The One Linear Filter to Rule them ALL (TOLFTRTA, pronounced " 'Toll' + 'Frittata'")

The engineering sub discipline of Digital Signal Processing (DSP) is a mature field since it started at the dawn of digital computers. Subsequently, it has a mature ecosystem of canonical textbooks, undergraduate curriculum, computer algorithms, etc, complete with vanity names for everything, from Kaiser Windows to Nyquist. While inevitable, this maturity often tends to make DSP seem more complicated and "magical" then perhaps necessary. Many of the debates of the early days of DSP are no longer relevant. For example, there are hundreds of chapters of textbooks comparing the relative merits of Finite Impulse Response Filters (FIR) versus Infinite Impulse Response filters (IIR). There are chapter upon chapter of filter design methods, again split into FIR methods and IIR methods. Are two 2nd order Biquads in series better or worse than an equripple elliptic filter versus a 23 tap FIR filter?

As a mathematician, one dimensional Linear Time Invariant (LTI) filtering is almost comically simple in the abstract.

Define x[n] as the input signal, sampled at sample rate R. For Audio, thats usually 48,000 samples per second (or 96,000), which because of the excellent work of Nyquist et al, completely and accurately represents and audio signal with a bandwidth of zero to 24kHz (thousand Hertz, or cycles per second).

And any LTI filter can be completely characterized by either its Impulse Response (IR) h[t], or its frequency response H[f]. Either can be transformed into the other by the Fourier Transform.

The output y[n] is then simply the convolution of the input signal x[n] and the impulse response h[t]

$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m].$$

[source: https://en.wikipedia.org/wiki/Convolution
Note f == x and h == g for this notation]

This is textbook stuff. Why repeat it? Well, there is a very famous algorithm called the FFT (Fast Fourier Transform), which efficiently computes the Fourier Transform. It is arguably the most important algorithm in applied mathematics, and high quality and robust FFT algorithms and computer code are available for all computer platforms, speciality DSP chips, Graphical Processing Units (GPU's).

Because we are in the linear world, it is possible to transform the convolution filter equation into the Fourier (frequency) domain.

lets define X[f] = FFT(x[n]), and H[f] = FFT(h[n]), and Y[f] = FFT(y[y])

The convolution equation is then

Y[f] = H[f] * X[f]

which is great since now we have an algorithm that consists entirely of FFTs andmultiplication, which is computationally efficient and obviously simple to describe and analyze.

Why is this so cool? We need one more ingredient. Overlap Save Block Convolution. As written, the convolution equations do not really work in a real-time DSP context. As written, you need all input x before you can calculate the output y. For "offline" processing (think re-mastering an audio tape, etc) it would be fine. But most DSP algorithms are for "real time" applications (music, video, etc). Overlap Save Block convolution decomposes the convolution equation into "blocks" (or "chunks"), and operates on these "chunks", and after every "chunk" outputs a "chunk" of output (y[t]) values. This allows the convolution equation to be used for real-time DSP filtering.

Thus, FFT based Overlap Save block convolution can be used for ANY linear filtering application.

This statement often surprises DSP engineers. They say "ha!, what about IIR filters -- the impulse response is, wait for it INFINITE, and thus the block convolution algorithms don't work." Then they sit back and smile. But not for long.

I say, ok. fine. not all IIR filters can be calculated with block convolution in real time. you got me. I say that to build up their confidence for what comes next.

For most real-world applications, we are only really interested in Stable and Causal IIR filters. Unstable IIR filters have an infinite output for some input. What does that mean? Well, guitar feedback is an example of an unstable LTI system. But (thankfully!) most guitar amps cannot produce infinite sound. At some point they get truncated (or kicked over...).

So, if we restrict ourselves to Stable and Causal linear filters (IIR and FIR), we get a nice property. ALL Stable and Causal LTI linear filters have an impulse response that decays to zero. The DSP engineer gets one last gasp of hope. See, it might not decay fast enough!

And this is why block convolution has been overlooked as an algorithm to implement real-time IIR filters in the real world A 2nd order linear filter (lets take the common "biquad" filter as an example) has a simple recursive structure that maybe takes fifty mathematical operations or 10 lines of computer code to implement. This is efficient. Lets say this particular (low frequency) biquad has an impulse response that slowly decays to zero. But because all computer calculations have a finite precision, all we need to do is to "wait" for the impulse

response to decay below a certain threshold (for single precision floating points its 1e-7, etc). The impulse response might be 10,000 "taps" long.

So, in order implement this IIR filter using FFT based Overlap Save Block Convolution, the algorithm would go something like this.

Take 256 samples of x[n].
Zero Pad to length 16,384
Do length 2^14 FFT(x)
Do length 2^4 FFT(h)
multiply Y[f] = X[f] * H[f]
Do length 2^14 y[t] = IFFT(Y[f])

The DSP engineer then calls mathematicians insane. From the Direct Form II implementation in 100 lines of C code, we have created an algorithm that computes a sixteen thousand point FFT and IFFT every 256 samples. All for a simple 2nd order biquad.

BUT, computers are now VERY,VERY fast. and GPU's are EVEN FASTER. Our desktop computers can easily do the FFT based overlap save block convolution described above, while STILL playing angry birds and running a fancy screen saver.

So now the DSP engineer has to admit that this one algorithm can calculate all linear filters. (Finite Impulse Response Filters, have, wait for it, a finite length IR, and so its obvious even to DSP engineers that overlap save block convolution can perform real time FIR filtering with no conceptual issues). How about Convolution Reverb? Oh, its the same algorithm. No new code needed. "Spectral filtering" -- sure, just change H[f] every chunk size. Done.

So, in the 21st century, why not use this algorithm for all linear filtering, despite it being "less efficient" than the hundreds of named algorithms in the canonical DSP textbooks.

Well, there are good reasons. Since now we are free of the baggage of the canonical DSP textbooks, we can implement any linear filter we want (as long as it is causal and stable, a very mild restriction). We can literally "draw" the frequency response of a filter we want, take the IFFT to find h[t], and bang - use it! (ok, its not quite that simple, but conceptually its that simple).

This brings us to **wavefield synthesis**. Wavefield synthesis utilizes hundreds (or thousands) of discrete transducers to create arbitrary and amazing sound fields. Conceptually we manipulate the magnitude and phase (at every frequency) at every transducer to make cool sound environments. Linear filtering on steroids. Classical textbooks are now useless.

Now we have a way of intelligently using the massive processing power of GPU's. Use the hand-tuned FFT routines from NVIDIA or AMD to turn a GPU into a parallel array of FFT based block convolution filters. Sample synchronicity is easy since we can zero pad all the h[t]'s to the

same length. There are no corner cases. No code branches. No FIR version or IIR version. One (simple) algorithm. The same algorithm running on all cores.

With LTI "solved", we can now completely focus our attention on the fascinating mathematics of wavefield synthesis, and never, ever, open a copy of Oppenheim and Schafer again.

((**THE** canonical textbook for DSP is:
Discrete-Time Signal Processing (3rd Edition) (Prentice-Hall Signal Processing Series) 3rd Edition
by Alan V. Oppenheim (Author), Ronald W. Schafer (Author)))