IIR滤波器的数值稳定性

在微控制器上实现同一个IIR（Infinite Impulse Response，无限冲激响应）滤波器时，不同的实现方式可能出现不同的数值稳定性问题，对于部分病态情况，会导致滤波器输出严重偏离真实值。现举例来说：

系统采样率，欲设计一个LPF（Low Pass Filter，低通滤波器），通带频率，截止频率，截止频率衰减。令表示输入，表示输出，对应的频域表示分别为、。使用Matlab fdatool 给出一个11阶的Butterworth滤波器：

(1-1)

其中，分子系数，

分母

增益。

频率响应、单位冲击响应如图.1-a、图.1-b：



图.1-a 频率响应



图.1-b 单位冲击响应

该IIR滤波器的时域表示为：

(1-2)

第一种滤波器的实现方法，可根据递推关系式(1-2)，使用个中间状态值临时存储，实现该IIR滤波器。

另一种是状态空间实现方案。该11阶Butterworth滤波器只有11个独立状态，令，采用可控标准型实现：

(1-3)

其中，，，，

，，

，

计算得出的第一行

。

采用该状态空间实现方案，在Cortex-M7内核芯片上使用C++实现，数值类型采用32bit float，计算单位冲激响应，发现很快以指数发散了，如图.2。由于，由式(1-3)知，发散意味着发散了，将绘出，如图.3。



图2. 使用状态空间实现的IIR滤波器，其单位冲击响应



图3. 单位冲击下的状态变化

由于，，故，发散意味着矩阵的第一列发散了。将分解为Jordan标准型，

，

发散也即存在大于的特征值。事实真的如此吗？使用Matlab计算的Jordan标准型分解：

>> [P,J]=jordan(A)

得到的个特征值：

模竟然均小于1！如果使用Matlab采用计算，发现也是收敛的；而若直接用Matlab指令A^n计算，结果发散……

基于如下原因，更倾向于相信的计算结果：

* 使用计算，进而计算，得到的结果与Matlab计算的单位冲激响应结果一致；
* 理论上使用方式的计算结果具有更高的数值稳定性。

那为何使用微控制器、Matlab的A^n指令直接计算结果发散了？原因在于矩阵病态(ill-conditioned)。使用矩阵奇异值分解计算的奇异值：

，，，，

矩阵的条件数。由于变量使用浮点数存储、计算存在舍入误差，设，计算相对误差增量的上界

因此每次计算矩阵乘法，将可能使状态变量的相对误差增加倍。

如果换用递推关系式(1-2)的实现方案呢？此时计算过程等价于选取状态变量，

此时，的奇异值为：，，，。当后，，计算过程变为了：

此时的就是上一种方式中的，仅仅改变了状态变量为输出变量，按照之前的分析，这种实现方式与第一种相比并没有改善数值稳定性，也是有可能发散的。

当然，使用Matlab计算的结果，是收敛的；第一种方案在Cortex-M7内核芯片上采用64bit double数值类型计算，结果也是收敛的。然而，没有改变问题本身的数值特性，仅依靠提高运算精度，算法是不可靠的。

合适的做法是将传递函数分解为多个二阶传递函数的乘积。如本问题，可以分解为5个二阶滤波器和1个一阶滤波器的级联，每个滤波器的传递函数为：

实现时使用作为状态变量以提高数值稳定性，这种实现方案在CMSIS-DSP库中被称为“Biquad Cascade IIR Filters Using Direct Form I Structure”，如图4所示。

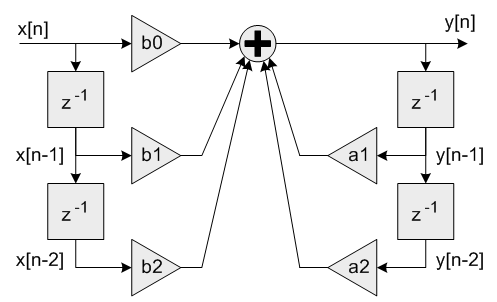


图4. 二阶IIR滤波器Direct Form I结构

6个滤波器的系数分别为：

，，；

，

，

，

，

，

。

若从状态空间的角度看计算过程，每个滤波器的系数矩阵为：

计算每个矩阵（）的右下子阵的条件数，分别为：

，

，

，

，

。

该计算问题是良态的。

思考：

使用个2阶矩阵级联构成一个阶的滤波器，内部独立的状态变量仍然只有个，整个计算过程可以用一个阶的状态方程等效替代。对于所有阶的状态空间实现，由于都是最小实现，相互间可以用一个非奇异线性变换相互转化，。因此原方案是病态的，之所以换用Direct Form I结构实现后变为良态的，就是使用了一个变换，使得条件数显著降低。