Wavelet-domain convolution for audio localization

Paul Hubbard
phubbard@anl.gov

Joint work with K.L. Umland, M.C. Pereyra, and T.P. Caudell, University of New Mexico

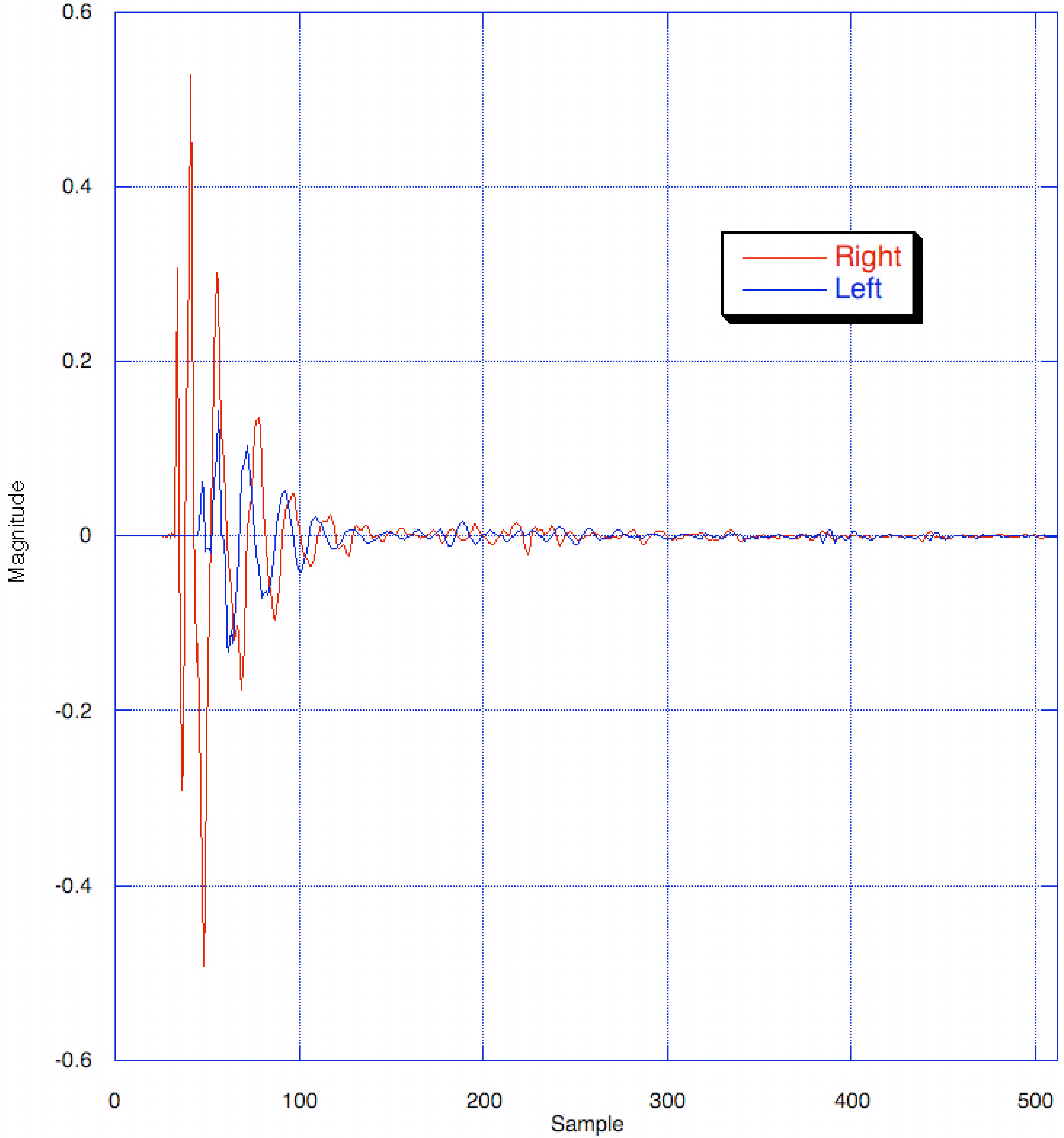
Overview of talk

- Background
  - Introduction to audio localization
  - HRTFs: what and why
  - Motivation
- Non-standard form
- Wavelet-domain convolution
- Results and discussion

Background

- How can you tell where a sound is coming from with your eyes closed?
  - Inter-aural time difference and the all-important pinnae
- HRTF: Head Related Transfer Function
  - Anything convolved with an HRTF will appear to originate where the HRTF was measured. This is audio localization.

