An SNR analysis of aLIGO circuit

Abstract

An SNR model for a mock up of LIGOs photodetector demodulation circuit.

Index Terms

SNR, noise

I. Introduction

This post builds on a few earlier posts of mine, An analysis of aLIGO PD circuit, LIGO modulation and RLC filters. In this post we will look into a mock up of the Advanced Laser Interferometer Gravitational-Wave Observatory (aLIGO) photodiode circuit and construct a model to study the noise and SNR. Once we figure the model out in the simplified scenario, we can extend it out to the full circuit, and see if we can design a new circuit with better noise characteristics than the existing circuit in aLIGO now. I will add a link here to the next post when it comes out. Please check back later.

The actual circuit in use at the moment is shown in Fig. 1, it is a bit too complicated to model for a beginner.

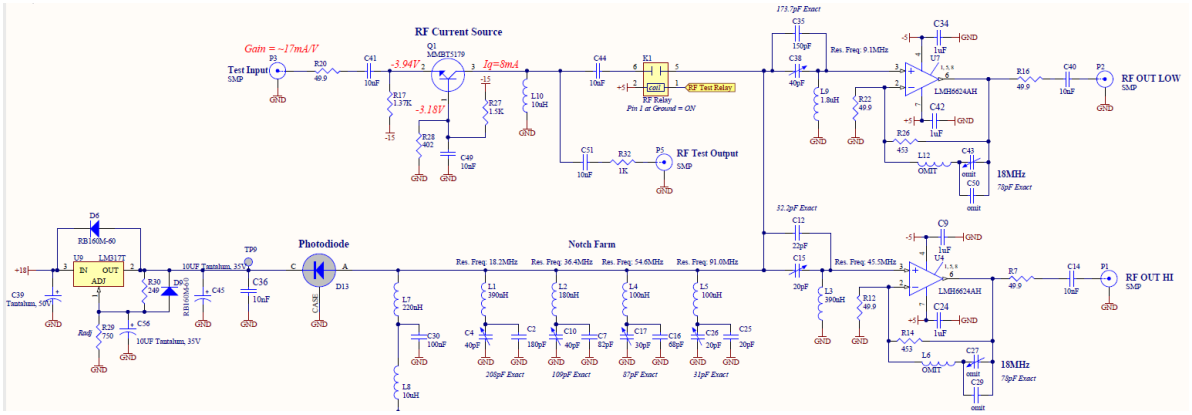

Figure 1: The actual amplifier circuit used in [the current set up.](https://dcc.ligo.org/LIGO-D1101124/public)

II. A mock up

This will be just a warm up exercise where we will reduce the problem to a toy model. We start with modeling the photodiode as a current source with a shunt capacitor and series inductance and capacitance. Its representation can be converted to the Thevenin equivalent as shown in Fig. 2.

email: quarktetra@gmail.com
Find the interactive HTML-document here.

Table I: Spectrum of power (in mW) at various ports and frequencies in two modes of operation [@LIGOT070247].

name	age	gender

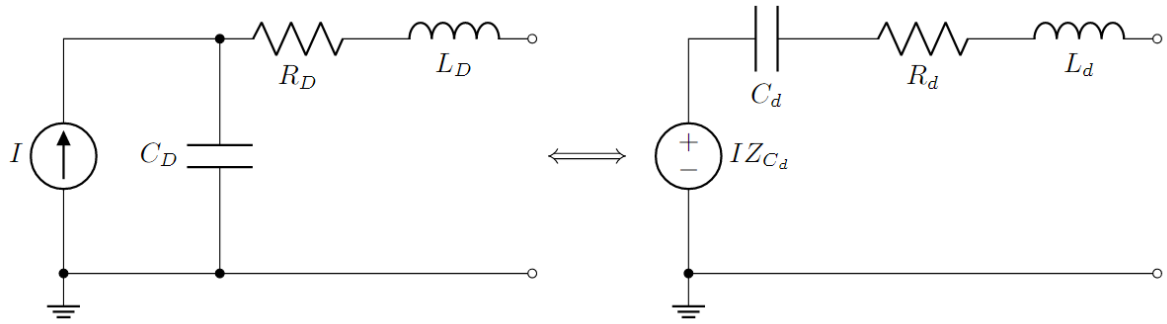

Figure 2: The model for the photodiode at operating frequency Ω. Typical values for the circuit elements are $R_d = 23\Omega$, $C_d = 1$pf, and $L_d = 5$nH. ... [+] The circuit on the right is the Thevenin equivalent.

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

Before we look at the circuit and the full spectrum with its all glory, let's test the waters a bit to define what is the signal and what is the noise for a simplified input spectrum. We will assume, for now, that we want to isolate the 9.1MHz signal at REFL port. We will also assume to only harmonic remaining is at 18.2MHz.

Now, consider connecting this photodiode to a single frequency selector to pick up 9.1MHz and a notch filter to dump the 18.2MHz, as in Fig. 3. We need to keep in mind that the output will be tied to the input of an OPAMP, which will have its noise current flowing through the circuit.

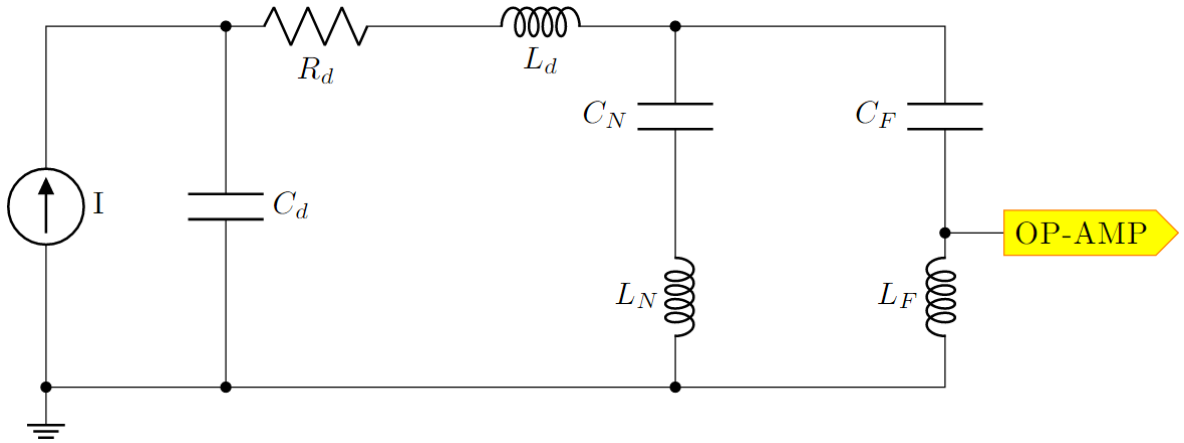

Figure 3: The circuit stripped down to the essentials. ... [+] We will later add more notches and selector .

III. SNR analysis

Let's define a set of impedances to simplify the notation:

$$\begin{aligned} \text{Diode impedance: } Z_{\mathrm{D}} &= Z_{C_D} + Z_{L_D} + R_D, \\ \text{Notch impedance: } Z_N &= Z_{C_N} + Z_{L_N}, \\ \text{Filter impedance: } Z_F &= Z_{C_F} + Z_{L_F}. \end{aligned} \tag{1}$$

Table II: OPAMP noise parameters.

Parameter	Description	Value	Unit
$i_{\rm op}$	Opamp current noise	2.0	$\frac{\rm pA}{\sqrt{\rm Hz}}$
$V_{\rm v}^o$	Opamp voltage noise	0.9	$\frac{\rm nV}{\sqrt{\rm Hz}}$

The total impedance observed by the diode as it is converted to a voltage source in the Thevenin equivalent circuit as shown in Fig. 2 reads:

$$Z_T = Z_{\rm D} + Z_N//Z_F. \tag{2}$$

where for given C or L, we will have $Z_C = \frac{1}{i\omega C}$ and $Z_L = i\omega L$ for ideal capacitors and inductors.

