

The **bodeplot** package*

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

October 31, 2021

Contents

1	Introduction	1
2	Usage	2
2.1	Bode plots	2
2.1.1	Basic components up to first order	7
2.1.2	Basic components of the second order	8
2.2	Nyquist plots	10
2.3	Nichols charts	12
3	Implementation	13
3.1	Initialization	13
3.2	Parametric function generators for poles, zeros, gains, and delays. .	15
3.3	Second order systems.	16
3.4	Commands for Bode plots	18
3.4.1	User macros	18
3.4.2	Internal macros	22
3.5	Nyquist plots	26
3.5.1	User macros	26
3.5.2	Internal commands	29
3.6	Nichols charts	29

1 Introduction

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

$$G(s) = e^{-Ts} \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0} \quad (1)$$

*This document corresponds to **bodeplot** ?, dated ?.

and the zero-pole-gain (ZPK) form

$$G(s) = K e^{-Ts} \frac{(s - z_1)(s - z_2) \cdots (s - z_m)}{(s - p_1)(s - p_2) \cdots (s - p_n)}. \quad (2)$$

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

2 Usage

2.1 Bode plots

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

Plots the Bode plot of a transfer function given in ZPK format using the `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 `obj/typ/{opt}`, or `obj/{opt}`, or just `{opt}`. Each tuple passes options to different `pgfplots` macros that generate the group, the axes, and the plots according to:

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

used in the description of the `\BodeTF` macro below to mark the gain crossover frequency on the Bode Magnitude plot.

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

The options `{opt}` can be any `key=value` options that are supported by the `pgfplots` macros they are added to. *Linear or asymptotic approximation of transfer functions that include a transport delay is not supported.*

For example, given a transfer function

$$G(s) = 10 \frac{s(s + 0.1 + 0.5i)(s + 0.1 - 0.5i)}{(s + 0.5 + 10i)(s + 0.5 - 10i)}, \quad (3)$$

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

```
\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}
```

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

```
\BodeZPK[plot/mag/{red,thick},plot/ph/{blue,thick},
  axes/mag/{ytick distance=40,xmajorticks=true,
  xlabel={Frequency (rad/s)}},axes/ph/{ytick distance=90},
  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}
  {0.01}{100}
```

which generates the plot in Figure 2.

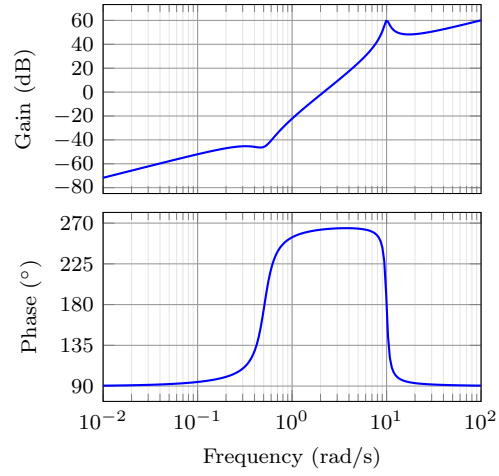


Figure 1: Output of the default `\BodeZPK` macro.

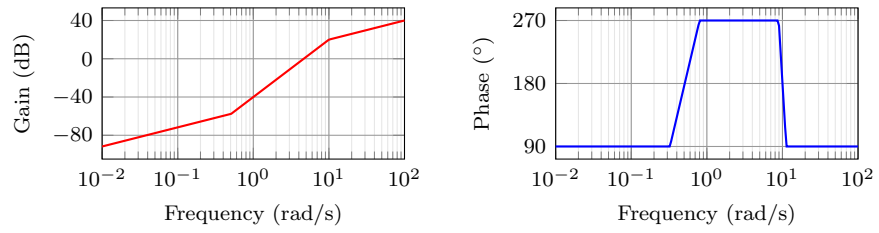


Figure 2: Customization of the default `\BodeZPK` macro.

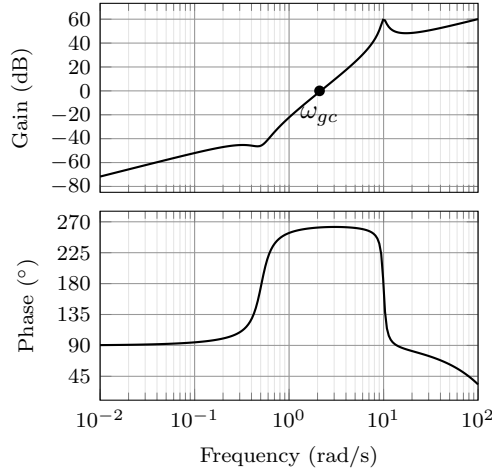


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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

For example, given the same transfer function as (3) in TF form and with a small transport delay,

$$G(s) = e^{-0.01s} \frac{s(10s^2 + 2s + 2.6)}{(s^2 + s + 100.25)}, \quad (4)$$

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

```
\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/{1,0.2,100},d/0.01}
{0.01}{100}
```

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 `\commands` option.

`BodePlot` `\begin{BodePlot} [{axis-options}] {min-frequency} {max-frequency}`

```

\addNode...
\end{NodePlot}

```

The `NodePlot` environment works in conjunction with the parametric function generator macros `\addNodeZPKPlots`, `\addNodeTFPlot`, and `\addNodeComponentPlot`. If supplied, `axis-options` are passed directly to the `semilogaxis` environment and the frequency limits are translated to the x-axis limits and the domain of the `semilogaxis` environment. Example usage in the description of `\addNodeZPKPlots`, `\addNodeTFPlot`, and `\addNodeComponentPlot`.

```

\addNodeZPKPlots \addNodeZPKPlots [approx1/{opt1}],approx2/{opt2}],...]
  {plot-type}
  {z/{zeros}},p/{poles}},k/{gain}},d/{delay}}]

```

Generates the appropriate parametric functions and supplies them to multiple `\addplot` macros, one for each `approx/{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/{opt}`, `linear/{opt}`, or `asymptotic/{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/{opt}` interface or directly in the optional argument of the `semilogaxis` environment. Use with the `NodePlot` 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 (3), its linear, asymptotic, and true Bode plots can be superimposed using

```

\begin{NodePlot}[ ylabel={Gain (dB)}, ytick distance=40,
  height=2cm, width=4cm] {0.01} {100}
\addNodeZPKPlots[
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}]
  {magnitude}
  {z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{{-0.1,-10},{-0.1,10}},k/10}
\end{NodePlot}

```

```

\begin{NodePlot}[ylabel={Phase ( $\circ$ )},
  height=2cm, width=4cm, ytick distance=90,] {0.01} {100}
\addNodeZPKPlots[
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}]
  {phase}
  {z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{{-0.1,-10},{-0.1,10}},k/10}
\end{NodePlot}

```

which generates the plot in Figure 4.

```

\addNodeTFPlot \addNodeTFPlot [plot-options]
  {plot-type}

```

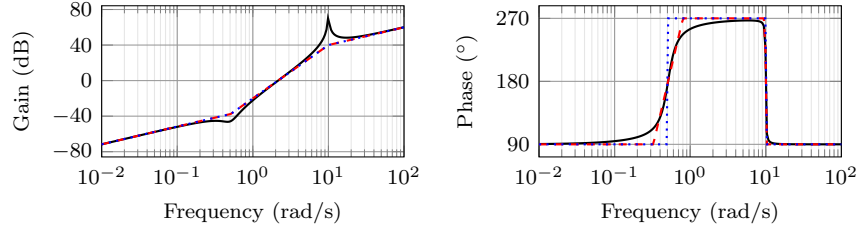


Figure 4: Superimposed approximate and true Bode plots using the `BodePlot` environment and the `\addBodeZPKPlots` macro.

