The \texttt{bodeplot} package\footnote{This document corresponds to \texttt{bodeplot} \texttt{v1.0.7}, dated January 18, 2021.}

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com
January 18, 2022

Contents

1 Introduction 2

2 TL;DR 2

3 Usage 7

3.1 Bode plots ........................................... 7

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 ............ 19

4.3 Second order systems. .................................. 20

4.4 Commands for Bode plots .............................. 22

4.4.1 User macros ....................................... 22

4.4.2 Internal macros ................................... 25

4.5 Nyquist plots .......................................... 29

4.5.1 User macros ....................................... 29

4.5.2 Internal commands ................................. 32

4.6 Nichols charts ......................................... 32
1 Introduction

Generate Bode, Nyquist, and Nichols plots for transfer functions in the canonical (TF) form

\[ G(s) = e^{-T_s \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0}} \] (1)

and the zero-pole-gain (ZPK) form

\[ G(s) = K e^{-T_s \frac{(s - z_1)(s - z_2) \cdots (s - z_m)}{(s - p_1)(s - p_2) \cdots (s - p_n)}}. \] (2)

In the equations above, \( b_m, \cdots, b_0 \) and \( a_n, \cdots, a_0 \) are real coefficients, \( T \geq 0 \) is the loop delay, \( z_1, \cdots, z_m \) and \( p_1, \cdots, p_n \) 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.

Limitation: in TF form, the phase angles are always between 0 and 360°. As such, the Bode, phase plots and the Nyquist and Nichols plots will have phase wrapping discontinuities. I do not know how this can be rectified, pull requests are welcome!

2 TL;DR

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

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

---

Bode plot in ZPK format

---

1 \texttt{Bode2PZPK()}  
2 \texttt{x/0,(-0.1,-0.5),(-0.1,0.5)}, \ 3 \texttt{y/((-0.5,-10),(-0.5,10))}, \ 4 \texttt{z/10}  
5 \texttt{y/((0,0.5))}, \ 6 \texttt{100}  
7 \texttt{()}
Bode plot in TF format with arrow decoration, transport delay, and color customization

\begin{verbatim}
1 \BodeTF[
2  plot/mag/{blue,thick},
3  plot/ph/{green,thick},
4  tikz/={>=latex},
5  commands/mag/{
6  \draw[->](axis cs:1,40) -- (axis cs:10,60);
7  \node at (axis cs: 0.8,30) {\tiny Resonant Peak};
8  }
9 ]
10 {num/{10,2,2.6,0},den/{1,1,100.25},d/0.01}
11 {0.01} {100}
\end{verbatim}

Linear approximation with customization

\begin{verbatim}
1 \BodeZPK%
2 plot/mag/{red,thick},
3 plot/ph/{blue,thick},
4 axes/mag/{ytick distance=40},
5 axes/ph/{ytick distance=90},
6 approx/linear
7 ]
8 {z/{0,{-0.1,-0.5},{-0.1,0.5}},
9 p/{{-0.5,-10},{-0.5,10}},
10 k/10}
11 {0.01}
12 {100}
\end{verbatim}
Plot with delay and customization

```
begin{BodePlot}[
% axes/{ylabel={Gain (dB)},
% ytick distance=40,
% height=2cm,
% width=4cm}
]
0.01
100
addBodeZPKPlots[
% true/{black,thick},
linear/{red,dashed,thick},
asymptotic/{blue,dotted,thick}
]
end{BodePlot}
```

Individual gain and phase plots with more customization

```
begin{BodePlot}[
% axes/{ylabel={Phase ($^{\circ}$)},
% ytick distance=90,
% height=2cm,
% width=4cm,
% ytick distance=90
]
0.01
100
addBodeZPKPlots[
% true/{black,thick},
linear/{red,dashed,thick},
asymptotic/{blue,dotted,thick}
]
end{BodePlot}
```
Nichols chart

Nichols chart in TF format

Multiple Nichols charts with customization
Nyquist plot

Nyquist plot in TF format with arrows

Multiple Nyquist plots with customization
Nyquist plots with additional commands, using two different macros

\begin{lstlisting}[language=TeX]
\NyquistPlot
\begin{tikz}
  \spy using outlines={
    circle,
    magnification=3,
    connect spies,
    size=2cm
  }
\end{tikz}
\end{NyquistPlot}

3 Usage

3.1 Bode plots

\BodeZPK \{\texttt{obj/typ1/\{\texttt{opt1}\}, obj2/typ2/\{\texttt{opt2}\},...}\}
  \{\texttt{z/\{\texttt{zeros}\}, p/\{\texttt{poles}\}, k/\{\texttt{gain}\}, d/\{\texttt{delay}\}}\}
  \{\texttt{min-freq}\}\{\texttt{max-freq}\}

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 \texttt{x}–axis, and (3) the higher end of the frequency range for the \texttt{x}–axis. The zeros and the poles are complex numbers, entered as a comma-separated list of comma-separated lists, of the form \{\texttt{real part 1,imaginary part 1}\}, \{\texttt{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

\end{lstlisting}
\addplot macro for the magnitude plot if typ is mag and the phase plot if typ is ph.

- \texttt{axes/typ/{opt}}: modify axis properties by adding options \texttt{opt} to the \texttt{nextgroupplot} macro for the magnitude plot if typ is mag and the phase plot if typ is 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 typ is mag and the phase plot if typ is 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 \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 groupplot environment.
  - \texttt{tikz/{opt}}: adds options \texttt{opt} to the 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 \addplot macros for both the magnitude and the phase plots.

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

For example, given a transfer function

\[ G(s) = 10^8 \frac{s(s + 0.1 + 0.5i)(s + 0.1 - 0.5i)}{(s + 0.5 + 10i)(s + 0.5 - 10i)}, \]

its Bode plot over the frequency range \([0.01, 100]\) can be generated using
\begin{verbatim}
\BodeZPK [blue,thick] {z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{{-0.5,-10},{-0.5,10}},k/10} {0.01}{100}
\end{verbatim}

which generates the plot in Figure 1. If a delay is not specified, it is assumed to be zero. If a gain is not specified, it is assumed to be 1. By default, 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 as shown in the example below.

