

The `bodeplot` package

version 1.1.3

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

November 2, 2022

Contents

1	Introduction	2
1.1	External Dependencies	2
1.2	Directory Structure	2
1.3	Limitations	2
2	TL;DR	3
3	Usage	8
3.1	Bode plots	8
3.1.1	Basic components up to first order	12
3.1.2	Basic components of the second order	13
3.2	Nyquist plots	14
3.3	Nichols charts	16
4	Implementation	18
4.1	Initialization	18
4.2	Parametric function generators for poles, zeros, gains, and delays.	20
4.3	Second order systems.	21
4.4	Commands for Bode plots	22
4.4.1	User macros	22
4.4.2	Internal macros	28
4.5	Nyquist plots	32
4.5.1	User macros	32
4.5.2	Internal commands	34
4.6	Nichols charts	35

1 Introduction

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

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

and the zero-pole-gain (ZPK) form

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

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 $K \in \mathfrak{R}$ is the loop gain.

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

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

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

1.1 External Dependencies

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

1.2 Directory Structure

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

1.3 Limitations

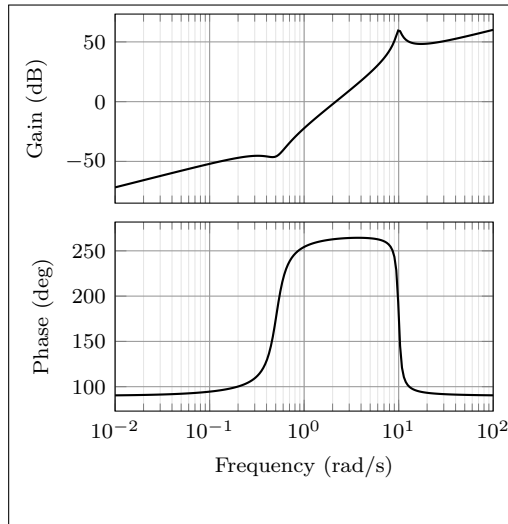
- In `pgf` mode, Bode phase plots and Nichols charts in TF form wrap angles so that they are always between 0 and 360° or 0 and 2π radian. As such, these plots will show phase wrapping discontinuities. Since v1.1.1, in `gnuplot` mode, the package uses the `smooth unwrap` filter to correct wrapping discontinuities. As of now, I have not found a way to do this in `pgf` mode, any merge requests or ideas you may have are welcome!
- Use of the `declutter` option with other directory management tools such as a `tikzexternalize` prefix is not recommended.

2 TL;DR

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

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

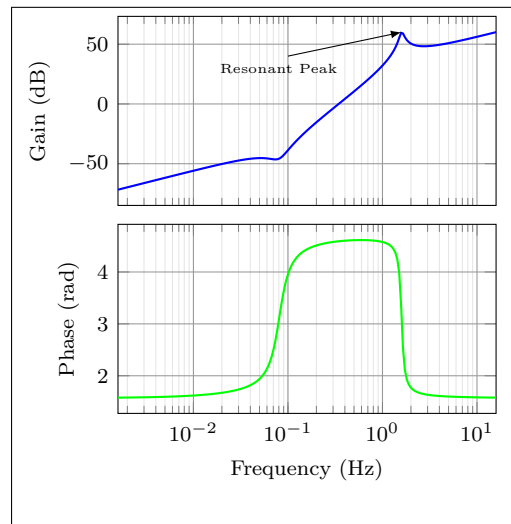
Bode plot in ZPK format



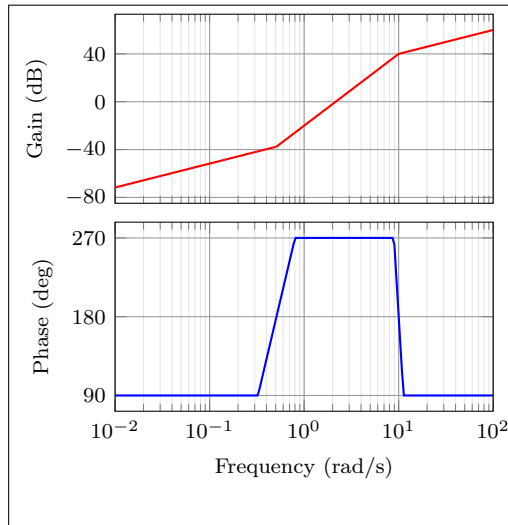
```
\BodeZPK{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

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

```
\BodeTF[%
samples=1000,
plot/mag/{blue,thick},
plot/ph/{green,thick},
tikz/{%
=>latex,
phase unit=rad,
frequency unit=Hz%
},
commands/mag/{
\draw[->](axis cs:0.1,40) -- (axis cs:{10/(2*pi)},60);
\node at (axis cs: 0.08,30) {\tiny Resonant Peak};
}%
]
{%
num/{10,2,2.6,0},
den/{1,1,100.25}%
}
{0.01/(2*pi)}
{100/(2*pi)}
```



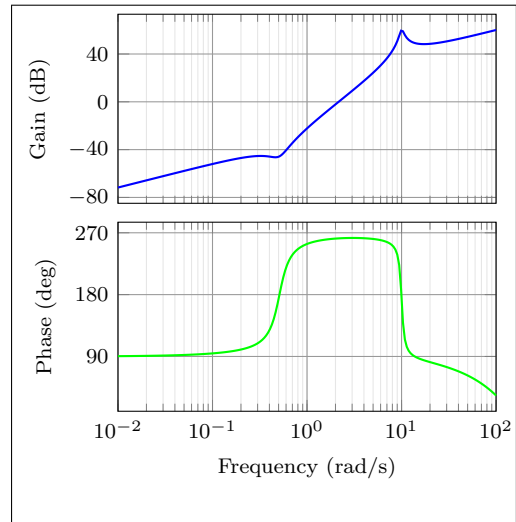
Linear approximation with customization



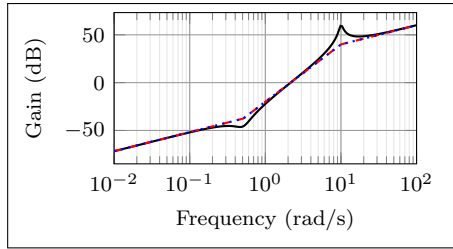
```
\BodeZPK[%
plot/mag/{red,thick},
plot/ph/{blue,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90},
approx/linear%
]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

Plot with delay and customization

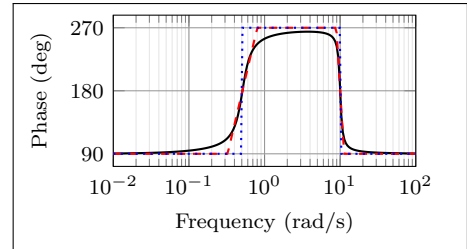
```
\BodeZPK[%
plot/mag/{blue,thick},
plot/ph/{green,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90%
}]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10,
d/0.01%
}
{0.01}
{100}
```



Individual gain and phase plots with more customization

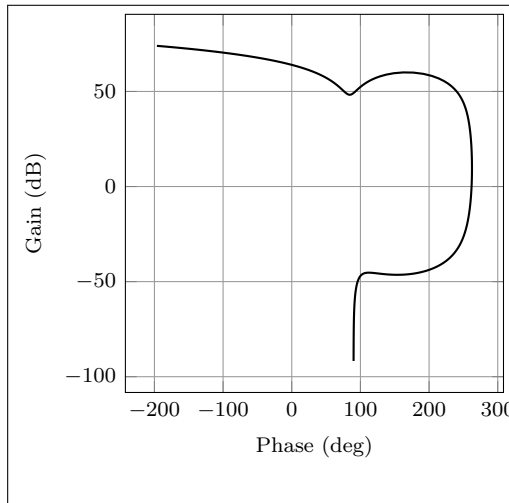


```
\begin{BodeMagPlot}[%
  axes/{height=2cm,
  width=4cm}
]
{0.01}
{100}
\addBodeZPKPlots[%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
]
{magnitude}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\end{BodeMagPlot}
```



```
\begin{BodePhPlot}[%
  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/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\end{BodePhPlot}
```

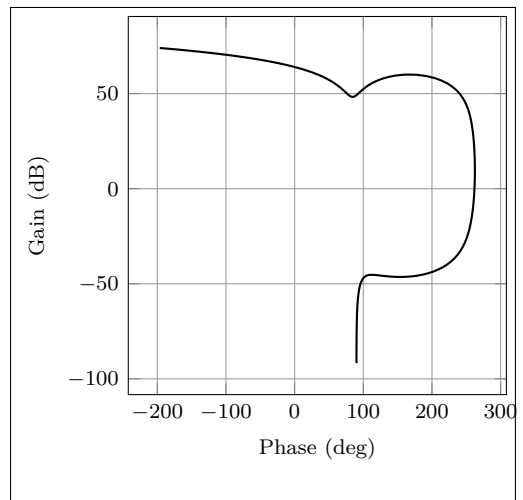
Nichols chart



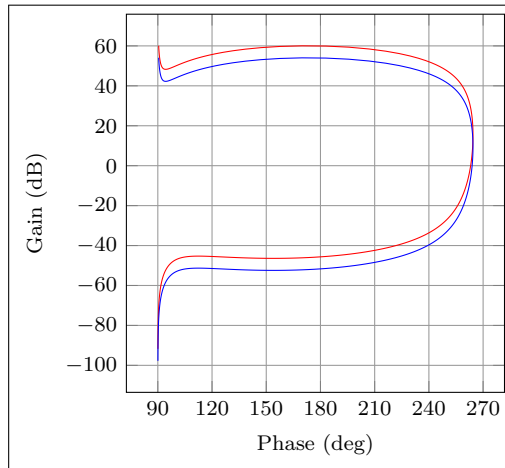
```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01%
}
{0.001}
{500}
```

