AI Term Project Report

Eric Wilson
Advanced Radar Research Center
The University of Oklahoma
Norman, Oklahoma, United States
Eric.J.Wilson-1@ou.edu

—, —,
–
—
—
—

***Abstract*—This class project aims to implement a machine learning based method of designing electrical analog filters using an unbalanced-ladder topology. The AI method developed is compared to standard filter prototypes using G-coefficients and element scaling.**

***Keywords*—AI in Analog Design, Filter Synthesis, G-coefficients, Filter prototypes, Element Scaling**

I. Description of Problem

Analog filters are used in many electronic devices to block unwanted power at specific frequencies from entering the system. Cell phones, radios and radar systems all use analog filters in the front end to prevent out of band frequencies from compressing sensitive receive electronics (leading to degraded performance). There are some closed form methods for computing filter component values, BUT they are limited to very basic filter shapes (like low-pass, high-pass, symmetric bandpass) AND almost always require a skilled human designer to tune the values afterward. Additionally, the closed form filter synthesis methods cannot easily account for higher-order real-world effects which have a big impact on component value selection. These higher order effects include uninteded coupling between circuit elements, component parasitics, and propagation delays for high-frequency distrubuted circuits.

The problem: Existing closed form filter synthesis methods are limited to very basic filter shapes, do not account for impactful higher order effects, and require significant time for human designers to tweak values by trial and error.

Overall Goal of Paper: Make progress towards a machine learning (ML) model that can:

1) generate filters with much more arbitrary frequency responses (not just simple low-pass, symmetric bandpass, etc.),
2) accurately account for the higher-order effects which closed-form solutions ignore,
3) significantly decrease the amount of human tuning required by generating more accurate component values.

Specific Project Scope: This project specifically tackles the design of 5th-order unbalanced ladder filters with series element first. The user gives model the desired filter frequency response (can be arbitrary shape), and the model designs a working filter.

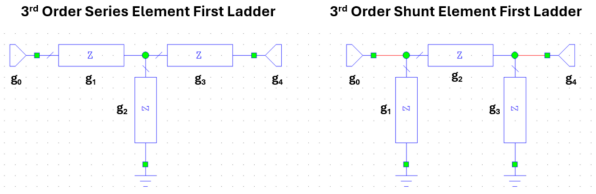

Fig. 1. 3rd order ladder representation: comes in series element first or shunt element first.

Figure 1 shows a 3rd order filter topology. For this design, each element of the ladder is made up of a capacitor and inductor. The series elements are made up of a series LC and the shunt elements are composed of a parallel LC. This means there are 10 different electrical component values to select for the filter design.

II. Description of Approach

To achieve the overall project goal the following approach was taken. Each entry has its own sub-section:

1) Research existing attempts at ML based circuit designers
2) Learn and implement the standard closed form filter synthesis methods *
3) Develop method to quickly evaluate circuit performance AND verify against commercial circuit solvers **
4) Generate Training Data
5) Model Training Details
   Importance of criterion selection
   Multi-Step Training Program to significantly increase training speed

* This is critical for publishing a paper in the future. Many of the machine learning papers on circuit design DO NOT compare against standard circuit design methods and lack sufficient electrical engineering expertise. The lack of standard EE knowledge is a large hole in the current literature as of 2025.

** generating sufficient data can be expensive in microwave design

A. Research existing attempts at ML based circuit designers

Existing work on ML based circuit designers can be broken into parameter design vs. topological design. Parameter designs have the AI only choose component values for a pre-chosen circuit, whereas in ML topological design the AI generates the entire circuit with very little human guidance.

My class presentation and survey essay go into depth on the recent ML microwave design literature published in 2023 2025. In those submissions the basics of the RF design process are explained, and the potential for limited AI assistance at each step is briefly discussed [1]. Next, the distinction between parameter optimization and topological design is made. [2] was used to demonstrate AI's ability to balance the many parameters required in an RF design whose topology is fixed. Then, the topological AI design algorithms described in [3], [4], [5] are described and methods compared. Particular attention is paid to the wavelet transform used in [3] and how it aids in inverse mapping. Additionally, the two-step learning method and fully connected layer dropout operations in [4] was discussed in depth.

Some key items to note are:

1) Collecting the required amount of training data is very difficult for microwave designs because the most accurate electromagnetic simulators are extremely compute intensive.
2) Many of the AI topological design papers involve training a model which quickly evaluates circuit performance with very little compute, this model is then used to guide an evolutionary algorithm which explores the design space.
3) This project takes the reverse approach of generating a model which directly generates the filter design, instead of using an evolutionary algorithm to search through many different designs every time the user asks for a new filter synthesis.

