线性系统应用的变换研究

Application Study of Linear Systems Transformation

张维玺
（常州技术师范学院电信系　江苏常州213001）

Abstract　This paper presents the principle and design method for digital filter with linear Transformation when transfer function of lowpass simulated filter is given by comprter. Experiments show the effctiveness of model. The algorthm is simple and converges quickly. Preliminary experiment results are included.

Keywords　Linear Transformation, Transfer function, Signal processes, Program, Filter

1. 前言

线性变换在自动控制、通信、无线电技术、滤波器设计、网络综合和分析等领域有着十分重要的用途。通过这种变换系统可以从模拟域变换到数字域，或者从数字域直接变换到模拟域，从理论上讲省去了滤波、采样、量化等环节，非常利于我们对系统进行分析和研究，历来受到人们的重视。

模拟域中系统的传输函数的一般形式为：

$$H(s)=\frac{A(s)}{B(s)}=\frac{a_m s^m+a_{m-1}s^{m-1}+\cdots+a_0}{b_n s^n+b_{n-1}s^{n-1}+\cdots+b_0} \qquad (1)$$

这里 H(s)对应系统的冲激响应是 h(t)。如果 H(s)是模拟域中的低通、高通、带通、带阻，用双线变换可以直接得到数字的低通、高通、带通、带阻滤波器。如果将模拟的低通滤波器 H(s)直接为数字的低通、高通、带通、带阻滤波器，则：

$$\text{数字低通}: H(z)=H(s)|_{s=\frac{z-1}{z+1}} \qquad (2)$$

$$\text{数字高通}: H(z)=H(s)|_{s=\frac{z+1}{z-1}} \qquad (3)$$

$$\text{数字带通}: H(z)=H(s)|_{s=\frac{z^2-2z\cos\omega_0+1}{z^2-1}} \qquad (4)$$

$$\text{数字带阻}: H(z)=H(s)|_{s=\frac{z^2-1}{z^2-2z\cos\omega_0+1}} \qquad (5)$$

考察式(2)、(3)、(4)、(5)可以看出：如果用于手工运算这些式子是十分麻烦的，在阶数较高的情况下是根本计算不出来的。为了克服这个问题我们提出了用计算机求解它的快速计算方法。

2. 快速双线性变换

2.1　*基本原理*　观察(2)、(3)、(4)、(5)式发现，在(4)式中若 $\cos\omega_0=1$　($\omega_0=0$)，则

$$S=\frac{z^2-2z\cos\omega_0+1}{z^2-1}=\frac{z^2-2z+1}{z^2-1}=\frac{z-1}{z+1} \qquad (6)$$

(6)式就是(2)式的变换形式。在(4)式中若 $\cos\omega_0=-1$　($\omega_0=\pi$)，则

$$S=\frac{z^2-2z\cos\omega_0+1}{z^2-1}=\frac{z^2+2z+1}{z^2-1}=\frac{z+1}{z-1} \qquad (7)$$

(7)式就是(3)式的变换形式。因此，我们只要搞清楚带通变换，让 $\omega_0=0$或 $\omega_0=\pi$ 就可以得到数字滤波器的其它三种形式。至于带阻变换，它与带通变换是倒量关系。以后的推导中就会发现它的运算过程和方法与带通变换完全一样。因此，我们只要用计算机程序完成模拟低通到数字带通的变换后改变程序中的有关参数就会随心所欲地得到数字滤波器的四种变换。

2.2　*数字带通变换*　不失一般性，在带通变换中 H(z)为：

$$H(z)=H(s)|_{s=\frac{z^2-2z\cos\omega_0+1}{z^2-1}}$$

$$=\frac{A(z)}{B(z)}=\frac{a_m\left(\frac{z^2-2z\cos\omega_0+1}{z^2-1}\right)^m+a_{m-1}\left(\frac{z^2-2z\cos\omega_0+1}{z^2-1}\right)^{m-1}+\cdots+a_0}{b_m\left(\frac{z^2-2z\cos\omega_0+1}{z^2-1}\right)^m+b_{m-1}\left(\frac{z^2-2z\cos\omega_0+1}{z^2-1}\right)^{m-1}+\cdots+b_0} \qquad (8)$$