Same Nichols chart in TF format (may show wrapping in pgf mode)

```
\NicholsTF[samples=1000]
{%
  num/{10,2,2.6,0},
  den/{1,1,100.25},
  d/0.01%
}
{0.001}
{500}
```



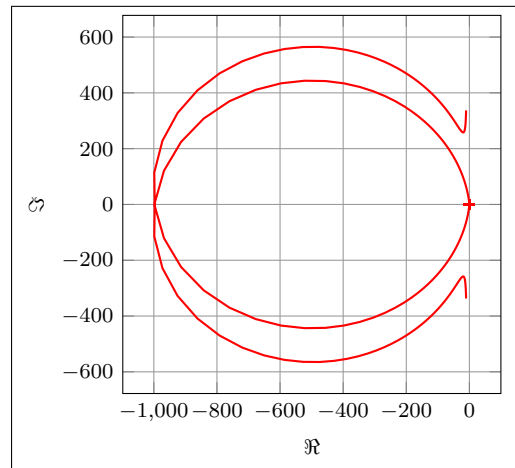
Multiple Nichols charts with customization



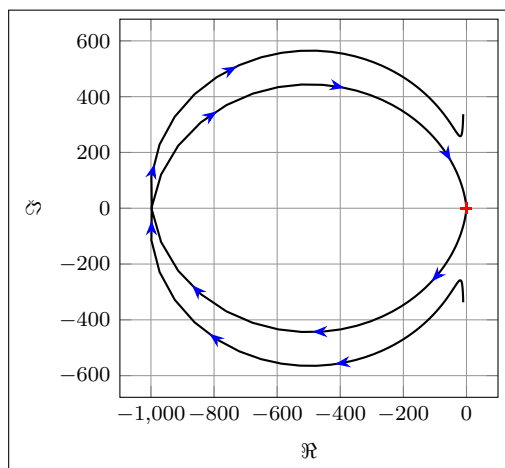
```
\begin{NicholsChart}[%
  ytick distance=20,
  xtick distance=30
]
{0.001}
{100}
\addNicholsZPKChart [red,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\addNicholsZPKChart [blue,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/5%
}
\end{NicholsChart}
```

Nyquist plot

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



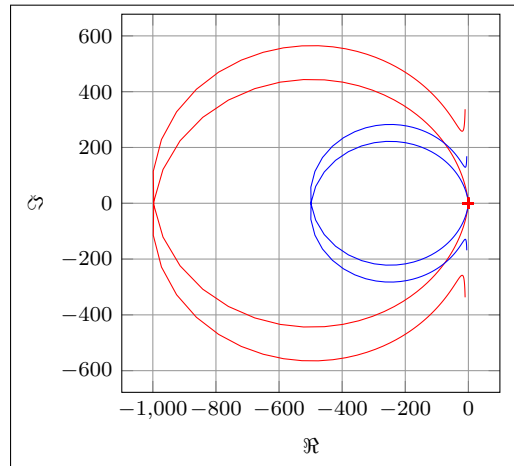
Nyquist plot in TF format with arrows



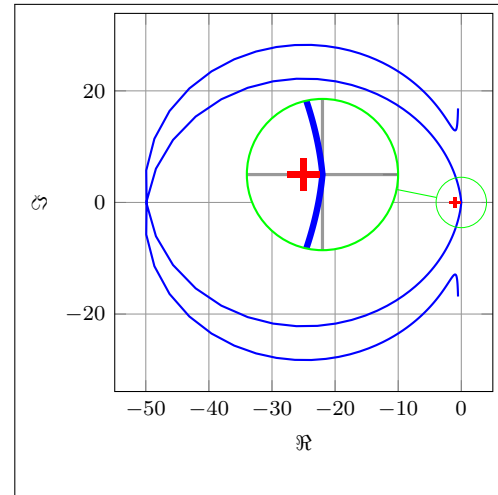
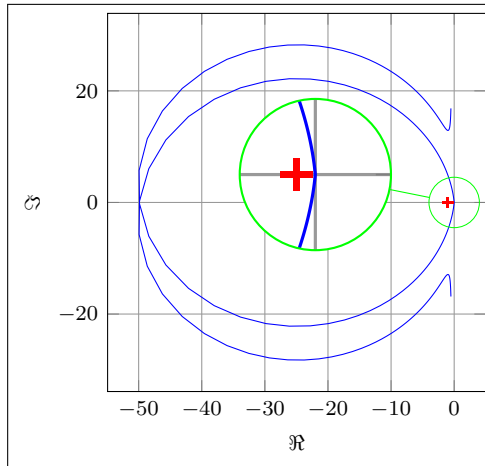
```
\NyquistTF[%
  plot/{%
    samples=1000,
    postaction=decorate,
    decoration={
      markings,
      mark=between positions 0.1 and 0.9 step 5em with {%
        \arrow{Stealth [length=2mm, blue]}
      }
    }
  }%
]
{%
  num/{10,2,2.6,0},
  den/{1,1,100.25}%
}
{-30}
{30}
```

Multiple Nyquist plots with customization

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



Nyquist plots with additional commands, using two different macros



```
\begin{NyquistPlot}{%
tikz/{
spy using outlines={%
circle,
magnification=3,
connect spies,
size=2cm
}
}%
}{-30}{30}
\addNyquistZPKPlot [blue,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/0.5%
}
\coordinate (spyon) at (axis cs:0,0);
\coordinate (spyat) at (axis cs:-22,5);
\spy [green] on (spyon) in
node [fill=white] at (spyat);
\end{NyquistPlot}
```

```
\NyquistZPK[%
plot/[blue,samples=1000],
tikz/{
spy using outlines={%
circle,
magnification=3,
connect spies,
size=2cm
}
},
commands/{
\coordinate (spyon) at (axis cs:0,0);
\coordinate (spyat) at (axis cs:-22,5);
\spy [green] on (spyon) in
node [fill=white] at (spyat);
}%
}{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/0.5%
}
}{-30}
}{30}
```

3 Usage

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

3.1 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 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 `{opt}` add all of the supplied options to `\addplot` macros for both the magnitude and the phase plots.

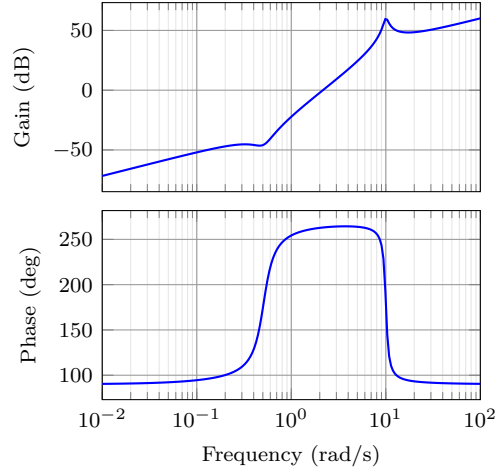


Figure 1: Output of the default `\BodeZPK` macro.

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

For example, given a transfer function

$$G(s) = 10 \frac{s(s + 0.1 + 0.5i)(s + 0.1 - 0.5i)}{(s + 0.5 + 10i)(s + 0.5 - 10i)}, \quad (4)$$

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/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.

```
\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.

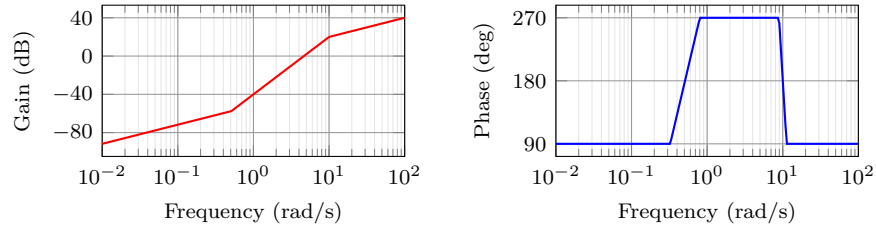


Figure 2: Customization of the default `\BodeZPK` macro.

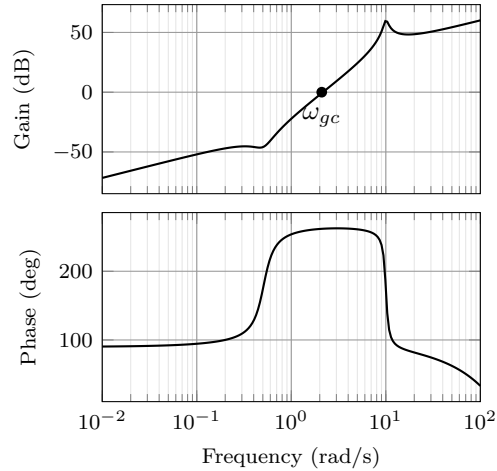


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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

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

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/{1,1,100.25},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.

```
BodeMagPlot (env.) \begin{BodeMagPlot}[\langle obj1/\langle opt1 \rangle \rangle, \langle obj2/\langle opt2 \rangle \rangle, \dots]
{\langle min-frequency \rangle}{\langle max-frequency \rangle}
\addBode...
\end{BodeMagPlot}
```

The `BodeMagPlot` environment works in conjunction with the parametric function generator macros `\addBodeZPKPlots`, `\addBodeTFPlot`, and `\addBodeComponentPlot`, intended to be used for magnitude plots. 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`.

