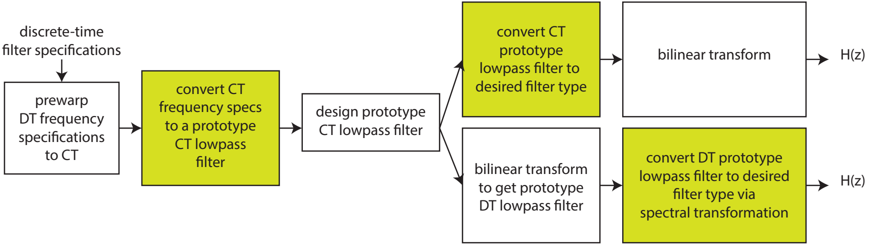
Digital Signal Processing
Complete Bandpass Filter Design Example

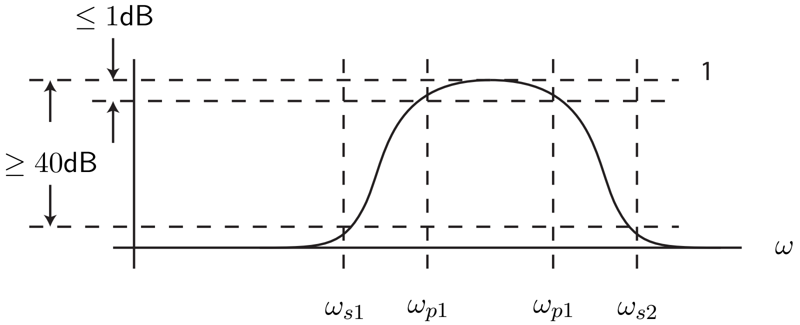
D. Richard Brown III

General Filter Design Procedure



Bilinear Transform Bandpass Filter Design Ex.

Desired discrete-time BPF specifications: $\omega_{p1} = 0.45\pi$, $\omega_{p2} = 0.65\pi$, $\omega_{s1} = 0.3\pi$, $\omega_{s2} = 0.75\pi$, maximum passband ripple 1 dB, minimum stopband attenuation 40 dB.



Step 1: Prewarp to CT Frequencies

We can assume an arbitrary sampling period T_d, so we will choose $T_d = 1$. The following MATLAB code computes the prewarped CT frequencies:

```
% set sampling period
T = 1;

% prewarp frequencies
omega = [0.3 0.45 0.65 0.75]*pi;
Omega_prewarped = (2/T)*tan(omega/2);
```

We get

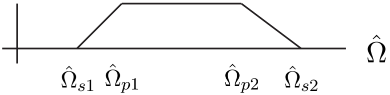
```
  Omega_prewarped =


      1.0191    1.7082    3.2637    4.8284
```

and note that $\hat{\Omega}_{p1}\hat{\Omega}_{p2} = 5.5749 > 4.9204 = \hat{\Omega}_{s1}\hat{\Omega}_{s2}$. We need to adjust one or more band edges so that $\hat{\Omega}_{p1}\hat{\Omega}_{p2} = \hat{\Omega}_{s1}\hat{\Omega}_{s2}$.

Step 2: Precompute Values for Prototype CT LPF

Since we need $\hat{\Omega}_0^2 = \hat{\Omega}_{p1}\hat{\Omega}_{p2} = \hat{\Omega}_{s1}\hat{\Omega}_{s2}$, we can increase $\hat{\Omega}_{s1}$ to shorten the left transition band.



The following MATLAB code makes this correction and also computes a convenient "bandwidth" variable.

```
% we want the two stopband frequencies to be GEOMETRICALLY symmetric
% around the GEOMETRIC center frequency of the passband
Omega_0 = sqrt(Omega_prewarped(2)*Omega_prewarped(3));
Omega_prewarped(1) = Omega_0^2/Omega_prewarped(4);

% compute useful bandwidth variable for later
BW = Omega_prewarped(3)-Omega_prewarped(2);
```

Step 3: Design Prototype CT LPF

We now use the frequency relation between the prototype LPF and the transformed BPF

$$\Omega = -\Omega_p \frac{\hat{\Omega}_0^2 - \hat{\Omega}^2}{\hat{\Omega}\left(\hat{\Omega}_{p2} - \hat{\Omega}_{p1}\right)}$$

to reverse transform the (prewarped and corrected) CT BPF specifications to an appropriate set of prototype CT LPF specifications. This can be accomplished in MATLAB as

```
% create prototype CT LPF filter specs via LPF<->BPF transformation
% (uses even symmetry of magnitude response)
Omega_p = 1; % arbitrary
Omega_s = (Omega_0^2-Omega_prewarped(1)^2)/(Omega_prewarped(1)*BW);

% design prototype CT LPF
% (can also do this by hand as shown in another screencast)
[N,Wn] = buttord(Omega_p,Omega_s,1,40,'s');
[B,A] = butter(N,Wn,'s');
```

At this point, we have a transfer function $H_P(s)$ for a prototype CT LPF.

Step 4: Transform from CT LPF to CT BPF

We have our prototype $H_P(s)$. To get $H_T(s)$ we have to apply the substitution

$$s \to \Omega_p \frac{s^2 + \hat{\Omega}_0^2}{s\left(\hat{\Omega}_{p2} - \hat{\Omega}_{p1}\right)}$$

For notational convenience, let $B = \hat{\Omega}_{p2} - \hat{\Omega}_{p1}$. This transform causes

$$H_P(s) = \frac{b}{\prod_{i=1}^N \left(1 - \frac{s}{\alpha_i}\right)} \to \frac{b}{\prod_{i=1}^N \left(1 - \frac{\Omega_p \frac{s^2+\hat{\Omega}_0^2}{sB}}{\alpha_i}\right)} = \frac{bB^N s^N}{\prod_{i=1}^N \left(Bs - \frac{\Omega_p s^2 + \hat{\Omega}_0^2}{\alpha_i}\right)}$$

Remarks:

- For each pole α_i of the prototype LPF, we now get two poles according to the solutions of
$$Bs - \frac{\Omega_p s^2 + \hat{\Omega}_0^2}{\alpha_i} = 0$$
- Note the addition of N zeros at $s = 0$

Step 4: Transform from CT LPF to CT BPF (MATLAB)

Here we do the transformation manually (see also MATLAB function `lp2bp`)

```
% compute poles of prototype CT LPF
LPFpoles = roots(A);

% compute new pole locations after transformation
BPFpoles = zeros(1,2*N);
index = 1;
for i=1:N,
    tmp = roots([1 -LPFpoles(i)*BW/Omega_p +Omega_0^2]);
    BPFpoles(index) = tmp(1);
    BPFpoles(index+1) = tmp(2);
    index = index+2;
end

% form numerator and denominator polynomial vectors
A_BPF = poly(BPFpoles);
B_BPF = [B(end)*BW^N zeros(1,N)];
```

Step 5: Bilinear Transform from CT BPF to DT BPF

The last step is to take $H_{BPF}(s)$ and perform the substitution

$$\boxed{s \rightarrow \frac{2}{T_d}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

using $T_d = 1$ as decided earlier. MATLAB has a built-in command to do this called `bilinear`. All we need to do is

```
[num,den] = bilinear(B_BPF,A_BPF,T);
```

to get the final discrete-time bandpass filter.

Bilinear Transform Lowpass Butterworth Filter Design Ex.