

# The **bodeplot** package\*

Rushikesh Kamalapurkar  
rlkamalapurkar@gmail.com

November 15, 2021

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Usage</b>	<b>2</b>
2.1	Bode plots . . . . .	2
2.1.1	Basic components up to first order . . . . .	7
2.1.2	Basic components of the second order . . . . .	9
2.2	Nyquist plots . . . . .	10
2.3	Nichols charts . . . . .	12
<b>3</b>	<b>Implementation</b>	<b>14</b>
3.1	Initialization . . . . .	14
3.2	Parametric function generators for poles, zeros, gains, and delays. . . . .	15
3.3	Second order systems. . . . .	17
3.4	Commands for Bode plots . . . . .	19
3.4.1	User macros . . . . .	19
3.4.2	Internal macros . . . . .	23
3.5	Nyquist plots . . . . .	28
3.5.1	User macros . . . . .	28
3.5.2	Internal commands . . . . .	31
3.6	Nichols charts . . . . .	31

## 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** v1.0.4, dated November 5, 2021.

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. **Limitation:** in TF form, the phase angles are always between 0 and  $360^\circ$ . 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 Usage

### 2.1 Bode plots

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

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}, \dots`. 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.
  - `tikz/{opt}`: adds options `{opt}` to the `tikzpicture` 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},
```

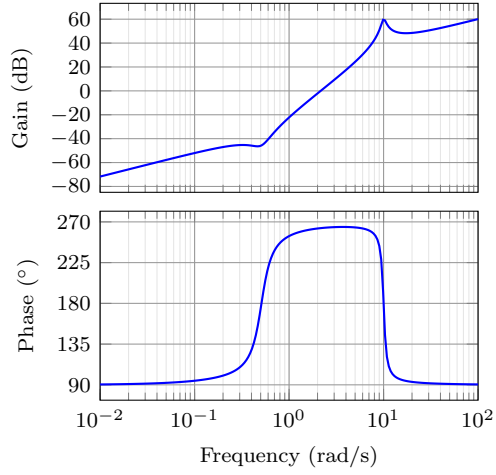


Figure 1: Output of the default `\BodeZPK` macro.

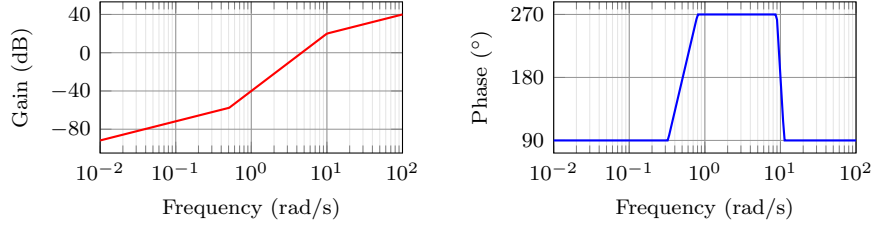


Figure 2: Customization of the default `\BodeZPK` macro.

```
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.

```
\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.

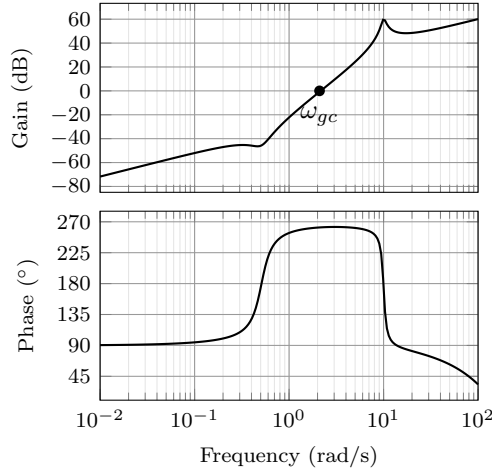


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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

The `BodePlot` environment works in conjunction with the parametric function generator macros `\addBodeZPKPlots`, `\addBodeTFPlot`, and `\addBodeComponentPlot`. 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 `semilogaxis` environment.
- `commands/{opt}`: add any valid TikZ commands inside `semilogaxis` 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 `semilogaxis` environment.

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

```
\addBodeZPKPlots \addBodeZPKPlots [(\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 `BodePlot` environment supplied with this package is recommended. The second mandatory argument, `plot-type` is either `magnitude` or `phase`. If it is not equal to `phase`, it is assumed to be `magnitude`. The last mandatory argument is the same as `\BodeZPK`.

For example, given the transfer function in (3), its linear, asymptotic, and true Bode plots can be superimposed using

```
\begin{BodePlot}[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}]
{magnitude}
{z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{{-0.1,-10},{-0.1,10}},k/10}
\end{BodePlot}

\begin{BodePlot}[ylabel={Phase ( $\sim^\circ$ )},
height=2cm, width=4cm, ytick distance=90] {0.01} {100}
\addBodeZPKPlots[
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{BodePlot}
```

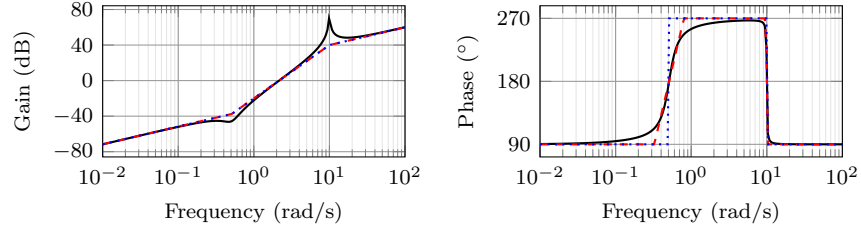


Figure 4: Superimposed approximate and true Bode plots using the `BodePlot` environment and the `\addBodeZPKPlots` macro.

which generates the plot in Figure 4.

```
\addBodeTFPlot [plot-options]
  {plot-type}
  {num/{coeffs}},den/{coeffs}},d/{delay}}
```

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 [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 2.1.1 below.

