**计算机辅助设计报告**

**三轮全向移动机器人**

**运动控制仿真**

**201103009004 余杨广**

**201103009019 沈阳**

**201103009031 陈斌**

**人员分工：**

**余杨广：总体负责，系统理解及控制器设计，PPT制作，后期报告审查及修改**

**陈斌：PPT制作，报告撰写**

**沈阳：资料收集，辅助其余两人完成任务**

目录

[**一、** **实验目的** 3](#_Toc392889598)

[**二、** **实验原理** 3](#_Toc392889599)

[**2.1** **控制对象——三轮全向机器人** 3](#_Toc392889600)

[**2.2 控制系统结构** 4](#_Toc392889601)

[**三、** **实验内容** 4](#_Toc392889602)

[**3.1电机模型** 4](#_Toc392889603)

[**3.1.1物理建模** 4](#_Toc392889604)

[**3.1.2 Simulink模块搭建** 5](#_Toc392889605)

[**3.1.3无刷直流电机仿真模型的验证** 8](#_Toc392889606)

[**3.2运动学模型** 10](#_Toc392889607)

[**3.2.1物理建模** 10](#_Toc392889608)

[**3.2.2 Simulink模块搭建** 10](#_Toc392889609)

[**3.3 路径规划** 11](#_Toc392889610)

[**3.4. 传感器设计** 12](#_Toc392889611)

[**3.5.控制器设计** 13](#_Toc392889612)

[**3.5.1 电机控制器设计** 13](#_Toc392889613)

[**3.5.2 运动控制器设计** 15](#_Toc392889614)

[**3.6 观测器** 19](#_Toc392889615)

[**四、结果验收** 20](#_Toc392889616)

[**4.1 x轴方向的误差** 20](#_Toc392889617)

[**4.2 y轴方向的误差** 20](#_Toc392889618)

[**4.3 前进方向偏角** 21](#_Toc392889619)

[**4.4 速度误差** 21](#_Toc392889620)

[**五、致谢** 22](#_Toc392889621)

[**六、附录（路径规划函数）** 22](#_Toc392889622)

1. **实验目的**
2. 建立三轮全向机器人系统的数学模型，然后基于simulink建立该系统的仿

真模型并设计控制器，最终满足控制要求；

1. 控制的最终目的是使该机器人能够良好跟踪预期的运动轨迹；
2. 通过对复杂系统的分析、建模、仿真、验证，全面提高利用计算机对复杂系

统进行辅助设计的能力；

1. 通过集体作业、分工完成任务的方式培养团队意识，提高团队集体攻关能力
2. **实验原理**
   1. **控制对象——三轮全向机器人**

三轮全向移动机器人其驱动轮由三个全向轮组成，径向对称安装，各轮互成120°角，滚柱垂直于各主轮。三个全向轮的大小和质量完全相同，而且由性能相同的电机驱动。

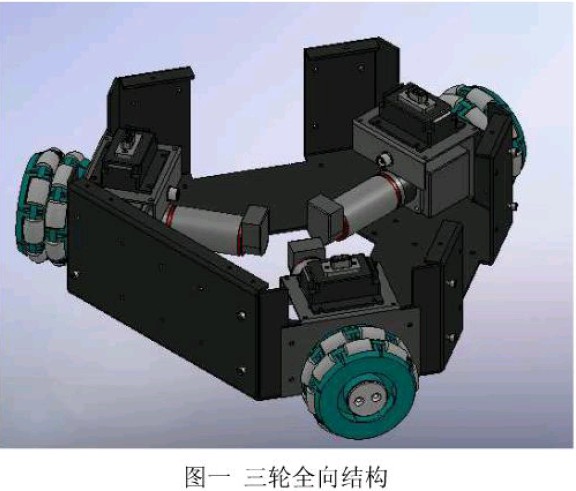


图 1 三轮全向移动机器人

**2.2 控制系统结构**

图 2基于运动学模型的分层控制框图

1. **实验内容**

**3.1电机模型**

**3.1.1物理建模**

瑞士的MAXON公司的无刷直流电机建模如下:

