Common Mode Board Simulation

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

Setup

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

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

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

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

Free Running Laser Noise

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

Import

In[∘]:=
```
path = NotebookDirectory[] <> "/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}];
```

Equations

Parallel and Serial Impedance

In[∘]:=

```
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

In[∘]:=

```
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]

In[∘]:=

```
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]

In[ ]:=
```
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.

In[ ]:=
```
$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

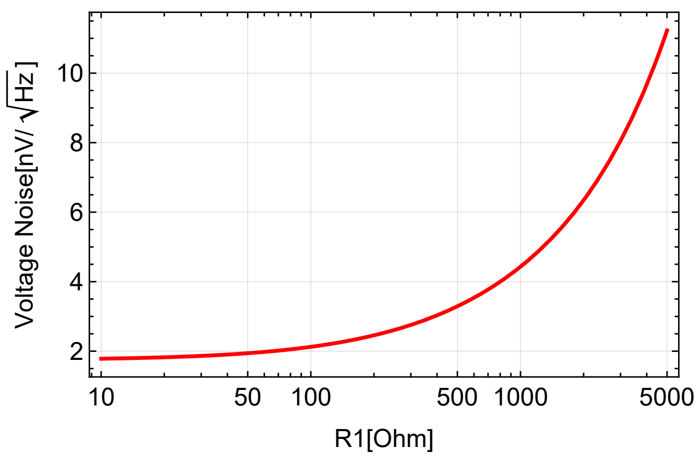
In[ ]:=
```
Clear[AD829, OP27]
AD829[f_] := {s → 2 π ⅈ f, en → enAD829, in → inAD829,
    enfloor → enfloorAD829, infloor → infloorAD829} //.
   {enAD829 → OpAmpNoiseFlicker[f, ekneeAD829, enfloorAD829],
    inAD829 → OpAmpNoiseFlicker[f, ikneeAD829, infloorAD829],
    ekneeAD829 → 50, ikneeAD829 → 100, (*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

```
In[ ]:= 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]]
```

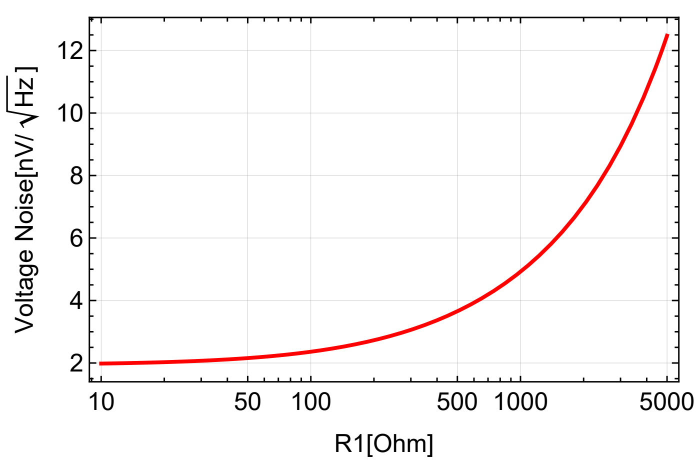
Out[ ]=



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

```
In[ ]:= 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]]
```

Out[ ]=



Series Product of OpAmps

Computes the transfer function of several OpAmps circuits in series.

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

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

In[ ]:=
```
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

In[ ]:=
```
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

In[ ]:=
```
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

In[ ]:=
```
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

```
In[∘]:= ugf = 20*^3;
```

First Stage: Differential Input Amplifier

```
In[∘]:= 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];
```

```
In[∘]:= {dB[opamp[1]], Phase[opamp[1]]} /. s → 2 π ⅈ ugf
BodePlotEx[opamp[1] /. s → 2 π ⅈ 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt];
Out[∘]=
{-0.0000274323, -0.144}
```

Second Stage: Gain Stage (+12dB)

```
In[∘]:= 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

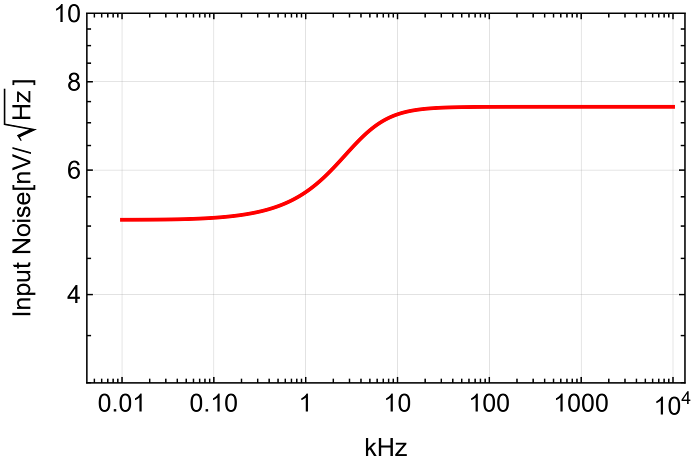
```
In[∘]:= n += 1;
z1[n] = 2000.;
z2[n] = par[2000., 1/(s 10*^-12)];
opamp[n] = OpAmp[-1, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[-1, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

