第 3 章

数字滤波算法及实践

数字滤波在电机控制应用中广泛使用，用来滤除信号波动、抑制干扰、平滑指令、提取指定信号等。本章从应用的角度介绍几种常用的数字滤波器，分析滤波器的幅频响应和相频响应，对滤波器在频域的特点做简单的展示；着重讨论滤波器的实现结构、离散化及代码编写，给出推导过程和示例代码；结合代码给出滤波器部分指标的工程计算方法，将实践与理论联系到一起。

滤波的本质是减小噪声。一个信号通常由有用的原始信号与噪声叠加而成，它们混在一起。放大有用信号，噪声也被放大；减小噪声，有用信号也跟着减小，一般没有通用的方法来改善信噪比。但是如果有两个信号，里面包含了相同的原始信号，把它们叠加在一起：原始信号是相干叠加，叠加后幅度变成原始信号的 2 倍，功率变成原始信号的 4 倍；噪声是非相干叠加，许多地方会相互抵消，功率只变成原始信号的 2 倍。这样，信噪比就变为原来的 2 倍，达到了滤波效果。

除典型的滤波器外，本章还介绍了数字滤波器的一些共性问题，如频响周期性、混叠等。这些问题主要是由采样和数据精度导致的，模拟滤波器没有类似的情况。

关于滤波，有以下几个基本概念。

（1）模拟频率：每秒多少个周期，单位为 Hz，以 f 表示。

（2）模拟角频率：每秒多少弧度，单位为 rad/s，以 w 表示，并且 $w=2\pi f$ 。

（3）数字角频率：两个采样点之间有多少弧度，以 ω 表示，满足 $\omega=2\pi f/f_s$，这是用采样频率 f_s 将模拟角频率归一化之后的结果。

（4）线性相位：相频响应 $\phi(\omega)$ 为数字角频率 ω 的线性函数，在数学上表示为 $\phi(\omega)=a\omega+b$，其中 a 和 b 为常数；通俗解释是信号经过滤波器后，各个频率分量的延时时间都是一样的。

（5）截止频率：以 f_c 表示。

3.1 常用 IIR 滤波器

N 阶 IIR 滤波器差分方程为

$$y(n)=\sum_{k=0}^{N-1}a_k x(n-k)+\sum_{k=0}^{N-1}b_k y(n-k) \tag{3-1}$$

式中，x 和 y 分别为滤波器的输入与输出；a_k 和 b_k 为滤波器系数，k=0,1,2,⋯,N−1。IIR 滤波器结构上存在反馈支路，属于递推型滤波器。滤波器的输出不仅和输入有关，还和过去的输出有关。在某一时刻，输入施加到滤波器中，若没有分辨率的限制，则输出永远不会下降到零，这就是所谓的无限冲激响应。递推型滤波器实现起来占用内存更少，运算也比较少，因此在资源紧张的电机控制领域应用十分广泛。

3.1.1 模拟原型

设计 IIR 滤波器时有多种模拟原型可以选择，不同模拟原型在细节上有所不同。例如，常见的巴特沃斯原型通带较为平坦，但过渡带下降缓慢；切比雪夫原型通带平坦，过渡带下降迅速，但阻带衰减有纹波。设计者根据需要选择模拟原型，一般情况下都希望滤波器通带增益平坦，过渡带下降迅速，同时在阻带能提供足够的衰减。虽然切比雪夫原型阻带衰减有纹波，但是只要最小衰减符合指标，整体上能充分抑制噪声，就不用担心纹波问题。

使用切比雪夫原型可以降低滤波器的阶次，或者在保持滤波器阶次不变的情况下获得更合理的参数。以某高阶滤波器的设计为例，设计参数为采样频率 $f_s = 160\text{kHz}$、通带频率 $f_{pass} = 2\text{kHz}$、阻带频率 $f_{stop} = 16\text{kHz}$、阻带衰减 $A_{stop} = 60\text{dB}$，采用巴特沃斯原型得到滤波器传递函数：

$$h_1(z) = 0.0032 \times \frac{1 + 2z^{-1} + z^{-2}}{1 - 1.9028z^{-1} + 0.9156z^{-2}}，\quad h_2(z) = 0.003 \times \frac{1 + 2z^{-1} + z^{-2}}{1 - 1.7956z^{-1} + 0.8077z^{-2}} \tag{3-2}$$

