基于 PID 算法控制两轮驱动小车

苏州大学机电工程学院 17 级电气一班 1729401127 李海诺

【摘要】PID 算法是工程控制领域常用的一种算法，其有着技术成熟，参数整定灵活，适用性强，鲁棒性强，控制效果好等优点，得到了广泛的应用，其参数整定对控制效果影响极大。本文根据大作业文档要求，提出了用 PID 算法分两步来控制两轮驱动小车的方法，同时用 Matlab 对整个过程进行了仿真。

【关键词】PID 算法 两轮驱动小车 Matlab 仿真

第一部分 总述与分析

通过仔细研读相关要求，我们需要明确小车大致需要完成两个任务。第一是从起始点逐步移动到目标点，第二则是在移动到目标点时需要保证小车朝向预定方向。根据网上所找到的资料来看，大部分对两轮小车的控制方法并没有对小车最后的方向进行限制，即控制小车跑到目标位置即可，显然这并没有完全满足大作业的要求，我们需要对方法进行完善。

如果要完成大作业中的两个任务，我们有两个选择，第一个是将两个任务合二为一同时完成，即最后到达目标点的时侯小车朝向也完全正确；第二个是先执行小车移动到目标点的操作，控制其移动至目标点，等到了目标点之后在进行方向的对准。第一个选择看似方便，但是我对第一个方法进行仿真过后发现它存在很大的问题。小车在运动过程中的方向可以逐渐收敛于目标方向，但无法到达目标点，即小车的运动被方向限定死了，只能在平行于过目标点的一条直线上运动。虽然在仿真中恰好可以完成大作业的目标，但是这只是因为起始点和目标点的特殊关系才能完成，并不能完成大部分其它任意点到任意点的任务，不具有普适性。而第二种方法则将两个任务分开进行，相对来说可行性更高，普适性也更强，最后通过仿真也证明了第二种分开控制的方法更好，所以最后我选择用第二种方法来对小车进行控制。

本篇文章分四个部分对小车的运动及控制进行了分析、建模、设计、仿真，证明我的设计能完全满足大作业的要求，同时也发现了本方法有继续改进和提升的空间。

第二部分 运动部分建模和控制器设计

I.运动部分

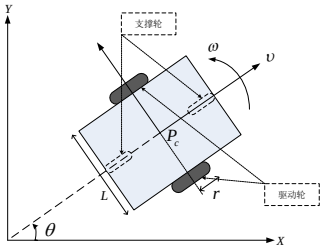

Fig.1 文档中给出的小车示意图

在大作业文档中已经明确了运动部分模型的一部分，即：

1. 全局坐标系为$X-Y$，小车前进方向与X轴夹角为θ，两轮间轴长度为L。
2. 设轴中心P_c与该小车的质心在地面的投影重合，小车实时位置可由点P_c在坐标系中的位置表示，设为列向量：

$$q = [x \quad y \quad \theta]^T \tag{1}$$

3. υ为小车前进方向的线速度，ω为偏离X轴方向的角速度，可根据左右驱动轮线速度υ_l、υ_r分别计算得到式(2)、(3)：

$$\upsilon = \frac{\upsilon_l + \upsilon_r}{2} \tag{2}$$

$$\omega = \frac{\upsilon_r - \upsilon_l}{L} \tag{3}$$

其中速度约束为$-0.5\ \mathrm{m/s} \le v_r \le 0.5\ \mathrm{m/s}$，$-0.5\ \mathrm{m/s} \le v_l \le 0.5\ \mathrm{m/s}$。

