4Gen TTFFS

Setup

```
Needs["Controls`LinearControl`"]
```

```
$TextStyle = {FontFamily → "Helvetica", FontSize → 13};
```

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

```
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
```

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

Equations

Schematics is D1700075-v1.

Parallel and Serial Impedance

```
par[r1_, r2_] := 1/(1/r1 + 1/r2)
par[r1_, r2_, r3_] := 1/(1/r1 + 1/r2 + 1/r3)
par[r1_, r2_, r3_, r4__] := 1/(1/r1 + 1/r2 + 1/r3 + Plus @@ (1/List[r4]))
ser[r1_, r2_] := r1 + r2
ser[r1_, r2_, r3_] := r1 + r2 + r3
ser[r1_, r2_, r3_, r4__] := r1 + r2 + r3 + Plus @@ List[r4]
```

Pole/zero

```
pole[s_, s0_] := 1/(1 + s/s0)
zero[s_, s0_] := 1 + s/s0
pole[s_, s0_, Q_] := 1/(1 + (1/Q) (s/s0) + (s/s0)^2)
zero[s_, s0_, Q_] := 1 + (1/Q) (s/s0) + (s/s0)^2
```

Transfer Function of an OpAmp

This function computes the transfer function of an idealized OpAmp circuit
g: +1 for non-inverting configuration or -1 for inverting configuration, 0 for differential configuration
z2: Impedance in feedback path [Ohm]
z1: Impedance of input path (inverting) or impedance to ground (non-inverting) [Ohm]

```
OpAmp[g_, z1_, z2_] :=
 Which[g > 0, 1 + z2/z1, g < 0, z2/z1, True, z2/z1]
```

Noise of an OpAmp

This function computes the equivalent input noise of an OpAmp circuit
g: +1 for non-inverting configuration or -1 for inverting configuration, 0 for differential configuration
z1: Impedance of input path (inverting) or impedance to ground (non-inverting) [Ohm]
z2: Impedance over feedback path [Ohm]
en: voltage noise [Volt]
in: current noise [Ampere]

```
FourKT = 1.62*^-20; (* V^2/Hz/Ohm; room temperature 20C *)
OpAmpNoise[g_, z1_, z2_, en_, in_] :=
 Which[g > 0, If[z1 == Infinity, Sqrt[en^2 + FourKT Abs[z2] + (in Abs[z2])^2],
    Sqrt[(en^2 + FourKT Abs[par[z1, z2]] + (in Abs[par[z1, z2]])^2)]],
  g < 0, Sqrt[(Abs[1 + z1/z2]^2 en^2 + Abs[z1]^2 (in^2 + Abs[FourKT/z1] + Abs[FourKT/z2]))],
  True, Sqrt[(Abs[1 + z1/z2]^2 en^2 + 2 Abs[z1]^2 (in^2 + Abs[FourKT/z1] + Abs[FourKT/z2]))]]
```

Flicker Noise: The variable $Flicker determines if flicker noise is added or not. It can also be explicitly specified with the option Flicker.

```
$Flicker = True;
Clear[OpAmpNoiseFlicker];
Options[OpAmpNoiseFlicker] = {Flicker :> $Flicker};
OpAmpNoiseFlicker[f_, fknee_, opts___] :=
  If[Flicker /. {opts} /. Options[OpAmpNoiseFlicker], Sqrt[fknee/f + 1], 1]
OpAmpNoiseFlicker[f_, fknee_, floor_, opts___] := floor OpAmpNoiseFlicker[f, fknee, opts]
```

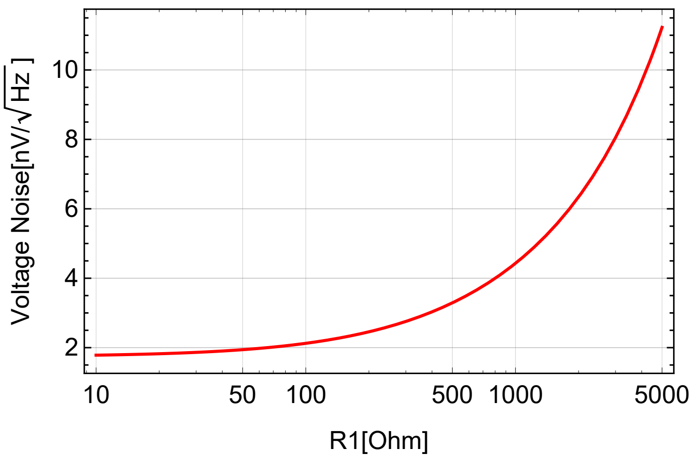
OpAmp Parameters

```
Clear[AD829, OP27]
AD829[f_] := {s → 2 π ⅈ f, en → enAD829, in → inAD829,
     enfloor → enfloorAD829, infloor → infloorAD829} //.
    {enAD829 → OpAmpNoiseFlicker[f, ekneeAD829, enfloorAD829],
     inAD829 → OpAmpNoiseFlicker[f, ikneeAD829, infloorAD829],
     ekneeAD829 → 50, ikneeAD829 → 100, (*guess*)
     enfloorAD829 → 1.7*^-9, infloorAD829 → 1.5*^-12};
OP27[f_] := {s → 2 π ⅈ f, en → enOP27, in → inOP27,
     enfloor → enfloorOP27, infloor → infloorOP27} //.
    {enOP27 → OpAmpNoiseFlicker[f, ekneeOP27, enfloorOP27],
     inOP27 → OpAmpNoiseFlicker[f, ikneeOP27, infloorOP27],
     ekneeOP27 → 2.7, ikneeOP27 → 140,
     enfloorOP27 → 3.0*^-9, infloorOP27 → 0.4*^-12};
LT1028[f_] := {s → 2 π ⅈ f, en → enLT1028, in → inLT1028,
     enfloor → enfloorLT1028, infloor → infloorLT1028} //.
    {enLT1028 → OpAmpNoiseFlicker[f, ekneeLT1028, enfloorLT1028],
     inLT1028 → OpAmpNoiseFlicker[f, ikneeLT1028, infloorLT1028],
     ekneeLT1028 → 3.5, ikneeLT1028 → 250,
     enfloorLT1028 → 0.85*^-9, infloorLT1028 → 1*^-12};
LT1128[f_] := {s → 2 π ⅈ f, en → enLT1128, in → inLT1128,
     enfloor → enfloorLT1128, infloor → infloorLT1128} //.
    {enLT1128 → OpAmpNoiseFlicker[f, ekneeLT1128, enfloorLT1128],
     inLT1128 → OpAmpNoiseFlicker[f, ikneeLT1128, infloorLT1128],
     ekneeLT1128 → 3.5, ikneeLT1128 → 250,
     enfloorLT1128 → 0.85*^-9, infloorLT1128 → 1*^-12};
AD797[f_] := {s → 2 π ⅈ f, en → enAD797, in → inAD797,
     enfloor → enfloorAD797, infloor → infloorAD797} //.
    {enAD797 → OpAmpNoiseFlicker[f, ekneeAD797, enfloorAD797],
     inAD797 → OpAmpNoiseFlicker[f, ikneeAD797, infloorAD797],
     ekneeAD797 → 50, ikneeAD797 → 100, (*guess*)
     enfloorAD797 → 0.9*^-9, infloorAD797 → 2*^-12};
LT1012[f_] := {s → 2 π ⅈ f, en → enLT1012, in → inLT1012,
     enfloor → enfloorLT1012, infloor → infloorLT1012} //.
    {enLT1012 → OpAmpNoiseFlicker[f, ekneeLT1012, enfloorLT1012],
     inLT1012 → OpAmpNoiseFlicker[f, ikneeLT1012, infloorLT1012],
     ekneeLT1012 → 2.5, ikneeLT1012 → 120, (*guess*)
     enfloorLT1012 → 14*^-9, infloorLT1012 → 6*^-15};
PA98A[f_] := {s → 2 π ⅈ f, en → enPA98A, in → inPA98A,
     enfloor → enfloorPA98A, infloor → infloorPA98A} //.
    {enPA98A → OpAmpNoiseFlicker[f, ekneePA98A, enfloorPA98A],
     inPA98A → OpAmpNoiseFlicker[f, ikneePA98A, infloorPA98A],
     ekneePA98A → 100(*guess*), ikneePA98A → 120, (*guess*)
     enfloorPA98A → 4*^-9, infloorPA98A → 1*^-12 (*guess*)};
```

Examples (AD829)