为了讨论方便起见，设

$f(z)=z^2-2z\cos\omega_0+1$　　$g(z)=z^2-1$　　　　于是(8)式是：

$$H(z)=\frac{g(z)^m}{g(z)^m}\cdot\frac{a_m f(z)^m+a_{m-1}f(z)^{m-1}g(z)+\cdots+a_1 f(z)g(z)^{m-1}+a_0 f(z)g(z)^m}{b_n f(z)^n+b_{n-1}f(z)^{n-1}g(z)+\cdots+b_1 f(z)g(z)^{m-1}+a_0 f(z)g(z)^m} \qquad (9)$$

按照同样的方法带阻变换的传输函数是：

$$H(z)=\frac{a_m g(z)^m+a_{m-1}g(z)^{m-1}f(z)+\cdots+a_1 g(z)f(z)^{m-1}+a_0 g(z)f(z)^m}{b_n g(z)^n+b_{n-1}g(z)^{n-1}f(z)+\cdots+b_1 g(z)f(z)^{n-1}+a_0 g(z)f(z)^m}\cdot\frac{f(z)^n}{f(z)^m} \tag{10}$$

2.3 H(z)标准形式的计算 要计算出H(z)的标准形式只要展开式(10)就可以了。为了便于计算机运算以式(9)为例说明它的计算方法。由于式(9)中的分子、分母组成结构类同。现以分子A(z)为例加以说明。

$$A(z)=a_n f(z)^m+a_{m-1}f(z)^{m-1}g(z)+\cdots+a_1 f(z)g(z)^{m-1}+a_0 f(z)g(z)^m \tag{11}$$

发现(11)式中：

(1)式中f(z)是以m为零到m降幂排列，g(z)是以同样的幂次升幂排列。它们的最高幂次m是H(s)中分子多项式的最高幂次。

(2)式中的每项是f(z)与g(z)各次幂的相乘，系数为{a_n}。

考虑到计算机运算的特点，为了提高运算速度，我们作如下变化。将f(z)逐次向外提出一次：

$$\begin{aligned}A(z)&=f(z)[f(z)[a_n f(z)^{m-2}+a_{n-1}f(z)^{m-3}g(z)+\cdots+a_2 g(z)^{m-2}]+a_1 g(z)^{m-1}]+a_0 g(z)^m\\&\vdots\\&=f(z)[f(z)[f(z)\cdots f(z)[a_m f(z)+a_{m-1}g(z)]+a_{m-2}g(z)^2]+a_{m-3}g(z)^3+\cdots]+a_0 g(z)^m\end{aligned} \tag{12}$$

这种展开式共有m层。举例说明这种方法：设m=5，则：

$$A(z)=a_5 f(z)^5+a_4 f(z)^4 g(z)+\cdots+a_1 f(z)g(z)^4+a_0 f(z)g(z)^5$$

$$A(z)=f(z)[f(z)[f(z)[a_5 f(z)+a_4 g(z)]+a_3 g(z)^2]+a_2 g(z)^3+\cdots]+a_0 g(z)^5$$

(12)式有以下5个特点：

(1)a_m乘$f(z)$加a_{m-1}乘$g(z)$后组成最里一层。

(2)最里层计算出来后再乘以$f(z)$加a_{m-2}乘$g(z)^2$后构成里一层，直到乘过m个$a_{m-1}g(z)^r(r=1,2,\cdots,m)$后H(z)的分子多项式就计算出来了。

(3)H(z)的分母多项式的计算方法与分子多项式的计算方法完全相同。

(4)整个运算过程很有规律，非常方便计算机实现。

(5)在计算数据运算中一直是"原位运算"节省了许多内存单元。

用同样的方法得出变换后数字带阻的传输函数分子、分母多项式，即式(10)中g(z)与式中(9)f(z)对调、式(10)中f(z)与式(9)中g(z)对调。于是：

$$\begin{aligned}A(z)&=a_m g(z)^m+a_{m-1}g(z)^{m-1}f(z)+\cdots+a_1 g(z)f(z)^{m-1}+a_0 g(z)f(z)^m\\&=g(z)[g(z)[g(z)\cdots g(z)[a_m g(z)+a_{m-1}f(z)]+a_{m-2}g(z)^2]+a_{m-3}g(z)^3+\cdots]+a_0 g(z)^m\end{aligned} \tag{13}$$

