

The bodeplot package

version 2.1

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

September 30, 2025

Contents

1	Introduction	2
1.1	External Dependencies	2
1.2	Directory Structure	2
1.3	Limitations	2
2	TL;DR	3
3	Usage	10
3.1	Bode plots	10
3.1.1	Basic components up to first order	14
3.1.2	Basic components of the second order	15
3.2	Nyquist plots	16
3.3	Nichols charts	18
3.4	Pole-zero maps	20
4	Implementation	21
4.1	Initialization	21
4.2	Parametric function generators for poles, zeros, gains, and delays.	23
4.3	Second order systems.	24
4.4	Commands for Bode plots	26
4.4.1	User macros	26
4.4.2	Internal macros	32
4.5	Nyquist plots	37
4.5.1	User macros	37
4.5.2	Internal commands	40
4.6	Nichols charts	47
	Index	56
	Change History	59

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 $K \in \Re$ 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

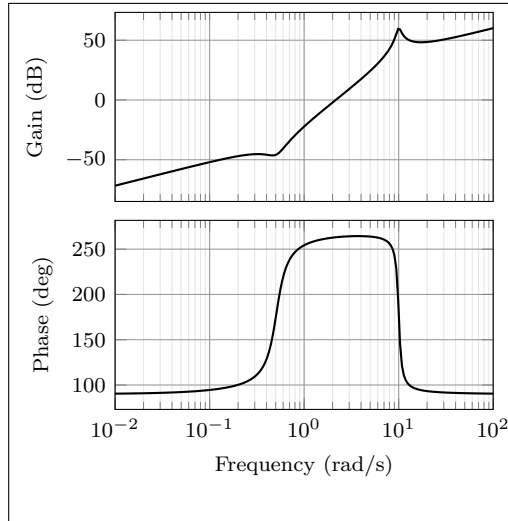
- Before version 1.2, in `pgf` mode, the package set `trig format plots` to `rad` globally. Version 1.2 onwards, this option is passed to each `addplot` command individually so that it does not affect other plots in the document. To roll back to the pre-1.2 behavior, load the package with `\usepackage[pgf]{bodeplot}[=2024-02-06]`.
- In `pgf` mode, Bode phase plots and Nichols charts in TF form wrap angles so that they are always between -180 and 180° or $-\pi$ and π 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! Since v1.1.4, you can redefine the `n@mod` macro using the commands `\makeatletter\renewcommand{\n@mod}{\n@mod@p}\makeatother` to wrap the phase between 0 and 360° or 0 and 2π radian. The commands `\makeatletter\renewcommand{\n@mod}{\n@mod@n}\makeatother` will wrap the phase between -360 and 0° or -2π and 0 radian.
- 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)

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

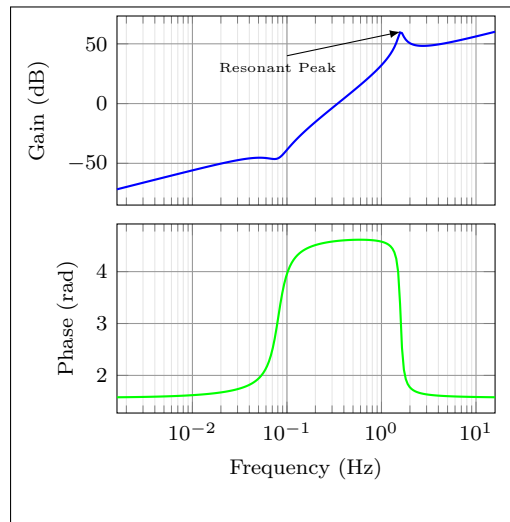
Bode plot in ZPK format



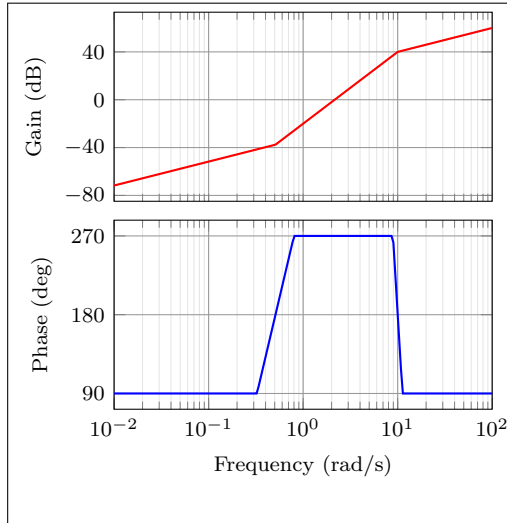
```
\BodeZPK{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

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)}
```



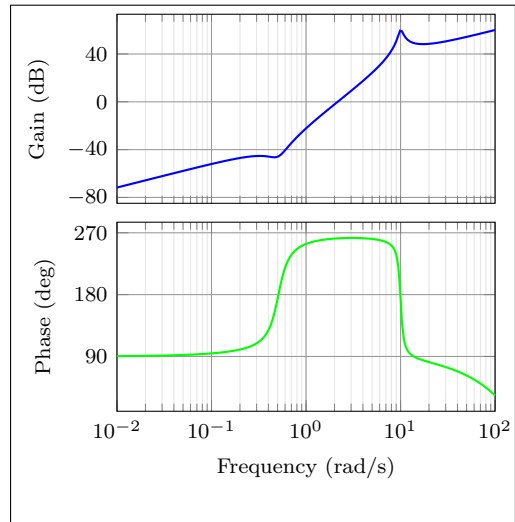
Linear approximation with customization



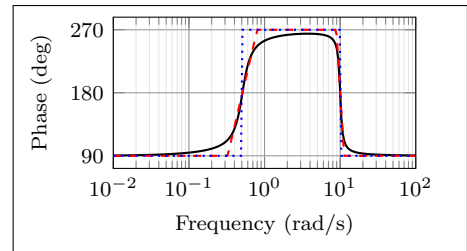
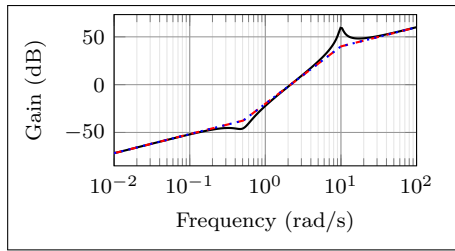
```
\BodeZPK[%
plot/mag/{red,thick},
plot/ph/{blue,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90},
approx/linear%
]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

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

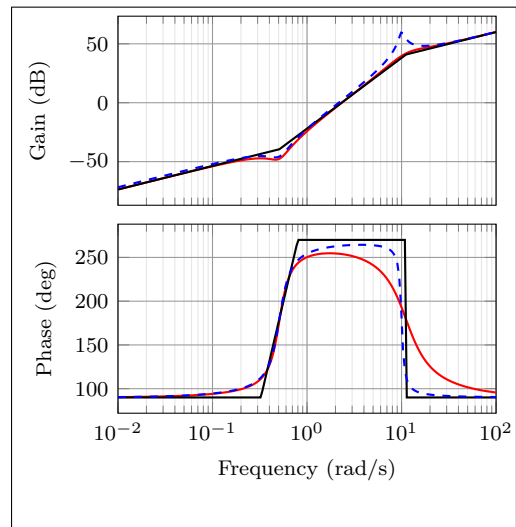


```
\begin{BodeMagPlot}[%
  axes/{height=2cm,
  width=4cm}%
]
{0.01}
{100}
\addBodeZPKPlots[%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
]
{magnitude}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\end{BodeMagPlot}
```

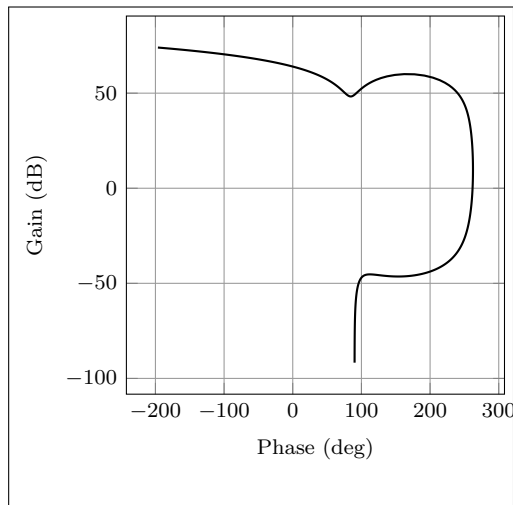
```
\begin{BodePhPlot}[%
  height=2cm,
  width=4cm,
  ytick distance=90
]
{0.01}
{100}
\addBodeZPKPlots[%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
]
{phase}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\end{BodePhPlot}
```

Multiple transfer functions in a single Bode plot using the **BodePlot** environment and the **\addBodePlot** macro introduced in v2.1.

```
\begin{BodePlot}{0.01}{100}
\addBodePlot[red,thick]{zpk}{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-5,-10},{-5,10}},
  k/10%
}
\addBodePlot[black,thick,linear]{zpk}{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-5,-10},{-5,10}},
  k/10%
}
\addBodePlot[blue,dashed]{tf}{%
  num/{10,2,2.6,0},
  den/{1,1,100.25}%
}
\end{BodePlot}
```



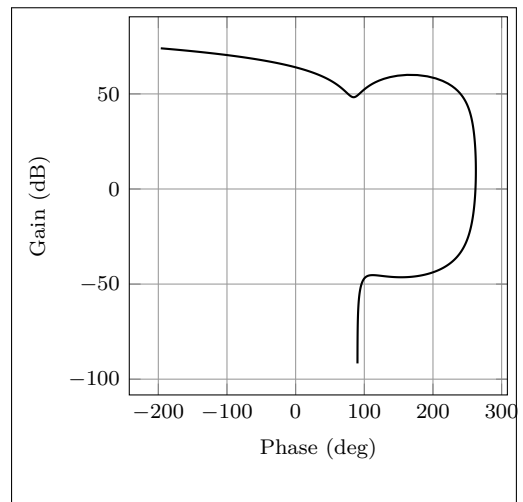
Nichols chart



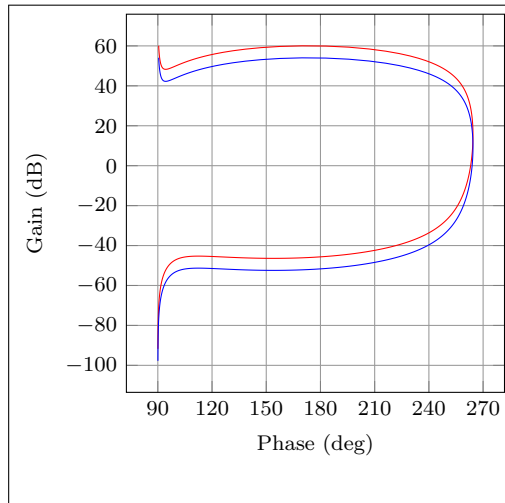
```
\NicholsZPK[samples=1000]
{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10,
d/0.01%
}
{0.001}
{500}
```

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}
```



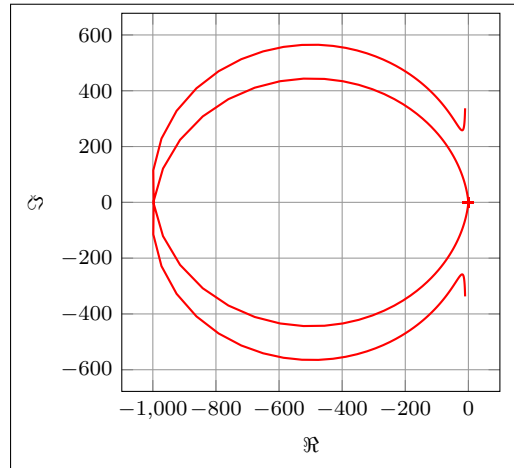
Multiple Nichols charts with customization



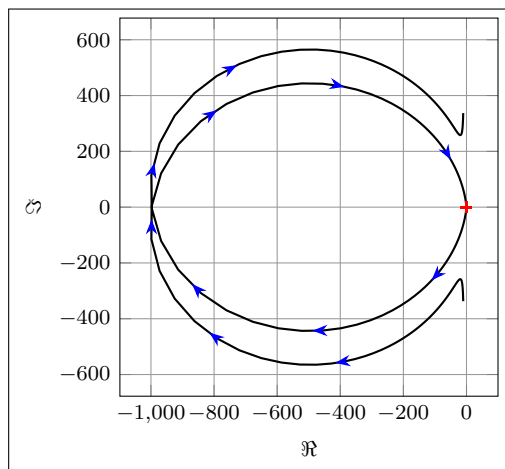
```
\begin{NicholsChart}[%
ytick distance=20,
xtick distance=30
]
{0.001}
{100}
\addNicholsZPKChart [red,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
\addNicholsZPKChart [blue,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/5%
}
\end{NicholsChart}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

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}
```



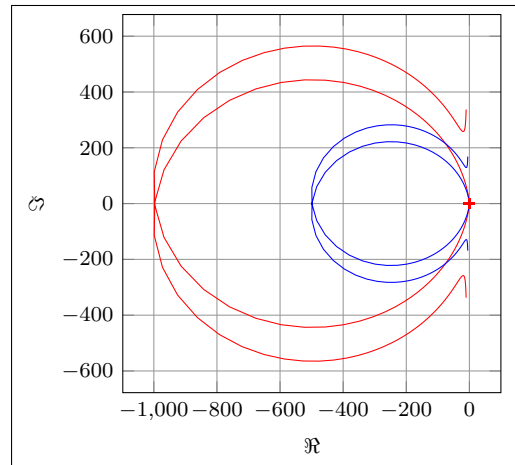
Nyquist plot in TF format with arrows



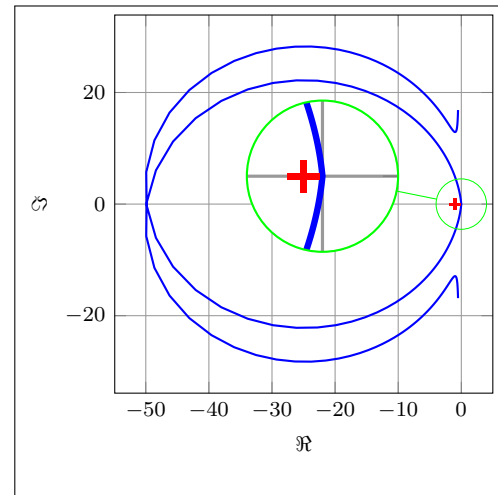
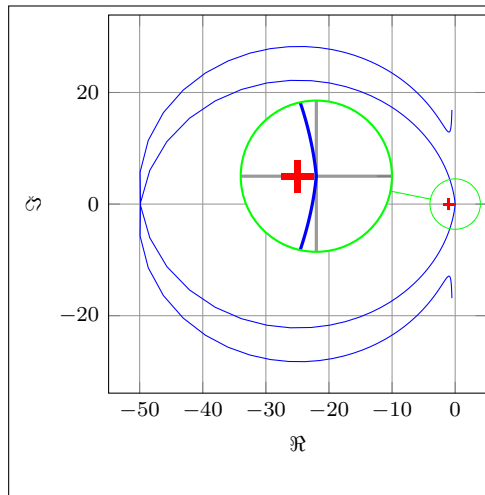
```
\NyquistTF[%
plot/{%
samples=1000,
postaction=decorate,
decoration={%
markings,
mark=between positions 0.1 and 0.9 step 5em with {%
\arrow{Stealth [length=2mm, blue]}
}
}%
}
]
{%
num/{10,2,2.6,0},
den/{1,1,100.25}%
}
{-30}
{30}
```

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}
```



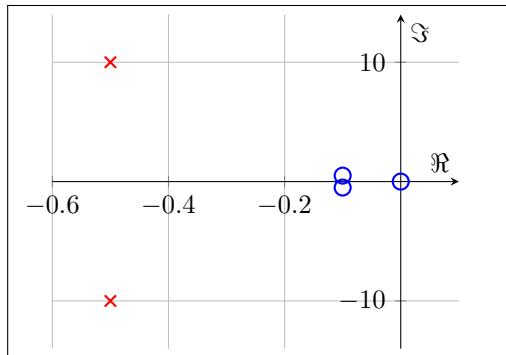
Nyquist plots with additional commands, using two different macros



```
\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);
}%
}
{-30}
{30}
```

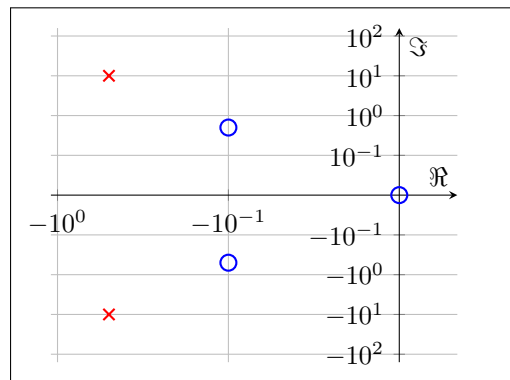

Pole-zero map



```
\PoleZeroMapZPK
{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
```

Pole-zero map (symmetric log scale)

```
\PoleZeroMapZPK[scale/{log}]
{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
```



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.

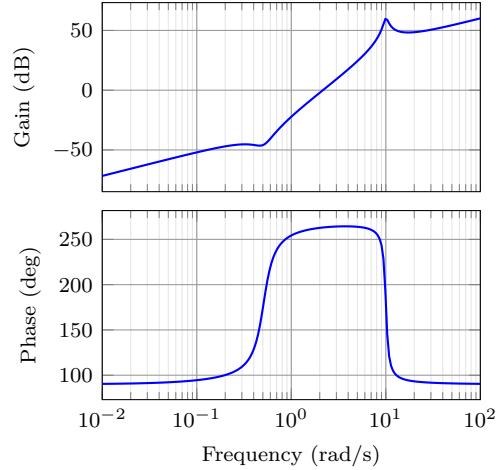


Figure 1: Output of the `\BodeZPK` macro.

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

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

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

which generates the plot in Figure 2.