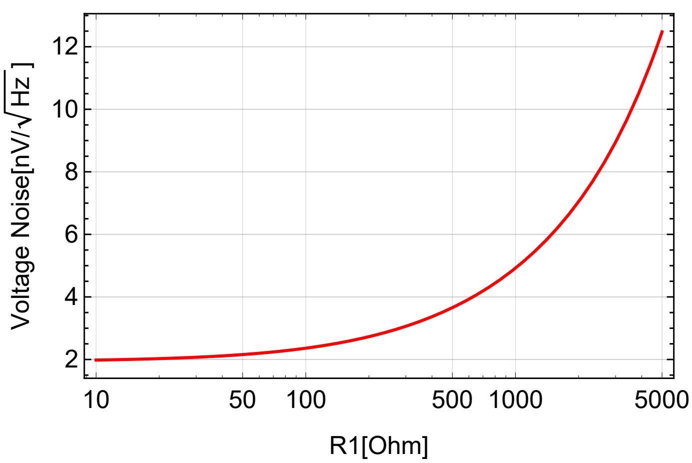
Non-Inverting configuration: input noise w/ gain of 10 as function of r1

```
LogLinearPlot[1*^9 OpAmpNoise[+1, r, 9 r, en, in] /. AD829[1000],
 {r, 10, 5000}, FrameLabel → {"R1[Ohm]", "Voltage Noise[nV/√Hz ]"},
 Frame → True, GridLines → Automatic, Evaluate[plotopt]]
```



Inverting configuration: input noise w/ gain of 10 as function of r1

```
LogLinearPlot[1*^9 OpAmpNoise[-1, r, 9 r, en, in] /. AD829[1000],
 {r, 10, 5000}, FrameLabel → {"R1[Ohm]", "Voltage Noise[nV/√Hz ]"},
 Frame → True, GridLines → Automatic, Evaluate[plotopt]]
```



Series Product of OpAmps

Computes the transfer function of several OpAmps circuits in series.

```
OpAmpProduct[t_, m_] := Product[t[i], {i, m}]
OpAmpProduct[t_List] := Product @@ t
```

Computes the equivalent input noise of several OpAmps circuits in series.

```
NoiseSum[prev_, {t_, n_}] := Sqrt[prev^2 + n^2] Abs[t]
OpAmpNoiseProduct[t_, n_, m_] :=
 Fold[NoiseSum, 0, Table[{t[i], n[i]}, {i, m}]] / Abs[OpAmpProduct[t, m]]
OpAmpNoiseProduct[t_List, n_List] := Fold[NoiseSum, 0, Transpose[t, n]] / Abs[OpAmpProduct[t]]
```

Spectrum Math

Propagate noise spectrum

```
SpecProp[prev_, t_] := {#[[1]], Abs[t /. s → 2. π #[[1]]] #[[2]]} & /@ prev
SpecProp[noise_, t_List] := FoldList[SpecProp, noise, t]
SpecProp[noise_, t_, m_] := FoldList[SpecProp, noise, Table[t[i], {i, m}]]
```

RMS of spectrum

```
Clear[SpecRMS];
SpecRMS[l_List ? (MatrixQ[#, NumberQ] &)] := Block[{i, sqr = 0},
  For[i = 1, i < Length[l], ++i,
   sqr += (l[[i + 1, 1]] - l[[i, 1]]) ((l[[i, 2]] + l[[i + 1, 2]]) / 2)^2];
  Sqrt[sqr]]
```

Integrated RMS spectrum

```
Clear[RMSSpec];
RMSSpec[l_List ? (MatrixQ[#, NumberQ] &), dir_: (-1)] := Block[{i, sqr = 0, r = N[l]},
  If[dir ≥ 0,
   For[i = 2, i ≤ Length[l], ++i,
    r[[i, 2]] = Sqrt[(r[[i - 1, 2]]^2 + r[[i, 2]]^2 (r[[i, 1]] - r[[i - 1, 1]]))]],
   For[i = Length[l] - 2, i >= 1, --i,
    r[[i, 2]] = Sqrt[(r[[i + 1, 2]]^2 + r[[i, 2]]^2 (r[[i + 1, 1]] - r[[i, 1]]))]]];
  r]
```

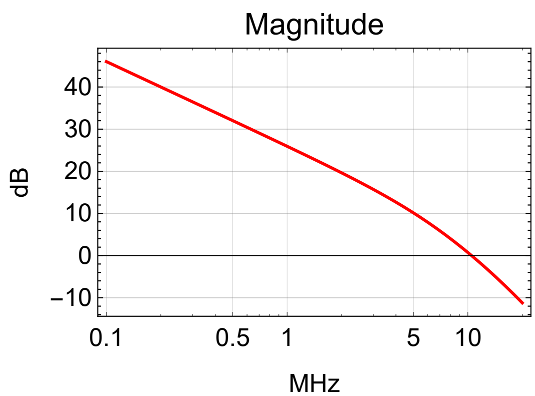
EOM Actuator Path

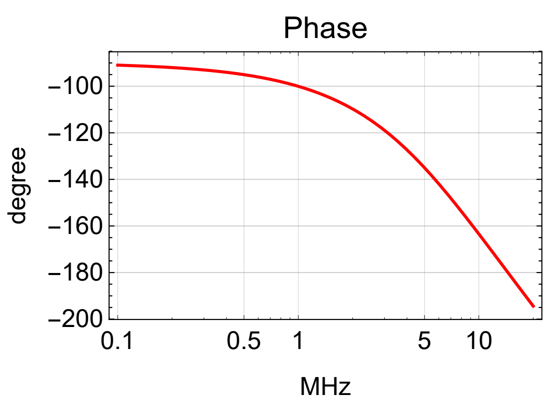
PA98 Open Loop Gain

Data sheet at www.apexanalog.com.

```
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 *)
```

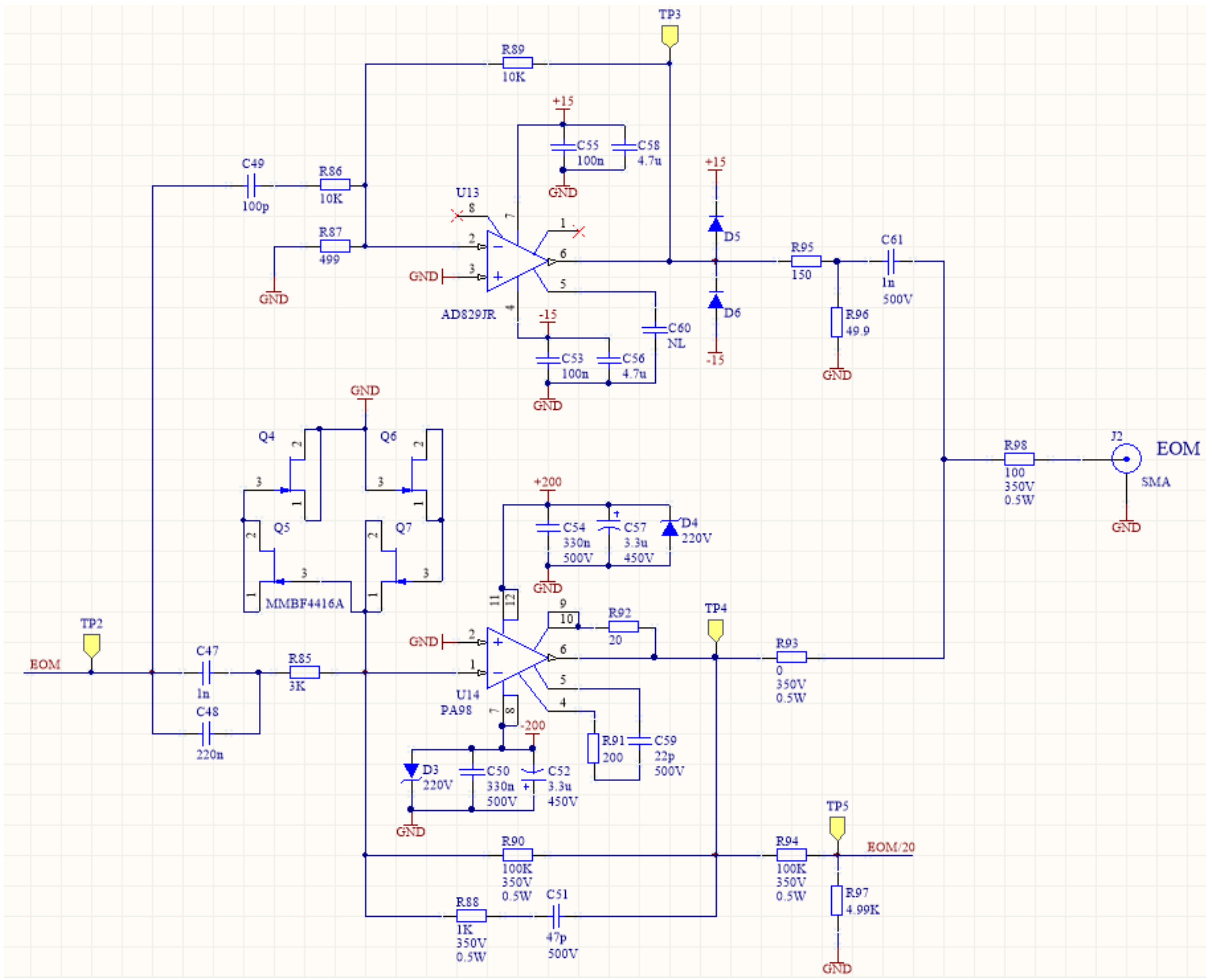
```
BodePlotEx[pa98[2 π ⅈ 1*^6 f] /. prmpa98, {f, 0.1, 20}, Evaluate[plotopt], XAxisLabel → "MHz"]
```