Forth Stage: Pole/Zero Pair

```
In[∘]:= 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];
```

In[∘]:=
```
{dB[opamp[4]], Phase[opamp[4]]} /. s → 2 π ⅈ ugf
BodePlotEx[opamp[4] /. s → 2 π ⅈ 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt];
```
Out[∘]=
```
{0.0827091, -11.1579}
```

In[∘]:=
```
ListLogLogPlot[Table[{10^i, 1*^9 opampnoise[4] /. s → 2 π ⅈ 1000 × 10^i}, {i, -2, 4, 0.01}],
 Joined → True, FrameLabel → {"kHz", "Input Noise[nV/ √Hz ]"},
 PlotRange → {3, 10}, plotoptn[1]]
```
Out[∘]=



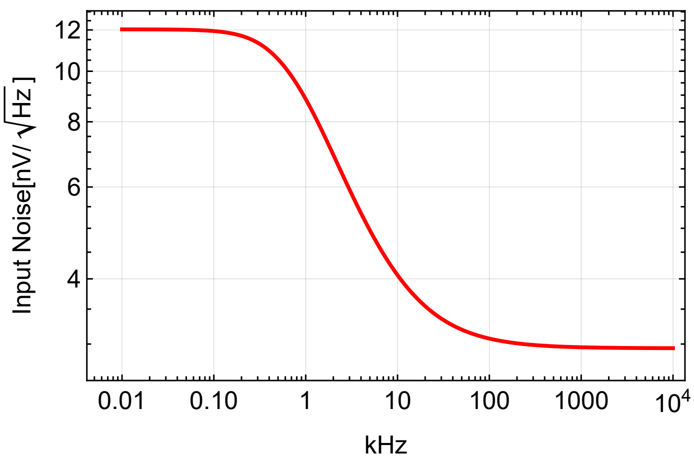
Fifth Stage: Boosts

In[∘]:=
```
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];
```

In[ ]:=
```
ListLogLogPlot[Table[{10^i, 1*^9 opampnoise[5] /. s → 2 π ⅈ 1000 × 10^i}, {i, -2, 4, 0.01}],
 Joined → True, FrameLabel → {"kHz", "Input Noise[nV/ √Hz ]"},
 PlotRange → All, plotoptn[1]]
```

Out[ ]=



Sixth Stage: Exc1

In[ ]:=
```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., s 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];
```

Seventh Stage: Generic

In[ ]:=
```
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

In[ ]:=
```
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

```
In[ ]:= n += 1;
z1[n] = 2000.;
z2[n] = par[2000., s 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];
```

Tenth Stage: Differential Input Amplifier

```
In[ ]:= 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)

```
In[ ]:= 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];
```

Twelfth Stage: Low Pass

```
In[ ]:= 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

In[∘]:=
```
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

In[∘]:=
```
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

In[∘]:=
```
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

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

Out[∘]=

