[常用数字滤波算法总结](https://www.cnblogs.com/qsyll0916/p/6691413.html)

在单片机进行数据采集时，会遇到数据的随机误差，随机误差是由随机干扰引起的，其特点是在相同条件下测量同一量时，其大小和符号会现无规则的变化而无法预测，但多次测量的结果符合统计规律。为克服随机干扰引起的误差，硬件上可采用滤波技术，软件上可采用软件算法实现数字滤波。滤波算法往往是系统测控算法的一个重要组成部分，实时性很强。

**采用数字滤波算法克服随机干扰的误差具有以下优点：**

1、数字滤波无需其他的硬件成本，只用一个计算过程，可靠性高，不存在阻抗匹配问题。尤其是数字滤波可以对频率很低的信号进行滤波，这是模拟滤波器做不到的。

2、数字滤波使用软件算法实现，多输入通道可共用一个滤波程序，降低系统开支。

3、只要适当改变滤波器的滤波程序或运算，就能方便地改变其滤波特性，这对于滤除低频干扰和随机信号会有较大的效果。

4、在单片机系统中常用的滤波算法有限幅滤波法、中值滤波法、算术平均滤波法、加权平均滤波法、滑动平均滤波等。

**(1)限幅滤波算法**

该运算的过程中将两次相邻的采样相减，求出其增量，然后将增量的绝对值，与两次采样允许的最大差值A进行比较。A的大小由被测对象的具体情况而定，如果小于或等于允许的最大差值，则本次采样有效;否则取上次采样值作为本次数据的样本。

算法的程序代码如下：

#defineA //允许的最大差值

chardata; //上一次的数据

char filter()

{

chardatanew; //新数据变量

datanew=get\_data(); //获得新数据变量

if((datanew-data)>A||(data-datanew>A))

return data;

else

returndatanew;

}

说明：限幅滤波法主要用于处理变化较为缓慢的数据，如温度、物体的位置等。使用时，关键要选取合适的门限制A。通常这可由经验数据获得，必要时可通过实验得到。

**(2)中值滤波算法**

该运算的过程是对某一参数连续采样N次(N一般为奇数)，然后把N次采样的值按从小到大排列，再取中间值作为本次采样值，整个过程实际上是一个序列排序的过程。

算法的程序代码如下：

#define N11 //定义获得的数据个数

char filter()

{

charvalue\_buff[N]; //定义存储数据的数组

char count,i,j,temp;

for(count=0;count

{

value\_buf[count]=get\_data();

delay(); //如果采集数据比较慢，那么就需要延时或中断

}

for(j=0;j

{

for(value\_buff[i]>value\_buff[i+1]

{

temp=value\_buff[i];

value\_buff[i]=value\_buff[i+1];

value\_buff[i+1]=temp;

}

}

returnvalue\_buff[(N-1)/2];

}

说明：中值滤波比较适用于去掉由偶然因素引起的波动和采样器不稳定而引起的脉动干扰。若被测量值变化比较慢，采用中值滤波法效果会比较好，但如果数据变化比较快，则不宜采用此方法。

**(3)算术平均滤波算法**

该算法的基本原理很简单，就是连续取N次采样值后进行算术平均。

算法的程序代码如下：

char filter()

{

int sum=0;

for(count=0;count

{

sum+=get\_data();

delay():

}

return (char)(sum/N);

}

说明：算术平均滤波算法适用于对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，信号在某一数值附近上下波动。信号的平均平滑程度完全到决于N值。当N较大时，平滑度高，灵敏度低;当N较小时，平滑度低，但灵敏度高。为了方便求平均值，N一般取4、8、16、32之类的2的整数幂，以便在程序中用移位操作来代替除法。