Example HRTF: KEMAR, 0 deg elevation, 40 deg azimuth, 'R' set



Localization procedure

- Load (monaural) source audio
- Choose an HRTF based on 3D location relative to the listener
    - Azimuth and elevation; distance via attenuation
- Convolve the source twice; once with the HRTF for each ear
- Example: Piano, 40 degrees right and level

Localization Notes

- Localization is resource intensive
  - Array of HRTFs - hemisphere, sampled every 5 degrees
  - Computation to perform convolution
- Sensitive to time delays
  - 100msec video-audio error budget
- HRTFs vary from person to person
- Results best heard via headphones (crosstalk)

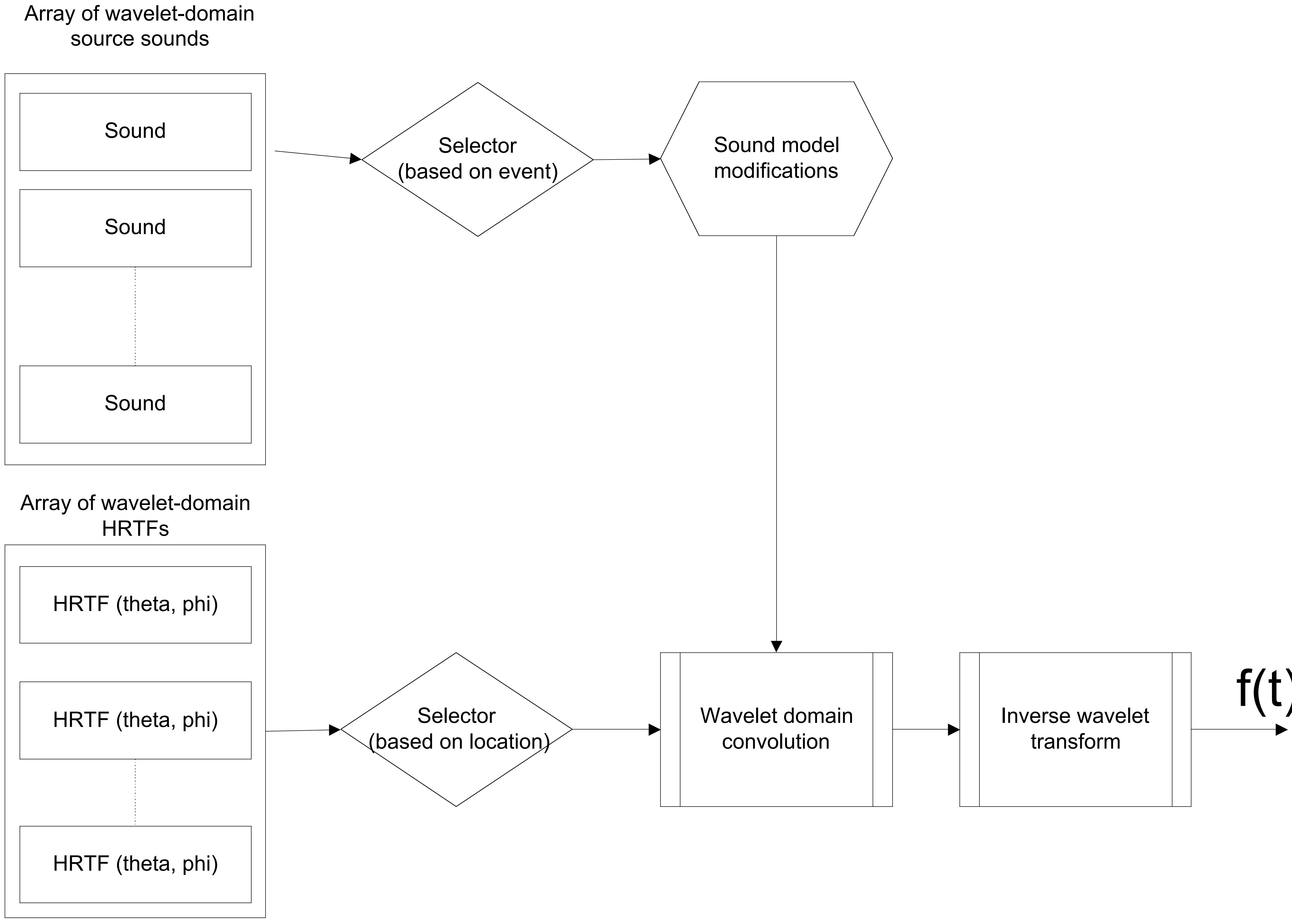
Operators and Matrices

- We can write each HRTF as a circulant matrix, and ‘filter’ by multiplying the matrix by the input audio clip
  - Usually not done this way because matrix-vector multiplication is less efficient than simple convolution
  - Have to deal with circulant wraparound

Why Wavelet-Domain Convolution?

- Existing (Miner) sound synthesis system that operates in the wavelet domain
- What if we could generate *and* localize in the wavelet domain?
  - Avoid extra transformations; this implies less latency and reduced CPU overhead
  - Q: What does convolution as an operator look like in the wavelet domain?

