ELECTRONOTES
1 Pheasant Lane
Ithaca, NY 14850
(607)-273-8030

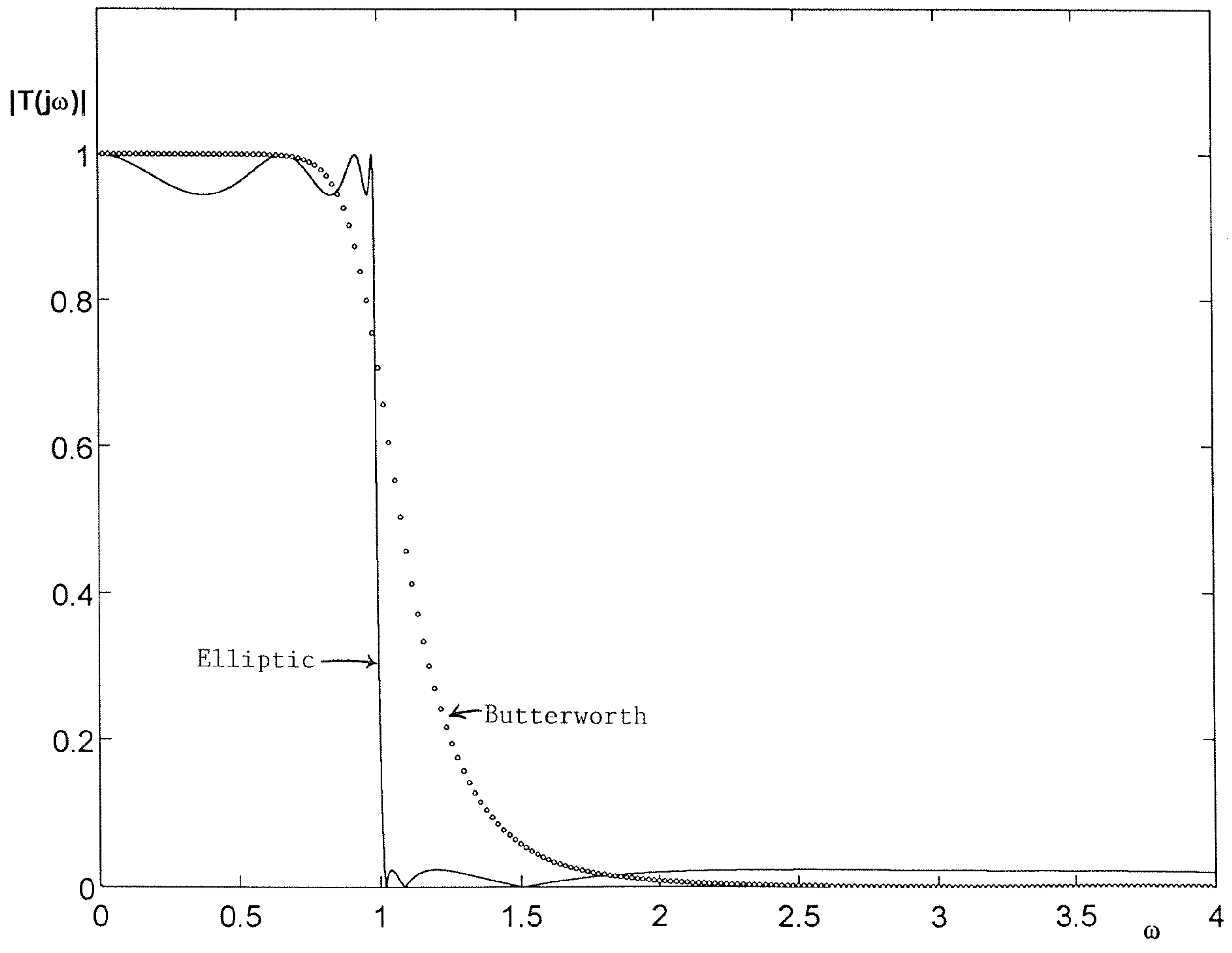
APPLICATION NOTE NO. 334

November 1995

DESIGN AND APPLICATIONS OF ELLIPTIC FILTERS

0. INTRODUCTION

Elliptic (Cauer) filters generally offer the sharpest cutoff of any of the commonly available analog filters. Like inverse Chebyshev low-pass filters (see AN-333), they have finite jω-axis zeros. However, unlike inverse Chebyshev, the passband (as well as the stopband) is allowed to ripple, and this means that the poles can be manipulated to increase the sharpness of cutoff. In particular, a close placement of the highest frequency pole pair and the lowest frequency zero pair results in a precipitous rolloff.



