

通信系统仿真

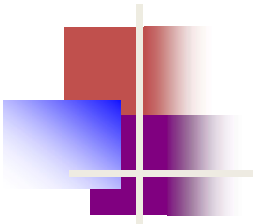
IIR滤波器

何晨光

哈尔滨工业大学

电子与信息工程学院

Communication Research Center



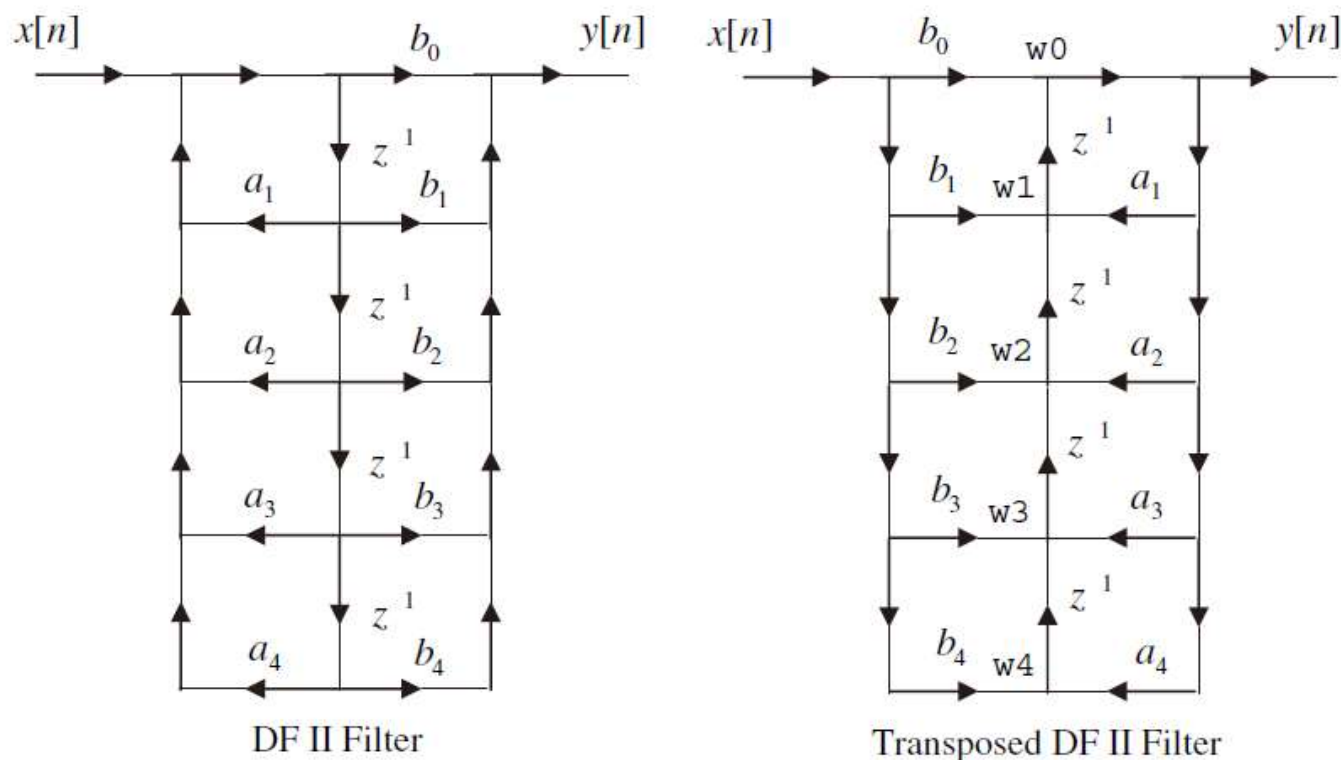
线性数字信号处理器(数字滤波器)计算过去的 N 个输出采样点 $y[n-k](1 \leq k \leq N)$ ，当前的输入采样点 $x[n]$ ，已寄过去的 N 个输入采样点 $x[n-k](1 \leq k \leq N)$ 的加权和作为当前的输出采样点 $y[n]$ 。也就是说，根据过去的输入信号及输出信号计算当前输出信号的算法：