```
15
```

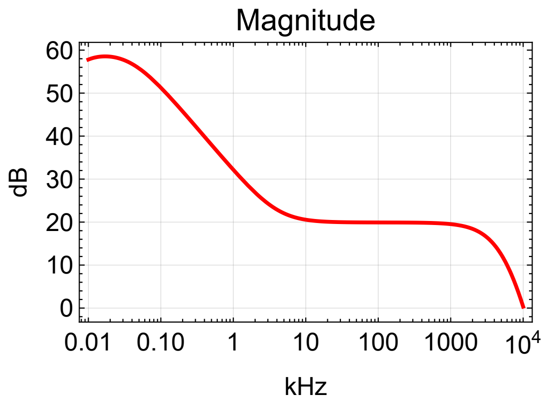
Plots

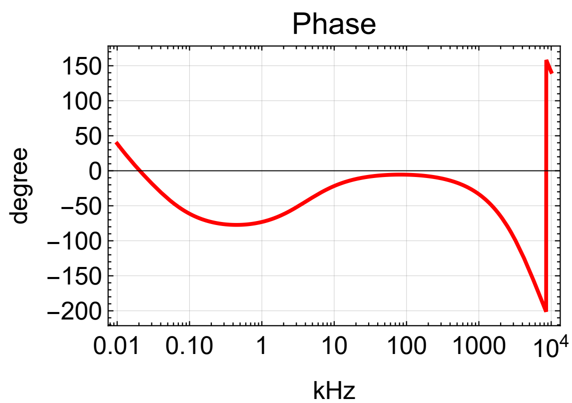
Transfer function

In[]:=
```
{dB[opamp[0]], Phase[opamp[0]]} /. s → 2 π i ugf
BodePlotEx[opamp[0] /. s → 2 π i 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotoptn[1]]
```

Out[]=
```
{20.0826, -11.799}
```

Out[]=

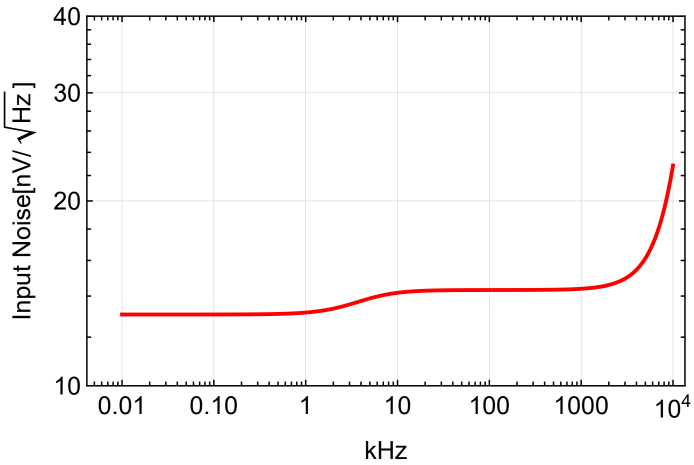




Equivalent Input Noise

In[•]:=
```
ListLogLogPlot[Table[{10^i, 1*^9 opampnoise[0] /. s → 2 π ⅈ 1000 × 10^i}, {i, -2, 4, 0.01}],
 Joined → True, FrameLabel → {"kHz", "Input Noise[nV/ √Hz ]"},
 PlotRange → {9.999, 40}, plotoptn[1]]
```

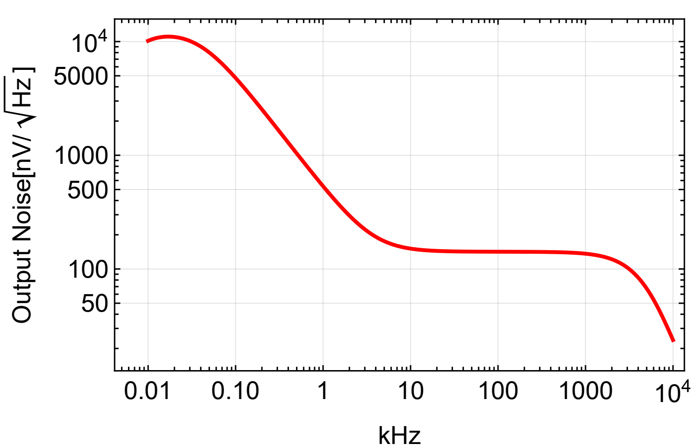
Out[•]=



Output Noise w/ Input Terminated

In[•]:=
```
ListLogLogPlot[
 Table[{10^i, 1*^9 opampnoise[0] Abs[opamp[0]] /. s → 2 π ⅈ 1000 × 10^i}, {i, -2, 4, 0.01}],
 Joined → True, FrameLabel → {"kHz", "Output Noise[nV/ √Hz ]"},
 PlotRange → All, plotoptn[1]]
```

Out[•]=



IFO Common Mode Noise Propagation and Slew Rate Limit

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

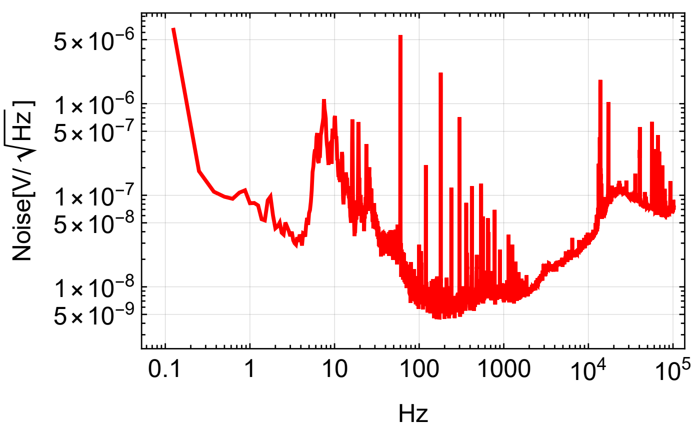
In[∘]:=
```
SpecRMS /@ (Drop[#, 7] & /@ signal)
SpecRMS /@ (Drop[#, 7] & /@ slew)
```

Out[∘]=
```
{0.0000329826, 0.0000328078, 0.000131231, 0.000130539, 0.000575068,
 0.000575068, 0.000575068, 0.000575068, 0.000575068, 0.000575068,
