Fast Digital Servo:
DAC Converter Module
T2400200-v1

Initialization

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

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

In[ ]:=
```
plotopt = Sequence @@ {GridLines → Automatic, Frame → True,
    FrameStyle → Thickness[0.0025], BaseStyle → {FontSize → 12}};
```

In[ ]:=
```
plotoptn[n_Integer ? (# > 0 ∧ # < 9 &)] :=
  Sequence @@ {GridLines → Automatic, Frame → True, FrameStyle → Thickness[0.0025],
    PlotStyle → Take[{Black, Gray, Darker[Yellow], Purple, Brown,
       Darker[Green], Blue, Red}, -n], BaseStyle → {FontSize → 12}};
plotoptn[n_Integer ? (# ≤ 0 ∨ # ≥ 9 &)] := plotopt
```

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

Parallel and Serial Impedance

In[•]:=
$$\text{par[r1\_, r2\_] := } \frac{1}{\frac{1}{r1} + \frac{1}{r2}}$$
$$\text{par[r1\_, r2\_, r3\_] := } \frac{1}{\frac{1}{r1} + \frac{1}{r2} + \frac{1}{r3}}$$
$$\text{par[r1\_, r2\_, r3\_, r4\_\_] := } \frac{1}{\frac{1}{r1} + \frac{1}{r2} + \frac{1}{r3} + \text{Plus @@ } \left(\frac{1}{\text{List[r4]}}\right)}$$

```
ser[r1_, r2_] := r1 + r2
ser[r1_, r2_, r3_] := r1 + r2 + r3
ser[r1_, r2_, r3_, r4__] := r1 + r2 + r3 + Plus @@ List[r4]
```

In[•]:=
```
fourkt = {FourKT → 1.62*^-20}; (* V^2/Hz/Ohm; room temperature 20C *)
```

Pole/Zero

In[•]:=
$$\text{pole[f\_, p\_] := } \frac{1}{1 + i\, f / p}$$
$$\text{zero[f\_, p\_] := } 1 + i\, f / p$$
$$\text{pole[f\_, p\_, Q\_] := } \frac{1}{1 + i \frac{1}{Q} \frac{f}{p} - (f / p)^2}$$
$$\text{zero[f\_, p\_, Q\_] := } 1 + i \frac{1}{Q} \frac{f}{p} - (f / p)^2$$

Parallel Capacitors

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[•]:=
$$\text{OpAmp[g\_, z1\_, z2\_] :=}$$
$$\text{Which}\left[g > 0,\ 1 + \frac{z2}{z1},\ g < 0,\ -\frac{z2}{z1},\ \text{True},\ \frac{z2}{z1}\right]$$

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[∘]:=
```
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]
```

Series Product of OpAmps

Computes the transfer function of several OpAmps circuits in series.

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

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]]
OpAmpNoiseProductOut[t_, n_, m_] := Fold[NoiseSum, 0, Table[{t[i], n[i]}, {i, m}]]
```

Spectrum Math

Propagate noise spectrum

```
In[∘]:= 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

```
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]
```

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};
LT1007[f_] := {s → 2 π ⅈ f, en → enLT1007, in → inLT1007,
     enfloor → enfloorLT1007, infloor → infloorLT1007} //.
    {enLT1007 → OpAmpNoiseFlicker[f, ekneeLT1007, enfloorLT1007],
     inLT1007 → OpAmpNoiseFlicker[f, ikneeLT1007, infloorLT1007],
     ekneeLT1007 → 2.0, ikneeLT1007 → 120,
     enfloorLT1007 → 2.5*^-9, infloorLT1007 → 0.4*^-12};
LT1037[f_] := {s → 2 π ⅈ f, en → enLT1037, in → inLT1037,
     enfloor → enfloorLT1037, infloor → infloorLT1037} //.
    {enLT1037 → OpAmpNoiseFlicker[f, ekneeLT1037, enfloorLT1037],
     inLT1037 → OpAmpNoiseFlicker[f, ikneeLT1037, infloorLT1037],
     ekneeLT1037 → 2.0, ikneeLT1037 → 120,
     enfloorLT1037 → 2.5*^-9, infloorLT1037 → 0.4*^-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};
LT1792[f_] := {s → 2 π ⅈ f, en → enLT1792, in → inLT1792,
     enfloor → enfloorLT1792, infloor → infloorLT1792} //.
    {enLT1792 → OpAmpNoiseFlicker[f, ekneeLT1792, enfloorLT1792],
     inLT1792 → OpAmpNoiseFlicker[f, ikneeLT1792, infloorLT1792],
     ekneeLT1792 → 30, ikneeLT1792 → 0, (*guess*)
     enfloorLT1792 → 4.2*^-9, infloorLT1792 → 10*^-15};
ADA4625[f_] := {s → 2 π ⅈ f, en → enADA4625, in → inADA4625,
     enfloor → enfloorADA4625, infloor → infloorADA4625} //.
    {enADA4625 → OpAmpNoiseFlicker[f, ekneeADA4625, enfloorADA4625],
     inADA4625 → OpAmpNoiseFlicker[f, ikneeADA4625, infloorADA4625],
     ekneeADA4625 → 15, ikneeADA4625 → 0, (*guess*)
     enfloorADA4625 → 3.3*^-9, infloorADA4625 → 4.5*^-15};
ADA4522[f_] := {s → 2 π ⅈ f, en → enADA4522, in → inADA4522,
```

