DESIGN AND IMPLEMENTATION OF FILTERS FOR AMBISONIC DECODERS

Aaron J. HELLER

Artificial Intelligence Center, SRI International
Menlo Park, CA, USA
heller@ai.sri.com

Eric M. BENJAMIN

Surround Research
Pacifica, CA, USA
ebenj@pacbell.net

ABSTRACT

Optimal Ambisonic reproduction requires two different filter types, one to implement dual-band decoding and a second to compensate for the near-field effect of having the loudspeakers close to the listener. We present the theory, design, and implementation of both types of filters. They comprise key components of a full Ambisonic decoder engine written completely in FAUST. The FAUST digital signal processing specification language was selected for the implementation so the filters may be easily incorporated into many different plugins and decoder engines.

1. INTRODUCTION

Ambisonics is a full-sphere, surround sound technique. One of its main advantages is that its transmission format is a loudspeaker-array-independent representation of the soundfield, which is then decoded for reproduction over a particular array. The Ambisonic Decoder Toolbox (ADT) is a MATLAB/Octave toolbox written by the present authors that computes decoder configurations given the geometry of the loudspeaker array and the set of transmission channels in use [1, 2, 3]. It implements three major design techniques, each with a number of variants, including techniques well suited for "difficult" configurations such as domes and stacked rings of loudspeakers.

Initial versions of the toolbox produced configuration files for existing decoder plugins that performed the signal processing. This limited its application to the audio processing platforms, decoder features, ambisonic orders, and numbers of loudspeakers supported by those engines. In order to get around this limitation, we implemented a new, full-featured decoder. We decided to implement it entirely in FAUST, because it could then be used with any audio processing platform that is supported by FAUST. In practice, most users compile the decoders to VST, MaxMSP, Pd, Jack, or Supercollider plugins.

By "full featured" we mean decoders that support all of the signal processing needed for reproduction of *Near-field Compensated High-Order Ambisonics (NFC-HOA)* as described by Daniel [4]. This includes

- Phase-matched, band-splitting filters to enable the use of separate low- and high-frequency decoding matrices, supporting the duplex theory of human directional hearing
- Per-speaker, near-field compensation due to the fact that most Ambisonic decoder design techniques assume the superposition of plane waves, whereas the loudspeakers are actually at a finite distance and radiating spherical waves
- Optional delay and level compensation to accommodate loudspeakers at varying distances from the center of the array

The first two of these features require frequency-dependent processing, while the third is simply a broadband delay and gain adjustment. While the need for these is acknowledged in the literature, there is little design guidance and even fewer correct implementations of the necessary signal processng. We note that in listening tests carried out by the authors, full-featured decoders were always preferred over decoders that use simply a broadband matrix multiplication.

2. PHASE-MATCHED, BAND-SPLITTING FILTERS

Human directional hearing operates over three frequency ranges [5]

- Low (< 800 Hz) using relative time-of-arrival cues (interaural time difference, ITD),
- High ($800 - 5000$ Hz) using relative intensity (interaural level difference, ILD), and
- Very high (> 5 kHz), where pinna effects dominate

The first two are listener-independent and are based on the duplex theory of human directional hearing [6]. Consequently, Ambisonics incorporates different criteria for low-frequency (LF) and high-frequency (HF) reproduction [7, 8, 9]. Specifically, the following are necessary for optimal Ambisonic reproduction

- Constant amplitude gain for all source directions
- Constant energy gain for all source directions
- At low frequencies, correct reproduced wavefront direction and velocity
- At high frequencies, maximum angular concentration of energy in the source direction
- Matching high- and low-frequency perceived directions

See [2, 3, 10, 11] for further details on how these criteria are used to design and evaluate decoders.

In practical terms, this means that the weights of spherical harmonic components for a given loudspeaker are different for low- and high-frequencies, with the transition frequency between 300 and 800 Hz depending on the size and Ambisonic order of the array. This places requirements on both the frequency and phase response of the filters; specifically, the phase response must be the same regardless of the relative gains of the LF and HF sections.

2.1. Design

The key idea is to treat the LF-to-HF transition as one would the crossover network feeding the LF and HF units in a loudspeaker.

