模糊滑模控制报告

18 测试 7 班　　郑超　　201820502010

一、 应用背景

本文立足于本人所研究的方向（滑模算法在电机控制中的应用），滑模算法由于自身特性，不可避免地在控制中存在抖振问题，结合课上所学，考虑引入模糊控制，设计模糊逻辑规则实现基于模糊切换增益调节的滑模控制。实验中为了验证方法的有效性，简化验证过程，输入正弦函数为位置指令信号，通过比较传统滑膜控制和模糊滑模控制的输出结果来说明引入模糊控制的优点。

二、 理论介绍

2.1 滑模变结构控制

滑模变结构控制理论是一种处理线性系统和非线性系统的鲁棒控制方法。在变结构控制系统中，系统的运动可分为两个阶段：第一阶段是到达阶段，即滑模控制中的趋近过程，在该过程中，由到达条件保证系统运动在有限时间内从任意初始状态到达切换面；第二阶段是系统在控制律作用下保持滑模运动。

从理论角度上讲，理想的滑模变结构系统的控制结构切换具有理想开关特性 u (x) = u } (x)sign(s(x))，切换频率为无限大，系统状态测量精确无误，控制量不受限制，能在切换面上生成滑动模态，且滑动模态总是降维的光滑运动并渐进稳定于原点。但是，在实际系统中，理想的开关特性是不可能实现的。例如，对实际的连续系统进行控制时，模拟切换装置总是存在着一定的延迟和滞后;在数字控制的场合，由于采样周期总有一个极限，而切换频率又不可能超过采样频率，所以也存在一种延迟作用。使得滑模运动并不产生在预定的切换平面上，而是在其两侧的附近区域内产生一种高频振动，滑动模态呈抖动形式，这种现象称为滑模变结构控制的“抖振”。

2.2 模糊滑模控制

模糊滑模控制将模糊控制与滑模变结构控制相结合，控制目标从跟踪误差转

为滑模切换函数，通过设计模糊规则，使滑模函数为零，从而使得跟踪误差也渐进到达零点。模糊滑模控制柔化了控制信号，减轻或避免了一般滑模控制的抖振现象，保持了变结构控制对参数摄动和干扰不灵敏的特点，能够达到保持良好的跟踪性能和抑制抖振的效果。

利用模糊规则自应适地调整符号函数的幅度，亦即切换增益。根据经验，以降低抖振来设计模糊规则，可有效地削弱滑模控制的抖振幅度。

三、 实例分析

通过一个实例说明模糊滑模的用法：

$$\ddot{\theta} = f\left(\theta, \dot{\theta}\right) + b(u(t) + E(t))$$

其中$f\left(\theta, \dot{\theta}\right)$为已知，b>0, E(t)为未知干扰。

取滑模函数为：

$$s = e + \dot{c}e$$

其中 c>0,e 为跟踪误差，$e = \theta_d - \theta$, θ_d为指令角度。

设计滑模控制器为：

$$u = \frac{1}{b}(-f(\theta) + \ddot{\theta}_d + c\dot{e} + K(t)\mathrm{sgn}(s))$$

其中$K(t) = \max|E(t)| + \varphi$, φ>0。

在滑模控制器中切换增益K(t)值是造成抖振的原因，K(t)用于补偿不确定项E(t),以保证滑模存在性条件得到满足。如果 E(t)时变，则为了降低抖振，K(t)也应该是时变。可采用模糊规则，根据经验实现 K(t)的变化。

滑模存在条件为$s\dot{s} < 0$,当系统到达滑模面后，将会保持在滑模面上，K(t)就是保证系统运动达到滑模面的增益，其值必须足以消除不确定项的影响，才能保证滑模存在条件$s\dot{s} < 0$成立。

