如下所示：

Q	T	a1	b1	L	t_i	Q	T	a1	b1	L	t_i	Q	T	a1	b1	L	t_i
-5	300	0.033318	-1.16189	0.554955	361.77	-1	335	0.073698	1.142729	0.958228	363.84	5	315	0.07076	0.332529	1.160331	363.13
-5	305	0.052236	-1.07484	1.603637	362.03	-1	340	0.07423	1.430649	0.887007	364.11	5	320	0.072273	0.66144	1.128679	363.14
-5	310	0.060555	-0.76014	1.560314	362.28	-1	345	0.07457	1.709817	0.82473	364.39	5	325	0.073254	0.942651	1.013012	363.66
-5	315	0.065286	-0.41592	1.438064	362.54	-1	350	0.074821	1.983237	0.771067	364.67	5	330	0.07378	1.230743	0.922953	363.92
-5	320	0.069303	-0.07632	1.319	362.81	1	300	0.056107	-0.96241	1.607569	362.13	5	335	0.074369	1.513608	0.868242	364.19
-5	325	0.070461	0.255797	1.200926	363.07	1	305	0.062655	-0.63291	1.518355	362.38	5	340	0.07461	1.785884	0.803596	364.46
-5	330	0.072069	0.57586	1.121011	363.34	1	310	0.066535	-0.29216	1.382668	362.64	5	345	0.074862	2.053829	0.756732	364.74
-5	335	0.072988	0.88407	1.011694	363.6	1	315	0.067616	0.040549	1.20611	362.9	5	350	0.075037	2.317267	0.71273	365.02
-5	340	0.073705	1.181489	0.942733	363.88	1	320	0.070875	0.363457	1.164111	363.16	7	300	0.064412	-0.50159	1.470955	362.48
-5	345	0.074258	1.471914	0.873382	364.15	1	325	0.072269	0.675335	1.068521	363.42	7	305	0.067924	-0.16653	1.386358	362.73
-5	350	0.07458	1.753068	0.811765	364.43	1	330	0.073296	0.976833	0.997977	363.69	7	310	0.070201	0.160122	1.250524	362.99
-3	300	0.044441	-1.18471	1.426443	361.89	1	335	0.073868	1.266886	0.920719	363.96	7	315	0.071464	0.473702	1.130813	363.25
-3	305	0.056762	-0.93948	1.605581	362.15	1	340	0.074348	1.550144	0.857379	364.23	7	320	0.072808	0.779012	1.057279	363.51
-3	310	0.063003	-0.6041	1.503637	362.4	1	345	0.074694	1.826844	0.797703	364.5	7	325	0.073585	1.07265	0.972607	363.77
-3	315	0.067135	-0.26207	1.401668	362.66	1	350	0.074945	2.097452	0.750706	364.78	7	330	0.074045	1.35513	0.898966	364.04
-3	320	0.070458	0.075461	1.30259	362.92	3	300	0.059558	-0.81275	1.57931	362.24	7	335	0.074498	1.632334	0.841472	364.31
-3	325	0.071092	0.397993	1.160438	363.19	3	305	0.064831	-0.47701	1.467805	362.5	7	340	0.074799	1.902803	0.787388	364.58
-3	330	0.072418	0.711652	1.058571	363.45	3	310	0.067622	-0.13865	1.280055	362.76	7	345	0.074968	2.166214	0.739106	364.86
-3	335	0.073437	1.015412	0.989252	363.72	3	315	0.069799	0.188263	1.180992	363.01	7	350	0.075105	2.42686	0.693176	365.13
-3	340	0.073977	1.307351	0.908499	363.99	3	320	0.071722	0.504931	1.134574	363.28	9	300	0.066277	-0.34626	1.416627	362.6
-3	345	0.074395	1.591203	0.846271	364.27	3	325	0.072847	0.810699	1.043736	363.54	9	305	0.0663	-0.0146	1.204627	362.85
-3	350	0.074702	1.868764	0.790466	364.55	3	330	0.073591	1.105056	0.96779	363.81	9	310	0.070735	0.304183	1.182921	363.11
-1	300	0.051342	-1.09406	1.594737	362.01	3	335	0.074125	1.391461	0.888851	364.07	9	315	0.072269	0.613606	1.109688	363.37
-1	305	0.060043	-0.78805	1.570328	362.26	3	340	0.074498	1.668692	0.832289	364.35	9	320	0.07299	0.909032	1.007714	363.63
-1	310	0.065147	-0.44864	1.462581	362.52	3	345	0.074781	1.941044	0.776825	364.62	9	325	0.073718	1.197366	0.933257	363.89
-1	315	0.068559	-0.10911	1.340804	362.78	3	350	0.074969	2.207335	0.728281	364.9	9	330	0.074267	1.477312	0.876032	364.16
-1	320	0.070454	0.22129	1.223427	363.04	5	300	0.062211	-0.65756	1.522982	362.36	9	335	0.074575	1.748534	0.81377	364.43
-1	325	0.071723	0.53838	1.103181	363.3	5	305	0.06622	-0.31986	1.378481	362.62	9	340	0.074847	2.015291	0.764113	364.7
-1	330	0.072977	0.846412	1.036612	363.57	5	310	0.078299	0.013341	1.554139	362.87	9	345	0.074998	2.27625	0.715674	364.97
												9	350	0.075158	2.534478	0.679062	365.25

图 13　打表辨识得到的参数

由于采集到的数据为离散数据，无法对随意的输入[u1,u2]进行处理。我们考虑了两种思路，对[u1,u2]进行离散化处理，以及对传递函数模型进行插值连续化处理。

对辨识得到的参数作出八组不同入料温度下的变化曲线图，下图以 b、L 参数为例，可以看到，这些参数基本呈线性变化的趋势，因而我们可以选用第二种方式，即对对传递函数模型进行插值连续化处理。

