## 增量式PID算法

 (2012-01-01 18:07:06)

[IMG_256](http://blog.sina.com.cn/s/javascript:;)转载▼

|  |  |
| --- | --- |
| 标签： [杂谈](http://search.sina.com.cn/?c=blog&q=%D4%D3%CC%B8&by=tag" \t "http://blog.sina.com.cn/s/_blank) |  |

|  |
| --- |
| 程序是自己编的，根据多方资料改写了一部分，为增量式PID算法。程序采用浮点形式，如果MCU速度不够快，可根据需求转换定点运算。     关键变量：ref——设置的参数；feb——采样反馈；根据实际情况修改这两个变量即可。     相关参数：Kp、Ti、Td、T：根据各项目的控制对象不同而修改其定义值。  //定义变量 double ref = 0;//设置参数 double feb = 0;//采样反馈   int pwm\_var = 0;    //PID调整量 Uint16 PWM\_out = 0; //PWM输出量   double Uo = 0; double Ek = 0; double Ei = 0; double Ed = 0;  #define Kp   8        //PID调节的比例常数 #define Ti   0.05     //PID调节的积分常数 #define Td   0.02     //PID调节的微分时间常数 #define T    0.02     //采样周期  #define Kpp   Kp \* ( 1 + (T / Ti) + (Td / T) ) #define Ki   (-Kp) \* ( 1 + (2 \* Td / T ) ) #define Kd    Kp \* Td / T  //#define Kpp 4 //#define Ki  0.8 //#define Kd  20  //误差的阀值，小于这个数值的时候，不做PID调整，避免误差较小时频繁调节引起震荡 #define Emin 3  //调整值限幅，防止积分饱和 #define Umax 100 #define Umin -100  //输出值限幅 #define Pmax 15500 #define Pmin 200   /////////////////////////////////////////////////////////////////// //////    PID运算    /////// void pid\_ctrl(void) {   Ek = ref - feb;       //差值运算     if( fabs(Ek) < Emin ) //误差的阀值(死区控制??)   {     pwm\_var = 0;   }   else   {     Uo = Kpp\*Ek + Ki\*Ei + Kd\*Ed;//PID计算     Ed = Ei;     Ei = Ek;         pwm\_var = (int)Uo;    //强制转化调整量,PWM为整数         if(pwm\_var >= Umax)pwm\_var = Umax;      //调整值限幅，防止积分饱和     if(pwm\_var <= Umin)pwm\_var = Umin;    //调整值限幅，防止积分饱和   }     PWM\_out += pwm\_var;        //调整PWM输出     if(PWM\_out > Pmax)PWM\_out = Pmax;    //输出值限幅   if(PWM\_out < Pmin)PWM\_out = Pmin;    //输出值限幅     TBCCR1 = PWM\_out;//输出给寄存器,改变PWM占空比  }  ///////////////////////////////////////////////////////////////////   调用示例：  void main(void) {   ref = 120.6;     while(1)   {     feb = adc\_sample();         pid\_ctrl();   } }  另外附上：****PID调节经验总结****（搜索资源） ****PID****控制器参数选择的方法很多，例如试凑法、临界比例度法、扩充临界比例度法等。但是，对于****PID****控制而言，参数的选择始终是一件非常烦杂的工作，需要经过不断的调整才能得到较为满意的控制效果。依据经验，一般****PID****参数确定的步骤如下： **(1)确定比例系数Kp**     确定比例系数Kp时，首先去掉****PID****的积分项和微分项，可以令Ti=0、Td=0，使之成为纯比例调节。输入设定为系统允许输出最大值的60％～70％，比例系数Kp由0开始逐渐增大，直至系统出现振荡；再反过来，从此时的比例系数Kp逐渐减小，直至系统振荡消失。记录此时的比例系数Kp，设定****PID****的比例系数Kp为当前值的60％～70％。 **(2)确定积分时间常数Ti**     比例系数Kp确定之后，设定一个较大的积分时间常数Ti，然后逐渐减小Ti，直至系统出现振荡，然后再反过来，逐渐增大Ti，直至系统振荡消失。记录此时的Ti，设定****PID****的积分时间常数Ti为当前值的 150％～180％。 **(3)确定微分时间常数Td**     微分时间常数Td一般不用设定，为0即可，此时****PID****调节转换为PI调节。如果需要设定，则与确定Kp的方法相同，取不振荡时其值的30％。 **(4)系统空载、带载联调**     对****PID****参数进行微调，直到满足性能要求。 |

