PYWDF: AN OPEN SOURCE LIBRARY FOR PROTOTYPING AND SIMULATING WAVE DIGITAL FILTER CIRCUITS IN PYTHON

Gustav Anthon, Xavier Lizarraga-Seijas and Frederic Font

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
anthon.gus1@gmail.com | xavier.lizarraga@upf.edu | frederic.font@upf.edu

ABSTRACT

This paper introduces a new open-source Python library for the modeling and simulation of wave digital filter (WDF) circuits. The library, called `pwydf`, allows users to easily create and analyze WDF circuit models in a high-level, object-oriented manner. The library includes a variety of built-in components, such as voltage sources, capacitors, diodes etc., as well as the ability to create custom components and circuits. Additionally, `pywdf` includes a variety of analysis tools, such as frequency response and transient analysis, to aid in the design and optimization of WDF circuits. We demonstrate the library's efficacy in replicating the nonlinear behavior of an analog diode clipper circuit, and in creating an all-pass filter that cannot be realized in the analog world. The library is well-documented and includes several examples to help users get started. Overall, `pywdf` is a powerful tool for anyone working with WDF circuits, and we hope it can be of great use to researchers and engineers in the field.

1. INTRODUCTION

Wave digital filters were initially developed by Alfred Fettweis in the '70s and '80s in order to digitize ladder and lattice circuits [1–3]. They have gained popularity in recent years as interest has grown in virtual analog (VA) modeling of audio and music applications [4,5]. Many analog audio effect circuits are exceedingly rare and/or expensive for the majority of music makers, so making these effects more accessible by faithfully recreating them in the digital domain has become an important goal of audio engineers and developers. [6, 7].

Wave digital modeling is a form of white box VA modeling that takes into account the entirety of a circuit's internal structure. A wave digital model of a circuit is composed by replicating each of the circuit's elements one by one, and connecting them with "adaptors", which inform what type of topology is configured [8]. Series and parallel adaptors of course connect elements in series and parallel and are the most common wave digital adaptors, while polarity inverters can be considered two-port adaptors, and R-type adaptors are used for more complicated topologies [9].

The elements and adaptors are arranged in an *SPQR* tree, with one element at the root and its children elements organized below it [10, 11]. This is done by representing the reference circuit as a graph, in which nodes are circuit nodes and edges are circuit ports. Then a graph decomposition algorithm is performed to yield the SPQR tree. Waves propagate throughout the tree from one wave digital element to the next to simulate the analog circuit.

Circuit elements are discretized locally and are therefore very modular as compared to traditional techniques of physical modeling entire circuits. This allows users to swap out components or change parameters without the need to recompute the entire system's transfer function. This also enables users to reuse element models in multiple circuits, needing only to change parameter values, or occasionally, methods of discretization for stateful components. WDFs work not with Kirchoff variables like voltage and current, but rather wave variables, namely 'incident' and 'reflected' waves at each circuit element's port. All WDF elements accept an incident wave, and propagate a reflected wave as their output[1]. The main work of deriving wave digital models of circuit elements involves computing the reflected wave based on the incoming incident wave.

In this paper we introduce a new open-source Python library for modeling and simulating WDF circuits called `pywdf`. Section 2 provides an overview of related work. Section 3 details the structure of the library and describes basic functionalities. Section 4 offers examples of circuits built with this library.

2. RELATED WORK

Several libraries for implementing wave digital filters tailored to audio circuits exist, notably including: `Faust` framework `wdmodels` [12], and `C++` libraries `chowdsp_wdf` [13] and `RT-WDF` [14]. These libraries are implemented such that the circuits can be reliably run and tested in real time. This however necessitates that a low level language, such as `C++`, or very specific knowledge about `Faust` programming is used to implement the models. The learning curve of `C++` is significantly steeper than that of higher level and less performant languages, which makes the barrier of entry quite high for researchers and engineers to begin experimenting with wave digital filters. This is the central motivation for developing this library; there is not currently a library for modeling wave digital filters in Python, where the process of prototyping and programming is much less difficult. As real time processing of audio in Python results in higher latency, the library is best suited for prototyping, though real time is still possible thanks to frameworks like `pyaudio`[2].

