Common Mode Board Simulation

Simulation of the mode cleaner board and the common mode board.

Initialization

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

Needs["Graphics`Graphics`"]

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

plotopt = PlotStyle → {{Thickness [0.007], RGBColor [1, 0, 0]},
              {Thickness [0.007], RGBColor [0, 0, 1]},
              {Thickness [0.007], RGBColor [0.1, 0.7, 0.2]},
              {Thickness [0.007], RGBColor [0.5, 0.5, 0.2]}};

path = "c:/user/daniel/protel/CommonMode/MCbaseline/";
mcbaseline = ReadList[path <> "MC_BASELINE.TXT", {Number, Number}];
cmbaseline = ReadList[path <> "CM_BASELINE.TXT", {Number, Number}];
mcslewbaseline = ReadList[path <> "MCSLEW_BASELINE.TXT", {Number, Number}];
cmslewbaseline = ReadList[path <> "CMSLEW_BASELINE.TXT", {Number, Number}];
```

Parallel and Serial Impedance

```
par[r1_, r2_] := 1 / (1/r1 + 1/r2)
ser[r1_, r2_] := r1 + r2
```

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

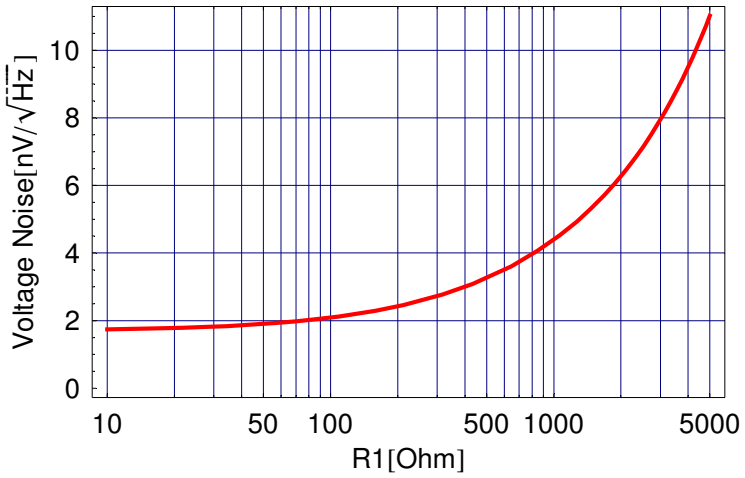
This function computes the equivalent input noise of an OpAmp circuit
g: +1 for non-inverting configuration or -1 for inverting configuration, 0 for differential configuration
z1: Impedance of input path (inverting) or impedance to ground (non-inverting) [Ohm]
z2: Impedance over feedback path [Ohm]
en: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

```
FourKT = 1.62*^-20;(* V^2/Hz/Ohm; room temperature 20 C*)
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]))]]
```

Examples (AD829)

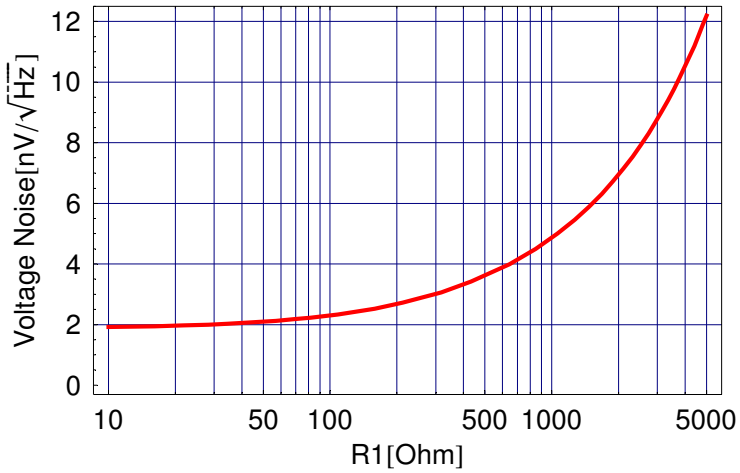
Non-Inverting configuration: input noise w/ gain of 10 as function of r1

```
LogLinearPlot[1*^9 OpAmpNoise[+1, r, 9 r, 1.7*^-9, 1.5*^-12],
  {r, 10, 5000}, FrameLabel → {"R1[Ohm]", "Voltage Noise[nV/Sqrt[Hz]]"},
  Frame → True, GridLines → Automatic, plotopt];
```



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

```
LogLinearPlot[1*^9 OpAmpNoise[-1, r, 10 r, 1.7*^-9, 1.5*^-12],
  {r, 10, 5000}, FrameLabel → {"R1[Ohm]", "Voltage Noise[nV/√Hz]"},
  Frame → True, GridLines → Automatic, plotopt];
```



Series Product of OpAmps

Computes the transfer function of several OpAmps circuits in series.

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

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

```
NoiseSum[prev_, {t_, n_}] := √(prev² + n²) Abs[t]
OpAmpNoiseProduct[t_, n_, m_] := Fold[NoiseSum, 0, Table[{t[i], n[i]}, {i, m}]] / Abs[OpAmpProduct[t, m]]
```

Spectrum Math

Propagate noise spectrum