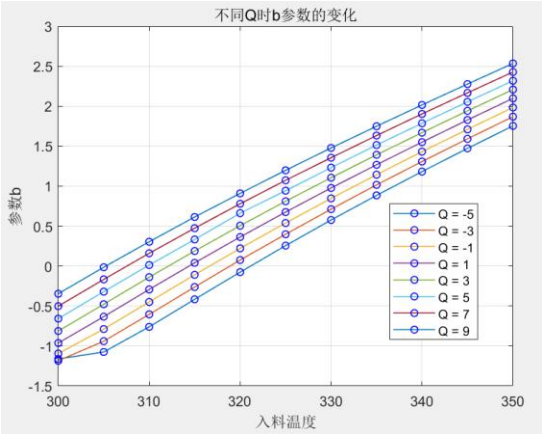

图 13 不同入料温度 b 参数的变化

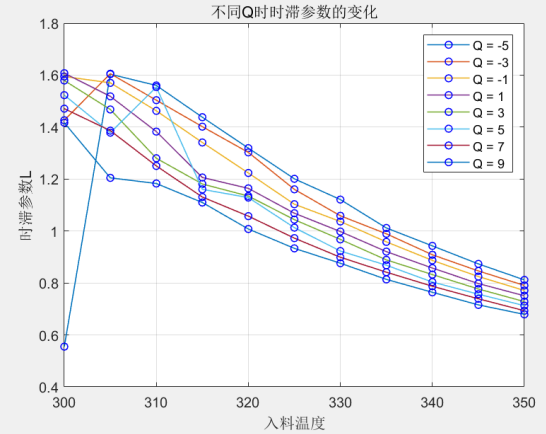

图 14 不同入料温度 L 参数的变化

这里我们选用经典的双线性插值方法对离散模型进行连续补全，即对下图所示的双自变量系统，先对 x 方向(入料温度 u1 方向)进行插值处理，再对 y 方向（散热功率参数 Q 方向）进行插值处理。

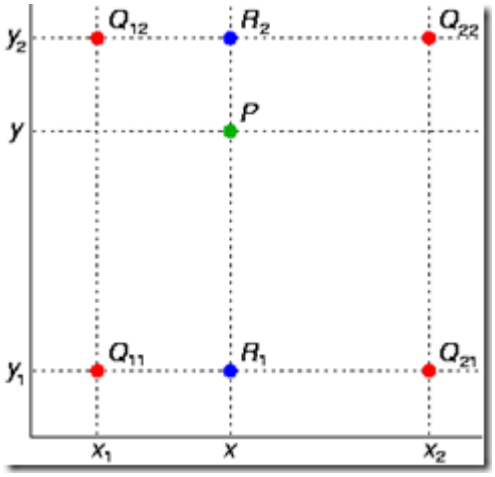

图 15 双线性插值算法

先对 x 方向进行插值处理：

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{Where } R_1 = (x, y_1) \qquad (3-11)$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{Where } R_2 = (x, y_2) \qquad (3-12)$$

然后再 y 方向进行插值：

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \qquad (3-13)$$

则插值得到的结果为：

$$f(x,y) \approx \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y)$$
$$+ \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1) \qquad (3-14)$$

3.4 辨识效果

我们在 2-100 实验室实地对模型辨识的结果与实际系统输出进行了验证，效果图如下所示。

在给定输入的情况下进行阶跃测试如下：

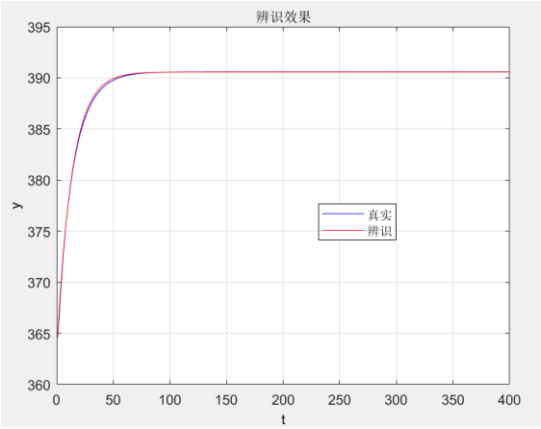

图 16 定输入阶跃测试 1

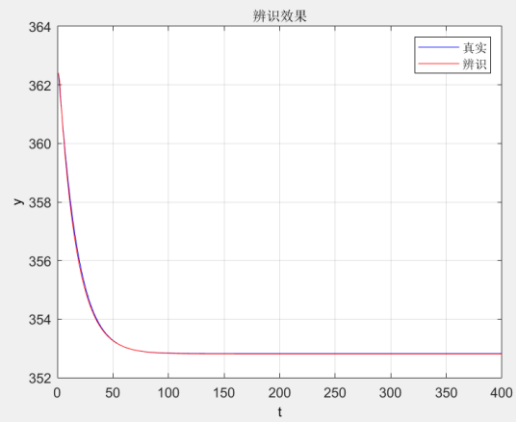

图 17 定输入阶跃测试 2

在连续多次变输入的情况下进行测试如下：

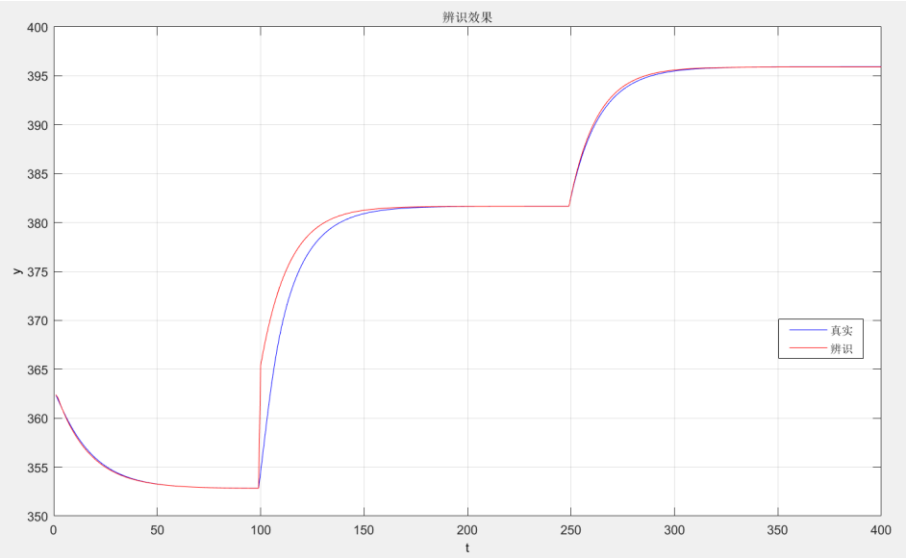

图 18 不同入料温度 b 参数的变化

在实验中，我们也发现某些输入下，辨识效果并非很好。我们认为这是我们初次采样的数据不多、采用插值方法产生的误差的缘故。在理想情况下，也就是采样打表足够细致的情况下，这一情况应该可以得到很大的改善。

