LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note **LIGO-T2100238–v3** 2021/09/13

Red Pitaya Digital Laser Controller

O.Elgabori, F.Salces, A.Gupta, R.Adhikari

Contents

1 Abstract

This report details using the Red Pitaya (125 MHz 14 bit) electronic board as a digital feedback controller for laser frequency stabilization. There is an exploration of various digital signal processing functionalities of a python interface to the onboard FPGA. Through a plant model approach, we attempt to validate the performance of Red Pitaya for feedback control. This is achieved by configuring the board to perform system identification and fitting frequency response data to a pole-residue model. The aim is to develop an automated device capable of determining the frequency response of some unknown plant and cancelling undesirable features (e.g. resonances) to produce a flat response.

2 Introduction

2.1 LIGO

The Laser Interferometer Gravitational-Wave Observatory (LIGO) detects and studies gravitational waves using laser interferometry. The detectors used by LIGO are advanced Michelson interferometers that are 4 km long with Fabry Perot cavities and power recycling mirrors that make them highly sensitive and capable of detecting length changes at 1/10,000th the width of a proton. One significant factor that impacts the sensitivity of these interferometers is a frequency stabilized laser. As such, it is highly desirable to have a feedback controller that makes adjustments to maintain a stable frequency response.

2.2 Motivation

The type of laser utilized by LIGO is a non-planar ring oscillator (NPRO), which are known to have a high intrinsic stability (i.e. no external stabilization) on the order of $10^4\text{Hz}/\sqrt{\text{Hz}}$ at 1 Hz [1]. This quantity denoted the free-running frequency noise of the laser. The laser frequency is stabilized by means of a piezoelectric transducer (PZT) . However, the stabilization is impacted by the mechanical resonances of the PZT as they limit the control bandwidth. These mechanical resonances can be suppressed through the implementation of a digital filter.

3 Background

3.1 Transfer Functions

In the Laplace s-domain, a transfer function is the ratio of system's output to the given input

$$H(s) = \frac{B(s)}{A(s)} \tag{1}$$

$$H(s) = \frac{\sum_{m=0}^{M} b_m s^m}{\sum_{n=0}^{N} a_n s^n} \tag{2}$$

where N must be greater than M for physically stable systems.

Typically, the most useful and preferred representation for a transfer function is the ZPK form,

$$H(s) = K\frac{\prod_{m=0}^{M}(s-z_m)}{\prod_{n=0}^{N}(s-p_n)} \tag{3}$$

where z_m are the zeroes of the transfer function(roots of the numerator polynomial), p_m are the poles of the transfer function (roots of the denominator polynomial), and K is an associated gain term that comes from factoring out the coefficients of the highest power term in each polynomial.

3.2 Digital Filters

There are two fundamental types of digital filters: infinite impulse response (IIR) and finite impulse response (FIR). Mathematically, these filters are defined by their impulse response (i.e. infinite or finite) and are represented by (FIX)

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k), \tag{4}$$

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k), \tag{5}$$

where y(n) are the outputs and x(n-k) are the inputs with associated coefficients h(k). The key difference between these two filters is that the FIR outputs depend solely on the provided input values, while IIR filters are recursive in nature so future output values depend on both the inputs and previous outputs. In practice, each filter has its own advantages and disadvantages. For instance, IIR filters consume less memory than FIR filters and have lower latency making them faster as well. However, FIR has linear phase characteristics and are more stable as their output values do not have feedback, and thus will not become unstable for any input signal unlike IIR filters. [2]

3.3 FPGA

A field-programmable gate array (FPGA) is an integrated circuit with programmable interconnects that can be customized for a particular application. As these interconnects can be reprogrammed to be used for other purposes other than the originally desired application, it makes this device quite versatile when used in the field. The Red Pitaya (Figure 1) is an electronic board that possesses a FPGA along with other useful components, such as digital to analog (DAC) and analog to digital (ADC) converters. The Red Pitaya is preferable to use over other electronic boards with an FPGA as it is low cost, has a user friendly interface, and possesses extensive documentation. In addition, other FPGA-based systems are not as readily usable as the Red Pitaya as this particular board has an open source python package called PyRPL.