```
SpecProp[prev_, t_] := {#[[1]], Abs[t /. s → 2. π #[[1]]] #[[2]]} & /@ prev
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];
  √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]
```

IFO Common Mode Transfer Functions & Noise

```
ugf = 20*^3;
```

■ First Stage: Differential Input Amplifier

```
n = 1;
z1[n] = 2000.;
z2[n] = par[2000, 1/(s 10*^-12)];
opamp[n] = OpAmp[0, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[0, z1[n], z2[n], 1.7*^-9, 1.5*^-12];

{dB[opamp[1]], Phase[opamp[1]]} /. s → 2 π i ugf
BodePlot[opamp[1] /. s → 2 π i 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt];
```

■ Second Stage: Gain Stage (+12dB)

```
n += 1;
z1[n] = Infinity;
z2[n] = 100.;
opamp[n] = 4 OpAmp[1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Third Stage: Summing Node

```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., 1/(s 10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Forth Stage: Pole/Zero Pair

```
n += 1;
z1[n] = 1210.;
z2[n] = par[121.*^3, ser[1210., 1/(s 33*^-9)]];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];

{dB[opamp[4]], Phase[opamp[4]]} /. s → 2 π i ugf
BodePlot[opamp[4] /. s → 2 π i 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt];

LogLogListPlot[Table[{10^i, 1*^9 opampnoise[4] /. s → 2 π i 1000 10^i}, {i, -2, 4, 0.01}],
  PlotJoined → True, FrameLabel → {"kHz", "Input Noise[nV/√Hz]"},
  Frame → True, PlotRange → {3, 10}, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```

■ Fifth Stage: Boosts

```
n += 1;
z1A[n] = Infinity;
z2A[n] = par[1580., 1/(s 100*^-9)];
z1B[n] = Infinity;
z2B[n] = par[1580., 1/(s 100*^-9)];
z1C[n] = Infinity;
z2C[n] = par[3160., 1/(s 100*^-9)];

opamp[n] = OpAmp[+1, z1A[n], z2A[n]] OpAmp[+1, z1B[n], z2B[n]] OpAmp[+1, z1C[n], z2C[n]];
opampnoise[n] = Sqrt[OpAmpNoise[+1, z1A[n], z2A[n], 1.7*^-9, 1.5*^-12]^2 +
    OpAmpNoise[+1, z1B[n], z2B[n], 1.7*^-9, 1.5*^-12]^2 +
    OpAmpNoise[+1, z1C[n], z2C[n], 1.7*^-9, 1.5*^-12]^2];

LogLogListPlot[Table[{10^i, 1*^9 opampnoise[5] /. s → 2 π i 1000 10^i}, {i, -2, 4, 0.01}],
  PlotJoined → True, FrameLabel → {"kHz", "Input Noise[nV/√Hz]"},
  Frame → True, PlotRange → All, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```

■ Sixth Stage: Exc1

```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., s 1/(10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Seventh Stage: Generic

```
n += 1;
z1[n] = Infinity;
z2[n] = 100;
opamp[n] = OpAmp[+1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[+1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Eighth Stage: Polarity

```
n += 1;
z1[n] = 3300;
z2[n] = 3300;
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Ninth Stage: Exc2

```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., s 1/(10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Tenth Stage: Differential Input Amplifier

```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000, 1/(s 10*^-12)];
opamp[n] = OpAmp[0, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[0, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Eleventh Stage: Gain Stage (8dB)

```
n += 1;
z1[n] = Infinity;
z2[n] = 100.;
opamp[n] = 2.5 OpAmp[1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Twelvth Stage: Low Pass

```
n += 1;
z1[n] = Infinity;
z2[n] = 100;
div[n] = 1600. / ser[1600, 1/(s 20*^-6)];
opamp[n] = div[n] OpAmp[+1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[+1, z1[n], z2[n], 1.7*^-9, 1.5*^-12] / Abs[div[n]];
```

■ Thirteenth Stage: Low Pass

```
n += 1;
z1[n] = Infinity;
z2[n] = 100;
div[n] = 1600. / ser[1600, 1/(s 20*^-6)];
opamp[n] = div[n] OpAmp[+1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[+1, z1[n], z2[n], 1.7*^-9, 1.5*^-13] / Abs[div[n]];
```

■ Fourteenth Stage: Limiter

```
n += 1;
z1[n] = Infinity;
z2[n] = 100;
opamp[n] = OpAmp[+1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[+1, z1[n], z2[n], 1.7*^-9, 1.5*^-13];
```

■ Fifteenth Stage: Output Driver