$$y[n] = \sum_{k=0}^N b_k x[n-k] - \sum_{k=1}^N a_k y[n-k]$$

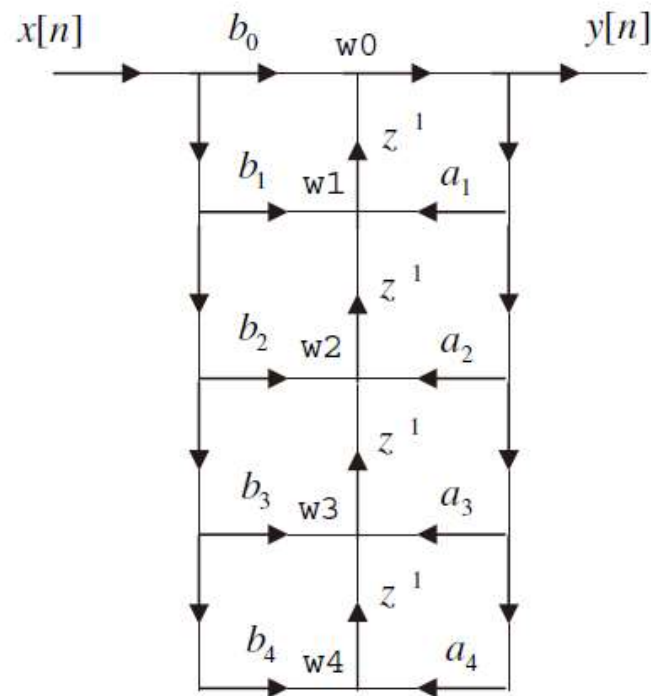
对上式两边进行 z 变换，可以得到传递传输 $H(z)$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^N b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}}$$

在仿真程序中实现IIR数字滤波器的一种有效方法是采用转置直接II型结构，可以很容易的从直接II型滤波器结构推导出来。

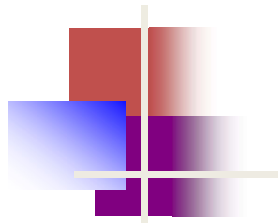


四阶转置直接II型滤波器，给定输入采样 $x[n]$ ，要计算输出信号 $y[n]$ ，第一步，计算状态变量 $w_j[n], j=0,1,2,3,4$ 。对于四阶滤波器，在公式中有五个状态变量。这五个状态变量的计算式如下：



Transposed DF II Filter

$$\begin{aligned}
 w_0[n] &= w_1[n-1] + b_0x[n] \\
 w_1[n] &= -a_1w_0[n] + w_2[n-1] + b_1x[n] \\
 w_2[n] &= -a_2w_0[n] + w_3[n-1] + b_2x[n] \\
 w_3[n] &= -a_3w_0[n] + w_4[n-1] + b_3x[n] \\
 w_4[n] &= -a_4w_0[n] + b_4x[n]
 \end{aligned}$$



$$w_0[n] = w_1[n-1] + b_0x[n]$$

$$w_1[n] = -a_1w_0[n] + w_2[n-1] + b_1x[n]$$

$$w_2[n] = -a_2w_0[n] + w_3[n-1] + b_2x[n]$$

$$w_3[n] = -a_3w_0[n] + w_4[n-1] + b_3x[n]$$

$$w_4[n] = -a_4w_0[n] + b_4x[n]$$

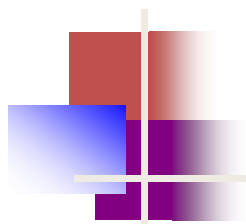
```
w1 = 0; w2 = 0; w3 = 0; w4 = 0;           % initialize state variables
for k = 1:npts                             % beginning of simulation loop

    :

    w0 = w1 + b0*x;
    w1 = -a1*w0 + w2 + b1*x;
    w2 = -a2*w0 + w3 + b2*x;
    w3 = -a3*w0 + w4 + b3*x;
    w4 = -a4*w0 + b4*x;
    y = w0;

    :

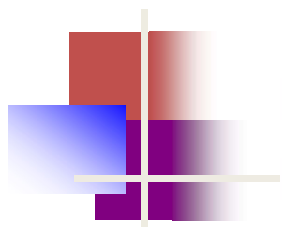
end                                         % end of simulation loop}
```



```
w1 = 0; w2 = 0; w3 = 0; w4 = 0;
for k = 1:npts
    :
    w0 = w1 + b0*x;
    w1 = -a1*w0 + w2 + b1*x;
    w2 = -a2*w0 + w3 + b2*x;
    w3 = -a3*w0 + w4 + b3*x;
    w4 = -a4*w0 + b4*x;
    y = w0;
    :
end
```

x和y分别代表滤波器的当前输入x[k]和当前输出y[k]。

状态变量w1,w2,w3,w4在首次进入仿真循环之前必须进行初始化，这一初始化导致滤波器输出一个瞬态响应。一般必须执行仿真循环多次之后，才能从仿真的输出中采集到有用的数据，这段时间通常称为“稳定时间”，其大小为滤波器贷款倒数的若干倍。



$$w_0[n] = w_1[n-1] + b_0x[n]$$

$$w_1[n] = -a_1w_0[n] + w_2[n-1] + b_1x[n]$$

$$w_2[n] = -a_2w_0[n] + w_3[n-1] + b_2x[n]$$

$$w_3[n] = -a_3w_0[n] + w_4[n-1] + b_3x[n]$$

$$w_4[n] = -a_4w_0[n] + b_4x[n]$$

单输入滤波器的一般表达式:

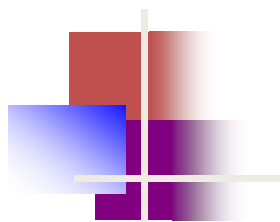
$$\mathbf{W}[n] = \mathbf{F}_c \mathbf{W}[n] + \mathbf{F}_d \mathbf{W}[n-1] + \mathbf{B}x[n]$$

$\mathbf{W}[n]$ 和 $\mathbf{W}[n-1]$ 是 $k \times 1$ 列向量, 分别表示当前和过去的状态变量。

\mathbf{F}_c 和 \mathbf{F}_d 是 $k \times k$ 的系数矩阵, \mathbf{B} 是 $k \times 1$ 的列向量, 用来将输入 $x[n]$ 耦合到状态变量。

单输出信号 $y[n]$ 的输出方程为 $y[n] = \mathbf{C}\mathbf{W}[n]$

\mathbf{C} 是一个 $1 \times k$ 的行向量。



$$w_0[n] = w_1[n-1] + b_0x[n]$$

$$w_1[n] = -a_1w_0[n] + w_2[n-1] + b_1x[n]$$

$$w_2[n] = -a_2w_0[n] + w_3[n-1] + b_2x[n]$$

$$w_3[n] = -a_3w_0[n] + w_4[n-1] + b_3x[n]$$

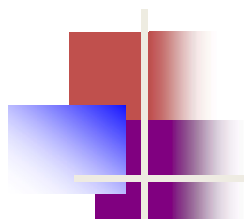
$$w_4[n] = -a_4w_0[n] + b_4x[n]$$

$$y[n] = w_0[n] = w_0[n] + b_0x[n]$$

$$\text{out} = \text{b}(1)*\text{in} + \text{sreg}(1,1);$$

$$\begin{bmatrix} w_0[n] \\ w_1[n] \\ w_2[n] \\ w_3[n] \\ w_4[n] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -a_1 & 0 & 0 & 0 & 0 \\ -a_2 & 0 & 0 & 0 & 0 \\ -a_3 & 0 & 0 & 0 & 0 \\ -a_4 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_0[n] \\ w_1[n] \\ w_2[n] \\ w_3[n] \\ w_4[n] \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_0[n-1] \\ w_1[n-1] \\ w_2[n-1] \\ w_3[n-1] \\ w_4[n-1] \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} x[n]$$

$$\begin{aligned} \text{sreg} &= \text{in}*\text{b} - \text{out}*\text{a} + \text{sreg}; \\ \text{sreg} &= [\text{sreg}(1,2:(\text{order}+1)),0]; \end{aligned}$$



```

out = b(1)*in + sreg(1,1);           % compute filter output
sreg = in*b - out*a + sreg;          % update shift register contents
sreg = [sreg(1,2:(order+1)),0];      % cycle shift register

```

$$y[n] = w_0[n] = w_0[n] + b_0 x[n]$$

```
out = b(1)*in + sreg(1,1);
```

$$\begin{bmatrix} w_0[n] \\ w_1[n] \\ w_2[n] \\ w_3[n] \\ w_4[n] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -a_1 & 0 & 0 & 0 & 0 \\ -a_2 & 0 & 0 & 0 & 0 \\ -a_3 & 0 & 0 & 0 & 0 \\ -a_4 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_0[n] \\ w_1[n] \\ w_2[n] \\ w_3[n] \\ w_4[n] \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_0[n-1] \\ w_1[n-1] \\ w_2[n-1] \\ w_3[n-1] \\ w_4[n-1] \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} x[n]$$

```

sreg = in*b - out*a + sreg;
sreg = [sreg(1,2:(order+1)),0];

```



步骤三：编写BER子函数

例6：采用快处理和串行（逐个采样点）处理方法，来确定一个四阶巴特沃思滤波器的冲激响应。

(代码见c212.m)

```

n = 40;                % number of samples
order = 4;             % filter order
[b, a] = butter(order, 0.1); % prototype
%
% The following program segment is the block processing implementation.
%

```

```

in1 = [1, zeros(1, n-1)]; % input vector
out1 = filter(b, a, in1); % output vector
%

```

```

% The following program segment is the sample-by-sample implementation.
%

```

```

sreg = zeros(1, order+1); % initialize shift register

```

```

for k=1:n
    if k==1
        in=1; % impulse input
    else
        in=0;
    end
    out = b(1)*in + sreg(1,1); % determine output
    sreg = in*b - out*a + sreg; % update register
    sreg = [sreg(1, 2:(order+1)), 0]; % shift
    out2(k) = out; % create output vector

```

```

end

```

Sreg代表长度为order+1的移位寄存器，
order为滤波器阶数

步骤三：编写BER子函数

例：采用快处理和串行（逐个采样点）处理方法，来确定一个四阶巴特沃思滤波器的冲激响应。