4. 控制器设计

我们完成了以下三种控制器的设计，即 PID 控制器、BangBang 控制器、PID-Bangbang 双模控制器。

4.1 PID控制器

PID 控制是最早发展起来的控制策略之一，由于其算法简单、鲁棒性好、可靠性高，参数易于整定等优点而被广泛应用[3]。PID 控制器包括比例环节 P、微分环节 I、积分环节 D，其基本控制流程方块图如下所示：

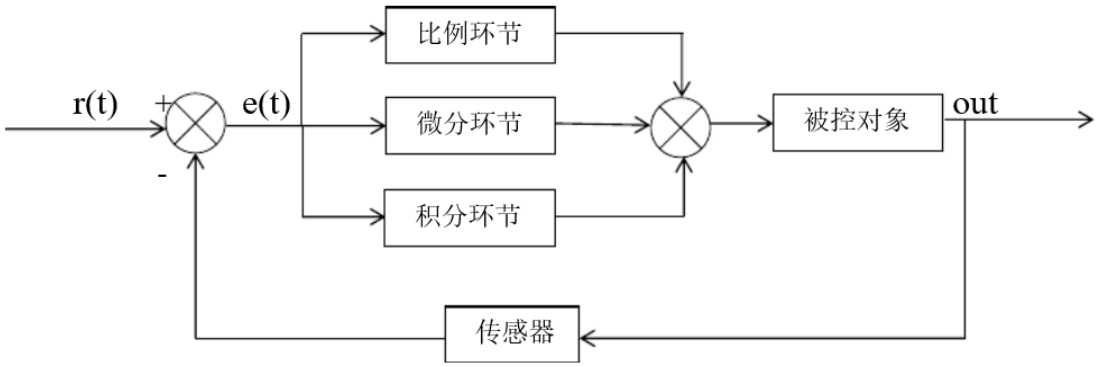

图 19 不同入料温度 b 参数的变化

PID 控制的基本方法是根据系统输入与预定输出的偏差的大小运用比例、积分、微分计算出一个控制量，将这个控制量输入系统，获得输出量，通过反馈回路再次检测该输出量的偏差，循环上述过程，以使输出达到预定值。

在 PID 算法中，比例环节 P 的作用是成比例地反映控制系统的偏差信号 e(t)，一旦产生偏差，比例控制环节立即产生控制作用以减小偏差。积分环节 I 的作用是消除静差，提高系统的无差度。微分环节 D 的作用是反映偏差信号的变化趋势，能够在偏差信号变化之前先引入一个有效的早期修正信号来提前修正偏差，加快系统的动作速度，减少调节时间。

位置式基本公式为：

$$u = K_{\mathrm{P}}e + K_{\mathrm{I}}\int e\mathrm{d}t + K_{\mathrm{D}}\dot{e} \qquad (4-1)$$

在本实验中，由于是对离散工业系统进行控制，因而要采用离散式 PID 方法，即：

$$u(k) = K_{\mathrm{P}}e(k) + K_{\mathrm{I}}Ts\sum_{j=0}^{k} e(j) + K_{\mathrm{D}}\cdot\frac{\mathrm{e}(k)-e(k-1)}{Ts} \qquad (4-2)$$

