课程总结报告

——互耦水槽液位控制的PID整定方法比较

1 任务重述

1.1 互耦水槽模型简介

互耦水槽系统是由两个相同水槽组成，如图 1 所示。两个相同的水槽 basin 1 和 basin 2 通过阀门 K3 连接，从而两水槽的液位 y_1、y_2 互相耦合，液位的测量由传感器完成；流入两个水槽的流量 Q 由水泵 pump 控制，水泵的 DC 电动机电枢电压受 u_1、u_2 控制，该水槽可以通过调节各个阀门的开闭实现多样的实验。

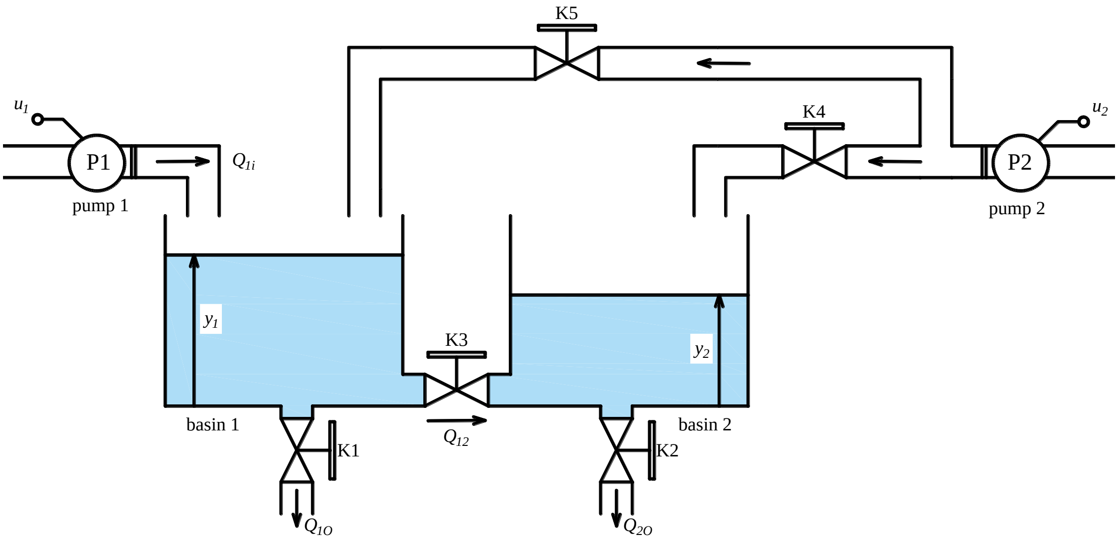

图 1　　互耦水槽系统原理图

另一方面，互耦水槽是一个大惯性、大时滞、非线性、时变的系统，互耦水槽的流量关系可以描述为：$Q_{12} = c_{12}\sqrt{H_1 - H_2}$ ，在建立模型时通常需要将其在工作点附近线性化处理。该模型可以用一个带纯滞后的一阶惯性环节（FOPDT）来近似描述，其传递函数的形式为：

$$G(s) = \frac{Ke^{-T_d s}}{Ts + 1}$$

1.2 任务内容

- 基于给定的实验数据进行传递函数建模，并根据验证指标验证模型的准确程度
- 对传递函数模型进行适当的频域分析，求出需要的指标、画出相应图线
- 利用几种不同的 PID 整定方法进行参数整定，比较几种方法的动态性能
- 提供建模和模型分析的 Matlab 程序清单

2 系统建模

2.1 实验数据分析

本报告使用的实验数据是 plant_data1.mat，此数据集由五部分变量组成，分别是：实验时间向量 t，水槽 basin 1 的输入输出数据 u_1、y_1，水槽 basin 2 的输入输出数据 u_2、y_2。系统的时域响应经过 plot 函数绘制后，我们发现输出相应的曲线的确能使用带纯滞后的一阶惯性环节替代。

通过观察向量 u_1、u_2 的值我们发现在索引 283 处发生输入的跳变，均为 5 跳变到 6，这是一个阶跃输入。根据索引找到对应的跳变时间 $t_0 = -437.6\text{s}$，我们做变换 $t = t - t(283)$，即可将跳变时间变为 0。

通过分析，实际上这是两次互相对称的实验数据，以一端为例，实验过程为：先将水泵 pump 1 的电枢电压设置为 $u_1 = U_0$，打开阀门 K3 使得水槽 basin 2 的液位存在一个初值 Y_0，待液位稳定一段时间，在 t_0 时刻将 u_2 设置成 U_f，即施加的是阶跃信号，水槽 basin 2 的液位将逐渐开始变化，直到稳定在一个终值 Y_f。

上述的量，通过数据集可以很容易地读出，建模时选取的是 y_2作为输出 y，且有：

$$U_0 = 5，U_f = 6，Y_0 = y(t_0) = 1.546，Y_f = 3.675$$

由于实验数据的误差特性，其中 Y_f 的值是一个在终值附近较为折中的值。

2.2 模型参数求解

［注］　该部分建立模型的 Matlab 程序清单为 build_mode.m

如图 2 所示，该模型参数的求解采用的是两点法。我们需要作两条高度相较于 Y_0 分别为 $0.283\Delta Y$ 和 $0.362\Delta Y$ 的平行直线（$\Delta Y = Y_f - Y_0$），与输出响应曲线交点对应的时间记作 t_1、t_2。