$\{\langle num/\{\langle coeffs \rangle\}, den/\{\langle coeffs \rangle\}, d/\{\langle delay \rangle\} \}$

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[\langle plot-options \rangle]{\langle plot-command \rangle}`

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 2.1.1 below.

2.1.1 Basic components up to first order

`\TypeFeatureApprox`

`\TypeFeatureApprox{\langle real-part \rangle}{\langle imaginary-part \rangle}`

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

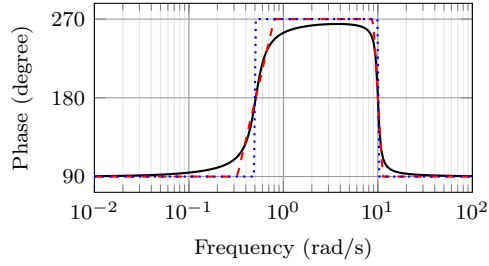


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePlot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

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 $G(s) = \frac{1}{s-a-ib}$.
- `\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 `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 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{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.

2.1.2 Basic components of the second order

`\TypeS0FeatureApprox` `\TypeS0FeatureApprox{<a1>}{<a0>}`

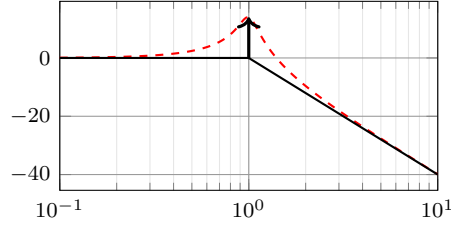


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

This entry describes 12 different macros of the form `\TypeS0FeatureApprox` 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.

`\MagS0FeaturePeak` `\MagS0FeaturePeak[$\langle draw-options \rangle$]{ $\langle a1 \rangle$ }{ $\langle a0 \rangle$ }`

This entry describes 2 different macros of the form `\MagS0FeaturePeak` 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{BodePlot}[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{BodePlot}
```

generates the plot in Figure 6.

`\TypeCSFeatureApprox` `\TypeCSFeatureApprox{ $\langle zeta \rangle$ }{ $\langle omega-n \rangle$ }`

This entry describes 12 different macros of the form `\TypeCSFeatureApprox` that take the damping ratio, ζ , and the natural frequency, ω_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.

`\MagCSFeaturePeak` `\MagCSFeaturePeak[$\langle draw-options \rangle$]{ $\langle zeta \rangle$ }{ $\langle omega-n \rangle$ }`

This entry describes 2 different macros of the form `\MagCSFeaturePeak` that take

the damping ratio, ζ , and the natural frequency, ω_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.

`\MagCCFeaturePeak`

`\MagCCFeaturePeak[$\langle draw-options \rangle$]{ $\langle real-part \rangle$ }{ $\langle imaginary-part \rangle$ }`

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.

2.2 Nyquist plots

`\NyquistZPK`

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

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 `\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` and `axes/{opt}`, which passes `{opt}` to the `axis` 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 ω 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
 $\{ \backslash arrow{Stealth [length=2mm, blue]} \}$ }}`

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[$\langle plot/\{opt\} \rangle$, $\langle axes/\{opt\} \rangle$]
 $\{ \langle num/\{coeffs\} \rangle$, $\langle den/\{coeffs\} \rangle$, $\langle d/\{delay\} \rangle \}$
 $\{ \langle min-freq \rangle \}$, $\{ \langle max-freq \rangle \}$`

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`

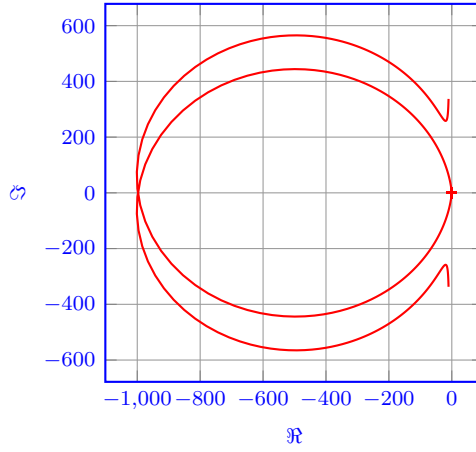


Figure 7: Output of the `\NyquistZPK` macro.

```
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}[\langle axis-options \rangle]{\langle min-frequency \rangle}{\langle max-frequency \rangle}
\addNyquist...
\end{NyquistPlot}
```

The `NyquistPlot` environment works in conjunction with the parametric function generator macros `\addNyquistZPKPlot` and `\addNyquistTFPlot`. If supplied, `axis-options` are passed directly to the `axis` environment and the frequency limits are translated to the x-axis limits and the domain of the `axis` environment.

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

Generates a two parametric functions for the magnitude and the phase 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[\langle plot-options \rangle]
{\langle num/\langle coeffs \rangle \rangle, den/\langle coeffs \rangle \rangle, d/\langle delay \rangle \rangle}
```

Similar to `\addNyquistZPKPlot`, with a transfer function input in the TF form.

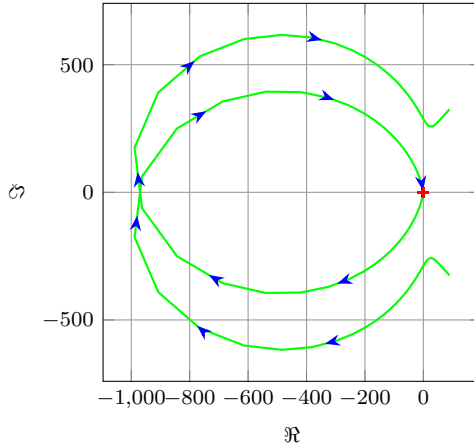


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

2.3 Nichols charts

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

Nichols chart of a transfer function given in ZPK format. Same arguments as `\NyquistZPK`.

`\NicholsTF` `\NicholsTF` [$\langle plot/\{opt\} \rangle$, $\langle axes/\{opt\} \rangle$]
 $\{ \langle num/\{coeffs\} \rangle, den/\{coeffs\} \rangle, d/\{delay\} \rangle \}$
 $\{ \langle min-freq \rangle \} \{ \langle max-freq \rangle \}$

