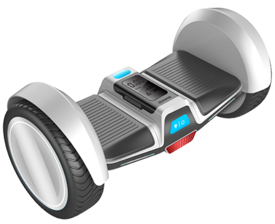
项目作业 平衡车控制

Author: 章翰宇
ID: 3220104133

Abstract

平衡车在日常生活中很常见，它脱胎于倒立摆的控制问题。本项目以平衡车为对象，将倒立摆问题深化，从一维导轨拓展到二维平面上的运动，从单一变量摆角的控制，拓展到三个变量的控制。项目中，使用模糊控制以及神经网络 PID 控制方法，设计出一种拥有比普通 PID 更平滑控制效果，且能在模型参数大幅度变化后仍然成功控制的方案。最终方案为：用两个模糊控制器与一个神经网络 PID 控制器，取代所有的普通 PID 控制器，达到更优的效果。

Keywords: 智能控制，平衡车，神经网络控制，模糊控制



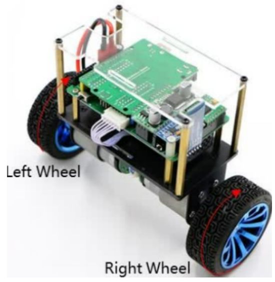


Table 1: 现实中的平衡小车，往往采用普通的 PID 控制策略

与"车载倒立摆"的区别

之前的车载倒立摆系统，考虑的固定在导轨上的小车（SISO），在本例中，没有导轨，有两个轮子（甚至可以转弯），两个轮子的转速不同将造成偏航角的变化，偏航角也将纳入被控变量考虑。之前，输入系统的控制信号是水平力F，此处，输入信号是两个电各自的加速度，和实际开发平衡车产品中的控制问题条件一致。

之前倒立摆作业没有控制住位移，仅仅控制了摆角，此处我们的目标是控制在二维平面上运动并且稳定下来，因此本质是x的跟踪问题，但是依赖于摆角的变化，需要协调两者之间的关系。

因此，此处的力学分析要更加复杂全面一些，由于是二维运动，是 MIMO 问题，操控变量有两个，被控变量有 3 个（位移、倾角、航向）。

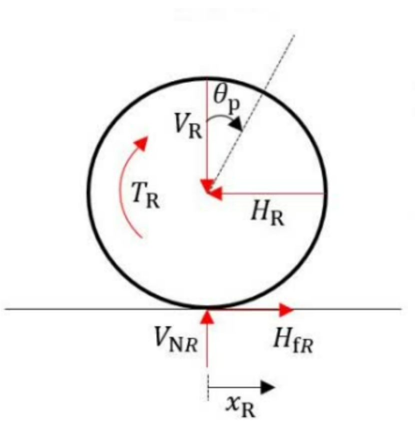
模型建立

符号说明

Symbol	Explanation
m	单个车轮的质量
M	车体的质量
x	两个轮子中心点的坐标
d	两个轮子之间的距离

J_p	车体绕质心俯仰转动的转动惯量
J_δ	车体绕质心偏航转动的转动惯量
I	一个轮子绕其轴转动的转动惯量
δ	整体的偏航角
θ_p	车体的俯仰角（前倾程度）
l	车体的质心到轮轴的距离
x_L, x_R	左，右轮子的水平方向位移
H_L, H_R	车体与左，右轮子之间的水平作用力大小
V_L, V_R	车体与左，右轮子之间的竖直作用力大小
H_{fL}, H_{fR}	左，右车轮与地面之间的摩擦力大小
T_L, T_R	左，右轮子处电机施加的矩大小

轮子受力分析



分析两个轮子的受力如上图，分别有合力公式与合力矩公式：

$$\begin{cases} \ddot{x}_R m = H_{fR} - H_R \\ \ddot{\theta} I = T_R - H_{fR} r \end{cases}$$

由于H_{fR}是地面给的摩擦力，是隐变量的地位，因此消去之：

$$\ddot{x}_R m = \frac{T_R - \ddot{\theta} I}{r} - H_R$$

根据无滑滚动约束条件，$\dot{x_R} = r\dot{\theta}$，因此将$\theta$消去，上式成为：

$$\ddot{x}_R \left(m + \frac{I}{r^2} \right) = \frac{T_R}{r} - H_R$$

同理：

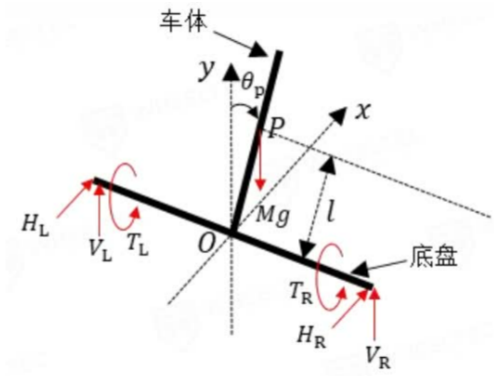
$$\ddot{x}_L \left(m + \frac{I}{r^2} \right) = \frac{T_L}{r} - H_L$$

由于$x=\frac{x_R+x_L}{2}$，因此：

$$\ddot{x}\left(m+\frac{I}{r^2}\right)=\frac{T_R+T_L}{2r}-\frac{H_L+H_R}{2} \tag{1}$$

$$(\ddot{x_L}-\ddot{x_R})\left(m+\frac{I}{r^2}\right)=\frac{T_L-T_R}{r}-(H_L-H_R) \tag{2}$$

车体受力分析



$$\begin{cases} M\frac{\mathrm{d}^2(x+l\sin\theta_p)}{\mathrm{d}t^2}=H_L+H_R \\ M\frac{\mathrm{d}^2(l\cos\theta_p)}{\mathrm{d}t^2}=V_L+V_R-Mg \\ J_p\ddot{\theta}_p=(V_L+V_R)l\sin\theta_p-(T_R+T_L)-(H_R+H_L)l\cos\theta_p \end{cases}$$

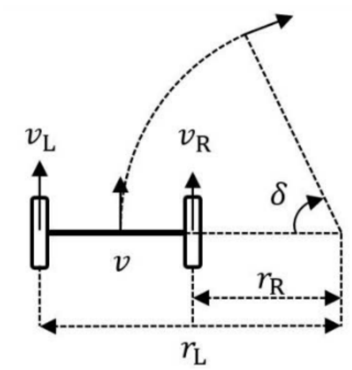
因此，结合轮子的受力分析消除内力(H_L+H_R)以及(V_L+V_R)部分，就能得到式子：

$$\begin{cases} \left(M+2m+\frac{2I}{r^2}\right)\ddot{x}-\frac{T_R+T_L}{r}+Ml\ddot{\theta}_p\cos\theta_p-Ml\dot{\theta}_p^2\sin\theta_p=0 \\ \left(\frac{J_p}{Ml}+l\right)\ddot{\theta}_p+\ddot{x}\cos\theta_p-g\sin\theta_p+\frac{T_L+T_R}{Ml}=0 \end{cases} \tag{3}$$

偏航转向分析

回顾d代表两个轮子之间的距离，因此：

$$J_\delta\ddot{\delta}=\frac{d}{2}(H_L-H_R)$$



由示意图的几何关系可得：

$$\dot{\delta} = \frac{\dot{x_L} - \dot{x_R}}{d}$$

把这个代入上式，并且结合 Equation 2，得到δ演化的动力学方程：

$$\ddot{\delta} = \frac{1}{r\left(md + \frac{Id}{r^2} + \frac{2J_\delta}{d}\right)}(T_L - T_R) \tag{4}$$

Simulink 建模

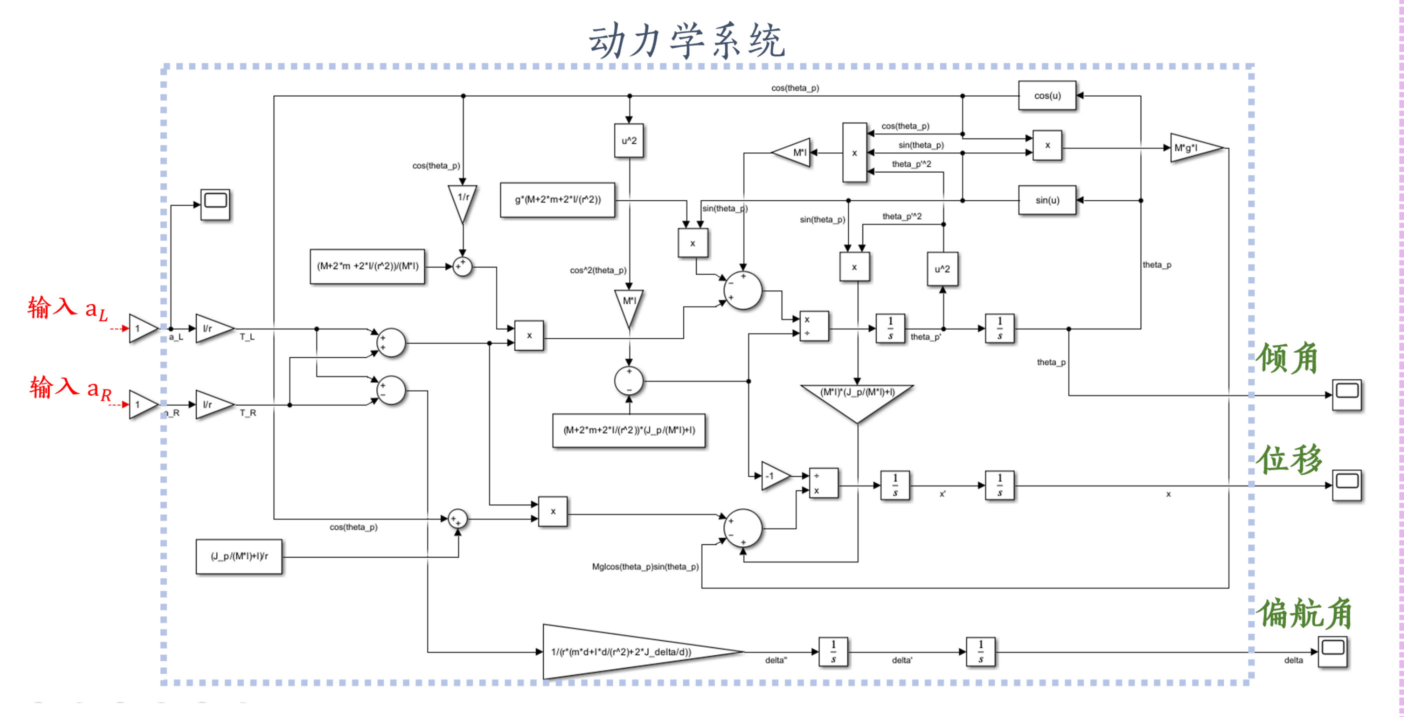
将 Equation 3 整理成为容易建模的形式如下（Equation 4 就不需要变了）：