式（3-2）所示的滤波器阶次为 4，由两个 2 阶滤波器串联构成，增益参数数值较小，分别为 $G_0 = 0.0032$、$G_1 = 0.003$。保持设计指标不变，采用切比雪夫 II 型原型，得到滤波器传递函数：

$$h_1(z) = 0.0202 \times \frac{1 - 1.5064z^{-1} + z^{-2}}{1 - 1.8933z^{-1} + 0.9033z^{-2}}，\quad h_2(z) = 0.0493 \times \frac{1 + z^{-1}}{1 - 0.9013z^{-1}} \tag{3-3}$$

式（3-3）所示为一个 3 阶滤波器，由一个 2 阶滤波器和一个 1 阶滤波器串联构成。子滤波器的增益参数较大，分别为 $G_0 = 0.0202$、$G_1 = 0.0493$，其在数值上比巴特沃斯原型滤波器大一个数量级。

滤波器阶次低，实现起来消耗内存资源更少，并且带来的相位滞后更小，参数值大不易受到量化误差的影响。很明显，在这方面，切比雪夫 II 型模拟原型更有优势。

图 3-1 所示为两种滤波器的幅频响应，可以看到，它们在通带和过渡带上的响应基本上是一致的，但是在阻带上，巴特沃斯原型滤波器的衰减倍数持续下降；而切比雪夫 II 型原型滤波器的衰减则是波动的，未持续下降。巴特沃斯原型滤波器能维持至少 60dB 的衰减，在满足设计指标的基础上还提供了冗余的性能。

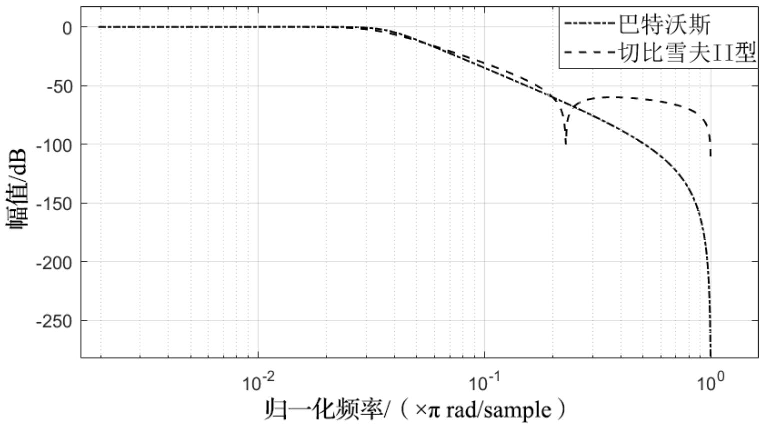

图 3-1　两种滤波器的幅频响应①

图 3-2 所示为两种滤波器的相频响应，在截止频率处，它们的相位滞后都超过了 130°，但在整个通带范围内，切比雪夫 II 型原型滤波器的相位滞后比巴特沃斯原型滤波器的相位滞后小。

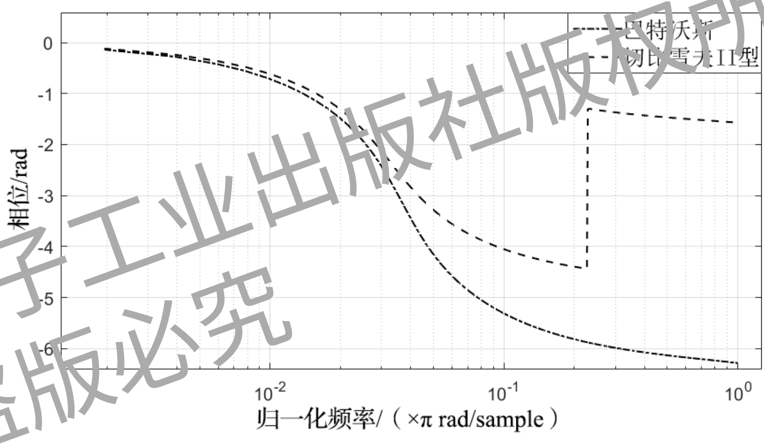

图 3-2　两种滤波器的相频响应

MATLAB 代码如下：

```
        % 幅频响应对比
        a=[9.584,38.337,57.506,38.337,9.584];
        a=a/1000000;
        b=[1,-3.698444,5.139971,-3.180835,0.739462];
        [h,w] = freqz(a,b);
        semilogx(w/pi,20*log10(abs(h)),'k-.','LineWidth',1);

        c=[0.001,-0.000508,-0.000508,0.001];
        d = [1,-2.794596,2.609760,-0.814173];
    [h,w] = freqz(c,d);
    hold on;
```