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}{100}
```

generates the Nichols chart in Figure 9.

`NicholsChart` `\begin{NicholsChart}` [$\langle axis-options \rangle$] $\{ \langle min-frequency \rangle \} \{ \langle max-frequency \rangle \}$
`\addNichols...`
`\end{NicholsChart}`

The `NicholsChart` environment works in conjunction with the parametric function generator macros `\addNicholsZPKChart` and `\addNicholsTFChart`. If supplied, `axis-options` are passed directly to the `axis` environment and the frequency limits are translated to the x-axis limits and the domain of the `axis` environment.

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

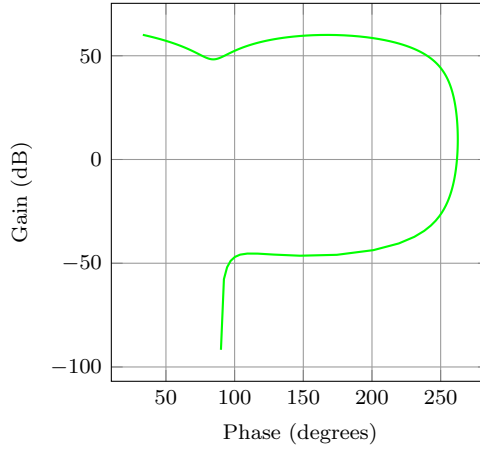


Figure 9: Output of the `\NyquistZPK` macro.

Generates a two 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[<plot-options>]  
                    {<num/{coeffs}>,den/{coeffs}>,d/{delay}>}
```

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

3 Implementation

3.1 Initialization

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

```
1 \RequirePackage{ifluatex}%  
2 \ifluatex  
3   \let\pdfstrcmp\pdf@strcmp  
4 \fi
```

`\n@mod` This code is needed to support both `pgfplots` and `gnuplot` simultaneously. New
`\n@pow` macros are defined for the `pow` and `mod` functions to address differences between
`idGnuplot` the two math engines. We start by processing the `pgf` class option.
`gnuplot def` 5 `\newif\if@pgfarg\@pgfargfalse`
`gnuplot degrees` 6 `\DeclareOption{pgf}{%`
`bodeStyle`

```

7 \pgfargtrue
8 }
9 \ProcessOptions\relax

```

Then, we define two new macros to unify pgfplots and gnuplot.

```

10 \ifpgfarg
11 \newcommand{\n@pow}[2]{(#1)^{(#2)}}%
12 \newcommand{\n@mod}[2]{mod((#1),(#2))}%
13 \else
14 \newcommand{\n@pow}[2]{(#1)**(#2)}%
15 \newcommand{\n@mod}[2]{(#1)-(floor((#1)/(#2))*(#2))}%

```

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.

```

16 \newcounter{idGnuplot}%
17 \setcounter{idGnuplot}{0}%
18 \tikzset{%
19   gnuplot def/.style={%
20     id=\arabic{idGnuplot},
21     prefix=gnuplot/
22   }%
23 }

```

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

```

24 \pgfplotsset{%
25   gnuplot degrees/.code={%
26     \ifnum\value{idGnuplot}=1
27       \xdef\pgfplots@gnuplot@format{\pgfplots@gnuplot@format set angles degrees;}%
28     \fi
29   }%
30 }
31 \fi

```

Default axis properties for all plot macros are collected in the following `pgf` style.

```

32 \pgfplotsset{%
33   bodeStyle/.style = {%
34     label style={font=\footnotesize},
35     tick label style={font=\footnotesize},
36     grid=both,
37     major grid style={color=gray!80},
38     minor grid style={color=gray!20},
39     x label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
40     y label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
41     scale only axis,
42     samples=200,
43     width=5cm,
44   }%
45 }

```

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

<code>\MagK</code>	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.
<code>\MagKAsymp</code>	
<code>\MagKLin</code>	
<code>\PhK</code>	
<code>\PhKAsymp</code>	
<code>\PhKLin</code>	<pre> 46 \newcommand*\MagK[2]{(20*log10(abs(#1)))} 47 \newcommand*\MagKAsymp{\MagK} 48 \newcommand*\MagKLin{\MagK} 49 \newcommand*\PhK[2]{(#1<0?-180:0)} 50 \newcommand*\PhKAsymp{\PhK} 51 \newcommand*\PhKLin{\PhK} </pre>
<code>\PhKAsymp</code>	True magnitude and phase parametric functions for a pure delay $G(s) = e^{-Ts}$.
<code>\PhKLin</code>	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.
	<pre> 52 \newcommand*\MagDel[2]{0} 53 \newcommand*\PhDel[2]{-#1*180*t/pi} </pre>
<code>\MagPole</code>	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.
<code>\MagPoleAsymp</code>	
<code>\MagPoleLin</code>	
<code>\PhPole</code>	
<code>\PhPoleAsymp</code>	
<code>\PhPoleLin</code>	<pre> 54 \newcommand*\MagPole[2] 55 {(-20*log10(sqrt(\n@pow{#1}{2} + \n@pow{t - (#2)}{2})))} </pre> <p>Parametric function for linear approximation of the magnitude of a complex pole.</p> <pre> 56 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ? 57 -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) : 58 -20*log10(t) 59)} </pre> <p>Parametric function for asymptotic approximation of the magnitude of a complex pole, same as linear approximation.</p> <pre> 60 \newcommand*\MagPoleAsymp{\MagPoleLin} </pre> <p>Parametric function for the true phase of a complex pole.</p> <pre> 61 \newcommand*\PhPole[2]{(#1 > 0 ? (#2 > 0 ? 62 (\n@mod{-atan2((t - (#2)),-(#1))+360}{360}) : 63 (-atan2((t - (#2)),-(#1)))) : 64 (-atan2((t - (#2)),-(#1))))} </pre> <p>Parametric function for linear approximation of the phase of a complex pole.</p> <pre> 65 \newcommand*\PhPoleLin[2]{% 66 (abs(#1)+abs(#2) == 0 ? -90 : 67 (t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) / 68 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))) ? 69 (-atan2(-(#2),-(#1))) : </pre>

```

70 (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
71 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2}))) ?
72 (#2>0?(\n@pow{#1}{2}:-90):-90) :
73 (-atan2(-(\n@pow{#1}{2}),-(\n@pow{#2}{2})) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
74 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} +
75 \n@pow{#2}{2})))))))*((\n@pow{#1}{2}:-90) + atan2(-(\n@pow{#1}{2}),-(\n@pow{#2}{2}))/
76 (log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
77 (\n@pow{#1}{2} + \n@pow{#2}{2})))))))))}

```

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

```

78 \newcommand*\PhPoleAsymp}[2]{(t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) ?
79 (-atan2(-(\n@pow{#1}{2}),-(\n@pow{#2}{2}))) :
80 (#2>0?(\n@pow{#1}{2}:-90):-90))}

```

\MagZero 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).

```

\MagZeroAsymp
\MagZeroLin 81 \newcommand*\MagZero{-\MagPole}
\PhZero 82 \newcommand*\MagZeroLin{-\MagPoleLin}
\PhZeroAsymp 83 \newcommand*\MagZeroAsymp{-\MagPoleAsymp}
\PhZeroLin 84 \newcommand*\PhZero{-\PhPole}
85 \newcommand*\PhZeroLin{-\PhPoleLin}
86 \newcommand*\PhZeroAsymp{-\PhPoleAsymp}

```

3.3 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.

```

\MagCSPoles Consider the canonical second order transfer function  $G(s) = \frac{1}{s^2 + 2\zeta\omega_n s + \omega_n^2}$ . We
\MagCSPolesAsymp start with true, linear, and asymptotic magnitude plots for this transfer function.
\MagCSPolesLin 87 \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles 88 - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2})))}
\PhCSPolesAsymp 89 \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : - 40*log10(t))}
\PhCSPolesLin 90 \newcommand*\MagCSPolesAsymp{-\MagCSPolesLin}

