4Gen TTFFS

Setup

In[11]:=
```
Needs["Controls`LinearControl`"]
```

In[12]:=
```
$TextStyle = {FontFamily → "Helvetica", FontSize → 13};
```

In[13]:=
```
plotopt =
  Sequence @@ {GridLines → Automatic, Frame → True, FrameStyle → Thickness[0.0025],
    PlotStyle → {Darker[Green], Blue, Red}, BaseStyle → {FontSize → 13}};
```

In[14]:=
```
plotoptn[n_Integer ? (# > 0 ∧ # < 8 &)] :=
  Sequence @@ {GridLines → Automatic, Frame → True, FrameStyle → Thickness[0.0025],
    PlotStyle → Take[{Gray, Orange, Purple, Brown, Darker[Green], Blue, Red}, -n],
    BaseStyle → {FontSize → 13}};
plotoptn[n_Integer ? (# ≤ 0 ∨ # ≥ 8 &)] := plotopt
```

In[16]:=
```
mylegend[labels_List, pos_ : Right] :=
 {Placed[LineLegend[labels, LabelStyle → {FontSize → 11}, LegendMargins → 2,
    LegendFunction → (Framed[#, Background → White] &)], pos]}
```

Free Running Laser Noise

In[17]:=
```
npro[f_] := 1*^4 / f  (*Hz/ √Hz *)
```

Equations

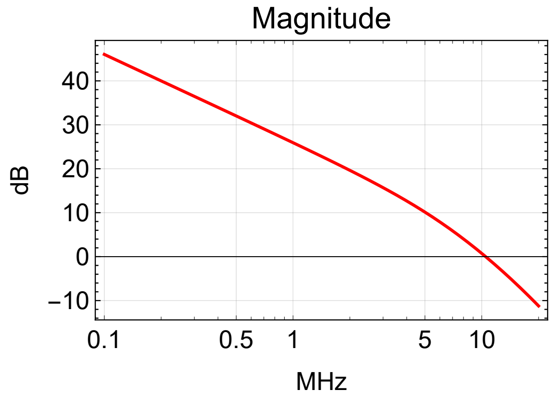
EOM Actuator Path

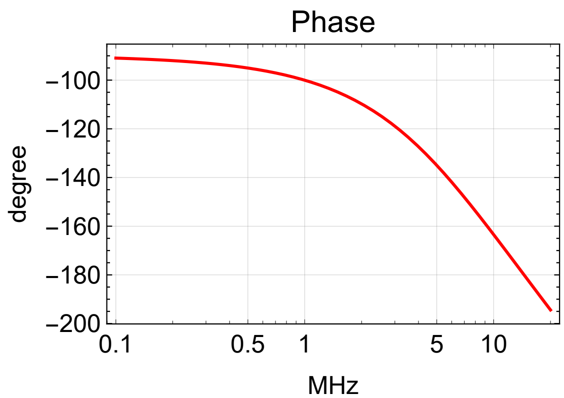
PA98 Open Loop Gain

Data sheet at www.apexanalog.com.

In[56]:=
```
prmpa98 = {gpa → 2*^5, spa → 2 π 100, spa2 → 2 π 7*^6, spa3 → 2 π 30*^6, rpa → 50};
pa98[s_] := gpa pole[s, spa] pole[s, spa2] × pole[s, spa3]
  (* heuristic model representing the puplished curves with Cc = 20 pF *)
```

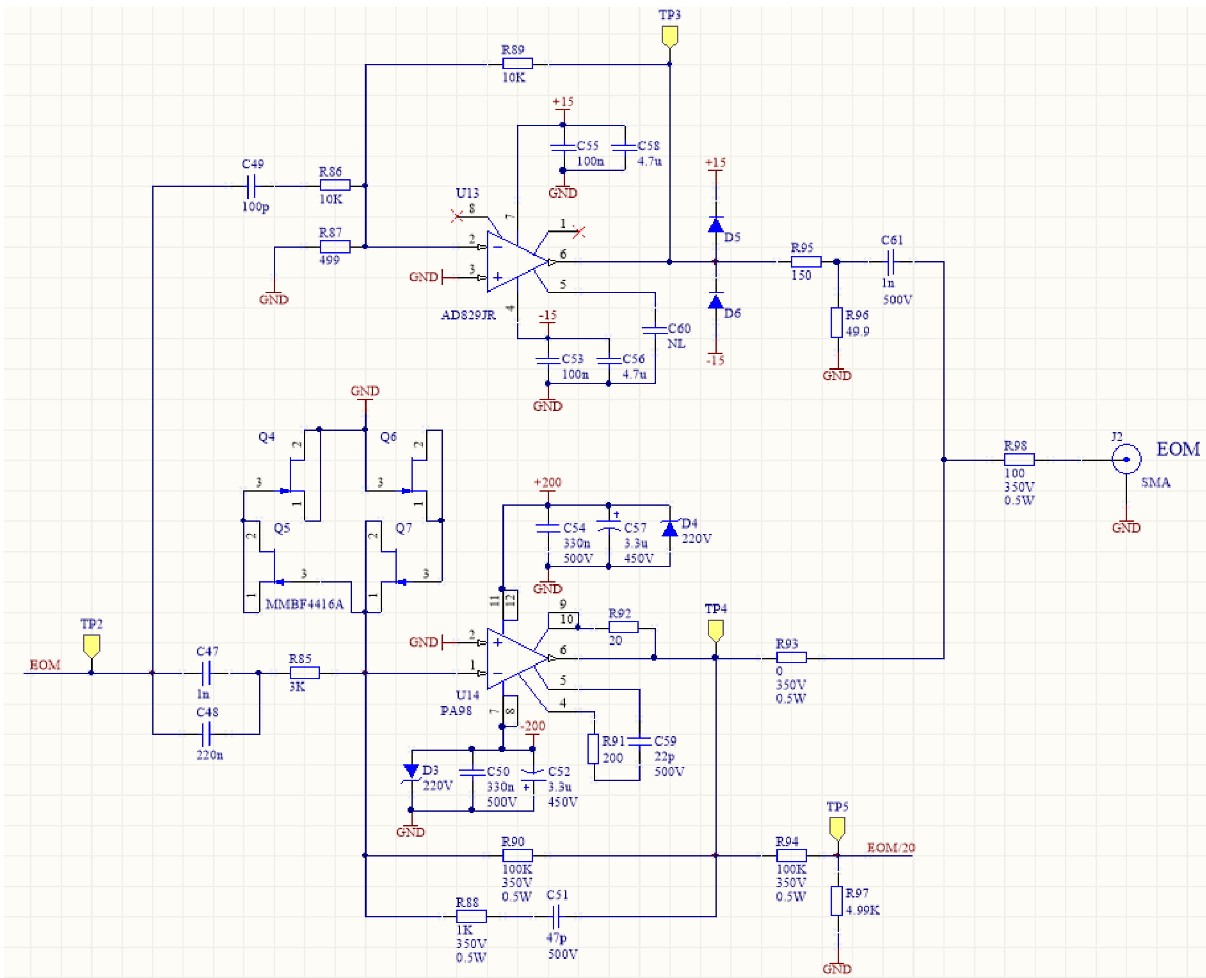
```
BodePlotEx[pa98[2 π i 1*^6 f] /. prmpa98,
 {f, 0.1, 20}, Evaluate[plotopt], XAxisLabel → "MHz"]
```





Old Double Path Configuration

Schematics



Transfer Function

In[58]:=

$$\text{prmFbEom} = \left\{\text{Zin} \to \text{R85} + \frac{1}{\text{s C48}},\ \text{Zfb} \to \text{par}\left[\text{R90},\ \text{R88} + \frac{1}{\text{s C51}}\right]\right\};$$

```
prmActEom2Path = {R90 → 100*^3, R88 → 1*^3, C51 → 47*^-12, R85 → 3*^3,
    C48 → 220*^-9, C49 → 100*^-12, R86 → 10*^3, R87 → 499, R89 → 10*^3,
    R93 → 0, R95 → 150, R96 → 50, C61 → 1*^-9, R98 → 100, Ceom → 10*^-12};
```

In[60]:=

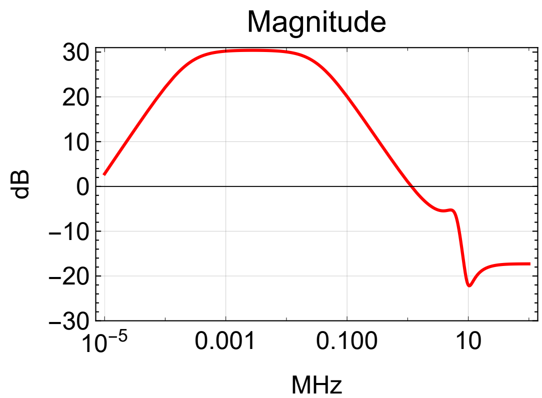
$$\text{u14[s\_]} := -\frac{\text{Zfb}}{\text{Zin}}\ /.\ \text{prmFbEom}$$

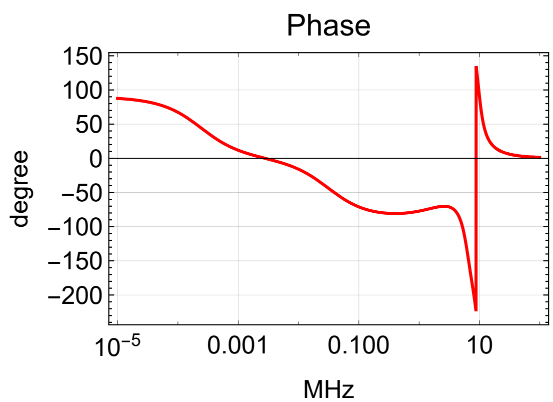
$$\text{u13[s\_]} := -\frac{\text{R89}}{\text{R86} + \frac{1}{\text{s C49}}}$$

Pole/zero Determination

Bode Plot

```
BodePlotEx[-eomact2Path[2 π ⅈ 1*^6 f] /. prmpa98 //. prmActEom2Path,
 {f, 0.00001, 100}, MagnitudeRange → {-30, 31}, Evaluate[plotopt], XAxisLabel → "MHz"]
```



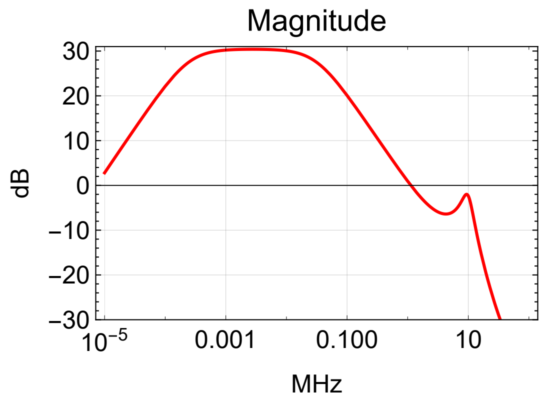


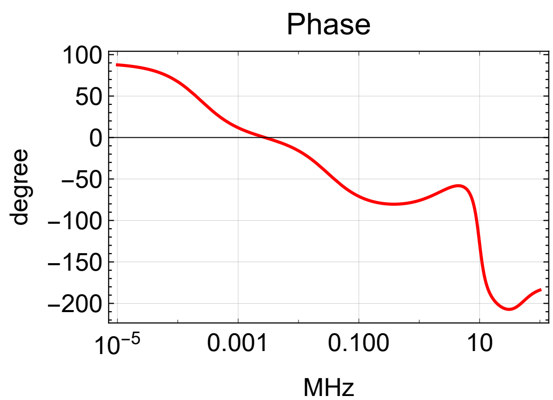
Single Path Configuration with Old Parameters

Remove C61 in AD829 path.

The AD829 path seems to reduce the gain peaking around 10 MHz, but otherwise has little effect below 1 MHz.

```
BodePlotEx[-eomact2Path[2 π ⅈ 1*^6 f] /. prmpa98 /. C61 → 0 //. prmActEom2Path,
 {f, 0.00001, 100}, MagnitudeRange → {-30, 31}, Evaluate[plotopt], XAxisLabel → "MHz"]
```





New Single Path Configuration (no PMC pole)

We add a passive low pass filter to the output and remove the U13 path all together.
C61 has changed to 560 pF and goes to ground with R96 → 0 and R95 → ∞.

Transfer Function

In[62]:=

```
prmActEom = {R90 → 100*^3, R88 → 1*^3, C51 → 47*^-12,
   R85 → 3*^3, C48 → 220*^-9, C49 → 0, R86 → 10*^3, R87 → 499, R89 → 10*^3,
   R93 → 100, R95 → ∞, R96 → 0, C61 → 560*^-12, R98 → 0, Ceom → 10*^-12};
paPole = {gPA → R90/R85, pPA1 → 1/(C48 R85), pPA2 → 1/(C51 (R90 + R88))} /. prmActEom;
eomPrm = Join[paPole, {coefEOM → 0.015 (* rad/V *)}];
eomActTF[s_] := gPA s/pPA1 pole[s, pPA1] × pole[s, pPA2]
eomCoeff[s_] := coefEOM s (* rad/s/V *)
{pPA1/(2. π), pPA2/(2. π)} /. eomPrm
```

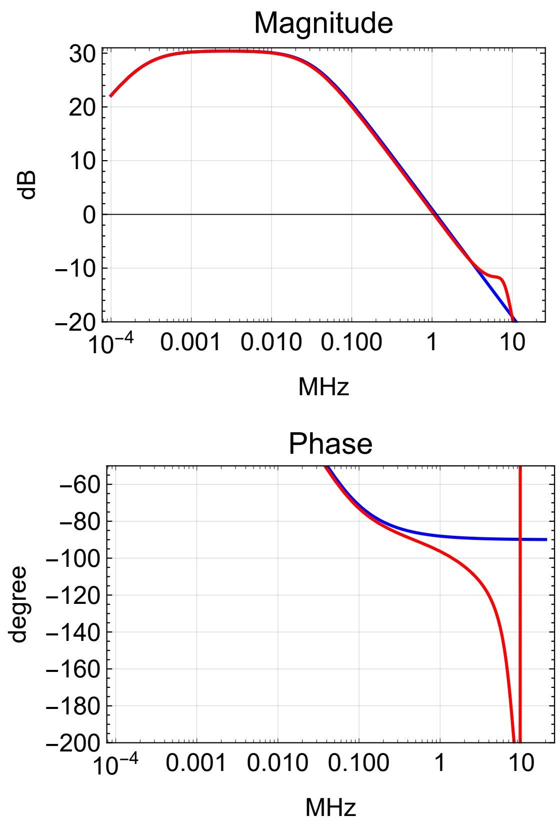
Out[67]=

```
{241.144, 33 527.5}
```

Pole/zero Determination

Bode Plot

```
BodePlotEx[{eomActTF[s] /. eomPrm /. s → 2 π ⅈ 1*^6 f,
  -eomact[s] /. prmpa98 /. prmActEom /. s → 2 π ⅈ 1*^6 f},
 {f, 0.0001, 20}, MagnitudeRange → {-20, 31}, PhaseRange → {-200, -50},
 Evaluate[plotoptn[2]], XAxisLabel → "MHz"]
```



New Single Path Configuration (with 600 kHz PMC pole)

We limit the gain roll-off above 600 kHz by increasing R88 to 5.62K. We also eliminate C61, since it is not needed.

