

The `bodeplot` package

version 1.1.7

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

February 6, 2024

Contents

1	Introduction	2
1.1	External Dependencies	2
1.2	Directory Structure	2
1.3	Limitations	2
2	TL;DR	3
3	Usage	8
3.1	Bode plots	8
3.1.1	Basic components up to first order	12
3.1.2	Basic components of the second order	13
3.2	Nyquist plots	14
3.3	Nichols charts	16
4	Implementation	18
4.1	Initialization	18
4.2	Parametric function generators for poles, zeros, gains, and delays.	20
4.3	Second order systems.	21
4.4	Commands for Bode plots	23
4.4.1	User macros	23
4.4.2	Internal macros	28
4.5	Nyquist plots	33
4.5.1	User macros	33
4.5.2	Internal commands	36
4.6	Nichols charts	36

1 Introduction

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

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

and the zero-pole-gain (ZPK) form

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

In the equations above, b_m, \dots, b_0 and a_n, \dots, a_0 are real coefficients, $T \geq 0$ is the loop delay, z_1, \dots, z_m and p_1, \dots, p_n are complex zeros and poles of the transfer function, respectively, and $K \in \mathfrak{R}$ is the loop gain.

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

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

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

1.1 External Dependencies

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

1.2 Directory Structure

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

1.3 Limitations

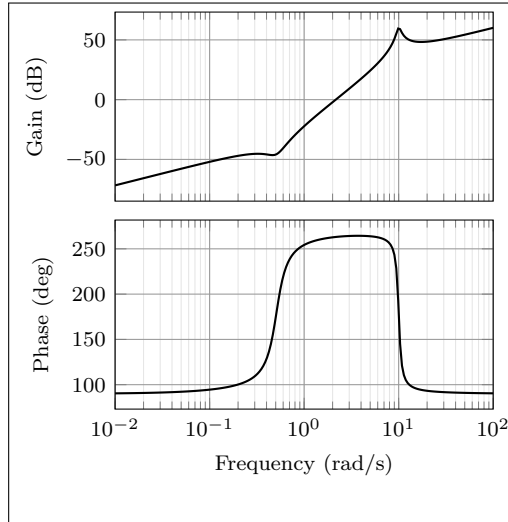
- In `pgf` mode, Bode phase plots and Nichols charts in TF form wrap angles so that they are always between -180 and 180° or $-\pi$ and π radian. As such, these plots will show phase wrapping discontinuities. Since v1.1.1, in `gnuplot` mode, the package uses the `smooth unwrap` filter to correct wrapping 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)

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

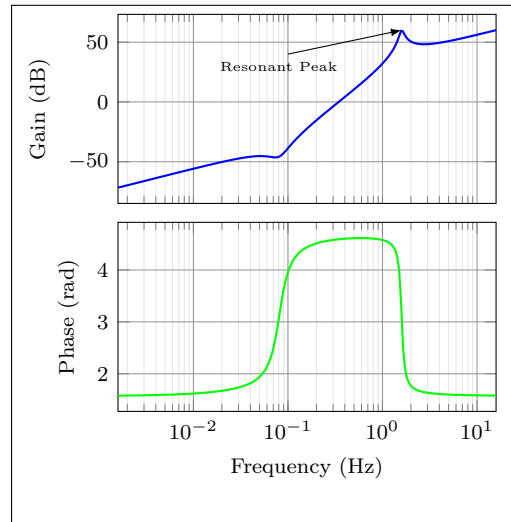
Bode plot in ZPK format



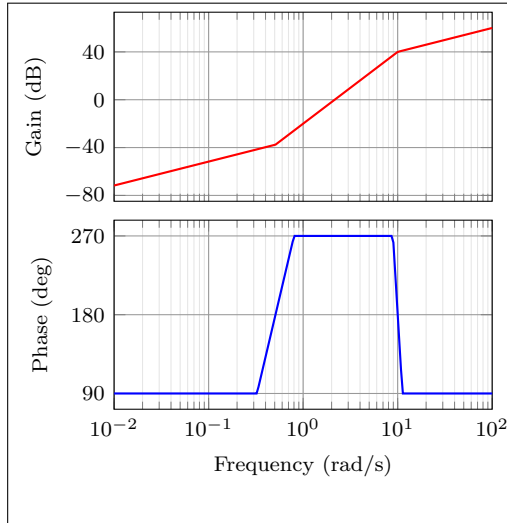
```
\BodeZPK{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

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)}
```



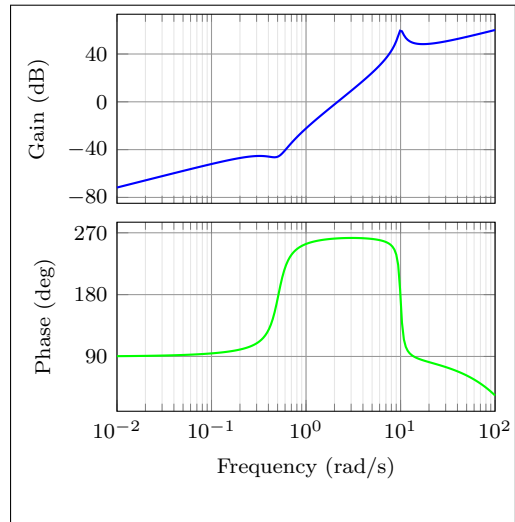
Linear approximation with customization



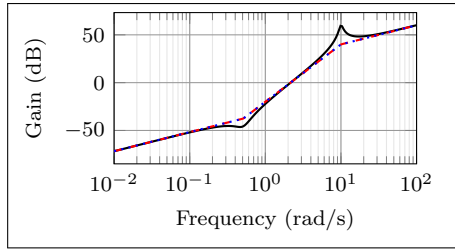
```
\BodeZPK[%
plot/mag/{red,thick},
plot/ph/{blue,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90},
approx/linear%
]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

Plot with delay and customization

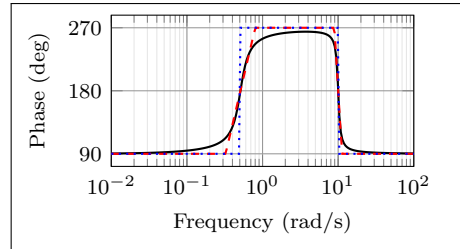
```
\BodeZPK[%
plot/mag/{blue,thick},
plot/ph/{green,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90%
}]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10,
d/0.01%
}
{0.01}
{100}
```



Individual gain and phase plots with more customization

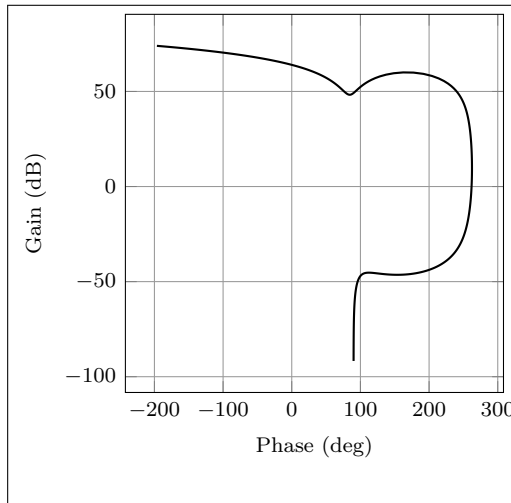


```
\begin{BodeMagPlot}[%
  axes/{height=2cm,
  width=4cm}
]
{0.01}
{100}
\addBodeZPKPlots[%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
]
{magnitude}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\end{BodeMagPlot}
```



```
\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}
```

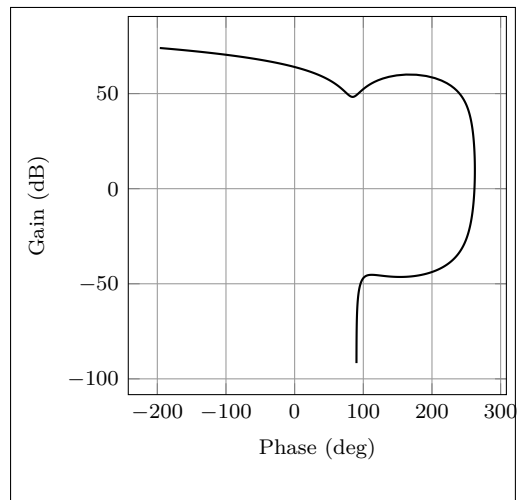
Nichols chart



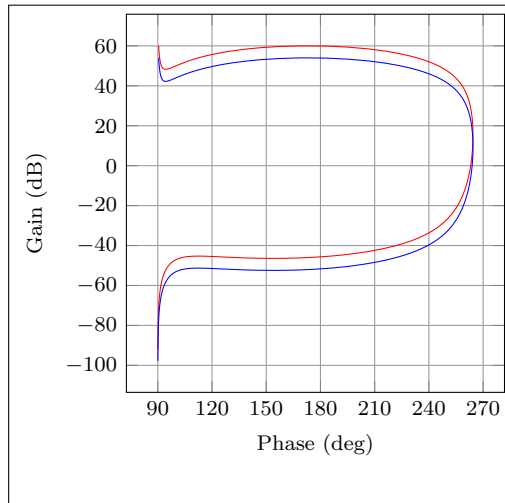
```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01%
}
{0.001}
{500}
```

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

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



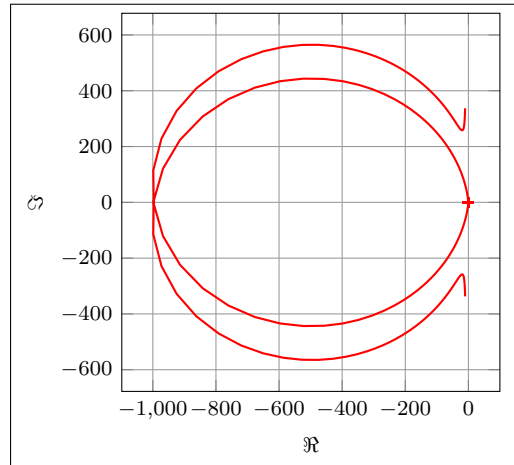
Multiple Nichols charts with customization



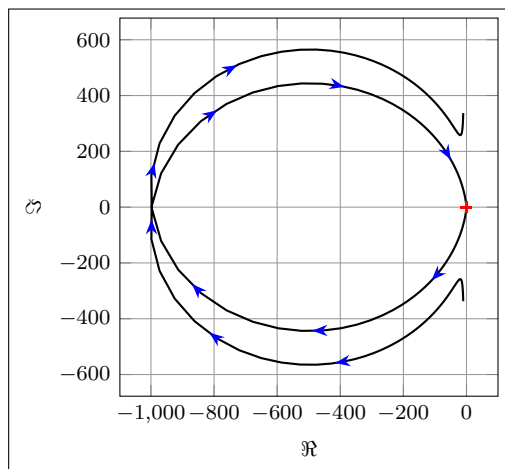
```
\begin{NicholsChart}[%
ytick distance=20,
xtick distance=30
]
{0.001}
{100}
\addNicholsZPKChart [red,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
\addNicholsZPKChart [blue,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/5%
}
\end{NicholsChart}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