```

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

```

\MagCSZeros 91 \newcommand*\PhCSPoles}[2]{(-atan2((2*(#1)*(#2)*t),(\n@pow{#2}{2}
\MagCSZerosLin 92 - \n@pow{t}{2})))}
\PhCSZeros 93 \newcommand*\PhCSPolesLin}[2]{(t < (#2 / (\n@pow{10}{abs(#1)})) ?
\PhCSZerosAsymp 94 0 :
\PhCSZerosLin 95 (t >= (#2 * (\n@pow{10}{abs(#1)})) ?
96 (#1>0 ? -180 : 180) :
97 (#1>0 ? (-180*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
98 (180*(log10(t*(\n@pow{10}{abs(#1})/#2))/(2*abs(#1))))}
99 \newcommand*\PhCSPolesAsymp}[2]{(#1>0?(t<#2?0:-180):(t<#2?0:180))}

```

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).


```

100 \newcommand*\MagCSZeros{0-\MagCSPoles}
101 \newcommand*\MagCSZerosLin{0-\MagCSPolesLin}
102 \newcommand*\MagCSZerosAsymp{0-\MagCSPolesAsymp}
103 \newcommand*\PhCSZeros{0-\PhCSPoles}
104 \newcommand*\PhCSZerosLin{0-\PhCSPolesLin}
105 \newcommand*\PhCSZerosAsymp{0-\PhCSPolesAsymp}

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

106 \newcommand*\MagCSPolesPeak{3}[]{}
107 \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
108 (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
109 }
110 \newcommand*\MagCSZerosPeak{3}[]{}
111 \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
112 (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
113 }

\MagSOPoles Consider a general second order transfer function  $G(s) = \frac{1}{s^2+as+b}$ . We start with
\MagSOPolesAsymp true, linear, and asymptotic magnitude plots for this transfer function.
\MagSOPolesLin 114 \newcommand*\MagSOPoles{2}{}
\PhSOPoles 115 (-20*log10(sqrt(\n@pow{#2} - \n@pow{t}{2}){2} + \n@pow{#1*t}{2})))
\PhSOPolesAsymp 116 \newcommand*\MagSOPolesLin{2}{}
\PhSOPolesLin 117 (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t))
\MagSOPoles 118 \newcommand*\MagSOPolesAsymp{\MagSOPolesLin}

\MagSOPolesAsymp Then, we have true, linear, and asymptotic phase plots for the general second
\MagSOPolesLin order transfer function.
\PhSOPoles 119 \newcommand*\PhSOPoles{2}{(-atan2((#1)*t,((#2) - \n@pow{t}{2})))}
\PhSOPolesAsymp 120 \newcommand*\PhSOPolesLin{2}{(#2>0 ?
\PhSOPolesLin 121 \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
122 (#1>0 ? -180 : 180))}
123 \newcommand*\PhSOPolesAsymp{2}{(#2>0 ?
124 \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
125 (#1>0 ? -180 : 180))}

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).

126 \newcommand*\MagSOPoles{0-\MagSOPoles}
127 \newcommand*\MagSOPolesLin{0-\MagSOPolesLin}
128 \newcommand*\MagSOPolesAsymp{0-\MagSOPolesAsymp}
129 \newcommand*\PhSOPoles{0-\PhSOPoles}
130 \newcommand*\PhSOPolesLin{0-\PhSOPolesLin}
131 \newcommand*\PhSOPolesAsymp{0-\PhSOPolesAsymp}

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

```

```

132 \newcommand*\MagSOPolesPeak}[3][]{%
133   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
134   (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
135     20*log10(abs(#2/sqrt(abs(#3))))});
136 }
137 \newcommand*\MagS0ZerosPeak}[3][]{%
138   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
139   (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
140     20*log10(abs(#2/sqrt(abs(#3))))});
141 }

```

3.4 Commands for Bode plots

3.4.1 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`.

```

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

```

Most of the work is done by the `\parse@opt` and the `\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 `\func@mag` and `\func@ph` that hold the magnitude and phase parametric functions.

```

143   \parse@opt{#1}%
144   \gdef\func@mag{}%
145   \gdef\func@ph{}%
146   \build@ZPK@plot{\func@mag}{\func@ph}{\opt@approx}{#2}%

```

The `\noexpand` macros below are needed so that only the macro `\opt@group` is expanded.

```

147   \edef\temp@cmd{\noexpand\begin{tikzpicture}\noexpand\begin{groupplot}}[%
148     bodeStyle,
149     xmin={#3},
150     xmax={#4},
151     domain=#3:#4,
152     height=2.5cm,
153     xmode=log,
154     group style = {group size = 1 by 2,vertical sep=0.25cm,},
155     \opt@group,]
156   \temp@cmd

```

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

```

157 \if@pgfarg
158   \expandafter\nextgroupplot\expandafter[ytick distance=20,
159     ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
160   \edef\temp@cmd{\noexpand\addplot[thick,\optmag@plot]}%
161   \temp@cmd {\func@mag};
162   \optmag@commands;
163   \expandafter\nextgroupplot\expandafter[ytick distance=45,
164     ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
165   \edef\temp@cmd{\noexpand\addplot[thick,\optph@plot]}%
166   \temp@cmd {\func@ph};
167   \optph@commands;
168 \else

```

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

```

169   \stepcounter{idGnuplot}
170   \expandafter\nextgroupplot\expandafter[ytick distance=20,
171     ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
172   \edef\temp@cmd{\noexpand\addplot[thick,\optmag@plot]}%
173   \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\func@mag};
174   \optmag@commands;
175   \stepcounter{idGnuplot}
176   \expandafter\nextgroupplot\expandafter[ytick distance=45,
177     ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
178   \edef\temp@cmd{\noexpand\addplot[thick,\optph@plot]}%
179   \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\func@ph};
180   \optph@commands;
181 \fi
182 \end{groupplot}\end{tikzpicture}
183 }

```

\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.

```

184 \newcommand{\BodeTF}[4][]{%
185   \parse@opt{#1}%
186   \gdef\func@mag{}%
187   \gdef\func@ph{}%
188   \build@TF@plot{\func@mag}{\func@ph}{#2}%
189   \edef\temp@cmd{\noexpand\begin{tikzpicture}\noexpand\begin{groupplot}[%
190     bodeStyle,
191     xmin={#3},
192     xmax={#4},
193     domain=#3:#4,
194     height=2.5cm,
195     xmode=log,
196     group style = {group size = 1 by 2,vertical sep=0.25cm},
197     \opt@group,]}
198   \temp@cmd

```

```

199 \if@pgfarg
200 \expandafter\nextgroupplot\expandafter[ytick distance=20,
201 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
202 \edef\temp@cmd{\noexpand\addplot[thick,\optmag@plot]}%
203 \temp@cmd {\func@mag};
204 \optmag@commands;%
205 \expandafter\nextgroupplot\expandafter[ytick distance=45,
206 ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
207 \edef\temp@cmd{\noexpand\addplot[thick,\optph@plot]}%
208 \temp@cmd {\func@ph};
209 \optph@commands;%
210 \else
211 \stepcounter{idGnuplot}%
212 \expandafter\nextgroupplot\expandafter[ytick distance=20,
213 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
214 \edef\temp@cmd{\noexpand\addplot[thick,\optmag@plot]}%
215 \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\func@mag};
216 \optmag@commands;%
217 \stepcounter{idGnuplot}%
218 \expandafter\nextgroupplot\expandafter[ytick distance=45,
219 ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
220 \edef\temp@cmd{\noexpand\addplot[thick,\optph@plot]}%
221 \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\func@ph};
222 \optph@commands;%
223 \fi
224 \end{groupplot}\end{tikzpicture}
225 }