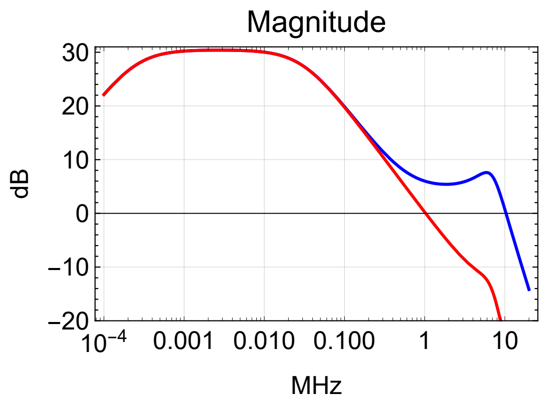
Transfer Function

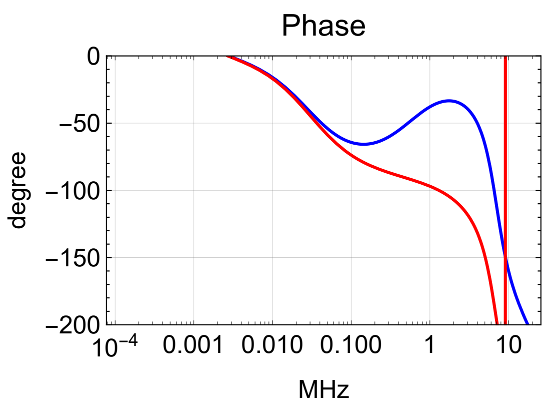
In[68]:=

```
prmActEomPMC = {R90 → 100*^3, R88 → 5.62*^3, C51 → 47*^-12,
    R85 → 3*^3, C48 → 220*^-9, C49 → 0, R86 → 10*^3, R87 → 499, R89 → 10*^3,
    R93 → 100, R95 → ∞, R96 → 0, C61 → 0*^-12, R98 → 0, Ceom → 10*^-12};
pmcPole = {pPMC → 2 π 600*^3};
```

Bode Plot

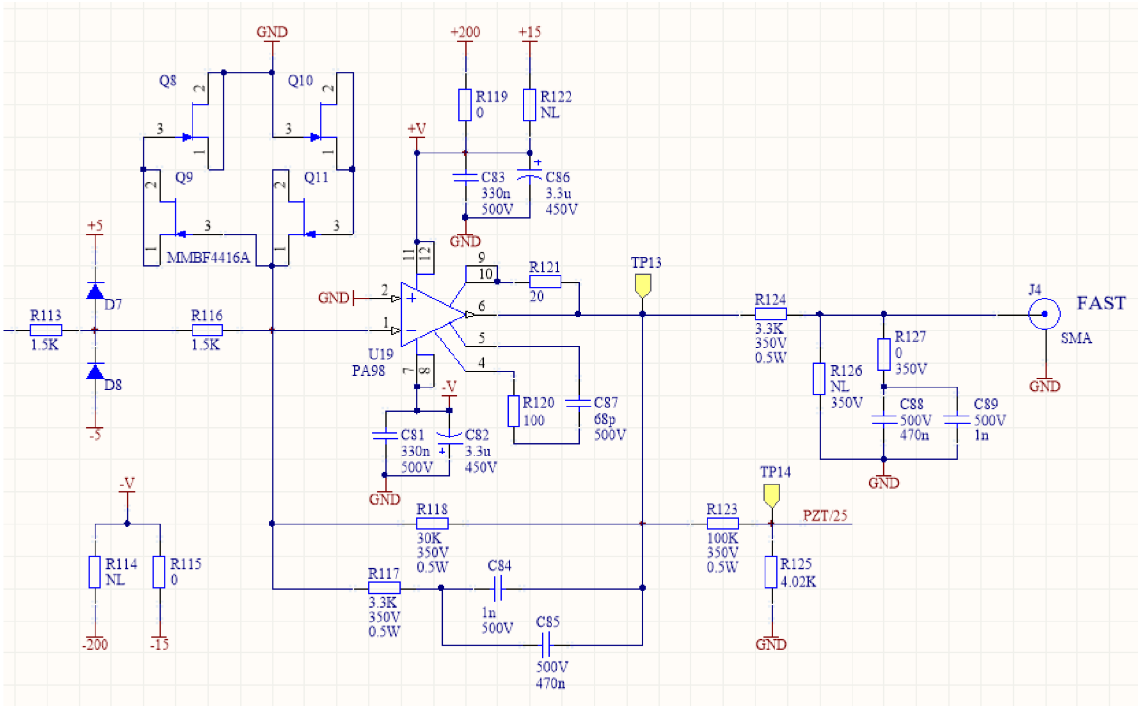
```
BodePlotEx[{eomTF[s] /. paPole /. s → 2 π ⅈ 1*^6 f,
  -eomact[s] /. prmpa98 /. prmActEomPMC /. s → 2 π ⅈ 1*^6 f,
  -eomact[s] × pole[s, pPMC] /. prmpa98 /. prmActEomPMC /. pmcPole /. s → 2 π ⅈ 1*^6 f},
 {f, 0.0001, 20}, MagnitudeRange → {-20, 31}, PhaseRange → {-200, 0},
 Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```





PZT Actuator Path

Schematics



Transfer Function

In[70]:=
```
prmFbPZT = {Zin → R113 + R116, Zfb → par[R118, R117 + 1/(s C85)]};
prmActPztPath = {R118 → 30*^3, R117 → 3.3*^3, C85 → 470*^-9, R113 → 1.5*^3,
    R116 → 1.5*^3, C88 → 470*^-9, R124 → 3.3*^3, R127 → 0, Cpzt → 40*^-12};
pztPole = {gPZT → R118/(R113 + R116), pPZT1 → 1/(C85 (R117 + R118))} /. prmActPztPath;
pztPrm = Join[pztPole, {coefPZT → 2 π 1*^6 (* rad/s/V *), bwPZT → 2 π 100*^3}];
pztTF[s_] := 1/2 gPZT pole[s, pPZT1] (*1/2 due ot gain in prev stage *)
pztCoeff[s_] := coefPZT pole[s, bwPZT] (* rad/s/V *)
{pPZT1/(2. π)} /. pztPrm
```

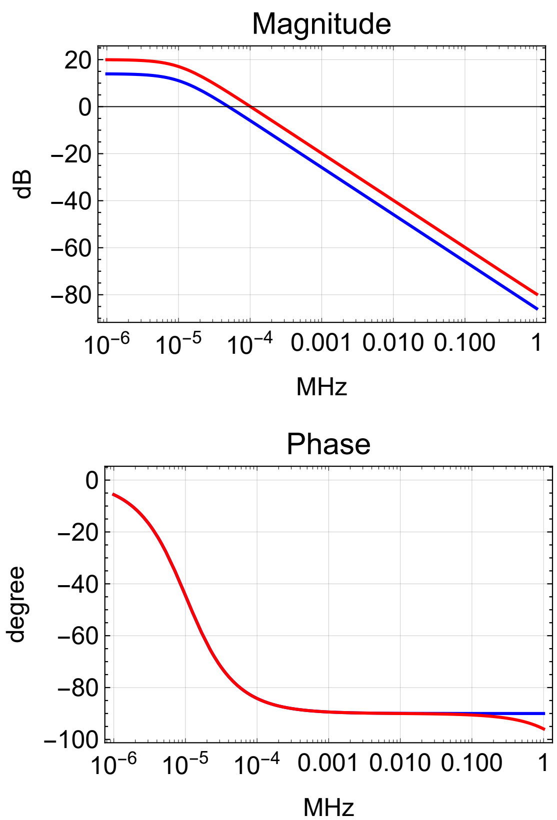
Out[76]=
```
{10.169}
```

In[77]:=
```
u19[s_] := -Zfb/Zin /. prmFbPZT
```

Pole/Zero Determination

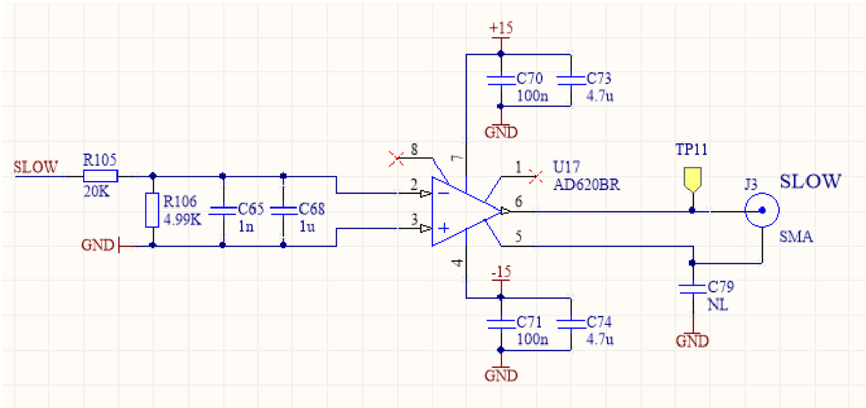
Bode Plot

```
BodePlotEx[
 {pztTF[2 π ⅈ 1*^6 f] /. pztPole, -pztactPath[2 π ⅈ 1*^6 f]/2 /. prmpa98 /. prmActPztPath},
 {f, 0.000001, 1}, Evaluate[plotoptn[2]], XAxisLabel → "MHz"]
```



Slow Actuator Path

Schematics



Transfer Function

In[78]:=

```
prmActSlowPath = {R105 → 20*^3, R106 → 4.99*^3, C68 → 1*^-6};
slowPole = {gSlow → R106/(R105 + R106), pSlow → 1/(C68 par[R105, R106])} /. prmActSlowPath;
slowTF[s_] := gSlow pole[s, pSlow]
slowCoeff[s_] := 2 π 3*^9 pole[s, 2 π 0.5] (* rad/s/V *)
{gSlow, pSlow/(2. π)} /. slowPole
```

Out[82]=

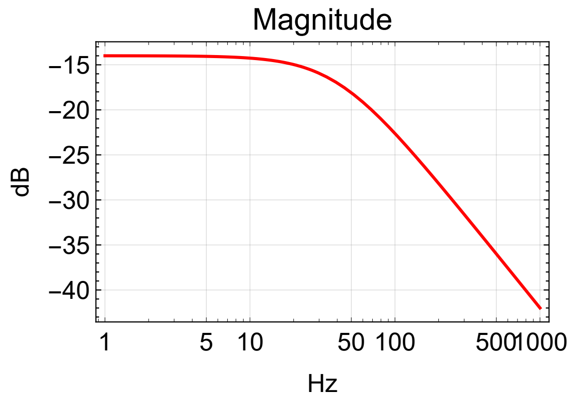
{0.19968, 39.8525}

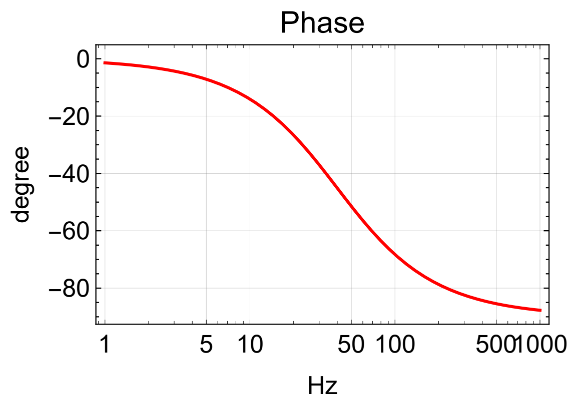
$$\frac{\text{par}\left[\text{R106}, \frac{1}{\text{s C65}}\right]}{\text{par}\left[\text{R106}, \frac{1}{\text{s C65}}\right] + \text{R105}} \text{ // Together}$$

$$\frac{\text{R106}}{\text{R105} + \text{R106} + \text{C65 R105 R106 s}}$$

Bode Plot

```
BodePlotEx[{slowTF[2 π ⅈ f] /. slowPole},
 {f, 1, 1000}, Evaluate[plotoptn[1]], XAxisLabel → "Hz"]
```





Sensing Path

Phase-Frequency Discriminator (Laser Locking)

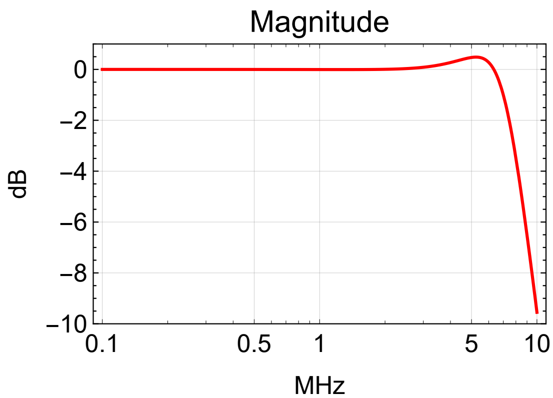
A phase-frequency discriminator is used for laser locking. The standard LIGO PFD circuit is described in LIGO-E1200114. The PCB LIGO-D1002471 is used with a modification to make it higher bandwidth. This is described in LIGO-E1700100.

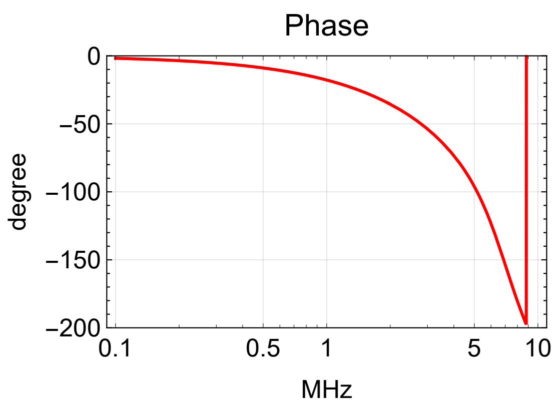
Transfer Function

In[83]:=
```
prmPFD =
  {sPFD1 → 2 π 5.72*^6, sPFD2 → 2 π 7.08*^6, qPFD2 → 1.44, sPFD3 → 169*^6, gPFD → 10/(2 π)};
pfd[s_] := pole[s, sPFD1] × pole[s, sPFD2, qPFD2] × pole[s, sPFD3]
pfdCoeff[s_] := gPFD 1/s (* V/(rad/s) *)
```

Bode Plot

```
BodePlotEx[pfd[2 π ⅈ f 1*^6] /. prmPFD, {f, 0.1, 10}, MagnitudeRange → {-10, 1},
 PhaseRange → {-200, 0}, Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```





Mixer (Cavity Locking)

The FET IQ demodulator is used for locking to a reference cavity. The standard LIGO FET IQ demodulator circuit is described in LIGO-E1200113. The PCB LIGO-D0902745 is used with a modification to make it ultra-fast bandwidth. This is described in LIGO-E1100044.

