# The bodeplot package

version 1.2

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

April 18, 2024

## Contents

1 **Introduction** 2
   1.1 External Dependencies ........................................ 2
   1.2 Directory Structure ........................................... 2
   1.3 Limitations .................................................... 2

2 **TL;DR** 3

3 **Usage** 8
   3.1 Bode plots .................................................... 8
      3.1.1 Basic components up to first order ......................... 12
      3.1.2 Basic components of the second order ....................... 13
   3.2 Nyquist plots ................................................ 14
   3.3 Nichols charts ............................................... 16

4 **Implementation** 18
   4.1 Initialization ............................................... 18
   4.2 Parametric function generators for poles, zeros, gains, and delays . 20
   4.3 Second order systems .......................................... 21
   4.4 Commands for Bode plots ...................................... 23
      4.4.1 User macros ............................................. 23
      4.4.2 Internal macros ......................................... 29
   4.5 Nyquist plots ................................................. 33
      4.5.1 User macros ............................................. 33
      4.5.2 Internal commands ....................................... 36
   4.6 Nichols charts ................................................. 36
1 Introduction

Generate Bode, Nyquist, and Nichols plots for transfer functions in the canonical (TF) form
\[ G(s) = e^{-T_s} b_m s^m + \cdots + b_1 s + b_0 \]
\[ a_n s^n + \cdots + a_1 s + a_0 \]
and the zero-pole-gain (ZPK) form
\[ G(s) = K e^{-T_s} \frac{(s - z_1) (s - z_2) \cdots (s - z_m)}{(s - p_1) (s - p_2) \cdots (s - p_n)}. \]

In the equations above, \( b_m, \ldots, b_0 \) and \( a_n, \ldots, a_0 \) are real coefficients, \( T_s \geq 0 \) is the loop delay, \( z_1, \ldots, z_m \) and \( p_1, \ldots, p_n \) are complex zeros and poles of the transfer function, respectively, and \( K \in \mathbb{R} \) is the loop gain.

For transfer functions in the ZPK format in (2) with zero delay, this package also supports linear and asymptotic approximation of Bode plots.

By default, all phase plots use degrees as units. Use the \texttt{rad} package option or the optional argument \texttt{tikz/phase unit=rad} to generate plots in radians. The \texttt{phase unit} key accepts either \texttt{rad} or \texttt{deg} as inputs and needs to be added to the \texttt{tikzpicture} environment that contains the plots.

By default, frequency inputs and outputs are in radians per second. Use the \texttt{Hz} package option or the optional argument \texttt{tikz/frequency unit=Hz} to generate plots in hertz. The \texttt{frequency unit} key accepts either \texttt{rad} or \texttt{Hz} as inputs and needs to be added to the \texttt{tikzpicture} environment that contains the plots.

1.1 External Dependencies

By default, the package uses \texttt{gnuplot} to do all the computations. If \texttt{gnuplot} is not available, the \texttt{pgf} package option can be used to do the calculations using the native \texttt{pgf} math engine. Compilation using the \texttt{pgf} math engine is typically slower, but the end result should be the identical (other than phase wrapping in the TF form, see limitations below).

1.2 Directory Structure

Since version 1.0.8, the \texttt{bodeplot} package places all \texttt{gnuplot} temporary files in the working directory. The package option \texttt{declutter} restores the original behavior where the temporary files are placed in a folder called \texttt{gnuplot}.

1.3 Limitations

- Before version 1.2, in \texttt{pgf} mode, the package set \texttt{trig format plots} to \texttt{rad} globally. Version 1.2 onwards, this option is passed to each \texttt{addplot} command individually so that it does not affect other plots in the document. To roll back to the pre-1.2 behavior, load the package with \texttt{\usepackage[pgf]{bodeplot}[=2024-02-06]}.
- In \texttt{pgf} mode, Bode phase plots and Nichols charts in TF form wrap angles so that they are always between -180° and 180° or \(-\pi\) and \(-\pi\) radian. As such, these plots will show phase wrapping discontinuities. Since v1.1.1, in \texttt{gnuplot} mode, the package uses the \texttt{smooth unwrap} filter to correct wrapping discontinuities. As of now, I have not found a way to do this in \texttt{pgf} mode, any merge requests or ideas you may have are welcome! Since v1.1.4, you can redefine the \texttt{n@mod} macro using the commands \texttt{\makeatletter\renewcommand\n@mod@p}{\n@mod@p}\makeatother to wrap the phase between 0 and \(360°\) or 0 and \(2\pi\) radian. The commands \texttt{\makeatletter\renewcommand\n@mod@n}{\n@mod@n}\makeatother will wrap the phase between \(-360°\) and 0° or \(-2\pi\) and 0 radian.
- Use of the \texttt{declutter} option with other directory management tools such as a \texttt{tikzexternalize} prefix is not recommended.
2 TL;DR

All Bode plots in this section are for the transfer function (with and without a transport delay)

\[
G(s) = \frac{10^s(s + 0.1 + 0.5i)(s + 0.1 - 0.5i)}{(s + 0.5 + 10i)(s + 0.5 - 10i)} = \frac{s(10s^2 + 2s + 2.6)}{(s^2 + s + 100.25)}. \quad (3)
\]

Bode plot in ZPK format

Same Bode plot over the same frequency range but supplied in Hz, in TF format with arrow decoration, transport delay, unit, and color customization (the phase plot may show wrapping if the \texttt{pgf} package option is used)
Linear approximation with customization

Plot with delay and customization
Individual gain and phase plots with more customization

\begin{BodeMagPlot}
\begin{itemize}
\item axes\{height=2cm, width=4cm\}
\item \{0.01\} \{100\}
\end{itemize}
\addBodeZPKPlots\begin{itemize}
\item true\{black, thick\}
\item linear\{red, dashed, thick\}
\item asymptotic\{blue, dotted, thick\}
\end{itemize}\begin{itemize}
\item magnitude
\item \{z\{0, {-0.1, -0.5}, {-0.1, 0.5}\}, \}
\item \{p\{(-0.5, -10), (-0.5, 10)\}, \}
\item k\{10\}
\end{itemize}
\end{BodeMagPlot}

\begin{BodePhPlot}
\begin{itemize}
\item height=2cm, width=4cm,
\item ytick distance=90
\end{itemize}
\addBodeZPKPlots\begin{itemize}
\item true\{black, thick\}
\item linear\{red, dashed, thick\}
\item asymptotic\{blue, dotted, thick\}
\end{itemize}\begin{itemize}
\item phase
\item \{z\{0, {-0.1, -0.5}, {-0.1, 0.5}\}, \}
\item \{p\{(-0.5, -10), (-0.5, 10)\}, \}
\item k\{10\}
\end{itemize}
\end{BodePhPlot}

Nichols chart

\begin{ NicholsZPK\begin{itemize}
\item samples=1000
\end{itemize}
\begin{itemize}
\item z\{0, (-0.1, -0.5), (-0.1, 0.5)\}, \}
\item p\{(-0.5, -10), (-0.5, 10)\}, \}
\item k\{10\}, \}
\item d\{0.01\}
\end{itemize}
\end{ NicholsZPK}

Same Nichols chart in TF format (may show wrapping in \texttt{pgf} mode)

\begin{ NicholsTF\begin{itemize}
\item samples=1000
\end{itemize}
\begin{itemize}
\item num\{10, 2, 2, 6, 0\}, \}
\item den\{1, 1, 100.25\}, \}
\item d\{0.01\}
\end{itemize}
\end{ NicholsTF}
Multiple Nichols charts with customization

\begin{NicholsChart}
\begin{tikzpicture}
\begin{axis}[
    ytick distance=20,
    xtick distance=30
]
\addNicholsZPKChart [red,samples=1000] {
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10
}
\addNicholsZPKChart [blue,samples=1000] {
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/5
}
\end{axis}
\end{tikzpicture}
\end{NicholsChart}

Nyquist plot

\NyquistZPK[plot/{red,thick,samples=1000}]
{
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10
}
{-30}
{30}

Nyquist plot in TF format with arrows

\NyquistTF[plot/{\text{\textcolor{blue}{\textbf{\textbullet}}},thick,samples=1000, postaction=decorate, decoration={markings, mark=at position 0.1 and 0.9 with {\arrow[Stealth \[length=2mm, blue\]}}},]
{
    num/{10,2,2.6,0},
    den/{1,1,100.25}
}
{-30}
{180}
Multiple Nyquist plots with customization

\begin{NyquistPlot}{-30}{30}
\addNyquistZPKPlot [red,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10
}
\addNyquistZPKPlot [blue,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/5
}\end{NyquistPlot}

Nyquist plots with additional commands, using two different macros

\begin{NyquistPlot}[
  \tikz/
  \spy using outlines={
    circle,
    magnification=3,
    connect spies,
    size=2cm
  },
  \]
\addNyquistZPKPlot [blue,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/0.5
}\coordinate (spyon) at (axis cs:0,0);
\coordinate (spyat) at (axis cs:-22,5);
\spy [green] on (spyon) in node [fill=white] at (spyat);
\end{NyquistPlot}
3 Usage

In all the macros described here, the frequency limits supplied by the user are assumed to be in rad/s unless either the Hz package option is used or the optional argument \texttt{tikz/frequency unit=Hz} is supplied to the \texttt{tikzpicture} environment. All phase plots are generated in degrees unless either the \texttt{rad} package option is used or the optional argument \texttt{tikz/frequency unit=rad} is supplied to the \texttt{tikzpicture} environment.

3.1 Bode plots

\begin{verbatim}
\BodeZPK \BodeZPK [{(obj1/typ1/\{opt1\},obj2/typ2/\{opt2\},...)}]
\{\{z/\{zeros\},p/\{poles\},k/\{gain\},d/\{delay\}\}\}
\{\{min-freq\}\{max-freq\}\}
\end{verbatim}

Plots the Bode plot of a transfer function given in ZPK format using the \texttt{groupplot} environment. The three mandatory arguments include: (1) a list of tuples, comprised of the zeros, the poles, the gain, and the transport delay of the transfer function, (2) the lower end of the frequency range for the \(x\)-axis, and (3) the higher end of the frequency range for the \(x\)-axis. The zeros and the poles are complex numbers, entered as a comma-separated list of comma-separated lists, of the form \{\{real part 1,imaginary part 1\}, \{real part 2,imaginary part 2\},...\}. If the imaginary part is not provided, it is assumed to be zero.