```
BodePhPlot (env.)  \begin{BodePhPlot}[\langle obj1/\langle opt1\rangle\rangle,\langle obj2/\langle opt2\rangle\rangle,\dots]
                  {\langle min-frequency\rangle}\{\langle max-frequency\rangle}
                  \addBode...
                  \end{BodePhPlot}
```

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

```
\addBodeZPKPlots  \addBodeZPKPlots [\langle approx1/\langle opt1\rangle\rangle,\langle approx2/\langle opt2\rangle\rangle,\dots]
                  {\langle plot-type\rangle}
                  {z/\langle zeros\rangle},p/\langle poles\rangle,k/\langle gain\rangle,d/\langle delay\rangle}}
```

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 (4), its linear, asymptotic, and true Bode plots can be superimposed using

```
\begin{BodeMagPlot}[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/{{-0.5,-10},{-0.5,10}},k/10}
\end{BodeMagPlot}
```

```
\begin{BodePhPlot}[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/{{-0.5,-10},{-0.5,10}},k/10}
\end{BodePhPlot}
```

which generates the plot in Figure 4.

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

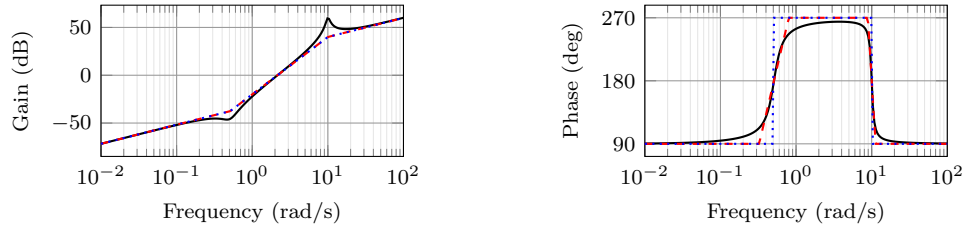


Figure 4: Superimposed approximate and true Bode plots using the `BodeMagPlot` and `BodePhPlot` environments and the `\addBodeZPKPlots` macro.

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

`\addBodeComponentPlot` `\addBodeComponentPlot`[*plot-options*]{*plot-command*}

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

3.1.1 Basic components up to first order

`\TypeFeatureApprox` `\TypeFeatureApprox`{*real-part*}{*imaginary-part*}

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

- `\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{BodePhPlot}[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} +
```

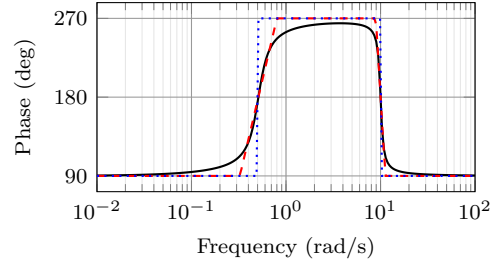


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePh-Plot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

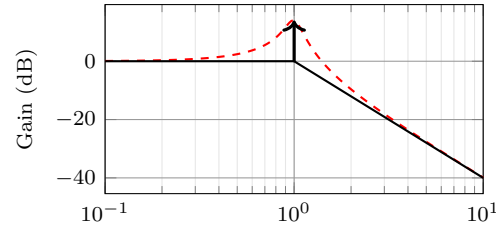


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

```

\PhZeroLin{-0.1}{-0.5} + \PhZeroLin{-0.1}{0.5} +
\PhPoleLin{-0.5}{-10} + \PhPoleLin{-0.5}{10} + \PhKLin{10}{20}
\addBodeComponentPlot[blue,dotted,thick] {\PhZeroAsymp{0}{0} +
\PhZeroAsymp{-0.1}{-0.5} + \PhZeroAsymp{-0.1}{0.5} +
\PhPoleAsymp{-0.5}{-10} + \PhPoleAsymp{-0.5}{10} + \PhKAsymp{10}{40}}
\end{BodePhPlot}

```

which gives us the plot in Figure 5.

3.1.2 Basic components of the second order

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

```

generates the plot in Figure 6.

`\TypeCSFeatureApprox` `\TypeCSFeatureApprox{<zeta>}{<omega-n>}`

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.

3.2 Nyquist plots

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

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`, `axes/{opt}`, which passes `{\opt}` to the `axis` environment, and `tikz/{opt}`, which passes `{\opt}` to the `tikzpicture` environment. Asymptotic/linear approximations are not supported in Nyquist plots. If just `{opt}` is provided as the optional argument, it is interpreted as `plot/{opt}`. Arrows to indicate the direction of increasing 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
{\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 [$\langle plot/\langle opt \rangle$], $\langle axes/\langle opt \rangle$]`
`{ $\langle num/\langle coeffs \rangle$ }, $\langle den/\langle coeffs \rangle$ }, $\langle d/\langle delay \rangle$ }}`
`{ $\langle min-freq \rangle$ }{ $\langle max-freq \rangle$ }`

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 with{\arrow{Stealth[length=2mm, blue]}}}]`

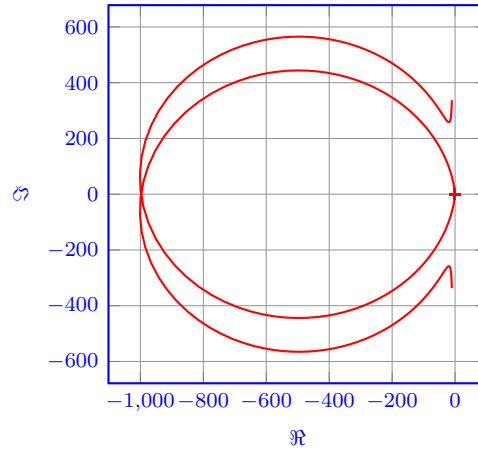


Figure 7: Output of the `\NyquistZPK` macro.

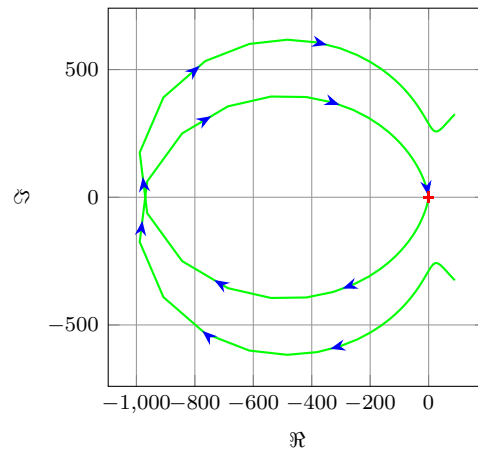


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

```
{num/{10,2,2.6,0},den/{1,1,100.25}}
{-30}{30}
```

generates the Nyquist plot in Figure 8.

```
NyquistPlot (env.) \begin{NyquistPlot}[(obj1/{opt1}],obj2/{opt2}],...]
  {<min-frequency>}{<max-frequency>}
  \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

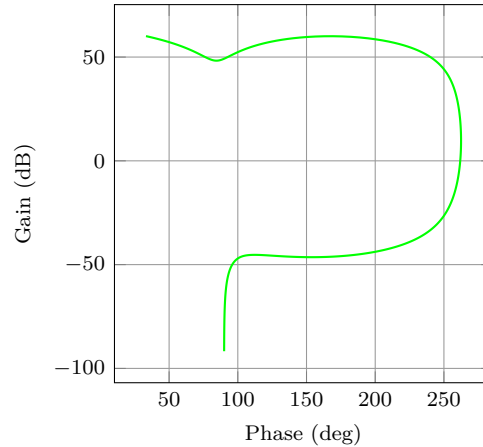


Figure 9: Output of the `\NyquistZPK` macro.

by semicolons as usual.

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

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

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

Generates a twp 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.

3.3 Nichols charts

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

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

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

Nichols chart of a transfer function given in TF format. Same arguments as `\NyquistTF`. For example, the command

```
\NicholsTF[plot/{green,thick,samples=2000}]
          {num/{10,2,2.6,0},den/{1,1,100.25},d/0.01}
          {0.001}{100}
```

generates the Nichols chart in Figure 9.

```
NicholsChart (env.) \begin{NicholsChart}[<obj1/{opt1}>,<obj2/{opt2}>,...]
                    {<min-frequency>}{<max-frequency>}
                    \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.

4 Implementation

4.1 Initialization

`\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 the
`gnuplot@id` two math engines. We start by processing the class options.