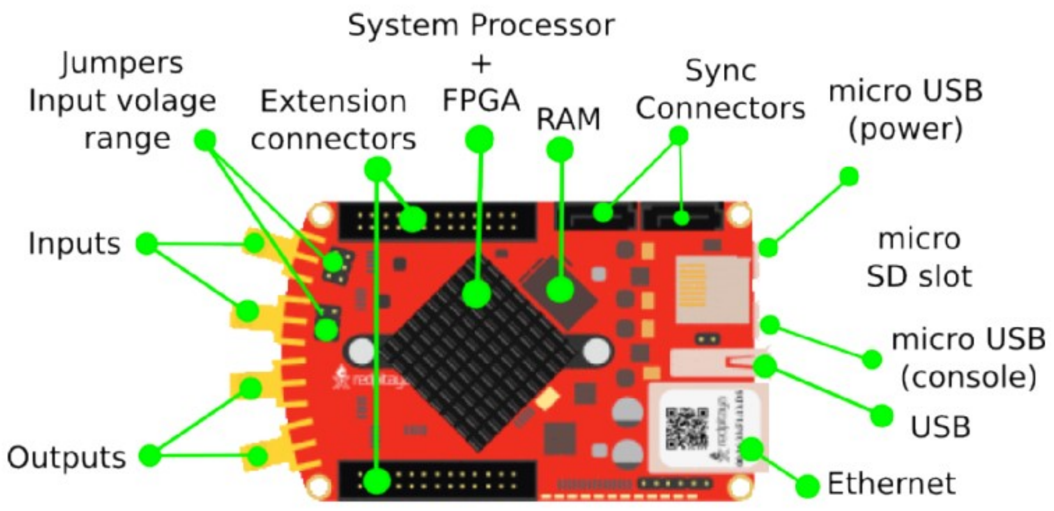

Figure 1: Schematic of the Red Pitaya board[4]

4 Approach

We aim to improve the stabilization of the laser by suppressing the mechanical resonances of the piezoelectric transducer by the Red Pitaya FPGA [3]. We will accomplish this by programming the FPGA on the Red Pitaya determine the optimal digital filter — IIR or FIR — to use for suppressing the resonant modes of the piezo. The feedback system is represented by a simple block diagram in figure 2.

As FPGA programming can be time consuming, we will use the customizable digital signal processing modules provided by PyRPL. This software package not only provides many instruments, including high order digital filters, it comes with a graphical user interface and has a python API.[6]

4.1 PyRPL

Despite its outdated documentation, the software PyRPL provides convenient modules and a user friendly interface to the Red Pitaya. The network analyzer module is of particular interest as it allows us to probe an unknown system's transfer function (the ratio of the output signal of a system to a given input signal). The network analyzer accomplishes this probe through IQ (In-phase/Quadrature) modulation and demodulation. This involves exciting the device under test with a frequency sweep of sine functions, demodulating the output with the same sine and a corresponding cosine, then low-pass filtering and extracting the phase and magnitude.[7] A bode plot of the transfer function for a 1.9 MHz low pass filter (Mini-Circuits Model BLP-1.9) provided by the network analyzer module is shown in figure 3 and 4.

Extracting the transfer function of an unknown system using the network analyzer module

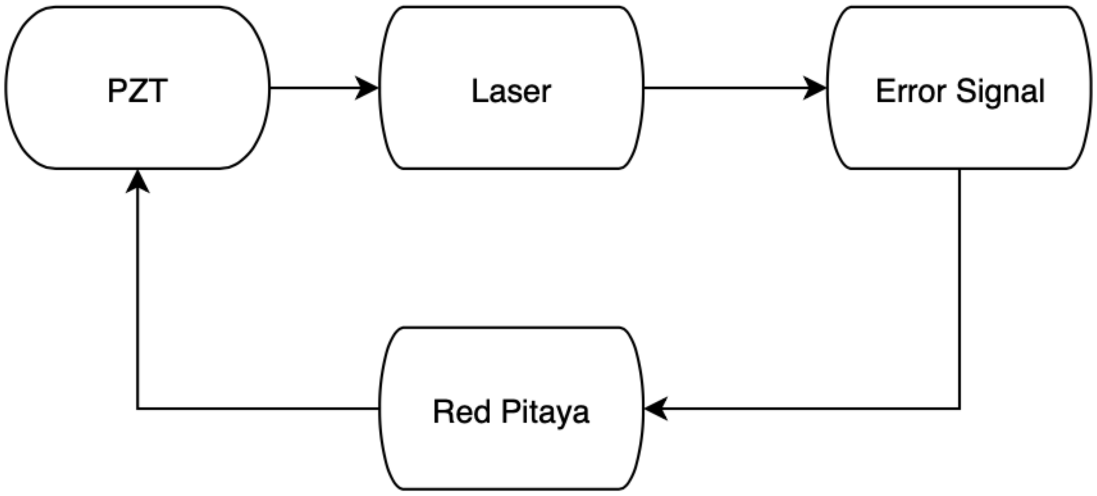

Figure 2: A closed-loop system of the NPRO laser PZT and Red Pitaya controller. The frequency of the laser is compared to a stable reference frequency to produce an error signal that is fed to the controller. The controller then corrects for this error by feeding into the PZT actuator that is used to control the laser frequency.

enables us to perform system identification. Based on this information, we can develop a model (analytical or numerical) and construct digital filters by taking the model's inverse. This can be achieved using the zpk (zero-pole-gain) representation of transfer functions.