Old Double Path Configuration

Schematics



Transfer Function

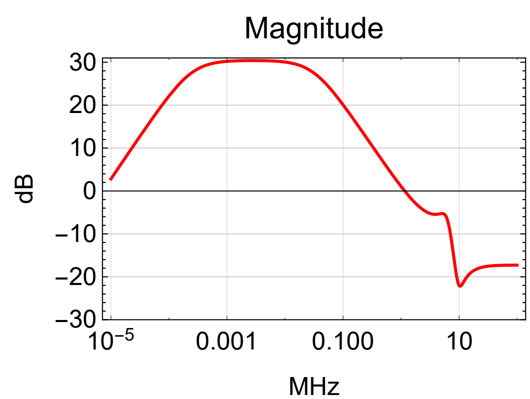
```
prmFbEom = {Zin → R85 + 1/(s C48), Zfb → par[R90, R88 + 1/(s C51)]};
prmActEom2Path = {R90 → 100*^3, R88 → 1*^3, C51 → 47*^-12, R85 → 3*^3,
    C48 → 220*^-9, C49 → 100*^-12, R86 → 10*^3, R87 → 499, R89 → 10*^3,
    R93 → 0, R95 → 150, R96 → 50, C61 → 1*^-9, R98 → 100, Ceom → 10*^-12};
```

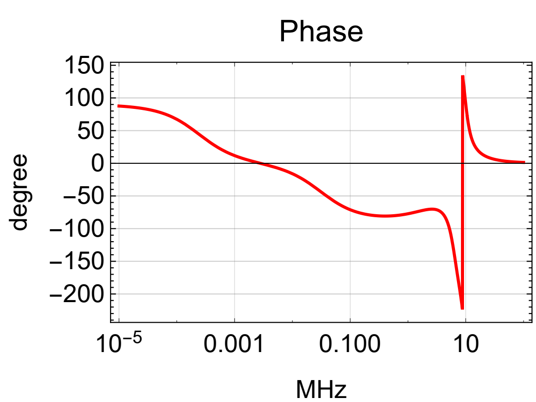
```
u14[s_] := - Zfb/Zin /. prmFbEom
u13[s_] := - R89/(R86 + 1/(s C49))
```

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"]
```



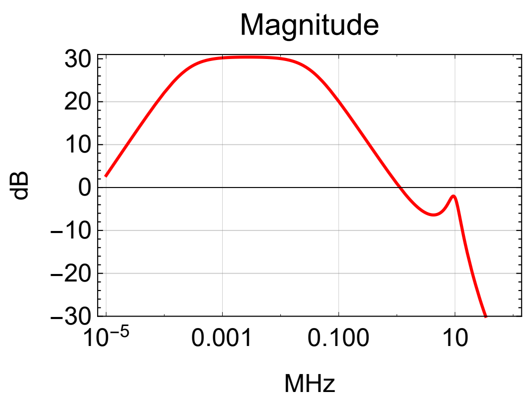


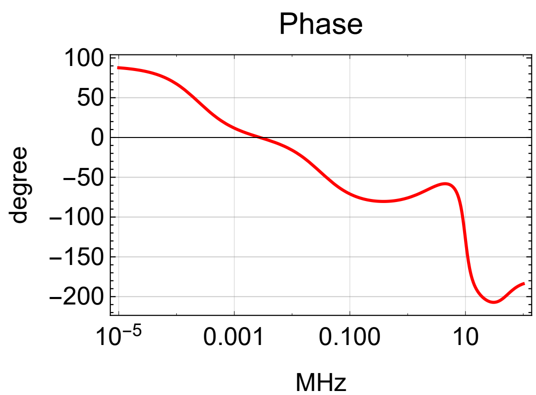
Single Path Configuration with Old Parameters

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

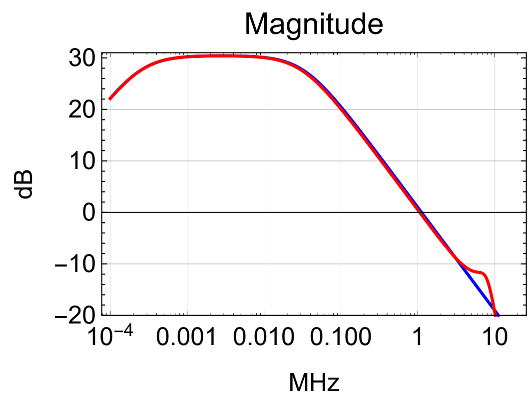
```
prmActEom = {R90 → 100*^3, R88 → 1*^3, C51 → 47*^-12,
    R85 → 3*^3, C48 → 220*^-9, C49 → 0, R86 → 10*^3, R87 → 499, R89 → 10*^3,
    R93 → 100, R95 → ∞, R96 → 0, C61 → 560*^-12, R98 → 0, Ceom → 10*^-12};
paPole = {gPA → R90/R85, pPA1 → 1/(C48 R85), pPA2 → 1/(C51 (R90 + R88))} /. prmActEom;
eomPrm = 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
```

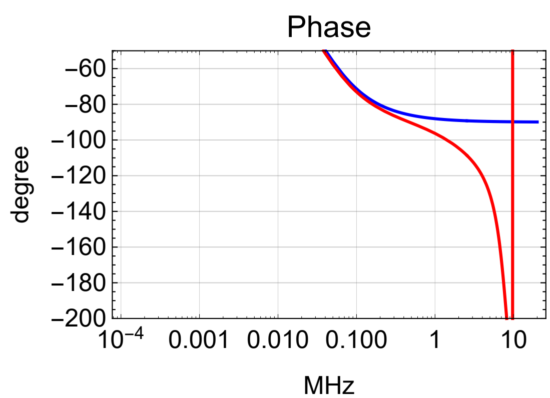
```
{241.144, 33527.5}
```

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)

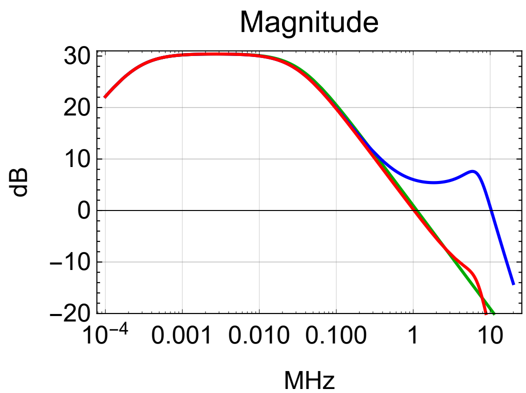
We limit the gain roll-off above 600 kHz by increasing R88 to 5.62K. We also eliminate C61, since it is not needed.

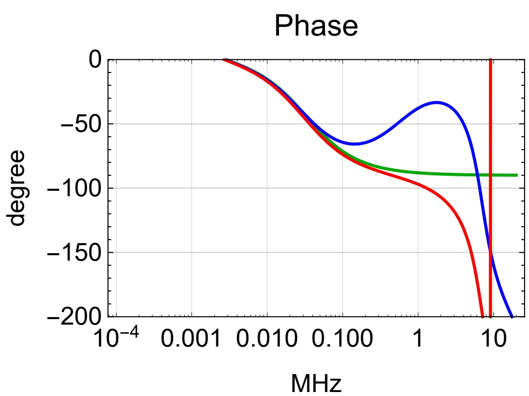
Transfer Function