As demonstrated in this example, if a single comma-separated list of options is passed, it applies to both the magnitude and the phase plots. Without any optional arguments, we get a thick black Bode plot.

A linear approximation of the Bode plot with customization of the plots, the axes, and the group can be generated using
\begin{verbatim}
\BodeZPK[plot/mag/{red,thick},plot/ph/{blue,thick},
     axes/mag/ytick distance=40,xmajor ticks=true,
     xlabel={Frequency (rad/s)},
     group/group style={group size=2 by 1, horizontal sep=2cm, width=4cm, height=2cm},
     approx/linear]
     {z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{{-0.5,-10},{-0.5,10}},k/10}
\end{verbatim}

Figure 1: Output of the default \texttt{BodeZPK} macro.

Figure 2: Customization of the default \texttt{BodeZPK} macro.
Figure 3: Output of the \texttt{\textbackslash BodeTF} macro with an optional TikZ command used to mark the gain crossover frequency.

\begin{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{BodeTF}

which generates the plot in Figure 2.

\begin{BodeTF}[\{obj1/typ1/{\langle opt1 \rangle},obj2/typ2/{\langle opt2 \rangle},...\}]
{\langle num/{\langle coeffs \rangle},den/{\langle coeffs \rangle},d/{\langle delay \rangle}\rangle}
{\langle min-freq \rangle}{\langle max-freq \rangle}
\end{BodeTF}

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 from the highest degree of $s$ to the lowest, with zeros for missing degrees. The optional arguments are the same as \texttt{\textbackslash 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,

$G(s) = e^{-0.01s} \frac{s(10s^2 + 2s + 2.6)}{(s^2 + s + 100.25)}$, \hfill (5)

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

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

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{\textbackslash commands} option.

\begin{BodePlot}[\begin{BodePlot}\{\texttt{\{obj1/\{opt1\},obj2/\{opt2\},...\}]}\}
\{\texttt{\langle min-frequency \rangle}{\langle max-frequency \rangle}\}
\end{BodePlot}\addBode...
\end{BodePlot}

The \texttt{\textbackslash BodePlot} environment works in conjunction with the parametric function generator macros \texttt{\textbackslash addBodeZPKPlots, \textbackslash addBodeTFPlot}, and \texttt{\textbackslash addBodeComponentPlot}. The optional argument is comprised of a comma separated list of tuples, either \texttt{\textbackslash\textbackslash obj/\{opt\}} or just \texttt{\textbackslash\textbackslash \{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}}:
  - \texttt{tikz/{opt}}: modify picture properties by adding options \texttt{opt} to the \texttt{tikzpicture} environment.
  - \texttt{axes/{opt}}: modify axis properties by adding options \texttt{opt} to the \texttt{semilogaxis} environment.
  - \texttt{commands/{opt}}: add any valid TikZ commands inside \texttt{semilogaxis} environment. The commands passed to \texttt{opt} need to be valid TikZ commands, separated by semicolons as usual.

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

The frequency limits are translated to the x-axis limits and the domain of the \texttt{semilogaxis} environment. Example usage in the description of \texttt{\addBodeZPKPlots}, \texttt{\addBodeTFPlot}, and \texttt{\addBodeComponentPlot}.

\begin{verbatim}
\addBodeZPKPlots \{true/{black,thick},
linear/{red,dashed,thick},
asymptotic/{blue,dotted,thick}\}
\end{verbatim}

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

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

\begin{verbatim}
\begin{BodePlot}\[ ylabel={Gain (dB)}, ytick distance=40, height=2cm, width=4cm\] {0.01} {100}
\addBodeZPKPlots\[
\{\text{magnitude}\}
\{z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{\{-0.5,-10},\{-0.5,10\}},k/10\}
\end{BodePlot}
\end{verbatim}

which generates the plot in Figure 4.

\begin{verbatim}
\addBodeTFPlot\{plot-options\}
\end{verbatim}
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). 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,

- \( \text{MagK}\{k\}\{0\} \) or \( \text{MagK}\{k\}\{400\} \) generates a parametric function for the true Bode magnitude of \( G(s) = k \).
- \( \text{PhPoleLin}\{a\}\{b\} \) generates a parametric function for the linear approximation of the Bode phase of \( G(s) = \frac{1}{s-a-i b} \).
- \( \text{PhDel}\{T\}\{200\} \) or \( \text{PhDel}\{T\}\{0\} \) 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 pgfplot 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
Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePlot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

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:

```latex
\begin{BodePlot}[ylabel={Phase (degree)},ytick distance=90]{0.01}{100}
  \addBodeComponentPlot[black,thick]{\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{BodePlot}
```

which gives us the plot in Figure 5.

### 3.1.2 Basic components of the second order

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.

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

```latex
\begin{BodePlot}[xlabel={},ylabel={Phase (degree)},ytick distance=90]{0.1}{10}
  \addBodeComponentPlot[red,dashed,thick]{\MagSOPoles{0.2}{1}}
  \addBodeComponentPlot[black,thick]{\MagSOPolesLin{0.2}{1}}
\end{BodePlot}
```

generates the plot in Figure 6.
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 Lin or Asymp to generate the true Bode plot, the linear approximation, or the asymptotic approximation, respectively.

This entry describes 2 different macros of the form \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 \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.

This entry describes 2 different macros of the form \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 \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.

\subsection{Nyquist plots}

\NyquistZPK \{
\plot/{{\opt}},\axes/{{\opt}}\}
\{
\zeros,\poles,\gain,\delay\}
\{\min-freq}\{\max-freq\}

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 \BodeZPK. Since there is only one plot in a Nyquist diagram, the \typ specifier in the optional argument tuples is not needed. As such, the supported optional argument tuples are \plot/{{\opt}}, which passes \{\opt\} to \addplot, \axes/{{\opt}}, which passes \{\opt\} to the axis environment, and \tikz/{{\opt}}, which passes \{\opt\} to the \tikzpicture environment. Asymptotic/-linear approximations are not supported in Nyquist plots. If just \{\opt\} is provided as the optional argument, it is interpreted as \plot/{{\opt}}. Arrows to indicate the direction of increasing \( \omega \) can be added by adding \usetikzlibrary{decorations.markings} and \usetikzlibrary{arrows.meta} to the preamble and then passing a tuple of the form \plot/{{postaction=decorate,decoration={markings,mark=between positions 0.1 and 0.9 step 5em with \arrow{Stealth [length=2mm, blue]}}}}.