Transfer Function

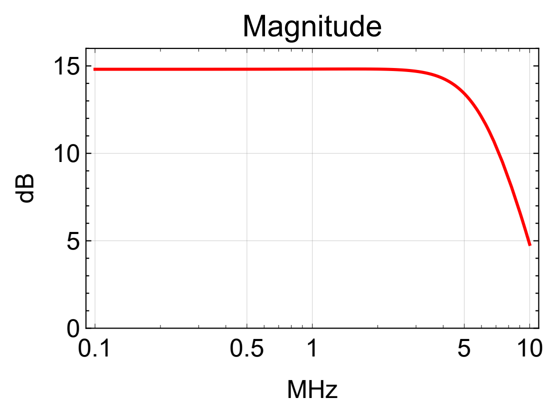
In[86]:=
```
prmDemod = {gDemod → 5.5, pDemod1 → 2 π 15.9*^6, pDemod2 → 2 π 6.17*^6, qDemod2 → 0.761};
demod[s_] := gDemod pole[s, pDemod1] × pole[s, pDemod2, qDemod2]
```

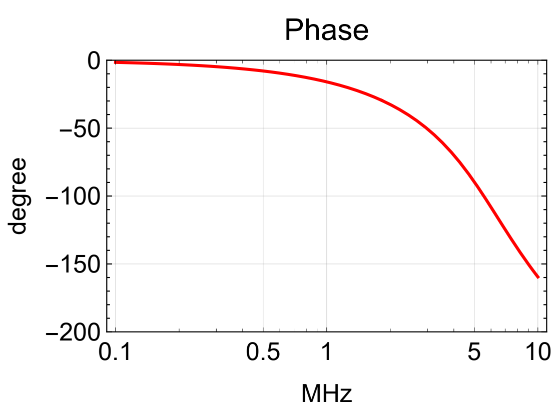
Reference Cavity

In[88]:=
```
prmPDH = {pRefCav → 2 π 77.5*^3, pwrRefCav → 10*^-3, gainRefCav → 1*^-6,
   gammaRefCav → 1.0, effPD → 0.8, transPD → 500, pPD → 2 π 2*^6}; (* estimates *)
pdh[s_] := 2 BesselJ[0, gammaRefCav] BesselJ[1, gammaRefCav]
  pwrRefCav gainRefCav pole[s, pRefCav] effPD transPD pole[s, pPD]
```

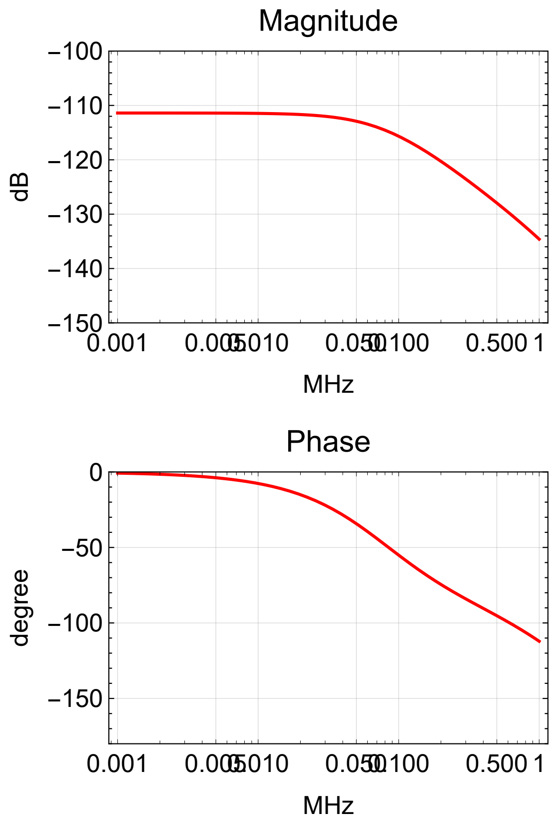
Bode Plot

```
BodePlotEx[demod[2 π ⅈ f 1*^6] /. prmDemod, {f, 0.1, 10}, MagnitudeRange → {0, 16},
 PhaseRange → {-200, 0}, Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```





```
BodePlotEx[pdh[2 π ⅈ f 1*^6] /. prmPDH, {f, 0.001, 1}, MagnitudeRange → {-150, -100},
 PhaseRange → {-180, 0}, Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```



Combined Sensing Path

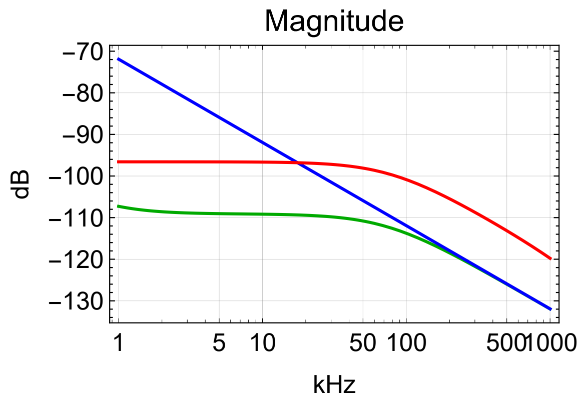
The sensing path transfer function combines one of the sensors, PFD or mixer, with an optional anti-boost. The anti-boost will make the PFD transfer function look more like the mixer one.

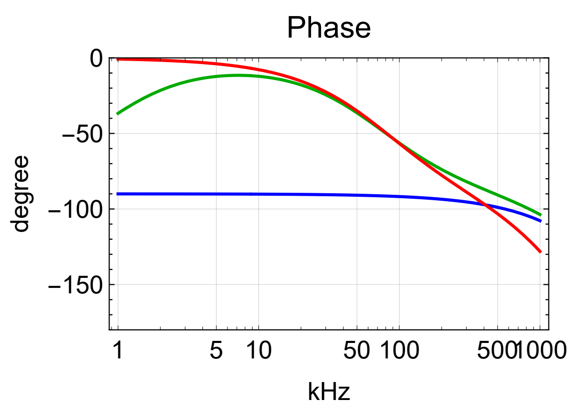
Transfer Function

```
In[90]:= prmSensing = {sBoostGain → 100, sBoostPole → 1/(Rs Cs)} /. {Rs → 1*^3, Cs → 2.2*^-9};
allSensing := Join[prmPFD, prmDemod, prmPDH, prmSensing]
sensingTF::unknownsensing = "Unknown sensing parameter `1`; must be PFD or Mixer.";
Options[sensingTF] = {Sensing → "PFD", sBoost → False};
sensingTF[s_, opts___] := Switch[Sensing /. {opts} /. Options[sensingTF],
    "PFD", pfdCoeff[s] × pfd[s],
    "Mixer", pdh[s] × demod[s],
    _, Message[sensingTF::unknownsensing, Sensing]; 0] *
   If[sBoost /. {opts} /. Options[sensingTF],
    1 / sBoostGain zero[s, sBoostPole / sBoostGain] × pole[s, sBoostPole], 1]
```

Bode Plot

```
BodePlotEx[{sensingTF[2 π ⅈ f 1*^3, Sensing → "PFD", sBoost → True] /. allSensing,
   sensingTF[2 π ⅈ f 1*^3, Sensing → "PFD"] /. allSensing,
   sensingTF[2 π ⅈ f 1*^3, Sensing → "Mixer"] /. allSensing}, {f, 1, 1000},
  MagnitudeRange → All, PhaseRange → {-180, 0}, Evaluate[plotoptn[3]], XAxisLabel → "kHz"]
```





TTFFS Servo

Common Path

Transfer Function

In[95]:=
```
poleCommon = {R89 → 3.16*^3, R87 → 3.16*^3, C101 → 330*^-9,
   R88 → 3.16*^3, R90 → 3.16*^3, C107 → 3.500*^-6};
prmCommon = {cGain → 10^(6/20), gCom1 → R87/R89, zCom1 → 1/(C101 R87),
     gCom2 → R90/R88, zCom2 → 1/(C107 R90)} /. poleCommon;
allCommon := prmCommon;

Options[commonTF] = {cBoost1 → False, cBoost2 → False};
commonTF[s_, opts___] := cGain/2 If[cBoost1, gCom1 zCom1/s zero[s, zCom1], gCom1] ×
     If[cBoost2, gCom2 zCom2/s zero[s, zCom2], gCom2] /. {opts} /. Options[commonTF]
```

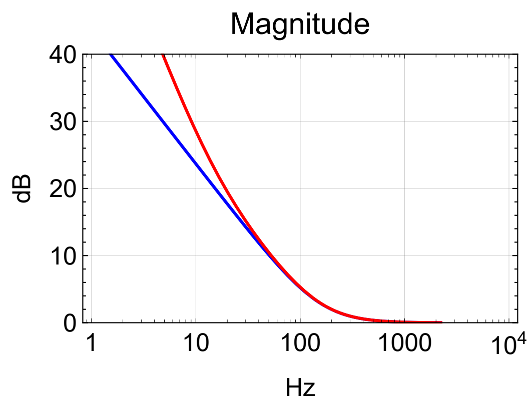
Parameters

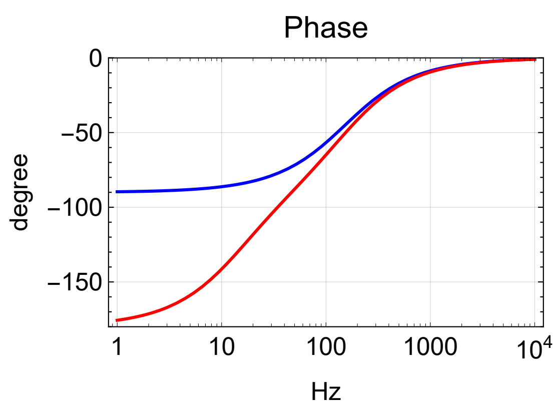
```
N[{gCom1, zCom1/(2 π)} /. allCommon]
N[{gCom2, zCom2/(2 π)} /. allCommon]
```

```
{1., 152.623}
{1., 14.3901}
```

Bode Plot

```
BodePlotEx[{commonTF[2 π ⅈ f , cBoost1 → True] /. prmCommon,
  commonTF[2 π ⅈ f , cBoost1 → True, cBoost2 → True] /. prmCommon},
 {f, 1, 10 000}, MagnitudeRange → {0, 40}, PhaseRange → {-180, 0},
 Evaluate[plotoptn[2]], XAxisLabel → "Hz"]
```





Fast Path Notches

Transfer Function

In[99]:=
```
poleFastNotch = {
   R96 → 499, R100 → 33, L1 → 470*^-6, C130 → 3.1*^-9,
   R97 → 499, R101 → 33, L2 → 470*^-6, C131 → 1.2*^-9,
   R98 → 499, R102 → 33, L3 → 220*^-6, C132 → 1.45*^-9,
   R99 → 249, R103 → 20, L4 → 470*^-6, C133 → 10.0*^-9};
NotchTF[s_, R1_, R2_, L_, C_] := R/(R1 + R) /. R → R2 + s L + 1/(s C)
NotchFreq[R1_, R2_, L_, C_] := 1/(2 π Sqrt[L C])
NotchZeroQ[R1_, R2_, L_, C_] := Sqrt[L]/(R2 Sqrt[C])
NotchPoleQ[R1_, R2_, L_, C_] := Sqrt[L]/((R1 + R2) Sqrt[C])
NotchParam[R1_, R2_, L_, C_] :=
 {NotchFreq[R1, R2, L, C], NotchZeroQ[R1, R2, L, C], NotchPoleQ[R1, R2, L, C]}

prmFastNotch = Flatten[{
     {fNotch1 → 2 π #[[1]], qzNotch1 → #[[2]], qpNotch1 → #[[3]]} &[
      NotchParam[R96, R100, L1, C130]],
     {fNotch2 → 2 π #[[1]], qzNotch2 → #[[2]], qpNotch2 → #[[3]]} &[
      NotchParam[R97, R101, L2, C131]],
     {fNotch3 → 2 π #[[1]], qzNotch3 → #[[2]], qpNotch3 → #[[3]]} &[
      NotchParam[R98, R102, L3, C132]],
     {fNotch4 → 2 π #[[1]], qzNotch4 → #[[2]], qpNotch4 → #[[3]]} &[
      NotchParam[R99, R103, L4, C133]]
    }] /. poleFastNotch;
Options[fastTFNotch] = {FastNotch → 4};
fastTFNotch[s_, opts___] :=
 If[FastNotch < 1 /. {opts} /. Options[fastTFNotch],
   1, zero[s, fNotch1, qzNotch1] × pole[s, fNotch1, qpNotch1]] *
  If[FastNotch < 2 /. {opts} /. Options[fastTFNotch],
   1, zero[s, fNotch2, qzNotch2] × pole[s, fNotch2, qpNotch2]] *
  If[FastNotch < 3 /. {opts} /. Options[fastTFNotch],
   1, zero[s, fNotch3, qzNotch3] × pole[s, fNotch3, qpNotch3]] *
  If[FastNotch < 4 /. {opts} /. Options[fastTFNotch],
   1, zero[s, fNotch4, qzNotch4] × pole[s, fNotch4, qpNotch4]]
```

Pole/zero Determination

Parameters

```
{fNotch1/(2 π), qzNotch1, qpNotch1} /. prmFastNotch
{fNotch2/(2 π), qzNotch2, qpNotch2} /. prmFastNotch
{fNotch3/(2 π), qzNotch3, qpNotch3} /. prmFastNotch
{fNotch4/(2 π), qzNotch4, qpNotch4} /. prmFastNotch
```

{131 853., 11.7992, 0.731908}

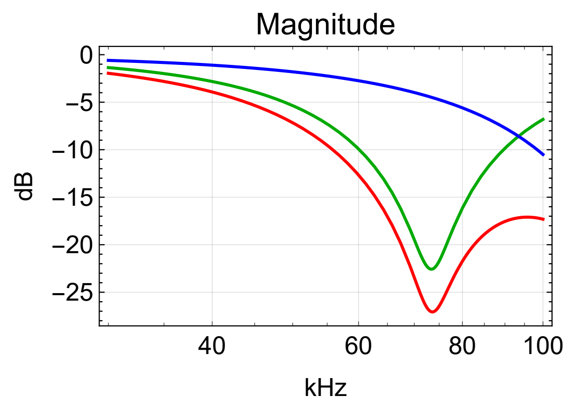
{211 924., 18.9646, 1.17638}

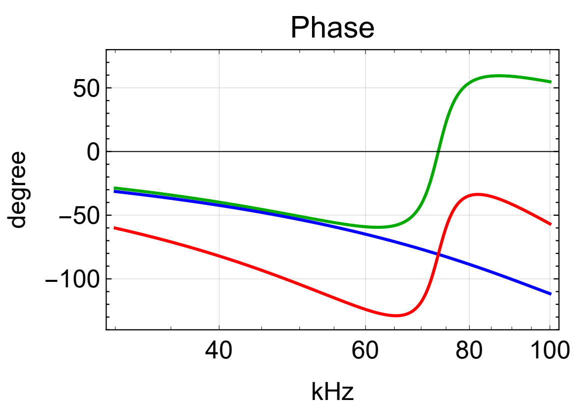
{281 789., 11.8036, 0.732176}

{73 412.7, 10.8397, 0.805929}

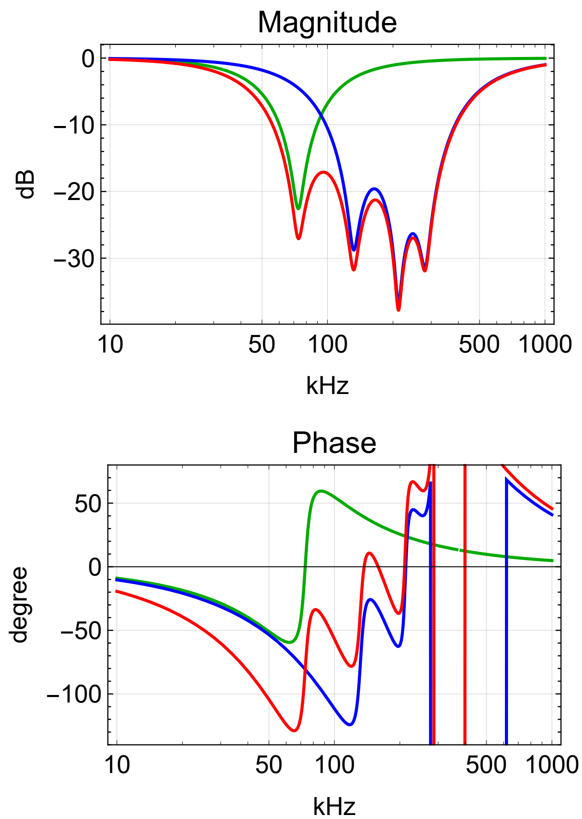
Bode Plot

```
BodePlotEx[{fastTFNotch[2 π ⅈ 1000 f, FastNotch → 4] / fastTFNotch[2 π ⅈ 1000 f, FastNotch → 3] /. prmFastNotch,
  fastTFNotch[2 π ⅈ 1000 f, FastNotch → 3] /. prmFastNotch,
  fastTFNotch[2 π ⅈ 1000 f, FastNotch → 4] /. prmFastNotch}, {f, 30, 100},
 MagnitudeRange → All, PhaseRange → {-140, 80}, Evaluate[plotoptn[3]], XAxisLabel → "kHz"]
```





```
BodePlotEx[{fastTFNotch[2 π ⅈ 1000 f, FastNotch → 4] / fastTFNotch[2 π ⅈ 1000 f, FastNotch → 3] /. prmFastNotch,
  fastTFNotch[2 π ⅈ 1000 f, FastNotch → 3] /. prmFastNotch,
  fastTFNotch[2 π ⅈ 1000 f, FastNotch → 4] /. prmFastNotch}, {f, 10, 1000},
 MagnitudeRange → All, PhaseRange → {-140, 80}, Evaluate[plotoptn[3]], XAxisLabel → "kHz"]
```