$$\left[Ml\cos^2\theta_p - \left(M + 2m + \frac{2I}{r^2}\right)\left(\frac{J_p}{Ml} + l\right)\right]\ddot{\theta}_p = Ml\sin\theta_p\cos\theta_p\dot{\theta}_p^2 - g\sin\theta_p\left(M + 2m + \frac{2I}{r^2}\right) + \left[\frac{\cos\theta_p}{r} + \frac{\left(M + 2m + \frac{2I}{r^2}\right)}{Ml}\right](T_R + T_L) \tag{5}$$

$$\left[\left(\frac{J_p}{Ml} + l\right)\left(M + 2m + \frac{2I}{r^2}\right) - Ml\cos^2\theta_p\right]\ddot{x} = Ml\sin\theta_p\left(\frac{J_p}{Ml} + l\right)\dot{\theta}_p^2 - Mgl\cos\theta_p\sin\theta_p + \left[\cos\theta_p + \frac{\left(\frac{J_p}{Ml} + l\right)}{r}\right](T_R + T_L) \tag{6}$$

在绘制 Simulink 框图时，最后T_R与T_L这两者由于代表电机输出转矩的大小，不好直接控制，因此转化为两个车轮在无摩擦情况下的加速度:$a_L = \frac{r}{I}T_L, a_R = \frac{r}{I}T_R$。

Simulink



上图为建立的动力学系统框图，体现了整个模型的结构，整个开环对象，可以看到，是 2 输入 3 输出的。从物理含义上来看，“位移x”代表了一维的运动，“倾角θ_p”代表了平衡车的姿势，“偏航角δ”代表了二维平面上的转向。

普通控制器设计

思路概述

我的想法是先实现普通的 PID 控制，在成功后，使用更高级的模糊控制以及神经网络控制提升控制效果，逐步迭代。

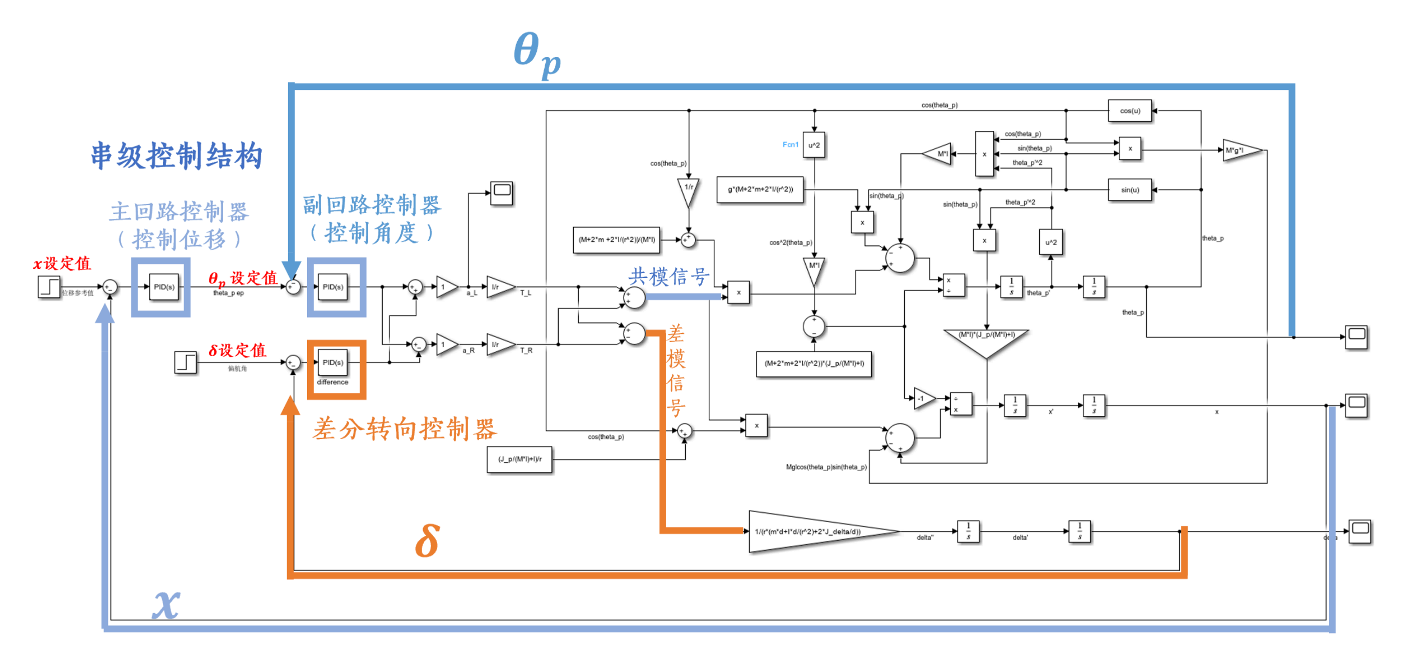
由于系统有三个输出，在实现简单的 PID 控制时，要考虑对象的工艺特性，即三个变量之间的逻辑关系。根据框图看出，可把两个输入视作其共模信号及差模信号。航向角δ的变化是双轮差动的结果，因此δ仅受到差模信号影响，共模信号对转向的贡献为0；同理，倾角与位移仅仅受到共模信号的影响，差模信号不会对倾角和位移产生作用。因此，可以很方便地解耦：把δ单独拎出去控制，操纵变量就是差模信号（用一个单回路控制它即可）；倾角和位移则统一用共模信号控制。

不难发现，位移和倾角都是由同一个操纵变量控制的！（2 输入 3 输出问题本身就蕴含“至少有两个变量是被同一个操纵变量控制”这一点）因此，不可能使用两个单回路控制成功。分析对象特性即知，倾角不为0时，车体是歪斜的，x也不可能稳定下来，因此两者的控制含有隐藏的逻辑：需先控制θ_p，再稳住x；x是时间常数大的主变量，而θ_p是响应速度快的副变量。可以借鉴**串级控制**思想。

注：可能会觉得这不是典型的串级控制。主副变量一个是x，一个是θ_p，光看 Simulink 框图可能难以看出其主/副层次关系。但是观察原微分方程，可以这样想：θ_p的变化确定后，x的各阶导数被相应的解出来。因此，为了简洁，下文中使用串级控制的概念解释。

至此，整个控制框架已经搭建，参见下图：

Simulink



接着具体分析三个控制器的特色：

1. 由于小车稳定下来以后，倾角一定是0，因此不管怎么样，控制倾角的控制器，都是一个 PD 控制器（不应该含有积分作用），故主回路控制器是 PD 控制器。
2. 由于差模信号非零时，一定会发生转向，因此到达稳态工作点时，差模信号必然为0，故，差分转向控制器也不用含积分项，同为 PD 控制器。

普通 *PID* 实现控制

（在文件 balance_car.slx 中）调整参数（Simulink 中的[P,I,D]值）:

- 主回路控制器=[0.01,0,0.2]
- 副回路控制器=[−280,-200,-120]
- 差分转向控制器=[1,0,0.8]