4. 因为使用了差速驱动方式，小车的左右轮间还存在速度差。而小车的左右轮线速度的不同还决定了其具备不同的三种运动状态，如 Fig.2：

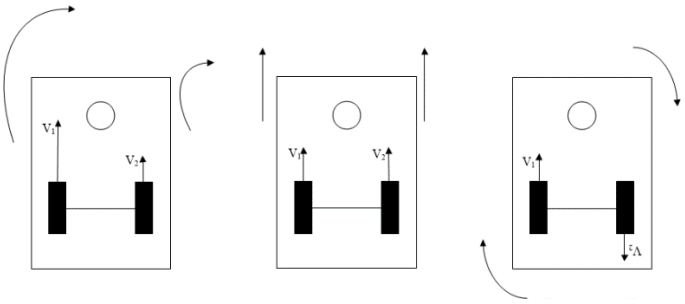

Fig.2 小车存在的三种运动状态

5. 选取控制向量为：

$$u = [v\ \ \omega]^T \tag{4}$$

6. 根据教材的有关知识，将q作为状态空间模型的状态向量，可得到系统状态方程如下：

$$\dot{q} = \begin{bmatrix} cos\,\theta & 0 \\ sin\,\theta & 0 \\ 0 & 1 \end{bmatrix} u = \begin{bmatrix} cos\,\theta & 0 \\ sin\,\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{5}$$

该式为连续时间下的非线性状态方程，即$\dot{q} = f(q,u)$。

因为计算机能处理的量只能是数字量，所以我们在实际控制一个系统的时候需要将连续时间状态方程化为离散时间状态方程。离散化即将方程(5)中的时间t变为KT，其中T为采样周期，K为整数，输入量$u(t)$只在采样时刻变化，在一个采样周期KT和$(K+1)T$之间是不变的。当采样周期T比较小时，离散化的数学过程可以近似表示为：

$$\frac{q[(K+1)T] - q[KT]}{T} = \begin{bmatrix} cos[\theta(KT)] & 0 \\ sin[\theta(KT)] & 0 \\ 0 & 1 \end{bmatrix} u(KT) \tag{6}$$

经(6)整理后可得：

$$q[(K+1)T] = q[KT] + \begin{bmatrix} cos[\theta(KT)] & 0 \\ sin[\theta(KT)] & 0 \\ 0 & 1 \end{bmatrix} u(KT) \cdot T \tag{7}$$

这样就表示出来了计算机可进行处理的系统的离散化状态方程。

II.控制器部分

在控制器设计方面，同样有两个选择，一种是构造一个势函数，形成一个人工势场，使其在控制域中仅有处于目标点处的极小值点，而控制向量则为势函数的负梯度，这样就可以使小车无论在哪个位置最后都可以运动至目标点。但是受限于我目前的知识水平以及能力，在短期内构造出合适的势函数并设计相应的控制方法比较困难，所以我就选择了第二种相对可行的用 PID 算法来对两轮小车进行控制。

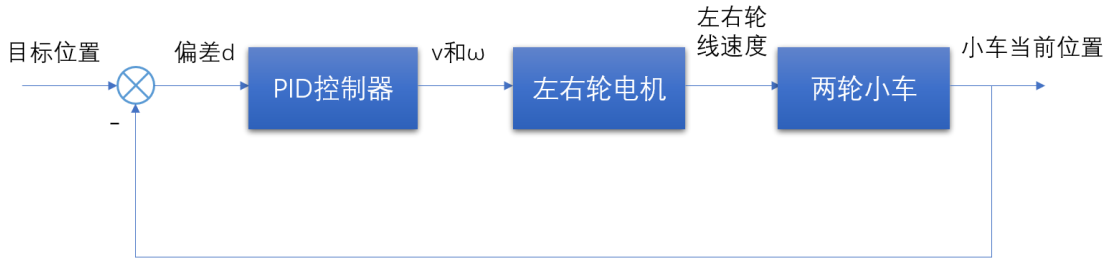

Fig.3 控制流程图

基于 PID 算法对目标对象进行控制的流程图如 Fig.3 所示，而在本次控制小车的应用中，运动控制器由两个并联的 PID 控制器组成，一个输出为线速度v，即距离决定速度。距离远速度大，距离近速度小。一个输出为角速度ω，即偏角误差决定转速。正偏左转，负偏右转；偏多转快，偏少转慢。