4.2 Capabilities of PyRPL

Using PyRPL, it is possible to digitally simulate transfer functions, such as a bandpass filter, on one Red Pitaya and extract this transfer function through the network analyzer on a separate Red Pitaya board. This simulation can be achieved through one of the available IQ modules that is provided or, for a more complicated transfer function, using the IIR filter (Figure 7). Furthermore, in our examination of the network analyzer module we probe the transfer functions of several different low pass filters. These filters serve as test systems in our attempt to validate the performance of PyRPL and the Red Pitaya to achieve system identification and feedback control. In order to obtain an estimation of the transfer function for these analog devices, the vector fitting algorithm is used to provide a fit to the measured data.

4.3 Vector Fitting

Vector fitting is a method developed by Bjørn Gustavsen to fit measured frequency response data given some set of an initial poles.[8] This method attempts to solve for the parameters of the partial fractions model

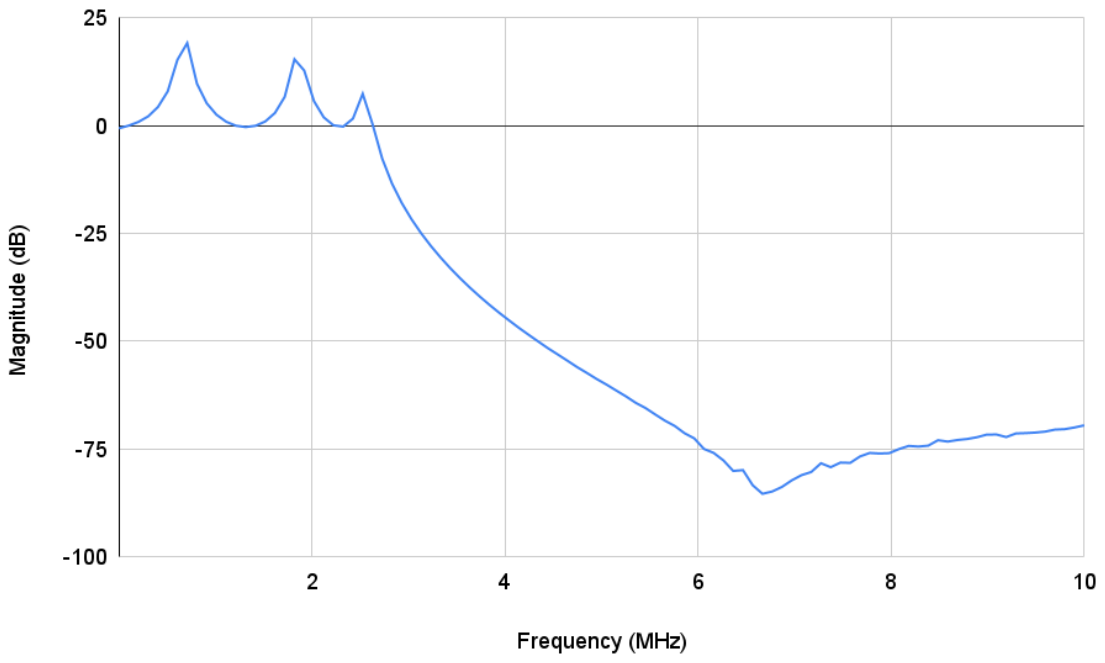

Figure 3: The magnitude of the transfer function for a 1.9 MHz low pass filter produced by the network analyzer.