 0.000574882, 0.00143721, 0.00115498, 0.000991745, 0.000991745, 0.000990664}
```

Out[∘]=
```
{10.2981, 10.2101, 40.8405, 40.4923, 42.8589, 42.8589, 42.8589, 42.8589, 42.8589,
 42.8589, 42.8589, 42.5016, 106.254, 106.244, 106.234, 106.234, 104.785}
```

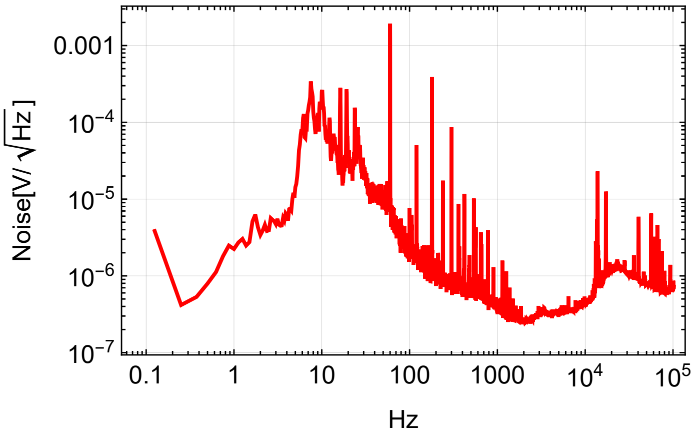
In[∘]:=
```
ListLogLogPlot[signal[[1]], Joined → True,
 FrameLabel → {"Hz", "Noise[V/ √Hz ]"}, PlotRange → All, plotoptn[1]]
```

Out[∘]=



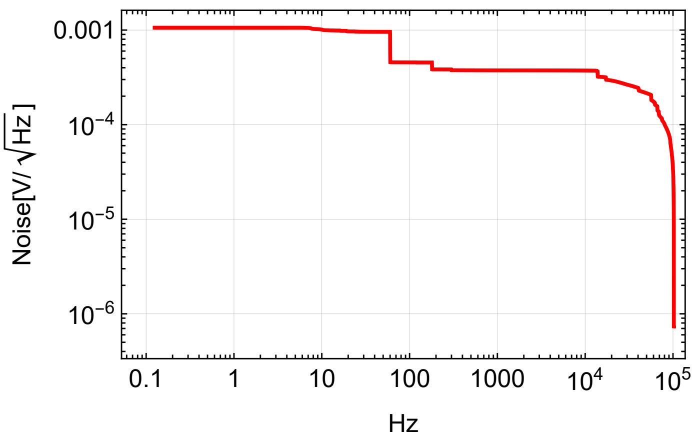
In[∘]:=
```
ListLogLogPlot[signal[[16]], Joined → True,
  FrameLabel → {"Hz", "Noise[V/√Hz]"}, PlotRange → All, plotoptn[1]]
```

Out[∘]=



In[∘]:=
```
ListLogLogPlot[RMSSpec[signal[[16]], -1], Joined → True,
  FrameLabel → {"Hz", "Noise[V/√Hz]"}, PlotRange → All, plotoptn[1]]
```

Out[∘]=



Mode Cleaner Board Transfer Functions & Noise

In[∘]:=
```
ugf = 100*^3;
```

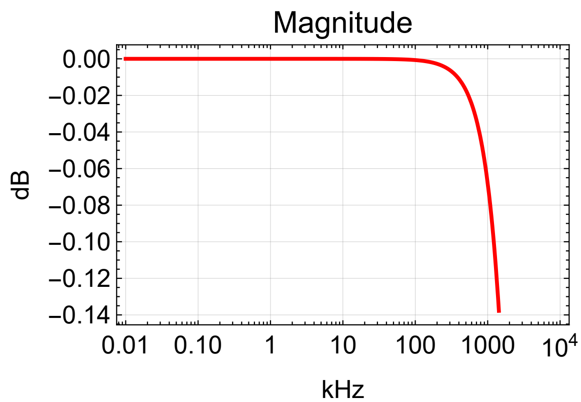
First Stage: Differential Input Amplifier

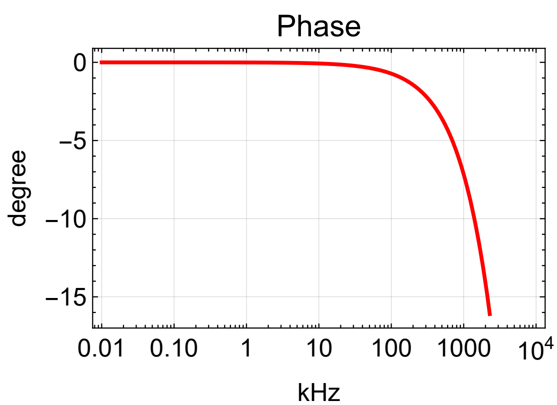
In[∘]:=
```
n = 1;
z1[n] = 2000.;
z2[n] = par[2000, 1/(s 10*^-12)];
opamp[n] = OpAmp[0, z1[n], z2[n]];
opampnoise[n] = OpAmpNoise[0, z1[n], z2[n], 1.7*^-9, 1.5*^-12];
```

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

Out[∘]=
```
{-0.000685756, -0.719962}
```

Out[∘]=





Second Stage: Gain Stage (0dB)

In[∘]:=
```
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

