Infinite Impulse Response (IIR) Digital Low-Pass Filter Design by Butterworth Method

Initialization Code (optional)

Manipulate

```
Manipulate[
 (
  ok = True;
  If [Not[normalized], {samplingUsed = sampling;
    If[filterOrderType == filterOrderMinumum, {
      If[stop ≥ sampling / 2, stop = sampling / 2 - 0.1];
      If[pass ≥ stop, pass = stop - 0.01]}, {
      If[pass ≥ sampling / 2, pass = sampling / 2 - 0.1];}];

    wpass = N[pass / sampling * 2 * Pi];
    wstop = N[stop / sampling * 2 * Pi];},
   {
    samplingUsed = 2; (*Nyquist=1*)
    If[filterOrderType == filterOrderMinumum,
     If[passNormalized ≥ stopNormalized, passNormalized = stopNormalized - 0.01]];

    wpass = N[passNormalized * Pi];
    wstop = N[stopNormalized * Pi];}
  ];

  If[passdb ≥ stopdb, passdb = stopdb - 0.01];

  Which[
   filterOrderType == filterOrderMinumum, {
    {filterOrderFoundByDesign, Ωc} = butterd[samplingUsed, wpass, wstop , -passdb, -stopdb, method];

    If[filterOrderFoundByDesign > maxFilterOrderAllowed , {
      ok = False;
      statusMsg = Text@
        Style[Row[{"Error: calculated filter order is ", filterOrderFoundByDesign, ". Limit is 10"}]]}]
   },

   filterOrderType == filterOrderSpecified, {
    filterOrderFoundByDesign = filterOrder;
    Ωc = If[method == bilinearMappingMethod, 2 * samplingUsed * Tan[wpass / 2], wpass * samplingUsed];
   }
  ];

  If[ok, {
    hspoles = N@getHsStablePoles[filterOrderFoundByDesign, Ωc];
    hsGain = Re@Chop@getHsGain[hspoles, method, samplingUsed];
    If[hsGain < minumumGainAllowed , {ok = False;
      statusMsg = "Error: Gain found to be too small"}
    ]}];
```

```
If[ok, {
  coeff = getPartialFractionsCoeff[hspoles, hsGain, s];
  hzpoles = Chop@getHzPoles[hspoles, method, samplingUsed];
  hsSecondOrderForm = Chop@getHsFromPolesSecondOrder[hspoles, s, coeff];
  fullFormHS = Chop@FullSimplify@Total@hsSecondOrderForm;
  fullFormHS = Chop@Expand[Numerator@fullFormHS] / Chop@Expand@Denominator@fullFormHS;

  Which[
   method == bilinearMappingMethod, (*bilinear*){
    hz = getHzBilinear[hsSecondOrderForm, samplingUsed, s, z];},

   method == impulseInvarianceMappingMethod, {
    hz = getHzImpulseInvariance[hspoles, coeff, samplingUsed, z];}
  ];

  fullFormHz = FullSimplify@Total@hz;
  fullFormHz = Expand[Numerator@fullFormHz] / Expand@Denominator@fullFormHz;

  Which[
   plotType == plotTypeResponseSpectrumDBScale,
   plots = magPhasePlot[hz, samplingUsed / 2, useDBScale, z, normalized],

   plotType == plotTypeResponseSpectrumLinearScale,
   plots = magPhasePlot[hz, samplingUsed / 2, useLinearScale, z, normalized],

   plotType == plotTypeButterworth,
   plots = doplotTypeButterworth[filterOrderFoundByDesign, Ωc],

   plotType == plotTypeShowPoles,
   plots = polesPlot[hzpoles, hspoles, Ωc, filterOrderFoundByDesign, method],

   plotType == plotTypeHSfirstOrder,
   plots = doHsPlot[hspoles, coeff, 1. / samplingUsed, s],

   plotType == plotTypeHSSecondOrder,
   If[filterOrderFoundByDesign == 1, plots = doHsPlot[hspoles, coeff, 1. / samplingUsed, s],
    plots = doHs2Plot[hsSecondOrderForm, s]
   ],

   plotType == plotTypeHSPolynomial,
   plots = doHsFullPlot[fullFormHS],

   plotType == plotTypeHzfirstOrder,
   plots = doHzPlot[hspoles, coeff, hzpoles, 1. / samplingUsed, z],

   plotType == plotTypeHzSecondOrder,
   If[filterOrderFoundByDesign == 1, plots = doHzPlot[coeff, hzpoles, 1. / samplingUsed, z],
    plots = doHz2Plot[hz, z]],

   plotType == plotTypeHzPolynomial,
   plots = doHzFullPlot[fullFormHz],

   plotType == plotTypeImpluseResponse,
   plots = doImpulseResponsePlot[fullFormHS, s, hz, z, 1 / samplingUsed, nSamplesForPlotting],

   plotType == plotTypeStepResponse,
   plots = doStepResponsePlot[fullFormHS, s, hz, z, 1 / samplingUsed, nSamplesForPlotting],

   plotType == plotTypeRampResponse,
   plots = doRampResponsePlot[fullFormHS, s, hz, z, 1 / samplingUsed, nSamplesForPlotting]
  ]
 }
];
```

```
 If[Not[ok], statusMsg = Text@Style[statusMsg, Red, 14],
  statusMsg = Text@Style[Row[{"Filter order ", Style["N", Italic], " = ",
        filterOrderFoundByDesign, ". Cutoff frequency ", "ω", " = ", Ωc, " (rad/sec)"}], 12]];

 Column[{Framed[plots, ImageSize → {390, 350}, FrameMargins → 4],
   Framed[statusMsg, ImageSize → {390, 22}, ImageMargins → 0, Alignment → {Center}, FrameMargins → 0]}]
),
(*-------------------------------------------*)
(*  C O N T R O L   L A Y O U T  LEFT SIDE  *)
(*-------------------------------------------*)
Item[
 Grid[{
   {
    Panel[
     Labeled[RadioButtonBar[Dynamic[method], {bilinearMappingMethod → Style["bilinear method", 10],
         impulseInvarianceMappingMethod → Style["impulse invariance", 10]},
       Appearance → "Vertical"],
      Style["mapping method", "Label", Bold], {{Top, Center}}],
     FrameMargins → 3
    ],
    Panel[
     Labeled[RadioButtonBar[Dynamic[normalized],
       {True → Style["normalized", 10], False → Style["Hz", 10]}, Appearance → "Vertical"],
      Style["frequency units", "Label", Bold], {{Top, Center}}],