The voltage at the output is given by the simple voltage division:

$$V_{\rm out} = I Z_{C_D} \frac{Z_N//Z_F}{Z_T} \frac{Z_{L_F}}{Z_F} = I \frac{Z_{C_D} Z_N Z_{L_F}}{Z_T(Z_N + Z_F)}. \tag{3}$$

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

$$Z_{\rm tr} \equiv \frac{V_{\rm out}}{I} = \frac{Z_{C_D} Z_N Z_{L_F}}{Z_T(Z_N + Z_F)}. \tag{4}$$

An input current of amplitude $I_{1,2}$ and angular frequency of $\Omega_{1,2}$ will create a voltage at filter output (also the OPAMP input) as

$$V_{1,2}^{\rm out} = I_{1,2} Z_{\rm tr}(\Omega_{1,2}). \tag{5}$$

We can calculate the current amplitudes $I_{1,2}$ using the power values at 9.1MHz and 18.2MHz in Tab. I and using a conversion efficiency of 0.86, which gives $I_1 = 0.07$mA and $I_2 = 17.4$mA.

A photodiode also has the shot noise, and it is transferred to the output just like the signal in Eq. (5), i.e.,

$$V_{\rm shot}^{\rm out} = I_{\rm shot} Z_{\rm tr}, \tag{6}$$

where $I_{\rm shot} = \sqrt{2eI_{\rm dc}}$.

In addition to the signal appearing at the OPAMP port, there will be the thermal noise. That noise is created by the real part of the equivalent impedance, which we will refer to as backwards impedance:

$$Z_{\rm BO} \quad \equiv \quad Z_{L_F} // \left(Z_D // Z_N + Z_{C_F}\right). \tag{7}$$

The thermal voltage reads:

$$\mathcal{V}_{\rm th} = \sqrt{4k_B T \Re\{Z_{\rm BO}\}} \tag{8}$$

The voltage induced by the OPAMP current noise is given by

$$V_{\rm i}^o = i_{\rm op} Z_{\rm BO}. \tag{9}$$

The noise parameters for the OPAMP TI, LMH6624 is given in Tab. II.

The total noise is:

$$\mathcal{N}^{\rm out} = \left|V_{\rm shot}^{\rm out}\right|^2 + \mathcal{V}_{\rm th}^2 + |V_{\rm i}^o|^2 + (V_{\rm v}^o)^2 = |I_{\rm shot} Z_{\rm tr}|^2 + 4k_B T \Re\{Z_{\rm BO}\} + |i_{\rm op} Z_{\rm BO}|^2 + (V_{\rm v}^o)^2, \tag{10}$$

where we included the voltage noise term from OPAMP, $V_{\rm v}^o$. Let us define the noise to signal ratio at frequency Ω_1:

$$\mathrm{NSR} = \frac{\mathcal{N}^{\rm out}}{|V_1^{\rm out}|^2} = \left|\frac{I_{\rm shot}}{I_1}\right|^2 + \frac{4k_B T \Re\{Z_{\rm BO}\} + |i_{\rm op} Z_{\rm BO}|^2 + (V_{\rm v}^o)^2}{|I_1 Z_{\rm tr}|^2}. \tag{11}$$

There are a couple of options to treat the first term in Eq. (11). We can keep the whole expression and use the exact values of I's to get the value of NSR. Or we can follow another convention, where we take the

input signal current I_1 as I_{shot}. In this case I_{shot} will the noise in the input current, and it will be negligibly small, and we will have the following expression for the flipped NSR, i.e.:

$$\text{pSNR} = \left.\frac{\left|I_{\text{shot}} Z_{\text{tr}}\right|^2}{4k_B T\Re\{Z_{\text{BO}}\} + \left|i_{\text{op}} Z_{\text{BO}}\right|^2 + \left(V_{\text{v}}^o\right)^2}\right|_{\Omega_1}, \tag{12}$$

which is not the SNR in the conventional definition, and that is why we defined it as a pseudo SNR as a measure of goodness. What do we do with the $\Omega_2 = 2\,\Omega_1$ term? Should it be a part of the SNR calculations? After all, we are after Ω_1 content and anything else should be considered as noise. However, we should remember that the output of the OPAMP will be fed to a mixer to isolate the Ω_1 content. In fact, we have shown in an earlier post, LIGO modulation, that, with square-wave demodulation, even multiples of the fundamental frequency will be eliminated from the output. Therefore, we don't need to do much about the Ω_2 term. But, if it doesn't hurt the SNR, why bother notching it? It is because there is a significant amount of current in that frequency, as we calculated earlier $I_2 = 17.4$mA, which gets multiplied by the transimpedance $Z_{\text{tr}}(\Omega_2)$. If we completely ignore that signal, it will create a large voltage in the OPAMP input saturating it. The OPAMP is typically set to provide a gain of 10 and its V_{CC} is $\pm$5V. To make sure we stay in the nice, linear region, we need to make sure that the OPAMP input voltage doesn't exceed about 0.1V peak. And that is the reason why we need to sufficiently notch out unwanted signals even though they might be eliminated later by the mixer. We will now turn to some numerical analysis and search for the best parameters that will optimize the SNR while keeping the unwanted signal below a critical voltage value.

Let's do a numerical analysis where we ignore the resistance component of coils- we will turn on the resistance in the next subsection to see how much of a difference they make. Since there are 4 parameters, we can do a complete sweep of parameters. We won't worry about the efficiency of the code for now. Find the full code here or copy it below.

Show the simulation code (R, Python, and LTspice)