① 图中的 rad/sample 代表弧度/样本。

```
semilogx(w/pi,20*log10(abs(h)),'k--','LineWidth',1);
grid on
title('Magnitude Response(dB)','FontWeight','Normal')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
legend('Butterworth','Chebyshev II')

%   相频响应对比
       a=[9.584,38.337,57.506,38.337,9.584];
       a=a/1000000;
       b=[1,-3.698444,5.139971,-3.180835,0.739462];
       [h,w] = freqz(b,a);
       semilogx(w/pi,-unwrap(angle(h)),'k-.','LineWidth',1);
       grid on

       hold on
c=[0.001,-0.000508,-0.000508,0.001];
d = [1,-2.794596,2.609760,-0.814173];
[h,w] = freqz(c,d);
semilogx(w/pi,unwrap(angle(h)),'k--','LineWidth',1);

title('Phase Response(dB)','FontWeight','Normal')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Phase (rad)')
legend('Butterworth','Chebyshev II')
```


图 3-3 所示为两种滤波器的阶跃响应，很明显，切比雪夫 II 型原型滤波器的上升时间和调整时间更短，这正是相位滞后小带来的好处。

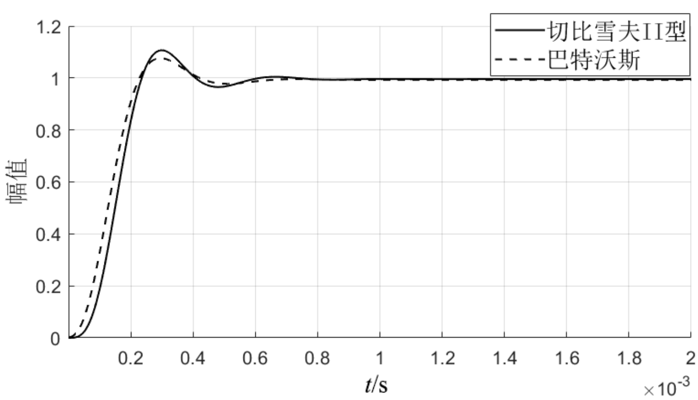

图 3-3　两种滤波器的阶跃响应

阶跃响应分析和绘图相关 MATLAB 代码如下：

```
       a=[9.584,38.337,57.506,38.337,9.584];
       a=a/1000000;
```

```
        b=[1,-3.698444,5.139971,-3.180835,0.739462];
        sys = tf(a,b,1/160000);

        c=[0.001,-0.000508,-0.000508,0.001];
        d = [1,-2.794596,2.609760,-0.814173];
        sys1 = tf(c,d,1/160000);

        close all;
hold on
grid minor;
[y,t] = step(sys ,0.0014);
plot(t,y,'k-','LineWidth',1)
[y,t] = step(sys1 ,0.0014);
plot(t,y,'k--','LineWidth',1)
legend('Chebyshev II','Butterworth')
xlabel('time (s)')
ylabel('Amplitude')
title('Step Response','FontWeight','Normal')
```

3.1.2 实现结构

滤波器传递函数可以有多种不同的表达形式，每种都对应着不同的算法，也对应着不同的实现结构。例如：

$$h_1(z)=\frac{1}{1-0.3z^{-1}-0.4z^{-2}} \tag{3-4}$$

可以分解为

$$h_1(z)=\frac{1}{1-0.8z^{-1}}\cdot\frac{1}{1+0.5z^{-1}} \tag{3-5}$$

或

$$h_1(z)=\frac{0.6154}{1-0.8z^{-1}}+\frac{0.3846}{1+0.5z^{-1}} \tag{3-6}$$

上述同一滤波器的 3 种不同的表达形式就对应着不同的实现结构。IIR 滤波器常见的结构形式有直接 I 型、直接 II 型（典范型）、级联型、并联型。通过差分方程能够画出的包含反馈结构的数字网络称为直接型。