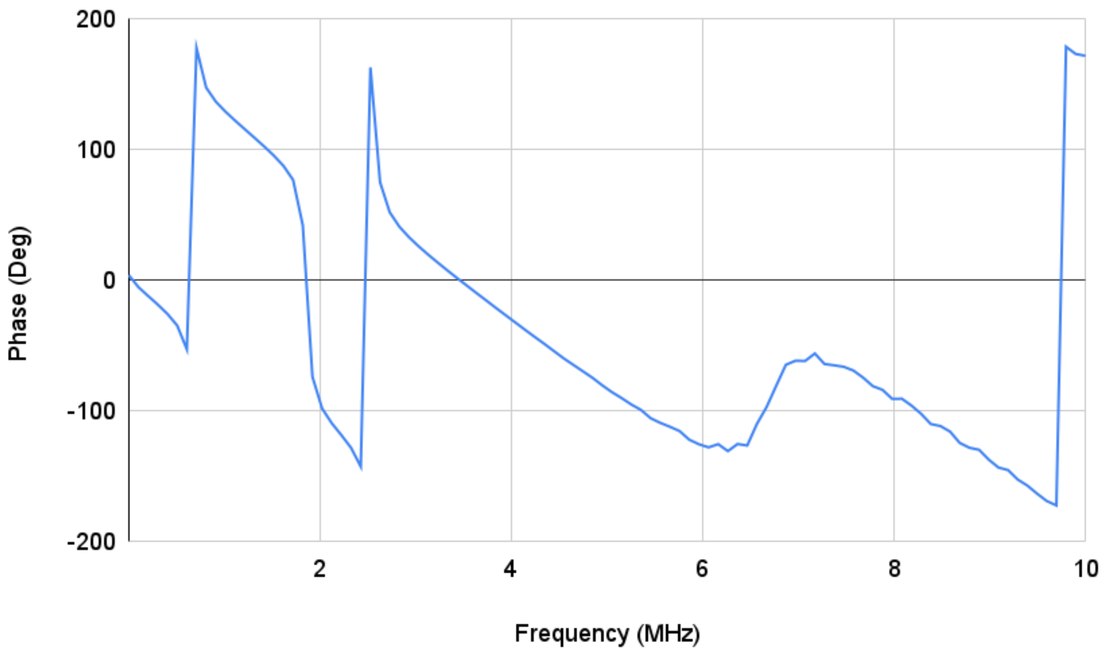

Figure 4: The phase of the transfer function for a 1.9 MHz low pass filter produced by the network analyzer.

$$H(s) = \sum_{m=1}^{N} \frac{r_m}{s - p_m} + d + sh \tag{6}$$

where r_m are the residues, p_m are the poles, and d and h are just some coefficients of the polynomial term of the model. Using some initial set of poles, we write

$$\sigma(s)H(s) = x(s) \tag{7}$$

$$(\sum_{m=1}^{N} \frac{\tilde{r}_m}{s - q_m} + 1)H(s) = \sum_{m=1}^{N} \frac{r_m}{s - q_m} + d + sh \tag{8}$$

where $\sigma(s)$ and $x(s)$ share the same initial poles q_m. Then, we solve an eigenvalue problem to obtain a new set of poles, and iterate the process until $\sigma(s)$ converges to 1 and $q_m = p_m$.

Using an implementation of the vector fitting method in python, we obtain fits for the transfer functions of the various low pass filters. However, the code requires the user to provide an initial guess to the number of poles. In order to automate this process and obtain the best fit, we develop a python function that loops through 1-16 poles as the initial guess for the method and uses chi square as the criterion for determining the best fit. The upper limit of 16 poles is a practical constraint as this highest order that PyRPL can currently provide.

5 Results

5.1 System Identification

The Red Pitaya's ability to perform system identification is tested using several low pass filters. The setup of the device for this process is indicated in figure 5. The magnitude and phase data from the network analyzer matches the expected results from these filters as shown by figures 8-10 (see appendix). In addition, the system identification is performed with sufficient speed. Thus, the Red Pitaya shows promise for generic plant identification.

5.2 System Control

After examining the system identification properties of the Red Pitaya, we examine its ability for system control through the IIR filter module (Figure 6). Although capable of cancelling resonances (Figure 11) and simulating the resonances typical of piezoelectric transducers, these tasks can only be achieved experimentally. In other words, the IIR module's graphical user interface must used in order achieve the desired response. The reason for this is that the IIR module fails to properly implement complex zeros. This poses a problem as we are attempting inverse filtering to cancel resonances, and therefore need to change complex poles into complex zeros. Furthermore, the IIR module can experience saturation issues preventing a desired transfer function from being properly implemented.