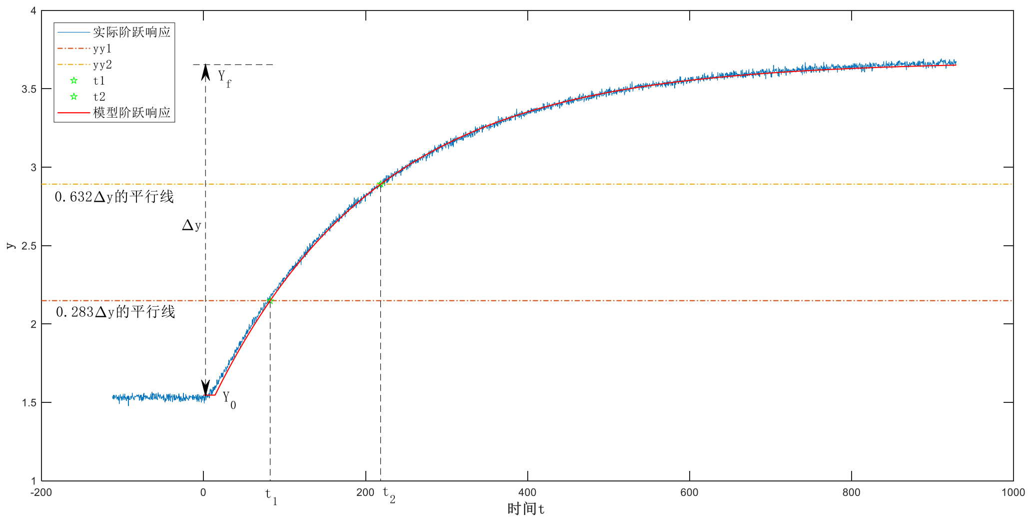

图 2　　模型建立

使用 for 循环求得：

$$t_1 = -355.2，t_2 = -218.8$$

根据以下公式：

$$K = \frac{Y_f - Y_0}{U_f - U_0}、T = 1.5(t_2 - t_1)、T_d = t_2 - T$$

如果上式求出 $T_d < 0$，则 $T = t_2$，$T_d = 0$。

最终：

$$K = 2.129、T = 204.6、T_d = 14.2$$

互耦水槽系统的传递函数为：

$$G(s) = \frac{2.129e^{-14.2s}}{204.6s + 1}$$

2.3 验证模型

［注］　**该部分验证模型的 Matlab 程序清单为 verify_model.m**

模型建立完成之后，从图 2 可看出模型的拟合效果还是比较好的，但是需要一个指标定量描述模型的优劣程度，记为 I_p，其表达式为：

$$I_p = \frac{1}{t_{fin} - t_{ini}} \int_{t_{ini}}^{t_{fin}} |y(t) - y_m(t)|^2 dt \sim \frac{1}{t_{fin} - t_{ini}} \left(\sum |y(t) - y_m(t)|^2 \Delta t\right)$$

其中，t_{ini} 为起始时间，t_{fin} 为终止时间，$y(t)$ 为实际阶跃响应，$y_m(t)$ 为模型的阶跃响应。$|\mathrm{y(t)} - \mathrm{y_m(t)}|^2$ 表征了实际响应与模型响应的偏差，再对时间求平均值得到的 I_p，若 I_p 越小，说明模型的近似程度越好。