The optional argument is comprised of a comma separated list of tuples, either \texttt{obj/typ/\{opt\}}, or \texttt{obj/\{opt\}}, or just \texttt{\{opt\}}. Each tuple passes options to different \texttt{pgfplots} macros that generate the group, the axes, and the plots according to:

- Tuples of the form \texttt{obj/typ/\{opt\}}:
  - \texttt{plot/typ/\{opt\}}: modify plot properties by adding options \texttt{\{opt\}} to the \texttt{\addplot} macro for the magnitude plot if \texttt{typ} is \texttt{mag} and the phase plot if \texttt{typ} is \texttt{ph}.
  - \texttt{axes/typ/\{opt\}}: modify axis properties by adding options \texttt{\{opt\}} to the \texttt{\nextgroupplot} macro for the magnitude plot if \texttt{typ} is \texttt{mag} and the phase plot if \texttt{typ} is \texttt{ph}.
  - \texttt{commands/typ/\{opt\}}: add any valid TikZ commands (including the the parametric function generator macros in this package, such as \texttt{\addBodeZPKPlots}, \texttt{\addBodeTFPlot}, and \texttt{\addBodeComponentPlot}) to the magnitude plot if \texttt{typ} is \texttt{mag} and the phase plot if \texttt{typ} is \texttt{ph}. The commands passed to \texttt{\{opt\}} need to be valid TikZ commands, separated by semicolons as usual. For example, a TikZ command is used in the description of the \texttt{\BodeTF} macro below to mark the gain crossover frequency on the Bode Magnitude plot.

- Tuples of the form \texttt{obj/\{opt\}}:
  - \texttt{plot/\{opt\}}: adds options \texttt{\{opt\}} to \texttt{\addplot} macros for both the magnitude and the phase plots.
  - \texttt{axes/\{opt\}}: adds options \texttt{\{opt\}} to \texttt{\nextgroupplot} macros for both the magnitude and the phase plots.
  - \texttt{group/\{opt\}}: adds options \texttt{\{opt\}} to the \texttt{\groupplot} environment.
  - \texttt{tikz/\{opt\}}: adds options \texttt{\{opt\}} to the \texttt{\tikzpicture} environment.
  - \texttt{approx/linear}: plots linear approximation.
  - \texttt{approx/asymptotic}: plots asymptotic approximation.

- Tuples of the form \texttt{\{opt\}} add all of the supplied options to \texttt{\addplot} macros for both the magnitude and the phase plots.
Figure 1: Output of the \BodeZPK macro.

The options \{opt\} can be any key=value options that are supported by the pgfplots macros they are added to.

For example, given a transfer function

\[ G(s) = \frac{10s(s + 0.1 + 0.5i)(s + 0.1 - 0.5i)}{(s + 0.5 + 10i)(s + 0.5 - 10i)}. \] (4)

its Bode plot over the frequency range [0.01, 100] can be generated using

\[
\BodeZPK \{\text{blue,thick}\}
\{z/\{0,\{-0.1,-0.5\},\{-0.1,0.5\}\},p/\{\{-0.5,-10\},\{-0.5,10\}\},k/10\}
\{0.01\}\{100\}
\]

which generates the plot in Figure 1. In this example, a delay is not specified, so it is assumed to be zero. A gain is not specified, so it is assumed to be 1. A single comma-separated list of options \{blue,thick\} is passed, so it is passed on to the \addplot commands in both the magnitude and the phase plots. The default plots are thick black lines and each of the axes, excluding ticks and labels, are 5cm wide and 2.5cm high.

The width and the height, along with other properties of the plots, the axes, and the group can be customized using native pgf keys. For example, a linear approximation of the Bode plot with customization of the plots, the axes, and the group can be generated using

\[
\BodeZPK[\
\text{plot/mag/\{red,thick\}},
\text{plot/ph/\{blue,thick\}},
\text{axes/mag/\{ytick distance=40,xmajorticks=true,xlabel={Frequency (rad/s)}\}},
\text{axes/ph/\{ytick distance=90\}},
\text{group/\{group style=\{group size=2 by 1, horizontal sep=2cm, width=4cm, height=2cm\}\}},
\text{approx/linear}]
\{z/\{0,\{-0.1,-0.5\},\{-0.1,0.5\}\},p/\{\{-0.5,-10\},\{-0.5,10\}\},k/10\}
\{0.01\}\{100\}
\]

which generates the plot in Figure 2.

\BodeTF \{\text{obj1/typ1/\{\{opt1\}\},obj2/typ2/\{\{opt2\}\},...}\}
\{\{\{num/\{\{coeffs\}\},den/\{\{coeffs\}\},d/\{\{delay\}\}\}\,
\{\{min-freq\}\}\{\{max-freq\}\}\}

Plots the Bode plot of a transfer function given in TF format. The three mandatory arguments include: (1) a list of tuples comprised of the coefficients in the numerator and the denominator of the transfer function and the transport delay, (2) the lower end of the frequency range for the x-axis, and (3) the higher end of the frequency range for the x-axis. The coefficients are entered as a comma-separated list, in order
Figure 2: Customization of the default \texttt{BodeZPK} macro.

Figure 3: Output of the \texttt{BodeTF} macro with an optional TikZ command used to mark the gain crossover frequency.

from the highest degree of $s$ to the lowest, with zeros for missing degrees. The optional arguments are the same as \texttt{BodeZPK}, except that linear/asymptotic approximation is not supported, so \texttt{approx/...} is ignored.

For example, given the same transfer function as (4) in TF form and with a small transport delay,

$$G(s) = e^{-0.01s} \frac{s(10s^2 + 2s + 2.6)}{(s^2 + s + 100.25)},$$

its Bode plot over the frequency range $[0.01, 100]$ can be generated using

\begin{verbatim}
\BodeTF[%
  commands/mag/{\node at (axis cs: 2.1,0) [circle,fill,inner sep=0.05cm, label=below:{$\omega_{gc}$}]{};}
  ]
  {num/{10,2,2.6,0},den/{1,1,100.25},d/0.01}
  {0.01}{100}
\end{verbatim}

which generates the plot in Figure 3. Note the 0 added to the numerator coefficients to account for the fact that the numerator does not have a constant term in it. Note the semicolon after the TikZ command passed to the \texttt{commands} option.

\begin{verbatim}
BodeMagPlot(\begin{BodeMagPlot}
\addBode...
\end{BodeMagPlot})
\end{verbatim}

The \texttt{BodeMagPlot} environment works in conjunction with the parametric function generator macros \texttt{addBodeZPKPlots}, \texttt{addBodeTFPlot}, and \texttt{addBodeComponentPlot}, intended to be used for magnitude plots. The optional argument is comprised of a comma separated list of tuples, either \texttt{obj/{opt}} or just \texttt{opt}. Each tuple passes options to different \texttt{pgfplots} macros that generate the axes and the plots according to:
• Tuples of the form \( \text{obj/\{opt\}} \):
  
  - \( \text{tikz/\{opt\}} \): modify picture properties by adding options \( \{\text{opt}\} \) to the `tikzpicture` environment.
  
  - \( \text{axes/\{opt\}} \): modify axis properties by adding options \( \{\text{opt}\} \) to the `semilogaxis` environment.
  
  - \( \text{commands/\{opt\}} \): add any valid TikZ commands inside `semilogaxis` environment. The commands passed to \( \text{opt} \) need to be valid TikZ commands, separated by semicolons as usual.

• Tuples of the form \( \{\text{opt}\} \) are passed directly to the `semilogaxis` environment. The frequency limits are translated to the x-axis limits and the domain of the `semilogaxis` environment. Example usage in the description of \( \text{addBodeZPKPlots} \), \( \text{addBodeTFPlot} \), and \( \text{addBodeComponentPlot} \).

\[ \text{BodePhPlot} \]
\[ \langle \text{obj1/\{opt1\}}, \text{obj2/\{opt2\}}, ... \rangle \]

\( \text{BodePhPlot}(\text{env.}) \)
\[ \{<\text{min-frequency}>\}, \{<\text{max-frequency}>\} \]

`\addBodePhPlot` Intended to be used for phase plots, otherwise same as the `BodeMagPlot` environment.

\[ \text{addBodeZPKPlots} \]
\[ \langle \text{approx1/\{opt1\}}, \text{approx2/\{opt2\}}, ... \rangle \]

\( \text{addBodeZPKPlots}(\langle \text{plot-type} \rangle) \)
\[ \{<\text{num}>\}, \{<\text{den}>\}, \{<\text{d}>\} \]

Generates the appropriate parametric functions and supplies them to multiple `addplot` macros, one for each `approx/{\text{opt}}` pair in the optional argument. If no optional argument is supplied, then a single `addplot` command corresponding to a thick true Bode plot is generated. If an optional argument is supplied, it needs to be one of `true/{\text{opt}}`, `linear/{\text{opt}}`, or `asymptotic/{\text{opt}}`. This macro can be used inside any `semilogaxis` environment as long as a domain for the x-axis is supplied through either the `approx/{\text{opt}}` interface or directly in the optional argument of the `semilogaxis` environment. Use with the `BodePlot` environment supplied with this package is recommended. The second mandatory argument, `plot-type` is either `magnitude` or `phase`. If it is not equal to `phase`, it is assumed to be `magnitude`. The last mandatory argument is the same as `BodeZPK`.

For example, given the transfer function in (4), its linear, asymptotic, and true Bode plots can be superimposed using

\[ \begin{BodeMagPlot}(\text{env.}) \]
\[ \{0.01\}, \{100\} \]

\[ \text{addBodeTFPlot}(\langle \text{plot-options} \rangle) \]
\[ \{<\text{plot-type}>\} \]
\[ \{<\text{num}>\}, \{<\text{den}>\}, \{<\text{d}>\} \]

Generates a single parametric function for either Bode magnitude or phase plot of a transfer function in TF form. The generated parametric function is passed to the
\addplot macro. This macro can be used inside any semilogaxis environment as long as a domain for the x-axis is supplied through either the plot-options interface or directly in the optional argument of the container semilogaxis environment. Use with the \BodePlot environment supplied with this package is recommended. The second mandatory argument, plot-type is either magnitude or phase. If it is not equal to phase, it is assumed to be magnitude. The last mandatory argument is the same as \BodeTF.

\addBodeComponentPlot \addBodeComponentPlot[⟨plot-options⟩]⟨plot-command⟩
Generates a single parametric function corresponding to the mandatory argument plot-command and passes it to the \addplot macro. The plot command can be any parametric function that uses $t$ as the independent variable. The parametric function must be gnuplot compatible (or pgfplots compatible if the package is loaded using the pgf option, with angles passed to trigonometric functions in radian). The intended use of this macro is to plot the parametric functions generated using the basic component macros described in Section 3.1.1 below.