Caution: with a high number of samples, adding arrows in this way may cause the error.
message ! Dimension too big.
For example, the command
\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.

\NyquistTF
\NyquistTF [{plot/{(opt)}, axes/{(opt)}},
{(num/{(coefs)}, den/{(coefs)}, d/{(delay)})}
{(min-freq)}{(max-freq)}]

Nyquist plot of a transfer function given in TF format. Same mandatory arguments as
\BodeTF and same optional arguments as \NyquistZPK. For example, the command
\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}]

generates the Nyquist plot in Figure 8.

NyquistPlot
\begin{NyquistPlot}[(obj1/{(opt1)}, obj2/{(opt2)}, ...)]

{(min-frequency)}{(max-frequency)}
\addNyquist...
\end{NyquistPlot}

The NyquistPlot environment works in conjunction with the parametric function gen-
erator macros \addNyquistZPKPlot and \addNyquistTFPlot. 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 en-
vironment.
  - commands/{opt}: add any valid TikZ commands inside axis environment.
The commands passed to opt need to be valid TikZ commands, separated by
• 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.

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

\begin{figure}
\centering
\begin{tikzpicture}
    \begin{axis}[
        xlabel=$\Re$, ylabel=$\Im$
    ]
        \addplot[mark=*,mark size=2pt] coordinates {(-2.0,4.0) (-1.0,1.0) (0.0,0.0) (1.0,-1.0) (2.0,-2.0)};
    \end{axis}
\end{tikzpicture}
\caption{Output of the \texttt{NyquistTF} macro with direction arrows. Increasing the number of samples can cause \texttt{decorations.markings} to throw errors.}
\end{figure}

3.3 Nichols charts

\begin{figure}
\centering
\begin{tikzpicture}
    \begin{axis}[
        xlabel=$\Re$, ylabel=$\Im$
    ]
        \addplot[mark=*,mark size=2pt] coordinates {(-2.0,4.0) (-1.0,1.0) (0.0,0.0) (1.0,-1.0) (2.0,-2.0)};
    \end{axis}
\end{tikzpicture}
\caption{Nichols chart of a transfer function given in ZPK format. Same arguments as \texttt{NyquistZPK}.}
\end{figure}

\begin{figure}
\centering
\begin{tikzpicture}
    \begin{axis}[
        xlabel=$\Re$, ylabel=$\Im$
    ]
        \addplot[mark=*,mark size=2pt] coordinates {(-2.0,4.0) (-1.0,1.0) (0.0,0.0) (1.0,-1.0) (2.0,-2.0)};
    \end{axis}
\end{tikzpicture}
\caption{Nichols chart of a transfer function given in TF format. Same arguments as \texttt{NyquistTF}.}
\end{figure}

For example, the command
\begin{verbatim}
\NicholsTF[plot/{green,thick,samples=2000}]
{num/{10,2,2.6,0},den/{1,1,100.25},d/0.01}
{0.001}{100}
\end{verbatim}
generates the Nichols chart in Figure 9.

\begin{verbatim}
\begin{NicholsChart}\{obj1/{opt1},obj2/{opt2}\,...\}
\end{NicholsChart}
\end{verbatim}
Figure 9: Output of the `\NyquistZPK` macro.

\addNichols...
\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.
  - `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.

\addNicholsZPKChart \addNicholsZPKChart[\langle plot-options \rangle]
\{(z/{\langle zeros \rangle}),p/{\langle poles \rangle},k/{\langle gain \rangle},d/{\langle delay \rangle})\}

Generates a twp parametric functions for the magnitude and the phase a transfer function in ZPK form. The generated magnitude and phase parametric functions are 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 `NicholsChart` environment supplied with this package is recommended. The mandatory argument is the same as `\BodeZPK`.

\addNicholsTFChart \addNicholsTFChart[\langle plot-options \rangle]
\{(num/{\langle coeffs \rangle}),den/{\langle coeffs \rangle},d/{\langle delay \rangle})\}

Similar to `\addNicholsZPKChart`, with a transfer function input in the TF form.
4 Implementation

4.1 Initialization

The package makes extensive use of the `\pdfstrcmp` macro to parse options. Since that macro is not available in `lualatex`, this code is needed.

```latex
\RequirePackage{ifluatex}\
\ifluatex\
  \let\pdfstrcmp\pdf@strcmp\\n\fi
```

This code is needed to support both `pgfplots` and `gnuplot` simultaneously. New macros are defined for the `pow` and `mod` functions to address differences between the two math engines. We start by processing the `pgf` class option.

```latex
\newif\ifpgfarg\@pgfargfalse\n\DeclareOption{pgf}{\@pgfargtrue}\n\ProcessOptions\relax\n
\if@pgfarg\n\newcommand{\n@pow}\[2\]{(#1)^{(#2)}}\n\newcommand{\n@mod}\[2\]{mod((#1),(#2))}\\n\else\n\newcommand{\n@pow}\[2\]{(#1)^{(#2)}}\n\newcommand{\n@mod}\[2\]{(#1)-(\text{floor}((#1)/(#2))*(#2))}\\n\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 `gnuplot`, which reduces compilation time.

```latex
\newcounter{gnuplot@id}\\n\setcounter{gnuplot@id}{0}\\n\tikzset{\n  gnuplot@prefix/.style={\n    id=\arabic{gnuplot@id},\n    prefix=gnuplot/\jobname\n  }\n}\n```

Then, we add `set angles degrees` to all `gnuplot` macros to avoid having to convert from degrees to radians everywhere.

```latex
\pgfplotsset{\n  gnuplot@degrees/.code={\n    \ifnum\value{gnuplot@id}=1\n      \xdef\pgfplots@gnuplot@format{\pgfplots@gnuplot@format set angles degrees;}\n    \fi\n  }\n}\n```

If the operating system is not Windows, we create the `gnuplot` folder if it does not already exist.