Nyquist plot

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



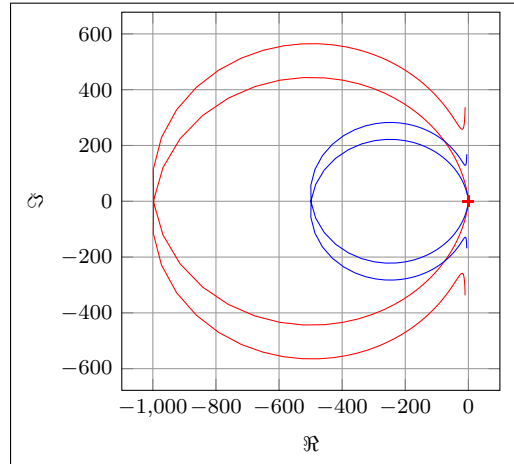
Nyquist plot in TF format with arrows



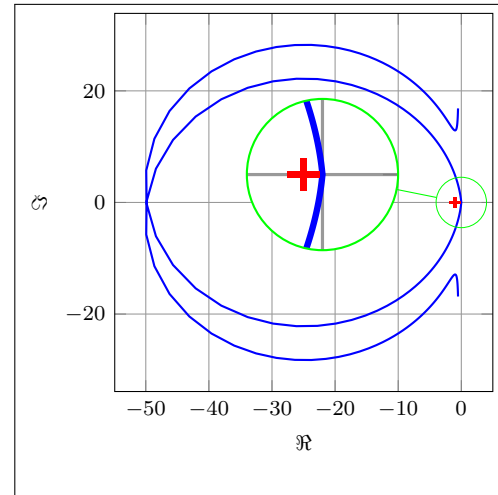
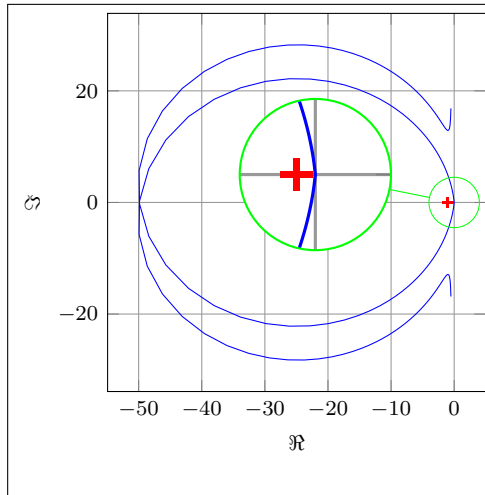
```
\NyquistTF[%
plot/{%
samples=1000,
postaction=decorate,
decoration={%
markings,
mark=between positions 0.1 and 0.9 step 5em with {%
\arrow{Stealth [length=2mm, blue]}
}
}%
}
]
{%
num/{10,2,2.6,0},
den/{1,1,100.25}%
}
{-30}
{30}
```

Multiple Nyquist plots with customization

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



Nyquist plots with additional commands, using two different macros



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

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

3 Usage

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

3.1 Bode plots

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

Plots the Bode plot of a transfer function given in ZPK format using the `groupplot` environment. The three mandatory arguments include: (1) a list of tuples, comprised of the zeros, the poles, the gain, and the transport delay of the transfer function, (2) the lower end of the frequency range for the x -axis, and (3) the higher end of the frequency range for the x -axis. The zeros and the poles are complex numbers, entered as a comma-separated list of comma-separated lists, of the form `{real part 1,imaginary part 1}, {real part 2,imaginary part 2},...`. If the imaginary part is not provided, it is assumed to be zero.

The optional argument is comprised of a comma separated list of tuples, either `obj/typ/{opt}`, or `obj/{opt}`, or just `{opt}`. Each tuple passes options to different `pgfplots` macros that generate the group, the axes, and the plots according to:

- Tuples of the form `obj/typ/{opt}`:
 - `plot/typ/{opt}`: modify plot properties by adding options `{opt}` to the `\addplot` macro for the magnitude plot if `typ` is `mag` and the phase plot if `typ` is `ph`.
 - `axes/typ/{opt}`: modify axis properties by adding options `{opt}` to the `\nextgroupplot` macro for the magnitude plot if `typ` is `mag` and the phase plot if `typ` is `ph`.
 - `commands/typ/{opt}`: add any valid TikZ commands (including the the parametric function generator macros in this package, such as `\addBodeZPKPlots`, `\addBodeTFPlot`, and `\addBodeComponentPlot`) to the magnitude plot if `typ` is `mag` and the phase plot if `typ` is `ph`. The commands passed to `opt` need to be valid TikZ commands, separated by semicolons as usual. For example, a TikZ command is used in the description of the `\BodeTF` macro below to mark the gain crossover frequency on the Bode Magnitude plot.
- Tuples of the form `obj/{opt}`:
 - `plot/{opt}`: adds options `{opt}` to `\addplot` macros for both the magnitude and the phase plots.
 - `axes/{opt}`: adds options `{opt}` to `\nextgroupplot` macros for both the magnitude and the phase plots.
 - `group/{opt}`: adds options `{opt}` to the `groupplot` environment.
 - `tikz/{opt}`: adds options `{opt}` to the `tikzpicture` environment.
 - `approx/linear`: plots linear approximation.
 - `approx/asymptotic`: plots asymptotic approximation.
- Tuples of the form `{opt}` add all of the supplied options to `\addplot` macros for both the magnitude and the phase plots.

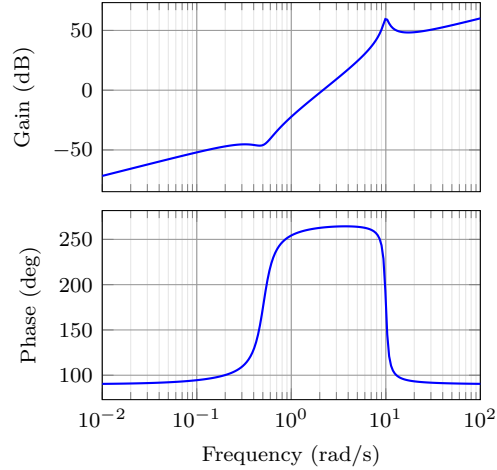


Figure 1: Output of the `\BodeZPK` macro.

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

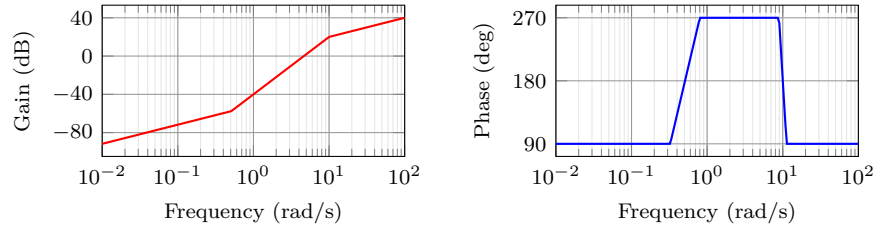


Figure 2: Customization of the default `\BodeZPK` macro.

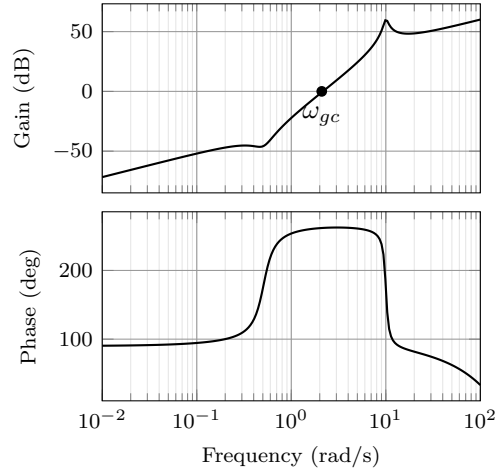


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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

The `BodeMagPlot` environment works in conjunction with the parametric function generator macros `\addBodeZPKPlots`, `\addBodeTFPlot`, and `\addBodeComponentPlot`, intended to be used for magnitude plots. The optional argument is comprised of a comma separated list of tuples, either `obj/opt` or just `opt`. Each tuple passes options to different `pgfplots` macros that generate the axes and the plots according to:

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

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

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

```
\addBodeZPKPlots  \addBodeZPKPlots [\langle approx1/\langle opt1 \rangle \rangle, \langle approx2/\langle opt2 \rangle \rangle, \dots]
                  {\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}
```

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

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

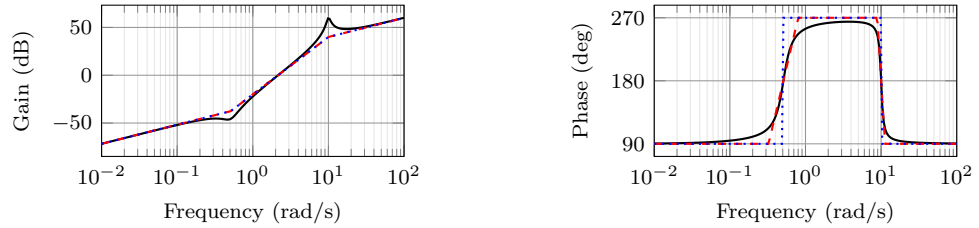


Figure 4: Superimposed approximate and true Bode plots using the `BodeMagPlot` and `BodePhPlot` environments and the `\addBodeZPKPlots` macro.

`\addplot` macro. This macro can be used inside any `semilogaxis` environment as long as a domain for the x-axis is supplied through either the `plot-options` interface or directly in the optional argument of the container `semilogaxis` environment. Use with the `BodePlot` environment supplied with this package is recommended. The second mandatory argument, `plot-type` is either `magnitude` or `phase`. If it is not equal to `phase`, it is assumed to be `magnitude`. The last mandatory argument is the same as `\BodeTF`.

