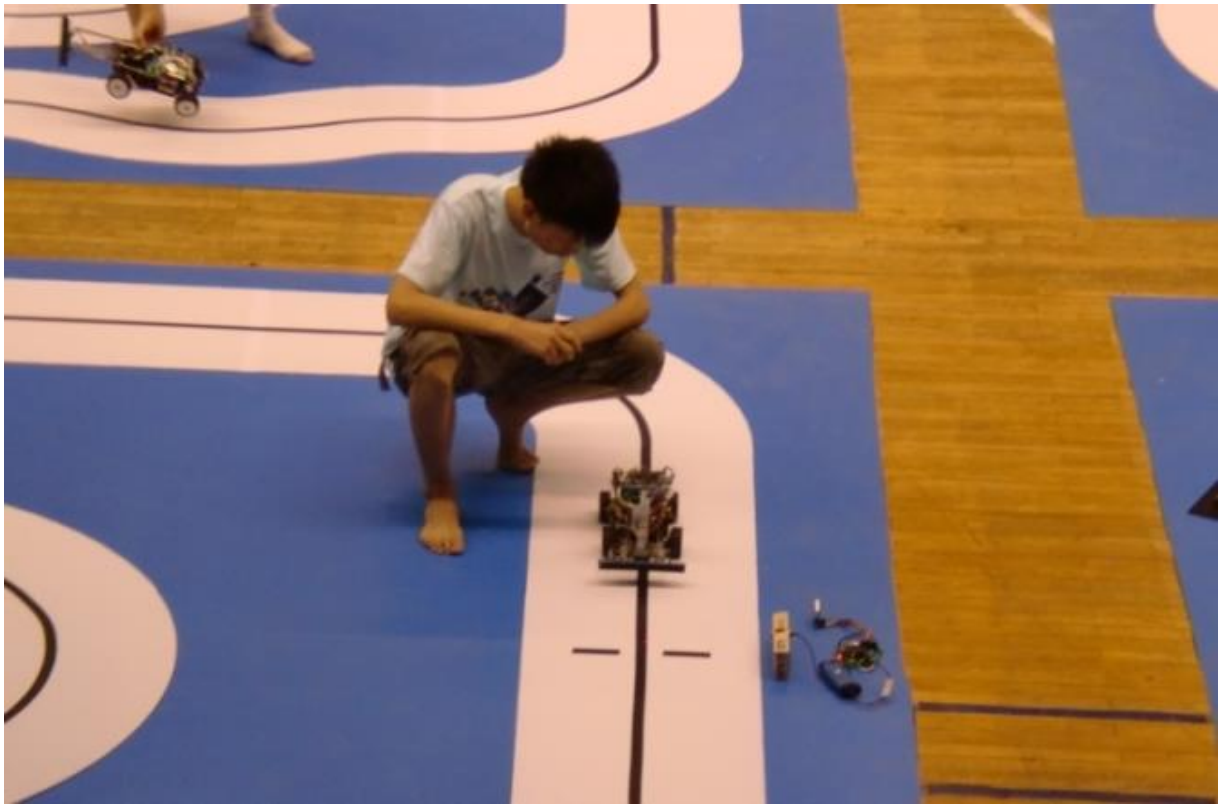
PID 算法深度技术帖*写给参加飞思卡尔智能车大赛的朋友

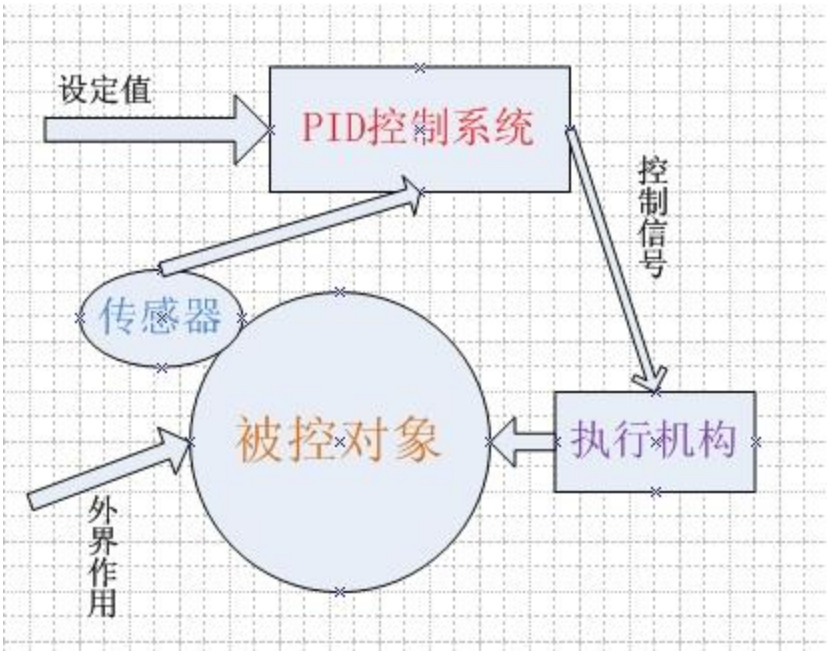
作者：王玉鹏 （郑州轻工业学院）

最近很多朋友问我关于 PID 的算法。这里我就简单整理了下：



PID 控制算法是最经典的自动控制算法。所有《自动控制原理》教材均有大幅篇章介绍。从数学模型到传递函数，公式推导不厌其烦。很多‘童鞋’似懂非懂。下面我就从程序的角度来讲：

下面是 PID 控制系统的原理图



PID 算法具体分两种：一种是位置式的 ，一种是增量式的。

在大多数的情况下，执行机构本身没有状态记忆功能，每时每刻都要靠控制信号驱动，如果失去控制信号，执行机构即失去功能。在这种场合必须采用位置式 PID 控制算法 。

也有一些执行机构具有记忆功能，即使失去驱动信号仍然可以维持原来的状态不变。例如在用步进电机作为执行机构的自动化控制系统中（数控机床最为典型）

下面主要针对位置式 PID 来讲，

以加温设备为例，软件采用 uc/os-ii 嵌入式操作系统环境

```
//采用位置式 PID 控制算法的任务函数

  float Set;//设定值
float KP;//比例常数
float KI;//积分常数
float KD;//微分常数
float LE=0;//上次误差
float SE;//累计误差

void TaskSampleCtrl(void *pdata)//采样控制函数
{
    float now,E,DE,out;//定义临时变量
```

```
    while(1)
    {
        now=SampleAdc();//传感器采样一次

        E=Set-now;          //计算当前误差
        SE+=E;              //计算当前误差的积分值
        DE=E-LE;            //计算当前误差的微分值
        out=KP*E            //计算比例项
            +KI*SE          //计算积分项
            +KD*DE;         //计算微分项

        CtrlOut(out);   //控制执行机构
        LE=E;               //保存误差值以进行下次计算
        OSTimeDly(25);  //心跳20ms  延时0.5s(采样周期)
    }
}
```

PID 参数调节：

1.如果被控制对象的温度很快达到设定值，并超过设定值很多才回调（此种称为超调），则应该减小 KP 和 KI，降低控制强度；加大 KD，抑止超

调。不一定3个参数同时都要调整，可重点调整 KP;