B. Learn and implement the standard closed form filter synthesis methods

One of the standard methods used for filter synthesis involves calculating a table of values called G-coefficients which define a prototype filter. It is called a prototype filter because its cuttoff frequency is 1 rad/sec. The G-coefficients must be specifically scaled and altered to achieve a different cuttoff frequency and/or to switch filter response shape (e.g. low pass, high pass, bandpass). Different families of filters can be synthesized by using a wide range of complicated recursive formulas. The Butterworth and Chebyshev filter styles are common place in the filter synthesis world.

Why this matters: Many of the existing ML microwave design papers discuss performance of circuits the AI generates, but the authors do not analyze the results against standard methods. For example, in [4] they discuss how amazing it is the AI designed an electrically small patch antenna, but they don't mention antenna efficiency or that electrically small antennas is an entire field of study which has defined upper bound limits of antenna performance for a specific size. [4] is one of the best ML microwave papers as of 2025, but it severely lacks sufficient EE background.

Below the method of computing Butterworth and Chebyshev filter coefficients is described.

For standard filter synthesis there are several commonly used designs which dictate where the poles and zeros of the filter lie. Butterworth and Chebyshev filters are common and are synthesized as follows.

— Butterworth Filter Synthesis

$$g_r = 2\sin\left(\frac{(2r-1)\pi}{2n}\right), \quad r = 1, \ldots, n \tag{1}$$

— Chebyshev Filter Synthesis

— Insertion Loss Ripple

$$L_{A\_dB} = -10 \cdot \log\left(1 - \left(10^{\frac{-L_{r\_dB}}{10}}\right)\right) \tag{2}$$

— Ripple Factor

$$\epsilon = \sqrt{10^{\frac{L_{A\_dB}}{10}} - 1} \tag{3}$$

— coefficient for input port is always 1

$$g_0 = 1 \tag{4}$$

— First Circuit Element

$$g_1 = \frac{2}{\eta}\sin\left(\frac{\pi}{2n}\right) \tag{5}$$

$$\eta = \sinh\left[\frac{1}{n}\sinh^{-1}\left(\frac{1}{\epsilon}\right)\right] \tag{6}$$

— Subsequent Elements

$$g_r g_{r+1} = \frac{4\sin\left(\frac{(2r-1)\pi}{2n}\right)\sin\left(\frac{(2r+1)\pi}{2n}\right)}{\eta^2 + \sin^2\left(\frac{r\pi}{n}\right)}, \quad r = 1, 2, \ldots, n-1 \tag{7}$$

— Load for Odd Order Filters

$$g_{\text{load}} = 1 \tag{8}$$

— Load for Even Order Filters From:

$$|S_{21}(0)|^2 = \frac{4g_{n+1}}{(g_n+1)^2} = \frac{1}{1+\epsilon^2} \tag{9}$$

Solve for g_{n+1}:
- For $S_{11}(0) \geq 0$:

$$g_{n+1} = \frac{\left(\epsilon + \sqrt{1+\epsilon^2}\right)^2}{1} \tag{10}$$

- For $S_{11}(0) \leq 0$:

$$g_{n+1} = \frac{1}{\left(\epsilon + \sqrt{1+\epsilon^2}\right)^2} \tag{11}$$

Chebyshev Lowpass Prototype Coef. (0.05dB ripple)

N	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_10	g_11	g_load shunt first
1	0.2152	1.0000										
2	0.6923	0.5585	1.2396									0.8067
3	0.8794	1.1132	0.8794	1.0000								
4	0.9588	1.2970	1.6078	0.7734	1.2396							0.8067
5	0.9984	1.3745	1.8283	1.3745	0.9984	1.0000						
6	1.0208	1.4141	1.9183	1.5475	1.7529	0.8235	1.2396					0.8067
7	1.0346	1.4369	1.9637	1.6162	1.9637	1.4369	1.0346	1.0000				
8	1.0436	1.4514	1.9899	1.6503	2.0457	1.6053	1.7992	0.8419	1.2396			0.8067
9	1.0499	1.4611	2.0065	1.6698	2.0858	1.6698	2.0065	1.4611	1.0499	1.0000		
10	1.0544	1.4679	2.0177	1.6820	2.1085	1.7009	2.0851	1.6277	1.8197	0.8506	1.2396	0.8067

Chebyshev Lowpass Prototype Coef. (10 dB ripple)