为便于计算机求解，将积分转换成了求和的形式，借助计算机工具，求得：

$$I_p = 3.0163 \times 10^{-4}$$

实际阶跃响应和模型阶跃响应如图 3 所示

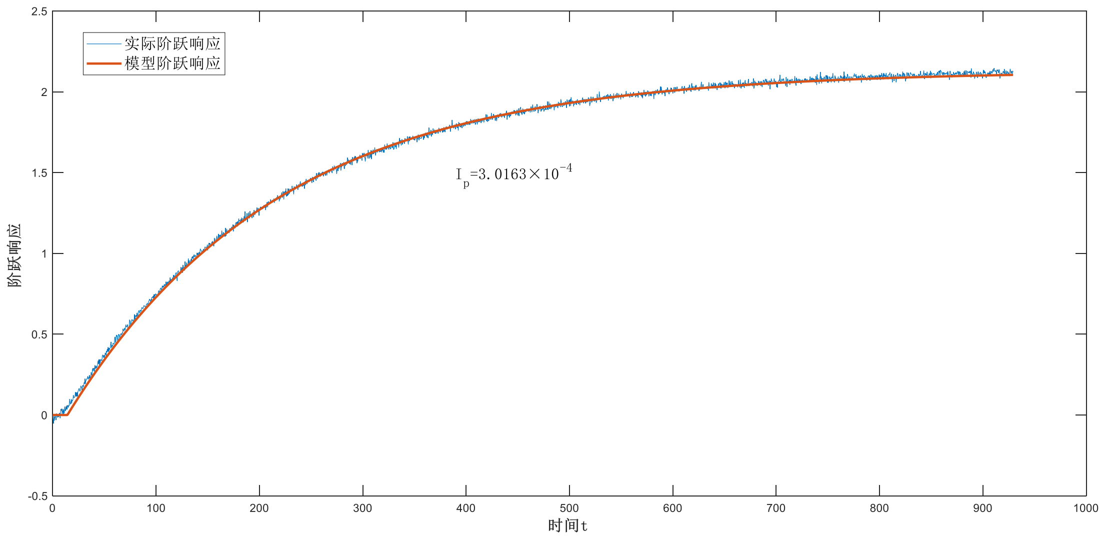

图 3　　模型响应和真实响应的示意

2.4 模型的频域分析

[注]　　该部分模型的频域分析 Matlab 程序清单为 f_analysis.m

利用 Matlab 下的 margin 函数可以较为方便求出：

幅值裕度：$G_m = 10.9309$

相位裕度：$P_m = 110.5418$

幅值穿越频率：$W_{pm} = 0.092$

相角穿越频率：$W_{gm} = 0.1136$

绘制的 Bode 图如图 4 所示：

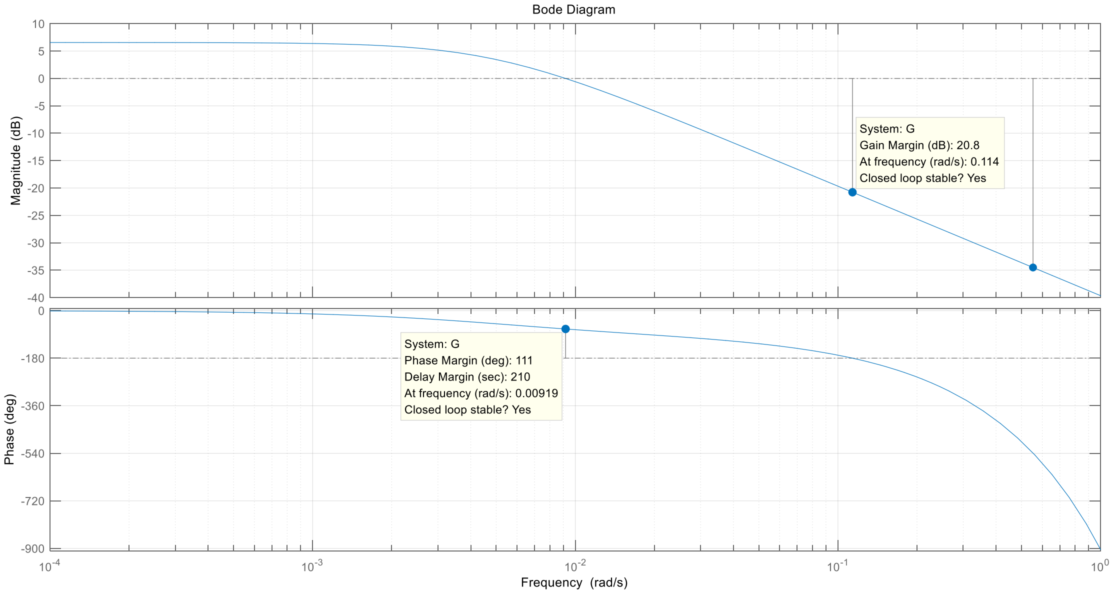

图 4　模型的 Bode 图

3　PID 控制器的建立

［注］　该部分建立控制器模型的 Matlab 程序清单为 build_pid.m

3.1 Ziegler-Nichols 整定法

- 方法简述：

对于过程对象常可采用一阶延迟模型（FOPDT）近似描述，而 Ziegler-Nichols 方法是一种较为经典的针对此模型的 PID 控制器设计方法。假设 PID 的控制器的标准形式为：

$$G_c(s) = K_p(1 + \frac{1}{T_{int}s} + T_{dif}s)$$

根据 Ziegler-Nichols 方法，上述控制器各个参数的按照表 1 确定，表中 T 表示受控对象模型的时间常数，K 表示增益，T_d 表示延迟时间。

控制器类型	K_p	T_{int}	T_{dif}
P	$\frac{T}{KT_d}$		
PI	$0.9\frac{T}{KT_d}$	$\frac{T_d}{0.3}$	
PID	$1.2\frac{T}{KT_d}$	$2.2T_d$	$0.5T_d$

表 1　Ziegler-Nichols 方法控制器参数

● 建立控制器模型

模型的建立首先是需要根据表 1 计算出控制器的各个参数：

PI 控制器：$K_p = 6.0909$、$T_{int} = 47.3333$

PID 控制器：$K_p = 8.1212$、$T_{int} = 31.2400$、$T_{dif} = 7.1000$

然后便得到两种控制器的传递函数模型：

PI 控制器：

$$G_{znpi}(s) = 6.0909(1 + \frac{1}{47.3333s})$$

PID 控制器：

$$G_{znpid}(s) = 8.1212(1 + \frac{1}{31.24s} + 7.1s)$$

3.2 SIMC 整定法

● 方法简述

SIMC 方法是一种由内模控制（IMC）演化来的 PID 参数整定法，一般采用串行 PID 控制器实现，控制回路的图示如所示

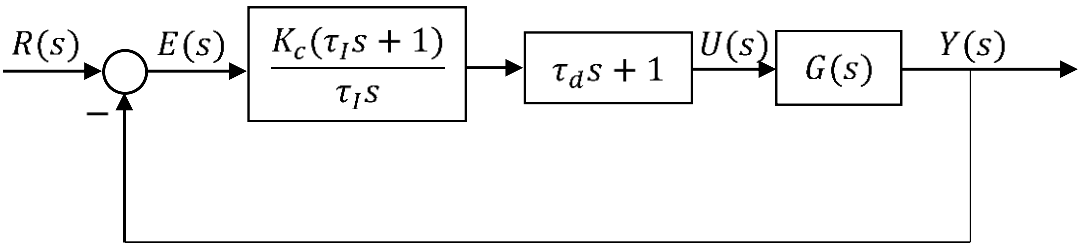

图 5　SIMC 整定法 PID 控制回路

如果采用 PI 控制器，则只需将图 5 的 $(\tau_d s + 1)$ 项舍去，或直接令 $\tau_d = 0$。各个参数的确定方式如下：

PI 控制器：

$$K_c = \frac{T}{K(\tau_c + T_d)}、\tau_c = T_d、\tau_I = min\{T, 4(\tau_c + T_d)\}$$

其中 τ_c 为闭环系统时间常数。

PID 控制器：

$$K_c = \frac{T}{K(\tau_c + T_d)}、\tau_c = T_d、\tau_I = min\{T, 4(\tau_c + T_d)\}、\tau_d = \frac{T_d}{3}$$