We desire a gradual transition, so second-order filters are used

$$LF(s) = \frac{1}{1 + 2sT + (sT)^2} \tag{1}$$

$$HF(s) = \frac{(sT)^2}{1 + 2sT + (sT)^2} \tag{2}$$

These have the -6 dB point at the crossover frequency $\omega = 1/T$ rad/sec. If you combine these, the outputs cancel at the crossover frequency, but reversing the polarity of the HF section makes its phase match that of the LF section and there is no cancellation at the crossover frequency. The output is

$$Total(s) = \frac{1 - (sT)^2}{1 + 2sT + (sT)^2} \tag{3}$$

$$= \frac{(1 + sT)(1 - sT)}{(1 + sT)(1 + sT)} \tag{4}$$

$$= \frac{1 - sT}{1 + sT} \tag{5}$$

which is a first-order, all-pass network. Hence, the phase response is the same as the LF section and is maintained *regardless of the relative levels of the LF and HF sections.*

Applying the bilinear transformation to implement these as digital infinite-impulse response (IIR) filters, the second-order pole

$$H(s) = \frac{1}{1 + 2sT + (sT)^2} \tag{6}$$

becomes

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \tag{7}$$

where, for the LF section

$$b_0 = \frac{k^2}{k^2 + 2k + 1} \tag{8}$$

$$b_1 = 2b_0 \tag{9}$$

$$b_2 = b_0 \tag{10}$$

$$a_0 = 1 \tag{11}$$

$$a_1 = \frac{2(k^2 - 1)}{k^2 + 2k + 1} \tag{12}$$

$$a_2 = \frac{k^2 - 2k + 1}{k^2 + 2k + 1} \tag{13}$$

and, for the HF section

$$b_0 = \frac{1}{k^2 + 2k + 1} \tag{14}$$

$$b_1 = -2b_0 \tag{15}$$

$$b_2 = b_0 \tag{16}$$

with a_0, a_1, a_2 as in the LF section and

$$k = \tan \frac{\pi F_c}{F_s} \tag{17}$$

and F_c is the crossover frequency in Hz and F_s is the sample rate in samples/second.

Note that the denominator is the same for the LF and HF sections, so we implement it as a pair of transposed-direct-form-I filters (Figure 1), where the all-pole section is shared by both filters.[1]

[1] Placing the all-pole section first can cause numeric overflow in fixed-point arithmetic. In that case, we recommend implementing it as two separate filters, with the all-zero section first, followed by the all-pole section.

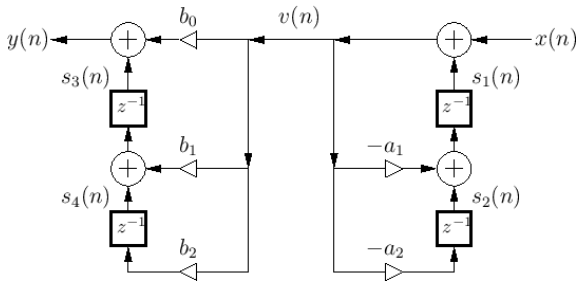
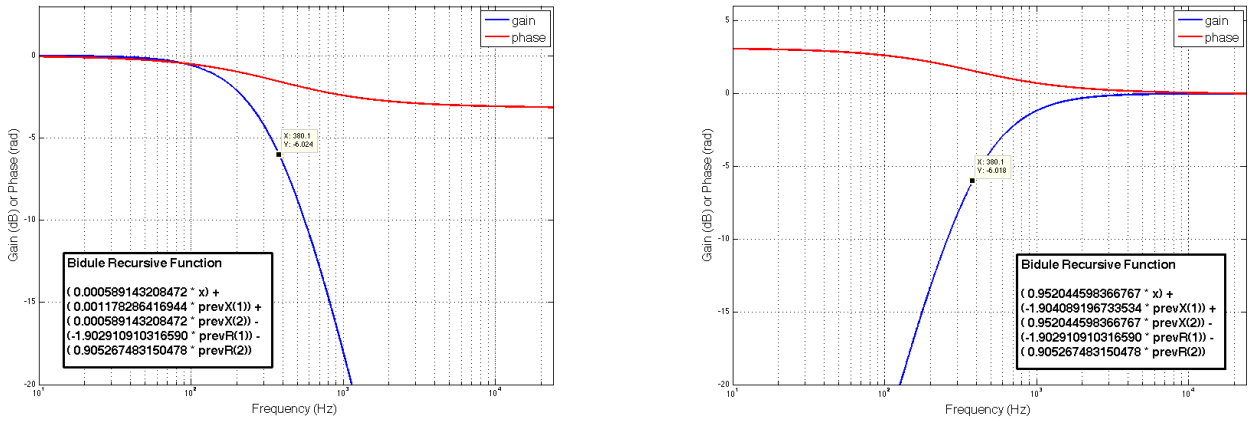