例如，某 N 阶差分方程：

$$h(z)=\sum_{k=1}^{N}a_k y(n-k)+\sum_{k=1}^{N}b_k x(n-k) \tag{3-7}$$

其网络结构可由差分方程直观地画出，如图 3-4 所示。

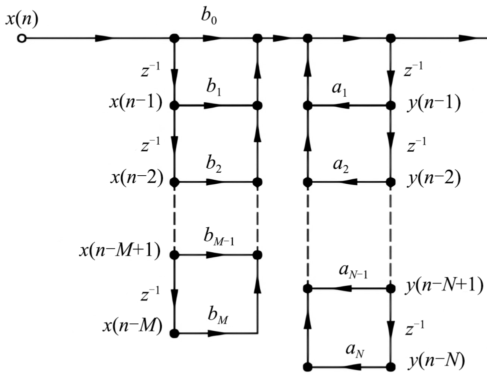

图 3-4　IIR 滤波器直接 I 型网络结构

图 3-4 所示的滤波器可以看作两个系统的串联，因为它们是线性系统，所以交换两个系统的顺序不影响输出结果，即传递函数可以写为

$$h(z) = B(z)\frac{1}{A(z)} = \frac{1}{A(z)}B(z) \tag{3-8}$$

交换顺序后滤波器的网络结构（IIR 滤波器直接 II 型网络结构）如图 3-5 所示，可以看到，延迟模块的输入和输出不再是输入信号或输出信号，取而代之的是一个中间变量。同时，由于两个子系统的延迟模块的输入相同，因此它们可以合并处理，如图 3-6 所示。

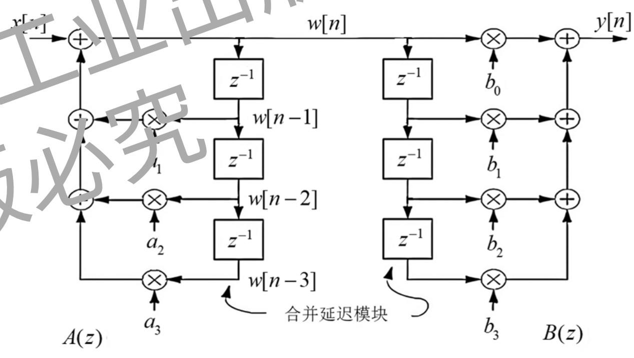

图 3-5　IIR 滤波器直接 II 型网络结构

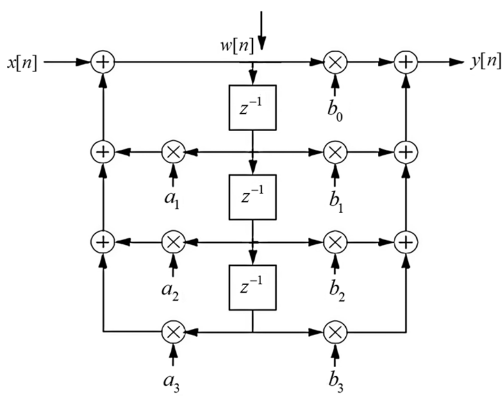

图 3-6　合并延迟模块后的网络结构

直接 II 型网络结构减少了延迟模块，对具体实现而言，这意味着可以减少存储单元的使用，并且减少延迟缓冲区的更新工作，有一定的积极意义。从运行速度上来说，计算中间变量打断了直接 I 型网络结构那样的两个表连续乘加的流程，同时增加了一次乘加运算，实际计算速度可能会慢一些。

直接型网络结构简单，使用的延迟器较少（N 和 M 中的较大者）。当使用 DSP 实现滤波器时，通过优化汇编可以获得其他网络结构无法达到的运行速度，但是直接型系数 a_k、b_k 对滤波器性能的控制关系不直接，因此调整参数不方便。具体在实现滤波器时，a_k、b_k 的量化误差可能使滤波器的频率响应产生一定程度的改变，甚至影响系统的稳定性。因此，直接型网络结构一般用于实现低阶系统。

串联型和并联型网络结构多用于高阶滤波器，将高阶滤波器拆分成多个子滤波器串联或并联的形式，每个子滤波器仍然采用直接型网络结构。这样处理的好处在于可以分别调整子滤波器的参数，并且不易受到参数量化误差的影响。