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

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

3.1.1 Basic components up to first order

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

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

- `\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} +
```

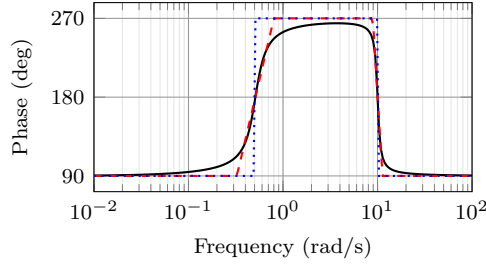


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePh-Plot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

```

\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>}`

This entry describes 2 different macros of the form `\MagS0FeaturePeak` that take the the coefficients a_1 and a_0 of a general second order system as inputs, and draw a resonant peak using the `\draw` TikZ macro. The **Feature** in the macro name should be replaced by either **Poles** or **Zeros** to generate a peak for poles and a valley for zeros, respectively. For example, the command

```

\begin{BodeMagPlot}[xlabel={}]{0.1}{10}
\addBodeComponentPlot[red,dashed,thick]{\MagS0Poles{0.2}{1}}
\addBodeComponentPlot[black,thick]{\MagS0PolesLin{0.2}{1}}
\MagS0PolesPeak[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

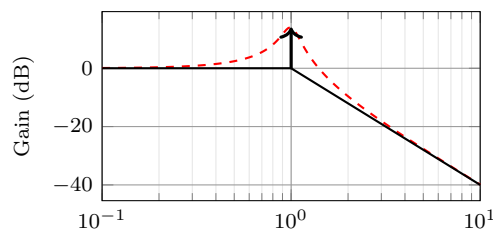


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

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*] $\{\text{real-part}\}\{\text{imaginary-part}\}$

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}*] $\{z/\{zeros\}, p/\{poles\}, k/\{gain\}, d/\{delay\}\}$ $\{\text{min-freq}\}\{\text{max-freq}\}$

Plots the Nyquist plot of a transfer function given in ZPK format with a thick red + marking the critical point (-1,0). The mandatory arguments are the same as `\BodeZPK`. Since there is only one plot in a Nyquist diagram, the `\typ` specifier in the optional argument tuples is not needed. As such, the supported optional argument tuples are `plot/{opt}`, which passes `{opt}` to `\addplot`, `axes/{opt}`, which passes `{\opt}` to the `axis` environment, and `tikz/{opt}`, which passes `{\opt}` to the `tikzpicture` environment. Asymptotic/linear approximations are not supported in Nyquist plots. If just `{opt}` is provided as the optional argument, it is interpreted as `plot/{opt}`. Arrows to indicate the direction of increasing 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}*] $\{\text{num}/\{\text{coeffs}\}, \text{den}/\{\text{coeffs}\}, d/\{\text{delay}\}\}$ $\{\text{min-freq}\}\{\text{max-freq}\}$

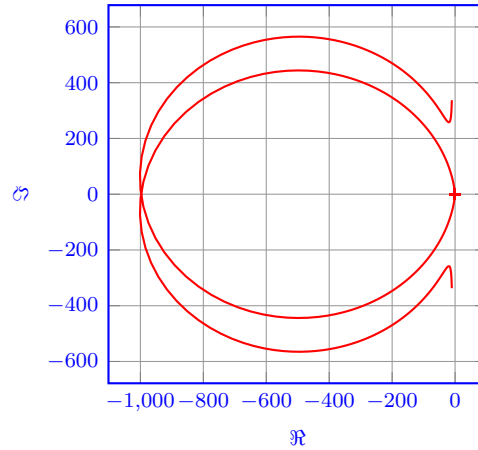


Figure 7: Output of the `\NyquistZPK` macro.

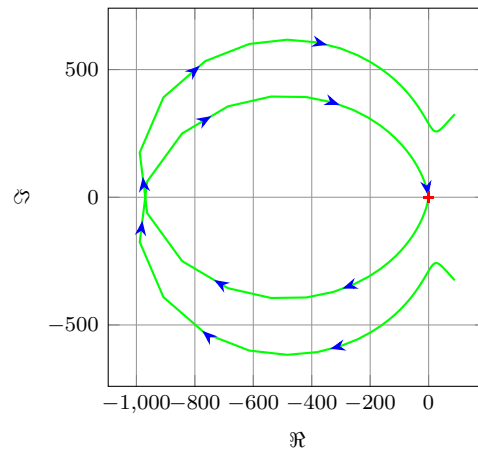


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

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}[\langle obj1/\{\langle opt1\}\rangle,\langle obj2/\{\langle opt2\}\rangle,\dots\rangle]
\addNyquist...
\end{NyquistPlot}
```

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

- Tuples of the form `obj/{opt}`:

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

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

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

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

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

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

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

3.3 Nichols charts

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

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

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

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

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

generates the Nichols chart in Figure 9.

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

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

- Tuples of the form `obj/{opt}`:
 - `tikz/{opt}`: modify picture properties by adding options `{opt}` to the `tikzpicture` environment.
 - `axes/{opt}`: modify axis properties by adding options `{opt}` to the `axis` environment.

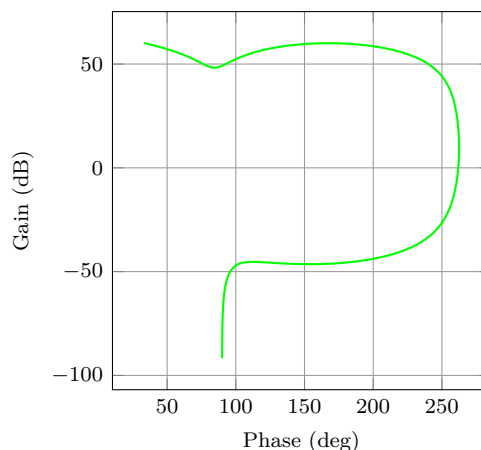


Figure 9: Output of the `\NyquistZPK` macro.

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

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

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

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

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

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

4 Implementation

4.1 Initialization

```
\n@mod 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 \pgfplotsset{
24 trig format plots=rad
25 }
26 \else
27 \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.

```
28 \newcounter{gnuplot@id}
29 \setcounter{gnuplot@id}{0}
30 \if@declutterarg
31 \edef\bodeplot@prefix{gnuplot/\jobname}
32 \else
33 \edef\bodeplot@prefix{\jobname}
34 \fi
35 \tikzset{
36 gnuplot@prefix/.style={
37 id=\arabic{gnuplot@id},
38 prefix=\bodeplot@prefix
39 }
40 }
```

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.

```
41 \ifwindows\else
42 \if@declutterarg
43 \immediate\write18{mkdir -p gnuplot}
44 \fi
45 \fi
46 \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](#).

```

47 \newif\if@babel\@babelfalse
48 \AtBeginDocument{%
49   \ifpackageloaded{babel}{%
50     \@babeltrue
51     \let\shorthand@list\@empty
52     \def\do#1{%
53       \begingroup
54         \lccode'\~='#1\relax
55         \lowercase{\ifbabelshorthand~{\g@addto@macro\shorthand@list{~}}{}}
56       \endgroup
57     }
58     \dospecials
59   }{}
60 }

```

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

```

61 \pgfplotsset{
62   bode@style/.style = {
63     label style={font=\footnotesize},
64     tick label style={font=\footnotesize},
65     grid=both,
66     major grid style={color=gray!80},
67     minor grid style={color=gray!20},
68     x label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
69     y label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
70     scale only axis,
71     samples=200,
72     width=5cm,
73     log basis x=10
74   }
75 }

```

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

```

```

101   \pgfplotsset{ph@y@label/.style = {ylabel={Phase (deg)}}}
102 },
103 phase unit/rad/.code={
104   \renewcommand{\ph@scale}{1}
105   \pgfplotsset{ph@y@label/.style = {ylabel={Phase (rad)}}}
106   \pgfplotsset{ph@x@label/.style = {xlabel={Phase (rad)}}}
107 },
108 frequency unit/.initial={rad},
109 frequency unit/.default={rad},
110 frequency unit/.is choice,
111 frequency unit/Hz/.code={
112   \renewcommand{\freq@scale}{2*pi}
113   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}
114   \ifpgfarg
115     \pgfplotsset{freq@filter/.style = {x filter/.expression={x-
log10(2*pi)}}}
116   \fi
117 },
118 frequency unit/rad/.code={
119   \renewcommand{\freq@scale}{1}
120   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (rad/s)}}}
121 }
122 }

```

`get@interval@start` Internal macros to extract start and end frequency limits from domain specifications.
`get@interval@end` 123 `\def\get@interval@start#1:#2\@nil{#1}`
124 `\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 125 \newcommand*\MagK[2]{(20*log10(abs(#1)))}
\PhKLin   126 \newcommand*\MagKAsymp{\MagK}
           127 \newcommand*\MagKLin{\MagK}
           128 \newcommand*\PhK[2]{((#1<0?-pi:0)*\ph@scale)}
           129 \newcommand*\PhKAsymp{\PhK}
           130 \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.