### 2.1.1 Basic components up to first order

```
\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

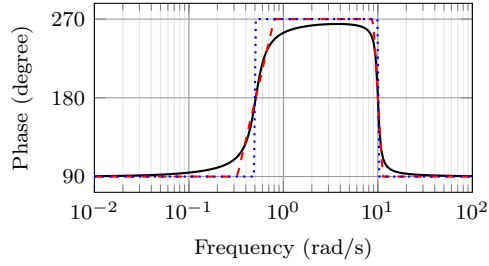


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePlot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

should either be removed, or it should be replaced by `Lin` or `Asymp` to generate the true Bode plot, the linear approximation, or the asymptotic approximation, respectively. If the `Feature` is set to `Del`, then `Approx` has to be removed. For example,

- `\MagK{k}{0}` or `\MagK{k}{400}` generates a parametric function for the true Bode magnitude of  $G(s) = k$
- `\PhPoleLin{a}{b}` generates a parametric function for the linear approximation of the Bode phase of  $G(s) = \frac{1}{s-a-ib}$ .
- `\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.



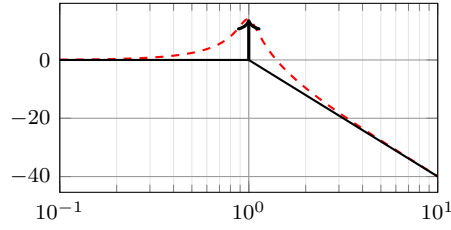


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

### 2.1.2 Basic components of the second order

`\TypeS0FeatureApprox` `\TypeS0FeatureApprox{⟨a1⟩}{⟨a0⟩}`  
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  $G(s) = \frac{1}{s^2 + a_1s + a_0}$  or  $G(s) = s^2 + a_1s + a_0$ , 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[⟨draw-options⟩]{⟨a1⟩}{⟨a0⟩}`  
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{⟨zeta⟩}{⟨omega-n⟩}`  
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  $G(s) = \frac{1}{s^2 + 2\zeta\omega_ns + \omega_n^2}$  or  $G(s) = s^2 + 2\zeta\omega_ns + \omega_n^2$ , 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.

<code>\MagCSFeaturePeak</code>	<code>\MagCSFeaturePeak[⟨draw-options⟩]{⟨zeta⟩}{⟨omega-n⟩}</code> This entry describes 2 different macros of the form <code>\MagCSFeaturePeak</code> 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 <code>\draw</code> TikZ macro. The <b>Feature</b> in the macro name should be replaced by either <b>Poles</b> or <b>Zeros</b> to generate a peak for poles and a valley for zeros, respectively.
<code>\MagCCFeaturePeak</code>	<code>\MagCCFeaturePeak[⟨draw-options⟩]{⟨real-part⟩}{⟨imaginary-part⟩}</code> This entry describes 2 different macros of the form <code>\MagCCFeaturePeak</code> 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 <code>\draw</code> TikZ macro. The <b>Feature</b> in the macro name should be replaced by either <b>Poles</b> or <b>Zeros</b> to generate a peak for poles and a valley for zeros, respectively.

## 2.2 Nyquist plots

<code>\NyquistZPK</code>	<code>\NyquistZPK [⟨plot/⟨opt⟩⟩,axes/⟨opt⟩⟩]   {⟨z/⟨zeros⟩⟩,p/⟨poles⟩⟩,k/⟨gain⟩⟩,d/⟨delay⟩⟩}   {⟨min-freq⟩}{⟨max-freq⟩}</code> 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 <code>\BodeZPK</code> . Since there is only one plot in a Nyquist diagram, the <code>\typ</code> specifier in the optional argument tuples is not needed. As such, the supported optional argument tuples are <code>plot/⟨opt⟩</code> , which passes <code>⟨opt⟩</code> to <code>\addplot</code> , <code>axes/⟨opt⟩</code> , which passes <code>⟨opt⟩</code> to the <code>axis</code> environment, and <code>tikz/⟨opt⟩</code> , which passes <code>⟨opt⟩</code> to the <code>tikzpicture</code> environment. Asymptotic/linear approximations are not supported in Nyquist plots. If just <code>⟨opt⟩</code> is provided as the optional argument, it is interpreted as <code>plot/⟨opt⟩</code> . Arrows to indicate the direction of increasing $\omega$ can be added by adding <code>\usetikzlibrary{decorations.markings}</code> and <code>\usetikzlibrary{arrows.meta}</code> to the preamble and then passing a tuple of the form <code>plot/{postaction=decorate,decoration={markings,   mark=between positions 0.1 and 0.9 step 5em with   {\arrow{Stealth [length=2mm, blue]}}}</code> <b>Caution:</b> with a high number of samples, adding arrows in this way may cause the error message ! Dimension too big. For example, the command <code>\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}</code> generates the Nyquist plot in Figure 7.
<code>\NyquistTF</code>	<code>\NyquistTF [⟨plot/⟨opt⟩⟩,axes/⟨opt⟩⟩]   {⟨num/⟨coeffs⟩⟩,den/⟨coeffs⟩⟩,d/⟨delay⟩⟩}   {⟨min-freq⟩}{⟨max-freq⟩}</code> Nyquist plot of a transfer function given in TF format. Same mandatory arguments as <code>\BodeTF</code> and same optional arguments as <code>\NyquistZPK</code> . For example, the command

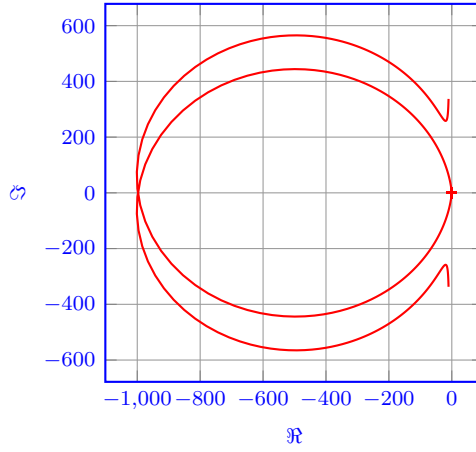


Figure 7: Output of the `\NyquistZPK` macro.