无刷直流电机的数学模型，其等效电路如下图所示：

根据上图，建立电机数学方程如下：

* 瞬态电压方程



* 电压方程



* 转矩方程



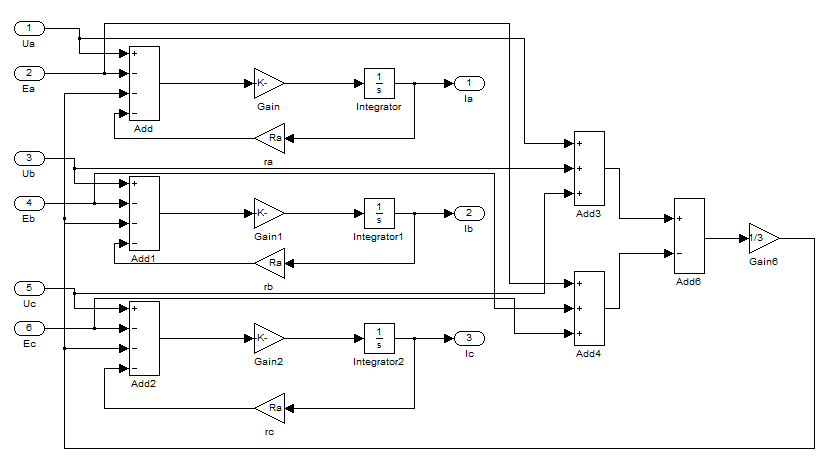
* 运动方程



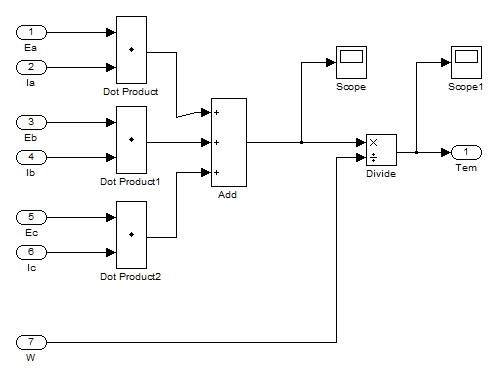
**3.1.2 Simulink模块搭建**

**根据以上数学模型，我们搭建电机的Simulink模块如下：**

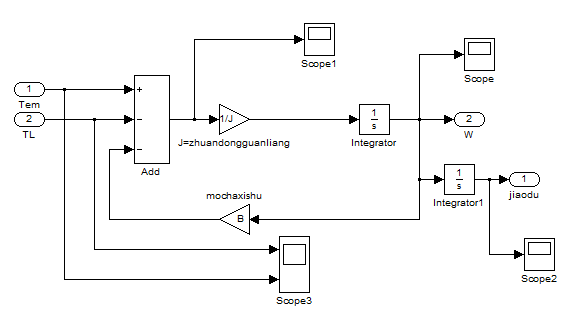
* 电压方程模块



* 转矩方程模块

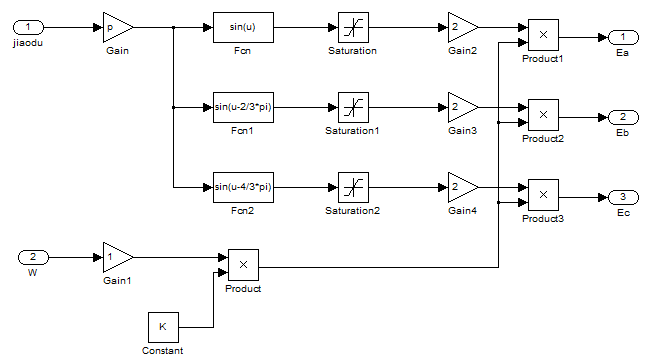


* 运动方程模块