```
n += 1;
z1[n] = 3300.;
z2[n] = par[3300., 1/(s 10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-13];
```

IFO Common Mode Overall Transfer Functions & Noise

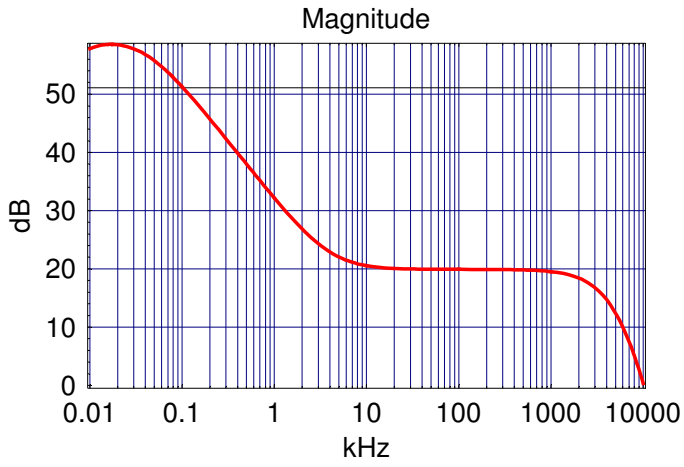
```
stages = n
opamp[0] = OpAmpProduct[opamp, stages];
opampnoise[0] = OpAmpNoiseProduct[opamp, opampnoise, stages];
```

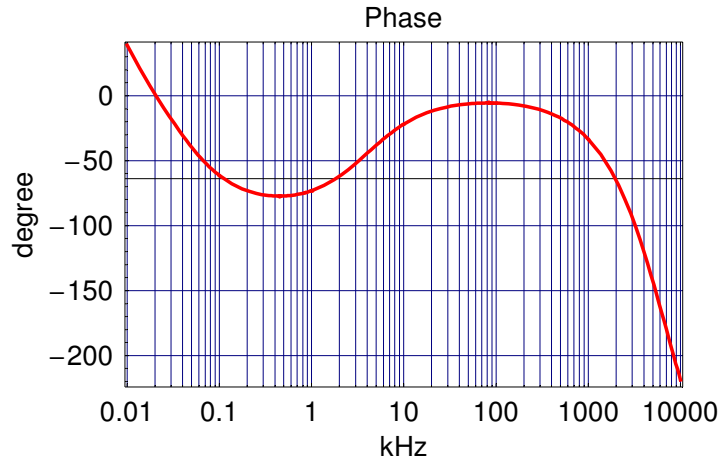
```
15
```

■ Transfer function

```
{dB[opamp[0]], Phase[opamp[0]]} /. s → 2 π i ugf
BodePlot[opamp[0] /. s → 2 π i 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt];
```

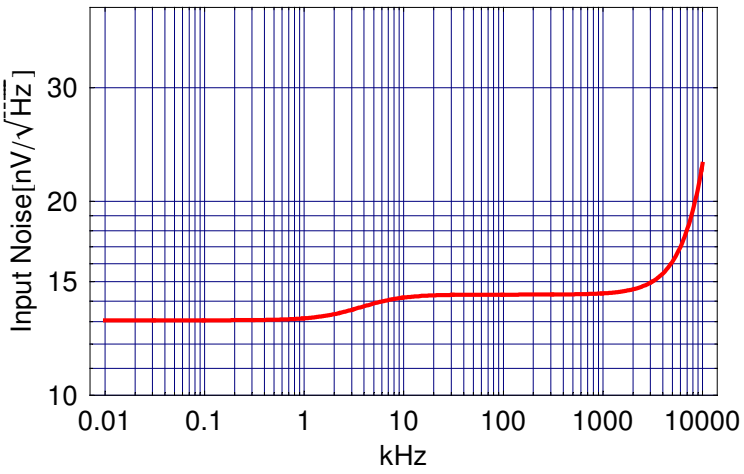
```
{20.0826, -11.799}
```





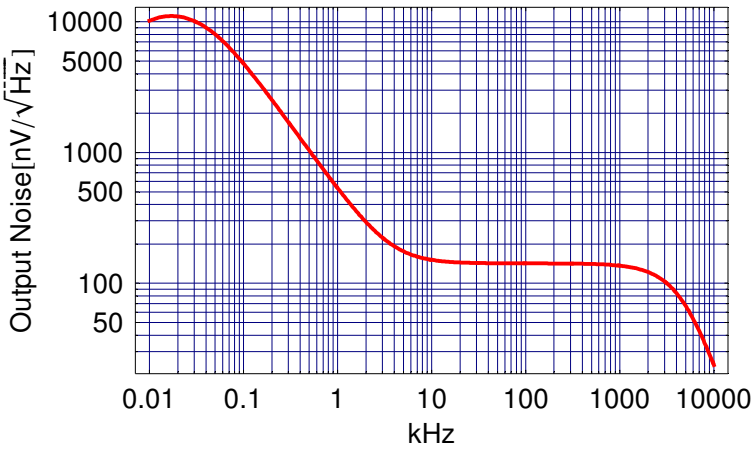
■ Equivalent Input Noise

```
LogLogListPlot[Table[{10^i, 1*^9 opampnoise[0] /. s → 2 π ⅈ 1000 10^i}, {i, -2, 4, 0.01}],
  PlotJoined → True, FrameLabel → {"kHz", "Input Noise[nV/√Hz]"},
  Frame → True, PlotRange → {9.999, 40}, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```