The contest between elliptic and Butterworth is illustrated by Fig. 1. We note that in general terms, Butterworth is a "better" low-pass filter for frequencies from 0 to about 0.8 (flatter response), and for frequencies greater than about 1.8 (greater rejection). In the region immediately around 1, the elliptic is clearly superior. The choice between one or the other is clearly involved, and application dependent. Elliptic is probably a good choice where we can tolerate ripple in the passband (e.g., for telephone speech), where we want a very sharp cutoff, and where we are looking for only a fixed amount of minimum rejection in the stopband.

Like the inverse Chebyshev, the presence of $j\omega$-axis zeros offers us the opportunity to place one zero right on top of some frequency that we may want to reject completely. This can be done even in the absence of easily invertable design equations simply by scaling all the poles and zeros. In addition, the elliptic analog filters are easily converted to digital filters using Bilinear-z. In fact, except for fairly low-order elliptic filters, it is quite difficult to actually realize analog elliptic filters due to sensitivities of the performance characteristics to component tolerances.

1. ANALOG FILTER DESIGN

The design equations for elliptic filters are not simple, and we really need computers to evaluate certain functions by iteration. (The design procedure itself is not iterative however, unlike the iterative "Parks-McClellan" FIR design procedure for similar-looking FIR responses.) Here we have simply taken an available FORTRAN program from Parks and Burris [1] and converted it to MATLAB™ [2] without much difficulty. The new program is given below as a main program and four other MATLAB functions serve in place of the subroutines in the FORTRAN program.

1-a: MAIN PROGRAM AEF.M

```
function [an,ad,az,ap]=aef(WP,WS,R1,R2)
%  function [an,ad,az,ap]=aef(WP,WS,R1,R2)
%
%                ANALOG ELLIPTIC FILTER
%
%    WP and WS in radians
%    R1 = passband ripple in db (e.g., 1 db)
%    R2 = stopband rejection in db (e.g., 20 db down)
%
%     example: [an,ad,az,ap]=aef(0.98,1.02,1,20)
%
%    B. Hutchins                      Fall 1995
%    After FORTRAN Program 9 from Parks & Burrus,
%      Digital Filter Design, Wiley, (1987)
%
%    Requires Functions:  cei, fk, elp, arcsc
```

```
E=sqrt(10^(0.1*R1)-1);
   K=WP/WS;
   KC=sqrt(1-K*K);
   K1=E/sqrt(10^(0.1*R2)-1);
   K1C=sqrt(1-K1*K1);
   KK=cei(KC);
   KKC=cei(K);
   KK1=cei(K1C);
   KK1C=cei(K1);
   XN=KK*KK1C/(KK1*KKC);
   N = floor(XN+1);
   disp('   FILTER ORDER N:');N
   K1=fk(N*KKC/KK);
   K1C=sqrt(1-K1*K1);
   KK1=cei(K1C);

N2=(N+1)/2;
if floor(N/2)==N/2; L=1; else L=0; end
LL=L;
V0=(KK/(KK1*N))*arcsc(1/E,K1);
[SN,CN,DN]=elp(V0,K);
SM=SN;
CM=CN;
DM=DN;
for J=1:N2
   ARG=KK*L/N;
   [SN,CN,DN]=elp(ARG,KC);
   ZR(J)=0;
   if L ~= 0; ZI(J)=WS/SN; end
   PR(J)=-WP*SM*CM*CN*DN/(1-((DN*SM)^2));
   PI(J)= WP*DM*SN/(1-((DN*SM)^2));
   L=L+2;
end
if LL==1;
     ap=[PR+j*PI,PR-j*PI];
     az=[j*ZI,-j*ZI];
end
if LL==0;
     ap=[ PR(1), PR(2:J)+j*PI(2:J), PR(2:J)-j*PI(2:J)];
     az=[ ZI(2:J)*j, -ZI(2:J)*j];
end
an=poly(az);
ad=poly(ap);
w=0:.002:4;
H=freqs(an,ad,w);
H=H/abs(H(1));
figure(1)
plot(w,abs(H))
figure(1)
pause
figure(2)
plot(real(ap),imag(ap),'x')
hold on
zz=zeros(1,length(az));
plot(zz,imag(az),'o')
ymax=ceil(max(abs(az)));
xmax=ceil(max(abs(real(ap))));
axis([-xmax, 0.5, -ymax, ymax]);
grid
hold off
figure(2)
```