我们首先需要完成两个任务中的第一个任务，即先让小车到达目标点而不管小车最后的目标方向。小车的实时姿态已由列向量(1)表示，设目标位置为：

$$g = [x_g \quad y_g \quad \theta_g]^T \tag{8}$$

由(1)、(8)可得出小车与目标点的实时距离差：

$$d_{err} = \sqrt{(x_g - x)^2 + (y_g - y)^2} \tag{9}$$

以及目标偏离角（注意和后文的方向偏离角加以区别）：

$$\delta = arctan\left(\frac{y_g - y}{x_g - x}\right) - \theta = \varphi - \theta \tag{10}$$

部分参数如 Fig.4 所示。

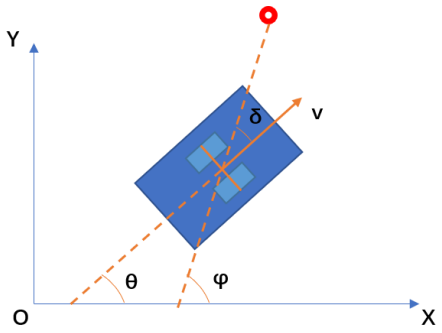

Fig.4 控制器第一部分有关参数图示

需要注意的是由控制器输出的线速度和角速度需要符合大作业中的限速要求，因此需要用当前的线速度和角速度根据式(2)、(3)解出左右轮的线速度v_l、v_r，看是否符合限速的要求，若有超过限速的情况，则将左右轮的线速度循环乘一个小于1的系数（仿真中为了效率是循环乘0.9），直到高速度的轮子低于限速要求为止。最后再用左右轮线速度反解出v和ω。

除了限速的要求，还有计算目标偏离角δ时需要注意将其归一化。即求出的δ大于π或小于等于$-\pi$时，需要将其化至$(-\pi, \pi]$。

而接下来就是对第一部分 PID 控制器进行设计并将其离散化。以下是普通连续时间的 PID 算法的式子：

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \tag{11}$$

其中偏差$e(t)$对于线速度来说是式(9)的d_{err}，对于角速度来说是式(10)的δ。

再对式(11)进行离散化，可得：

$$u(KT) = K_p e(KT) + K_i \sum_{i=0}^{k} e(i)T + K_d \frac{e(KT) - e[(K-1)T]}{T} \tag{12}$$

将式(12)代入式(7)即可得到第一部分完整的离散化状态方程。

第二个任务的完成相对第一个任务来说要简单许多。因为此时小车已经运动至目标地点，随后只需在原地旋转至目标方向即可。我们可以把此时小车的运动线速度v设为0，只对角速度ω进行控制。

我们依旧可以采用第一个任务中用来控制角度的 PID 控制器，但注意$e(t)$不再是目标偏离角而是方向偏离角，即目标方向和小车方向的夹角：

$$\delta_g = \theta_g - \theta \tag{13}$$

示意图如 Fig.5 所示。

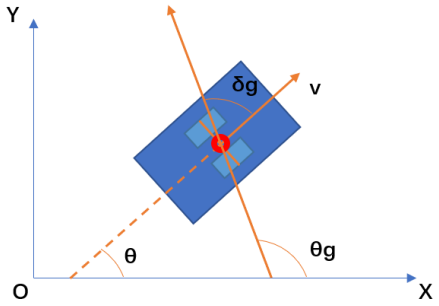

Fig.5 控制器第一部分有关参数图示

在第一项任务中关于角度及角速度有关参数的注意事项以及左右轮电机的限速在第二项任务中也同样需要注意，在这里就不多赘述了。

第三部分 使用 Matlab 进行仿真

在进行仿真之前，我们需要一些明确相关参数。首先是小车的起始点、起始方向以及目标点与目标方向，作业要求中已经明确起始位置为$[-2 \quad -2 \quad -\pi/4]^T$，目标位置为$[0 \quad 0 \quad \pi/4]^T$。其次是左右轮的限速为$-0.5\ \mathrm{m/s} \leq v_r, v_l \leq 0.5\ \mathrm{m/s}$。由于题目中并未给出轮间距$L$，所以在仿真时设轮间距$L$为$0.2\mathrm{m}$。