3.1.1 Basic components up to first order

\TypeFeatureApprox \TypeFeatureApprox{⟨real-part⟩}{⟨imaginary-part⟩}
This entry describes 20 different macros of the form \TypeFeatureApprox that take the real part and the imaginary part of a complex number as arguments. The Type in the macro name should be replaced by either Mag or Ph to generate a parametric function corresponding to the magnitude or the phase plot, respectively. The Feature in the macro name should be replaced by one of K, Pole, Zero, or Del, to generate the Bode plot of a gain, a complex pole, a complex zero, or a transport delay, respectively. If the Feature is set to either K or Del, the imaginary-part mandatory argument is ignored. The Approx in the macro name should either be removed, or it should be replaced by Lin or Asymp to generate the true Bode plot, the linear approximation, or the asymptotic approximation, respectively. If the Feature is set to Del, then Approx has to be removed. For example,

- \MagK{k}{0} or \MagK{k}{400} generates a parametric function for the true Bode magnitude of $G(s) = k$
- \PhPoleLin{a}{b} generates a parametric function for the linear approximation of the Bode phase of $G(s) = \frac{1}{s-a-i\beta}$.
- \PhDel{T}{200} or \PhDel{T}{0} generates a parametric function for the Bode phase of $G(s) = e^{-Ts}$.

All 20 of the macros defined by combinations of Type, Feature, and Approx, and any gnuplot (or pgfplots if the pgf class option is loaded) compatible function of the 20 macros can be used as plot-command in the \addBodeComponentPlot macro. This is sufficient to generate the Bode plot of any rational transfer function with delay. For example, the Bode phase plot in Figure 4 can also be generated using:

\begin{BodePhPlot}[ytick distance=90]{0.01}{100}
\addBodeComponentPlot[black,thick]{%
Figure 5: Superimposed approximate and true Bode Phase plot using the BodePhPlot environment, the \addBodeComponentPlot macro, and several macros of the \TypeFeatureApprox form.

\begin{BodePhPlot}
\PhZero{0}{0} + \PhZero{-0.1}{-0.5} + \PhZero{-0.1}{0.5} + \\
\PhPole{-0.5}{-10} + \PhPole{-0.5}{10} + \PhK{10}{0}\}
\addBodeComponentPlot[red,dashed,thick] {%
\PhZeroLin{0}{0} + \PhZeroLin{-0.1}{-0.5} + \PhZeroLin{-0.1}{0.5} + \\
\PhPoleLin{-0.5}{-10} + \PhPoleLin{-0.5}{10} + \PhKLin{10}{20}}
\addBodeComponentPlot[blue,dotted,thick] {%
\PhZeroAsymp{0}{0} + \PhZeroAsymp{-0.1}{-0.5} + \PhZeroAsymp{-0.1}{0.5} + \\
\PhPoleAsymp{-0.5}{-10} + \PhPoleAsymp{-0.5}{10} + \PhKAsymp{10}{40}}
\end{BodePhPlot}

which gives us the plot in Figure 5.

3.1.2 Basic components of the second order

\TypeSOFeatureApprox{\langle a_1 \rangle}{\langle a_0 \rangle}

This entry describes 12 different macros of the form \TypeSOFeatureApprox that take the coefficients $a_1$ and $a_0$ of a general second order system as inputs. The Feature in the macro name should be replaced by either Poles or Zeros to generate the Bode plot of $G(s) = \frac{1}{s^2 + a_1 s + a_0}$ or $G(s) = s^2 + a_1 s + a_0$, respectively. The Type in the macro name should be replaced by either Mag or Ph to generate a parametric function corresponding to the magnitude or the phase plot, respectively. The Approx in the macro name should either be removed, or it should be replaced by Lin or Asymp to generate the true Bode plot, the linear approximation, or the asymptotic approximation, respectively.

\MagSOFeaturePeak{\langle draw-options \rangle}{\langle a_1 \rangle}{\langle a_0 \rangle}

This entry describes 2 different macros of the form \MagSOFeaturePeak that take the the coefficients $a_1$ and $a_0$ of a general second order system as inputs, and draw a resonant peak using the \draw TikZ macro. The Feature in the macro name should be replaced by either Poles or Zeros to generate a peak for poles and a valley for zeros, respectively. For example, the command

\begin{BodeMagPlot}[xlabel={}]{0.1}{10}
\addBodeComponentPlot[red,dashed,thick]{\MagSOPoles{0.2}{1}}
\addBodeComponentPlot[black,thick]{\MagSOPolesLin{0.2}{1}}
\MagSOPolesPeak[thick]{0.2}{1}
\end{BodeMagPlot}

generates the plot in Figure 6.

\TypeCSFeatureApprox{\langle \zeta \rangle}{\langle \omega_n \rangle}

This entry describes 12 different macros of the form \TypeCSFeatureApprox that take the damping ratio, $\zeta$, and the natural frequency, $\omega_n$ of a canonical second order system as inputs. The Type in the macro name should be replaced by either Mag or Ph to generate a parametric function corresponding to the magnitude or the phase plot, respectively. The Feature in the macro name should be replaced by either Poles or Zeros to generate the Bode plot of $G(s) = \frac{1}{s^2 + 2\zeta \omega_n s + \omega_n^2}$ or $G(s) = s^2 + 2\zeta \omega_n s + \omega_n^2$, respectively. The Approx in the macro name should either be removed, or it should be
replaced by \texttt{Lin} or \texttt{Asymp} to generate the true Bode plot, the linear approximation, or the asymptotic approximation, respectively.

\texttt{MagCSFeaturePeak} \quad | \quad \texttt{MagCSFeaturePeak}[\langle draw-options \rangle,\langle zeta \rangle,\langle omega-n \rangle]

This entry describes 2 different macros of the form \texttt{MagCSFeaturePeak} that take the damping ratio, \( \zeta \), and the natural frequency, \( \omega_n \) of a canonical second order system as inputs, and draw a resonant peak using the \texttt{draw} TikZ macro. The \texttt{Feature} in the macro name should be replaced by either \texttt{Poles} or \texttt{Zeros} to generate a peak for poles and a valley for zeros, respectively.

\texttt{MagCCFeaturePeak} \quad | \quad \texttt{MagCCFeaturePeak}[\langle draw-options \rangle,\langle real-part \rangle,\langle imaginary-part \rangle]

This entry describes 2 different macros of the form \texttt{MagCCFeaturePeak} that take the real and imaginary parts of a pair of complex conjugate poles or zeros as inputs, and draw a resonant peak using the \texttt{draw} TikZ macro. The \texttt{Feature} in the macro name should be replaced by either \texttt{Poles} or \texttt{Zeros} to generate a peak for poles and a valley for zeros, respectively.

3.2 Nyquist plots

\texttt{NyquistZPK} \quad \texttt{NyquistZPK} [\langle plot/\{\opt\},axes/\{\opt\} \rangle]
\{\langle z/\{\texttt{zeros}\},p/\{\texttt{poles}\},k/\{\texttt{gain}\},d/\{\texttt{delay}\} \rangle\}
\{\langle \texttt{min-freq} \rangle\}{\langle \texttt{max-freq} \rangle}

Plots the Nyquist plot of a transfer function given in ZPK format with a thick red + marking the critical point (-1,0). The mandatory arguments are the same as \texttt{BodeZPK}. Since there is only one plot in a Nyquist diagram, the \texttt{typ} specifier in the optional argument tuples is not needed. As such, the supported optional argument tuples are \texttt{plot/\{\opt\}}, which passes \{\opt\} to \texttt{addplot, axes/\{\opt\}}, which passes \{\opt\} to the \texttt{axis} environment, and \texttt{tikz/\{\opt\}}, which passes \{\opt\} to the \texttt{tikzpicture} environment. Asymptotic/linear approximations are not supported in Nyquist plots. If just \{\opt\} is provided as the optional argument, it is interpreted as \texttt{plot/\{\opt\}}. Arrows to indicate the direction of increasing \( \omega \) can be added by adding \texttt{usetikzlibrary{decorations.markings}} and \texttt{usetikzlibrary{arrows.meta}} to the preamble and then passing a tuple of the form

\texttt{plot/\{postaction=decorate,decoration=\{markings,}
mark=\{\texttt{between positions 0.1 and 0.9 step 5em with \%
\texttt{\arrow{\texttt{Stealth}}||[length=2mm,\textcolor{blue}{\texttt{blue}}]}\}\}}}\}

\textbf{Caution:} with a high number of samples, adding arrows in this way may cause the error message \texttt{Dimension too big}.

For example, the command

\texttt{NyquistZPK[plot/\{red,thick,samples=2000\},axes/\{blue,thick\}]
\{z/\{0,{-0.1,-0.5},{-0.1,0.5}\},p/\{\{-0.5,-10\},\{-0.5,10\}\},k/10\}
\{-30\}{30}}

generates the Nyquist plot in Figure 7.

\texttt{NyquistTF} \quad | \quad \texttt{NyquistTF}[\langle plot/\{\opt\},axes/\{\opt\} \rangle]
\{\langle num/\{\texttt{coeffs}\},den/\{\texttt{coeffs}\},d/\{\texttt{delay}\} \rangle\}
\{\langle \texttt{min-freq} \rangle\}{\langle \texttt{max-freq} \rangle}
Nyquist plot of a transfer function given in TF format. Same mandatory arguments as \texttt{\textbackslash BodeTF} and same optional arguments as \texttt{\textbackslash NyquistZPK}. For example, the command

\begin{verbatim}
\NyquistTF[plot/{green,thick,samples=500,postaction=decorate, 
  decoration={markings, 
    mark=between positions 0.1 and 0.9 step 5em 
    with{\arrow{Stealth[length=2mm, blue]}}}} 
{num/{10,2,2.6,0},den/{1,1,100.25}} 
{-30}{30} 
\end{verbatim}

generates the Nyquist plot in Figure 8.

The \texttt{\textbackslash NyquistPlot} environment works in conjunction with the parametric function generator macros \texttt{\textbackslash addNyquistZPKPlot} and \texttt{\textbackslash addNyquistTFPlot}. The optional argument is comprised of a comma separated list of tuples, either \texttt{obj/\{opt\}} or just \texttt{\{opt\}}. Each tuple passes options to different \texttt{pgfplots} macros that generate the axes and the plots according to:

- Tuples of the form \texttt{obj/\{opt\}}:
- `tikz/{opt}`: modify picture properties by adding options `{opt}` to the `tikzpicture` environment.
- `axes/{opt}`: modify axis properties by adding options `{opt}` to the `axis` environment.
- `commands/{opt}`: add any valid TikZ commands inside `axis` environment. The commands passed to `{opt}` need to be valid TikZ commands, separated by semicolons as usual.

- Tuples of the form `{opt}` are passed directly to the `axis` environment.

The frequency limits are translated to the x-axis limits and the domain of the `axis` environment.