```

131 \newcommand*\MagDel[2]{0}
132 \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 133 \newcommand*\MagPole[2]
\PhPole     134 {(-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 135 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
136 -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :
137 -20*log10(t)
138 )}

```

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

```
139 \newcommand*\MagPoleAsymp\{MagPoleLin}
```

Parametric function for the true phase of a complex pole.

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

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

```
144 \newcommand*\PhPoleLin[2]{
145 ((abs(#1)+abs(#2) == 0 ? -pi/2 :
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 (-atan2(- (#2), -(#1))) :
149 (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
150 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))))) ?
151 (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) :
152 (-atan2(- (#2), -(#1)) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
153 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} +
154 \n@pow{#2}{2})))))))*(#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) + atan2(-
155 (#2), -(#1)))/
156 (log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
157 (\n@pow{#1}{2} + \n@pow{#2}{2})))))))*\ph@scale)}
```

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

```
157 \newcommand*\PhPoleAsymp[2]{((t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}))) ?
158 (-atan2(- (#2), -(#1))) :
159 (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2))*\ph@scale)}
```

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

```
\MagZeroAsymp
\MagZeroLin 160 \newcommand*\MagZero{0-\MagPole}
\PhZero     161 \newcommand*\MagZeroLin{0-\MagPoleLin}
\PhZeroAsymp 162 \newcommand*\MagZeroAsymp{0-\MagPoleAsymp}
\PhZeroLin  163 \newcommand*\PhZero{0-\PhPole}
             164 \newcommand*\PhZeroLin{0-\PhPoleLin}
             165 \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
\MagCSPolesLin 166 \newcommand*\MagCSPoles[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles     167 - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2}))))}
\PhCSPolesAsymp 168 \newcommand*\MagCSPolesLin[2]{(t < #2 ? -40*log10(#2) : -
\MagCSZeros   169 \newcommand*\MagCSPolesAsymp\{MagCSPolesLin}
```

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

```
\PhCSZeros     170 \newcommand*\PhCSPoles[2]{((-atan2((2*(#1)*(#2)*t), (\n@pow{#2}{2}
\PhCSZerosAsymp 171 - \n@pow{t}{2}))))*\ph@scale)}
\PhCSZerosLin  172 \newcommand*\PhCSPolesLin[2]{((t < (#2 / (\n@pow{10}{abs(#1)})) ?
173 0 :
174 (t >= (#2 * (\n@pow{10}{abs(#1)})) ?
175 (#1>0 ? -pi : pi) :
176 (#1>0 ? (-pi*(log10(t*(\n@pow{10}{#1})/#2))/ (2*#1)) :
177 (pi*(log10(t*(\n@pow{10}{abs(#1)})/#2)) / (2*abs(#1)))))*\ph@scale)}
178 \newcommand*\PhCSPolesAsymp[2]{((#1>0?(t<#2?0:-
pi):(t<#2?0:pi))*\ph@scale)}
```

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

```

179 \newcommand*\MagCSZeros{\theta-\MagCSPoles}
180 \newcommand*\MagCSZerosLin{\theta-\MagCSPolesLin}
181 \newcommand*\MagCSZerosAsymp{\theta-\MagCSPolesAsymp}
182 \newcommand*\PhCSZeros{\theta-\PhCSPoles}
183 \newcommand*\PhCSZerosLin{\theta-\PhCSPolesLin}
184 \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.

```

185 \newcommand*\MagCSPolesPeak}[3][]{
186   \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
187   (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
188 }
189 \newcommand*\MagCSZerosPeak}[3][]{
190   \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
191   (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
192 }

```

`\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 193 \newcommand*\MagS0Poles}[2]{
\PhS0Poles     194   (-20*log10(sqrt(\n@pow{#2} - \n@pow{t}{2}){2} + \n@pow{#1*t}{2})))}
\PhS0PolesAsymp 195 \newcommand*\MagS0PolesLin}[2]{
\PhS0PolesLin  196   (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t))}
\MagS0Zeros    197 \newcommand*\MagS0PolesAsymp{\MagS0PolesLin}

```

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

```

\PhS0Zeros     198 \newcommand*\PhS0Poles}[2]{((-atan2((#1)*t,((#2) -
\PhS0ZerosAsymp \n@pow{t}{2}))) * \ph@scale)}
\PhS0ZerosLin  199 \newcommand*\PhS0PolesLin}[2]{((#2>0 ?
200   \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
201   (#1>0 ? -pi : pi))}
202 \newcommand*\PhS0PolesAsymp}[2]{((#2>0 ?
203   \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
204   (#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).

```

205 \newcommand*\MagS0Zeros{\theta-\MagS0Poles}
206 \newcommand*\MagS0ZerosLin{\theta-\MagS0PolesLin}
207 \newcommand*\MagS0ZerosAsymp{\theta-\MagS0PolesAsymp}
208 \newcommand*\PhS0Zeros{\theta-\PhS0Poles}
209 \newcommand*\PhS0ZerosLin{\theta-\PhS0PolesLin}
210 \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.

```

211 \newcommand*\MagS0PolesPeak}[3][]{
212   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
213   (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
214     20*log10(abs(#2/sqrt(abs(#3)))});
215 }
216 \newcommand*\MagS0ZerosPeak}[3][]{
217   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
218   (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
219     20*log10(abs(#2/sqrt(abs(#3)))});
220 }

```

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`.

```
221 \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.

```
222 \parse@opt{#1}
223 \gdef\func@mag{}
224 \gdef\func@ph{}
225 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
226 \temp@cmd
227 \build@ZPK@plot{\func@mag}{\func@ph}{\opt@approx}{#2}
228 \edef\temp@cmd{\noexpand\begin{groupplot}[
229     bode@style,
230     xmin=#3,
231     xmax=#4,
232     domain=#3*\freq@scale:#4*\freq@scale,
233     height=2.5cm,
234     xmode=log,
235     group style = {group size = 1 by 2,vertical sep=0.25cm},
236     \opt@group
237 ]}
238 \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.

```
239 \edef\temp@mag@cmd{\noexpand\nextgroupplot [yla-
bel={Gain (dB)}, xmajorticks=false, \optmag@axes]
240 \noexpand\addplot [freq@filter, variable=t, thick, \opt-
mag@plot]}
241 \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
242 \noexpand\addplot [freq@filter, variable=t, thick, \optph@plot]}
243 \if@pgfarg
244 \temp@mag@cmd {\func@mag};
245 \optmag@commands
246 \temp@ph@cmd {\func@ph};
247 \optph@commands
248 \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.

```
249 \stepcounter{gnuplot@id}
250 \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
251 { set table $meta;
252 set dummy t;
253 set logscale x 10;
254 set xrange [#3*\freq@scale:#4*\freq@scale];
```

```

255         set samples \pgfkeysvalueof{/pgfplots/samples};
256         plot \func@mag;
257         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
258         plot "$meta" using ($1/(\freq@scale)):($2);
259     };
260     \optmag@commands
261     \stepcounter{gnuplot@id}
262     \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
263     { set table $meta;
264       set dummy t;
265       set logscale x 10;
266       set xrange [#3*\freq@scale:#4*\freq@scale];
267       set samples \pgfkeysvalueof{/pgfplots/samples};
268       plot \func@ph;
269       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
270       plot "$meta" using ($1/(\freq@scale)):($2);
271     };
272     \optph@commands
273     \fi
274     \end{groupplot}
275     \end{tikzpicture}
276 }

```

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

```

277 \AtBeginDocument{%
278   \if@babel
279   \let\Orig@BodeZPK\BodeZPK
280   \renewcommand{\BodeZPK}{%
281     \expandafter\shorthandoff\expandafter{\shorthand@list}
282     \BodeZPK@Shorthandoff
283   }
284   \newcommand{\BodeZPK@Shorthandoff}[4][]{%
285     \Orig@BodeZPK[#1]{#2}{#3}{#4}
286     \expandafter\shorthandon\expandafter{\shorthand@list}
287   }
288   \fi
289 }

```

\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.

```

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

```



```

310     \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
311     \noexpand\addplot [freq@filter, variable=t, thick, \optph@plot]}
312     \if@pgfarg
313         \temp@mag@cmd {\func@mag};
314         \optmag@commands
315         \temp@ph@cmd {\n@mod{\func@ph}{2*pi*\ph@scale}};
316         \optph@commands
317     \else
318         \stepcounter{gnuplot@id}
319         \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
320         { set table $meta;
321           set dummy t;
322           set logscale x 10;
323           set xrange [#3*\freq@scale:#4*\freq@scale];
324           set samples \pgfkeysvalueof{/pgfplots/samples};
325           plot \func@mag;
326           set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
327           plot "$meta" using ($1/(\freq@scale)):($2);
328         };
329         \optmag@commands
330         \stepcounter{gnuplot@id}
331         \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
332         { set table $meta;
333           set dummy t;
334           set logscale x 10;
335           set trange [#3*\freq@scale:#4*\freq@scale];
336           set samples \pgfkeysvalueof{/pgfplots/samples};
337           plot '+' using (t) : ((\func@ph)/(\ph@scale)) smooth unwrap;
338           set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
339           plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
340         };
341         \optph@commands
342     \fi
343 \end{groupplot}
344 \end{tikzpicture}
345 }

```

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

```

346 \AtBeginDocument{
347     \if@babel
348     \let\Orig@BodeTF\BodeTF
349     \renewcommand{\BodeTF}{%
350         \expandafter\shorthandoff\expandafter{\shorthand@list}
351         \BodeTF@Shorthandoff
352     }
353     \newcommand{\BodeTF@Shorthandoff}[4][[]]{%
354         \Orig@BodeTF[#1]{#2}{#3}{#4}
355         \expandafter\shorthandon\expandafter{\shorthand@list}
356     }
357     \fi
358 }

```

\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.

```

359 \newcommand{\addBodeZPKPlots}[3][true/{}]{
360     \foreach \approx/\opt in {#1} {
361         \gdef\plot@macro{
362             \gdef\temp@macro{
363                 \ifnum\pdf@strcmp{#2}{phase}=0
364                 \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}
365                 \else
366                 \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}
367             \fi
368         \if@pgfarg

```

```

369     \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, thick, \opt]}
370     \temp@cmd {\plot@macro};
371     \else
372     \stepcounter{gnuplot@id}
373     \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \opt]}
374     \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
375     { set table $meta;
376       set dummy t;
377       set logscale x 10;
378       set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
379       set samples \pgfkeysvalueof{/pgfplots/samples};
380       plot \plot@macro;
381       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
382       plot "$meta" using ($1/(\freq@scale)):($2);
383     };
384     \fi
385   }
386 }

```

`\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.

```

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

```

```

422         set samples \pgfkeysvalueof{/pgfplots/samples};
423         plot \plot@macro;
424         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
425         plot "$meta" using ($1/(\freq@scale)):($2);
426     };
427     \fi
428 \fi
429 }

```

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

```

430 \newcommand{\addBodeComponentPlot}[2][thick]{
431     \if@pgfarg
432         \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, #1]}
433         \temp@cmd {#2};
434     \else
435         \stepcounter{gnuplot@id}
436         \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
437         { set table $meta;
438           set dummy t;
439           set logscale x 10;
440           set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
441           set samples \pgfkeysvalueof{/pgfplots/samples};
442           plot #2;
443           set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
444           plot "$meta" using ($1/(\freq@scale)):($2);
445         };
446     \fi
447 }

```

`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`.

```

448 \AtBeginDocument{%
449     \if@babel
450         \AddToHook{env/BodePhPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand
451         \AddToHook{env/BodePhPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@L
452     \fi
453 }
454 \NewDocumentEnvironment{BodePhPlot}{0}{mm+b}{
455     \parse@env@opt{#1}
456     \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
457     \temp@cmd
458     \edef\temp@cmd{\noexpand\begin{semilogaxis}[
459         ph@y@label,
460         freq@label,
461         bode@style,
462         xmin={#2},
463         xmax={#3},
464         domain=#2:#3,
465         height=2.5cm,
466         \unexpanded\expandafter{\opt@axes}
467     ]}
468     \temp@cmd
469     #4
470     \end{semilogaxis}
471 \end{tikzpicture}
472 }{}

