Audio EQ Cookbook HP15c CE software pac

Pepin Torres, P.E. pepin[dot]torres[at]gmail[dot]com

Introduction:

This software pac calculates all the filters coefficients and related parameters as defined in the classic document Cookbook formulae for audio equalizer biquad filter coefficients.

Who is this for?

This software pac is for any audio enthusiants, audio engineers, or software engineers that need to **generate the coefficients** for any of the 9 different filters in the cookbook (when there is no computer in sight). Furthermore, this pac also **calculates the complex frequency response** of any of the 9 filters at any arbitrary frequency f_x. One use-case would be to generate the coefficients of a filter with center frequency f_0 and calculate the complex response for a set of frequencies f_x (think of being able to plot the response by sampling at different f_x frequencies). Another use-case would be to run this program for a set of filters with different center frequencies f_0 and evaluate each filter at a fixed f_x to understand the gain/phase contributions of each filter at f_x (think of an EQ with multiple bands that overlap f_x).

Main Programs:

Program C – Calculate all filter coefficients

Registers required before running: `R9`, `R.0`, `R.1`, `R.2`

where:

`R9` = Fs (sampling frequency in Hz)

`R.0` = f_0 (center frequency in Hz)

`R.1` = Q (set manually, or run Prog A or Prog B to do so automatically, Q>0)

`R.2` = A (see Prog A, A>0)

where f_0 is the center frequency of the filter (or cutoff frequency for shelf filters)

Registers modified: `R.3`, `R.4`, `R.5`, `R.6`, `R.7`, `R.8`, `R.9`

This program will use the parameters in `R9`, `R.0`, `R.1`, and `R.2` to calculate all the coeffcients for the 9 digital filters defined in the Cookbook. Given the repetition of the a_n coeffcients for the Low-Pass, High-Pass, Band-Pass Q peak, Band-Pass 0dB peak, Notch, and All-Pass filters, the `a0`, `a1`, and `a2` calculation is performed and stored only once.

The output is placed into two matrices, the `A` matrix containing all the a_n coeffcients of size [4-by-3] and the `B` matrix containing all the b_n coeffcients of size [9-by-3].

Example:

R9 = 48KHz = 48,000.0
R.0 = f_0 = 1KHz = 1,000.0
R.1 = Q = 2.8627260504
R.2 = A = 1.0

Key Strokes	Display	Description
GSB C	0.000001330	Stack has no useful data. All results are in Matrix A and Matrix B

Matrices after running Prog C:

<table>
<tr><th></th><th colspan="3">MATRIX B</th><th colspan="3">MATRIX A</th><th></th></tr>
<tr><th></th><th>b0</th><th>b1</th><th>b2</th><th>a0</th><th>a1</th><th>a2</th><th>Type</th></tr>
<tr><td>1</td><td>0.004278</td><td>0.008555</td><td>0.004278</td><td rowspan="6">1.022798</td><td rowspan="6">-1.98289</td><td rowspan="6">0.977202</td><td>LPF</td></tr>
<tr><td>2</td><td>0.995722</td><td>-1.991445</td><td>0.995722</td><td>HPF</td></tr>
<tr><td>3</td><td>0.065263</td><td>0</td><td>-0.065263</td><td>BPF Q</td></tr>
<tr><td>4</td><td>0.022798</td><td>0</td><td>-0.022798</td><td>BOF 0dB</td></tr>
<tr><td>5</td><td>1</td><td>-1.98289</td><td>1</td><td>Notch</td></tr>
<tr><td>6</td><td>0.977202</td><td>-1.98289</td><td>1.022798</td><td>APF</td></tr>
<tr><td>7</td><td>1.022798</td><td>-1.98289</td><td>0.977202</td><td>1.022798</td><td>-1.98289</td><td>0.977202</td><td>Peaking*</td></tr>
<tr><td>8</td><td>2.045595</td><td>-3.965779</td><td>1.954405</td><td>2.045595</td><td>-3.965779</td><td>1.954405</td><td>Low Shelf</td></tr>
<tr><td>9</td><td>2.045595</td><td>-3.965779</td><td>1.954405</td><td>2.045595</td><td>-3.965779</td><td>1.954405</td><td>High Shelf</td></tr>
</table>

