基于专家 PID 的温度控制系统

一、实验目的

1. 掌握 PID 控制算法的基本原理及实现方法
2. 了解 PID 控制算法中各个超参数对算法的影响
3. 掌握专家 PID 的控制原理
4. 熟悉 MATLAB 编程

二、实验环境

软件环境：MATLAB R2020a

系统环境：Windows10

三、系统描述

PID 控制是一种结构简单、易于工程化的经典闭环控制算法，常用于控制温度、压力、流量等过程量信号。PID 控制时域数学模型见式（1），该式为经典的线性控制算法，其输出为对偏差进行比例、微分、积分计算后的加权求和值。

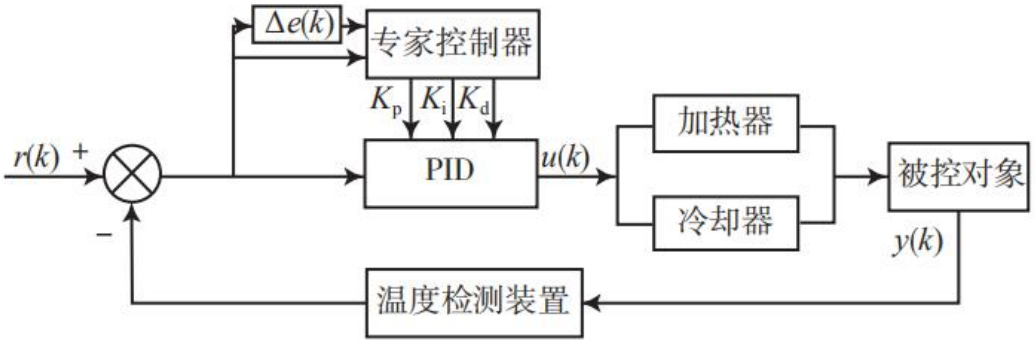
$$u(t)=K_p e(t)+K_i\int e(t)\mathrm{d}t+K_d\frac{\mathrm{d}e(t)}{\mathrm{d}t} \qquad (1)$$

式中：$U(t)$表示输出；Kp,Ki,Kd 分别表示比例系数、微分系数、积分系数；$e(t)$表示偏差。

挤塑机温度控制系统属于经典的非线性，大滞后系统，采用传统的 PID 控制难以获取理想的响应。针对非线性控制系统，PID 控制的不足之处在于比例、积分、微分参数由工程技术人员根据被控系统模型和经验整定而来，且为离线调整，不能根据系统的变化实时调整。基于此，提出了专家 PID 控制。如图所示：

图中：$\Delta e(k)$表示温度偏差变化率; $r(k)$表示输入温度，℃;

$u(k)$表示输出控制信号; $y(k)$表示反馈温度，℃;



四、专家 PID 控制原理

与传统 PID 相比，专家 PID 控制根据现场温度及温度偏差变化率实时调整优化 PID 控制参数，其参数优化规则是根据专家工程经验制定，一定程度上符合温度控制的要求，具体参数优化规则如下：

1. 若$|e(k)| \geq$误差绝对值上限（Max1,本实验中 Max1 取 4）,此时$|e(k)|$非常大，应该取较大的 Kp（本实验中 Kp 取基础值的 10 倍），Kd 较小（本实验中取基础值的 1/10），Ki=0

2. 若误差绝对值中间值（Max2，本实验中取 0.05）$\leq |e(k)|, \Delta e(k) * e(k) > 0$ 此时 $e(k)$依然较大，应增大 Kp（本实验中取基础值的 10 倍）保持 $Ki,\ Kd$ 不变。

3. 若 Max2$\geq |e(k)|, \Delta e(k) * e(k) > 0$ 此时$|e(k)|$大，但 $e(k)$不大，保持 $Ki,\ Kd, Kp$ 不变。

4. 若$\Delta e(k) * e(k) < 0, \Delta e(k) * \Delta e(k-1) > 0$ 且 Max2> $|e(k)|$,此时 $e(k)$处于拐点，$e(k)$比较小，应当取较小的 Kp（本实验中为基础值的 0.06 倍），使 $Ki = 0,\ Kd = 0$

5. 若$\Delta e(k) * e(k) < 0, \Delta e(k) * \Delta e(k-1) < 0$ 且 Max2> $|e(k)|$, 此时 $e(k)$处于拐点，$e(k)$比较大，应当取较大的 Kp（本实验中为基础值的 3 倍），使 $Ki = 0,\ Kd = 0$