```
\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}[\langle obj1/\{opt1\}\rangle, obj2/\{opt2\}\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`. 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.

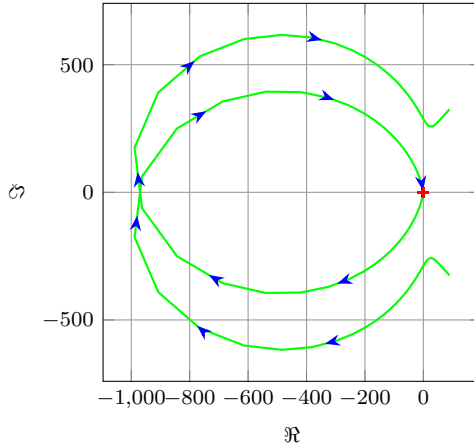


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

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

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

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

```
\addNyquistTFPlot \addNyquistTFPlot[plot-options]  
                  {num/{coeffs}},den/{coeffs}},d/{delay}}}
```

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

## 2.3 Nichols charts

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

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

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

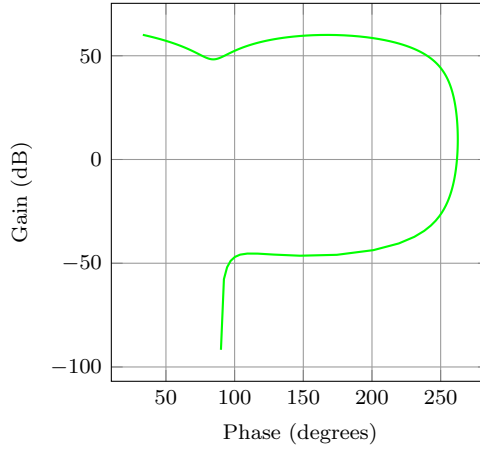


Figure 9: Output of the `\NyquistZPK` macro.

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}[{obj1/{\langle opt1 \rangle},{obj2/{\langle opt2 \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`. 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[plot-options]  
                    {z/zeros},p/poles},k/gain},d/delay}}
```

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}{%  
7   \@pgfargtrue  
8 }  
9 \ProcessOptions\relax
```

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

```
10 \if@pgfarg  
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/\jobname
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 }

```

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

```

31 \ifwindows\else
32   \immediate\write18{mkdir -p gnuplot}%
33 \fi
34 \fi

```

**bodeStyle** Default axis properties for all plot macros are collected in this `pgf` style.

```

35 \pgfplotsset{%
36   bodeStyle/.style = {%
37     label style={font=\footnotesize},
38     tick label style={font=\footnotesize},
39     grid=both,
40     major grid style={color=gray!80},
41     minor grid style={color=gray!20},
42     x label style={at={{ticklabel cs:0.5}},anchor=near ticklabel},
43     y label style={at={{ticklabel cs:0.5}},anchor=near ticklabel},
44     scale only axis,
45     samples=200,
46     width=5cm,
47   }%
48 }