我们编写了 pid_solve 函数，对系统使用 PID 进行控制，控制效果图如下所示，具体参数为 K_p=0.03, K_i=0.03, K_d=0。

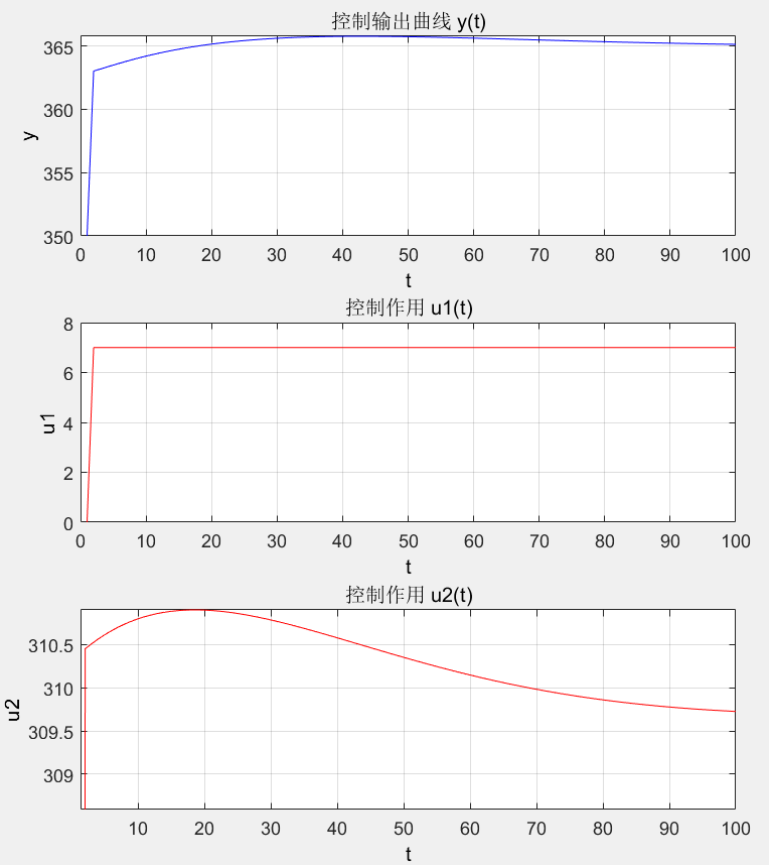

图 20 PID 控制器

当然,PID 算法受参数的影响很大也会,图 21-22 是不同 PID 参数下的控制效果图，图 22 有着较大的 P 参数，控制更快、更有效。

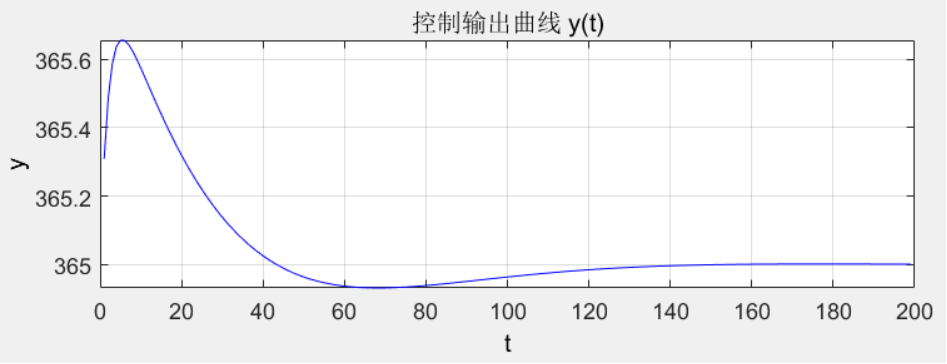

图 21 较小 Kp 下 PID 控制器效果图

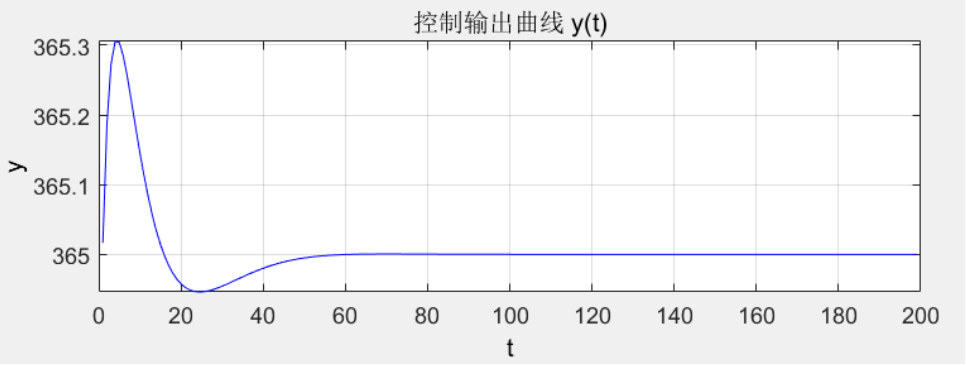

图 22 较大 Kp 下 PID 控制器效果图

4.2 Bang-Bang控制器

Bang—Bang 控制也称为开关控制或者最小时间控制，其主要的控制目的是在控制域内使系统以最快的速度从一个状态到另一个状态[4]。Bang-Bang 控制的基本原理是将状态空间划分为多个区域，一个区域对应一种控制变量的输出。这区域之间的分界面称为开关面，决定 Bang-Bang 控制的关键就是开关面的设定。

Bang-Bang 控制基本公式为:

$$u=\begin{cases}U_{max} & e>\varepsilon \\ U_{set} & -\varepsilon<e<\varepsilon \\ -U_{max} & e<-\varepsilon\end{cases} \qquad (4-3)$$

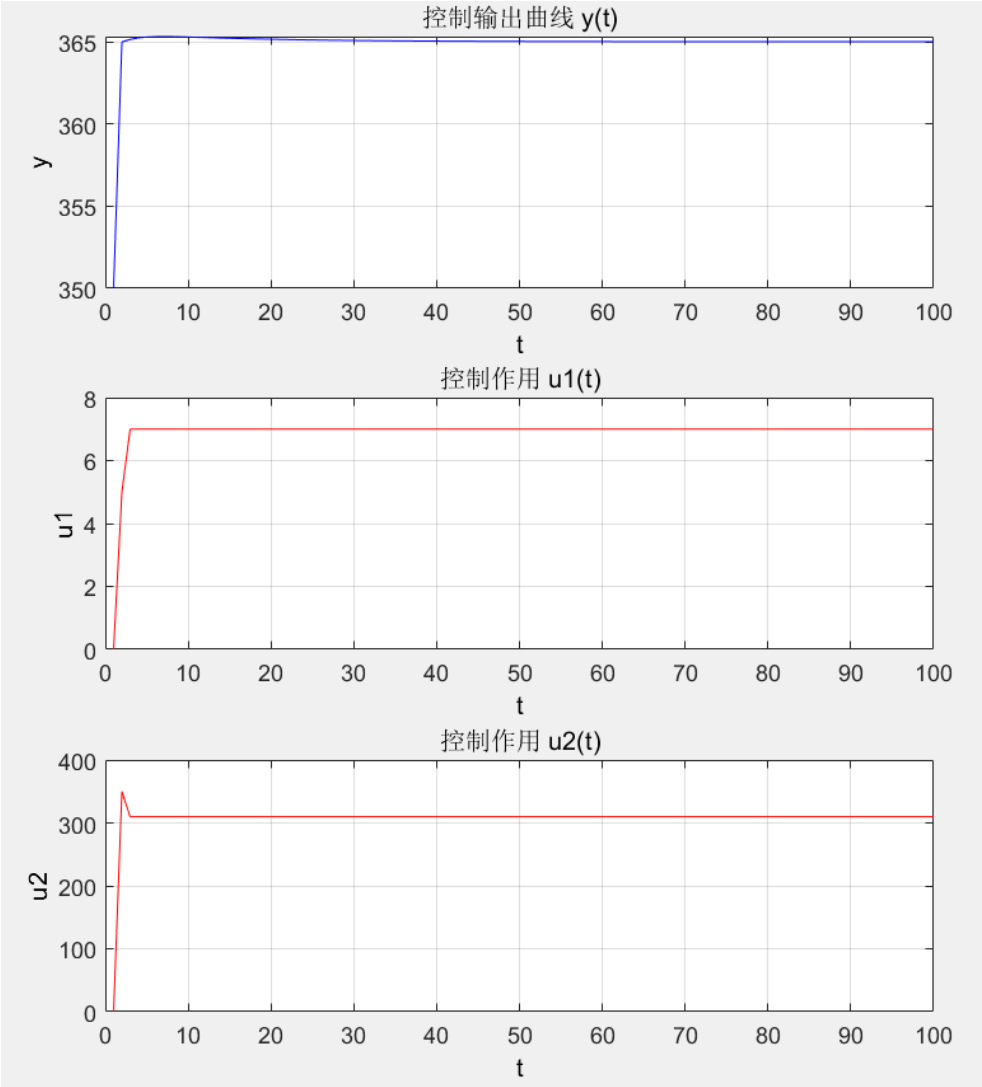

图 21 Bang-Bang 控制器

从上图可以看到，设计 Bang-Bang 控制器在采用合适的开关面与变量输出后，系统很快地达到设定值，动态性能较好。但是 Bang-Bang 控制在应用时对切换时刻和切换次数有严格要求，选取不当会导致整个控制过程紊乱。