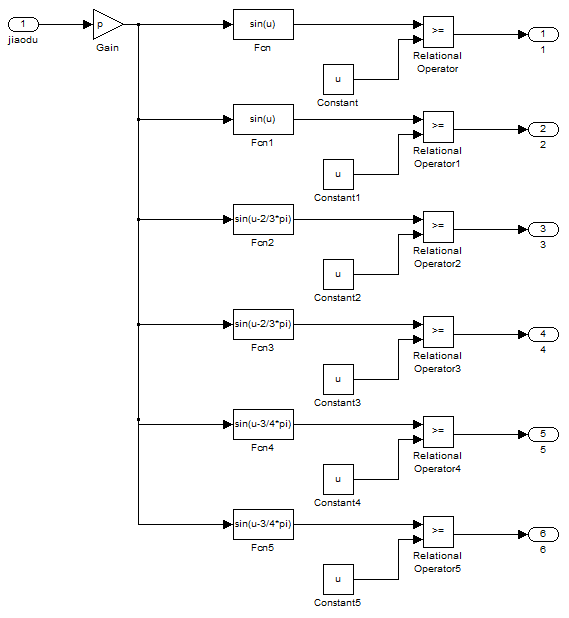


* 其他必要模块设计

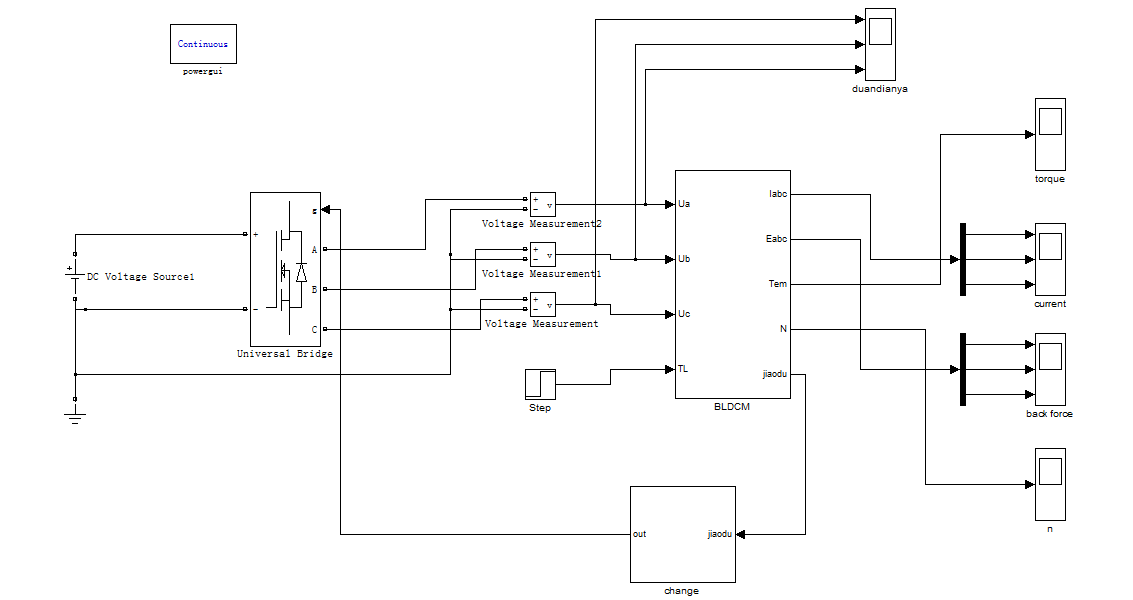
反电动势模块

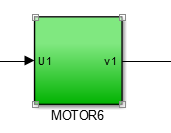


逻辑换向模块



* 模块组装——电机仿真模型

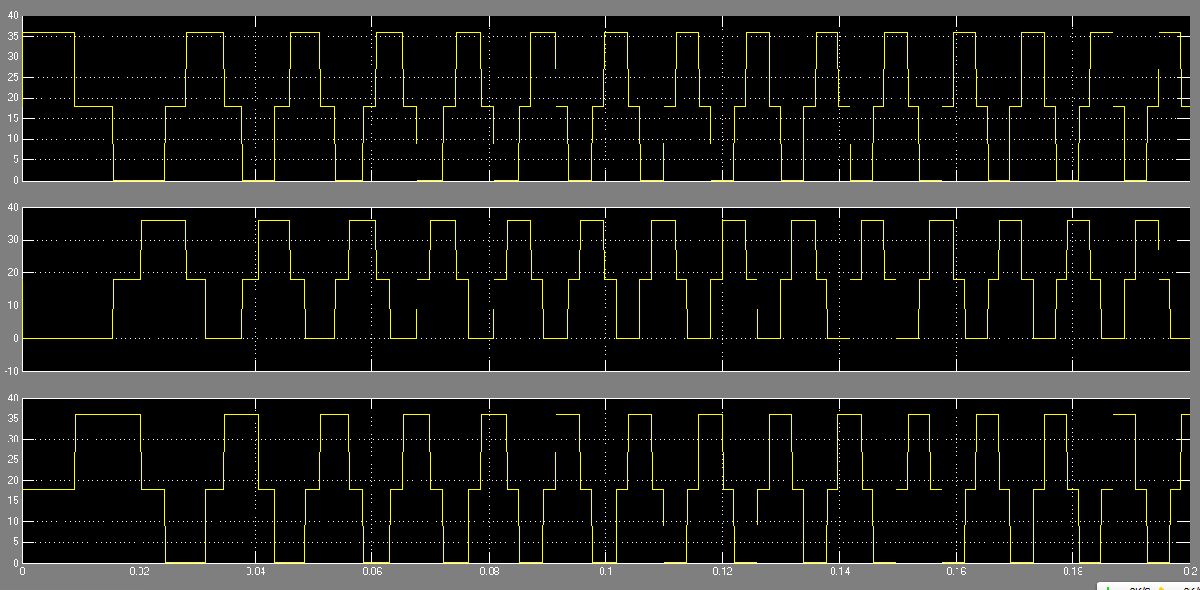


* 驱动电机模块封装

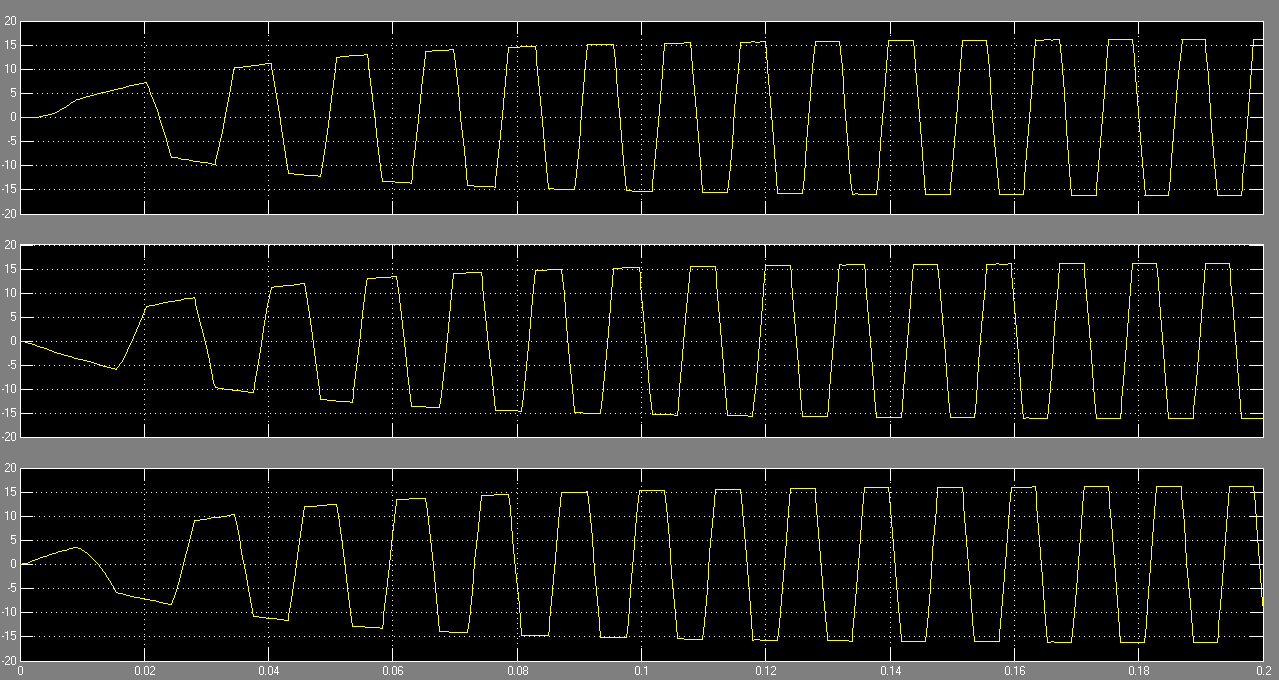
**3.1.3无刷直流电机仿真模型的验证**

到此电机的建模就算完成了，但其正确性还需要结果来验证，以下是仿真结果：

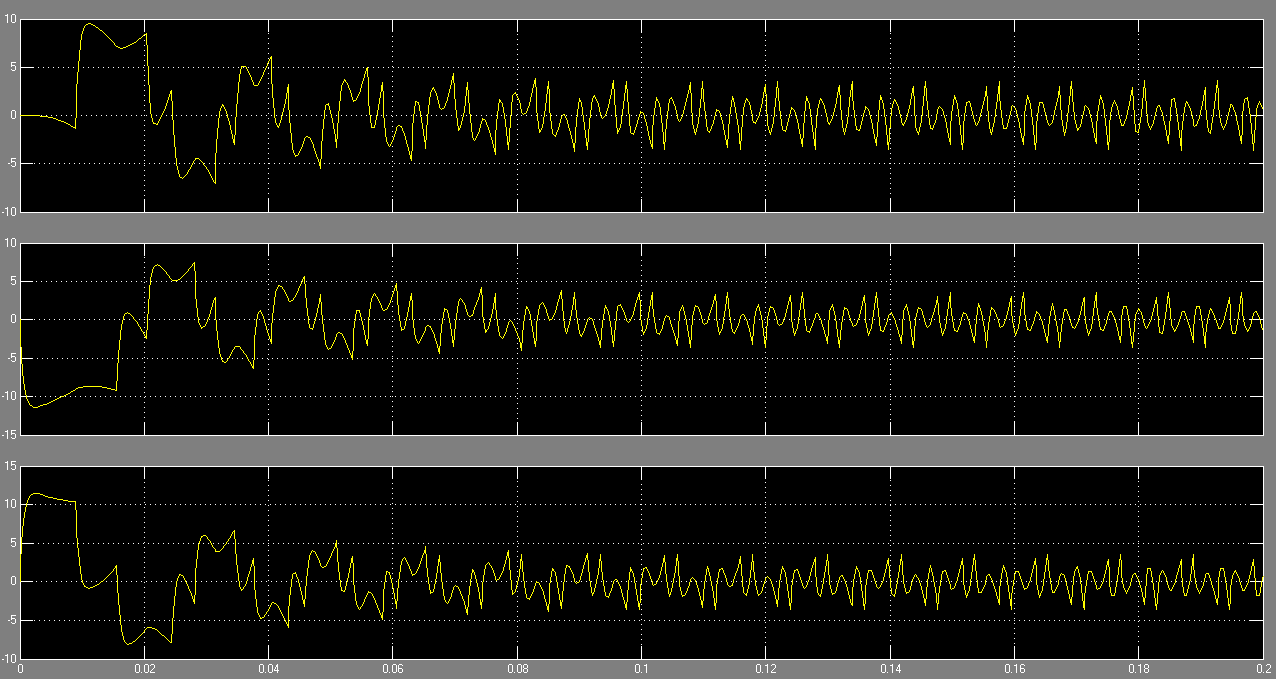
1. 绕组端电压波形：