```
     enfloor → enfloorADA4522, infloor → infloorADA4522} //.
    {enADA4522 → OpAmpNoiseFlicker[f, ekneeADA4522, enfloorADA4522],
     inADA4522 → OpAmpNoiseFlicker[f, ikneeADA4522, infloorADA4522],
     ekneeADA4522 → 0, ikneeADA4522 → 0, (*chopper*)
     enfloorADA4522 → 7*^-9, infloorADA4522 → 1.2*^-12};
ADA4898[f_] := {s → 2 π ⅈ f, en → enADA4898, in → inADA4898,
     enfloor → enfloorADA4898, infloor → infloorADA4898} //.
    {enADA4898 → OpAmpNoiseFlicker[f, ekneeADA4898, enfloorADA4898],
     inADA4898 → OpAmpNoiseFlicker[f, ikneeADA4898, infloorADA4898],
     ekneeADA4898 → 10, ikneeADA4898 → 20,
     enfloorADA4898 → 0.9*^-9, infloorADA4898 → 2.4*^-12};
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*)};
```

ADA4898

DAC Whitening and Gain

Parameters

```
dacdewhite = {w1 → False, w2 → False};
(*ADA4898[f] or AD829[f]*)
```

DAC (AD3552R) & Driver (AD8065)

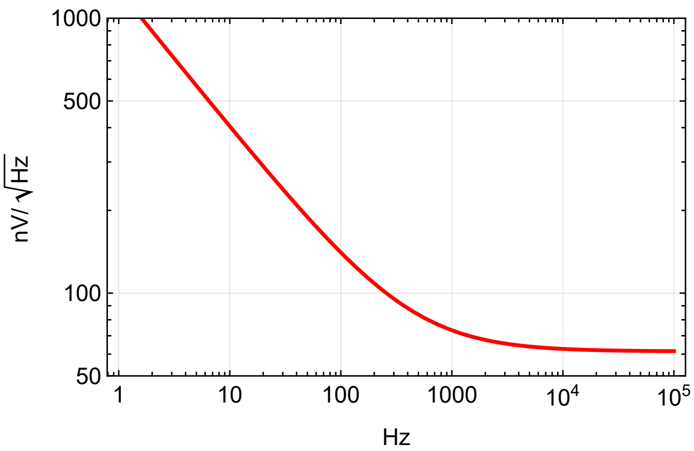
Assume this is all part of the DAC output noise which is specified at ~95dB THD for ±10V range.

In[ ]:= $\text{prmAD3552R} = \left\{\text{FS} \to 10,\ \text{THD} \to 10^{\frac{95}{20}},\ \text{BW} \to 2^{22},\ \text{flicker10} \to 400{*}\text{^-9}\right\};$

In[ ]:= $\text{dacnoise}[\text{f\_}] := \sqrt{\frac{\text{flicker10}^2}{\text{f}/10} + \left(\frac{\text{FS}}{\sqrt{2}\ \text{THD}}\ \frac{1}{\sqrt{\text{BW}}}\right)^2}\ /.\ \text{prmAD3552R}\ (*\ \text{nV}/\sqrt{\text{Hz}}\ *)$

In[∘]:=
```
LogLogPlot[1*^9 dacnoise[f], {f, 1, 100*^3}, PlotRange → {50, 1000},
 FrameLabel → {"Hz", "nV/ √Hz "}, Evaluate[plotoptn[1]]]
```

Out[∘]=



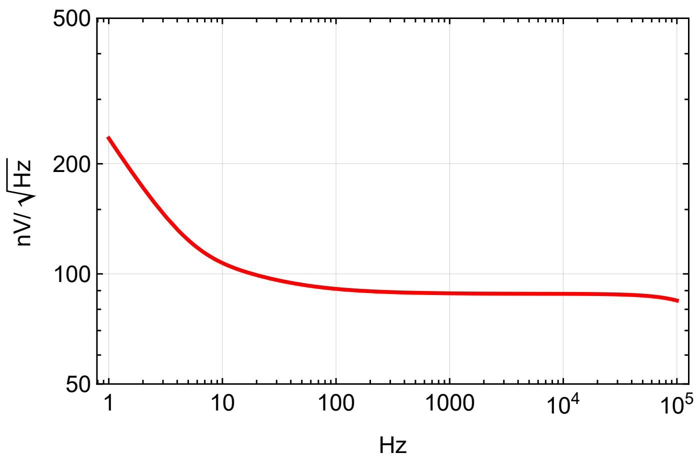
Snubber

First Stage: Dewhitening

In[∘]:=
```
z1A[1] := 1580 + 1/(s 10*^-6)

z2A[1] := par[15 000, 4990 + 1/(s 24.7*^-12)]

z1B[1] := ∞
z2B[1] := 50

opamp[1] := If[w1, z1A[1]/(z1A[1] + z2A[1]) OpAmp[1, z1B[1], z2B[1]], 1]