1-b: REQUIRED FUNCTIONS

1-b-1: PROGRAM CEI.M

```
function y=cei(KC)
%  function y=cei(KC)
%       Complete Elliptic Integral
% Function is support of aef.m
disp('Function cei.m called')
    A=1;
    B=KC;
    for J=1:20
       AT=(A+B)/2;
       B = sqrt(A*B);
       A=AT;
       if ((A-B)/A) < 1.2e-7; break; end
    end

    y=1.5707963/A;
```

1-b-2: PROGRAM FK.M

```
function y=fk(U)
% function fk(U)
% Function in support of aef.m
disp('Function fk.m  called')
Q=exp(-pi*U);
A=1;
B=1;
C=1;
D=Q;
for J=1:15
     A=A+2*C*D;
     C=C*D*D;
     B=B+C;
     D=D*Q;
     if C < 0.1e-7; break; end
end
y=4*sqrt(Q)*(B/A)^2;
```

1-b-3: PROGRAM ARCSC.M

```
function y=arcsc(U,KC)
%  function y=arcsc(U,KC)
%    Arc elliptic tangent
%  function in support of aef.m
disp('Function arcsc.m called')
A=1;
B=KC;
Y=1/U;
L=0;
for J=1:1:15
     BT=A*B;
     A=A+B;
     B=2*sqrt(BT);
     Y=Y-BT/Y;
       if Y==0; Y=sqrt(BT)*1e-10; end
       if abs(A-B) < (A*1.2e-7); break; end
       L=2*L;
       if Y<0; L=L+1; end
end
if Y<0; L=L+1; end
y=( atan(A/Y) + pi*L )/A;
```

1-b-4: PROGRAM ELP.M

```
function [SN,CN,DN]=elp(X,KC)
% function [SN,CN,DN]=elp(X,KC)
%    Elliptic Function
% function in support of aef.m
disp('Function elp.m called')

if X==0; SN=0; CN=1; DN=1; end

if X~=0;
A=1;
B=KC;

for I=1:20
     AA(I)=A;
     BB(I)=B;
     AT=(A+B)/2;
     B=sqrt(A*B);
     A=AT;
     if ((A-B)/2) < 1.3e-7; break; end
end
C=A/tan(X*A);
D=1;
i=I;
for I=i:-1:1
     E=C*C/A;
     C=C*D;
     A=AA(I);
     D=(E+BB(I))/(E+A);
end
SN=1/sqrt(1+C*C);
CN=SN*C;
DN=D;
end
```

2. CONVERSION TO DIGITAL FILTERS

The design of our IIR digital filter begins with an appropriate analog prototype, to which we will then apply the Bilinear z-Transform. Here we will start with the example 7th-order elliptic filter shown in the example of Fig. 1. Fig. 2 shows the corresponding pole/zero plot for the analog filter. Note that Fig. 2 shows seven poles and six zeros. However, from Fig. 1 we note that there is apparently a seventh zero at infinity since the response is headed back down one final time for high frequencies.

Now, let's assume that we want a digital filter that has a cutoff at a digital frequency $f_d = f_s/5 = 0.2f_s$. This means that the analog prototype for Bilinear z-Transform design should be "prewarped" to an analog frequency:

$$f_a = (f_s/\pi) \tan(\pi f_d/f_s) = 0.2313 f_s \qquad (1)$$

This means that we need to expand the poles and zeros of a normalized analog prototype by 0.2313 f_s. For specificity, let's take f_s=10 kHz so that the desired