Fast Path

Transfer Function

In[108]:=

```
poleFast = {R35 → 499, R33 → 3.16*^3, C36 → 10*^-12, R30 → 28*^3, C43 → 2.2*^-9,
   R36 → 499, R31 → 3.16*^3, C37 → 390*^-12,
   R80 → 499, R81 → 1.58*^3, C77 → 330*^-9, R82 → 14.3*^3,
   R83 → 1000, (*R84→66.7*^3,C95→2.2*^-9,*) R85 → 1000, R86 → 100*^3, C79 → 330*^-9
  };
prmFast = {fGain → 10^(0/20),
    gFast1 → -R30/R35, pFast1 → 1/(C43 (R30 + R33)), zFast1 → 1/(C43 R33),
    gFast2 → -R31/R36, pFast2 → 1/(C37 R31),
    gFast3 → -R82/R80, pFast3 → 1/(C77 (R81 + R82)), zFast3 → 1/(C77 R81),
    gFast4 → -R86/R83, pFast4A → 1/(C79 (R85 + R86)), zFast4A → 1/(C79 R85),
    (*pFast4B→1/(C95 R83),zFast4B→1/(C95 (R83+R84)),*)
    gFast5 → -1

   } /. poleFast;
allFast := Join[pztPrm, prmFast, prmFastNotch]

Options[fastTF] = Join[{FastOnly → False, FastFilter → True}, Options[fastTFNotch]];
fastTF[s_, opts___] :=
 If[FastOnly /. {opts} /. Options[fastTF],
   If[FastFilter /. {opts} /. Options[fastTF],
    gFast3 pole[s, pFast3] × zero[s, zFast3] *
     gFast4 pole[s, pFast4A] × zero[s, zFast4A],
    (gFast3 pFast3)/zFast3 (gFast4 pFast4A)/zFast4A],
   fGain/2 gFast1 pole[s, pFast1] × zero[s, zFast1] gFast2 pole[s, pFast2]] *
  gFast5 *
  fastTFNotch[s, opts]
```

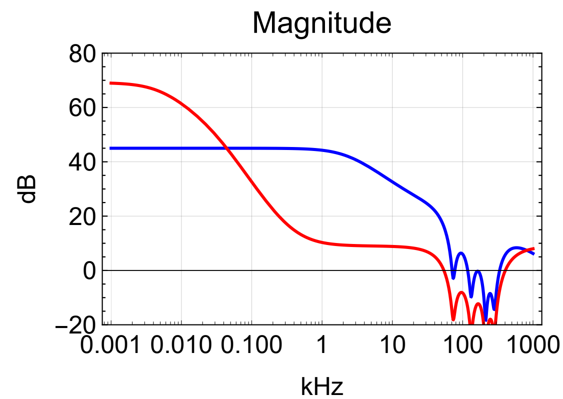
Parameters

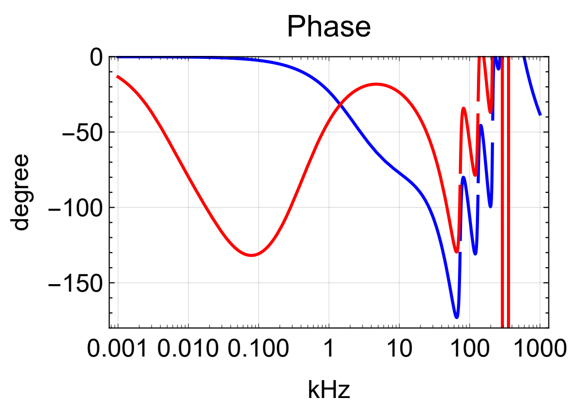
```
N[{gFast1, pFast1/(2 π), zFast1/(2 π)} /. allFast]
N[{gFast2, pFast2/(2 π)} /. allFast]
N[{gFast3, pFast3/(2 π), zFast3/(2 π)} /. allFast]
N[{gFast4, pFast4A/(2 π), zFast4A/(2 π) (*, pFast4B/(2 π), zFast4B/(2π) *)} /. allFast]
{-56.1122, 2321.67, 22893.4}
{-6.33267, 129142.}
{-28.6573, 30.3708, 305.245}
{-100., 4.77513, 482.288}
```

Pole/zero Determination

Bode Plot

```
BodePlotEx[
 {-fastTF[2 π ⅈ f 1*^3] //. allFast, -fastTF[2 π ⅈ f 1*^3, FastOnly → True] //. allFast},
 {f, 0.001, 1000}, MagnitudeRange → {-20, 80},
 PhaseRange → {-180, 0}, Evaluate[plotoptn[2]], XAxisLabel → "kHz"]
```





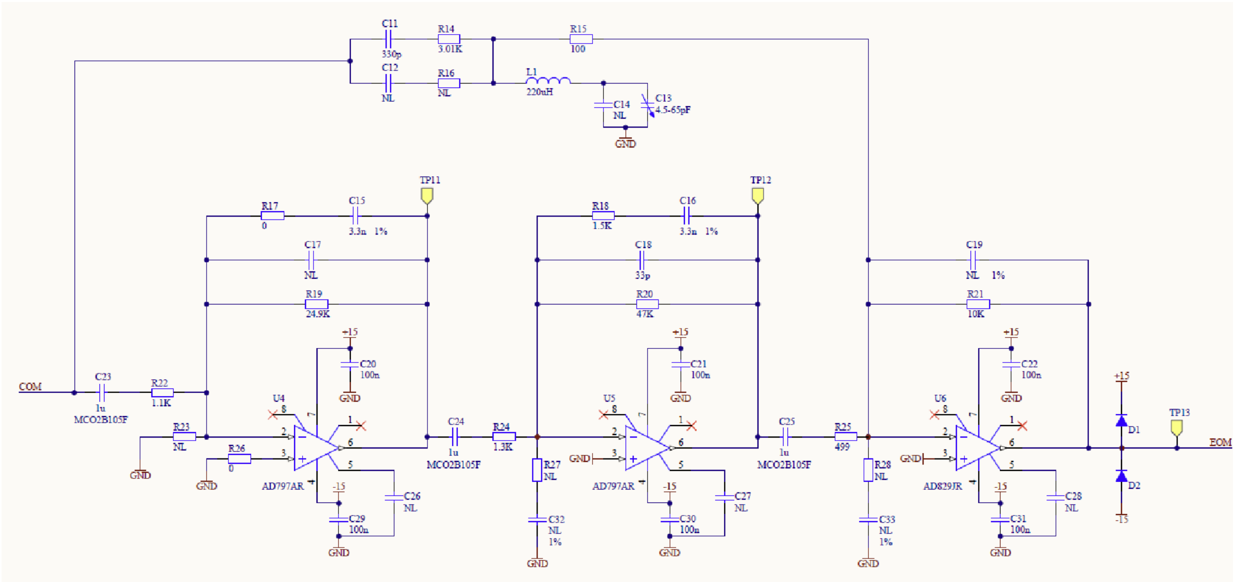
TF Data

```
{dB[#/(4 × 25)], Phase[#]} &@ (-fastTF[2 π ⅈ f ] × pztTF[2 π ⅈ f]) /. pztPole //. allFast /.
 f → 100 000.
```

```
{-99.8956, -196.119}
```

EOM Path

Schematics



Extra Lead

In[113]:=

```
poleEomLead = {R14 → 3.16*^3, C11 → 1*^-9, R16 → 2*^3, C12 → 100.*^-12,
   L1 → 220*^-6, C13 → 30*^-12, R15 → 100, R21 → 10*^3, C19 → 0.1*^-12};
g1[s_] := Simplify[R21 / (1/(s C11) + R14 + R15)]
g2[s_] := FullSimplify[Together[R21 / (par[1/(s C11) + R14, 1/(s C12) + R16] + R15)]]
prmEomLead = Join[
    {gEOM4 → -Limit[g2[s]/s, s → 0]},
    MapThread[ReplaceAll,
     {{zEOM4a → -s, zEOM4aa → -s}, Solve[Numerator[g2[s]] == 0, s]}],
    MapThread[ReplaceAll,
     {{pEOM4a → -s, pEOM4aa → -s}, Solve[Denominator[g2[s]] == 0, s]}],
    {zEOM4b → 1/Sqrt[C13 L1], pEOM4b → 1/Sqrt[C13 L1],
     qEOM4b → Sqrt[C13 L1]/(C13 par[R14, R15]), pEOM4c → 1/(C19 R21)}] /. poleEomLead;
Options[eomTF] = {};
eom2TF[s_, opts___] := gEOM4 s pole[s, pEOM4a] × zero[s, zEOM4aa] × pole[s, pEOM4aa] × zero[
     s, zEOM4b, ∞] × pole[s, pEOM4b, qEOM4b] × pole[s, pEOM4c] /. {opts} /. Options[eomTF]
```

Parameters

$$\left\{-\frac{\text{R21}}{\text{R15}+\text{par}[\text{R14},\text{R16}]},\ \frac{\text{pEOM4aa}}{2.\ \pi},\ \frac{\text{zEOM4aa}}{2.\ \pi},\ \frac{\text{pEOM4a}}{2.\ \pi}\right\}\ \text{/. prmEomLead /. poleEomLead}$$

$$\left\{\frac{\text{pEOM4b}}{2.\ \pi},\ \text{qEOM4b},\ \frac{\text{pEOM4c}}{2.\ \pi}\right\}\ \text{/. prmEomLead /. poleEomLead}$$

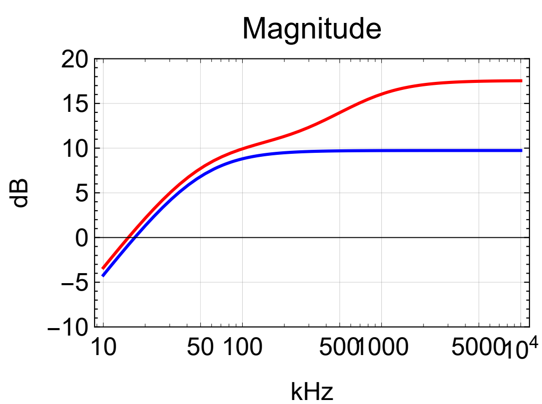
```
{-7.54827, 48815.6, 339284., 759066.}
```

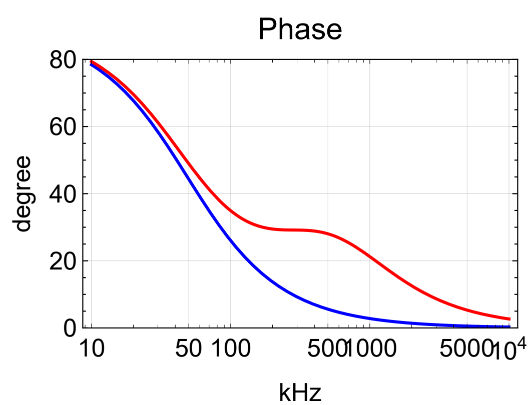
$\{1.95906\times10^{6},\ 27.9371,\ 1.59155\times10^{8}\}$

```
{dB[#], Phase[#]} &@
 ((eomActTF[s] /. eomPrm /. s → 2 π ⅈ 1*^6 f) (eomTF[2 π ⅈ f 1*^6] //. allEom) /. f → .0001)
```

```
{32.565, -222.885}
```

```
BodePlotEx[{g1[2 π ⅈ f 1*^3] //. poleEomLead, g2[2 π ⅈ f 1*^3] //. poleEomLead},
 {f, 10, 10000}, MagnitudeRange → {-10, 20},
 PhaseRange → {0, 80}, Evaluate[plotoptn[2]], XAxisLabel → "kHz"]
```





Transfer Function

In[117]:=

```
poleEom = Join[poleEomLead,
   {R22 → 1*^3, C23 → 100*^-9, R17 → 0.1, C15 → 3.3*^-9,
    R19 → 28*^3, C17 → 0.5*^-12, R24 → 1.58*^3, C24 → 100*^-9, R18 → 1.58*^3,
    C16 → 3.3*^-9, R20 → 48.7*^3, C18 → 47*^-12, R25 → 499., C25 → 100*^-9}];
prmEom = {gEOM1 → -R19/R22, pEOM1a → 1/(C23 R22),
    pEOM1b → 1/(C15 (R17 + R19)), zEOM1b → 1/(C15 R17), pEOM1c → 1/(par[R17, R19] C17),
    gEOM2 → -R20/R24, pEOM2a → 1/(C24 R24),
    pEOM2b → 1/(C16 (R18 + R20)), zEOM2b → 1/(C16 R18), pEOM2c → 1/(par[R18, R20] C18),
    gEOM3 → -R21/R25, pEOM3a → 1/(C25 R25), pEOM3b → 1/(C19 R21)} /. poleEom;
allEom := Join[eomPrm, prmEom, prmEomLead]

Options[eomTF] = {};
eom1TF[s_, opts___] :=
 gEOM1 s/pEOM1a pole[s, pEOM1a] × pole[s, pEOM1b] × zero[s, zEOM1b] × pole[s, pEOM1c] *
    gEOM2 s/pEOM2a pole[s, pEOM2a] × pole[s, pEOM2b] × zero[s, zEOM2b] × pole[s, pEOM2c] *
    gEOM3 s/pEOM3a pole[s, pEOM3a] × pole[s, pEOM3b] /. {opts} /. Options[eomTF]
eomTF[s_, opts___] := eom1TF[s, opts] + eom2TF[s, opts]
```

Parameters

```
{gEOM1, pEOM1a/(2. π), pEOM1b/(2 π), zEOM1b/(2 π), pEOM1c/(2 π)} /. prmEom /. poleEom
{gEOM2, pEOM2a/(2 π), pEOM2b/(2 π), zEOM2b/(2 π), pEOM2c/(2 π)} /. prmEom /. poleEom
{gEOM3, pEOM3a/(2. π), pEOM3b/(2 π)} /. prmEom /. poleEom
```

$\{-28, 1591.55, 1722.45, 4.82288\times10^{8}, 3.18311\times10^{12}\}$

$\{-30.8228, 1007.31, 959.204, 30524.5, 2.21275\times10^{6}\}$

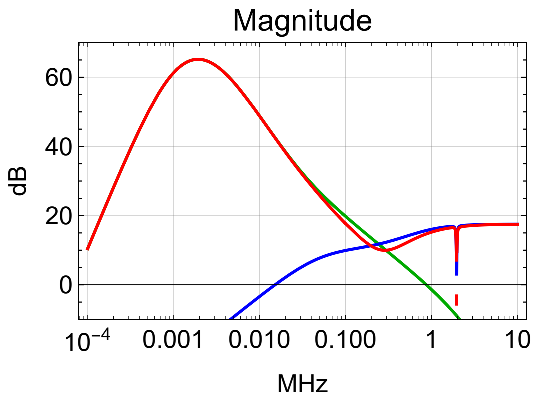
$\{-20.0401, 3189.48, 1.59155\times10^{8}\}$

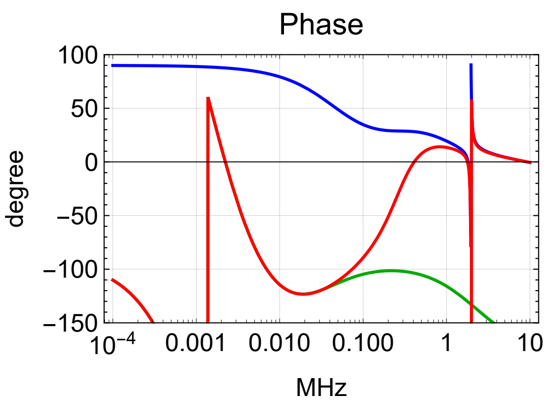
```
{dB[#], Phase[#]} &@
 ((eomActTF[s] /. eomPrm /. s → 2 π ⅈ 1*^6 f) (eomTF[2 π ⅈ f 1*^6] //. allEom) /. f → .0001)
```

```
{32.565, -222.885}
```