N	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_10	g_11	g_load shunt first
1	0.6667	1.0000										
2	1.3601	0.7066	1.9250									0.5195
3	1.5529	1.1058	1.5529	1.0000								
4	1.6269	1.2051	2.3198	0.8452	1.9250							0.5195
5	1.6625	1.2436	2.4956	1.2436	1.6625	1.0000						
6	1.6821	1.2626	2.5617	1.3308	2.4304	0.8738	1.9250					0.5195
7	1.6940	1.2733	2.5940	1.3627	2.5940	1.2733	1.6940	1.0000				
8	1.7019	1.2801	2.6123	1.3781	2.6527	1.3571	2.4641	0.8841	1.9250			0.5195
9	1.7072	1.2846	2.6238	1.3867	2.6805	1.3867	2.6238	1.2846	1.7072	1.0000		
10	1.7111	1.2877	2.6315	1.3921	2.6960	1.4006	2.6797	1.3671	2.4789	0.8889	1.9250	0.5195

Fig. 2. Computed Chebyshev G-coefficients. These values have been verified against other sources.

C. Develop method to quickly evaluate circuit performance AND verify against commercial circuit solvers

A significant amount of time went into writing the code to compute the G-coefficients and to scale the values to get a standard filter design.

I wrote a program that calculates the G-coefficients for Butterworth and Chebyshev filters using the recursive equations above. I also wrote the scripts that scale those G-coefficients to get electrical component values for inductors and capacitors. Finally, I wrote a circuit simulator that has been checked against actual circuit simulator to verify accuracy. Figure 2 shows the G-coefficients calculated for Chebyshev filter prototypes of different orders.

I also wrote scripts which simulates the filter circuit to measure its frequency response. The results of my circuit simulator were compared against a commercial microwave circuit simulator (Microwave Office AWR). Figure 3 shows that my simulated filter performance exactly aligns with the commercial solver. Notice the precise alignment between the commercial solver (AWR) and my own circuit simulator script. It was critical to write the circuit solver so that it could run on the GPU for fast simulation time and so that an ML model training on the GPU could easily differentiate through the weights of the circuit solver. This allows the training ML model to have its loss calculated off of the error of circuit element values OR the difference in the input frequency response versus the generated circuit's simulated frequency response. The difference between these methods is further discussed below.

D. Generate Training Data

I wrote a full set of functions which allowed me to generate filters with different parameters. I used Latin Hyper Cube sampling to generate a random set of training data, validation data, and testing data. I also added the ability to introduce

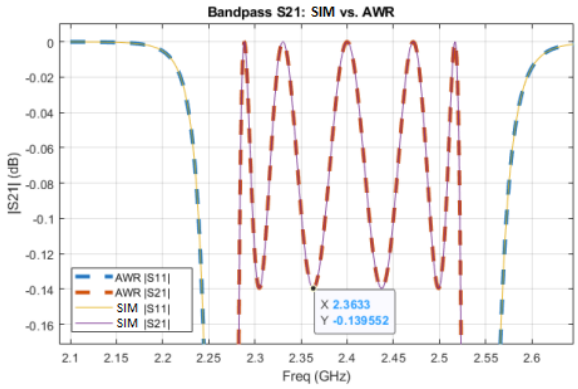

Fig. 3. S21 of a bandpass filter simulated with AWR and my own script. Notice the very zoomed in vertical axis to accurately measure the filter passband ripple. My circuit simulator lines up exactly with the commercial solver.

varying levels of component variation in the generated filter designs.

I used this to generate three categories of training data that is used in my training (more about training steps in next section.)

1) Perfect Chebyshev Filter Designs (no random component variation)
2) Chebyshev filters with varying level of random component variation
3) Completely randomly generated component values

For the Chebyshev family of filters there are three main parameters: Center Frequency, Passband Ripple (dB), and Fractional Bandwidth. The Hyper Cube Sampling ensures that a random sample is taken in each section of this 3-dimensional space. LHCS has the advantage of better coverage for fewer samples than using the Monte-Carlo sampling approach.

TABLE I
DESCRIPTION OF TRAINING DATA STYLES

Function	Description
`gen_cheby_data(...)`	Ideal Chebyshev filters with no component variation; serves as baseline.
`gen_cheby_data(..., comp_var=0.1)`	Chebyshev filters with $\pm 10\%$ component variation for moderate fabrication noise.
`gen_cheby_data(..., comp_var=0.2)`	Chebyshev filters with $\pm 20\%$ variation to represent higher manufacturing variation.
`gen_cheby_data(..., comp_var=0.5)`	Chebyshev filters with $\pm 50\%$ variation to simulate extreme uncertainty.
`gen_rand_data(...)`	Filters with fully random component values (10^{-14}–10^{-6}), unconstrained by a prototype.