opampnoise[1] := If[w1, Abs[(z1A[1] + z2A[1])/z1A[1]]
   √(OpAmpNoise[1, z1A[1], z2A[1], en, in]^2 + FourKT Abs[z3] + (in Abs[z3])^2) /.
  z3 → par[z1A[1], z2A[1]], 0]
```

In[∘]:=
```
LogLogPlot[1*^9 opampnoise[1] /. w1 → True //. ADA4898[f] /. fourkt, {f, 1, 100*^3},
  PlotRange → {50, 500}, FrameLabel → {"Hz", "nV/ √Hz "}, Evaluate[plotoptn[1]]]
```

Out[∘]=



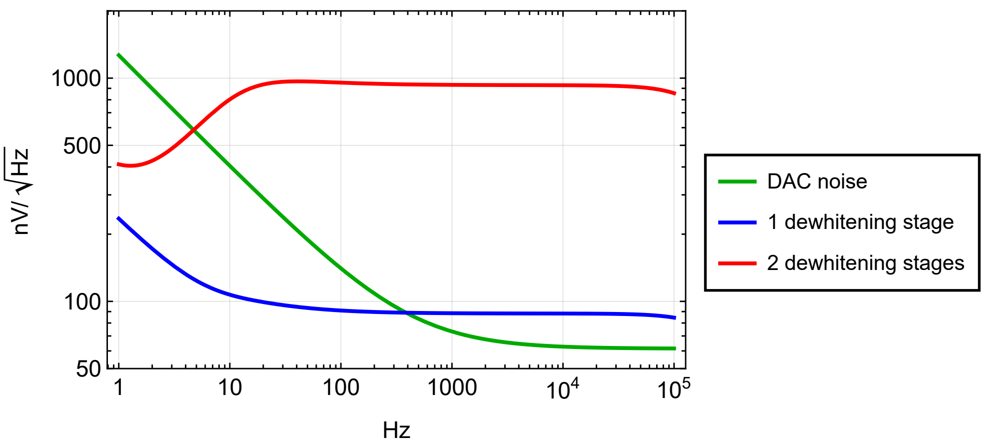
Second Stage: Dewhitening

In[∘]:=
```
z1A[2] := 1580 + 1/(s 10*^-6)
z2A[2] := par[15000, 4990 + 1/(s 24.7*^-12)]
z1B[2] := ∞
z2B[2] := 50
opamp[2] := If[w2, z1A[2]/(z1A[2] + z2A[2]) OpAmp[1, z1B[2], z2B[2]], 1]
opampnoise[2] := If[w2, Abs[(z1A[2] + z2A[2])/z1A[2]]
    √(OpAmpNoise[1, z1A[2], z2A[2], en, in]^2 + FourKT Abs[z3] + (in Abs[z3])^2) /.
   z3 → par[z1A[2], z2A[2]], 0]
```

Input Referred Noise

In[ ]:=
```
LogLogPlot[
 Evaluate[1*^9 {dacnoise[f], opampnoise[1], OpAmpNoiseProduct[opamp, opampnoise, 2]} /.
    {w1 → True, w2 → True} //. ADA4898[f] /. fourkt], {f, 1, 100*^3},
 PlotRange → {50, 2000}, FrameLabel → {"Hz", "nV/ √Hz "}, Evaluate[plotoptn[3]],
 PlotLegends → mylegend[{"DAC noise", "1 dewhitening stage", "2 dewhitening stages"}]]
```

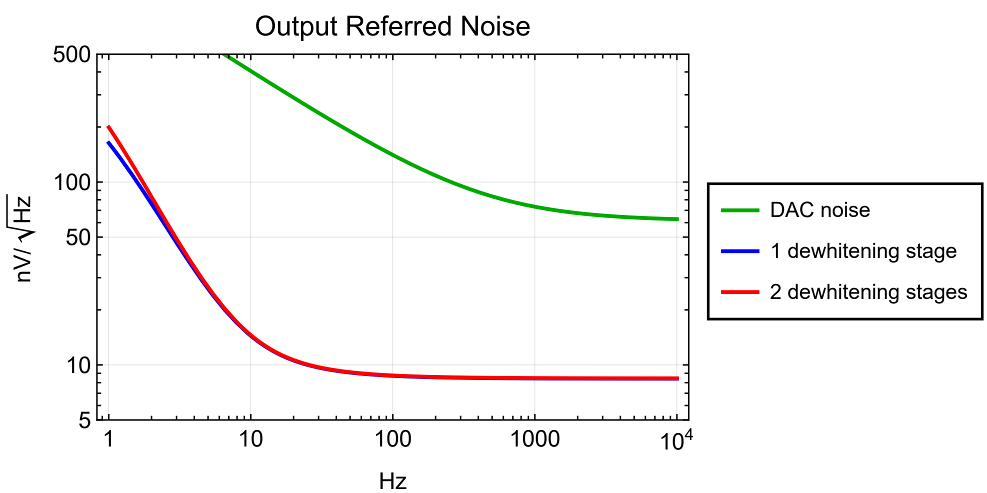
Out[ ]=



Output Referred Noise

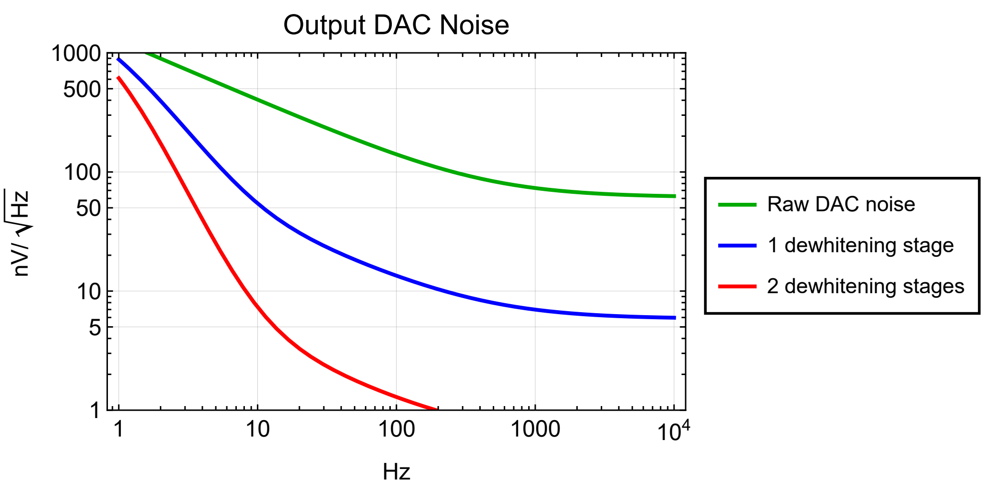
```
In[◦]:= LogLogPlot[Evaluate[1*^9 {dacnoise[f],
        OpAmpNoiseProductOut[opamp, opampnoise, 2] /. {w1 → True, w2 → False},
        OpAmpNoiseProductOut[opamp, opampnoise, 2] /. {w1 → True, w2 → True}} //.
      ADA4898[f] /. fourkt], {f, 1, 10*^3}, PlotRange → {5, 500},
  PlotLabel → "Output Referred Noise", FrameLabel → {"Hz", "nV/ √Hz "},