■ Output Noise w/ Input Terminated

```
LogLogListPlot[
  Table[{10^i, 1*^9 opampnoise[0] Abs[opamp[0]] /. s → 2 π i 1000 10^i}, {i, -2, 4, 0.01}],
  PlotJoined → True, FrameLabel → {"kHz", "Output Noise[nV/√Hz]"},
  Frame → True, PlotRange → All, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```



IFO Common Mode Noise Propagation and Slew Rate Limit

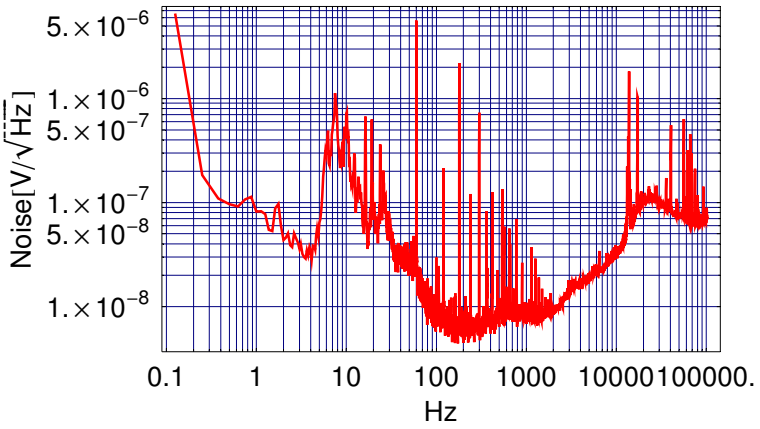
```
signal = SpecProp[cmbaseline, opamp, stages];
slew = SpecProp[cmslewbaseline, opamp, stages];

SpecRMS /@ (Drop[#, 7] & /@ signal)
SpecRMS /@ (Drop[#, 7] & /@ slew)
```

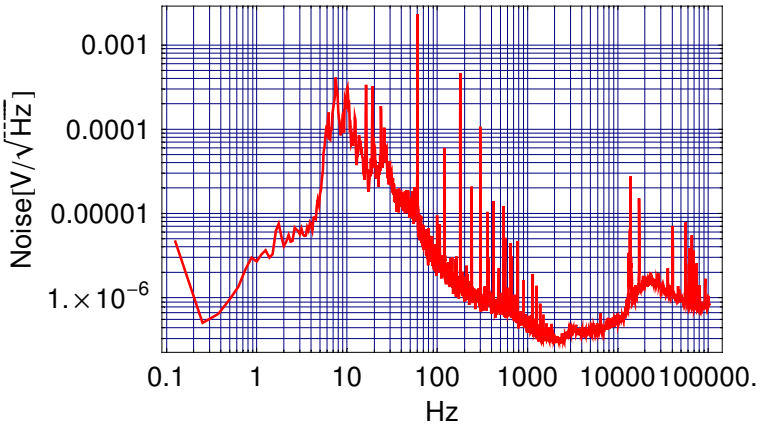
```
{0.0000329826, 0.0000328078, 0.000131231, 0.000130539, 0.000575068,
 0.000575068, 0.000575068, 0.000575068, 0.000575068, 0.000575068,
 0.000574882, 0.00172465, 0.00138597, 0.00119009, 0.00119009, 0.0011888}

{10.2981, 10.2101, 40.8405, 40.4923, 42.8589, 42.8589, 42.8589, 42.8589,
 42.8589, 42.8589, 42.5016, 127.505, 127.493, 127.481, 127.481, 125.742}
```

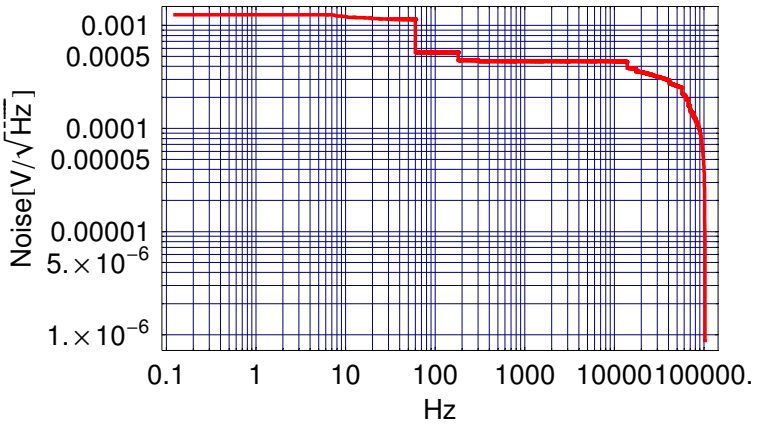
```
LogLogListPlot[signal[[1]], PlotJoined → True,
  FrameLabel → {"Hz", "Noise[V/√Hz]"}, Frame → True, PlotRange → All,
  GridLines → Automatic, PlotStyle → {Thickness[0.005], RGBColor[1, 0, 0]}];
```