除了空间参数以外，控制器的参数也非常重要。先设置时间步长为$0.1s$，再对控制线速度和角速度的 PID 控制器进行调整。首先确定两个控制器的K_p参数，用来控制线速度的K_p不可太大也不可太小，太大了会绕远路，使走的路径过长；太小了会使运动速度变慢，到目标点的时间变长。用来控制角速度的K_p不能太小，可以调的相对大一些，这样转向速度会比较快。其次确定K_i参数。由于我直接将控制器离散化得到的式子是位置型的离散化 PID 公式，会有积分饱和现象，所以会导致累计误差项非常大，如果用 PID 增量算法则不会有这样的问题。鉴于此状况，我决定直接将累计误差项去除，以保证控制器在整个区域内的性能而不是单单只满足大作业的要求。最后再对参数K_d进行调整，K_d的参数相对来说不能太大，如果太大话会增大系统的阻尼，使最后阶段的收敛速度变慢。经过一段时间的调整，最终用来控制线速度的 PID 参数为$K_p = 0.3, K_i = 0, K_d = 0.2$，控制角速度的 PID 参数为$K_p = 2, K_i = 0, K_d = 0.1$。

明确相关数据后，经过仿真可得到最后的结果。由于计算机处理的局限性，我们无法准确的停在目标点，只能规定一个精度。在仿真中规定了任务一中车与目标点距离不超过$0.01m$，任务二中车的方向与目标方向偏离不超过 0.001rad，精度相对来说较高，但是时间相对来说也会长一些。

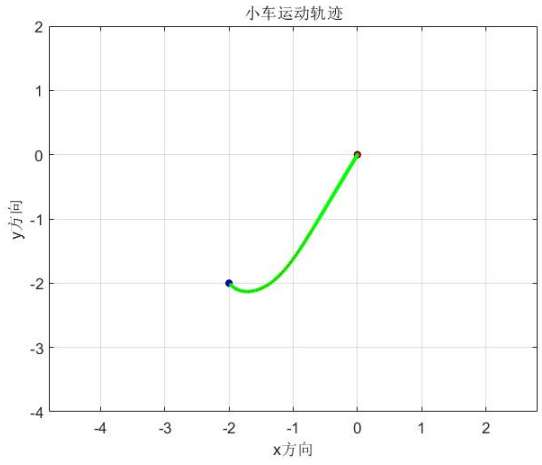

Fig.6 小车移动轨迹

小车的移动轨迹如 Fig.6 所示。蓝点为出发点，红点为目标点，轨迹由离散的点以及点与点之间的连线组成。Fig.7a 和 Fig.7b 分别通过离散点展示了小车的两部分的行进速度和角速度，其中可明显看出通过 PID 控制使偏差趋于零的过程，大作业文档中要求的限速也间接体现了出来。