     FrameMargins → 3]
   },
   {
    Panel[Labeled[Column[{
        Control[{{passNormalized, .2, Style["F_pass ", 10]}, .01, .98, .01,
          Appearance → "Labeled", ImageSize → Small, Enabled → Dynamic[normalized === True]}],

        Control[{{stopNormalized, .3, Style["F_stop ", 10]}, .02, .99, .01, Appearance → "Labeled",
          ImageSize → Small, Enabled → Dynamic[normalized === True && filterOrderType == filterOrderMinumum]}]
       }],
      Style["normalized frequency input", "Label", Bold], {{Top, Center}}]
     , FrameMargins → 3],
    SpanFromLeft
   }, {
    Panel[Labeled[Column[{
        Control[{{pass, 2, Style["F_pass ", 10]}, 0.1, 49.8, 0.1,
          Appearance → "Labeled", ImageSize → Small, Enabled → Dynamic[normalized === False]}],

        Control[{{stop, 3, Style["F_stop ", 10]}, 0.2, 49.9, 0.1, Appearance → "Labeled", ImageSize → Small,
          Enabled → Dynamic[normalized === False && filterOrderType == filterOrderMinumum]}],

        Control[{{sampling, 10, Style["F_samp", 10]}, 1, 100, 0.1,
          Appearance → "Labeled", ImageSize → Small, Enabled → Dynamic[normalized === False]}]

       }], Style["Hz frequency input", "Label", Bold], {{Top, Center}}]
     , FrameMargins → 3],
    SpanFromLeft
   }, {
    Panel[Labeled[Column[{
        Control[{{passdb, 1, Style["A_pass ", 10]}, 0.1, 99.0, 0.1, Appearance → "Labeled",
          ImageSize → Small, Enabled → Dynamic[filterOrderType == filterOrderMinumum]}],

        Control[{{stopdb, 15, Style["A_stop ", 10]}, 0.2, 100, 0.1, Appearance → "Labeled",
          ImageSize → Small, Enabled → Dynamic[filterOrderType == filterOrderMinumum]}]
       }], Style["attenuation in db", "Label", Bold], {{Top, Center}}]
     , FrameMargins → 3],
    SpanFromLeft
   }
  }], ControlPlacement → Left
],
```

```
(*-----------------------------------------*)
(*  C O N T R O L   L A Y O U T  TOP  SIDE  *)
(*-----------------------------------------*)
Item[
 Grid[{
   {Panel[ doPlotButterworthSpecs[2.3, 5.3, -7, -26, 2, 3]],
    Panel[Column[{Labeled[
         RadioButtonBar[Dynamic[filterOrderType], {filterOrderSpecified → Style["specific order", 10],
           filterOrderMinumum → Style["minumum", 10]}, Appearance → "Horizontal"
         ], Style["method to determine filter order", "Label", Bold], {{Top, Center}}
        ],
        Framed@Labeled[
          Control[{{filterOrder, 3, ""}, 1, maxFilterOrderAllowed , 1, Appearance → "Labeled",
            ImageSize → Small, Enabled → Dynamic[filterOrderType == filterOrderSpecified]}
          ], Style["filter order", "Label", Bold], {{Top, Center}}
         ]
       }, Alignment → Center, Spacings → 1
     ], FrameMargins → 4, ImageMargins → 1], (*Panel*)
    Panel[Column[{
       Labeled[Control[{ {plotType, 1, ""}, {
            plotTypeResponseSpectrumDBScale -> Style["digital filter spectrum db", 11],
            plotTypeResponseSpectrumLinearScale → Style["digital filter spectrum linear", 11],
            plotTypeButterworth → Style["Butterworth gain plot", 11],
            plotTypeShowPoles → Style["locations of poles and zeros", 11],
            plotTypeHSfirstOrder → Style["H(s) first-order sections", 11],
            plotTypeHSSecondOrder → Style["H(s) second-order sections", 11],
            plotTypeHSPolynomial → Style["H(s) single section", 11],
            plotTypeHzfirstOrder → Style["H(z) first-order sections", 11],
            plotTypeHzSecondOrder → Style["H(z) second-order sections", 11],
            plotTypeHzPolynomial → Style["H(z) single section", 11],
            plotTypeImpluseResponse → Style["filter impulse response", 11],
            plotTypeStepResponse → Style["filter step response", 11],
            plotTypeRampResponse → Style["filter ramp response", 11],
           }, ControlType → PopupMenu, ImageSize → Medium, FrameMargins → 2}
         ], Style["select result to display", "Label", Bold], {{Top, Center}}],

       Framed@Labeled[
         Control[{{nSamplesForPlotting, 100, Style["", 10]},
           10, 500, 1, Appearance → "Labeled", ImageSize → Small}]
         , Style["number of samples to plot", "Label", Bold], {{Top, Center}}
        ]
      }, Alignment → Center, Spacings → 1], FrameMargins → 4
    ](*Panel*)
   }}],
 ControlPlacement → Top],

{{filterOrderType, 2}, ControlType → None},
{{method, 1}, ControlType → None},
{{samplingUsed, 0}, ControlType → None},
{{normalized, True}, ControlType → None},
{{plots, 0}, ControlType → None},
{{ok, True}, ControlType → None},
{{wpass, 1}, ControlType → None},
{{wstop, 1}, ControlType → None},
{{hsSecondOrderForm, 2}, ControlType → None},
{{hsGain, 0}, ControlType → None},
{{fullFormHS, 0}, ControlType → None},
{{fullFormHz, 0}, ControlType → None},
{{statusMsg, ""}, ControlType → None},
{{filterOrderFoundByDesign, 0}, ControlType → None},
{{coeff, 0}, ControlType → None},
{{Ωc, 0}, ControlType → None},
{{hz, 0}, ControlType → None},
{{hspoles, 0}, ControlType → None},
```

```