模糊规则如下：

If $s\dot{s} > 0$,则 K(t)应增大
If $s\dot{s} < 0$,则 K(t)应减小

据此规则设计$s\dot{s}$和ΔK之间关系的模糊系统，该系统中，$s\dot{s}$为输入，ΔK 为输出，系统输入输出的模糊集分别定义如下：

$s\dot{s}$={NB　NM　ZO　PM　PB}

$$\Delta K=\{NB \quad NM \quad ZO \quad PM \quad PB\}$$

其中，NB 为负大，NM 为负中，ZO 为零，PM 为正中，PB 为正大。

模糊系统的输入输出隶属函数为：

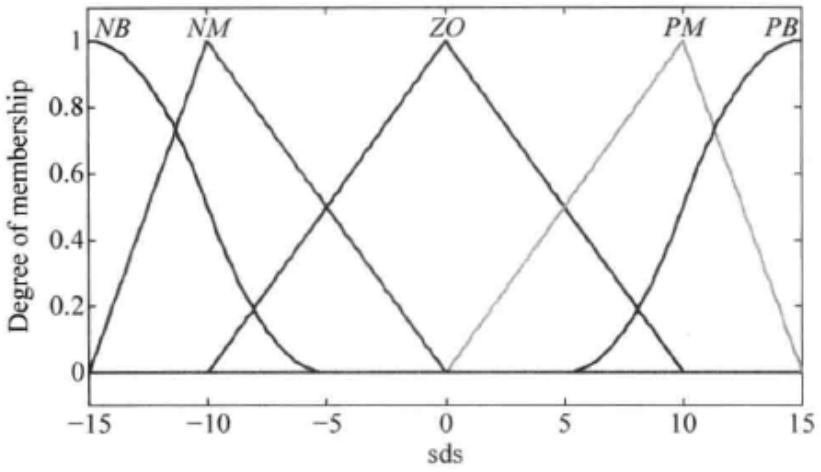

图 3.1 模糊输入的隶属函数

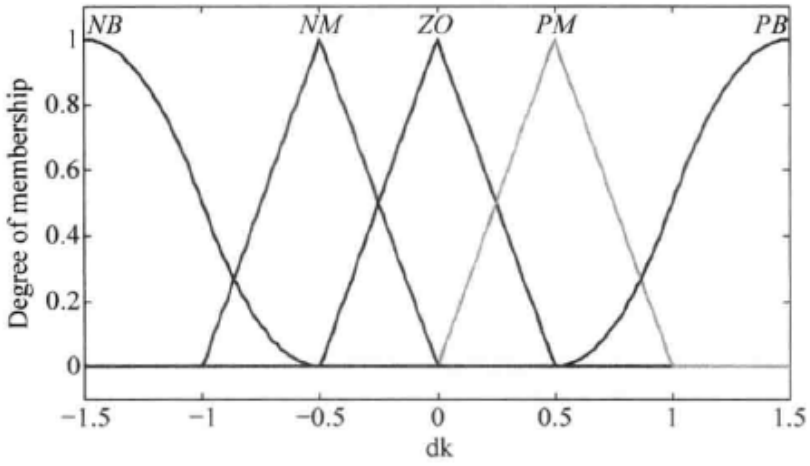

图 3.2 模糊输出的隶属函数

模糊规则设计如下：

R1: IF $s\dot{s}$ is PB THEN ΔK is PB

R2: IF $s\dot{s}$ is PM THEN ΔK is PM

R3: IF $s\dot{s}$ is ZO THEN ΔK is ZO

R4: IF $s\dot{s}$ is NM THEN ΔK is NM

R5: IF $s\dot{s}$ is NB THEN ΔK is NB

采用积分的方法对$\widehat{K}(t)$的上界进行估计：

$$\widehat{K}(t)=G\int_0^t \Delta K dt$$

其中，G 为比例系数，根据经验确定。

将经模糊控制器得到的$\hat{\mathrm{K}}(\mathrm{t})$代入到控制律中为：

$$\mathrm{u}=\frac{1}{b}(-f(\theta)+\ddot{\theta}_d+c\dot{e}+\hat{K}(\mathrm{t})sgn(s))$$