```
\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

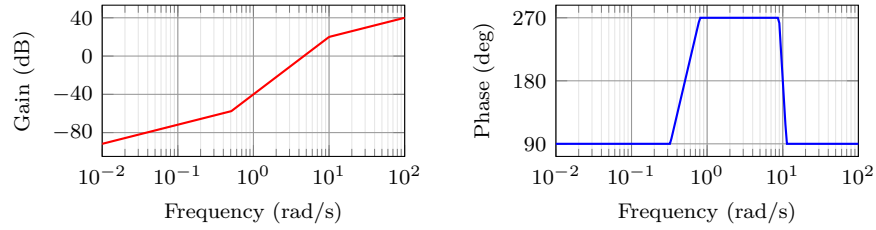


Figure 2: Customization of the default `\BodeZPK` macro.

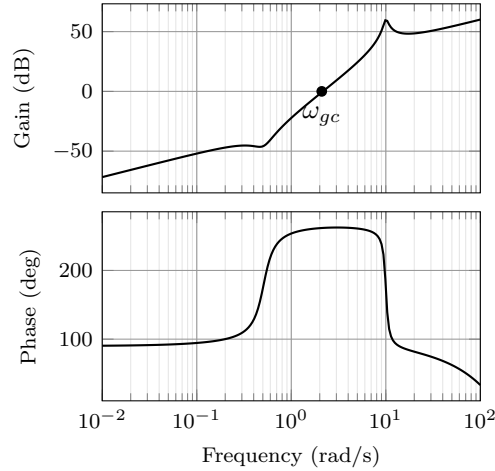


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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

For example, given the same transfer function as (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} [{obj1/{opt1}},obj2/{opt2},...]
  {<min-frequency>}{<max-frequency>}
  \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.
- 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,...]
                  {\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,...]
                  {\langle plot-type\rangle}
                  {\langle z/\langle zeros\rangle\rangle,\langle p/\langle poles\rangle\rangle,\langle k/\langle gain\rangle\rangle,\langle d/\langle delay\rangle\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,\langle d/\langle delay\rangle\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

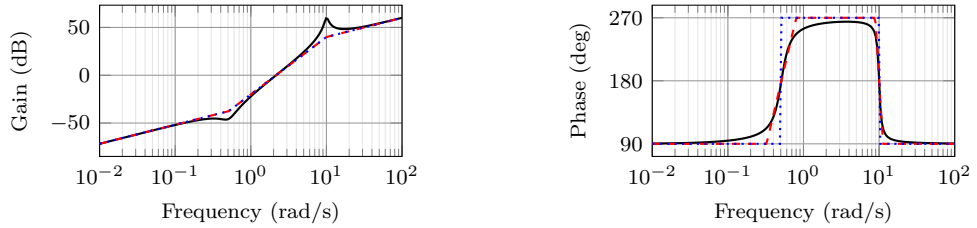


Figure 4: Superimposed approximate and true Bode plots using the `BodeMagPlot` and `BodePhPlot` environments and the `\addBodeZPKPlots` macro.

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, **with angles passed to trigonometric functions in radian**). The intended use of this macro is to plot the parametric functions generated using the basic component macros described in Section 3.1.1 below.

`\addBodePlot` `\addBodePlot` [*plot-options*] {*system-type*} {*system-data*}

Unified macro to add Bode plots for both ZPK and TF system representations. Unlike the `\addBodeZPKPlots` and `\addBodeTFPlots` macros, this macro can only be used inside a `BodePlot` environment. **It will not function if used in a generic semilogxaxis environment or in the `BodeMagPlot` and `BodePhPlot` environments.** The second mandatory argument `system-type` should be either `zpk` or `tf` to specify the system representation. For ZPK systems, the `system-data` has the same format as `\addBodeZPKPlots`. For TF systems, it has the same format as `\addBodeTFPlot`. The optional arguments are the same as the `\addBodeTfPlot` macro, with two additions. Passing `linear` or `asymptotic` options will generate linear or asymptotic approximations if the system representation is ZPK. These two options are ignored for TF systems.

`BodePlot (env.)` `\begin{BodePlot}` [*obj1*/*opt1*], *obj2*/*opt2*], ...]
`{` *min-frequency* `}` {*max-frequency*}
`\addBodePlot` ...
`\end{BodePlot}`

Version 2.1 revives the the deprecated `BodePlot` environment exclusively to host multiple instances of the new unified `\addBodePlot` macro. The `\addBodeZPKPlots` and `\addBodeTFPlots` macros **will not function inside this environment**. The purpose is to plot a Bode plot that contains several different transfer functions. The optional argument is the same as the original `\BodeZPK` macro, supporting `plot/`, `axes/`, `group/`, `tikz/` options with `mag` and `ph` sub-types.

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

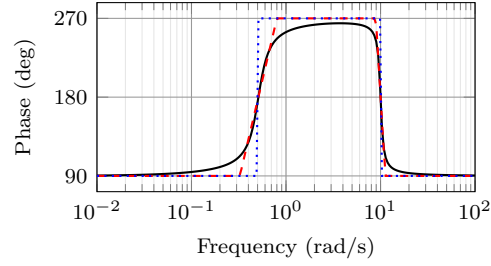


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePhPlot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

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} + \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>}`

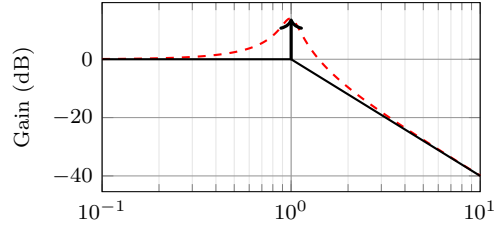


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

This entry describes 2 different macros of the form `\MagSOFeaturePeak` that take 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[<draw-options>]{<zeta>}{<omega-n>}`

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[<draw-options>]{<real-part>}{<imaginary-part>}`

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

Plots the Nyquist plot of a transfer function given in ZPK format with a thick red + marking the critical point (-1,0). The mandatory arguments are the same as `\BodeZPK`. Since there is only one plot in a Nyquist diagram, the `\typ` specifier in the optional argument tuples is not needed. As such, the supported optional argument tuples are `plot/{opt}`, which passes `{opt}` to `\addplot`, `axes/{opt}`, which passes `{\opt}` to the `axis` environment, and `tikz/{opt}`, which passes `{\opt}` to the `tikzpicture` environment. Asymptotic/linear approximations are

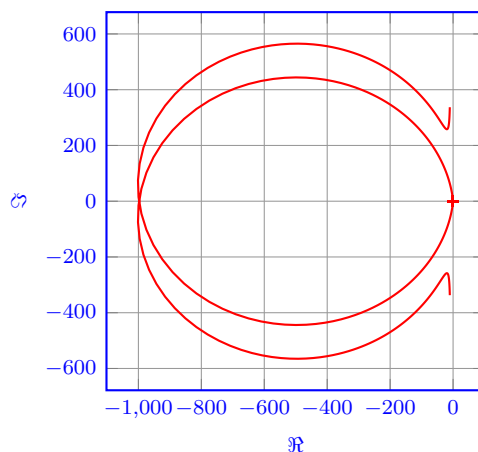


Figure 7: Output of the `\NyquistZPK` macro.

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

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

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

generates the Nyquist plot in Figure 8.

```
NyquistPlot (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}`:

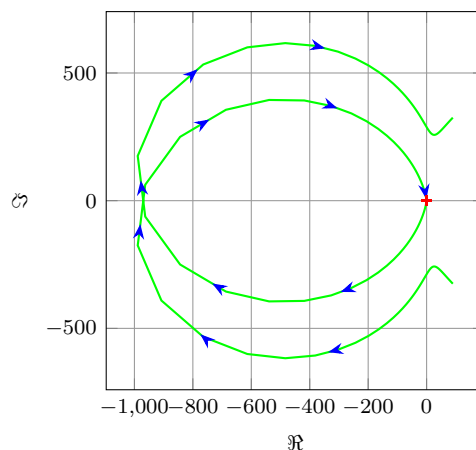


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

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

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

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

Generates a two parametric functions for the magnitude and the phase 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}]
```

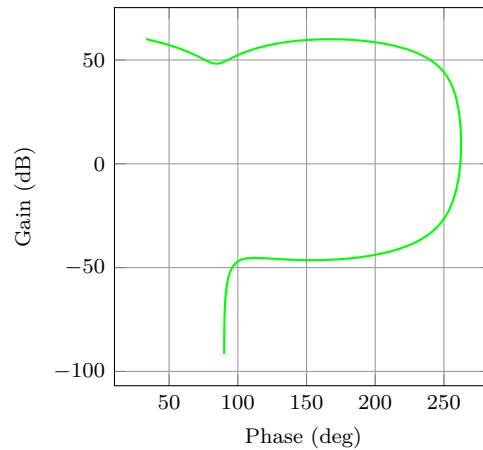


Figure 9: Output of the `\NyquistZPK` macro.

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

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

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

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

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

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

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

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

3.4 Pole-zero maps

```
\PoleZeroMapZPK \PoleZeroMapZPK [plot/opt],axes/opt],scale/log}]
  {z/zeros},p/poles},k/gain}]}
```

Plots the pole-zero map of a transfer function given in ZPK format, similar to MATLAB's `pzmap` function. The poles are marked with red 'x' symbols and the zeros are marked with blue 'o' symbols. The mandatory argument contains the same ZPK specification as `\BodeZPK`, but the delay parameter `d` is ignored since delays do not affect pole-zero locations. The optional argument supports the same tuples as `\NyquistZPK`: `plot/opt` passes options to `\addplot`, `axes/opt` passes options to the `axis` environment, `scale/opt` sets the scaling type, and `tikz/opt` passes options to the `tikzpicture` environment. If just `{opt}` is provided, it is interpreted as `axes/opt`.

By default, the axes use linear scaling with unequal scaling to better distribute poles and zeros across the available plot area. This provides improved readability when poles and zeros have different ranges in real and imaginary parts. To force equal axes (square aspect ratio), use `axes/{axis equal}`.

The `scale/opt` option enables symmetric logarithmic (symlog) scaling for both axes. This is particularly useful for systems with poles and zeros spanning multiple decades. The symlog scaling preserves the sign of coordinates while applying logarithmic scaling to their magnitude, allowing visualization of both positive and negative values on the same plot.

For example, the command

```
\PoleZeroMapZPK[axes/{grid=major,xlabel={Real Part},ylabel={Imaginary Part}}]
  {z/0,{-0.1,-0.5},{-0.1,0.5}},p/{-0.5,-10},{-0.5,10}},k/10}
```

generates a standard linear pole-zero map with the zeros at the origin and at $-0.1 \pm 0.5j$, and poles at $-0.5 \pm 10j$.

```
\PoleZeroMapZPK[scale/log]{z/{-1000,-10,-0.1}},p/{-100000,-1000,-100,-1}},k/1}
```

```
\PoleZeroMapZPK[scale/log]{z/{-1,2}},{-1,-2}},p/{-10,5},{-10,-5}},-  
0.01},k/1}
```

The first example creates a symlog pole-zero map with real poles and zeros; the threshold will be automatically set to the smallest real part. The second example includes complex poles and zeros; the threshold will be set to the smallest real part.

4 Implementation

4.1 Initialization

```
\n@mod We start by processing the class options.
\n@mod@p 1 \newif\if@pgfarg\@pgfargfalse
\n@mod@n 2 \DeclareOption{pgf}{
\n@pow 3 \@pgfargtrue
gnuplot@id 4 }
gnuplot@prefix 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 new macros to unify `pgfplots` and `gnuplot`. New macros are defined for the `pow` and `mod` functions to address differences between the two math engines.

```
18 \newcommand{\n@mod}[2]{(#1)-((round((#1)/(#2)))*(#2))}
19 \newcommand{\n@mod@p}[2]{(#1)-((floor((#1)/(#2)))*(#2))}
20 \newcommand{\n@mod@n}[2]{(#1)-((floor((#1)/(#2))+1)*(#2))}
21 \if@pgfarg
22 \newcommand{\n@pow}[2]{(#1)^(#2)}
23 \else
24 \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.

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

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.

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

`\if@babel` Check if the `babel` package is loaded and generate a list of shorthands if it is. The code `\shorthand@list` is based on [this stackexchange answer](#).

```
44 \newif\if@babel\@babelfalse
45 \AtBeginDocument{%
```

```

46 \@ifpackageloaded{babel}{%
47   \@babeltrue
48   \let\shorthand@list\@empty
49   \def\do#1{%
50     \begingroup
51     \lccode'\~='#1\relax
52     \lowercase{\ifbabelshorthand~{\g@addto@macro\shorthand@list{~}}{}}
53     \endgroup
54   }
55   \dospecials
56 }{}
57 }

```

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

```

58 \pgfplotsset{
59   bode@style/.style = {
60     label style={font=\footnotesize},
61     tick label style={font=\footnotesize},
62     grid=both,
63     major grid style={color=gray!80},
64     minor grid style={color=gray!20},
65     x label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
66     y label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
67     scale only axis,
68     samples=200,
69     width=5cm,
70     log basis x=10
71   }
72 }

```

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, respectively.

```

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

```

```

102   \pgfplotsset{ph@y@label/.style = {ylabel={Phase (rad)}}}
103   \pgfplotsset{ph@x@label/.style = {xlabel={Phase (rad)}}}
104 },
105 frequency unit/.initial={rad},
106 frequency unit/.default={rad},
107 frequency unit/.is choice,
108 frequency unit/Hz/.code={
109   \renewcommand{\freq@scale}{2*pi}
110   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}
111   \ifpgfarg
112     \pgfplotsset{freq@filter/.style = {x filter/.expression={x-
log10(2*pi)}}}
113   \fi
114 },
115 frequency unit/rad/.code={
116   \renewcommand{\freq@scale}{1}
117   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (rad/s)}}}
118 }
119 }

```

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

```

get@interval@end 120 \def\get@interval@start#1:#2\@nil{#1}
                  121 \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 122 \newcommand*\MagK[2]{(20*log10(abs(#1)))}
\PhKLin   123 \newcommand*\MagKAsymp{\MagK}
           124 \newcommand*\MagKLin{\MagK}
           125 \newcommand*\PhK[2]{((#1<0?-pi:0)*\ph@scale)}
           126 \newcommand*\PhKAsymp{\PhK}
           127 \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.