In[]:=
```
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];
```

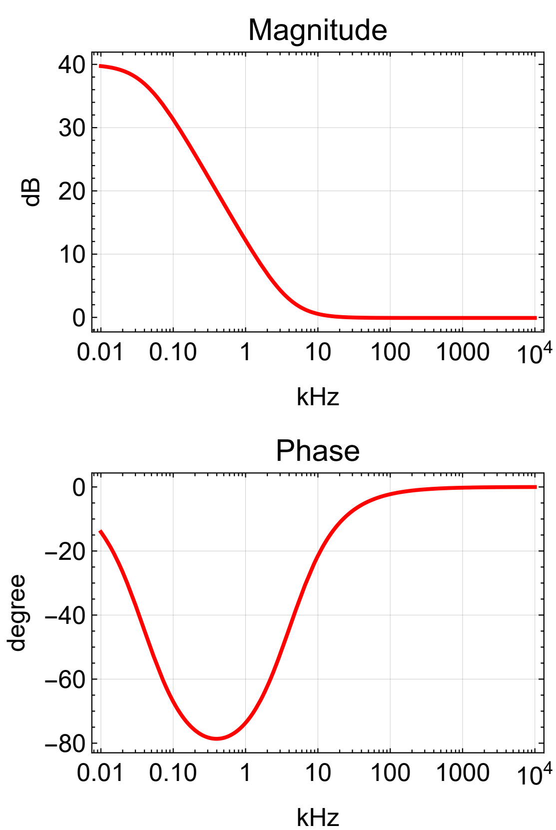
Forth Stage: Pole/Zero Pair

In[]:=
```
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];
```

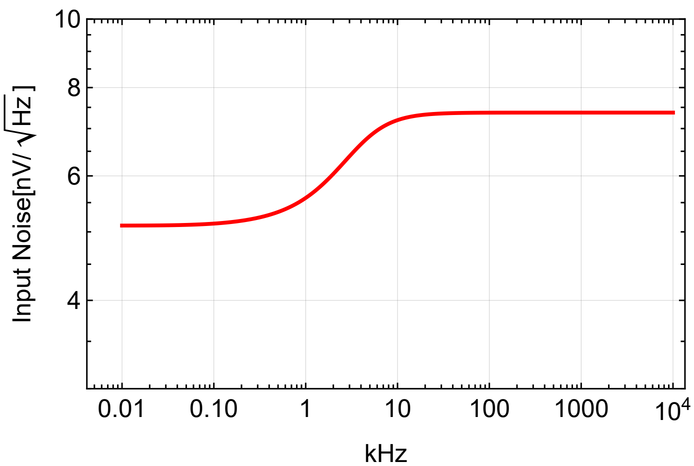
In[]:=
```
{dB[opamp[4]], Phase[opamp[4]]} /. s → 2 π ⅈ ugf
BodePlotEx[opamp[4] /. s → 2 π ⅈ 1000 f, {f, 0.01, 10*^3}, XAxisLabel → "kHz", plotopt]
```

