CUT ONLY (use POSITIVE dB for CUT - THIS IS BAD!!!!!) fix me !

```
function [b,a,h,H] = cp10h(centerHz,octave3dB,gaindB,f_samp,P)
```

CP-10 2nd Order Biquad Filter

The transfer function $H(s)$ of the continuous-time parametric equalizer (2nd order biquad) filter is given by

$$H_B(s) = \frac{s^2 + G\frac{\omega_c}{Q}s + \omega_c^2}{s^2 + \frac{\omega_c}{Q}s + \omega_c^2} \tag{1}$$

for boost, and

$$H_C(s) = \frac{1}{H_B(s)} \tag{2}$$

for cut.

ω_c is 2π times the center frequency of the bell in f_c in Hertz. G is the linear gain at the center frequency, usually in the range 1–10. For cut, G represents the attenuation at the center frequency. Q is the quality factor which controls the width of the boost or cut region. In this paper we will use another parameter $BW = 1.43/Q$, as an approximation to the filter's bandwidth in octaves. $H_B(s)$ and $H_C(s)$, formed by the ratio of two second-order polynomials, are called *biquadratic* transfer functions—'biquads' for short. In the Galileo, we use three parameters to set a single filter $H(s)$. The "high level" parameters are the center frequency f_c, the "Bandwidth" BW, and the Gain (in dB) G_{dB}, where $G = 10^{(|G_{\mathrm{dB}}|/20)}$. If $G_{\mathrm{dB}} >= 0$, then the boost filter $H_B(s)$ is used, if $G_{\mathrm{dB}} < 0$, then the cut filter $H_C(s)$ is used. These three are sometimes knows as the "CP–10" parameters, since they are modeled after a CP–10, an analog parametric EQ designed by MSLI almost 20 years ago. One of the more straightforward ways to realize a discrete-time filter from a continuous-time filter is to use the well-known bilinear transform. The bilinear transform generates a z-domain transfer function from the s-domain prototype by making the substitution

$$s = P\,\frac{z-1}{z+1}.$$

where P is a pre–warping factor.

$$P = \frac{2\pi f_p}{\tan\left(\pi\frac{f_p}{f_s}\right)}$$

where f_p the frequency where the discrete time filter will match most closely the continous time filter and f_s is the sampling frequency. solving for z yields:

$$Y(z) = \frac{\frac{P^2Q+P\omega_c+Q\omega_c^2}{P^2Q+GP\omega_c+Q\omega_c^2} + \frac{-2P^2Q+2Q\omega_c^2}{P^2Q+GP\omega_c+Q\omega_c^2}z^{-1} + \frac{P^2Q-P\omega_c+Q\omega_c^2}{P^2Q+GP\omega_c+Q\omega_c^2}z^{-2}}{1 + \frac{-2P^2Q+2Q\omega_c^2}{P^2Q+GP\omega_c+Q\omega_c^2}z^{-1} + \frac{P^2Q-GP\omega_c+Q\omega_c^2}{P^2Q+GP\omega_c+Q\omega_c^2}z^{-2}}X(z)$$

and using the [DTSP notation]

$$Y(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} X(z)$$

where by tradtion $a_0 = 1$.

fourier transform

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

where $z = e^{j\omega}$.

$$\omega = 2\pi \frac{n}{N}, \quad n = 0, 1, 2, \ldots, N-1$$

This gives a linear constant coefficient differnce equation:

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

[DTSP eq. (5.16) p. 245] since

$$Y(z) + a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) = b_0 X(z) + b_1 z^{-1} X(z) + b_2 z^{-2} X(z)$$

using the inverse $\mathcal{Z}$ transform

$$y[n] + a_1 y[n-1] + a_2 y[n-2] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$$

Now lets get the Impulse response from Partial fraction expansion

$$H(z) = B_0 + \frac{A_1}{1 - \alpha z^{-1}} + \frac{A_2}{1 - \beta z^{-1}}$$

where α, β are roots, and B_1, A_1, A_2 are constants. from long divsion [DTSP p. 115]

$$H(z) = \frac{b_2}{a_2} + \frac{\left(b_1 - \frac{b_2}{a_2} a_1\right) z^{-1} + \left(b_0 - \frac{b_2}{a_2}\right)}{\left(1 - \alpha z^{-1}\right)\left(1 - \beta z^{-1}\right)}$$

$$H(z) = B + \frac{c_1 z^{-1} + c_0}{\left(1 - \alpha z^{-1}\right)\left(1 - \beta z^{-1}\right)}$$

then using simple algebra to solve for A_1, A_2

$$A_1(1 - \beta z^{-1}) + A_2(1 - \alpha z^{-1}) = c_1 z^{-1} + c_0$$

$$A_2 = \frac{\frac{c_1}{\beta} + c_0}{1 - \frac{\alpha}{\beta}}$$

and

$$A_1 = \frac{\frac{c_1}{\alpha} + c_0}{1 - \frac{\beta}{\alpha}}$$

and the roots are from the quadratic equation

$$\alpha, \beta = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2}}{2}$$

(backround defs): δ is the "unit sample sequence" or the "discrete–time impulse respoonse"

$$\delta[n] = \begin{cases} 0, & \text{if } n \neq 0, \\ 1, & \text{if } n = 0. \end{cases}$$

and $u[n]$ is the "unit step sequence"