For filters 1-6, the a_n coefficients are identical
* For Peaking filter, the a_n coefficients will be different if $A \neq 1$

Register states after running:

R9	F_s
R.0	f_0
R.1	Q
R.2	A
R.3	$\sin(\omega_0)$
R.4	$\cos(\omega_0)$
R.5	α
R.6	A-1
R.7	A+1
R.8	$2\sqrt{A}\alpha$
R.9	BW

Program D – Calculate filter response $H(\omega_x)$ at arbitrary frequecy f_x

Given Matrices `A` and `B` as calculated for center frequency f_0 at sampling rate F_s, program D will calculate the frequency response of the filter at frequency $\mathbf{f_x}$ returned as a complex number on the x-register.

Registers needed: All the registers and matrices from running Program C.
Registers modified: `R6`, `R7`, `R8`

For this you need to look up the row index in **matrix B** corresponding to the filter you want to use. The following table provides this:

Filter	**Coeffiecients Location Row Matrix A**	**Coeffiecients Location Row Matrix B**
Low Pass	1	1
High Pass	1	2
Band Pass Q peak gain	1	3
Band Pass 0dB peak gain	1	4
Notch Filter	1	5
All-Pass Filter	1	6
Peaking	2	7
Low Shelf	3	8
High Shelf	4	9

Example:

Get frequency response for the Bandpass Filter 0dB peak gain (with center frequency f_0 = 1KHz and F_s=48KHz) evaluated at f_x= 840Hz:

t-regsiter	-
z-register	-
y-register	4
x-register	840

Key Strokes	Display	Description
4	4	Row 4 of Matrix B
ENTER	4.0000	
840	840	f_x, Frequency of interest
GSB D	0.4971	Real of $H(\omega_x)$
f (i)	0.4999	Imag of $H(\omega_x)$

Note: Calculator will go into Complex Mode after this program

Running program D results in 3 more matrices being created C, D and E:

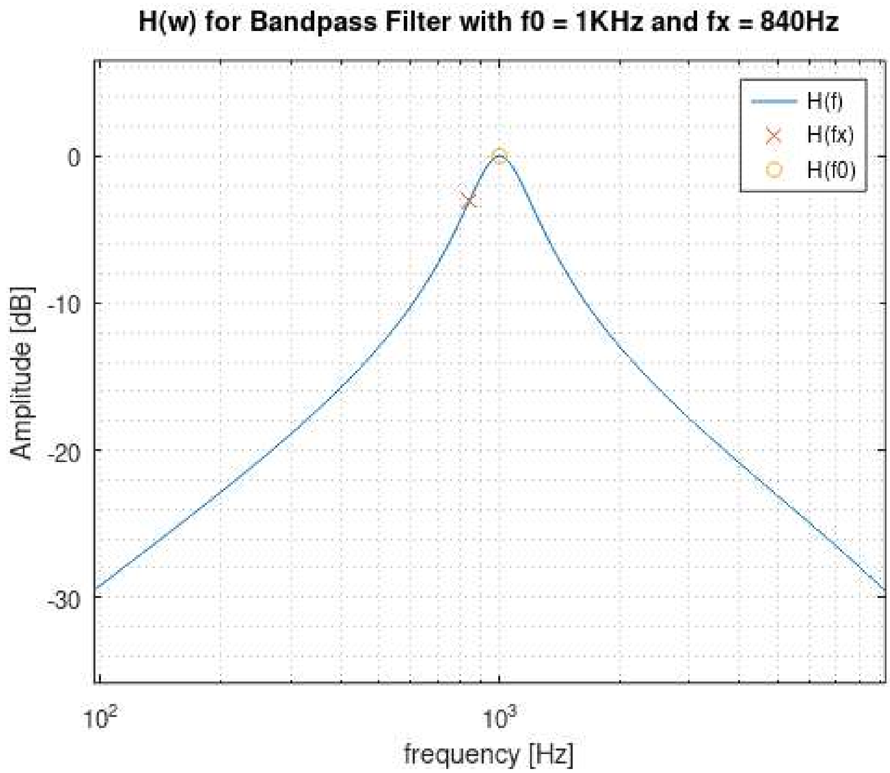
Matrix `C` [3x2] contains the real and imaginary parts of $z=e^{j2\pi fx/Fs}$, z^{-1}, and z^{-2}.
Matrix `D` [4x2] contains the real and imaginary parts of $\Sigma\ a_n z^{-n}$ where `n={0,1,2}`
Matrix `E` [9x2] contains the real and imaginary parts of $\Sigma\ b_n z^{-n}$ where `n={0,1,2}`

```
             Powers of z for frequency fx = 840.0000
             z^0 = (1.00000000,0.00000000)
             z^-1 = (0.99396096,-0.10973431)
             z^-2 = (0.97591676,-0.21814324)

             Intermediate complex vectors.
             E = sum(bn * zx^-n)              D = sum(an * zx^-n)
LPF          E(1) 0.016956 + -0.001872j       D(1) 0.005551 + 0.004421j
HPF          E(2) -0.011954 + 0.001320j       D(2) 0.005551 + 0.004421j
BPF Q        E(3) 0.001572 + 0.014237j        D(3) 0.005551 + 0.004421j
BPF 0dB      E(4) 0.000549 + 0.004973j        D(4) 0.005551 + 0.004421j
NOTCH        E(5) 0.005002 + -0.000552j       D(5) 0.005551 + 0.004421j
ALL-PASS     E(6) 0.004453 + -0.005525j       D(6) 0.005551 + 0.004421j
PEAKING      E(7) 0.005551 + 0.004421j        D(7) 0.005551 + 0.004421j
LOW SHELF    E(8) 0.011102 + 0.008842j        D(8) 0.011102 + 0.008842j
HIGH SHELF   E(9) 0.011102 + 0.008842j        D(9) 0.011102 + 0.008842j

             Frequency response evaluated at fx = 840.0000 [ H(x) =
             E(wx)/D(wx) ]
LPF          H(2*pi*840Hz) = 1.70469312 + -1.69492177j
HPF          H(2*pi*840Hz) = -1.20181890 + 1.19493003j
BPF Q        H(2*pi*840Hz) = 1.42313489 + 1.43133938j
BPF 0dB      H(2*pi*840Hz) = 0.49712577 + 0.49999174j
NOTCH        H(2*pi*840Hz) = 0.50287423 + -0.49999174j
ALL-PASS     H(2*pi*840Hz) = 0.00574846 + -0.99998348j
PEAKING      H(2*pi*840Hz) = 1.00000000 + 0.00000000j
LOW SHELF    H(2*pi*840Hz) = 1.00000000 + 0.00000000j
HIGH SHELF   H(2*pi*840Hz) = 1.00000000 + 0.00000000j
```

The complex-valued $H(\omega_x)$ in the x-register comes from:

```
H(f_x)   =    (Σ b_n·z^-n) / (Σ a_n·z^-n)
         =    (E[idxB,1]+jE[idxB,2])/(D[idxA,1]+jD[idxA,2])
```

Plot of BPF filter with center frequency f_0 and transfer function evaluaed at f_x



Auxilliary Programs:

Program A – Calculate A

Registers modified: `R.1`, `R.2`

This program is used to calculate the $A = 10^{dBgain/40}$ parameter when designing Peaking, Low-Shelf and High-Shelf filters. Put the gain in dB in x-register and press `GSB A`. The A value will be placed in the x-register as well as stored in register `R.2` automatically. If not interested in those filters, put 0.0 in the x-register then press `GSB A` to populate R.2 (or manually store 1.0 in `R.2`).