```

128 \newcommand*\MagDel[2]{0}
129 \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 130 \newcommand*\MagPole[2]
\PhPole     131 {(-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 132 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
133   -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :
134   -20*log10(t)
135 )}

```

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

```

136 \newcommand*\MagPoleAsymp{\MagPoleLin}

```

Parametric function for the true phase of a complex pole.

```

137 \newcommand*\PhPole}[2]{((#1 > 0 ? (#2 > 0 ?
138 (\n@mod@p{-atan2((t - (#2)), -(#1))}{2*pi}) :
139 (-atan2((t - (#2)), -(#1)))) :
140 (-atan2((t - (#2)), -(#1))))*\ph@scale)}

```

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

```

141 \newcommand*\PhPoleLin}[2]{
142 ((abs(#1)+abs(#2) == 0 ? -pi/2 :
143 (t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
144 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))))) ?
145 (-atan2(- (#2), -(#1))) :
146 (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
147 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))))) ?
148 (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) :
149 (-atan2(- (#2), -(#1)) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
150 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} +
151 \n@pow{#2}{2})))))))*((#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) + atan2(-
152 (#2), -(#1)))/
153 (log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
154 (\n@pow{#1}{2} + \n@pow{#2}{2})))))))*\ph@scale)}

```

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

```

154 \newcommand*\PhPoleAsymp}[2]{((t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}))) ?
155 (-atan2(- (#2), -(#1))) :
156 (#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 \newcommand*\MagZero{0-\MagPole}
\MagZeroLin   \newcommand*\MagZeroLin{0-\MagPoleLin}
\PhZero      \newcommand*\MagZeroAsymp{0-\MagPoleAsymp}
\PhZeroAsymp \newcommand*\PhZero{0-\PhPole}
\PhZeroLin   \newcommand*\PhZeroLin{0-\PhPoleLin}
\PhZeroAsymp \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 $G(s) = \frac{1}{s^2 + 2\zeta w_n s + w_n^2}$. We start with true, linear, and asymptotic magnitude plots for this transfer function.

```

\MagCSPolesAsymp \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\MagCSPolesLin   \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
\PhCSPoles      \newcommand*\PhCSPoles}[2]{(t < #2 ? -40*log10(#2) : -
\PhCSPolesAsymp \newcommand*\PhCSPolesAsymp}[2]{(t < #2 ? -40*log10(#2) : -
\PhCSPolesLin   \newcommand*\PhCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
\MagCSZeros     \newcommand*\MagCSZeros{0-\MagCSPolesLin}

```

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

```

\MagCSZerosLin   \newcommand*\MagCSZerosLin{0-\MagCSPolesLin}
\PhCSZeros      \newcommand*\PhCSZeros}[2]{((-atan2((2*(#1)*(#2)*t), (\n@pow{#2}{2}
\PhCSZerosAsymp \newcommand*\PhCSZerosAsymp}[2]{(t < #2 / (\n@pow{10}{abs(#1)})) ?
\PhCSZerosLin   \newcommand*\PhCSZerosLin}[2]{(t < #2 / (\n@pow{10}{abs(#1)})) ?
170 0 :
171 (t >= (#2 * (\n@pow{10}{abs(#1)})) ?
172 (#1>0 ? -pi : pi) :
173 (#1>0 ? (-pi*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
174 (pi*(log10(t*(\n@pow{10}{abs(#1)})/#2))/(2*abs(#1))))*\ph@scale)}
175 \newcommand*\PhCSPolesAsymp}[2]{((#1>0?(t<#2?0:-
176 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 θ - is necessary due to a bug in `gnuplot` (fixed in version 5.4, patchlevel 3).

```

176 \newcommand*\MagCSZeros{\theta-\MagCSPoles}
177 \newcommand*\MagCSZerosLin{\theta-\MagCSPolesLin}
178 \newcommand*\MagCSZerosAsymp{\theta-\MagCSPolesAsymp}
179 \newcommand*\PhCSZeros{\theta-\PhCSPoles}
180 \newcommand*\PhCSZerosLin{\theta-\PhCSPolesLin}
181 \newcommand*\PhCSZerosAsymp{\theta-\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.

```

182 \newcommand*\MagCSPolesPeak}[3][]{
183   \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
184   (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
185 }
186 \newcommand*\MagCSZerosPeak}[3][]{
187   \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
188   (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
189 }

```

`\MagS0Poles` `\MagS0PolesAsymp` 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.

```

\MagS0PolesLin 190 \newcommand*\MagS0Poles}[2]{
\PhS0Poles     191   (-20*log10(sqrt(\n@pow{#2} - \n@pow{t}{2}){2} + \n@pow{#1*t}{2})))}
\PhS0PolesAsymp 192 \newcommand*\MagS0PolesLin}[2]{
\PhS0PolesLin  193   (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t))}
\MagS0Zeros    194 \newcommand*\MagS0PolesAsymp{\MagS0PolesLin}

```

`\MagS0ZerosAsymp` `\MagS0ZerosLin` Then, we have true, linear, and asymptotic phase plots for the general second order transfer function.

```

\PhS0Zeros     195 \newcommand*\PhS0Poles}[2]{((-atan2((#1)*t,((#2) -
\PhS0ZerosAsymp \n@pow{t}{2}))) * \ph@scale)}
\PhS0ZerosLin  196 \newcommand*\PhS0PolesLin}[2]{((#2>0 ?
197   \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
198   (#1>0 ? -pi : pi))}
199 \newcommand*\PhS0PolesAsymp}[2]{((#2>0 ?
200   \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
201   (#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 θ - is necessary due to a bug in `gnuplot` (fixed in version 5.4, patchlevel 3).

```

202 \newcommand*\MagS0Zeros{\theta-\MagS0Poles}
203 \newcommand*\MagS0ZerosLin{\theta-\MagS0PolesLin}
204 \newcommand*\MagS0ZerosAsymp{\theta-\MagS0PolesAsymp}
205 \newcommand*\PhS0Zeros{\theta-\PhS0Poles}
206 \newcommand*\PhS0ZerosLin{\theta-\PhS0PolesLin}
207 \newcommand*\PhS0ZerosAsymp{\theta-\PhS0PolesAsymp}

```

`\MagS0PolesPeak` `\MagS0ZerosPeak` 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.

```

208 \newcommand*\MagS0PolesPeak}[3][]{
209   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
210   (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
211     20*log10(abs(#2/sqrt(abs(#3)))});
212 }
213 \newcommand*\MagS0ZerosPeak}[3][]{
214   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
215   (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
216     20*log10(abs(#2/sqrt(abs(#3)))});
217 }

```

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`.

```
218 \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 to so that only the macro `\opt@group` is expanded.

```
219 \parse@opt{#1}
220 \gdef\func@mag{}
221 \gdef\func@ph{}
222 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
    panded\expandafter{\opt@tikz}]}
223 \temp@cmd
224 \build@ZPK@plot{\func@mag}{\func@ph}{\opt@approx}{#2}
225 \edef\temp@cmd{\noexpand\begin{groupplot}[
226     bode@style,
227     xmin=#3,
228     xmax=#4,
229     domain=#3*\freq@scale:#4*\freq@scale,
230     height=2.5cm,
231     xmode=log,
232     group style = {group size = 1 by 2,vertical sep=0.25cm},
233     \opt@group
234 ]}
235 \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.

```
236 \edef\temp@mag@cmd{\noexpand\nextgroupplot [yla-
    bel={Gain (dB)}, xmajorticks=false, \optmag@axes]
237 \noexpand\addplot [freq@filter, variable=t, thick, \opt-
    mag@plot]}
238 \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
239 \noexpand\addplot [freq@filter, variable=t, thick, trig for-
    mat plots=rad, \optph@plot]}
240 \ifpgfarg
241 \temp@mag@cmd {\func@mag};
242 \optmag@commands
243 \temp@ph@cmd {\func@ph};
244 \optph@commands
245 \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.

```
246 \stepcounter{gnuplot@id}
247 \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
248 { set table $meta;
249   set dummy t;
250   set logscale x 10;
```

```

251         set xrange [#3*\freq@scale:#4*\freq@scale];
252         set samples \pgfkeysvalueof{/pgfplots/samples};
253         plot \func@mag;
254         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
255         plot "$meta" using ($1/(\freq@scale)):($2);
256     };
257     \optmag@commands
258     \stepcounter{gnuplot@id}
259     \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
260     { set table $meta;
261       set dummy t;
262       set logscale x 10;
263       set xrange [#3*\freq@scale:#4*\freq@scale];
264       set samples \pgfkeysvalueof{/pgfplots/samples};
265       plot \func@ph;
266       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
267       plot "$meta" using ($1/(\freq@scale)):($2);
268     };
269     \optph@commands
270   \fi
271 \end{groupplot}
272 \end{tikzpicture}
273 }

```

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

```

274 \AtBeginDocument{%
275   \if@babel
276   \let\Orig@BodeZPK\BodeZPK
277   \renewcommand{\BodeZPK}{%
278     \expandafter\shorthandoff\expandafter{\shorthand@list}
279     \BodeZPK@Shorthandoff
280   }
281   \newcommand{\BodeZPK@Shorthandoff}[4][]{%
282     \Orig@BodeZPK[#1]{#2}{#3}{#4}
283     \expandafter\shorthandon\expandafter{\shorthand@list}
284   }
285   \fi
286 }

```

\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.

```

287 \newcommand{\BodeTF}[4][]{
288   \parse@opt{#1}
289   \gdef\func@mag{}
290   \gdef\func@ph{}
291   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
292     panded\expandafter{\opt@tikz}]}
293   \temp@cmd
294   \build@TF@plot{\func@mag}{\func@ph}{#2}
295   \edef\temp@cmd{\noexpand\begin{groupplot}[
296     bode@style,
297     xmin=#3,
298     xmax=#4,
299     domain=#3*\freq@scale:#4*\freq@scale,
300     height=2.5cm,
301     xmode=log,
302     group style = {group size = 1 by 2,vertical sep=0.25cm},
303     \opt@group
304   ]}
305   \temp@cmd
306   \edef\temp@mag@cmd{\noexpand\nextgroupplot [yla-
307     bel={Gain (dB)}, xmajorticks=false, \optmag@axes]

```

```

306     \noexpand\addplot [freq@filter, variable=t, thick, \opt-
mag@plot]}
307     \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes
308     \noexpand\addplot [freq@filter, variable=t, thick, trig for-
mat plots=rad, \optph@plot]}
309     \if@pgfarg
310     \temp@mag@cmd {\func@mag};
311     \optmag@commands
312     \temp@ph@cmd {\n@mod{\func@ph}{2*pi*\ph@scale}};
313     \optph@commands
314     \else
315     \stepcounter{gnuplot@id}
316     \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
317     { set table $meta;
318     set dummy t;
319     set logscale x 10;
320     set xrange [#3*\freq@scale:#4*\freq@scale];
321     set samples \pgfkeysvalueof{/pgfplots/samples};
322     plot \func@mag;
323     set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
324     plot "$meta" using ($1/(\freq@scale)):($2);
325     };
326     \optmag@commands
327     \stepcounter{gnuplot@id}
328     \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
329     { set table $meta;
330     set dummy t;
331     set logscale x 10;
332     set xrange [#3*\freq@scale:#4*\freq@scale];
333     set samples \pgfkeysvalueof{/pgfplots/samples};
334     plot '+' using (t) : ((\func@ph)/(\ph@scale)) smooth unwrap;
335     set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
336     plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
337     };
338     \optph@commands
339     \fi
340     \end{groupplot}
341     \end{tikzpicture}
342 }

```

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

```

343 \AtBeginDocument{
344   \if@babel
345   \let\Orig@BodeTF\BodeTF
346   \renewcommand{\BodeTF}{%
347     \expandafter\shorthandoff\expandafter{\shorthand@list}
348     \BodeTF@Shorthandoff
349   }
350   \newcommand{\BodeTF@Shorthandoff}[4][[]]{%
351     \Orig@BodeTF[#1]{#2}{#3}{#4}
352     \expandafter\shorthandon\expandafter{\shorthand@list}
353   }
354   \fi
355 }

```

`\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.

```

356 \newcommand{\addBodeZPKPlots}[3][true/{}]{
357   \foreach \approx/\opt in {#1} {
358     \gdef\plot@macro{
359     \gdef\temp@macro{
360     \ifnum\pdf@strcmp{#2}{phase}=0
361     \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}
362     \else

```

```

363     \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}
364   \fi
365   \if@pgfarg
366     \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, thick, trig format plots=rad, \opt]}
367     \temp@cmd {\plot@macro};
368   \else
369     \stepcounter{gnuplot@id}
370     \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \opt]}
371     \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
372     { set table $meta;
373       set dummy t;
374       set logscale x 10;
375       set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
376       set samples \pgfkeysvalueof{/pgfplots/samples};
377       plot \plot@macro;
378       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
379       plot "$meta" using ($1/(\freq@scale)):($2);
380     };
381   \fi
382 }
383 }

```

`\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.

```

384 \newcommand{\addBodeTFPlot}[3][thick]{
385   \gdef\plot@macro{
386     \gdef\temp@macro{
387       \ifnum\pdf@stricmp{#2}{phase}=0
388         \build@TF@plot{\temp@macro}{\plot@macro}{#3}
389       \else
390         \build@TF@plot{\plot@macro}{\temp@macro}{#3}
391       \fi
392     \if@pgfarg
393       \ifnum\pdf@stricmp{#2}{phase}=0
394         \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, trig format plots=rad, #1]}
395         \temp@cmd {\n@mod{\plot@macro}{2*pi}};
396       \else
397         \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, #1]}
398         \temp@cmd {\plot@macro};
399       \fi
400     \else
401       \stepcounter{gnuplot@id}
402       \ifnum\pdf@stricmp{#2}{phase}=0
403         \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
404         { set table $meta;
405           set dummy t;
406           set logscale x 10;
407           set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
408           set samples \pgfkeysvalueof{/pgfplots/samples};
409           plot '+' using (t) : ((\plot@macro)/(\ph@scale)) smooth un-
wrap;
410           set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
411           plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
412         };
413       \else
414         \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
415         { set table $meta;

```

```

416         set dummy t;
417         set logscale x 10;
418         set xrange [ \freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
419         set samples \pgfkeysvalueof{/pgfplots/samples};
420         plot \plot@macro;
421         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
422         plot "$meta" using ($1/(\freq@scale)):(2);
423     };
424 \fi
425 \fi
426 }

```

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

```

427 \newcommand{\addBodeComponentPlot}[2][thick]{
428   \if@pgfarg
429     \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, trig format plots=rad, #1]}
430     \temp@cmd {#2};
431   \else
432     \stepcounter{gnuplot@id}
433     \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
434     { set table $meta;
435       set dummy t;
436       set logscale x 10;
437       set xrange [ \freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
438       set samples \pgfkeysvalueof{/pgfplots/samples};
439       plot #2;
440       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
441       plot "$meta" using ($1/(\freq@scale)):(2);
442     };
443   \fi
444 }

```

`\addBodePlot` Unified macro to add Bode plots supporting both ZPK and TF system representations. Automatically generates both magnitude and phase plot commands for later insertion.

```

445 \newcommand{\addBodePlot}[3][]{
446   \parse@add@Bode@opt{#1}
447   \gdef\plot@macro{}
448   \gdef\temp@macro{}
449   \ifnum\pdf@strcmp{#2}{zpk}=0
450     \build@ZPK@plot{\plot@macro}{\temp@macro}{\opt@approx}{#3}
451   \else
452     \ifnum\pdf@strcmp{#2}{tf}=0
453       \build@TF@plot{\plot@macro}{\temp@macro}{#3}
454     \else
455       \PackageError {bodeplot} {Unknown system representation '#2'.}
456       {Supported representations are 'zpk' and 'tf' .}
457     \fi
458   \fi
459   \if@pgfarg
460     \xdef\plot@a@cmd{\unexpanded\expandafter{\plot@a@cmd}\noexpand\addplot [freq@fil-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}, vari-
able=t, thick, trig format plots=rad, \opt@plot] {\plot@macro};}
461     \xdef\plot@b@cmd{\unexpanded\expandafter{\plot@b@cmd}\noexpand\addplot [freq@fil-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}, vari-
able=t, thick, trig format plots=rad, \opt@plot] {\temp@macro};}
462   \else
463     \stepcounter{gnuplot@id}
464     \edef\gnu@id{\arabic{gnuplot@id}}
465     \build@gnu@plot{\plot@macro}{\gnu@id}