```latex
\ifwindows\else\n  \immediate\write18{mkdir -p gnuplot}\n\fi\n```

Default axis properties for all plot macros are collected in this `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,
  }%
}

4.2 Parametric function generators for poles, zeros, gains, and delays.

\textbf{\texttt{\MagK}} \texttt{\MagKAsymp} \texttt{\MagKLin} \textbf{\texttt{\PhK}} \texttt{\PhKAsymp} \texttt{\PhKLin}\linebreak

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.

\begin{verbatim}
\newcommand*{\MagK}[2]{(20*log10(abs(#1)))}
\newcommand*{\MagKAsymp}{\MagK}
\newcommand*{\MagKLin}{\MagK}
\end{verbatim}

\begin{verbatim}
\newcommand*{\PhK}[2]{(#1<0?-180:0)}
\newcommand*{\PhKAsymp}{\PhK}
\newcommand*{\PhKLin}{\PhK}
\end{verbatim}

True magnitude and phase parametric functions for a pure delay \( G(s) = e^{-Ts} \). 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.

\begin{verbatim}
\newcommand*{\MagDel}[2]{0}
\newcommand*{\PhDel}[2]{-#1*180*t/pi}
\end{verbatim}

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*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]{(t < sqrt(#1^2 + #2^2) ? \{-20*log10(sqrt(#1^2 + #2^2))\} : \{-20*log10(t)\})}
\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]{(\n@mod{-atan2((t - (#2)),(-#1))+360}{360}) : \{-atan2((t - (#2)),(-#1))\})
\end{verbatim}
Parametric function for linear approximation of the phase of a complex pole.

\[ \text{\texttt{\textbackslash newcommand*\{\texttt{PhPoleLin}\}[2]\{}} \]
\[ (\text{abs}(#1)+\text{abs}(#2) == 0 ? -90 : \]
\[ (t < (\text{sqrt}(\text{n@pow}(#1){2}) + \text{n@pow}(#2){2})) / \]
\[ (\text{n@pow}(10){\text{sqrt}(\text{n@pow}(#1){2})/(\text{n@pow}(#1){2} + \text{n@pow}(#2){2}))}) ]\]
\[ (\text{atan2}(\text{#2},-(\text{#1}))) ; \]
\[ (t >= (\text{sqrt}(\text{n@pow}(#1){2} * \text{n@pow}(#2){2})) / \]
\[ (\text{n@pow}(10){\text{sqrt}(\text{n@pow}(#1){2})/(\text{n@pow}(#1){2} + \text{n@pow}(#2){2}))}) ]\]
\[ (\text{#2>0?#1>0?270:-90} : -90) ; \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1})) + (\text{log10}(t/(\text{sqrt}(\text{n@pow}(#1){2} + \text{n@pow}(#2){2})))))/*(\text{#2>0?#1>0?270:-90} : -90)) ; \]
\[ (\text{n@pow}(#1){2} + \text{n@pow}(#2){2}))}))]]\]

Parametric function for asymptotic approximation of the phase of a complex pole.

\[ \text{\texttt{\textbackslash newcommand*\{\texttt{PhPoleAsymp}\}[2]\{}} \]
\[ (t < (\text{sqrt}(\text{n@pow}(#1){2} + \text{n@pow}(#2){2})) ? \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1}))) : \]
\[ (\text{#2>0?#1>0?270:-90} : -90)) \]
\[ (\text{n@pow}(10){\text{sqrt}(\text{n@pow}(#1){2} + \text{n@pow}(#2){2})})/\]
\[ (\text{#2>0?#1>0?270:-90}) : \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1})) + (\log10(t/(\text{sqrt}(\text{n@pow}(#1){2} + \text{n@pow}(#2){2})))))/*(\text{#2>0?#1>0?270:-90}) : \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1})) + (\text{log10}(t/(\text{sqrt}(\text{n@pow}(#1){2} + \text{n@pow}(#2){2}))))/\]
\[ (\text{log10}(\text{n@pow}(10){\text{sqrt}(\text{n@pow}(#1){2})/\}
\[ (\text{#2>0?#1>0?270:-90}) : \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1})) + (\log10(t/(\text{sqrt}(\text{n@pow}(#1){2} + \text{n@pow}(#2){2})))))/*(\text{#2>0?#1>0?270:-90})) \]

\text{\textbackslash newcommand*\{\texttt{MagZero}\}{0-\texttt{MagPole}}} \]
\text{\textbackslash newcommand*\{\texttt{MagZeroLin}\}{0-\texttt{MagPoleLin}}} \]
\text{\textbackslash newcommand*\{\texttt{MagZeroAsymp}\}{0-\texttt{MagPoleAsymp}}} \]
\text{\textbackslash newcommand*\{\texttt{PhZero}\}{0-\texttt{PhPole}}} \]
\text{\textbackslash newcommand*\{\texttt{PhZeroLin}\}{0-\texttt{PhPoleLin}}} \]
\text{\textbackslash newcommand*\{\texttt{PhZeroAsymp}\}{0-\texttt{PhPoleAsymp}}} \]

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.

\[ \text{\texttt{\textbackslash newcommand*\{\texttt{MagCSPoles}\}[2]\{}} \]
\[ (-20*\text{log10}(\text{n@pow}(\text{n@pow}(#2)/2}}} \]
\[ - \text{pow}(\text{sqrt}(\text{n@pow}(\text{n@pow}(#2)}2}}} \]
\[ - \text{pow}(\text{pow}(\text{#1})2}}} \]
\[ (\text{#1>0?270:-90}) ; \]
\[ (-\text{atan2}(\text{#2},-(\text{#1})) + (\text{log10}(\text{#1})/\text{pow}(\text{sqrt}(\text{n@pow}(#1)}2}}} \]
\[ (\text{n@pow}(#1}{2} + \text{n@pow}(#2}{2}))//\]
\[ (\text{#1>0?270:-90}) : \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1})) + (\log10(t/(\text{sqrt}(\text{n@pow}(#1}{2} + \text{n@pow}(#2}{2}))))/)\]
\[ (\text{#2>0?#1>0?270:-90}) : \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1})) + (\text{log10}(\text{#2}/(\text{sqrt}(\text{n@pow}(#1}{2} + \text{n@pow}(#2}{2}))))/\]
\[ (\text{#1>0?#1>0?270:-90}) : \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1})) + (\text{log10}(\text{#2}/(\text{sqrt}(\text{n@pow}(#1}{2} + \text{n@pow}(#2}{2}))))/*(\text{#2>0?#1>0?270:-90})) \]

Plots of zeros are defined to be negative of plots of poles. The 0- is necessary due to a bug in \texttt{gnuplot} (fixed in version 5.4, patchlevel 3).