4.3 PID-BangBang双模控制器

考虑 BangBang 控制的快速性与 PID 控制的稳定性能，因而我们又设计了 PID-BangBang 双模控制器，其基本思路如下图所示：

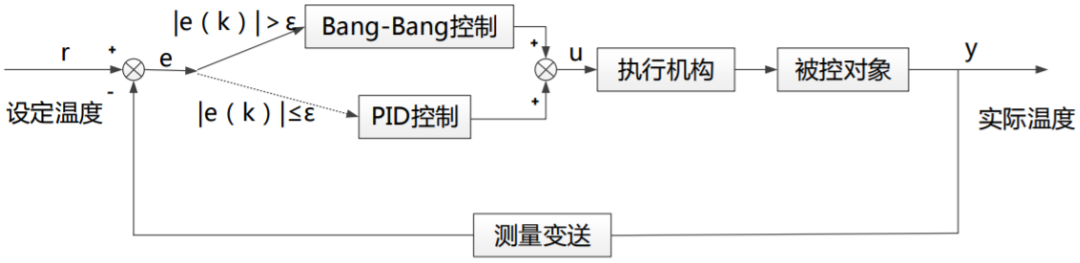

图 22 PID-BangBang 双模控制器

在实验平台进行测试，得到效果图如下所示：

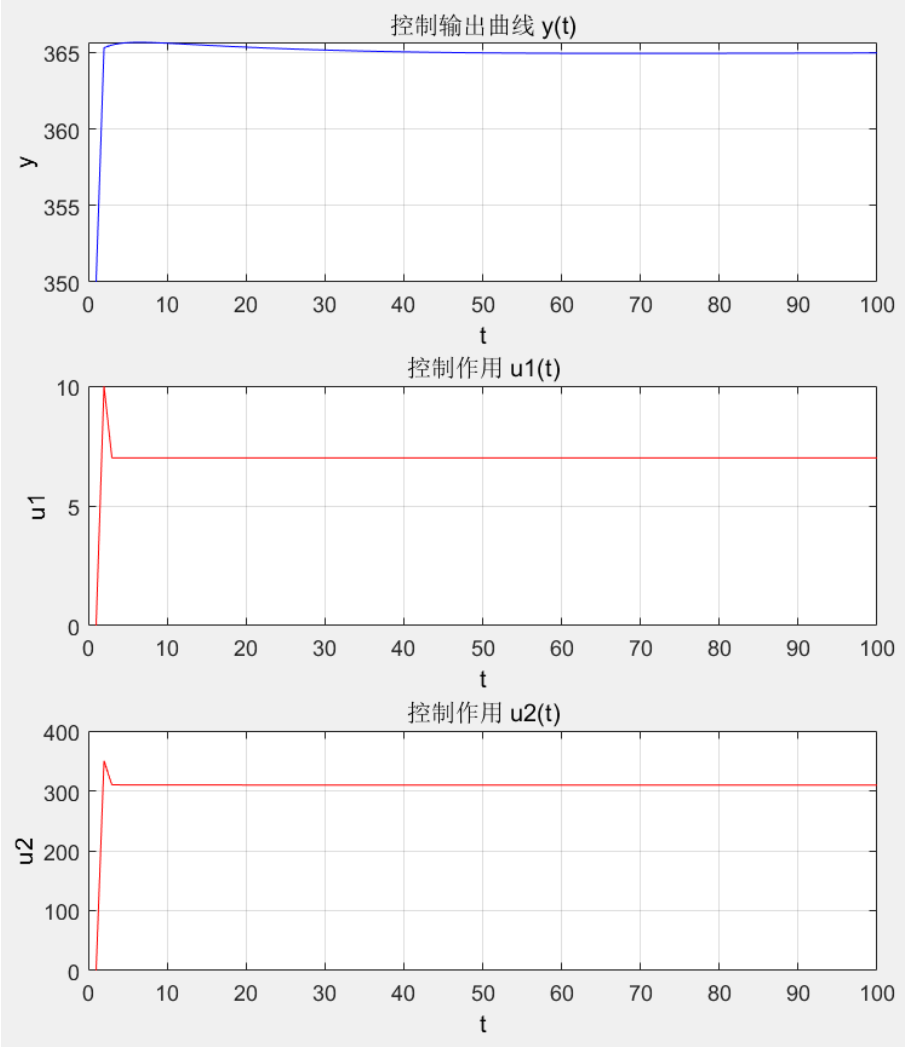

图 23 PID-BangBang 控制器

4.4 三种控制器对比

基于以上实验，三种控制器在 CSTR 系统的控制的应用上都具有一定的可行性，在合适的参数作用下，都能起到较好的控制效果。

Bang—Bang 控制具有结构简单及相应时间短等特点，在设定合适的开关线后可以迅速地达到设定值。PID 控制器则具有鲁棒性好、可靠性高，参数易于整定等优点。BangBang-PID 双模控制器则结合了两种方法的优点，有着更好的控制效果。