```

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.

```

473 \AtBeginDocument{%
474   \if@babel
475     \AddToHook{env/BodeMagPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand@
476     \AddToHook{env/BodeMagPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@
477   \fi
478 }
479 \NewDocumentEnvironment{BodeMagPlot}{0}{mm+b}{
480   \parse@env@opt{#1}
481   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
482     \temp@cmd
483     \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
484       bode@style,
485       freq@label,
486       xmin={#2},
487       xmax={#3},
488       domain=#2:#3,
489       height=2.5cm,
490       ylabel={Gain (dB)},
491       \unexpanded\expandafter{\opt@axes}
492     ]}
493     \temp@cmd
494     #4
495     \end{semilogxaxis}
496   \end{tikzpicture}
497 }{}

```

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

```

498 \AtBeginDocument{%
499   \if@babel
500     \AddToHook{env/BodePlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand@
501     \AddToHook{env/BodePlot/end}{\expandafter\shorthandon\expandafter{\shorthand@lis
502   \fi
503 }
504 \NewDocumentEnvironment{BodePlot}{0}{mm+b}{
505   \parse@env@opt{#1}
506   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
507     \temp@cmd
508     \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
509       bode@style,
510       freq@label,
511       xmin={#2},
512       xmax={#3},
513       domain=#2:#3,
514       height=2.5cm,
515       \unexpanded\expandafter{\opt@axes}
516     ]}
517     \temp@cmd
518     #4
519     \end{semilogxaxis}
520   \end{tikzpicture}
521 }{}

```

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](#).

```

522 \newcommand*\add@feature}[3]{
523   \ifcat$\detokenize\expandafter{#1}$
524     \xdef#1{\unexpanded\expandafter{#1 0+#2}}
525   \else
526     \xdef#1{\unexpanded\expandafter{#1+#2}}
527   \fi
528   \foreach \y [count=\n] in #3 {
529     \xdef#1{\unexpanded\expandafter{#1}{\y}}
530     \xdef\Last@LoopValue{\n}
531   }
532   \ifnum\Last@LoopValue=1
533     \xdef#1{\unexpanded\expandafter{#1}{0}}
534   \fi
535 }

```

`\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.

```

536 \newcommand{\build@ZPK@plot}[4]{
537   \foreach \feature/\values in {#4} {
538     \ifnum\pdf@strcmp{\feature}{z}=0
539       \foreach \z in \values {
540         \ifnum\pdf@strcmp{#3}{linear}=0
541           \add@feature{#2}{\PhZeroLin}{\z}
542           \add@feature{#1}{\MagZeroLin}{\z}
543         \else
544           \ifnum\pdf@strcmp{#3}{asymptotic}=0
545             \add@feature{#2}{\PhZeroAsymp}{\z}
546             \add@feature{#1}{\MagZeroAsymp}{\z}
547           \else
548             \add@feature{#2}{\PhZero}{\z}
549             \add@feature{#1}{\MagZero}{\z}
550           \fi
551         \fi
552       }
553     \fi
554     \ifnum\pdf@strcmp{\feature}{p}=0
555       \foreach \p in \values {
556         \ifnum\pdf@strcmp{#3}{linear}=0
557           \add@feature{#2}{\PhPoleLin}{\p}
558           \add@feature{#1}{\MagPoleLin}{\p}
559         \else
560           \ifnum\pdf@strcmp{#3}{asymptotic}=0
561             \add@feature{#2}{\PhPoleAsymp}{\p}
562             \add@feature{#1}{\MagPoleAsymp}{\p}
563           \else
564             \add@feature{#2}{\PhPole}{\p}
565             \add@feature{#1}{\MagPole}{\p}
566           \fi
567         \fi
568       }
569     \fi
570     \ifnum\pdf@strcmp{\feature}{k}=0
571       \ifnum\pdf@strcmp{#3}{linear}=0
572         \add@feature{#2}{\PhKLin}{\values}

```

```

573     \add@feature{#1}{\MagKLin}{\values}
574   \else
575     \ifnum\pdf@strcmp{#3}{asymptotic}=0
576       \add@feature{#2}{\PhKAsymp}{\values}
577       \add@feature{#1}{\MagKAsymp}{\values}
578     \else
579       \add@feature{#2}{\PhK}{\values}
580       \add@feature{#1}{\MagK}{\values}
581     \fi
582   \fi
583 \fi
584 \ifnum\pdf@strcmp{\feature}{d}=0
585   \ifnum\pdf@strcmp{#3}{linear}=0
586     \PackageError {bodeplot} {Linear approximation for pure de-
lays is not
587     supported.} {Plot the true Bode plot using 'true' in-
stead of 'linear'.}
588   \else
589     \ifnum\pdf@strcmp{#3}{asymptotic}=0
590       \PackageError {bodeplot} {Asymptotic approxima-
tion for pure delays is not
591       supported.} {Plot the true Bode plot using 'true' in-
stead of 'asymptotic'.}
592     \else
593       \ifdim\values pt < 0pt
594         \PackageError {bodeplot} {Delay needs to be a posi-
tive number.}
595       \fi
596       \add@feature{#2}{\PhDel}{\values}
597       \add@feature{#1}{\MagDel}{\values}
598     \fi
599   \fi
600 \fi
601 }
602 }

```

`\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).

```

603 \newcommand{\build@TF@plot}[3]{
604   \gdef\num@real{0}
605   \gdef\num@im{0}
606   \gdef\den@real{0}
607   \gdef\den@im{0}
608   \gdef\loop@delay{0}
609   \foreach \feature/\values in {#3} {
610     \ifnum\pdf@strcmp{\feature}{num}=0
611       \foreach \numcoeff [count=\numpow] in \values {
612         \xdef\num@degree{\numpow}
613       }
614       \foreach \numcoeff [count=\numpow] in \values {
615         \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}
616         \ifnum\currentdegree = 0
617           \xdef\num@real{\num@real+\numcoeff}
618         \else
619           \ifodd\currentdegree
620             \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-
1}}{(\currentdegree-1)/2})*%
621             (\n@pow{t}{\currentdegree}))}
622           \else
623             \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-
1}}{(\currentdegree)/2})*%

```

```

624         (\n@pow{t}{\currentdegree}))}
625     \fi
626     \fi
627 }
628 \fi
629 \ifnum\pdf@strcmp{\feature}{den}=0
630     \foreach \dencoeff [count=\denpow] in \values {
631         \xdef\den@degree{\denpow}
632     }
633     \foreach \dencoeff [count=\denpow] in \values {
634         \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}
635         \ifnum\currentdegree = 0
636             \xdef\den@real{\den@real+\dencoeff}
637         \else
638             \ifodd\currentdegree
639                 \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-
1}}{(\currentdegree-1)/2}))*%
640                     (\n@pow{t}{\currentdegree}))}
641             \else
642                 \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-
1}}{(\currentdegree)/2}))*%
643                     (\n@pow{t}{\currentdegree}))}
644         \fi
645     \fi
646 }
647 \fi
648 \ifnum\pdf@strcmp{\feature}{d}=0
649     \xdef\loop@delay{\values}
650 \fi
651 }
652 \xdef#2{((atan2((\num@im),(\num@real))-atan2((\den@im),%
653     (\den@real))-\loop@delay*t)*(\ph@scale))}
654 \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2}))) -
%
655     20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2}))))}
656 }

```

`\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.

```

657 \newcommand{\parse@opt}[1]{
658     \gdef\optmag@axes{}
659     \gdef\optph@axes{}
660     \gdef\optph@plot{}
661     \gdef\optmag@plot{}
662     \gdef\opt@group{}
663     \gdef\opt@approx{}
664     \gdef\optph@commands{}
665     \gdef\optmag@commands{}
666     \gdef\opt@tikz{}
667     \foreach \obj/\typ/\opt in {#1} {
668         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
669             \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
670                 \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
671             \else
672                 \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
673                     \xdef\optph@plot{\unexpanded\expandafter{\opt}}
674                 \else

```

```

675         \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
676         \xdef\optph@plot{\unexpanded\expandafter{\opt}}
677     \fi
678 \fi
679 \else
680 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
681 \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
682 \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
683 \else
684 \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
685 \xdef\optph@axes{\unexpanded\expandafter{\opt}}
686 \else
687 \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
688 \xdef\optph@axes{\unexpanded\expandafter{\opt}}
689 \fi
690 \fi
691 \else
692 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{group}=0
693 \xdef\opt@group{\unexpanded\expandafter{\opt}}
694 \else
695 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{approx}=0
696 \xdef\opt@approx{\unexpanded\expandafter{\opt}}
697 \else
698 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
699 \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
700 \xdef\optph@commands{\unexpanded\expandafter{\opt}}
701 \else
702 \xdef\optmag@commands{\unexpanded\expandafter{\opt}}
703 \fi
704 \else
705 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
706 \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
707 \else
708 \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
709 \unexpanded\expandafter{\obj}}
710 \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
711 \unexpanded\expandafter{\obj}}
712 \fi
713 \fi
714 \fi
715 \fi
716 \fi
717 \fi
718 }
719 }

```

`\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.

```

720 \newcommand{\parse@env@opt}[1]{
721 \gdef\opt@axes{}
722 \gdef\opt@tikz{}
723 \foreach \obj/\opt in {#1} {
724 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
725 \xdef\opt@axes{\unexpanded\expandafter{\opt}}
726 \else
727 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
728 \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
729 \else
730 \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
731 \unexpanded\expandafter{\obj}}

```



```

732     \fi
733     \fi
734 }
735 }

```

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!

```

736 \newcommand{\NyquistZPK}[4][]{
737   \parse@N@opt{#1}
738   \gdef\func@mag{}
739   \gdef\func@ph{}
740   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
       panded\expandafter{\opt@tikz}]}
741   \temp@cmd
742   \build@ZPK@plot{\func@mag}{\func@ph}{{#2}
743     \edef\temp@cmd{\noexpand\begin{axis}[
744       bode@style,
745       domain=#3*\freq@scale:#4*\freq@scale,
746       height=5cm,
747       xlabel={\$Re\$},
748       ylabel={\$Im\$},
749       samples=500,
750       \unexpanded\expandafter{\opt@axes}
751     ]}
752   \temp@cmd
753     \addplot [only marks,mark=+,thick,red] (-1 , 0);
754     \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \unex-
       panded\expandafter{\opt@plot}]}
755     \if@pgfarg
756       \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
757         {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
758     \opt@commands
759     \else
760       \stepcounter{gnuplot@id}
761       \temp@cmd gnuplot [parametric, gnuplot@prefix] {
762         \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
763         \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
764       };
765     \opt@commands
766     \fi
767   \end{axis}
768 \end{tikzpicture}
769 }