```

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

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

`\MagKAsymp`

`\MagKLin`

`\PhK`

`\PhKAsymp`

`\PhKLin`

```

49 \newcommand*{\MagK}[2]{(20*log10(abs(#1)))}

```

```

50 \newcommand*\MagKAsymp{\MagK}
51 \newcommand*\MagKLin{\MagK}
52 \newcommand*\PhK[2]{\if1<0?-180:0}
53 \newcommand*\PhKAsymp{\PhK}
54 \newcommand*\PhKLin{\PhK}

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

55 \newcommand*\MagDel[2]{0}
56 \newcommand*\PhDel[2]{-#1*180*pi}

\MagPole These macros are the building blocks for most of the plotting functions provided
\MagPoleAsymp by this package. We start with Parametric function for the true magnitude of a
\MagPoleLin complex pole.
\PhPole 57 \newcommand*\MagPole[2]
\PhPoleAsymp 58 {(-20*log10(sqrt(\n@pow{#1}{2} + \n@pow{t - (#2)}{2})))}
\PhPoleLin Parametric function for linear approximation of the magnitude of a complex pole.

59 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
60 -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :
61 -20*log10(t)
62 )}

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

63 \newcommand*\MagPoleAsymp{\MagPoleLin}

Parametric function for the true phase of a complex pole.

64 \newcommand*\PhPole[2]{\if1 > 0 ? (\if2 > 0 ?
65 (\n@mod{-atan2((t - (#2)),-(#1))+360}{360}) :
66 (-atan2((t - (#2)),-(#1)))) :
67 (-atan2((t - (#2)),-(#1))))}

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

68 \newcommand*\PhPoleLin[2]{%
69 (abs(#1)+abs(#2) == 0 ? -90 :
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 (-atan2(-(#2),-(#1))) :
73 (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
74 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))) ?
75 (#2>0?(\if1>0?270:-90):-90) :
76 (-atan2(-(#2),-(#1)) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
77 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} +
78 \n@pow{#2}{2})))))))*((#2>0?(\if1>0?270:-90):-90) + atan2(-(#2),-(#1)))/
79 (log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
80 (\n@pow{#1}{2} + \n@pow{#2}{2})))))))))}

```



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

```
81 \newcommand*\PhPoleAsymp}[2]{(t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) ?
82 (-atan2(-( #2), -(#1))) :
83 (#2>0?(#1>0?270:-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 84 \newcommand*\MagZero{0-\MagPole}
\PhZero 85 \newcommand*\MagZeroLin{0-\MagPoleLin}
\PhZeroAsymp 86 \newcommand*\MagZeroAsymp{0-\MagPoleAsymp}
\PhZeroLin 87 \newcommand*\PhZero{0-\PhPole}
88 \newcommand*\PhZeroLin{0-\PhPoleLin}
89 \newcommand*\PhZeroAsymp{0-\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 90 \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles 91 - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2})))}
\PhCSPolesAsymp 92 \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : - 40*log10(t))}
\PhCSPolesLin 93 \newcommand*\MagCSPolesAsymp{\MagCSPolesLin}
\MagCSZeros Then, we have true, linear, and asymptotic phase plots for the canonical second
\MagCSZerosAsymp order transfer function.
\MagCSZerosLin 94 \newcommand*\PhCSPoles}[2]{(-atan2((2*(#1)*(#2)*t), (\n@pow{#2}{2}
\PhCSZeros 95 - \n@pow{t}{2})))}
\PhCSZerosAsymp 96 \newcommand*\PhCSPolesLin}[2]{(t < (#2 / (\n@pow{10}{abs(#1)})) ?
\PhCSZerosLin 97 0 :
98 (t >= (#2 * (\n@pow{10}{abs(#1)})) ?
99 (#1>0 ? -180 : 180) :
100 (#1>0 ? (-180*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
101 (180*(log10(t*(\n@pow{10}{abs(#1})/#2))/(2*abs(#1))))}
102 \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).

```
103 \newcommand*\MagCSZeros{0-\MagCSPoles}
104 \newcommand*\MagCSZerosLin{0-\MagCSPolesLin}
105 \newcommand*\MagCSZerosAsymp{0-\MagCSPolesAsymp}
106 \newcommand*\PhCSZeros{0-\PhCSPoles}
107 \newcommand*\PhCSZerosLin{0-\PhCSPolesLin}
108 \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.

```

109 \newcommand*\MagCSPolesPeak}[3][]{%
110   \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
111   (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
112 }
113 \newcommand*\MagCSZerosPeak}[3][]{%
114   \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
115   (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
116 }

```

`\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 117 \newcommand*\MagSOPoles}[2]{%
\PhSOPoles 118   (-20*log10(sqrt(\n@pow{#2} - \n@pow{t}{2}){2} + \n@pow{#1*t}{2})))}
\PhSOPolesAsymp 119 \newcommand*\MagSOPolesLin}[2]{%
\PhSOPolesLin 120   (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t)))}
\MagSOPoles 121 \newcommand*\MagSOPolesAsymp[\MagSOPolesLin]

```

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

```

\PhSOPoles 122 \newcommand*\PhSOPoles}[2]{(-atan2((#1)*t,((#2) - \n@pow{t}{2})))}
\PhSOPolesAsymp 123 \newcommand*\PhSOPolesLin}[2]{(#2>0 ?
\PhSOPolesLin 124   \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
125   (#1>0 ? -180 : 180))}
126 \newcommand*\PhSOPolesAsymp}[2]{(#2>0 ?
127   \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
128   (#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).

```

129 \newcommand*\MagSOPoles{0-\MagSOPoles}
130 \newcommand*\MagSOPolesLin{0-\MagSOPolesLin}
131 \newcommand*\MagSOPolesAsymp{0-\MagSOPolesAsymp}
132 \newcommand*\PhSOPoles{0-\PhSOPoles}
133 \newcommand*\PhSOPolesLin{0-\PhSOPolesLin}
134 \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.

```

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

```

```

141 \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
142 (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
143 20*log10(abs(#2/sqrt(abs(#3))))});
144 }

```

## 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`.

```

145 \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.

```

146 \parse@opt{#1}%
147 \gdef\func@mag{}%
148 \gdef\func@ph{}%
149 \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.