Python is also a programming language that is typically

[1] Some wave digital adaptors accept multiple incident waves and propagate multiple reflected waves, such as R-type adaptors

[2] `https://people.csail.mit.edu/hubert/pyaudio/`

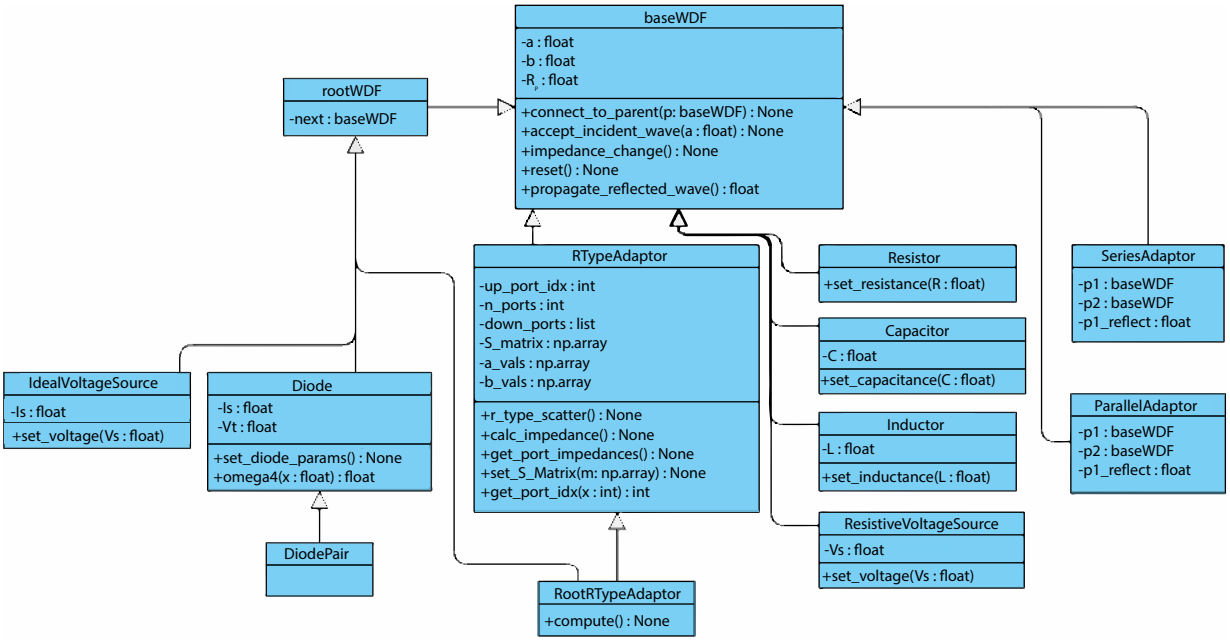

Figure 1: `pywdf` circuit elements UML class diagram

used for research and prototyping as frameworks like `numpy`[3], `scipy`[4], and `matplotlib` [5] allow easy access for plotting and visualizing the behavior of a system. This process is significantly easier than building and rendering a `C++` plugin, opening a DAW session, and configuring the correct channel strip settings to ascertain the same information about the system. Python has also become one of the more popular environments for experimenting with Machine Learning (ML). Recent works such as [15, 16] have united the world of wave digital filters and machine learning, so we believe a WDF library implemented in Python will further facilitate research and development at the intersection of these two subjects.

3. STRUCTURE

The `pywdf` library[6] is built following the object oriented paradigm used in the `C++ chowdsp_wdf` library[7]. The base class from which all wave digital elements and adaptors inherit basic functionalities is called `baseWDF`. This class initializes variables like incident and reflected waves, parent elements, and contains functions to probe and calculate the port resistance at each element and connect elements to one another according to the composition of the *SPQR*-tree representing the circuit. Basic wave digital elements in the repository include resistors, ideal and resistive voltage sources, capacitors and inductors, 3-port series and parallel adaptors and more. Also included is the the diode and diode pair, the nonlinear behavior of which we model using the reflected wave equation derived by Werner et al. in [17]:

$$b = a - 2\lambda V_T \left[\mathcal{W}\left(\frac{R_p I_s}{V_T} e^{\frac{\lambda a}{V_T}} \right) + \mathcal{W}\left(-\frac{R_p I_s}{V_T} e^{-\frac{\lambda a}{V_T}} \right) \right] \quad (1)$$

