

The bodeplot package

version 1.2

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

April 18, 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	29
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

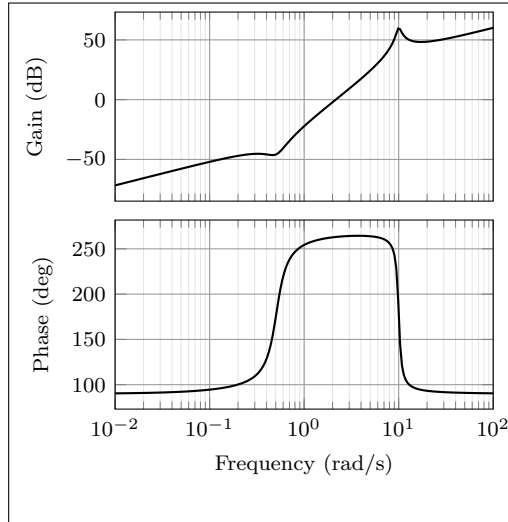
- Before version 1.2, in `pgf` mode, the package set `trig format plots to rad` globally. Version 1.2 onwards, this option is passed to each `addplot` command individually so that it does not affect other plots in the document. To roll back to the pre-1.2 behavior, load the package with `\usepackage[pgf]{bodeplot}[=2024-02-06]`.
- In `pgf` mode, Bode phase plots and Nichols charts in TF form wrap angles so that they are always between -180 and 180° or $-\pi$ and π radian. As such, these plots will show phase wrapping discontinuities. Since v1.1.1, in `gnuplot` mode, the package uses the `smooth unwrap` filter to correct wrapping discontinuities. As of now, I have not found a way to do this in `pgf` mode, any merge requests or ideas you may have are welcome! Since v1.1.4, you can redefine the `n@mod` macro using the commands `\makeatletter\renewcommand{\n@mod}{\n@mod@p}\makeatother` to wrap the phase between 0 and 360° or 0 and 2π radian. The commands `\makeatletter\renewcommand{\n@mod}{\n@mod@n}\makeatother` will wrap the phase between -360 and 0° or -2π and 0 radian.
- Use of the `declutter` option with other directory management tools such as a `tikzexternalize` prefix is not recommended.

2 TL;DR

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

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

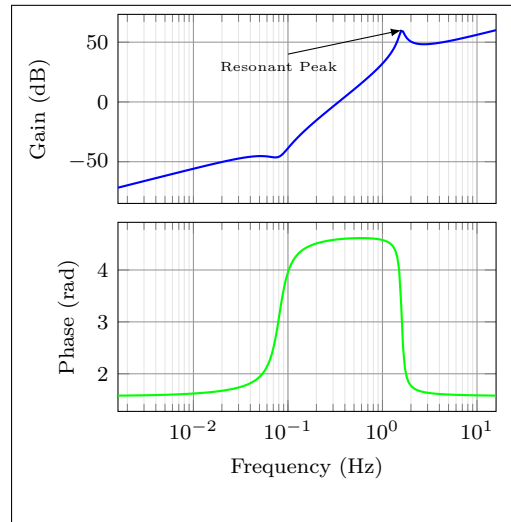
Bode plot in ZPK format



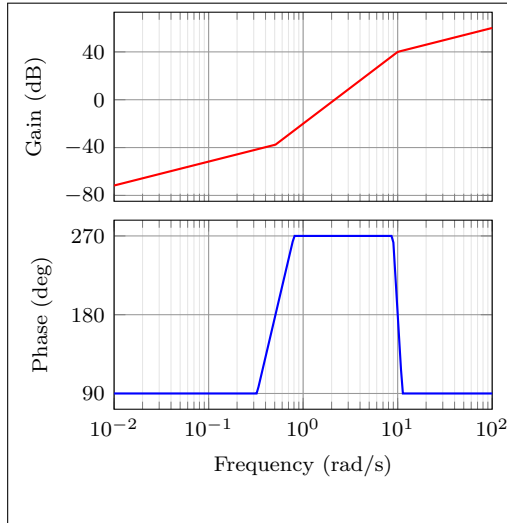
```
\BodeZPK{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

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

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



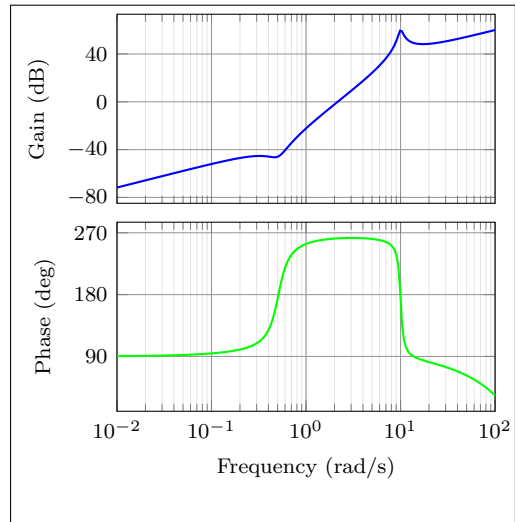
Linear approximation with customization



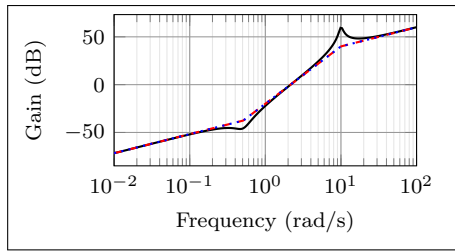
```
\BodeZPK[%
plot/mag/{red,thick},
plot/ph/{blue,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90},
approx/linear%
]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