$$u[n] = \begin{cases} 1, & n \geq 0, \\ 0, & n < 0, \end{cases}$$

Right–Sided Exponential Sequence [DTSP eq. (3.10) p. 98] consider

$$x[n] = a^n u[n]$$

the $\mathcal{Z}$ transform is

$$X(z) = \sum_{n=-\infty}^{\infty} a^n u[n] z^{-n} = \sum_{n=0}^{\infty} (az^{-1})^n = \frac{1}{1 - az^{-1}}$$

$$Y(z) = H(z)X(z)$$

thus, from the inverse $\mathcal{Z}$ transform table, the **impulse response** of a single 2nd order biquad is:

$$h[n] = B\delta[n] + A_1\alpha^n u[n] + A_2\beta^n u[n]$$

so by the inverse $\mathcal{Z}$ transform and the convolution sum

$$y[n] = \sum_{k=-\infty}^{\infty} h[n-k]x[k]$$

```
% from Mathematica and cp10_s_z: (this is CUT equation)

% f_samp = 96e3;
f_nyq = f_samp / 2;
T = 1 ./ f_samp;

%  CP-10  ("biquad") symetric 2nd order parametric filter

% gaindB = 6;  % in dB - ranges from -15dB to +15dB in ? 0.25 dB steps?
% centerHz = 1e3; % ranges from 20Hz to 20KHz in log steps?
% octave3dB = 1; % ranges from 0.1 to 1.1 in ? steps?

fcp10 = logspace(log10(20),log10(20e3),512);
fcp10 = fcp10';

% Pf is prewarped frequency (from Matlab "doc bilinear")
f_p = centerHz;
Pf = (2 .* pi .* f_p) / ( tan(pi .* (f_p / f_samp)));
G = 10^(gaindB / 20);
Q = 1.43 / octave3dB;

W = 2 * pi * centerHz;
b = [ 1 , W/Q , W.^2];
a = [ 1 , G * W / Q , W.^2];
[bz,az] = bilinear(b,a,f_samp,f_p);
Hbzaz = freqz(bz,az,fcp10,f_samp);

s = 2 * pi * fcp10 * i;
Htop = s.^2 + (W/Q) .* s + W.^2;
Hbot = s.^2 + (G .* W/Q) .* s + W.^2;
Hpsm = Htop ./ Hbot;

% from Mathematica
bzmath(1,1) = Pf.^2 .* Q + Pf .* W + Q .* W.^2;
bzmath(1,2) = -2 .* Pf.^2 .* Q + 2 .* Q .* W.^2;
bzmath(1,3) = Pf.^2 .*Q - Pf .* W + Q .* W.^2;

azmath(1,1) = Pf.^2 .* Q + G .* Pf .* W + Q .* W.^2;
azmath(1,2) =  -2 .* Pf.^2 .* Q + 2 .* Q .* W.^2;
azmath(1,3) =  Pf.^2 .*Q - G .* Pf .* W + Q .* W.^2;

bzmath = bzmath ./ azmath(1,1);
azmath = azmath ./ azmath(1,1);

% bzmath =
%      9.854807538490463e-01    -1.953340696279021e+00     9.720511685386910e-01
% azmath =
%      1.000000000000000e+00    -1.953340696279021e+00     9.575319223877373e-01

% Matlab Bilinaer and Mathematica Algebra Match to double precision!
% >> bz - bzmath
% ans =
%      1.110223024625157e-16    -2.220446049250313e-16     1.110223024625157e-16
% >> az - azmath
% ans =
%                          0    -2.220446049250313e-16     2.220446049250313e-16

% from testinversebypartialfraction.m

b0 = bzmath(1);
b1 = bzmath(2);
b2 = bzmath(3);

a0 = azmath(1);
a1 = azmath(2);
a2 = azmath(3);

b = bzmath;
```

```
a = azmath;

% calculationg H(e^{j w}) (Fourier transform)
n = 0:(P-1);
n = n';

z = exp(i .* 2 .* pi .* (n ./ P));
Htop  = b0 + b1 .* z.^(-1) + b2 .* z.^(-2);
Hbot = 1 + a1 .* z.^(-1) + a2 .* z.^(-2);
H = Htop ./ Hbot;

B = b2/a2;
c1 = b1 - ((b2/a2)*a1);
c0 = b0 - (b2/a2);

alpha = (-a1 + sqrt(a1^2 - 4 * a2))/2;
beta = (-a1 - sqrt(a1^2 - 4 * a2))/2;

A1 = ((c1 / alpha) + c0) / ( 1 - (beta / alpha));

A2 = ( (c1 / beta) + c0) / ( 1 - (alpha / beta));

n = 0:(P-1);
n = n';

h(n(1)+1) = B + A1 + A2;
h( n(2:end) + 1,1) = A1 .* alpha .^( n(2:end)) + A2 .* beta.^(n(2:end));
```

NOTE that MATLAB uses $b(1) = b_0$ ACK

from Matlab

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(nb+1)z^{-nb}}{1 + a(2)z^{-1} + \ldots + a(na+1)z^{-na}} X(z)$$

```
% The filter is then:
%
% y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
%                  - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
```

End of MSLI Literate TeX processing on file: cp10h.m
Printed on: Fri Nov 21 16:22:35 PST 2008
git info:
path: Spectral_Convolution/cp10h.m
SHA1: 6f28234b06543345092995f2d45a864f178238fb
mode: 100644
url :
ssh://perrin@crunch8.eng.msli.com:22/home/perrin/GITSERVER/MAPP3DFT