3.1.3　一阶低通滤波器

3.1.3.1　公式推导

一阶低通滤波器的传递函数为

$$H(s)=\frac{1}{\tau s+1} \tag{3-9}$$

式中，τ 为时间常数，数值上 $\tau=1/(2\pi f_{\mathrm{c}})$，其中 f_{c} 为滤波器的截止频率。

利用后向差分将式（3-9）离散化，即令

$$s=(1-z^{-1})/T_{\mathrm{s}} \tag{3-10}$$

式中，T_{s} 为采样周期。将式（3-10）代入式（3-9）可得如下差分方程：

$$y_n=\frac{\tau}{T_{\mathrm{s}}+\tau}y_{n-1}+\frac{T_{\mathrm{s}}}{T_{\mathrm{s}}+\tau}x_n \tag{3-11}$$

式（3-11）为一阶滞后滤波最常见的差分形式，便于递推实现，是非常典型的 IIR 滤波器。除利用后向差分实现离散化以外，还可以使用双线性变换实现离散化，这里直接给出差分方程：

$$y_n=\frac{2\tau-T_{\mathrm{s}}}{2\tau+T_{\mathrm{s}}}y_{n-1}+\frac{T_{\mathrm{s}}}{2\tau+T_{\mathrm{s}}}x_n+\frac{T_{\mathrm{s}}}{2\tau+T_{\mathrm{s}}}x_{n-1} \tag{3-12}$$

看起来，式（3-12）可能会不稳定，当 y_{n-1} 的系数为负数并且输入为零时，滤波输出将会出现振荡。实际上，这里涉及系统离散化时采样频率的选择问题，实际应用中总是要求采样周期远小于其时间常数，因此并不会出现这种系数为负的情况。

将连续传递函数离散化的方法有很多种，一种常见的一阶滞后滤波器的 z 域传递函数形式为

$$M=\frac{z(1-\mathrm{e}^{-wT_\mathrm{s}})}{z-\mathrm{e}^{-wT_\mathrm{s}}} \tag{3-13}$$

式（3-13）同样是一阶滞后滤波器的 s 域传递函数在 z 域的近似（这里保证离散化得到的数字滤波器与模拟原型具有相同的阶跃响应），但是采用的方式并不是前向差分或后向差分，或者双线性变换法。此时，差分方程为

$$y_k=\mathrm{e}^{-wT_\mathrm{s}}y_{k-1}+(1-\mathrm{e}^{-wT_\mathrm{s}})x_k \tag{3-14}$$

式（3-11）和式（3-14）具有相同的形式，即

$$y_k=(1-a)y_{k-1}+ax_k \tag{3-15}$$

式中，a 为滤波器系数，$0<a<1$。a 越大，滤波器的输出越依赖输入，滤波器作用越小；反之则越依赖滤波器上一次的输出，滤波器作用越大。

3.1.3.2　定点编码实现

式（3-15）常见的 C 代码实现如下：

```
y = ((long)y<<16) + ((long)x<<12) - ((long)y<<12)  >>16;       // a = 1/16
```

其中，x 为输入变量，y 为输出变量。将变量左移 16 位本质上是在利用 Q16 格式数据处理滤波器系数，即常系数 a = 1/16 = (1<<12)>>16。因为变量 x 和 y 都是 16 位的数据，所以左移为 32 位数据不用考虑数据溢出的问题。这种实现方式的缺点是滤波器稳态输出存在静差，如给定 x=200，最终稳态输出 y = 185。

保持实现方式不变，增加四舍五入算法，当输入 x=200 时，最终稳态输出 y = 193。此时，稳态误差有所减小，但仍然存在，代码如下：

```
y = ((long)y<<16) + ((long)x<<12) - ((long)y<<12) +32768  >>16;
```

稍加分析可以发现，上述代码中精度的损失主要来源于表达式的右移运算，右移 16 位时表达式低 16 位的信息被丢弃，因此可以添加一个缓存变量来累积丢弃部分，代码如下：

```
temp_32 = temp_32 + ((long)x<<12) - (temp_32 >>4);
result = temp_32 + 32768  >> 16;
```

上述代码增加了一个中间变量 temp_32，计算结果的低 16 位得到累积，从而跟踪精度大大提高。当给定 x=100 时，result 可到 99；当 x=200 时，result 可上升至 199。如果在增加中间变量的基础上引入四舍五入算法，那么基本上可以完全跟踪，从而解决稳态误差问题。

编写代码时选择正确的数据类型很重要，如果输入变量是有符号整型，那么中间变量应该是有符号长型，否则，当输入为负值时，程序结果将出现错误。