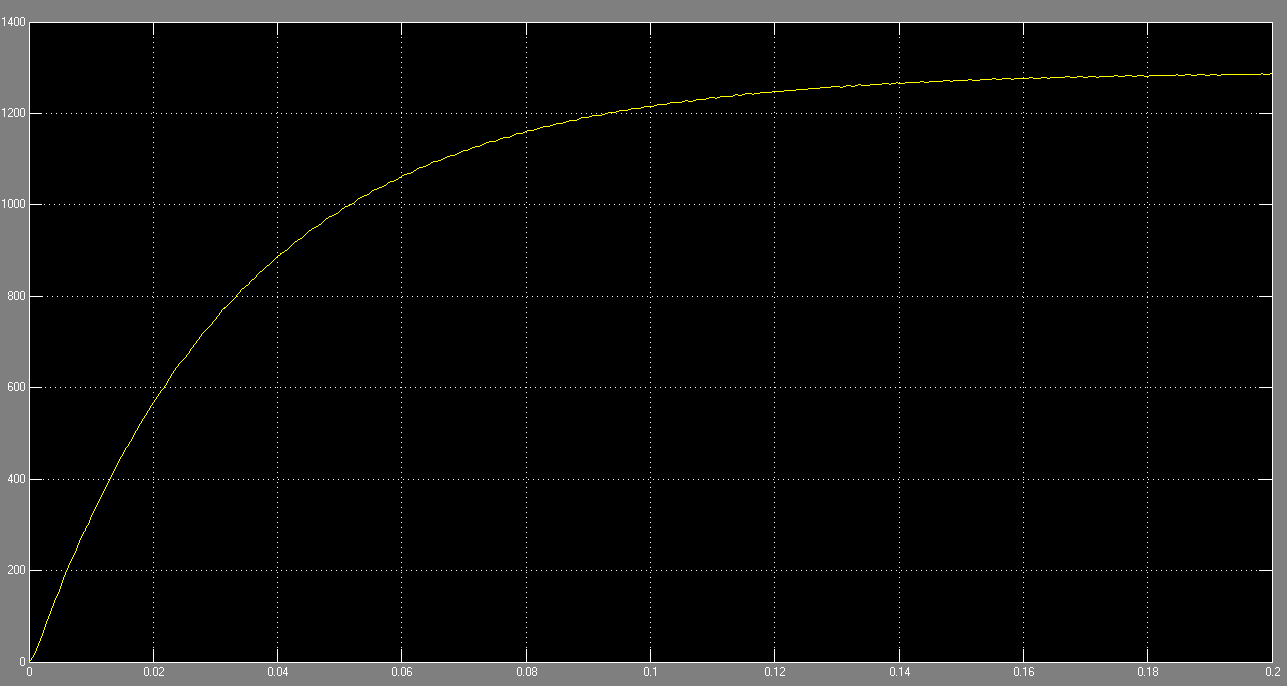
1. 反电动势波形



1. 电流波形



1. 转速波形



根据图像可知，仿真结果跟实际是相吻合的。

**3.2运动学模型**

为了实现现实世界速度与机器人三个电机转速之间的转换，我们建立起运动学模型如下：

**3.2.1物理建模**

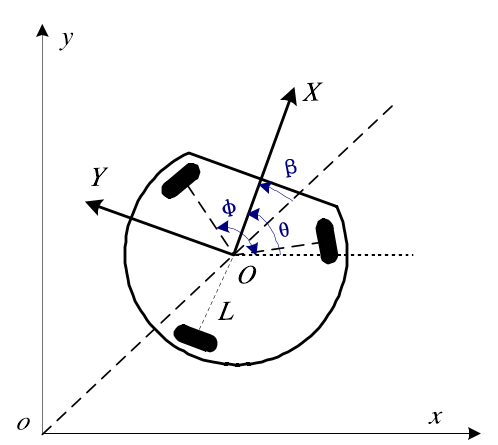
1. 建立如图所示的世界坐标系xoy和机器人坐标系XOY 。

图 3 三轮全向轮式机器人示意图

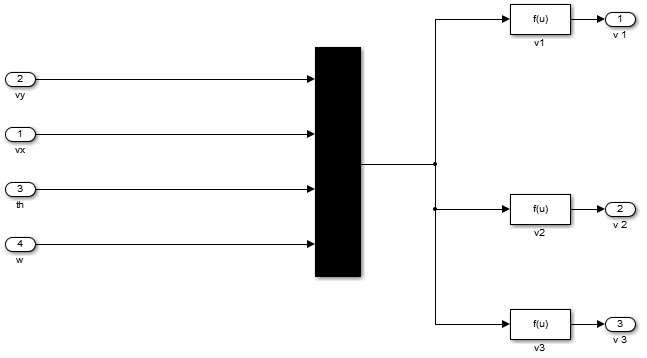
图中，θ为机器人坐标系与世界坐标系之间的夹角；φ为驱动轮间的夹角，φ =120；L为机器人中心到轮子中心的水平距离。

