

【1】PID离散化

连续形式PID：

out(t) = Kp * error(t) + Ki * ∫0^t error(t)dt + Kd * derror(t)/dt

离散形式PID：

out(k) = Kp * error(k) + Ki * T * Σ_{j=0}^k error(j) + Kd * (error(k) - error(k - 1))/T

若将T并入Ki和Kd，则：

out(k) = Kp * error(k) + Ki * Σ_{j=0}^k error(j) + Kd * (error(k) - error(k - 1))

位置式PID与增量式PID公式

位置式PID：

out(k) = Kp * error(k) + Ki * Σ_{j=0}^k error(j) + Kd * (error(k) - error(k - 1))

当k = k - 1时：

out(k - 1) = Kp * error(k - 1) + Ki * Σ_{j=0}^{k-1} error(j) + Kd * (error(k - 1) - error(k - 2))

两式相减，得到增量式PID：

Δout(k) = Kp * (error(k) - error(k - 1)) + Ki * error(k) + Kd * (error(k) - 2error(k - 1) + error(k - 2))

而位置式PID，输出限幅和积分限幅得分开进行

【2】PID在函数中的实现

PID程序实现

确定一个调控周期T，每隔时间T，程序执行一次PID调控

```
1 #include "stm32f10x.h" // Device header
2 #include "Delay.h"
3 #include "Timer.h"
4 int main(void)
5 {
6     while (1)
7     {
8         //在此处执行PID控制
9         Delay_ms(1000); //延时时间1s
10    }
11 }
```

```
1 #include "stm32f10x.h" // Device header
2 #include "Delay.h"
3 #include "Timer.h"
4 int main(void)
5 {
6     Timer_Init();
7     while (1)
8     {
9         //每隔时间T，程序执行到这里一次
10        //在此处执行PID控制
11        TIM2_ITStatus(TIM2, TIM_IT_Update) == SET;
12        //每隔时间T，程序执行到这里一次
13        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
14    }
15 }
```

```
1 #include "stm32f10x.h" // Device header
2 #include "Delay.h"
3 #include "Timer.h"
4 int main(void)
5 {
6     Timer_Init();
7     while (1)
8     {
9         if (Flag == 1)
10        {
11            //在此处执行PID控制
12            TIM2_ITStatus(TIM2, TIM_IT_Update) == SET;
13            //每隔时间T，程序执行到这里一次
14            TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
15        }
16    }
17 }
```

out(k) = Kp * error(k) + Ki * Σ_{j=0}^k error(j) + Kd * (error(k) - error(k - 1))

```
1 #include "stm32f10x.h" // Device header
2 #include "Delay.h"
3 #include "Timer.h"
4 //定义变量
5 float Target, Actual, Out; //目标值，实际值，输出值
6 float Kp = 1, Ki = 0.01, Kd = 0.01; //比例项，积分项，微分项的权重
7 float Error0, Error1, ErrorInt; //本次误差，上次误差，误差积分
8
9 int main(void)
10 {
11     Timer_Init();
12     while (1)
13     {
14         //在此处执行PID控制
15         Delay_ms(1000);
16     }
17 }
```

```
1 void TIM2_IRQHandler(void)
2 {
3     if (TIM_GetITStatus(TIM2, TIM_IT_Update) == SET)
4     {
5         //每隔时间T，程序执行到这里一次
6         //在此处执行PID控制
7         Actual = 读取传感器();
8         //获取本次误差和上次误差
9         Error1 = Error0;
10        Error0 = Target - Actual;
11        //调整积分（累加）
12    }
13 }
```

$$\bullet \quad out(k) = K_p * error(k) + K_i * \sum_{j=0}^k error(j) + K_d * (error(k) - error(k-1))$$

```

1 #include "stm32f10x.h" // Device header
2 #include "Delay.h"
3 #include "Timer.h"
4
5 /*定义变量*/
6 float Target, Actual, Out; //目标值, 实际值, 输出值
7 float Kp = 1, Ki = 1, Kd = 1; //比例项, 积分项, 微分项的权重
8 float Error0, Error1, ErrorInt; //本次误差, 上次误差, 误差积分
9
10 int main(void)
11 {
12     Timer_Init();
13     while (1)
14     {
15         /*用户在此处根据需求写入PID控制器的目标值*/
16         Target = 用户指定的一个值;
17     }
18 }

```

```

21 void TIM2_IRQHandler(void)
22 {
23     if (TIM_GetITStatus(TIM2, TIM_IT_Update) == SET)
24     {
25         /*每隔时间t, 程序执行到这里一次*/
26
27         /******执行PID调控******/
28         /*获取实际值*/
29         Actual = 读取传感器();
30
31         /*获取本次误差和上次误差*/
32         Error1 = Error0;
33         Error0 = Target - Actual;
34
35         /*误差积分(累加)*/
36         ErrorInt += Error0;
37
38         /*PID计算*/
39         Out = Kp * Error0 + Ki * ErrorInt + Kd * (Error0 - Error1);
40
41         /*输出限幅*/
42         if (Out > 上限) Out = 上限;
43         if (Out < 下限) Out = 下限;
44
45         /*执行控制*/
46         输出至被控对象(Out);
47         /*******/
48         TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
49     }
50 }
51