Hide

```
# SNR optimizer, 4/6/23, S.O.
# find the details here: https://tetraquark.netlify.app/ebooks/aligorfphotodetectors/aligorfphotodetect
library(plotly);require(utils);library(optimization);library('pracma')
inductorRes=0;# include or exclude inductor resistances
exponentiate <- function(x) {return(10^x)}
paraV<-function(vect) { # recursively calculates the parallel impedance
  if(length(vect)==1){return=vect[1]}; if(length(vect)==2){return=vect[1]*vect[2]/(vect[1]+vect[2])}
  if(length(vect)>2){ vect=c(paraV(vect[1:2]),vect[3:length(vect)]);return=paraV(vect)}
  return;
  }
t1 <- list(family ="Times New Roman",color ="black",size=60);m <- list( l = 50, r = 50, b = 100, t = 10
e_c = 1.602*10^-19;          # electron charge [Coulomb]
kB  = 1.38*10^-23;           # Boltzmann constant [J/K]
Tk   = 300;  # room temperature [deg K]
# opamp LMH6624  https://www.analog.com/media/en/technical-documentation/data-sheets/MAX4106-MAX4107.pd
in0 = 2.0*10^-12;en0 = 0.9*10^-9;
Rf=453 # feedback resistor
Rm=49.9 # V- to GND
oGain=1+Rf/Rm # non-inverting gain #https://dcc.ligo.org/LIGO-T070247/public
f0=0 # use this to inject the DC power into the logplots. should have been 0, but can't take log of it.
pf0= 90.14 # DC power

vppspec=0.1 # max acceptible voltage at the OPAMP input to avoid saturation
f1 = 9.1*10^6;w1=2*pi*f1; pf1=0.08; #power at this frequency, in mW
f2 =2 * f1;   w2=2*pi*f2; pf2=20.19; #power at this frequency, in mW
PDsensitivity=0.86  #ma/mW# https://dcc.ligo.org/LIGO-T070247/public page11 with quantum efficiency ~1
```

```
fmin=1*10^6;fmax=100*10^6 #Range of  frequencies to show in the plots
Idc = 1; #in mA
ishot = sqrt(2 * e_c * Idc *10^(-3))# shot noise
Rd=23;Cd=110*10^-12;Ld=5*10^-9; # diode parameters; serial res, shunt cap, serial inductance.
################################################
Cv=1000*10^(-9);Lv=2.72*10^(-6);# selector filter, these are just guesses
NCv=7.3e-12;NLv=10.4755e-06; # notch filter
normalizer=c(Lv, Cv,NLv,NCv)/100 # will use this in the simulated annealing later

logbaseL=seq(-0.3, 0.3, by = 0.05) # create a grid of equally spaced numbers for L sweep
logbaseC=seq(-0.3, 0, by = 0.05) # create a grid of equally spaced numbers for C sweepa
NlogbaseL=seq(-0.7, 0.7, by = 0.05) # create a grid of equally spaced numbers for L sweep
NlogbaseC=seq(-0.7, 0.7, by = 0.05) # create a grid of equally spaced numbers for C sweep

logbasef=seq(log10(fmin), log10(fmax), by = 0.01) # create a grid of equally spaced numbers for f sweep
fVals=sapply(logbasef,exponentiate);fVals=sort(c(fVals,c(c(f0,f1,f2),c(f1,f2)*1.1,c(f1,f2)*0.9))) # inj
fVals=unique(fVals); # just making sure we don't inject already existing values;  remove duplicates.
DFf=data.frame("fVal"=fVals);DFf$"wVal"=2*pi*DFf$fVal # will use this DF to store various quantities wh


sweepL=unlist(lapply(logbaseL,exponentiate)) # exponentiate these to power 10
sweepC=unlist(lapply(logbaseC,exponentiate))
NsweepL=unlist(lapply(NlogbaseL,exponentiate)) # exponentiate these to power 10
NsweepC=unlist(lapply(NlogbaseC,exponentiate))
s1Lvals=Lv*sweepL;s1Cvals=Cv*sweepC; n1Lvals=NLv*NsweepL;n1Cvals=NCv*NsweepC
Nresf=round(10^(-6)/(NCv*NLv)^0.5/(2*pi),2)



#####################################################################
spectrum<-function(f) {if(f==f1){pf1}else if(f==f2)(pf2)else if(f==f0)(pf0)else{0}}

DFf$'spectPower'<-  mapply(spectrum,DFf$fVal)
DFf$'spectmA'<- DFf$'spectPower'*PDsensitivity
pf2*PDsensitivity
DFfsub<-DFf[DFf$fVal<f2*1.05,] # remove high freq for the plot only
figSp<-plot_ly()%>%  add_trace( x = DFfsub$fVal/10^6,   y =  DFfsub$'spectPower', mode = 'markers+lines
figSp <- figSp %>% layout(
      xaxis = list(ztype="log",title = "TeX($$f[\\text{MHz}]$$)", font = t1,dtick =1 ,range=list(-0.5,2
      yaxis = list(type="log",title = "TeX($$S(f)[\\text{mW}]$$)",font = t1 )
    )
#figSp%>% config(mathjax = "cdn")

DFf<-DFf[DFf$fVal>f0,] # drop f0 at this point


zd <-function(omega) { Rd+ 1i*Ld*omega-  1i/(omega* Cd)} # diode serial impedance
zC<-function(omega,Cv){- 1i/(omega*Cv)}  #  impedance of a capacitor
zL<-function(omega,Lv){
  1i*omega*Lv +  0*1.05*10^(-6)*omega^0.5 *Lv  + inductorRes* 0.058*Lv^0.5 *10^4.5
   }  #impedance of an inductor  second term is the skin depth effect, 3rd is the serial res
zLC<- function(omega,Lv,Cv) {zC(omega,Cv)+zL(omega,Lv)} # impedance of a  series L and C.
zTot<-function(omega,Lv,Cv,NLv,NCv) {paraV(c(zLC(omega,Lv,Cv),zLC(omega,NLv,NCv))) +zd(omega) }
```

```
ztr<- function(omega,Lv,Cv,NLv,NCv) {zC(omega,Cd)*zLC(omega,NLv,NCv)* zL(omega,Lv)/(zTot(omega,Lv,Cv,NL

voltsw2<- function(Lv,Cv,NLv,NCv) {abs(ztr(w2,Lv,Cv,NLv,NCv) )*pf2*PDsensitivity/1000 }


zbo<- function(omega,Lv,Cv,NLv,NCv) {paraV(c(zL(omega,Lv) ,paraV(c(zd(omega),zLC(omega,NLv,NCv))) +zC(o

V1outsq<- function(omega,Lv,Cv,NLv,NCv) {abs(ztr(omega,Lv,Cv,NLv,NCv))^2 * ishot^2};
VXoutsq<- function(Lv,Cv,NLv,NCv) {abs(ztr(w2,Lv,Cv,NLv,NCv))^2 * ishot^2};

N1outth<- function(omega,Lv,Cv,NLv,NCv) {4*kB*Tk*abs(Re(zbo(omega,Lv,Cv,NLv,NCv)))}
N1outOPv<- function(omega,Lv,Cv,NLv,NCv) {en0^2}
N1outOPi<- function(omega,Lv,Cv,NLv,NCv) {abs(in0*zbo(omega,Lv,Cv,NLv,NCv))^2}

N1out<- function(omega,Lv,Cv,NLv,NCv) {N1outth(omega,Lv,Cv,NLv,NCv)+N1outOPv(omega,Lv,Cv,NLv,NCv)+N1out
SNRoutdb<-function(Lv,Cv,NLv,NCv) { (V1outsq(w1,Lv,Cv,NLv,NCv)/N1out(w1,Lv,Cv,NLv,NCv))^0.5}