模糊滑模控制系统的结构如图所示：

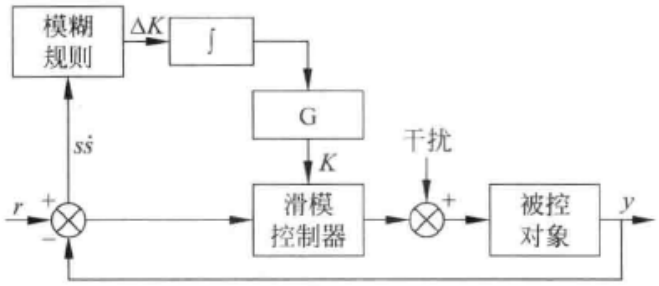

图 3.3 模糊滑模控制系统结构

四、建模仿真与分析

经过以上分析已确定系统对象和参数，通过 Matlab\Simulink 进行建模仿真，为了方便地改变系统参数，所用到的模块采用 S 函数编写。主要包括模糊系统设计 S 函数，控制器 S 函数，被控对象 S 函数，模糊系统 S 函数。模型如下：

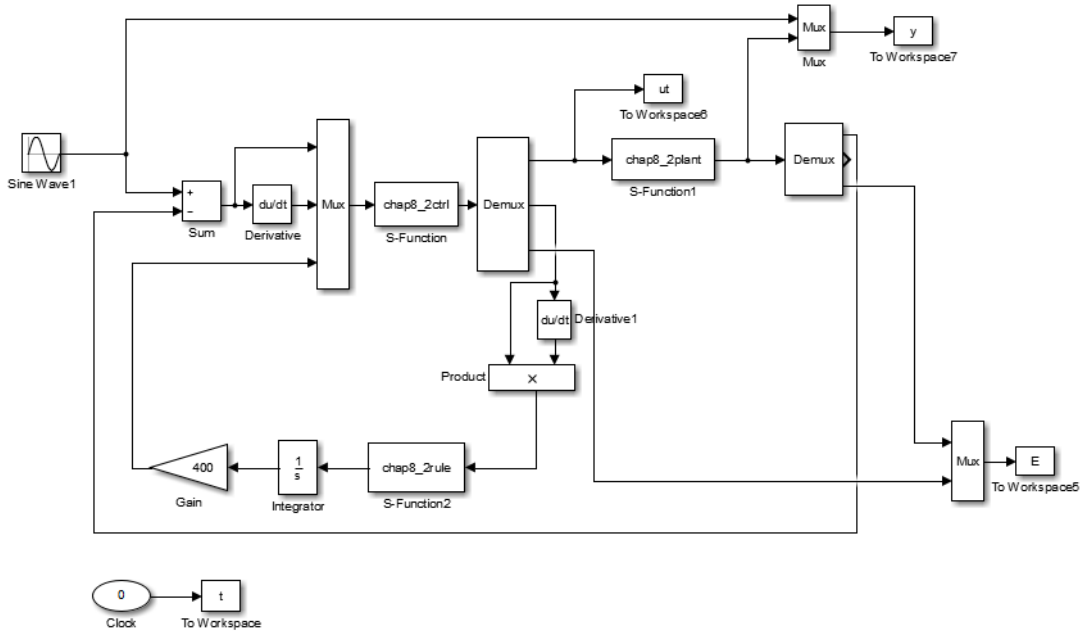

图 4.1 系统仿真结构模型

在这个模型中被控对象为：

$$\ddot{\theta} = f(\theta, \dot{\theta}) + b(u(t) + E(t))$$

其中$f(\theta, \dot{\theta}) = -25\dot{\theta}$，b=133。

采用高斯函数的形式表达不确定项：