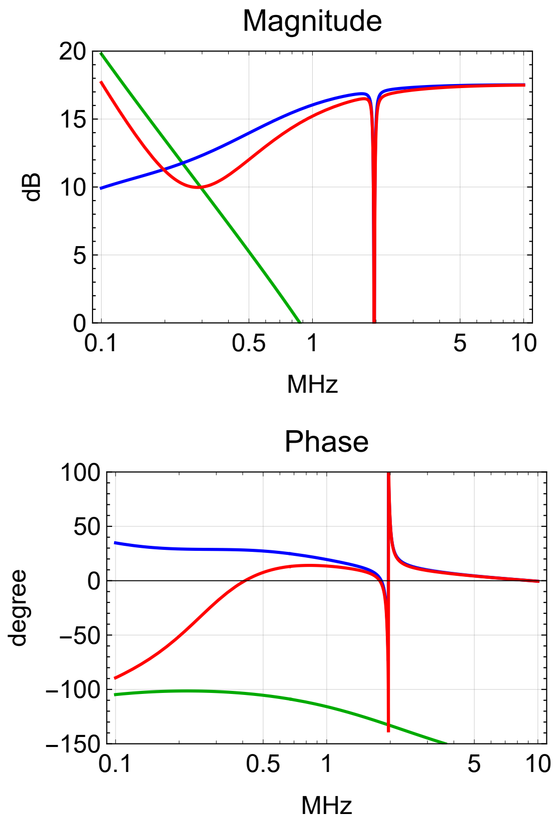
Bode Plot

```
BodePlotEx[{-eom1TF[2 π ⅈ f 1*^6], -eom2TF[2 π ⅈ f 1*^6], -eomTF[2 π ⅈ f 1*^6]} //. allEom,
 {f, 0.0001, 10}, MagnitudeRange → {-10, 70},
 PhaseRange → {-150, 100}, Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```

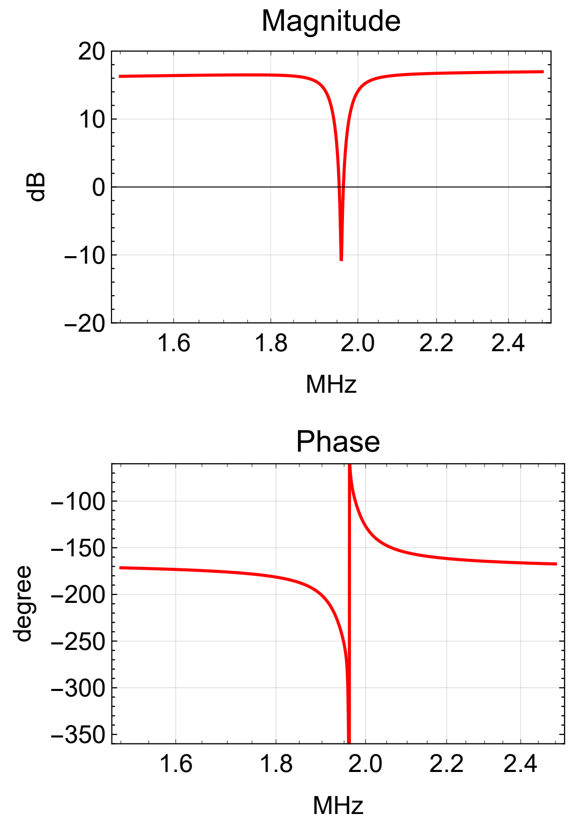




```
BodePlotEx[{-eom1TF[2 π ⅈ f 1*^6], -eom2TF[2 π ⅈ f 1*^6], -eomTF[2 π ⅈ f 1*^6]} //. allEom,
 {f, 0.1, 10}, MagnitudeRange → {0, 20}, PhaseRange → {-150, 100},
 Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```



```
BodePlotEx[eomTF[2 π ⅈ f 1*^6] /. allEom, {f, 1.5, 2.5}, MagnitudeRange → {-20, 20},
 PhaseRange → {-360, -60}, Evaluate[plotoptn[1]], XAxisLabel → "MHz"]
```



Overall Transfer Functions

In[123]:=

```
ttfssCom[s_, opts___] := -sensingTF[s, opts] × commonTF[s, opts]
ttfssFastSplit[s_, opts___] := fastTF[s, opts] × pztTF[s] × pztCoeff[s]
ttfssEomSplit[s_, opts___] := eomTF[s, opts] × eomActTF[s] × eomCoeff[s]
ttfssFast[s_, opts___] := ttfssCom[s, opts] × ttfssFastSplit[s, opts]
ttfssEom[s_, opts___] := ttfssCom[s, opts] × ttfssEomSplit[s, opts]
ttfssCrossTF[s_, opts___] := ttfssFastSplit[s, opts] / ttfssEomSplit[s, opts]
```

In[129]:=

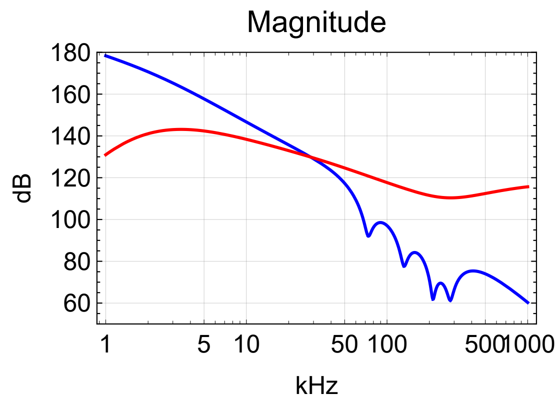
```
allTTFSS := Join[allSensing, prmCommon, allFast, allEom]
Options[ttfssTF] = Join[Options[sensingTF],
   Options[commonTF], Options[fastTF], Options[eomTF]];
ttfssTF[s_, opts___] :=
 ttfssFast[s, opts] + If[FastOnly /. {opts} /. allTTFSS, 0, ttfssEom[s, opts]]
```

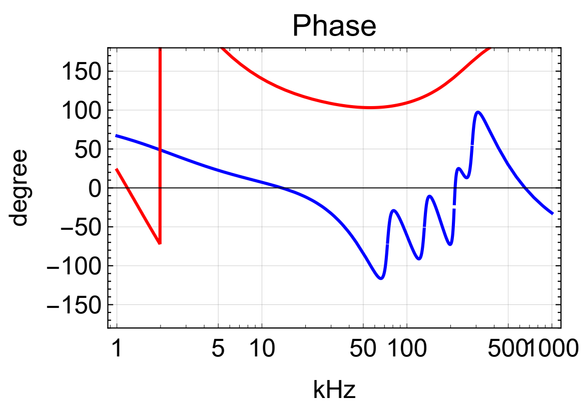
```
Options[ttfssTF]
```

```
{Sensing → PFD, sBoost → False, cBoost1 → False,
 cBoost2 → False, FastOnly → False, FastFilter → True, FastNotch → 4}
```

Split

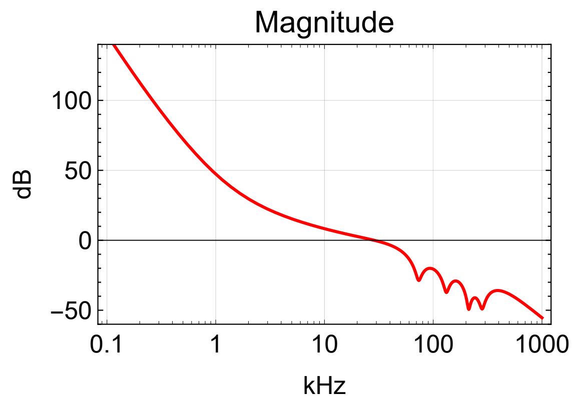
```
BodePlotEx[{ttfssFastSplit[2 π ⅈ f 1*^3, FastOnly → False] /. fGain → 10^(24/20) /. allTTFSS,
  ttfssEomSplit[2 π ⅈ f 1*^3] /. allTTFSS}, {f, 1, 1000}, MagnitudeRange → {50, 180},
 PhaseRange → {-180, +180}, Evaluate[plotoptn[2]], XAxisLabel → "kHz"]
```

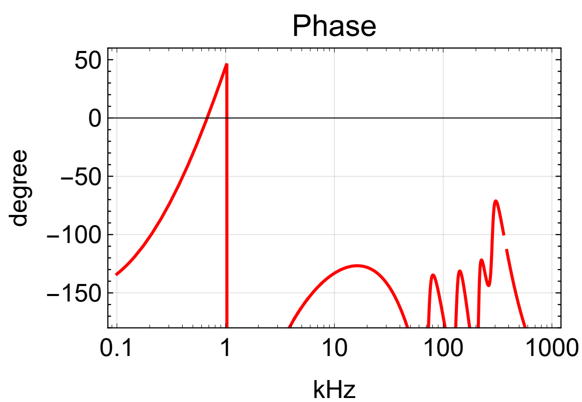




Crossover

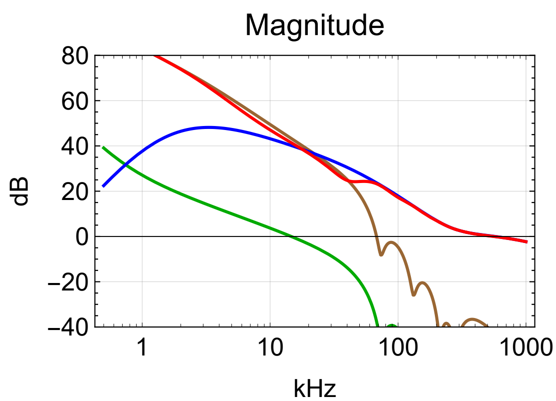
```
BodePlotEx[ttfssCrossTF[2 π ⅈ f 1*^3, FastOnly → False] /. fGain → 10^(24/20) /. allTTFSS,
 {f, 0.1, 1000}, MagnitudeRange → {-60, 140},
 PhaseRange → {-180, +60}, Evaluate[plotoptn[1]], XAxisLabel → "kHz"]
```

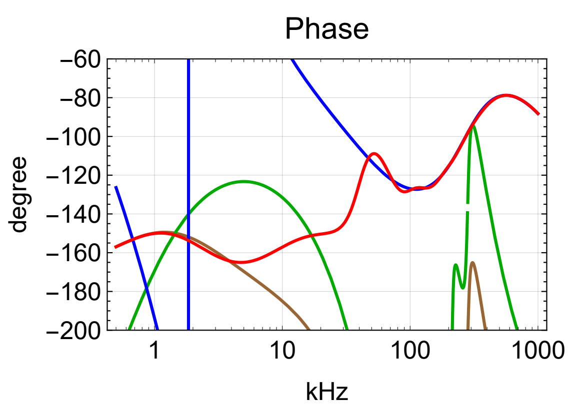




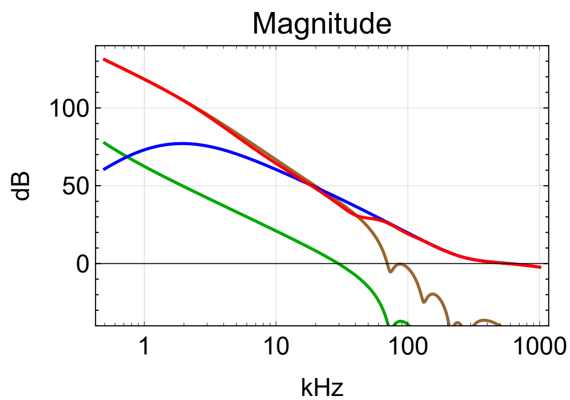
Components

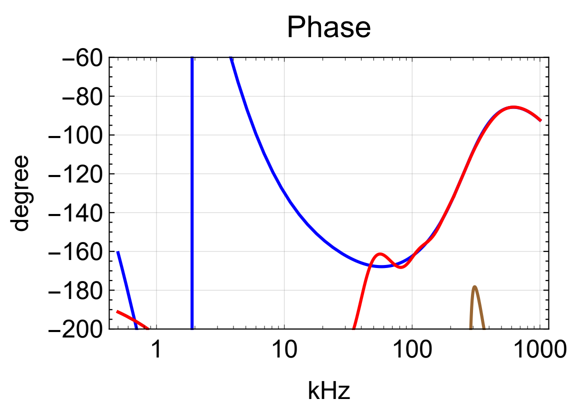
```
opt = Sequence[sBoost → True, FastNotch → 4];
prm = {cGain → 10^(20/20), fGain → 10^(22/20)};
BodePlotEx[{ttfssFast[2 π ⅈ f 1*^3, FastOnly → False, opt] /. prm /. allTTFSS,
  ttfssFast[2 π ⅈ f 1*^3, FastOnly → True, opt] /. prm /. allTTFSS,
  ttfssEom[2 π ⅈ f 1*^3, FastOnly → False, opt] /. prm /. allTTFSS,
  ttfssTF[2 π ⅈ f 1*^3, FastOnly → False, opt] /. prm /. allTTFSS},
 {f, 0.5, 1000}, MagnitudeRange → {-40, 80}, PhaseRange → {-200, -60},
 Evaluate[plotoptn[4]], XAxisLabel → "kHz"]
```





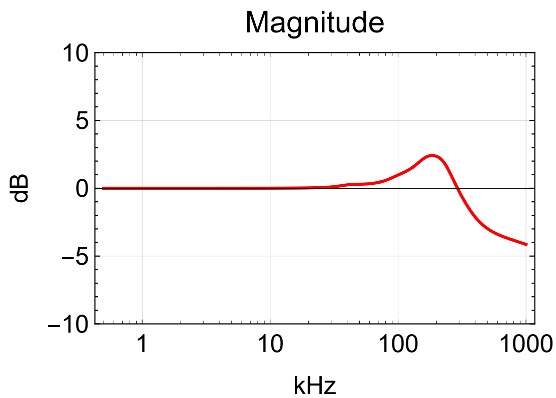
```
opt = Sequence[sBoost → False, FastNotch → 4];
prm = {cGain → 10^(20/20), fGain → 10^(22/20)};
BodePlotEx[{ttfssFast[2 π ⅈ f 1*^3, FastOnly → False, opt] /. prm /. allTTFSS,
  ttfssFast[2 π ⅈ f 1*^3, FastOnly → True, opt] /. prm /. allTTFSS,
  ttfssEom[2 π ⅈ f 1*^3, FastOnly → False, opt] /. prm /. allTTFSS,
  ttfssTF[2 π ⅈ f 1*^3, FastOnly → False, opt] /. prm /. allTTFSS},
 {f, 0.5, 1000}, MagnitudeRange → {-40, 140}, PhaseRange → {-200, -60},
 Evaluate[plotoptn[4]], XAxisLabel → "kHz"]
```

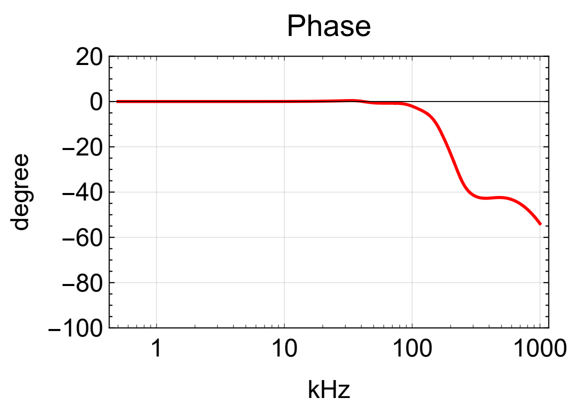




Closed Loop

```
opt = Sequence[sBoost → False, FastNotch → 4];
prm = {cGain → 10^(20/20), fGain → 10^(22/20)};
BodePlotEx[{ttfssTF[2 π ⅈ f 1*^3, FastOnly → False, opt] / (1 + ttfssTF[2 π ⅈ f 1*^3, FastOnly → False, opt]) /. prm /. allTTFSS},
 {f, 0.5, 1000}, MagnitudeRange → {-10, 10},
 PhaseRange → {-100, 20}, Evaluate[plotoptn[4]], XAxisLabel → "kHz"]
```