```
gnuplot@prefix 1 \newif\if@pgfarg\@pgfargfalse
                2 \DeclareOption{pgf}{
                3   \@pgfargtrue
                4 }
                5 \newif\if@declutterarg\@declutterargfalse
                6 \DeclareOption{declutter}{
                7   \@declutterargtrue
                8 }
                9 \newif\if@radarg\@radargfalse
               10 \DeclareOption{rad}{
               11   \@radargtrue
               12 }
               13 \newif\if@hzarg\@hzargfalse
               14 \DeclareOption{Hz}{
               15   \@hzargtrue
               16 }
               17 \ProcessOptions\relax
```

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

```
               18 \newcommand{\n@mod}[2]{(#1) - (floor((#1)/(#2))* (#2))}
               19 \if@pgfarg
               20   \newcommand{\n@pow}[2]{(#1)^(#2)}
               21   \pgfplotsset{
               22     trig format plots=rad
               23   }
               24 \else
               25   \newcommand{\n@pow}[2]{(#1)**(#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. The `declutter` option is used to enable the `gnuplot` directory to declutter the working directory.

```
               26   \newcounter{gnuplot@id}
               27   \setcounter{gnuplot@id}{0}
               28   \if@declutterarg
               29     \edef\bodeplot@prefix{gnuplot/\jobname}
               30   \else
               31     \edef\bodeplot@prefix{\jobname}
               32   \fi
               33   \tikzset{
               34     gnuplot@prefix/.style={
               35       id=\arabic{gnuplot@id},
               36       prefix=\bodeplot@prefix
               37     }
               38   }
```

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

```
               39   \ifwindows\else
               40     \if@declutterarg
               41       \immediate\write18{mkdir -p gnuplot}
               42     \fi
               43   \fi
               44 \fi
```

`bode@style` Default axis properties for all plot macros are collected in this `pgf` style.

```
               45 \pgfplotsset{
```

```

46 bode@style/.style = {
47   label style={font=\footnotesize},
48   tick label style={font=\footnotesize},
49   grid=both,
50   major grid style={color=gray!80},
51   minor grid style={color=gray!20},
52   x label style={at={{ticklabel cs:0.5}},anchor=near ticklabel},
53   y label style={at={{ticklabel cs:0.5}},anchor=near ticklabel},
54   scale only axis,
55   samples=200,
56   width=5cm,
57   log basis x=10
58 }
59 }

```

freq@filter These macros handle the Hz and rad class options and two new pgf keys named
freq@label frequency unit and phase unit for conversion of frequency and phase units, re-
freq@scale spectively.

```

ph@scale 60 \pgfplotsset{freq@filter/.style = {}}
ph@x@label 61 \def\freq@scale{1}
ph@y@label 62 \pgfplotsset{freq@label/.style = {xlabel = {Frequency (rad/s)}}}
63 \pgfplotsset{ph@x@label/.style = {xlabel={Phase (deg)}}}
64 \pgfplotsset{ph@y@label/.style = {ylabel={Phase (deg)}}}
65 \def\ph@scale{180/pi}
66 \if@radarg
67   \pgfplotsset{ph@y@label/.style = {ylabel={Phase (rad)}}}
68   \pgfplotsset{ph@x@label/.style = {xlabel={Phase (rad)}}}
69   \def\ph@scale{1}
70 \fi
71 \if@hzarg
72   \def\freq@scale{2*pi}
73   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}
74   \if@pgfarg
75     \pgfplotsset{freq@filter/.style = {x filter/.expression={x-
76       log10(2*pi)}}}
77   \fi
78 \tikzset{
79   phase unit/.initial={deg},
80   phase unit/.default={deg},
81   phase unit/.is choice,
82   phase unit/deg/.code={
83     \renewcommand{\ph@scale}{180/pi}
84     \pgfplotsset{ph@x@label/.style = {xlabel={Phase (deg)}}}
85     \pgfplotsset{ph@y@label/.style = {ylabel={Phase (deg)}}}
86   },
87   phase unit/rad/.code={
88     \renewcommand{\ph@scale}{1}
89     \pgfplotsset{ph@y@label/.style = {ylabel={Phase (rad)}}}
90     \pgfplotsset{ph@x@label/.style = {xlabel={Phase (rad)}}}
91   },
92   frequency unit/.initial={rad},
93   frequency unit/.default={rad},
94   frequency unit/.is choice,
95   frequency unit/Hz/.code={
96     \renewcommand{\freq@scale}{2*pi}
97     \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}
98     \if@pgfarg
99       \pgfplotsset{freq@filter/.style = {x filter/.expression={x-
100         log10(2*pi)}}}
101     \fi
102     frequency unit/rad/.code={

```

```

103   \renewcommand{\freq@scale}{1}
104   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (rad/s)}}}
105   }
106 }

```

`get@interval@start` Internal macros to extract start and end frequency limits from domain specifications.

```

get@interval@end 107 \def\get@interval@start#1:#2\@nil{#1}
108 \def\get@interval@end#1:#2\@nil{#2}

```

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

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

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

```

\PhKAsymp 109 \newcommand*\MagK[2]{(20*log10(abs(#1)))}
\PhKLin 110 \newcommand*\MagKAsymp{\MagK}
111 \newcommand*\MagKLin{\MagK}
112 \newcommand*\PhK[2]{((#1<0?-pi:0)*\ph@scale)}
113 \newcommand*\PhKAsymp{\PhK}
114 \newcommand*\PhKLin{\PhK}

```

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

```

115 \newcommand*\MagDel[2]{0}
116 \newcommand*\PhDel[2]{(-#1*t*\ph@scale)}

```

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

```

\MagPoleLin 117 \newcommand*\MagPole[2]
\PhPole 118 {(-20*log10(sqrt(\n@pow{#1}{2} + \n@pow{t - (#2)}{2})))}

```

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

```

\PhPoleLin 119 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
120 -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :
121 -20*log10(t)
122 )}

```

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

```

123 \newcommand*\MagPoleAsymp{\MagPoleLin}

```

Parametric function for the true phase of a complex pole.

```

124 \newcommand*\PhPole[2]{((#1 > 0 ? (#2 > 0 ?
125 (\n@mod{-atan2((t - (#2)), -(#1))}{2*pi}) :
126 (-atan2((t - (#2)), -(#1)))) :
127 (-atan2((t - (#2)), -(#1))))*\ph@scale)}

```

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

```

128 \newcommand*\PhPoleLin[2]{
129 ((abs(#1)+abs(#2) == 0 ? -pi/2 :
130 (t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
131 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))))) ?
132 (-atan2(-(#2), -(#1))) :
133 (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
134 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))))) ?

```

```

135 (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) :
136 (-atan2(-(#2),-(#1)) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
137 (\n@pow{10}{sqrt(\n@pow{#1}{2})/(\n@pow{#1}{2} +
138 \n@pow{#2}{2})))))))*((#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) + atan2(-
    (#2),-(#1)))/
139 (log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
140 (\n@pow{#1}{2} + \n@pow{#2}{2})))))))*\ph@scale)}

```

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

```

141 \newcommand*\PhPoleAsymp}[2]{((t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) ?
142 (-atan2(-(#2),-(#1))) :
143 (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2))*\ph@scale)}

```

`\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 144 \newcommand*\MagZero{0-\MagPole}
\PhZero 145 \newcommand*\MagZeroLin{0-\MagPoleLin}
\PhZeroAsymp 146 \newcommand*\MagZeroAsymp{0-\MagPoleAsymp}
\PhZeroLin 147 \newcommand*\PhZero{0-\PhPole}
148 \newcommand*\PhZeroLin{0-\PhPoleLin}
149 \newcommand*\PhZeroAsymp{0-\PhPoleAsymp}

```

4.3 Second order systems.

Although second order systems can be dealt with using the macros defined so far, the following dedicated macros for second order systems involve less computation.

```

\MagCSPoles Consider the canonical second order transfer function  $G(s) = \frac{1}{s^2 + 2\zeta\omega_n s + \omega_n^2}$ . We start
\MagCSPolesAsymp with true, linear, and asymptotic magnitude plots for this transfer function.
\MagCSPolesLin 150 \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles 151 - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2}))))}
\PhCSPolesAsymp 152 \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
\PhCSPolesLin 40*log10(t))}
\MagCSZeros 153 \newcommand*\MagCSPolesAsymp{\MagCSPolesLin}

```

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

```

\PhCSZeros 154 \newcommand*\PhCSPoles}[2]{((-atan2((2*(#1)*(#2)*t),(\n@pow{#2}{2}
\MagCSZerosAsymp 155 - \n@pow{t}{2}))))*\ph@scale)}
\PhCSZerosLin 156 \newcommand*\PhCSPolesLin}[2]{((t < (#2 / (\n@pow{10}{abs(#1)})) ?
157 0 :
158 (t >= (#2 * (\n@pow{10}{abs(#1)})) ?
159 (#1>0 ? -pi : pi) :
160 (#1>0 ? (-pi*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
161 (pi*(log10(t*(\n@pow{10}{abs(#1})/#2))/(2*abs(#1)))))*\ph@scale)}
162 \newcommand*\PhCSPolesAsymp}[2]{((#1>0?(t<#2?0:-
    pi):(t<#2?0:pi))*\ph@scale)}

```

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).

```

163 \newcommand*\MagCSZeros{0-\MagCSPoles}
164 \newcommand*\MagCSZerosLin{0-\MagCSPolesLin}
165 \newcommand*\MagCSZerosAsymp{0-\MagCSPolesAsymp}
166 \newcommand*\PhCSZeros{0-\PhCSPoles}
167 \newcommand*\PhCSZerosLin{0-\PhCSPolesLin}
168 \newcommand*\PhCSZerosAsymp{0-\PhCSPolesAsymp}

```

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

```

169 \newcommand*\MagCSPolesPeak}[3][[]]{
170 \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --

```

```

171 (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
172 }
173 \newcommand*\MagCSZerosPeak}[3][[]]{
174 \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
175 (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
176 }

```

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

```

\MagSOPolesAsymp
\MagSOPolesLin 177 \newcommand*\MagSOPoles}[2]{
\PhSOPoles 178 (-20*log10(sqrt(\n@pow{#2} - \n@pow{t}{2}){2} + \n@pow{#1*t}{2})))}
\PhSOPolesAsymp 179 \newcommand*\MagSOPolesLin}[2]{
\PhSOPolesLin 180 (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t))}
\MagSOPolesAsymp 181 \newcommand*\MagSOPolesAsymp}{\MagSOPolesLin}

```

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

```

\PhSOPoles 182 \newcommand*\PhSOPoles}[2]{((-atan2((#1)*t,((#2) -
\PhSOPolesAsymp \n@pow{t}{2}))) * \ph@scale)}
\PhSOPolesLin 183 \newcommand*\PhSOPolesLin}[2]{((#2>0 ?
184 \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
185 (#1>0 ? -pi : pi))}
186 \newcommand*\PhSOPolesAsymp}[2]{((#2>0 ?
187 \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
188 (#1>0 ? -pi : pi))}

```

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