$$E(t) = 200\exp(-\frac{(t - c_i)^2}{2b_i^2})$$

其中$b_i = 0.50, c_i = 0.50$。

为了证明模糊滑膜控制的有效性，输入位置指令信号$\theta_d = \sin(2\pi t)$,观察两种情况下位置与速度跟踪曲线，输出结果如下图所示。

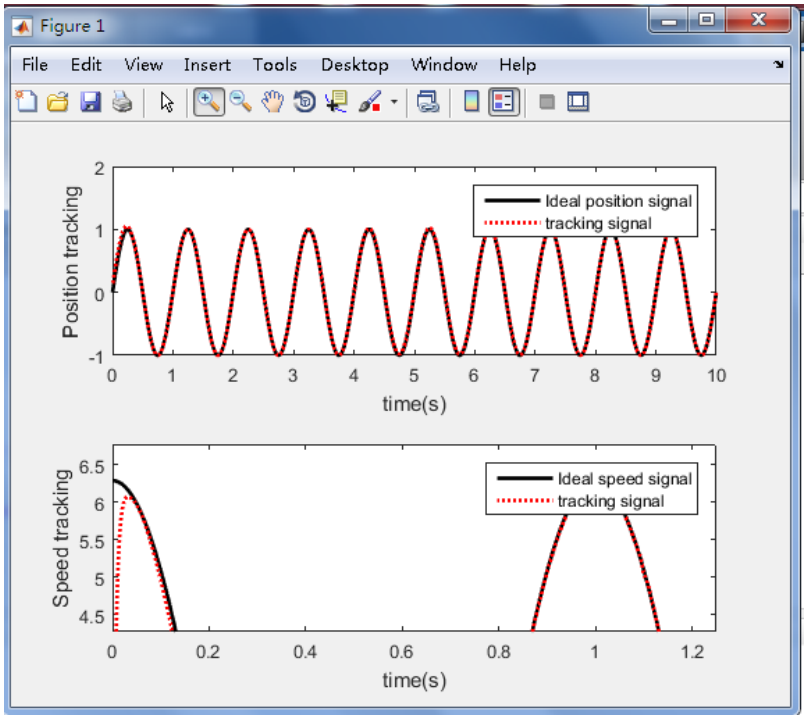

图 4.2 采用模糊滑模控制器时的位置和速度跟踪

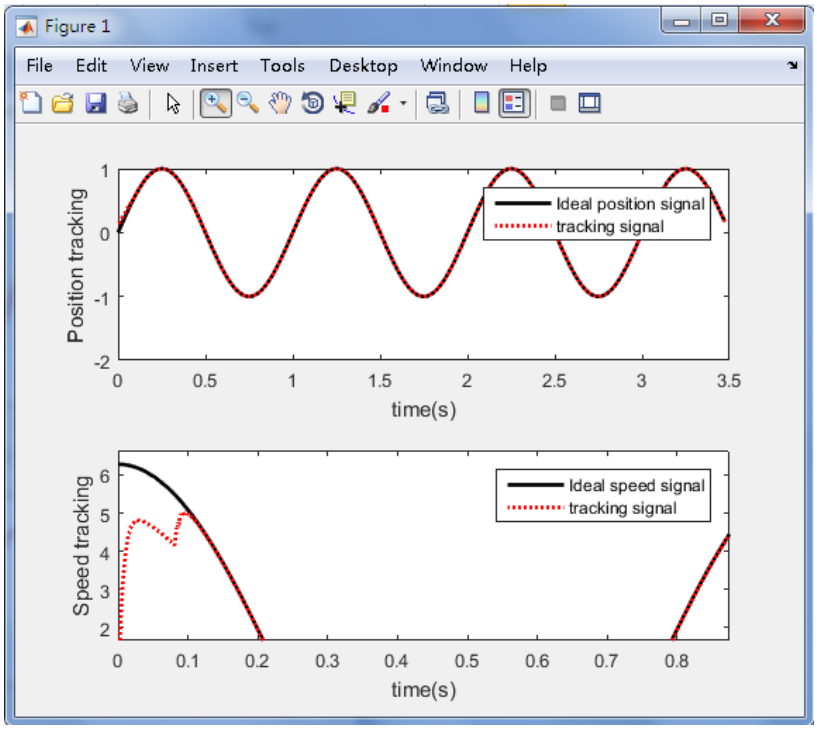

图 4.3 采用传统滑膜控制器时的位置和速度跟踪

由实验结果得：图一图二局部放大图对比可知模糊控制环节的加入使得跟踪曲线更加快速准确，明显减少了抖振。采用基于模糊规则的模糊滑模控制方法，可有效地通过切换增益消除干扰项，从而消除抖振。反应在电机上使得电机转速更加稳定，这在实际应用中显得意义重大。