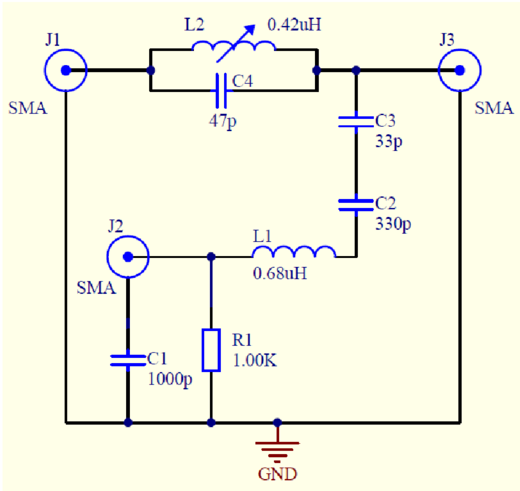




Summing Node

Schematics

This is D040469-B.
J1: Low frequency input
J2: RF modulation input, nominal 35.5 MHz
J3: Output to Pockels cell



Parameters

In[131]:=

```
prmSN = {L2 → 0.439*^-6, L2R → 0.019, C4 → 47*^-12, C3 → 33*^-12, C2 → 330*^-12,
    L1 → 0.68*^-6, L1R → 0.55, R1 → 1000, C1 → 0*^-12, Ceom → 20*^-12, Rterm → 50};
```

$$\frac{1}{2\pi\sqrt{\text{L2 C4}}} \text{ /. prmSN}$$

$$\frac{1}{2\pi\sqrt{\text{L1 par[C3, C2]}}} \text{ /. prmSN}$$

3.5038×10^{7}

3.52375×10^{7}

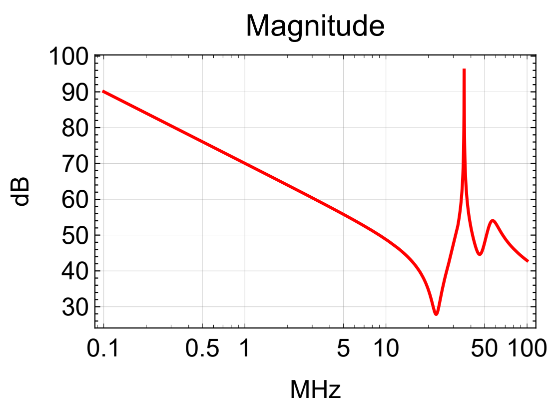
Input Impedance

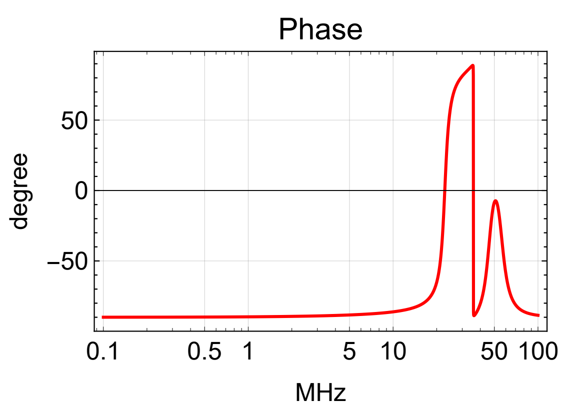
Low Frequency Input

```
impSN1 = par[s L2 + L2R, 1/(s C4)] + par[1/(s C3) + 1/(s C2) + s L1 + L1R + par[R1, Rterm], 1/(s Ceom)];
Limit[impSN1 s, s → 0]
N[1/% /. prmSN] (* effective capacitance seen *)
BodePlotEx[impSN1 /. s → 2 π ⅈ 1*^6 f //. prmSN, {f, 0.1, 100},
 MagnitudeRange → All, PhaseRange → All, XAxisLabel → "MHz", plotoptn[1]]
```

$$\frac{C2 + C3}{C3\ Ceom + C2\ (C3 + Ceom)}$$

$5. \times 10^{-11}$



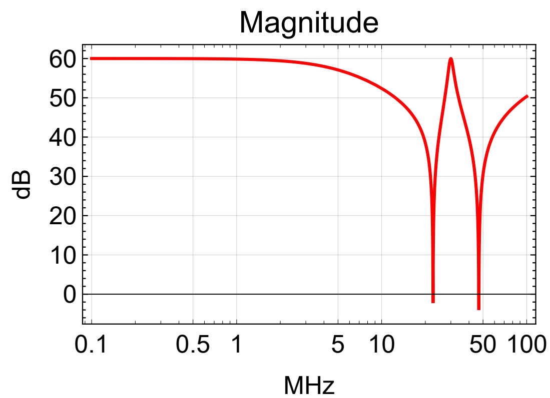


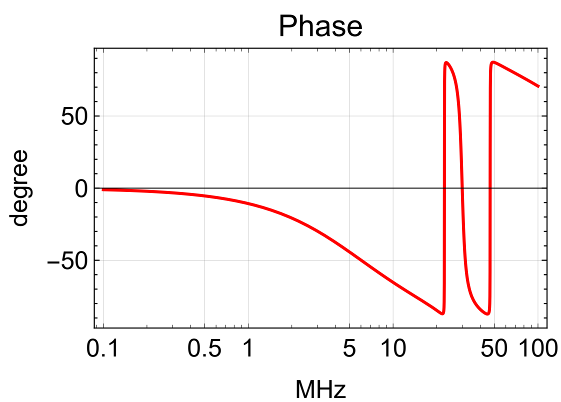
Modulation Input

$$\text{impSN2} = \text{par}\left[\text{R1},\ \text{s L1} + \text{L1R} + \frac{1}{\text{s C3}} + \frac{1}{\text{s C2}} + \text{par}\left[\text{s L2} + \text{L2R},\ \frac{1}{\text{s C4}},\ \frac{1}{\text{s Ceom}}\right]\right];$$

```
impSN2 /. s → 2 π ⅈ 35.5*^6 /. prmSN
BodePlotEx[impSN2 /. s → 2 π ⅈ 1*^6 f //. prmSN, {f, 0.1, 100},
 MagnitudeRange → All, PhaseRange → All, XAxisLabel → "MHz", plotoptn[1]]
```

51.6224 − 219.827 ⅈ





Equations

$$\text{eqSN1} = \frac{\text{Vout} - \text{Vin}}{\text{par}\left[\text{s L2} + \text{L2R},\ \frac{1}{\text{s C4}}\right]} + \frac{\text{Vout} - \text{Vm}}{\frac{1}{\text{s C3}} + \frac{1}{\text{s C2}} + \text{s L1} + \text{L1R}} + \frac{\text{Vout}}{\frac{1}{\text{s Ceom}}} == 0;$$

$$\text{eqSN2} = \frac{\text{Vm} - \text{Vout}}{\frac{1}{\text{s C3}} + \frac{1}{\text{s C2}} + \text{s L1} + \text{L1R}} + \frac{\text{Vm} - \text{Vmod}}{\text{Rterm}} + \frac{\text{Vm}}{\text{R1}} == 0;$$

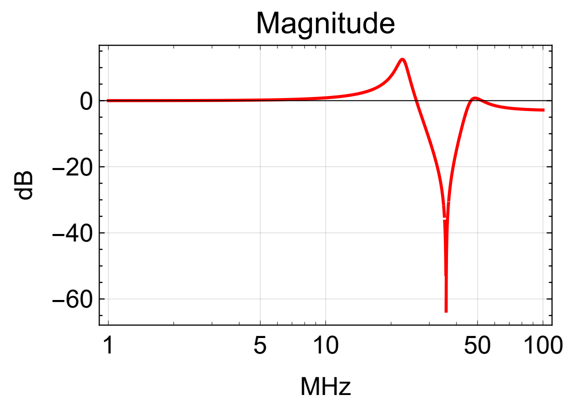
$$\text{solSN1} = \text{Simplify}\left[\frac{\text{Vout}}{\text{Vin}}\ \text{/. Solve[\{eqSN1, eqSN2\}, \{Vout\}, \{Vm\}][[1]] /. Vmod} \to 0\right];$$

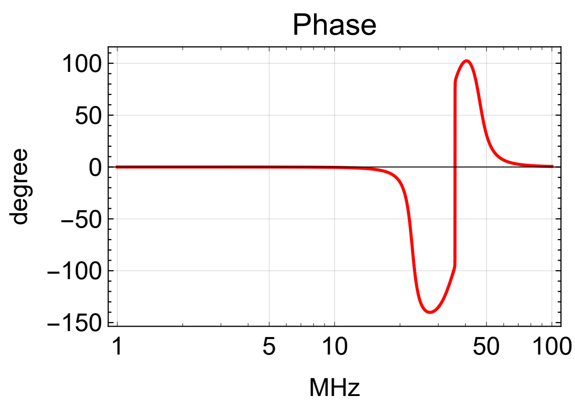
$$\text{solSN2} = \text{Simplify}\left[\frac{\text{Vout}}{\text{Vmod}}\ \text{/. Solve[\{eqSN1, eqSN2\}, \{Vout\}, \{Vm\}][[1]] /. Vin} \to 0\right];$$

Plots

Low Frequency Input

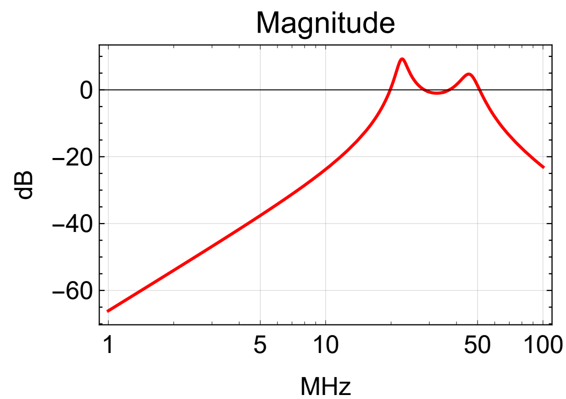
```
BodePlotEx[solSN1 /. s → 2 π ⅈ 1*^6 f //. prmSN, {f, 1, 100},
 MagnitudeRange → All, PhaseRange → All, XAxisLabel → "MHz", plotoptn[1]]
```

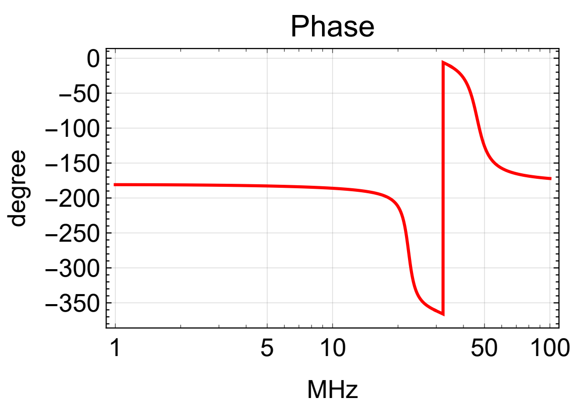




Modulation Input

```
BodePlotEx[solSN2 /. s → 2 π ⅈ 1*^6 f //. prmSN, {f, 1, 100},
 MagnitudeRange → All, PhaseRange → All, XAxisLabel → "MHz", plotoptn[1]]
```





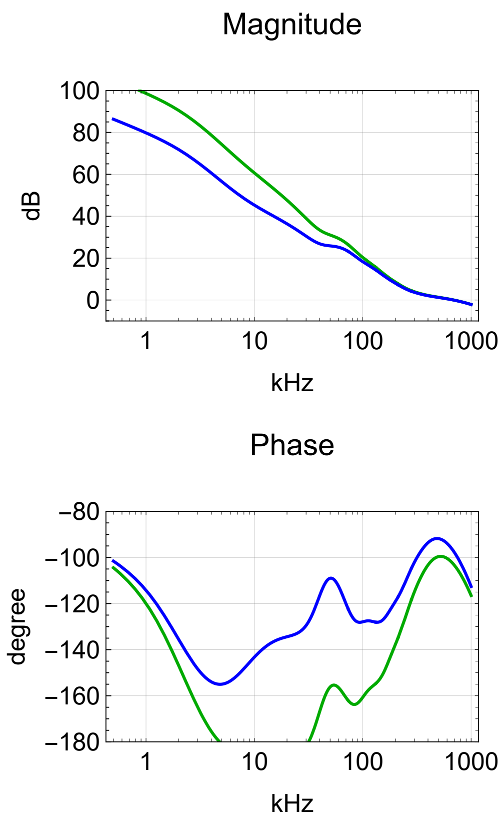
Frequency Noise Suppression

PSL

In[132]:=

```
pslOpt = Sequence[Sensing → "Mixer", sBoost → False, FastOnly → False, FastNotch → 4];
psl[f_] := Norm[{fPslSense pole[f, fVcoBW] Abs[ttfssTF[2 π ⅈ f, pslOpt] / (1 + ttfssTF[2 π ⅈ f, pslOpt])],
    npro[f] Abs[1 / (1 + ttfssTF[2 π ⅈ f, pslOpt])]}]
pslPrm[1] = {fPslSense → 2*^-3, fVcoBW → 100*^3};
pslPrm[2] = {fPslSense → 2*^-2, fVcoBW → 100*^3};
pslPrm[3] = {fPslSense → 1*^-1, fVcoBW → 100*^3};
```

```
In[•]:= prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[1], allTTFSS];
        BodePlotEx[{77.5/8.8 pole[ⅈ 1000 f, 8.8*^3] × zero[ⅈ 1000 f, 77.5*^3] × ttfssTF[2 π ⅈ 1000 f,
              pslOpt] /. prm, ttfssTF[2 π ⅈ 1000 f, pslOpt] /. prm}, {f, 0.5, 1000}, MagnitudeRange →
           {-10, 100}, PhaseRange → {-180, -80}, Evaluate[plotoptn[3]], XAxisLabel → "kHz"]
Out[•]=
```