● 建立控制器模型

模型的建立首先根据前一目给出的公式计算各个参数的值：

PI 控制器：$\mathrm{K_c} = 3.3839$、$\tau_I = 113.6000$

PID 控制器：$\mathrm{K_c} = 3.3839$、$\tau_I = 113.6000$、$\tau_d = 4.7333$

之后根据控制器的结构形式便可建立控制器模型：

PI 控制器：

$$G_{sipi}(s)=\frac{3.3839(113.6s+1)}{113.6s}$$

PID 控制器：

$$G_{sipid}(s)=\frac{3.3839(113.6s+1)(4.7333s+1)}{113.6s}$$

4 四种 PID 整定方法的比较

［注］ 该部分整定方法比较的 Matlab 程序清单为 compare_pid.m

此部分四种整定方法前面已经给出，分别简记为 Z-N_PI、Z-N_PID、SIMC_PI、SIMC_PID，我们需要比较的性能指标将逐一在各个小节中描述。

4.1 跟踪性能分析

本部分涉及到的指标为：上升时间、超调量、调整时间、平方偏差积分、绝对偏差积分，下面分别讨论这些指标。

- 上升时间、超调量、调整时间

由于可以利用 Matlab 自带的阶跃响应函数 step 绘制出阶跃响应曲线后直接读图得出，故放在同一个小节求解。首先需要建立一个闭环模型，然后画出相应的阶跃响应曲线，最终使用鼠标点按的方法在图形窗口上得到需要的性能指标。四种整定方法对应的闭环阶跃响应曲线及其性能指标如图 6 所示。其中各个性能指标的数值已统计在表 2 中。