nSNRoutdbX<-function(x) {  Lv <- normalizer[1]*x[1];Cv<- normalizer[2]*x[2];NLv<- normalizer[3]*x[3];NC
returnV=-(V1outsq(w1,Lv,Cv,NLv,NCv)/N1out(w1,Lv,Cv,NLv,NCv))^0.5
if(voltsw2(Lv,Cv,NLv,NCv)>vppspec){returnV=99999999}
return(returnV)
}

dtdoe2=expand.grid(s1Lval = s1Lvals, s1Cval=s1Cvals,n1Lval = n1Lvals, n1Cval=n1Cvals)
dtdoe2$snr<-mapply( SNRoutdb,dtdoe2$s1Lval,dtdoe2$s1Cval,dtdoe2$n1Lval,dtdoe2$n1Cval)
dtdoe2$'ztrw2'<-mapply( ztr,w2,dtdoe2$s1Lval,dtdoe2$s1Cval,dtdoe2$n1Lval,dtdoe2$n1Cval)
dtdoe2$'ztrw1'<-mapply( ztr,w1,dtdoe2$s1Lval,dtdoe2$s1Cval,dtdoe2$n1Lval,dtdoe2$n1Cval)
dtdoe2$'zTot'<-mapply( zTot,w1,dtdoe2$s1Lval,dtdoe2$s1Cval,dtdoe2$n1Lval,dtdoe2$n1Cval)

dtdoe2$'volts'<-abs(dtdoe2$'ztrw2')*pf2*PDsensitivity/1000 # in volts

dtdoe2s=dtdoe2[dtdoe2$'volts'<vppspec,] ##
opt2=dtdoe2s[dtdoe2s$snr==max(dtdoe2s$snr),]

dtdoe2Backup<-dtdoe2
dtdoe2<-dtdoe2[dtdoe2$n1Lval==opt2$n1Lval[1]&dtdoe2$n1Cval==opt2$n1Cval[1], ]
Cbest=opt2$s1Cval[1];Lbest=opt2$s1Lval[1];wbest=1/sqrt(Cbest*Lbest)


SNRbest=round(opt2$snr[1],2)

print(opt2)

#dtres=dtdoe2[abs(dtdoe2$'ztrw1')>0.99*max(abs(dtdoe2$'ztrw1')),]

dtres=dtdoe2[abs(dtdoe2$'zTot')<1.01*min(abs(dtdoe2$'zTot')),]

optParam=results$par*normalizer; #print(optParam)

oLv=opt2$s1Lval[1]
oCv=opt2$s1Cval[1]
oNLv=opt2$n1Lval[1]
oNCv=opt2$n1Cval[1]
```

```
oNresf=round(10^(-6)/(oNCv*oNLv)^0.5/(2*pi),2)




DFf$'ztr'<-  mapply(ztr,DFf$wVal,oLv, oCv,oNLv,oNCv)  ;DFf$'absztr'<-abs(DFf$'ztr')
DFf$'zbo'<-  mapply(zbo,DFf$wVal,oLv, oCv,oNLv,oNCv)  ;DFf$'abszbo'<-abs(DFf$'zbo')
DFf$'Volt'<-DFf$'ztr'*DFf$spectmA;DFf$'VoltRMS'<-abs(DFf$'Volt')
DFf$'N1outsq'<-  mapply(N1out,DFf$wVal,oLv, oCv,oNLv,oNCv);DFf$'N1out'<-(DFf$'N1outsq')^0.5*10^9
DFf$'N1outthsq'<-  mapply(N1outth,DFf$wVal,oLv, oCv,oNLv,oNCv);DFf$'N1outth'<-(DFf$'N1outthsq')^0.5*10^9
DFf$'N1outOPisq'<-  mapply(N1outOPi,DFf$wVal,oLv, oCv,oNLv,oNCv);DFf$'N1outOPi'<-(DFf$'N1outOPisq')^0.5*
DFf$'N1outOPvsq'<-  mapply(N1outOPv,DFf$wVal,oLv, oCv,oNLv,oNCv);DFf$'N1outOPv'<-(DFf$'N1outOPvsq')^0.5*
Dresf=round(10^(-6)*DFf$fVal[DFf$'absztr'==max(DFf$'absztr')],1)

print(paste0("circuit res:",Dresf))

dtResonate=data.frame("s1Cval"=unique(dtdoe2$s1Cval))
dtResonate$LofCf1<-1/(w1^2*dtResonate$s1Cval)
dtResonate$LofCf2<-1/(w2^2*dtResonate$s1Cval)
dtResonate1<-dtResonate[dtResonate$LofCf1>=min(dtdoe2$s1Lval),]
dtResonate2<-dtResonate[dtResonate$LofCf2>=min(dtdoe2$s1Lval),]

dtdoe2$s1Cval<-round(dtdoe2$s1Cval*10^9)/10^9
dtdoe2$s1Lval<-round(dtdoe2$s1Lval*10^7)/10^7
if(oCv*10^9>1){CbestText=paste0(round(oCv*10^9,1),"nF") }else{CbestText=paste0(round(oCv*10^12,1),"pF")}
if(oCv*10^6>1){CbestText=paste0(round(oCv*10^6,1),"\\mu F") }
NresText=paste0(" \\quad    C_N=",round(oNCv*10^12,1), "pF; L_N=",round(oNLv*10^6,1),"\\mu H;")
print(NresText)

fig<-plot_ly(
   hovertemplate = "Value: %{z:.2f}<br>C: %{x}<br>L: %{y}",

  x = dtdoe2$s1Cval, y = dtdoe2$s1Lval, z =  dtdoe2$snr, type = "contour",name="SNR")%>%
 # add_trace( x = dtResonate1$s1Cval,   y =  dtResonate1$LofCf1, name = paste0('f1:',round(f1*10^-6,1),
  add_trace( x = dtResonate2$s1Cval, y =  dtResonate2$LofCf2, name = paste0('f2:',round(f2*10^-6,1),"MH
  add_trace( x = dtres$s1Cval, y =  dtres$s1Lval, name = paste0(round(f1*10^-6,1),"MHz resonance"), mod
             hoverinfo='skip'
            )
fig <- fig %>% layout(
 title=paste0("TeX($$ \\text{Notch parameters:}",  NresText," \\text{and  sweep of filter parameters} $$
      xaxis = list(type="log",title = "TeX($$C_F[\\text{F}]$$)", font = t1  ),
      yaxis = list(type="log",title = "TeX($$L_F[\\text{H}]$$)",font = t1 )
    )
xpos=-20*log10(Cbest/Cv*10^12)/(log10(max(Cvals)/Cv*10^12)-log10(min(Cvals)/Cv*10^12))
a <- list( x =log10(Cbest),y = log10(Lbest), bgcolor="white",
  text = paste0("$$\\text{Best SNR=}",SNRbest," \\text{ at }  C_F=",CbestText,"; L_F=",round(Lbest*10^6
  xref = "x", yref = "y", showarrow = TRUE,arrowhead = 7,arrowcolor='white',ax = xpos,ay = -50)

fig <- fig %>% layout(annotations = a)
fig%>%config(mathjax = "cdn")


dtdoe2s<-dtdoe2Backup
```

```
dtdoe2s<-dtdoe2s[dtdoe2s$s1Lval==opt2$s1Lval[1], ]
dtdoe2s<-dtdoe2s[dtdoe2s$s1Cval==opt2$s1Cval[1], ]

Cbestn=opt2$n1Cval[1]
Lbestn=opt2$n1Lval[1]
wbestn=1/sqrt(Cbestn*Lbestn)
opt2mVolts= signif(opt2$volts[1]*1000,1)

ndtResonate=data.frame("n1Cval"=unique(dtdoe2s$n1Cval))
ndtResonate$LofCf1<-1/(w1^2*ndtResonate$n1Cval)
ndtResonate$LofCf2<-1/(w2^2*ndtResonate$n1Cval)

ndtResonate1<-ndtResonate[ndtResonate$LofCf1>=min(dtdoe2s$n1Lval),]
ndtResonate2<-ndtResonate[ndtResonate$LofCf2>=min(dtdoe2s$n1Lval),]
ndtResonate1<-ndtResonate[ndtResonate$LofCf1<=max(dtdoe2s$n1Lval),]
ndtResonate2<-ndtResonate[ndtResonate$LofCf2<=max(dtdoe2s$n1Lval),]


#dtdoe2s$n1Cval<-round(dtdoe2s$n1Cval*10^14)/10^14
#dtdoe2s$n1Lval<-round(dtdoe2s$n1Lval*10^9)/10^9

fig2<-plot_ly(
     hovertemplate = "Value: %{z:.2f}<br>C: %{x}<br>L: %{y}",

  x = dtdoe2s$n1Cval, y = dtdoe2s$n1Lval, z =  dtdoe2s$snr, type = "contour", name="Value")%>%  add_tra
  add_trace( x = ndtResonate2$n1Cval, y =  ndtResonate2$LofCf2, name = paste0('f2:',round(f2*10^-6,1),"
fig2 <- fig2 %>% layout(
  title=paste0("TeX($$ \\text{Filter parameters: }   C_F=",CbestText,"; L_F=",round(Lbest*10^6,2),"\\mu
      xaxis = list(type="log",title = "TeX($$C_N[\\text{F}]$$)", font = t1  ),