\addNyquistZPKPlot \addNyquistZPKPlot[{plot-options}]
\{\{z/\{zeros\},p/\{poles\},k/\{gain\},d/\{delay\}\}\}
Generates a two parametric functions for the magnitude and the phase of a transfer function in ZPK form. The generated magnitude and phase parametric functions are converted to real and imaginary part parametric functions and passed to the `addplot` macro. This macro can be used inside any `axis` environment as long as a domain for the x-axis is supplied through either the `plot-options` interface or directly in the optional argument of the container `axis` environment. Use with the `NyquistPlot` environment supplied with this package is recommended. The mandatory argument is the same as `BodeZPK`.

\addNyquistTFPlot \addNyquistTFPlot[{plot-options}]
\{\{num/\{coeffs\},den/\{coeffs\},d/\{delay\}\}\}
Similar to `\addNyquistZPKPlot`, with a transfer function input in the TF form.

### 3.3 Nichols charts

\NicholsZPK \NicholsZPK[{plot/\{opt\},axes/\{opt\}}]
\{\{z/\{zeros\},p/\{poles\},k/\{gain\},d/\{delay\}\}\}
\{\{min-freq\}\{max-freq\}\}
Nichols chart of a transfer function given in ZPK format. Same arguments as `\NyquistZPK`.

\NicholsTF \NicholsTF[{plot/\{opt\},axes/\{opt\}}]
\{\{num/\{coeffs\},den/\{coeffs\},d/\{delay\}\}\}
\{\{min-freq\}\{max-freq\}\}
Nichols chart of a transfer function given in TF format. Same arguments as `\NyquistTF`. For example, the command

\NicholsTF[plot/\{green,thick,samples=2000\}]
\{num/\{10,2,2.6,0\},den/\{1,1,100.25\},d/0.01\}
\{0.001\{10\}

generates the Nichols chart in Figure 9.

\NicholsChart (env.) \begin{NicholsChart}\{\{obj1/\{opt1\},obj2/\{opt2\}\},...\}\}
\{\{min-frequency\}\{max-frequency\}\}
\end{NicholsChart}

The `NicholsChart` environment works in conjunction with the parametric function generator macros `\addNicholsZPKChart` and `\addNicholsTFChart`. The optional argument is comprised of a comma separated list of tuples, either `obj/\{opt\}` or just `{opt}`. Each tuple passes options to different `pgfplots` macros that generate the axes and the plots according to:

- Tuples of the form `obj/\{opt\}`:
  - `tikz/\{opt\}`: modify picture properties by adding options `{opt}` to the `tikzpicture` environment.
  - `axes/\{opt\}`: modify axis properties by adding options `{opt}` to the `axis` environment.
\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{nyquist_zpkExample}
\caption{Output of the \texttt{NyquistZPK} macro.}
\end{figure}

- \texttt{commands/\{opt\}}: add any valid TikZ commands inside \texttt{axis} environment. The commands passed to \texttt{opt} need to be valid TikZ commands, separated by semicolons as usual.

- Tuples of the form \texttt{\{opt\}} are passed directly to the \texttt{axis} environment.

The frequency limits are translated to the x-axis limits and the domain of the \texttt{axis} environment.

\begin{verbatim}
\addNicholsZPKChart \addNicholsZPKChart\{\textit{plot-options}\}
\{z/\{\textit{zeros}\},p/\{\textit{poles}\},k/\{\textit{gain}\},d/\{\textit{delay}\}\}
\end{verbatim}
Generates a two parametric functions for the magnitude and the phase of a transfer function in ZPK form. The generated magnitude and phase parametric functions are passed to the \texttt{addplot} macro. This macro can be used inside any \texttt{axis} environment as long as a domain for the x-axis is supplied through either the \texttt{plot-options} interface or directly in the optional argument of the container \texttt{axis} environment. Use with the \texttt{NicholsChart} environment supplied with this package is recommended. The mandatory argument is the same as \texttt{BodeZPK}.

\begin{verbatim}
\addNicholsTFChart \addNicholsTFChart\{\textit{plot-options}\}
\{num/\{\textit{coeffs}\},den/\{\textit{coeffs}\},d/\{\textit{delay}\}\}
\end{verbatim}
Similar to \texttt{addNicholsZPKChart}, with a transfer function input in the TF form.
4 Implementation

4.1 Initialization

We start by processing the class options.

\newif\if@pgfarg\@pgfargfalse
\DeclareOption{pgf}{
  \@pgfargtrue
}
\newif\if@declutterarg\@declutterargfalse
\DeclareOption{declutter}{
  \@declutterargtrue
}
\newif\if@radarg\@radargfalse
\DeclareOption{rad}{
  \@radargtrue
}
\newif\if@hzarg\@hzargfalse
\DeclareOption{Hz}{
  \@hzargtrue
}
\ProcessOptions\relax

Then, we define new macros to unify \texttt{pgfplots} and \texttt{gnuplot}. New macros are defined for the \texttt{pow} and \texttt{mod} functions to address differences between the two math engines.

\newcommand{\n@mod}[2]{(#1)-((\text{round}((#1)/(#2)))*(#2))}
\newcommand{\n@mod@p}[2]{(#1)-((\text{floor}((#1)/(#2)))*(#2))}
\newcommand{\n@mod@n}[2]{(#1)-((\text{floor}((#1)/(#2))+1)*(#2))}
\if@pgfarg
\newcommand{\n@pow}[2]{(#1)^{(#2)}}
\else
\newcommand{\n@pow}[2]{(#1)**(#2)}
\fi

Then, we create a counter so that a new data table is generated and for each new plot. If the plot macros have not changed, the tables, once generated, can be reused by \texttt{gnuplot}, which reduces compilation time. The \texttt{declutter} option is used to enable the \texttt{gnuplot} directory to declutter the working directory.

\newcounter{gnuplot@id}
\setcounter{gnuplot@id}{0}
\if@declutterarg
\edef\bodeplot@prefix{gnuplot/\jobname}
\else
\edef\bodeplot@prefix{\jobname}
\fi
\tikzset{
  gnuplot@prefix/.style={
    id=arabic{gnuplot@id},
    prefix=\bodeplot@prefix
  }
}\if@windows\else
\if@declutterarg
\immediate\write18{mkdir -p gnuplot}
\fi
\fi
\if@babel
\shorthand@list
\newif\if@babel\@babelfalse
\AtBeginDocument{\relax
\gdef\shorthand{}%
\@ifpackageloaded{babel}{%
@babeltrue
\let\shorthand@list@empty
\def\do#1{%
\begingroup
\lccode’~='#1\relax
\lowercase{\ifbabelshorthand~{\g@addto@macro\shorthand@list{~}}{}\endgroup}
\dospecials
}}{}

\bode@style
Default axis properties for all plot macros are collected in this \texttt{pgf} style.
\pgfplotsset{\bode@style/.style = {
    label style={font=\footnotesize},
    tick label style={font=\footnotesize},
    grid=both,
    major grid style={color=gray!80},
    minor grid style={color=gray!20},
    x label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
    y label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
    scale only axis,
    samples=200,
    width=5cm,
    log basis x=10
}}

\freq@filter
These macros handle the \texttt{Hz} and \texttt{rad} class options and two new \texttt{pgf} keys named \texttt{frequency unit} and \texttt{phase unit} for conversion of frequency and phase units, respectively.
\pgfplotsset{\freq@filter/.style = {}}
\def\freq@scale{1}
\pgfplotsset{\freq@label/.style = {xlabel = {Frequency (rad/s)}}}
\pgfplotsset{\ph@x@label/.style = {xlabel={Phase (deg)}}}
\pgfplotsset{\ph@y@label/.style = {ylabel={Phase (deg)}}}
\def\ph@scale{180/pi}
\if@radarg
\pgfplotsset{\ph@y@label/.style = {ylabel={Phase (rad)}}}
\pgfplotsset{\ph@x@label/.style = {xlabel={Phase (rad)}}}
\def\ph@scale{1}
\fi
\if@hzarg
\def\freq@scale{2*pi}
\pgfplotsset{\freq@label/.style = {xlabel = {Frequency (Hz)}}}
\if@pgfarg
\pgfplotsset{\freq@filter/.style = {x filter/.expression={x-log10(2*pi)}}}
\fi
\fi
\tikzset{
    phase unit/.initial={deg},
    phase unit/.default={deg},
    phase unit/.is choice,
    phase unit/deg/.code={
        \renewcommand{\ph@scale}{180/pi}
        \pgfplotsset{\ph@x@label/.style = {xlabel={Phase (deg)}}}
        \pgfplotsset{\ph@y@label/.style = {ylabel={Phase (deg)}}}
    },
    phase unit/rad/.code={
        \renewcommand{\ph@scale}{1}
    }
}
4.2 Parametric function generators for poles, zeros, gains, and delays.

All calculations are carried out assuming that frequency inputs are in \textit{rad/s}. Magnitude outputs are in \textit{dB} and phase outputs are in degrees or radians, depending on the value of \texttt{ph\@scale}.

\begin{verbatim}
\newcommand*{\MagK}[2]{(20*\text{log10}(\text{abs}(#1)))}
\newcommand*{\MagKAsymp}{\MagK}
\newcommand*{\MagKLin}{\MagK}
\newcommand*{\PhK}[2]{((#1<0?-\pi:0)*\text{ph\@scale})}
\newcommand*{\PhKAsymp}{\PhK}
\newcommand*{\PhKLin}{\PhK}
\end{verbatim}

\texttt{\MagK} True, linear, and asymptotic magnitude and phase parametric functions for a pure gain \(G(s) = k + 0i\). The macros take two arguments corresponding to real and imaginary part of the gain to facilitate code reuse between delays, gains, poles, and zeros, but only real gains are supported. The second argument, if supplied, is ignored. The second argument, if supplied, is ignored.

\begin{verbatim}
\newcommand*{\MagDel}[2]{0}
\newcommand*{\PhDel}[2]{(-#1*t*\text{ph\@scale})}
\end{verbatim}

\texttt{\MagPole} These macros are the building blocks for most of the plotting functions provided by this package. We start with Parametric function for the true magnitude of a complex pole.

\begin{verbatim}
\newcommand*{\MagPole}[2]{(-20*\text{log10}(\sqrt{#1^2 + (t - (#2))^2}))}
\end{verbatim}

Parametric function for linear approximation of the magnitude of a complex pole.

\begin{verbatim}
\newcommand*{\MagPoleLin}[2]{((-20*\text{log10}(\sqrt{\text{pow(#1,2)} + \text{pow(t - (#2),2)})))}
\end{verbatim}

Parametric function for asymptotic approximation of the magnitude of a complex pole, same as linear approximation.

\begin{verbatim}
\newcommand*{\MagPoleAsymp}{\MagPoleLin}
\end{verbatim}
Parametric function for the true phase of a complex pole.
\begin{verbatim}
\newcommand*{\PhPole}[2]{((#1 > 0 ? (#2 > 0 ?
\@mod@p{-\atan2((t - (#2)),-(#1))}{2*pi}) :
\@atan2((t - (#2)),-(#1)))) :
\@atan2((t - (#2)),-(#1))))\*\ph@scale}
\end{verbatim}