Figure 1: *Transposed-Direct-Form-I implementation of a second-order IIR digital filter. The two-pole section is on the left, followed by a single two-zero section. In our implementation, there are two two-zero sections for low- and high-frequencies respectively. Note that the input signal comes in from the right and the output is on the left. (Diagram taken from [12])*

2.2. Implementation

Writing this in FAUST, note that the equations are transcribed into the FAUST definitions directly, minimizing the possibility of errors. Recall that the desired phase response is obtained by subtracting the output of these sections, so that after scaling according to the desired response the output signals must be differenced, not summed. Figure 2 shows the measured frequency response and phase response of the filters with $F_c = 380$ Hz.

```
xover(freq,n) = par(i,n,xover1) with {
    k = tan(m.PI*float(freq)/m.SR);
    k2 = k^2;
    d = (k2 + 2*k + 1);

    // shared all-pole section
    a = (2*(k2-1)/d, (k2-2*k+1)/d);

    // hf all-zero section
    b_hf = (1/d,-2/d,1/d);

    // lf all-zero section
    b_lf = (k2/d, 2*k2/d, k2/d);

    rsub(x,y) = y-x;
    xover1 = _ : rsub ~ fir(a)
             <: fir(b_lf), fir(b_hf)
             : _, *(-1);
};

// shelf filter
shelf(freq,g_lf,g_hf) = xover(freq,1) :
    gain(g_lf), gain(g_hf):>_;

// fir filter
fir(c) = R(c) :>_ with {
  R((c,lc)) = _<: R(c), (mem:R(lc));
  R(c)      = gain(c);
};

// bus with gains
gain(c) = R(c) with {
  R((c,cl)) = R(c),R(cl);
  R(1)      = _;
  R(0)      = !:0;
  R(float(0)) = R(0);
  R(float(1)) = R(1);
  R(c)      = *(c);
};
```



(a) low-pass filter

(b) high-pass filter

Figure 2: *Frequency and phase response of phase-matched, band-splitting filters with $F_c = 380$ Hz.*

3. NEAR-FIELD COMPENSATION FILTERS

Ambisonics is based on a spherical-coordinate representation of the soundfield. The mathematical formalism arises from solving the Helmholtz wave equation

$$(\nabla^2 + k)p = 0 \tag{18}$$

in spherical coordinates, with p the pressure and $k = 2\pi f/c$. The solution leads to the Fourier-Bessel description of the soundfield [13]

$$p(kr, \theta, \phi) = \sum_{l=0}^{+\infty} j^l j_l(kr) \sum_{m=-l}^{+l} B_{lm} Y_{lm}(\theta, \phi) \tag{19}$$

where B_{lm} are the Ambisonic signals with the time dependency, $e^{-j\omega t}$, implicit. In practice, we truncate the series after some number of terms, so the upper limit on the first sum becomes O, which sets the Ambisonic order of the system. Note that the solutions decompose into the product of an angular part, the real spherical harmonics, $Y_{lm}(\theta, \phi)$, and a radial part, the spherical Bessel functions, $j_l(kr)$. Only the radial part is frequency dependent.

Taking advantage of this decomposition into angular and radial parts, most decoder design techniques assume that the loudspeakers are at an infinite distance and are thus producing plane waves. The decoder then uses separate filters to compensate for the fact that the speakers are at a finite distance and producing spherical waves. Unfortunately, many decoders simply ignore the radial effects completely.

In spherical waves, the pressure gradient arises from both the direction of propagation, as in plane waves, (the real part) as well as the $1/r$ decrease in pressure due to the spreading of energy across the expanding spherical wavefront (the reactive part, normal to the direction of propagation). This second factor is the cause of the near-field effect, manifesting itself as a low-frequency boost and phase shift.[2] Because the higher-degree spherical harmonics represent higher-order spatial derivatives of the soundfield, the frequency at which this effect starts, as well as the rate of change of the phase, increases with degree.