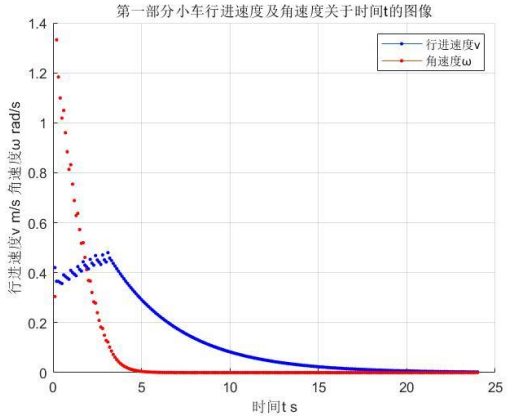

Fig.7a 第一部分 v 与ω关于 t 的图像

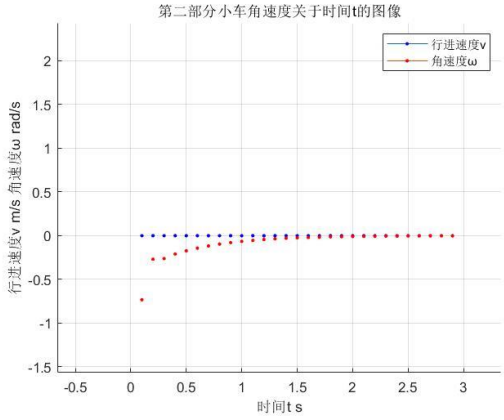

Fig.7b 第二部分 v 与ω关于 t 的图像

Fig.8a 体现出了小车的朝向，可以看出经过控制分成了两部分，并且经过第二部分控制后最后小车与 X 轴之间的夹角为0.78611rad，即$\pi/4$。

Fig.8b 则体现出了小车左右轮的线速度。从图中可以发现明显的限速迹象，而且在第一部分和第二部分的控制过程中，左右轮的线速度都被有效地限制在了范围内。

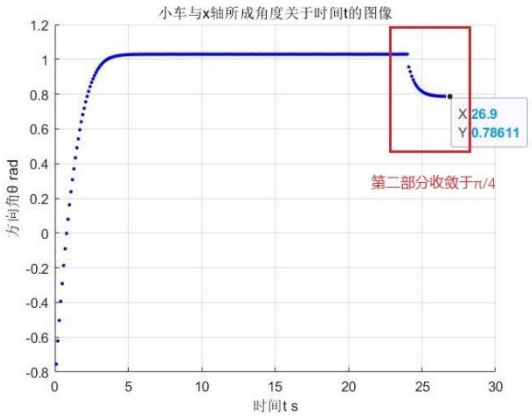

Fig.8a θ关于 t 的图像

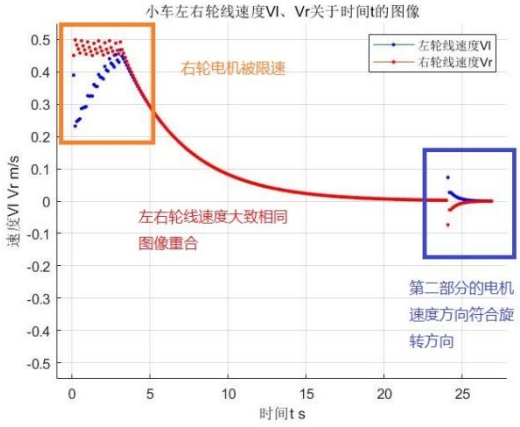

Fig.8b Vl、Vr 关于 t 的图像

Fig.9a 与 Fig.9b 则证明了此方法的普适性，即可以让小车从任意点到任意点并朝向任意目标方向。

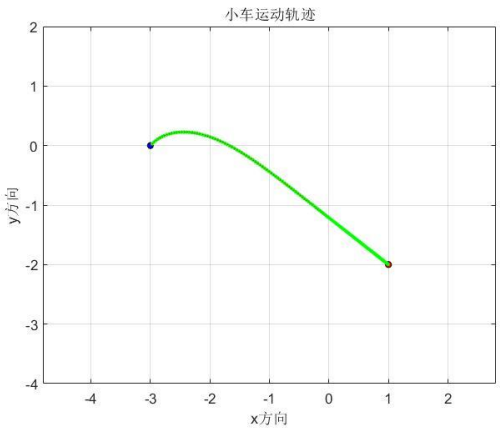

Fig.9a 更换目标位置之后的运动轨迹

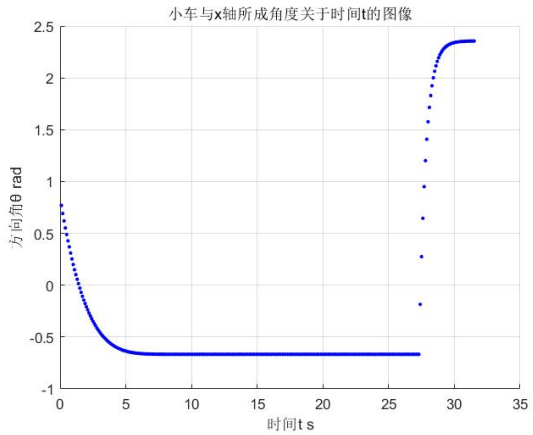

Fig.9b 更换目标位置之后的方向角θ

第四部分 结论

本文设计的基于 PID 算法控制的两轮小车通过将任务分为两个部分来完成相应的要求，较好地实现了大作业的目标，控制器的性能也很优良，整个系统也具有一定的普适性，可以在多种环境下运作。但任务的完成并非说明本文提出的方法一定是最好的，此种方法依然有提升的空间，比如可使用增量型离散 PID 算法代替位置型离散 PID 算法，使积分环节不再出现积分饱和的现象。其他方面也有很多可提高之处，还有待我一一探索。