Where a is the diode pair's incident wave, λ is $\mathrm{signum}(a)$ (defined in equation 2), V_t is thermal voltage, $\mathcal{W}$ is the Lambert W function, R_p is port resistance, and I_s is the reverse bias saturation current.

$$\mathrm{signum}(x) = \begin{cases} -1 & , \quad x < 0 \\ 0 & , x = 0 \\ +1 & , \quad x > 0 \end{cases} \quad (2)$$

The thermal voltage is typically about 25.85 mV at room temperature, but depends on the number of antiparallel diodes used in series, giving us the ability to modify the number of diodes used in the circuit. While the reverse bias saturation current varies from diode to diode, we use an I_s value of 2.52e-9 which is standard for silicon diodes such as the common 1N4148. In practice, we employ the fast approximation of the Lambert W function published by D'Angelo et al. in [18].

Also included are *R*-type adaptors, which allow for implementations of circuits whose topologies cannot be broken down into strictly series or parallel, such as the bridged T circuit. Additionally, they can be used to implement models of operational amplifiers. We include *R*-type adaptors that are unadaptable and can only be used at the root of a connection tree, as well as adaptable ones that can be used more flexibly. The outputs of *R*-type adaptors are computed with scattering matrices, which can be found using methods from Modified Nodal Analysis [19]. The full structure of the library's circuit elements is depicted by the Unified Modeling Language (UML) class diagram in figure 1, and shows how and from where each WDF element inherits its variables and methods.

There also exists a Github repository with a Python script to generate a scattering matrix for a `chowdsp_wdf` circuit given a

[3] `https://numpy.org/`
[4] `https://scipy.org/`
[5] `https://matplotlib.org/`
[6] `https://github.com/gusanthon/pywdf`
[7] `https://github.com/Chowdhury-DSP/chowdsp_wdf`

circuit's netlist [8]. This repository also contains a fork that allows users to generate a scattering matrix[9] compatible with `pywdf` circuits.

Lastly we have created a `Circuit` class from which any wave digital circuit built using this library can inherit basic functionalities. These functions are useful for research and analysis and include:

- `process_sample()` : a function that contains a generic method to process a single sample of a wave digital circuit
- `process_signal()` : uses `process_sample` to process entire signals with a circuit
- `get_impulse_response()` : uses `process_signal()` to process a Dirac delta function and returns the output
- `plot_freqz()` : uses `get_impulse_response()` and takes the Fast Fourier Transform (FFT) to plot the system's magnitude and phase responses
- `plot_freqz_list()` : allows user to visualize how the system's frequency response changes as a parameter is varied. Figure 9 was generated with this function

4. EXAMPLES

In this section we will describe some of the examples offered by the library such as the Diode Clipper and a Passive All Pass Filter. Although the library includes additional circuits such as the RCA Mark II Sound Effects Filter [6] and the Bassman Tone Stack [20, 21], among others.

4.1. Diode Clipper Evaluation

This library was initially developed to thoroughly examine how effectively wave digital filters can replicate the nonlinear behavior of an analog diode clipper [22] . We do so by examining frequency response comparisons, AC transient analysis, and harmonic series analysis. The Diode Clipper WDF model was constructed by converting the circuit to an *SPQR* tree as shown in 3. We then instantiate these components in a `DiodeClipper` class `__init__` function, as shown in listing 1. The parameters used to instantiate these components such as the resistance and capacitance are calculated according to the cutoff value provided by the user.

Listing 1: Instantiating WDF elements of diode clipper

```
self.R1 = Resistor(self.R)
self.Vs = ResistiveVoltageSource()

self.S1 = SeriesAdaptor(self.Vs, self.R1)
self.C1 = Capacitor(self.C, self.fs)

self.P1 = ParallelAdaptor(self.S1, self.C1)
self.Dp = DiodePair(self.P1, 2.52e-9,
    n_diodes=n_diodes)
```

[8] https://github.com/jatinchowdhury18/R-Solver
[9] https://github.com/gusanthon/R-Solver

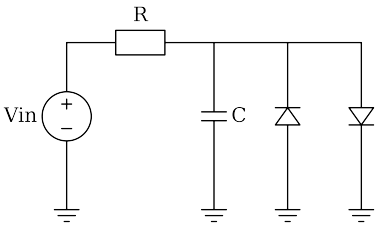

Figure 2: Diode clipper circuit.