Proposed method

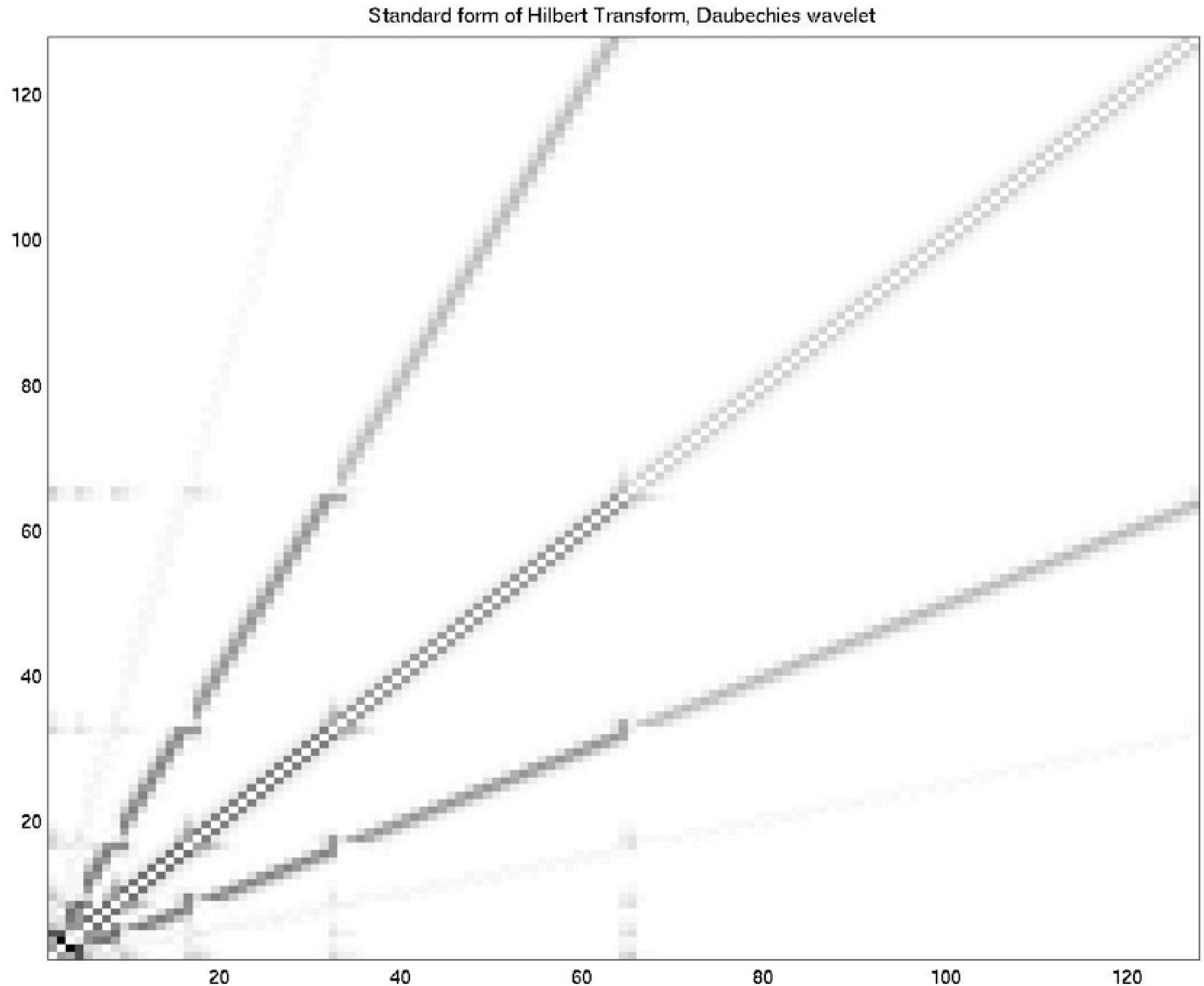


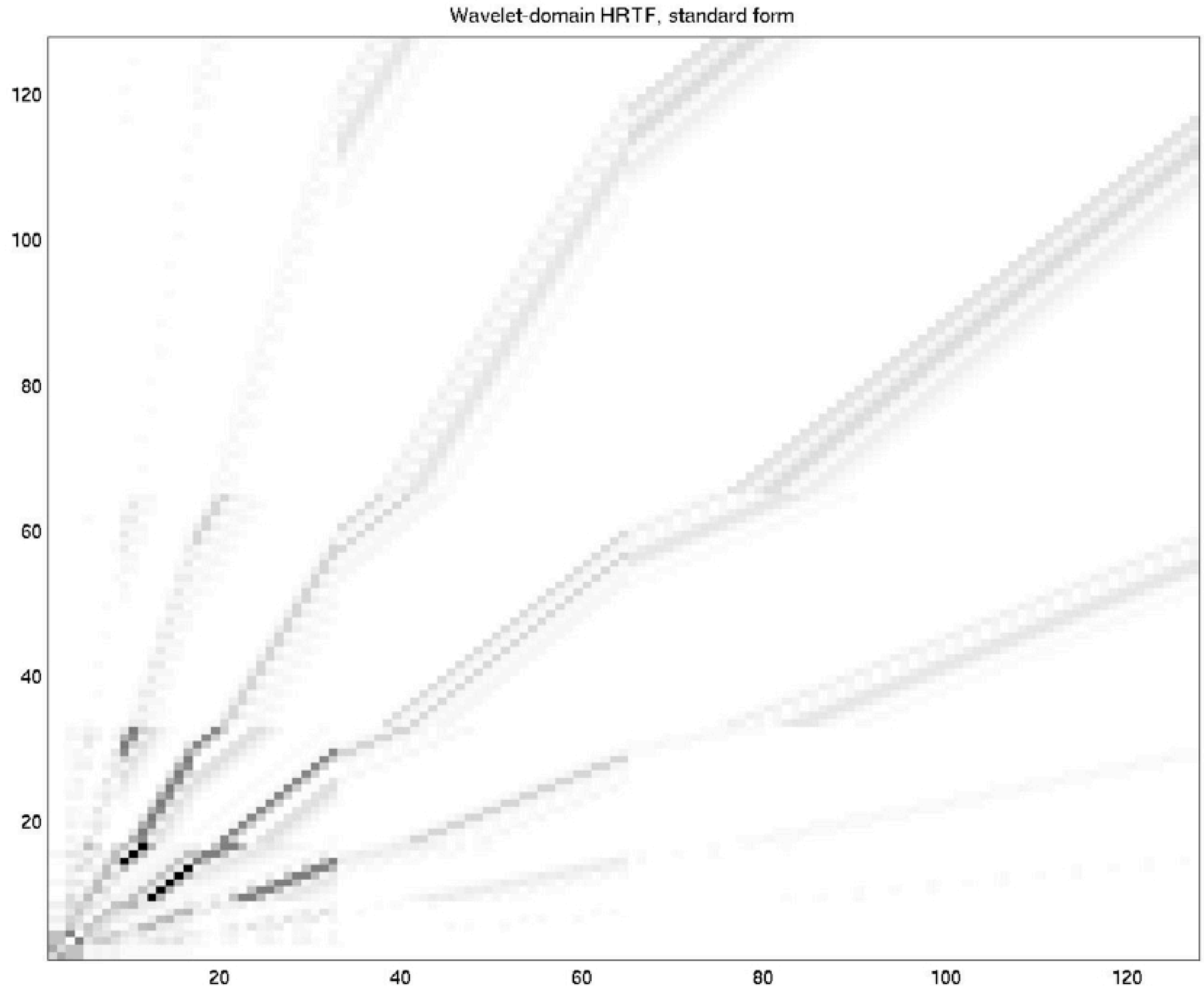
Non-Standard Matrices

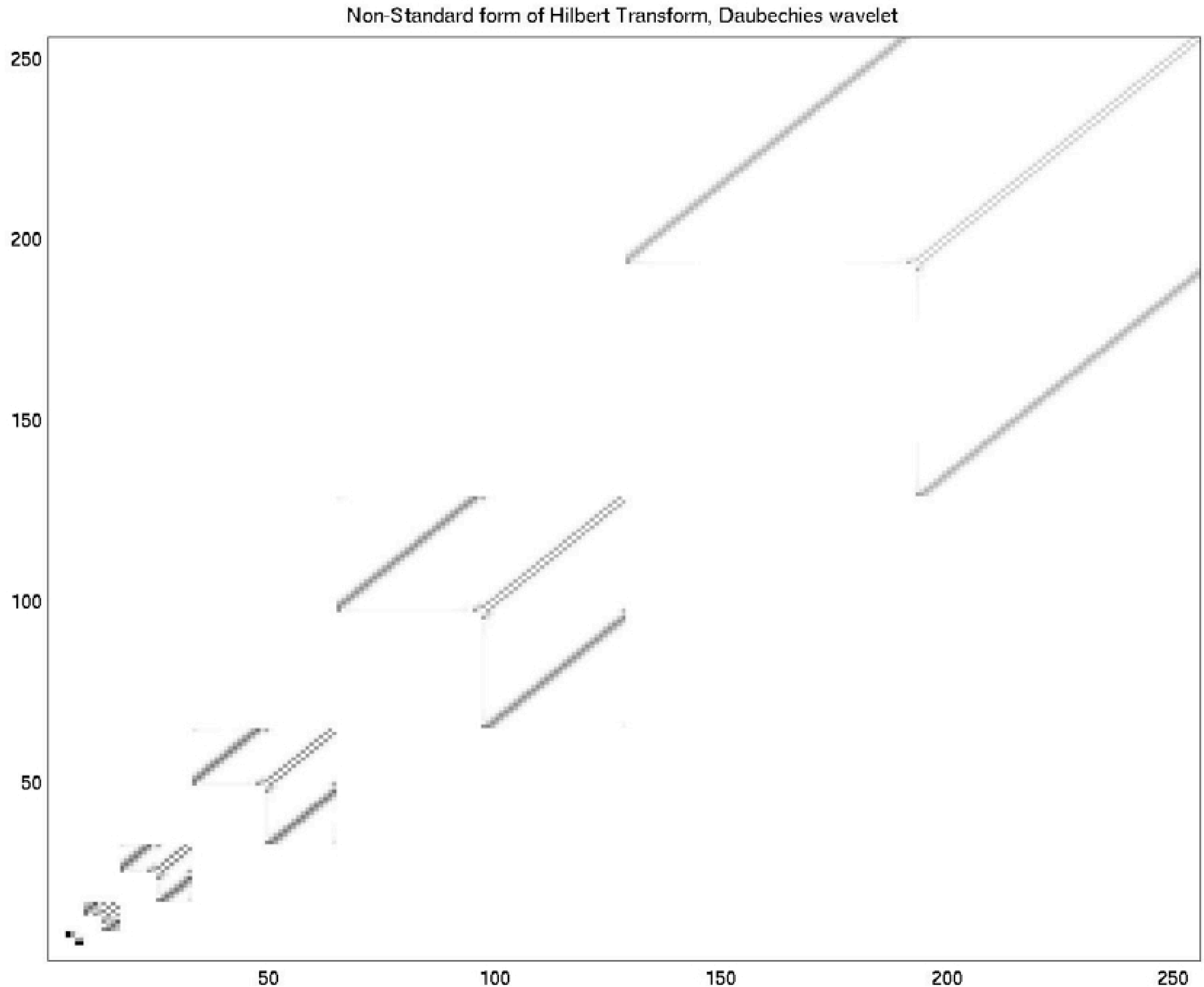
- Beylkin et al derived a method for approximating an operator matrix to an arbitrary degree of accuracy in the wavelet domain, and a fast matrix-vector multiplication to apply it
- If the operator compresses well, potentially more efficient than the FFT. However, matrix area quadruples.
- This presentation is an application of their results.
- See Beylkin's papers for more details
  - (Wavelets, Multiresolution Analysis, and Fast Numerical Algorithms: A draft on INRIA lectures, G. Beylkin)