Parametric function for linear approximation of the phase of a complex pole.
\begin{verbatim}
\newcommand*{\PhPoleLin}[2]{((abs(#1)+abs(#2) == 0 ? -pi/2 :
(t < (sqrt(\@pow{#1}{2} + \@pow{#2}{2}) /
\@pow{10}{sqrt((4*\@pow{#1}{2})/
\@pow{#1}{2} + \@pow{#2}{2}))}) ?
(-atan2(-(#2),-(#1)) :
(t >= (sqrt(\@pow{#1}{2} * \@pow{10}{sqrt((4*\@pow{#1}{2})/
\@pow{#1}{2} + \@pow{#2}{2}))})))*((#1>0?(#1>0?3*pi/2:-pi/2):-pi/2) :
(-atan2(-(#2),-(#1)) + (log10((#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) + atan2(-
(#2),-(#1))))/}
(\log10(\@pow{#1}{2}/\@pow{10}{abs(#1)})))))))*\ph@scale}
\end{verbatim}

Parametric function for asymptotic approximation of the phase of a complex pole.
\begin{verbatim}
\newcommand*{\PhPoleAsymp}[2]{((t < (sqrt(\@pow{#1}{2} + \@pow{#2}{2})) ?
(-atan2(-(#2),-(#1)) :
(#2>0?(#1>0?3*pi/2:-pi/2):-pi/2))*\ph@scale)
\end{verbatim}

Plots of zeros are defined to be negative of plots of poles. The 0- is necessary due to a bug in gnuplot (fixed in version 5.4, patchlevel 3).
\begin{verbatim}
\newcommand*{\MagZero}{0-\MagPole}
\newcommand*{\MagZeroLin}{0-\MagPoleLin}
\newcommand*{\MagZeroAsymp}{0-\MagPoleAsymp}
\newcommand*{\PhZero}{0-\PhPole}
\newcommand*{\PhZeroLin}{0-\PhPoleLin}
\newcommand*{\PhZeroAsymp}{0-\PhPoleAsymp}
\end{verbatim}

4.3 Second order systems.

Although second order systems can be dealt with using the macros defined so far, the following dedicated macros for second order systems involve less computation.
\begin{verbatim}
\newcommand*{\MagCSPoles}[2]{(-20*log10(sqrt(\@pow{#2}{2} - 
\@pow{#1}{2})/2) + \@pow{2*#1*#2*t}{2})))}
\newcommand*{\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
40*log10(t)}/
\@pow{10}{abs(#1)})))/}
\@pow{10}{abs(#1)}))))})*\ph@scale}
\end{verbatim}

Consider the canonical second order transfer function $G(s) = \frac{1}{s^2 + 2\zeta\omega_n s + \omega_n^2}$. We start with true, linear, and asymptotic magnitude plots for this transfer function.
\begin{verbatim}
\newcommand*{\MagCSPoles}[2]{(-20*log10(sqrt(\@pow{#2}{2} - 
\@pow{#1}{2})/2) + \@pow{2*#1*#2*t}{2})))}
\newcommand*{\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
40*log10(t))}
\end{verbatim}

Then, we have true, linear, and asymptotic phase plots for the canonical second order transfer function.
\begin{verbatim}
\newcommand*{\PhCSPoles}[2]{((-atan2((2*(#1)*(#2)*t),\@pow{#2}{2} - 
\@pow{#1}{2})))*\ph@scale)
\end{verbatim}

\begin{verbatim}
\newcommand*{\PhCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
40*log10(t))}
\newcommand*{\PhCSPolesAsymp}[2]{(t < #2 ? -40*log10(#2) : -}
40*log10(t))}
\end{verbatim}

\begin{verbatim}
\newcommand*{\MagCSZeros}[2]{(-20*log10(sqrt(\@pow{#2}{2} - 
\@pow{#1}{2})/2) + \@pow{2*#1*#2*t}{2})))}
\newcommand*{\MagCSZerosLin}[2]{(t < #2 ? -40*log10(#2) : -
40*log10(t))}
\end{verbatim}

\begin{verbatim}
\newcommand*{\PhCSZeros}[2]{((-atan2((2*(#1)*(#2)*t),\@pow{#2}{2} - 
\@pow{#1}{2})))*\ph@scale)
\end{verbatim}

\begin{verbatim}
\newcommand*{\PhCSZerosLin}[2]{(t < #2 ? -40*log10(#2) : -
40*log10(t))}
\newcommand*{\PhCSZerosAsymp}[2]{(t < #2 ? -40*log10(#2) : -}
40*log10(t))}
\end{verbatim}
Plots of the inverse function $G(s) = s^2 + 2\zeta\omega_n s + \omega_n^2$ are defined to be negative of plots of poles. The 0- is necessary due to a bug in gnuplot (fixed in version 5.4, patchlevel 3).

\newcommand*{\MagCSZeros}{0\text{-MagCSPoles}}
\newcommand*{\MagCSZerosLin}{0\text{-MagCSPolesLin}}
\newcommand*{\MagCSZerosAsymp}{0\text{-MagCSPolesAsymp}}
\newcommand*{\PhCSZeros}{0\text{-PhCSPoles}}
\newcommand*{\PhCSZerosLin}{0\text{-PhCSPolesLin}}
\newcommand*{\PhCSZerosAsymp}{0\text{-PhCSPolesAsymp}}

These macros are used to add a resonant peak to linear and asymptotic plots of canonical second order poles and zeros. Since the plots are parametric, a separate \draw command is needed to add a vertical arrow.

\newcommand*{\MagCSPolesPeak}{\draw[#1,->] (axis cs:{#2},{-20*log10(#3)}) -- (axis cs:{#2},{-20*log10(#3)-20*log10(2*abs(#2))})}
\newcommand*{\MagSOZerosPeak}{\draw[#1,->] (axis cs:{#2},{20*log10(#3)}) -- (axis cs:{#2},{20*log10(#3)+20*log10(2*abs(#2))})}

Consider a general second order transfer function $G(s) = \frac{1}{s^2 + as + b}$. We start with true, linear, and asymptotic magnitude plots for this transfer function.

\newcommand*{\MagSOPoles}{(-20*log10(sqrt(#3^2 + #1^2)))}
\newcommand*{\MagSOPolesLin}{(t < sqrt(#2) ? -20*log10(#2) : -40*log10(t))}
\newcommand*{\MagSOPolesAsymp}{\MagSOPolesLin}

Then, we have true, linear, and asymptotic phase plots for the general second order transfer function.

\newcommand*{\PhSOPoles}{((#2>0 ? -#1/(2*sqrt(#2)) : #1>0 ? pi : -pi))}
\newcommand*{\PhSOPolesLin}{((#2>0 ? -#1/(2*sqrt(#2)) : #1>0 ? pi : -pi))}
\newcommand*{\PhSOPolesAsymp}{((#2>0 ? #1/(2*sqrt(#2)) : #1>0 ? -pi : pi))}

Plots of the inverse function $G(s) = s^2 + as + b$ are defined to be negative of plots of poles. The 0- is necessary due to a bug in gnuplot (fixed in version 5.4, patchlevel 3).

\newcommand*{\MagSOZeros}{0\text{-MagSOPoles}}
\newcommand*{\MagSOZerosLin}{0\text{-MagSOPolesLin}}
\newcommand*{\MagSOZerosAsymp}{0\text{-MagSOPolesAsymp}}
\newcommand*{\PhSOZeros}{0\text{-PhSOPoles}}
\newcommand*{\PhSOZerosLin}{0\text{-PhSOPolesLin}}
\newcommand*{\PhSOZerosAsymp}{0\text{-PhSOPolesAsymp}}

These macros are used to add a resonant peak to linear and asymptotic plots of general second order poles and zeros. Since the plots are parametric, a separate \draw command is needed to add a vertical arrow.
4.4 Commands for Bode plots

4.4.1 User macros

\textbf{BodeZPK} This macro takes lists of complex poles and zeros of the form \{\texttt{re}, \texttt{im}\}, and values of gain and delay as inputs and constructs parametric functions for the Bode magnitude and phase plots. This is done by adding together the parametric functions generated by the macros for individual zeros, poles, gain, and delay, described above. The parametric functions are then plotted in a \texttt{tikzpicture} environment using the \texttt{addplot} macro. Unless the package is loaded with the option \texttt{pgf}, the parametric functions are evaluated using \texttt{gnuplot}.

\begin{verbatim}
\newcommand{\BodeZPK}[4][\approx/true]{
  Most of the work is done by the \texttt{parse@opt} and the \texttt{build@ZPK@plot} macros, described in the 'Internal macros' section. The former is used to parse the optional arguments and the latter to extract poles, zeros, gain, and delay from the first mandatory argument and to generate macros \texttt{func@mag} and \texttt{func@ph} that hold the magnitude and phase parametric functions. The \texttt{noexpand} macros below are needed so that only the macro \texttt{opt@group} is expanded.

\par
\texttt{parse@opt}{\#1}
\texttt{gdef\func@mag{}}
\texttt{gdef\func@ph{}}
\edef\temp@cmd{\noexpand\begin{tikzpicture} \unexpanded\expandafter{\opt@tikz}}
\temp@cmd
\build@ZPK@plot{\func@mag}{\func@ph}{\opt@approx}{\#2}
\edef\temp@cmd{\noexpand\begin{groupplot}\[\bode@style,
  xmin=#3,
  xmax=#4,
  domain=#3*\freq@scale:#4*\freq@scale,
  height=2.5cm,
  xmode=log,
  group style = {group size = 1 by 2, vertical sep=0.25cm},
  \opt@group\]}
\temp@cmd
\end{verbatim}

To ensure frequency tick marks on magnitude and the phase plots are always aligned, we use the \texttt{groupplot} library. The \texttt{noexpand} and \texttt{unexpanded\expandafter} macros below are used to expand macros in the plot and group optional arguments.

\begin{verbatim}
\edef\temp@mag@cmd{\noexpand\nextgroupplot \[ylabel={Gain (dB)}, xmajor ticks=false, \optmag@axes\]
  \noexpand\addplot \[freq@filter, variable=t, thick, \optmag@plot\]}
\edef\temp@ph@cmd{\noexpand\nextgroupplot \[ph@y@label, freq@label, \optph@axes\]
  \noexpand\addplot \[freq@filter, variable=t, thick, trig format plots=rad, \optph@plot\]}
\if@pgfarginput
  \temp@mag@cmd \{\func@mag\};
  \optmag@commands
  \temp@ph@cmd \{\func@ph\};
  \optph@commands
\else
  In gnuplot mode, we increment the gnuplot@id counter before every plot to make sure that new and reusable .\texttt{gnuplot} and .\texttt{table} files are generated for every plot. We use raw \texttt{gnuplot} to make sure that the tables generated by \texttt{gnuplot} use the correct phase and frequency units as supplied by the user.