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

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.

\[ \text{\texttt{\textbackslash newcommand*\{\texttt{PhCSPoles}\}[2]\{}} (-20*\text{log10}(\text{n@pow}(\text{n@pow}(#2}{2}}} \]
\[ - \text{pow}(\text{sqrt}(\text{n@pow}(\text{n@pow}(#2}{2}}} \]
\[ - \text{pow}(\text{pow}(\text{#1})2}}} \]
\[ (\text{#1>0?270:-90}) ; \]
\[ (-\text{atan2}(\text{#2},-(\text{#1})) + (\text{log10}(\text{#1})/\text{pow}(\text{sqrt}(\text{n@pow}(#1}{2} + \text{n@pow}(#2}{2}))))/\]
\[ (\text{#1>0?270:-90}) : \]
\[ (-\text{atan2}(-\text{#2},-(\text{#1})) + (\log10(t/(\text{sqrt}(\text{n@pow}(#1}{2} + \text{n@pow}(#2}{2}))))/*(\text{#1>0?270:-90})) \]

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

\[ \text{\texttt{\textbackslash newcommand*\{\texttt{PhCSPolesLin}\}[2]\{}} (t < #2 ? -40*\text{log10}(#2) : - 40*\text{log10}(t)) \]
\text{\texttt{\textbackslash newcommand*\{\texttt{MagCSPolesLin}\}}} \]

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 \texttt{gnuplot} (fixed in version 5.4, patchlevel 3).
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*{\PhCSZerosLin}{0-\PhCSPolesLin}
\newcommand*{\PhCSZerosAsymp}{0-\PhCSPolesAsymp}
\newcommand*{\MagCSZerosPeak}{3}{%}
{\draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
(axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2)))}}
\newcommand*{\MagCSZerosLin}{3}{%}
{\draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
(axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2)))}}
\newcommand*{\PhCSZerosPeak}{3}{%}
\PhCSPolesLin{#1}{#2}{#3}{%}
\draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
(axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) - 10*log10(abs(#2/sqrt(abs(#3))))});
\newcommand*{\PhCSZerosLin}{3}{%}
\PhCSPolesLin{#1}{#2}{#3}{%}
\draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
(axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) + 10*log10(abs(#2/sqrt(abs(#3))))});

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}{2}{%}
\MagSOPolesLin[2]{%}
\MagSOPolesAsymp[2]{%}
\PhSOPolesLin[2]{%}
\PhSOPolesAsymp[2]{%}
\newcommand*{\MagSOPolesLin}{2}{%}
\MagSOPolesLin[2]{%}
\PhSOPolesLin[2]{%}
\PhSOPolesAsymp[2]{%}
\newcommand*{\MagSOZeros}{0\MagSOPoles}{%}
\MagSOZerosLin[0\MagSOPoles]{%}
\PhSOZerosLin[0\PhSOPoles]{%}
\PhSOZerosAsymp[0\PhSOPoles]{%}
\newcommand*{\MagSOZerosLin}{0\MagSOPoles}{%}
\MagSOZerosLin[0\MagSOPoles]{%}
\PhSOZerosLin[0\PhSOPoles]{%}
\PhSOZerosAsymp[0\PhSOPoles]{%}
\newcommand*{\MagSOZerosAsymp}{0\MagSOPoles}{%}
\MagSOZerosAsymp[0\MagSOPoles]{%}
\PhSOZerosAsymp[0\PhSOPoles]{%}
\PhSOZerosAsymp[0\PhSOPoles]{%}

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

\newcommand*{\PhSOPoles}{2}{%}
\PhSOPolesLin[2]{%}
\PhSOPolesAsymp[2]{%}
\newcommand*{\PhSOPolesLin}{2}{%}
\PhSOPolesLin[2]{%}
\PhSOPolesAsymp[2]{%}
\newcommand*{\PhSOZeros}{0\PhSOPoles}{%}
\PhSOZerosLin[0\PhSOPoles]{%}
\PhSOZerosAsymp[0\PhSOPoles]{%}
\newcommand*{\PhSOZerosLin}{0\PhSOPoles}{%}
\PhSOZerosLin[0\PhSOPoles]{%}
\PhSOZerosAsymp[0\PhSOPoles]{%}
\newcommand*{\PhSOZerosAsymp}{0\PhSOPoles}{%}
\PhSOZerosAsymp[0\PhSOPoles]{%}
\PhSOZerosAsymp[0\PhSOPoles]{%}

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-\MagSOPoles}{%}
\newcommand*{\MagSOZerosLin}{0-\MagSOPolesLin}{%}
\newcommand*{\MagSOZerosAsymp}{0-\MagSOPolesAsymp}{%}
\newcommand*{\PhSOZeros}{0-\PhSOPoles}{%}
\newcommand*{\PhSOZerosLin}{0-\PhSOPolesLin}{%}
\newcommand*{\PhSOZerosAsymp}{0-\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.

\newcommand*{\MagSOZerosPeak}{3}{%}
{\draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
(axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))})
\newcommand*{\PhSOZerosPeak}{3}{%}
{\draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
(axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))} +
20*log10(abs(#3))});
4.4 Commands for Bode plots

4.4.1 User macros

\BodeZPK

This macro takes lists of complex poles and zeros of the form \{re,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 \tikzpicture environment using the \addplot macro. Unless the package is loaded with the option \pgf, the parametric functions are evaluated using gnuplot.

\newcommand{\BodeZPK}[4][approx/true]{%

Most of the work is done by the \parseopt and the \buildZPKplot 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 \funcmag and \funcph that hold the magnitude and phase parametric functions.