将三种控制器的控制效果图进行对比如图 24-26 所示。

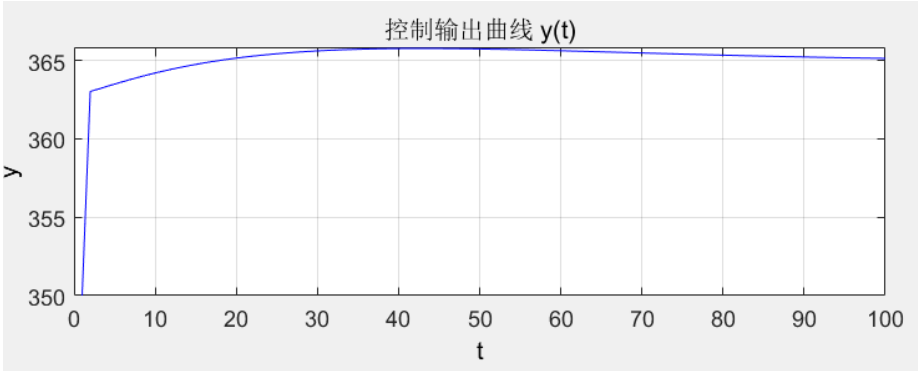

图 24　PID 控制器控制效果图

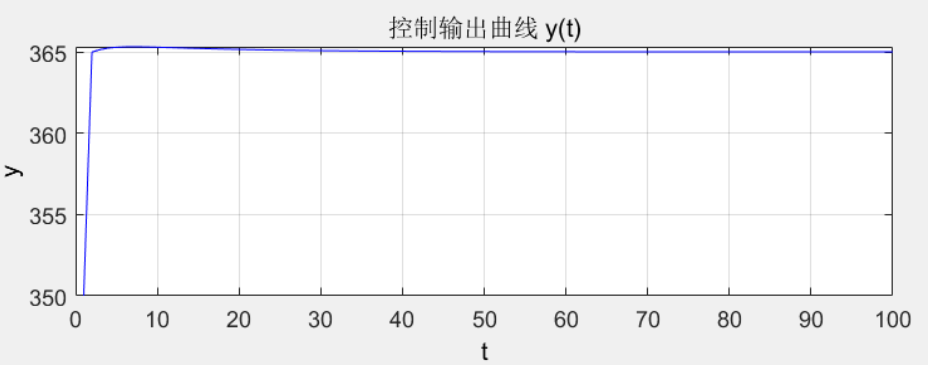

图 25　BangBang 控制器效果图

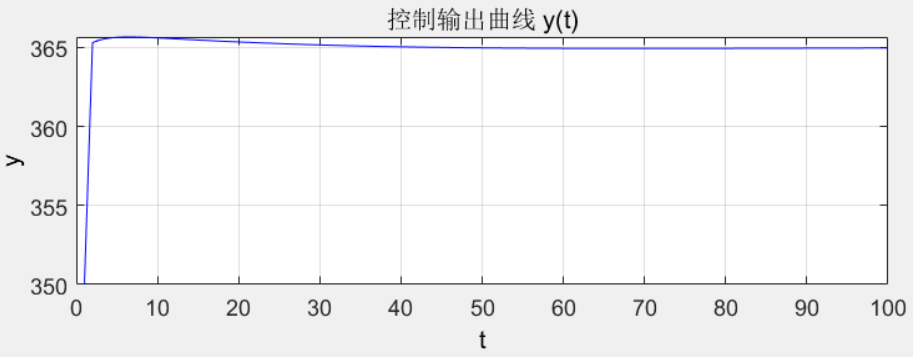

图 26　PID-BangBang 双模控制

从上面三种方法的对比图可以看出，含有 BangBang 控制的控制器在快速性上有着较好的优势。

下面我们对控制的抗扰性能进行测试，加入脉冲扰动，各控制器的效果图如图所示。可以看到，PID-BangBang 控制器作为两种控制器的融合，在抗扰性能上有了较大的提升。

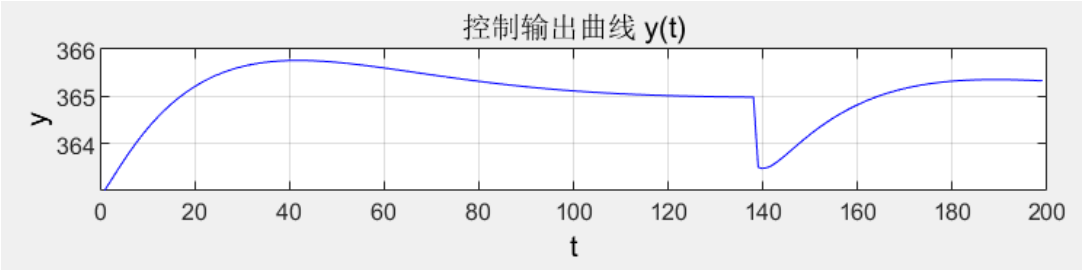

图 27 PID 控制器抗扰性能

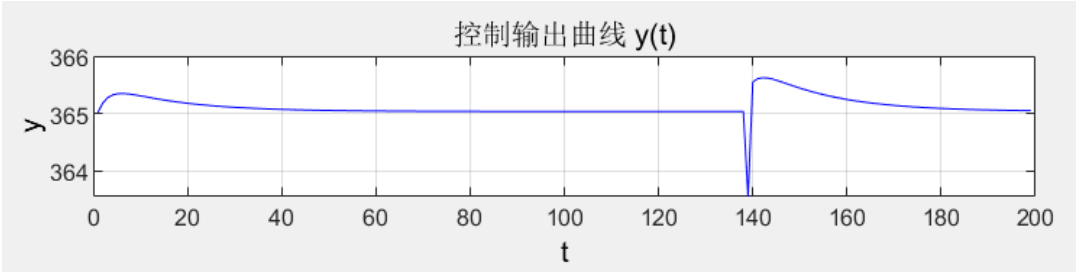

图 28 BangBang 控制器抗扰性能

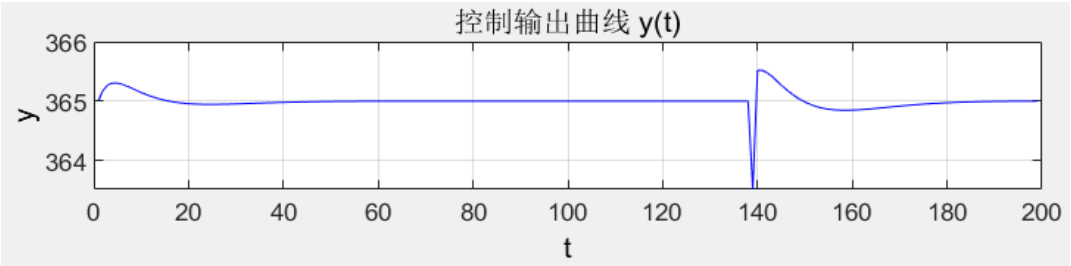

图 29 PID-BangBang 双模控制器抗扰性能

5. 实验总结与致谢

5.1 总结与展望

本次**《过程控制系统课程设计》**针对非线性连续搅拌反应釜进行了系统辨识与控制器设计。

在**模型辨识**方面，我采用采用局部线性化的基本思路，将系统简化为一阶加纯滞后模型，基于黑箱思想与阶跃测试，使用最小二乘法进行求解，对不同条件下的模型进行打表辨识，从而得到系统在不同输入参数下的模型参数。