      yaxis = list(type="log",title = "TeX($$L_N[\\text{H}]$$)",font = t1 )
    )
if(Cbestn*10^9>1){CbestTextn=paste0(round(Cbestn*10^9,1),"nF") }else{CbestTextn=paste0(round(Cbestn*10^
xpos=50
a <- list( x =log10(Cbestn),y = log10(Lbestn), bgcolor="white",
  text = paste0("$$ \\text{ Bet SNR}=",SNRbest,"\\text{ at } C_N=",CbestTextn,"; L_N=",round(Lbestn*10^
  xref = "x", yref = "y", showarrow = TRUE,arrowhead = 7,arrowcolor='white',ax = xpos,ay = 50)
fig2 <- fig2 %>%
  layout(annotations = a)
fig2%>% config(mathjax = "cdn")



fig2V<-plot_ly(
   hovertemplate = "Value: %{z:.2f}<br>C: %{x}<br>L: %{y}",
  x = dtdoe2s$n1Cval, y = dtdoe2s$n1Lval, z =  log10(dtdoe2s$volts), type = "contour", name="Value")%>%
  add_trace( x = ndtResonate1$n1Cval,   y =  ndtResonate1$LofCf1, name = paste0('f1:',round(f1*10^-6,1)
  add_trace( x = ndtResonate2$n1Cval, y =  ndtResonate2$LofCf2, name = paste0('f2:',round(f2*10^-6,1),"
fig2V <- fig2V %>% layout(title="TeX($$\\text{Log10 of the voltage at the OPAMP input at 18.2 MHz: }
      xaxis = list(type="log",title = "TeX($$C_N[\\text{F}]$$)", font = t1  ),
      yaxis = list(type="log",title = "TeX($$L_N[\\text{H}]$$)",font = t1 )
    )
xpos=150
a <- list( x =log10(Cbestn),y = log10(Lbestn), bgcolor="white",
```

```
  text = paste0("C:",CbestTextn,"; L:",round(Lbestn*10^6,2),"uH; f:",round(wbestn/10^6/(2*pi) ,1), "MHz
  xref = "x", yref = "y", showarrow = TRUE,arrowhead = 7,arrowcolor='white',ax = xpos,ay = 50)
fig2V <- fig2V %>%
  layout(annotations = a)
fig2V%>% config(mathjax = "cdn")

xpos=log10(wbestn/(2*pi))
dtdoe2s$fn<-(dtdoe2s$n1Cval*dtdoe2s$n1Lval)^(-0.5)/(2*pi)
dtdoe2s2<-dtdoe2s[dtdoe2s$fn<wbestn/(2*pi)*1.2&dtdoe2s$fn>wbestn/(2*pi)*0.8,]

fig2VT<-plot_ly(
     hovertemplate = "Value: %{z:.2f}<br>C: %{x}<br>L: %{y}",

  x = dtdoe2s2$fn, y = dtdoe2s2$n1Lval, z =  log10(dtdoe2s2$volts), type = "contour", name="out")
fig2VT <- fig2VT %>% layout(title="TeX($$\\text{Log10 of the voltage at the OPAMP input at 18.2 MHz: }
      xaxis = list(type="log",title = "TeX($$f_\\text{N}$$)", font = t1  ),
      yaxis = list(type="log",title = "TeX($$\\text{L}[\\text{H}]$$)",font = t1 )
    )

a <- list( x =xpos,y = log10(Lbestn), bgcolor="white",
  text = paste0("$$C_N=",CbestTextn,"; L_N:",round(Lbestn*10^6,2),"\\mu H; f=",round(wbestn/10^6/(2*pi)
  xref = "x", yref = "y", showarrow = TRUE,arrowhead = 7,arrowcolor='white',ax = 0.5,ay = 50)

fig2VT <- fig2VT %>% layout(annotations = a)

fig2VT%>% config(mathjax = "cdn")


figF<-plot_ly()%>%  add_trace(line=list (shape = 'spline', smoothing= 1.3), x = DFf$fVal,   y =  DFf$'al
figF <- figF %>% layout(
      title=paste0("$C_F=",CbestText,"; L_F=",round(oLv*10^6,2),"\\mu H; ", NresText, "$" ),
      xaxis = list(type="log",title = "TeX($$f[\\text{MHz}]$$)", font = t1 ),
      yaxis = list(type="log",title = "TeX($$|Z_\\text{tr}(f)|[\\Omega]$$)",font = t1 ,hoverformat = ".
      yaxis2 = list(type="log",title = "TeX($$V_\\text{in}(f)[\\text{mV}]$$)",overlaying = "y", side =
    )

figF%>% config(mathjax = "cdn")

figS <-plot_ly(z = ~xtabs(snr ~ s1Lval + s1Cval, data = dtdoe2),x = unique(dtdoe2$s1Cval), y = unique(dt
figS <- figS %>% layout(
    scene = list(
      xaxis=list(title="C",type="log"),
      yaxis=list(title="L",type="log"),
      zaxis=list(title="SNR"),
      camera=list(
        eye = list(x=1.87, y=0.88, z=-0.64)
        )
      )
)
#figS


pVect=c(paste0(".param Cv " ,round(oCv*10^9,2),'n'),
```

```
  paste0(".param Lv " ,round(oLv*10^9,2),'n'),
  paste0(".param Rv " ,round(0.001+inductorRes*0.058*oLv^0.5 *10^4.5,3)),
  paste0(".param NCv " ,round(oNCv*10^12,2),'p'),
  paste0(".param NLv " ,round(oNLv*10^9,2),'n'),
  paste0(".param NRv " ,round(0.001+inductorRes*0.058*oNLv^0.5 *10^4.5,3))
)



#fileConn<-file("LSC RFPD Simulation Files/paramE.txt");writeLines(pVect, fileConn);close(fileConn)

thepath="C:\\Users\\451516\\Documents\\github\\aLIGOrfPhotoDetectors\\LSC RFPD Simulation Files\\9_45_LS
thepath="C:\\Users\\451516\\Documents\\github\\aLIGOrfPhotoDetectors\\LSC RFPD Simulation Files\\9_45_LS
thepath="C:\\Users\\451516\\Documents\\github\\aLIGOrfPhotoDetectors\\LSC RFPD Simulation Files\\9_45_LS

dfLT<-as.data.frame(read.table(thepath,sep="\t", header=TRUE,check.names = FALSE))
#cat(paste('<script>  LTdata_SE= ',toJSON(df),'</script>', sep=""))

figN<-plot_ly()%>%  add_trace(line=list (shape = 'spline', smoothing= 1.3,color='red'), x = DFf$fVal,
  add_trace(line=list (shape = 'spline', smoothing= 1.3,color='red',dash='dot'), x = dfLT$frequency,
  add_trace(line=list (shape = 'spline', smoothing= 1.3,color='green'), x = DFf$fVal,   y =  DFf$'N1out
  #add_trace(line=list (shape = 'spline', smoothing= 1.3,color='green',dash='dot'), x = dfLT$frequency,
  add_trace(line=list (shape = 'spline', smoothing= 1.3,color="orange"), x = DFf$fVal,   y =  DFf$'N1ou

figN <- figN %>% layout(      title=paste0("$\\text{NOISE break out vs } f \\text{ at }  C_F=",CbestTex
      xaxis = list(type="log",title = "TeX($$f[\\text{MHz}]$$)", font = t1 ,range=list(6.7,7.7) ),
      yaxis = list(type="log",title = "TeX($$|\\mathcal{N}(f)|[nV/\\sqrt{Hz}]$$)",font = t1 ,range=list
      yaxis2 = list(type="log",title = "TeX($$V_\\text{in}(f)[\\text{mV}]$$)",overlaying = "y", side =
    )

figN%>% config(mathjax = "cdn")

#ztrRatio=DFf$'absztr'[DFf$fVal==f1]/DFf$'absztr'[DFf$fVal==f2]
```