{{hzpoles, 0}, ControlType → None},
{{scale, 0}, ControlType → None},

ContinuousAction → False,
SynchronousUpdating → True,
AutorunSequencing → {1, 2},
FrameMargins → 0,
ImageMargins → 0,
TrackedSymbols :→ {filterOrderType, filterOrder, method, normalized, stop, pass,
  passNormalized, stopNormalized, sampling, passdb, stopdb, plotType, nSamplesForPlotting},

Initialization :→
 {
  filterOrderMinumum = 2;
  filterOrderSpecified = 1;
  impulseInvarianceMappingMethod = 2;
  bilinearMappingMethod = 1;
  plotTypeResponseSpectrumDBScale = 1;
  plotTypeResponseSpectrumLinearScale = 2;
  plotTypeShowPoles = 4;
  plotTypeHSfirstOrder = 5;
  plotTypeHSSecondOrder = 6;
  plotTypeHSPolynomial = 7;
  plotTypeHzfirstOrder = 8;
  plotTypeHzSecondOrder = 9;
  plotTypeHzPolynomial = 10;
  plotTypeImpluseResponse = 11;
  plotTypeStepResponse = 12;
  plotTypeRampResponse = 13;
  plotTypeButterworth = 14;
  useDBScale = 1;
  useLinearScale = 2;
  useSquaredScale = 3;
  maxFilterOrderAllowed = 10;
  minumumGainAllowed = 0.000001;
  orderLimitForDisplay = 7;

  (*-----------------------------------------------*)
  makehs[
    hspoles_ /; VectorQ[hspoles, NumberQ], hsgain_ /; NumberQ[hsgain] && Im[hsgain] == 0, s_]
   := Module[{i},
    hsgain / Product[s - hspoles[[i]], {i, 1, Length[hspoles]}]
   ];

  (*-----------------------------------------------*)
  getPartialFractionsCoeff[
    hspoles_List /; VectorQ[hspoles, NumberQ],
    hsgain_ /; NumberQ[hsgain] && Im[hsgain] == 0, s_] := Module[{i},

    Chop@Table[ Limit[makehs[ Delete[hspoles, i], hsgain , s ], s → hspoles[[i]]], {i, 1, Length[hspoles]}]
   ];

  (*-----------------------------------------------*)
  ticksForMag[] := Module[{}, {
     {.2 Pi, ".2"}, {.4 Pi, ".4"}, {.6 Pi, ".6"}, {.8 Pi, ".8"}, {Pi, "1"}}
   ];

  (*-----------------------------------------------*)
  ticksForMag[ nyquist_ /; NumberQ[nyquist] && Positive[nyquist] ] := Module[{},
    {{.2 Pi , Style[  .2 * nyquist ] }, {.4 Pi , Style[  .4 * nyquist ]},
     {.6 Pi , Style[  .6 * nyquist ] }, {.8 Pi , Style[  .8 * nyquist ] },
     {Pi , Style[  N[nyquist] ]}}
   ];
```

```
(*-----------------------------------------------*)
formatPolynomialFormHZ[hz_, z_] := Module[{num, den, cnum, cden, formNumerator, formDen, len, hzz, i},
  hzz = Together[hz];
  num = Numerator[hzz];
  den = Denominator[hzz];
  cnum = CoefficientList[num, z];
  cden = CoefficientList[den, z];
  len = Length[cden];
  formDen = Table[If[i == 1, 1, Superscript[z, -(i - 1)]], {i, 1, len}];
  len = Length[cnum];
  formNumerator = Table[If[i == 1, 1, Superscript[z, -(i - 1)]], {i, 1, len}];
  formDen = Reverse[formDen];
  formNumerator = Reverse[formNumerator];
  Dot[formNumerator, cnum] / Dot[formDen, cden]
 ];

(*-----------------------------------------------*)
polesPlot[
  hzpoles_ /; VectorQ[hzpoles, NumberQ],
  hspoles_ /; VectorQ[hspoles, NumberQ],
  Ωc_ /; NumberQ[Ωc] && Positive[Ωc],
  filterOrder_ /; NumberQ[filterOrder] && Positive[filterOrder],
  method_ /; NumberQ[method] && Positive[method]
 ] := Module[{i, hsPolesLocations, hzPolesLocations,
   hsPolesPlot, hzPolesPlot, t, commonPlotOptions, maxPolesToShow = 10},

  hsPolesLocations =
   Table[{ComplexExpand[Re[hspoles[[i]]]], ComplexExpand[Im[hspoles[[ i]]]]}, {i, 1, Length[hspoles]}];

  hzPolesLocations =
   Table[{ComplexExpand[Re[hzpoles[[i]]]], ComplexExpand[Im[hzpoles[[ i]]]]}, {i, 1, Length[hzpoles]}];

  commonPlotOptions = {ImageMargins → 1, ImagePadding → All, ImageSize → Full, Frame → True,
    AspectRatio → .7, Ticks → None, PlotStyle → {Thin, Black}, TicksStyle → Directive[10],
    Exclusions → None, GridLines → Automatic};

  hsPolesPlot = Plot[0, {t, -Ωc, Ωc},
    PlotRange → {{-1.1 Ωc, 1.1 Ωc}, {-1.1 Ωc, 1.1 Ωc}}, Evaluate[commonPlotOptions],
    PlotLabel → Text@Row[{Style[Row[{Style["H", Italic], "(", Style["s", Italic],
           ") poles, ", "Ω"Style["c",Italic], " = ", numIt[N[Ωc], 6, 4]}], 10]}],
    Epilog → {Circle[{0, 0}, Ωc],
      Table[
       Text[Style["×", Thick, Red, 18], hsPolesLocations[[i]]], {i, 1, Length[hsPolesLocations]}]}];

  hzPolesPlot = Plot[0, {t, -1, 1.1},
    Evaluate[commonPlotOptions], PlotRange → {{-1.1, 1.1 }, {-1.1, 1.1}},
    PlotLabel → Style[Row[{Style["H", Italic], "(", Style["z", Italic],
        ") poles/zeros, ", Style["N", Italic], " = ", Style[filterOrder, 10]}], 10],
    Epilog → {
      Circle[{0, 0}, 1],
      Table[
       Text[Style["×", Thick, Red, 18], hzPolesLocations[[i]]], {i, 1, Length[hzPolesLocations]}],
      If[method == bilinearMappingMethod, {
        Text[Style["o", Thick, Black, 20], {-1, 0}],
        Text[Style[Length[hzPolesLocations], Black, 12], {-.9, 0}, {0, -1}]},
       Text["", {0, 0}]]}];

  Grid[{
    {GraphicsRow[{hsPolesPlot, hzPolesPlot}, ImageSize → Full]},
    {GraphicsRow[
      {TableForm[ Table[{numIt[hsPolesLocations[[i, 1]], 6, 4], numIt[hsPolesLocations[[ i, 2] ], 6, 4] },
         {i, 1, If[filterOrder > maxPolesToShow, maxPolesToShow, filterOrder]}],
```

```