在三个被控变量$\ddot{\delta}, \ddot{x}, \ddot{\theta}_p$的处叠加上白噪音（作为理想模型之外的干扰），得到对比效果如下：

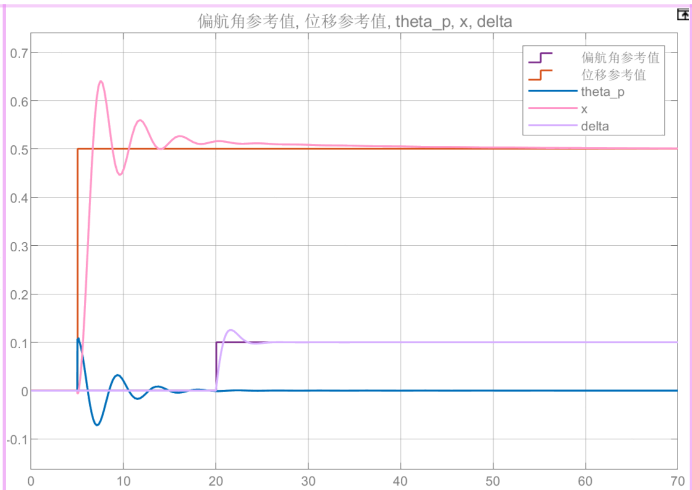
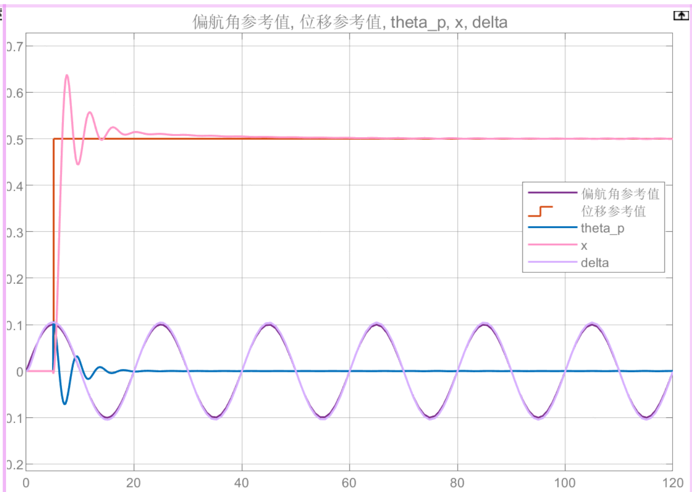
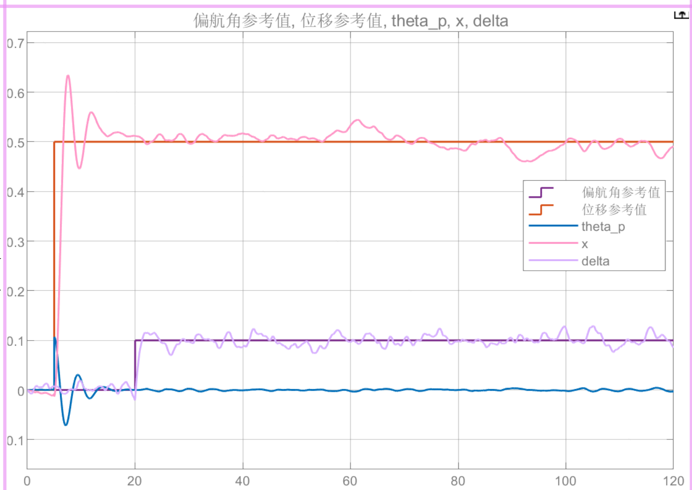
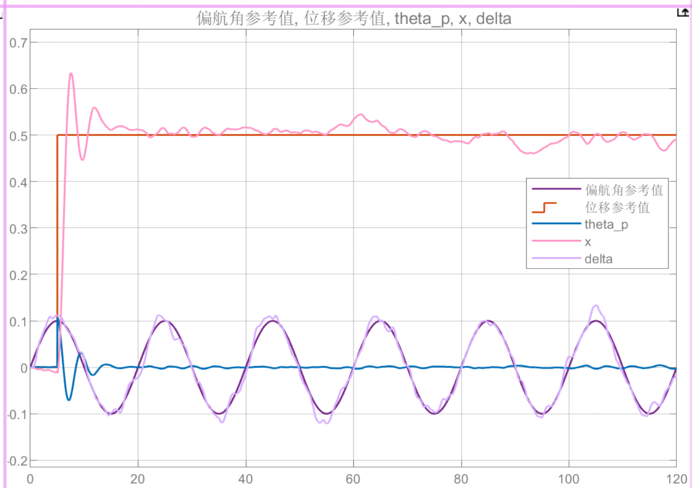
无噪音		
有噪音		

Table 2: PID 控制效果，Baseline

加入模糊控制

由于模糊控制器本身相当于 PD 控制器，因此可以将回路中的两个 PD 控制器（主回路、差分转向控制器）都用模糊控制器取代，实现更加平滑的控制效果。

FIS 使用 **Mamdani** 类型，下面展示本次实验使用的模糊控制器：

Fuzzification

模糊控制器的首个输入定义为位移误差偏移量e（即与目标稳态点差值），第二个输入为$\dot{e}$，现在，将位移偏差，和速度偏差的论域按照如下定义，设计两者的各自的一组隶属度函数：