\stepcounter{gnuplot@id}
\temp@mag@cmd gnuplot \[raw gnuplot, gnuplot@prefix\]
\{ set table $meta;
  set dummy t;
  set logscale x 10;
\end{verbatim}
The following code handles active characters to avoid conflicts with ‘babe."

\AtBeginDocument{%
\if@babel
  \let\Orig@BodeZPK\BodeZPK
  \renewcommand{\BodeZPK}{\expandafter\shorthandoff\expandafter{\shorthand@list}}
  \BodeZPK@Shorthandoff
}{%\Orig@BodeZPK[#1]{#2}{#3}{#4}
  \shorthandon\expandafter{\shorthand@list}
}\fi
%
\BodeTF Implementation of this macro is very similar to the \BodeZPK macro above. The only difference is the lack of linear and asymptotic plots and slightly different parsing of the mandatory arguments.

\newcommand{\BodeTF}[4][]{
  \parse@opt{#1}
  \gdef\func@mag{}
  \gdef\func@ph{}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} \[unexpanded\]expandafter{\opt@tikz}}
  \temp@cmd
  \build@TFplot{\func@mag}{\func@ph}{#2}
  \edef\temp@cmd{\noexpand\begin{groupplot}[ylabel={Gain (dB)}, xmajor ticks=false, \optmag@axes]
  \temp@cmd
}
\noexpand\addplot [freq@filter, variable=t, thick, \opt-
mag@plot]
\edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
\noexpand\addplot [freq@filter, variable=t, thick, trig for-
mat plots=rad, \optph@plot]}
\if@pgfarg
\temp@mag@cmd {\func@mag};
\optmag@commands
\temp@ph@cmd {\mod{\func@ph}{2*pi*\ph@scale}};
\optph@commands
\else
\stepcounter{gnuplot@id}
\temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
{ set table $meta;
 set logscale x 10;
 set xrange [#3*freq@scale:#4*freq@scale];
 set samples \pgfkeysvalueof{/pgfplots/samples};
 plot \func@mag;
 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
 plot "$meta" using ($1/(\freq@scale)):($2);
};
\optmag@commands
\stepcounter{gnuplot@id}
\temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
{ set table $meta;
 set dummy t;
 set logscale x 10;
 set xrange [#3*freq@scale:#4*freq@scale];
 set samples \pgfkeysvalueof{/pgfplots/samples};
 plot '+' using (t) : ((\func@ph)/(\ph@scale)) smooth unwrap;
 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
 plot "$meta" using ($1/(\freq@scale)):(2*$\ph@scale);
};
\optph@commands
\fi
\end{groupplot}
\end{tikzpicture}

\AtBeginDocument{
\if@babel
\let\Orig@BodeTF\BodeTF
\renewcommand\BodeTF{\expandafter\shorthandoff\expandafter{\shorthand@list}
\BodeTF@Shorthandoff}
\newcommand\BodeTF@Shorthandoff[4][{}]{\Orig@BodeTF[1][#1][#2][#3][#4]}
\expandafter\shorthandon\expandafter{\shorthand@list}
\fi
\end{tikzpicture}

\addBodeZPKPlots This macro is designed to issues multiple \addplot macros for the same set of poles, zeros, gain, and delay. All of the work is done by the \build@ZPK@plot macro.
\newcommand\addBodeZPKPlots[3][true/{}]{\foreach \approx/\opt in {#3}{\gdef\plot@macro{}
\gdef\temp@macro{\ifnum\pdf@strcmp{#2}{phase}=0
\build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#2\approx}\else
\fi}}
\addBodeTFPlot This macro is designed to issue a single \addplot macros for the set of coefficients and delay. All of the work is done by the \build@TF@plot macro.

\newcommand{\addBodeTFPlot}{[3][thick]}{
\gdef\plot@macro{}
\gdef\temp@macro{}
\ifnum\pdf@strcmp{#2}{phase}=0
  \build@TF@plot{\temp@macro}{\plot@macro}{#3}
\else
  \build@TF@plot{\plot@macro}{\temp@macro}{#3}
\fi
\if@pgfarg
\ifnum\pdf@strcmp{#2}{phase}=0
  \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main={freq@scale*pgfkeysvalueof{/pgfplots/domain})*\freq@scale, vari-
able=t, thick, trig format plots=rad, \opt]}
  \temp@cmd {\r@mod{\plot@macro}{2*pi}};
\else
  \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main={freq@scale*pgfkeysvalueof{/pgfplots/domain})*\freq@scale, vari-
able=t, \opt]}
  \temp@cmd {\plot@macro};
\fi
\else
\stepcounter{gnuplot@id}
\edef\temp@cmd{\noexpand\addplot [variable=t, thick, \opt]}
\temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
{ set table $meta;
  set dummy t;
  set logscale x 10;
  set xrange [{freq@scale*pgfkeysvalueof{/pgfplots/domain})*\freq@scale};
  set samples \pgfkeysvalueof{/pgfplots/samples};
  plot \plot@macro;
  set table "$bodeplot@prefix\arabic{gnuplot@id}.table";
  plot "$meta" using ($1/(\freq@scale)):($2);
};
\fi
}
This macro is designed to create a single \addplot macro capable of plotting linear combinations of the basic components described in Section 3.1.1. The only work to do here is to handle the \pgf package option.

\newcommand{\addBodeComponentPlot}[2][thick]{
  \if@pgfarg
    \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
      main={freq@scale*\pgfkeysvalueof{/pgfplots/domain}*freq@scale, vari-
      able=t, trig format plots=rad, #1}]
    \temp@cmd {#2};
  \else
    \stepcounter{gnuplot@id}
    \addplot [variable=t, #1] gnuplot {\raw gnuplot, gnuplot@prefix}
    { set table $meta;
      set dummy t;
      set logscale x 10;
      set xrange [{freq@scale*\pgfkeysvalueof{/pgfplots/domain}*freq@scale}];
      set samples \pgfkeysvalueof{/pgfplots/samples};
      plot \plot@macro;
      set table \bodeplot@prefix\arabic{gnuplot@id}.table";
      plot "$meta" using ($1/({freq@scale})):($2);
    }
  \fi
}

BodePhPlot (env.) An environment to host phase plot macros that pass parametric functions to \addplot macros. Uses the defaults specified in bode@style to create a shortcut that includes the \tikzpicture and \semilogaxis environments. The body of the environment is grabbed as a macro to maintain compatibility with externalization in \tikz.

\AtBeginDocument{%
  \if@babel
    \AddToHook{env/BodePhPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand@list}}
    \AddToHook{env/BodePhPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@list}}
  \fi
}
\NewDocumentEnvironment{BodePhPlot}{O{}mm+b}{
\parse@env@opt{#1}
\edef\temp@cmd{\noexpand\begin{tikzpicture} {\unex-
  panded}\opt@tikz}}
\temp@cmd
\edef\temp@cmd{\noexpand\begin{semilogaxis} {\unex-
  panded}\opt@axes} ph@y@label, freq@label, bode@style, x@min={#2}, x@max={#3}, domain={#2:#3}, height=2.5cm, \temp@cmd
\}}
BodeMagPlot (env.) An environment to host magnitude plot macros that pass parametric functions to \addplot macros. Uses the defaults specified in \texttt{bode@style} to create a shortcut that includes the tikzpicture and semilogaxis environments.

\AtBeginDocument{%\if@babel\AddToHook{env/BodeMagPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand@list}}\AddToHook{env/BodeMagPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@list}}\fi\NewDocumentEnvironment{BodeMagPlot}{O{}mm+b}{\parse@env@opt{#1}\edef\temp@cmd{\noexpand\begin{tikzpicture} \unexpanded\expandafter{\opt@tikz}}\temp@cmd\edef\temp@cmd{\noexpand\begin{semilogxaxis}[bode@style,\opt@axes]}\temp@cmd#4\end{semilogxaxis}\end{tikzpicture}}{}}

BodePlot (env.) Same as BodeMagPlot. The BodePlot environment is deprecated as of v1.1.0, please use the BodePhPlot and BodeMagPlot environments instead.

\AtBeginDocument{%\if@babel\AddToHook{env/BodePlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand@list}}\AddToHook{env/BodePlot/end}{\expandafter\shorthandon\expandafter{\shorthand@list}}\fi\NewDocumentEnvironment{BodePlot}{O{}mm+b}{\parse@env@opt{#1}\edef\temp@cmd{\noexpand\begin{tikzpicture} \unexpanded\expandafter{\opt@tikz}}\temp@cmd\edef\temp@cmd{\noexpand\begin{semilogxaxis}[\opt@axes]}\temp@cmd#4\end{semilogxaxis}\end{tikzpicture}}{}}
4.4.2 Internal macros

\add@feature This is an internal macro to add a basic component (pole, zero, gain, or delay), described using one of the macros in Section 3.1.1 (input #2), to a parametric function stored in a global macro (input #1). The basic component value (input #3) is a complex number of the form \{re,im\}. If the imaginary part is missing, it is assumed to be zero. Implementation made possible by this StackExchange answer.