```
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

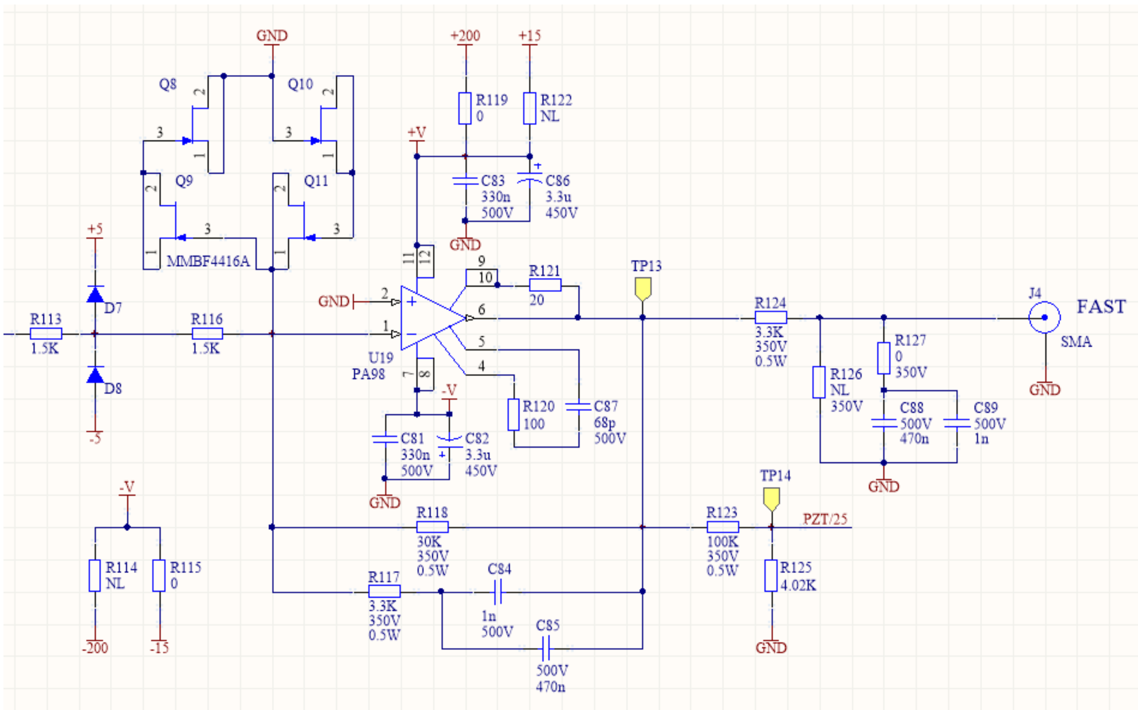
```
BodePlotEx[{eomTF[s] /. paPole /. s → 2 π ⅈ 1*^6 f,
  -eomact[s] /. prmpa98 /. prmActEomPMC /. s → 2 π ⅈ 1*^6 f,
  -eomact[s] pole[s, pPMC] /. prmpa98 /. prmActEomPMC /. pmcPole /. s → 2 π ⅈ 1*^6 f},
 {f, 0.0001, 20}, MagnitudeRange → {-20, 31}, PhaseRange → {-200, 0},
 Evaluate[plotoptn[3]], XAxisLabel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





PZT Actuator Path

Schematics



Transfer Function

```
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_] := gPZT pole[s, pPZT1]
pztCoeff[s_] := coefPZT pole[s, bwPZT] (* rad/s/V *)
{pPZT1/(2. π)} /. pztPrm
```

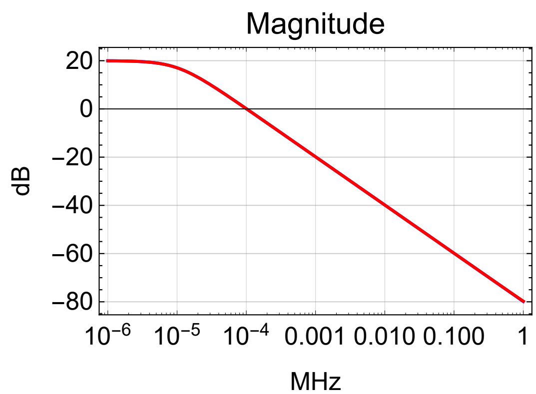
{10.169}

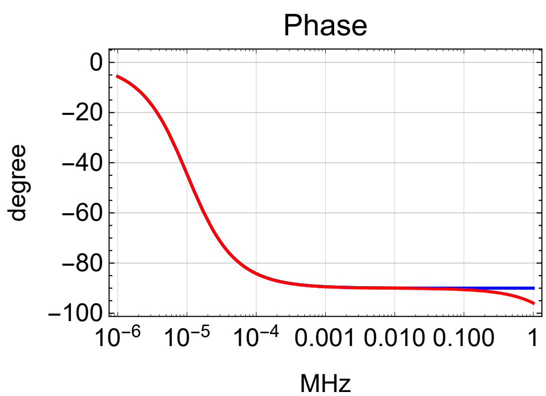
```
u19[s_] := -Zfb/Zin /. prmFbPZT
```

Pole/Zero Determination

Bode Plot

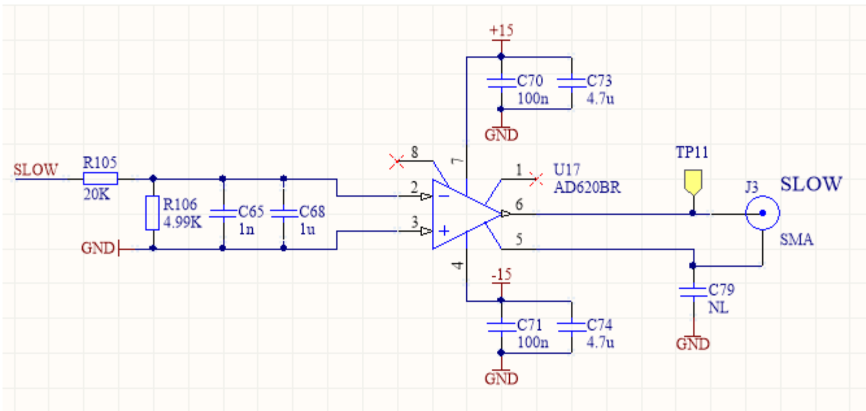
```
BodePlotEx[
 {pztTF[2 π ⅈ 1*^6 f] /. pztPole, -pztactPath[2 π ⅈ 1*^6 f] /. prmpa98 /. prmActPztPath},
 {f, 0.000001, 1}, Evaluate[plotoptn[2]], XAxisLabel → "MHz"]
```





Slow Actuator Path

Schematics



Transfer Function

```
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
```

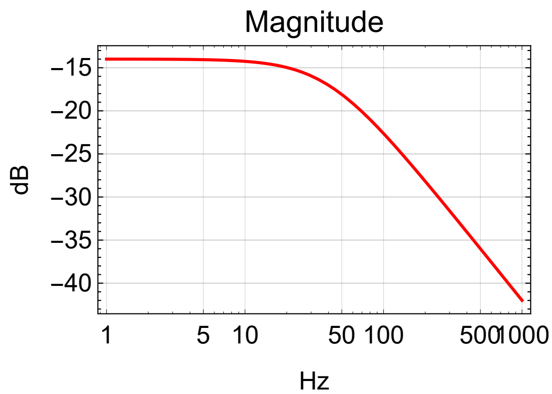
{0.19968, 39.8525}

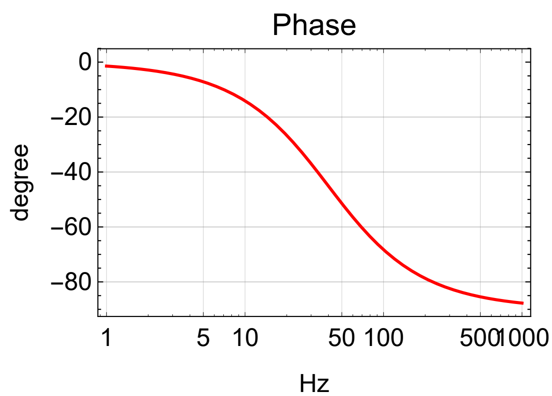
$$\frac{\text{par}\left[\text{R106}, \frac{1}{\text{s C65}}\right]}{\text{par}\left[\text{R106}, \frac{1}{\text{s C65}}\right] + \text{R105}} \text{ // Together}$$

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

Bode Plot

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





Sensing Path

Phase-Frequency Discriminator (Laser Locking)

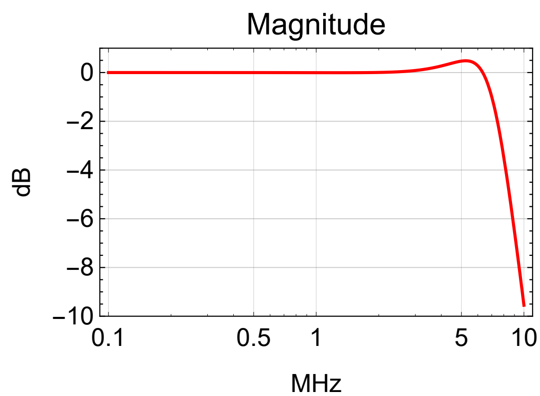
A phase-frequency discriminator is used for laser locking. The standard LIGO PFD circuit is described in LIGO-E1200114. The PCB LIGO-D1002471 is used with a modification to make it higher bandwidth. This is described in LIGO-E1700100.

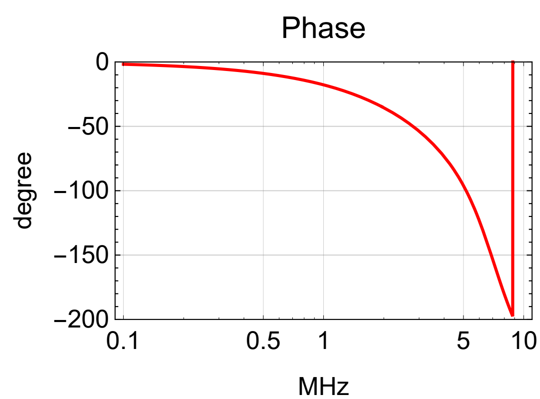
Transfer Function