```
LogLogListPlot[signal[[16]], PlotJoined → True,
  FrameLabel → {"Hz", "Noise[V/√Hz]"}, Frame → True, PlotRange → All,
  GridLines → Automatic, PlotStyle → {Thickness[0.005], RGBColor[1, 0, 0]}];
```



```
LogLogListPlot[RMSSpec[signal[[16]], -1], PlotJoined → True,
  FrameLabel → {"Hz", "Noise[V/√Hz ]"}, Frame → True, PlotRange → All,
  GridLines → Automatic, PlotStyle → {Thickness [0.007], RGBColor [1, 0, 0]}];
```



Mode Cleaner Board Transfer Functions & Noise

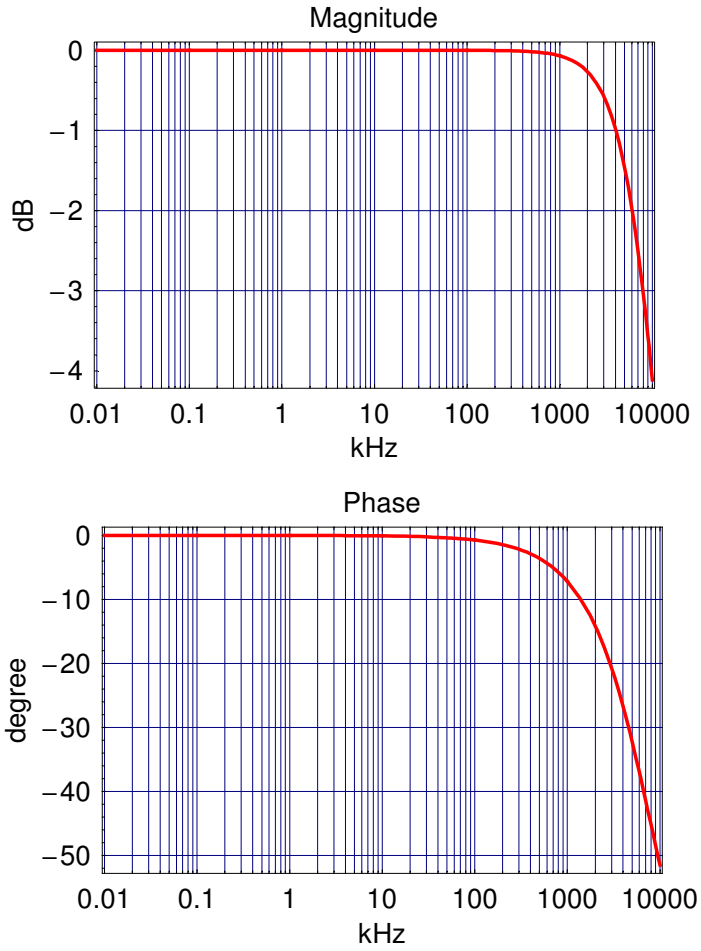
```
ugf = 100*^3;
```

■ First Stage: Differential Input Amplifier

```
n = 1;
z1[n] = 2000.;
z2[n] = par[2000, 1/(s 10*^-12)];
opamp[n] = OpAmp[0, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[0, z1[n], z2[n], 1.7*^-9, 1.5*^-12];

{dB[opamp[1]], Phase[opamp[1]]} /. s → 2 π i ugf
BodePlot[opamp[1] /. s → 2 π i 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt];
```

```
{-0.000685756, -0.719962}
```



■ Second Stage: Gain Stage (0dB)

```
n += 1;
z1[n] = Infinity;
z2[n] = 100.;
opamp[n] = OpAmp[1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Third Stage: Summing Node

```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., 1/(s 10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Forth Stage: Pole/Zero Pair

```
n += 1;
z1[n] = 1210.;
z2[n] = par[121.*^3, ser[1210., 1/(s 33*^-9)]];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];

{dB[opamp[4]], Phase[opamp[4]]} /. s → 2 π i ugf
BodePlot[opamp[4] /. s → 2 π i 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt];

LogLogListPlot[Table[{10^i, 1*^9 opampnoise[4] /. s → 2 π i 1000 10^i}, {i, -2, 4, 0.01}],
  PlotJoined → True, FrameLabel → {"kHz", "Input Noise[nV/√Hz]"},
  Frame → True, PlotRange → {3, 10}, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```

■ Fifth Stage: Boosts

```
n += 1;
z1A[n] = 82.5;
z2A[n] = par[1580., 1/(s 100*^-9)];
z1B[n] = 82.5;
z2B[n] = par[1580., 1/(s 100*^-9)];
z1C[n] = Infinity;
z2C[n] = par[3160., 1/(s 100*^-9)];

opamp[n] = OpAmp[+1, z1A[n], z2A[n]] OpAmp[+1, z1B[n], z2B[n]] OpAmp[+1, z1C[n], z2C[n]];
opampnoise[n] = Sqrt[OpAmpNoise[+1, z1A[n], z2A[n], 1.7*^-9, 1.5*^-12]^2 +
    OpAmpNoise[+1, z1B[n], z2B[n], 1.7*^-9, 1.5*^-12]^2 / Abs[OpAmp[+1, z1A[n], z2A[n]]]^2 +
    OpAmpNoise[+1, z1C[n], z2C[n], 1.7*^-9, 1.5*^-12]^2 / Abs[OpAmp[+1, z1A[n], z2A[n]] OpAmp[+1, z1B[n], z2B[n]]]^2];

LogLogListPlot[Table[{10^i, 1*^9 opampnoise[5] /. s → 2 π i 1000 10^i}, {i, -2, 4, 0.01}],
  PlotJoined → True, FrameLabel → {"kHz", "Input Noise[nV/√Hz]"},
  Frame → True, PlotRange → All, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```

■ Sixth Stage: Exc1