```

`\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.

```

226 \newcommand{\addBodeZPKPlots}[3][true/{}]{%
227 \foreach \approx/\opt in {#1} {%
228 \gdef\plot@macro{%
229 \gdef\temp@macro{%
230 \ifnum\pdfstrcmp{#2}{phase}=0
231 \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}%
232 \else
233 \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}%
234 \fi
235 \if@pgfarg
236 \edef\temp@cmd{\noexpand\addplot[thick,\opt]}%
237 \temp@cmd {\plot@macro};
238 \else
239 \stepcounter{idGnuplot}%
240 \edef\temp@cmd{\noexpand\addplot[thick,\opt]}
241 \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\plot@macro};
242 \fi
243 }%
244 }

```

`\addBodeTFPlot` 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.

```

245 \newcommand{\addBodeTFPlot}[3][thick]{%
246   \gdef\plot@macro{%
247     \gdef\temp@macro{%
248       \ifnum\pdfstrcmp{#2}{phase}=0
249         \build@TF@plot{\temp@macro}{\plot@macro}{#3}%
250       \else
251         \build@TF@plot{\plot@macro}{\temp@macro}{#3}%
252       \fi
253     \if@pgfarg
254       \addplot[#1]{\plot@macro};
255     \else
256       \stepcounter{idGnuplot}%
257       \addplot[#1] gnuplot[gnuplot degrees, gnuplot def] {\plot@macro};
258     \fi
259 }

```

`\addBodeComponentPlot` This macro is designed to issue a single `\addplot` macro capable of plotting linear combinations of the basic components described in Section 2.1.1. The only work to do here is to handle the `pgf` package option.

```

260 \newcommand{\addBodeComponentPlot}[2][thick]{%
261   \if@pgfarg
262     \addplot[#1]{#2};
263   \else
264     \stepcounter{idGnuplot}%
265     \addplot[#1] gnuplot[gnuplot degrees,gnuplot def] {#2};
266   \fi
267 }

```

`BodePlot` An environment to host macros that pass parametric functions to `\addplot` macros. Uses the defaults specified in `bodeStyle` to create a shortcut that includes the `tikzpicture` and `semilogaxis` environments.

```

268 \newenvironment{BodePlot}[3][{}]{%
269   \begin{tikzpicture}
270     \begin{semilogxaxis}[%
271       bodeStyle,
272       xmin={#2},
273       xmax={#3},
274       domain=#2:#3,
275       height=2.5cm,
276       xlabel={Frequency (rad/s)},
277       #1]
278   }{
279     \end{semilogxaxis}
280     \end{tikzpicture}
281 }

```

3.4.2 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 2.1.1 (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.

```

282 \newcommand*{\add@feature}[3]{%
283   \ifcat$\detokenize\expandafter{#1}$%
284     \xdef#1{\unexpanded\expandafter{#1 0+#2}}%
285   \else
286     \xdef#1{\unexpanded\expandafter{#1+#2}}%
287   \fi
288   \foreach \y [count=\n] in #3 {%
289     \xdef#1{\unexpanded\expandafter{#1}{\y}}%
290     \xdef\Last@LoopValue{\n}%
291   }%
292   \ifnum\Last@LoopValue=1%
293     \xdef#1{\unexpanded\expandafter{#1}{0}}%
294   \fi
295 }
```

\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 2.1.1) 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.

```

296 \newcommand{\build@ZPK@plot}[4]{%
297   \foreach \feature/\values in {#4} {%
298     \ifnum\pdfstrcmp{\feature}{z}=0
299       \foreach \z in \values {%
300         \ifnum\pdfstrcmp{#3}{linear}=0
301           \add@feature{#2}{\PhZeroLin}{\z}%
302           \add@feature{#1}{\MagZeroLin}{\z}%
303         \else
304           \ifnum\pdfstrcmp{#3}{asymptotic}=0
305             \add@feature{#2}{\PhZeroAsymp}{\z}%
306             \add@feature{#1}{\MagZeroAsymp}{\z}%
307           \else
308             \add@feature{#2}{\PhZero}{\z}%
309             \add@feature{#1}{\MagZero}{\z}%
310           \fi
311         \fi
312       }%
313     \fi
314   }
```

```

314 \ifnum\pdfstrcmp{\feature}{p}=0
315 \foreach \p in \values {%
316 \ifnum\pdfstrcmp{#3}{linear}=0
317 \add@feature{#2}{\PhPoleLin}{\p}%
318 \add@feature{#1}{\MagPoleLin}{\p}%
319 \else
320 \ifnum\pdfstrcmp{#3}{asymptotic}=0
321 \add@feature{#2}{\PhPoleAsymp}{\p}%
322 \add@feature{#1}{\MagPoleAsymp}{\p}%
323 \else
324 \add@feature{#2}{\PhPole}{\p}%
325 \add@feature{#1}{\MagPole}{\p}%
326 \fi
327 \fi
328 }%
329 \fi
330 \ifnum\pdfstrcmp{\feature}{k}=0
331 \ifnum\pdfstrcmp{#3}{linear}=0
332 \add@feature{#2}{\PhKLin}{\values}%
333 \add@feature{#1}{\MagKLin}{\values}%
334 \else
335 \ifnum\pdfstrcmp{#3}{asymptotic}=0
336 \add@feature{#2}{\PhKAsymp}{\values}%
337 \add@feature{#1}{\MagKAsymp}{\values}%
338 \else
339 \add@feature{#2}{\PhK}{\values}%
340 \add@feature{#1}{\MagK}{\values}%
341 \fi
342 \fi
343 \fi
344 \ifnum\pdfstrcmp{\feature}{d}=0
345 \ifnum\pdfstrcmp{#3}{linear}=0
346 \PackageError {bodeplot} {Linear approximation for pure delays is not
347 supported.} {Plot the true Bode plot using 'true' instead of 'linear'.}
348 \else
349 \ifnum\pdfstrcmp{#3}{asymptotic}=0
350 \PackageError {bodeplot} {Asymptotic approximation for pure delays is not
351 supported.} {Plot the true Bode plot using 'true' instead of 'asymptotic'..}
352 \else
353 \ifdim\values pt < 0pt
354 \PackageError {bodeplot} {Delay needs to be a positive number.}
355 \fi
356 \add@feature{#2}{\PhDel}{\values}%
357 \add@feature{#1}{\MagDel}{\values}%
358 \fi
359 \fi
360 \fi
361 }%
362 }