```
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

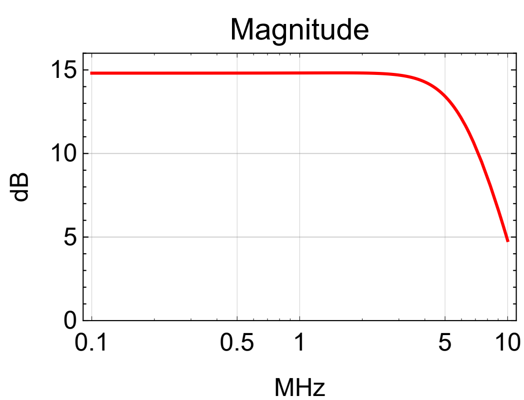
```
prmDemod = {gDemod → 5.5, pDemod1 → 2 π 15.9*^6, pDemod2 → 2 π 6.17*^6, qDemod2 → 0.761};
demod[s_] := gDemod pole[s, pDemod1] pole[s, pDemod2, qDemod2]
```

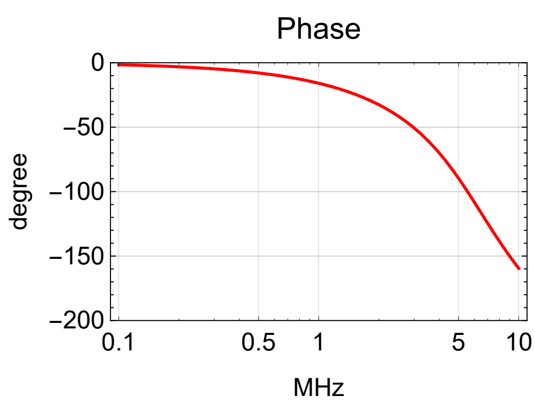
Reference Cavity

```
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]
```

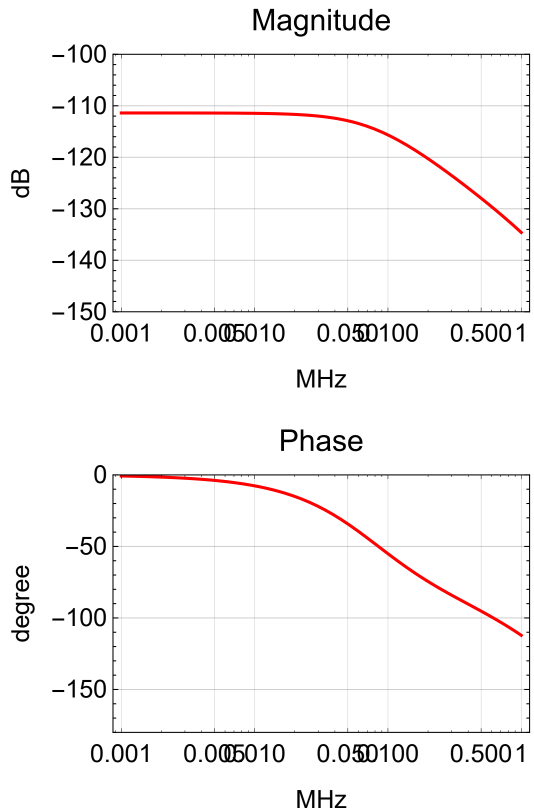
Bode Plot

```
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

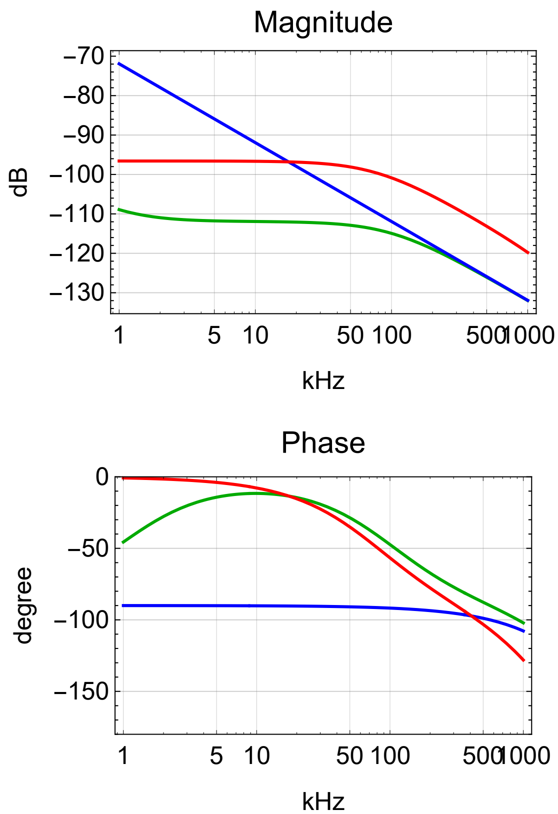
The sensing path transfer function combines one of the sensors, PFD or mixer, with an optional anti-boost. The anti-boost will make the PFD transfer function look more like the mixer one.

Transfer Function

```
prmSensing = {sBoostGain → 100, sBoostPole → 2 π 100 000};
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

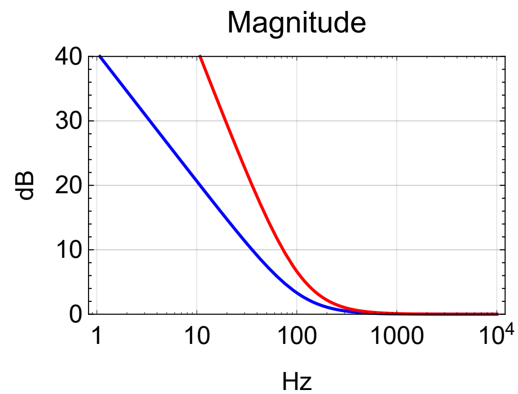
```
poleCommon = {R89 → 3.16*^3, R87 → 3.16*^3, C101 → 330*^-9,
    R88 → 3.16*^3, R90 → 3.16*^3, C107 → 330*^-9};
prmCommon = {cGain → 10^(0/20), gCom1 → R87/R89, zCom1 → 1/(C101 R87),
     gCom2 → R90/R88, zCom2 → 1/(C107 R90)} /. poleCommon;
allCommon := prmCommon;

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

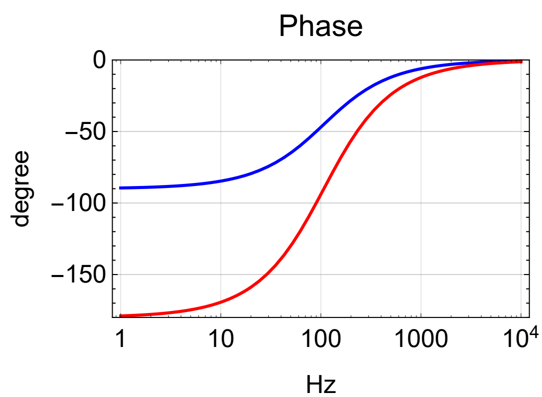
Parameters

```
N[{gCom1, zCom1/(2 π)} /. allCommon]
N[{gCom2, zCom2/(2 π)} /. allCommon]
{1., 107.161}