Plot with delay and customization

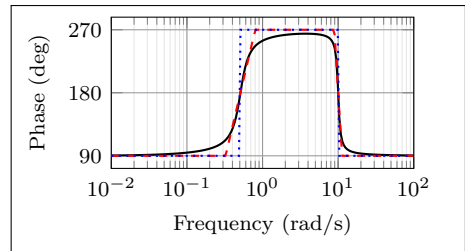
```
\BodeZPK[%
plot/mag/{blue,thick},
plot/ph/{green,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90%
}]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10,
d/0.01%
}
{0.01}
{100}
```



Individual gain and phase plots with more customization

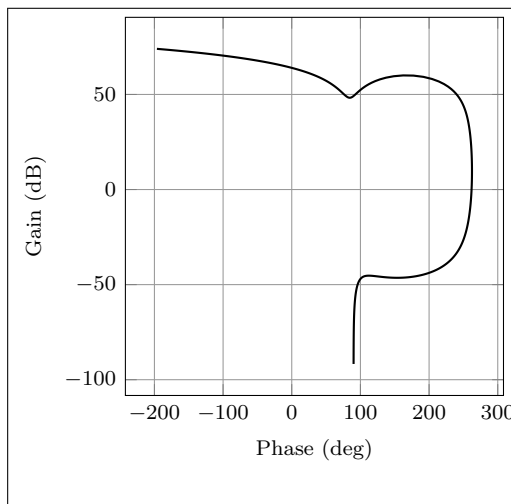


```
\begin{BodeMagPlot}[%
  axes/{height=2cm,
  width=4cm}
]
{0.01}
{100}
\addBodeZPKPlots[%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
]
{magnitude}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\end{BodeMagPlot}
```



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

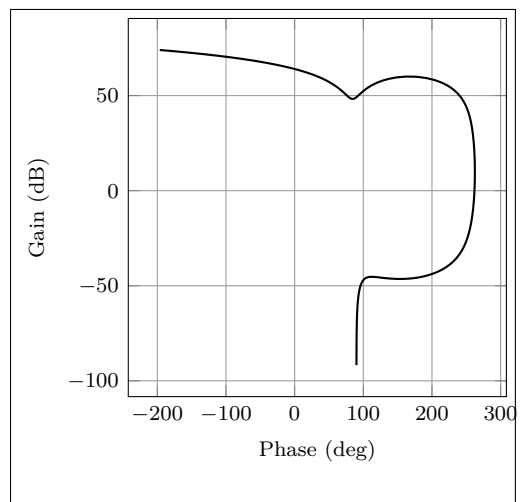
Nichols chart



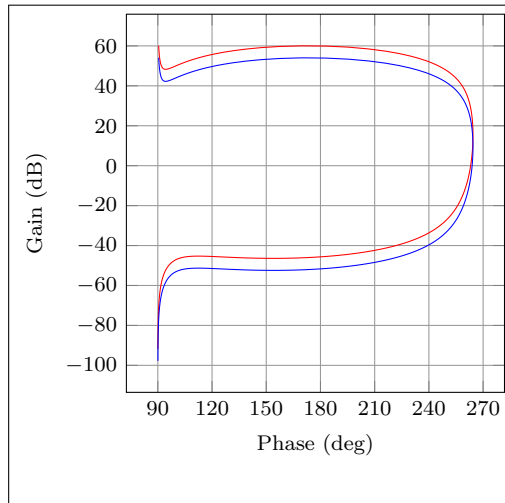
```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01%
}
{0.001}
{500}
```