```

150 \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]}%
151 \noexpand\begin{groupplot}[%
152     bodeStyle,
153     xmin={#3},
154     xmax={#4},
155     domain=#3:#4,
156     height=2.5cm,
157     xmode=log,
158     group style = {group size = 1 by 2,vertical sep=0.25cm},
159     \opt@group
160 ]%
161 }%
162 \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.

```

163 \if@pgfarg
164 \expandafter\nextgroupplot\expandafter[ytick distance=20,
165 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]

```

```

166         \edef\temp@cmd{\noexpand\addplot[thick,\optmag@plot]}%
167         \temp@cmd {\func@mag};
168         \optmag@commands;
169         \expandafter\nextgroupplot\expandafter[ytick distance=45,
170             ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
171         \edef\temp@cmd{\noexpand\addplot[thick,\optph@plot]}%
172         \temp@cmd {\func@ph};
173         \optph@commands;
174     \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.

```

175     \stepcounter{idGnuplot}
176     \expandafter\nextgroupplot\expandafter[ytick distance=20,
177         ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
178     \edef\temp@cmd{\noexpand\addplot[thick,\optmag@plot]}%
179     \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\func@mag};
180     \optmag@commands;
181     \stepcounter{idGnuplot}
182     \expandafter\nextgroupplot\expandafter[ytick distance=45,
183         ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
184     \edef\temp@cmd{\noexpand\addplot[thick,\optph@plot]}%
185     \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\func@ph};
186     \optph@commands;
187 \fi
188 \end{groupplot}
189 \end{tikzpicture}
190 }

```

`\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.

```

191 \newcommand{\BodeTF}[4][]{%
192     \parse@opt{#1}%
193     \gdef\func@mag{}%
194     \gdef\func@ph{}%
195     \build@TF@plot{\func@mag}{\func@ph}{#2}%
196     \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]}%
197     \noexpand\begin{groupplot}[%
198         bodeStyle,
199         xmin={#3},
200         xmax={#4},
201         domain=#3:#4,
202         height=2.5cm,
203         xmode=log,
204         group style = {group size = 1 by 2,vertical sep=0.25cm},
205         \opt@group
206     ]%
207 }%

```

```

208 \temp@cmd
209 \if@pgfarg
210 \expandafter\nextgroupplot\expandafter[ytick distance=20,
211 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
212 \edef\temp@cmd{\noexpand\addplot[thick,\optmag@plot]}%
213 \temp@cmd {\func@mag};
214 \optmag@commands;
215 \expandafter\nextgroupplot\expandafter[ytick distance=45,
216 ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
217 \edef\temp@cmd{\noexpand\addplot[thick,\optph@plot]}%
218 \temp@cmd {\func@ph};
219 \optph@commands;
220 \else
221 \stepcounter{idGnuplot}%
222 \expandafter\nextgroupplot\expandafter[ytick distance=20,
223 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
224 \edef\temp@cmd{\noexpand\addplot[thick,\optmag@plot]}%
225 \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\func@mag};
226 \optmag@commands;
227 \stepcounter{idGnuplot}%
228 \expandafter\nextgroupplot\expandafter[ytick distance=45,
229 ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
230 \edef\temp@cmd{\noexpand\addplot[thick,\optph@plot]}%
231 \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\func@ph};
232 \optph@commands;
233 \fi
234 \end{groupplot}
235 \end{tikzpicture}
236 }

```

`\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.

```

237 \newcommand{\addBodeZPKPlots}[3][true/{}]{%
238 \foreach \approx/\opt in {#1} {%
239 \gdef\plot@macro{%
240 \gdef\temp@macro{%
241 \ifnum\pdfstrcmp{#2}{phase}=0
242 \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}%
243 \else
244 \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}%
245 \fi
246 \if@pgfarg
247 \edef\temp@cmd{\noexpand\addplot[thick,\opt]}%
248 \temp@cmd {\plot@macro};
249 \else
250 \stepcounter{idGnuplot}%
251 \edef\temp@cmd{\noexpand\addplot[thick,\opt]}
252 \temp@cmd gnuplot[gnuplot degrees,gnuplot def] {\plot@macro};
253 \fi

```

```

254 }%
255 }

```

**\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.

```

256 \newcommand{\addBodeTFPlot}[3][thick]{%
257   \gdef\plot@macro{%
258     \gdef\temp@macro{%
259       \ifnum\pdfstrcmp{#2}{phase}=0
260         \build@TF@plot{\temp@macro}{\plot@macro}{#3}%
261       \else
262         \build@TF@plot{\plot@macro}{\temp@macro}{#3}%
263       \fi
264       \ifpgfarg
265         \addplot[#1]{\plot@macro};
266       \else
267         \stepcounter{idGnuplot}%
268         \addplot[#1] gnuplot[gnuplot degrees, gnuplot def] {\plot@macro};
269       \fi
270 }

```

**\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.

```

271 \newcommand{\addBodeComponentPlot}[2][thick]{%
272   \ifpgfarg
273     \addplot[#1]{#2};
274   \else
275     \stepcounter{idGnuplot}%
276     \addplot[#1] gnuplot[gnuplot degrees,gnuplot def] {#2};
277   \fi
278 }

```

**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.

```

279 \newenvironment{BodePlot}[3][]{%
280   \parse@env@opt{#1}%
281   \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]
282     \noexpand\begin{semilogaxis}[%
283       bodeStyle,
284       xmin={#2},
285       xmax={#3},
286       domain=#2:#3,
287       height=2.5cm,
288       xlabel={Frequency (rad/s)},
289       \unexpanded\expandafter{\opt@axes}
290     ]%
291   }%

```

```

292 \temp@cmd
293 }{
294     \end{semilogxaxis}
295 \end{tikzpicture}
296 }

```

### 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](#).