位移偏移

主回路控制器中e的论域为$[-0.8, 0.8]$，取 5 个分类，中间用三角形函数，左右用 z 形和 s 形函数：

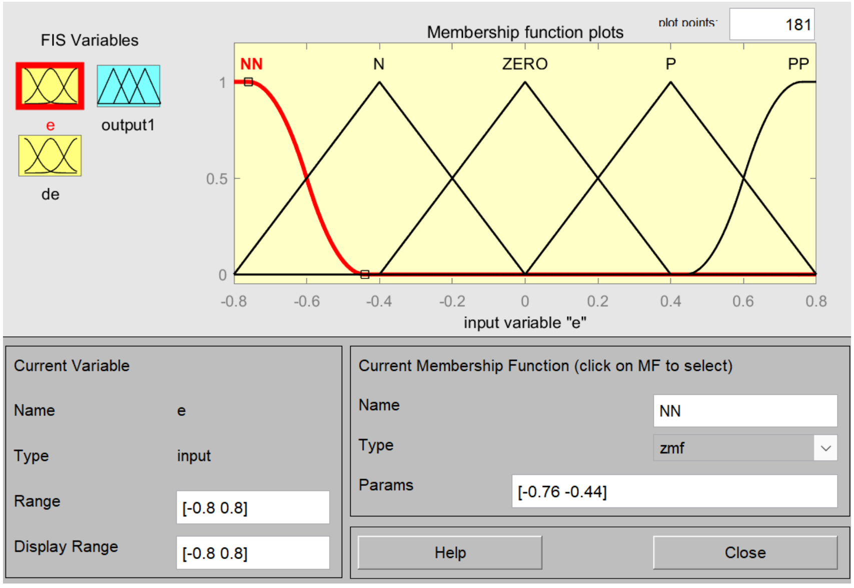


Figure 4: Δx的模糊规则

差分转向控制器中e的论域为[−0.3 0.3]，其余同。

误差导数

其论域为$[-0.5, 0.5]$，中间用三角形函数，左右用 z 形和 s 形函数：

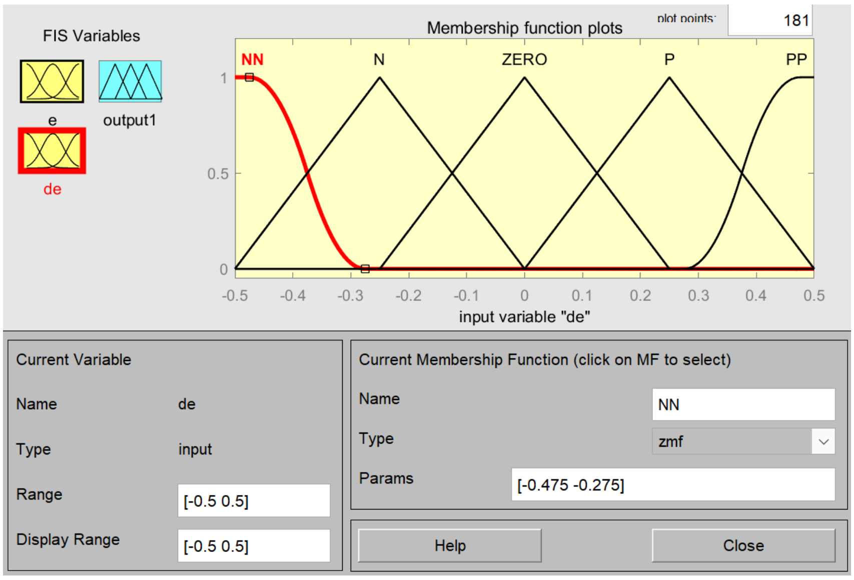


Figure 5: Δx的模糊规则

差分转向控制器中$\dot{e}$的论域为[−0.3 0.3]，其余同。

操作变量

输出的θ_p角度设定值的论域为$[-0.2, 0.2]$，取了 7 个分类，也均在中间用三角形函数，左右用 z 形和 s 形函数

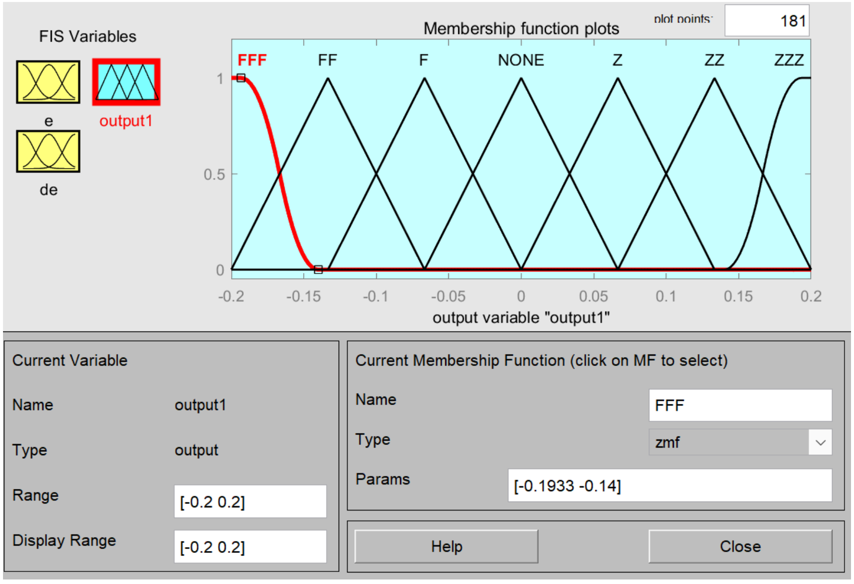


Figure 6: U的模糊规则

差分转向控制器中输出（θ_p设定值）的论域为$[-0.2\ 0.2]$，其余同。

Inference

在我的两个控制器的模糊推断操作中，使用隶属度函数的max来作为 OR（或）操作，用min来作为 AND（和）操作，如下图所示：

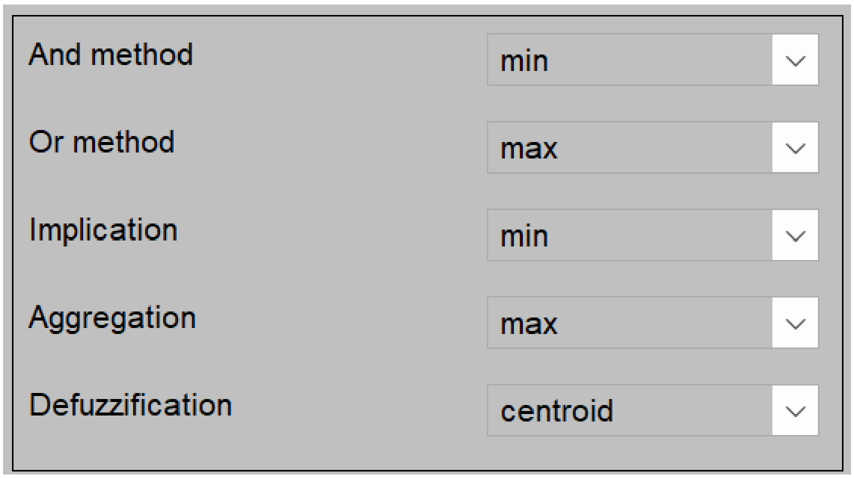


Figure 7: 模糊逻辑 推断规则

Defuzzification

两个控制器均使用 centroid 方法（见上图），即解模糊化的方法是计算两个区域的面积再取重心的横坐标（参见后文效果图）

Rules

两个控制器均采用下面的模糊规则。该控制规则的设计思路仍是：“远则快速接近，近则微调”。以主回路控制器为例说明：对于位移和速度均有负偏差时（说明小车不仅位置正向超前，而且速度也

是正向太快了)，需要输出较大的负向角度，对于位移正向超前但速度为负向时，说明已经在返回的路上了，因此只用少量的负向角度，甚至偏正的角度，赶紧慢下来刹车就行。对于正反互换的情况完全同理，因此整张表呈现出中心对称的特点。

		e				
		NN	N	ZERO	P	PP
	NN	FFF	FFF	FF	F	NONE
	N	FFF	FF	F	NONE	Z
$\dot{e}$	ZERO	FF	F	NONE	Z	ZZ
	P	F	NONE	Z	ZZ	ZZZ
	PP	NONE	Z	ZZ	ZZZ	ZZZ

Visualization

这是 centroid 解模糊化规则的可视化（以主回路控制器为例，转向控制器同）:

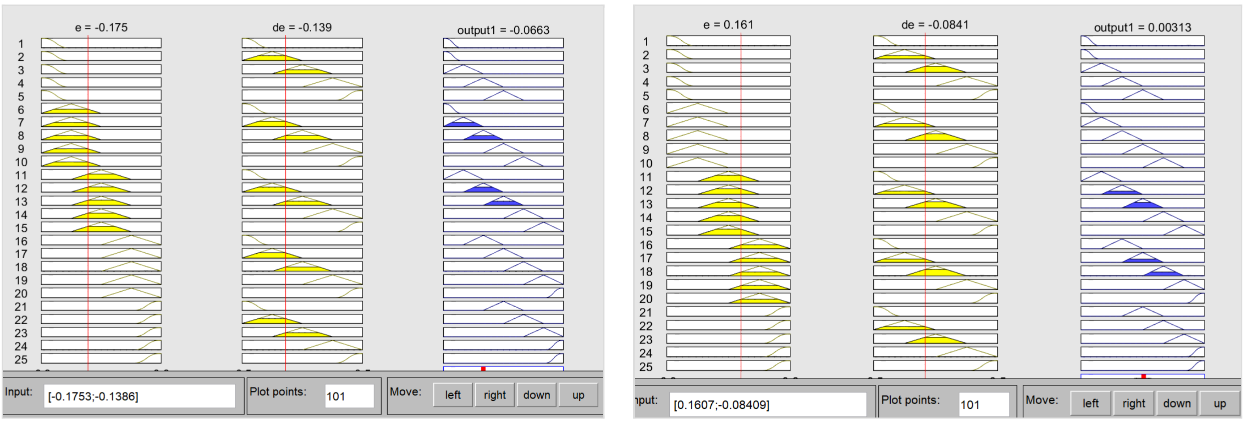


Table 3: 模糊规则的可视化

以下是通过控制曲面的方法可视化模糊控制器（以主回路控制器为例，转向控制器同）:

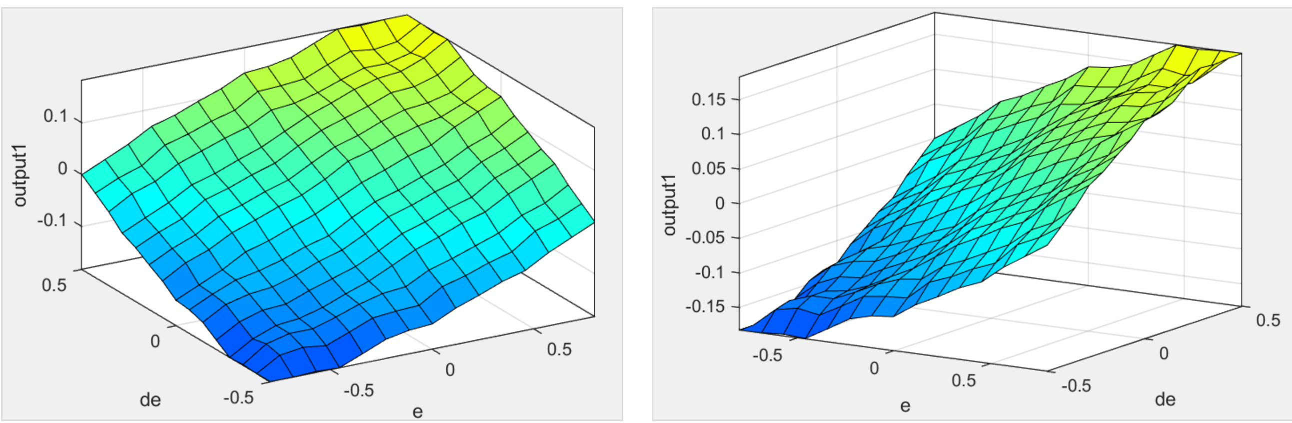
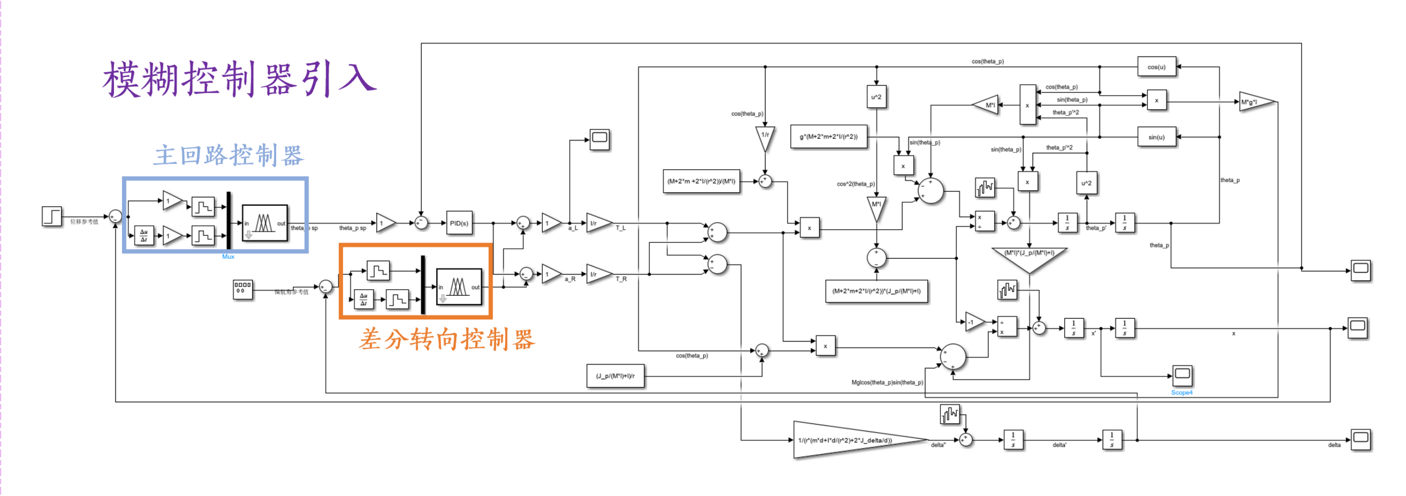