```

189 \newcommand*\MagSOPoles}{0-\MagSOPoles}
190 \newcommand*\MagSOPolesLin}{0-\MagSOPolesLin}
191 \newcommand*\MagSOPolesAsymp}{0-\MagSOPolesAsymp}
192 \newcommand*\PhSOPoles}{0-\PhSOPoles}
193 \newcommand*\PhSOPolesLin}{0-\PhSOPolesLin}
194 \newcommand*\PhSOPolesAsymp}{0-\PhSOPolesAsymp}

```

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

```

195 \newcommand*\MagSOPolesPeak}[3][[]]{
196 \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
197 (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
198 20*log10(abs(#2/sqrt(abs(#3))))});
199 }
200 \newcommand*\MagSOPolesPeak}[3][[]]{
201 \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
202 (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
203 20*log10(abs(#2/sqrt(abs(#3))))});
204 }

```

4.4 Commands for Bode plots

4.4.1 User macros

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

```

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

```

206 \parse@opt{#1}
207 \gdef\func@mag{}
208 \gdef\func@ph{}
209 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
210 \temp@cmd
211 \build@ZPK@plot{\func@mag}{\func@ph}{\opt@approx}{#2}
212 \edef\temp@cmd{\noexpand\begin{groupplot}[
213     bode@style,
214     xmin=#3,
215     xmax=#4,
216     domain=#3*\freq@scale:#4*\freq@scale,
217     height=2.5cm,
218     xmode=log,
219     group style = {group size = 1 by 2,vertical sep=0.25cm},
220     \opt@group
221 ]}
222 \temp@cmd

```

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

```

223 \edef\temp@mag@cmd{\noexpand\nextgroupplot [yla-
bel={Gain (dB)}, xmajorticks=false, \optmag@axes]
224 \noexpand\addplot [freq@filter, variable=t, thick, \optmag@plot]}
225 \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
226 \noexpand\addplot [freq@filter, variable=t, thick, \optph@plot]}
227 \if@pgfarg
228 \temp@mag@cmd {\func@mag};
229 \optmag@commands
230 \temp@ph@cmd {\func@ph};
231 \optph@commands
232 \else

```

In `gnuplot` mode, we increment the `gnuplot@id` counter before every plot to make sure that new and reusable `.gnuplot` and `.table` files are generated for every plot. We use `raw gnuplot` to make sure that the tables generated by `gnuplot` use the correct phase and frequency units as supplied by the user.

```

233 \stepcounter{gnuplot@id}
234 \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
235 { set table $meta;
236     set dummy t;
237     set logscale x 10;
238     set xrange [#3*\freq@scale:#4*\freq@scale];
239     set samples \pgfkeysvalueof{/pgfplots/samples};
240     plot \func@mag;
241     set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
242     plot "$meta" using ($1/(\freq@scale)):$2;
243 };
244 \optmag@commands
245 \stepcounter{gnuplot@id}
246 \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
247 { set table $meta;
248     set dummy t;
249     set logscale x 10;
250     set xrange [#3*\freq@scale:#4*\freq@scale];
251     set samples \pgfkeysvalueof{/pgfplots/samples};

```

```

252         plot \func@ph;
253         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
254         plot "$meta" using ($1/(\freq@scale)):($2);
255     };
256     \optph@commands
257     \fi
258 \end{groupplot}
259 \end{tikzpicture}
260 }

```

\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.

```

261 \newcommand{\BodeTF}[4][]{
262   \parse@opt{#1}
263   \gdef\func@mag{}
264   \gdef\func@ph{}
265   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
266   \temp@cmd
267   \build@TF@plot{\func@mag}{\func@ph}{#2}
268   \edef\temp@cmd{\noexpand\begin{groupplot}[
269     bode@style,
270     xmin=#3,
271     xmax=#4,
272     domain=#3*\freq@scale:#4*\freq@scale,
273     height=2.5cm,
274     xmode=log,
275     group style = {group size = 1 by 2,vertical sep=0.25cm},
276     \opt@group
277   ]}
278   \temp@cmd
279   \edef\temp@mag@cmd{\noexpand\nextgroupplot [yla-
bel={Gain (dB)}, xmajor ticks=false, \optmag@axes]
280   \noexpand\addplot [freq@filter, variable=t, thick, \optmag@plot]}
281   \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
282   \noexpand\addplot [freq@filter, variable=t, thick, \optph@plot]}
283   \if@pgf@arg
284     \temp@mag@cmd {\func@mag};
285     \optmag@commands
286     \temp@ph@cmd {\n@mod{\func@ph}{2*pi*\ph@scale}};
287     \optph@commands
288   \else
289     \stepcounter{gnuplot@id}
290     \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
291     { set table $meta;
292       set dummy t;
293       set logscale x 10;
294       set xrange [#3*\freq@scale:#4*\freq@scale];
295       set samples \pgfkeysvalueof{/pgfplots/samples};
296       plot \func@mag;
297       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
298       plot "$meta" using ($1/(\freq@scale)):($2);
299     };
300     \optmag@commands
301     \stepcounter{gnuplot@id}
302     \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
303     { set table $meta;
304       set dummy t;
305       set logscale x 10;
306       set trange [#3*\freq@scale:#4*\freq@scale];
307       set samples \pgfkeysvalueof{/pgfplots/samples};
308       plot '+' using (t) : ((\func@ph)/(\ph@scale)) smooth unwrap;

```



```

309         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
310         plot "$meta" using ($1/(\freq@scale)):(($2*\ph@scale));
311     };
312     \optph@commands
313 \fi
314 \end{groupplot}
315 \end{tikzpicture}
316 }

```

\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.

```

317 \newcommand{\addBodeZPKPlots}[3][true/{}]{
318 \foreach \approx/\opt in {#1} {
319 \gdef\plot@macro{}
320 \gdef\temp@macro{}
321 \ifnum\pdf@strcmp{#2}{phase}=0
322 \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}
323 \else
324 \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}
325 \fi
326 \if@pgfarg
327 \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, thick, \opt]}
328 \temp@cmd {\plot@macro};
329 \else
330 \stepcounter{gnuplot@id}
331 \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \opt]}
332 \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
333 { set table $meta;
334 set dummy t;
335 set logscale x 10;
336 set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
337 set samples \pgfkeysvalueof{/pgfplots/samples};
338 plot \plot@macro;
339 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
340 plot "$meta" using ($1/(\freq@scale)):($2);
341 };
342 \fi
343 }
344 }

```

\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.

```

345 \newcommand{\addBodeTFPlot}[3][thick]{
346 \gdef\plot@macro{}
347 \gdef\temp@macro{}
348 \ifnum\pdf@strcmp{#2}{phase}=0
349 \build@TF@plot{\temp@macro}{\plot@macro}{#3}
350 \else
351 \build@TF@plot{\plot@macro}{\temp@macro}{#3}
352 \fi
353 \if@pgfarg
354 \ifnum\pdf@strcmp{#2}{phase}=0
355 \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, #1]}
356 \temp@cmd {\n@mod{\plot@macro}{2*pi}};
357 \else
358 \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, #1]}
359 \temp@cmd {\plot@macro};

```

```

360   \fi
361 \else
362   \stepcounter{gnuplot@id}
363   \ifnum\pdf@strcmp{#2}{phase}=0
364     \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
365     { set table $meta;
366       set dummy t;
367       set logscale x 10;
368       set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
369       set samples \pgfkeysvalueof{/pgfplots/samples};
370       plot '+' using (t) : ((\plot@macro)/(\ph@scale)) smooth unwrap;
371       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
372       plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
373     };
374 \else
375   \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
376   { set table $meta;
377     set dummy t;
378     set logscale x 10;
379     set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
380     set samples \pgfkeysvalueof{/pgfplots/samples};
381     plot \plot@macro;
382     set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
383     plot "$meta" using ($1/(\freq@scale)):($2);
384   };
385   \fi
386 \fi
387 }

```

`\addBodeComponentPlot` This macro is designed to issue a single `\addplot` macro capable of plotting linear combinations of the basic components described in Section 3.1.1. The only work to do here is to handle the `pgf` package option.

```

388 \newcommand{\addBodeComponentPlot}[2][thick]{
389   \if@pgfarg
390     \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
391       main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
392       able=t, #1]}
393     \temp@cmd {#2};
394   \else
395     \stepcounter{gnuplot@id}
396     \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
397     { set table $meta;
398       set dummy t;
399       set logscale x 10;
400       set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
401       set samples \pgfkeysvalueof{/pgfplots/samples};
402       plot #2;
403       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
404       plot "$meta" using ($1/(\freq@scale)):($2);
405     };
406   \fi
407 }

```

`BodePhPlot` (*env.*) An environment to host phase plot macros that pass parametric functions to `\addplot` macros. Uses the defaults specified in `bode@style` to create a shortcut that includes the `tikzpicture` and `semilogaxis` environments.

```

406 \NewEnviron{BodePhPlot}[3][{}]{
407   \parse@env@opt{#1}
408   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
409     panded\expandafter{\opt@tikz}]
410     \noexpand\begin{semilogaxis}[
411       ph@y@label,
412       freq@label,

```

```

412     bode@style,
413     xmin={#2},
414     xmax={#3},
415     domain=#2:#3,
416     height=2.5cm,
417     \unexpanded\expandafter{\opt@axes}
418   ]
419 }
420 \temp@cmd
421   \BODY
422 \end{semilogxaxis}
423 \end{tikzpicture}
424 }