2. 设v1，v2，v3为全向轮线速度，vx，vy分别为机器人在XOY坐标系X轴和Y轴的速度分量；ω为机器人自转的角速度。那么，机器人在世界坐标系中的速度与驱动轮速度之间的关系为：

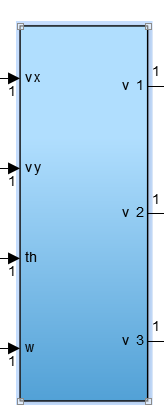


**3.2.2 Simulink模块搭建**

根据2.1中的公式，我们搭建出Simulink模块如下：

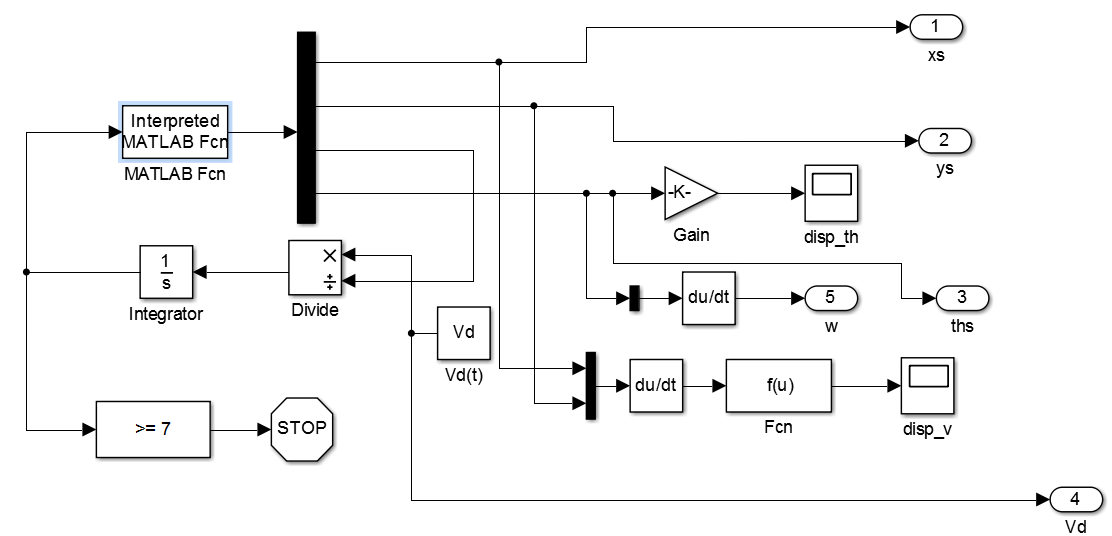


图中三个函数模块实现2.1中的矩阵转换。最终，模块封装如下：

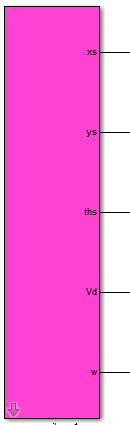


**3.3 路径规划**

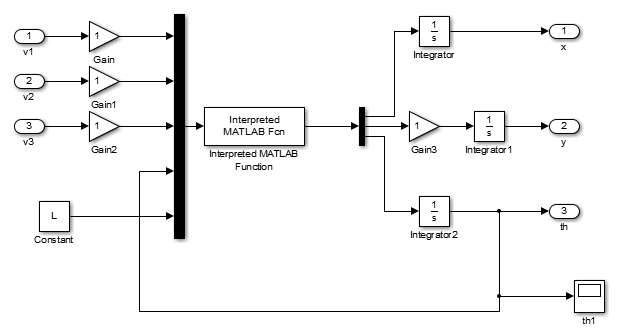
路径规划方面，我们利用B样条曲线规划方法，取了几个点作为控制点，生成了一条类似S的曲线，来检验小车的跟随情况。

**Simulink模块如下：**

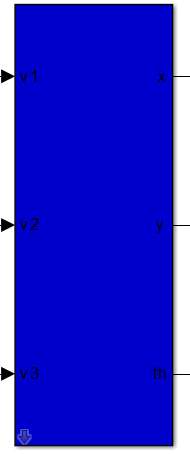
其中，MATLAB FCN为核心算法，因代码较长，放至附录部分，在此不一一赘述。模块中，Vd为待输入的期望速度值，xs为期望的x轴位置，ys为期望的y轴位置, ths为期望的小车前进偏角。

最后，曲线生成模块封装图如下图所示：

**3.4. 传感器设计**

为了实现对小车自身状态（包括自身位置信息和朝向信息）的感知，我们设计了一个传感器模块，以便引入反馈。搭建的Simulink模块如下：

其中，v1,v2,v3为三个电机的线速度，中间的函数实现电机线速度到真实世界中Vx,Vy和w的转化。最后，我们将传感器模块封装，封装图如下：

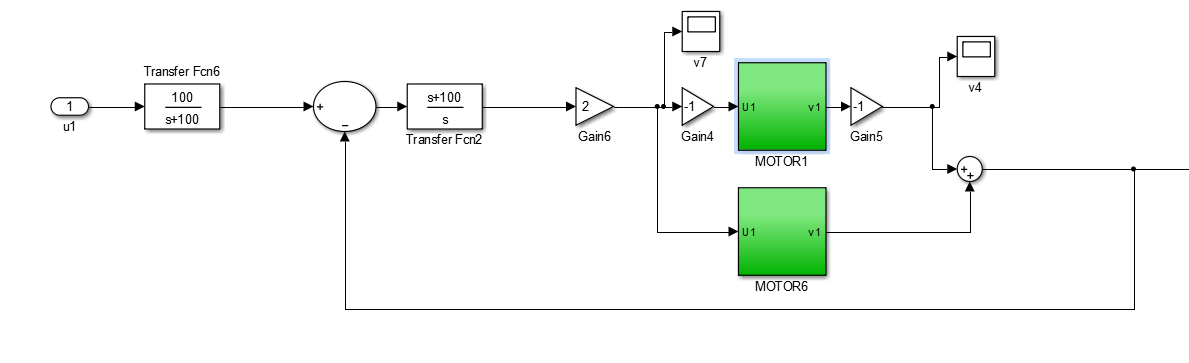


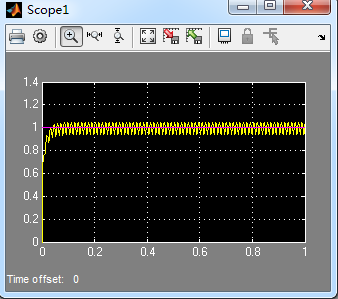
**3.5.控制器设计**

对于小车的控制，我们认为无非要解决两个问题，一个是小车是否受控听话即电机的控制问题；第二个问题是小车是否足够聪明知道该怎样走，即运动学控制问题。下面我们将就这两个方面展开论述。