```

`\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).

```

363 \newcommand{\build@TF@plot}[3]{%
364   \gdef\num@real{0}%
365   \gdef\num@im{0}%
366   \gdef\den@real{0}%
367   \gdef\den@im{0}%
368   \gdef\loop@delay{0}%
369   \foreach \feature/\values in {#3} {%
370     \ifnum\pdfstrcmp{\feature}{num}=0
371       \foreach \numcoeff [count=\numpow] in \values {%
372         \xdef\num@degree{\numpow}%
373       }%
374       \foreach \numcoeff [count=\numpow] in \values {%
375         \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}%
376         \ifnum\currentdegree = 0
377           \xdef\num@real{\num@real+\numcoeff}%
378         \else
379           \ifodd\currentdegree
380             \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-1}{(\currentdegree-1)/2}))*%
381               (\n@pow{t}{\currentdegree}))}%
382           \else
383             \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-1}{(\currentdegree)/2}))*%
384               (\n@pow{t}{\currentdegree}))}%
385           \fi
386         \fi
387       }%
388     \fi
389     \ifnum\pdfstrcmp{\feature}{den}=0
390       \foreach \dencoeff [count=\denpow] in \values {%
391         \xdef\den@degree{\denpow}%
392       }%
393       \foreach \dencoeff [count=\denpow] in \values {%
394         \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}%
395         \ifnum\currentdegree = 0
396           \xdef\den@real{\den@real+\dencoeff}%
397         \else
398           \ifodd\currentdegree
399             \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-1}{(\currentdegree-1)/2}))*%
400               (\n@pow{t}{\currentdegree}))}%
401           \else
402             \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-1}{(\currentdegree)/2}))*%
403               (\n@pow{t}{\currentdegree}))}%
404           \fi
405         \fi
406       }%
407     \fi

```



```

408 \ifnum\pdfstrcmp{\feature}{d}=0
409 \xdef\loop@delay{\values}%
410 \fi
411 }%
412 \xdef#2{(\n@mod{atan2((\num@im),(\num@real))-atan2((\den@im),%
413 (\den@real))+360}{360}-\loop@delay*180*t/pi)}%
414 \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2})))-%
415 20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2})))}%
416 }

```

\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. The input `\obj` is either `plot`, `axes`, `group` or `approx`, and the corresponding `\opt` are passed to the `\addplot` macro, the `\nextgroupplot` macro, the `groupplot` environment, and the `\build@ZPK@plot` macros, respectively. The input tuples should not contain any macros that need to be passed to respective `pgf` macros unexpanded. If an input tuple needs to contain such a macro, the `\xdef` macros below need to be defined using `\unexpanded\expandafter{\opt}` instead of just `\opt`. For example, the `\parse@N@opt` macro in Section 3.5.2 can pass macros in its arguments, unexpanded, to `pgf` plot macros and environments, which is useful, for example, when the user wishes to add direction arrows to Nyquist plots. I did not think such a use case would be encountered when plotting Bode plots.

```

417 \newcommand{\parse@opt}[1]{%
418 \gdef\optmag@axes{}%
419 \gdef\optph@axes{}%
420 \gdef\optph@plot{}%
421 \gdef\optmag@plot{}%
422 \gdef\opt@group{}%
423 \gdef\opt@approx{}%
424 \xdef\optph@commands{}%
425 \xdef\optmag@commands{}%
426 \foreach \obj/\typ/\opt in {#1} {%
427 \ifnum\pdfstrcmp{\obj}{plot}=0
428 \ifnum\pdfstrcmp{\typ}{mag}=0
429 \xdef\optmag@plot{\optmag@plot,\opt}%
430 \else
431 \ifnum\pdfstrcmp{\typ}{ph}=0
432 \xdef\optph@plot{\optph@plot,\opt}%
433 \else
434 \xdef\optmag@plot{\optmag@plot,\opt}%
435 \xdef\optph@plot{\optph@plot,\opt}%
436 \fi
437 \fi
438 \else
439 \ifnum\pdfstrcmp{\obj}{axes}=0
440 \ifnum\pdfstrcmp{\typ}{mag}=0
441 \xdef\optmag@axes{\optmag@axes,\opt}%
442 \else
443 \ifnum\pdfstrcmp{\typ}{ph}=0

```

```

444         \xdef\optph@axes{\optph@axes,\opt}%
445     \else
446         \xdef\optmag@axes{\optmag@axes,\opt}%
447         \xdef\optph@axes{\optph@axes,\opt}%
448     \fi
449 \fi
450 \else
451     \ifnum\pdfstrcmp{\obj}{group}=0
452         \xdef\opt@group{\opt@group,\opt}%
453     \else
454         \ifnum\pdfstrcmp{\obj}{approx}=0
455             \xdef\opt@approx{\opt}%
456         \else
457             \ifnum\pdfstrcmp{\obj}{commands}=0
458                 \ifnum\pdfstrcmp{\typ}{phase}=0
459                     \xdef\optph@commands{\unexpanded\expandafter{\opt}}%
460                 \else
461                     \xdef\optmag@commands{\unexpanded\expandafter{\opt}}%
462                 \fi
463             \else
464                 \xdef\optmag@plot{\optmag@plot,\obj}%
465                 \xdef\optph@plot{\optph@plot,\obj}%
466             \fi
467         \fi
468     \fi
469 \fi
470 \fi
471 }%
472 }

```

3.5 Nyquist plots

3.5.1 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 ω . Logarithmic sampling makes it worse, perhaps inverse logarithmic sampling will help, merge requests are welcome!

```

473 \newcommand{\NyquistZPK}[4][]{%
474     \parse@N@opt{#1}%
475     \gdef\func@mag{%
476         \gdef\func@ph{%
477             \build@ZPK@plot{\func@mag}{\func@ph}{#2}%

```

```

478 \edef\temp@cmd{\noexpand\begin{tikzpicture}\noexpand\begin{axis}[%
479     bodeStyle,
480     domain=#3:#4,
481     height=5cm,
482     xlabel={$\Re$},
483     ylabel={$\Im$},
484     samples=500,
485     \opt@axes,]}%
486 \temp@cmd
487 \addplot [only marks,mark=+,thick,red] (-1 , 0);
488 \edef\temp@cmd{\noexpand\addplot [thick,\unexpanded\expandafter{\opt@plot}]}%
489 \ifpgfarg
490 \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
491             {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
492 \else
493 \stepcounter{idGnuplot}%
494 \temp@cmd gnuplot[parametric,gnuplot degrees,gnuplot def] {%
495     \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
496     \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
497 \fi
498 \end{axis}
499 \end{tikzpicture}
500 }

```

\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**.

```

501 \newcommand{\NyquistTF}[4] [] {%
502 \parse@N@opt{#1}%
503 \gdef\func@mag{}%
504 \gdef\func@ph{}%
505 \build@TF@plot{\func@mag}{\func@ph}{#2}%
506 \edef\temp@cmd{\noexpand\begin{tikzpicture}\noexpand\begin{axis}[%
507     bodeStyle,
508     domain=#3:#4,
509     height=5cm,
510     xlabel={$\Re$},
511     ylabel={$\Im$},
512     samples=500,
513     \opt@axes,]}%
514 \temp@cmd
515 \addplot [only marks,mark=+,thick,red] (-1 , 0);
516 \edef\temp@cmd{\noexpand\addplot [thick,\unexpanded\expandafter{\opt@plot}]}%
517 \ifpgfarg
518 \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
519             {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
520 \else
521 \stepcounter{idGnuplot}%
522 \temp@cmd gnuplot[parametric,gnuplot degrees,gnuplot def] {%
523     \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),

```

```

524         \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
525     \fi
526 \end{axis}
527 \end{tikzpicture}
528 }

```

\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.

```

529 \newcommand{\addNyquistZPKPlot}[2] [] {%
530   \gdef\func@mag{%
531     \gdef\func@ph{%
532       \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
533     \ifpgfarg
534       \addplot [#1] ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
535         {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
536     \else
537       \stepcounter{idGnuplot}%
538       \addplot [#1] gnuplot[parametric,gnuplot degrees,gnuplot def]{%
539         \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
540         \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
541     \fi
542 }