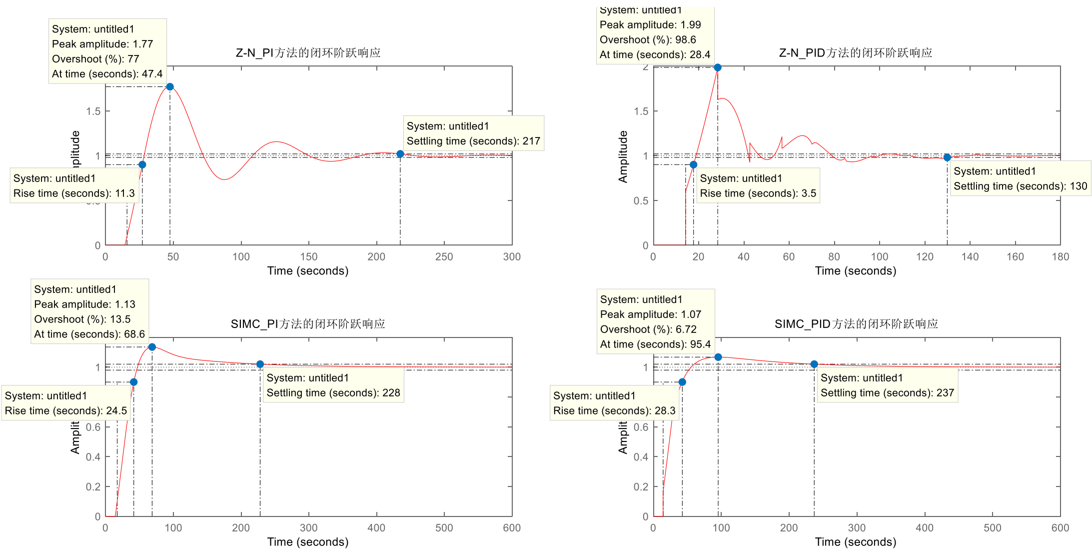

图 6　四种 PID 整定方法建立的闭环模型阶跃响应比较

- 平方偏差积分、绝对偏差积分

首先需要明确的是偏差的定义是：$E(s)=R(s)-Y(s)$，从而可得偏差传递

函数为：

$$\Phi_e(s) = \frac{1}{1+G_cG}$$

式中，G_c 是 PID 控制器的传递函数，G 是受控对象的传递函数。

其次需要明确的是两种指标的定义，平方偏差积分（ISE）的定义为：

$$ISE = \int_0^{\infty} e^2(t)dt$$

绝对偏差积分的定义为：

$$IAE = \int_0^{\infty} |e(t)|\, dt$$

求解时需要对其做近似处理，使用和式子代替积分，较大的上限代替无穷限，即有以下表达式：

$$ISE = \sum e^2(t)\Delta t$$

$$ISE = \sum |e(t)|\Delta t$$

借助 Matlab 得到的误差曲线如图 7 所示，计算得到的 ISE、IAE 的数值已统计在表 2 中。

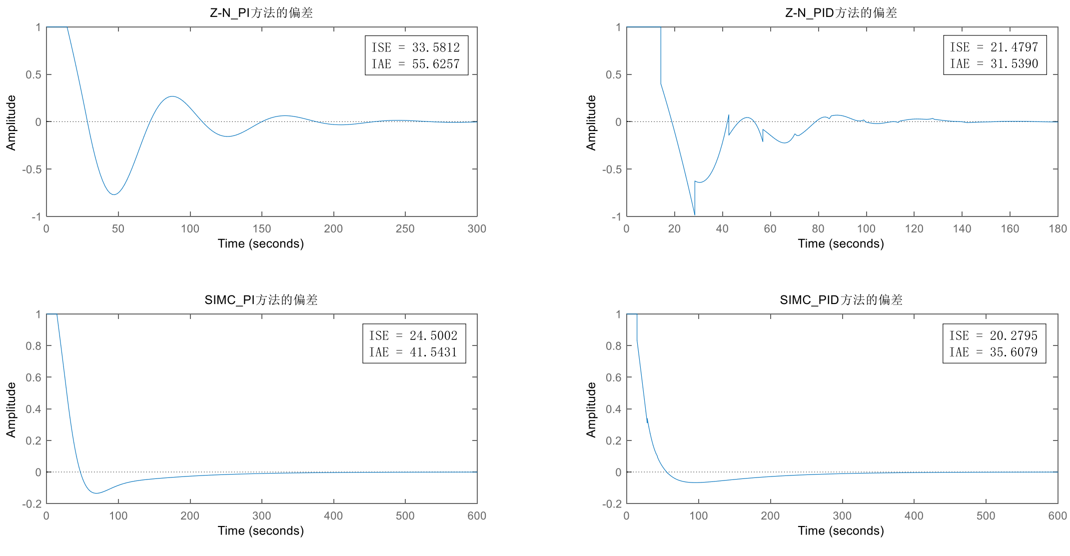

图 7　四种 PID 整定方法的偏差曲线

4.2 抗干扰性能分析

- 指标解释

这一部分有两个指标描述分别是动态降落和恢复时间，二者都表征系统的抗干扰性能。

动态降落 $\Delta C_{max}\%$ 的定义是：系统稳定运行时，突然加上一定数值的扰动，引起原稳态值 $C_{\infty 1}$ 向新的稳态值 $C_{\infty 2}$ 转变，在转变过程中，引起输出值的最大降

落的百分数称为动态降落。

恢复时间 t_v 的定义是：在上述的转变过程中，从阶跃扰动作用开始，到输出量基本恢复稳态值 $C_{\infty 2}$（基本恢复的含义是恢复到 $C_{\infty 2}$ 附近的 2% 或 5%）所需要的最短时间。

- 求解指标

求解该指标利用传统的 Matlab 语言的方法很难求解，问题在于定义的 PID 控制器是理想控制器，是物理不可实现的，借助 Simulink 工具建立模型更为直观简便。

在 Simulink 下，控制器的并行结构传递函数表达式为：

$$G_c(s) = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

其中，第三项是微分环节的近似，做了近似处理，以便物理实现，其中 N 的值默认取 100。我编写了一个名为 get_PID 的函数，用于由前面讨论的控制器模型得到 P、I、D 三个 PID 参数的数值，详见 get_PID.m

Simulink 模型的建立的结果如图 8 所示，其中输入信号是阶跃信号，由前面的讨论可知，在 500 秒后系统输出几乎已经稳定，所以给出的扰动信号是跳变时刻为 500 的阶跃函数，且是由 0 跳变到 -1 的。

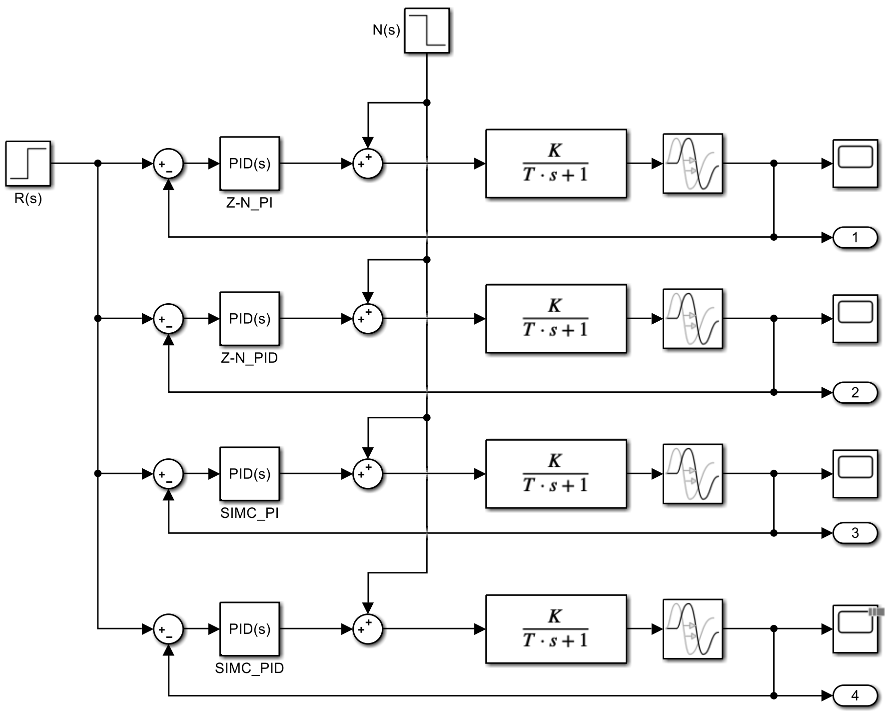

图 8　　用于观察抗干扰性能的 Simulink 模型

在图 8 所示的模型中，我们将输入置零，经过示波器示波，我们发现当扰动

信号单独作用时，总会恢复到 0，所以我们可以说原稳态值和新稳态值均为 1。我们画出响应曲线如图 9 所示。此处我们利用数值方法求得 ΔC_{max}%、t_v（取 5%），求得的结果已统计在表 2 中。

