Re-optimizing aLIGO RF filter

2025-12-08

Reoptimizing filter circuits for better noise performance.

blog: https://tetraquark.vercel.app/posts/ligo_snr_reopt/?src=pdf

email: quarktetra@gmail.com

This is the final one of a series of posts on LIGOs photodetector circuits. It builds on a few earlier posts of mine:

- Real coils are not purely imaginary,
- An analysis of aLIGO PD circuit,
- LIGO modulation,
- RLC filters.

In this post we will look into the Advanced Laser Interferometer Gravitational-Wave Observatory (aLIGO) photodiode circuit and compute its noise performance. We then apply what we have learned in these four posts above to reoptimize the circuit and show that we can design a circuit with substantially better performance.

The model

The photodiode connects to the rest of the circuit as shown in Figure 1.

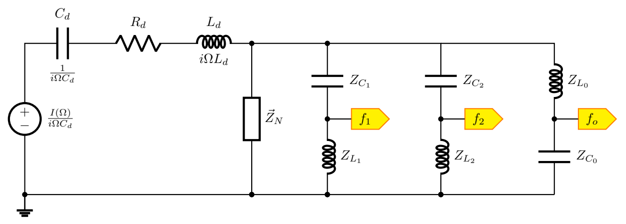

Figure 1: The photodiode connected to the notch filters, collapsed into a single impedance, and the selectors. Notch filters knock out the unwanted harmonics and the frequency selectors will take in the targeted frequency while rejecting everything else.

Although we will be dealing with a fixed number of notches and readouts in this particular problem, it will be worthwhile to build the model in a generic way so that it will be applicable to an arbitrary number of notches and readouts. As we are developing the model in the most generic sense, we will want to develop the numerical analysis code in the same way so that it can handle arbitrary circuits. Furthermore, at any point in the numerical analysis, we will want to be able to automatically create a circuit in LTspice, simulate it there and collect and analyze the resulting data. This is more complicated than simply passing parameters to LTspice since the topology of the circuit itself may change. We will need to get creative to make this work! Let us start with the mathematical model.

In Figure 1, we already collapsed the notch filters into a block and called the impedance of that block $\vec{Z}_N \equiv \{Z_N^1, Z_N^2, \cdots, Z_N^n\}$, and we will also collapse all the readout filters into $\vec{Z}_R \equiv \{Z_R^1, Z_R^2, \cdots, Z_R^m\}$. With these blocked elements, the total impedance is simple to write down:

$$Z_T = Z_d + \text{para}\left(\vec{Z}_N, \vec{Z}_R\right), \tag{1}$$

where para() returns the parallel impedance.

The voltage at the i^{th} readout port is given by the simple voltage division:

$$V^i = I Z_{C_D} \frac{\text{para}\left(\vec{Z}_N, \vec{Z}_R\right)}{Z_T} \frac{Z^i_{\text{R,out}}}{Z^i_R}, \tag{2}$$

where $Z^i_{\text{R,out}}$ is the impedance of the circuit element the readout voltage is measured on. It will be a capacitor in parallel with a resistor for the DC output. It may be an inductor for the other readouts.

We can define a trans-impedance which converts the input current I to V_{out}:

$$Z^i_{\text{tr}} \equiv \frac{V^i_{\text{out}}}{I} = Z_{C_D} \frac{\text{para}\left(\vec{Z}_N, \vec{Z}_R\right)}{Z_T} \frac{Z^i_{\text{R,out}}}{Z^i_R}. \tag{3}$$

An input current of amplitude I_j and angular frequency of Ω_j will create a voltage at the readout port i (also the OPAMP input) as

$$V^i_j = I_j Z^i_{\text{tr}}(\Omega_j). \tag{4}$$

The spectrum of the input signal from the photodiode is given in Table 1, and the spectrum shows very narrow spikes at multiple frequencies [1].

Table 1: Spectrum of power (in mW) at various ports and frequencies in two modes of operation [1].

Table 1: Spectrum of power (in mW) at various ports and frequencies in two modes of operation [1].

name	age	gender

We can calculate the current amplitudes I_j using the power values at various frequency in Table 1 and using a conversion efficiency of 0.86.

The thermal noise will appear at the OPAMP port. That noise is created by the real part of the equivalent impedance, which we will refer to as backwards impedance. Consider the read out port i:

$$Z^i_{\text{BO}} \equiv \text{para}\left(Z^i_{\text{R,out}}, \text{para}\left(\vec{Z}_N, \vec{Z}^{-i}_R\right) + Z^i_{\text{R,in}}\right). \tag{5}$$

The notation needs some explanation:

1. $Z^i_{\text{R,out}}$: the filter element the read out is measured on
2. $Z^i_{\text{R,in}}$: the complementary element of the filter.
3. $\vec{Z}^{-i}_R$: the read out impedance vector excluding the i^{th} filter.