附录 Matlab 程序

```
%两轮小车程序仿真
clc;
clear;
tic;

g=[0,0];%目标点
g_theta=pi/4;%目标方向

car=[-2,-2];%小车起始点
car_theta=-pi/4;%小车起始方向

L=0.2;%设小车轮间距为 0.2m

Kp=0.3;%小车运动速度参数
Ki=0;
Kd=0.2;

Kpo=2;%小车转动角速度参数
Kio=0;
Kdo=0.1;

tstep=0.1;%时间步长
t=0;%时间累积

qk=[car(1)
    car(2)
    car_theta];%小车的当前位置及朝向
accerrd=0;%位置累计误差
accerrt=0;%目标偏离角累计误差
v=0;%小车速度
omega=0;%小车转动速度
carp=[qk(1),qk(2)];%小车当前位置
delta_dis=norm(carp-g);%距终点距离
delta_theta_offset=atan((g(2)-qk(2))/(g(1)-qk(1)))-qk(3);%目标偏离角
deltadisp=0;%上一点的移动距离
deltatp=0;%上一点的转动角度

i=0;

figure(1);%画起始点
r=0.05;
alpha=0:pi/50:2*pi;
x=qk(1)+r*cos(alpha);
y=qk(2)+r*sin(alpha);
plot(x,y,''-'');
fill(x,y,''b'');
grid on;
```

```
hold on;

alpha=0:pi/50:2*pi;%画结束点
x=g(1)+r*cos(alpha);
y=g(2)+r*sin(alpha);
plot(x,y,''-'');
fill(x,y,''r'');
hold on;
x1=xlabel('x 方向');
x2=ylabel('y 方向');
title('小车运动轨迹');

%第一部分：从起始点运动至目标点
while delta_dis>0.01

    t=t+tstep;
    i=i+1;
    vot(i,1)=t;%存入时间点
    ptt(i,1)=t;

    axis([-4 2 -4 2]);%在更换目标点时需改变显示区域以得到较好的显示效果
    axis equal;

    carp=[qk(1),qk(2)];%小车当前坐标
    delta_dis=norm(carp-g);%距终点距离
    accerrd=accerrd+delta_dis;

    if g(1)>qk(1)%目标偏离角需根据目标位置和与小车当前位置所形成角度分情况讨论
        delta_theta_offset=atan((g(2)-qk(2))/(g(1)-qk(1)))-qk(3);
    elseif g(2)>qk(2)
            delta_theta_offset=atan((g(2)-qk(2))/(g(1)-qk(1)))+pi-qk(3);
    else
        delta_theta_offset=atan((g(2)-qk(2))/(g(1)-qk(1)))-pi-qk(3);
    end
    accerrt=accerrt+delta_theta_offset;

    while delta_theta_offset>pi || delta_theta_offset<=-pi %对目标偏移角进行处理，使其落在(-pi,pi]
    if delta_theta_offset>pi
        delta_theta_offset=delta_theta_offset-2*pi;
    end
    if delta_theta_offset<=-pi
        delta_theta_offset=delta_theta_offset+2*pi;
    end
    end

    v=Kp*delta_dis+Ki*tstep*accerrd+Kd*(delta_dis-deltadisp)/tstep;%PID 调整速度

    omega=Kpo*delta_theta_offset+Kio*tstep*accerrt+Kdo*(delta_theta_offset-deltatp)/tstep;%PID 调整角度
```

```
    Vr=(2*v+L*omega)/2;%通过运动方程解出左轮和右轮的线速度
    Vl=(2*v-L*omega)/2;

    while Vr>0.5 || Vl>0.5 || Vr<-0.5 || Vl<-0.5 %Vr Vl 限幅