We first show the spectrum in Fig. 4. We will take the photodiode conversion sensitivity as 0.86 mA/mW [1] to convert the laser power to current.

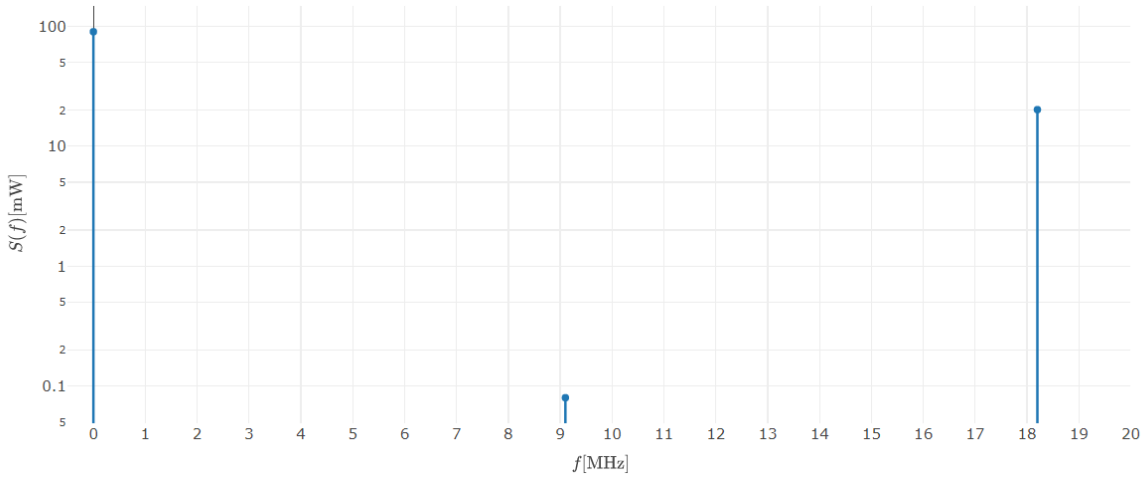

Figure 4: The power spectrum for the toy model. We keep DC power and the content at 9.1 and 18.2 MHz as listed in Tab.
reftab:spect.

We will do a sweep for all four parameters:

1. C_F : [501, 562, 631, 708, 794, 891, 1000] nF
2. L_F : [1.4, 1.5, 1.7, 1.9, 2.2, 2.4, 2.7, 3.1, 3.4, 3.8, 4.3, 4.8, 5.4] μ H
3. C_N : [1.5, 1.6, 1.8, 2.1, 2.3, 2.6, 2.9, 3.3, 3.7, 4.1, 4.6, 5.2, 5.8, 6.5, 7.3, 8.2, 9.2, 10.3, 11.6, 13, 14.6, 16.3, 18.3, 20.6, 23.1, 25.9, 29.1, 32.6, 36.6] pF
4. L_N : [2.1, 2.3, 2.6, 3, 3.3, 3.7, 4.2, 4.7, 5.3, 5.9, 6.6, 7.4, 8.3, 9.3, 10.5, 11.8, 13.2, 14.8, 16.6, 18.6, 20.9, 23.5, 26.3, 29.5, 33.1, 37.2, 41.7, 46.8, 52.5] μ H

After the sweep, we pick the best SNR point while maintaining $V_p < 0.1V$. Figure 5 shows the contour plot for the SNR after the notch filter parametes are locked to their values as shown in the plot title.

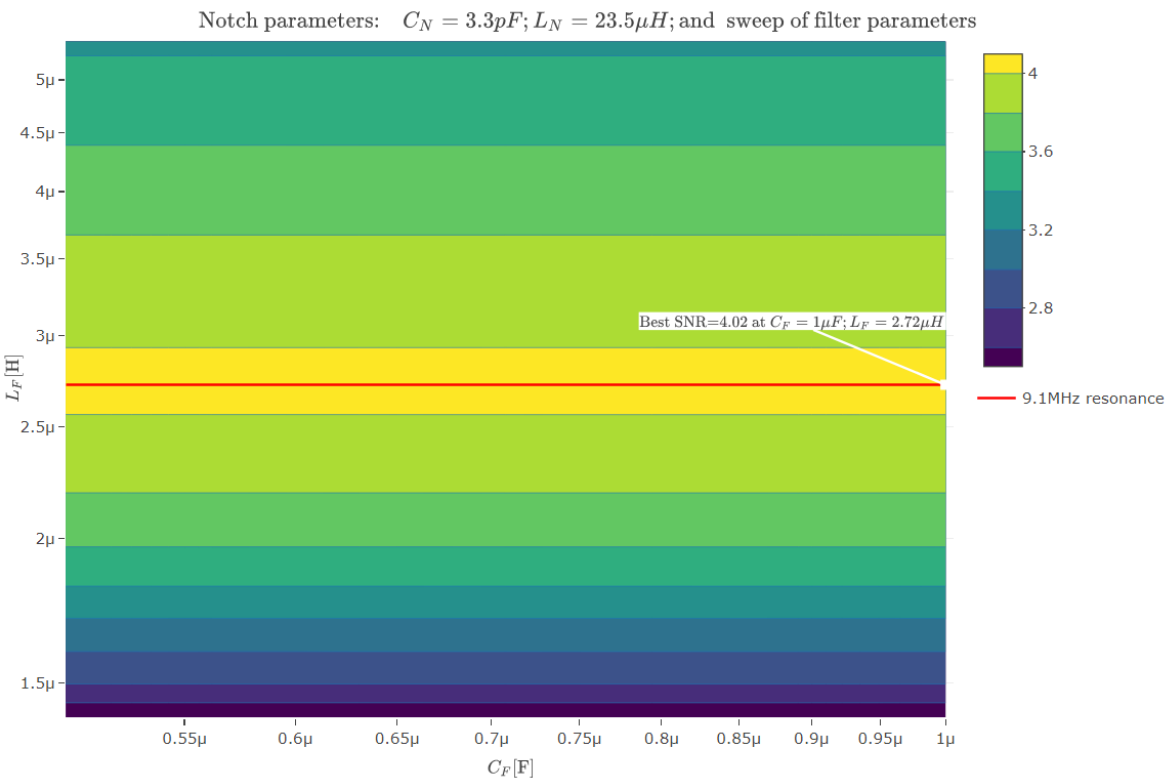

Figure 5: Contour plot for SNR after the notch filter parameters are fixed to maintain the voltage below the V_p spec.

We can now fix the filter parameters to the values that maximize the SNR and look at the change in SNR, as in Fig. 6.