    程序是自己编的，根据多方资料改写了一部分，为增量式PID算法。程序采用浮点形式，如果MCU速度不够快，可根据需求转换定点运算。  
    关键变量：ref——设置的参数；feb——采样反馈；根据实际情况修改这两个变量即可。  
    相关参数：Kp、Ti、Td、T：根据各项目的控制对象不同而修改其定义值。  
  
//定义变量  
double ref = 0;//设置参数  
double feb = 0;//采样反馈  
  
  
int pwm\_var = 0;    //PID调整量  
Uint16 PWM\_out = 0; //PWM输出量  
  
  
double Uo = 0;  
double Ek = 0;  
double Ei = 0;  
double Ed = 0;  
  
#define Kp   8        //PID调节的比例常数  
#define Ti   0.05     //PID调节的积分常数  
#define Td   0.02     //PID调节的微分时间常数  
#define T    0.02     //采样周期  
  
#define Kpp   Kp \* ( 1 + (T / Ti) + (Td / T) )  
#define Ki   (-Kp) \* ( 1 + (2 \* Td / T ) )  
#define Kd    Kp \* Td / T  
  
//#define Kpp 4  
//#define Ki  0.8  
//#define Kd  20  
  
//误差的阀值，小于这个数值的时候，不做PID调整，避免误差较小时频繁调节引起震荡  
#define Emin 3  
  
//调整值限幅，防止积分饱和  
#define Umax 100  
#define Umin -100  
  
//输出值限幅  
#define Pmax 15500  
#define Pmin 200  
  
  
///////////////////////////////////////////////////////////////////  
//////    PID运算    ///////  
void pid\_ctrl(void)  
{  
  Ek = ref - feb;       //差值运算  
   
  if( fabs(Ek) < Emin ) //误差的阀值(死区控制??)  
  {  
    pwm\_var = 0;  
  }  
  else  
  {  
    Uo = Kpp\*Ek + Ki\*Ei + Kd\*Ed;//PID计算  
    Ed = Ei;  
    Ei = Ek;  
     
    pwm\_var = (int)Uo;    //强制转化调整量,PWM为整数  
     
    if(pwm\_var >= Umax)pwm\_var = Umax;      //调整值限幅，防止积分饱和  
    if(pwm\_var <= Umin)pwm\_var = Umin;    //调整值限幅，防止积分饱和  
  }  
   
  PWM\_out += pwm\_var;        //调整PWM输出  
   
  if(PWM\_out > Pmax)PWM\_out = Pmax;    //输出值限幅  
  if(PWM\_out < Pmin)PWM\_out = Pmin;    //输出值限幅  
   
  TBCCR1 = PWM\_out;//输出给寄存器,改变PWM占空比  
  
}  
  
///////////////////////////////////////////////////////////////////  
  
  
调用示例：  
  
void main(void)  
{  
  ref = 120.6;  
   
  while(1)  
  {  
    feb = adc\_sample();  
     
    pid\_ctrl();  
  }  
}  
  
另外附上：****PID调节经验总结****（搜索资源）  
****PID****控制器参数选择的方法很多，例如试凑法、临界比例度法、扩充临界比例度法等。但是，对于****PID****控制而言，参数的选择始终是一件非常烦杂的工作，需要经过不断的调整才能得到较为满意的控制效果。依据经验，一般****PID****参数确定的步骤如下：  
**(1)确定比例系数Kp**  
    确定比例系数Kp时，首先去掉****PID****的积分项和微分项，可以令Ti=0、Td=0，使之成为纯比例调节。输入设定为系统允许输出最大值的60％～70％，比例系数Kp由0开始逐渐增大，直至系统出现振荡；再反过来，从此时的比例系数Kp逐渐减小，直至系统振荡消失。记录此时的比例系数Kp，设定****PID****的比例系数Kp为当前值的60％～70％。  
**(2)确定积分时间常数Ti**  
    比例系数Kp确定之后，设定一个较大的积分时间常数Ti，然后逐渐减小Ti，直至系统出现振荡，然后再反过来，逐渐增大Ti，直至系统振荡消失。记录此时的Ti，设定****PID****的积分时间常数Ti为当前值的 150％～180％。  
**(3)确定微分时间常数Td**  
    微分时间常数Td一般不用设定，为0即可，此时****PID****调节转换为PI调节。如果需要设定，则与确定Kp的方法相同，取不振荡时其值的30％。  
**(4)系统空载、带载联调**  
    对****PID****参数进行微调，直到满足性能要求。