```

```

466 \xdef\plot@a@cmd{\unexpanded\expandafter{\plot@a@cmd}\noexpand\addplot [vari-
able=t, thick, \opt@plot]\gnu@cmd}
467 \stepcounter{gnuplot@id}
468 \edef\gnu@id{\arabic{gnuplot@id}}
469 \ifnum\pdf@stricmp{#2}{zpk}=0
470 \build@gnu@plot{\temp@macro}{\gnu@id}
471 \else
472 \ifnum\pdf@stricmp{#2}{tf}=0
473 \build@gnu@unwrap@plot{\temp@macro}{\gnu@id}
474 \else
475 \PackageError {bodeplot} {Unknown system representa-
tion '#2'.}
476 {Supported representations are 'zpk' and 'tf'.}
477 \fi
478 \fi
479 \xdef\plot@b@cmd{\unexpanded\expandafter{\plot@b@cmd}\noexpand\addplot [vari-
able=t, thick, \opt@plot]\gnu@cmd}
480 \fi
481 }

```

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

```

482 \AtBeginDocument{%
483 \if@babel
484 \AddToHook{env/BodePhPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand@
485 \AddToHook{env/BodePhPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@
486 \fi
487 }
488 \NewDocumentEnvironment{BodePhPlot}{0}{mm+b}{
489 \parse@env@opt{#1}
490 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
491 \temp@cmd
492 \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
493 ph@y@label,
494 freq@label,
495 bode@style,
496 xmin={#2},
497 xmax={#3},
498 domain=#2:#3,
499 height=2.5cm,
500 \unexpanded\expandafter{\opt@axes}
501 ]}
502 \temp@cmd
503 #4
504 \end{semilogxaxis}
505 \end{tikzpicture}
506 }{}

```

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 `semilogaxis` environments.

```

507 \AtBeginDocument{%
508 \if@babel
509 \AddToHook{env/BodeMagPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand@
510 \AddToHook{env/BodeMagPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@
511 \fi
512 }
513 \NewDocumentEnvironment{BodeMagPlot}{0}{mm+b}{
514 \parse@env@opt{#1}

```

```

515 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
516 \temp@cmd
517 \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
518     bode@style,
519     freq@label,
520     xmin={#2},
521     xmax={#3},
522     domain=#2:#3,
523     height=2.5cm,
524     ylabel={Gain (dB)},
525     \unexpanded\expandafter{\opt@axes}
526 ]}
527 \temp@cmd
528 #4
529 \end{semilogxaxis}
530 \end{tikzpicture}
531 }{}

```

BodePlot (*env.*) Enhanced version of the **BodePlot** environment that works with the unified **\addBodePlot** macro. Creates a grouped plot with magnitude on top and phase on bottom, automatically collecting and inserting plot commands generated by **\addBodePlot** calls within the environment body.

```

532 \NewDocumentEnvironment{BodePlot}{0}{mm+b}{
533   \gdef\plot@a@cmd{}
534   \gdef\plot@b@cmd{}
535   \parse@opt{#1}
536   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
537   \temp@cmd
538   \edef\temp@cmd{\noexpand\begin{groupplot}[
539     bode@style,
540     xmin=#2,
541     xmax=#3,
542     domain=#2*\freq@scale:#3*\freq@scale,
543     height=2.5cm,
544     xmode=log,
545     group style = {group size = 1 by 2,vertical sep=0.25cm},
546     \opt@group
547 ]}
548   \temp@cmd
549   #4
550   \edef\temp@mag@cmd{\noexpand\nextgroupplot [yla-
bel={Gain (dB)}, xmajorticks=false, \optmag@axes]}
551   \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]}
552   \temp@mag@cmd
553   \plot@a@cmd
554   \temp@ph@cmd
555   \plot@b@cmd
556   \end{groupplot}
557   \end{tikzpicture}
558 }{}

```

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](#).

```

559 \newcommand*\add@feature}[3]{
560   \ifcat$\detokenize\expandafter{#1}$

```



```

561     \xdef#1{\unexpanded\expandafter{#1 0+#2}}
562 \else
563     \xdef#1{\unexpanded\expandafter{#1+#2}}
564 \fi
565 \foreach \y [count=\n] in #3 {
566     \xdef#1{\unexpanded\expandafter{#1}{\y}}
567     \xdef\Last@LoopValue{\n}
568 }
569 \ifnum\Last@LoopValue=1
570     \xdef#1{\unexpanded\expandafter{#1}{0}}
571 \fi
572 }

```

`\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.

```

573 \newcommand{\build@ZPK@plot}[4]{
574   \foreach \feature/\values in {#4} {
575     \ifnum\pdf@strcmp{\feature}{z}=0
576       \foreach \z in \values {
577         \ifnum\pdf@strcmp{#3}{linear}=0
578           \add@feature{#2}{\PhZeroLin}{\z}
579           \add@feature{#1}{\MagZeroLin}{\z}
580         \else
581           \ifnum\pdf@strcmp{#3}{asymptotic}=0
582             \add@feature{#2}{\PhZeroAsymp}{\z}
583             \add@feature{#1}{\MagZeroAsymp}{\z}
584           \else
585             \add@feature{#2}{\PhZero}{\z}
586             \add@feature{#1}{\MagZero}{\z}
587           \fi
588         \fi
589       }
590     \fi
591     \ifnum\pdf@strcmp{\feature}{p}=0
592       \foreach \p in \values {
593         \ifnum\pdf@strcmp{#3}{linear}=0
594           \add@feature{#2}{\PhPoleLin}{\p}
595           \add@feature{#1}{\MagPoleLin}{\p}
596         \else
597           \ifnum\pdf@strcmp{#3}{asymptotic}=0
598             \add@feature{#2}{\PhPoleAsymp}{\p}
599             \add@feature{#1}{\MagPoleAsymp}{\p}
600           \else
601             \add@feature{#2}{\PhPole}{\p}
602             \add@feature{#1}{\MagPole}{\p}
603           \fi
604         \fi
605       }
606     \fi
607     \ifnum\pdf@strcmp{\feature}{k}=0
608       \ifnum\pdf@strcmp{#3}{linear}=0
609         \add@feature{#2}{\PhKLin}{\values}
610         \add@feature{#1}{\MagKLin}{\values}
611       \else
612         \ifnum\pdf@strcmp{#3}{asymptotic}=0
613           \add@feature{#2}{\PhKAsymp}{\values}
614           \add@feature{#1}{\MagKAsymp}{\values}

```

```

615         \else
616             \add@feature{#2}{\PhK}{\values}
617             \add@feature{#1}{\MagK}{\values}
618         \fi
619     \fi
620 \fi
621 \ifnum\pdf@strcmp{\feature}{d}=0
622     \ifnum\pdf@strcmp{#3}{linear}=0
623         \PackageError {bodeplot} {Linear approximation for pure de-
624 lays is not
625         supported.} {Plot the true Bode plot using 'true' in-
626 stead of 'linear'.}
627     \else
628         \ifnum\pdf@strcmp{#3}{asymptotic}=0
629             \PackageError {bodeplot} {Asymptotic approxima-
630 tion for pure delays is not
631 supported.} {Plot the true Bode plot using 'true' in-
632 stead of 'asymptotic'.}
633     \else
634         \ifdim\values pt < 0pt
635             \PackageError {bodeplot} {Delay needs to be a posi-
636 tive number.}
637         \fi
638         \add@feature{#2}{\PhDel}{\values}
639         \add@feature{#1}{\MagDel}{\values}
640     \fi
641 \fi
642 \fi
643 }
644 }

```

`\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).

```

640 \newcommand{\build@TF@plot}[3]{
641     \gdef\num@real{0}
642     \gdef\num@im{0}
643     \gdef\den@real{0}
644     \gdef\den@im{0}
645     \gdef\loop@delay{0}
646     \foreach \feature/\values in {#3} {
647         \ifnum\pdf@strcmp{\feature}{num}=0
648             \foreach \numcoeff [count=\numpow] in \values {
649                 \xdef\num@degree{\numpow}
650             }
651             \foreach \numcoeff [count=\numpow] in \values {
652                 \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}
653                 \ifnum\currentdegree = 0
654                     \xdef\num@real{\num@real+\numcoeff}
655                 \else
656                     \ifodd\currentdegree
657                         \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-
658 1}{(\currentdegree-1)/2}))*%
659                             (\n@pow{t}{\currentdegree}})}
660                     \else
661                         \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-
662 1}{(\currentdegree)/2}))*%
663                             (\n@pow{t}{\currentdegree}})}
664                 \fi
665             \fi

```

```

666     \ifnum\pdf@strcmp{\feature}{den}=0
667     \foreach \dencoeff [count=\denpow] in \values {
668     \xdef\den@degree{\denpow}
669     }
670     \foreach \dencoeff [count=\denpow] in \values {
671     \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}
672     \ifnum\currentdegree = 0
673     \xdef\den@real{\den@real+\dencoeff}
674     \else
675     \ifodd\currentdegree
676     \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-
677     1}{(\currentdegree-1)/2})*%
678     (\n@pow{t}{\currentdegree}))}
679     \else
680     \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-
681     1}{(\currentdegree)/2})*%
682     (\n@pow{t}{\currentdegree}))}
683     \fi
684     \fi
685     \ifnum\pdf@strcmp{\feature}{d}=0
686     \xdef\loop@delay{\values}
687     \fi
688     }
689     \xdef#2{((atan2((\num@im),(\num@real))-atan2((\den@im),%
690     (\den@real))-\loop@delay*t)*(\ph@scale))}
691     \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2})))-%
692     20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2}))))}
693 }

```

`\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.

```

694 \newcommand{\parse@opt}[1]{
695   \gdef\optmag@axes{}
696   \gdef\optph@axes{}
697   \gdef\optph@plot{}
698   \gdef\optmag@plot{}
699   \gdef\opt@group{}
700   \gdef\opt@approx{}
701   \gdef\optph@commands{}
702   \gdef\optmag@commands{}
703   \gdef\opt@tikz{}
704   \foreach \obj/\typ/\opt in {#1} {
705     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
706     \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
707     \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
708     \else
709     \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
710     \xdef\optph@plot{\unexpanded\expandafter{\opt}}
711     \else
712     \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
713     \xdef\optph@plot{\unexpanded\expandafter{\opt}}
714     \fi
715     \fi
716     \else

```

```

717 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
718 \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
719 \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
720 \else
721 \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
722 \xdef\optph@axes{\unexpanded\expandafter{\opt}}
723 \else
724 \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
725 \xdef\optph@axes{\unexpanded\expandafter{\opt}}
726 \fi
727 \fi
728 \else
729 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{group}=0
730 \xdef\opt@group{\unexpanded\expandafter{\opt}}
731 \else
732 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{approx}=0
733 \xdef\opt@approx{\unexpanded\expandafter{\opt}}
734 \else
735 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
736 \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
737 \xdef\optph@commands{\unexpanded\expandafter{\opt}}
738 \else
739 \xdef\optmag@commands{\unexpanded\expandafter{\opt}}
740 \fi
741 \else
742 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
743 \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
744 \else
745 \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
746 \unexpanded\expandafter{\obj}}
747 \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
748 \unexpanded\expandafter{\obj}}
749 \fi
750 \fi
751 \fi
752 \fi
753 \fi
754 \fi
755 }
756 }

```

`\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.

```

757 \newcommand{\parse@env@opt}[1]{
758 \gdef\opt@axes{}
759 \gdef\opt@tikz{}
760 \foreach \obj/\opt in {#1} {
761 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
762 \xdef\opt@axes{\unexpanded\expandafter{\opt}}
763 \else
764 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
765 \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
766 \else
767 \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
768 \unexpanded\expandafter{\obj}}
769 \fi
770 \fi
771 }
772 }

```

`\parse@add@Bode@opt` Parses options for the unified `\addBodePlot` macro. Handles `linear` and `asymptotic` approximation options plus general plot styling options.

```

773 \newcommand{\parse@add@Bode@opt}[1]{
774   \gdef\opt@plot{}
775   \gdef\opt@approx{}
776   \foreach \obj/\opt in {#1} {
777     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{linear}=0
778       \xdef\opt@approx{\unexpanded\expandafter{\opt}}
779     \else
780       \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{asymptotic}=0
781         \xdef\opt@approx{\unexpanded\expandafter{\opt}}
782       \else
783         \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
784           \unexpanded\expandafter{\obj}}
785       \fi
786     \fi
787   }
788 }

```

`\build@gnu@plot` Helper macro to assemble a general `gnuplot` invocation for parametric plots.

```

789 \newcommand{\build@gnu@plot}[2]{
790   \xdef\gnu@cmd{ gnuplot [raw gnuplot, id=#2, prefix=\bodeplot@prefix]
791     { set table $meta;
792       set dummy t;
793       set logscale x 10;
794       set xrange [\pgfkeysvalueof{/pgfplots/domain}];
795       set samples \pgfkeysvalueof{/pgfplots/samples};
796       plot #1;
797       set table "\bodeplot@prefix#2.table";
798       plot "$meta" using ($1/(\freq@scale)):($2);
799     };}
800 }

```

`\build@gnu@unwrap@plot` Helper macro to assemble a `gnuplot` invocation with phase unwrapping for TF phase plots.

```

801 \newcommand{\build@gnu@unwrap@plot}[2]{
802   \xdef\gnu@cmd{ gnuplot [raw gnuplot, id=#2, prefix=\bodeplot@prefix]
803     { set table $meta;
804       set dummy t;
805       set logscale x 10;
806       set trange [\pgfkeysvalueof{/pgfplots/domain}];
807       set samples \pgfkeysvalueof{/pgfplots/samples};
808       plot '+' using (t) : ((#1)/(\ph@scale)) smooth unwrap;
809       set table "\bodeplot@prefix#2.table";
810       plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
811     };}
812 }

```

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!

```

813 \newcommand{\NyquistZPK}[4][]{

```

```

814 \parse@N@opt{#1}
815 \gdef\func@mag{}
816 \gdef\func@ph{}
817 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
818 \temp@cmd
819 \build@ZPK@plot{\func@mag}{\func@ph}{#2}
820 \edef\temp@cmd{\noexpand\begin{axis}[
821     bode@style,
822     domain=#3*\freq@scale:#4*\freq@scale,
823     height=5cm,
824     xlabel={\Re$},
825     ylabel={\Im$},
826     samples=500,
827     \unexpanded\expandafter{\opt@axes}
828 ]}
829 \temp@cmd
830 \addplot [only marks,mark=+,thick,red] (-1 , 0);
831 \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig for-
mat plots=rad, \unexpanded\expandafter{\opt@plot}]}
832 \if@pgfarg
833 \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
834             {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
835 \opt@commands
836 \else
837 \stepcounter{gnuplot@id}
838 \temp@cmd gnuplot [parametric, gnuplot@prefix] {
839     \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
840     \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
841 };
842 \opt@commands
843 \fi
844 \end{axis}
845 \end{tikzpicture}
846 }

```

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

```

847 \AtBeginDocument{%
848     \if@babel
849     \let\Orig@NyquistZPK\NyquistZPK
850     \renewcommand{\NyquistZPK}{%
851         \expandafter\shorthandoff\expandafter{\shorthand@list}
852         \NyquistZPK@Shorthandoff
853     }
854     \newcommand{\NyquistZPK@Shorthandoff}[4][]{%
855         \Orig@NyquistZPK[#1]{#2}{#3}{#4}
856         \expandafter\shorthandon\expandafter{\shorthand@list}
857     }
858     \fi
859 }