\build@ZPK@plot This is an internal macro to build parametric Bode magnitude and phase plots by concatenating basic component (pole, zero, gain, or delay) macros (Section 3.1.1) to global magnitude and phase macros (inputs #1 and #2). The \add@feature macro is used to do the concatenation. The basic component macros are inferred from a feature/values list, where feature is one of z,p,k, and d, for zeros, poles, gain, and delay, respectively, and \{values\} is a comma separated list of comma separated lists (complex numbers of the form \{re,im\}). If the imaginary part is missing, it is assumed to be zero.
\ifnum\pdf@strcmp{\feature}{k}=0
  \add@feature{#2}{\PhKLin}{\values}
  \add@feature{#1}{\MagKLin}{\values}
\else
  \ifnum\pdf@strcmp{#3}{linear}=0
    \add@feature{#2}{\PhKLin}{\values}
    \add@feature{#1}{\MagKLin}{\values}
  \else
    \add@feature{#2}{\PhK}{\values}
    \add@feature{#1}{\MagK}{\values}
  \fi
\fi
\fi
\fi
\fi
\fi
\fi
\fi
\fi
\fi
\PackageError{bodeplot}{Linear approximation for pure delays is not supported.}{Plot the true Bode plot using 'true' instead of 'linear'.}
\fi
\fi
\PackageError{bodeplot}{Asymptotic approximation for pure delays is not supported.}{Plot the true Bode plot using 'true' instead of 'asymptotic'.}
\else
  \ifdim\values pt < 0pt
    \PackageError{bodeplot}{Delay needs to be a positive number.}
  \fi
  \add@feature{#2}{\PhDel}{\values}
  \add@feature{#1}{\MagDel}{\values}
\fi
\fi
\fi
\fi
\fi
\PackageError{bodeplot}{
\build@TF@plot This is an internal macro to build parametric Bode magnitude and phase functions by computing the magnitude and the phase given numerator and denominator coefficients and delay (input #3). The functions are assigned to user-supplied global magnitude and phase macros (inputs #1 and #2).
\newcommand{\build@TF@plot}[3]{
  \gdef\num@real{0}
  \gdef\num@im{0}
  \gdef\den@real{0}
  \gdef\den@im{0}
  \gdef\loop@delay{0}
  \foreach \feature/\values in {#3} {
    \ifnum\pdf@strcmp{\feature}{num}=0
      \foreach \numcoeff \[count=\numpow\] in \values {
        \xdef\num@degree{\numpow}
        \foreach \numcoeff \[count=\numpow\] in \values {
          \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}
          \ifnum\currentdegree = 0
            \xdef\num@real{\num@real+\numcoeff}
          \else
            \ifodd\currentdegree
              \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-\numpow})}
            \else
              \xdef\num@im{\num@im*(\n@pow{-\numpow})}
          \fi
        }
      }
      \add@feature{#2}{\PhK}{\values}
      \add@feature{#1}{\MagK}{\values}
    \else
      \add@feature{#2}{\PhKLin}{\values}
      \add@feature{#1}{\MagKLin}{\values}
    \fi
  }
\parse@opt
Parses options supplied to the main Bode macros. A for loop over tuples of the form \obj/\typ/\opt with a long list of nested if-else statements does the job. If the input \obj is plot, axes, group, approx, or tikz the corresponding \opt are passed, unexpanded, to the \addplot macro, the \nextgroupplot macro, the groupplot environment, the \build@ZPK@plot macro, and the tikzpicture environment, respectively. If \obj is commands, the corresponding \opt are stored, unexpanded, in the macros \optph@commands and \optmag@commands, to be executed in appropriate axis environments.

\newcommand{\parse@opt}[1]{
  \gdef\optmag@axes{}
  \gdef\optph@axes{}
  \gdef\optph@plot{}
  \gdef\optmag@plot{}
  \gdef\opt@group{}
  \gdef\opt@approx{}
  \gdef\optph@commands{}
  \gdef\optmag@commands{}
  \gdef\opt@tikz{}
  \foreach \obj/\typ/\opt in {#1} {
    \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
        \gdef\optmag@axes{\unexpanded\expandafter{\typ}}
      \else
        \gdef\optph@axes{\unexpanded\expandafter{\typ}}
      \fi
    \else
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
        \gdef\optph@axes{\unexpanded\expandafter{\typ}}
      \else
        \gdef\optph@plot{\unexpanded\expandafter{\typ}}
      \fi
    \fi
  }
}
Parses options supplied to the Bode, Nyquist, and Nichols environments. A for loop over tuples of the form \obj/\opt, processed using nested if-else statements does the job. The input \obj should either be axes or tikz, and the corresponding \opt are passed, unexpanded, to the axis environment and the tikzpicture environment, respectively.

\newcommand{\parseenvopt}[1]{
  \gdef\opt@axes{}
  \gdef\opt@tikz{}
  \foreach \obj/\opt in {#1} {
    \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
      \xdef\opt@axes{\unexpanded\expandafter{\opt}}
    \else
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
        \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
      \else
        \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{approx}=0
          \xdef\opt@approx{\unexpanded\expandafter{\opt}}
        \else
          \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
            \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
              \xdef\opt@commands{\unexpanded\expandafter{\opt}}
            \else
              \xdef\opt@commands{\unexpanded\expandafter{\optmag@plot}, \optph@plot, \optmag@plot}
            \fi
            \xdef\opt@approx{\unexpanded\expandafter{\optmag@plot}, \optph@plot, \optmag@plot}
          \else
            \xdef\opt@commands{\unexpanded\expandafter{\optmag@plot}, \optph@plot, \optmag@plot}
          \fi
        \fi
      \fi
    \fi
  }
}
4.5 Nyquist plots

4.5.1 User macros

\NyquistZPK Converts magnitude and phase parametric functions built using \build@ZPK@plot into real part and imaginary part parametric functions. A plot of these is the Nyquist plot. The parametric functions are then plotted in a \tikzpicture environment using the \addplot macro. Unless the package is loaded with the option pgf, the parametric functions are evaluated using gnuplot. A large number of samples is typically needed to get a smooth plot because frequencies near 0 result in plot points that are very close to each other. Linear frequency sampling is unnecessarily fine near zero and very coarse for large \( \omega \). Logarithmic sampling makes it worse, perhaps inverse logarithmic sampling will help, pull requests to fix that are welcome!

\newcommand{\NyquistZPK}{[4][{}]{
\parse@N@opt{#1}
\gdef\func@mag{}
\gdef\func@ph{}
\edef\temp@cmd{\begin{tikzpicture}\unexpanded{\opt@tikz}}
\temp@cmd
\build@ZPK@plot{\func@mag}{\func@ph}{}{#2}
\edef\temp@cmd{\begin{axis}[bode@style,
domain=#3*\freq@scale:#4*\freq@scale,
height=5cm,
xlabel={$\Re$},
ylabel={$\Im$},
samples=500,]
\unexpanded{\opt@axes}
\}]
\temp@cmd
\addplot [only marks, mark=+, thick, red] (-1, 0);
\edef\temp@cmd{\addplot [variable=t, thick, trig format plots=rad, \unexpanded{\opt@plot}]
\if@pgfarg
\temp@cmd ( \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale)) \);}
\opt@commands
\else
\stepcounter{gnuplot@id}
\temp@cmd gnuplot [parametric, gnuplot@prefix] {
\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
};
\opt@commands
\fi
\end{axis}
\end{tikzpicture}}

The following code handles active characters to avoid conflicts with ‘babel.’
\AtBeginDocument{\if@babel
}
\let\Orig@NyquistZPK\NyquistZPK
\renewcommand{\NyquistZPK}{\expandafter\shorthandoff\expandafter{\shorthand@list}\NyquistZPK@Shorthandoff}
\if@yu
\newcommand{\NyquistZPK@Shorthandoff}[4][]{\Orig@NyquistZPK[#1]{#2}{#3}{#4}}
\expandafter\shorthandon\expandafter{\shorthand@list}\fi

\NyquistTF
Implementation of this macro is very similar to the \NyquistZPK macro above. The only difference is a slightly different parsing of the mandatory arguments via \build@TF@plot.

\newcommand{\NyquistTF}[4][]{
\let\temp@cmd\noexpand\begin{tikzpicture}\unexpanded\expandafter{\opt@tikz}\temp@cmd
\build@TF@plot{\func@mag}{\func@ph}{#2}
\edef\temp@cmd{\noexpand\begin{axis}[
\bode@style,
domain=#3*\freq@scale:#4*\freq@scale,
height=5cm,
xlabel={$\Re$},
ylabel={$\Im$},
samples=500,
\unexpanded\expandafter{\opt@axes}\]}\temp@cmd
\addplot [only marks, mark=+, thick, red] (-1 , 0);
\edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig format plots=rad, \unexpanded\expandafter{\opt@plot}\temp@cmd]};
\if@pgfarg
\temp@cmd ( {{\n@pow{10}((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
{\n@pow{10}((\func@mag)/20)}*sin((\func@ph)/(\ph@scale)) ));
\opt@commands\else
\stepcounter{gnuplot@id}
\temp@cmd gnuplot [parametric, gnuplot@prefix] {
\n@pow{10}((\func@mag)/20)*cos((\func@ph)/(\ph@scale)),
\n@pow{10}((\func@mag)/20)*sin((\func@ph)/(\ph@scale))};
\opt@commands\fi
\end{axis}
\end{tikzpicture}}

The following code handles active characters to avoid conflicts with ‘babel.’

\AtBeginDocument{%
\if@babel
\let\Orig@NyquistTF\NyquistTF
\renewcommand{\NyquistTF}{%\expandafter\shorthandoff\expandafter{\shorthand@list}\NyquistTF@Shorthandoff
\if@yu
\newcommand{\NyquistTF@Shorthandoff}[4][]{\Orig@NyquistTF[#1]{#2}{#3}{#4}}
\expandafter\shorthandon\expandafter{\shorthand@list}\fi
}
\addNyquistZPKPlot Adds Nyquist plot of a transfer function in ZPK form. This macro is designed to pass two parametric function to an \addplot macro. The parametric functions for phase (func@ph) and magnitude (func@mag) are built using the \build@ZPK@plot macro, converted to real and imaginary parts and passed to \addplot commands.

\newcommand{\addNyquistZPKPlot}[2][]{
\gdef\func@mag{}
\gdef\func@ph{}
\build@ZPK@plot{\func@mag}{\func@ph}{#2}
\if@pgfarg
\edef\temp@cmd{\noexpand\addplot [domain=freq@scale*pgfkeysvalueof{/pgfplots/domain}*freq@scale, variable=t, trig format plots=rad, #1]}
\temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
{\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
\else
\stepcounter{gnuplot@id}
\edef\temp@cmd{\noexpand\addplot [domain=freq@scale*pgfkeysvalueof{/pgfplots/domain}*freq@scale, variable=t, #1]}
\temp@cmd gnuplot [parametric, gnuplot@prefix] {
\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
};
\fi
\}

\addNyquistTFPlot Adds Nyquist plot of a transfer function in TF form. This macro is designed to pass two parametric function to an \addplot macro. The parametric functions for phase (func@ph) and magnitude (func@mag) are built using the \build@TF@plot macro, converted to real and imaginary parts and passed to \addplot commands.

\newcommand{\addNyquistTFPlot}[2][]{
\gdef\func@mag{}
\gdef\func@ph{}
\build@TF@plot{\func@mag}{\func@ph}{#2}
\if@pgfarg
\edef\temp@cmd{\noexpand\addplot [domain=freq@scale*pgfkeysvalueof{/pgfplots/domain}*freq@scale, variable=t, trig format plots=rad, #1]}
\temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
{\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
\else
\stepcounter{gnuplot@id}
\edef\temp@cmd{\noexpand\addplot [domain=freq@scale*pgfkeysvalueof{/pgfplots/domain}*freq@scale, variable=t, #1]}
\temp@cmd gnuplot [parametric, gnuplot@prefix] {
\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
};
\fi
\}

\NyquistPlot An environment to host \addNyquist... macros that pass parametric functions to \addplot. Uses the defaults specified in bode@style to create a shortcut that includes the tikzpicture and axis environments.

