10种经典的软件滤波方法

看见代码就想敲　于 2020-03-22 15:40:01 发布

在AD采集中经常要用到数字滤波，而不同情况下又有不同的滤波需求，下面是10种经典的软件滤波方法的程序和优缺点 分析：

1、限幅滤波法（又称程序判断滤波法）
2、中位值滤波法
3、算术平均滤波法
4、递推平均滤波法（又称滑动平均滤波法）
5、中位值平均滤波法（又称防脉冲干扰平均滤波法）
6、限幅平均滤波法
7、一阶滞后滤波法
8、加权递推平均滤波法
9、消抖滤波法
10、限幅消抖滤波法

1、限副滤波

A、方法：
根据经验判断，确定两次采样允许的最大偏差值（设为A）
每次检测到新值时判断：
如果本次值与上次值之差<=A,则本次值有效
如果本次值与上次值之差>A,则本次值无效,放弃本次值,用上次值代替本次值
B、优点：
能有效克服因偶然因素引起的脉冲干扰
C、缺点
无法抑制那种周期性的干扰
平滑度差

程序：


```
1 /* A值可根据实际情况调整
```

```
value为有效值，new_value为当前采样值
   滤波程序返回有效的实际值 */



#define A 10

char value;

char filter()
{

char new_value;
new_value = get_ad();
if ( ( new_value - value > A ) || ( value - new_value > A ) )

return value;

else

return new_value;

}
```

2、中位值滤波法

A、方法：
连续采样N次（N取奇数），把N次采样值按大小排列，取中间值为本次有效值
B、优点：
能有效克服因偶然因素引起的波动干扰，对温度、液位的变化缓慢的被测参数有良好的滤波效果
C、缺点：
对流量、速度等快速变化的参数不宜

程序：

```
/* N值可根据实际情况调整
排序采用冒泡法*/

#define N 11

char filter()
{
```

```
8
  9 | char value_buf[N];
10 char count,i,j,temp;
11     for ( count=0;count<N;count++)
12     {
13
14     value_buf[count] = get_ad();
15
16     delay();
17     }
18
19     for (j=0;j<N-1;j++)
20
21     {
22
23         for (i=0;i<N-j一1;i++)
24         {
25
26             if ( value_buf[i]>value_buf[i+1] )
27             {
28             temp = value_buf[i];
29             value_buf[i] = value_buf[i+1];
30             value_buf[i+1] = temp;
31             }
32
33         }
34
35     }
36 return value_buf[(N-1)/2];
37 }
```

3、算术平均滤波法

A、方法：
连续取N个采样值进行算术平均运算
N值较大时：信号平滑度较高，但灵敏度较低
N值较小时：信号平滑度较低，但灵敏度较高
N值的选取：一般流量，N=12；压力：N=4
B、优点：
适用于对一般具有随机干扰的信号进行滤波
这样信号的特点是有一个平均值，信号在某一数值范围附近上下波动
C、缺点：

对于测量速度较慢或要求数据计算速度较快的实时控制不适用
比较浪费RAM

程序：

```
#define N 12

char filter()
{

int sum = 0;
    for ( count=0;count<N;count++)
    {
    sum + = get_ad();
    delay();
    }
return (char)(sum/N);

}
```

4、递推平均滤波法（又称滑动平均滤波法）（FIR前身）

A、方法：
把连续取N个采样值看成一个队列
队列的长度固定为N
每次采样到一个新数据放入队尾,并扔掉原来队首的一次数据.(先进先出原则)
把队列中的N个数据进行算术平均运算,就可获得新的滤波结果
N值的选取：流量，N=12；压力：N=4；液面，N=4~12；温度，N=1~4
B、优点：
对周期性干扰有良好的抑制作用，平滑度高
适用于高频振荡的系统
C、缺点：
灵敏度低
对偶然出现的脉冲性干扰的抑制作用较差
不易消除由于脉冲干扰所引起的采样值偏差
不适用于脉冲干扰比较严重的场合
比较浪费RAM

程序：

```
#define N 12

char value_buf[N];
char i=0;

char filter()
{

char count;
int sum=0;
value_buf[i++] = get_ad();
    if ( i == N ) i = 0;
    for ( count=0;count<N,count++)
    sum+ = value_buf[count];

return (char)(sum/N);

}
```

5、中位值平均滤波法（又称防脉冲干扰平均滤波法）

A、方法：
相当于“中位值滤波法”+“算术平均滤波法”
连续采样N个数据，去掉一个最大值和一个最小值
然后计算N-2个数据的算术平均值
N值的选取：3~14
B、优点：
融合了两种滤波法的优点
对于偶然出现的脉冲性干扰，可消除由于脉冲干扰所引起的采样值偏差
C、缺点：
测量速度较慢，和算术平均滤波法一样
比较浪费RAM