        TableAlignments → Center, TableHeadings → {None, {Style["Re", 11], Style["Im", 11]}}] ,
       TableForm[ Table[{numIt[hzPolesLocations[[i, 1]], 6, 4], numIt[hzPolesLocations[[ i, 2] ], 6, 4] ,
          numIt[EuclideanDistance[{0, 0}, {hzPolesLocations[[i, 1]], hzPolesLocations[[i, 2]]}], 6, 4]},
         {i, 1, If[filterOrder > maxPolesToShow, maxPolesToShow, filterOrder]}],
        TableAlignments → Center, TableHeadings → {None, {Style["Re", 11],
           Style["Im", 11], Style[Row[{, "|", Style["z", Italic], "|"}], 11]}}]
      }, ImageSize → Full]}}]
 ];

(*-----------------------------------------------*)
magPhasePlot[hz_, (* H(z) *)
  (nyquist_ /; NumberQ[nyquist] && Positive[nyquist]),
  (vscale_ /; IntegerQ[vscale]),  (* for vertical scale, 1 or 2 or 3 *)
  z_ , (* H(z) variable*)
  isNormalized_
 ] := Module[{dtft, ω, mag, phase, absMag, yTitle, data, i, commonPlotOptions},

  dtft = hz /. z → Exp[I ω] ;
  dtft = Total@dtft;
  absMag = Abs[ComplexExpand@dtft];
  Which[
   vscale == useLinearScale, yTitle = Style["magnitude", 12],
   vscale == useDBScale, {absMag = 10 Log[10, absMag], yTitle = Style["gain (db)", 12]}
  ];

  commonPlotOptions = {ImagePadding → {{65, 10}, {40, 5}}, ImageSize → Full, ImageMargins → 1,
    AxesOrigin → {0, 0}, Frame → True, AspectRatio → .40,
    Frame → True, PlotStyle → {Thick, Red}, TicksStyle → Directive[10], Joined → True,
    GridLines → {Range[0, Pi, .1 Pi], Automatic}};

  data = Table[{i, absMag /. ω → i}, {i, 0, Pi, Pi / 100}];

  mag = ListPlot[ data,
    Evaluate[commonPlotOptions], GridLines → {Range[0, Pi, .1 Pi], Automatic},
    FrameTicks → {{Automatic, None}, {If[isNormalized, ticksForMag[], ticksForMag[nyquist]], None}},
    PlotRange → {{0, Pi}, Automatic}, FrameLabel → {{yTitle, None},
      {If[isNormalized, Style["normalized frequency", 12], Style["frequency (hz)", 12]], None}}
   ];

  data = Table[{i, 180 / Pi Arg[ComplexExpand[dtft /. ω → i]]}, {i, 0, Pi, Pi / 100}];

  phase = ListPlot[data,
    Evaluate[commonPlotOptions], FrameTicks →
     {{Range[-200, 200, 40], None}, {If[isNormalized, ticksForMag[], ticksForMag[nyquist]], None}},
    PlotRange → {{0, Pi}, {-200, 200}}, FrameLabel → {{Style["phase (degree)", 12], None},
      {If[isNormalized, Style["normalized frequency", 12], Style["frequency (hz)", 12]], None}}
   ];

  Labeled[GraphicsColumn[{mag, phase}, ImageSize → Full],
   Text@Style["digital filter frequency response", 12, Bold], {{Top, Center}}]
 ];

(*-----------------------------------------------*)
numIt[v_?(NumberQ[#] &),
  s1_?(NumberQ[#] && Positive[#] &), s2_?(NumberQ[#] && Positive[#] &)] := Module[{},

  Style[
   ToString[AccountingForm[Chop[v], {s1, s2}, NumberPadding → {" ", "0"}, NumberSigns → {"-", ""}]], 11]
 ];

(*-----------------------------------------------*)
str[expr_] := Module[{}, StringReplace[
   ToString[expr, FormatType → TraditionalForm], c : LetterCharacter ~~ "$" ~~ DigitCharacter .. :> c]
```

```
 ];

(*-----------------------------------------------*)
butterd[
  fs_?(NumberQ[#] && Positive[#] &),      (* sampling frequency*)
  wpass_?(NumberQ[#] && Positive[#] &),   (* passband corner frequency*)
  wstop_?(NumberQ[#] && Positive[#] &),   (* stopband corner frequency*)
  δp_?(NumberQ[#] && Negative[#] &),      (* attenuation at passband in db*)
  δs_?(NumberQ[#] && Negative[#] &),      (* attenuation at stopband in db*)
  method_?(NumberQ[#] && Positive[#] &)  (* bilinear or impulse invariance*)

 ] := Module[{T = 1 / fs, αs, αp, filterOrder, Ωc, Ωpass, Ωstop},

  If[method == bilinearMappingMethod, {Ωpass = N[2/T Tan[wpass / 2]]; Ωstop = N[2/T Tan[wstop / 2]]},
   {Ωpass = N[ wpass / T]; Ωstop = N[wstop / T]}];
  αs = N[10^(-δs/10)];
  αp = N[10^(-δp/10)];
  filterOrder = 1/2 ((Log[10, αp - 1] - Log[10, αs - 1]) / (Log[10, Ωpass] - Log[10, Ωstop]));
  filterOrder = Ceiling[filterOrder];
  Ωc = 0;
  If[filterOrder <= 10,
   Ωc =
    N[If[method == bilinearMappingMethod, Ωstop / 10^(1/(2 filterOrder) Log[10,αs-1]), Ωpass / 10^(1/(2 filterOrder) Log[10,αp-1])]]];

  {filterOrder, Ωc}
 ];

(*-----------------------------------------------*)
getHsStablePoles[filterOrder_?(NumberQ[#] && Positive[#] &),
  Ωc_?(NumberQ[#] && Positive[#] &)] := Module[{i, poles},

  poles = Table[0, {i, filterOrder}];
  Do[{
    poles[[i + 1]] = Ωc (Cos[(Pi (1 + 2 i + filterOrder)) / (2 filterOrder)] +
        I Sin[(Pi (1 + 2 i + filterOrder)) / (2 filterOrder)]);}
   , {i, 0, Length[poles] - 1}
  ];
  poles
 ];

(*-----------------------------------------------*)
getHsGain[
  hspoles_ /; VectorQ[hspoles, NumberQ],
  method_?(NumberQ[#] && Positive[#] &),
  fs_?(NumberQ[#] && Positive[#] &)] := Module[{k,  T = 1 / fs, i},

  k = Chop[ Product[-hspoles[[i]], {i, 1, Length[hspoles]}]];
  If[method == bilinearMappingMethod, k, T * k ]
 ];

(*-----------------------------------------------*)
getHsFromPolesSecondOrder[
  hspoles_ /; VectorQ[hspoles, NumberQ], s_, coeff_ /; VectorQ[coeff, NumberQ]
 ] := Module[{i, hs, expr},
```