```

\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.

```

543 \newcommand{\addNyquistTFPlot}[2] [] {%
544   \gdef\func@mag{%
545     \gdef\func@ph{%
546       \build@TF@plot{\func@mag}{\func@ph}{#2}%
547     \ifpgfarg
548       \addplot [#1] ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
549         {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
550     \else
551       \stepcounter{idGnuplot}%
552       \addplot [#1] gnuplot[parametric,gnuplot degrees,gnuplot def]{%
553         \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
554         \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
555     \fi
556 }

```

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

```

557 \newenvironment{NyquistPlot}[3][]{%
558   \begin{tikzpicture}
559     \begin{axis}[%
560       bodeStyle,
561       height=5cm,
562       domain=#2:#3,
563       xlabel={\Re$},
564       ylabel={\Im$},
565       #1]
566     \addplot [only marks,mark=+,thick,red] (-1 , 0);
567 ]{%
568   \end{axis}
569 \end{tikzpicture}
570 }

```

3.5.2 Internal commands

\parse@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. The input **\obj** is either **plot** or **axes**, and the corresponding **\opt** are passed to the **\addplot** macro and the **axis** environment, respectively. If the input tuples contain macros, they are to be passed to respective **pgf** macros unexpanded.

```

571 \newcommand{\parse@N@opt}[1]{%
572   \gdef\opt@axes{}%
573   \gdef\opt@plot{}%
574   \foreach \obj/\opt in {#1} {%
575     \ifnum\pdfstrcmp{\obj}{axes}=0
576       \xdef\opt@axes{\unexpanded\expandafter{\opt}}%
577     \else
578       \ifnum\pdfstrcmp{\obj}{plot}=0
579         \xdef\opt@plot{\unexpanded\expandafter{\opt}}%
580       \else
581         \xdef\opt@plot{\unexpanded\expandafter{\obj}}%
582       \fi
583     \fi
584   }%
585 }

```

3.6 Nichols charts

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

```

NicholsChart 586 \newcommand{\NicholsZPK}[4][]{%
\addNicholsZPKChart 587   \parse@N@opt{#1}%
\addNicholsTFChart 588   \gdef\func@mag{}%
589   \gdef\func@ph{}%
590   \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
591   \edef\temp@cmd{\noexpand\begin{tikzpicture}\noexpand\begin{axis}[%

```

```

592         bodeStyle,
593         domain=#3:#4,
594         height=5cm,
595         xlabel={Phase (degrees)},
596         ylabel={Gain (dB)},
597         samples=500,
598         \opt@axes}}
599 \temp@cmd
600 \edef\temp@cmd{\noexpand\addplot[thick,\opt@plot]}%
601 \ifpgfarg
602     \temp@cmd ( {\func@ph} , {\func@mag} );
603 \else
604     \stepcounter{idGnuplot}%
605     \temp@cmd gnuplot[parametric, gnuplot degrees, gnuplot def]
606         { \func@ph , \func@mag };
607 \fi
608 \end{axis}
609 \end{tikzpicture}
610 }
611 \newcommand{\NicholsTF}[4][]{%
612     \parse@N@opt{#1}%
613     \gdef\func@mag{}%
614     \gdef\func@ph{}%
615     \build@TF@plot{\func@mag}{\func@ph}{#2}%
616     \edef\temp@cmd{\noexpand\begin{tikzpicture}\noexpand\begin{axis}[%
617         bodeStyle,
618         domain=#3:#4,
619         height=5cm,
620         xlabel={Phase (degrees)},
621         ylabel={Gain (dB)},
622         samples=500,
623         \opt@axes}}
624 \temp@cmd
625 \edef\temp@cmd{\noexpand\addplot[thick,\opt@plot]}%
626 \ifpgfarg
627     \temp@cmd ( {\func@ph} , {\func@mag} );
628 \else
629     \stepcounter{idGnuplot}%
630     \temp@cmd gnuplot[parametric, gnuplot degrees, gnuplot def]
631         { \func@ph , \func@mag };
632 \fi
633 \end{axis}
634 \end{tikzpicture}
635 }
636 \newenvironment{NicholsChart}[3][]{%
637     \begin{tikzpicture}
638         \begin{axis}[%
639             bodeStyle,
640             domain=#2:#3,
641             height=5cm,

```

```

642     ytick distance=20,
643     xtick distance=15,
644     xlabel={Phase (degrees)},
645     ylabel={Gain (dB)},
646     #1]
647 }{
648     \end{axis}
649 \end{tikzpicture}
650 }
651 \newcommand{\addNicholsZPKChart}[2][{}]{%
652     \gdef\func@mag{}%
653     \gdef\func@ph{}%
654     \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
655     \if@pgfarg
656         \addplot [#1] ( {\func@ph} , {\func@mag} );
657     \else
658         \stepcounter{idGnuplot}%
659         \addplot [#1] gnuplot[parametric,gnuplot degrees,gnuplot def]
660             {\func@ph} , \func@mag};
661     \fi
662 }
663 \newcommand{\addNicholsTFChart}[2][{}]{%
664     \gdef\func@mag{}%
665     \gdef\func@ph{}%
666     \build@TF@plot{\func@mag}{\func@ph}{#2}%
667     \if@pgfarg
668         \addplot [#1] ( {\func@ph} , {\func@mag} );
669     \else
670         \stepcounter{idGnuplot}%
671         \addplot [#1] gnuplot[gnuplot degrees,gnuplot def]
672             {\func@ph} , \func@mag};
673     \fi
674 }

```

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	324, 325, 332,	<code>\addNicholsZPKChart</code> <u>586</u>
<code>\@pgfargfalse</code> 5	333, 336, 337,	<code>\addNyquistTFPlot</code> . <u>543</u>
<code>\@pgfargtrue</code> 7	339, 340, 356, 357	<code>\addNyquistZPKPlot</code> . <u>529</u>
A	<code>\addBodeComponentPlot</code>	B
<code>\add@feature</code> <u>282</u> , 301, <u>260</u>	<code>BodePlot</code> (environ-
302, 305, 306,	<code>\addBodeTFPlot</code> <u>245</u>	ment) <u>268</u>
308, 309, 317,	<code>\addBodeZPKPlots</code> . . <u>226</u>	<code>\bodeStyle</code> <u>5</u>
318, 321, 322,	<code>\addNicholsTFChart</code> . <u>586</u>	<code>\BodeTF</code> <u>184</u>