Out[]=
```
{-0.079534, -2.2599}
```

Out[]=



In[◦]:=
```
ListLogLogPlot[Table[{10^i, 1*^9 opampnoise[4] /. s → 2 π ⅈ 1000 × 10^i}, {i, -2, 4, 0.01}],
 Joined → True, FrameLabel → {"kHz", "Input Noise[nV/ √Hz ]"},
 PlotRange → {3, 10}, plotoptn[1]]
```
Out[◦]=



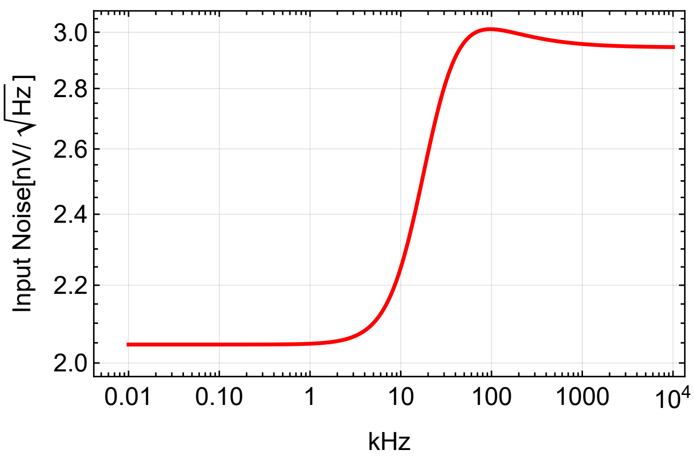
Fifth Stage: Boosts

In[◦]:=
```
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];
```

In[∘]:=
```
ListLogLogPlot[Table[{10^i, 1*^9 opampnoise[5] /. s → 2 π ⅈ 1000 × 10^i}, {i, -2, 4, 0.01}],
  Joined → True, FrameLabel → {"kHz", "Input Noise[nV/ √Hz ]"},
  PlotRange → All, plotoptn[1]]
```

Out[∘]=



Sixth Stage: Exc1

In[∘]:=
```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., s 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];
```

Seventh Stage: Generic

In[∘]:=
```
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

In[∘]:=
```
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

In[∘]:=
```
n += 1;
z1[n] = 2000.;
z2[n] = par[2000., s 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];
```

Tenth Stage: Differential Input Amplifier

In[∘]:=
```
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

In[∘]:=
```
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];
```

Twelfth Stage: Lead Compensation

In[∘]:=
```
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

In[∘]:=
```
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

In[ ]:=
```
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