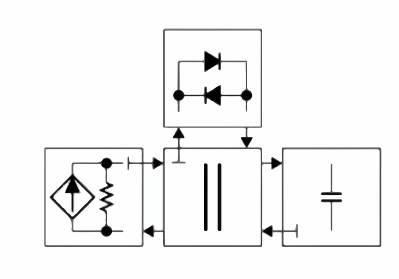

Figure 3: Diode clipper *SPQR* tree.

4.1.1. Frequency Response Comparison

We compare the magnitude and phase responses generated by the `pywdf` circuit model to those of a SPICE[10] circuit model, with the frequency cutoff at the following values:

$$F_c = \{70, 150, 250, 500, 1000, 2000, 4000, 8000, 16000\}[\text{Hz.}] \tag{3}$$

At each of the following sample rates:

$$F_s = \{44100, 48000, 88200, 96000\}[\text{Hz.}] \tag{4}$$

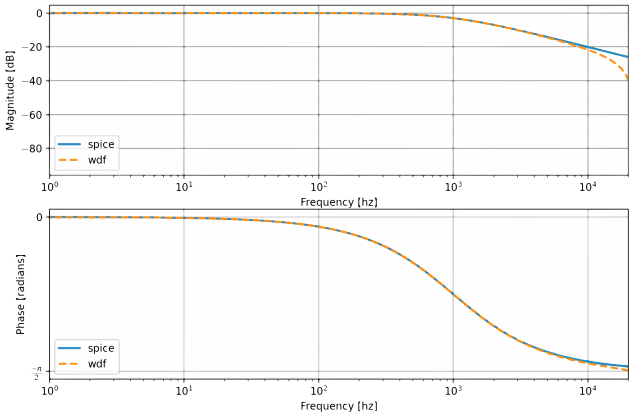

Figure 4: Spice vs `pywdf` frequency response, with 44.1kHz sample rate and cutoff frequency at 1kHz

[10] http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/

Figure 4 shows this frequency response comparison with a sample rate of 44.1 kHz and a cutoff parameter of 1 kHz. We observe only slight deviations as the WDF model's frequency approaches Nyquist - where its behavior is technically undefined. We also compute error metrics between the two models using Mean Square Error (MSE)[11] and Error-to-Signal ratio (ESR).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{5}$$

$$ESR = \frac{\sum_{n=-\infty}^{\infty}|y_p[n] - \hat{y}_p[n]|^2}{\sum_{n=-\infty}^{\infty}|y_p[n]|^2} \tag{6}$$

The MSE and ESR results averaged across all parameter changes for each sample rate are listed in table 1.

	Magnitude [dB]		Phase [rad]	
Sample Rate [Hz]	MSE	ESR	MSE	ESR
44100	7.215	1.703	0.001	0.019
48000	6.837	1.324	0.015	0.015
88200	7.035	0.416	0.003	0.003
96000	6.660	0.344	0.003	0.003

Table 1: Averaged MSE and ESR of magnitude and phase across all parameter changes by sample rate

4.1.2. AC Transient Analysis

A diode clipper typically includes an input gain stage, to raise the level of the input signal and consequently cause it to be clipped even harder. We examine how sinusoidal inputs respond to raising the input gain parameter and show how the signal becomes a square wave in Figure 5. One can observe that even at negative input gain values the signal is being saturated, which implies that the diodes contribute nonlinearities even when the input signal is not crossing the clipping voltage. Table 2 shows the amount of total harmonic distortion and noise (THD+N) introduced at different input gain levels, which also indicates that negative and small input gain values result in additional saturation to the input.

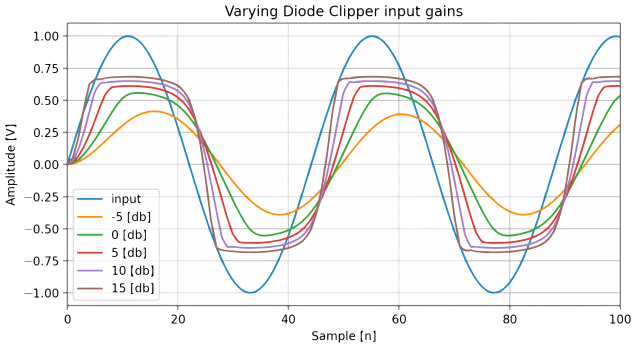

Figure 5: AC transient analysis varying input gains.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html