Table 4: 控制曲面

Simulink Model

Simulink



Result

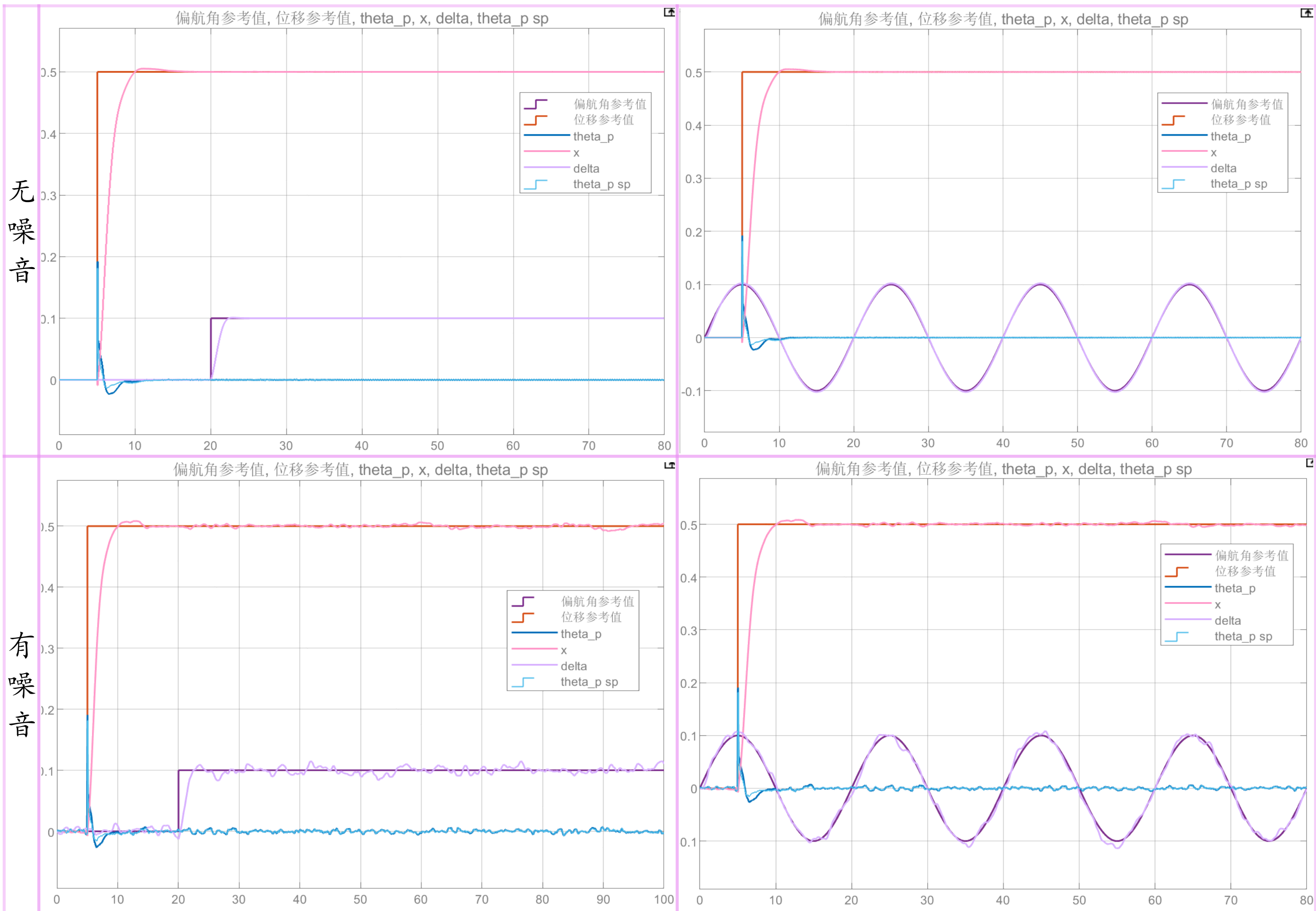


Table 5: 模糊控制效果

仍然使用之前 PID 的那几个输入波形，尝试跟踪，发现明显效果好于普通 PID，追踪十分平滑，没有振荡。

加入神经网络控制

神经网络 引入原因

尽管模糊控制取得了很好的效果，但是由于副回路还是用了 PID 控制器（积分项的作用难以用模糊控制器代替），因此还是有缺陷：**当模型参数发生变化，或者数据不准确时，有可能失控**，而果将副回路的控制器设计为神经网络 PID 控制器，让它自己找到合适的 P、I、D 参数，那么即使参数发生较大的变化，也能够取得良好的控制效果。

在我的架构里，主控制器是负责“上层指挥”，即根据x的测量值决定θ_p的设定值，因此模型参数发生变化，主回路控制器也并不需要调整。但是，具体θ_p怎么达到其设定值，是副回路的工作，因此副回路控制器才是与实际参数相关的，这就是把副回路的控制器用神经网络 PID 代替的理由。

总之，使用神经网络 PID **并不是因为它效果好**（普通 PID 只要参数设置正确，效果和神经网络 PID 当然差不多！）而是因为它能够**自己去找合适的三个参数**，因此带来巨大的便捷，非常 robust，模型**参数大幅度变化**，仍然能控制。

神经网络框架

下面，写了一个三层的 BP 神经网络，输出为 P、I、D 三个参数：

输入为$\begin{pmatrix} e(k) & e(k-1) & e(k-2) & e(k-3) \end{pmatrix}^\top$，首先乘一个上权重矩阵$W$，得到$\overrightarrow{net}$：

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{pmatrix} \begin{pmatrix} e(k) \\ e(k-1) \\ e(k-2) \\ e(k-3) \end{pmatrix} = \begin{pmatrix} \text{net1} \\ \text{net3} \\ \text{net3} \end{pmatrix} = \overrightarrow{net}$$

接着经过 sigmoid 函数激活: $\sigma(\overrightarrow{net}) = \overrightarrow{hidden\_output}$

$\overrightarrow{hidden_output}$作为下一层（输出层）的输入，输出层首先用权重矩阵$K$与之相乘：

$$\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} \text{hidden\_output1} \\ \text{hidden\_output2} \\ \text{hidden\_output3} \end{pmatrix} = \begin{pmatrix} \text{sum1} \\ \text{sum2} \\ \text{sum3} \end{pmatrix}$$

最后激活一下，作为**归一化**后的P, I, D三个参数的输出：

$$\begin{pmatrix} P \\ I \\ D \end{pmatrix} = \begin{pmatrix} \sigma(\text{sum1}) \\ \sigma(\text{sum2}) \\ \sigma(\text{sum3}) \end{pmatrix}$$

最后分别乘上K_p, K_i, K_d作为最终的三个参数，输出给 varying PID 模块。

数学分析

下面推导反向传播公式，将系统抽象为$u \to \text{system} \to y$，而$u$是根据 PID 规则给出的，而 PID 参数又是网络给出的，因此可以根据系统的输出y来反向传播，更新网络参数。首先取 loss 函数为$J = \frac{1}{2}e^2 = \frac{1}{2}(r-y)^2$，因此

$$\frac{\partial J}{\partial y} = e(k)$$

由于系统动力学模型未知（这也是智能控制的假设），可以用符号函数代替输入输出之间的偏导关系(这一项需要把观测到的两个时刻的y以及u都收集起来才能算):

$$\frac{\partial y}{\partial u} \approx \operatorname{sgn}\left(\frac{y(k)-y(k-1)}{u(k)-u(k-1)}\right) = \operatorname{sgn}((y(k)-y(k-1)) \times (u(k)-u(k-1)))$$

接下来计算输入u与三个参数的关系（回想增量式 PID 即得):

$$\frac{\partial u}{\partial P} = K_p(e(k)-e(k-1))$$
$$\frac{\partial u}{\partial I} = K_i e(k)$$
$$\frac{\partial u}{\partial D} = K_d(e(k)+e(k-2)-2e(k-1))$$

再根据 sigmoid 函数的导数，即得:

$$\frac{\partial P}{\partial \mathrm{sum1}} = P(1-P)$$
$$\frac{\partial I}{\partial \mathrm{sum2}} = I(1-I)$$
$$\frac{\partial D}{\partial \mathrm{sum3}} = D(1-D)$$

而剩下的就是常规的神经网络反向传播,

$$\frac{\partial\ \mathrm{sum}i}{\partial k_{ij}} = \sigma(\mathrm{net}_j) \quad i = 1,2,3$$

最后，对于输出层的更新为:

$$\frac{\partial J}{\partial k_{1j}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial P}\frac{\partial P}{\partial \mathrm{sum1}}\frac{\partial\ \mathrm{sum1}}{\partial k_{1j}}$$
$$\frac{\partial J}{\partial k_{2j}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial I}\frac{\partial I}{\partial \mathrm{sum2}}\frac{\partial\ \mathrm{sum2}}{\partial k_{2j}}$$
$$\frac{\partial J}{\partial k_{3j}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial D}\frac{\partial D}{\partial \mathrm{sum3}}\frac{\partial\ \mathrm{sum3}}{\partial k_{3j}}$$

对于隐藏层更加复杂，为:

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial u}\left(\frac{\partial u}{\partial P}\frac{\partial P}{\partial \mathrm{sum1}}\frac{\partial\ \mathrm{sum1}}{\partial \mathrm{hid\_output}i} + \frac{\partial u}{\partial I}\frac{\partial I}{\partial \mathrm{sum2}}\frac{\partial\ \mathrm{sum2}}{\partial \mathrm{hid\_output}i} + \frac{\partial u}{\partial D}\frac{\partial D}{\partial \mathrm{sum3}}\frac{\partial\ \mathrm{sum3}}{\partial \mathrm{hid\_output}i}\right) \times \frac{\partial\ \mathrm{hid\_output}i}{\partial \mathrm{net}i}\frac{\partial\ \mathrm{net}i}{\partial w_{ij}}$$

其中，$\frac{\partial\ \mathrm{sum1}}{\partial \mathrm{hid_output}i}$显然就是$K$矩阵中的元素；$\frac{\partial\ \mathrm{hid_output}i}{\partial \mathrm{net}i}$这一项就是 sigmoid 求导，即 $\mathrm{hid\_output}i(1-\mathrm{hid\_output}i)$；$\frac{\partial\ \mathrm{net}i}{\partial w_{ij}}$这一项即$e(k-j+1)$，至此，反向传播的每一项都清楚了，最后更新的结果即为:

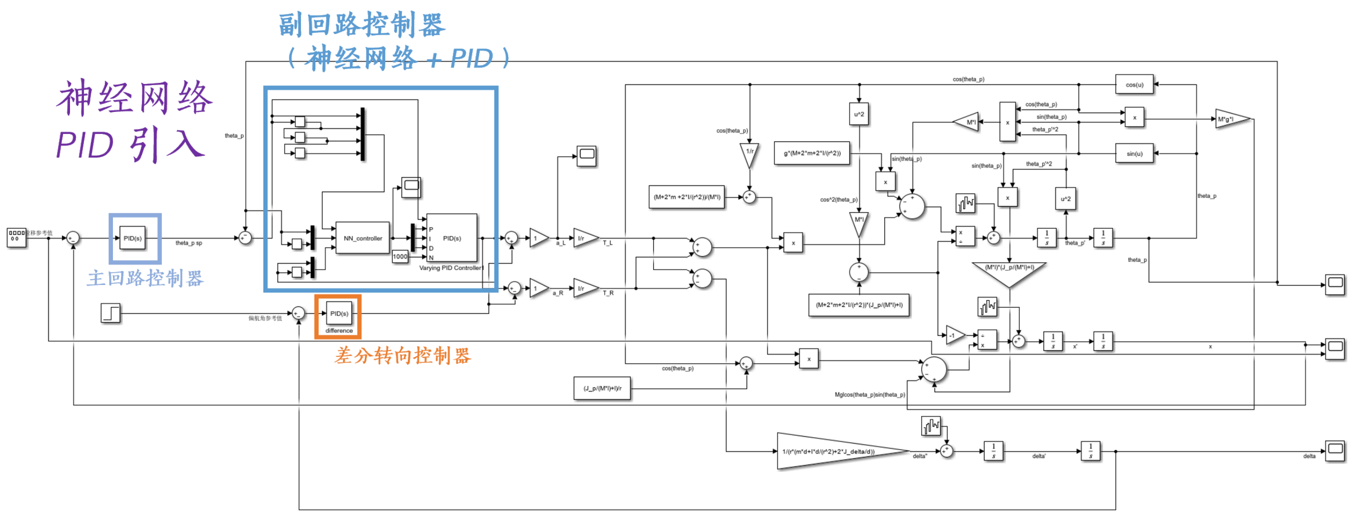
$$w_{ij} \leftarrow w_{ij} - \alpha \frac{\partial J}{\partial w_{ij}} \qquad k_{ij} \leftarrow k_{ij} - \alpha \frac{\partial J}{\partial k_{ij}}$$

沿袭以上记号，具体代码在文件 NN_controller.m 中，用 S-function 实现。

Simulink Model

（以下内容在 balance_car_with_NN.slx 文件）副回路控制器改用这个神经网络 PID 控制器，而主回路控制器和差分转向控制器仍然用普通 PID。

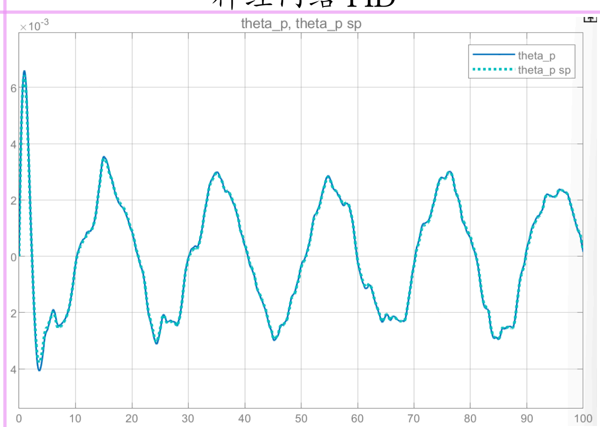
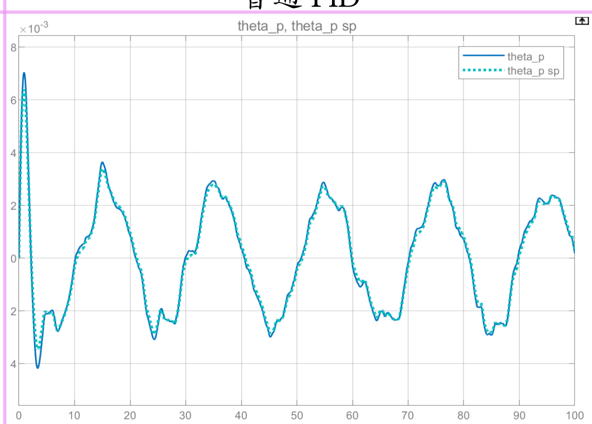
Simulink

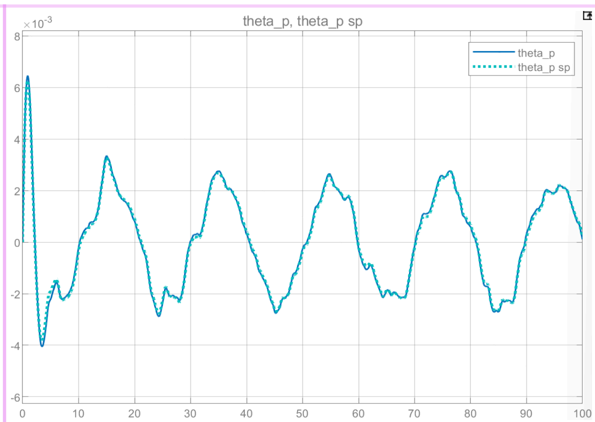
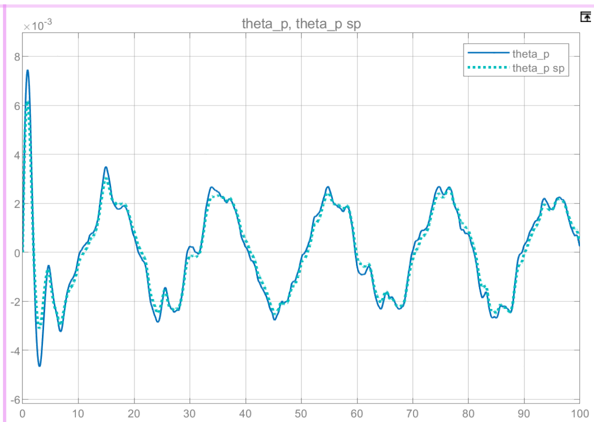
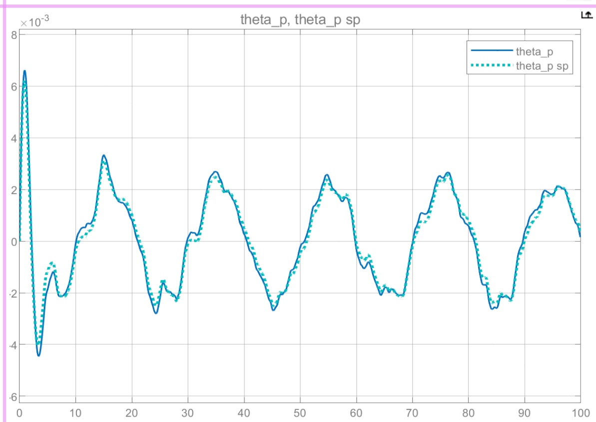
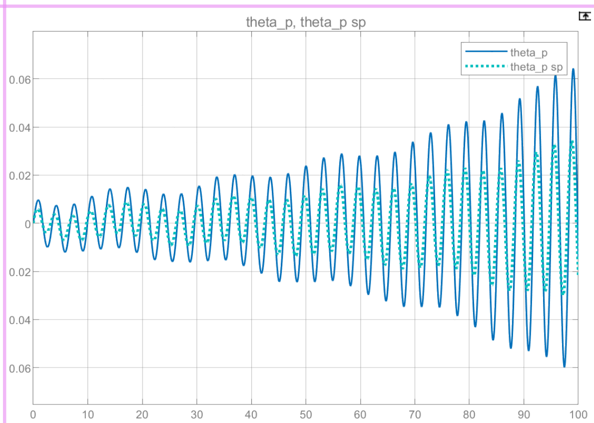


以下为副回路控制器用“神经网络 PID”和“普通 PID”效果的对比，运行 new_param.m 文件后，更改其中的m参数，即质量发生变化后，看看哪个控制器更 robust。位移x的参考信号是正弦函数。

以下为θ_p与其设定值的对比，在各个情况中，神经网络 PID 效果都很好，而普通 PID 随着参数m的变小，效果越来越差：

```
m => 变化 %% 以下曲线使用参数如下
r = 0.07/2; I = 0.5*m*r^2; M = 0.757-2*m; l = 0.5*0.8;
J_p = (1/12)*M*(0.3^2+0.08^2); d = 0.1612; J_delta = (1/12)*M*(0.0930^2+0.0530^2);
```

m	神经网络 PID	普通 PID
0.2		

0.1		
0.04		

在m参数值不同的时候，可以在 scope 中观察神经网络输出向量，发现 P/I/D 在m不同时确实会自动变的不同。（可以点开 balance_car_with_NN.slx 文件中的 scope，看神经网络输出的那个向量值）