This data was saved to files using the pickle python package for quick loading into a workspace for training.

E. Model Training Details

Scaling Model Inputs and Outputs:

Originally, I used the MSELoss() function to compute the error between the ideal component values and the inferred values. I had much more successes computing the mean square percentage due to the values being larger and easier to understand. Additionally, I convert the filter frequency response data to dB so that the model doesn't need to contend with such a large dynamic range. Using dB also aligns with standard practice in EE.

The original proposed training approach involved two stages:

1) Training with perfect Chebyshev filters (with loss calculated from component value error)
2) Once a baseline of performance is reached more randomly varied filters are introduced into training data (eases model into designs different from perfect Chebyshev)
3) Finally, Switch to loss calculated from difference of filter frequency responses (not the actual circuit element values)

   this allows understanding of which components values are more sensitive to filter freq. response

The rationale behind having training datasets with different levels of randomness is to "teach" the model with simple consistent data first (the clean, non-altered data) and then once the model is partially established start introducing more random data.

By training only perfect Chebyshev filters to start, the model can benefit from a pre-established way of thinking without having to start from scratch. It is then hoped the randomly varied filters will allow the model to "understand" the sensitivity of each component in the circuit.

III. Evaluation of Method

In the end amazing filter design accuracy was achieved using an MLP that took 1000 values in (each corresponding the S21 in dB) and generating 10 circuit element parameters out. The final testing loss 0.001% percent error in the calculated circuit element values after 2000 epochs taking around 2 minutes to train. See figure 4 for the training loss curves.

IV. Conclusion

A program has been written that can calculate the G-coefficients for Butterworth and Chebyshev filters. More scripts have been written that can scale those G-coefficients to get electrical component values for inductors and capacitors. Finally, I wrote a circuit simulator that has been checked against actual circuit simulator to verify accuracy. These scripts will be used to generate the training data and the circuit simulator will also be used to evaluate model performance.

By understanding the different aspects and limitation of ANN like different training methods, data scaling to minimize dynamic range, etc. I created a model which can accurately generate 10th order analog filters to meet the speciations of a near arbitrary desired frequency response.

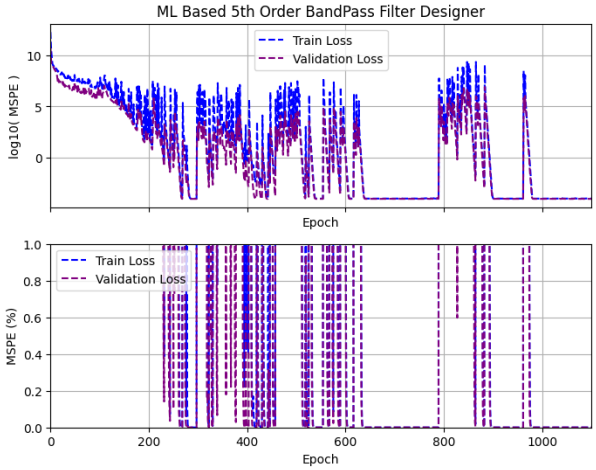

Fig. 4. Notice MLP reaches error well below 1% of theoretical component values. It should be noted many electrical components have a tolerance of +/-1%

References

[1] P.-Y. Lee and T.-C. Chen, "Ai-driven innovations in ic designs: From planning to implementation," in *2024 International VLSI Symposium on Technology, Systems and Applications (VLSI TSA)*, April 2024, pp. 1–2.

[2] L. Xue, H. Fan, Y. Ding, and B. Liu, "A design methodology of mmic power amplifiers using ai-driven design techniques," in *2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, July 2023, pp. 1–4.

[3] K. Xu *et al.*, "Flexible inverse design of microwave filter customized on demand with wavelet transform deep learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2024.

[4] E. A. Karahan, J. Zhou, Z. Liu, Z. Shao, S. Fisher, and K. Sengupta, "Deep learning enabled design of rf/mmwave ic and antennas," in *2024 IEEE 67th International Midwest Symposium on Circuits and Systems (MWSCAS)*, August 2024, pp. 769–772.

[5] H. Taşkıran, E. Sağlıcan, and E. Afacan, "Ann-based analog/rf ic synthesis featuring reinforcement learning-based fine-tuning," in *2024 20th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, July 2024, pp. 1–4.