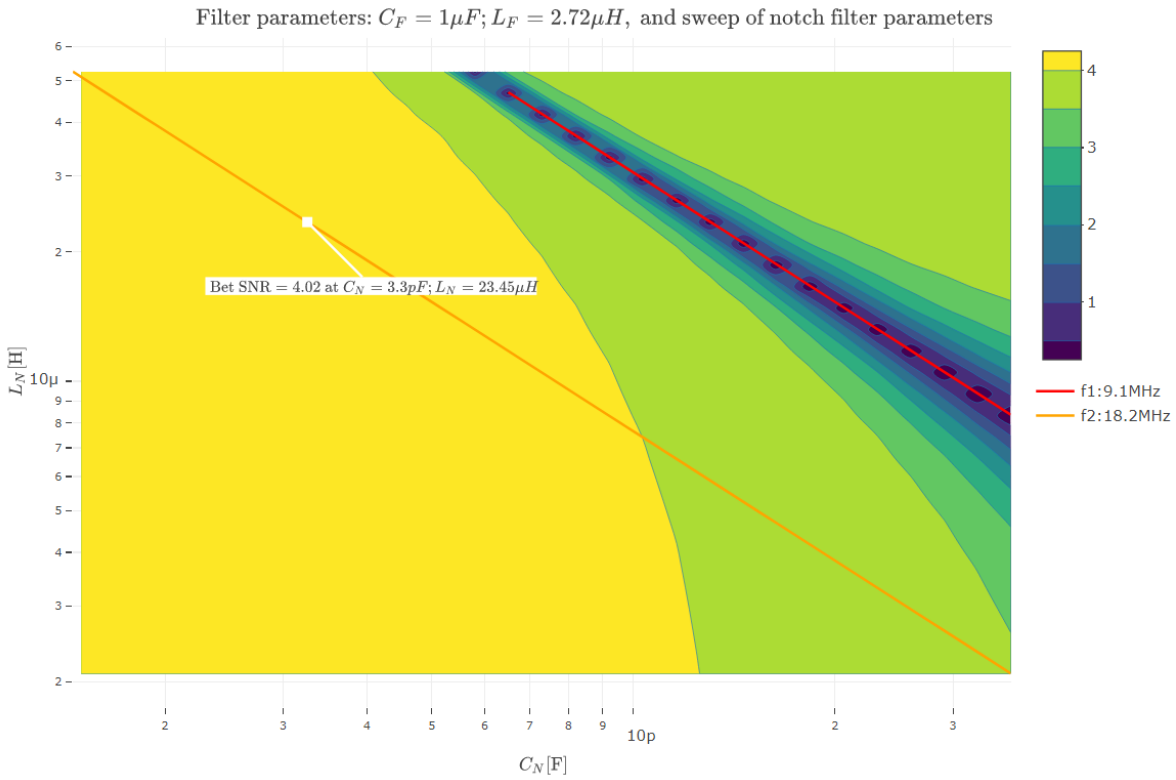

Figure 6: Contour plot for SNR after the notch filter parameters are fixed to maintain the voltage below the maximum voltage spec. The lines show the values that give 9.1 and 18.2 MHz frequencies. The notch parameters end up at the 18.2 MHz line, as on would predict.

Figure 6 shows that SNR has very little sensitivity around the point we selected. This is expected since the main purpose of the notch is to get rid of the 18.2MHz. Let's take a look at the voltage in Fig. 7.

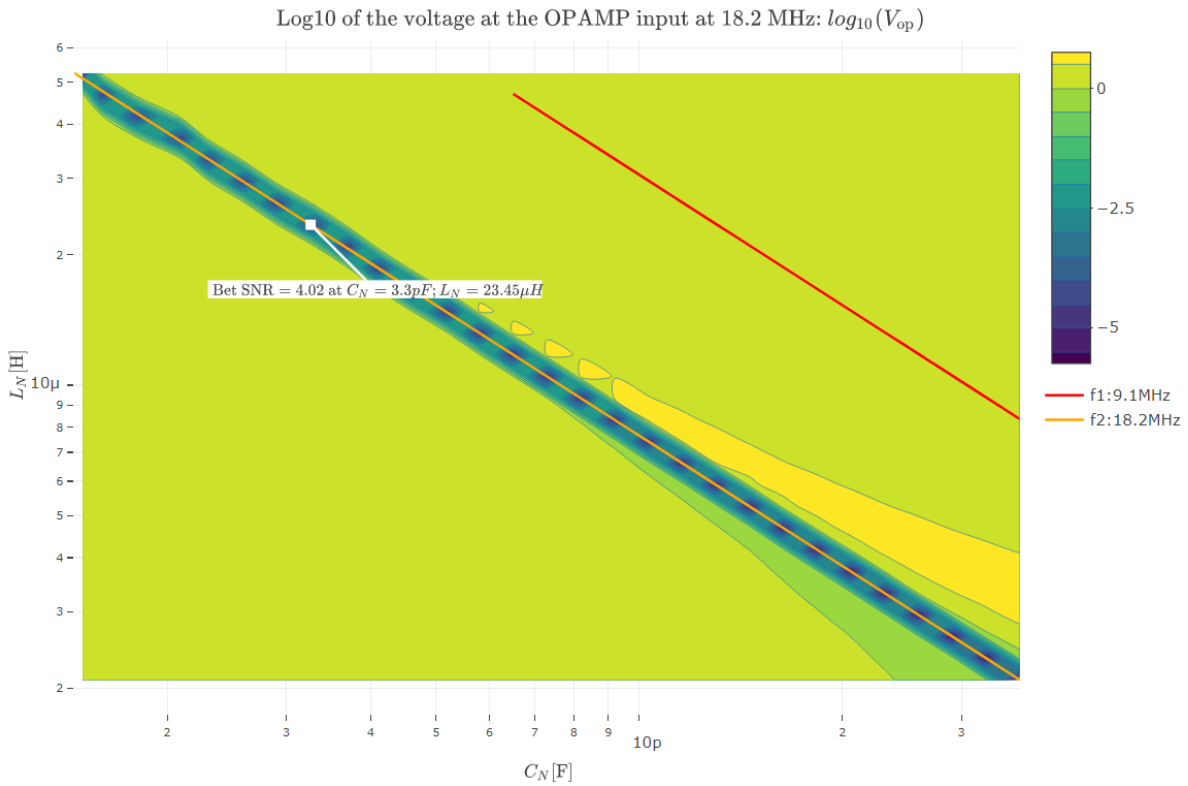

Figure 7: Contour plot for voltage after the notch filter parameters are fixed to maintain the voltage below the maximum voltage spec. .

It is hard to zoom into the diagonal line in that figure. We can rotate it by defining $f_N = \frac{1}{2\pi\sqrt{L_N C_N}}$, and replot as in Fig. 8.