  Evaluate[plotoptn[3]],
  PlotLegends → mylegend[{"DAC noise", "1 dewhitening stage", "2 dewhitening stages"}]]
```

Out[◦]=



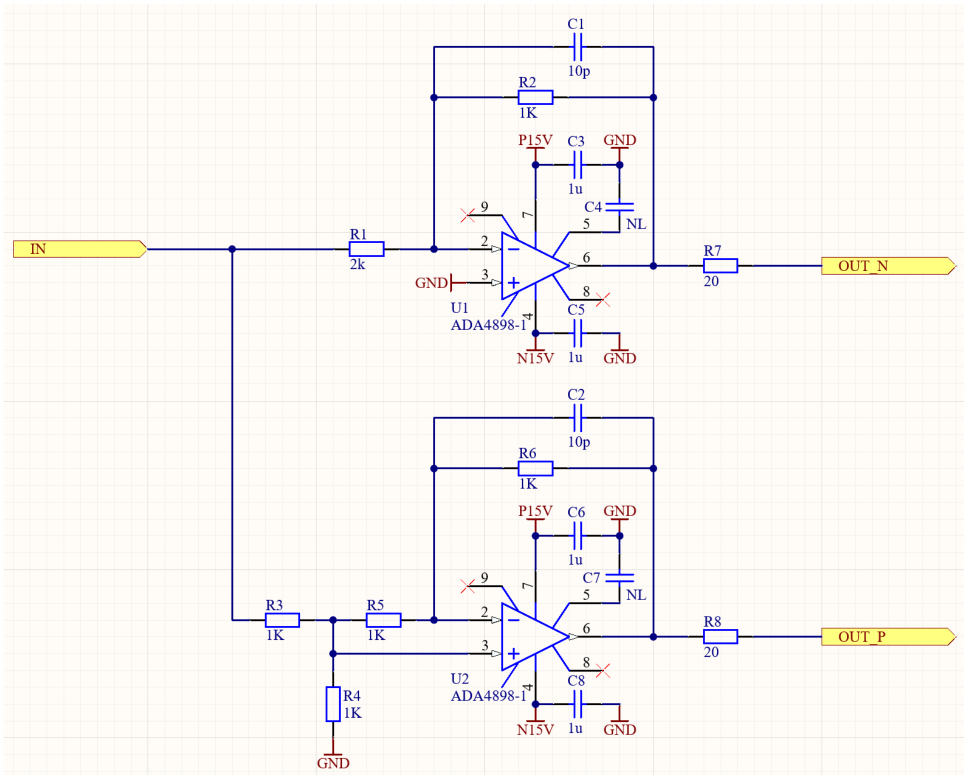
In[ ]:=
```
LogLogPlot[Evaluate[
  1*^9 {dacnoise[f], Abs[OpAmpProduct[opamp, 2]] dacnoise[f] /. {w1 → True, w2 → False},
      Abs[OpAmpProduct[opamp, 2]] dacnoise[f] /. {w1 → True, w2 → True}} //. ADA4898[f] /.
   fourkt], {f, 1, 10*^3}, PlotRange → {1, 1000}, PlotLabel → "Output DAC Noise",
 FrameLabel → {"Hz", "nV/ √Hz "}, Evaluate[plotoptn[3]], PlotLegends →
  mylegend[{"Raw DAC noise", "1 dewhitening stage", "2 dewhitening stages"}]]
```

Out[ ]=



Differential Driver

Schematics



Negative Leg

In[∘]:=
```
zz1[1] := R1
zz2[1] := R2
oopamp[1] := OpAmp[-1, zz1[1], zz2[1]]
oopampnoise[1] := OpAmpNoise[-1, zz1[1], zz2[1], en, in]
```

In[∘]:=
```
PowerExpand[OpAmpNoise[-1, zz1[1], zz2[1], 0, 0]^2,
 Assumptions → {R1 > 0, R2 > 0, FourKT > 0}]
Simplify[% /. R1 → 2 R2]
```

Out[∘]=

$$\text{FourKT R1} + \frac{\text{FourKT R1}^2}{\text{R2}}$$

Out[∘]=

6 FourKT R2

In[∘]:=
```
PowerExpand[OpAmpNoise[-1, zz1[1], zz2[1], en, 0]^2 /. FourKT → 0,
 Assumptions → {R1 > 0, R2 > 0}]
Simplify[% /. R1 → 2 R2]
```

Out[∘]=

$$en^2 \left(1 + \frac{R1}{R2}\right)^2$$

Out[∘]=

$$9\ en^2$$

In[∘]:=
```
PowerExpand[OpAmpNoise[-1, zz1[1], zz2[1], 0, in]^2 /. FourKT → 0,
 Assumptions → {R1 > 0, R2 > 0}]
Simplify[% /. R1 → 2 R2]
```

Out[∘]=

$$in^2\ R1^2$$

Out[∘]=

$$4\ in^2\ R2^2$$

Positive Leg

Equations

In[∘]:=

$$eq1 = \frac{vin - v1}{R3} + \frac{-v1}{R4} + \frac{v2 - v1}{R5} == 0$$

$$eq2 = \frac{vout - v2}{R6} + \frac{v1 - v2}{R5} == 0$$