In[ ]:=
```
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

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

Out[ ]=
```
15
```

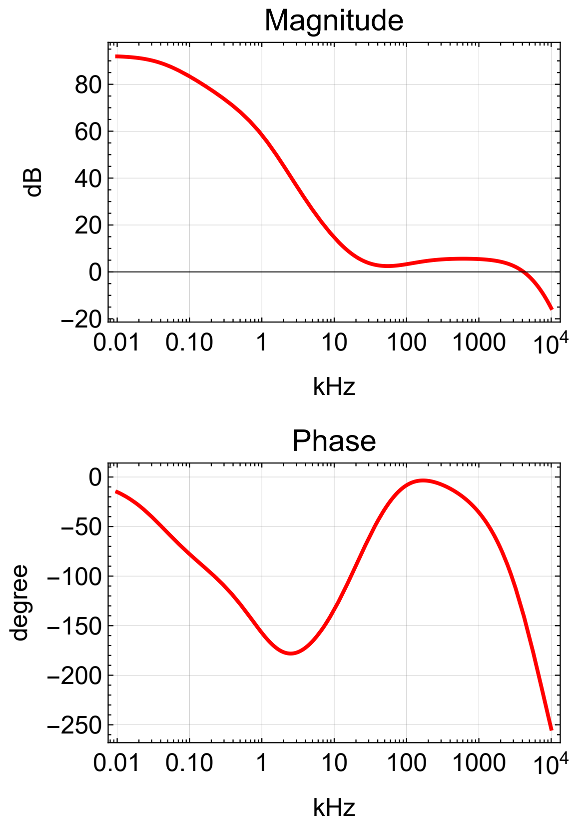
Plots

Transfer function

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

Out[∘]=
```
{3.28828, -8.33801}
```

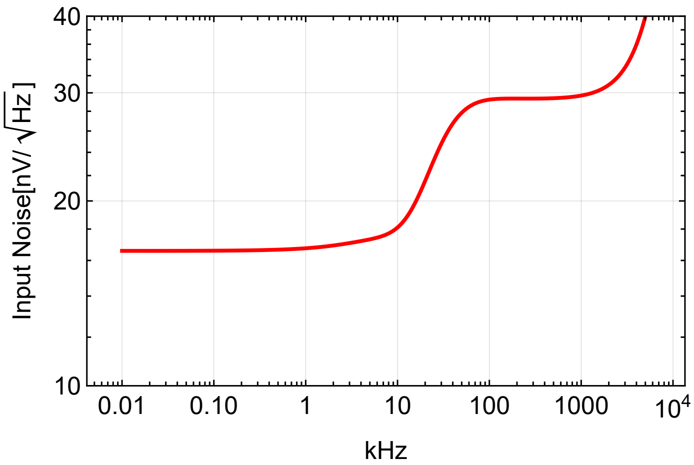
Out[∘]=



Equivalent Input Noise

In[ ]:=
```
ListLogLogPlot[Table[{10^i, 1*^9 opampnoise[0] /. s → 2 π i 1000 × 10^i}, {i, -2, 4, 0.01}],
 Joined → True, FrameLabel → {"kHz", "Input Noise[nV/ √Hz ]"},
 PlotRange → {9.999, 40}, plotoptn[1]]
```

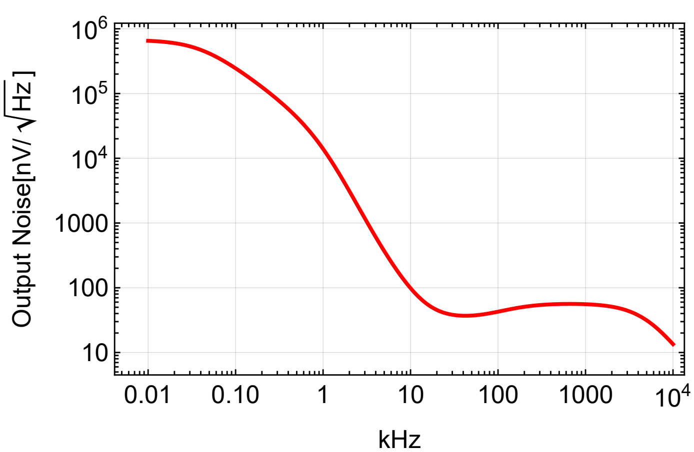
Out[ ]=



Output Noise w/ Input Terminated

In[ ]:=
```
ListLogLogPlot[
 Table[{10^i, 1*^9 opampnoise[0] Abs[opamp[0]] /. s → 2 π i 1000 × 10^i}, {i, -2, 4, 0.01}],
 Joined → True, FrameLabel → {"kHz", "Output Noise[nV/ √Hz ]"},
 PlotRange → All, plotoptn[1]]
```

Out[ ]=



Mode Cleaner Noise Propagation and Slew Rate Limit

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

In[•]:=
```
signal⟦1, 2100⟧
slew⟦1, 2100⟧
```

Out[•]=

$\{1000, 1.22563 \times 10^{-6}\}$

Out[•]=

{1000, 0.00770086}

In[•]:=
```
SpecRMS /@ (Drop[#, 2100] & /@ signal)
SpecRMS /@ (Drop[#, 2100] & /@ slew)
```

Out[•]=

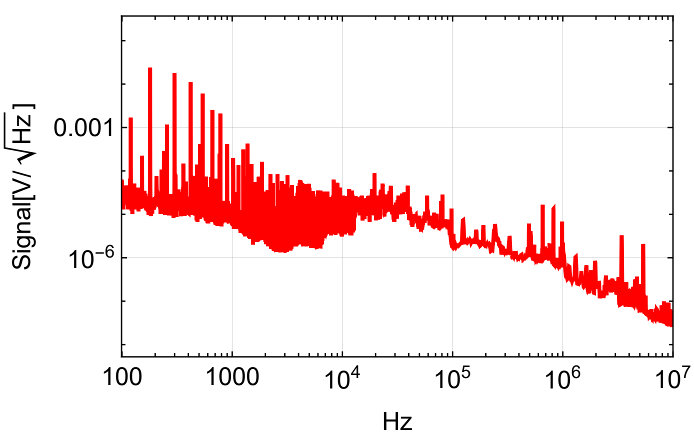