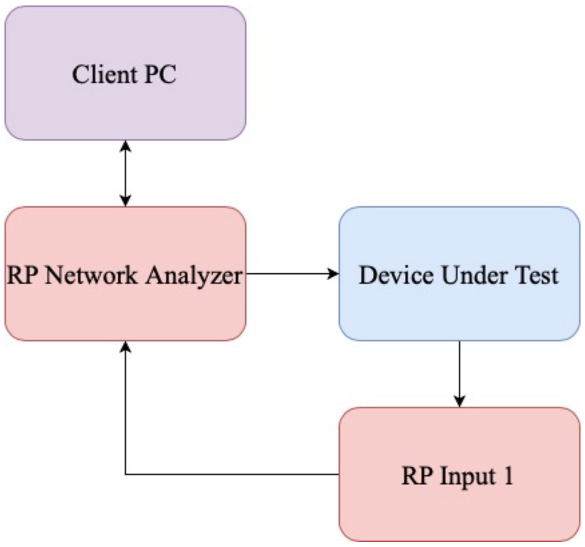

Figure 5: A flow diagram that details the setup of the Red Pitaya for system identification. The client PC sets up the network analyzer to probe the transfer function of a device under test. The response of the device is sent back to the Red Pitaya to be recorded by the network analyzer and viewed on the PC by the user.

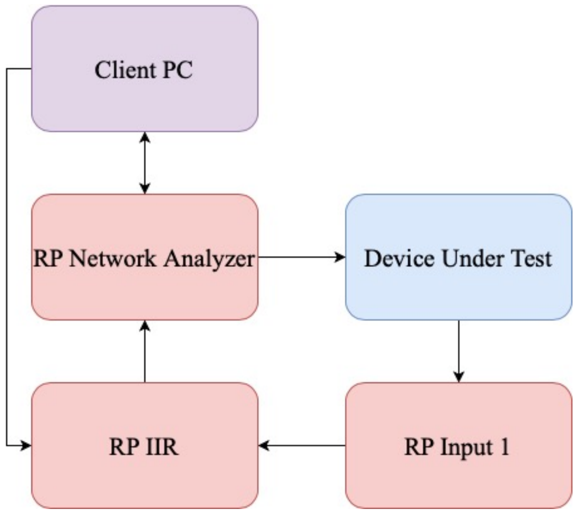

Figure 6: A flow diagram that details the setup of the Red Pitaya for filtering after system identification. The client PC sets up the IIR module based on the system identification performed. Then, the network analyzer is setup to probe the result of passing the device under test through the IIR module.

6 Future Work

For the purpose of filtering PZT resonances efficiently, it is necessary to improve the IIR filter module where it fails as mentioned previously. For purpose of generic plant identification and optimal system control beyond the cancellation of PZT resonances, it would be valuable to include an FIR filter module in PyRPL. This would involve learning the subtleties of FPGA programming using the architecture of PyRPL as a foundation to develop this module. Accomplishing both of these goals will likely involve removing other modules of PyRPL to free FPGA resources and divert them to the development of the IIR and FIR filters.

References

[1] B. Willke , S. Brozek, K. Danzmann, V. Quetschke, and S. Gossler *Frequency stabilization of a monolithic Nd:YAG ring laser by controlling the power of the laser-diode pump source*. Optics Letters 25 (2000)

[2] *Difference between IIR and FIR filters: a practical design guide* (2020)
https://www.advsolned.com/difference-between-iir-and-fir-filters-a-practical-design-

[3] M. Okada, T. Serikawa, J. Dannatt, M. Kobayashi, A. Sakaguchi, I. Petersen, and A. Furusawa *Extending the piezoelectric transducer bandwidth of an optical interferometer by suppressing resonance using a high dimensional IIR filter implemented on an FPGA*. Review of Scientific Instruments 91, 055102 (2020).

[4] *Red Pitaya Developer's Guide: Red Pitaya boards comparison* https://redpitaya.readthedocs.io/en/latest/developerGuide/125-10/vs.html

[5] L. Neuhaus, S. Deléglise *Basics of the PyRPL Architecture* https://pyrpl.readthedocs.io/en/latest/basics.html

[6] L. Neuhaus, R. Metzdorff, S. Chua, T. Jacqmin, T. Briant, A. Heidmann, P.-F. Cohadon, and S. Deléglise *PyRPL (Python Red Pitaya Lockbox) — An open-source software package for FPGA-controlled quantum optics experiments*. European Quantum Electronics Conference (2017)

[7] L. Neuhaus, S. Deléglise. *Network analyzer* https://pyrpl.readthedocs.io/en/latest/api.html#network-analyzer

[8] B. Gustavsen and A. Semlyen *Rational approximation of frequency domain responses by vector fitting* IEEE Trans. Power Delivery, vol. 14, no. 3, pp. 1052-1061, July 1999.