Input gain [dBFS]	THD+N [%]
-20	0.017
-15	0.017
-10	0.029
-5	0.437
0	6.897
5	15.519
10	21.859
15	26.425
20	29.611
25	31.978
30	33.559
35	35.055
40	37.115

Table 2: THD+N% introduced at each input gain value with 44.1 khz sample rate and cutoff frequency at 1 khz

4.1.3. Harmonic Series Analysis

We also perform a swept sine analysis as first described by Farina in [23], which involves feeding an exponentially swept sine wave across all frequencies below Nyquist into a nonlinear system, and convolving the output with an inverse filter of the input sweep to create a multidimensional impulse response (IR). Because it is a nonlinear system, the diode clipper adds additional frequencies (harmonics) to its input. The multidimensional IR represents the impulse response of each of the harmonics introduced by the system. The multidimensional IR of the diode clipper is shown in 6, with each vertical line corresponding to the IR of each harmonic. We can isolate each individual IR from the multidimensional IR and take its FFT to visualize each harmonic's magnitude response, as shown in figure 7.[12] It is interesting to note that the 0th harmonic has very low magnitude below 20 Hz, which is likely due to the bandwidth of the sweep tone being limited between 20 Hz and 20 kHz. It is also interesting to note that the bandwidth of each successive harmonic decreases, and is essentially high-passed further and further.

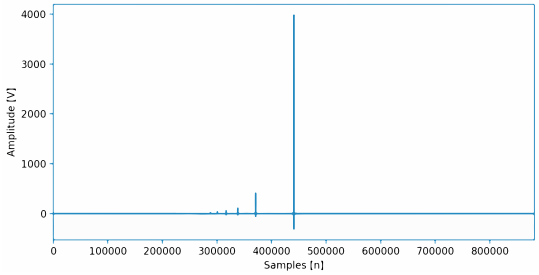

Figure 6: Diode Clipper multidimensional impulse response.

To assess the contribution of each harmonic in terms of the signal, we reused the idea of the Signal-to-Noise Ratio (SNR) in this scenario. It drove us to define the Harmonic-to-Signal Ratio (HSR), that allows us to compare the magnitude level of the 0th-harmonic (the desired signal) to each higher harmonic.

[12] Responses were normalized 0 dB

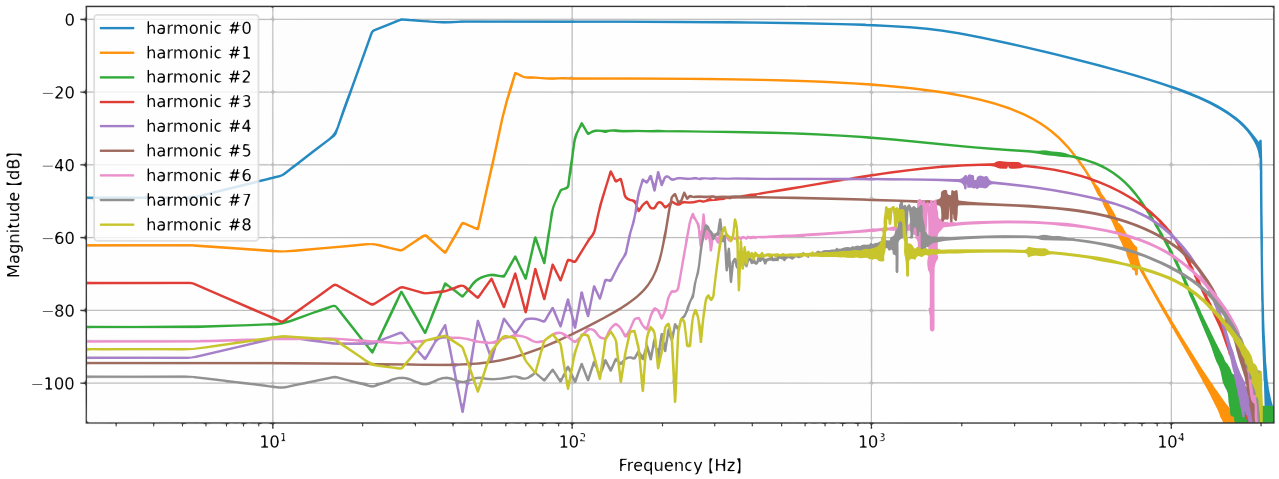

Figure 7: Magnitude response of each harmonic.