3.1. Design

Our goal is to design filters for each Ambisonic degree that compensate for the near-field effect and, hence, are called *near-field compensation* (NFC) filters. In their seminal paper, Krall and Frink [14] show that, with appropriate variable substitution, the Bessel polynomials are solutions to the radial part of the wave equation.

The Bessel polynomials are solutions to the second-order differential equation

$$x^2 {y_n}'' + 2(x+1){y_n}' - n(n+1)y_n = 0 \tag{20}$$

and are readily obtained from the recurrence relation

$$y_n(x) = (2n-1)xy_{n-1}(x) + y_{n-2}(x) \tag{21}$$

with $y_0(x) = 1$ and $y_1(x) = x + 1$.

Following Daniel's derivation [4], ignoring the broadband delay and $1/r$ attenuation with distance from the source, the transfer function of the near-field effect for an l^{th}-degree Ambisonic signal is the l^{th}-order Bessel polynomial

$$F_l(s) = \sum_{m=0}^{l} \frac{(l+m)!}{(l-m)!m!} \left(\frac{x}{2}\right)^m \tag{22}$$

where $x = c/sr$, c is the speed of sound, and s is the Laplace transform variable. Hence to compensate for the near-field effect, we need filters with the inverse transfer function.

To obtain the frequency response, we let $s = j\omega = j2\pi f$. We note that at a distance, r, from a point source the real and reactive parts of the soundfield are equal at frequency $f = c/2\pi r$, hence the pressure gradient is 3dB too large and is phase shifted $-45°$ [15, p. 311]. To compensate, the realized first-order filter should have the complementary amplitude and phase response.

[2] The familiar proximity effect in directional microphones arises from the same phenomenon.

3.2. Implementation

To implement these filters, Daniel transforms them from low-pass to high-pass filters and uses the bilinear transformation to move from the continuous s-domain to the discrete z-domain. He then factors them into a series of second-order (SOS), direct-form II, IIR filters, also called biquad filter sections. There are two problems with this

- These are high-pass filters with a critical frequency much lower than the sample rate, $F_c << F_s$. Implementing these as digital biquads is ill-conditioned numerically, requiring high-precision filter coefficients and computation.
- An important characteristic of Bessel filters is linear-phase response over the passband of the filter, which preserves the waveform shape. The frequency warping inherent in the bilinear transform destroys the linear-phase characteristics of the analog filter [16, p. 692].

Instead, we implement these as a series of second-order, digital state-variable filters [17], using the high-pass output from each section to effect the low-pass to high-pass transformation and obtain the near-field effect compensating filters. The filter comprises a series of integrators and connected-feedback configurations that implement the continuous s-domain transfer function directly, avoiding numerical precision problems. The block diagram of the FAUST implementation is shown in Figure 3.

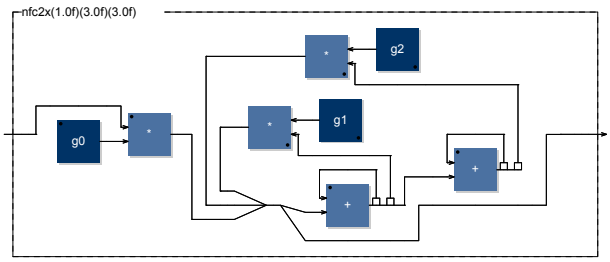


Figure 3: *The block diagram of the digital state-variable filter used to implement the near-field compensation filters. Consult the* FAUST *listing for the calculation of the g_i as a function of distance and sample rate.*

The first step is to generate the Bessel polynomial for a given order (corresponding to the Ambisonic degree of the signal) and then factor it into the product of linear and quadratic polynomials.[3] We have implemented this in Symbolic Python (SymPy), which is convenient for representing and manipulating polynomials and supports arbitrary-precision arithmetic. The package [18] also includes functions that write out the FAUST code that implements the filter for each Ambisonic degree, as well as a FAUST implementation of an NFC filter bank suitable for use on the input of another decoder engine that lacks such filters.

[3]Because the coefficients of the Bessel polynomials are real (in fact, integers), we know, by the fundamental theorem of algebra, that the roots will be real or occur in complex-conjugate pairs. The former give rise to linear factors and the latter, quadratic factors. In practice, it is easier to factor the reverse Bessel polynomial because the leading coefficient is 1, then take the reciprocal of the roots to get the roots of the original polynomial, and compose those into the first- and second-order factors.