7 Appendix

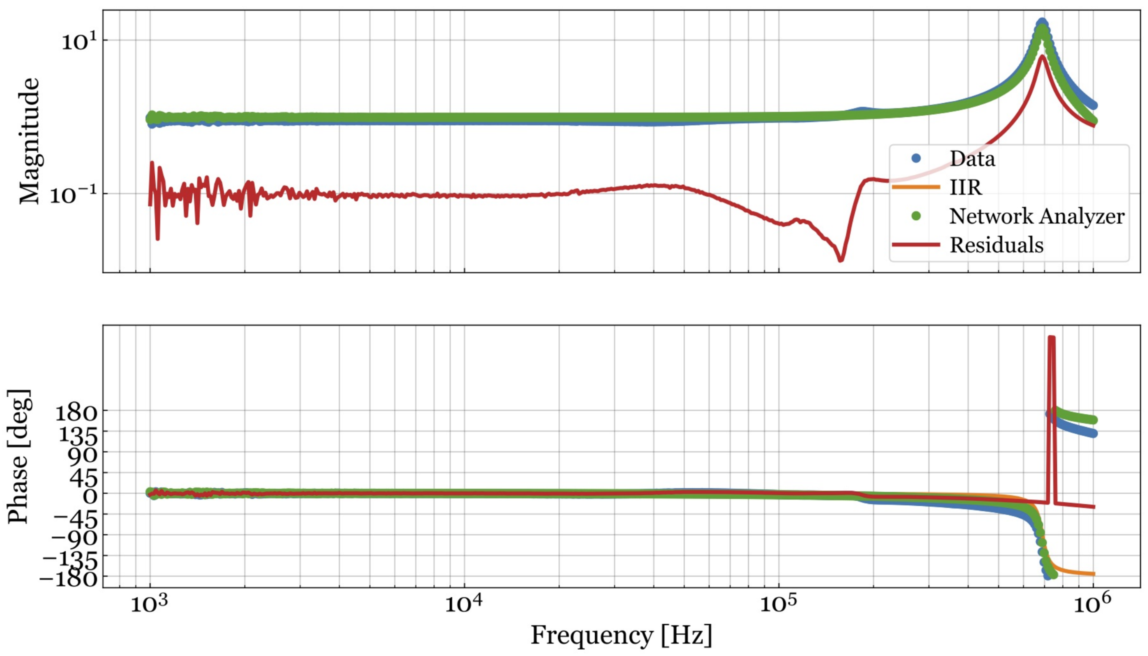

Figure 7: The simulated transfer function of the 1.9 MHz Minicircuits low pass filter obtained from the IIR filter module. The IIR data (orange) is covered by the network analyzer data (green), which is necessary to verify that the transfer function has been correctly implemented.