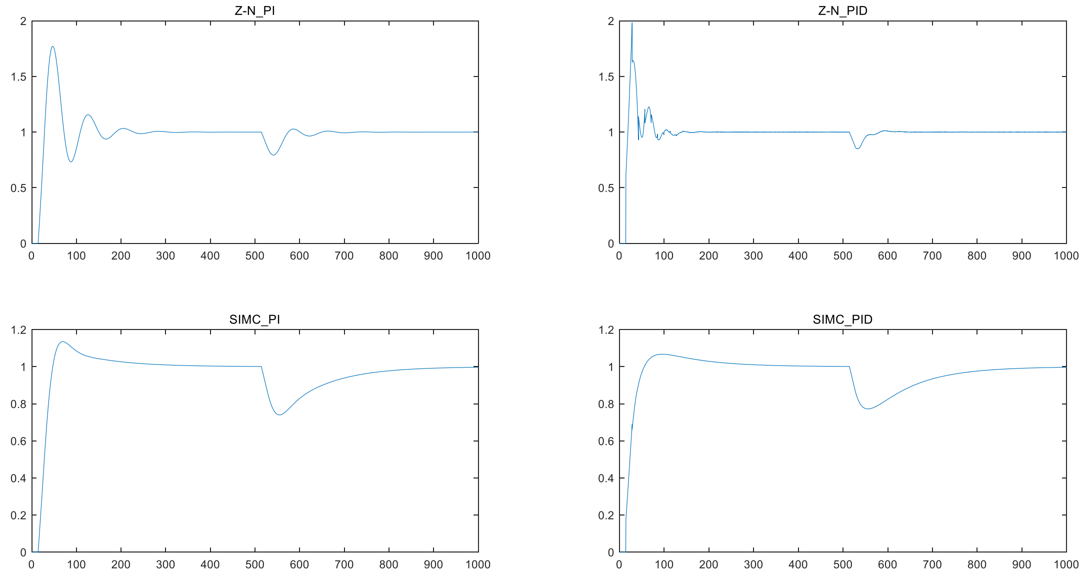

图 9　　用于观察抗干扰性能的响应曲线

4.3 控制器输出信号平滑性

在 4.3 节中我们讨论了 PID 控制器的物理可实现形式，控制器本身是物理不可实现的，基于 Simulink 我们可以较为便捷地得到控制器输出，本节的 Simulink 模型只需在前一节的基础上稍加修改，建立的模型如图 10 所示。其中我们将 PID 控制器的滤波系数 N 设置成了 10，并且使用了集线器，仿真的步长设置成了 0.01 秒，仿真时间是 300 秒。

基于建立好的 Simulink 模型，我们可以画出控制器的输出曲线如图 11 所示，由图可知，Z-N_PID 存在过冲现象，实际使用时需要前置一个饱和非线性环节。下面需要根据控制器输出信号的平滑性定义：