The code to transform the polynomial coefficients into the gain parameters for the state-variable filters, as well as the filters themselves, is implemented in FAUST.

```
// second-order state-variable filter
svf2(g0,g1,g2) = _ : *(g0)
    : (((_,_,_):> _ <: +~_, _ ) ~ *(g1)
    : +~_, _) ~ *(g2) : !,_ ;

// first-order NFC filter section
nfc1x(radius,a1) = svf2(g0,g1,0)
  with {
    x = c/(2*float(radius)*SR);
    b1 = a1 * x;
    g0 = 1.0 / (1.0 + b1);
    g1 = 0.0 - (2.0*b1) * g0;
  };

// second-order NFC filter section
nfc2x(radius,a1,a2) = svf2(g0,g1,g2)
  with {
    x = c/(2*float(radius)*SR);
    b1 = a1 * x;
    b2 = a2 * x * x;
    g0 = 1.0 / (1.0 + b1 + b2);
    g1 = 0.0 - (2.0*b1 + 4.0*b2) * g0;
    g2 = 0.0 - (4.0*b2) * g0;
  };
```

The filters are written out by the SymPy program directly in terms of the coefficients of the factored polynomials. The listing shows the fifth-order filter and how it is decomposed into a cascade of two second-order sections followed by a first-order section. The coefficients are computed to 38 digits to support FAUST's quad-precision mode. The frequency and phase responses for $r = 1$ meter are shown in Figure 4. The callouts show that at $f = c/2\pi r = 55.6$ Hz, the amplitude response is $\frac{\sqrt{2}}{2}$ (-3 dB) and the phase response is $45°$, validating the digital realization of the first-order filter. Similar checks were made for the higher order filters.

```
nfc(5, distance) = _ :
 nfc2x(distance,
  4.6493486063632904542320018653568278917,
  18.156315313452237137021293936926965218):
 nfc2x(distance,
  6.7039127983070662860328284984837266881,
  14.272480513279948265230498114959597159):
 nfc1x(distance,
  3.6467385953296432597351696361594454202):
 _ ;
```

Because the required filter response is a function of the distance to the given speaker, if the array includes speakers at different distances, each speaker will need its own NFC filter bank, in general $L \times O$ where L is the number of loudspeakers, and O is the Ambisonic order of the decoder. For example, a 50-loudspeaker dome array supporting sixth-order playback will require 300 individual NFC filter instances. If all of the loudspeakers are at the same distance (or nearly so), the NFC filters can be placed at the input to the decoder, reducing the required number of NFC filters to one per Ambisonic program channel—49 in the previous example, and in general $(O + 1)^2$ in the 3D case and $2O + 1$ in the 2D case. This is controlled by the global variables `nfc_input` and `nfc_output`. The ADT will set these accordingly from the geometry of the loudspeaker array.