```

297 \newcommand*{\add@feature}[3]{%
298 \ifcat$\detokenize\expandafter{#1}$%
299     \xdef#1{\unexpanded\expandafter{#1 0+#2}}%
300 \else
301     \xdef#1{\unexpanded\expandafter{#1+#2}}%
302 \fi
303 \foreach \y [count=\n] in #3 {%
304     \xdef#1{\unexpanded\expandafter{#1}{\y}}%
305     \xdef\Last@LoopValue{\n}%
306 }%
307 \ifnum\Last@LoopValue=1%
308     \xdef#1{\unexpanded\expandafter{#1}{0}}%
309 \fi
310 }

```

**\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.

```

311 \newcommand{\build@ZPK@plot}[4]{%
312 \foreach \feature/\values in {#4} {%
313     \ifnum\pdfstrcmp{\feature}{z}=0
314         \foreach \z in \values {%
315             \ifnum\pdfstrcmp{#3}{linear}=0
316                 \add@feature{#2}{\PhZeroLin}{\z}%
317                 \add@feature{#1}{\MagZeroLin}{\z}%
318             \else
319                 \ifnum\pdfstrcmp{#3}{asymptotic}=0
320                     \add@feature{#2}{\PhZeroAsymp}{\z}%
321                     \add@feature{#1}{\MagZeroAsymp}{\z}%

```

```

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

```



```

372      \add@feature{#1}{\MagDel}{\values}%
373      \fi
374    \fi
375  \fi
376 }%
377 }

```

`\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).

```

378 \newcommand{\build@TF@plot}[3]{%
379   \gdef\num@real{0}%
380   \gdef\num@im{0}%
381   \gdef\den@real{0}%
382   \gdef\den@im{0}%
383   \gdef\loop@delay{0}%
384   \foreach \feature/\values in {#3} {%
385     \ifnum\pdfstrcmp{\feature}{num}=0
386       \foreach \numcoeff [count=\numpow] in \values {%
387         \xdef\num@degree{\numpow}%
388       }%
389       \foreach \numcoeff [count=\numpow] in \values {%
390         \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}%
391         \ifnum\currentdegree = 0
392           \xdef\num@real{\num@real+\numcoeff}%
393         \else
394           \ifodd\currentdegree
395             \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-1}{(\currentdegree-1)/2})*%
396               (\n@pow{t}{\currentdegree}))}%
397           \else
398             \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-1}{(\currentdegree)/2})*%
399               (\n@pow{t}{\currentdegree}))}%
400           \fi
401         \fi
402       }%
403     \fi
404     \ifnum\pdfstrcmp{\feature}{den}=0
405       \foreach \dencoeff [count=\denpow] in \values {%
406         \xdef\den@degree{\denpow}%
407       }%
408       \foreach \dencoeff [count=\denpow] in \values {%
409         \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}%
410         \ifnum\currentdegree = 0
411           \xdef\den@real{\den@real+\dencoeff}%
412         \else
413           \ifodd\currentdegree
414             \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-1}{(\currentdegree-1)/2})*%
415               (\n@pow{t}{\currentdegree}))}%
416           \else

```

```

417         \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-1}{(\currentdegree)/2})*%
418         (\n@pow{t}{\currentdegree}))}%
419     \fi
420 \fi
421 }%
422 \fi
423 \ifnum\pdfstrcmp{\feature}{d}=0
424     \xdef\loop@delay{\values}%
425 \fi
426 }%
427 \xdef#2{(\n@mod{atan2((\num@im),(\num@real))-atan2((\den@im),%
428     (\den@real))+360}{360}-\loop@delay*180*t/pi)}%
429 \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2})))-%
430     20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2})))}%
431 }

```

**\parse@opt** Parses options supplied to the main Bode macros. A for loop over tuples of the form `\obj/\typ/\opt` with a long list of nested if-else statements does the job. If the input `\obj` is `plot`, `axes`, `group`, `approx`, or `tikz` the corresponding `\opt` are passed, unexpanded, to the `\addplot` macro, the `\nextgroupplot` macro, the `groupplot` environment, the `\build@ZPK@plot` macro, and the `tikzpicture` environment, respectively. If `\obj` is `commands`, the corresponding `\opt` are stored, unexpanded, in the macros `\optph@commands` and `\optmag@commands`, to be executed in appropriate axis environments.

```

432 \newcommand{\parse@opt}[1]{%
433     \gdef\optmag@axes{}%
434     \gdef\optph@axes{}%
435     \gdef\optph@plot{}%
436     \gdef\optmag@plot{}%
437     \gdef\opt@group{}%
438     \gdef\opt@approx{}%
439     \gdef\optph@commands{}%
440     \gdef\optmag@commands{}%
441     \gdef\opt@tikz{}%
442     \foreach \obj/\typ/\opt in {#1} {%
443         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{plot}=0
444             \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{mag}=0
445                 \xdef\optmag@plot{\unexpanded\expandafter{\opt}}%
446             \else
447                 \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
448                     \xdef\optph@plot{\unexpanded\expandafter{\opt}}%
449                 \else
450                     \xdef\optmag@plot{\unexpanded\expandafter{\opt}}%
451                     \xdef\optph@plot{\unexpanded\expandafter{\opt}}%
452                 \fi
453             \fi
454         \else
455             \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0
456                 \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{mag}=0

```