仔细观察$m = 0.04$的情况如下，普通 PID 控不住θ_p，导致了x也发生灾难性的发散。

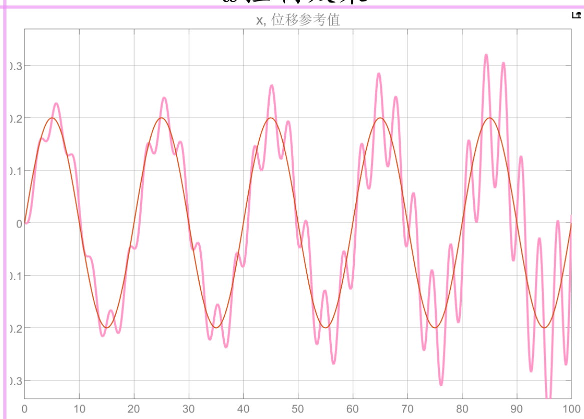
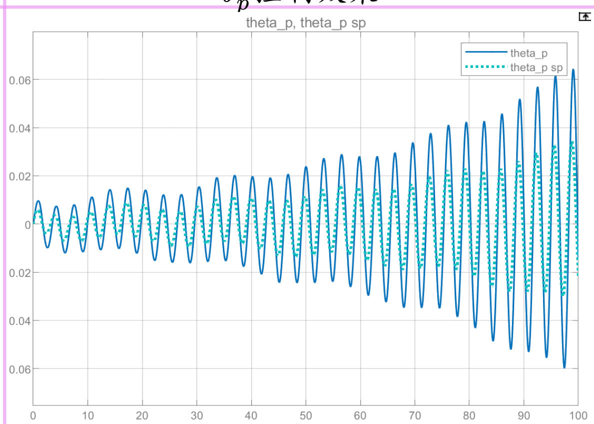
参数	x控制效果	θ_p控制效果
m = 0.04 PID		

Table 8: m=0.04 时的普通 PID 控制效果

但是神经网络 PID 却仍然可行。对比说明神经网络 PID 的确可以适应模型参数的变化：

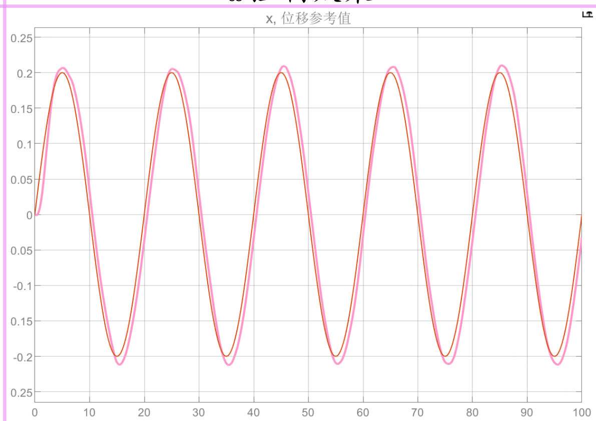
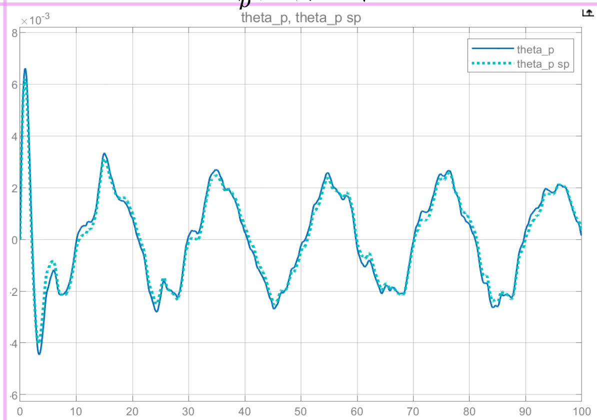
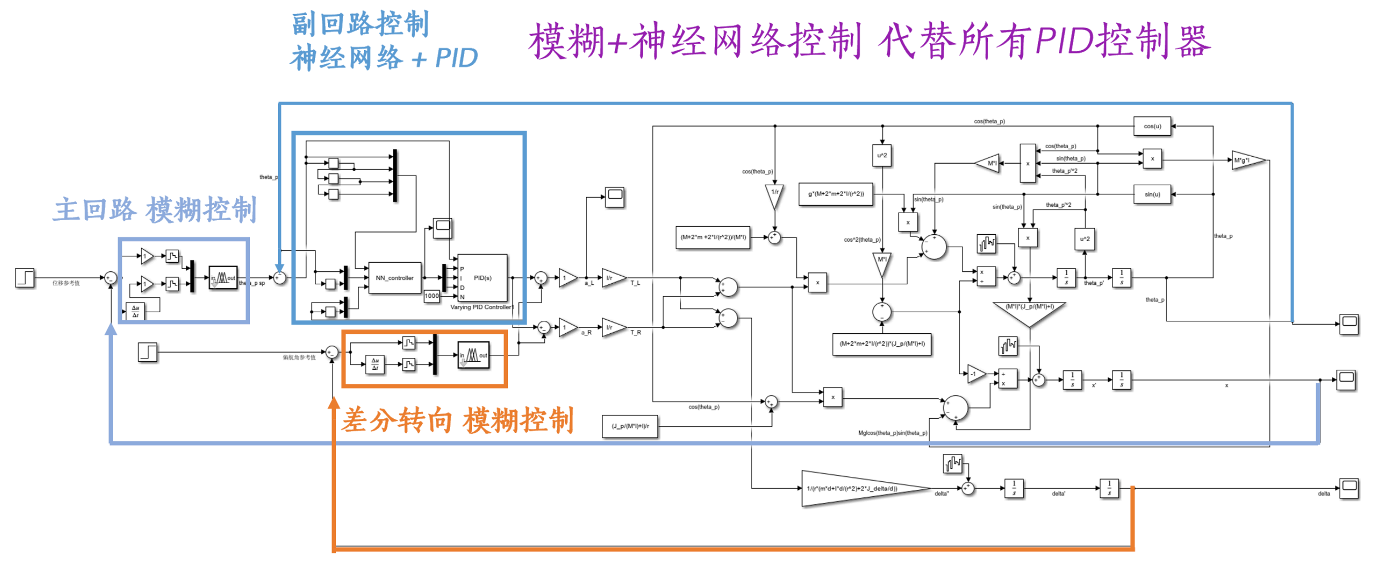
参数	x控制效果	θ_p控制效果
m = 0.04 NN		

Table 9: m=0.04 时的神经网络 PID 控制效果

神经网络 PID 与模糊控制 综合

（以下内容在 balance_with_fuzzy_and_NN.slx 文件中）模糊控制能够平滑实现x的跟踪（输出比 PID 更优的θ_p），而神经网络 PID 则对参数的变化有更强的适应性，结合两者，得到最终的控制结构，至此，已经将所有的普通 PID 都替换掉了！

Simulink



这个方案的优势在于，能够实现平滑阶跃跟踪，而且对于模型参数的变化 robust，下面是实验，接连改动 param.m 文件中的质量m以及l两个参数，两个参数都变化较大的倍数，看看是否仍然能控制（在控制过程中在是哪个变量$\ddot{x}, \ddot{\delta}, \ddot{\theta}_p$处都叠加加上白噪音噪音，模拟真实情况）。

```
%% 以下曲线使用参数如下
m => 变动; l => 变动;
r = 0.0672/2; I = 0.5*m*r^2;
M = 0.757-2*m; J_p = (1/12)*M*(0.0903^2+0.0530^2);
d = 0.1612; J_delta = (1/12)*M*(0.0930^2+0.0530^2);
```

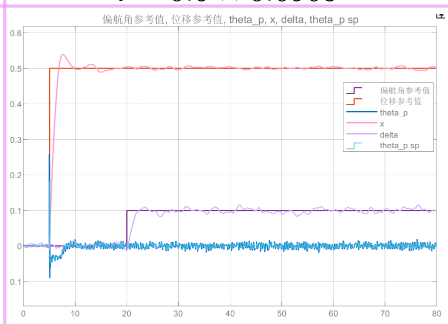
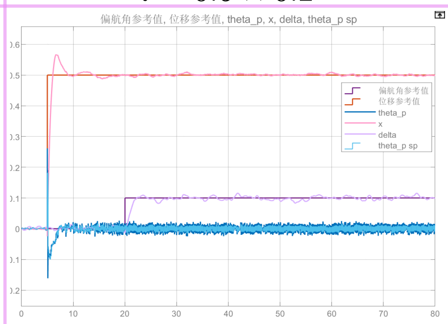
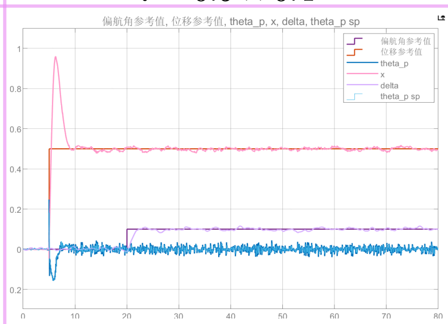
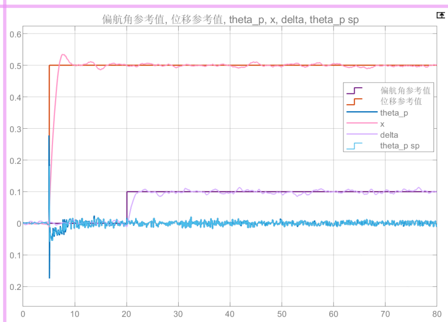
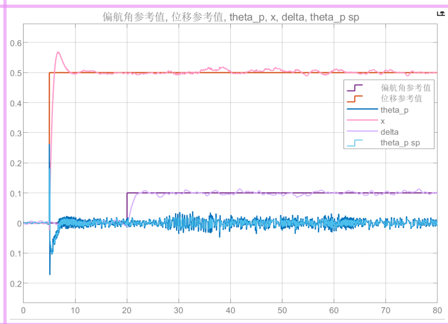
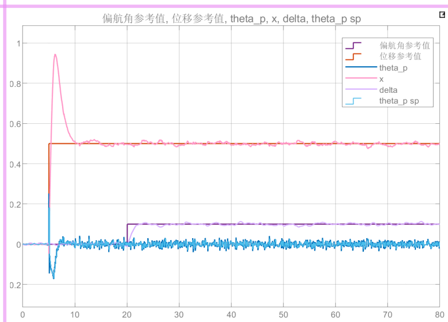
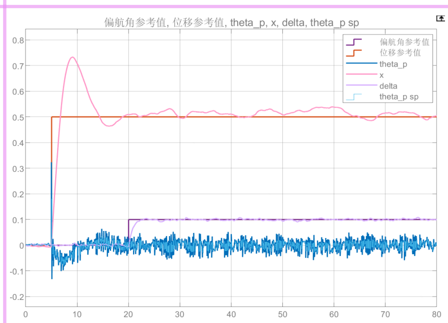
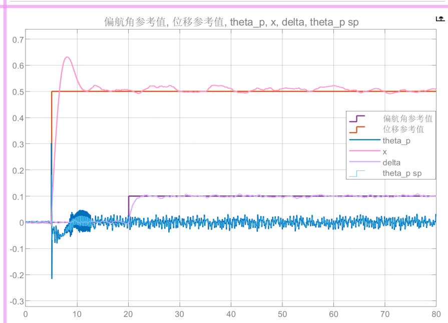
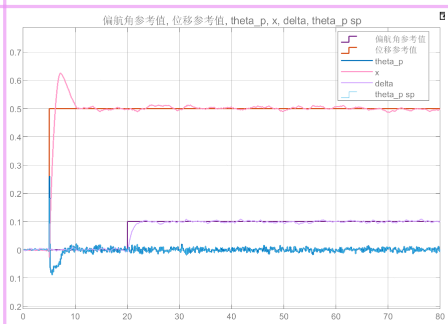
m	$l = 0.5 \times 0.0903$	$l = 0.5 \times 0.2$	$l = 0.5 \times 0.4$
0.035			
0.05			
0.3			