\edef\tempcmd{\noexpand\begin{tikzpicture}\unexpanded{\opttikz}\]
\noexpand\begin{groupplot}[bode@style,
xmin={#3},
xmax={#4},
domain=#3:#4,
height=2.5cm,
xmode=log,
group style = {group size = 1 by 2,vertical sep=0.25cm},
\optgroup
]

To ensure frequency tick marks on magnitude and the phase plots are always aligned, we use the \tikzlibrary. The \expandafter chain below is used to expand macros in the plot and group optional arguments.

\edef\tempcmd{\noexpand\addplot[variable=t,thick,\optmag@plot] \funcmag; \optmag@commands
\noexpand\addplot[variable=t,thick,\optph@plot] \funcph; \optph@commands
}

In gnuplot mode, we increment the \gnuplotid counter before every plot to make sure that new and reusable .\gnuplot and .\table files are generated for every plot.

\stepcounter{gnuplotid}
\edef\tempcmd{\noexpand\addplot[variable=t,thick,\optmag@plot] \funcmag; \optmag@commands
\noexpand\addplot[variable=t,thick,\optph@plot] \funcph; \optph@commands
}
\else

In gnuplot mode, we increment the \gnuplotid counter before every plot to make sure that new and reusable .\gnuplot and .\table files are generated for every plot.

\stepcounter{gnuplotid}
\edef\tempcmd{\noexpand\addplot[variable=t,thick,\optmag@plot] \funcmag; \optmag@commands
\noexpand\addplot[variable=t,thick,\optph@plot] \funcph; \optph@commands
}
\temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\func@mag};
\optmag@commands
\stepcounter{gnuplot@id}
\expandafter\nextgroupplot\expandafter[ytick distance=45,
ylabel={Phase ($^\circ$)},xlabel={Frequency (rad/s)},\optph@axes]
\edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optph@plot]}
\temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\func@ph};
\optph@commands
\fi
\end{groupplot}
\end{tikzpicture}
}

\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{}
\build@TF@plot{\func@mag}{\func@ph}{#2}%
\edef\temp@cmd{\noexpand\begin{tikzpicture}\{\unexpanded\expandafter{\opt@tikz}\}\
\opt@group
\begin{groupplot}[bode@style,
xmin={#3},
xmax={#4},
domain=#3:#4,
height=2.5cm,
xmode=log,
group style = {group size = 1 by 2,vertical sep=0.25cm},
\opt@group
\end{groupplot}[%
\temp@cmd
\if@pgfarg
\expandafter\nextgroupplot\expandafter[ytick distance=20,
ylabel={Gain (dB)},xmajor ticks=false,\optmag@axes]
\edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optmag@plot]}
\temp@cmd {\func@mag};
\optmag@commands
\expandafter\nextgroupplot\expandafter[ytick distance=45,
ylabel={Phase ($^\circ$)},xlabel={Frequency (rad/s)},\optph@axes]
\edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optph@plot]}
\temp@cmd {\func@ph};
\optph@commands
\else
\stepcounter{gnuplot@id}
\expandafter\nextgroupplot\expandafter[ytick distance=20,
ylabel={Gain (dB)},xmajor ticks=false,\optmag@axes]
\edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optmag@plot]}
\temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\func@mag};
\optmag@commands
\stepcounter{gnuplot@id}
\expandafter\nextgroupplot\expandafter[ytick distance=45,
ylabel={Phase ($^\circ$)},xlabel={Frequency (rad/s)},\optph@axes]
\edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optph@plot]}
\temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\func@ph};
\optph@commands
\fi
23
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 {#1} {%
  \gdef\plot@macro{}%
  \gdef\temp@macro{}%
  \ifnum\pdfstrcmp{#2}{phase}=0
    \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}%
  \else
    \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}%
  \fi
  \if@pgfarg
    \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt]{\plot@macro};}
  \else
    \stepcounter{gnuplot@id}%
    \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt] gnuplot[gnuplot@degrees,gnuplot@prefix] {\plot@macro};}
  \fi
}\}%

This macro is designed to issues 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\pdfstrcmp{#2}{phase}=0
  \build@TF@plot{\temp@macro}{\plot@macro}{#3}%
\else
  \build@TF@plot{\plot@macro}{\temp@macro}{#3}%
\fi
\if@pgfarg
  \edef\temp@cmd{\noexpand\addplot[variable=t,#1]{\plot@macro};}
\else
  \stepcounter{gnuplot@id}%
  \edef\temp@cmd{\noexpand\addplot[variable=t,#1] gnuplot[gnuplot@degrees,gnuplot@prefix] {\plot@macro};}
\fi
}

This macro is designed to issue 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
  \addplot[variable=t,#1]{\plot@macro};
\else
  \stepcounter{gnuplot@id}%
  \addplot[variable=t,#1] gnuplot[gnuplot@degrees,gnuplot@prefix] {\plot@macro};
\fi
}