```

BodeMagPlot (*env.*) An environment to host magnitude plot macros that pass parametric functions to `\addplot` macros. Uses the defaults specified in `bode@style` to create a shortcut that includes the `tikzpicture` and `semilogxaxis` environments.

```

425 \NewEnviron{BodeMagPlot}[3][]{
426   \parse@env@opt{#1}
427   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
428     panded\expandafter{\opt@tikz}]
429     \noexpand\begin{semilogxaxis}[
430       bode@style,
431       freq@label,
432       xmin={#2},
433       xmax={#3},
434       domain=#2:#3,
435       height=2.5cm,
436       ylabel={Gain (dB)},
437       \unexpanded\expandafter{\opt@axes}
438     ]
439   }
440   \temp@cmd
441   \BODY
442   \end{semilogxaxis}
443   \end{tikzpicture}
444 }

```

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

```

444 \NewEnviron{BodePlot}[3][]{
445   \parse@env@opt{#1}
446   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
447     panded\expandafter{\opt@tikz}]
448     \noexpand\begin{semilogxaxis}[
449       bode@style,
450       freq@label,
451       xmin={#2},
452       xmax={#3},
453       domain=#2:#3,
454       height=2.5cm,
455       \unexpanded\expandafter{\opt@axes}
456     ]
457   }
458   \temp@cmd
459   \BODY
460   \end{semilogxaxis}
461   \end{tikzpicture}
462 }

```

4.4.2 Internal macros

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

```

462 \newcommand*\add@feature}[3]{
463   \ifcat$\detokenize\expandafter{#1}$
464   \xdef#1{\unexpanded\expandafter{#1 0+#2}}
465   \else
466   \xdef#1{\unexpanded\expandafter{#1+#2}}
467   \fi
468   \foreach \y [count=\n] in #3 {
469     \xdef#1{\unexpanded\expandafter{#1}{\y}}
470     \xdef\Last@LoopValue{\n}
471   }
472   \ifnum\Last@LoopValue=1
473     \xdef#1{\unexpanded\expandafter{#1}{0}}
474   \fi
475 }

```

`\build@ZPK@plot` This is an internal macro to build parametric Bode magnitude and phase plots by concatenating basic component (pole, zero, gain, or delay) macros (Section 3.1.1) to global magnitude and phase macros (inputs #1 and #2). The `\add@feature` macro is used to do the concatenation. The basic component macros are inferred from a `feature/{values}` list, where `feature` is one of `z,p,k`, and `d`, for zeros, poles, gain, and delay, respectively, and `{values}` is a comma separated list of comma separated lists (complex numbers of the form $\{re, im\}$). If the imaginary part is missing, it is assumed to be zero.

```

476 \newcommand{\build@ZPK@plot}[4]{
477   \foreach \feature/\values in {#4} {
478     \ifnum\pdf@strcmp{\feature}{z}=0
479       \foreach \z in \values {
480         \ifnum\pdf@strcmp{#3}{linear}=0
481           \add@feature{#2}{\PhZeroLin}{\z}
482           \add@feature{#1}{\MagZeroLin}{\z}
483         \else
484           \ifnum\pdf@strcmp{#3}{asymptotic}=0
485             \add@feature{#2}{\PhZeroAsymp}{\z}
486             \add@feature{#1}{\MagZeroAsymp}{\z}
487           \else
488             \add@feature{#2}{\PhZero}{\z}
489             \add@feature{#1}{\MagZero}{\z}
490           \fi
491         \fi
492       }
493     \fi
494     \ifnum\pdf@strcmp{\feature}{p}=0
495       \foreach \p in \values {
496         \ifnum\pdf@strcmp{#3}{linear}=0
497           \add@feature{#2}{\PhPoleLin}{\p}
498           \add@feature{#1}{\MagPoleLin}{\p}
499         \else
500           \ifnum\pdf@strcmp{#3}{asymptotic}=0
501             \add@feature{#2}{\PhPoleAsymp}{\p}
502             \add@feature{#1}{\MagPoleAsymp}{\p}
503           \else
504             \add@feature{#2}{\PhPole}{\p}
505             \add@feature{#1}{\MagPole}{\p}
506           \fi
507         \fi

```

```

508     }
509   \fi
510   \ifnum\pdf@strcmp{\feature}{k}=0
511     \ifnum\pdf@strcmp{#3}{\linear}=0
512       \add@feature{#2}{\PhKLin}{\values}
513       \add@feature{#1}{\MagKLin}{\values}
514     \else
515       \ifnum\pdf@strcmp{#3}{\asymptotic}=0
516         \add@feature{#2}{\PhKAsymp}{\values}
517         \add@feature{#1}{\MagKAsymp}{\values}
518       \else
519         \add@feature{#2}{\PhK}{\values}
520         \add@feature{#1}{\MagK}{\values}
521       \fi
522     \fi
523   \fi
524   \ifnum\pdf@strcmp{\feature}{d}=0
525     \ifnum\pdf@strcmp{#3}{\linear}=0
526       \PackageError {bodeplot} {Linear approximation for pure de-
527 lays is not
528 supported.} {Plot the true Bode plot using 'true' in-
529 stead of 'linear' .}
530     \else
531       \ifnum\pdf@strcmp{#3}{\asymptotic}=0
532         \PackageError {bodeplot} {Asymptotic approxima-
533 tion for pure delays is not
534 supported.} {Plot the true Bode plot using 'true' in-
535 stead of 'asymptotic' .}
536       \else
537         \ifdim\values pt < 0pt
538           \PackageError {bodeplot} {Delay needs to be a positive num-
539 ber.}
540         \fi
541         \add@feature{#2}{\PhDel}{\values}
542         \add@feature{#1}{\MagDel}{\values}
543       \fi
544     \fi
545   \fi
546 }
547 }

```

`\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).

```

543 \newcommand{\build@TF@plot}[3]{
544   \gdef\num@real{0}
545   \gdef\num@im{0}
546   \gdef\den@real{0}
547   \gdef\den@im{0}
548   \gdef\loop@delay{0}
549   \foreach \feature/\values in {#3} {
550     \ifnum\pdf@strcmp{\feature}{num}=0
551       \foreach \numcoeff [count=\numpow] in \values {
552         \xdef\num@degree{\numpow}
553       }
554       \foreach \numcoeff [count=\numpow] in \values {
555         \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}
556         \ifnum\currentdegree = 0
557           \xdef\num@real{\num@real+\numcoeff}
558         \else
559           \ifodd\currentdegree
560             \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-

```

```

1}{(\currentdegree-1)/2})*%
561      (\n@pow{t}{\currentdegree}}))}
562      \else
563      \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-
1}{(\currentdegree)/2})*%
564      (\n@pow{t}{\currentdegree}}))}
565      \fi
566      \fi
567    }
568  \fi
569  \ifnum\pdf@strcmp{\feature}{den}=0
570  \foreach \dencoeff [count=\denpow] in \values {
571  \xdef\den@degree{\denpow}
572  }
573  \foreach \dencoeff [count=\denpow] in \values {
574  \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}
575  \ifnum\currentdegree = 0
576  \xdef\den@real{\den@real+\dencoeff}
577  \else
578  \ifodd\currentdegree
579  \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-
1}{(\currentdegree-1)/2})*%
580  (\n@pow{t}{\currentdegree}}))}
581  \else
582  \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-
1}{(\currentdegree)/2})*%
583  (\n@pow{t}{\currentdegree}}))}
584  \fi
585  \fi
586  }
587  \fi
588  \ifnum\pdf@strcmp{\feature}{d}=0
589  \xdef\loop@delay{\values}
590  \fi
591  }
592  \xdef#2{((atan2((\num@im),(\num@real))-atan2((\den@im),%
593  (\den@real))-\loop@delay*t)*(\ph@scale))}
594  \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2}))) -
%
595  20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2}))))}
596  }

```

`\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.

```

597 \newcommand{\parse@opt}[1]{
598   \gdef\optmag@axes{}
599   \gdef\optph@axes{}
600   \gdef\optph@plot{}
601   \gdef\optmag@plot{}
602   \gdef\opt@group{}
603   \gdef\opt@approx{}
604   \gdef\optph@commands{}
605   \gdef\optmag@commands{}
606   \gdef\opt@tikz{}
607   \foreach \obj/\typ/\opt in {#1} {
608     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
609     \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0

```

```

610     \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
611   \else
612     \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
613       \xdef\optph@plot{\unexpanded\expandafter{\opt}}
614     \else
615       \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
616       \xdef\optph@plot{\unexpanded\expandafter{\opt}}
617     \fi
618   \fi
619 \else
620   \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
621     \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
622       \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
623     \else
624       \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
625         \xdef\optph@axes{\unexpanded\expandafter{\opt}}
626       \else
627         \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
628         \xdef\optph@axes{\unexpanded\expandafter{\opt}}
629       \fi
630     \fi
631   \else
632     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{group}=0
633       \xdef\opt@group{\unexpanded\expandafter{\opt}}
634     \else
635       \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{approx}=0
636         \xdef\opt@approx{\unexpanded\expandafter{\opt}}
637       \else
638         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
639           \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
640             \xdef\optph@commands{\unexpanded\expandafter{\opt}}
641           \else
642             \xdef\optmag@commands{\unexpanded\expandafter{\opt}}
643           \fi
644         \else
645           \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
646             \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
647           \else
648             \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
649               \unexpanded\expandafter{\obj}}
650             \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
651               \unexpanded\expandafter{\obj}}
652           \fi
653         \fi
654       \fi
655     \fi
656   \fi
657 \fi
658 }
659 }