Non-Standard Form, continued

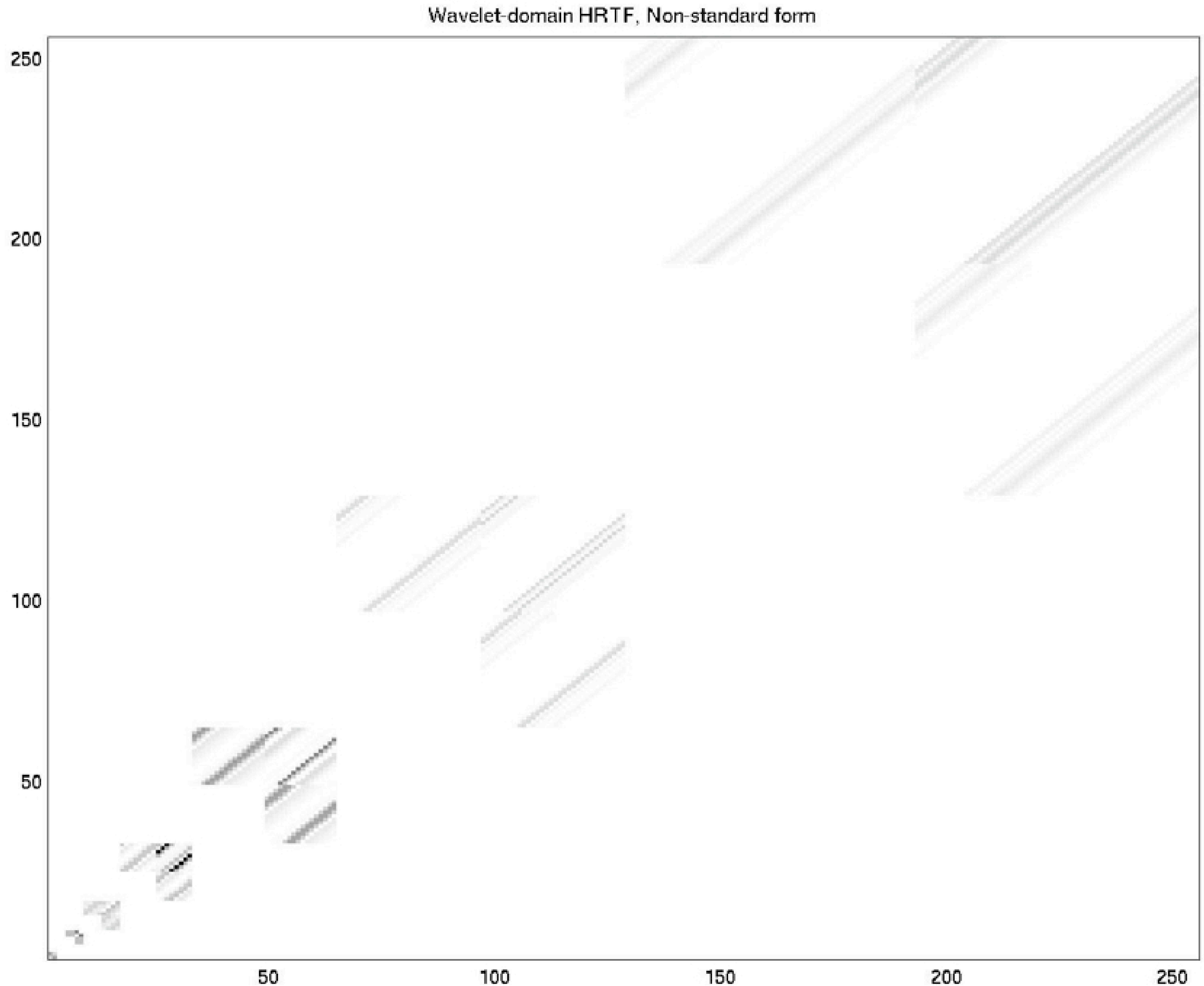
- We are attempting to compress the HRTFs via choice of a suitable wavelet basis and error threshold
- Beylkin et al predict that compressibility will scale with the number of vanishing moments in the wavelet basis.
- Best case: Matrix-vector multiplication requiring O(N) computations