程序：

```
#define N 12

char filter()
{

```

```
char count,i,j;
char value_buf[N];
int sum=0;
    for (count=0;count<N;count++)
    {
    value_buf[count] = get_ad();
    delay();
    }
    for (j=0;j<N-1;j++)
    {
        for (i=0;i<N-j-1;i++)
        {
            if ( value_buf[i]>value_buf[i+1] )
            {
            temp = value_buf[i];
            value_buf[i] = value_buf[i+1];
            value_buf[i+1] = temp;
            }
        }
    }

    for(count=1;count<N-1;count++)
    sum += value[count];
return (char)(sum/(N-2));
}
```

6、限幅平均滤波法

A、方法:
相当于“限幅滤波法”+“递推平均滤波法”
每次采样到的新数据先进行限幅处理，
再送入队列进行递推平均滤波处理
B、优点:
融合了两种滤波法的优点 对于偶然出现的脉冲性干扰，可消除由于脉冲干扰所引起的采样值偏差
C、缺点:
比较浪费RAM
程序:

```
#define FILTER_A 1
int Filter() {
  int i;
```

```
  int filter_sum = 0; 5    filter_buf[FILTER_N - 1] = Get_data();
  if(((filter_buf[FILTER_N - 1] - filter_buf[FILTER_N - 2]) > FILTER_A) || ((filter_buf[FILTER_N - 2] - filter_buf[FILTER_N - 1]) > FILTER_A))
    filter_buf[FILTER_N - 1] = filter_buf[FILTER_N - 2];
  for(i = 0; i < FILTER_N - 1; i++) {
    filter_buf[i] = filter_buf[i + 1];
    filter_sum += filter_buf[i];
  }
  return (int)filter_sum / (FILTER_N - 1);
}
```

7、一阶滞后滤波法

A、方法：
取a=0~1
本次滤波结果=（1-a）*本次采样值+a*上次滤波结果
B、优点：
对周期性干扰具有良好的抑制作用 适用于波动频率较高的场合
C、缺点：
相位滞后，灵敏度低 滞后程度取决于a值大小 不能消除滤波频率高于采样频率的1/2的干扰信号

程序：

```
/* 为加快程序处理速度假定基数为100，a=0~100 */

#define a 50

char value;

char filter()
{

char new_value;
new_value = get_ad();
return ((100-a)*value + a*new_value);

}
```


8、加权递推平均滤波法

A、方法：
是对递推平均滤波法的改进，即不同时刻的数据加以不同的权
通常是，越接近现时刻的数据，权取得越大。
给予新采样值的权系数越大，则灵敏度越高，但信号平滑度越低
B、优点：
适用于有较大纯滞后时间常数的对象
和采样周期较短的系统
C、缺点：
对于纯滞后时间常数较小，采样周期较长，变化缓慢的信号 不能迅速反应系统当前所受干扰的严重程度，滤波效果差

程序：

```
/* coe数组为加权系数表，存在程序存储区。*/

#define N 12

char code coe[N] = {1,2,3,4,5,6,7,8,9,10,11,12};
char code sum_coe = 1+2+3+4+5+6+7+8+9+10+11+12;

char filter()
{

char count;
char value_buf[N];
int sum=0;
    for (count=0,count<N;count++)
    {
    value_buf[count] = get_ad();
    delay();

    }
for (count=0,count<N;count++)
sum += value_buf[count]*coe[count];

return (char)(sum/sum_coe);
}
```

9、消抖滤波法

A、方法：
设置一个滤波计数器

将每次采样值与当前有效值比较：
如果采样值 = 当前有效值，则计数器清零
如果采样值<>当前有效值，则计数器+1，并判断计数器是否>=上限N(溢出)
如果计数器溢出,则将本次值替换当前有效值,并清计数器
B、优点：
对于变化缓慢的被测参数有较好的滤波效果,
可避免在临界值附近控制器的反复开/关跳动或显示器上数值抖动
C、缺点：
对于快速变化的参数不宜
如果在计数器溢出的那一次采样到的值恰好是干扰值,则会将干扰值当作有效值导入系统

程序：

```
#define N 12

char filter()

{

char count=0;
char new_value;
new_value = get_ad();

    while (value !=new_value)
    {
    count++;
    if (count>=N) return new_value;
    delay();
    new_value = get_ad();
    }

return value;
}
```

10、限幅消抖滤波法

A、方法：
相当于“限幅滤波法”+“消抖滤波法”
先限幅,后消抖

B、优点:

继承了“限幅”和“消抖”的优点

改进了“消抖滤波法”中的某些缺陷,避免将干扰值导入系统

C、缺点:

对于快速变化的参数不宜

程序:

```
#define FILTER_A 1
#define FILTER_N 5
int i = 0;
int Filter() {
  int NewValue;
  int new_value;
  NewValue = Get_data();
  if(((NewValue - Value) > FILTER_A) || ((Value - NewValue) > FILTER_A))
    new_value = Value;
  else
    new_value = NewValue;
if(Value != new_value) {
    i++;
    if(i > FILTER_N) {
      i = 0;
      Value = new_value;
    }
  }
  else
    i = 0;
  return Value;
}
```