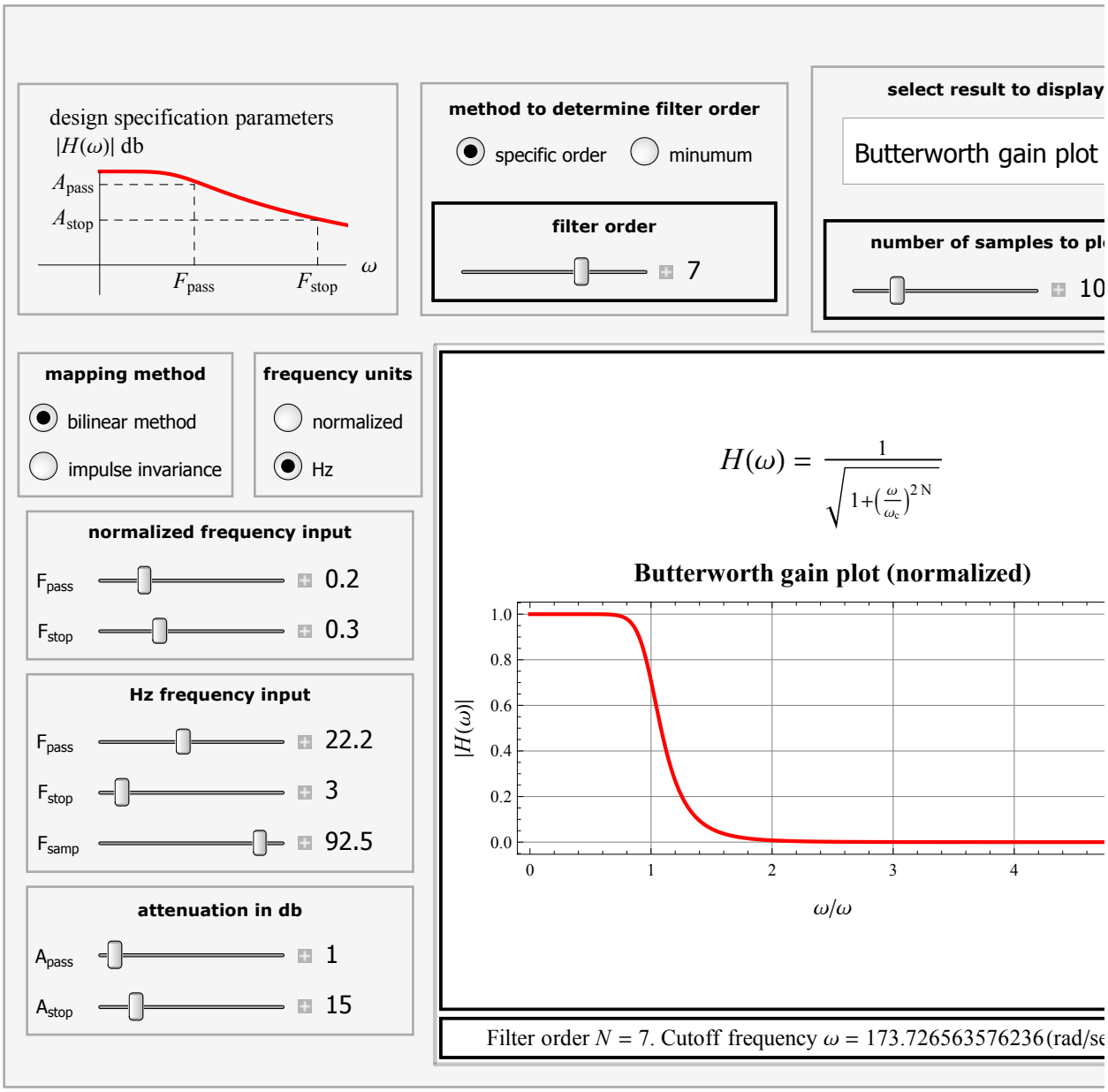
```
-

  If[EvenQ[Length[hspoles]], {
    hs = Table[0, {i, Length[hspoles] / 2}];
    Do[{
      expr = coeff[[i]] / (s - hspoles[[i]]) + Conjugate@coeff[[i]] / (s - Conjugate@hspoles[[i]]);
      expr = Normal@Chop@FullSimplify[ComplexExpand[expr]];
      expr = ComplexExpand[Numerator[expr]] / ComplexExpand[Denominator[expr]];
      hs[[i]] = expr;
     },
     {i, 1, Length[hs]}
    ];},
   {
    hs = Table[0, {i, 1 + ((Length[hspoles] - 1) / 2)}];
    Do[{
      expr = coeff[[i]] / (s - hspoles[[i]]) + Conjugate@coeff[[i]] / (s - Conjugate@hspoles[[i]]);
      expr = Normal@Chop@FullSimplify[ComplexExpand[expr]];
      expr = ComplexExpand[Numerator[expr]] / ComplexExpand[Denominator[expr]];
      hs[[i]] = expr;
     },
     {i, 1, Length[hs] - 1}
    ];
    hs[[-1]] = coeff[[(Length[hspoles] - 1) / 2 + 1]] / (s - hspoles[[(Length[hspoles] - 1) / 2 + 1]]);
   }
  ];
  hs
 ];

(*-----------------------------------------------*)
getHzPoles[
  hspoles_ /; VectorQ[hspoles, NumberQ], method_?(NumberQ[#] && Positive[#] &),
  fs_?(NumberQ[#] && Positive[#] &)] := Module[{T = 1 / fs, i},

  If[method == bilinearMappingMethod,
   Table[(1 + (T/2) hspoles[[i]]) / (1 - (T/2) hspoles[[i]]), {i, 1, Length[hspoles]}],
   Table[Exp[T * hspoles[[i]]], {i, 1, Length[hspoles]}]
  ]
 ];

(*-----------------------------------------------*)
getHzBilinear[hsSecondOrderForm_List,
  sampling_?(NumberQ[#] && Positive[#] &), s_, z_] := Module[{T = 1 / sampling, i},
  Table[
   FullSimplify[hsSecondOrderForm[[i]] /. s → (2/T) (1 - z^-1) / (1 + z^-1)], {i, 1, Length[hsSecondOrderForm]}]
 ];

(*-----------------------------------------------*)
getHzImpulseInvariance[hspoles_ /; VectorQ[hspoles, NumberQ],
```

```
  coeff_ /; VectorQ[coeff, NumberQ], sampling_?(NumberQ[#] && Positive[#] &), z_] :=
 Module[{T = 1 / sampling, i, hz, expr},

  If[EvenQ[Length[hspoles]], {
    hz = Table[0, {i, Length[hspoles] / 2}];
    Do[{
      expr = z coeff[[i]] / (z - Exp[hspoles[[i]] T]) + (z Conjugate@coeff[[i]]) / (z - Exp[Conjugate@hspoles[[i]] T]);

      expr = Normal@Chop@FullSimplify[ComplexExpand[expr]];
      expr = ComplexExpand[Numerator[expr]] / ComplexExpand[Denominator[expr]];
      hz[[i]] = expr;
     },
     {i, 1, Length[hz]}
    ];
   }, {
    hz = Table[0, {i, 1 + ((Length[hspoles] - 1) / 2)}];
    Do[{
      expr = z coeff[[i]] / (z - Exp[hspoles[[i]] T]) + (z Conjugate@coeff[[i]]) / (z - Exp[Conjugate@hspoles[[i]] T]);
      expr = Normal@Chop@FullSimplify[ComplexExpand[expr]];
      expr = ComplexExpand[Numerator[expr]] / ComplexExpand[Denominator[expr]];
      hz[[i]] = expr;
     },
     {i, 1, Length[hz] - 1}
    ];
    hz[[-1]] = (z coeff[[(Length[hspoles] - 1) / 2 + 1]]) / (z - Exp[T hspoles[[(Length[hspoles] - 1) / 2 + 1]]]);
   }
  ];

  hz
 ];

(*-----------------------------------------------*)
doHsPlot[hspoles_ /; VectorQ[hspoles, NumberQ], coeff_ /; VectorQ[coeff, NumberQ],
  T_?(NumberQ[#] && Positive[#] &), s_] := Module[{i, hs, fontSize},

  hs = Total@Table[ coeff[[i]] / (s - hspoles[[i]]), {i, 1, Length[hspoles]}];
  If[Length[hs] < orderLimitForDisplay, fontSize = 24, fontSize = 16];
  Grid[{
    {Text@TraditionalForm[Style[hs, fontSize]]}
   }, Alignment → Center, Spacings → 0]
 ];

(*-----------------------------------------------*)
doHzPlot[hspoles_ /; VectorQ[hspoles, NumberQ], coeff_ /; VectorQ[coeff, NumberQ],
  hzpoles_ /; VectorQ[hzpoles, NumberQ],
  T_?(NumberQ[#] && Positive[#] &), z_] := Module[{i, hz, fontSize},

  hz = Total@Table[ (coeff[[i]]) / (1 - z^-1 * Exp[hspoles[[i]] T]) , {i, 1, Length[hspoles]}];
  If[Length[hz] < orderLimitForDisplay, fontSize = 24, fontSize = 16];
  Grid[{
    {Text@TraditionalForm[Style[hz, fontSize]]}
   }, Alignment → Center, Spacings → 0]
 ];
```

```
(*-----------------------------------------------*)
doHsFullPlot[hs_] := Module[{fontSize},
  fontSize = 16;
  Grid[{
    {Text@TraditionalForm[Style[hs, fontSize]]}}, Alignment → Center, Spacings → 0]
 ];

(*-----------------------------------------------*)
doHzFullPlot[hz_] := Module[{fontSize},
  fontSize = 16;
  Grid[{{Text@TraditionalForm[Style[hz, fontSize]]}}, Alignment → Center, Spacings → 0]
 ];

(*-----------------------------------------------*)
doHs2Plot[hs_, s_] := Module[{i, myhs, c, fontSize},
  myhs = Table[0, {i, Length[hs]}];
  Do[
   {
    myhs[[i]] = Expand@FullSimplify[Numerator[hs[[i]]]] / Expand@FullSimplify[Denominator[hs[[i]]]];

    c = CoefficientList[Denominator[hs[[i]]], s];
    If[Length[c] == 3, {
      c = c[[3]];
      myhs[[i]] =