```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., s 1/(10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Seventh Stage: Generic

```
n += 1;
z1[n] = Infinity;
z2[n] = 100;
opamp[n] = OpAmp[+1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[+1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Eighth Stage: Polarity

```
n += 1;
z1[n] = 3300;
z2[n] = 3300;
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Ninth Stage: Exc2

```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., s 1/(10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Tenth Stage: Differential Input Amplifier

```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000, 1/(s 10*^-12)];
opamp[n] = OpAmp[0, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[0, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Eleventh Stage: Gain Stage

```
n += 1;
z1[n] = Infinity;
z2[n] = 100.;
opamp[n] = OpAmp[1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Twelvth Stage: Lead Compensation

```
n += 1;
z1[n] = par[1130., 1130. + 1/(s 1*^-9)];
z2[n] = par[1130, 1/(s 10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

■ Thirteenth Stage: Identity

```
n += 1;
z1[n] = Infinity;
z2[n] = 100;
opamp[n] = OpAmp[+1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[+1, z1[n], z2[n], 1.7*^-9, 1.5*^-13];
```

■ Fourteenth Stage: Limiter

```
n += 1;
z1[n] = Infinity;
z2[n] = 100;
opamp[n] = OpAmp[+1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[+1, z1[n], z2[n], 1.7*^-9, 1.5*^-13];
```

■ Fifteenth Stage: Output Driver

```
n += 1;
z1[n] = 3300.;
z2[n] = par[3300., 1/(s 10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-13];
```

Mode Cleaner Overall Transfer Functions & Noise

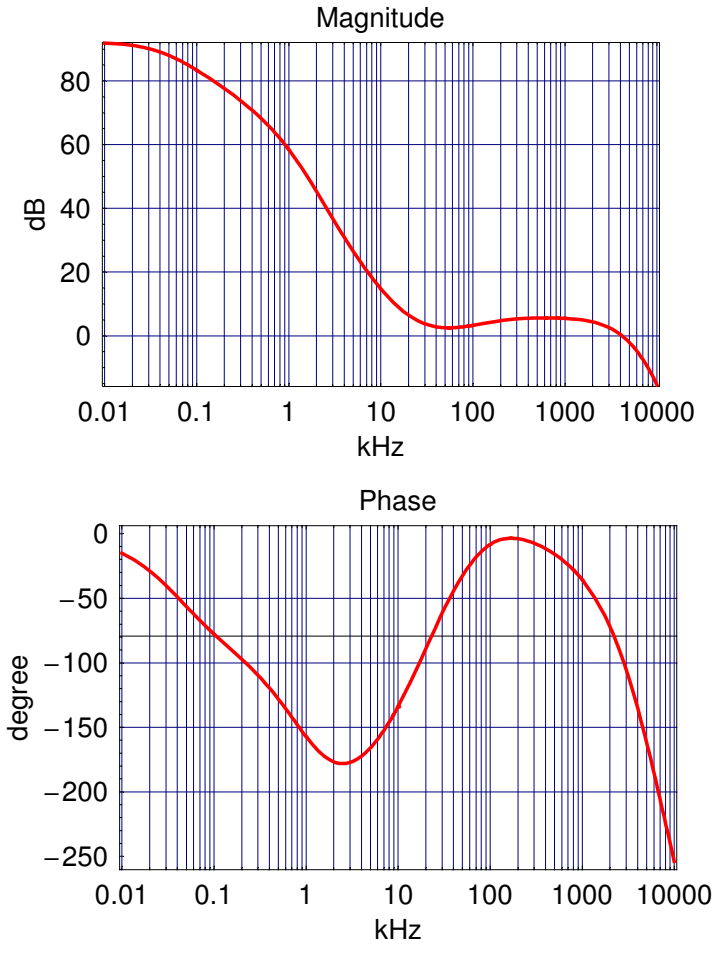
```
stages = n
opamp[0] = OpAmpProduct[opamp, stages];
opampnoise[0] = OpAmpNoiseProduct[opamp, opampnoise, stages];
```

```
15
```