We can now define the pseudo SNR at readout port i as follows:

$$\text{pSNR}^i = \left.\frac{\left|I_{\text{shot}} Z^i_{\text{tr}}\right|^2}{4k_B T \Re\{Z^i_{\text{BO}}\} + \left|i_{\text{op}} Z^i_{\text{BO}}\right|^2 + (V^o_{\text{v}})^2}\right|_{\Omega_i}. \tag{6}$$

We may have m such readout ports, to which we can assign weights and define a weighted average pSNR:

$$\overline{\mathrm{pSNR}} = \frac{1}{m} \sum_{i=1}^{m} \mathrm{w}_i \, \mathrm{pSNR}^i, \tag{7}$$

which is the metric we will want to maximize. We will require that the unwanted content in the spectrum to not saturate the OPAMP input:

$$V^i \equiv \sum_{j \neq i} V_j^{\mathrm{i}} = \sum_{j \neq i} I_j Z_{\mathrm{tr}}^i(\Omega_j) < V_{\mathrm{sat}} \,, \forall \, i, \tag{8}$$

where the summation is done for all terms in the spectrum except for the targeted readout frequency, i.e., only the cross terms are added up. With Eqs. 7 and 8 the mathematical formalism to optimize the circuit is completed. We can now turn to building the numerical analysis machinery for the computation.

The code

In order to pass the circuit created in the optimization loop to LTspice in automated way, we create a set of building block templates in LTspice, as shown in Figure 2. These blocks are completely defined by their input and output terminals and the place holder values for the elements.

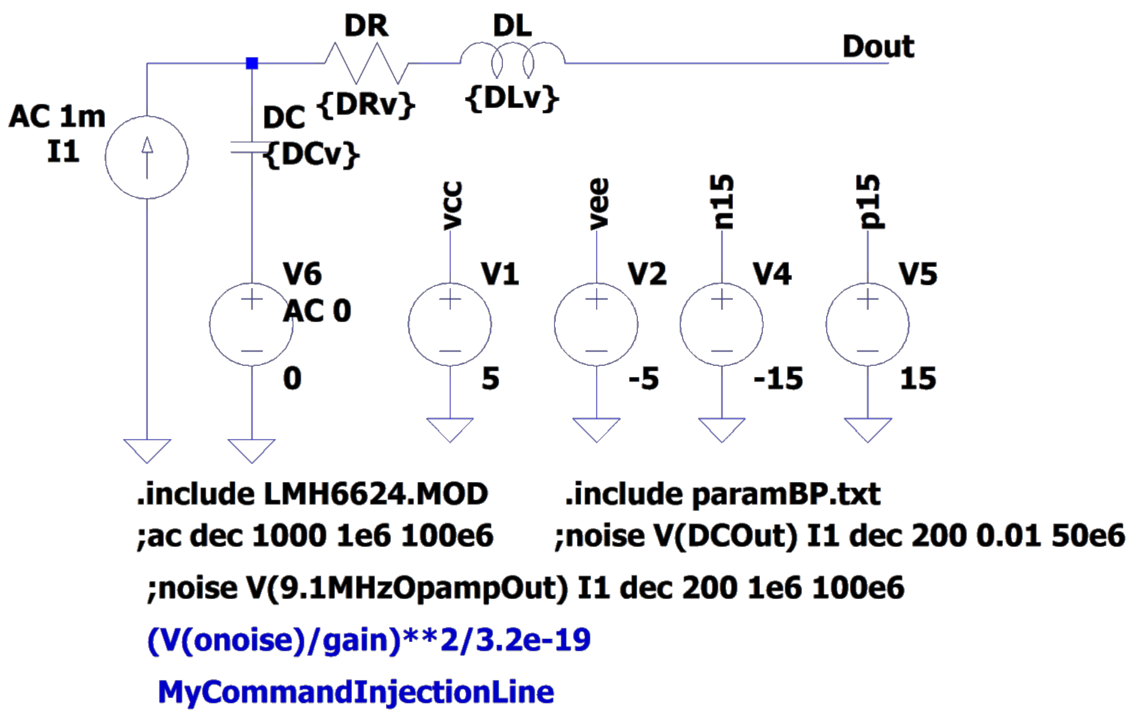

Figure 2: Blueprint templates for LTspice circuit.

The optimization script is coded in R language, and it can load these building blocks, assign values to the terminals to cross connect them to build the full circuit. Once the circuit is built, a Python subroutine is initiated to run the circuit in LTspice, and the resulting data is collected with Python code and analyzed. An outline of the optimization process is shown in Figure 3.

```
Algorithm 1: The optimization algorithm
Data: Diode parameters, power spectrum
Result: pSNR, circuit element parameters
parameters ← initital parameters;
bestparameters ← initital parameters;
bestSNR←pSNR(parameters);
$N$ ← number of iterations;
while $j < N$ do
    pSNR←pSNR(parameters);
    if $pSNR > bestSNR$ and $voltage < voltage\ spec$ then
        bestparameters ← parameters;
        bestpSNR ← pSNR;
        parameters ← new parameters ;      /* parameters pulled from a set */
    else
        parameters ← new parameters;
    end
    $j \leftarrow j + 1$;
end
LTspiceCircuit ← buildCircuit(bestparameters);
Windows CMD ← Python ← LTspiceCircuit;
csv ← Python ← LTspice Raw data;
plots ← R ← csv;
```

Figure 3: The flowchart for the algorithm.

The results

The optimization routine is a brute force search at the moment, however it will work well with smarter techniques such as simulated annealing. Also note that although the LTspice run is pushed to the end to simulate the final configuration, it can be pulled into the loop if there is a more complicated circuit to analyze. This might be a good tool for later projects. The optimization returns the best parameters, which are tabulated and compared to the values in the current circuit in **?@tbl-descriptiontablec**.

[1] R. Abbott et al, "AdvLIGO interferometer sensing and control conceptual design." [Online]. Available:https://dcc.ligo.org/LIGO-T070247/public