```
Solve[{eq1, eq2, v1 == v2}, {vout}, {v1, v2}]
```

$$\frac{vout}{vin} \ /.\ \%[[1]]$$

Out[∘]=

$$-\frac{v1}{R4} + \frac{-v1 + v2}{R5} + \frac{-v1 + vin}{R3} == 0$$

Out[∘]=

$$\frac{v1 - v2}{R5} + \frac{-v2 + vout}{R6} == 0$$

Out[∘]=

$$\left\{\left\{vout \to \frac{R4\ vin}{R3 + R4}\right\}\right\}$$

Out[∘]=

$$\frac{R4}{R3 + R4}$$

Noise

$$\text{ipp} = \frac{\text{vin} - \text{v1}}{\text{R3}} + \frac{-\text{v1}}{\text{R4}} + \frac{\text{v2} - \text{v1}}{\text{R5}} + \text{g3}\sqrt{\frac{\text{FourKT}}{\text{R3}}} + \text{g4}\sqrt{\frac{\text{FourKT}}{\text{R4}}} + \text{g5}\sqrt{\frac{\text{FourKT}}{\text{R5}}} == 0;$$

$$\text{inn} = \frac{\text{vout} - \text{v2}}{\text{R6}} + \frac{\text{v1} - \text{v2}}{\text{R5}} + \text{g5}\sqrt{\frac{\text{FourKT}}{\text{R5}}} + \text{g6}\sqrt{\frac{\text{FourKT}}{\text{R6}}} == 0;$$