{1., 107.161}
```

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

Transfer Function

```
poleFast = {R35 → 1000, R33 → 3.16*^3, C36 → 10*^-12, R30 → 28*^3, C43 → 2.2*^-9,
   R36 → 1000, 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]

Options[fastTF] = {FastOnly → False};
fastTF[s_, opts___] :=
 If[FastOnly /. {opts} /. Options[fastTF],
     gFast3 pole[s, pFast3] zero[s, zFast3] * -1
     (*gFast4 pole[s,pFast4A]zero[s,zFast4A]*),
     fGain gFast1 pole[s, pFast1] zero[s, zFast1] gFast2 pole[s, pFast2]] *
    gFast5 /. {opts} /. Options[fastTF]
```

Parameters

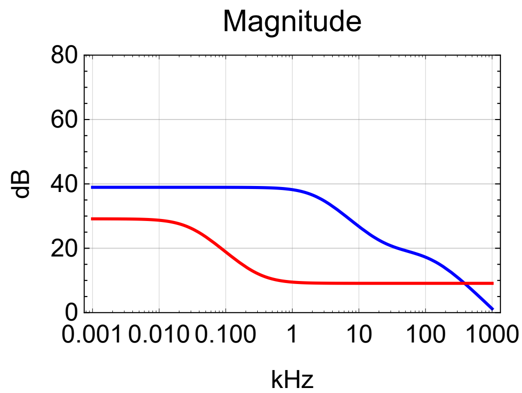
```
N[{gFast1, pFast1/(2 π), zFast1/(2 π)} /. allFast]
N[{gFast2, pFast2/(2 π)} /. allFast]
N[{gFast3, pFast3/(2 π), zFast3/(2 π)} /. allFast]
N[{gFast4, pFast4A/(2 π), zFast4A/(2 π) (*, pFast4B/(2 π), zFast4B/(2π) *)} /. allFast]
```

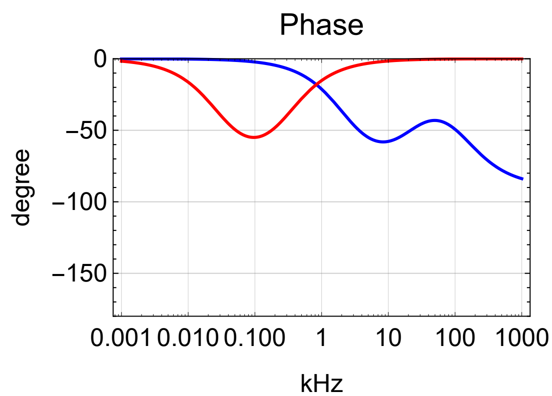
```
{-28., 2321.67, 22893.4}

{-3.16, 129142.}

{-28.6573, 30.3708, 305.245}

{-100., 4.77513, 482.288}
```

Pole/zero Determination

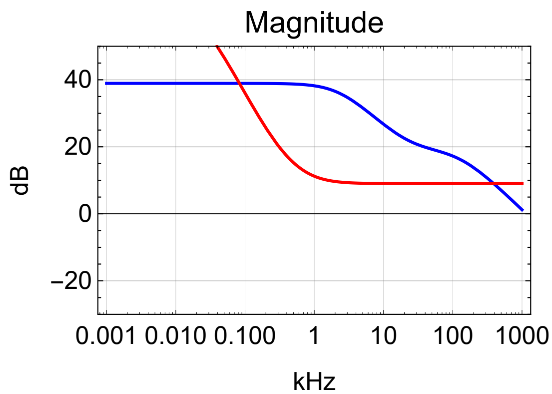
Bode Plot

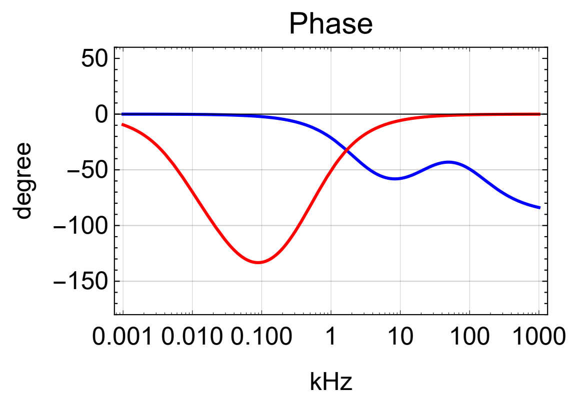
```
BodePlotEx[
 {-fastTF[2 π ⅈ f 1*^3] //. allFast, -fastTF[2 π ⅈ f 1*^3, FastOnly → True] //. allFast},
 {f, 0.001, 1000}, MagnitudeRange → {0, 80}, PhaseRange → {-180, 0},
 Evaluate[plotoptn[2]], XAxisLabel → "kHz"]
```





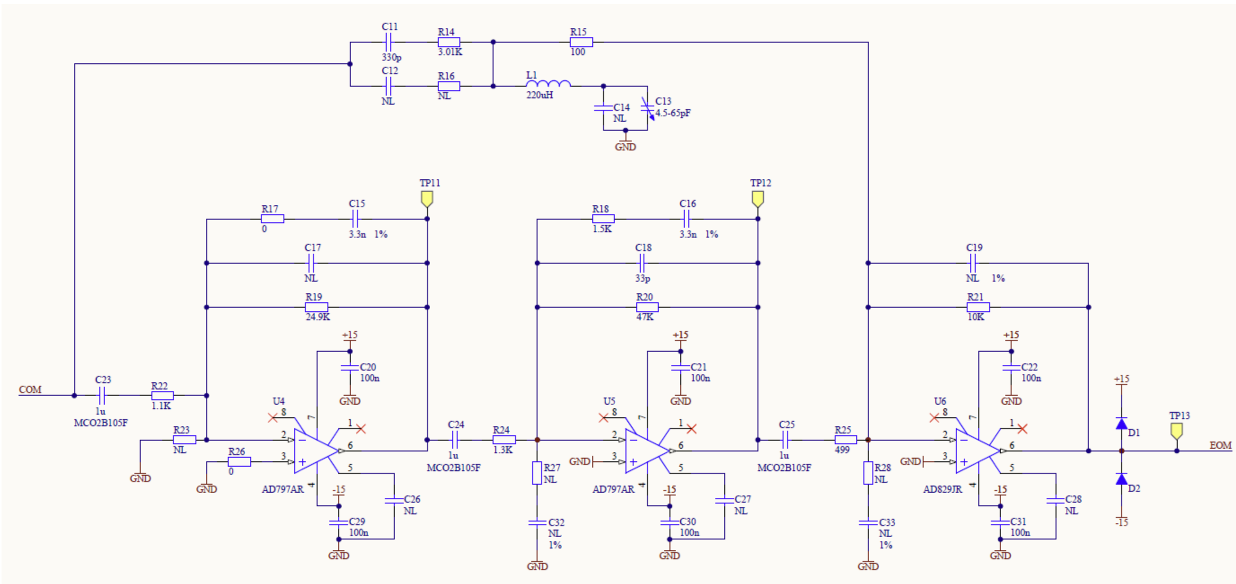
```
BodePlotEx[
 {-fastTF[2 π ⅈ f 1*^3] //. allFast, -fastTF[2 π ⅈ f 1*^3, FastOnly → True] //. allFast},
 {f, 0.001, 1000}, MagnitudeRange → {-30, 50},
 PhaseRange → {-180, 60}, Evaluate[plotoptn[2]], XAxisLabel → "kHz"]
```





EOM Path

Schematics



Transfer Function

```
poleEom = {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, R14 → 3.16*^3, C11 → 1*^-9, L1 → 220*^-6,
   C13 → 30*^-12, R15 → 100, R25 → 499., C25 → 100*^-9, R21 → 1*^3, C19 → 0.1*^-12};
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),
    gEOM4 → -R21/(R14 + R15), pEOM4a → 1/(C11 (R14 + R15)), zEOM4b → 1/Sqrt[C13 L1],
    pEOM4b → 1/Sqrt[C13 L1], qEOM4b → Sqrt[C13 L1]/(C13 par[R14, R15]), pEOM4c → 1/(C19 R21)} /. poleEom;
allEom := Join[eomPrm, prmEom]