```

457         \xdef\optmag@axes{\unexpanded\expandafter{\opt}}%
458     \else
459         \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
460             \xdef\optph@axes{\unexpanded\expandafter{\opt}}%
461         \else
462             \xdef\optmag@axes{\unexpanded\expandafter{\opt}}%
463             \xdef\optph@axes{\unexpanded\expandafter{\opt}}%
464         \fi
465     \fi
466 \else
467     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{group}=0
468         \xdef\opt@group{\unexpanded\expandafter{\opt}}%
469     \else
470         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{approx}=0
471             \xdef\opt@approx{\unexpanded\expandafter{\opt}}%
472         \else
473             \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{commands}=0
474                 \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
475                     \xdef\optph@commands{\unexpanded\expandafter{\opt}}%
476                 \else
477                     \xdef\optmag@commands{\unexpanded\expandafter{\opt}}%
478                 \fi
479             \else
480                 \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{tikz}=0
481                     \xdef\opt@tikz{\unexpanded\expandafter{\opt}}%
482                 \else
483                     \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
484                     \unexpanded\expandafter{\obj}}%
485                     \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
486                     \unexpanded\expandafter{\obj}}%
487                 \fi
488             \fi
489         \fi
490     \fi
491 \fi
492 \fi
493 }%
494 }

```

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

```

495 \newcommand{\parse@env@opt}[1]{%
496     \gdef\opt@axes{}%
497     \gdef\opt@tikz{}%
498     \foreach \obj/\opt in {#1} {%
499         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0

```

```

500     \xdef\opt@axes{\unexpanded\expandafter{\opt}}}%
501   \else
502     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{\tikz}=0
503       \xdef\opt@tikz{\unexpanded\expandafter{\opt}}}%
504     \else
505       \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
506         \unexpanded\expandafter{\obj}}}%
507     \fi
508   \fi
509 }%
510 }

```

## 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  $\omega$ . Logarithmic sampling makes it worse, perhaps inverse logarithmic sampling will help, pull requests to fix that are welcome!

```

511 \newcommand{\NyquistZPK}[4][]{%
512   \parse@N@opt{#1}%
513   \gdef\func@mag{%
514     \gdef\func@ph{%
515       \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
516       \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]{%
517         \noexpand\begin{axis}[%
518           bodeStyle,
519           domain=#3:#4,
520           height=5cm,
521           xlabel={\Re$},
522           ylabel={\Im$},
523           samples=500,
524           \unexpanded\expandafter{\opt@axes}
525         ]%
526       }%
527       \temp@cmd
528       \addplot [only marks,mark=+,thick,red] (-1 , 0);
529       \edef\temp@cmd{\noexpand\addplot[thick,\unexpanded\expandafter{\opt@plot}]{%
530         \if@pgfarg
531           \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}}*\cos(\func@ph)},
532           {\n@pow{10}{((\func@mag)/20)}}*\sin(\func@ph)} );
533         \opt@commands;

```

```

534     \else
535         \stepcounter{idGnuplot}%
536         \temp@cmd gnuplot[parametric,gnuplot degrees,gnuplot def] {%
537             \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
538             \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
539         \opt@commands;
540     \fi
541 \end{axis}
542 \end{tikzpicture}
543 }

```

**\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**.

```

544 \newcommand{\NyquistTF}[4] [] {%
545     \parse@N@opt{#1}%
546     \gdef\func@mag{%
547     \gdef\func@ph{%
548     \build@TF@plot{\func@mag}{\func@ph}{#2}%
549     \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}}%
550         \noexpand\begin{axis}[%
551             bodeStyle,
552             domain=#3:#4,
553             height=5cm,
554             xlabel={\$Re\$},
555             ylabel={\$Im\$},
556             samples=500,
557             \unexpanded\expandafter{\opt@axes}
558         ]%
559     }%
560     \temp@cmd
561     \addplot [only marks,mark=+,thick,red] (-1 , 0);
562     \edef\temp@cmd{\noexpand\addplot[thick,\unexpanded\expandafter{\opt@plot}}}%
563     \if@pgfarg
564         \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
565             {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
566     \opt@commands;
567     \else
568         \stepcounter{idGnuplot}%
569         \temp@cmd gnuplot[parametric,gnuplot degrees,gnuplot def]{%
570             \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
571             \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
572         \opt@commands;
573     \fi
574 \end{axis}
575 \end{tikzpicture}
576 }

```

**\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.

```

577 \newcommand{\addNyquistZPKPlot}[2] [] {%
578   \gdef\func@mag{}%
579   \gdef\func@ph{}%
580   \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
581   \if@pgfarg
582     \addplot [#1] ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
583       {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
584   \else
585     \stepcounter{idGnuplot}%
586     \addplot [#1] gnuplot[parametric,gnuplot degrees,gnuplot def]{%
587       \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
588       \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
589   \fi
590 }

```

`\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.

```

591 \newcommand{\addNyquistTFPlot}[2] [] {%
592   \gdef\func@mag{}%
593   \gdef\func@ph{}%
594   \build@TF@plot{\func@mag}{\func@ph}{#2}%
595   \if@pgfarg
596     \addplot [#1] ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
597       {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
598   \else
599     \stepcounter{idGnuplot}%
600     \addplot [#1] gnuplot[parametric,gnuplot degrees,gnuplot def]{%
601       \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
602       \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
603   \fi
604 }

```

`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.