很明显，式(13)与(12)在结构形式上完全一样，不言而喻也具有上述5个特点。

2.4 计算机实现方法 要计算出H(z)只要编出两个多项式相乘的子程序就可以了，即：

$$[c_e x^e+c_{e-1}x^{e-1}+\cdots+c_0][d_f x^f+d_{f-1}x^{f-1}+\cdots+d_0]$$

这里e、f是两个非负整数，可以相同，也可以不同，编制该子程序的目的在于计算$g(z)^m$方便，因为前面推导过程中要依次用到$g(z)$、$g(z)^2$、…、$g(z)^m$、$g(z)^n$、$g(z)^{n-m}$(在带阻变换中依次用到$f(z)$、$f(z)^2$、…、$f(z)^m$、$f(z)^n$、$f(z)^{n-m}$等)。计算$g(z)$时可以不必调用此子程序，不过为了统一起见，仍调用此子程序，只不过令其中一个多项式为$g(z)$另一个为1而已。第二步计算$g(z)^2$时在调用子程序前给第一步中赋1的多项式重新赋值，令其为$g(z)$，这样就可调用子程序计算出来$g(z)^2$了。计算$g(z)^3$的过程也类似：原来的$g(z)$多项式不变，而另一个多项式赋于上一次的$g(z)^2$，这样调用子程序后就计算出$g(z)^3$了，以此类推计算出其它项的$g(z)$(或$f(z)$各次幂项)，这个子程序还便于下式的计算：

$$[a_m f(z)^m+a_{m-1}f(z)^{m-1}g(z)+\cdots+a_1 f(z)g(z)^{m-1}+a_0 g(z)^m]\cdot g(z)^{n-m} \tag{14}$$

$$[a_m f(z)^m+a_{m-1}f(z)^{m-1}g(z)+\cdots+a_1 f(z)g(z)^{m-1}+a_0 g(z)^m]\cdot f(z)^{n-m} \tag{15}$$

上面式(14)、(15)是已知的，$g(z)^{n-m}$、$f(z)^{n-m}$也是已知的，这两个多项式的乘积仍可以调用子程序而方便地求出最终结果。

在具体运算中，我们把$f(z)$的方幂多项式展开过程编为子程序，便于改变参数计算分子、分母展开式，用数组M(N)、N(N)分别表示$f(z)$、$g(z)$的系数。

对于$f(z)=z^2-2z\cos\omega_0+1$中，令M(Z)=1 M(1)=2z cos ω_0 M(0)=1；对于$g(z)=z^2-1$中，令M(Z)=1 M(1)=1 M(0)=−1

2.5 计算速度比较 计算机运算速度主要取决于算法的乘法次数。利用快速双线性变换的乘法次数：

$$H(z)=\frac{f(z)[f(z)\cdots f(z)[a_n f(z)+a_{n-1}g(z)]+\cdots]+a_0 g(z)^m}{f(z)[f(z)\cdots f(z)[b_n f(z)+b_{n-1}g(z)]+\cdots]+b_0 g(z)^m}\cdot g(z)^{n-m}$$

当N<M时，乘法次数：

$(m-1)(6m+6)+(n-1)(6n+6)+3(m-n)^2+(2n+1)(2m-2n+1)$

当N>M时，乘法次数：

$(m-1)(6m+6)+(n-1)(6n+6)+3(m-n)^2+(2m+1)(2n-2m+1)$

(下转第103页)

Comm 事件的处理函数 OnOnCommMscomm()，对接收的数据进行处理，在这里所接收的数据是有一定格式的命令报文("命令＋数据")，调用协议转换函数对数据报文进行相关的处理。