BodePlot  An environment to host 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.
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 \texttt{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.
\add@feature{#1}{\MagZeroAsymp}\%
\else\n\add@feature{#2}{\PhZero}\%
\add@feature{#1}{\MagZero}\%
\fi\n\fi\n\fi}%
\fi\n\ifnum\pdfstrcmp{\feature}{p}=0\n\foreach \p in \values {%
\ifnum\pdfstrcmp{#3}{linear}=0\n\add@feature{#2}{\PhPoleLin}{\p}\%
\add@feature{#1}{\MagPoleLin}{\p}\%
\else\n\ifnum\pdfstrcmp{#3}{asymptotic}=0\n\add@feature{#2}{\PhPoleAsymp}{\p}\%
\add@feature{#1}{\MagPoleAsymp}{\p}\%
\else\n\add@feature{#2}{\PhPole}{\p}\%
\add@feature{#1}{\MagPole}{\p}\%
\fi\n\fi\n\fi\n\ifnum\pdfstrcmp{\feature}{k}=0\n\ifnum\pdfstrcmp{#3}{linear}=0\n\PackageError{bodeplot}{Linear approximation for pure delays is not supported.}{Plot the true Bode plot using ‘true’ instead of ‘linear’.}\
\else\n\ifnum\pdfstrcmp{#3}{asymptotic}=0\n\PackageError{bodeplot}{Asymptotic approximation for pure delays is not supported.}{Plot the true Bode plot using ‘true’ instead of ‘asymptotic’.}\
\else\n\ifdim\values pt < 0pt\n\PackageError{bodeplot}{Delay needs to be a positive number.}\
\fi\n\fi\n\fi\n\fi
\ifnum\pdfstrcmp{\feature}{d}=0\n\ifnum\pdfstrcmp{#3}{linear}=0\n\PackageError{bodeplot}{Linear approximation for pure delays is not supported.}{Plot the true Bode plot using ‘true’ instead of ‘linear’.}\
\else\n\ifnum\pdfstrcmp{#3}{asymptotic}=0\n\PackageError{bodeplot}{Asymptotic approximation for pure delays is not supported.}{Plot the true Bode plot using ‘true’ instead of ‘asymptotic’.}\
\else\n\ifdim\values pt < 0pt\n\PackageError{bodeplot}{Delay needs to be a positive number.}\
\fi\n\fi\n\fi
\fi}\n\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\pdfstrcmp{\feature}{num}=0
      \foreach \numcoeff [{count=\numpow}] in \values {
        \xdef\num@degree{\numpow}%
        \\
      }
    \else
      \ifodd\num@degree
        \xdef\num@real{\num@real+\numcoeff*({@pow{-1}{(\num@degree-1)/2}}*({\n@pow{t}{\num@degree}}))}%
      \else
        \xdef\num@real{\num@real+(\numcoeff*({@pow{-1}{(\num@degree)/2}}*({\n@pow{t}{\num@degree}})))}%
      \fi
      \fi
    \fi
    \foreach \numcoeff [{count=\numpow}] in \values {
      \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}%
      \ifnum\currentdegree = 0
        \xdef\num@real{\num@real+\numcoeff}%
      \else
        \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}%
        \ifodd\currentdegree
          \xdef\num@im{\num@im+(\numcoeff*({@pow{-1}{(\num@degree-1)/2}}*({\n@pow{t}{\num@degree}}))}%
        \else
          \xdef\num@real{\num@real+(\numcoeff*({@pow{-1}{(\num@degree)/2}}*({\n@pow{t}{\num@degree}})))}%
        \fi
      \fi
    }
  \fi
  \ifnum\pdfstrcmp{\feature}{den}=0
    \foreach \dencoeff [{count=\denpow}] in \values {
      \xdef\den@degree{\denpow}%
      \\
    }
  \else
    \ifodd\den@degree
      \xdef\den@im{\den@im+(\dencoeff*({@pow{-1}{(\den@degree-1)/2}}*({\n@pow{t}{\den@degree}}))}%
    \else
      \xdef\den@real{\den@real+(\dencoeff*({@pow{-1}{(\den@degree)/2}}*({\n@pow{t}{\den@degree}})))}%
    \fi
  \fi
  \xdef\loop@delay{\values}%
\}

\xdef#2{({@atan2((\num@im),\num@real)}-{@atan2((\den@im),\den@real)}+\loop@delay*180/t/pi)}%
\xdef#1{20*\log10(sqrt((\num@real)^2+(\num@im)^2))-20*\log10(sqrt((\den@real)^2+(\den@im)^2)))}%

\parse@opt Parses options supplied to the main Bode macros. A \texttt{for} loop over tuples of the form
\input{parseopt}

\begin{verbatim}
\newcommand\parseopt[1]{\% 
  \foreach \obj/\typ/\opt in {#1} {\% 
    \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{plot}=0 
      \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{mag}=0 
        \edef\optmag@plot{\unexpanded\expandafter{\opt}}\% 
      \else 
        \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0 
          \edef\optph@plot{\unexpanded\expandafter{\opt}}\% 
        \else 
          \edef\optmag@plot{\unexpanded\expandafter{\opt}}\% 
          \edef\optph@plot{\unexpanded\expandafter{\opt}}\% 
        \fi 
      \fi 
    \else 
      \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0 
        \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{mag}=0 
          \edef\optmag@axes{\unexpanded\expandafter{\opt}}\% 
        \else 
          \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0 
            \edef\optph@axes{\unexpanded\expandafter{\opt}}\% 
          \else 
            \edef\optmag@axes{\unexpanded\expandafter{\opt}}\% 
            \edef\optph@axes{\unexpanded\expandafter{\opt}}\% 
          \fi 
        \fi 
      \fi 
    \else 
      \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{group}=0 
        \edef\opt@group{\unexpanded\expandafter{\opt}}\% 
      \else 
        \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{approx}=0 
          \edef\opt@approx{\unexpanded\expandafter{\opt}}\% 
        \else 
          \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{commands}=0 
            \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0 
              \edef\optph@commands{\unexpanded\expandafter{\opt}}\% 
            \else 
              \edef\optmag@commands{\unexpanded\expandafter{\opt}}\% 
            \fi 
          \else 
            \edef\optmag@commands{\unexpanded\expandafter{\opt}}\% 
          \fi 
        \else 
          \edef\opt@tikz{\unexpanded\expandafter{\opt}}\% 
        \fi 
      \fi 
    \fi 
  \fi 
}\end{verbatim}
\parse@env@opt

Parses options supplied to the Bode, Nyquist, and Nichols environments. A \texttt{for} loop over tuples of the form \texttt{\obj/\opt}, processed using nested if-else statements does the job. The input \texttt{\obj} should either be \texttt{axes} or \texttt{tikz}, and the corresponding \texttt{\opt} are passed, unexpanded, to the \texttt{axis} environment and the \texttt{tikzpicture} environment, respectively.

\newcommand{\parse@env@opt}[]{\}

\newcommand{\NyquistZPK}[]{\}

\texttt{\NyquistZPK} Converts magnitude and phase parametric functions built using \texttt{\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 \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}. 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}[]{\}

4.5 Nyquist plots

4.5.1 User macros

\texttt{\NyquistZPK} Converts magnitude and phase parametric functions built using \texttt{\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 \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}. 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!
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.
\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
    \addplot[variable=t,#1] ( \n@pow{10}{((\func@mag)/20)}*\cos(\func@ph),
    \n@pow{10}{((\func@mag)/20)}*\sin(\func@ph) );
  \else
    \_stepcounter{gnuplot@id}%
    \addplot[variable=t,#1] gnuplot[parametric,gnuplot@degrees,gnuplot@prefix]{%
      \n@pow{10}{((\func@mag)/20)}*\cos(\func@ph),
      \n@pow{10}{((\func@mag)/20)}*\sin(\func@ph)};
  \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
    \addplot[variable=t,#1] ( \n@pow{10}{((\func@mag)/20)}*\cos(\func@ph),
    \n@pow{10}{((\func@mag)/20)}*\sin(\func@ph) );
  \else
    \_stepcounter{gnuplot@id}%
    \addplot[variable=t,#1] gnuplot[parametric,gnuplot@degrees,gnuplot@prefix]{%
      \n@pow{10}{((\func@mag)/20)}*\cos(\func@ph),
      \n@pow{10}{((\func@mag)/20)}*\sin(\func@ph)};
  \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.