■ Transfer function

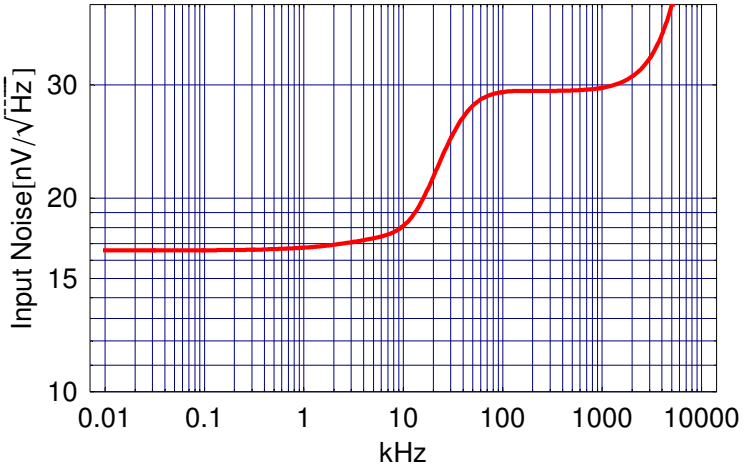
```
{dB[opamp[0]], Phase[opamp[0]]} /. s → 2 π i ugf
BodePlot[opamp[0] /. s → 2 π i 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt];
```

```
{3.28828, -8.33801}
```



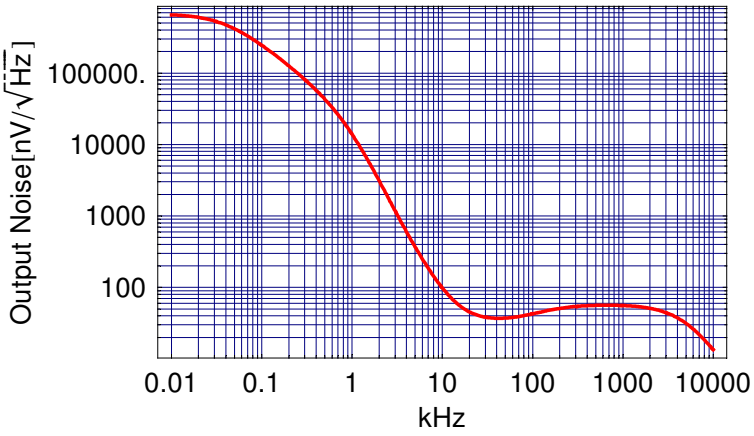
■ Equivalent Input Noise

```
LogLogListPlot[Table[{10^i, 1*^9 opampnoise[0] /. s → 2 π i 1000 10^i}, {i, -2, 4, 0.01}],
  PlotJoined → True, FrameLabel → {"kHz", "Input Noise[nV/√Hz]"},
  Frame → True, PlotRange → {9.999, 40}, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```



■ Output Noise w/ Input Terminated

```
LogLogListPlot[
  Table[{10^i, 1*^9 opampnoise[0] Abs[opamp[0]] /. s → 2 π i 1000 10^i}, {i, -2, 4, 0.01}],
  PlotJoined → True, FrameLabel → {"kHz", "Output Noise[nV/√Hz]"},
  Frame → True, PlotRange → All, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```



Mode Cleaner Noise Propagation and Slew Rate Limit

```
signal = SpecProp[mcbaseline, opamp, stages];
slew = SpecProp[mcslewbaseline, opamp, stages];

signal[[1, 2100]]
slew[[1, 2100]]
```

```
{1000, 1.22563×10^-6}
```

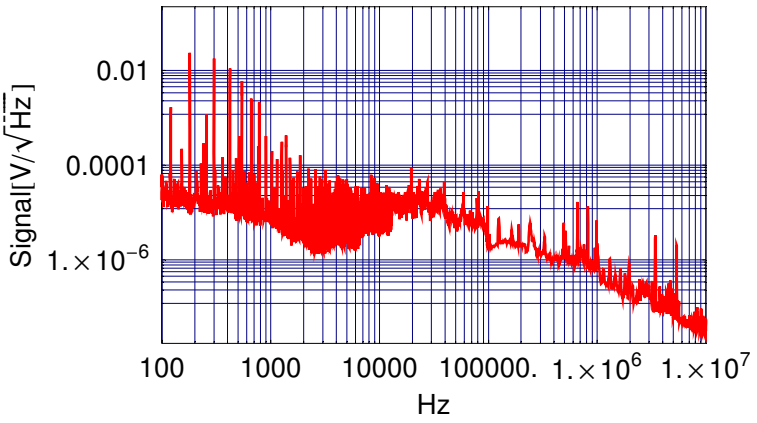
```
{1000, 0.00770086}
```

```
SpecRMS /@ (Drop[#, 2100] & /@ signal)
SpecRMS /@ (Drop[#, 2100] & /@ slew)
```