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}
```



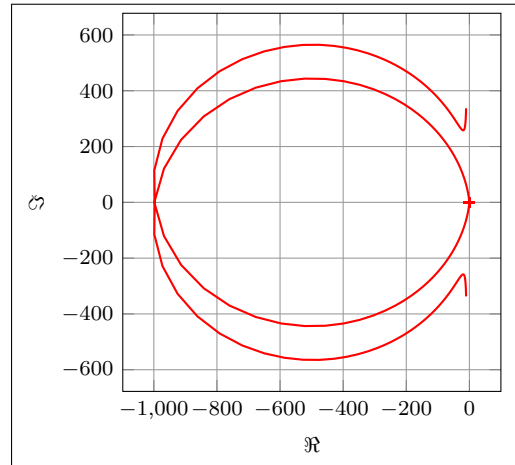
Multiple Nichols charts with customization



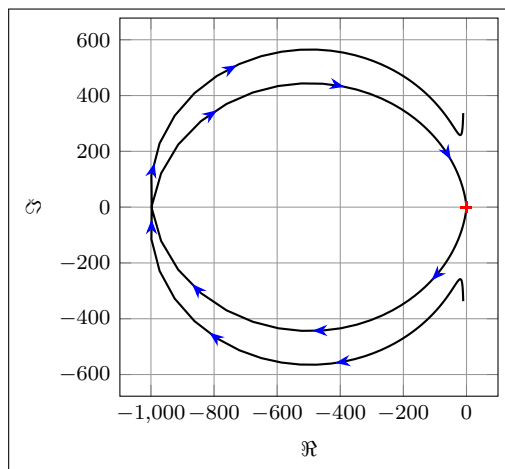
```
\begin{NicholsChart}[%
ytick distance=20,
xtick distance=30
]
{0.001}
{100}
\addNicholsZPKChart [red,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
\addNicholsZPKChart [blue,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/5%
}
\end{NicholsChart}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