\newenvironment{NyquistPlot}[3][]{%
  \parse@env@opt{#1}%
  \edef\temp{\noexpand\begin{tikzpicture}\unexpanded\expandafter{\opt@tikz}}%
  \noexpand\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} %
}
4.5.2 Internal commands

\parse@N@opt

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

\newcommand{\parse@N@opt}[1]{%
  \gdef\opt@axes{}
  \gdef\opt@plot{}
  \gdef\opt@commands{}
  \gdef\opt@tikz{}
  \foreach \obj/\opt in {#1} {%
    \ifequ \pdfstrcmp{\unexpanded\expandafter{\obj}}{axes} 0
      \xdef\opt@axes{\unexpanded\expandafter{\opt}}%
    \else
      \ifequ \pdfstrcmp{\unexpanded\expandafter{\obj}}{plot} 0
        \xdef\opt@plot{\unexpanded\expandafter{\opt}}%
      \else
        \ifequ \pdfstrcmp{\unexpanded\expandafter{\obj}}{commands} 0
          \xdef\opt@commands{\unexpanded\expandafter{\opt}}%
        \else
          \ifnum\pdfstrcmp{\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
  \fi
}\}

4.6 Nichols charts

These macros and the NicholsChart environment generate Nichols charts, and they are implemented similar to their Nyquist counterparts.

\NicholsZPK
\NicholsTF
NicholsChart
\addNicholsZPKChart
\addNicholsTFChart

\newcommand{\NicholsZPK}[4]{%
  \parse@N@opt{#1}%
  \gdef\func@mag{}
  \gdef\func@ph{}
  \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
  \edef{\temp@cmd}{\noexpand\begin{tikzpicture}{\unexpanded\expandafter{\opt@tikz}}%
    \noexpand\begin{axis}[]{bode@style, domain=#3:#4, height=5cm, xlabel={Phase (degrees)}, ylabel={Gain (dB)}, samples=500, \unexpanded\expandafter{\opt@axes}]
  \temp@cmd
  \edef{\temp@cmd}{\noexpand\addplot[variable=t,thick,\opt@plot]}
}
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.

Symbols
\@pgfargfalse ........ 5
\@pgfargtrue ........ 7
\add@feature ........ 297, 316, 317, 320, 321, 323, 324, 325, 333, 335, 337, 339, 340, 347, 348, 351, 352, 354, 355, 371, 372
\addBodeComponentPlot ............... 271
\addBodeTFPlot ........ 256
\addBodeZPKPlots ... 247
\addNicholsTFChart 649
\addNicholsZPKChart 649
\addNyquistTFPlot ... 591
\addNyquistZPKPlot ... 577
\addplot[variable=t,#1] ( {\func@ph} , {\func@mag} );
\addplot[variable=t,#1] gnuplot[gnuplot@degrees,gnuplot@prefix]
{\func@ph , \func@mag};
\build@TF@plot ....... 195, 260, 262, 378, 548, 594, 683, 741
\func@mag 147, 149, 167, 179, 193, 195, 213, 225, 513, 515, 531, 532, 537, 538, 546, 548, 564, 565, 570, 571, 578, 580, 582, 583, 587, 588, 592, 594, 596, 597, 601, 602, 652, 653, 668, 673, 681, 683, 698, 703, 727, 729, 731, 735, 739, 741, 743, 747
\build@ZPK@plot ....... 149, 242, 244, 311, 515, 580, 653, 729
\currentdegree ....... 390, 391, 394–396, 398, 399, 409, 410, 413–415, 417, 418
\circ .... 170, 183, 216, 229
\den@degree .... 406, 409
\den@im .... 382, 414, 427, 430
\den@real ....
\dencoeff ....
\denpow .... 405, 406, 408, 409
\func@mag 179, 193, 195, 213, 225, 513, 515, 531, 532, 537, 538, 546, 548, 564, 565, 570, 571, 578, 580, 582, 583, 587, 588, 592, 594, 596, 597, 601, 602, 652, 653, 668, 673, 681, 683, 698, 703, 727, 729, 731, 735, 739, 741, 743, 747
\func@ph .... 148, 149, 172, 185, 194, 195, 218, 231, 514, 515, 531, 532, 537, 538, 547, 548, 564, 565, 570, 571, 579, 580, 582, 583, 587, 588, 593, 594, 596, 597, 601, 602, 652, 653, 668, 673, 682, 683, 698, 703, 728, 729, 731, 735, 740, 741, 743, 747
Change History

v1.0
General: Initial release .................... 1

v1.0.1
\BodeZPK: Pass arbitrary TikZ commands as options. .............. 22
\addBodeZPKPlots: Improved optional argument handling. ............ 24

v1.0.2
\gnuplotdegrees: Fixed issue #1 ..... 18

v1.0.3
\BodeTF: Added Tikz option .......... 23
\BodeZPK: Added Tikz option .......... 22
\NicholsTF: Added commands and tikz options ..................... 32
\NicholsZPK: Added commands and tikz options ..................... 32
\NyquistTF: Added commands and tikz options ..................... 30
\NyquistZPK: Added commands and tikz options ..................... 29
\parse@N@opt: Added commands and tikz options ..................... 32

v1.0.4
General: Fixed unintended optional argument macro expansion ...... 1

v1.0.5
\parse@opt: Fixed a bug .............. 28

v1.0.6
General: Fixed issue #3 .............. 1

v1.0.7
General: Removed unnecessary semicolons ......................... 1
Updated documentation ...................... 1