在**控制器设计**方面，结合课内所学知识，设计了 PID、BangBang 以及 PID-Bangbang 双模控制器，这三种方法在 CSTR 系统的控制的应用上都具有一定的可行性，在合适的参数作用下，都能起到较好的控制效果。其中 BangBang 控制器有着较好的快速性能，PID-BangBang 控制器则在此基础上又具有了较好的抗扰性能，有着更好的实用价值。

当然，本次实验其实也有很多**不足之处**。模型辨识方面，想完全对非线性模型进行建模并不容易，我们所采用的也是局部线性及模型简化的方法进行实现。在某些输入情况下，会有误差，而多次误差累计则会造成模型的不精确。当然，这一问题也有可能是由于辨识数据不足，双线性插值并不完美造成的。

控制器设计方面，我们采用了并不高级的方法进行实验。本考虑结合《过程控制系统》课程所学知识，采用 MPC、神经网络等先进控制方法进行实验设计，但脱离实验环境，没有足够的时间很难完成这一任务，最终只能采用了经典的控制方法进行。

5.2 致谢

首先需要感谢**李少远老师和邹媛媛老师**为本课程提供了良好的实验平台与实验设备。先进控制系统的学习与设计很难在书本中得到实例化的理解，本次课程设计针对实际的工业系统进行，拓宽了我们的视野，使我们对工业系统的控制有了更深的理解与感悟。也要感谢《过程控制系统》这一课程的郑毅老师，感谢您在课后为我们解答了关于 CSTR 非线性系统的部分疑惑，谢谢！

其次，需要感谢负责这门课程的几位**助教老师**，感谢各位老师对实验平台的讲解与群内的答疑解惑。也要感谢你们在答疑课程上的解答，帮助我们明确了探究的方向，确定了基本的实验方法，避免了盲目尝试所带来的碰壁。

之后，需要感谢 gPROMS-4 小组的其它同学，在微信群以及两次线下的讨论中一起探讨实验的相关知识，获益良多！

最后，还要感谢**自动化系**同一实验的同学，一起讨论实验方法，交互验证，共同进步，获得了许多过程控制系统的相关知识。

【参考文献】

[1]邵汝倩. 非线性 CSIR 系统的控制方法研究[D]. 北京：北京化工大学，2018. DOI:10.7666/d.Y3390108.

[2]李长鹏. 基于 CSTR 温度系统的模糊神经网络预测控制研究[D]. 哈尔滨工程大学，2009.

[3]葛芦生，陶永华. 新型 PID 控制及其应用 [J]. 工业仪表与自动化装置，1998(03):55-59.

[4]李明骏，罗雄麟等. 变工况条件下 Bang-Bang 控制与常规 PID 控制的集成设计与分析[J] 化工自动化及仪表，2016，43(9):901-905. DOI:10.3969/j.issn.1000-3932.2016.09.003.

Appendices

Appendix A Code: fitmodel.m

```
%%
%  @ model_fit.m
%  @ Use least squares method to identify
%        CSTR data as an identification first-order model
%  @ C_zihao
%  @ chaizihao@sjtu.edu.cn
%  @ 2021.5
clear;clc;
%%
model = zeros(88, 6);
for u1 = -5:2:9
    for u2 = 300:5:350
        FileName = sprintf('./data-4/u1=%d-u2=%d.txt', u1, u2)
        data = importdata(FileName);
        [a1, b1, L] = yijie_recog(data);
        % save the param in model
        model(k, 1) = u1;
        model(k, 2) = u2;
        model(k, 3) = a1;
        model(k, 4) = b1;
        model(k, 5) = L;
        model(k, 6) = data(1, 4);
        k = k + 1;
    end
end
%%
function [a1, b1, L] = yijie_recog(data)
    % Gets the step response y1 from the transfer function
    y = data(:, 4) - data(1, 4);
    Fai = zeros(length(y), 3);
    for i = 1:length(y)
        Fai(i, :) = [-1 * sum(y(1:i)), -1, i];
    end

    %Using Least square method
    theta = inv(Fai' * Fai) * Fai' * y;
    a1 = theta(1);
    b1 = theta(3);
    L = theta(2) / theta(3);
    disp([a1, b1, L])
    if (L < 0)
        L = 0
    end
    s = tf('s');

    try
        % The final transfer func
        M1 = (b1) * exp(-L * s) / (s + a1);
        % Draw the image to contrast
        figure;
        plot(y, 'r')
        hold on;
        step(M1);
        grid on;
        title('Model identification effect diagram');
        xlabel('Time');
        ylabel('Temporature');
        legend('Real', 'Recognise')
    catch err
        disp(["wrong"])
    end
end
```