Note: `R.2` > 0.0 is required by the main Program C.

Example: dBGain = 3dB

Key Strokes	Display	Description
`3`	`3`	Gain in dB
`GSB A`	`1.1885`	A is also stored in `R.2`

After running:

Q is in `R.1`

A is in `R.2`

Program B – Calculate Bandwidth given Q

Registers required before running: `R9`, `R.0`

where:

`R9 = ` Fs (sampling frequency in Hz)

`R.0` = $\omega_0 = 2\pi f_0/F_s$ (angular frequency in rad)

This program is used to calculate the filter bandwidth (in multiples of an octave) given Q, f_0, F_s and Q. It also calculate ω_0 which is used extensively in Program C.

Example:
Filter center frequency: f_0 = 1KHz
Sampling frequency: F_s = 48KHz
Quality Factor: Q = 2.862726050

Key Strokes	Display	Description
`0.5`	`0.5`	Bandwidth (in multiples of an octave)
`GSB B`	`1.0`	

After running:

Q is stored in `R.1`

A is stored in `R.2`

Bandwidth is stored in `R.9` (for reference only)

Extra Mini Programs:

These are two very short programs in the .bin file (but not in the listing file) that convert filter gain to and from dB.

Program 0 – Calculate Linear → dB

This program will take the absolute value of the x-register and do 20log10(abs(x-reg))

Program 1 – Calculate dB → Linear

This program will take the absolute value of the x-register and do 10^(x-reg / 20)

Files included:

Below is a listing of the files included in the package and their SHA256.

```
3f72a6f1ca8649270bee4ca1fefaca92292abbe936cc5a15aa2f24ba661345e6  audio_eq_cookbook.15c
80ab83e487e402f3493d0ffbc7932921eef339068c76729596e633604b703808  audio_eq_cookbook.bin
ff656720283d6aa524ba73588fea86c5bc0eff13eb8c9a59ab67c68f73fd0c89  audio_eq_cookbook.hex
6f43a8fa4ce6195ec19f0e1a038b6923af48e6e2a4a1b486b59e650a1ec8fdab  audio_eq_cookbook.jrn
a077ab03a33a90b114ec45bdb854539d3618bc54d1b2d4c8a036a4868525733d  audio_eq_cookbook.k15
14e75d3ebf8f52c9c462437a46f22628583d05ca5703865ec060324edf45551b  audio_eq_cookbook.m
78b0436cc708674e6de97f937de374dd4c0b9d2cb857df1004330f34b349209a  audio_eq_cookbook.tch
7efa6c698847e7225b5ab6df8746f026a363401810e85d85a00532bc6a37ea61  audio_eq_cookbook.txt
b7e5eccbc65389498ef2039b943de5577fd98f5ca6ef1bf304317c7b3020d3cb  audio_eq_cookbook.xlsx
```

Technical Notes:

Given how much program memory this code needs (528 bytes) and the size of the matrices A thru E (568 bytes), the provided image can only be **run in 15.2 mode** and the author strongly recommends keeping the allocation to the default amount of 19 memory registers. The programs will not run in default 15 mode and allocating more than 19 registers might cause unexpected behavior up to and including freezing (requiring a hard reset.)

Supporting Docs:

audio_eq_cookbook.xlsx – This sheet has the full program listing and the state of the stack for every instruction. It also provides a worked out example for a given set of parameters so users can verify all calculations are correct.

audio_eq_cookbook.m – This Matlab/Octave script calculates all the parameters and is meant to be a way to validate the calculator output. It will plot the filter response and place markers on f_0 and f_x.

```
function [C,Hx,Hsweep] = sanity_check(Fs,f0,BW,A,fx)
```