$$\sum |u(k+1)-u(k)|$$

借助 Matlab 可求出，其值已统计在表 2 中。

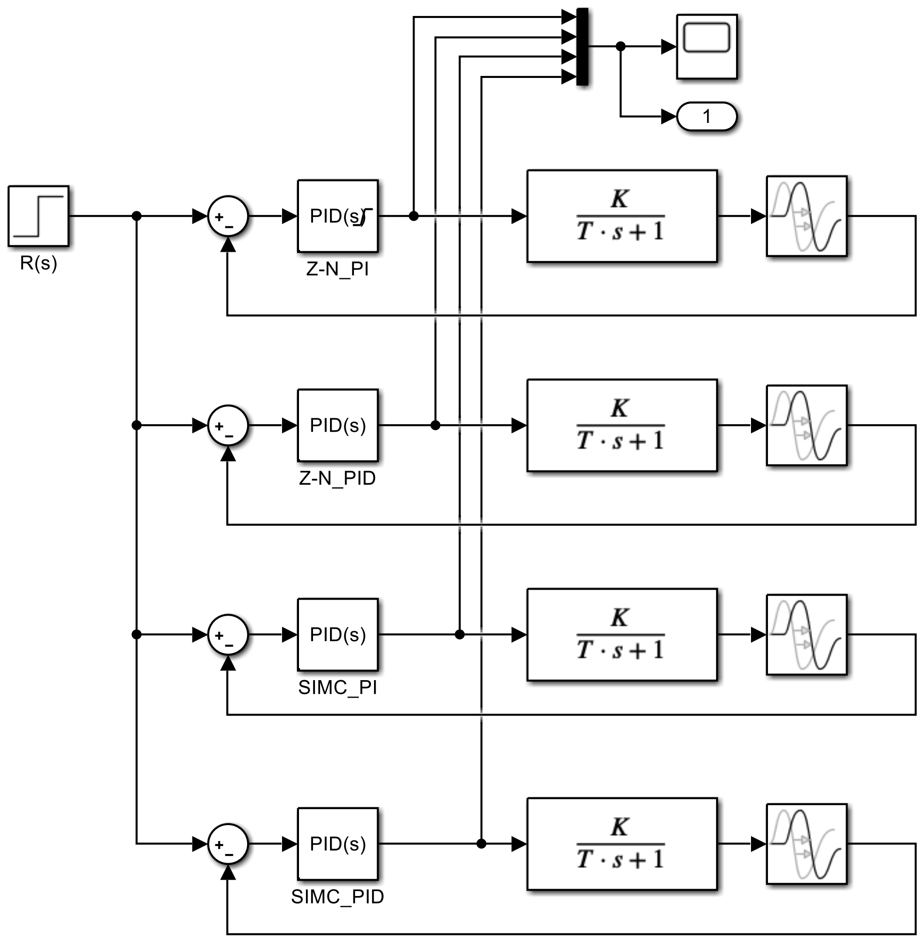

图 10 控制器输出的 Simulink 模型

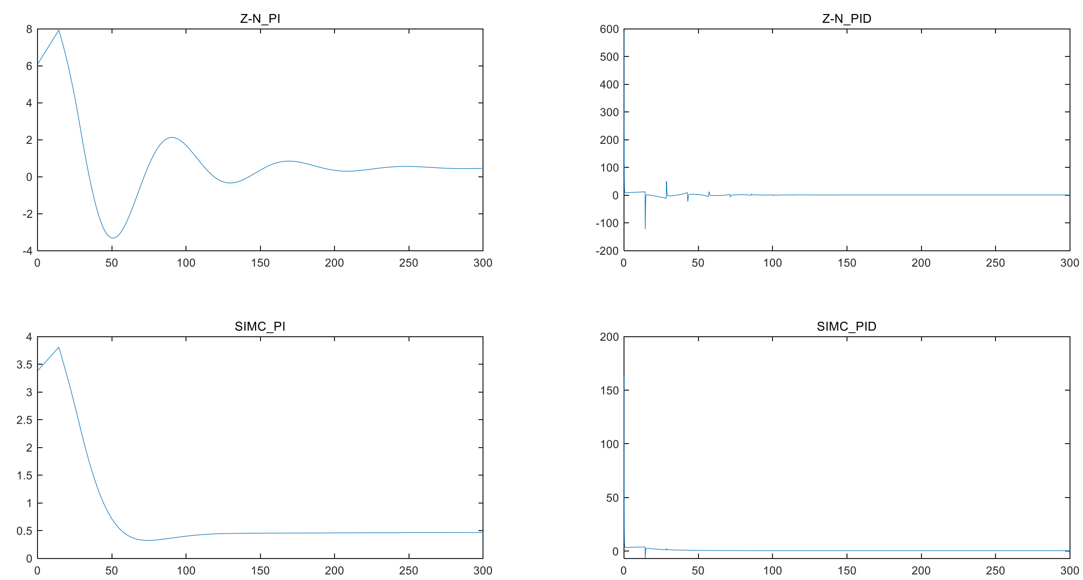

图 11 控制器输出曲线

4.4 结论

四种 PID 整定方法的性能指标见表 2：

	Z-N_PI	Z-N_PID	SIMC_PI	SIMC_PID
上升时间	11.3	3.5	24.5	28.3
超调量	77%	98.6%	13.5%	6.72%
调整时间	217	130	228	237
控制器输出信号平滑性	23.0966	1122.5	4.0491	185.6582
ISE	33.5812	21.4797	24.5002	20.2795
IAE	55.6257	31.5390	41.5431	35.6079
$\Delta C_{max}\%$	20.67%	14.99%	25.99%	22.79%
t_v	566.3818	549.2542	719.5936	726.7005

表 2 四种 PID 整定方法的性能指标

从表中可以看出，Z-N_PID 方法的上升时间最快，调整时间最快，恢复时间也最快，说明其对输入的响应是最敏感的，但是带来了超调量大，控制器输出信号平滑性太差等不利，实际使用中要尽可能抑制控制器输出过冲的现象。与之对应的 Z-N_PI 方法，由于减少了微分项，控制器的输出信号平滑性明显改善，但同时超调量大，其他指标也并不是很突出。SIMC_PID 方法上升时间、调整时间、回复时间虽然最大，但相较于其他，这一不利可以容忍，带来的好处是超调量大大减少，平滑性较同样是 PID 的 Z-N_PID 得到了较大改善，偏差也是最小的。SIMC_PI 方法相较于 SIMC_PID 减少了微分环节，平滑性大幅度改善，并且继承了 SIMC_PID 的优点。

综合起来考虑，SIMC_PI 方法是四种方法中较为优秀的整定方法。

参考资料

[1] 陈力,胡刚. 基于内模控制的互耦水槽建模与仿真[J]. 计算机仿真,2011

[2] 东方. 基于 Ziegler_Nichols 法则的 PID 控制器参数整定[J]. 自动化与仪器仪表,2015

[3] 薛定宇. 控制系统计算机辅助设计(第二版)[M]. 北京:清华大学出版社,2006

程序清单

build_model.m

```
%% 设定常量的值
% 其中 283 是阶跃信号跳变时的向量索引
load('plant_data1.mat');
t = t-t(283);          % 通过坐标变换将阶跃信号的跳变时间设置为 0
y = y2;
t0 = -437.6; U0 = 5; Uf = 6;
Y0 = 1.546; Yf = 3.675;
% 两条平行线的纵坐标
yy1 = 0.283*(Yf-Y0)+Y0;
yy2 = 0.632*(Yf-Y0)+Y0;

%% 绘制图像
plot(t,y); hold on;
fplot(yy1); fplot(yy2);

%% 求解模型参数
dis1 = zeros(size(y));
dis2 = zeros(size(y));
for i = 1:2606
    dis1(i) = abs(y(i)-yy1);
    dis2(i) = abs(y(i)-yy2);
end
[mdis1,index1] = min(dis1);
[mdis2,index2] = min(dis2);
t1 = t(index1); t2 = t(index2);
plot(t1,y(index1),'rp',t2,y(index2),'gp')    % 绘制出 t1、t2 用五角星标记
K = (Yf-Y0)/(Uf-U0);                 % 模型各个参数的求解
T = 1.5*(t2-t1); Td = t2-T;
if Td < 0
    T = t2; Td = 0;
end
```

```
%% 输出模型
G = tf(K,[T,1]); G.IODelay = Td;
ym = step(G,t(283:end));            % 基于模型的阶跃响应
plot(t(283:end),ym+Y0);             % 此处变换纵坐标，加上 Y0，更为直观
```

verify_model.m

```
%% 画出模型和实际的阶跃响应
load('plant_data1.mat');
t = t-t(283);
t = t(283:end);
y = y2-Y0;
y = y(283:end);
ym = step(G,t);
plot(t,y);
hold on; plot(t,ym);

%% 求 Ip 的值
Ip = 0;
dt = t(2)-t(1);
for k = 1:2324
    Ip = Ip+(1/(t(end)-t(1)))*(abs(y(k)-ym(k)))^2*dt;
end
```

f_analysis.m

```
%% 求解频域指标
% Gm 为幅值裕度，Pm 为相位裕度，
% Wgm 为幅值穿越频率，Wpm 为相角穿越频率
[Gm,Pm,Wgm,Wpm] = margin(G);

%% 画出系统的 Bode 图
bode(G); grid;
```