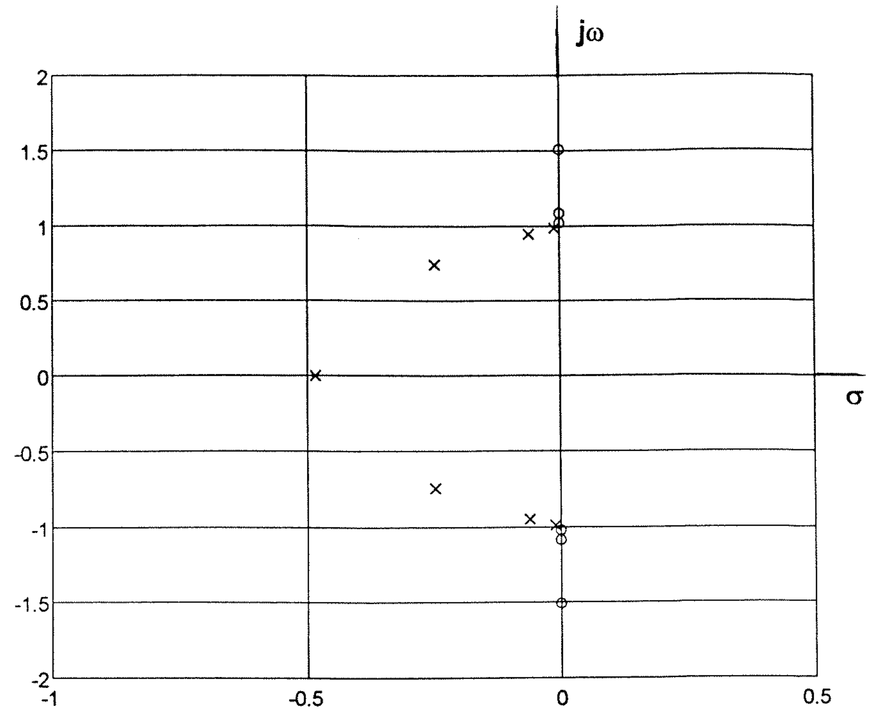


Fig. 2 Pole/Zero Plot of Analog Prototype Elliptic of Fig. 1

cutoff is f_d= 2000 Hz and therefore f_a=2313 Hz. We are now in a position to complete the design by properly expanding the poles/zeros of the analog prototype and then making the usual Bilinear z-Transform substitution:

$$z = (2f_s + s) / (2f_s - s) \tag{2}$$

The one remaining detail is to deal with the "missing" zero at analog infinity. For practical purposes, this can be just a matter of putting a seventh zero at some large number (at 100000 here). The MATLAB design steps follow:

[an,ad,az,ap] = aef(0.985,1.015,0.5,25) (3)

This gives the normalized analog prototype with cutoff at 1. These normalized poles/zeros are expanded for a cutoff at 2313 Hz as follows:

wap = 2313*az/ 2 (4)

waz = 2313*az/ 2 (5a)

waz = [waz 100000] (5b)

where the initial w signifies warped values, in Hertz. The Bilinear z-Transform is now implemented by the function bz.m, previously described in AN-333, and is reprinted here just above the references at the end.