```

\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**.

```

860 \newcommand{\NyquistTF}[4][]{
861     \parse@N@opt{#1}
862     \gdef\func@mag{}
863     \gdef\func@ph{}
864     \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
865     \temp@cmd
866     \build@TF@plot{\func@mag}{\func@ph}{#2}
867     \edef\temp@cmd{\noexpand\begin{axis}[
868         bode@style,

```

```

869     domain=#3*\freq@scale:#4*\freq@scale,
870     height=5cm,
871     xlabel={\$Re\$},
872     ylabel={\$Im\$},
873     samples=500,
874     \unexpanded\expandafter{\opt@axes}
875   ]]
876   \temp@cmd
877     \addplot [only marks, mark=+, thick, red] (-1 , 0);
878     \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig for-
mat plots=rad, \unexpanded\expandafter{\opt@plot}]}
879     \if@pgfarg
880       \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
881         {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
882     \opt@commands
883   \else
884     \stepcounter{gnuplot@id}
885     \temp@cmd gnuplot [parametric, gnuplot@prefix] {
886       \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
887       \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
888     };
889     \opt@commands
890   \fi
891 \end{axis}
892 \end{tikzpicture}
893 }

```

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

```

894 \AtBeginDocument{%
895   \if@babel
896   \let\Orig@NyquistTF\NyquistTF
897   \renewcommand{\NyquistTF}{%
898     \expandafter\shorthandoff\expandafter{\shorthand@list}
899     \NyquistTF@Shorthandoff
900   }
901   \newcommand{\NyquistTF@Shorthandoff}[4][]{%
902     \Orig@NyquistTF[#1]{#2}{#3}{#4}
903     \expandafter\shorthandon\expandafter{\shorthand@list}
904   }
905   \fi
906 }

```

`\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.

```

907 \newcommand{\addNyquistZPKPlot}[2][]{
908   \gdef\func@mag{}
909   \gdef\func@ph{}
910   \build@ZPK@plot{\func@mag}{\func@ph}{#2}
911   \if@pgfarg
912     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, trig format plots=rad, #1]}
913     \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
914       {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
915   \else
916     \stepcounter{gnuplot@id}
917     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
918     \temp@cmd gnuplot [parametric, gnuplot@prefix] {
919       \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
920       \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
921     };
922   \fi

```

923 }

\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.

```

924 \newcommand{\addNyquistTFPlot}[2][]{
925   \gdef\func@mag{}
926   \gdef\func@ph{}
927   \build@TF@plot{\func@mag}{\func@ph}{#2}
928   \if@pgfarg
929     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, trig format plots=rad, #1]}
930     \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
931               {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
932   \else
933     \stepcounter{gnuplot@id}
934     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
935     \temp@cmd gnuplot [parametric, gnuplot@prefix]{
936       \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
937       \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
938     };
939   \fi
940 }
```

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.

```

941 \AtBeginDocument{%
942   \if@babel
943     \AddToHook{env/NyquistPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand
944     \AddToHook{env/NyquistPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@
945   \fi
946 }
947 \NewDocumentEnvironment{NyquistPlot}{0}{mm+b}{
948   \parse@env@opt{#1}
949   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
950   \temp@cmd
951   \edef\temp@cmd{\noexpand\begin{axis}[
952     bode@style,
953     height=5cm,
954     domain=#2:#3,
955     xlabel={\$Re\$},
956     ylabel={\$Im\$},
957     \unexpanded\expandafter{\opt@axes}
958   ]}
959   \temp@cmd
960   \addplot [only marks,mark=+,thick,red] (-1 , 0);
961   #4
962   \end{axis}
963 \end{tikzpicture}
964 }{}
```

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**, **scale**, or **tikz** then the corresponding **\opt** are passed, unexpanded, to the **\addplot** macro, the **axis** environment, the scaling option, and the **tikzpicture** environment, respectively.


```

965 \newcommand{\parse@N@opt}[1]{
966   \gdef\opt@axes{}
967   \gdef\opt@plot{}
968   \gdef\opt@commands{}
969   \gdef\opt@tikz{}
970   \gdef\opt@scale{linear}
971   \foreach \obj/\opt in {#1} {
972     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
973       \xdef\opt@axes{\unexpanded\expandafter{\opt}}
974     \else
975       \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
976         \xdef\opt@plot{\unexpanded\expandafter{\opt}}
977       \else
978         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
979           \xdef\opt@commands{\unexpanded\expandafter{\opt}}
980         \else
981           \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
982             \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
983           \else
984             \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{scale}=0
985               \xdef\opt@scale{\unexpanded\expandafter{\opt}}
986             \else
987               \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
988                 \unexpanded\expandafter{\obj}}
989             \fi
990           \fi
991         \fi
992       \fi
993     \fi
994   }
995 }

```

`\min@real@ZPK` Computes the minimum nonzero absolute value of real parts from all poles and zeros in ZPK format. This is used for automatically setting the threshold in logarithmic pole-zero maps. The result is stored in `\min@re@threshold@result`.

```

996 \newcommand{\min@real@ZPK}[1]{
997   % Initialize with large default value
998   \gdef\min@re@threshold@result{1000}
999   \def\@min@false{false}
1000  \gdef\min@threshold@found{false}
1001  % Keep a float version in FPU format for safe comparisons under Lu-
    aLaTeX
1002  \global\let\min@thresh@float\relax
1003  % Track maximum absolute value for axis sizing
1004  \pgfkeys{/pgf/fpu=true}
1005  \pgfmathparse{0}
1006  \global\let\max@re@float=\pgfmathresult
1007  % Track positive and negative sides separately
1008  \pgfmathparse{0}
1009  \global\let\max@re@pos@float=\pgfmathresult
1010  \pgfmathparse{0}
1011  \global\let\max@re@neg@float=\pgfmathresult
1012  \pgfkeys{/pgf/fpu=false}
1013  \gdef\max@re@value{0}
1014  \gdef\has@positive@values{false}
1015  \gdef\has@negative@values{false}
1016  \foreach \feature/\values in {#1} {
1017    \ifnum\pdf@strcmp{\feature}{z}=0
1018      % Process zeros
1019      \foreach \z in \values {
1020        \foreach \y [count=\zcnt] in \z {
1021          \ifnum\zcnt=1
1022            % Compute absolute value using PGF FPU to avoid TeX di-

```

```

men overflows
1023     \pgfkeys{/pgf/fpu=true}
1024     \pgfmathparse{abs(\y)}
1025     \let\abs@valuefloat=\pgfmathresult
1026     \pgfmathfloattofixed{\abs@valuefloat}
1027     \edef\abs@value{\pgfmathresult}
1028     \pgfkeys{/pgf/fpu=false}
1029     % Skip if zero (string compare avoids numeric parser)
1030     \ifnum\pdf@strcmp{\abs@value}{0}=0\else
1031     \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
1032     % Check if value is positive or nega-
tive and track separately
1033     \pgfkeys{/pgf/fpu=true}
1034     \pgfmathparse{\y >= 0 ? 1 : 0}
1035     \pgfmathfloattoint{\pgfmathresult}
1036     \pgfkeys{/pgf/fpu=false}
1037     \ifnum\pgfmathresult=1
1038     % Positive value
1039     \gdef\has@positive@values{true}
1040     \pgfkeys{/pgf/fpu=true}
1041     \pgfmathparse{\abs@valuefloat > \max@re@pos@float ? 1 : 0}
1042     \pgfmathfloattoint{\pgfmathresult}
1043     \pgfkeys{/pgf/fpu=false}
1044     \ifnum\pgfmathresult=1
1045     \global\let\max@re@pos@float=\abs@valuefloat
1046     \fi
1047     \else
1048     % Negative value
1049     \gdef\has@negative@values{true}
1050     \pgfkeys{/pgf/fpu=true}
1051     \pgfmathparse{\abs@valuefloat > \max@re@neg@float ? 1 : 0}
1052     \pgfmathfloattoint{\pgfmathresult}
1053     \pgfkeys{/pgf/fpu=false}
1054     \ifnum\pgfmathresult=1
1055     \global\let\max@re@neg@float=\abs@valuefloat
1056     \fi
1057     \fi
1058     % Update overall maximum tracker
1059     \pgfkeys{/pgf/fpu=true}
1060     \pgfmathparse{\abs@valuefloat > \max@re@float ? 1 : 0}
1061     \pgfmathfloattoint{\pgfmathresult}
1062     \pgfkeys{/pgf/fpu=false}
1063     \ifnum\pgfmathresult=1
1064     \global\let\max@re@float=\abs@valuefloat
1065     \xdef\max@re@value{\abs@value}
1066     \fi
1067     \ifx\min@threshold@found\@min@false
1068     % First valid nonzero value
1069     \xdef\min@re@threshold@result{\abs@value}
1070     \global\let\min@thresh@float=\abs@valuefloat
1071     \gdef\min@threshold@found{true}
1072     \else
1073     % Compare floats safely with FPU; then trun-
cate boolean to an int
1074     \pgfkeys{/pgf/fpu=true}
1075     \pgfmathparse{\abs@valuefloat < \min@thresh@float ? 1 : 0}
1076     \pgfmathfloattoint{\pgfmathresult}
1077     \pgfkeys{/pgf/fpu=false}
1078     \ifnum\pgfmathresult=1
1079     \xdef\min@re@threshold@result{\abs@value}
1080     \global\let\min@thresh@float=\abs@valuefloat
1081     \fi
1082     \fi

```

```

1083         \fi
1084     \fi
1085 \fi
1086 }
1087 }
1088 \fi
1089 \ifnum\pdf@strcmp{\feature}{p}=0
1090 % Process poles
1091 \foreach \p in \values {
1092     \foreach \y [count=\pcnt] in \p {
1093         \ifnum\pcnt=1
1094             % Compute absolute value using PGF FPU to avoid TeX di-
men overflows
1095             \pgfkeys{/pgf/fpu=true}
1096             \pgfmathparse{abs(\y)}
1097             \let\abs@valuefloat=\pgfmathresult
1098             \pgfmathfloattofixed{\abs@valuefloat}
1099             \edef\abs@value{\pgfmathresult}
1100             \pgfkeys{/pgf/fpu=false}
1101             % Skip if zero (string compare avoids numeric parser)
1102             \ifnum\pdf@strcmp{\abs@value}{0}=0\else
1103                 \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
1104                     % Check if value is positive or nega-
tive and track separately
1105                     \pgfkeys{/pgf/fpu=true}
1106                     \pgfmathparse{\y >= 0 ? 1 : 0}
1107                     \pgfmathfloattoint{\pgfmathresult}
1108                     \pgfkeys{/pgf/fpu=false}
1109                     \ifnum\pgfmathresult=1
1110                         % Positive value
1111                         \gdef\has@positive@values{true}
1112                         \pgfkeys{/pgf/fpu=true}
1113                         \pgfmathparse{\abs@valuefloat > \max@re@pos@float ? 1 : 0}
1114                         \pgfmathfloattoint{\pgfmathresult}
1115                         \pgfkeys{/pgf/fpu=false}
1116                         \ifnum\pgfmathresult=1
1117                             \global\let\max@re@pos@float=\abs@valuefloat
1118                         \fi
1119                     \else
1120                         % Negative value
1121                         \gdef\has@negative@values{true}
1122                         \pgfkeys{/pgf/fpu=true}
1123                         \pgfmathparse{\abs@valuefloat > \max@re@neg@float ? 1 : 0}
1124                         \pgfmathfloattoint{\pgfmathresult}
1125                         \pgfkeys{/pgf/fpu=false}
1126                         \ifnum\pgfmathresult=1
1127                             \global\let\max@re@neg@float=\abs@valuefloat
1128                         \fi
1129                     \fi
1130                 \ifx\min@threshold@found\@min@false
1131                     % First valid nonzero value
1132                     \xdef\min@re@threshold@result{\abs@value}
1133                     \global\let\min@thresh@float=\abs@valuefloat
1134                     \gdef\min@threshold@found{true}
1135                 \else
1136                     % Compare floats safely with FPU; then trun-
cate boolean to an int
1137                     \pgfkeys{/pgf/fpu=true}
1138                     \pgfmathparse{\abs@valuefloat < \min@thresh@float ? 1 : 0}
1139                     \pgfmathfloattoint{\pgfmathresult}
1140                     \pgfkeys{/pgf/fpu=false}
1141                     \ifnum\pgfmathresult=1
1142                         \xdef\min@re@threshold@result{\abs@value}

```

```

1143         \global\let\min@thresh@float=\abs@valuefloat
1144     \fi
1145     \fi
1146     \fi
1147     \fi
1148     \fi
1149     }
1150 }
1151 \fi
1152 }
1153 % If no valid values found, use default
1154 \ifx\min@threshold@found\@min@false
1155     \gdef\min@re@threshold@result{0.01}
1156 \fi
1157 \xdef\min@threshold@result{\min@re@threshold@result}
1158 % Compute \min@re@pow@10 such that  $10^{\min@re@pow@10}$  is the clos-
est power of 10 smaller than or equal to \min@re@threshold@result
1159 \pgfkeys{/pgf/fpu=true}
1160 \pgfmathparse{log10(\min@re@threshold@result)}
1161 \let\log@result=\pgfmathresult
1162 % Add small epsilon to handle floating-point precision is-
sues with exact powers of 10
1163 \pgfmathparse{\log@result + 1e-5}
1164 \let\log@adjusted=\pgfmathresult
1165 \pgfmathparse{floor(\log@adjusted)}
1166 \pgfmathfloattofixed{\pgfmathresult}
1167 \xdef\min@re@pow@10{\pgfmathresult}
1168 \xdef\min@pow@10{\min@re@pow@10}
1169 % Compute separate maximum exponents for positive and negative sides
1170 % Positive side
1171 \ifx\has@positive@values\@min@false
1172     \xdef\max@re@pos@pow@10{\min@re@pow@10}
1173 \else
1174     \pgfmathparse{log10(max(\max@re@pos@float,1e-100))}
1175     \let\log@max@re@pos=\pgfmathresult
1176     \pgfmathparse{\log@max@re@pos + 1e-5}
1177     \let\log@max@re@pos@adjusted=\pgfmathresult
1178     \pgfmathparse{ceil(\log@max@re@pos@adjusted)}
1179     \pgfmathfloattoint{\pgfmathresult}
1180     \xdef\max@re@pos@pow@10{\pgfmathresult}
1181 \fi
1182 % Negative side
1183 \ifx\has@negative@values\@min@false
1184     \xdef\max@re@neg@pow@10{\min@re@pow@10}
1185 \else
1186     \pgfmathparse{log10(max(\max@re@neg@float,1e-100))}
1187     \let\log@max@re@neg=\pgfmathresult
1188     \pgfmathparse{\log@max@re@neg + 1e-5}
1189     \let\log@max@re@neg@adjusted=\pgfmathresult
1190     \pgfmathparse{ceil(\log@max@re@neg@adjusted)}
1191     \pgfmathfloattoint{\pgfmathresult}
1192     \xdef\max@re@neg@pow@10{\pgfmathresult}
1193 \fi
1194 % Keep overall maximum for backward compatibility
1195 \pgfmathparse{max(\max@re@pos@float > 0 ? \max@re@pos@float : 0, \max@re@neg@float)}
1196 \let\max@re@valuefloat=\pgfmathresult
1197 \pgfmathparse{\max@re@valuefloat > 0 ? \max@re@valuefloat : \min@re@threshold@resu}
1198 \let\max@re@valuefloat=\pgfmathresult
1199 \pgfmathparse{log10(max(\max@re@valuefloat,1e-100))}
1200 \let\log@max@re=\pgfmathresult
1201 \pgfmathparse{\log@max@re + 1e-5}
1202 \let\log@max@re@adjusted=\pgfmathresult
1203 \pgfmathparse{ceil(\log@max@re@adjusted)}

```

```

1204 \pgfmathfloattoint{\pgfmathresult}
1205 \xdef\max@re@pow@10{\pgfmathresult}
1206 \pgfkeys{/pgf/fpu=false}
1207 }
1208
1209