```

605 \newenvironment{NyquistPlot}[3] [] {%
606   \parse@env@opt{#1}%
607   \edef\temp@cmd{\noexpand\begin{tikzpicture}}[\unexpanded\expandafter{\opt@tikz}]%
608   \noexpand\begin{axis}%
609     bodeStyle,
610     height=5cm,
611     domain=#2:#3,
612     xlabel={\$Re\$},

```

```

613     ylabel={\Im$},
614     \unexpanded\expandafter{\opt@axes}
615   ]%
616 }%
617 \temp@cmd
618 \addplot [only marks,mark=+,thick,red] (-1 , 0);
619 }{%
620 \end{axis}
621 \end{tikzpicture}
622 }

```

### 3.5.2 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.

```

623 \newcommand{\parse@N@opt}[1]{%
624   \gdef\opt@axes{}%
625   \gdef\opt@plot{}%
626   \gdef\opt@commands{}%
627   \gdef\opt@tikz{}
628   \foreach \obj/\opt in {#1} {%
629     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0
630       \xdef\opt@axes{\unexpanded\expandafter{\opt}}%
631     \else
632       \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{plot}=0
633         \xdef\opt@plot{\unexpanded\expandafter{\opt}}%
634       \else
635         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{commands}=0
636           \xdef\opt@commands{\unexpanded\expandafter{\opt}}%
637         \else
638           \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{tikz}=0
639             \xdef\opt@tikz{\unexpanded\expandafter{\opt}}%
640           \else
641             \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
642               \unexpanded\expandafter{\obj}}%
643           \fi
644         \fi
645       \fi
646     }%
647   }%
648 }

```

### 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 649 \newcommand{\NicholsZPK}[4] [] {%
\addNicholsZPKChart 650 \parse@N@opt{#1}%
\addNicholsTFChart 651 \gdef\func@mag{%
652 \gdef\func@ph{%
653 \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
654 \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]]%
655 \noexpand\begin{axis}[%
656 bodeStyle,
657 domain=#3:#4,
658 height=5cm,
659 xlabel={Phase (degrees)},
660 ylabel={Gain (dB)},
661 samples=500,
662 \unexpanded\expandafter{\opt@axes}
663 ]%
664 }%
665 \temp@cmd
666 \edef\temp@cmd{\noexpand\addplot[thick,\opt@plot]]%
667 \ifpgfarg
668 \temp@cmd ( {\func@ph} , {\func@mag} );
669 \opt@commands;
670 \else
671 \stepcounter{idGnuplot}%
672 \temp@cmd gnuplot[parametric, gnuplot degrees, gnuplot def]
673 { \func@ph , \func@mag };
674 \opt@commands;
675 \fi
676 \end{axis}
677 \end{tikzpicture}
678 }
679 \newcommand{\NicholsTF}[4] [] {%
680 \parse@N@opt{#1}%
681 \gdef\func@mag{%
682 \gdef\func@ph{%
683 \build@TF@plot{\func@mag}{\func@ph}{#2}%
684 \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]]%
685 \noexpand\begin{axis}[%
686 bodeStyle,
687 domain=#3:#4,
688 height=5cm,
689 xlabel={Phase (degrees)},
690 ylabel={Gain (dB)},
691 samples=500,
692 \unexpanded\expandafter{\opt@axes}
693 ]%
694 }%

```



```

695 \temp@cmd
696 \edef\temp@cmd{\noexpand\addplot[thick,\opt@plot]}%
697 \if@pgfarg
698 \temp@cmd ( {\func@ph} , {\func@mag} );
699 \opt@commands;
700 \else
701 \stepcounter{idGnuplot}%
702 \temp@cmd gnuplot[parametric, gnuplot degrees, gnuplot def]
703 { \func@ph , \func@mag };
704 \opt@commands;
705 \fi
706 \end{axis}
707 \end{tikzpicture}
708 }
709 \newenvironment{NicholsChart}[3] []{%
710 \parse@env@opt{#1}%
711 \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]}%
712 \noexpand\begin{axis}[%
713 bodeStyle,
714 domain=#2:#3,
715 height=5cm,
716 xlabel={Phase (degrees)},
717 ylabel={Gain (dB)},
718 \unexpanded\expandafter{\opt@axes}
719 ]%
720 ]%
721 \temp@cmd
722 }{
723 \end{axis}
724 \end{tikzpicture}
725 }
726 \newcommand{\addNicholsZPKChart}[2] []{%
727 \gdef\func@mag{}%
728 \gdef\func@ph{}%
729 \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
730 \if@pgfarg
731 \addplot [#1] ( {\func@ph} , {\func@mag} );
732 \else
733 \stepcounter{idGnuplot}%
734 \addplot [#1] gnuplot[parametric,gnuplot degrees,gnuplot def]
735 {\func@ph , \func@mag};
736 \fi
737 }
738 \newcommand{\addNicholsTFChart}[2] []{%
739 \gdef\func@mag{}%
740 \gdef\func@ph{}%
741 \build@TF@plot{\func@mag}{\func@ph}{#2}%
742 \if@pgfarg
743 \addplot [#1] ( {\func@ph} , {\func@mag} );
744 \else

```

```

745     \stepcounter{idGnuplot}%
746     \addplot [#1] gnuplot[gnuplot degrees,gnuplot def]
747         {\func@ph , \func@mag};
748     \fi
749 }

```