Figure 8: 模糊+神经网络 PID，参数大幅度变动也能控制

可见上图中，当m参数变化 8 倍以上，而l参数变化 4 倍以上时，仍然能够控制，并不需要重新修改任何控制器参数，这体现出神经网络 PID 的优越性。

与普通 *PID* 对比

可以使用 balance_car.slx 中的普通 PID 控制进行对照，在这两个参数变动时控制失败：

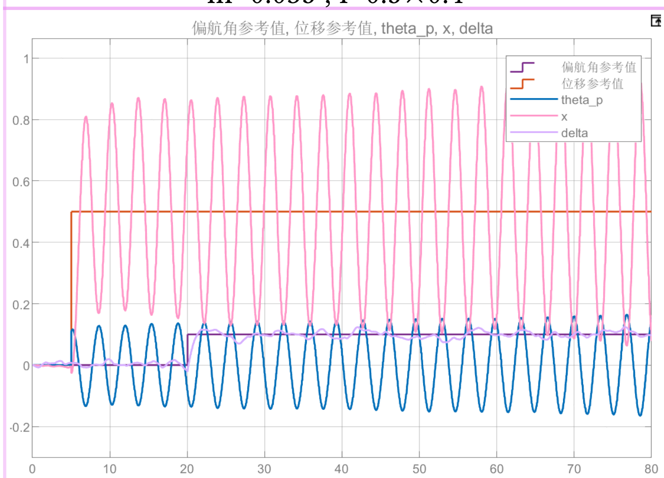
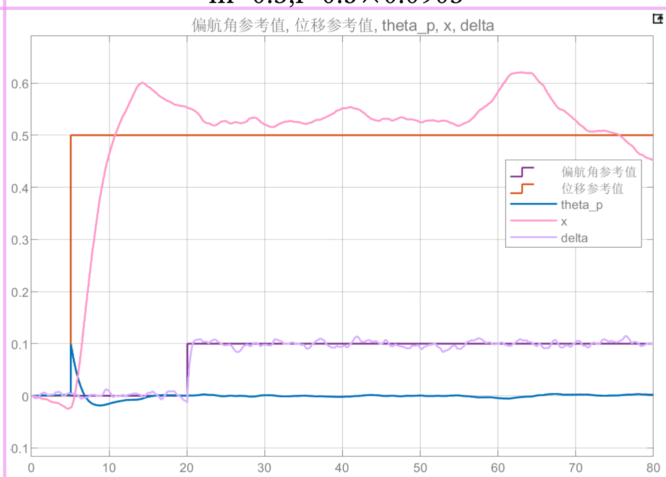
m=0.035 , l=0.5×0.4	m=0.3,l=0.5×0.0903
	

Table 10: 普通 PID，参数变动时控制失败

与仅用模糊控制对比

神经网络 PID+模糊控制器，与仅仅添加了模糊控制的模型对比，也是具有优越性：运行 new_param.m 后，运行 balance_car_with_fuzzy_logic.slx 文件，运行效果十分振荡，而运行 balance_with_fuzzy_and_NN.slx 中的神经网络 PID+模糊的控制效果仍然不错：

```
%% 以下曲线使用参数如下
m = 0.022; r = 0.07/2; I = 0.5*m*r^2; M = 0.757-2*m; l = 0.5*0.8; J_p =
(1/12)*M*(0.3^2+0.08^2); d = 0.1612; J_delta = (1/12)*M*(0.0930^2+0.0530^2);
```

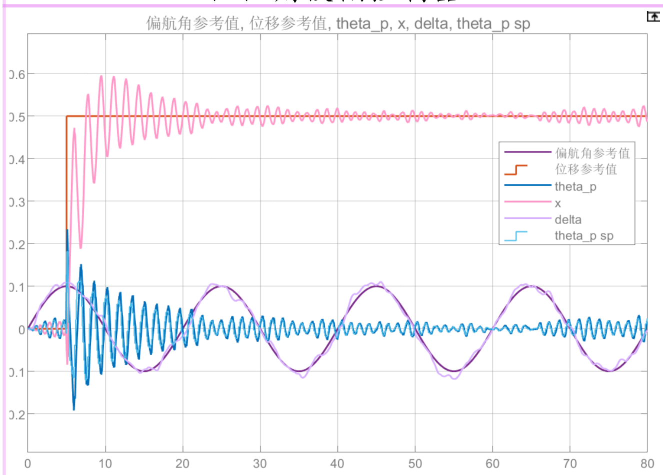
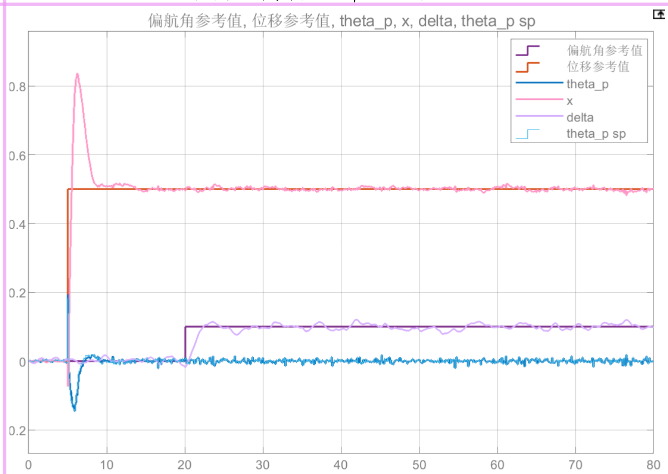
<table>
<tr><th>仅仅用模糊控制器</th><th>模糊控制器+神经网络 PID</th></tr>
<tr><td>
</td><td>
</td></tr>
</table>

Table 11: 仅用模糊控制，参数变动时控制失败

总结

本项目的对象是平衡车，2 输入 3 输出，首先借鉴了串级控制的思路搭建控制框架，然后用普通 PID 进行初步控制，效果作为 baseline。

引入模糊控制改进控制效果，使得跟踪更加平滑，效果明显优于 baseline。但是，模型参数变动后，仍然需要重新调参（因为副回路的普通 PID 控制器仍然依赖于模型参数信息），因此使用神经网络 PID 弥补这一点，使得控制更 robust。

对比“神经网络 PID 搭配两个普通 PID”与“三个普通 PID”之间的差异，发现在原参数下，两者效果接近（因为原本 PID 已经调到较优秀的参数了），但是当模型参数变动后，普通 PID 就失控了，然而神经网络 PID 仍然有较好的控制效果。

最终，将两个模糊控制器与一个神经网络 PID 一道放入回路，完全取代了原本的 PID，控制效果与 robustness 均优于普通 PID。又和之前仅用模糊控制的方法对比，发现该最终方案更 robust。

代码附件与运行说明

> 使用的 Matlab 版本为 2023b

附件含有：

> balance_car.slx 仿真模型，使用三个普通 PID 控制
> balance_car_with_fuzzy_logic.slx 仿真模型，使用两个模糊控制器+一个普通 PID 控制
> balance_car_with_NN.slx 仿真模型，使用一个神经网络 PID 控制器+两个普通 PID 控制
> balance_with_fuzzy_and_NN.slx 最终仿真模型，使用两个模糊控制器+一个神经网络 PID 控制

> parameter.m 模型参数（来源于真实数据）
> new_parameter.m 人为改动后的参数（为了测试 robustness）

> fuzzy_controller.fis 主回路模糊控制器
> fuzzy_delta.fis 差分转向模糊控制器
> NN_controller.m 神经网络 PID 的 level2 S-function 文件

测试时，请先运行 parameter.m 文件或者 new_parameter.m 文件（使得 workspace 中有参数的数据）然后即可运行 slx 文件。

参考资料

模型来自该处提供的实物数据：https://pan.baidu.com/e/1igIu6VU-7f1i702oJFCTrQ（但该资料的模型是线性化后的，我的项目中特意没有线性化，使用了精确的非线性方程）

如有疑问，请访问 https://github.com/Maythics/Control_simulink.git