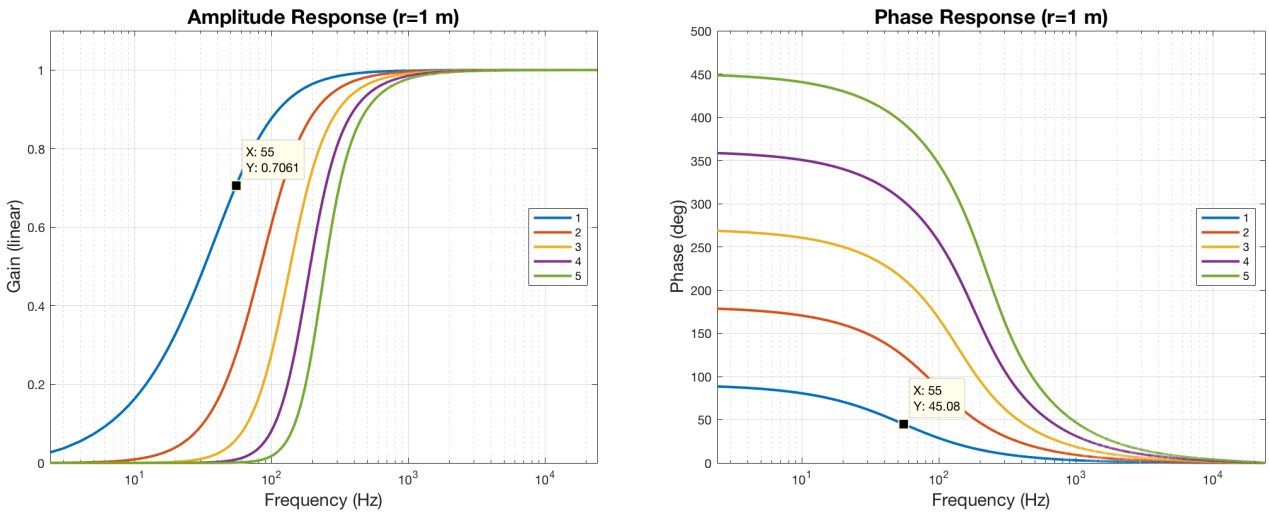


Figure 4: *Measured amplitude and phase response of first- to fifth-order NFC filters for a speaker distance of 1 meters. The callouts show that at* $f = c/2\pi r = 55.6\,Hz$*, the amplitude response is* $\frac{\sqrt{2}}{2}$ *(-3 dB) and the phase response is* $45°$.

Because the number of filter instances can be large, it is important that the implementation is computationally efficient. One of the advantages of FAUST is that it allows the author to focus on the math and signal flow, leaving the task of producing highly-optimized code to the FAUST compiler.

4. DISTANCE AND LEVEL COMPENSATION

In the derivation of the NFC filters, the broadband delay and $1/r$ attenuation with distance were omitted. Best practice is to determine these as part of the overall in-situ calibration process for the loudspeaker array [19]. When that is not possible, the ADT-generated decoders include code that approximates these for free-field (anechoic) propagation. The inclusion of these in the signal processing is controlled by the global variables `delay_comp` and `level_comp`.

```
delay(dc,r)  = R(dc,r) with {
 R(0,r) = _;  // delay_comp off
 R(1,0) = _;  // delay_comp on, zero delay
 R(1,float(0)) = R(1,0);
 R(1,r) = @(meters2samples(r));
 meters2samples(r) = int(m.SR * (float(r)/
     float(c)) + 0.5);
};

level(lc,r,rmax) = R(lc,r,rmax) with{
 R(0,r,rmax) = _;  // level_comp off
 R(1,r,rmax) = *(float(r)/float(rmax));
};

delay_level(r) = R(r) with {
 R((r,rl)) = R(r), R(rl);
 R(r)      = delay(delay_comp,(r_max-r)) :
     level(level_comp,r,r_max);
};
```

where `c` is the speed of sound in meters/sec, `r` is a list of the distances to each loudspeaker in meters, and `r_max` is the largest value of `r` in the list. We note that many widely-used decoder plugins skip this entirely, which can lead to confusing results when some loudspeakers, typically the overhead ones, are somewhat closer to the listener than others. In this case, the precedence effect draws the sound to the nearest loudspeakers [20] and any sense of direction breaks down completely.

5. CONCLUSIONS

We have presented the theory, design, and implementation of appropriate filters for use in decoders for Near-field Compensated High-Order Ambisonics. These are available as free-standing modules, as well as being an integral part of the decoders produced by the Ambisonics Decoder Toolbox. All of the code discussed is open source and is available from The FAUST code, in particular, is distributed under the BSD 3-Clause license to facilitate its integration into and use with other systems.

One key benefit of using FAUST is that we can almost directly transcribe the math into the implementation and then rely on the compiler to produce an efficient realization. This has resulted in code that is easier to maintain and verify. The other key advantage is that a single implementation can target a wide variety of host systems, with the compiler and architecture files handling all the low-level details transparently.

A decade ago, the present authors wrote the paper “Is My Decoder Ambisonic?” [21] in reaction to the poor quality of the Ambisonic decoders available at the time. It discussed the theory, design, and testing of first-order decoders, but did not provide implementations that could be employed directly in widely-used audio processing environments. The current situation is somewhat better, although we still see many poorly-performing or incomplete

decoders in use. The current work extends our previous efforts to high-order Ambisonics and provides implementations that can be used in a wide variety of audio processing systems.

6. ACKNOWLEDGMENTS

The authors thank Fernando Lopez-Lezcano and Natasha Barrett for creating the need for higher-order NFC filters with their large loudspeaker arrays and high-order Ambisonic compositions and Cameron Taylor for discussions and encouragement while writing this paper. The design of the band-splitting filters was originally proposed by Richard Lee. We also thank the anonymous reviewers who made many suggestions that improved this paper.

