【算法】ADC滤波的10种经典算法

原创 | 位元组 | 发于 2022-04-04 17:20:55 修改 | 3394 | 收藏 80

分享笔记 | 算法 | 嵌入式开发 | 分享笔记 | 算法 | 数据结构 | 人工智能

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/Guanfei_yuhi/article/details/119939283

1、限幅滤波法（又称程序判断滤波法）

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）
每次检测到一新值时判断：
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

```
/* 此参数需自定义：value传进来，new_value是程序获取，程序决定用实时还是缓存 */
#define A 10
char Value;
char Filter()
{
  char new_Value;
  new_value = get_ad();
  if(abs(new_Value - Value) > A)  return Value;          //超限则保留
  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个采样值按大小排列
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：
对于频繁、速度较快速变化的参数不宜

```
#define N 11
char Filter()
{
  char value_buf[N];
  char count_i,j,temp;
  for(count = 0;count < N;count++) //获取采样值
  {
    value_buf[count] = get_ad();
    delay();
  }
  for(j = 0;j<(N-1);j++)
  {
    for(i = 0;i<(N-j);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;
      }
    }
  }
  return value_buf[N/2];
}
```

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：
适用于一般具有随机干扰的信号进行滤波
这种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多数据计算速度要较快的实时控制不适用
比较消费RAM

```
#define N 12
char Filter()
{
  int sum = 0;
  for(count = 0;count<N;count++)
  {
    sum += get_ad();
    delay();
  }
  return (char)(sum/N);
}
```

4、滑动平均滤波法（又称滑动平均值滤波法）

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，就可获得新的滤波值
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
不适用脉动较大场合或有效信号频率接近采样频率的场合
比较消费RAM

```
/* 此参数需自定义：value传进来，new_value是程序获取，程序决定用实时还是缓存 */
#define A 10
char Value;
char Filter()
{
  char new_Value;
  new_value = get_ad();
  if(abs(new_Value - Value) > A)  return Value;          //超限则保留
  return new_Value;
}
```

5、中位值平均滤波法（又称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，可以将由于脉冲干扰所引起的采样值偏差较
不稳定的情况下数据不稳定的情况产生的影响
比较消费RAM

```
/* 此参数需自定义：value传进来，new_value是程序获取，程序决定用实时还是缓存 */
#define A 20
char Value;
char Filter();
{
  char new_Value;
  new_value = get_ad();
  if(abs(new_Value - Value) > A)  return Value;          //超限则保留
  return new_Value;
}
```

6、限幅平均滤波法

A、方法：
相当于"限幅法"+"算术平均滤波法"
每次采样到的新数据先进行限幅处理，
再进入算术平均滤波法进行平均滤波处理

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，可以将由于脉冲干扰所引起的采样值偏差

C、缺点：
比较消费RAM

```
#define A 10
#define N 12
char value_i=0;
char value_buf[N];
char Filter()
{
  char new_value,sum0;
  new_value=get_ad();
  if(abs(new_value-value[value_i])> A)  value_buf[i++]=new_value;
  if(i>=N) i=0;
  for(count=0 ;count<N;count++)
  {
    sum+=value_buf[count];
  }
  return (char)(sum/N);
}
```

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：
对偶然出现的干扰具有良好的抑制作用
适用于变化缓慢的被测参数场合

C、缺点：
相位滞后，灵敏度低
滤波幅度取决于a值大小
不能消除周期性质较高于采样频率的1/2的干扰信号

```
/* 此参数需自定义：函数名-char*/
#define a 30
char value;
char Filter();
{
  char new_value;
  new_value=get_ad();
  return (100-a)*value + a*new_value;
}
```

8、加权滤波平均滤波法

A、方法：
是对滤波平均滤波法的改进，即不同时间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较大，历史数值扬弃，这样使得平均数对于现时状况的反应是较灵敏的。

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、变化规律比较复杂的系统，
不能选择合适的权系数来反映各个时刻的"重要程度"，滤波效果差

```
/* coe数组为加权系数表 */
#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：
设置一个按键计数器
每隔一个时间计时器写当前有效键比较：
如果该时键=当前有效键，则计数器清零
如果该时键≠第一次所有有效键，则计数器+1，并判断计数器是否=一上限N(溢出)
如果计数器溢出，则将本次键取代当前有效键，并清计数器

B、优点：
对于会间歇性地接通或断开的开关信号进行滤波消抖，
可避免在程序编制过程中编制繁的延时并节省程序编程时间。

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处理系统

```
#define N 10
char Filter()
{
  char count=0,new_value;
  new_value=get_ad();
  while(value!=new_value)
  {
    count++;
    if(count>=N) return new_value;
    new_value=get_ad();
  }
  return value;
}
```

10、限幅消抖滤波法

A、方法：
相当于"限幅法"和"消抖法"的
为四种法，已N/A

B、优点：
继承了"限幅"和"消抖"的优点
改进了"消抖法"的"长期保持一种错误信息"的缺陷，避免由于干扰导入系统

C、缺点：
对于快速变化的参数不宜

```
#define N 10
#define M 12
char value;
char Filter()
{
  char new_value,count=0;
  new_value=get_ad();
  while(value!=new_value);
  {
    if(abs(value-new_value)>A)
    {
      count++;
      if(count>=M)
      {
        count=0;
        value=new_value;
      }
    }
    else
    {
      if(count>0) count--;
    }
    new_value=get_ad();
    delay();
  }
  return value;
}
```