```

`\min@im@ZPK` Computes the minimum nonzero absolute value of imaginary parts from all poles and zeros in ZPK format. This is used for automatically setting thresholds in logarithmic pole-zero maps for imaginary axis scaling. The result is stored in `\min@im@threshold@result` and the corresponding power of 10 is stored in `\min@im@pow@10`.

```

1210 \newcommand{\min@im@ZPK}[1]{
1211   % Initialize with large default value
1212   \gdef\min@im@threshold@result{1000}
1213   \def\min@false{false}
1214   \gdef\min@im@threshold@found{false}
1215   % Keep a float version in FPU format for safe comparisons under Lu-
LaTeX
1216   \global\let\min@im@thresh@float\relax
1217   % Track maximum absolute value for axis sizing
1218   \pgfkeys{/pgf/fpu=true}
1219   \pgfmathparse{0}
1220   \global\let\max@im@float=\pgfmathresult
1221   \pgfkeys{/pgf/fpu=false}
1222   \gdef\max@im@value{0}
1223   \foreach \feature/\values in {#1} {
1224     \ifnum\pdf@strcmp{\feature}{z}=0
1225       % Process zeros
1226       \foreach \z in \values {
1227         \foreach \y [count=\zcnt] in \z {
1228           \ifnum\zcnt=2
1229             % Second element is imaginary part - compute abso-
lute value using PGF FPU
1230             \pgfkeys{/pgf/fpu=true}
1231             \pgfmathparse{abs(\y)}
1232             \let\abs@valuefloat=\pgfmathresult
1233             \pgfmathfloattofixed{\abs@valuefloat}
1234             \edef\abs@value{\pgfmathresult}
1235             \pgfkeys{/pgf/fpu=false}
1236             % Skip if zero (string compare avoids numeric parser)
1237             \ifnum\pdf@strcmp{\abs@value}{0}=0\else
1238               \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
1239                 % Update maximum tracker
1240                 \pgfkeys{/pgf/fpu=true}
1241                 \pgfmathparse{\abs@valuefloat > \max@im@float ? 1 : 0}
1242                 \pgfmathfloattoint{\pgfmathresult}
1243                 \pgfkeys{/pgf/fpu=false}
1244                 \ifnum\pgfmathresult=1
1245                   \global\let\max@im@float=\abs@valuefloat
1246                   \xdef\max@im@value{\abs@value}
1247                 \fi
1248                 \ifx\min@im@threshold@found\min@false
1249                   % First valid nonzero value
1250                   \xdef\min@im@threshold@result{\abs@value}
1251                   \global\let\min@im@thresh@float=\abs@valuefloat
1252                   \gdef\min@im@threshold@found{true}
1253                 \else
1254                   % Compare floats safely with FPU; then trun-
cate boolean to an int
1255                   \pgfkeys{/pgf/fpu=true}
1256                   \pgfmathparse{\abs@valuefloat < \min@im@thresh@float ? 1 : 0}
1257                   \pgfmathfloattoint{\pgfmathresult}

```

```

1258         \pgfkeys{/pgf/fpu=false}
1259         \ifnum\pgfmathresult=1
1260             \xdef\min@im@threshold@result{\abs@value}
1261             \global\let\min@im@thresh@float=\abs@valuefloat
1262         \fi
1263     \fi
1264 \fi
1265 \fi
1266 \fi
1267 }
1268 }
1269 \fi
1270 \ifnum\pdf@strcmp{\feature}{p}=0
1271     % Process poles
1272     \foreach \p in \values {
1273         \foreach \y [count=\pcnt] in \p {
1274             \ifnum\pcnt=2
1275                 % Second element is imaginary part - compute absolute value using PGF FPU
1276                 \pgfkeys{/pgf/fpu=true}
1277                 \pgfmathparse{abs(\y)}
1278                 \let\abs@valuefloat=\pgfmathresult
1279                 \pgfmathfloattofixed{\abs@valuefloat}
1280                 \edef\abs@value{\pgfmathresult}
1281                 \pgfkeys{/pgf/fpu=false}
1282                 % Skip if zero (string compare avoids numeric parser)
1283                 \ifnum\pdf@strcmp{\abs@value}{0}=0\else
1284                     \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
1285                         \ifx\min@im@threshold@found\min@false
1286                             % First valid nonzero value
1287                             \xdef\min@im@threshold@result{\abs@value}
1288                             \global\let\min@im@thresh@float=\abs@valuefloat
1289                             \gdef\min@im@threshold@found{true}
1290                         \else
1291                             % Compare floats safely with FPU; then truncate boolean to an int
1292                             \pgfkeys{/pgf/fpu=true}
1293                             \pgfmathparse{\abs@valuefloat < \min@im@thresh@float ? 1 : 0}
1294                             \pgfmathfloattoint{\pgfmathresult}
1295                             \pgfkeys{/pgf/fpu=false}
1296                             \ifnum\pgfmathresult=1
1297                                 \xdef\min@im@threshold@result{\abs@value}
1298                                 \global\let\min@im@thresh@float=\abs@valuefloat
1299                             \fi
1300                         \fi
1301                     \fi
1302                 \fi
1303             \fi
1304         }
1305     }
1306 \fi
1307 }
1308 % If no valid values found, use default
1309 \ifx\min@im@threshold@found\min@false
1310     \gdef\min@im@threshold@result{0.01}
1311 \fi
1312 % Compute \min@im@pow@10 such that  $10^{\min@im@pow@10}$  is the closest power of 10 smaller than or equal to \min@im@threshold@result
1313 \pgfkeys{/pgf/fpu=true}
1314 \pgfmathparse{log10(\min@im@threshold@result)}
1315 \let\log@result=\pgfmathresult
1316 % Add small epsilon to handle floating-point precision issues with exact powers of 10

```

```

1317 \pgfmathparse{\log@result + 1e-5}
1318 \let\log@adjusted=\pgfmathresult
1319 \pgfmathparse{floor(\log@adjusted)}
1320 \pgfmathfloattofixed{\pgfmathresult}
1321 \xdef\min@im@pow@10{\pgfmathresult}
1322 \xdef\min@Im@pow@10{\min@im@pow@10}
1323 % Compute maximum exponent to determine axis extent for imagi-
      nary parts
1324 \pgfmathparse{\max@im@float > 0 ? \max@im@float : \min@im@threshold@result}
1325 \let\max@im@valuefloat=\pgfmathresult
1326 \pgfmathparse{log10(max(\max@im@valuefloat,1e-100))}
1327 \let\log@max@im=\pgfmathresult
1328 \pgfmathparse{\log@max@im + 1e-5}
1329 \let\log@max@im@adjusted=\pgfmathresult
1330 \pgfmathparse{ceil(\log@max@im@adjusted)}
1331 \pgfmathfloattoint{\pgfmathresult}
1332 \xdef\max@im@pow@10{\pgfmathresult}
1333 \pgfkeys{/pgf/fpu=false}
1334 }
1335
1336

```

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 1337 \newcommand{\NicholsZPK}[4][{}]{
\addNicholsZPKChart 1338 \parse@N@opt{#1}
\addNicholsTFChart 1339 \gdef\func@mag{}
1340 \gdef\func@ph{}
1341 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
      panded\expandafter{\opt@tikz}]}
1342 \temp@cmd
1343 \build@ZPK@plot{\func@mag}{\func@ph}{{#2}
1344 \edef\temp@cmd{\noexpand\begin{axis}[
1345 ph@x@label,
1346 bode@style,
1347 domain=#3*\freq@scale:#4*\freq@scale,
1348 height=5cm,
1349 ylabel={Gain (dB)},
1350 samples=500,
1351 \unexpanded\expandafter{\opt@axes}
1352 ]}
1353 \temp@cmd
1354 \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig for-
      mat plots=rad, \opt@plot]}
1355 \ifpgfarg
1356 \temp@cmd ( {\func@ph} , {\func@mag} );
1357 \opt@commands
1358 \else
1359 \stepcounter{gnuplot@id}
1360 \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
1361 { set table $meta;
1362 set logscale x 10;
1363 set dummy t;
1364 set samples \pgfkeysvalueof{/pgfplots/samples};
1365 set trange [#3*\freq@scale:#4*\freq@scale];
1366 plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1367 unset logscale x;
1368 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1369 plot "$meta" using ($2*\ph@scale):($1);
1370 };
1371 \opt@commands

```

```

1372     \fi
1373     \end{axis}
1374 \end{tikzpicture}
1375 }
1376 \AtBeginDocument{%
1377   \if@babel
1378   \let\Orig@NicholsZPK\NicholsZPK
1379   \renewcommand{\NicholsZPK}{%
1380     \expandafter\shorthandoff\expandafter{\shorthand@list}
1381     \NicholsZPK@Shorthandoff
1382   }
1383   \newcommand{\NicholsZPK@Shorthandoff}[4][[]]{%
1384     \Orig@NicholsZPK[#1]{#2}{#3}{#4}
1385     \expandafter\shorthandon\expandafter{\shorthand@list}
1386   }
1387   \fi
1388 }
1389 \newcommand{\NicholsTF}[4][[]]{
1390   \parse@N@opt{#1}
1391   \gdef\func@mag{}
1392   \gdef\func@ph{}
1393   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
1394     panded\expandafter{\opt@tikz}]}
1395   \temp@cmd
1396     \build@TF@plot{\func@mag}{\func@ph}{#2}
1397     \edef\temp@cmd{\noexpand\begin{axis}[
1398       ph@x@label,
1399       bode@style,
1400       domain=#3*\freq@scale:#4*\freq@scale,
1401       height=5cm,
1402       ylabel={Gain (dB)},
1403       samples=500,
1404       \unexpanded\expandafter{\opt@axes}
1405     ]}
1406     \temp@cmd
1407     \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig for-
1408     mat plots=rad, \opt@plot]}
1409     \if@pgfarg
1410       \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
1411       \opt@commands
1412     \else
1413       \stepcounter{gnuplot@id}
1414       \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
1415       { set table $meta1;
1416         set logscale x 10;
1417         set dummy t;
1418         set samples \pgfkeysvalueof{/pgfplots/samples};
1419         set trange [#3*\freq@scale:#4*\freq@scale];
1420         plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1421         unset logscale x;
1422         set table $meta2;
1423         plot "$meta1" using ($1):($2) smooth unwrap;
1424         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1425         plot "$meta2" using ($2*\ph@scale):($1);
1426       };
1427     \opt@commands
1428     \fi
1429   \end{axis}
1430 \end{tikzpicture}
1431 }
1432 \AtBeginDocument{
1433   \if@babel
1434   \let\Orig@NicholsTF\NicholsTF

```



```

1433 \renewcommand{\NicholsTF}{%
1434 \expandafter\shorthandoff\expandafter{\shorthand@list}
1435 \NicholsTF@Shorthandoff
1436 }
1437 \newcommand{\NicholsTF@Shorthandoff}[4][[]]{%
1438 \Orig@NicholsTF[#1]{#2}{#3}{#4}
1439 \expandafter\shorthandon\expandafter{\shorthand@list}
1440 }
1441 \AddToHook{env/NicholsChart/begin}{\expandafter\shorthandoff\expandafter{\shorthand@list}}
1442 \AddToHook{env/NicholsChart/end}{\expandafter\shorthandon\expandafter{\shorthand@list}}
1443 \fi
1444 }
1445 \NewDocumentEnvironment{NicholsChart}{0}{mm+b}{
1446 \parse@env@opt{#1}
1447 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
1448 \temp@cmd
1449 \edef\temp@cmd{\noexpand\begin{axis}[
1450 ph@x@label,
1451 bode@style,
1452 domain=#2:#3,
1453 height=5cm,
1454 ylabel={Gain (dB)},
1455 \unexpanded\expandafter{\opt@axes}
1456 ]}
1457 \temp@cmd
1458 #4
1459 \end{axis}
1460 \end{tikzpicture}
1461 ]{}
1462 \newcommand{\addNicholsZPKChart}[2][[]]{
1463 \gdef\func@mag{}
1464 \gdef\func@ph{}
1465 \build@ZPK@plot{\func@mag}{\func@ph}{#2}
1466 \if@pgfarg
1467 \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
1468 \temp@cmd ( {\func@ph} , {\func@mag} );
1469 \else
1470 \stepcounter{gnuplot@id}
1471 \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1472 { set table $meta;
1473 set logscale x 10;
1474 set dummy t;
1475 set samples \pgfkeysvalueof{/pgfplots/samples};
1476 set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1477 plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1478 unset logscale x;
1479 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1480 plot "$meta" using ($2*\ph@scale):($1);
1481 };
1482 \fi
1483 }
1484 \newcommand{\addNicholsTFChart}[2][[]]{
1485 \gdef\func@mag{}
1486 \gdef\func@ph{}
1487 \build@TF@plot{\func@mag}{\func@ph}{#2}
1488 \if@pgfarg
1489 \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
1490 \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
1491 \else
1492 \stepcounter{gnuplot@id}

```

```

1493 \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1494 { set table $meta1;
1495   set logscale x 10;
1496   set dummy t;
1497   set samples \pgfkeysvalueof{/pgfplots/samples};
1498   set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1499   plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1500   unset logscale x;
1501   set table $meta2;
1502   plot "$meta1" using ($1):($2) smooth unwrap;
1503   set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1504   plot "$meta2" using ($2*\ph@scale):($1);
1505 };
1506 \fi
1507 }

```

\PoleZeroMapZPK Creates a pole-zero map similar to MATLAB's `pzmap` function. Poles are plotted as 'x' markers and zeros as 'o' markers on the complex s-plane. The gain parameter is ignored since it does not affect pole-zero locations, and delay is also ignored since it does not add poles or zeros.