\BodeZPK <u>142</u>	491, 495, 496,	\MagPole . . . <u>54</u> , 81, 325
\build@TF@plot 188,	504, 505, 518,	\MagPoleAsymp <u>54</u> , 83, 322
249, 251, <u>363</u> ,	519, 523, 524,	\MagPoleLin . <u>54</u> , 82, 318
505, 546, 615, 666	531, 532, 534,	\MagSOPoles <u>114</u>
\build@ZPK@plot 146,	535, 539, 540,	\MagSOPolesAsymp . . <u>114</u>
231, 233, <u>296</u> ,	545, 546, 548,	\MagSOPolesLin <u>114</u>
477, 532, 590, 654	549, 553, 554,	\MagSOPolesPeak . . . <u>132</u>
	589, 590, 602,	\MagSOPolesAsymp . . <u>114</u>
C	606, 614, 615,	\MagSOPolesLin <u>114</u>
\circ . 164, 177, 206, 219	627, 631, 653,	\MagSOPolesPeak . . . <u>132</u>
\currentdegree	654, 656, 660,	\MagSOPolesAsymp . . <u>114</u>
. . . . 375, 376,	665, 666, 668, 672	\MagSOPolesLin <u>114</u>
379–381, 383,		\MagSOPolesPeak . . . <u>132</u>
384, 394, 395,	G	\MagZero <u>81</u> , 309
398–400, 402, 403	\gnuplot_def <u>5</u>	\MagZeroAsymp . . <u>81</u> , 306
	\gnuplot_degrees . . . <u>5</u>	\MagZeroLin <u>81</u> , 302
D		N
\den@degree . . . 391, 394	I	\n 288, 290
\den@im 367, 399, 412, 415	\idGnuplot <u>5</u>	\n@mod <u>5</u> , 62, 412
\den@real 366,	\ifcat 283	\n@pow . <u>5</u> , 55–57, 67,
396, 402, 413, 415	\ifluatex 2	68, 70, 71, 73–
\dencoeff 390,	\ifnum 26, 230,	78, 87, 88, 91–
393, 396, 399, 402	248, 292, 298,	93, 95, 97, 98,
\denpow 390, 391, 393, 394	300, 304, 314,	115, 119, 380,
	316, 320, 330,	381, 383, 384,
E	331, 335, 344,	399, 400, 402,
environments:	345, 349, 370,	403, 414, 415,
BodePlot <u>268</u>	376, 389, 395,	490, 491, 495,
	408, 427, 428,	496, 518, 519,
F	431, 439, 440,	523, 524, 534,
\func@mag . 144, 146,	443, 451, 454,	535, 539, 540,
161, 173, 186,	457, 458, 575, 578	548, 549, 553, 554
188, 203, 215,	L	\newcounter 16
475, 477, 490,	\Last@LoopValue 290, 292	\newenvironment . . .
491, 495, 496,	\loop@delay 368, 409, 413 268, 557, 636
503, 505, 518,		\NicholsChart <u>586</u>
519, 523, 524,	M	\NicholsTF <u>586</u>
530, 532, 534,	\MagCSPoles <u>87</u>	\NicholsZPK <u>586</u>
535, 539, 540,	\MagCSPolesAsymp . . <u>87</u>	\num@degree . . . 372, 375
544, 546, 548,	\MagCSPolesLin <u>87</u>	\num@im 365, 380, 412, 414
549, 553, 554,	\MagCSPolesPeak . . . <u>106</u>	\num@real 364,
588, 590, 602,	\MagCSZeros <u>87</u>	377, 383, 412, 414
606, 613, 615,	\MagCSZerosAsymp . . <u>87</u>	\numcoeff 371,
627, 631, 652,	\MagCSZerosLin <u>87</u>	374, 377, 380, 383
654, 656, 660,	\MagCSZerosPeak . . . <u>106</u>	\numpow 371, 372, 374, 375
664, 666, 668, 672	\MagDel 52, 357	\NyquistPlot <u>557</u>
\func@ph . . 145, 146,	\MagK <u>46</u> , 340	\NyquistTF <u>501</u>
166, 179, 187,	\MagKAsymp <u>46</u> , 337	\NyquistZPK <u>473</u>
188, 208, 221,	\MagKLin <u>46</u> , 333	O
476, 477, 490,		\opt@approx 146, 423, 455

<code>\opt@axes</code> . 485, 513, 572, 576, 598, 623	349, 370, 389, 408, 427, 428, 431, 439, 440, 443, 451, 454, 457, 458, 575, 578	256, 264, 493, 521, 537, 551, 604, 629, 658, 670
<code>\opt@group</code> 155, 197, 422, 452		
<code>\opt@plot</code> 488, 516, 573, 579, 581, 600, 625	<code>\pgfplots@gnuplot@format</code> 27	T <code>\temp@cmd</code> 147, 156, 160, 161, 165, 166, 172, 173, 178, 179, 189, 198, 202, 203, 207, 208, 214, 215, 220, 221, 236, 237, 240, 241, 478, 486, 488, 490, 494, 506, 514, 516, 518, 522, 591, 599, 600, 602, 605, 616, 624, 625, 627, 630
<code>\optmag@axes</code> 159, 171, 201, 213, 418, 441, 446	<code>\pgfplotsset</code> 24, 32 <code>\PhCSPoles</code> 87 <code>\PhCSPolesAsymp</code> 87, 124 <code>\PhCSPolesLin</code> . . 87, 121 <code>\PhCSZeros</code> 87 <code>\PhCSZerosAsymp</code> . . . 87 <code>\PhCSZerosLin</code> 87 <code>\PhDel</code> 53, 356 <code>\PhK</code> 46, 339 <code>\PhKAsymp</code> . . 46, 52, 336 <code>\PhKLin</code> 46, 52, 332 <code>\PhPole</code> 54, 84, 324 <code>\PhPoleAsymp</code> 54, 86, 321 <code>\PhPoleLin</code> . 54, 85, 317 <code>\PhSOPoles</code> 114 <code>\PhSOPolesAsymp</code> . . . 114 <code>\PhSOPolesLin</code> 114 <code>\PhSOZeros</code> 114 <code>\PhSOZerosAsymp</code> . . . 114 <code>\PhSOZerosLin</code> 114 <code>\PhZero</code> 81, 308 <code>\PhZeroAsymp</code> . . . 81, 305 <code>\PhZeroLin</code> 81, 301 <code>\plot@macro</code> 228, 231, 233, 237, 241, 246, 249, 251, 254, 257	<code>\temp@macro</code> 229, 231, 233, 247, 249, 251
<code>\optmag@commands</code> 162, 174, 204, 216, 425, 461		
<code>\optmag@plot</code> . . 160, 172, 202, 214, 421, 429, 434, 464		
<code>\optph@axes</code> 164, 177, 206, 219, 419, 444, 447		V <code>\value</code> 26 <code>\values</code> . . . 297, 299, 315, 332, 333, 336, 337, 339, 340, 353, 356, 357, 369, 371, 374, 390, 393, 409
<code>\optph@commands</code> 167, 180, 209, 222, 424, 459		
<code>\optph@plot</code> . . . 165, 178, 207, 220, 420, 432, 435, 465		
P		
<code>\p</code> . . . 315, 317, 318, 321, 322, 324, 325		
<code>\parse@N@opt</code> . . 474, 502, 571, 587, 612		
<code>\parse@opt</code> 143, 185, 417, 571		
<code>\pdf@strcmp</code> 3		Y <code>\y</code> 288, 289
<code>\pdfstrcmp</code> . . 1, 230, 248, 298, 300, 304, 314, 316, 320, 330, 331, 335, 344, 345,	S <code>\setcounter</code> 17 <code>\stepcounter</code> 169, 175, 211, 217, 239,	Z <code>\z</code> . . . 299, 301, 302, 305, 306, 308, 309

Change History

v1.0	commands as options. 18
General: Initial release 1	
v1.0.1	<code>\addBodeZPKPlots</code> : Improved optional argument handling. . . 20
<code>\BodeZPK</code> : Pass arbitrary TikZ	