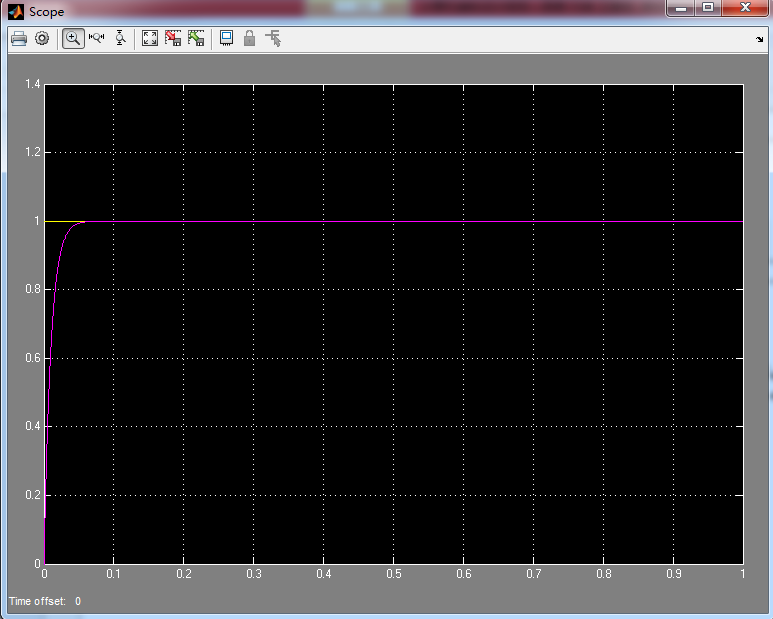
**3.5.1 电机控制器设计**

从电机模型一节中可以看出，电机模型的数学公式非常复杂，推算电压V和转速w之间的关系非常的困难，我们推算了很久也没有成功。后来我们想到，不管是交流电动机或者是直流电动机，他们的本质都是电动机。而电动机的传递函数都是一阶的，对于一阶模型我们利用P控制就能够实现很好的控制，而且也非常好调试，比例环节K的值越大越好，总能够实现较好的动态性能。最终我们出于消除稳态误差的考虑，采用了PI控制加上前置滤波的控制方式。经过调试，最终我们的PI控制器为，前置滤波器为。最终的控制simulink图如下图所示：



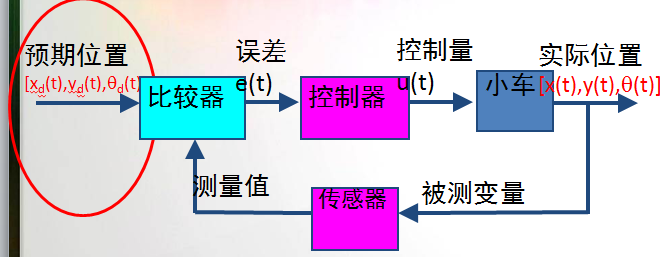
为了检验我们控制器的效果，我们特意与学长的控制器进行了对比，对比方式为控制下的电机阶跃响应性能。对比图如下图所示：