```

1508 \newcommand{\PoleZeroMapZPK}[2][[]]{
1509   \parseN@opt{#1}
1510   % Set threshold for log scale if needed
1511   \ifnum\pdf@strcmp{\opt@scale}{log}=0
1512   % Compute automatic threshold as the minimum nonzero absolute real part
1513   % from the provided ZPK data.
1514   \min@real@ZPK{#2}
1515   \min@im@ZPK{#2}
1516   % Prepare axis scaling helpers
1517   \pgfkeys{/pgf/fpu=true}
1518   % Compute separate tick counts for positive and negative x-axis
1519   \ifx\has@positive@values\@min@false
1520     \xdef\PoleZeroMapZPK@ticksXPos{0}
1521   \else
1522     \pgfmathparse{max(\max@re@pos@pow@10 - \min@re@pow@10 + 1, 1)}
1523     \pgfmathfloatoint{\pgfmathresult}
1524     \xdef\PoleZeroMapZPK@ticksXPos{\pgfmathresult}
1525   \fi
1526   \ifx\has@negative@values\@min@false
1527     \xdef\PoleZeroMapZPK@ticksXNeg{0}
1528   \else
1529     \pgfmathparse{max(\max@re@neg@pow@10 - \min@re@pow@10 + 1, 1)}
1530     \pgfmathfloatoint{\pgfmathresult}
1531     \xdef\PoleZeroMapZPK@ticksXNeg{\pgfmathresult}
1532   \fi
1533   \pgfkeys{/pgf/fpu=false}
1534   \def\PoleZeroMapZPK@formatXTick##1{%
1535     \pgfmathtruncatemacro{\PoleZeroMapZPK@tick}{##1}%
1536     \ifnum\PoleZeroMapZPK@tick=0
1537       $0$
1538     \else
1539       \pgfmathtruncatemacro{\PoleZeroMapZPK@exp}{\min@re@pow@10 + abs(\PoleZeroMapZPK@tick)}%
1540     }%
1541     \ifnum\PoleZeroMapZPK@tick>0
1542       $10^{\PoleZeroMapZPK@exp}$%
1543     \else
1544       $-10^{\PoleZeroMapZPK@exp}$%
1545     \fi
1546   \fi
1547 }
1548 \def\PoleZeroMapZPK@formatYTick##1{%
1549   \pgfmathtruncatemacro{\PoleZeroMapZPK@tick}{##1}%

```

```

1549     \ifnum\PoleZeroMapZPK@tick=0
1550         $0$
1551     \else
1552         \pgfmathtruncatemacro{\PoleZeroMapZPK@exp}{\min@im@pow@10 + abs(\PoleZeroMapZ
1553         1)%
1554         \ifnum\PoleZeroMapZPK@tick>0
1555             $10^{\PoleZeroMapZPK@exp}$%
1556         \else
1557             $-10^{\PoleZeroMapZPK@exp}$%
1558         \fi
1559     \fi
1560     \def\PoleZeroMapZPK@xticklabel{\PoleZeroMapZPK@formatXTick{\tick}}
1561     \def\PoleZeroMapZPK@yticklabel{\PoleZeroMapZPK@formatYTick{\tick}}
1562     \fi
1563     \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
1564     panded\expandafter{\opt@tikz}]}
1565     \temp@cmd
1566     % Set up axis based on scale option
1567     \ifnum\pdf@strcmp{\opt@scale}{log}=0
1568         \edef\temp@cmd{\noexpand\begin{axis}[
1569         xlabel={$\Re$},
1570         ylabel={$\Im$},
1571         axis lines=center,
1572         grid=major,
1573         height=6cm,
1574         enlarge x limits=0.2,
1575         enlarge y limits=0.2,
1576         xtick distance=1,
1577         ytick distance=1,
1578         xticklabel=\noexpand\PoleZeroMapZPK@xticklabel,
1579         yticklabel=\noexpand\PoleZeroMapZPK@yticklabel,
1580         x filter/.expression={abs(x) < \min@re@threshold@result ? 0 : (x >= 0 ? (log:
1581         100)) - \min@re@pow@10 + 1) : (-log10(max(min(-x, 1e100), 1e-
1582         100)) + \min@re@pow@10 - 1))},
1583         y filter/.expression={abs(y) < \min@im@threshold@result ? 0 : (y >= 0 ? (log:
1584         100)) - \min@im@pow@10 + 1) : (-log10(max(min(-y, 1e100), 1e-
1585         100)) + \min@im@pow@10 - 1))},
1586         \unexpanded\expandafter{\opt@axes}
1587     ]}
1588     \else
1589         \edef\temp@cmd{\noexpand\begin{axis}[
1590         xlabel={$\Re$},
1591         ylabel={$\Im$},
1592         axis lines=center,
1593         grid=major,
1594         height=6cm,
1595         enlarge x limits=0.2,
1596         enlarge y limits=0.2,
1597         \unexpanded\expandafter{\opt@axes}
1598     ]}
1599     \fi
1600     \temp@cmd
1601     % Plot poles and zeros from ZPK data
1602     \foreach \feature/\values in {#2} {
1603         \ifnum\pdf@strcmp{\feature}{z}=0
1604             \foreach \z in \values {
1605                 \foreach \y [count=\zcnt] in \z {
1606                     \ifnum\zcnt=1
1607                         \xdef\zreal{\y}
1608                     \fi
1609                     \ifnum\zcnt=2
1610                         \xdef\zimag{\y}

```

```

1606         \fi
1607         \xdef\Last@LoopValue@z{\zcnt}
1608     }
1609     \ifnum\Last@LoopValue@z=1
1610         \xdef\zimag{0}
1611     \fi
1612     \edef\temp@cmd{\noexpand\addplot [only marks, mark=o, mark size=3pt, thi
expanded\expandafter{\opt@plot}]]
1613         \temp@cmd coordinates {(\zreal,\zimag)};
1614     }
1615     \fi
1616     \ifnum\pdf@strcmp{\feature}{p}=0
1617         \foreach \p in \values {
1618             \foreach \y [count=\pcnt] in \p {
1619                 \ifnum\pcnt=1
1620                     \xdef\preal{\y}
1621                 \fi
1622                 \ifnum\pcnt=2
1623                     \xdef\pimag{\y}
1624                 \fi
1625                 \xdef\Last@LoopValue@p{\pcnt}
1626             }
1627             \ifnum\Last@LoopValue@p=1
1628                 \xdef\pimag{0}
1629             \fi
1630             \edef\temp@cmd{\noexpand\addplot [only marks, mark=x, mark size=3pt, thi
expanded\expandafter{\opt@plot}]]
1631                 \temp@cmd coordinates {(\preal,\pimag)};
1632         }
1633     \fi
1634 }
1635 \opt@commands
1636 \end{axis}
1637 \end{tikzpicture}
1638 }

```

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