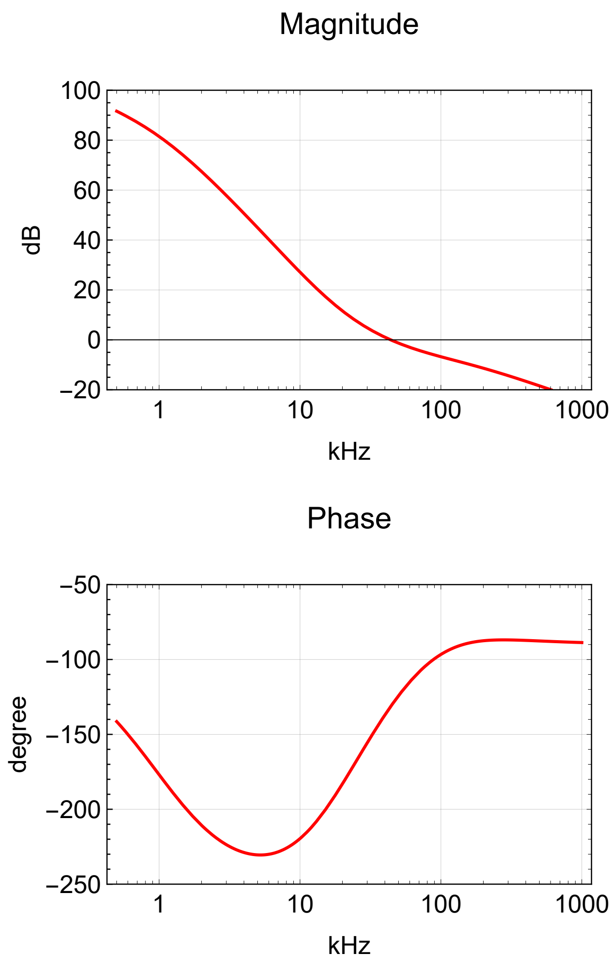
Mode Cleaner

In[137]:=

```
mcTF[f_] := fMcGain * pole[ⅈ f, fMCpole] *
   fMcComZero / fMcComPole pole[ⅈ f, fMcComPole] × zero[ⅈ f, fMcComZero] *
   (fMcBoostZero / fMcBoostPole pole[ⅈ f, fMcBoostPole] × zero[ⅈ f, fMcBoostZero])^2 *
   zero[ⅈ f, fMcFastZero] × pole[ⅈ f, fMcFastPole]
mcPrm = {fMcGain → 3.5, fMCpole → 8.8*^3,
    fMcComPole → 1.6, fMcComZero →  17*^3, fMcBoostPole → 1*^3,
    fMcBoostZero → 20*^3, fMcFastZero → 70*^3, fMcFastPole → 140*^3};
mcsensPrm[1] = {fMCSense → 4*^-6, fMCpole → 8.8*^3};
mcsensPrm[2] = {fMCSense → 1*^-4, fMCpole → 8.8*^3};
mc[f_] :=
  Norm[{fMCSense Abs[mcTF[f] / (1 + mcTF[f])], pole[ⅈ f, fMCpole] × psl[f] Abs[1 / (1 + mcTF[f])]}]
```

In[∘]:=
```
BodePlotEx[{mcTF[f 1*^3] /. mcPrm}, {f, 0.5, 1000}, MagnitudeRange → {-20, 100},
  PhaseRange → {-250, -50}, Evaluate[plotoptn[1]], XAxisLabel → "kHz"]
```

Out[∘]=



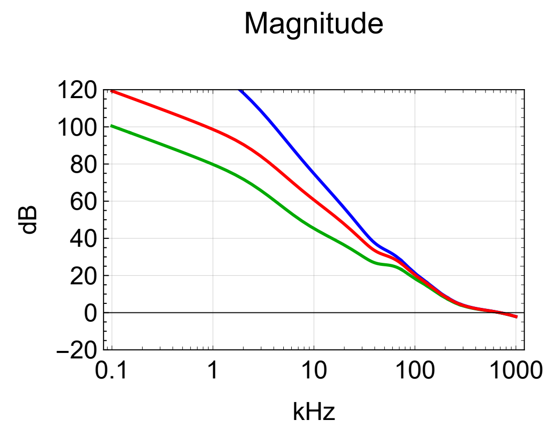
Mode Cleaner without Reference Cavity

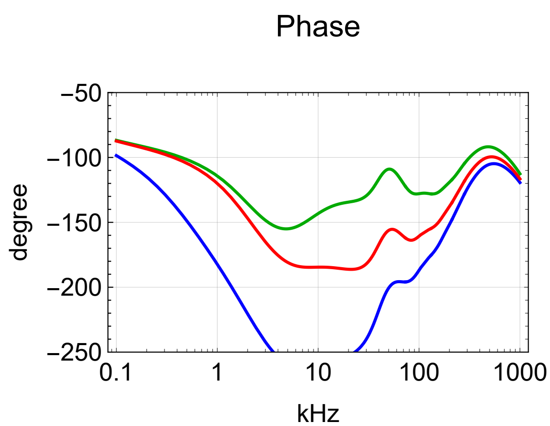
In[142]:=

```
mcttfssOpt =
  Sequence[Sensing → "Mixer", sBoost → False, FastOnly → False, FastNotch → 4];
mcttfssTF[f_] :=
 ttfssTF[2 π ⅈ f, pslOpt] (pRefCav/(2 π fMCpole)) pole[ⅈ f, fMCpole] × zero[2 π ⅈ f, pRefCav]
mcttfssTF2[f_] := mcttfssTF[f] ((50*^3/5*^2) pole[ⅈ f, 5*^2] × zero[ⅈ f, 50*^3])^1
mcttfss[f_] :=
 Norm[{fMCSense Abs[G/(1 + G)], npro[f ] × pole[ⅈ f, fMCpole] Abs[1/(1 + G)]}] /. G → mcttfssTF[f]
mcttfss2[f_] :=
 Norm[{fMCSense Abs[G/(1 + G)], npro[f ] × pole[ⅈ f, fMCpole] Abs[1/(1 + G)]}] /. G → mcttfssTF2[f]
```

In[ ]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[1], mcsensPrm[1], allTTFSS];
BodePlotEx[{ttfssTF[2 π i 1000 f, pslOpt] /. prm,
   mcttfssTF2[1000 f] /. prm, mcttfssTF[1000 f] /. prm},
  {f, 0.1, 1000}, MagnitudeRange → {-20, 120},
  PhaseRange → {-250, -50}, Evaluate[plotoptn[3]], XAxisLabel → "kHz"]
```

Out[ ]=

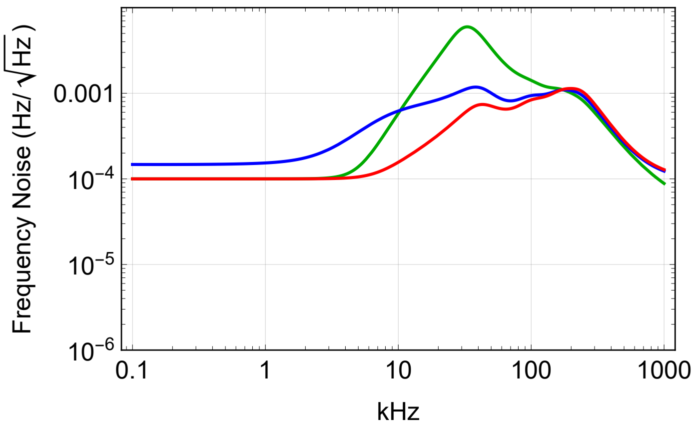




Current IMC Sensing Noise

In[]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[2], allTTFSS];
LogLogPlot[{mc[1000 f] /. mcPrm /. prm, mcttfss[1000 f] /. prm, mcttfss2[1000 f] /. prm},
 {f, 0.1, 1000}, PlotRange → {1*^-6, 1*^-2}, Evaluate[plotoptn[3]],
 FrameLabel → {"kHz", "Frequency Noise (Hz/ √Hz )"}]
```

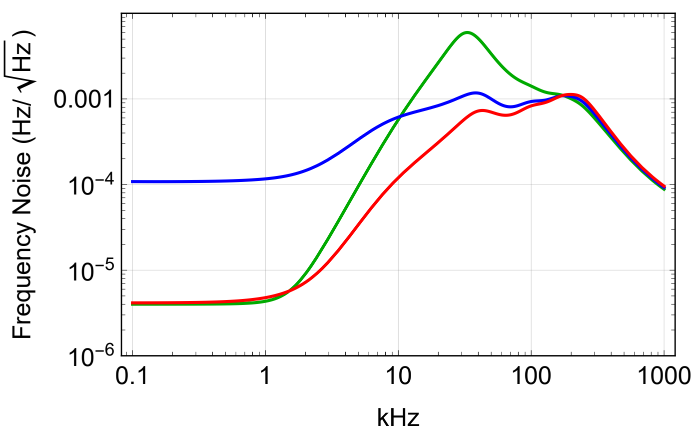
Out[]=



Ideal IMC Sensing Noise

In[]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[1], allTTFSS];
LogLogPlot[{mc[1000 f] /. mcPrm /. prm, mcttfss[1000 f] /. prm, mcttfss2[1000 f] /. prm},
 {f, 0.1, 1000}, PlotRange → {1*^-6, 1*^-2}, Evaluate[plotoptn[3]],
 FrameLabel → {"kHz", "Frequency Noise (Hz/ √Hz )"}]
```

Out[]=



Mode Cleaner with Separate EOM & TTFSS

In[147]:=

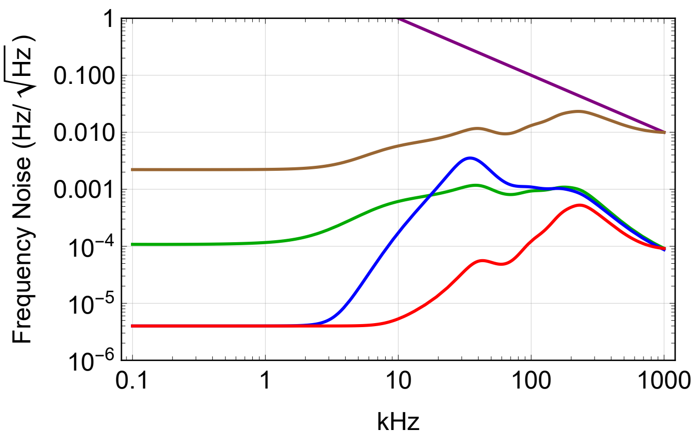
```
mc2[f_] := Norm[
  {fMCSense Abs[mcttfssTF[f] / (1 + mcttfssTF[f])], pole[ⅈ f, fMCpole] × psl[f] Abs[1 / (1 + mcttfssTF[f])]}]
```

Advanced LIGO VCO

In[∘]:=

```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[1], mcsensPrm[1], allTTFSS];
LogLogPlot[{npro[f 1*^3],
   psl[f 1*^3] /. prm, mcttfss[1000 f] /. prm, mc[1000 f] /. mcPrm /. prm, mc2[1000 f] /. prm},
  {f, 0.1, 1000}, PlotRange → {1*^-6, 1*^0}, Evaluate[plotoptn[5]],
  FrameLabel → {"kHz", "Frequency Noise (Hz/ √Hz )"}]
```

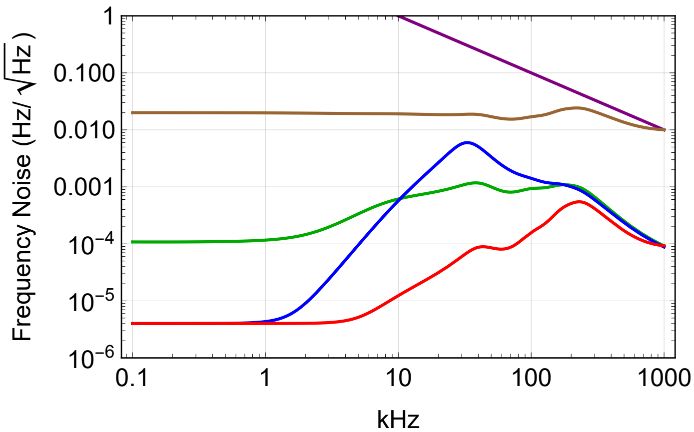
Out[∘]=



iLIGO VCO

In[ ]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[1], allTTFSS];
LogLogPlot[{npro[f 1*^3],
   psl[f 1*^3] /. prm, mcttfss[1000 f] /. prm, mc[1000 f] /. mcPrm /. prm, mc2[1000 f] /. prm},
  {f, 0.1, 1000}, PlotRange → {1*^-6, 1*^0}, Evaluate[plotoptn[5]],
  FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

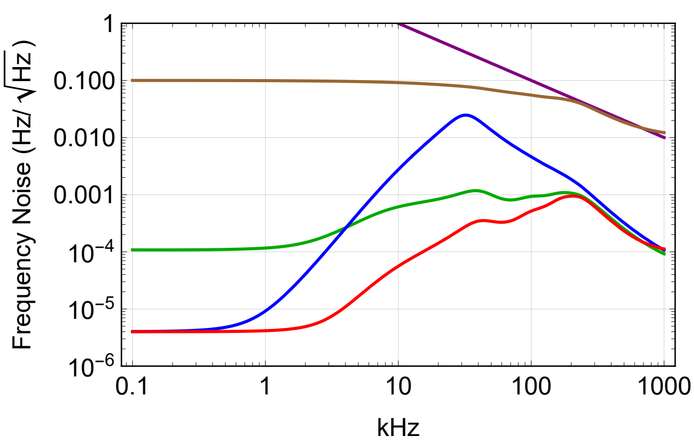
Out[ ]=



LLO VCO

In[ ]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[3], mcsensPrm[1], allTTFSS];
LogLogPlot[{npro[f 1*^3],
   psl[f 1*^3] /. prm, mcttfss[1000 f] /. prm, mc[1000 f] /. mcPrm /. prm, mc2[1000 f] /. prm},
  {f, 0.1, 1000}, PlotRange → {1*^-6, 1*^0}, Evaluate[plotoptn[5]],
  FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

Out[ ]=

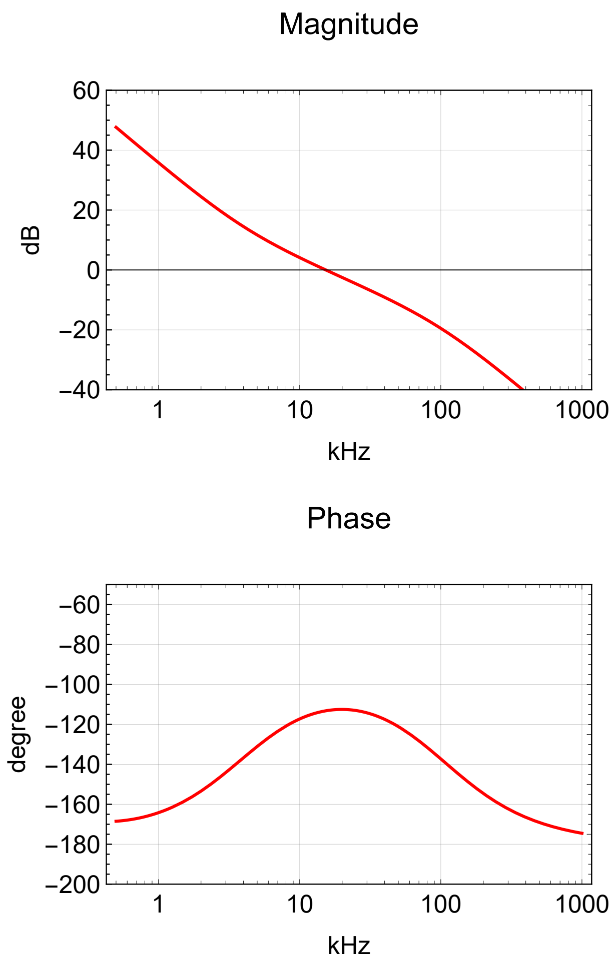


Common Mode

In[148]:=

```
cmTF[f_] := fCmGain * pole[ⅈ f, fDoubleCavPole] *
   fCmComZero/fCmComPole pole[ⅈ f, fCmComPole] × zero[ⅈ f, fCmComZero] × pole[ⅈ f, fCmBW]
cmPrm = {fCmGain → 2*^4, fDoubleCavPole → 0.75,
    fCmComPole → 40, fCmComZero → 4*^3, fCmBW → 100*^3, fCMSense → 3*^-10};
cm[f_] :=
  Norm[{fCMSense zero[ⅈ f, fDoubleCavPole] Abs[cmTF[f]/(1 + cmTF[f])], mc[f] Abs[1/(1 + cmTF[f])]}]
cm2[f_] := Norm[
   {fCMSense zero[ⅈ f, fDoubleCavPole] Abs[cmTF[f]/(1 + cmTF[f])], mcttfss[f] Abs[1/(1 + cmTF[f])]}]
cm3[f_] :=
  Norm[{fCMSense zero[ⅈ f, fDoubleCavPole] Abs[cmTF[f]/(1 + cmTF[f])], mc2[f] Abs[1/(1 + cmTF[f])]}]
```