AtBeginDocument{%
\if@babel
\AddToHook{env/NyquistPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand@list}}
\AddToHook{env/NyquistPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@list}}
\fi
\}
NewDocumentEnvironment{NyquistPlot}{O{}mm+b}{
parse=env/opt[#1]
\begin{tikzpicture} \begin{axis}[bode@style, height=5cm, domain=#2:#3, xlabel={$\Re$}, ylabel={$\Im$}, \unexpanded\expandafter{\opt@axes}] \addplot [only marks,mark=+,thick,red] (-1,0); \end{axis} \end{tikzpicture}

\section{Internal commands}
\texttt{\parse@N@opt} Parses options supplied to the main Nyquist and Nichols macros. A \texttt{for} loop over tuples of the form \texttt{\obj/\opt}, processed using nested if-else statements does the job. If the input \texttt{\obj} is \texttt{plot}, \texttt{axes}, or \texttt{tikz} then the corresponding \texttt{\opt} are passed, unexpanded, to the \texttt{\addplot} macro, the \texttt{axis} environment, and the \texttt{tikzpicture} environment, respectively.

\begin{verbatim}
\newcommand{\parse@N@opt}[1]{
  \gdef\opt@axes{}
  \gdef\opt@plot{}
  \gdef\opt@commands{}
  \gdef\opt@tikz{}
  \foreach \obj/\opt in {#1} {
    \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
      \xdef\opt@axes{\unexpanded\expandafter{\opt}}
    \else
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
        \xdef\opt@plot{\unexpanded\expandafter{\opt}}
      \else
        \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
          \xdef\opt@commands{\unexpanded\expandafter{\opt}}
        \else
          \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
            \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
          \else
            \xdef\opt@plot{\unexpanded\expandafter{\opt@plot}, \unexpanded\expandafter{\obj}}
          \fi
        \fi
      \fi
    \fi
  }
}
\end{verbatim}

\section{Nichols charts}
These macros and the \texttt{NicholsChart} environment generate Nichols charts, and they are implemented similar to their Nyquist counterparts.

\begin{verbatim}
\newcommand{\NicholsZPK}[4][]{}
\newcommand{\NicholsTF}[4][]{}
\newcommand{\NicholsChart}[4][]{}
\end{verbatim}
\edef\func@mag{}\\
\edef\func@ph{}\\
\build@TF@plot{\func@mag}{\func@ph}{#2}\\
\if@pgfarg\\
  \edef\temp@cmd{\noexpand\addplot \[domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}, has tick label 여전히, trig format plots=rad, #1\]}\\
  \temp@cmd ( {\func@ph} , {\func@mag} );\\
\else\\
  \stepcounter{gnuplot@id}\\
  \addplot \[#1\] gnuplot \[raw gnuplot, gnuplot@prefix\]\\
  { set table $meta;\\
    set logscale x 10;\\
    set dummy t;\\
    set samples \pgfkeysvalueof{/pgfplots/samples};\\
    set trange {[\freq@scale*\pgfkeysvalueof{/pgfplots/domain}]*\freq@scale];\\
    plot '+' using (\func@mag) : (((\func@ph)/(\ph@scale));\\
    unset logscale x;\\
    set table "\bodeplot@prefix\arabic{gnuplot@id}.table";\\
    plot "$\meta" using ($2*\ph@scale):(1);\\
  };}\\
\fi\\
\newcommand{\addNicholsTFChart}[2][]{\\
  \edef\func@mag{}\\
  \edef\func@ph{}\\
  \build@TF@plot{\func@mag}{\func@ph}{#2}\\
  \if@pgfarg\\
    \edef\temp@cmd{\noexpand\addplot \[domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}, has tick label 여전히, trig format plots=rad, #1\]}\\
    \temp@cmd ( \n@mod\func@ph\[2*pi*\ph@scale] \func@mag \});\\
  \else\\
    \stepcounter{gnuplot@id}\\
    \addplot \[#1\] gnuplot \[raw gnuplot, gnuplot@prefix\]\\
    { set table $meta1;\\
      set logscale x 10;\\
      set dummy t;\\
      set samples \pgfkeysvalueof{/pgfplots/samples};\\
      set trange {[\freq@scale*\pgfkeysvalueof{/pgfplots/domain}]*\freq@scale];\\
      plot '+' using (\func@mag) : (((\func@ph)/(\ph@scale));\\
      unset logscale x;\\
      set table $meta2;\\
      plot "$\meta1" using ($1):($2) smooth unwrap;\\
      set table "\bodeplot@prefix\arabic{gnuplot@id}.table";\\
      plot "$\meta2" using ($2*\ph@scale):(1);\\
    };\\
  \fi\\
}\n
\Index\\

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>@ifpackage loaded ... 46</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>@babelfalse ... 44</td>
<td>@n1 ... 120, 121</td>
<td>\add@feature ... 519, 538, 539, 542,</td>
</tr>
<tr>
<td>@babeltrue ... 47</td>
<td>@pgfargfalse ... 1</td>
<td>543, 545, 546, 554,</td>
</tr>
<tr>
<td>@declutterargfalse ... 5</td>
<td>@pgfargtrue ... 3</td>
<td>555, 558, 559, 561,</td>
</tr>
<tr>
<td>@declutterargtrue ... 7</td>
<td>@radargfalse ... 9</td>
<td>562, 569, 570, 573,</td>
</tr>
<tr>
<td>@empty ... 48</td>
<td>@radargtrue ... 11</td>
<td>574, 576, 577, 593, 594</td>
</tr>
<tr>
<td>@hzargfalse ... 13</td>
<td>@ ... 51</td>
<td>\addBodeComponentPlot</td>
</tr>
<tr>
<td>@hzargtrue ... 15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

39
Change History

v1.0
General: Initial release

v1.0.1
\addBodeZPKPlots: Improved optional argument handling
\BodeZPK: Pass arbitrary TikZ commands as options

v1.0.2
gnuplot@prefix: Fixed issue #1

v1.0.3
BodePlot: Added tikz option to environments
\BodeTF: Added Tikz option
\BodeZPK: Added Tikz option
NicholsChart: Added tikz option to environments
\NicholsTF: Added commands and tikz options
\NicholsZPK: Added commands and tikz options
gnuplot@prefix: Added jobname to gnuplot prefix
NyquistTF: Added commands and tikz options
NyquistZPK: Added commands and tikz options
\parse@env@opt: Added tikz option to environments
\parse@N@opt: Added commands and tikz options
\parse@opt: Added Tikz option
NyquistPlot: Added tikz option to environments

v1.0.4
General: Fixed unintended optional argument macro expansion

v1.0.5
\parse@opt: Fixed a bug

v1.0.6
General: Fixed issue #3

v1.0.7
General: Removed unnecessary semicolons
Updated documentation

v1.0.8
General: Added a new class option ‘dechutter’
\build@TF@plot: Included phase due to delay in wrapping
gnuplot@prefix: Fixed issue #6

v1.1.0
General: Fixed phase wrapping in gnuplot mode
\addBodeTFPlot: Fixed phase wrapping in gnuplot mode
\BodeTF: Fixed phase wrapping in gnuplot mode
BodeMagPlot: Added separate environments for phase and magnitude plots
BodePhPlot: Added separate environments for phase and magnitude plots
BodePlot: Deprecated BodePlot environment
\BodeTF: Fixed phase wrapping in gnuplot mode

v1.1.1
General: Enable Hz and rad units
\addBodeComponentPlot: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\addBodeTFPlot: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\addBodeZPKPlots: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\addNicholsTFChart: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\addNyquistTFPlot: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\addNyquistZPKPlot: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\BodeTF: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\BodeZPK: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively

BodeMagPlot: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
BodePhPlot: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
BodePlot: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\BodeTF: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\BodeZPK: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively
\buildTFplot: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively 30
getintervalend: New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively 20
\phaseLabel: New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively 19
\NyquistTF: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively 34
\NyquistZPK: Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively 33

v1.1.2
\BodeMagPlot: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization 28
\BodePhPlot: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization 27
\BodePlot: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization 28
\NicholsChart: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization 36
\PhSOZerosLin: Fix scaling bug introduced in v1.1.1 22
\NyquistPlot: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization 35

v1.1.3
\addBodeComponentPlot: Changed implementation to respect user-supplied domain 27
\addBodeTFPlot: Changed implementation to respect user-supplied domain 26
\addBodeZPKPlots: Changed implementation to respect user-supplied domain 25
\addNicholsTFChart: Changed implementation to respect user-supplied domain 36
\addNicholsZPKChart: Changed implementation to respect user-supplied domain 36
\addNyquistTFPlot: Changed implementation to respect user-supplied domain 35
\addNyquistZPKPlot: Changed implementation to respect user-supplied domain 35

v1.1.4
\addBodeTFPlot: Changed phase wrapping in pgf mode 26
\addNicholsTFChart: Changed phase wrapping in pgf mode 36
\BodeTF: Changed phase wrapping in pgf mode 24
\gnuplotprefix: Changed phase wrapping in pgf mode 18

v1.1.5
\BodeMagPlot: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters 28
\BodePhPlot: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters 27
\BodePlot: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters 28
\NicholsChart: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters 36
\NicholsTF: Added code to handle active characters 36
\NicholsZPK: Added code to handle active characters 36
\NyquistTF: Added code to handle active characters 34
\NyquistZPK: Added code to handle active characters 33
\NyquistPlot: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters 35

v1.1.6
\shorthandlist: Detect ‘babel-french’ using ‘frenchsetup’ 18

v1.1.7
General: Detect and turn off shorthands to improve ‘babel’ compatibility 1
\BodeMagPlot: Use auto-generated list of active characters instead of manually entering them. 28
\BodePhPlot: Use auto-generated list of active characters instead of manually entering them. 27
\BodePlot: Use auto-generated list of active characters instead of manually entering them. 28
\BodeTF: Use auto-generated list of shorthands instead of manually specifying them 25

43
\BodeZPK: Use auto-generated list of shorthands instead of manually specifying them .............. 24
\NicholsChart: Use auto-generated list of active characters instead of manually entering them. ....... 36
\NicholsTF: Use auto-generated list of shorthands instead of manually specifying them .......... 36
\NicholsZPK: Use auto-generated list of shorthands instead of manually specifying them .......... 36
\NyquistTF: Use auto-generated list of active characters instead of manually entering them. ....... 35
\NyquistZPK: Use auto-generated list of shorthands instead of manually specifying them .......... 33
\shorthand@list: Directly detect shorthands instead of detecting the language. .................... 18
\NyquistPlot: Use auto-generated list of active characters instead of manually entering them. ....... 35

v1.2
General: Removed global option to process pgf commands in radians . 1
gnuplot@prefix: Removed global option to process pgf commands in radians .................... 18