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]
eom2TF[s_, opts___] :=
 gEOM4 s/pEOM4a pole[s, pEOM4a] zero[s, zEOM4b, ∞] pole[s, pEOM4b, qEOM4b] pole[s, pEOM4c] /.
   {opts} /. Options[eomTF]
eomTF[s_, opts___] := eom1TF[s, opts] + eom2TF[s, opts]
```

Parameters

$$\left\{\text{gEOM1}, \frac{\text{pEOM1a}}{2.\,\pi}, \frac{\text{pEOM1b}}{2\,\pi}, \frac{\text{zEOM1b}}{2\,\pi}, \frac{\text{pEOM1c}}{2\,\pi}\right\} \text{ /. prmEom /. poleEom}$$

$$\left\{\text{gEOM2}, \frac{\text{pEOM2a}}{2\,\pi}, \frac{\text{pEOM2b}}{2\,\pi}, \frac{\text{zEOM2b}}{2\,\pi}, \frac{\text{pEOM2c}}{2\,\pi}\right\} \text{ /. prmEom /. poleEom}$$

$$\left\{\text{gEOM3}, \frac{\text{pEOM3a}}{2.\,\pi}, \frac{\text{pEOM3b}}{2\,\pi}\right\} \text{ /. prmEom /. poleEom}$$

$$\left\{\text{gEOM4}, \frac{\text{pEOM4a}}{2.\,\pi}, \frac{\text{pEOM4b}}{2.\,\pi}, \text{qEOM4b}, \frac{\text{pEOM4c}}{2.\,\pi}\right\} \text{ /. prmEom /. poleEom}$$

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

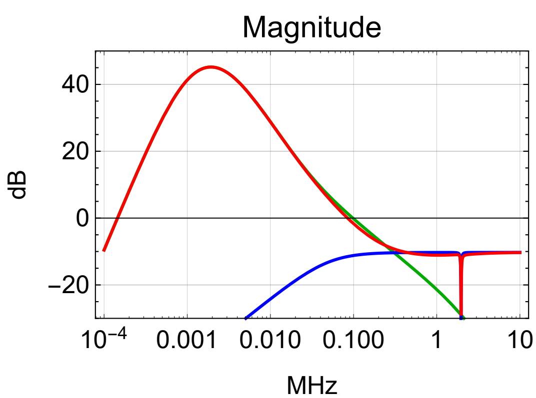
$\{-30.8228, 1007.31, 959.204, 30524.5, 2.21275\times10^{6}\}$

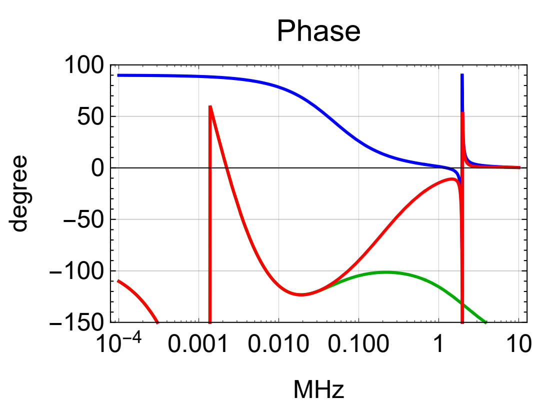
$\{-2.00401, 3189.48, 1.59155\times10^{9}\}$

$\{-0.306748, 48820.5, 1.95906\times10^{6}, 27.9371, 1.59155\times10^{9}\}$

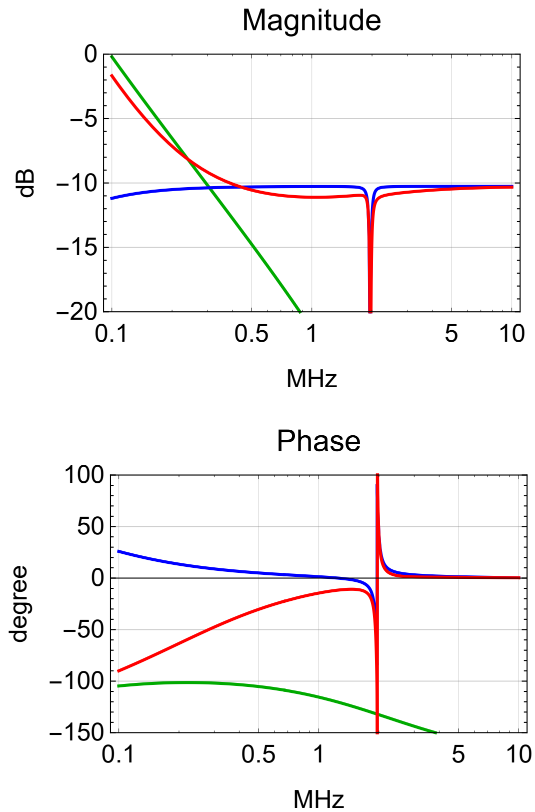
Bode Plot

```
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 → {-30, 50},
 PhaseRange → {-150, 100}, Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```

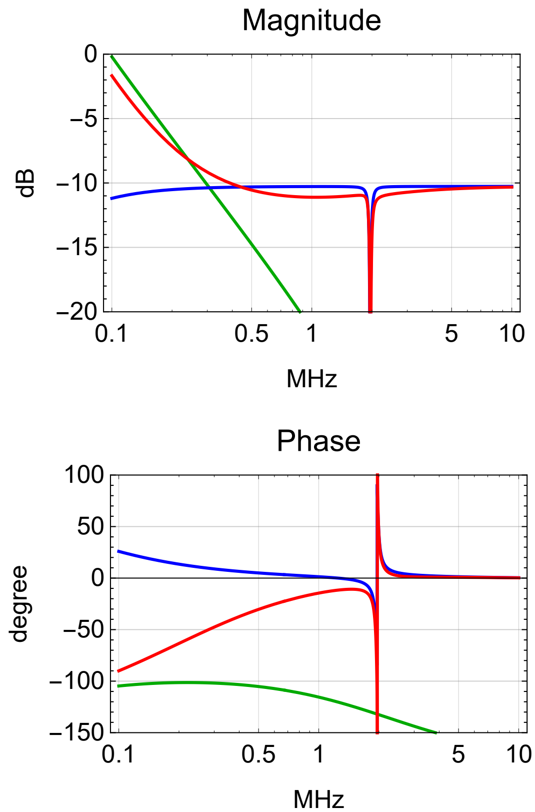




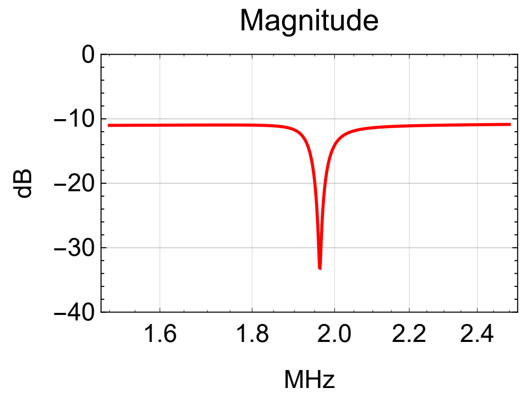
```
BodePlotEx[{-eom1TF[2 π ⅈ f 1*^6], -eom2TF[2 π ⅈ f 1*^6], -eomTF[2 π ⅈ f 1*^6]} //. allEom,
 {f, 0.1, 10}, MagnitudeRange → {-20, 0}, PhaseRange → {-150, 100},
 Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```

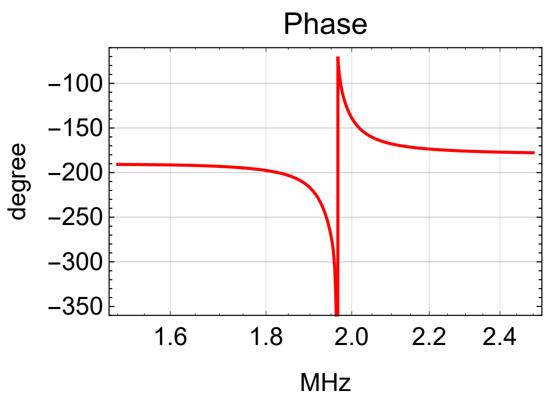


```
BodePlotEx[{-eom1TF[2 π ⅈ f 1*^6], -eom2TF[2 π ⅈ f 1*^6], -eomTF[2 π ⅈ f 1*^6]} //. allEom,
 {f, 0.1, 10}, MagnitudeRange → {-20, 0}, PhaseRange → {-150, 100},
 Evaluate[plotoptn[3]], XAxisLabel → "MHz"]
```



```
BodePlotEx[eomTF[2 π ⅈ f 1*^6] /. allEom, {f, 1.5, 2.5}, MagnitudeRange → {-40, 0},
 PhaseRange → {-360, -60}, Evaluate[plotoptn[1]], XAxisLabel → "MHz"]
```





Overall Transfer Functions