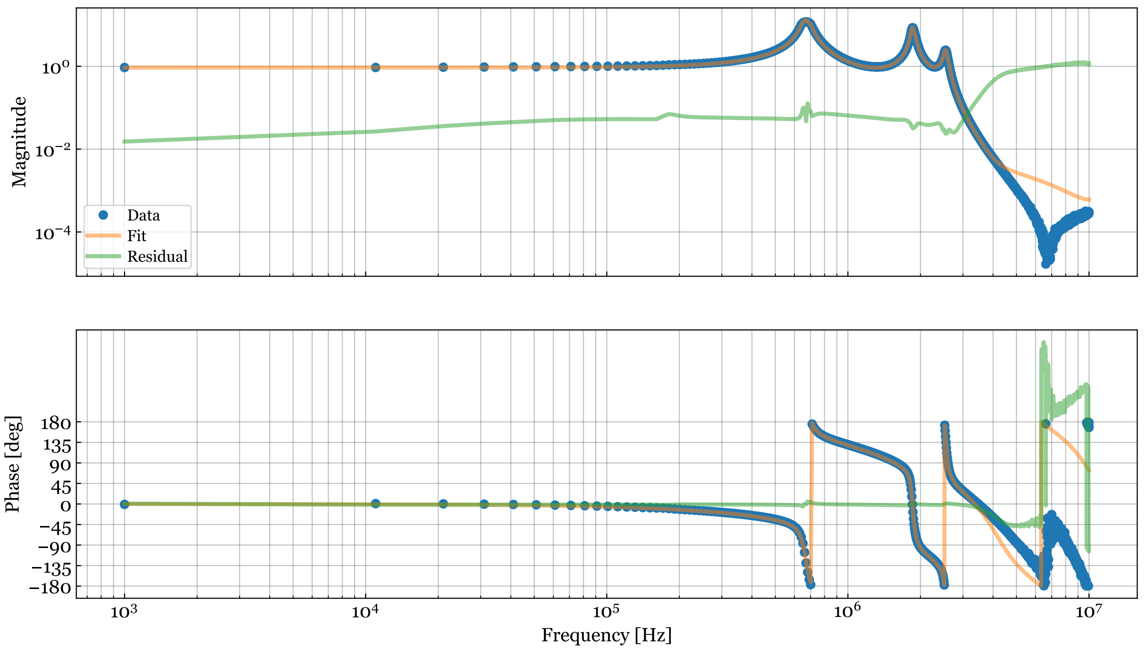

Figure 8: The best fit of the frequency response for a Minicircuits 1.9 MHz low pass filter. The fit is consistent with the data up until approximately 5 MHz, where the fit fails as it attempts to match unphysical behavior.

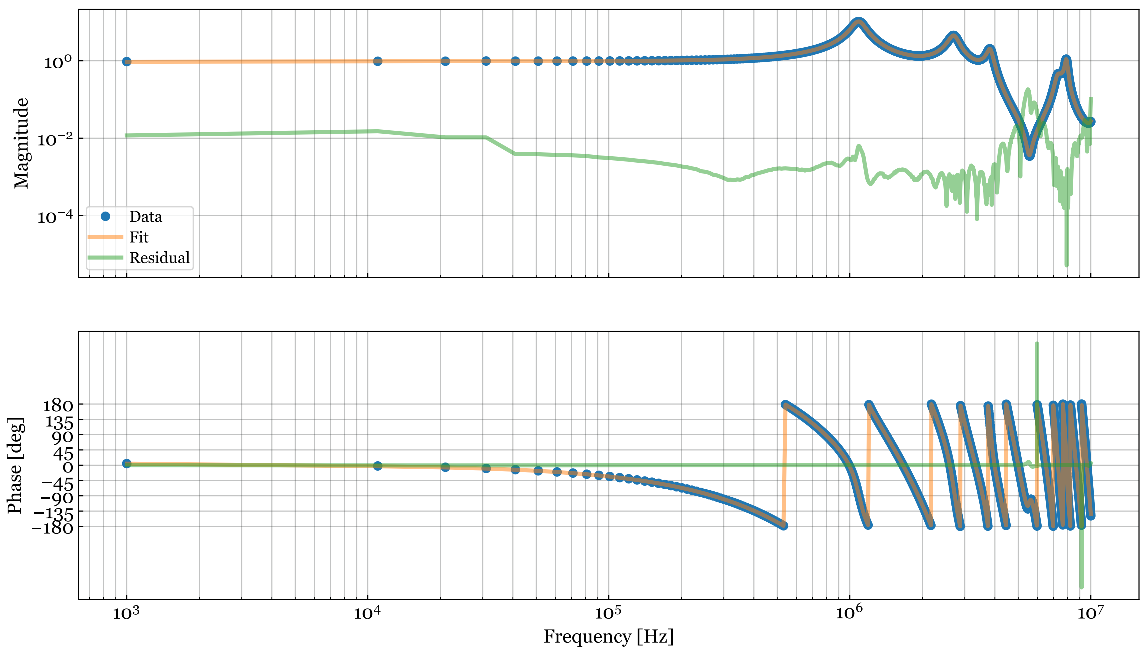

Figure 9: The best fit of the frequency response for a 10 MHz low pass filter with a notch located at 33 MHz.