```
In[*]:= BodePlotEx[{cmTF[f 1*^3] /. cmPrm}, {f, 0.5, 1000}, MagnitudeRange → {-40, 60},
         PhaseRange → {-200, -50}, Evaluate[plotoptn[1]], XAxisLabel → "kHz"]
Out[*]=
```



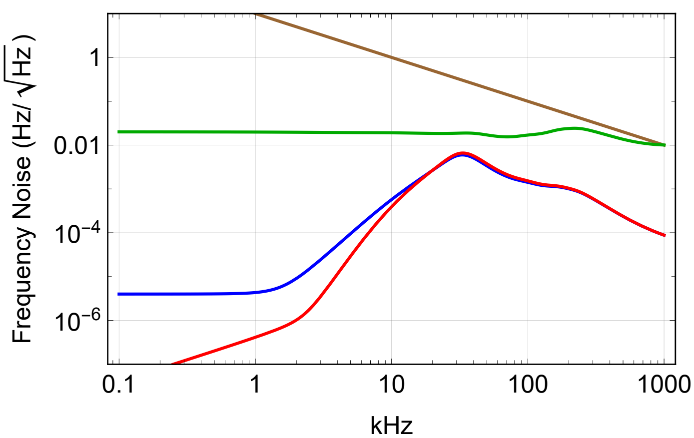
Interferometer Noise Budget

Advanced LIGO Standard

Ideal IMC Sensing Noise

In[•]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[1], allTTFSS];
LogLogPlot[{npro[f 1*^3],
  psl[f 1*^3] /. prm,
  mc[f 1*^3] /. prm /. mcPrm,
  cm[f 1*^3] /. prm /. mcPrm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^1},
 Evaluate[plotoptn[4]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

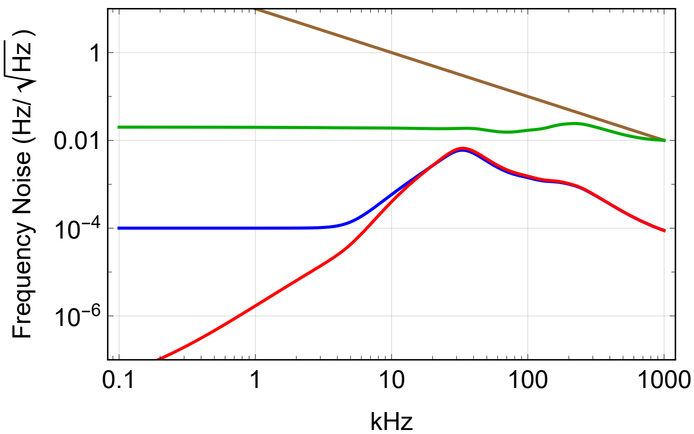
Out[•]=



Current IMC Sensing Noise

```
In[•]:= prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[2], allTTFSS];
LogLogPlot[{npro[f 1*^3],
   psl[f 1*^3] /. prm,
   mc[f 1*^3] /. prm /. mcPrm,
   cm[f 1*^3] /. prm /. mcPrm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^1},
 Evaluate[plotoptn[4]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

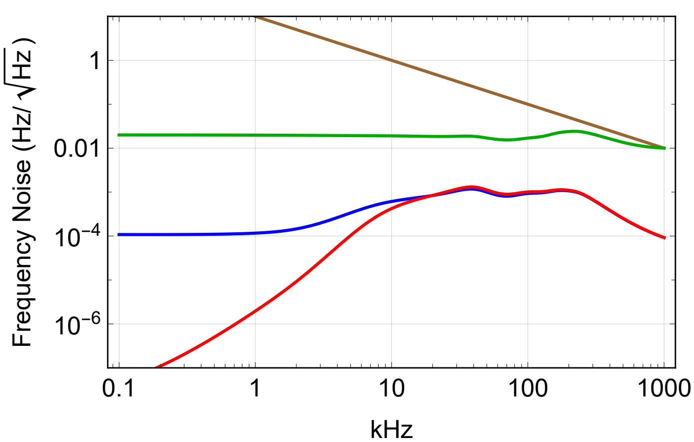
Out[•]=



Mode Cleaner without Reference Cavity

```
In[•]:= prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[1], allTTFSS];
LogLogPlot[{npro[f 1*^3],
   psl[f 1*^3] /. prm,
   mcttfss[f 1*^3] /. prm,
   cm2[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^1},
 Evaluate[plotoptn[4]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

Out[•]=

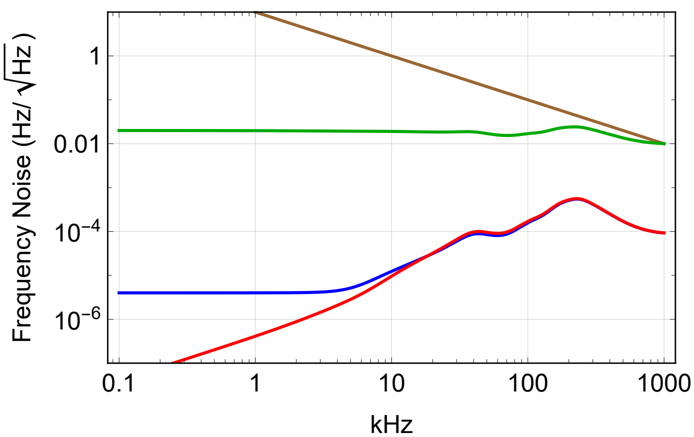


Mode Cleaner with Separate EOM & TTFSS

Ideal IMC Sensing Noise

In[ ]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[1], allTTFSS];
LogLogPlot[{npro[f 1*^3],
  psl[f 1*^3] /. prm,
  mcG2[f 1*^3] /. prm,
  cm3[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^1},
 Evaluate[plotoptn[4]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

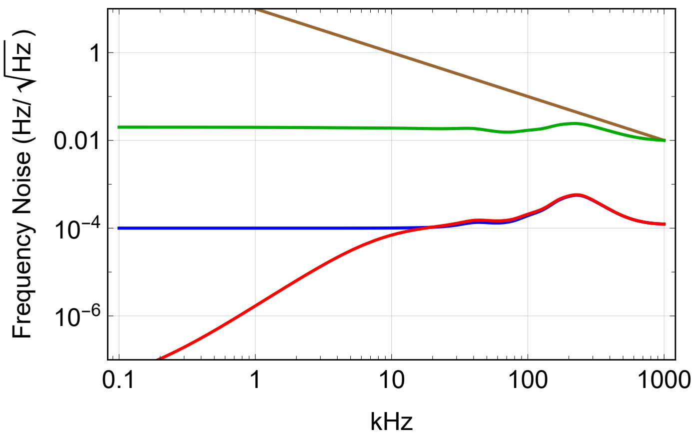
Out[ ]=



Current IMC Sensing Noise

In[•]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[2], allTTFSS];
LogLogPlot[{npro[f 1*^3],
   psl[f 1*^3] /. prm,
   mc2[f 1*^3] /. prm,
   cm3[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^1},
 Evaluate[plotoptn[4]], FrameLabel → {"kHz", "Frequency Noise (Hz/ √Hz )"}]
```

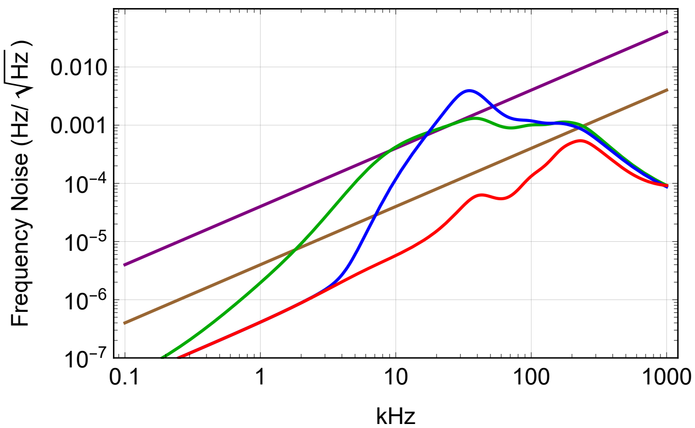
Out[•]=



CM Comparison: Advanced LIGO VCO

Ideal IMC Sensing Noise

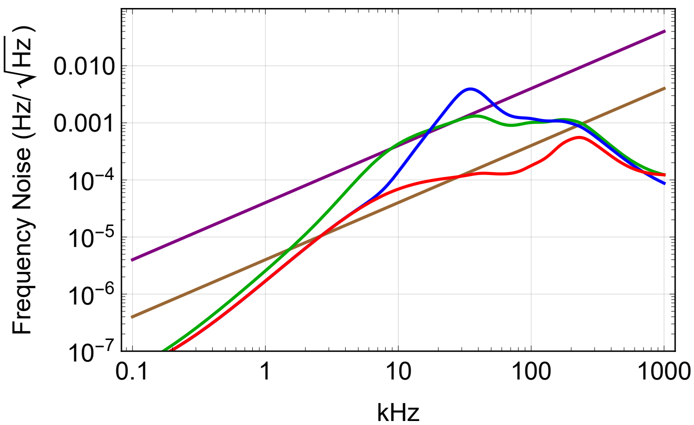
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[1], mcsensPrm[1], allTTFSS];
LogLogPlot[{4*^-5 f (*DARM*), 4*^-6 f (*est. current REFL shot*),
   cm2[f 1*^3] /. prm /. cmPrm,
   cm[f 1*^3] /. prm /. mcPrm /. cmPrm,
   cm3[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^-1},
 Evaluate[plotoptn[5]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```



Current IMC Sensing Noise

In[•]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[1], mcsensPrm[2], allTTFSS];
LogLogPlot[{4*^-5 f (*DARM*), 4*^-6 f (*est. current REFL shot*),
  cm2[f 1*^3] /. prm /. cmPrm,
  cm[f 1*^3] /. prm /. mcPrm /. cmPrm,
  cm3[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^-1},
 Evaluate[plotoptn[5]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

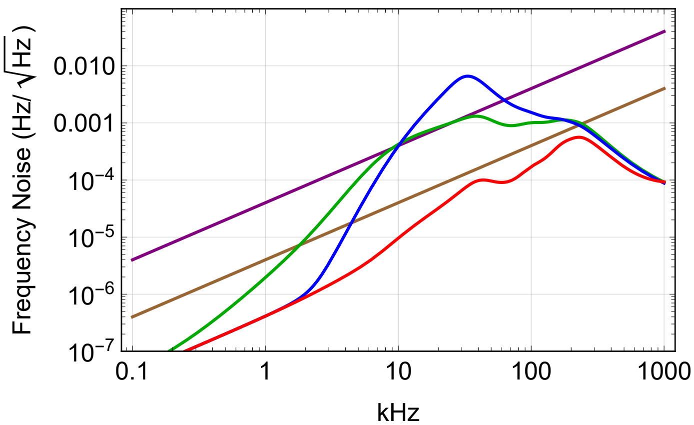
Out[•]=



CM Comparison: iLIGO VCO

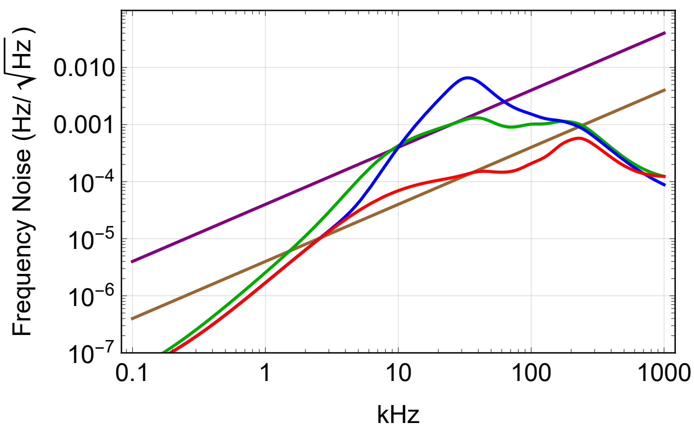
Ideal IMC Sensing Noise

```
In[•]:= prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[1], allTTFSS];
LogLogPlot[{4*^-5 f (*DARM*), 4*^-6 f (*est. current REFL shot*),
   cm2[f 1*^3] /. prm /. cmPrm,
   cm[f 1*^3] /. prm /. mcPrm /. cmPrm,
   cm3[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^-1},
 Evaluate[plotoptn[5]], FrameLabel → {"kHz", "Frequency Noise (Hz/ √Hz )"}]
```



Current IMC Sensing Noise

```
In[•]:= prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[2], mcsensPrm[2], allTTFSS];
LogLogPlot[{4*^-5 f (*DARM*), 4*^-6 f (*est. current REFL shot*),
   cm2[f 1*^3] /. prm /. cmPrm,
   cm[f 1*^3] /. prm /. mcPrm /. cmPrm,
   cm3[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^-1},
 Evaluate[plotoptn[5]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

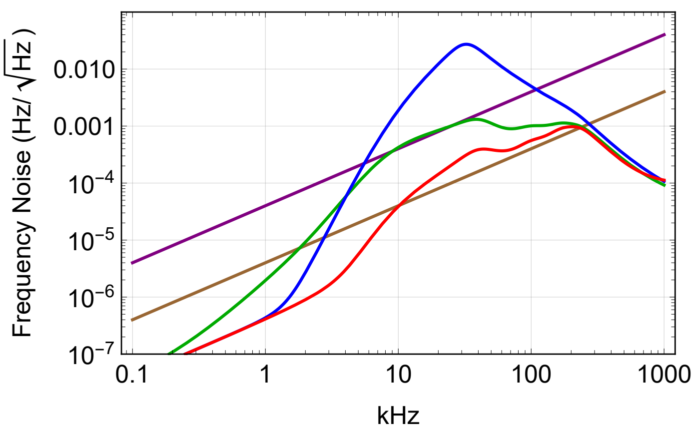


CM Comparison: LL VCO

Ideal IMC Sensing Noise

In[*]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[3], mcsensPrm[1], allTTFSS];
LogLogPlot[{4*^-5 f, 4*^-6 f,
   cm2[f 1*^3] /. prm /. cmPrm,
   cm[f 1*^3] /. prm /. mcPrm /. cmPrm,
   cm3[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^-1},
 Evaluate[plotoptn[5]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

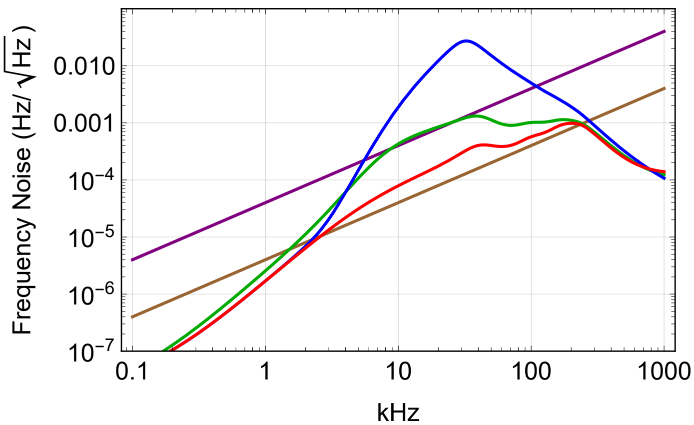
Out[*]=



Current IMC Sensing Noise

In[ ]:=
```
prm = Join[{cGain → 10^(8/20), fGain → 10^(20/20)}, pslPrm[3], mcsensPrm[2], allTTFSS];
LogLogPlot[{4*^-5 f, 4*^-6 f,
   cm2[f 1*^3] /. prm /. cmPrm,
   cm[f 1*^3] /. prm /. mcPrm /. cmPrm,
   cm3[f 1*^3] /. prm /. cmPrm}, {f, 0.1, 1000}, PlotRange → {1*^-7, 1*^-1},
  Evaluate[plotoptn[5]], FrameLabel → {"kHz", "Frequency Noise (Hz/√Hz)"}]
```

Out[ ]=



Additive Offset