|  |
| --- |
| 程序是自己编的，根据多方资料改写了一部分，为增量式PID算法。程序采用浮点形式，如果MCU速度不够快，可根据需求转换定点运算。     关键变量：ref——设置的参数；feb——采样反馈；根据实际情况修改这两个变量即可。     相关参数：Kp、Ti、Td、T：根据各项目的控制对象不同而修改其定义值。  //定义变量 double ref = 0;//设置参数 double feb = 0;//采样反馈   int pwm\_var = 0;    //PID调整量 Uint16 PWM\_out = 0; //PWM输出量   double Uo = 0; double Ek = 0; double Ei = 0; double Ed = 0;  #define Kp   8        //PID调节的比例常数 #define Ti   0.05     //PID调节的积分常数 #define Td   0.02     //PID调节的微分时间常数 #define T    0.02     //采样周期  #define Kpp   Kp \* ( 1 + (T / Ti) + (Td / T) ) #define Ki   (-Kp) \* ( 1 + (2 \* Td / T ) ) #define Kd    Kp \* Td / T  //#define Kpp 4 //#define Ki  0.8 //#define Kd  20  //误差的阀值，小于这个数值的时候，不做PID调整，避免误差较小时频繁调节引起震荡 #define Emin 3  //调整值限幅，防止积分饱和 #define Umax 100 #define Umin -100  //输出值限幅 #define Pmax 15500 #define Pmin 200   /////////////////////////////////////////////////////////////////// //////    PID运算    /////// void pid\_ctrl(void) {   Ek = ref - feb;       //差值运算     if( fabs(Ek) < Emin ) //误差的阀值(死区控制??)   {     pwm\_var = 0;   }   else   {     Uo = Kpp\*Ek + Ki\*Ei + Kd\*Ed;//PID计算     Ed = Ei;     Ei = Ek;         pwm\_var = (int)Uo;    //强制转化调整量,PWM为整数         if(pwm\_var >= Umax)pwm\_var = Umax;      //调整值限幅，防止积分饱和     if(pwm\_var <= Umin)pwm\_var = Umin;    //调整值限幅，防止积分饱和   }     PWM\_out += pwm\_var;        //调整PWM输出     if(PWM\_out > Pmax)PWM\_out = Pmax;    //输出值限幅   if(PWM\_out < Pmin)PWM\_out = Pmin;    //输出值限幅     TBCCR1 = PWM\_out;//输出给寄存器,改变PWM占空比  }  ///////////////////////////////////////////////////////////////////   调用示例：  void main(void) {   ref = 120.6;     while(1)   {     feb = adc\_sample();         pid\_ctrl();   } }  另外附上：****PID调节经验总结****（搜索资源） ****PID****控制器参数选择的方法很多，例如试凑法、临界比例度法、扩充临界比例度法等。但是，对于****PID****控制而言，参数的选择始终是一件非常烦杂的工作，需要经过不断的调整才能得到较为满意的控制效果。依据经验，一般****PID****参数确定的步骤如下： **(1)确定比例系数Kp**     确定比例系数Kp时，首先去掉****PID****的积分项和微分项，可以令Ti=0、Td=0，使之成为纯比例调节。输入设定为系统允许输出最大值的60％～70％，比例系数Kp由0开始逐渐增大，直至系统出现振荡；再反过来，从此时的比例系数Kp逐渐减小，直至系统振荡消失。记录此时的比例系数Kp，设定****PID****的比例系数Kp为当前值的60％～70％。 **(2)确定积分时间常数Ti**     比例系数Kp确定之后，设定一个较大的积分时间常数Ti，然后逐渐减小Ti，直至系统出现振荡，然后再反过来，逐渐增大Ti，直至系统振荡消失。记录此时的Ti，设定****PID****的积分时间常数Ti为当前值的 150％～180％。 **(3)确定微分时间常数Td**     微分时间常数Td一般不用设定，为0即可，此时****PID****调节转换为PI调节。如果需要设定，则与确定Kp的方法相同，取不振荡时其值的30％。 **(4)系统空载、带载联调**     对****PID****参数进行微调，直到满足性能要求。 |