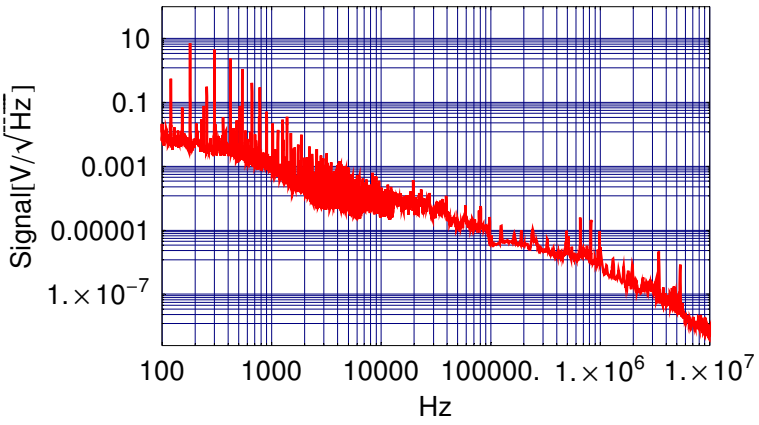
```
{0.00436962, 0.00407199, 0.00407199, 0.00389906,
 0.00448875, 0.0982917, 0.0982917, 0.0982917, 0.0982917, 0.0982917,
 0.0982654, 0.0982654, 0.099494, 0.099494, 0.099494, 0.0994429}
```

```
{51276.2, 32248.3, 32248.3, 21135.9, 20961.7, 21379.2, 21379.2, 21379.2,
 21379.2, 21379.2, 14839.1, 14839.1, 23225.5, 23225.5, 23225.5, 15395.8}
```

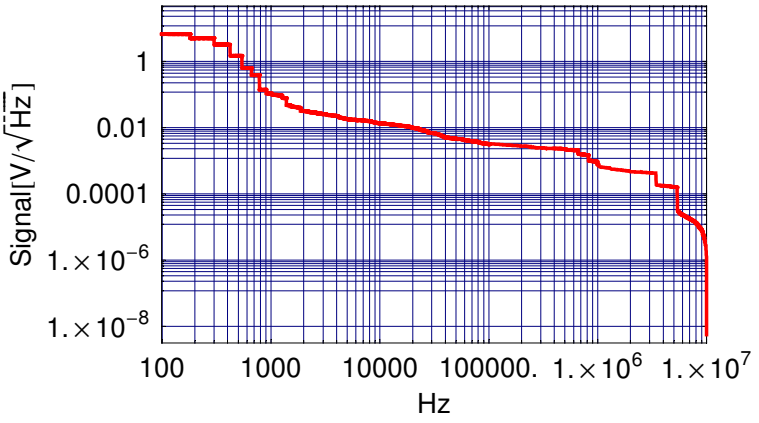
```
LogLogListPlot[signal[[5]], PlotJoined → True,
  FrameLabel → {"Hz", "Signal[V/√Hz]"}, Frame → True, PlotRange → {{99, 1*^7}, All},
  GridLines → Automatic, PlotStyle → {Thickness[0.005], RGBColor[1, 0, 0]}];
```



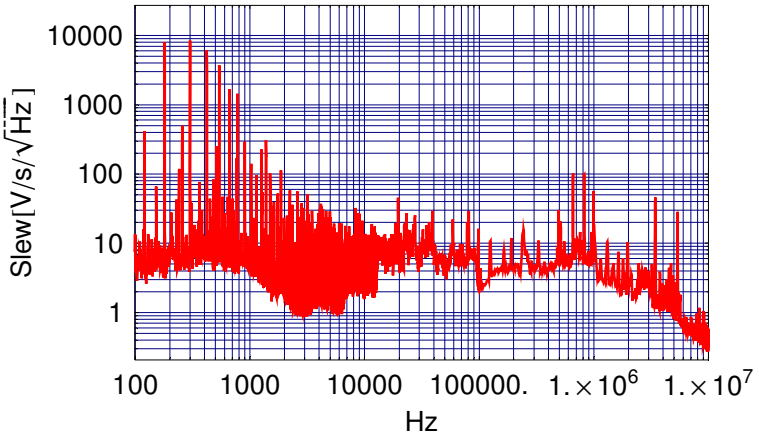
```
LogLogListPlot[signal[[16]], PlotJoined → True,
  FrameLabel → {"Hz", "Signal[V/√Hz]"}, Frame → True, PlotRange → {{99, 1*^7}, All},
  GridLines → Automatic, PlotStyle → {Thickness[0.005], RGBColor[1, 0, 0]}];
```




```
LogLogListPlot[RMSSpec[signal[[16]], -1], PlotJoined → True,
  FrameLabel → {"Hz", "Signal[V/√Hz]"}, Frame → True, PlotRange → {{99, 1*^7}, All},
  GridLines → Automatic, PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```



```
LogLogListPlot[slew[[16]], PlotJoined → True,
  FrameLabel → {"Hz", "Slew[V/s/√Hz]"}, Frame → True, PlotRange → {{99, 1*^7}, All},
  GridLines → Automatic, PlotStyle → {Thickness[0.005], RGBColor[1, 0, 0]}];
```




```
LogLogListPlot[RMSSpec[Drop[slew[[16]], 2100], 1],
  PlotJoined → True, FrameLabel → {"Hz", "Slew[V/s/√Hz]"}, Frame → True,
  PlotRange → {{999, 1*^7}, {1*^2, 1*^5}}, GridLines → Automatic,
  PlotStyle → {Thickness[0.007], RGBColor[1, 0, 0]}];
```