# Harmonic	HSR [dB]
0	0.0
1	34.63
2	60.51
3	73.69
4	81.27
5	91.58
6	101.77
7	108.91
8	115.14

Table 3: Harmonic-to-Signal Ratio (HSR) for the first 8 harmonics.

$$HSR_{dB_i} = 2(20\log_{10}(H_i) - (20\log_{10}(H_0))) \quad (7)$$

HSR is defined in Equation 7 [13]. The measurements of each harmonic's HSR can be seen in Table 3. The HSR of the 0th harmonic (fundamental) is of course 0 dB as it is being compared to itself. Each successive harmonic's HSR is greater than its previous, as the difference between the harmonic and the fundamental increases as the harmonic increases.

4.2. All Pass Filter

[24] describes a new resistor-capacitor (RC) circuit realization of a first-order all-pass filter (APF). The implementation is very particular because the APF scheme is composed of a single grounded capacitor, and three resistors, one of which is negative. Negative resistance means acceptance and it is of course not possible in the real world, but is an interesting experiment when simulating circuits digitally. This demonstrates that the digital domain allows us to modify the behavior of analog circuits, for example when each of the resistors are positive the circuit behaves as a high shelving filter.

APF is an important filter function for analog processing design because it provides phase shifting whereas the magnitude is constant at all frequencies, keeping the amplitude of the input constant over the frequency bandwidth of interest. APFs can be used to correct undesired phase change as a result of a processing signal such as the high-pass filtering in a loudspeaker crossover [25].

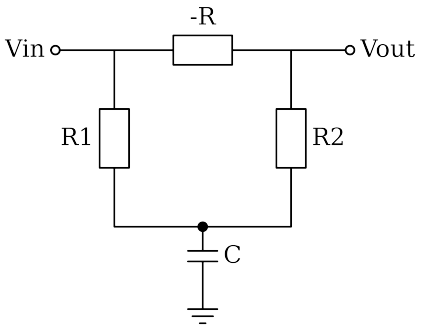

Figure 8: APF circuit.

This model was first implemented in SPICE and then we built the netlist of the R-type adaptor with the approach based on a graph decomposition of the reference circuit [10, 11]. Later we used *R*-Solver to compute the scattering matrix that allows us to calculate the impedance of the R-type adaptor, which was computed without an adapted port. Listing 2 shows the instantiation of each wave digital component in the APF class `__init__` function.

Figure 9 depicts the frequency response for a list of cutoffs, that was generated using the `plot_freqz_list()` command. We can observe how the magnitude is unaltered while the phase is more affected by the circuit with a 180° delay in the frequency components below the cutoff frequency. The phase of the first-order APF varies from 0 at $\omega = 0$ to -π at $\omega = \infty$ and the pole ω_p

[13] `https://wikimedia.org/api/rest_v1/media/math/render/svg/ed42497b9008934f5bcbab43fc64c4d815b142ee`

and zero ω_z frequencies are calculated as [24] indicates,

$$\omega_p = \omega_z = \frac{1}{CR} \tag{8}$$

where C is the capacitance and R the resistance value. Equation 8 enabled us to implement the `set_cutoff` method to modulate the cutoff in our APF-WDF, which may be useful to help fix unwanted phase shifting. It also can be used to implement a phasing/flanging effect for music production applications, which could be achieved by modulating the cutoff frequency with an `LCOscillator`, also included in this library.

Listing 2: Instantiating WDF elements of APF

```
# Port B
self.R1 = Resistor(self.R1_value)

# Port C
self.R2 = Resistor(self.R2_value)

# Port D
self.R3 = Resistor(self.R3_value)

# Port E
self.C1 = Capacitor(self.C1_value, self.fs)

# define R-TypeAdaptor
self.R_adaptor = PolarityInverter(
    RTypeAdaptor([self.R1, self.R2, self.R3,
        self.C1], self.impedance_calc, 0)
)

self.Vin = IdealVoltageSource(self.
    R_adaptor)
```

This APF configuration is unique in that it cannot be realized in the analog domain, because negative resistance cannot be achieved. We hope that `pywdf` can further allow users to experiment with physically impossible circuits in the digital domain, either by aiming to replicate known filter behaviors, or by tweaking existing circuits in ways that otherwise could not be done.