2.如果控制对象的温度很难达到设定值，或者经常低于设定值，则应该加大 KP 和 KI。增加控制的强度。

3.如果控制对象的温度能够控制在设定值上，但上下波动较大。则应该加大 KD，抑止波动。同时适当减少 KI，降低积分控制对微分的控制的对

冲作用。

4.如果容易受环境影响而导致温度波动，则应该加大 KD，抑止波动。

下面是用在智能车的 PID 上的算法实现

```
//飞思卡尔光电智能车大赛  光电组
```

```
//设计者：王玉鹏
//最后修改时间：2010.07. 23.    凌晨1点36
/****************命名规则统一开始************ /
匈牙利命名法
(变量首部加上)
全局变量加 g_
有符号整型：i
无符号整型：u
布尔量：b
无符号 char 型：uch
宏定义 全大写
两个大写字母的单词用 ‘ _ ’字符 分开
*****************命名规则统一结束************/
/*********统一数据类型定义开始********/
//说明：便于移植，书写
typedef unsigned int       INT16U;//无符号16位整数
typedef int                INT16S;//有符号16为整数
typedef unsigned char      CHAR8U;//无符号8位整数
typedef char               CHAR8S;//有符号8位整数
/*********统一数据类型定义结束********/
#define SPEED_PER    10 //10       SPEED 10000=pwm100%    千万别乱改
#define SPEED_TIMER    3 //速度累计采集定时  SPEED_TIMER*4  （ms)     12ms

INT16U    g_uSpeedTemp=0;//系统用
INT16U    g_uSpTimeCnt=0;//速度采集定时用  系统用    0-60000
INT16U    g_uRelSpeedNow=0;//实际速度       0-60000
INT16U    g_uRelSpeedOld=0;//实际速度       0-60000


/////电机 PID 手工设定  开始
//动态 PID       P 参数     ///数组元素地址  0:S 弯道  1：直道    2:保留
INT16S g_iMotP[3]={20,20,20};
//动态 PID       D 参数          ///数组元素地址  0:S 弯道  1：直道    2:保留
INT16S g_iMotD[3]={5,5,5};
/////电机 PID 手工设定  结束



//////////////SpeedPID 开始///////
INT16S iSpeedPidSmart(int iSpPidTempP,int iSpPidTempD);//系统函数，用户禁止调用
void vSpeedPidDo(int iPidTemp); //系统函数，用户禁止调用
//////////////SpeedPID 结束///////
```

```
INT16S iSpeedSet=0;//用户速度设定   ***重要***


//////电机值更新赋值函数 开始/////////
void vFlashSpeedUserSet(void);//用户速度控制设定更新   时间中断保持实时更新
//////电机值更新赋值函数 结束/////////




void main(void)
{
  while(1)
  {
     //其它 code...
     iSpeedSet=50;//直接设定   定时器自动调节电机
     //其它 code...
  }
}



void interrupt 8   ic0_int(void)//路程中断
{
    //其它 code...

    //////////定时速度采集          开始
    g_uSpeedTemp++;//速度反馈
    if(g_uSpeedTemp>60000) g_uSpeedTemp=0;    //确定范围0-60000
    //////////定时速度采集          结束

    //其它 code...

}

void interrupt 16 Timer4MsOver(void)//timer interrupt BUS=32M timer=4ms
{
   //其它 code...
   ////////////////////速度定时采集，即更新实测速度反馈 开始
   if(g_uSpTimeCnt>SPEED_TIMER)//速度采集定时  g_uSpTimeCnt*4  （ms)
   {
      g_uRelSpeedOld=g_uRelSpeedNow;//保存上一次的值
```

```
        g_uRelSpeedNow=g_uSpeedTemp;
        g_uSpeedTemp=0; //测量速度的脉冲清零
        g_uSpTimeCnt=0;
    }
    ////////////////////速度定时采集，即更新实测速度反馈 结束
    g_uSpTimeCnt++;
    vFlashSpeedUserSet();///速度定时更新 只在定时中断里

    //其它 code...

}



/*******电机 PID 理论值计算函数开始************************/

INT16S   iSpeedPidSmart(int iSpPidTempP,int iSpPidTempD)
{
     INT16S   iNow_SetRule;
     INT16S   iOld_SetRule;
     iNow_SetRule=g_uRelSpeedNow-iSpeedSet;//差值量
     iOld_SetRule=g_uRelSpeedOld-iSpeedSet;//差值量

     return ( iNow_SetRule*iSpPidTempP+
                (iNow_SetRule-iOld_SetRule)*iSpPidTempD   );

}
/********电机 PID 理论值计算函数结束************************/

/*******电机 PID 理论值转换为电机 PWM 脉宽  开始***************/
void vSpeedPidDo(int iPidTemp)
{

     //这里注意和舵机的区别 在 pid 计算的负值代表要增加的值
  //被减数本应*SPEED_PER,但发现效果不好
     iTempF=iSpeedSet*SPEED_PER-iPidTemp;
     //容错处理
     if(iTempF>=0)//正向力矩
     {

          if(iTempF>1000) iTempF=1000;//正向最大力矩
          SPEED_F=(INT16U)iTempF;
```

```
            SPEED_B=0;
        }
        else
        {
            if(iTempF<-1000) iTempF=-1000;//反向最大力矩
            SPEED_F=0;
            SPEED_B=(INT16U)(0-iTempF);
        }
}

/****电机 PID 理论值转换为电机 PWM 脉宽结束***************/



/********电机控制动作更新开始************/
void vFlashSpeedUserSet(void)
 {

    if((g_uchTopLedFlag==1)||(uRoadFlag==0)||(uRoadFlag==1))
 //车在直道上
    {
    //电机 PWM 赋值
        vSpeedPidDo(iSpeedPidSmart(g_iMotP[1],g_iMotD[1])) ;
    }
    else//车在 S 上
    {
    //电机 PWM 赋值
        vSpeedPidDo(iSpeedPidSmart(g_iMotP[0],g_iMotD[0])) ;
    }
 }
/********电机控制动作更新结束************/
```

结尾：

PID 的三个控制参数调整不易。最新的算法有“自整定”算法和动态调整算法，属于高端科

研课题，有兴趣的读者可以阅读相关资料。谢谢！

照片名称：全国飞思卡尔智能车大赛
所属相册：生命中经历的点
照片描述：校队照