       Expand@FullSimplify[Numerator[myhs[[i]]] / c] / Expand@FullSimplify[Denominator[myhs[[i]]] / c] ;}
    ];}, {i, 1, Length[hs]}];

  If[Length[hs] < orderLimitForDisplay, fontSize = 24, fontSize = 18];
  Grid[{{Text@TraditionalForm[Style[Total@myhs, fontSize]]}}, Alignment → Center, Spacings → 0]
 ];


(*-----------------------------------------------*)
doHz2Plot[hz_, z_] := Module[{i, myhz, c, fontSize},
  myhz = Table[0, {i, Length[hz]}];
  Do[{myhz[[i]] = formatPolynomialFormHZ[hz[[i]], z];}
   , {i, 1, Length[hz]}];

  If[Length[myhz] < orderLimitForDisplay, fontSize = 24, fontSize = 18];
  Grid[{
    {Text@TraditionalForm[Style[Total@myhz, fontSize]]}}, Alignment → Center, Spacings → 0]
 ];

(*-----------------------------------------------*)
inverseZtransform[hz_, z_, n_?(IntegerQ[#] && Positive[#] &)] := Module[{num, den, h},
  num = Chop[Numerator[hz], 10^-5];
  If[num == 0, num = 1];
  den = Denominator[hz];
  h = Normal[Series[num / den, {z, Infinity, n}]];
  CoefficientList[h, z^(-1)]
 ];

(*-----------------------------------------------*)
doStepResponsePlot[hs_, s_, hz_, z_, sampleTime_, nSamplesForPlotting_] :=
 Module[{contResponse, discreteResponse = 0, t, i, p1, p2, commonPlotOptions, maxTime},

  maxTime = nSamplesForPlotting * sampleTime;
  Do[
   {discreteResponse =
      discreteResponse + inverseZtransform[(1 / (1 - z^(-1))) hz[[i]], z, nSamplesForPlotting];
   }, {i, 1, Length[hz]}
  ];
```

```
  contResponse = Chop@InverseLaplaceTransform[hs / s, s, t];
  commonPlotOptions = {ImagePadding → {{70, 10}, {40, 30}}, Frame → True, ImageSize → Full,
    AxesOrigin → {0, 0}, ImageMargins → 1, AspectRatio → .35, Frame → True, TicksStyle → Directive[10]};

  p1 = Plot[Chop@contResponse, {t, 0, maxTime},
    Evaluate[commonPlotOptions], PlotRange → {{0, maxTime}, {0, 1.3}},
    FrameLabel → {{Text@Style["amplitude", 12], None}, {Text@Style["time (sec)", 12],
       Text@Style[Row[{Style["h", Italic], "(", Style["t", Italic], ") analog step response"}], 12]}}
   ];

  p2 = ListPlot[discreteResponse,
    Evaluate[commonPlotOptions], PlotRange → {All, {0, 1.3}},
    FrameLabel → {{Style["amplitude", 12], None},
      {Text@Style["sample number", 12], Text@Style[Row[{Style["h", Italic], "(",
           Style["n", Italic], ") discrete unit step response"}], 12]}}, Filling → Axis];

  GraphicsColumn[{p1, p2}, ImageSize → Full]
 ];

(*-----------------------------------------------*)
doImpulseResponsePlot[hs_, s_, hz_, z_, sampleTime_, nSamplesForPlotting_] :=
 Module[{contResponse, discreteResponse = 0, t, i, p1, p2, commonPlotOptions, maxTime},
  maxTime = nSamplesForPlotting * sampleTime;
  Do[
   {discreteResponse = discreteResponse + inverseZtransform[hz[[i]], z, nSamplesForPlotting];},
   {i, 1, Length[hz]}
  ];

  contResponse = Chop@InverseLaplaceTransform[hs, s, t];
  commonPlotOptions = {ImagePadding → {{70, 10}, {40, 30}}, ImageSize → Full,
    ImageMargins → 1, PlotRange → All, AxesOrigin → {0, 0}, Frame → True, AspectRatio → .35,
    Frame → True, TicksStyle → Directive[10]};

  p1 = Plot[Chop@contResponse, {t, 0, maxTime},
    Evaluate[commonPlotOptions],
    FrameLabel → {{Text@Style["amplitude", 12], None}, {Text@Style["time (sec)", 12], Text@
         Style[Row[{Style["h", Italic], "(", Style["t", Italic], ") analog impulse response"}], 12]}}
   ];

  p2 = ListPlot[discreteResponse, Evaluate[commonPlotOptions],
    FrameLabel → {{Text@Style["amplitude", 12], None},
      {Text@Style["sample number", 12], Text@Style[Row[{Style["h", Italic], "(",
           Style["n", Italic], ") discrete unit impulse response"}], 12]}}, Filling → Axis];

  GraphicsColumn[{p1, p2}, ImageSize → Full]
 ];

(*-----------------------------------------------*)
doRampResponsePlot[hs_, s_, hz_, z_, sampleTime_, nSamplesForPlotting_] :=
 Module[{contResponse, discreteResponse = 0, t, i, p1, p2, maxTime},
  maxTime = nSamplesForPlotting * sampleTime;
  Do[
   {
    discreteResponse =
      discreteResponse + inverseZtransform[hz[[i]] * (sampleTime * z / (z - 1)^2), z, nSamplesForPlotting];
   }, {i, 1, Length[hz]}];

  contResponse = Chop@InverseLaplaceTransform[hs * 1 / s^2, s, t];
  p1 = Plot[Chop@contResponse, {t, 0, maxTime}, ImagePadding → {{65, 10}, {40, 30}},
    ImageSize → Full, PlotRange → All, Frame → True,
    FrameLabel → {{Text@Style["amplitude", 12], None}, {Text@Style["time (sec)", 12],
       Text@Style[Row[{Style["h", Italic], "(", Style["t", Italic], ") analog ramp response"}], 12]}},
    AxesOrigin → {0, 0}, AspectRatio → .35, TicksStyle → Directive[10]];
```