```

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

```

770 \AtBeginDocument{%
771   \if@babel
772   \let\Orig@NyquistZPK\NyquistZPK
773   \renewcommand{\NyquistZPK}{%
774     \expandafter\shorthandoff\expandafter{\shorthand@list}
775     \NyquistZPK@Shorthandoff
776   }

```

```

777 \newcommand{\NyquistZPK@Shorthandoff}[4][]{%
778   \Orig@NyquistZPK[#1]{#2}{#3}{#4}
779   \expandafter\shorthandon\expandafter{\shorthand@list}
780 }
781 \fi
782 }

```

`\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`.

```

783 \newcommand{\NyquistTF}[4][]{
784   \parse@N@opt{#1}
785   \gdef\func@mag{}
786   \gdef\func@ph{}
787   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
788   \temp@cmd
789   \build@TF@plot{\func@mag}{\func@ph}{#2}
790   \edef\temp@cmd{\noexpand\begin{axis}[
791     bode@style,
792     domain=#3*\freq@scale:#4*\freq@scale,
793     height=5cm,
794     xlabel={\Re$},
795     ylabel={\Im$},
796     samples=500,
797     \unexpanded\expandafter{\opt@axes}
798   ]}
799   \temp@cmd
800   \addplot [only marks, mark=+, thick, red] (-1 , 0);
801   \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \unex-
panded\expandafter{\opt@plot}]}
802   \if@pgfarg
803     \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
804               {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
805   \opt@commands
806   \else
807     \stepcounter{gnuplot@id}
808     \temp@cmd gnuplot [parametric, gnuplot@prefix] {
809       \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
810       \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
811     };
812   \opt@commands
813   \fi
814   \end{axis}
815 \end{tikzpicture}
816 }

```

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

```

817 \AtBeginDocument{%
818   \if@babel
819   \let\Orig@NyquistTF\NyquistTF
820   \renewcommand{\NyquistTF}{%
821     \expandafter\shorthandoff\expandafter{\shorthand@list}
822     \NyquistTF@Shorthandoff
823   }
824   \newcommand{\NyquistTF@Shorthandoff}[4][]{%
825     \Orig@NyquistTF[#1]{#2}{#3}{#4}
826     \expandafter\shorthandon\expandafter{\shorthand@list}
827   }
828   \fi
829 }

```

`\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.

```

830 \newcommand{\addNyquistZPKPlot}[2][]{
831   \gdef\func@mag{}
832   \gdef\func@ph{}
833   \build@ZPK@plot{\func@mag}{\func@ph}{#2}
834   \ifpgfarg
835     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
836     \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
837     {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
838   \else
839     \stepcounter{gnuplot@id}
840     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
841     \temp@cmd gnuplot [parametric, gnuplot@prefix] {
842       \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
843       \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
844     };
845   \fi
846 }

```

`\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.

```

847 \newcommand{\addNyquistTFPlot}[2][]{
848   \gdef\func@mag{}
849   \gdef\func@ph{}
850   \build@TF@plot{\func@mag}{\func@ph}{#2}
851   \ifpgfarg
852     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
853     \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
854     {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
855   \else
856     \stepcounter{gnuplot@id}
857     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
858     \temp@cmd gnuplot [parametric, gnuplot@prefix]{
859       \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
860       \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
861     };
862   \fi
863 }

```

`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.

```

864 \AtBeginDocument{%
865   \if@babel
866     \AddToHook{env/NyquistPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand
867     \AddToHook{env/NyquistPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@
868   \fi
869 }
870 \NewDocumentEnvironment{NyquistPlot}{0}{mm+b}{
871   \parse@env@opt{#1}
872   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
873   \temp@cmd
874   \edef\temp@cmd{\noexpand\begin{axis}[
875     bode@style,

```

```

876     height=5cm,
877     domain=#2:#3,
878     xlabel={\Re$},
879     ylabel={\Im$},
880     \unexpanded\expandafter{\opt@axes}
881   ]]
882   \temp@cmd
883     \addplot [only marks,mark=+,thick,red] (-1 , 0);
884     #4
885   \end{axis}
886 \end{tikzpicture}
887 }{}

```

4.5.2 Internal commands

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

```

888 \newcommand{\parse@N@opt}[1]{
889   \gdef\opt@axes{}
890   \gdef\opt@plot{}
891   \gdef\opt@commands{}
892   \gdef\opt@tikz{}
893   \foreach \obj/\opt in {#1} {
894     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
895       \xdef\opt@axes{\unexpanded\expandafter{\opt}}
896     \else
897       \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
898         \xdef\opt@plot{\unexpanded\expandafter{\opt}}
899       \else
900         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
901           \xdef\opt@commands{\unexpanded\expandafter{\opt}}
902         \else
903           \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
904             \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
905           \else
906             \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
907               \unexpanded\expandafter{\obj}}
908           \fi
909         \fi
910       \fi
911     \fi
912   }
913 }

```

4.6 Nichols charts

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

```

NicholsChart
\addNicholsZPKChart
\addNicholsTFChart
914 \newcommand{\NicholsZPK}[4][[]]{
915   \parse@N@opt{#1}
916   \gdef\func@mag{}
917   \gdef\func@ph{}
918   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
919     panded\expandafter{\opt@tikz}]}
920   \temp@cmd
921   \build@ZPK@plot{\func@mag}{\func@ph}{}{#2}
922   \edef\temp@cmd{\noexpand\begin{axis}[
923     ph@x@label,
923     bode@style,

```

```

924     domain=#3*\freq@scale:#4*\freq@scale,
925     height=5cm,
926     ylabel={Gain (dB)},
927     samples=500,
928     \unexpanded\expandafter{\opt@axes}
929   ]]
930   \temp@cmd
931     \edef\temp@cmd{\noexpand\addplot [variable=t,thick,\opt@plot]}
932     \ifpgfarg
933       \temp@cmd ( {\func@ph} , {\func@mag} );
934       \opt@commands
935     \else
936       \stepcounter{gnuplot@id}
937       \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
938       { set table $meta;
939         set logscale x 10;
940         set dummy t;
941         set samples \pgfkeysvalueof{/pgfplots/samples};
942         set trange [#3*\freq@scale:#4*\freq@scale];
943         plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
944         unset logscale x;
945         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
946         plot "$meta" using ($2*\ph@scale):($1);
947       };
948       \opt@commands
949     \fi
950   \end{axis}
951 \end{tikzpicture}
952 }
953 \AtBeginDocument{%
954   \if@babel
955   \let\Orig@NicholsZPK\NicholsZPK
956   \renewcommand{\NicholsZPK}{%
957     \expandafter\shorthandoff\expandafter{\shorthand@list}
958     \NicholsZPK@Shorthandoff
959   }
960   \newcommand{\NicholsZPK@Shorthandoff}[4][]{%
961     \Orig@NicholsZPK[#1]{#2}{#3}{#4}
962     \expandafter\shorthandon\expandafter{\shorthand@list}
963   }
964   \fi
965 }
966 \newcommand{\NicholsTF}[4][]{
967   \parse@N@opt{#1}
968   \gdef\func@mag{
969   \gdef\func@ph{
970   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
971     panded\expandafter{\opt@tikz}]}
972   \temp@cmd
973     \build@TF@plot{\func@mag}{\func@ph}{#2}
974     \edef\temp@cmd{\noexpand\begin{axis}[
975       ph@x@label,
976       bode@style,
977       domain=#3*\freq@scale:#4*\freq@scale,
978       height=5cm,
979       ylabel={Gain (dB)},
980       samples=500,
981       \unexpanded\expandafter{\opt@axes}
982     ]}
983     \temp@cmd
984     \edef\temp@cmd{\noexpand\addplot [variable=t,thick, \opt@plot]}
985     \ifpgfarg
986       \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );

```

```

986     \opt@commands
987   \else
988     \stepcounter{gnuplot@id}
989     \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
990     { set table $metal;
991       set logscale x 10;
992       set dummy t;
993       set samples \pgfkeysvalueof{/pgfplots/samples};
994       set trange [#3*\freq@scale:#4*\freq@scale];
995       plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
996       unset logscale x;
997       set table $meta2;
998       plot "$metal" using ($1):($2) smooth unwrap;
999       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1000      plot "$meta2" using ($2*\ph@scale):($1);
1001    };
1002    \opt@commands
1003  \fi
1004  \end{axis}
1005  \end{tikzpicture}
1006 }
1007 \AtBeginDocument{
1008   \if@babel
1009     \let\Orig@NicholsTF\NicholsTF
1010     \renewcommand{\NicholsTF}{%
1011       \expandafter\shorthandoff\expandafter{\shorthand@list}
1012       \NicholsTF@Shorthandoff
1013     }
1014     \newcommand{\NicholsTF@Shorthandoff}[4][[]]{%
1015       \Orig@NicholsTF[#1]{#2}{#3}{#4}
1016       \expandafter\shorthandon\expandafter{\shorthand@list}
1017     }
1018     \AddToHook{env/NicholsChart/begin}{\expandafter\shorthandoff\expandafter{\shorthand@list}}
1019     \AddToHook{env/NicholsChart/end}{\expandafter\shorthandon\expandafter{\shorthand@list}}
1020   \fi
1021 }
1022 \NewDocumentEnvironment{NicholsChart}{0}{mm+b}{
1023   \parse@env@opt{#1}
1024   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
1025   \temp@cmd
1026   \edef\temp@cmd{\noexpand\begin{axis}[
1027     ph@x@label,
1028     bode@style,
1029     domain=#2:#3,
1030     height=5cm,
1031     ylabel={Gain (dB)},
1032     \unexpanded\expandafter{\opt@axes}
1033   ]}
1034   \temp@cmd
1035   #4
1036   \end{axis}
1037   \end{tikzpicture}
1038 }{}
1039 \newcommand{\addNicholsZPKChart}[2][[]]{
1040   \gdef\func@mag{
1041     \gdef\func@ph{
1042       \build@ZPK@plot{\func@mag}{\func@ph}{#2}
1043     \if@pgfarg
1044       \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
1045         able=t, #1]}
1046       \temp@cmd ( {\func@ph} , {\func@mag} );
1047     \else