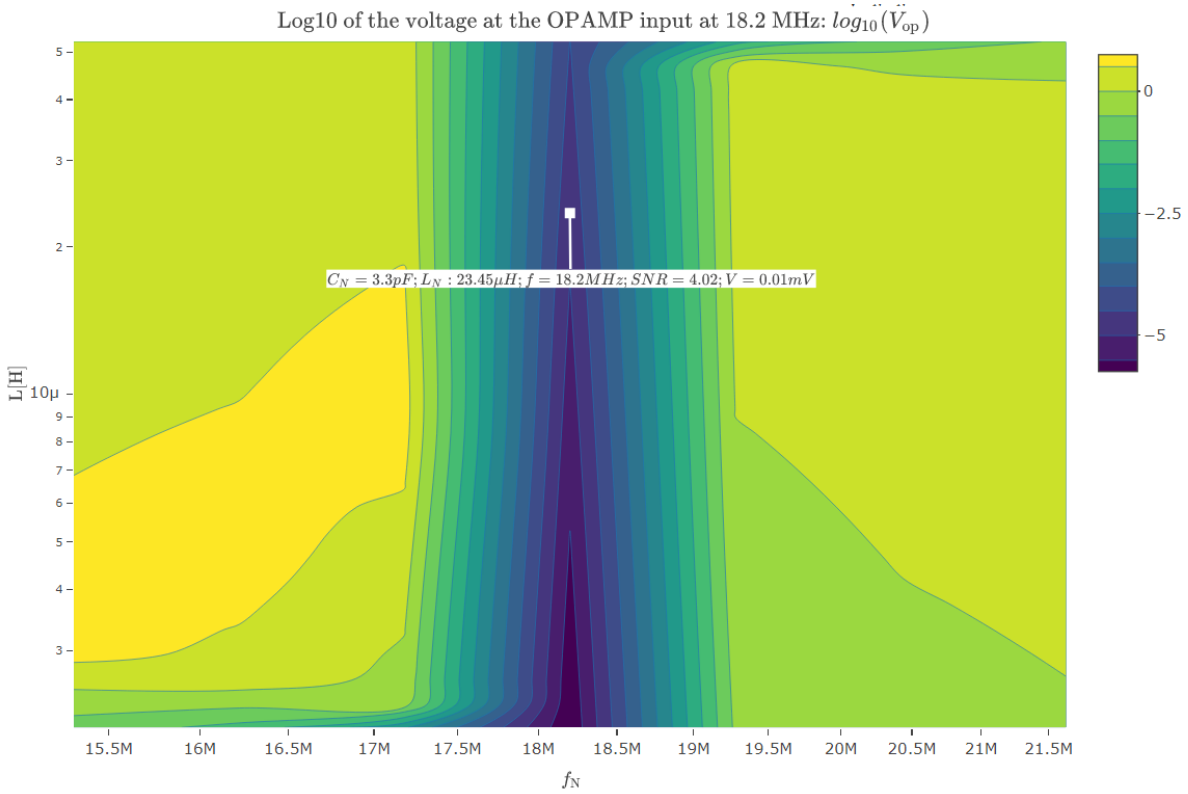

Figure 8: Contour plot for voltage after the selector filter parameters are fixed at the best SNR point. The voltage can be kept below the maximum voltage spec if the notch parameters are set around the 18.2MHz frequency.

We plot Fig. 9 to show the transimpedance of the circuit and the expected voltages[rigth axis] at the opamp input at the two frequencies of interest.

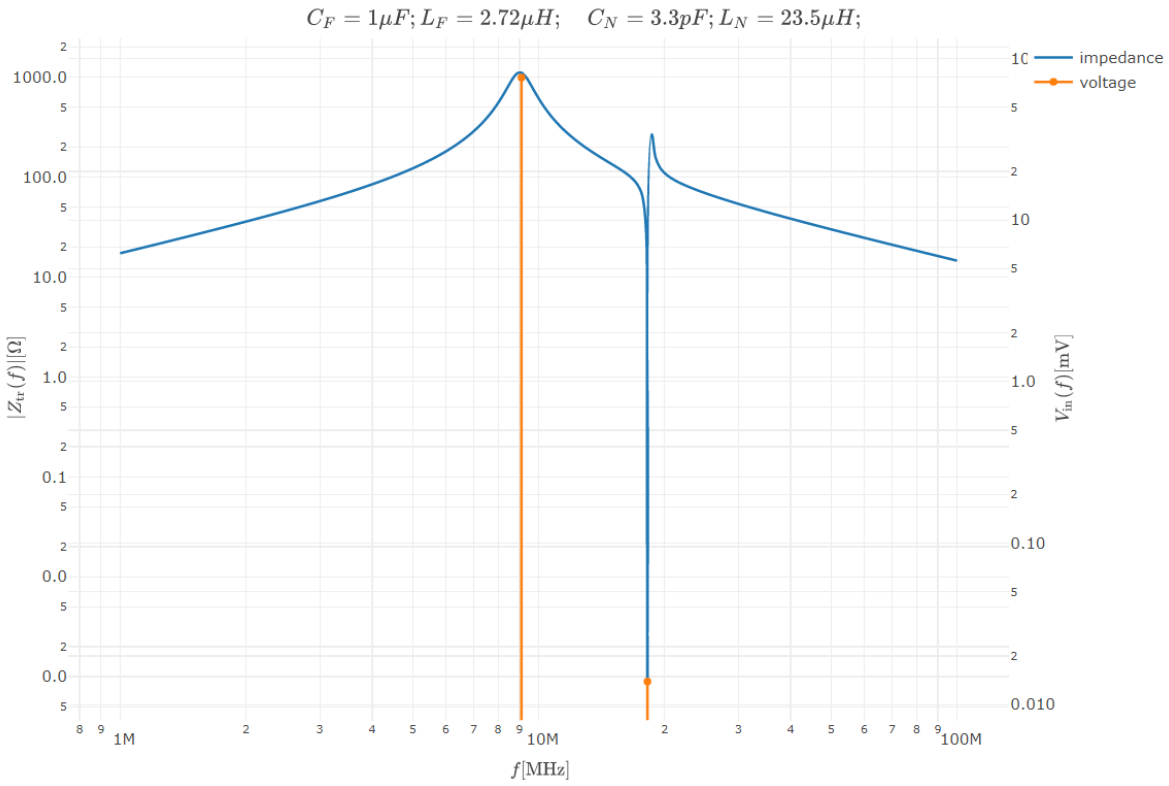

Figure 9: Transimpedance and the voltage at the OPAMP input[right axis]. The notch at 18.2 MHz is very narrow since we ignored the resistances of coils. Once we add that, the notch will widen.

We will treat the LTspice simulation as the golden standard, and it is crucial that the results from this analysis are in agreement with the ones from LTspice. As we will have to do this many times, it is worth automating the process. We will do the optimization in R (or possibly in Python later), find the parameters; pass them to LTspice. We can collect the results of the simulation by parsing the *.raw* data and compare and visualize it using R/Javascript.

Figure 10 shows the total noise computed here and in LTspice and the break out into the components.

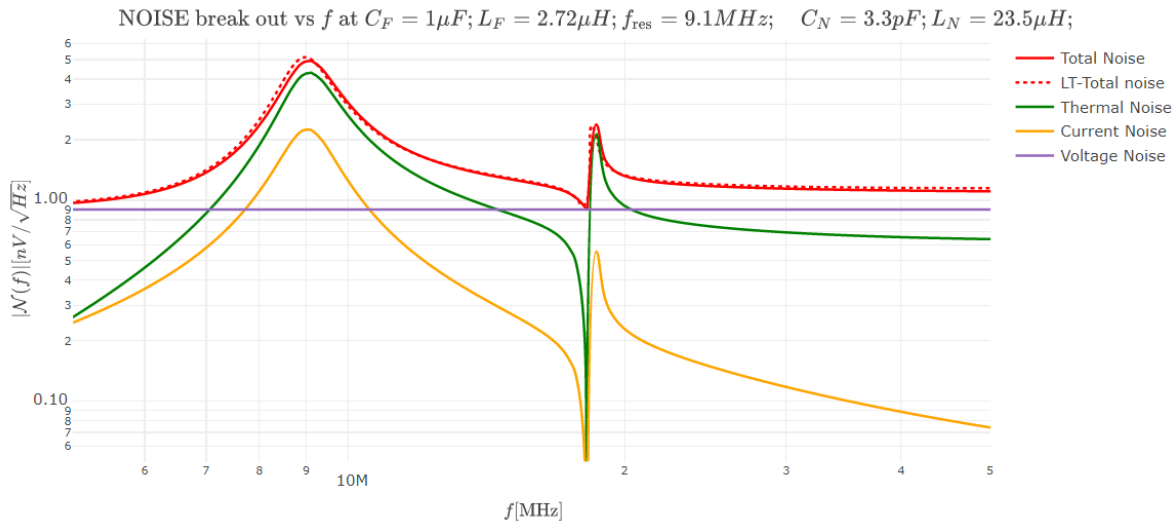

Figure 10: Total noise and the break out.

We see in Fig. 10 that the analysis here matches the results from LT spice simulation, which can be treated as the golden standard, and matching it with our mathematical model is very promising.

IV. What is next?

Now that we have the model for the simplified circuit, we can apply the same formalism to the full circuit. Before we do that, we need to take care of one more thing: coils have resistances. We first need to build a realistic model of coils and then use that for the complete circuit. I will include a link here when that analysis out. Stay tuned.

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