```

`\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.

```

660 \newcommand{\parse@env@opt}[1]{
661   \gdef\opt@axes{}
662   \gdef\opt@tikz{}
663   \foreach \obj/\opt in {#1} {
664     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
665       \xdef\opt@axes{\unexpanded\expandafter{\opt}}
666     \else

```

```

667 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
668 \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
669 \else
670 \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
671 \unexpanded\expandafter{\obj}}
672 \fi
673 \fi
674 }
675 }

```

4.5 Nyquist plots

4.5.1 User macros

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

```

676 \newcommand{\NyquistZPK}[4][]{
677 \parse@N@opt{#1}
678 \gdef\func@mag{}
679 \gdef\func@ph{}
680 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
681 \panded\expandafter{\opt@tikz}]}
682 \temp@cmd
683 \build@ZPK@plot{\func@mag}{\func@ph}{#2}
684 \edef\temp@cmd{\noexpand\begin{axis}[
685 bode@style,
686 domain=#3*\freq@scale:#4*\freq@scale,
687 height=5cm,
688 xlabel={$\Re$},
689 ylabel={$\Im$},
690 samples=500,
691 \unexpanded\expandafter{\opt@axes}
692 ]}
693 \temp@cmd
694 \addplot [only marks,mark=+,thick,red] (-1 , 0);
695 \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \unex-
696 \panded\expandafter{\opt@plot}]}
697 \if@pgfarg
698 \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
699 {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
700 \opt@commands
701 \else
702 \stepcounter{gnuplot@id}
703 \temp@cmd gnuplot [parametric, gnuplot@prefix] {
704 \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
705 \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
706 };
707 \opt@commands
708 \fi
709 \end{axis}
710 \end{tikzpicture}
711 }

```

`\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.
710 \newcommand{\NyquistTF}[4][[]]{
711   \parse@N@opt{#1}
712   \gdef\func@mag{}
713   \gdef\func@ph{}
714   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
715   \temp@cmd
716     \build@TF@plot{\func@mag}{\func@ph}{#2}
717     \edef\temp@cmd{\noexpand\begin{axis}[
718       bode@style,
719       domain=#3*\freq@scale:#4*\freq@scale,
720       height=5cm,
721       xlabel={$\Re$},
722       ylabel={$\Im$},
723       samples=500,
724       \unexpanded\expandafter{\opt@axes}
725     ]}
726     \temp@cmd
727       \addplot [only marks, mark=+, thick, red] (-1 , 0);
728       \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \unex-
panded\expandafter{\opt@plot}]}
729       \if@pgfarg
730         \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
731           {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
732         \opt@commands
733       \else
734         \stepcounter{gnuplot@id}
735         \temp@cmd gnuplot [parametric, gnuplot@prefix] {
736           \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
737           \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
738         };
739         \opt@commands
740       \fi
741     \end{axis}
742   \end{tikzpicture}
743 }

```

\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.

```

744 \newcommand{\addNyquistZPKPlot}[2][[]]{
745   \gdef\func@mag{}
746   \gdef\func@ph{}
747   \build@ZPK@plot{\func@mag}{\func@ph}{#2}
748   \if@pgfarg
749     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do-
able=t, #1]}
750     \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
751       {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
752     \else
753       \stepcounter{gnuplot@id}
754       \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do-
able=t, #1]}
755       \temp@cmd gnuplot [parametric, gnuplot@prefix] {
756         \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
757         \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
758       };
759       \fi
760 }

```

\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.

```

761 \newcommand{\addNyquistTFplot}[2][]{
762   \gdef\func@mag{}
763   \gdef\func@ph{}
764   \build@TF@plot{\func@mag}{\func@ph}{#2}
765   \if@pgfarg
766     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/doma
       able=t, #1]}
767       \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
768         {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
769   \else
770     \stepcounter{gnuplot@id}
771     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/doma
       able=t, #1]}
772       \temp@cmd gnuplot [parametric, gnuplot@prefix]{
773         \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
774         \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
775       };
776   \fi
777 }

```

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

```

778 \NewEnviron{NyquistPlot}[3][]{
779   \parse@env@opt{#1}
780   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
       panded\expandafter{\opt@tikz}]
781     \noexpand\begin{axis}[
782       bode@style,
783       height=5cm,
784       domain=#2:#3,
785       xlabel={\$Re\$},
786       ylabel={\$Im\$},
787       \unexpanded\expandafter{\opt@axes}
788     ]
789   }
790   \temp@cmd
791     \addplot [only marks,mark=+,thick,red] (-1 , 0);
792     \BODY
793   \end{axis}
794 \end{tikzpicture}
795 }

```

4.5.2 Internal commands

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

```

796 \newcommand{\parse@N@opt}[1]{
797   \gdef\opt@axes{}
798   \gdef\opt@plot{}
799   \gdef\opt@commands{}
800   \gdef\opt@tikz{}
801   \foreach \obj/\opt in {#1} {
802     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
803       \xdef\opt@axes{\unexpanded\expandafter{\opt}}
804     \else

```

```

805 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
806 \xdef\opt@plot{\unexpanded\expandafter{\opt}}
807 \else
808 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
809 \xdef\opt@commands{\unexpanded\expandafter{\opt}}
810 \else
811 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
812 \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
813 \else
814 \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
815 \unexpanded\expandafter{\obj}}
816 \fi
817 \fi
818 \fi
819 \fi
820 }
821 }

```

4.6 Nichols charts

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

```

NicholsChart 822 \newcommand{\NicholsZPK}[4][]{
\addNicholsZPKChart 823 \parse@N@opt{#1}
\addNicholsTFChart 824 \gdef\func@mag{
825 \gdef\func@ph{
826 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
827 \temp@cmd
828 \build@ZPK@plot{\func@mag}{\func@ph}{#2}
829 \edef\temp@cmd{\noexpand\begin{axis}[
830 ph@x@label,
831 bode@style,
832 domain=#3*\freq@scale:#4*\freq@scale,
833 height=5cm,
834 ylabel={Gain (dB)},
835 samples=500,
836 \unexpanded\expandafter{\opt@axes}
837 ]}
838 \temp@cmd
839 \edef\temp@cmd{\noexpand\addplot [variable=t,thick,\opt@plot]}
840 \if@pgfarg
841 \temp@cmd ( {\func@ph} , {\func@mag} );
842 \opt@commands
843 \else
844 \stepcounter{gnuplot@id}
845 \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
846 { set table $meta;
847 set logscale x 10;
848 set dummy t;
849 set samples \pgfkeysvalueof{/pgfplots/samples};
850 set trange [#3*\freq@scale:#4*\freq@scale];
851 plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
852 unset logscale x;
853 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
854 plot "$meta" using ($2*\ph@scale):($1);
855 };
856 \opt@commands
857 \fi
858 \end{axis}
859 \end{tikzpicture}
860 }
861 \newcommand{\NicholsTF}[4][]{

```

```

862 \parse@N@opt{#1}
863 \gdef\func@mag{}
864 \gdef\func@ph{}
865 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
866 \temp@cmd
867 \build@TF@plot{\func@mag}{\func@ph}{#2}
868 \edef\temp@cmd{\noexpand\begin{axis}[
869 ph@x@label,
870 bode@style,
871 domain=#3*\freq@scale:#4*\freq@scale,
872 height=5cm,
873 ylabel={Gain (dB)},
874 samples=500,
875 \unexpanded\expandafter{\opt@axes}
876 ]}
877 \temp@cmd
878 \edef\temp@cmd{\noexpand\addplot [variable=t,thick, \opt@plot]}
879 \if@pgfarg
880 \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
881 \opt@commands
882 \else
883 \stepcounter{gnuplot@id}
884 \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
885 { set table $meta1;
886 set logscale x 10;
887 set dummy t;
888 set samples \pgfkeysvalueof{/pgfplots/samples};
889 set trange [#3*\freq@scale:#4*\freq@scale];
890 plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
891 unset logscale x;
892 set table $meta2;
893 plot "$meta1" using ($1):($2) smooth unwrap;
894 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
895 plot "$meta2" using ($2*\ph@scale):($1);
896 };
897 \opt@commands
898 \fi
899 \end{axis}
900 \end{tikzpicture}
901 }
902 \NewEnviron{NicholsChart}[3][]{
903 \parse@env@opt{#1}
904 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
905 \noexpand\begin{axis}[
906 ph@x@label,
907 bode@style,
908 domain=#2:#3,
909 height=5cm,
910 ylabel={Gain (dB)},
911 \unexpanded\expandafter{\opt@axes}
912 ]
913 }
914 \temp@cmd
915 \BODY
916 \end{axis}
917 \end{tikzpicture}
918 }
919 \newcommand{\addNicholsZPKChart}[2][]{
920 \gdef\func@mag{}
921 \gdef\func@ph{}
922 \build@ZPK@plot{\func@mag}{\func@ph}{#2}

```