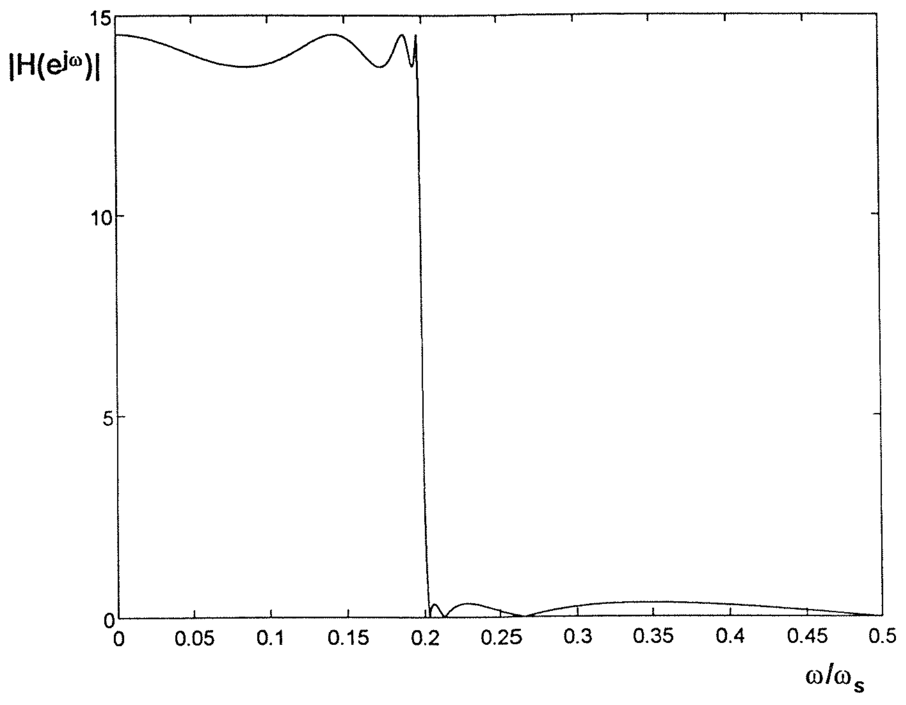


Fig. 3 Magnitude Plot of Digital Elliptic Filter

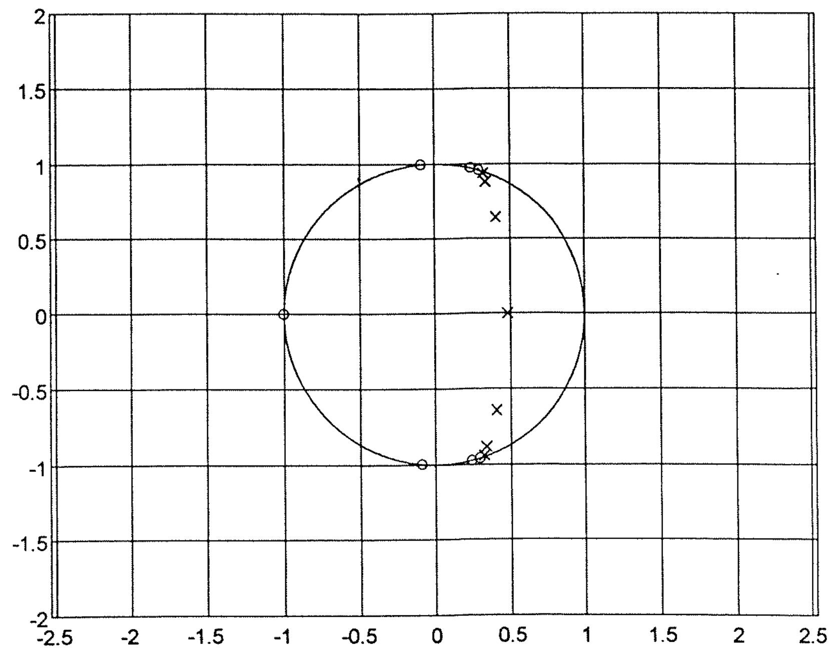


Fig. 4 Pole/Zero Plot of Digital Elliptic Filter

The example digital filter design proceeds with the command line:

[dn,dd,dz,dp]=bz(10000, 2*pi*waz, 2*pi*wap) (6)

Fig. 3 shows the frequency response of the final digital filter. Note that the cutoff is at 2000 Hz = 0.2 f_s, as we desired. Fig. 4 shows the corresponding pole/zero plot. As expected, the desired properties of the analog prototype are carried over to the digital response. Note also that the extra zero which we added at s=100000 has mapped here to z=-1 in the z-plane (0.5 in the frequency response).

3. PROGRAM BZ.M

```
function [dn,dd,dz,dp]=bz(fs,az,ap)
%function [dn,dd,dz,dp]=bz(fs,az,ap)
% convert analog poles/zeros to digital poles/zeros
% az=analog zeros          ap=analog poles
% dz=digital zeros         dp=digital poles
% dn=digital numerator     dd=digital denominator
%
%  Uses  s <- (2/T)(z-1)/(z+1) the usual Bilinear-z substitution
%    transformed to the inverse:
%
%            z = (2/T + s)/(2/T - s)
%
%  Note that 2/T is the sampling frequency fs in Hz while s is
%    thought of in terms of rad/sec.  The two are supposed to
%    be in different units.  If your poles/zeros to be
%    transformed are in Hz, multiply by 2*pi.

%  B. Hutchins                                 Fall 1995

dp=(2*fs + ap)./(2*fs - ap);    % digital poles
dz=(2*fs + az)./(2*fs - az);    % digital zeros
dn=poly(dz);                    % digital numerator
dd=poly(dp);                    % digital denominator
```

References:

[1] T.W. Parks & C.S. Burrus, Digital Filter Design, Wiley (1987), pp 179-184

[2] Anon, The Student Edition of MATLAB, Prentice-Hall (1992)