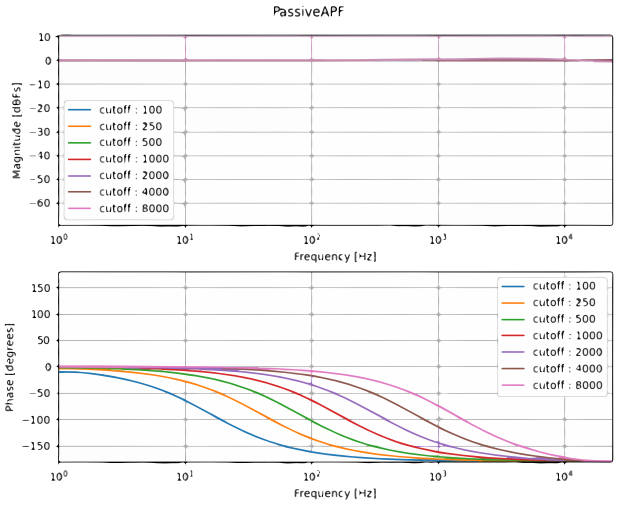

Figure 9: Cutoffs in the APF-WDF implementation.

5. CONCLUSIONS AND FUTURE WORK

This paper presents `pywdf`, an open-source Python library to prototype and evaluate WDFs. The library is available under an MIT license on Github and it provides examples of implementation and usage for different circuits. Currently, the library models each electrical component, such as Resistor, Capacitor, Inductor, Switch, Diode Pair, Adaptors and Sources. We have shown its ease of use in prototyping and analyzing digitally modeled analog circuits, as well as its efficacy in replicating their behavior. Overall, we believe `pywdf` to be a powerful resource with a low barrier of entry to begin experimenting with virtual analog modeling.

Future work for this project can involve developing additional circuit elements and models. For example, existing literature discusses nullors, which are helpful in modeling ideal operational amplifiers and transistors [26]. This would allow for much greater flexibility in building circuits with the library. We also hope to add support for differentiable wave digital filters, and improved integration with solving R-type adaptors' scattering matrices.

Further, while we have presently implemented a tolerance parameter for the Capacitor model, we would like to add this to more circuit elements to get more realistic modelling, and include analysis options on tolerance behavior such as is described in [27].

Additionally, stateful components like capacitors and inductors are only currently implemented as discretized by the bilinear transform. It would be helpful to add support for additional conformal maps such as the forwards and backwards Euler transforms and others, which are included as parameters for stateful components in `chowdsp_wdf`.

6. ACKNOWLEDGEMENTS

This research was carried out under the project Musical AI - PID2019- 111403GB-I00/AEI/10.13039/501100011033, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación. Many thanks to the great number of anonymous reviewers! The authors would also like to thank Xavier Serra and Jatin Chowdhury for sharing their help and knowledge throughout the process.

7. REFERENCES

[1] A. Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

[2] Alfred Fettweis, "Pseudo-passivity, sensitivity, and stability of wave digital filters," *IEEE Transactions on Circuit Theory*, vol. 19, no. 6, pp. 668–673, 1972.

[3] Alfred Fettweis and K Meerkotter, "On adaptors for wave digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 6, pp. 516–525, 1975.

[4] Giovanni De Sanctis and Augusto Sarti, "Virtual analog modeling in the wave-digital domain," *IEEE transactions on audio, speech, and language processing*, vol. 18, no. 4, pp. 715–727, 2009.

[5] Mattia Verasani, Alberto Bernardini, and Augusto Sarti, "Modeling Sallen-Key audio filters in the Wave Digital domain," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*, March. 2017, pp. 431–435.

[6] Kurt James Werner, Ezra J. Teboul, Seth Cluett, and Emma Azelborn, "Modeling and Extending the Rca Mark Ii Sound Effects Filter," in *Proceedings of the 25-th Int. Conf. on Digital Audio Effects (DAFx20in22)*, G. Evangelista and N. Holighaus, Eds., Sept. 2022, vol. 3, pp. 25–32.

[7] B Psenicka, Francisco J García-Ugalde, and Andrés Romero Mier y Terán, "Synthesis of the low-pass and high-pass wave digital filters.," in *ICINCO-SPSMC*, 2008, pp. 225–231.