```

923 \ifpgfarg
924   \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do
able=t, #1]}
925   \temp@cmd ( {\func@ph} , {\func@mag} );
926 \else
927   \stepcounter{gnuplot@id}
928   \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
929   { set table $meta;
930     set logscale x 10;
931     set dummy t;
932     set samples \pgfkeysvalueof{/pgfplots/samples};
933     set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
934     plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
935     unset logscale x;
936     set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
937     pplot "$meta" using ($2*\ph@scale):($1);
938   };
939 \fi
940 }
941 \newcommand{\addNicholsTFChart}[2][]{
942   \gdef\func@mag{}
943   \gdef\func@ph{}
944   \build@TF@plot{\func@mag}{\func@ph}{#2}
945   \ifpgfarg
946     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do
able=t, #1]}
947     \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
948   \else
949     \stepcounter{gnuplot@id}
950     \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
951     { set table $meta1;
952       set logscale x 10;
953       set dummy t;
954       set samples \pgfkeysvalueof{/pgfplots/samples};
955       set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
956       plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
957       unset logscale x;
958       set table $meta2;
959       plot "$meta1" using ($1):($2) smooth unwrap;
960       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
961       plot "$meta2" using ($2*\ph@scale):($1);
962     };
963   \fi
964 }

```

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	A	
<code>\@declutterargfalse</code> 5	<code>\add@feature</code>	<code>\addBodeZPKPlots</code> 317
<code>\@declutterargtrue</code> 7	462 , 481 , 482 , 485 ,	<code>\addNicholsTFChart</code> 822
<code>\@hzargfalse</code> 13	486 , 488 , 489 , 497 ,	<code>\addNicholsZPKChart</code> 822
<code>\@hzargtrue</code> 15	498 , 501 , 502 , 504 ,	<code>\addNyquistTFPlot</code> 761
<code>\@nil</code> 107 , 108	505 , 512 , 513 , 516 ,	<code>\addNyquistZPKPlot</code> 744
<code>\@pgfargfalse</code> 1	517 , 519 , 520 , 536 , 537	
<code>\@pgfargtrue</code> 3	<code>\addBodeComponentPlot</code>	B
<code>\@radargfalse</code> 9 388	<code>\bode@style</code> 45
<code>\@radargtrue</code> 11	<code>\addBodeTFPlot</code> 345	<code>BodeMagPlot (env.)</code> 425
		<code>BodePhPlot (env.)</code> 406

<code>\optmag@commands</code>	229, 244, 285, 300, 605, 642	182, 286, 308, 310, 370, 372, 593, 696, 697, 702, 703, 730, 731, 736, 737, 750, 751, 756, 757, 767, 768, 773, 774, 851, 854, 880, 890, 895, 934, 937, 947, 956, 961	393, 700, 734, 753, 770, 844, 883, 927, 949
<code>\optmag@plot</code>	... 224, 280, 601, 610, 615, 648		
<code>\optph@axes</code>	... 225, 281, 599, 625, 628		
<code>\optph@commands</code>	231, 256, 287, 312, 604, 640		
<code>\optph@plot</code>	... 226, 282, 600, 613, 616, 650		
P			
<code>\p</code>	... 495, 497, 498, 501, 502, 504, 505		
<code>\parse@env@opt</code>	407, 426, 445, 660, 779, 903		
<code>\parse@N@opt</code>	... 677, 711, 796, 823, 862		
<code>\parse@opt</code>	206, 262, 597		
<code>\pdf@strcmp</code>	... 321, 348, 354, 363, 478, 480, 484, 494, 496, 500, 510, 511, 515, 524, 525, 529, 550, 569, 588, 608, 609, 612, 620, 621, 624, 632, 635, 638, 639, 645, 664, 667, 802, 805, 808, 811		
<code>\pgfkeysvalueof</code>	... 239, 251, 295, 307, 327, 336, 337, 355, 358, 368, 369, 379, 380, 390, 398, 399, 749, 754, 766, 771, 849, 888, 924, 932, 933, 946, 954, 955		
<code>\pgfplotsset</code>	... 21, 45, 60, 62, 63, 64, 67, 68, 73, 75, 84, 85, 89, 90, 97, 99, 104		
<code>\ph@scale</code>	... 60, 65, 69, 83, 88, 112, 116, 127, 140, 143, 155, 161, 162,		
<code>\ph@x@label</code>	... 60		
<code>\ph@y@label</code>	... 60		
<code>\PhCSPoles</code>	... 150		
<code>\PhCSPolesAsymp</code>	150, 187		
<code>\PhCSPolesLin</code>	.. 150, 184		
<code>\PhCSZeros</code>	... 150		
<code>\PhCSZerosAsymp</code>	... 150		
<code>\PhCSZerosLin</code>	... 150		
<code>\PhDel</code>	... 116, 536		
<code>\PhK</code>	... 109, 519		
<code>\PhKAsymp</code>	.. 109, 115, 516		
<code>\PhKLin</code>	... 109, 115, 512		
<code>\PhPole</code>	... 117, 147, 504		
<code>\PhPoleAsymp</code>	117, 149, 501		
<code>\PhPoleLin</code>	.. 117, 148, 497		
<code>\PhSOPoles</code>	... 177		
<code>\PhSOPolesAsymp</code>	... 177		
<code>\PhSOPolesLin</code>	... 177		
<code>\PhS0Zeros</code>	... 177		
<code>\PhS0ZerosAsymp</code>	... 177		
<code>\PhS0ZerosLin</code>	... 177		
<code>\PhZero</code>	... 144, 488		
<code>\PhZeroAsymp</code>	... 144, 485		
<code>\PhZeroLin</code>	... 144, 481		
<code>\plot@macro</code>	... 319, 322, 324, 328, 338, 346, 349, 351, 356, 359, 370, 381		
R			
<code>\renewcommand</code>	... 83, 88, 96, 103		
S			
<code>\setcounter</code>	... 27		
<code>\stepcounter</code>	233, 245, 289, 301, 330, 362,		
T			
<code>\temp@cmd</code>	209, 210, 212, 222, 265, 266, 268, 278, 327, 328, 331, 332, 355, 356, 358, 359, 390, 391, 408, 420, 427, 439, 446, 457, 680, 681, 683, 692, 694, 696, 701, 714, 715, 717, 726, 728, 730, 735, 749, 750, 754, 755, 766, 767, 771, 772, 780, 790, 826, 827, 829, 838, 839, 841, 845, 865, 866, 868, 877, 878, 880, 884, 904, 914, 924, 925, 946, 947		
<code>\temp@macro</code>	... 320, 322, 324, 347, 349, 351		
<code>\temp@mag@cmd</code>	.. 223, 228, 234, 279, 284, 290		
<code>\temp@ph@cmd</code>	... 225, 230, 246, 281, 286, 302		
V			
<code>\values</code>	... 477, 479, 495, 512, 513, 516, 517, 519, 520, 533, 536, 537, 549, 551, 554, 570, 573, 589		
W			
<code>\write</code>	... 41		
Y			
<code>\y</code>	... 468, 469		
Z			
<code>\z</code>	... 479, 481, 482, 485, 486, 488, 489		

Change History

v1.0		<code>\BodeTF</code> : Added Tikz option	... 24
General: Initial release	... 1	<code>\BodeZPK</code> : Added Tikz option	... 23
v1.0.1		<code>NicholsChart</code> : Added tikz option to environments	... 35
<code>\addBodeZPKPlots</code> : Improved optional argument handling.	... 25	<code>\NicholsTF</code> : Added commands and tikz options	... 35
<code>\BodeZPK</code> : Pass arbitrary TikZ commands as options.	... 22	<code>\NicholsZPK</code> : Added commands and tikz options	... 35
v1.0.2		<code>gnuplot@prefix</code> : Fixed issue #1	.. 18
<code>BodePlot</code> : Added tikz option to environments	... 27	<code>gnuplot@prefix</code> : Added jobname to gnuplot prefix	... 18
v1.0.3		<code>\NyquistTF</code> : Added commands and tikz options	... 33

\NyquistZPK : Added commands and tikz options	32	BodeMagPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27
\parse@env@opt : Added tikz option to environments	31	BodePhPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	26
\parse@N@opt : Added commands and tikz options	34	BodePlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27
\parse@opt : Added Tikz option	30	\BodeTF : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	24
NyquistPlot : Added tikz option to environments	34	\BodeZPK : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	23
v1.0.4		\build@TF@plot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	29
General: Fixed unintended optional argument macro expansion	1	get@interval@end : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	20
v1.0.5		ph@y@label : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	19
\parse@opt : Fixed a bug	30	\NyquistTF : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	33
v1.0.6		\NyquistZPK : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	32
General: Fixed issue #3	1	v1.1.2	
v1.0.7		BodeMagPlot : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	27
General: Removed unnecessary semicolons	1	BodePhPlot : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	26
Updated documentation	1	BodePlot : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	27
v1.0.8		NicholsChart : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	35
General: Added a new class option ‘declutter’	1	\PhS0ZerosLin : Fix scaling bug introduced in v1.1.1	22
\build@TF@plot : Included phase due to delay in wrapping.	29	NyquistPlot : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	34
gnuplot@prefix : Fixed issue #6	18	v1.1.3	
v1.1.0		\addBodeComponentPlot : Changed implementation to respect user-supplied domain	26
General: Fixed phase wrapping in gnuplot mode	1	\addBodeTFPlot : Changed implementation to respect user-supplied domain	25
\addBodeTFPlot : Fixed phase wrapping in gnuplot mode	25	\addBodeZPKPlots : Changed implementation to respect user-supplied domain	25
BodeMagPlot : Added separate environments for phase and magnitude plots	27		
BodePhPlot : Added separate environments for phase and magnitude plots	26		
BodePlot : Deprecated BodePlot environment	27		
\BodeTF : Fixed phase wrapping in gnuplot mode	24		
v1.1.1			
General: Enable Hz and rad units	1		
\addBodeComponentPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	26		
\addBodeTFPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	25		
\addBodeZPKPlots : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	25		
\addNicholsTFChart : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	35		
\addNyquistTFPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	34		
\addNyquistZPKPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	33		

\addNicholsTFChart: Changed implementation to respect user-supplied domain	35	\addNyquistTFPlot: Changed implementation to respect user-supplied domain	34
\addNicholsZPKChart: Changed implementation to respect user-supplied domain	35	\addNyquistZPKPlot: Changed implementation to respect user-supplied domain	33