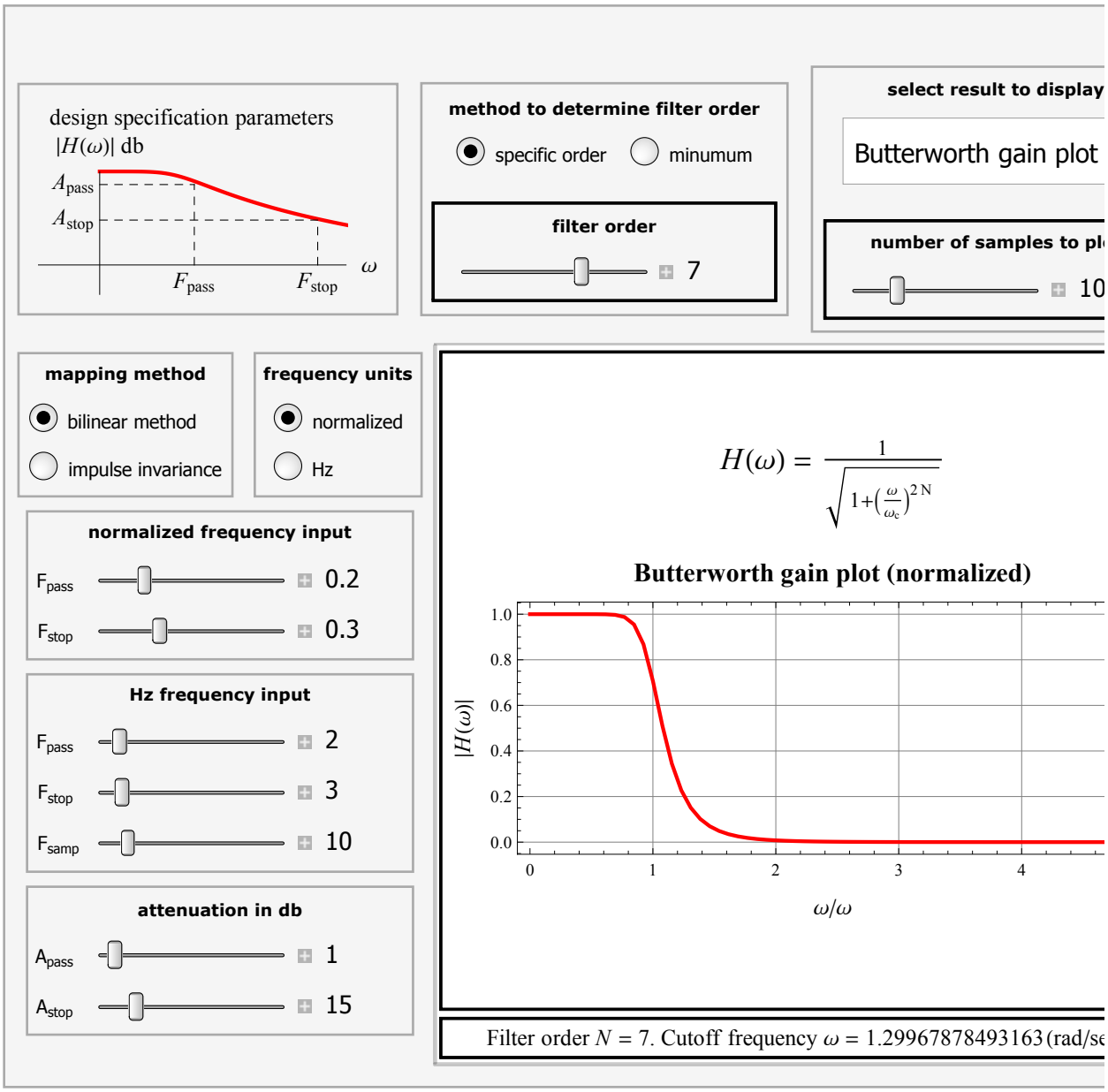
```
  p2 = ListPlot[discreteResponse, Frame → True, ImagePadding → All, ImageSize → Full,
    FrameLabel → {{Text@Style["amplitude", 12], None}, {Text@Style["sample number", 12], Text@Style[
         Row[{Style["h", Italic], "(", Style["t", Italic], ") discrete unit ramp response"}], 12]}},
    Filling → Axis, PlotRange → All, AxesOrigin → {0, 0}, AspectRatio → .45,
    TicksStyle → Directive[10]];

  GraphicsColumn[{p1, p2}, ImageSize → Full]
 ];

(*--------------------------------------------*)
doplotTypeButterworth[filterOrderFoundByDesign_, Ωc_] :=
 Module[{data, w, commonPlotOptions, maxX = 5 * Ωc, title},
  commonPlotOptions = {ImagePadding → {{35, 10}, {40, 90}}, ImageSize → Full,
    ImageMargins → 1, AxesOrigin → {0, 0}, Frame → True, AspectRatio → .40, Frame → True,
    PlotStyle → {Thick, Red}, TicksStyle → Directive[10], Joined → True, GridLines → Automatic};

  title = Column[{Text@Style[Row[{Style["H", Italic], "(ω)  = ", "1/Sqrt[1 + (ω/ωc)^(2 N)]"}], 16],

     Spacer[3], Style["Butterworth gain plot (normalized)", Bold, 14]}, Alignment → Center];

  data = Table[{w / Ωc, 1 / Sqrt[1 + (w / Ωc) ^ (2 filterOrderFoundByDesign)]}, {w, 0, maxX, .1}];
  ListPlot[data, AxesOrigin → {0, 0}, Evaluate@commonPlotOptions,
   FrameLabel → {{Text@Style[Row[{"|", Style["H", Italic], "(ω)|"}], 12], None},
     {Text@Style["ω/ω", 12], title}}, PlotRange → All, Frame → True, Joined → True]
 ];


(*--------------------------------------------*)
doPlotButterworthSpecs[fpass_, fstop_, apass_, astop_, Ωc_, filterOrderFoundByDesign_] := Module[
  {data, w, commonPlotOptions, maxX = 5 * Ωc, title},

  commonPlotOptions = {ImagePadding → All, ImageMargins → 1, ImageSize → 180,
    AspectRatio → .4, PlotStyle → {Thick, Red}, TicksStyle → Directive[10], Ticks → None,
    Joined → True};

  title = Style["design specification parameters", 12];
  data = Table[{w, 20 Log[10, 1 / Sqrt[1 + (w / Ωc) ^ (2 filterOrderFoundByDesign)]]}, {w, 0, maxX, .1}];

  Show[
   ListPlot[data, AxesOrigin → {0, -50}, Evaluate@commonPlotOptions,
    PlotRange → {{-1.5, 6}, {0, -66}},
    AxesLabel → {Text@Style["ω", 12], Text@Style[Row[{"|", Style["H", Italic], "(ω)| db"}], 12]},
    PlotLabel -> title, PlotRange → All, Joined → True,
    Epilog → {
      {Dashed, Line[{{0, apass}, {fpass, apass}}]},
      {Dashed, Line[{{fpass, -50}, {fpass, apass}}]},
      {Dashed, Line[{{0, astop}, {fstop, astop}}]},
      {Dashed, Line[{{fstop, -50}, {fstop, astop}}]}
     }],
   Graphics@Text[Style["Apass", 12], {-.65, apass}],
   Graphics@Text[Style["Astop", 12], {-.65, astop}],
   Graphics@Text[Style["Fpass", 12], {fpass, -60}],
   Graphics@Text[Style["Fstop", 12], {fstop, -60}]]];
 }
]
```