6. 若$|e(k)| < \varepsilon$，此时保持 $Kp,\ Ki$ 不变，使得 $Kd = 0$

   其中 0<Max2<Max1, ε为很小的正数（该实验中取 0.005）

五、实验步骤

首先需要获取连续系统的传递函数，挤塑机温度控制系统具有时变形、非线性和滞后性，其数学模型见式（2）

$$G(s)=\frac{Ke^{-\tau s}}{Ts+1} \qquad (2)$$

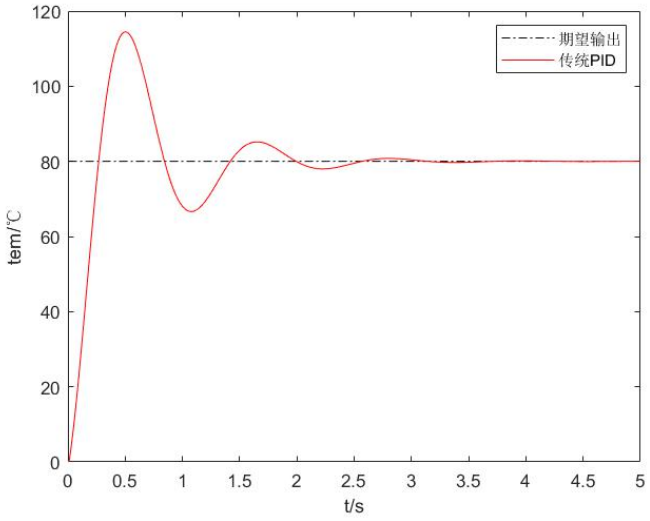
式中：K 表示稳态增益； τ 表示滞后时间；T 表示惯性常数；G(s)，s 分别表示拉氏变换的通用输出和变量。

由于是数字 PID 仿真，我们需要选取一个时间采样时间，本实验中取 0.005s，在对其进行数字 PID 控制前，我们首先将连续系统离散化，求解出 Z 变换表达式。在 PID 仿真过程中我们需要求解出时域表达式，因此需要借助差分方程解决。对刚才求解出的 Z 变换表达式进行逆 Z 变换，得到差分方程。然后构造出 PID 控制器，不断循环，求解出系统离散的响应点。在完成一个基本的 PID 之后，改变 PID 三个参数 Kp，Ki，Kd 的值，探究三个参数带来的效果。

在基础的 PID 系统中加入前文所述的专家规则，构造专家 PID 系统。最后对比 PID 和专家 PID 系统的输出结果，比较两者的差异，分析专家 PID 系统的特点。