```
sol = Solve[{ipp, inn, v1 == v2}, {vout}, {v1, v2}][[1]]
```

$$\text{gain} = \frac{\text{vout}}{\text{vin}} \;/.\; \text{sol} \;/.\; \{\text{g3} \to 0, \text{g4} \to 0, \text{g5} \to 0, \text{g6} \to 0\}$$

$$\text{t3} = \text{Simplify}\left[\frac{\text{vout}}{\text{gain}} \;/.\; \text{sol} \;/.\; \text{vin} \to 0 \;/.\; \{\text{g3} \to 1, \text{g4} \to 0, \text{g5} \to 0, \text{g6} \to 0\}\right];$$

$$\text{t4} = \text{Simplify}\left[\frac{\text{vout}}{\text{gain}} \;/.\; \text{sol} \;/.\; \text{vin} \to 0 \;/.\; \{\text{g3} \to 0, \text{g4} \to 1, \text{g5} \to 0, \text{g6} \to 0\}\right];$$

$$\text{t5} = \text{Simplify}\left[\frac{\text{vout}}{\text{gain}} \;/.\; \text{sol} \;/.\; \text{vin} \to 0 \;/.\; \{\text{g3} \to 0, \text{g4} \to 0, \text{g5} \to 1, \text{g6} \to 0\}\right];$$

$$\text{t6} = \text{Simplify}\left[\frac{\text{vout}}{\text{gain}} \;/.\; \text{sol} \;/.\; \text{vin} \to 0 \;/.\; \{\text{g3} \to 0, \text{g4} \to 0, \text{g5} \to 0, \text{g6} \to 1\}\right];$$

$$\text{FullSimplify}\left[\text{t3}^2 + \text{t4}^2 + \text{t5}^2 + \text{t6}^2\right]$$

$$\% \;/.\; \{\text{R3} \to \text{R6}, \text{R4} \to \text{R6}, \text{R5} \to \text{R6}\}$$

$$\%\% \;/.\; \left\{\text{R3} \to \frac{4}{3}\text{R6}, \text{R4} \to \frac{4}{3}\text{R6}, \text{R5} \to \frac{4}{3}\text{R6}\right\}$$

Out[]=

$$\Big\{\text{vout} \to -\frac{1}{\text{R3} + \text{R4}}\left(-\text{g3}\sqrt{\frac{\text{FourKT}}{\text{R3}}}\,\text{R3 R4} - \text{g4 R3}\sqrt{\frac{\text{FourKT}}{\text{R4}}}\,\text{R4} - \text{g5 R3 R4}\sqrt{\frac{\text{FourKT}}{\text{R5}}} + \text{g5 R3}\sqrt{\frac{\text{FourKT}}{\text{R5}}}\,\text{R6} + \text{g5 R4}\sqrt{\frac{\text{FourKT}}{\text{R5}}}\,\text{R6} + \text{g6 R3}\sqrt{\frac{\text{FourKT}}{\text{R6}}}\,\text{R6} + \text{g6 R4}\sqrt{\frac{\text{FourKT}}{\text{R6}}}\,\text{R6} - \text{R4 vin}\right)\Big\}$$

Out[]=

$$\frac{\text{R4}}{\text{R3} + \text{R4}}$$

Out[]=

$$\text{FourKT}\left(\text{R3} + \frac{\text{R3}^2}{\text{R4}} + \frac{(\text{R3} + \text{R4})^2\,\text{R6}}{\text{R4}^2} + \frac{(\text{R3 R4} - (\text{R3} + \text{R4})\,\text{R6})^2}{\text{R4}^2\,\text{R5}}\right)$$

Out[]=

7 FourKT R6

Out[]=

7 FourKT R6

In[•]:=
$$\text{ipp} = \frac{\text{vin}-\text{v1}}{\text{R3}} + \frac{-\text{v1}}{\text{R4}} + \frac{\text{v2}-\text{v1}}{\text{R5}} == 0;$$
$$\text{inn} = \frac{\text{vout}-\text{v2}}{\text{R6}} + \frac{\text{v1}-\text{v2}}{\text{R5}} + \text{in} == 0;$$

```
sol = Solve[{ipp, inn, v1 == v2 + en}, {vout}, {v1, v2}][[1]]
```

$$\text{gain} = \frac{\text{vout}}{\text{vin}} \text{ /. sol /. } \{\text{in} \to 0, \text{en} \to 0\}$$
$$\text{t1} = \text{Simplify}\left[\frac{\text{vout}}{\text{gain}} \text{ /. sol /. vin} \to 0 \text{ /. } \{\text{en} \to 0\}\right];$$
$$\text{t2} = \text{Simplify}\left[\frac{\text{vout}}{\text{gain}} \text{ /. sol /. vin} \to 0 \text{ /. } \{\text{in} \to 0\}\right];$$
$$\text{FullSimplify}\left[\text{t1}^2 + \text{t2}^2\right]$$