Appendix B Code: recognise.m

```
%%
%  @ recognise.m
%  @ Get model using Bilinear interpolation
%  @ C_zihao
%  @ chaizihao@sjtu.edu.cn
%  @ 2021.5

%% Read models from "model.xlsx"
%  T1ąćQ  input environment , t sampling time
%  model_data: the model read from xlsx ,y0: now temporature
function [yout] = ml_response(T1, Q, t, model_data, y0)

    [a1, b1, L, t_i] = model_get(T1, Q, model_data);
    sys = tf(b1, [1, a1], 'inputdelay', L);

    step_out = step(sys, 1:1:t + 2000) + t_i;
    [point, ~] = nearest_func(step_out(1:2000), y0);
    yout = step_out(point + 1) - y0
end

function yout = reco(T1, Q, model_data, y0)
    yout = ml_response(T1, Q, 1, model_data, y0);
end

%% Get model using Bilinear interpolation
function [a0, b0, L0, t_i] = model_get(T1, Q, model_data)

    [~, near_Q] = nearest_func(model_data(:, 1), Q);

    [low_Q, large_Q] = low_large(Q, near_Q, -5, 9, 2);

    [a1, b1, L1, t1] = model_get_T(T1, low_Q, model_data);
    [a2, b2, L2, t2] = model_get_T(T1, large_Q, model_data);

    %  interpolation
    a0 = Interpolation(low_Q, Q, a1, a2, 2);
    b0 = Interpolation(low_Q, Q, b1, b2, 2);
    L0 = Interpolation(low_Q, Q, L1, L2, 2);
    t_i = Interpolation(low_Q, Q, t1, t2, 2);

end

%% get T param from model_data
function [a0, b0, L0, t_i] = model_get_T(T1, Q, model_data)

    [~, near_T] = nearest_func(model_data(:, 2), T1);

    [low_T, large_T] = low_large(T1, near_T, 300, 350, 5);

    [a1, b1, L1, t1] = param_search(Q, low_T, model_data);
    [a2, b2, L2, t2] = param_search(Q, large_T, model_data);

    a0 = Interpolation(low_T, T1, a1, a2, 5);
    b0 = Interpolation(low_T, T1, b1, b2, 5);
    L0 = Interpolation(low_T, T1, L1, L2, 5);
    t_i = Interpolation(low_T, T1, t1, t2, 5);
end

%% get Q param from model_data
function [a0, b0, L0, t_i] = model_get_Q(T1, Q, model_data)
    [~, near_Q] = nearest_func(model_data(:, 1), T1);
    if (Q == near_Q || near_Q == 300 || near_Q == 350)
        low_Q = near_Q;
        large_Q = near_Q;
    elseif (Q > near_Q)
        low_Q = near_Q -5;
        large_Q = near_Q +5;
```

```
68      else
69          low_Q = near_Q - 5;
70          large_Q = near_Q;
71      end
72      % search the low_Q and large_Q model
73      [a1, b1, L1, t1] = param_search(low_Q, T1, model_data);
74      [a2, b2, L2, t2] = param_search(large_Q, T1, model_data);
75      % Interpolate in the Q direction
76      a0 = Interpolation(low_Q, Q, a1, a2, 5);
77      b0 = Interpolation(low_Q, Q, b1, b2, 5);
78      L0 = Interpolation(low_Q, Q, L1, L2, 5);
79      t_i = Interpolation(low_Q, Q, t1, t2, 5);
80  end
81
82  %% MINMAX
83  function [output] = constrain(min, input, max)
84      if (input > max)
85          output = max;
86      elseif (input < min)
87          output = min;
88      else
89          output = input;
90      end
91  end
92
93  %%
94  function [low, large] = low_large(input, near, min, max, gap)
95      if (input == near || near == min || near == max)
96          low = near;
97          large = near;
98      elseif (input > near)
99          low = near -gap;
100         large = near +gap;
101     else
102         low = near - gap;
103         large = near;
104     end
105 end
106
107 %% get the nearest from array
108 function [p, y] = nearest_func(A, b)
109     [~, index] = sort(abs(A(:) - b));
110     y = A(index(1));
111     p = index(1);
112 end
113
114 %% One-way interpolation
115 function [k] = Interpolation(min, actual, key_min, key_max, gap)
116     k = key_min + (key_max - key_min) * abs(actual - min) / gap;
117 end
118
119 %% Search param from data
120 function [a, b, L, t] = param_search(Q, T, model_data)
121     [col, ~] = size(model_data);
122     for i = 1:1:col
123         if (model_data(i, 1) == Q && model_data(i, 2) == T)
124             a = model_data(i, 3);
125             b = model_data(i, 4);
126             L = model_data(i, 5);
127             t = model_data(i, 6);
128             break;
129         end
130     end
131 end
```

Appendix C Code: demo.m

```
clc, clear, close all

%% named data storage file
nameName = datestr(now, 29);
FileName = strcat('./data/', 'CSTR', nameName);
FileNameY = strcat(FileName, 'Y.txt');

%% Start gOMATLAB--standard format
gOMATLAB('startONLY');
gOMATLAB('select', 'Simulate_CSTR', 'Simulate_CSTR');
gOMATLAB('simulate', 'Simulate_CSTR');

%% Get the analog value in gPROMs cyclically
% gOMATLAB('evaluate',u,5) is a standard sentence, please do not modify
% u(1) = heat transfer temperature Q, u(2) = feed temperature T_in
% The initial value of% u is [0, 315.5]
% y is the measurement signal,  y(3) = reaction temperature
% y(4:7) = molar concentration of four components in CSTR
% u(1) value range [-10,10], u(2) value range [300,350];
% y(3) The expected value of the reaction temperature is 365

simu_time = 200;
y = zeros(simu_time, 9);
ut = zeros(simu_time, 2);
aim_temp = 365;
error = 0;
i_error = 0;
d_error = 0;
T1 = 365;
y(1, 4) = 350;
%%
for i = 2:simu_time

    if y(i - 1, 4) < 364
        Q = 5;
        T1 = 350;
    elseif y(i - 1, 4) > 366
        Q = -10;
        T1 = 300;
    else
        d_error = aim_temp - y(i - 1, 4) - error;
        error = aim_temp - y(i - 1, 4);
        T1 = 310 + pid_solve(2, 0.2, 0, 10000000, error, i_error, d_error);
        T1 = constrain(300, T1, 350);
        i_error = i_error + error;
        Q = 7;
    end

    %     if i == 140
    %         T1 = 300;
    %         Q = -10;
    %     end

    ut(i, :) = [Q, T1];
    %According to the controller you designed, calculate the value of u
    u = [Q T1];

    y(i, :) = [i, u, gOMATLAB('evaluate', u, 5)];

    % % Write the value of y to the file FileNameY
    dlmwrite(FileNameY, y)

    % If the last return value of gOMATLBA is 1, it means that the simulation is norm
    if y(i, end) == 1
        disp([num2str(i) '  worked!'])
    else
        disp('failed!')
```

```
68      end
69
70  end
71
72  gOMATLAB('stop'); % Stop calling gOMATLAB, standard statement
73
74  %% Draw the image
75  begin = 1;
76  subplot(311);
77  plot(y(begin:simu_time, 4));
78  title(' y(t)');
79  xlabel('t');
80  ylabel('y');
81  grid on
82
83  subplot(312);
84  plot(ut(begin:simu_time, 1), 'r');
85  title(' u1(t)');
86  xlabel('t');
87  ylabel('u1');
88  grid on
89
90  subplot(313);
91  plot(ut(begin:simu_time, 2), 'r');
92  title(' u2(t)');
93  xlabel('t');
94  ylabel('u2');
95  grid on
96
97  %% solve the problem using pid
98  function [T1] = pid_solve(kp, ki, kd, i_max, error, i_error, d_error)
99      p_out = kp * error
100
101     i_out = ki * i_error
102     i_out = constrain(-i_max, i_out, i_max);
103
104     d_out = kd * d_error
105
106     T1 = p_out + i_out + d_out;
107 end
108
109 %% MINMAX
110 function [output] = constrain(min, input, max)
111
112     if (input > max)
113         output = max;
114     elseif (input < min)
115         output = min;
116     else
117         output = input;
118     end
119
120 end
```