五 附录

为了修改参数方便，建模时没有调用 simulink 自带的模块，采用 S 函数编写。部分函数如下所示：

模糊规则 S 函数：

```
clear all;
close all;
a=newfis('smc_fuzz');
f1=5;
a=addvar(a,'input','sds',[-3*f1,3*f1]);
a=addmf(a,'input',1,'NB','zmf',[-3*f1,-1*f1]);
a=addmf(a,'input',1,'NM','trimf',[-3*f1,-2*f1,0]);
a=addmf(a,'input',1,'Z','trimf',[-2*f1,0,2*f1]);
a=addmf(a,'input',1,'PM','trimf',[0,2*f1,3*f1]);
a=addmf(a,'input',1,'PB','smf',[1*f1,3*f1]);
```

```
f2=0.5;
a=addvar(a,'output','dk',[-3*f2,3*f2]);
a=addmf(a,'output',1,'NB','zmf',[-3*f2,-1*f2]);
a=addmf(a,'output',1,'NM','trimf',[-2*f2,-1*f2,0]);
a=addmf(a,'output',1,'Z','trimf',[-1*f2,0,1*f2]);
a=addmf(a,'output',1,'PM','trimf',[0,1*f2,2*f2]);
a=addmf(a,'output',1,'PB','smf',[1*f2,3*f2]);

rulelist=[1 1 1 1;
        2 2 1 1;
        3 3 1 1;
        4 4 1 1;
        5 5 1 1];

a1=addrule(a,rulelist);  ò
a1=setfis(a1,'DefuzzMethod','centroid');  .¨
writefis(a1,'smc_fuzz');
a1=readfis('smc_fuzz');
figure(1);
plotmf(a1,'input',1);   figure(2);
plotmf(a1,'output',1);
```

被控对象 S 函数：

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
   [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
   sys=mdlDerivatives(t,x,u);
case 3,
   sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
   sys = [];
otherwise
   error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
```

```
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[0.15,0];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
%bi=0.05;ci=5;
bi=0.5;ci=5;
dt=200*exp(-(t-ci)^2/(2*bi^2));   %rbf_func.m
%dt=0;

sys(1)=x(2);
sys(2)=-25*x(2)+133*u+dt;
function sys=mdlOutputs(t,x,u)
%bi=0.05;ci=5;
bi=0.5;ci=5;
dt=200*exp(-(t-ci)^2/(2*bi^2));   %rbf_func.m
%dt=0;

sys(1)=x(1);
sys(2)=x(2);
sys(3)=dt;
```

控制器S函数

```
function [sys,x0,str,ts]=s_function(t,x,u,flag)
switch flag,
case 0,
   [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
   sys=mdlOutputs(t,x,u);
case {2, 4, 9 }
   sys = [];
otherwise
   error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
```

```
sizes.NumSampleTimes = 0;
sys=simsizes(sizes);
x0=[];
str=[];
ts=[];
function sys=mdlOutputs(t,x,u)
persistent s0
e=u(1);
de=u(2);

c=150;
thd=sin(2*pi*t);
dthd=2*pi*cos(2*pi*t);
ddthd=-(2*pi)^2*sin(2*pi*t);

x1=thd-e;
x2=dthd-de;

fx=-25*x2;b=133;

s=c*e+de;

D=200;xite=1.0;

M=2;
if M==1
   K=D+xite;
elseif M==2   %Estimation for K with fuzzy
   K=abs(u(3))+xite;
end

ut=1/b*(-fx+ddthd+c*de+K*sign(s));

sys(1)=ut;
sys(2)=s;
sys(3)=K;
```