Caption

This Demonstration shows the implementation of a design for an infinite impulse response (IIR) low-pass digital filter. The input consists of the design specifications for the desired Butterworh analog filter. These specifications are used to determine the Butterworth (analog) filter transfer function $H(s)$, which is then mapped to the digital filter transfer function $H(z)$. The transformation of $H(s)$ to $H(z)$ can be made using a bilinear transformation or impulse invariance. A number of plots are made available to examine different aspects of the final design result, such as the locations of the poles and zeros, the response of the filter to common test input signals, and the display of $H(s)$ and $H(z)$ in symbolic form. Due to space limitations, the design is limited to a filter of order 10. You can get the design by asking for a minimum order filter or by specifying the filter order required. Both normalized units and hertz units can be used in the frequency specification.

Snapshots



Details (optional)

Carry out the design by inputting the filter specifications using the sliders. There are five filter specification parameters:

- F_{samp} is the sampling frequency;
- F_{pass} is the passband end frequency;
- F_{stop} is the start of stopband frequency;
- A_{pass} is the attenuation in db at F_{pass};
- A_{stop} is the attenuation in db at F_{stop}.

The plot at the top-left helps you see the locations of these parameters. These are standard digital filter design parameters that can be found in a number of digital signal processing textbooks.
You can do the design using normalized units (0 to 1) or in Hz. If you select normalized units, then 1 corresponds to the Nyquist

frequency and F_{pass} and F_{stop} will be between 0 and 1. Next select the analog-to-digital filter mapping method, which can be bilinear or impulse invariance. Next select the order of the filter. You can select a minimum order to be designed or specify the order yourself using the slider. The design supports a filter order up to 10. Finally, using the pulldown menu, select the plot type.

Some inputs will become disabled or enabled depending on what you select for the filter order and for the units. If you select a specific filter order, then only F_{pass} will be enabled, as F_{stop} is not required in this case; attenuation input is also disabled in this case, as attenuation A_{pass} will be fixed at $3\,\text{db}$. If you select normalized units, then the slider F_{samp} is disabled, as Nyquist is fixed at 1.

The design status is displayed below the main display window. Two possible design errors are detected and are displayed in red text. The first occurs if the filter order calculated is over 10 (when you select minimum filter order). In this case you need to change the design parameters until the filter order becomes less than 10. This error can occur if you select A_{stop} much larger than A_{pass}. The second error occurs if the gain is found to be too small. This can occur if you select F_{pass} and F_{stop} that are too close to each other.

Reference

[1] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1975, Chapter 5.

Control Suggestions (optional)

- [x] Resize Images
- [ ] Rotate and Zoom in 3D
- [ ] Drag Locators
- [ ] Create and Delete Locators
- [x] Slider Zoom
- [ ] Gamepad Controls
- [x] Automatic Animation
- [ ] Bookmark Animation

Search Terms (optional)

infinite impulse response
IIR
bilinear transformation
impulse invariance
butterworth filter
low pass filter
digital filter
analog filter
filter design
Z transform
Laplace transform

Related Links (optional)

Laplace Transform

Authoring Information

Contributed by: Nasser M. Abbasi