{0.00436962, 0.00407199, 0.00407199, 0.00389906,
0.00448875, 0.0982917, 0.0982917, 0.0982917, 0.0982917, 0.0982917,
0.0982654, 0.0982654, 0.099494, 0.099494, 0.099494, 0.0994429}

Out[•]=

{51 276.2, 32 248.3, 32 248.3, 21 135.9, 20 961.7, 21 379.2, 21 379.2, 21 379.2,
21 379.2, 21 379.2, 14 839.1, 14 839.1, 23 225.5, 23 225.5, 23 225.5, 15 395.8}

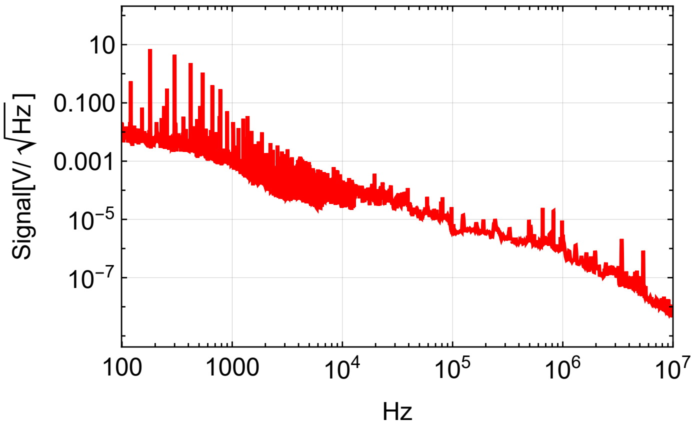
In[•]:=
```
ListLogLogPlot[signal⟦5⟧, Joined → True,
 FrameLabel → {"Hz", "Signal[V/ √Hz ]"}, PlotRange → {{99, 1*^7}, All}, plotoptn[1]]
```

Out[•]=



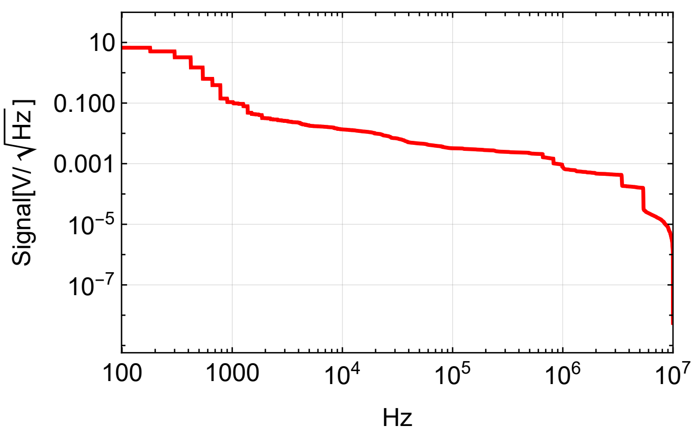
In[*]:=
```
ListLogLogPlot[signal[[16]], Joined → True,
  FrameLabel → {"Hz", "Signal[V/ √Hz ]"}, PlotRange → {{99, 1*^7}, All}, plotoptn[1]]
```

Out[*]=



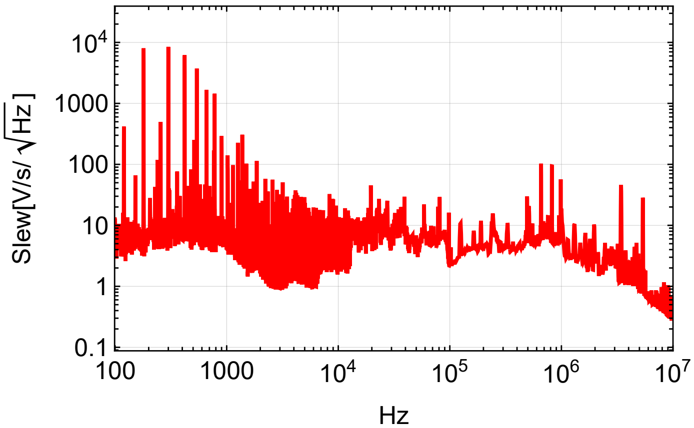
In[*]:=
```
ListLogLogPlot[RMSSpec[signal[[16]], -1], Joined → True,
  FrameLabel → {"Hz", "Signal[V/ √Hz ]"}, PlotRange → {{99, 1*^7}, All}, plotoptn[1]]
```

Out[*]=



In[◦]:=
```
ListLogLogPlot[slew〚16〛, Joined → True,
  FrameLabel → {"Hz", "Slew[V/s/ √Hz ]"}, PlotRange → {{99, 1*^7}, All}, plotoptn[1]]
```

Out[◦]=



In[◦]:=
```
ListLogLogPlot[RMSSpec[Drop[slew〚16〛, 2100], 1],
  Joined → True, FrameLabel → {"Hz", "Slew[V/s/ √Hz ]"},
  PlotRange → {{999, 1*^7}, {1*^2, 1*^5}}, plotoptn[1]]
```

Out[◦]=