7. REFERENCES

[1] Aaron J. Heller, "The Ambisonic Decoder Toolbox (ADT)," `https://bitbucket.org/ambidecodertoolbox/adt.git`, accessed 31-March-2018.

[2] Aaron J. Heller and Eric M. Benjamin, "The Ambisonic Decoder Toolbox: Extensions for Partial-Coverage Loudspeaker Arrays," in *Linux Audio Conference 2014*, Karlsruhe, May 2014.

[3] Aaron J. Heller, Eric M. Benjamin, and Richard Lee, "A Toolkit for the Design of Ambisonic Decoders," in *Linux Audio Conference 2012*, Stanford, Mar. 2012, pp. 1–12.

[4] Jerome Daniel, "Spatial Sound Encoding Including Near Field Effect: Introducing Distance Coding Filters and a Viable, New Ambisonic Format ," *Preprints 23rd AES International Conference, Copenhagen*, 2003.

[5] Wikipedia contributors, "Sound localization — Wikipedia, The Free Encyclopedia," 2017, Online, accessed 31-March-2018.

[6] Lord Rayleigh (J W Strutt), "On our perception of sound direction," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 13, no. 74, pp. 214–232, 1907, DOI: 10.1080/14786440709463595.

[7] Michael Gerzon, "Surround-Sound Psychoacoustics," *Wireless World*, vol. 80, no. 1468, pp. 483–486, December 1974, Available from `http://www.audiosignal.co.uk/Gerzon%20archive.html`, online, accessed 1-June-2006.

[8] Michael A. Gerzon, "General Metatheory of Auditory Localisation," in *92nd Audio Engineering Society Convention Preprints*, Vienna, March 1992, number 3306, AES E-lib `http://www.aes.org/e-lib/browse.cfm?elib=6827`.

[9] Eric M Benjamin, Aaron J. Heller, and Richard Lee, "Why Ambisonics Does Work," in *AES 129th Convention*, San Francisco, Nov. 2010.

[10] Michael A. Gerzon and Geoffrey J. Barton, "Ambisonic Decoders for HDTV," in *92nd Audio Engineering Society Convention Preprints*, Vienna, March 1992, number 3345, AES E-lib `http://www.aes.org/e-lib/browse.cfm?elib=6788`.

[11] Davide Scaini and Daniel Arteaga, "Decoding of High Order Ambisonics to Irregular Periphonic Loudspeaker Arrays," in *AES 55th Conference*, Helsinki, Aug. 2014, pp. 1–8.

[12] Julius O. Smith, "Transposed direct-forms," in *Introduction to Digital Filters with Audio Applications*. Online book, 2007 edition, accessed 31-March-2018.

[13] Earl G. Williams, *Fourier Acoustics: Sound Radiation and Nearfield Acoustical Holography*, Academic Press, San Diego, 1999.

[14] H L Krall and Orrin Frink, "A New Class of Orthogonal Polynomials: The Bessel Polynomials," *Transactions of the American Mathematical Society*, vol. 65, no. 1, pp. 100–115, Jan. 1949.

[15] Philip M. Morse and K. Uno Ingard, *Theoretical Acoustics*, chapter 7, Princeton University Press, 1986, ISBN: 0691024014.

[16] John G Proakis and Dimitris G Manolakis, *Digital Signal Processing*, Prentice-Hall International, third edition, 1996.

[17] Julius O Smith, "Digital State-Variable Filters," May 2013, `https://ccrma.stanford.edu/~jos/svf/`, online accessed 31-March-2018.

[18] Aaron J. Heller, "Ambisonic Near-field Compensation Filters," `https://bitbucket.org/ajheller/nfc-filters`, accessed 31-March-2018.

[19] Fernando Lopez-Lezcano, "Searching for the GRAIL," *Computer Music Journal*, vol. 40, no. 4, pp. 91–103, 2016.

[20] Wikipedia contributors, "Precedence effect — Wikipedia, The Free Encyclopedia," 2018, online, accessed 31-March-2018.

[21] Aaron J. Heller, Richard Lee, and Eric M. Benjamin, "Is My Decoder Ambisonic?," *125th Audio Engineering Society Convention, San Francisco*, pp. 1–21, Dec. 2008.