```

$$\bullet \quad \Delta out(k) = K_p * (error(k) - error(k-1)) + K_i * error(k) + K_d * (error(k) - 2error(k-1) + error(k-2))$$

```

1 #include "stm32f10x.h" // Device header
2 #include "Delay.h"
3 #include "Timer.h"
4
5 /*定义变量*/
6 float Target, Actual, Out; //目标值, 实际值, 输出值
7 float Kp = 1, Ki = 1, Kd = 1; //比例项, 积分项, 微分项的权重
8 float Error0, Error1, Error2; //本次误差, 上次误差, 上上次误差
9
10 int main(void)
11 {
12     Timer_Init();
13     while (1)
14     {
15         /*用户在此处根据需求写入PID控制器的目标值*/
16         Target = 用户指定的一个值;
17     }
18 }

```

```

21 void TIM2_IRQHandler(void)
22 {
23     if (TIM_GetITStatus(TIM2, TIM_IT_Update) == SET)
24     {
25         /*每隔时间t, 程序执行到这里一次*/
26
27         /******执行PID调控******/
28         /*获取实际值*/
29         Actual = 读取传感器();
30
31         /*获取本次误差, 上次误差和上上次误差*/
32         Error2 = Error1;
33         Error1 = Error0;
34         Error0 = Target - Actual;
35
36         /*PID计算*/
37         Out = Kp * (Error0 - Error1) + Ki * Error0
38             + Kd * (Error0 - 2 * Error1 + Error2);
39
40         /*输出限幅*/
41         if (Out > 上限) Out = 上限;
42         if (Out < 下限) Out = 下限;
43
44         /*执行控制*/
45         输出至被控对象(Out);
46         /*******/
47         TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
48     }
49 }
50
51

```

【3】平衡小车PID调参实例

```

1 #include "pid.h"
2
3 float Vertical_Kp, Vertical_Kd;
4 float Velocity_Kp, Velocity_Ki;
5 uint8_t stop;
6
7 //直立环PID控制器
8 //输入: 期望角度、真实角度、角速度
9 int Vertical(float Med, float Angle, float gyro_Y)
10 {
11     int temp;
12     temp = Vertical_Kp * (Angle - Med) + Vertical_Kd * gyro_Y;
13     return temp;
14 }
15
16 //速度环PI控制器
17 //输入: 期望速度、左编码器、右编码器
18 int Velocity(int Target, int encoder_L, int encoder_R)
19 {
20     static int Err_LowOut_last, Encoder_S;
21     static float a = 0.7;
22     int Err, Err_LowOut, temp;
23     //1、计算偏差值
24     Err = (encoder_L + encoder_R) - Target;
25     //2、低通滤波
26     Err_LowOut = (1 - a) * Err + a * Err_LowOut_last;
27     Err_LowOut_last = Err_LowOut;
28     //3、积分
29     Encoder_S += Err_LowOut;
30     //4、积分限幅(-20000~20000)
31     Encoder_S = Encoder_S > 20000 ? 20000 : Encoder_S < -20000 ? -20000 : Encoder_S;
32     if (stop == 1) Encoder_S = 0, stop = 0;
33     //5、速度环计算
34     temp = Velocity_Kp * Err_LowOut + Velocity_Ki * Encoder_S;
35     return temp;
36 }
37
38 //转向环PI控制器
39 //输入: 期望速度、左编码器、右编码器

```

```

46 //转向环PD控制器
47 //输入: 角速度、角度值
48 int Turn(float gyro_Z, int Target_turn)
49 {
50     int temp;
51     temp = Turn_Kp * Target_turn + Turn_Kd * gyro_Z;
52     return temp;
53 }
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

超调之后系数乘0.6

```

//闭环控制中间变量
int Vertical_out, Velocity_out, Turn_out, Target_Speed, Target_turn, MOT01, MOT02;
float Med_Angle = 7.0; //平衡时角度值偏移量(机械中值)
//参数
float Vertical_Kp, Vertical_Kd; //直立环 数量级(Kp: 0~1000, Kd: 0~10)
float Velocity_Kp, Velocity_Ki; //速度环 数量级(Kp: 0~1)
float Turn_Kp, Turn_Kd; //转向环
uint8_t stop;
extern TIM_HandleTypeDef htim2;

```

所以我们接下来要给它乘一个系数0.6

(直立环) kp: 大幅机械机械振荡; kd: 小幅高频机械振荡
(速度环) 主要是看平衡的时候, 不要让电机产生太大的反向力矩
(转向环) kp不重要, 主要是控制的小车转向的速度, kd是比较重要, 控制小车不转向的时候走直线