```
PowerExpand[% /. {R3 → R6, R4 → R6, R5 → R6}, Assumptions → R6 > 0]
```

$$\text{PowerExpand}\left[\%\% \text{ /. } \left\{\text{R3} \to \frac{4}{3}\text{R6}, \text{R4} \to \frac{4}{3}\text{R6}, \text{R5} \to \frac{4}{3}\text{R6}\right\}, \text{Assumptions} \to \text{R6} > 0\right]$$

Out[•]=
$$\left\{\text{vout} \to -\frac{\text{en R3 R4} + \text{en R3 R5} + \text{en R4 R5} + \text{en R3 R6} + \text{en R4 R6} + \text{in R3 R5 R6} + \text{in R4 R5 R6} - \text{R4 R5 vin}}{(\text{R3}+\text{R4})\ \text{R5}}\right\}$$

Out[•]=
$$\frac{\text{R4}}{\text{R3}+\text{R4}}$$

Out[•]=
$$\frac{\text{in}^2\ (\text{R3}+\text{R4})^2\ \text{R6}^2 + \frac{\text{en}^2\ (\text{R4}\ (\text{R5}+\text{R6}) + \text{R3}\ (\text{R4}+\text{R5}+\text{R6}))^2}{\text{R5}^2}}{\text{R4}^2}$$

Out[•]=
$$25\ \text{en}^2 + 4\ \text{in}^2\ \text{R6}^2$$

Out[•]=
$$\frac{81\ \text{en}^2}{4} + 4\ \text{in}^2\ \text{R6}^2$$

Total

In[•]:=
$$\text{OpAmpNoiseDiffDriverPos[r3\_, r4\_, r5\_, r6\_, en\_, in\_] :=}$$
$$\sqrt{\left(\frac{(\text{r4}\ (\text{r5}+\text{r6}) + \text{r3}\ (\text{r4}+\text{r5}+\text{r6}))^2}{\text{r5}^2\ \text{r4}^2}\ \text{en}^2 + \text{in}^2\ \frac{(\text{r3}+\text{r4})^2}{\text{r4}^2}\ \text{r6}^2 + \text{FourKT}\left(\text{r3} + \frac{\text{r3}^2}{\text{r4}} + \frac{(\text{r3}+\text{r4})^2\ \text{r6}}{\text{r4}^2} + \frac{(\text{r3 r4} - (\text{r3}+\text{r4})\ \text{r6})^2}{\text{r4}^2\ \text{r5}}\right)\right)}$$
$$\text{OpAmpNoiseDiffDriver[r1\_, r2\_, r3\_, r4\_, r5\_, r6\_, en\_, in\_] :=}$$
$$\sqrt{\text{OpAmpNoise}[-1, \text{r1}, \text{r2}, \text{en}, \text{in}]^2 + \text{OpAmpNoiseDiffDriverPos}[\text{r3}, \text{r4}, \text{r5}, \text{r6}, \text{en}, \text{in}]^2}$$

Gain $\frac{1}{2}$ Differential Driver

In[ ]:=
```
noise = FullSimplify[PowerExpand[
   OpAmpNoiseDiffDriver[2 R, R, R, R, R, R, en, in], Assumptions → {R > 0, FourKT > 0}]]
```

Out[ ]=

$$\sqrt{34\,\text{en}^2 + R\,(13\,\text{FourKT} + 8\,\text{in}^2\,R)}$$

In[ ]:=
```
Evaluate[1*^9 {OpAmpNoise[-1, R, R, en, in], noise /. R → 1000} //. ADA4898[f] /. fourkt]
```

Out[ ]=

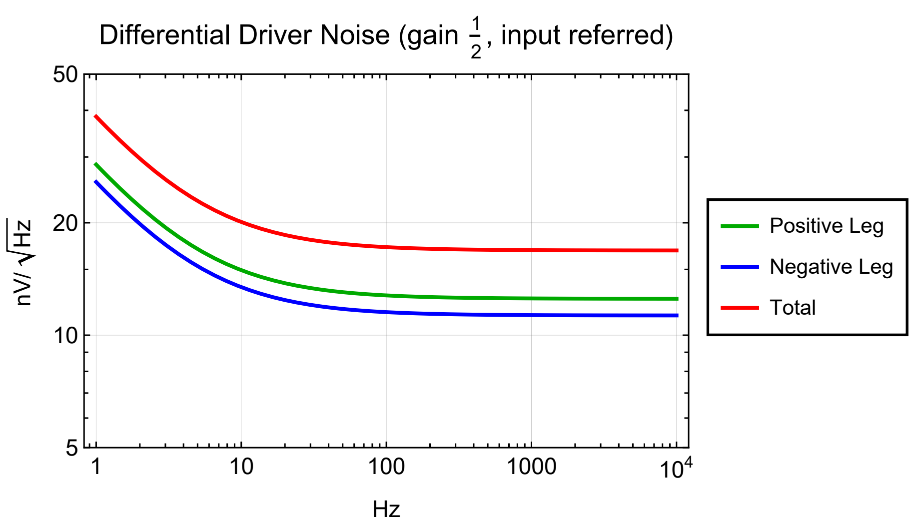
$$\left\{1\,000\,000\,000\,\sqrt{3.24\times10^{-18}\left(1+\frac{10}{f}\right)+\left(5.76\times10^{-24}\left(1+\frac{20}{f}\right)+\frac{3.24\times10^{-20}}{\text{Abs}[R]}\right)\text{Abs}[R]^2},\right.$$
$$\left.1\,000\,000\,000\,\sqrt{1000\left(2.106\times10^{-19}+4.608\times10^{-20}\left(1+\frac{20}{f}\right)\right)+2.754\times10^{-17}\left(1+\frac{10}{f}\right)}\right\}$$

In[ ]:=
```
1*^9 noise /. R → 1000 //. ADA4898[f] /. fourkt /. f → 1000
LogLogPlot[
 Evaluate[1*^9 {OpAmpNoiseDiffDriverPos[R, R, R, R, en, in], OpAmpNoise[-1, 2 R, R, en, in],
     noise} /. R → 1000 //. ADA4898[f] /. fourkt], {f, 1, 10*^3}, PlotRange → {5, 50},
 PlotLabel → "Differential Driver Noise (gain 1/2, input referred)",
 FrameLabel → {"Hz", "nV/√Hz "}, Evaluate[plotoptn[3]],
 PlotLegends → mylegend[{"Positive Leg", "Negative Leg", "Total"}]]
```

Out[ ]=

16.8943

Out[ ]=



Gain 1 Differential Driver

In[∘]:=
```
noise1 = FullSimplify[PowerExpand[
    OpAmpNoiseDiffDriver[R, R, 0, r4, R, R, en, in], Assumptions → {R > 0, FourKT > 0}]]
```

Out[∘]=

$$\sqrt{8\,\text{en}^2 + 2\,R\,\left(2\,\text{FourKT} + \text{in}^2\,R\right)}$$

In[∘]:=
```
Evaluate[
 1*^9 {OpAmpNoiseDiffDriverPos[0, r4, R, R, en, in], OpAmpNoise[-1, R, R, en, in], noise1} /.
    R → 2000 //. ADA4898[f] /. fourkt]
```

Out[∘]=

$$\left\{1\,000\,000\,000\,\sqrt{6.48\times10^{-17} + 3.24\times10^{-18}\left(1+\frac{10}{f}\right) + 2.304\times10^{-17}\left(1+\frac{20}{f}\right)},\right.$$

$$1\,000\,000\,000\,\sqrt{4\,000\,000\left(1.62\times10^{-23} + 5.76\times10^{-24}\left(1+\frac{20}{f}\right)\right) + 3.24\times10^{-18}\left(1+\frac{10}{f}\right)},$$

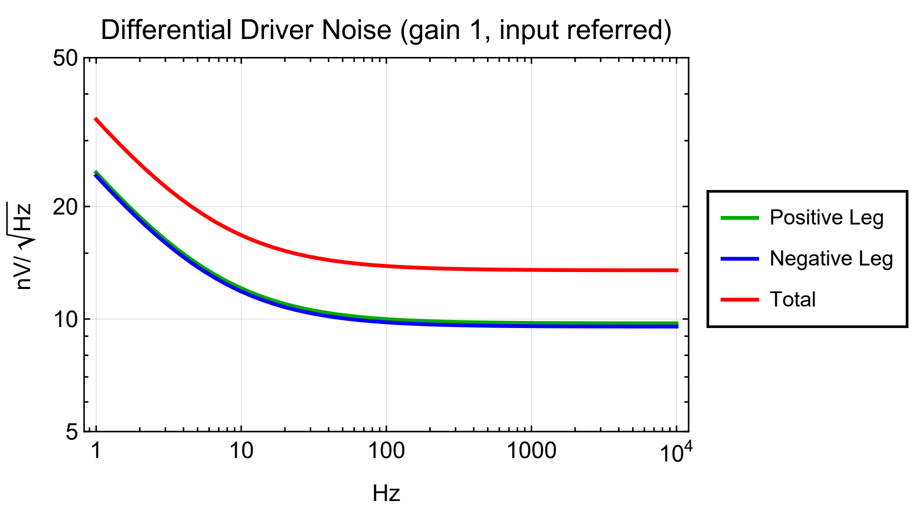
$$\left.1\,000\,000\,000\,\sqrt{4000\left(3.24\times10^{-20} + 1.152\times10^{-20}\left(1+\frac{20}{f}\right)\right) + 6.48\times10^{-18}\left(1+\frac{10}{f}\right)}\right\}$$

In[∘]:=
```
1*^9 noise /. R → 2000 //. ADA4898[f] /. fourkt /. f → 1000
LogLogPlot[Evaluate[1*^9 {1.02 ×
        OpAmpNoiseDiffDriverPos[0, r4, R, R, en, in], OpAmpNoise[-1, R, R, en, in], noise1} /
     R → 2000 //. ADA4898[f] /. fourkt], {f, 1, 10*^3}, PlotRange → {5, 50},
 PlotLabel → "Differential Driver Noise (gain 1, input referred)",
 FrameLabel → {"Hz", "nV/ √Hz "}, Evaluate[plotoptn[3]],
 PlotLegends → mylegend[{"Positive Leg", "Negative Leg", "Total"}]]
```

Out[∘]=

13.5332

Out[∘]=



Whitening/Dewhitening Filters

Whitening

In[∘]:=
```
prmWhitening = {r1 → 15*^3, c1 → 33*^-12, r2 → 1.58*^3, c2 → 10*^-6};
```

$$\text{sol} = \text{Together}\left[1 + \frac{\text{par}\left[\text{r1}, \frac{1}{\text{s c1}}\right]}{\text{r2} + \frac{1}{\text{s c2}}}\right]$$