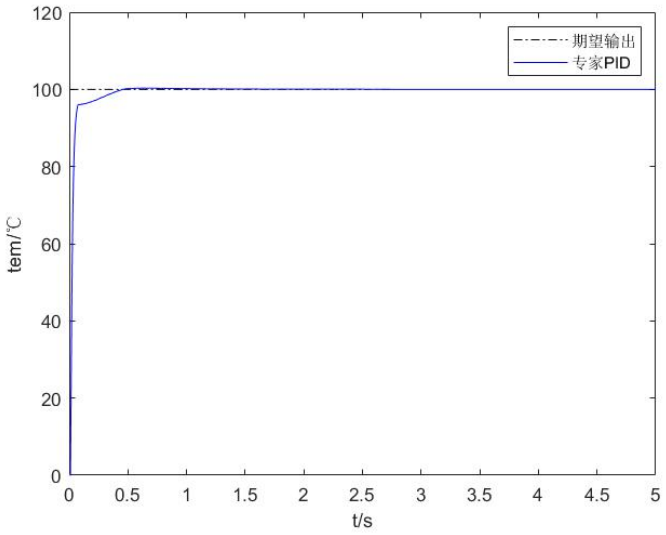
六、结果分析

利用 MATLAB 对 PID 和专家 PID 控制系统进行仿真，结果如图所示：



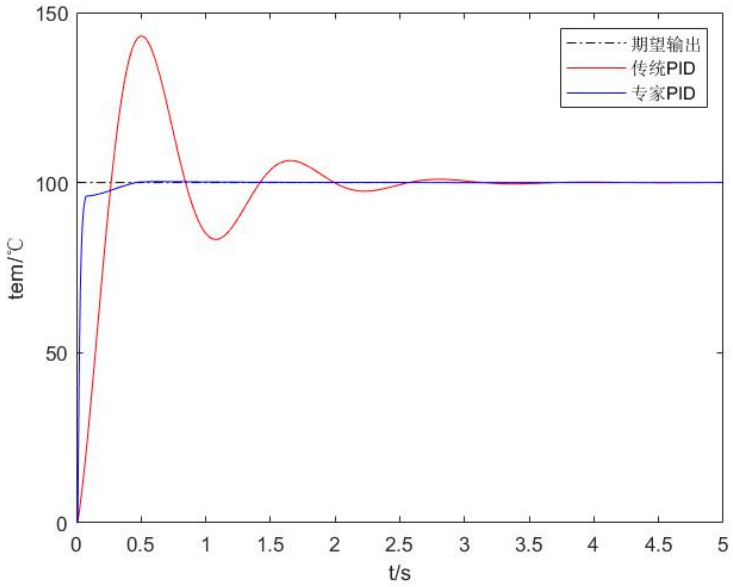
图表 1 传统 PID

传统 PID 的上升时间为 0.27s，超调量为 40%，稳定时间为 3.5s



图表 2 专家 PID

专家 PID 的上升时间为 0.40s，超调量为 0，稳定时间为 0.40s



图表 3 两种 PID 对比图

易得，在加入六条专家规则后，PID 系统有了更好的效果。专家 PID 系统几乎没有超调，且能在极短时间内达到稳态。

七、参考文献

[1].陈文科，刘　强，王健雄 基于专家 PID 的挤塑机温度控制系统设计

八、附录

```
%%
%清理屏幕
clc
clear
%%
%设置超参数
Kp=0.03;
Ki=0.002;
Kd=0.002;
tem_obj=100;%目标温度 80℃
time_end=5; %观察 5s
%%
%设置传递函数
T=1;%惯性系数
K=80;%稳态增益
tao=0.5;%滞后时间
ts=0.005; %采样时间=0.005s
s=tf('s');
sys=K*exp(-tao*s)/(T*s+1);%传递函数
dsys=c2d(sys,ts,'z');    %离散化
[num,den]=tfdata(dsys,'v'); %求解系数
%%
%初始化
e_before=0;      %前一时刻的误差
e_sum=0;         %累积偏差
e_now=0;         %现时刻的误差
u_before=0;    %前一时刻的控制量
u_now=0;         %现时刻的控制器输出
time=zeros(1,time_end/ts); %时刻点
y_before=0;      %前一时刻输出为 0
y=zeros(1,time_end/ts);   %普通系统输出
r=ones(1,time_end/ts)*tem_obj; %期望输出
%%
%传统 PID
for k=1:1:time_end/ts
   time(k)=k*ts;   %时间参数
   y(k)=-1*den(2)*y_before+num(2)*u_before+num(1)*u_now;%系统响应输出序列
   e_now=tem_obj-y(k);  %误差信号
   u_now=Kp*e_before+Ki*e_sum+Kd*(e_now-e_before); %系统 PID 控制器输出序列
   e_sum=e_sum+e_now;   %误差的累加和
   u_before=u_now;    %前一个的控制器输出值
   y_before=y(k);    %前一个的系统响应输出值
```

```
    e_before=e_now;   %前一个误差信号的值
end
plot(time,r,'k-.') %指令信号的曲线（即期望输入）
hold on;
plot(time,y,'r');%传统 pid
hold on;
xlabel('t/s');
ylabel('tem/℃');
%%
%初始化
e_before=0;       %前一时刻的误差
e_sum=0;          %累积偏差
e_now=0;          %现时刻的误差
u_before=0;     %前一时刻的控制量
u_now=0;          %现时刻的控制器输出
time=zeros(1,time_end/ts); %时刻点
y_before=0;       %前一时刻输出为 0
y=zeros(1,time_end/ts);   %专家系统输出
r=ones(1,time_end/ts)*tem_obj; %期望输出
%%
for k=1:1:time_end/ts
    deta_e=0;
    Kp=0.05;
    Ki=0.002;
    Kd=0.001;
    time(k)=k*ts;   %时间参数
    y(k)=-1*den(2)*y_before+num(2)*u_before+num(1)*u_now;%系统响应输出序列
    e_now=tem_obj-y(k);  %误差信号
    max1=4;
    max2=0.05;
    %规则 1
    if(abs(e_now)>=max1)
        Kp=Kp*10;
        Ki=0;
        Kd=Kd/10;
    %规则 2
    elseif(abs(e_now)>=max2&&e_now*(e_now-e_before)>=0)
        Kp=Kp*10;
        Ki=Ki;
        Kd=Kd;
    %规则 3
    elseif(abs(e_now)<max2&&e_now*(e_now-e_before)>=0)
        Kp=Kp;
        Ki=Ki;
```

```
    Kd=Kd;
%规则 4
elseif(e_now*(e_now-e_before)<0 && deta_e*(e_now-e_before)>0 && abs(e_now)<max2)
    Kp=0.06*Kp;
    Ki=0;
    Kd=0;
%规则 5
elseif(e_now*(e_now-e_before)<0 && deta_e*(e_now-e_before)<0 && abs(e_now)<max2)
    Kp=3*Kp;
    Ki=0;
    Kd=0;
 %规则 6
elseif(abs(e_now)<0.005)
    Kp=Kp;
    Ki=Ki;
    Kd=0;
end
deta_e=e_now-e_before; %记录 deta_e
u_now=Kp*e_before+Ki*e_sum+Kd*(e_now-e_before); %系统 PID 控制器输出序列
e_sum=e_sum+e_now;   %误差的累加和
u_before=u_now;    %前一个的控制器输出值
y_before=y(k);    %前一个的系统响应输出值
e_before=e_now;    %前一个误差信号的值
end
plot(time,y,'b');%专家 pid
hold on;
legend('期望输出','传统 PID','专家 PID');
```