build_pid.m

```
%% Ziegler-Nichols 方法的 PI 控制器
Kp1 = 0.9*T/(K*Td);
Tint1 = Td/0.3;
s = tf('s');
Gznpi = Kp1*(1+1/(Tint1*s));

%% Ziegler-Nichols 方法的 PID 控制器
Kp2 = 1.2*T/(K*Td);
Tint2 = 2.2*Td;
Tdif2 = 0.5*Td;
Gznpid = Kp2*(1+1/(Tint2*s)+Tdif2*s);

%% SIMC 方法的 PI 控制器

tauc = Td;
Kc = T/K/(tauc+Td);
tauI = min([T,4*(tauc+Td)]);
Gsipi = Kc*(tauI*s+1)/(tauI*s);

%% SIMC 方法的 PID 控制器
taud = Td/3;
Gsipid = Kc*(tauI*s+1)*(taud*s+1)/(tauI*s);
```

compare_pid

```
%% 跟踪性能分析
% 涉及指标：上升时间、超调量、调整时间及平方偏差积分和绝对偏差积分

% 上升时间、超调量、调整时间
% 计算方法：画出阶跃响应曲线读图得到
subplot(2,2,1);
```

```
step(feedback(G*Gznpi,1)); title('Z-N\_PI 方法的闭环阶跃响应')
subplot(2,2,2);
step(feedback(G*Gznpid,1)); title('Z-N\_PID 方法的闭环阶跃响应')
subplot(2,2,3);
step(feedback(G*Gsipi,1)); title('SIMC\_PI 方法的闭环阶跃响应')
subplot(2,2,4);
step(feedback(G*Gsipid,1)); title('SIMC\_PID 方法的闭环阶跃响应')

% 平方偏差积分和绝对偏差积分
% 计算方法：画偏差曲线，按照定义求解
subplot(2,2,1); step(1/(1+Gznpi*G)); title('Z-N\_PI 方法的偏差')
subplot(2,2,2); step(1/(1+Gznpid*G)); title('Z-N\_PID 方法的偏差')
subplot(2,2,3); step(1/(1+Gsipi*G)); title('SIMC\_PI 方法的偏差')
subplot(2,2,4); step(1/(1+Gsipid*G)); title('SIMC\_PID 方法的偏差')
% 偏差
e1 = step(1/(1+Gznpi*G), 0:0.01:1000);
e2 = step(1/(1+Gznpid*G), 0:0.01:1000);
e3 = step(1/(1+Gsipi*G), 0:0.01:1000);
e4 = step(1/(1+Gsipid*G), 0:0.01:1000);
% 计算 ISE，Delta t 为 0.01
ISE1 = sum(e1.^2*0.01);
ISE2 = sum(e2.^2*0.01);
ISE3 = sum(e3.^2*0.01);
ISE4 = sum(e4.^2*0.01);
ISE = [ISE1 ISE2 ISE3 ISE4];
% 计算 IAE，Delta t 为 0.01
IAE1 = sum(abs(e1)*0.01);
IAE2 = sum(abs(e2)*0.01);
IAE3 = sum(abs(e3)*0.01);
IAE4 = sum(abs(e4)*0.01);
IAE = [IAE1 IAE2 IAE3 IAE4];

%% 抗干扰性能分析
% 涉及指标：动态降落和恢复时间
```

```
% 计算方法：Simulink 仿真

% 扰动信号加在受控对象的输入位置
% 扰动信号是在时间 t=500 时，由 0 跳变到 -1 的阶跃信号
% 建立 Simulink 模型时 PID 控制器的 PID 参数由函数 get_PID 得到
[P1, I1, D1] = get_PID(Gznpi);
[P2, I2, D2] = get_PID(Gznpid);
[P3, I3, D3] = get_PID(Gsipi);
[P4, I4, D4] = get_PID(Gsipid);
sim('check_robust');
% 画出响应曲线
subplot(2,2,1); plot(tout,yout(:,1)); title('Z-N\_PI')
subplot(2,2,2); plot(tout,yout(:,2)); title('Z-N\_PID')
subplot(2,2,3); plot(tout,yout(:,3)); title('SIMC\_PI')
subplot(2,2,4); plot(tout,yout(:,4)); title('SIMC\_PID')
% 数值方法求解动态降落
DeltaC1 = 1 - min(yout(find(tout==500):end,1));
DeltaC2 = 1 - min(yout(find(tout==500):end,2));
DeltaC3 = 1 - min(yout(find(tout==500):end,3));
DeltaC4 = 1 - min(yout(find(tout==500):end,4));
DeltaC = [DeltaC1 DeltaC2 DeltaC3 DeltaC4];
% 数值方法求解恢复时间，取 5%
tv = [0 0 0 0];
for i = 1:4
    for index = find(tout==500):size(tout)
        if abs(1-yout(index,i)) >= 0.05*1 && abs(1-yout(index+1,i)) <= 0.05*1
            break;
        end
        tv(i) = tout(index);
    end
end

%% 控制器输出信号平滑性
% 计算方法：Simulink 仿真
```

```
sim('pid_out');
subplot(2,2,1);  plot(tout1,yout1(:,1));  title('Z-N\_PI')
subplot(2,2,2);  plot(tout1,yout1(:,2));  title('Z-N\_PID')
subplot(2,2,3);  plot(tout1,yout1(:,3));  title('SIMC\_PI')
subplot(2,2,4);  plot(tout1,yout1(:,4));  title('SIMC\_PID')
% 求平滑性的值
PingHua  =  [0  0  0  0];
for  i  =  1:4
    for  k  =  1:30000
        PingHua(i)  =  PingHua(i)  +  abs(yout1(k+1,i)-yout1(k,i));
    end
end
```

get_PID.m

```
function  [P,  I,  D]  =  get_PID(Gc)
% 此函数可以求取以 Simulink 并行 PID 控制器的参数
% 此函数仅仅适用于本课程报告给出的物理不可实现控制器形式

temp  =  Gc.num{1,1}  /  Gc.den{1,1}(end-1);
try
    P  =  temp(2);
    I  =  temp(3);
    D  =  temp(1);
catch
    P  =  temp(1);
    I  =  temp(2);
    D  =  0;
end
```