        Vr=Vr*0.9;
        Vl=Vl*0.9;
    end

    v=(Vr+Vl)/2;%反解出小车的速度与角速度
    omega=(Vr-Vl)/L;

    vot(i,3)=omega;%存入当前时间点小车运动角速度
    vot(i,2)=v;%存入当前时间点小车运动速度

    deltadisp=delta_dis;%保存当前的距目标点距离以及偏移角，为下一次循环使用准备
    deltatp=delta_theta_offset;

    qkp=qk;
    qk=qk+[v*tstep*cos(qk(3)) %更新当前位置
           v*tstep*sin(qk(3))
           omega*tstep];

    figure(1);
    plot([qkp(1),qk(1)],[qkp(2),qk(2)],''-r','LineWidth',0.5); %画路径
    hold on;
    plot(qk(1),qk(2),''o','MarkerEdgeColor','g',...%画每个采样点
                                'MarkerFaceColor','g',...
                                'MarkerSize',2);
    hold on;

    ptt(i,2)=qk(3);
    ptt(i,3)=Vl;
    ptt(i,4)=Vr;

end

figure(2);%画第一部分小车行进速度及角速度关于时间 t 的图像
grid on;
x1=xlabel('时间 t s');
x2=ylabel('行进速度 v m/s  角速度 ω rad/s');
title('第一部分小车行进速度及角速度关于时间 t 的图像');
hold on;

for j=1:i
    plot(vot(j,1),vot(j,2),''-o','MarkerEdgeColor','b','MarkerFaceColor','b',''MarkerSize'',2);
    plot(vot(j,1),vot(j,3),''-o','MarkerEdgeColor','r','MarkerFaceColor','r',''MarkerSize'',2);
    hold on;
end
```

```
legend('行进速度 v','角速度 ω');

figure(4);%画小车与 x 轴所成角度时间 t 的图像
grid on;
hold on;
for j=1:i
    plot(ptt(j,1),ptt(j,2),''-o'','MarkerEdgeColor','b','MarkerFaceColor','b',''MarkerSize'',2);
    hold on;
end

figure(5);;%画小车左右轮线速度 Vl、Vr 关于时间 t 的图像
grid on;
hold on;
for j=1:i
    plot(ptt(j,1),ptt(j,3),''-o'','MarkerEdgeColor','b','MarkerFaceColor','b',''MarkerSize'',2);
    plot(ptt(j,1),ptt(j,4),''-o'','MarkerEdgeColor','r','MarkerFaceColor','r',''MarkerSize'',2);
    hold on;
end

%第二部分，将小车原地旋转至目标方向
dire_offset=g_theta-qk(3);%计算方向偏离角
tp=t;
n=i;
v=0;%到目标点后设小车的速度为 0
t=0;
i=0;

accerrdo=0;%方向偏离角累计误差

while abs(dire_offset)>0.001

    t=t+tstep;
    i=i+1;
    vo2t(i,1)=t;%存入时间点
    ptt(i+n,1)=tp+t;

    dire_offset=g_theta-qk(3);%计算方向偏离角

    while dire_offset>pi || dire_offset<=-pi %对偏移角进行处理，使其落在(-pi,pi]
    if dire_offset>pi
        dire_offset=dire_offset-2*pi;
    end
    if dire_offset<=-pi
        dire_offset=dire_offset+2*pi;