```
Solve[Numerator[sol] == 0, s];
```

$$\frac{-s}{2.\ \pi}\ /.\ \%\ /.\ \text{prmWhitening}$$

```
Solve[Denominator[sol] == 0, s];
```

$$\frac{-s}{2.\ \pi}\ /.\ \%\ /.\ \text{prmWhitening}$$

Out[∘]=

$$\frac{1 + \text{c1 r1 s} + \text{c2 r1 s} + \text{c2 r2 s} + \text{c1 c2 r1 r2 s}^2}{(1 + \text{c1 r1 s})\ (1 + \text{c2 r2 s})}$$

Out[∘]=

$$\{3.37399 \times 10^6,\ 0.959919\}$$

Out[∘]=

{321 525., 10.0731}

Dewhitening

In[•]:=
```
prmDewhitening = {r1 → 15*^3, r3 → 5*^3, c1 → 22*^-12 + 2.7*^-12, r2 → 1.58*^3, c2 → 10*^-6};
```

$$\text{sol} = \text{Simplify}\left[\frac{r2 + \frac{1}{s\,c2}}{\text{par}\left[r1, \frac{1}{s\,c1} + r3\right] + r2 + \frac{1}{s\,c2}}\right]$$

```
Solve[Numerator[sol] == 0, s]
```

$$\frac{-s}{2.\,\pi} \text{ /. \% /. prmDewhitening}$$

```
Solve[Denominator[sol] == 0, s];
```

$$\frac{-s}{2.\,\pi} \text{ /. \% /. prmDewhitening}$$

Out[•]=

$$\frac{(1 + c2\,r2\,s)\,(1 + c1\,(r1 + r3)\,s)}{1 + c2\,(r1 + r2)\,s + c1\,s\,(r1 + r3 + c2\,r2\,r3\,s + c2\,r1\,(r2 + r3)\,s)}$$

Out[•]=

$$\left\{\left\{s \to -\frac{1}{c2\,r2}\right\}, \left\{s \to -\frac{1}{c1\,(r1 + r3)}\right\}\right\}$$

Out[•]=

{10.0731, 322176.}

Out[•]=

$\{1.00219 \times 10^6, 0.959919\}$

Plots

In[•]:=

$$\text{Show}\Big[\text{BodePlotEx}\Big[\Big\{\text{Together}\left[1 + \frac{\text{par}\left[r1, \frac{1}{s\,c1}\right]}{r2 + \frac{1}{s\,c2}}\right] \text{ /. } s \to 2\,\pi\,i\,f \text{ /. prmWhitening},$$

$$\text{Simplify}\left[\frac{r2 + \frac{1}{s\,c2}}{\text{par}\left[r1, \frac{1}{s\,c1} + r3\right] + r2 + \frac{1}{s\,c2}}\right] \text{ /. } s \to 2\,\pi\,i\,f \text{ /. prmDewhitening},$$

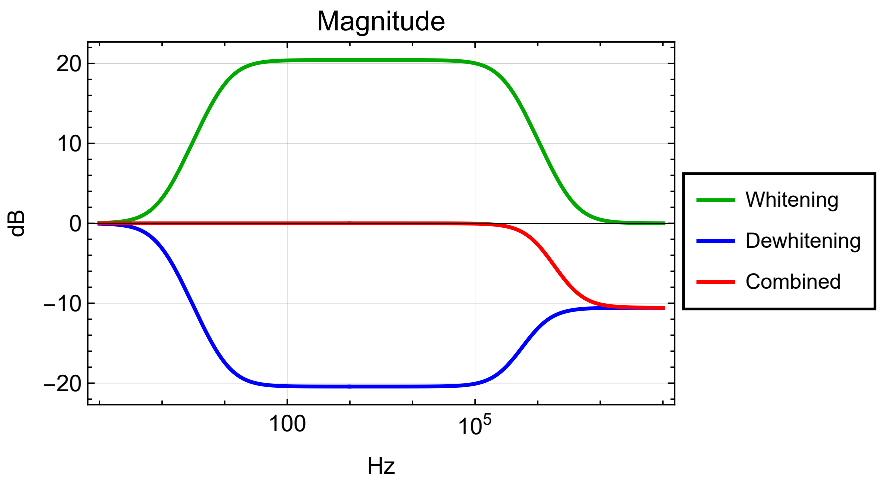
$$\left(\text{Together}\left[1 + \frac{\text{par}\left[r1, \frac{1}{s\,c1}\right]}{r2 + \frac{1}{s\,c2}}\right] \text{ /. } s \to 2\,\pi\,i\,f \text{ /. prmWhitening}\right)$$

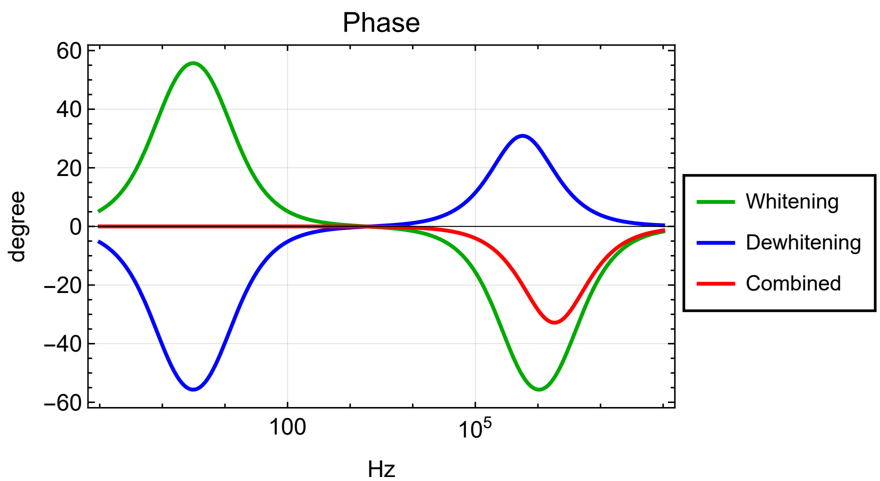
$$\left(\text{Simplify}\left[\frac{r2 + \frac{1}{s\,c2}}{\text{par}\left[r1, \frac{1}{s\,c1} + r3\right] + r2 + \frac{1}{s\,c2}}\right] \text{ /. } s \to 2\,\pi\,i\,f \text{ /. prmDewhitening}\right)\Big\}, \{f, 0.1, 1*\text{^}8\},$$

```
  PlotRange → {5, 50}, PlotLabel → "Whitening/Dewhitening Transfer Function",
  Evaluate[plotoptn[3]],
  PlotLegends → mylegend[{"Whitening", "Dewhitening", "Combined"}],
  ImageSize → 350], ImageSize → 500]
```

Whitening/Dewhitening Transfer Function

Out[ ]=


Magnitude
dB
20
10
0
−10
−20
100
10^5
Hz
Whitening
Dewhitening
Combined


Phase
degree
60
40
20
0
−20
−40
−60
100
10^5
Hz
Whitening
Dewhitening
Combined