Standard form of Hilbert Transform, Daubechies wavelet
120
100
80
60
40
20
20
40
60
80
100
120


Wavelet-domain HRTF, standard form
120
100
80
60
40
20
20
40
60
80
100
120



Non-Standard form of Hilbert Transform, Daubechies wavelet
250
200
150
100
50
50
100
150
200
250


Wavelet-domain HRTF, Non-standard form
250
200
150
100
50
50
100
150
200
250

Test setup

- For a chosen audio clip, compare exact convolution with lossy wavelet-domain convolution using different basis functions and error thresholds
- Daubechies 4, 20, Coiflet 5, & Beylkin bases
- Not tested: Detail level parameter (discard all scales below parameter k)
- Simple subjective evaluations of results

Results and Evaluation

Wavelet	Epsilon	Percent NZ	Megaflops	Evaluation
Daubechies-4	0.0001	9.80	868.5	Excellent
Daubechies-4	0.0002	8.33	744.1	Passable
Daubechies-4	0.0004	6.05	551.4	Poor
Daub.-20	0.0001	7.83	842.2	Excellent
Daub.-20	0.0002	5.79	669.7	Passable
Daub.-20	0.0004	4.02	520.0	Poor
Coiflet-5	0.0001	7.76	928.7	Excellent
Coiflet-5	0.0002	5.72	756.5	Passable
Coiflet-5	0.0004	4.04	614.1	Poor
Beylkin	0.0001	7.23	765.0	Excellent
Beylkin	0.0002	5.33	607.1	Passable
Beylkin	0.0004	4.34	528.6	Good

A quick demonstration

- Source clip
- Localized via time domain convolution (MATLAB's 'convolve' function)
- Done via wavelet-domain convolution:
  - Daubechies-20, epsilon at 0.0001
  - Daubechies-20, epsilon at 0.0002
  - Daubechies-20, epsilon at 0.0004
  - Beylkin, epsilon at 0.0004
- Note: All used the same (0,40,'R') HRTF

By Way of Comparison

- Reference time-domain convolution required 52.9 megaflops
  - Our *best* result is 520 megaflops, and that with lesser audio fidelity
- Reference convolution also required a lot less memory
  - 4096x4096 matrix, 10^5 entries
- Did not observe much variation in compression as correlated with the number of vanishing moments

Why is it so inefficient?

- These filters (HRTFs) exhibit extremely poor compression; this causes the number of non-zero matrix entries to rise rapidly
- Furthermore, the artifacts of this compression are readily audible even at moderate error thresholds
- Lack of an analytic HRTF hampers our understanding

Other Possible Approaches

- Psycho-acoustic masking
  - Also known as (MP3, AAC, Ogg Vorbis, etc) encoding
  - 10:1 compression is routine, with excellent results
  - Uses knowledge of auditory perception to decide, instead of coefficient magnitudes
- Best basis search to find a basis that has better compression behavior

Closing Remarks

- http://www.phfactor.net/wdconv for code, audio clips, etc
- WD-convolution might be useful for filters with better compressibility