    end
    end

    omega=Kpo*dire_offset+Kio*tstep*accerrdo+Kdo*(dire_offset-deltatp)/tstep;
    deltatp=dire_offset;
```

```
    Vr=(2*v+L*omega)/2;%通过运动方程反解出左轮和右轮的线速度
    Vl=(2*v-L*omega)/2;

    while Vr>0.5 || Vl>0.5 || Vr<-0.5 || Vl<-0.5 %Vr Vl 限幅
        Vr=Vr*0.9;
        Vl=Vl*0.9;
    end

    v=(Vr+Vl)/2;
    omega=(Vr-Vl)/L;

    vo2t(i,2)=v;
    vo2t(i,3)=omega;

    qkp=qk;
    qk=qk+[v*tstep*cos(qk(3)) %更新当前位置
           v*tstep*sin(qk(3))
           omega*tstep];

    ptt(i+n,2)=qk(3);
    ptt(i+n,3)=Vl;
    ptt(i+n,4)=Vr;

end

figure(3);%画小车第二部分角速度关于时间 t 的图像
grid on;
x1=xlabel('时间 t s');
x2=ylabel('行进速度 v m/s  角速度 ω rad/s');
title('第二部分小车角速度关于时间 t 的图像');
axis([-0.5 3.5 -0.5 3.5]);%在更换目标点时需改变显示区域以得到较好的显示效果
hold on;
for j=1:i
    plot(vo2t(j,1),vo2t(j,2),''-o'','MarkerEdgeColor','b','MarkerFaceColor','b',''MarkerSize'',2);
    plot(vo2t(j,1),vo2t(j,3),''-o'','MarkerEdgeColor','r','MarkerFaceColor','r',''MarkerSize'',2);
    hold on;
end

legend('行进速度 v','角速度 ω');

figure(4);%画小车与 x 轴所成角度关于时间 t 的图像
grid on;
x1=xlabel('时间 t s');
x2=ylabel('方向角 θ rad');
title('小车与 x 轴所成角度关于时间 t 的图像');
hold on;
for j=n+1:n+i
    plot(ptt(j,1),ptt(j,2),''-o'','MarkerEdgeColor','b','MarkerFaceColor','b',''MarkerSize'',2);
```

```
    hold on;
end

figure(5);%画小车左右轮线速度 Vl、Vr 关于时间 t 的图像
grid on;
x1=xlabel('时间 t s');
x2=ylabel('速度 Vl Vr m/s');
title('小车左右轮线速度 Vl、Vr 关于时间 t 的图像');
hold on;
for j=n+1:n+i
    plot(ptt(j,1),ptt(j,3),''-o'','MarkerEdgeColor','b','MarkerFaceColor','b',''MarkerSize'',2);
    plot(ptt(j,1),ptt(j,4),''-o'','MarkerEdgeColor','r','MarkerFaceColor','r',''MarkerSize'',2);
    hold on;
end
axis([-1 ptt(n+i,1)+1 -0.55 0.55]);
legend('左轮线速度 Vl','右轮线速度 Vr');
```

参考文献：

【1】 现代控制理论的案例教学探讨 高瑜 苏州大学机电工程学院 2016

【2】 两轮差速移动机器人运动分析、建模和控制 https://blog.csdn.net/iProphet/article/details/83661753 2018

【3】 PID 算法完全讲解 https://www.arduino.cn/thread-12813-1-1.html 2015

【4】 离散 PID、PID 参数整定 https://wenku.baidu.com/view/93c83e11866fb84ae45c8dc1.html 2020

【5】 现代控制理论（第三版） 刘豹 唐万生 2015