3.2 远程通信部分的设计与实现

网关与控制台的远程通信是通过局域网，Intranet，Internet 等多种方式来实现。通信所采用的协议根据网络类型的不同而不同。如果通过 Internet，采用的协议是 TCP/IP 协议，这时控制台与网关之间的通信可以采用连接方式和不连接方式进行通信。对于连接方式，当有多个控制台连接到网关上时，需要对每一个控制台建立一个连接。对于无连接方式，则通过侦听数据报的方式来对请求进行响应。另外，为了达到时实时的服务，一般采用数据报方式服务；对于重要的参数，则采用确认方式的数据报服务。

下面是在 TCP/IP 协议基础上实现网关和控制台之间的数据报通信的方法：

为了捕获到在接受到消息及连接完成等情况下触发的事件，需要用 CAsyncSocket 派生一个子类 CMySocket，为子类的 OnAccept、OcReceive，OnSend、OnClose 消息添加处理函数，添加代码以执行各种事件被触发时需要完成的任务。这可以通过 Class Wizard 来完成。

网关作为服务提供方，需要至少两个 CMySocket 对象，一个用于侦听连接请求，另一个用于通信。这里将两个 CMySocket 对象命名为 m_ListenSocket 和 m_ConnectSocket。网关首先要在某一个端口进行侦听，以探测是否有远程控制台的访问请求。

```
m_ListenSocket.Create(m_iPort);//m_iPort 变量是侦听的端口号
m_ListenSocket.Listen();//调用父类的侦听函数
```

当接收到控制台的连接请求时，m_ListenSocket 对象的 OnAccept 事件被触发，用一个新的 CMySocket 对象进行连接：

```
m_ListenSocket.Accept(m_ConnectSocket);// m_ConnectSocket 是用来建立连接的对象
```

之后网关接收数据并对数据进行处理，如访问授权的验证，命令的解释，信息的反馈等。

对于远程控制台来说，建立一个控制台程序(Console)，只需要一个 CMySocket 对象就可以实现通信了。

```
m_ConnectSocket.Create();
m_ConnectSocket.Connect(m_strGatewayURL,m_iPort);
//m_strGatewayURL 为网关的 IP 地址，m_iPort 为网关进行侦听的端口号
```

连接建立后，就可以用网关和控制台上的连接对象 m_ConnectSocket 的 Send 和 Receive 方法来收发数据进行通信了。限于篇幅该方法的使用请参考 MSDN 里 CAsyncSocket 部分。

参考文献

1 涂时亮，张有德著. 单片微机控制技术. 复旦大学出版社
2 张力，吕振肃，赵庆林. VC++下串行通信的编程方法. 计算机应用，1999(3)
3 Jerry Anderson 著. 张之一，史元春译. Visual C++5 ActiveX 编程指南. 清华大学出版社
4 TCP/IP Illustrated , Volume 2: The Implementation. Gray R. Wright and W. Richard Stevens
5 Microsoft Corp. Introduction to ActiveX Controls. MSDN

(上接第129页)

当 N=M 时，乘法次数：

$(m-1)(6m+6)+(n-1)(6n+6)$

用常规方法计算时推导可得的乘法次数是：

$3m^2+3(m-1)^2+\cdots+3+(2m-1)\cdot 3+(2m-3)\cdot 5+\cdots+3m+3m^2+3n^2+(2m+1)(2n+1)+3n^2+3(n-1)^2+\cdots+3+(2n-1)\cdot 3+(2n-3)\cdot 5+\cdots+3(2n-1)+3m^2+3n^2+(2n+1)(2m+1)$

明显看出快速双线性变换的乘法次数少于常规方法。

结束语 本方法简单、明了具有一定的实际应用价值，其特点是：(1)设计完成可以用计算机一次成功，参数改变很容易，方便地得到低通、高通、带通、带阻等各种滤波器，既可以输出图形，又可以输出数据，图文并茂，便于分析，便于修改，具有很大的通用性。(2)在算法上多次应用迭代、子程序、原位运算等，因此运算速度高，占用内存单元少。(3)程序编制由于用了汉字兼容的方法，所有提示、输出全用汉字，所以十分利于工程人员研究使用。