学长电机控制的阶跃响应图



我们的电机控制阶跃响应图

从图中我们可以看出，不管是从调节时间还是从稳定性、超调量来看，我们的控制器都有着绝对的优越性。

**3.5.2 运动控制器设计**

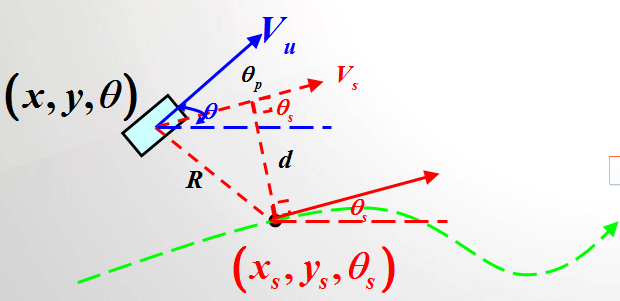
对于运动控制即位置控制，我们认为控制框图应该如下图所示：

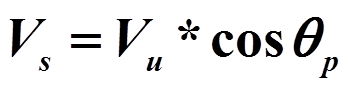
为了运动控制有更深层次的理解，我们建立如下数学模型：

一个在真实坐标戏中运动的小车模型如下图所示：

设Vu为轴向前进速度；为小车中轴线与水平线夹角；[x，y]为小车中心位置,L为车中心到小车前端的距离，Vs为速度误差，d为位置误差，则根据上图可建立以下公式：

小车在前进过程中，轨迹与预期的轨迹之间的误差如下图所示：



其中，



要达到良好的控制效果，设计上必须要确定一个控制目标。在设计中，我们要达到的目的有3个：

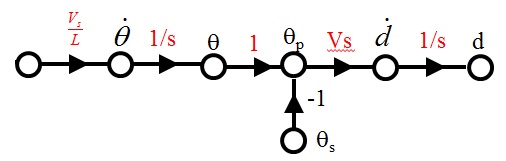
1.θp→0

2. Vs→期望速度Vd

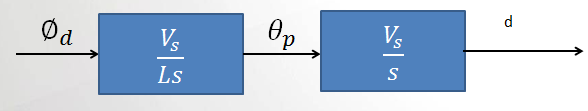
3. d→0

因为我们能够控制电机的转速，所以速度控制暂不考虑，我们只需考虑小车的期望向角。而这三个目标只要能够保证d->0,其他目标就能够实现。

根据以上公式，可以推导出到d的函数传递关系图如下：

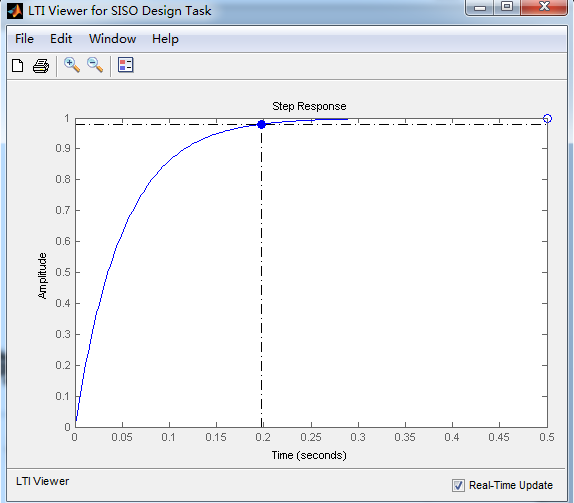


根据以上函数传递图，可以推导出以下关系图：



我们要得到的是预期输入d为0，输出也为0；我们采用内外环控制方式，引入反馈d和,因为内外环均是一阶环节，采用比例控制就能获得很好的控制效果。最终我们的内环的比例控制是100，外环的比例控制是30，因此-3000d-30；