```

1639 \AtBeginDocument{%
1640     \if@babel
1641     \let\Orig@PoleZeroMapZPK\PoleZeroMapZPK
1642     \renewcommand{\PoleZeroMapZPK}{%
1643         \expandafter\shorthandoff\expandafter{\shorthand@list}
1644         \PoleZeroMapZPK@Shorthandoff
1645     }
1646     \newcommand{\PoleZeroMapZPK@Shorthandoff}[2][]{%
1647         \Orig@PoleZeroMapZPK[#1]{#2}
1648         \expandafter\shorthandon\expandafter{\shorthand@list}
1649     }
1650     \fi
1651 }

```

Index

Numbers written in *italics* 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	<code>\@declutterargtrue</code> .. 7	<code>\@ifpackageloaded</code> .. 46
<code>\@babelfalse</code> 44	<code>\@empty</code> 48	<code>\@min@false</code> 999, 1067,
<code>\@babeltrue</code> 47	<code>\@hzargfalse</code> 13	1130, 1154, 1171,
<code>\@declutterargfalse</code> . 5	<code>\@hzargtrue</code> 15	1183, 1213, 1248,

	1285, 1309, 1519, 1526		
\@nil	120, 121	\BodeTF	287
\@pgfargfalse	1	\BodeTF@Shorthandoff	348, 350
\@pgfargtrue	3	\BodeZPK	218
\@radargfalse	9	\BodeZPK@Shorthandoff	279, 281
\@radargtrue	11	\build@gnu@plot	465, 470, 789
\~	51	\build@gnu@unwrap@plot	473, 801
A		\build@TF@plot	293, 388, 390, 453, 640, 866, 927, 1395, 1487
\abs@value	1027, 1030, 1031, 1065, 1069, 1079, 1099, 1102, 1103, 1132, 1142, 1234, 1237, 1238, 1246, 1250, 1260, 1280, 1283, 1284, 1287, 1297	\build@ZPK@plot	224, 361, 363, 450, 573, 819, 910, 1343, 1465
\abs@valuefloat	1025, 1026, 1041, 1045, 1051, 1055, 1060, 1064, 1070, 1075, 1080, 1097, 1098, 1113, 1117, 1123, 1127, 1133, 1138, 1143, 1232, 1233, 1241, 1245, 1251, 1256, 1261, 1278, 1279, 1288, 1293, 1298	C	
\add@feature	559, 578, 579, 582, 583, 585, 586, 594, 595, 598, 599, 601, 602, 609, 610, 613, 614, 616, 617, 633, 634	\currentdegree	652, 653, 656, 657, 658, 660, 661, 671, 672, 675, 676, 677, 679, 680
\addBodeComponentPlot	427	D	
\addBodePlot	445	\den@degree	668, 671
\addBodeTFPlot	384	\den@im	644, 676, 689, 692
\addBodeZPKPlots	356	\den@real	643, 673, 679, 690, 692
\addNicholsTFChart	1337	\dencoeff	667, 670, 673, 676, 679
\addNicholsZPKChart	1337	\denpow	667, 668, 670, 671
\addNyquistTFPlot	924	\do	49
\addNyquistZPKPlot	907	\dospecials	55
\AddToHook	484, 485, 509, 510, 943, 944, 1441, 1442	E	
\AtBeginDocument	45, 274, 343, 482, 507, 847, 894, 941, 1376, 1430, 1639	\endgroup	53
B		environments:	
\beginngroup	50	BodeMagPlot	507
\bode@style	58	BodePhPlot	482
BodeMagPlot (env.)	507	BodePlot	532
BodePhPlot (env.)	482	F	
BodePlot (env.)	532	\freq@filter	73
\bodeplot@prefix	28, 30, 35, 254, 266, 323, 335, 378, 410, 421, 440, 790, 797, 802, 809, 1368, 1422, 1479, 1503	\freq@label	73
		\freq@scale	73, 74, 85, 109, 116, 229, 251, 255, 263, 267, 298, 320, 324, 332, 336, 366, 375, 379, 394, 397, 407, 411, 418, 422, 429, 437, 441, 460, 461, 542, 798, 810, 822, 869, 912, 917, 929, 934, 1347, 1365, 1399, 1417, 1467, 1476, 1489, 1498
		\func@mag	220, 224, 241, 253, 289, 293, 310, 322, 815, 819, 833, 834, 839, 840, 862, 866, 880, 881, 886, 887, 908, 910, 913, 914,
			919, 920, 925, 927, 930, 931, 936, 937, 1339, 1343, 1356, 1366, 1391, 1395, 1408, 1418, 1463, 1465, 1468, 1477, 1485, 1487, 1490, 1499
		\func@ph	221, 224, 243, 265, 290, 293, 312, 334, 816, 819, 833, 834, 839, 840, 863, 866, 880, 881, 886, 887, 909, 910, 913, 914, 919, 920, 926, 927, 930, 931, 936, 937, 1340, 1343, 1356, 1366, 1392, 1395, 1408, 1418, 1464, 1465, 1468, 1477, 1486, 1487, 1490, 1499
		G	
		\g@addto@macro	52
		\get@interval@end	120, 121
		\get@interval@start	120, 120
		\global	1002, 1006, 1009, 1011, 1045, 1055, 1064, 1070, 1080, 1117, 1127, 1133, 1143, 1216, 1220, 1245, 1251, 1261, 1288, 1298
		\gnu@cmd	466, 479, 790, 802
		\gnu@id	464, 465, 468, 470, 473
		\gnuplot@id	1
		\gnuplot@prefix	1
		H	
		\has@negative@values	1015, 1049, 1121, 1183, 1526
		\has@positive@values	1014, 1039, 1111, 1171, 1519
		I	
		\if@babel	44
		\if@hzarg	13, 84
		\ifbabelshorthand	52
		\ifcat	560
		\ifnum	360, 387, 393, 402, 449, 452, 469, 472, 569, 575, 577, 581, 591, 593, 597, 607, 608, 612, 621, 622, 626, 647, 653, 666, 672, 685, 705, 706, 709, 717, 718, 721, 729, 732, 735, 736, 742,

761, 764, 777, 780, 972, 975, 978, 981, 984, 1017, 1021, 1030, 1031, 1037, 1044, 1054, 1063, 1078, 1089, 1093, 1102, 1103, 1109, 1116, 1126, 1141, 1224, 1228, 1237, 1238, 1244, 1259, 1270, 1274, 1283, 1284, 1296, 1511, 1536, 1540, 1549, 1553, 1566, 1598, 1601, 1604, 1609, 1616, 1619, 1622, 1627	\MagKAsymp 122, 614	1142, 1155, 1157, 1158, 1160, 1197, 1579
\ifwindows 38	\MagKLin 122, 610	\min@real@ZPK . 996, 1514
\ifx 1067, 1130, 1154, 1171, 1183, 1248, 1285, 1309, 1519, 1526	\MagPole . . . 130, 157, 602	\min@thresh@float . 1002, 1070, 1075, 1080, 1133, 1138, 1143
\immediate 40	\MagPoleAsymp 130, 159, 599	\min@thresh@found 1000, 1067, 1071, 1130, 1134, 1154
J	\MagPoleLin 130, 158, 595	\min@thresh@result 1157
\jobname 28, 30	\MagSOPoles 190	N
L	\MagSOPolesAsymp . . 190	\n 565, 567
\Last@LoopValue 567, 569	\MagSOPolesLin 190	\n@mod 1, 312, 395, 1408, 1490
\Last@LoopValue@p 1625, 1627	\MagSOPolesPeak . . . 208	\n@mod@n 1
\Last@LoopValue@z 1607, 1609	\MagS0Zeros 190	\n@mod@p 1, 138
\lccode 51	\MagS0ZerosAsymp . . 190	\n@pow 1, 131, 132, 133, 143, 144, 146, 147, 149, 150, 151, 152, 153, 154, 163, 164, 167, 168, 169, 171, 173, 174, 191, 195, 657, 658, 660, 661, 676, 677, 679, 680, 691, 692, 833, 834, 839, 840, 880, 881, 886, 887, 913, 914, 919, 920, 930, 931, 936, 937
\log@adjusted . 1164, 1165, 1318, 1319	\MagS0ZerosLin 190	\newcounter 25
\log@max@im . . 1327, 1328	\MagS0ZerosPeak . . . 208	\NewDocumentEnvironment 488,
\log@max@im@adjusted 1329, 1330	\MagZero 157, 586	513, 532, 947, 1445
\log@max@re . . 1200, 1201	\MagZeroAsymp . . 157, 583	\NicholsChart 1337
\log@max@re@adjusted 1202, 1203	\MagZeroLin 157, 579	\NicholsTF 1337
\log@max@re@neg 1187, 1188	\max@im@float . 1220, 1241, 1245, 1324	\NicholsTF@Shorthandoff 1435, 1437
\log@max@re@neg@adjusted 1189, 1190	\max@im@pow@ 1332	\NicholsZPK 1337
\log@max@re@pos 1175, 1176	\max@im@value 1222, 1246	\NicholsZPK@Shorthandoff 1381, 1383
\log@max@re@pos@adjusted 1177, 1178	\max@im@valuefloat 1325, 1326	\num@degree 649, 652
\log@result . . . 1161, 1163, 1315, 1317	\max@re@float 1006, 1060, 1064	\num@im 642, 657, 689, 691
\loop@delay 645, 686, 690	\max@re@neg@float 1011, 1051, 1055, 1123, 1127, 1186, 1195	\num@real 641, 654, 660, 689, 691
\lowercase 52	\max@re@neg@pow@ 1184, 1192, 1529	\numcoeff 648, 651, 654, 657, 660
M	\max@re@pos@float 1009, 1041, 1045, 1113, 1117, 1174, 1195	\numpy 648, 649, 651, 652
\MagCSPoles 163	\max@re@pos@pow@ 1172, 1180, 1522	\NyquistPlot 941
\MagCSPolesAsymp . . 163	\max@re@pow@ 1205	\NyquistTF 860
\MagCSPolesLin 163	\max@re@value 1013, 1065	\NyquistTF@Shorthandoff 899, 901
\MagCSPolesPeak . . . 182	\max@re@valuefloat 1196, 1197, 1198, 1199	\NyquistZPK 813
\MagCSZeros 163	\min@Im@pow@ 1322	\NyquistZPK@Shorthandoff 852, 854
\MagCSZerosAsymp . . 163	\min@im@pow@ . . 1312, 1321, 1322, 1552, 1580	O
\MagCSZerosLin 163	\min@im@thresh@float 1216, 1251, 1256, 1261, 1288, 1293, 1298	\opt@approx 224, 450, 700, 733, 775, 778, 781
\MagCSZerosPeak . . . 182	\min@im@thresh@found 1214, 1248, 1252, 1285, 1289, 1309	\opt@axes 500, 525, 758, 762, 767, 827, 874, 957,
\MagDel 128, 634	\min@im@threshold@result 1212, 1250, 1260, 1287, 1297, 1310, 1312, 1314, 1324, 1580	
\MagK 122, 617	\min@im@ZPK . . 1210, 1515	
	\min@pow@ 1168	
	\min@re@pow@ 1158, 1167, 1168, 1172, 1184, 1522, 1529, 1539, 1579	
	\min@re@thresh@result 998, 1069, 1079, 1132,	

966, 973, 1351, 1403, 1455, 1581, 1592	608, 612, 621, 622, 626, 647, 666, 685, 705, 706, 709, 717, 718, 721, 729, 732, 735, 736, 742, 761, 764, 777, 780, 972, 975, 978, 981, 984, 1017, 1030, 1031, 1089, 1102, 1103, 1224, 1237, 1238, 1270, 1283, 1284, 1511, 1566, 1598, 1616	1025, 1027, 1035, 1037, 1042, 1044, 1052, 1054, 1061, 1063, 1076, 1078, 1097, 1099, 1107, 1109, 1114, 1116, 1124, 1126, 1139, 1141, 1161, 1164, 1166, 1167, 1175, 1177, 1179, 1180, 1187, 1189, 1191, 1192, 1196, 1198, 1200, 1202, 1204, 1205, 1220, 1232, 1234, 1242, 1244, 1257, 1259, 1278, 1280, 1294, 1296, 1315, 1318, 1320, 1321, 1325, 1327, 1329, 1331, 1332, 1523, 1524, 1530, 1531
\opt@commands	\pgfkeys	\pgfplotsset
835, 842, 882, 889, 968, 979, 1357, 1371, 1409, 1425, 1635	1004, 1012, 1023, 1028, 1033, 1036, 1040, 1043, 1050, 1053, 1059, 1062, 1074, 1077, 1095, 1100, 1105, 1108, 1112, 1115, 1122, 1125, 1137, 1140, 1159, 1206, 1218, 1221, 1230, 1235, 1240, 1243, 1255, 1258, 1276, 1281, 1292, 1295, 1313, 1333, 1517, 1533	58, 73, 75, 76, 77, 80, 81, 86, 88, 97, 98, 102, 103, 110, 112, 117
\opt@group	\pgfkeysvalueof	\ph@scale
233, 302, 546, 699, 730	252, 264, 321, 333, 366, 375, 376, 394, 397, 407, 408, 418, 419, 429, 437, 438, 460, 461, 794, 795, 806, 807, 912, 917, 929, 934, 1364, 1416, 1467, 1475, 1476, 1489, 1497, 1498	73, 78, 82, 96, 101, 125, 129, 140, 153, 156, 168, 174, 175, 195, 312, 334, 336, 409, 411, 690, 808, 810, 833, 834, 839, 840, 880, 881, 886, 887, 913, 914, 919, 920, 930, 931, 936, 937, 1366, 1369, 1408, 1418, 1423, 1477, 1480, 1490, 1499, 1504
\opt@plot 460, 461, 466, 479, 774, 783, 831, 878, 967, 976, 987, 1354, 1406, 1612, 1630	\pgfmathfloattofixed	\ph@x@label
\opt@scale	1026, 1098, 1166, 1233, 1279, 1320	73
970, 985, 1511, 1566	\pgfmathfloattoint	\ph@y@label
\opt@tikz	1035, 1042, 1052, 1061, 1076, 1107, 1114, 1124, 1139, 1179, 1191, 1204, 1242, 1257, 1294, 1331, 1523, 1530	73
222, 291, 490, 515, 536, 703, 743, 759, 765, 817, 864, 949, 969, 982, 1341, 1393, 1447, 1563	\pgfmathparse	\PhCSPoles
\optmag@axes	1005, 1008, 1010, 1024, 1034, 1041, 1051, 1060, 1075, 1096, 1106, 1113, 1123, 1138, 1160, 1163, 1165, 1174, 1176, 1178, 1186, 1188, 1190, 1195, 1197, 1199, 1201, 1203, 1219, 1231, 1241, 1256, 1277, 1293, 1314, 1317, 1319, 1324, 1326, 1328, 1330, 1522, 1529	163
236, 305, 550, 695, 719, 724	\PhK	\PhCSPolesAsymp 163, 200
\optmag@commands 242, 257, 311, 326, 702, 739	\PhKAsymp	\PhCSPolesLin
\optmag@plot	122, 128, 613	163, 197
237, 306, 698, 707, 712, 745	\PhKLin	\PhCSZeros
\optph@axes	130, 160, 601	163
238, 307, 551, 696, 722, 725	\PhPole	\PhCSZerosAsymp
\optph@commands 244, 269, 313, 338, 701, 737	\PhPoleAsymp 130, 162, 598	163
\optph@plot	\PhPoleLin . 130, 161, 594	\PhCSZerosLin
239, 308, 697, 710, 713, 747	\PhSOPoles	163
\Orig@BodeTF	\PhSOPolesAsymp	\PhDel
345, 351	190	129, 633
\Orig@BodeZPK	\PhSOPolesLin	\PhK
276, 282	190	122, 616
\Orig@NicholsTF	\PhS0Zeros	\PhKAsymp
1432, 1438	190	122, 128, 609
\Orig@NicholsZPK	\PhS0ZerosAsymp	\PhKLin
1378, 1384	190	130, 160, 601
\Orig@NyquistTF 896, 902	\PhS0ZerosLin	\PhPoleAsymp 130, 162, 598
\Orig@NyquistZPK 849, 855	190	\PhPoleLin . 130, 161, 594
\Orig@PoleZeroMapZPK	\PhS0Zeros	\PhSOPoles
1641, 1647	190	\PhSOPolesAsymp
P	\PhS0ZerosAsymp	190
\p	\PhS0ZerosLin	\PhS0Zeros
592, 594, 595, 598, 599, 601, 602, 1091, 1092, 1272, 1273, 1617, 1618	190	190
\parse@add@Bode@opt	\PhZero	\PhS0ZerosAsymp
446, 773	157, 585	190
\parse@env@opt	\PhZeroAsymp	\PhS0ZerosLin
489, 514, 757, 948, 1446	157, 582	190
\parse@N@opt 814, 861, 965, 1338, 1390, 1509	\PhZeroLin	\PhZero
\parse@opt	1623, 1628, 1631	157, 585
219, 288, 535, 694	\pimag	157, 582
\pcnt		157, 588
1092, 1093, 1273, 1274, 1618, 1619, 1622, 1625		157, 582
\pdf@strcmp		157, 588
360, 387, 393, 402, 449, 452, 469, 472, 575, 577, 581, 591, 593, 597, 607,		157, 588

<code>\plot@a@cmd</code>	851, 856, 898, 903,	<code>\temp@macro</code>	359, 361, 363,
.. 460, 466, 533, 553	943, 944, 1380,	386, 388, 390, 448,	450, 453, 461, 470, 473
<code>\plot@b@cmd</code>	1385, 1434, 1439,	<code>\temp@mag@cmd</code>	236, 241, 247,
.. 461, 479, 534, 555	1441, 1442, 1643, 1648	305, 310, 316, 550, 552	
<code>\plot@macro</code>	<code>\shorthandoff</code>	<code>\temp@ph@cmd</code>	238, 243, 259,
358, 361, 363, 367,	.. 278, 347, 484,	307, 312, 328, 551, 554	
377, 385, 388, 390,	509, 851, 898, 943,	<code>\tick</code>	1560, 1561
395, 398, 409, 420,	1380, 1434, 1441, 1643		
447, 450, 453, 460, 465	<code>\shorthandon</code>		
<code>\PoleZeroMapZPK</code> ..	283, 352, 485,		
<code>\PoleZeroMapZPK@exp</code>	510, 856, 903, 944,		
..... 1539, 1541,	1385, 1439, 1442, 1648		
1543, 1552, 1554, 1556	<code>\stepcounter</code> ...		
<code>\PoleZeroMapZPK@formatXTick</code>	258, 315, 327, 369,		
..... 1534, 1560	401, 432, 463, 467,		
<code>\PoleZeroMapZPK@formatYTick</code>	837, 884, 916, 933,		
..... 1547, 1561	1359, 1411, 1470, 1492		
<code>\PoleZeroMapZPK@Shorthandoff</code>			
..... 1644, 1646			
<code>\PoleZeroMapZPK@tick</code>	T		
..... 1535,	<code>\temp@cmd</code>		
1536, 1539, 1540,	222, 223, 225, 235,		
1548, 1549, 1552, 1553	291, 292, 294, 304,		
<code>\PoleZeroMapZPK@ticksXNeg</code>	366, 367, 370, 371,		
..... 1527, 1531	394, 395, 397, 398,		
<code>\PoleZeroMapZPK@ticksXPos</code>	429, 430, 490, 491,		
..... 1520, 1524	492, 502, 515, 516,		
<code>\PoleZeroMapZPK@xticklabel</code>	517, 527, 536, 537,		
..... 1560, 1577	538, 548, 817, 818,		
<code>\PoleZeroMapZPK@yticklabel</code>	820, 829, 831, 833,		
..... 1561, 1578	838, 864, 865, 867,		
<code>\preal</code>	876, 878, 880, 885,		
1620, 1631	912, 913, 917, 918,		
	929, 930, 934, 935,		
	949, 950, 951, 959,		
	1341, 1342, 1344,		
	1353, 1354, 1356,		
	1360, 1393, 1394,		
	1396, 1405, 1406,		
	1408, 1412, 1447,		
	1448, 1449, 1457,		
	1467, 1468, 1489,		
	1490, 1563, 1564,		
	1567, 1584, 1595,		
	1612, 1613, 1630, 1631		
R			
<code>\renewcommand</code>			
... 96, 101, 109,			
116, 277, 346, 850,			
897, 1379, 1433, 1642			
S			
<code>\setcounter</code>	26		
<code>\shorthand@list</code> ..	44,		
278, 283, 347, 352,			
484, 485, 509, 510,			
		V	
		<code>\values</code>	574,
		576, 592, 609, 610,	
		613, 614, 616, 617,	
		630, 633, 634, 646,	
		648, 651, 667, 670,	
		686, 1016, 1019,	
		1091, 1223, 1226,	
		1272, 1597, 1599, 1617	
		W	
		<code>\write</code>	40
		Y	
		<code>\y</code>	565, 566,
		1020, 1024, 1034,	
		1092, 1096, 1106,	
		1227, 1231, 1273,	
		1277, 1600, 1602,	
		1605, 1618, 1620, 1623	
		Z	
		<code>\z</code>	576, 578,
		579, 582, 583, 585,	
		586, 1019, 1020,	
		1226, 1227, 1599, 1600	
		<code>\zcnt</code>	1020,
		1021, 1227, 1228,	
		1600, 1601, 1604, 1607	
		<code>\zimag</code> ..	1605, 1610, 1613
		<code>\zreal</code>	1602, 1613

Change History

v1.0			
General: Initial release	1		
v1.0.1			
\addBodeZPKPlots : Improved optional argument handling.	28		
\BodeZPK : Pass arbitrary TikZ commands as options.	26		
v1.0.2			
gnuplot@prefix : Fixed issue #1	21		
v1.0.3			
BodePlot : Added tikz option to environments	32		
\BodeTF : Added Tikz option	27		
\BodeZPK : Added Tikz option	26		
NicholsChart : Added tikz option to environments	47		
\NicholsTF : Added commands and tikz options	47		
\NicholsZPK : Added commands and tikz options	47		
gnuplot@prefix : Added jobname to gnuplot prefix	21		
\NyquistTF : Added commands and tikz options	38		
\NyquistZPK : Added commands and tikz options	37		
\parse@env@opt : Added tikz option to environments	36		
\parse@N@opt : Added commands and tikz options	40		
\parse@opt : Added Tikz option	35		
NyquistPlot : Added tikz option to environments	40		
v1.0.4			
General: Fixed unintended optional argument macro expansion	1		
v1.0.5			
\parse@opt : Fixed a bug	35		
v1.0.6			
General: Fixed issue #3	1		
v1.0.7			
General: Removed unnecessary semicolons	1		
Updated documentation	1		
v1.0.8			
General: Added a new class option ‘declutter’	1		
\build@TF@plot : Included phase due to delay in wrapping.	34		
gnuplot@prefix : Fixed issue #6	21		
v1.1.0			
General: Fixed phase wrapping in gnuplot mode	1		
\addBodeTFPlot : Fixed phase wrapping in gnuplot mode	29		
BodeMagPlot : Added separate environments for phase and magnitude plots	31		
BodePhPlot : Added separate environments for phase and magnitude plots	31		
BodePlot : Deprecated BodePlot environment	32		
\BodeTF : Fixed phase wrapping in gnuplot mode	27		
v1.1.1			
General: Enable Hz and rad units	1		
\addBodeComponentPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	30		
\addBodeTFPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	29		
\addBodeZPKPlots : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	28		
\addNicholsTFChart : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	47		
\addNyquistTFPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	40		
\addNyquistZPKPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	39		
BodeMagPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	31		
BodePhPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	31		
BodePlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	32		
\BodeTF : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27		
\BodeZPK : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	26		
\build@TF@plot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	34		
get@interval@end : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	23		
ph@y@label : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	22		
\NyquistTF : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	38		
\NyquistZPK : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	37		
v1.1.2			
BodeMagPlot : Defined using the ‘NewEnviron’ command from the			

‘environ’ package to fix conflicts with externalization	31	BodePlot: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	32
BodePhPlot: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	31	\BodeTF: Added code to handle active characters	28
BodePlot: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	32	\BodeZPK: Added code to handle active characters	27
NicholsChart: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	47	NicholsChart: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	47
\PhSOZerosLin: Fix scaling bug introduced in v1.1.1	25	\NicholsTF: Added code to handle active characters	47
NyquistPlot: Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	40	\NicholsZPK: Added code to handle active characters	47
v1.1.3		\NyquistTF: Added code to handle active characters	39
\addBodeComponentPlot: Changed implementation to respect user-supplied domain	30	\NyquistZPK: Added code to handle active characters	38
\addBodeTFPlot: Changed implementation to respect user-supplied domain	29	NyquistPlot: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	40
\addBodeZPKPlots: Changed implementation to respect user-supplied domain	28	v1.1.6	
\addNicholsTFChart: Changed implementation to respect user-supplied domain	47	\shorthand@list: Detect ‘babel-french’ using ‘frenchbsetup’	21
\addNicholsZPKChart: Changed implementation to respect user-supplied domain	47	v1.1.7	
\addNyquistTFPlot: Changed implementation to respect user-supplied domain	40	General: Detect and turn off shorthands to improve ‘babel’ compatibility	1
\addNyquistZPKPlot: Changed implementation to respect user-supplied domain	39	BodeMagPlot: Use auto-generated list of active characters instead of manually entering them.	31
v1.1.4		BodePhPlot: Use auto-generated list of active characters instead of manually entering them.	31
\addBodeTFPlot: Changed phase wrapping in pgf mode	29	BodePlot: Use auto-generated list of active characters instead of manually entering them	32
\addNicholsTFChart: Changed phase wrapping in pgf mode	47	\BodeTF: Use auto-generated list of shorthands instead of manually specifying them	28
\BodeTF: Changed phase wrapping in pgf mode	27	\BodeZPK: Use auto-generated list of shorthands instead of manually specifying them	27
gnuplot@prefix: Changed phase wrapping in pgf mode	21	NicholsChart: Use auto-generated list of active characters instead of manually entering them.	47
v1.1.5		\NicholsTF: Use auto-generated list of shorthands instead of manually specifying them	47
BodeMagPlot: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	31	\NicholsZPK: Use auto-generated list of shorthands instead of manually specifying them	47
BodePhPlot: Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	31	\NyquistTF: Use auto-generated list of shorthands instead of manually specifying them	39
		\NyquistZPK: Use auto-generated list of shorthands instead of manually	

specifying them	38	functions to a single Bode plot	1
\shorthand@list : Directly detect		Added unified Bode plot macro	
shorthands instead of detecting		supporting both ZPK and TF	
the language.	21	systems	14
NyquistPlot : Use auto-generated list		Enhanced BodePlot environment	
of active characters instead of		with unified plot command	
manually entering them.	40	collection	14
v1.2		\addBodePlot : Added unified Bode	
General: Removed global option to		plot macro supporting both ZPK	
process pgf commands in radians	1	and TF systems	30
gnuplot@prefix : Removed global		BodePlot : Enhanced BodePlot	
option to process pgf commands		environment with unified plot	
in radians	21	command collection	32
v2.0		\build@gnu@plot : Added general	
General: Added pole zero maps	1	gnuplot helper macro	37
Added pole-zero map functionality	20	\build@gnu@unwrap@plot : Added	
\PoleZeroMapZPK : Added pole-zero		gnuplot helper macro with phase	
map functionality	50	unwrapping	37
v2.1		\parse@add@Bode@opt : Added	
General: Added an environment and		option parser for unified Bode plot	
macros to add multiple transfer		macro	37