**(4)加权平均滤波算法**

由于前面所说的“算术平均滤波算法”存在平滑度和灵敏度之间的矛盾。为了协调平滑度和灵敏度之间的关系，可采用加权平均滤波。它的原理是对连续N次采样值分别乘上不同的加权系数之后再求累加，加权系数一般先小后大，以突出后面若干采样的效果，加强系统对参数变化趋势的认识。各个加权系数均小于1的小数，且满足总和等于1的结束条件。这样加权运算之后的累加和即为有效采样值。其中加权平均数字滤波的数学模型是：

式中：D为N个采样值的加权平均值：XN-i为第N-i次采样值;N为采样次数;Ci为加权系数。加权系数Ci体现了各种采样值在平均值中所占的比例。一般来说采样次数越靠后，取的比例越大，这样可增加新采样在平均值中所占的比重。加权平均值滤波法可突出一部分信号抵制另一部分信号，以提高采样值变化的灵敏度。

样例程序代码如下：

char codejq[N]={1,2,3,4,5,6,7,8,9,10,11,12}; //code数组为加权系数表，存在程序存储区

char codesum\_jq=1+2+3+4+5+6+7+8+9+10+11+12;

char filter()

{

char count;

char value\_buff[N];

int sum=0;

for(count=0;count

{

value\_buff[count]=get\_data();

delay();

}

for(count=0;count

sum+=value\_buff[count]\*jq[count];

return(char)(sum/sum\_jq);

}

**(5)滑动平均滤波算法**

以上介绍和各种平均滤波算法有一个共同点，即每获取一个有效采样值必须连续进行若干次采样，当采速度慢时，系统的实时得不到保证。这里介绍的滑动平均滤波算法只采样一次，将一次采样值和过去的若干次采样值一起求平均，得到的有效采样值即可投入使用。如果取N个采样值求平均，存储区中必须开辟N个数据的暂存区。每新采集一个数据便存入暂存区中，同时去掉一个最老数据，保存这N个数据始终是最新更新的数据。采用环型队列结构可以方便地实现这种数据存放方式。

程序代码如下：

char value\_buff[N];

char i=0;

char filter()

{

char count;

int sum=0;

value\_buff[i++]=get\_data();

if(i==N)

i=0;

for(count=0;count

sum=value\_buff[count];

return (char)(sum/N);

}

**(6)低通滤波**

将普通硬件RC低通滤波器的微分方程用差分方程来表求，变可以采用软件算法来模拟硬件滤波的功能，经推导，低通滤波算法如下：

Yn=a\* Xn+(1-a) \*Yn-1

式中 Xn——本次采样值

Yn-1——上次的滤波输出值;

，a——滤波系数，其值通常远小于1;

Yn——本次滤波的输出值。

由上式可以看出，本次滤波的输出值主要取决于上次滤波的输出值(注意不是上次的采样值，这和加权平均滤波是有本质区别的)，本次采样值对滤波输出的贡献是比较小的，但多少有些修正作用，这种算法便模拟了具体有教大惯性的低通滤波器功能。滤波算法的截止频率可用以下式计算：

fL=a/2Pit pi为圆周率3.14…

式中 a——滤波系数;

， t——采样间隔时间;

例如：当t=0.5s(即每秒2次)，a=1/32时;

fL=(1/32)/(2\*3.14\*0.5)=0.01Hz

当目标参数为变化很慢的物理量时，这是很有效的。另外一方面，它不能滤除高于1/2采样频率的干搅信号，本例中采样频率为2Hz，故对1Hz以上的干搅信号应采用其他方式滤除，

低通滤波算法程序于加权平均滤波相似，但加权系数只有两个：a和1-a。为计算方便，a取一整数，1-a用256-a，来代替，计算结果舍去最低字节即可，因为只有两项，a和1-a，均以立即数的形式编入程序中，不另外设表格。虽然采样值为单元字节(8位A/D)。为保证运算精度，滤波输出值用双字节表示，其中一个字节整数，一字节小数，否则有可能因为每次舍去尾数而使输出不会变化。

设Yn-1存放在30H(整数)和31H(小数)两单元中，Yn存放在32H(整数)和33H(小数)中。滤波程序如下：

虽千万里，吾往矣。