```

```

1047 \stepcounter{gnuplot@id}
1048 \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1049 { set table $meta;
1050   set logscale x 10;
1051   set dummy t;
1052   set samples \pgfkeysvalueof{/pgfplots/samples};
1053   set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1054   plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1055   unset logscale x;
1056   set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1057   plot "$meta" using ($2*\ph@scale):($1);
1058 };
1059 \fi
1060 }
1061 \newcommand{\addNicholsTFChart}[2][]{
1062 \gdef\func@mag{}
1063 \gdef\func@ph{}
1064 \buildTF@plot{\func@mag}{\func@ph}{#2}
1065 \if@pgfarg
1066 \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
1067 \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
1068 \else
1069 \stepcounter{gnuplot@id}
1070 \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1071 { set table $meta1;
1072   set logscale x 10;
1073   set dummy t;
1074   set samples \pgfkeysvalueof{/pgfplots/samples};
1075   set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1076   plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1077   unset logscale x;
1078   set table $meta2;
1079   plot "$meta1" using ($1):($2) smooth unwrap;
1080   set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1081   plot "$meta2" using ($2*\ph@scale):($1);
1082 };
1083 \fi
1084 }

```

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	A	
<code>\@babelfalse</code>	<i>47</i>	<code>\AddToHook</code>
<code>\@babeltrue</code>	<i>50</i>	<i>475, 476, 500, 501,</i>
<code>\@declutterargfalse</code>	<i>5</i>	<i>866, 867, 1018, 1019</i>
<code>\@declutterargtrue</code>	<i>7</i>	<code>\AtBeginDocument</code>
<code>\@empty</code>	<i>51</i>	<i>48, 277, 346,</i>
<code>\@hzargfalse</code>	<i>13</i>	<i>448, 473, 498, 770,</i>
<code>\@hzargtrue</code>	<i>15</i>	<i>817, 864, 953, 1007</i>
<code>\@ifpackageloaded</code>	<i>49</i>	<code>\addBodeComponentPlot</code>
<code>\@nil</code>	<i>123, 124</i> <i>430</i>
<code>\@pgfargfalse</code>	<i>1</i>	<code>\addBodeTFPlot</code>
<code>\@pgfargtrue</code>	<i>3</i>	<i>387</i>
<code>\@radargfalse</code>	<i>9</i>	<code>\addBodeZPKPlots</code>
<code>\@radargtrue</code>	<i>11</i>	<i>359</i>
<code>\~</code>	<i>54</i>	<code>\addNicholsTFChart</code>
		<i>914</i>
		<code>\addNicholsZPKChart</code>
		<i>914</i>
		<code>\addNyquistTFPlot</code>
		<i>847</i>
		<code>\addNyquistZPKPlot</code>
		<i>830</i>
		B
		<code>\begingroup</code>
		<i>53</i>
		<code>\bode@style</code>
		<i>61</i>
		<code>BodeMagPlot</code> (env.)
		<i>473</i>
		<code>BodePhPlot</code> (env.)
		<i>448</i>
		<code>BodePlot</code> (env.)
		<i>498</i>
		<code>\bodeplot@prefix</code>

..... 31, 33, 38,
 257, 269, 326, 338,
 381, 413, 424, 443,
 945, 999, 1056, 1080
 \BodeTF [290](#)
 \BodeTF@Shorthandoff
 [351](#), [353](#)
 \BodeZPK [221](#)
 \BodeZPK@Shorthandoff
 [282](#), [284](#)
 \build@TF@plot
[296](#), [391](#), [393](#), [603](#),
[789](#), [850](#), [972](#), [1064](#)
 \build@ZPK@plot ...
[227](#), [364](#), [366](#), [536](#),
[742](#), [833](#), [920](#), [1042](#)

C

\currentdegree . [615](#),
[616](#), [619](#), [620](#), [621](#),
[623](#), [624](#), [634](#), [635](#),
[638](#), [639](#), [640](#), [642](#), [643](#)

D

\den@degree [631](#), [634](#)
 \den@im [607](#), [639](#), [652](#), [655](#)
 \den@real [606](#),
[636](#), [642](#), [653](#), [655](#)
 \dencoeff [630](#),
[633](#), [636](#), [639](#), [642](#)
 \denpow [630](#), [631](#), [633](#), [634](#)
 \do [52](#)
 \dospecials [58](#)

E

\endgroup [56](#)
 environments:
 BodeMagPlot [473](#)
 BodePhPlot [448](#)
 BodePlot [498](#)

F

\freq@filter [76](#)
 \freq@label [76](#)
 \freq@scale [76](#),
[77](#), [88](#), [112](#), [119](#),
[232](#), [254](#), [258](#), [266](#),
[270](#), [301](#), [323](#), [327](#),
[335](#), [339](#), [369](#), [378](#),
[382](#), [397](#), [400](#), [410](#),
[414](#), [421](#), [425](#), [432](#),
[440](#), [444](#), [745](#), [792](#),
[835](#), [840](#), [852](#), [857](#),
[924](#), [942](#), [976](#), [994](#),
[1044](#), [1053](#), [1066](#), [1075](#)
 \func@mag .. [223](#), [227](#),
[244](#), [256](#), [292](#), [296](#),
[313](#), [325](#), [738](#), [742](#),
[756](#), [757](#), [762](#), [763](#),
[785](#), [789](#), [803](#), [804](#),
[809](#), [810](#), [831](#), [833](#),
[836](#), [837](#), [842](#), [843](#),
[848](#), [850](#), [853](#), [854](#),
[859](#), [860](#), [916](#), [920](#),
[933](#), [943](#), [968](#), [972](#),
[985](#), [995](#), [1040](#),
[1042](#), [1045](#), [1054](#),
[1062](#), [1064](#), [1067](#), [1076](#)

G

\g@addto@macro [55](#)
 \get@interval@end .
 [123](#), [124](#)
 \get@interval@start
 [123](#), [123](#)
 \gnuplot@id [1](#)
 \gnuplot@prefix [1](#)

I

\if@babel [47](#)
 \if@hzarg [13](#), [87](#)
 \if@babelshorthand .. [55](#)
 \ifcat [523](#)
 \ifnum [363](#), [390](#),
[396](#), [405](#), [532](#), [538](#),
[540](#), [544](#), [554](#), [556](#),
[560](#), [570](#), [571](#), [575](#),
[584](#), [585](#), [589](#), [610](#),
[616](#), [629](#), [635](#), [648](#),
[668](#), [669](#), [672](#), [680](#),
[681](#), [684](#), [692](#), [695](#),
[698](#), [699](#), [705](#), [724](#),
[727](#), [894](#), [897](#), [900](#), [903](#)

J

\ifwindows [41](#)
 \immediate [43](#)

L

\jobname [31](#), [33](#)

L

\Last@LoopValue [530](#), [532](#)
 \lccode [54](#)
 \loop@delay [608](#), [649](#), [653](#)
 \lowercase [55](#)

M

\MagCSPoles [166](#)
 \MagCSPolesAsymp .. [166](#)
 \MagCSPolesLin [166](#)
 \MagCSPolesPeak ... [185](#)
 \MagCSZeros [166](#)
 \MagCSZerosAsymp .. [166](#)
 \MagCSZerosLin [166](#)
 \MagCSZerosPeak ... [185](#)
 \MagDel [131](#), [597](#)

\MagK [125](#), [580](#)
 \MagKAsymp [125](#), [577](#)
 \MagKLin [125](#), [573](#)
 \MagPole ... [133](#), [160](#), [565](#)
 \MagPoleAsymp
 [133](#), [162](#), [562](#)
 \MagPoleLin [133](#), [161](#), [558](#)
 \MagSOPoles [193](#)
 \MagSOPolesAsymp .. [193](#)
 \MagSOPolesLin [193](#)
 \MagSOPolesPeak ... [211](#)
 \MagSOPolesZero [193](#)
 \MagSOPolesZeroAsymp .. [193](#)
 \MagSOPolesZeroLin [193](#)
 \MagSOPolesZeroPeak ... [211](#)
 \MagZero [160](#), [549](#)
 \MagZeroAsymp .. [160](#), [546](#)
 \MagZeroLin [160](#), [542](#)

N

\n [528](#), [530](#)
 \n@mod [1](#), [315](#), [398](#), [985](#), [1067](#)
 \n@mod@n [1](#)
 \n@mod@p [1](#), [141](#)
 \n@pow [1](#),
[134](#), [135](#), [136](#), [146](#),
[147](#), [149](#), [150](#), [152](#),
[153](#), [154](#), [155](#), [156](#),
[157](#), [166](#), [167](#), [170](#),
[171](#), [172](#), [174](#), [176](#),
[177](#), [194](#), [198](#), [620](#),
[621](#), [623](#), [624](#), [639](#),
[640](#), [642](#), [643](#), [654](#),
[655](#), [756](#), [757](#), [762](#),
[763](#), [803](#), [804](#), [809](#),
[810](#), [836](#), [837](#), [842](#),
[843](#), [853](#), [854](#), [859](#), [860](#)

\newcounter [28](#)
 \NewDocumentEnvironment
 [454](#),
[479](#), [504](#), [870](#), [1022](#)
 \NicholsChart [914](#)
 \NicholsTF [914](#)
 \NicholsTF@Shorthandoff
 [1012](#), [1014](#)
 \NicholsZPK [914](#)
 \NicholsZPK@Shorthandoff
 [958](#), [960](#)
 \num@degree [612](#), [615](#)
 \num@im [605](#), [620](#), [652](#), [654](#)
 \num@real [604](#),
[617](#), [623](#), [652](#), [654](#)
 \numcoeff [611](#),
[614](#), [617](#), [620](#), [623](#)
 \numpow [611](#), [612](#), [614](#), [615](#)
 \NyquistPlot [864](#)
 \NyquistTF [783](#)
 \NyquistTF@Shorthandoff
 [822](#), [824](#)
 \NyquistZPK [736](#)
 \NyquistZPK@Shorthandoff
 [775](#), [777](#)