```
allTTFSS := Join[allSensing, prmCommon, allFast, allEom]
Options[ttfssTF] = Join[Options[sensingTF],
   Options[commonTF], Options[fastTF], Options[eomTF]];

ttfssCom[s_, opts___] := -sensingTF[s, opts] commonTF[s, opts]
ttfssFastSplit[s_, opts___] := fastTF[s, opts] pztTF[s] pztCoeff[s]
ttfssEomSplit[s_, opts___] := 10 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]
```

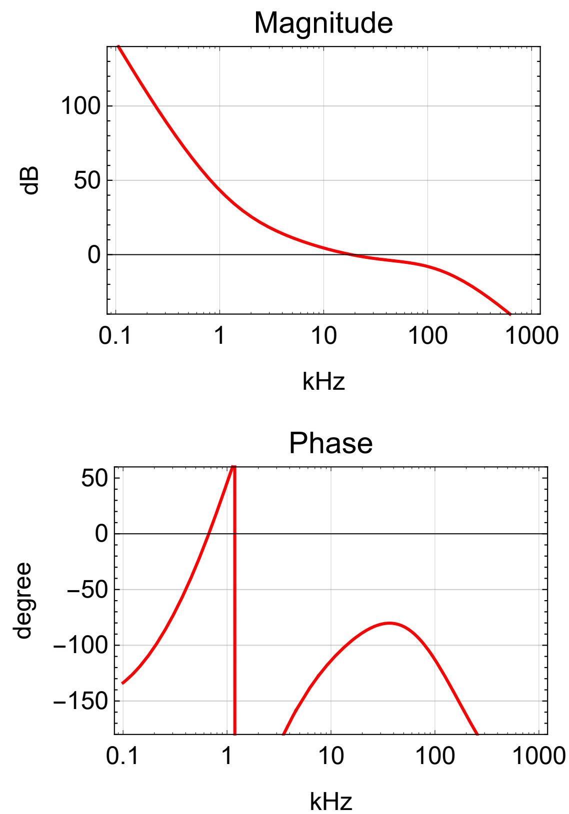
```
prmTTFSS = {FastOnly → False};
allTTFSS := Join[allSensing, prmCommon, allFast, allEom]
Options[ttfssTF] = Join[Options[sensingTF],
  Options[commonTF], Options[fastTF], Options[eomTF]];
ttfssTF[s_, opts___] :=
 ttfssFast[s, opts] + If[FastOnly /. {opts} /. allTTFSS, 0, ttfssEom[s, opts]]
```

```
Options[ttfssTF]
```

```
{Sensing → PFD, sBoost → False, cBoost1 → False, cBoost2 → False, FastOnly → False}
```

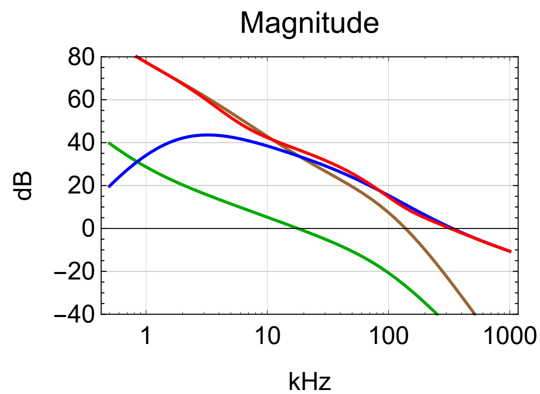
Crossover

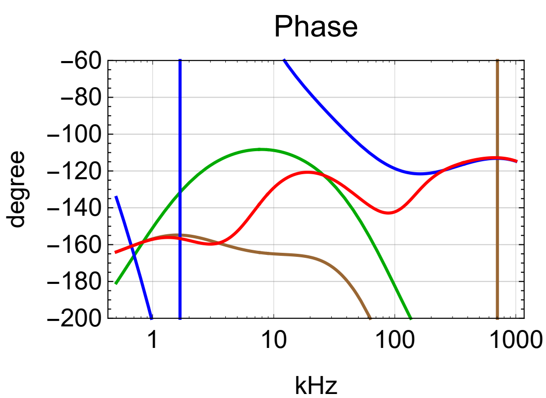
```
BodePlotEx[ttfssCrossTF[2 π ⅈ f 1*^3, FastOnly → False] /. fGain → 10^(20/20) /. allTTFSS,
 {f, 0.1, 1000}, MagnitudeRange → {-40, 140},
 PhaseRange → {-180, +60}, Evaluate[plotoptn[1]], XAxisLabel → "kHz"]
```



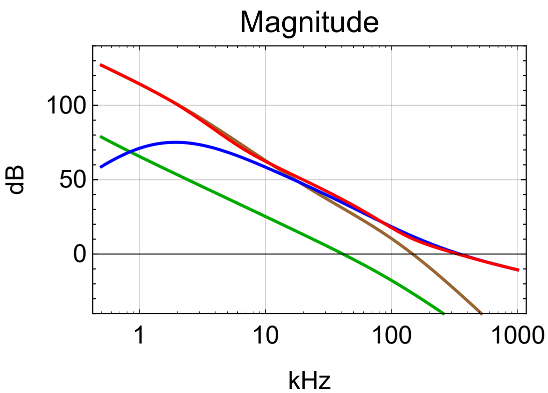
Components

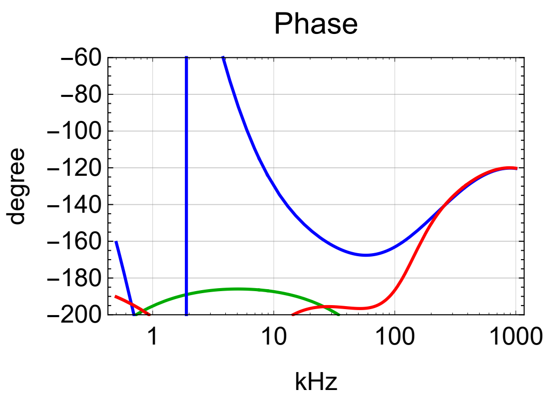
```
opt = Sequence[sBoost → True];
prm = {cGain → 10^(12/20), fGain → 10^(20/20)};
BodePlotEx[{ttfssFast[2 π ⅈ f 1*^3, FastOnly → False, opt] /. prm /. allTTFSS,
  ttfssFast[2 π ⅈ f 1*^3, FastOnly → True, opt] /. prm /. allTTFSS,
  ttfssEom[2 π ⅈ f 1*^3, FastOnly → False, opt] /. prm /. allTTFSS,
  ttfssTF[2 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];
prm = {cGain → 10^(12/20), fGain → 10^(20/20)};
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"]
```



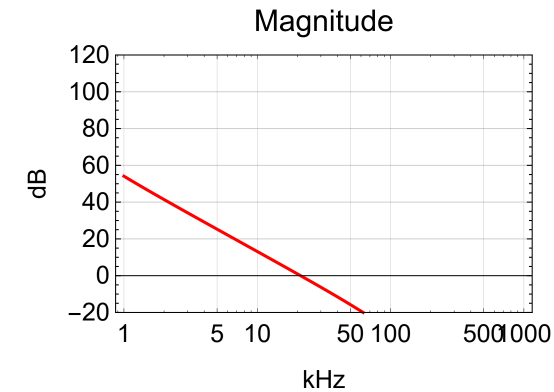


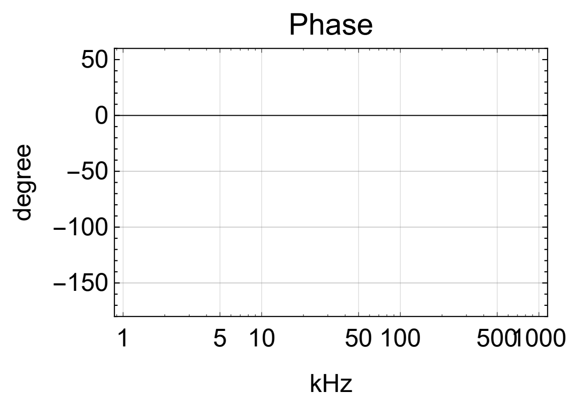
Combined Servo Path

Transfer Function

Bode Plot: PFD & Fast/EOM

```
BodePlotEx[ttfssTF[2 π ⅈ f 1*^3, FastOnly → True, sBoost → False] /. fGain → 10^(-10/20) /. allTTFSS,
 {f, 1, 1000}, MagnitudeRange → {-20, 120}, PhaseRange → {-180, +60},
 Evaluate[plotoptn[1]], XAxisLabel → "kHz"]
```





TTFFS Servo

Additive Offset