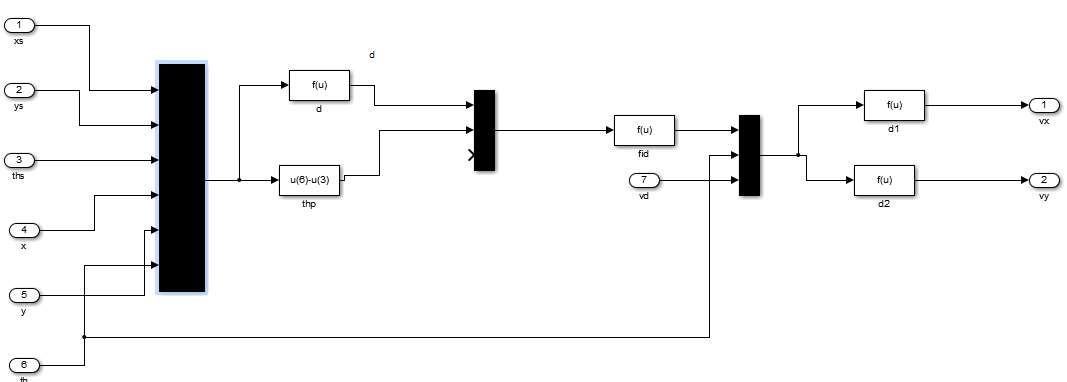
控制器的阶跃响应图如下图所示：



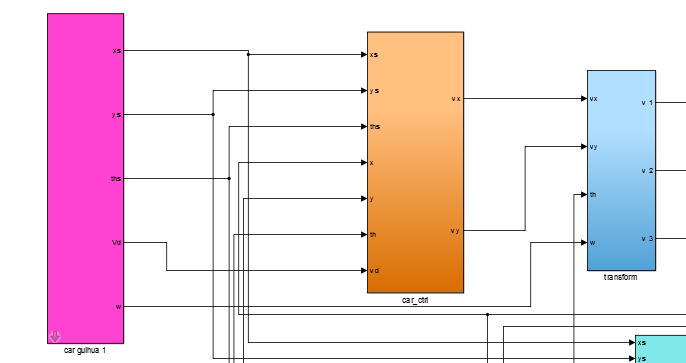
最后，得到真实世界坐标系中的预期期望速度：

最后，经过世界坐标系到机器坐标系的转换，得到预期的电机线速度。至此，整个运动学控制结束。

根据以上推导，我们搭建的Simulink模块如下图所示：

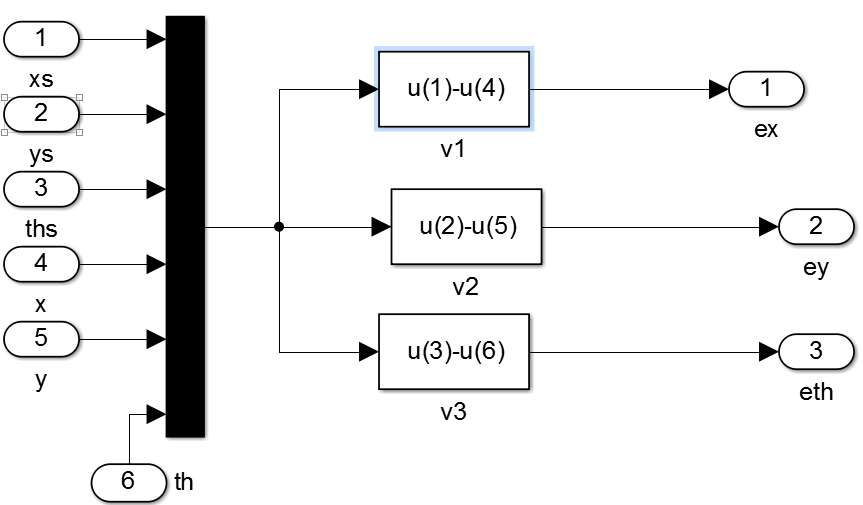


整个封装与路径规划模块、坐标转换模块的连接如下图所示：

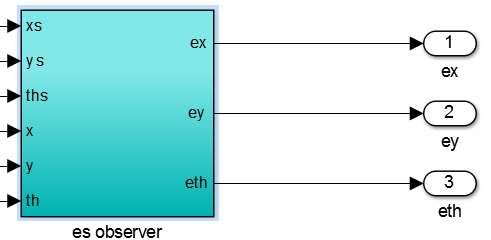


**3.6 观测器**

为了方便观察预期与实际的误差，我们设计了一个观测器模块如下：

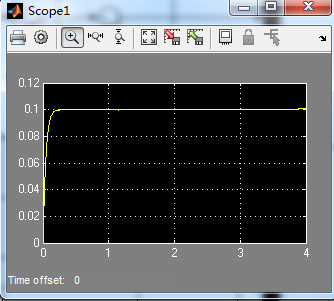


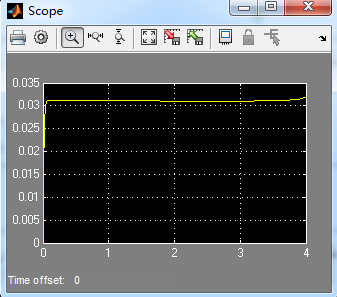
观测器封装如下：



**四、结果验收**

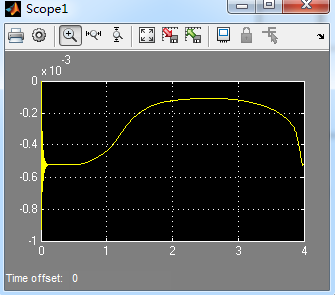
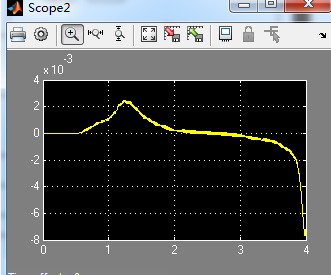
为了对我们的设计效果进行评估，我们再次利用学长做对照，比较了x轴方向的误差，y轴方向的误差，前进方向偏角，和速度误差，结果如下：

**4.1 x轴方向的误差**

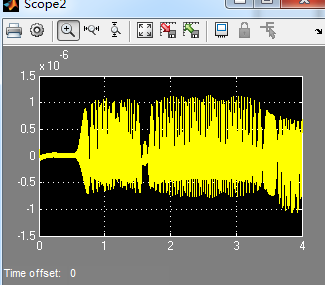
  **我们的设计**  **学长的设计**

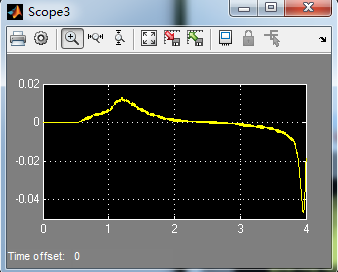
从两图对比可以看出，我们的性能比学长的提高了300%。

**4.2 y轴方向的误差**

**我们的设计**  **学长的设计**

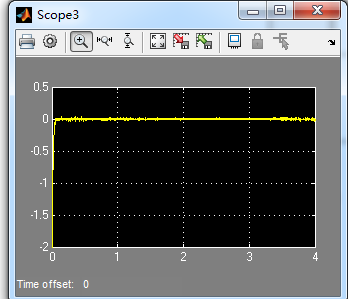
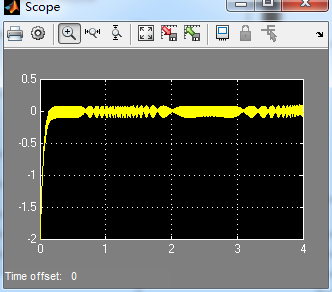
从两图对比可以看出，我们的性能比学长的提高了800%。

**4.3 前进方向偏角**

**我们的设计**  **学长的设计**

从两图对比可以看出，我们的性能比学长的提高了10000倍。

**4.4 速度误差**



**我们的设计** **学长的设计**

从两图对比可以看出，我们的设计速度保持的较好而学长的速度波动较大。

**五、致谢**

首先，非常感谢徐明老师给了我们这个机会，让我们了解了一些三轮全向机器人的基本原理，以及提高了对MATLAB运用的熟练程度。

其次，要向我们素未谋面的为我们提供了一份样板的师兄们。他们的工作为我们节约了大量的时间。虽然在文中我们用他们的结果作对比，来证明我们控制的优越性，但并不意味着我们比他们高明，因为他们才是开拓者，是他们完成了从无到有的过程，在此向他们致敬！

**六、附录（路径规划函数）**

function P=cal\_P(t)

%P=cal\_P(t)

%ÊäÈët£ºBÑùÌõÇúÏß²ÎÊýt

%Êä³öP:[x(t);y(t);V(t);thelta(t)]

%±¾³ÌÐòÓÃÓÚsimulink·ÂÕæ

%Ô­Ê¼¿ØÖÆµã

C=[0 1 2 3 4 5

0 0 0 0.5 4 5];

V1=[1;0];

V2=[1;0];

L=0.254;

NC=length(C);

%Ôö¼Ó¿ØÖÆµã£¬±£Ö¤¹ýÆðµãºÍÖÕµã

RC=[C(:,1)-V1\*L,C(:,1),C(:,1)+V1\*L,C(:,2:NC-1),C(:,NC)-V2\*L,C(:,NC),C(:,NC)+V2\*L];

N=length(RC);

OUT=0;

if(t>=N-3)

I=N-4;

OUT=1;

dt=t-(N-3);

t=1;

else

I=fix(t);

t=mod(t,1);

end

f1=(1-t).^3/6;

f2=(3\*t.^3-6\*t.^2+4)/6;

f3=(-3\*t.^3+3\*t.^2+3\*t+1)/6;

f4=t.^3/6;

d1 =-1/2\*(1-t)^2;

d2 =3/2\*t^2-2\*t;

d3 =-3/2\*t^2+t+1/2;

d4 =1/2\*t^2;

a1 = 1-t;

a2 = 3\*t-2;

a3 = -3\*t+1;

a4 = t;

P(1,1)=f1\*RC(1,I+1)+f2\*RC(1,I+2)+f3\*RC(1,I+3)+f4\*RC(1,I+4);

P(2,1)=f1\*RC(2,I+1)+f2\*RC(2,I+2)+f3\*RC(2,I+3)+f4\*RC(2,I+4);

V(1,1)=d1\*RC(1,I+1)+d2\*RC(1,I+2)+d3\*RC(1,I+3)+d4\*RC(1,I+4);

V(2,1)=d1\*RC(2,I+1)+d2\*RC(2,I+2)+d3\*RC(2,I+3)+d4\*RC(2,I+4);

A(1,1)=a1\*RC(1,I+1)+a2\*RC(1,I+2)+a3\*RC(1,I+3)+a4\*RC(1,I+4);

A(2,1)=a1\*RC(2,I+1)+a2\*RC(2,I+2)+a3\*RC(2,I+3)+a4\*RC(2,I+4);

vv=sqrt(V(1,1)^2+V(2,1)^2);

angle=atan2(V(2,1),V(1,1));

if(OUT==1)

P=P+V\*dt;

end

P=[P;vv;angle];