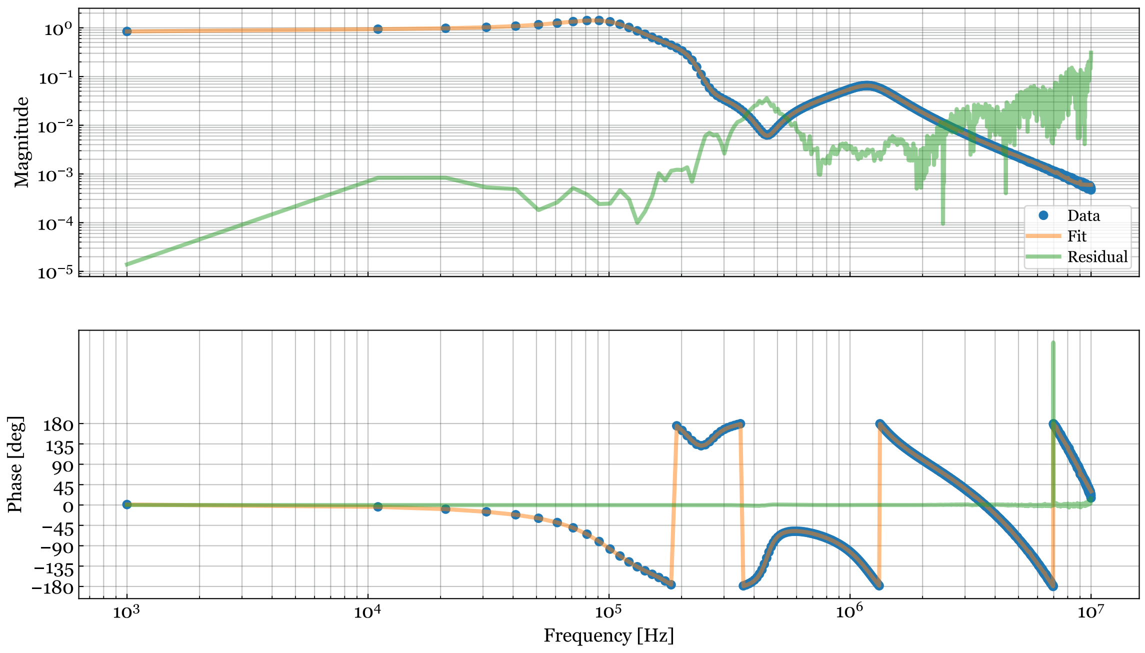

Figure 10: The best fit of the frequency response for a 5^{th} order elliptic low pass filter.

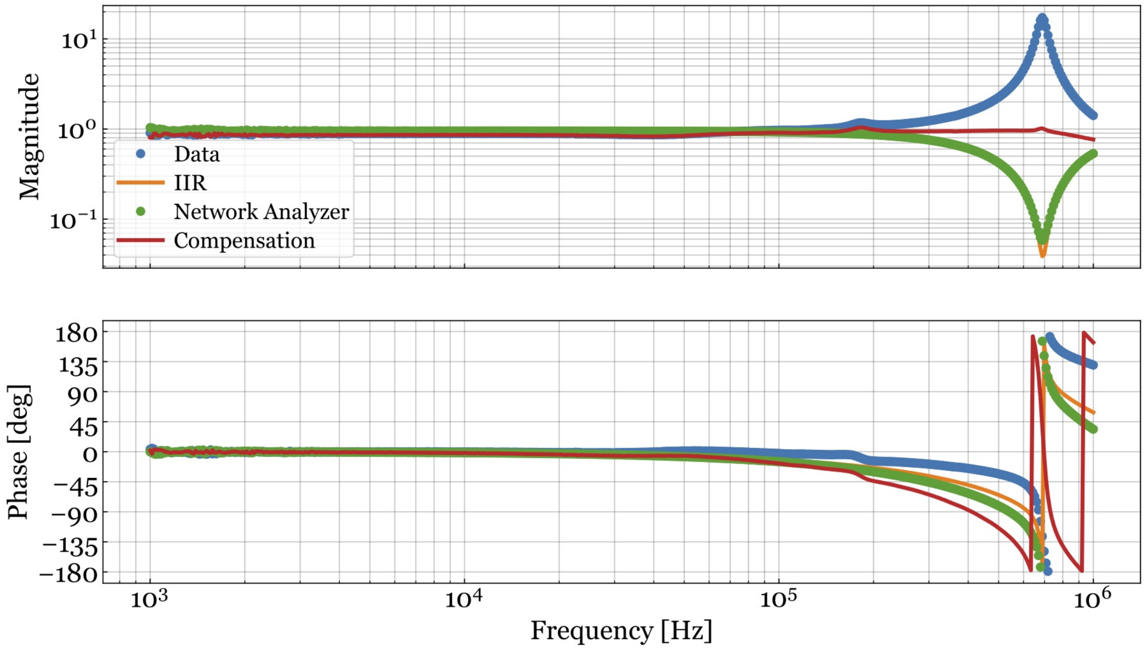

Figure 11: The cancellation of one resonance from the 1.9 MHz low pass filter using the IIR filter module to produce a desirable flat response. The blue is data taken from the network analyzer for system identification, the orange is the inverse transfer function implemented by the IIR module, the green is the network analyzer data for verifying the IIR transfer function, and the red is the result of filtering the analog low pass filter transfer function.