O		
<code>\opt@approx</code>	227, 663, 696	410, 411, 421, 422,
<code>\opt@axes</code>	.. 466, 491,	432, 440, 441, 835,
	515, 721, 725, 730,	840, 852, 857, 941,
	750, 797, 880, 889,	993, 1044, 1052,
	895, 928, 980, 1032	1053, 1066, 1074, 1075
<code>\opt@commands</code>	<code>\pgfplotsset</code>
 758, 765, 23, 61, 76,
	805, 812, 891, 901,	78, 79, 80, 83, 84,
	934, 948, 986, 1002	89, 91, 100, 101,
<code>\opt@group</code>	105, 106, 113, 115, 120
	.. 236, 305, 662, 693	<code>\ph@scale</code>
<code>\opt@plot</code>	.. 754, 801,	. 76, 81, 85,
	890, 898, 906, 931, 983	99, 104, 128, 132,
<code>\opt@tikz</code> 225,	143, 156, 159, 171,
	294, 456, 481, 506,	177, 178, 198, 315,
	666, 706, 722, 728,	337, 339, 412, 414,
	740, 787, 872, 892,	653, 756, 757, 762,
	904, 918, 970, 1024	763, 803, 804, 809,
<code>\optmag@axes</code>	... 239,	810, 836, 837, 842,
	308, 658, 682, 687	843, 853, 854, 859,
<code>\optmag@commands</code>	245,	860, 943, 946, 985,
	260, 314, 329, 665, 702	995, 1000, 1054,
<code>\optmag@plot</code>	... 240,	1057, 1067, 1076, 1081
	309, 661, 670, 675, 708	<code>\ph@x@label</code>
<code>\optph@axes</code>	... 241, 76
	310, 659, 685, 688	<code>\ph@y@label</code>
<code>\optph@commands</code>	247, 76
	272, 316, 341, 664, 700	<code>\PhCSPoles</code>
<code>\optph@plot</code>	... 242, 166
	311, 660, 673, 676, 710	<code>\PhCSPolesAsymp</code>
<code>\Orig@BodeTF</code>	... 348, 354	166, 203
<code>\Orig@BodeZPK</code>	.. 279, 285	<code>\PhCSPolesLin</code>
<code>\Orig@NicholsTF</code> 166, 200
 1009, 1015	<code>\PhCSZeros</code>
<code>\Orig@NicholsZPK</code>	955, 961 166
<code>\Orig@NyquistTF</code>	819, 825	<code>\PhCSZerosAsymp</code>
<code>\Orig@NyquistZPK</code>	772, 778	... 166
		<code>\PhCSZerosLin</code>
	 166
		<code>\PhDel</code>
	 132, 596
		<code>\PhK</code>
	 125, 579
		<code>\PhKAsymp</code>
		.. 125, 131, 576
		<code>\PhKLin</code>
		... 125, 131, 572
		<code>\PhPole</code>
		... 133, 163, 564
		<code>\PhPoleAsymp</code>
		133, 165, 561
		<code>\PhPoleLin</code>
		. 133, 164, 557
		<code>\PhSOPoles</code>
	 193
		<code>\PhSOPolesAsymp</code>
		... 193
		<code>\PhSOPolesLin</code>
	 193
		<code>\PhS0Zeros</code>
	 193
		<code>\PhS0ZerosAsymp</code>
		... 193
		<code>\PhS0ZerosLin</code>
	 193
		<code>\PhZero</code>
	 160, 548
		<code>\PhZeroAsymp</code>
		... 160, 545
		<code>\PhZeroLin</code>
	 160, 541
		<code>\plot@macro</code>
	
		.. 361, 364, 366,
		370, 380, 388, 391,
		393, 398, 401, 412, 423
		R
		<code>\renewcommand</code>
		99, 104,
		112, 119, 280, 349,
		773, 820, 956, 1010
		S
		<code>\setcounter</code>
	 29
		<code>\shorthand@list</code>
		...
		.. 47, 281, 286,
		350, 355, 450, 451,
		475, 476, 500, 501,
		774, 779, 821, 826,
		866, 867, 957, 962,
		1011, 1016, 1018, 1019
		<code>\shorthandoff</code>
	
		.. 281, 350, 450,
		475, 500, 774, 821,
		866, 957, 1011, 1018
		<code>\shorthandon</code>
	
		.. 286, 355, 451,
		476, 501, 779, 826,
		867, 962, 1016, 1019
		<code>\stepcounter</code>
	
		.. 249, 261, 318,
		330, 372, 404, 435,
		760, 807, 839, 856,
		936, 988, 1047, 1069
		T
		<code>\temp@cmd</code>
	
		225, 226, 228, 238,
		294, 295, 297, 307,
		369, 370, 373, 374,
		397, 398, 400, 401,
		432, 433, 456, 457,
		458, 468, 481, 482,
		483, 493, 506, 507,
		508, 517, 740, 741,
		743, 752, 754, 756,
		761, 787, 788, 790,
		799, 801, 803, 808,
		835, 836, 840, 841,
		852, 853, 857, 858,
		872, 873, 874, 882,
		918, 919, 921, 930,
		931, 933, 937, 970,
		971, 973, 982, 983,
		985, 989, 1024,
		1025, 1026, 1034,
		1044, 1045, 1066, 1067
		<code>\temp@macro</code>
		... 362,
		364, 366, 389, 391, 393
		<code>\temp@mag@cmd</code>
		... 239,
		244, 250, 308, 313, 319
		<code>\temp@ph@cmd</code>
		... 241,
		246, 262, 310, 315, 331
		V
		<code>\values</code>
	 537,
		539, 555, 572, 573,
		576, 577, 579, 580,
		593, 596, 597, 609,
		611, 614, 630, 633, 649
		W
		<code>\write</code>
	 43
		Y
		<code>\y</code>
	 528, 529
		Z
		<code>\z</code>
	 539, 541,
		542, 545, 546, 548, 549

Change History

v1.0			
General: Initial release	1		
v1.0.1			
\addBodeZPKPlots : Improved optional argument handling.	25		
\BodeZPK : Pass arbitrary TikZ commands as options.	23		
v1.0.2			
gnuplot@prefix : Fixed issue #1	18		
v1.0.3			
BodePlot : Added tikz option to environments	28		
\BodeTF : Added Tikz option	24		
\BodeZPK : Added Tikz option	23		
NicholsChart : Added tikz option to environments	36		
\NicholsTF : Added commands and tikz options	36		
\NicholsZPK : Added commands and tikz options	36		
gnuplot@prefix : Added jobname to gnuplot prefix	18		
\NyquistTF : Added commands and tikz options	34		
\NyquistZPK : Added commands and tikz options	33		
\parse@env@opt : Added tikz option to environments	32		
\parse@N@opt : Added commands and tikz options	36		
\parse@opt : Added Tikz option	31		
NyquistPlot : Added tikz option to environments	35		
v1.0.4			
General: Fixed unintended optional argument macro expansion	1		
v1.0.5			
\parse@opt : Fixed a bug	31		
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.	30		
gnuplot@prefix : Fixed issue #6	18		
v1.1.0			
General: Fixed phase wrapping in gnuplot mode	1		
\addBodeTFPlot : Fixed phase wrapping in gnuplot mode	26		
BodeMagPlot : Added separate environments for phase and magnitude plots	28		
BodePhPlot : Added separate environments for phase and magnitude plots	27		
BodePlot : Deprecated BodePlot environment	28		
\BodeTF : Fixed phase wrapping in gnuplot mode	24		
v1.1.1			
General: Enable Hz and rad units	1		
\addBodeComponentPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27		
\addBodeTFPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	26		
\addBodeZPKPlots : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	25		
\addNicholsTFChart : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	36		
\addNyquistTFPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	35		
\addNyquistZPKPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	35		
BodeMagPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	28		
BodePhPlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27		
BodePlot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	28		
\BodeTF : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	24		
\BodeZPK : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	23		
\build@TF@plot : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	30		
get@interval@end : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	20		
ph@y@label : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	19		
\NyquistTF : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	34		
\NyquistZPK : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	33		
v1.1.2			
BodeMagPlot : Defined using the ‘NewEnviron’ command from the			

‘environ’ package to fix conflicts with externalization	28	BodePlot : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	28
BodePhPlot : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	27	\BodeTF : Added code to handle active characters	25
BodePlot : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	28	\BodeZPK : Added code to handle active characters	24
NicholsChart : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	36	NicholsChart : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	36
\PhSOZerosLin : Fix scaling bug introduced in v1.1.1	22	\NicholsTF : Added code to handle active characters	36
NyquistPlot : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	35	\NicholsZPK : Added code to handle active characters	36
v1.1.3		\NyquistTF : Added code to handle active characters	34
\addBodeComponentPlot : Changed implementation to respect user-supplied domain	27	\NyquistZPK : Added code to handle active characters	33
\addBodeTFPlot : Changed implementation to respect user-supplied domain	26	NyquistPlot : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	35
\addBodeZPKPlots : Changed implementation to respect user-supplied domain	25	v1.1.6	
\addNicholsTFChart : Changed implementation to respect user-supplied domain	36	\shorthand@list : Detect ‘babel-french’ using ‘frenchbsetup’	19
\addNicholsZPKChart : Changed implementation to respect user-supplied domain	36	v1.1.7	
\addNyquistTFPlot : Changed implementation to respect user-supplied domain	35	General: Detect and turn off shorthands to improve ‘babel’ compatibility	1
\addNyquistZPKPlot : Changed implementation to respect user-supplied domain	35	BodeMagPlot : Use auto-generated list of active characters instead of manually entering them.	28
v1.1.4		BodePhPlot : Use auto-generated list of active characters instead of manually entering them.	27
\addBodeTFPlot : Changed phase wrapping in pgf mode	26	BodePlot : Use auto-generated list of active characters instead of manually entering them.	28
\addNicholsTFChart : Changed phase wrapping in pgf mode . . .	36	\BodeTF : Use auto-generated list of shorthands instead of manually specifying them	25
\BodeTF : Changed phase wrapping in pgf mode	24	\BodeZPK : Use auto-generated list of shorthands instead of manually specifying them	24
gnuplot@prefix : Changed phase wrapping in pgf mode	18	NicholsChart : Use auto-generated list of active characters instead of manually entering them.	36
v1.1.5		\NicholsTF : Use auto-generated list of shorthands instead of manually specifying them	36
BodeMagPlot : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	28	\NicholsZPK : Use auto-generated list of shorthands instead of manually specifying them	36
BodePhPlot : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	27	\NyquistTF : Use auto-generated list of shorthands instead of manually specifying them	34
		\NyquistZPK : Use auto-generated list of shorthands instead of manually	

specifying them	33	NyquistPlot: Use auto-generated list
\shorthand@list: Directly detect		of active characters instead of
shorthands instead of detecting		manually entering them.
the language.	19	35