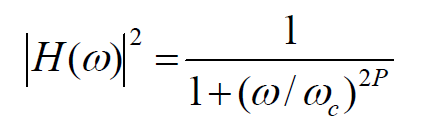


**ButterWorth滤波器的实现：**

低通滤波器：



其中*ｗｃ*为截止频率，P为滤波器的阶数。

利用双线性变换将其模拟滤波器变为数字滤波器。

其中T为采样周期，这就是模拟s和数字z的联系，理论上可以直接用，但是指数函数用起来比较复杂，不方便，我们需要一个更简单的形式。

稍微变一下形：



函数*ex*的泰勒展开为：



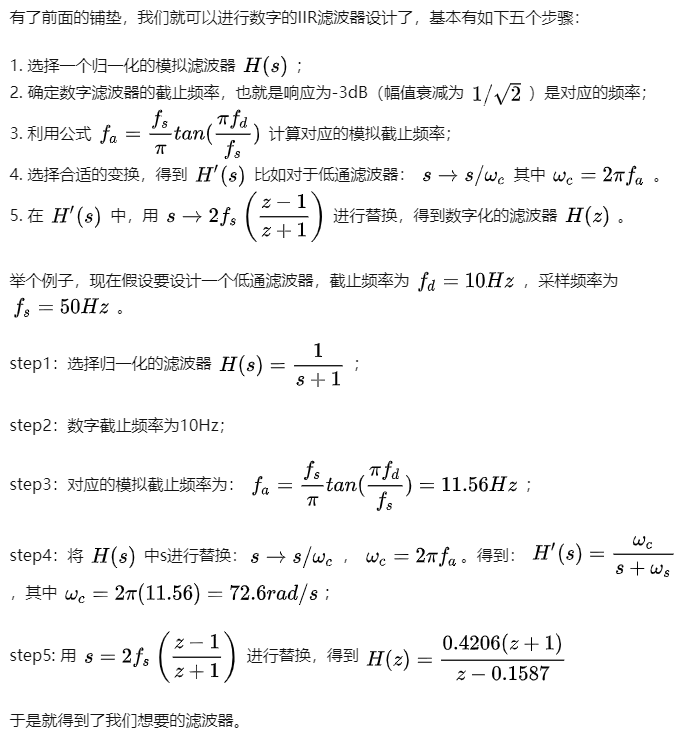
利用上式分别对分子和分母进行泰勒展开，并只取前两项：



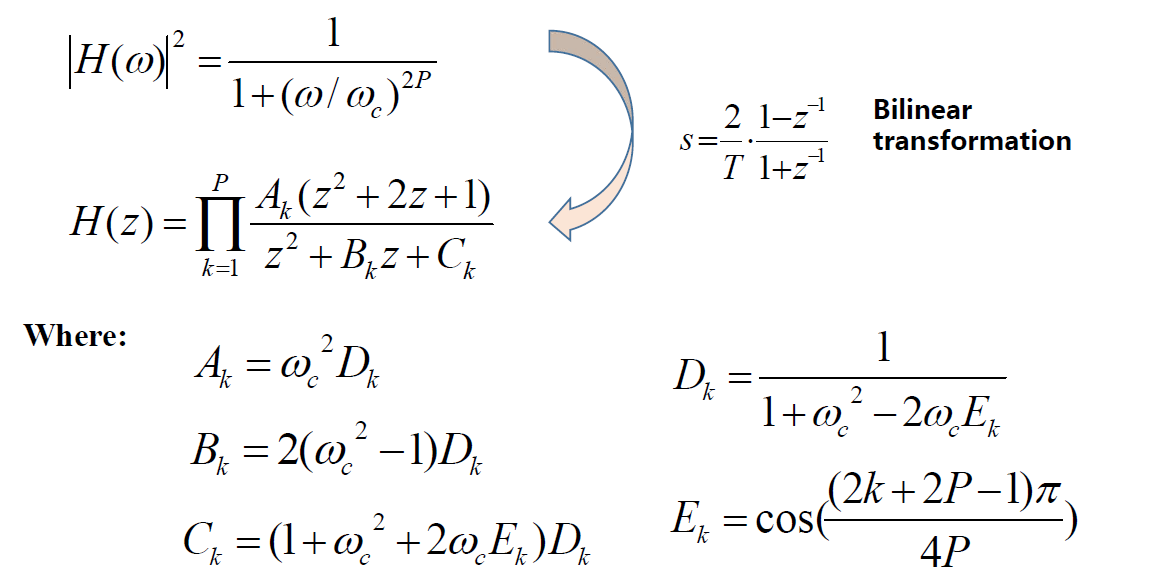
所以：



这就是双线性变换。因此，双线性变换本质就是就是将s域（模拟量）无穷大的频率映射到z域的  ，因为根据香浓采样定理，数字信号的包含的最大分量的频率就是一半的采样频率，否则就会产生混叠。



对于低通滤波器进行双线性变换后可以得到：



利用程序计算其中的系数即可：

A=zeros(1,num);

B=zeros(1,num);

C=zeros(1,num);

D=zeros(1,num);

E=zeros(1,num);

for k=1:num

E(k)=cos((2\*k+2\*num-1)\*pi/(4\*num));

D(k)=1/(1+low^2-2\*low\*E(k));

A(k)=low^2\*D(k);

B(k)=2\*(low^2-1)\*D(k);

C(k)=(1+low^2+2\*low\*E(k))\*D(k);

end

（因为这个实验中没有想到合理的滤波方式，所以直接使用了MATLAB自带的滤波函数）

b=[A(k),2\*A(k),A(k)];

a=[1,B(k),C(k)];

x2= filter(b,a,x1);

（这里，未考虑高阶的，只考虑了P=1时的情况）

高通滤波器的实现：考虑到高通滤波器实则为原信号减去低通的部分所以实现时，将上述代码中的low值换为high值，获得滤波后的函数后，用原信号减去即可：

for k=1:num

E(k)=cos((2\*k+2\*num-1)\*pi/(4\*num));

D(k)=1/(1+high^2-2\*high\*E(k));

A(k)=high^2\*D(k);

B(k)=2\*(high^2-1)\*D(k);

C(k)=(1+high^2+2\*high\*E(k))\*D(k);

end

b=[A(k),2\*A(k),A(k)];

a=[1,B(k),C(k)];

x3= filter(b,a,x1);

plot(handles.axes5,t,x1-x3,'linewidth',2);

axis(handles.axes5,[0, 0.1, -1.5,1.5]);

其中x1为原信号，x3为利用high计算出来的低通信号。

带通滤波器的实现：考虑到带通滤波器为原信号减去低通与高通的部分而通过的信号，即等于以低通频率为high计算出来的结果减去以低通频率为low计算出来的结果，所以实现如下述：

plot(handles.axes5,t, x3-x2,'linewidth',2);

axis([0, 0.1, -1.5,1.5]);

其中，x2为利用low计算出的低通，x3为利用high计算出的低通。

带阻滤波器的实现：带阻滤波器为低通部分加高通部分，所以带阻滤波器的实现为：

plot(handles.axes5,t,x2+x1-x3,'linewidth',2);

axis([0, 0.1, -1.5,1.5]);

从视频中可以看出，自己编写的函数与MATLAB自带的函数结果有差距，这里分析如下：

自己编写的函数中：*wc*的值为其中，而自带函数中的*wn*为：，对于同一个滤波器而言，*wc*和*wn*的值理论上是不同的，而在程序中，两值是相同的，因此造成结果有一定差距。