|  |
| --- |
| 程序是自己编的，根据多方资料改写了一部分，为增量式PID算法。程序采用浮点形式，如果MCU速度不够快，可根据需求转换定点运算。     关键变量：ref——设置的参数；feb——采样反馈；根据实际情况修改这两个变量即可。     相关参数：Kp、Ti、Td、T：根据各项目的控制对象不同而修改其定义值。  //定义变量 double ref = 0;//设置参数 double feb = 0;//采样反馈   int pwm\_var = 0;    //PID调整量 Uint16 PWM\_out = 0; //PWM输出量   double Uo = 0; double Ek = 0; double Ei = 0; double Ed = 0;  #define Kp   8        //PID调节的比例常数 #define Ti   0.05     //PID调节的积分常数 #define Td   0.02     //PID调节的微分时间常数 #define T    0.02     //采样周期  #define Kpp   Kp \* ( 1 + (T / Ti) + (Td / T) ) #define Ki   (-Kp) \* ( 1 + (2 \* Td / T ) ) #define Kd    Kp \* Td / T  //#define Kpp 4 //#define Ki  0.8 //#define Kd  20  //误差的阀值，小于这个数值的时候，不做PID调整，避免误差较小时频繁调节引起震荡 #define Emin 3  //调整值限幅，防止积分饱和 #define Umax 100 #define Umin -100  //输出值限幅 #define Pmax 15500 #define Pmin 200   /////////////////////////////////////////////////////////////////// //////    PID运算    /////// void pid\_ctrl(void) {   Ek = ref - feb;       //差值运算     if( fabs(Ek) < Emin ) //误差的阀值(死区控制??)   {     pwm\_var = 0;   }   else   {     Uo = Kpp\*Ek + Ki\*Ei + Kd\*Ed;//PID计算     Ed = Ei;     Ei = Ek;         pwm\_var = (int)Uo;    //强制转化调整量,PWM为整数         if(pwm\_var >= Umax)pwm\_var = Umax;      //调整值限幅，防止积分饱和     if(pwm\_var <= Umin)pwm\_var = Umin;    //调整值限幅，防止积分饱和   }     PWM\_out += pwm\_var;        //调整PWM输出     if(PWM\_out > Pmax)PWM\_out = Pmax;    //输出值限幅   if(PWM\_out < Pmin)PWM\_out = Pmin;    //输出值限幅     TBCCR1 = PWM\_out;//输出给寄存器,改变PWM占空比  }  ///////////////////////////////////////////////////////////////////   调用示例：  void main(void) {   ref = 120.6;     while(1)   {     feb = adc\_sample();         pid\_ctrl();   } }  另外附上：****PID调节经验总结****（搜索资源） ****PID****控制器参数选择的方法很多，例如试凑法、临界比例度法、扩充临界比例度法等。但是，对于****PID****控制而言，参数的选择始终是一件非常烦杂的工作，需要经过不断的调整才能得到较为满意的控制效果。依据经验，一般****PID****参数确定的步骤如下： **(1)确定比例系数Kp**     确定比例系数Kp时，首先去掉****PID****的积分项和微分项，可以令Ti=0、Td=0，使之成为纯比例调节。输入设定为系统允许输出最大值的60％～70％，比例系数Kp由0开始逐渐增大，直至系统出现振荡；再反过来，从此时的比例系数Kp逐渐减小，直至系统振荡消失。记录此时的比例系数Kp，设定****PID****的比例系数Kp为当前值的60％～70％。 **(2)确定积分时间常数Ti**     比例系数Kp确定之后，设定一个较大的积分时间常数Ti，然后逐渐减小Ti，直至系统出现振荡，然后再反过来，逐渐增大Ti，直至系统振荡消失。记录此时的Ti，设定****PID****的积分时间常数Ti为当前值的 150％～180％。 **(3)确定微分时间常数Td**     微分时间常数Td一般不用设定，为0即可，此时****PID****调节转换为PI调节。如果需要设定，则与确定Kp的方法相同，取不振荡时其值的30％。 **(4)系统空载、带载联调**     对****PID****参数进行微调，直到满足性能要求。 |