[8] Augusto Sarti and Giovanni De Sanctis, "Systematic methods for the implementation of nonlinear wave-digital structures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 2, pp. 460–472, 2008.

[9] Kurt James Werner, Vaibhav Nangia, Julius O. Smith, and Jonathan S. Abel, "A general and explicit formulation for wave digital filters with multiple/multiport nonlinearities and complicated topologies," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2015, pp. 1–5.

[10] D. Franken, Jörg Ochs, and Karlheinz Ochs, "Generation of wave digital structures for networks containing multiport elements," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, pp. 586 – 596, 04 2005.

[11] D. Franken, J. Ochs, and K. Ochs, "Generation of wave digital structures for connection networks containing ideal transformers," in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03.*, 2003, vol. 3, pp. III–III.

[12] Dirk Roosenburg, Eli Stine, Romain Michon, and Jatin Chowdhury, "A Wave-Digital Modeling Library for the Faust Programming Language," June 2021, Zenodo.

[13] Jatin Chowdhury, "chowdsp_wdf: An advanced c++ library for wave digital circuit modelling," *arXiv preprint arXiv:2210.12554*, 2022.

[14] Maximilian Rest, W. Ross Dunkel, Kurt James Werner, and Julius O. Smith III, "Rtwdf—a modular wave digital filter library with support for arbitrary topologies and multiple nonlinearities," in *International Conference on Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, 09/2016 2016.

[15] Jatin Chowdhury and Christopher Johann Clarke, "Emulating diode circuits with differentiable wave digital filters," in *Proceedings of the 19th Sound and Music Computing Conference. Zenodo, Saint-Etienne, France*, 2022, pp. 2–9.

[16] Champ C. Darabundit, Dirk Roosenburg, and Julius O. Smith III, "Neural Net Tube Models for Wave Digital Filters," in *Proceedings of the 25-th Int. Conf. on Digital Audio Effects (DAFx20in22)*, G. Evangelista and N. Holighaus, Eds., Sept. 2022, vol. 3, pp. 153–160.

[17] Kurt Werner, Vaibhav Nangia, Alberto Bernardini, Julius Smith, and Augusto Sarti, "An improved and generalized diode clipper model for wave digital filters," 10 2015.

[18] Stefano D'Angelo, Leonardo Gabrielli, and Luca Turchet, "Fast approximation of the lambert w function for virtual analog modelling," 09 2019.

[19] Kurt Werner, Vaibhav Nangia, Julius Smith, and Jonathan Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," 11 2015.

[20] David T. Yeh and Julius Orion Smith, "Discretization of the '59 Fender Bassman tone stack," in *Proc. Int. Conf. Digital Audio Effects (DAFx-06)*, Sept. 2006.

[21] Jatin Chowdhury, "Wave Digital Circuit Models with R-Type Adaptors — jatinchowdhury18.medium.com," https://jatinchowdhury18.medium.com/wave-digital-filter-circuit-models-with-r-type-adaptors-39ad0ad658ce, Oct. 2022.

[22] Gustav Anthon, *Evaluation of Nonlinearities in a Diode Clipper Circuit based on Wave Digital Filters*, Ph.D. thesis, Universitat Pompeu Fabra, Sept. 2022.

[23] Angelo Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," *Journal of the Audio Engineering Society*, pp. 1–24, Feb. 2000.

[24] Norbert Herencsar, Jaroslav Koton, Kamil Vrba, Shahram Minaei, and Izzet Cem Göknar, "Voltage-mode all-pass filter passive scheme based on floating negative resistor and grounded capacitor," in *2015 European Conference on Circuit Theory and Design (ECCTD)*, 2015, pp. 1–4.

[25] Stanley P Lipshitz and John Vanderkooy, "In-phase crossover network design," *Journal of the Audio Engineering Society*, vol. 34, no. 11, pp. 889–894, 1986.

[26] Kurt James Werner, "Virtual analog modeling of audio circuitry using wave digital filters," Dissertation, Stanford University, Stanford, 12/2016 2016, .

[27] Jatin Chowdhury, "Bad Circuit Modelling Episode 1: Component Tolerances — jatinchowdhury18.medium.com," https://jatinchowdhury18.medium.com/bad-circuit-modelling-episode-1-component-tolerances-3ffdbe4e980c.