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}
```



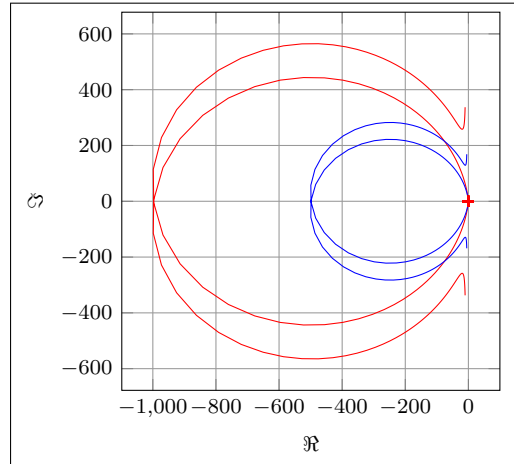
Nyquist plot in TF format with arrows



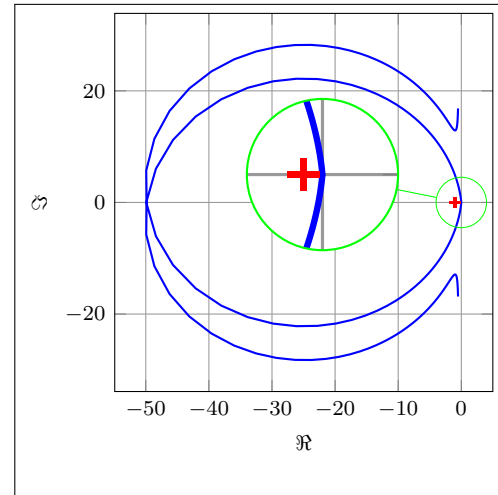
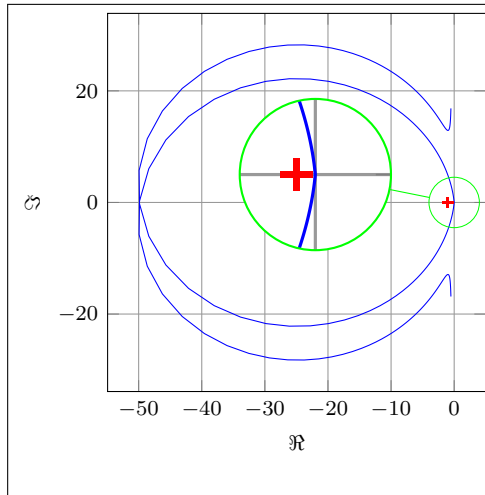
```
\NyquistTF[%
plot/{%
samples=1000,
postaction=decorate,
decoration={%
markings,
mark=between positions 0.1 and 0.9 step 5em with {%
\arrow{Stealth [length=2mm, blue]}
}
}%
}
]
{%
num/{10,2,2.6,0},
den/{1,1,100.25}%
}
{-30}
{30}
```

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.

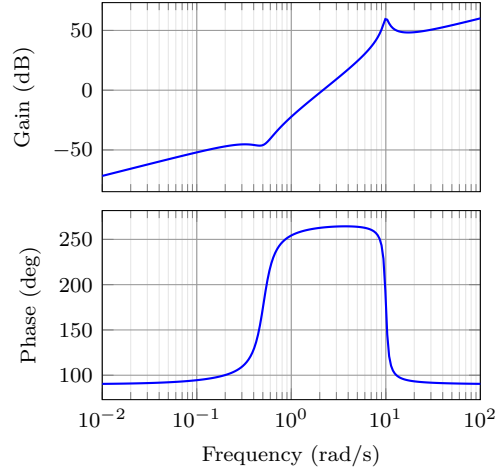


Figure 1: Output of the `\BodeZPK` macro.

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

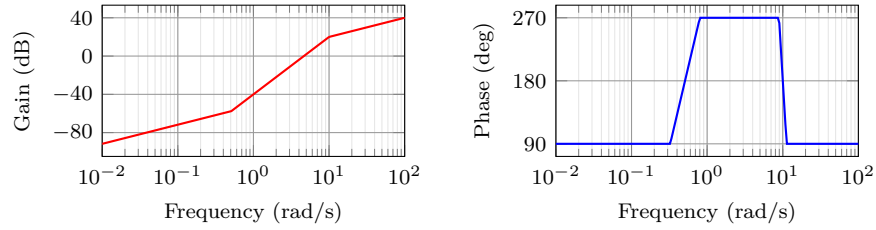


Figure 2: Customization of the default `\BodeZPK` macro.

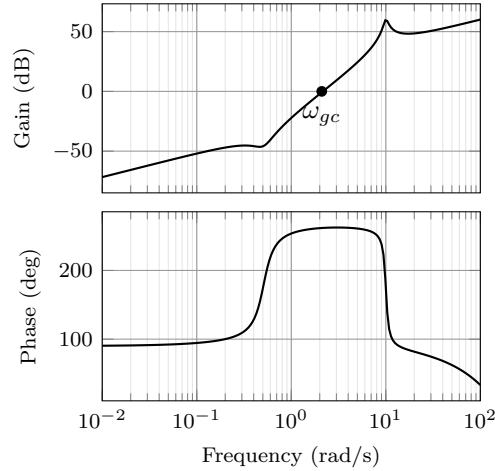


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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,...]
                  {\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,...]
                  {\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

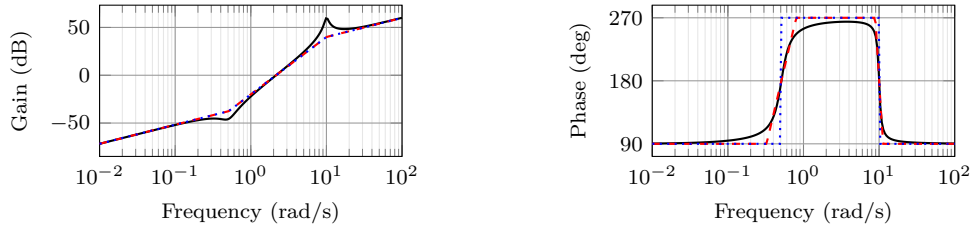


Figure 4: Superimposed approximate and true Bode plots using the `BodeMagPlot` and `BodePhPlot` environments and the `\addBodeZPKPlots` macro.

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

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]{%
```

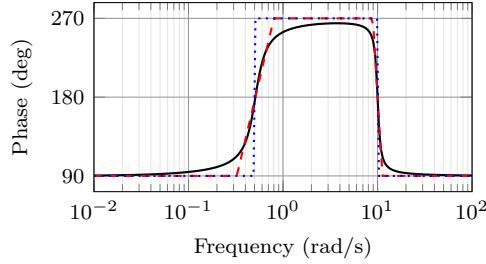


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePh-Plot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

```

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

```

which gives us the plot in Figure 5.

3.1.2 Basic components of the second order

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

This entry describes 12 different macros of the form `\TypeS0FeatureApprox` that take the coefficients a_1 and a_0 of a general second order system as inputs. The **Feature** in the macro name should be replaced by either **Poles** or **Zeros** to generate the Bode plot of $G(s) = \frac{1}{s^2+a_1s+a_0}$ or $G(s) = s^2 + a_1s + a_0$, respectively. The **Type** in the macro name should be replaced by either **Mag** or **Ph** to generate a parametric function corresponding to the magnitude or the phase plot, respectively. The **Approx** in the macro name should either be removed, or it should be replaced by **Lin** or **Asymp** to generate the true Bode plot, the linear approximation, or the asymptotic approximation, respectively.

`\MagS0FeaturePeak` `\MagS0FeaturePeak[<draw-options>]{<a1>}{<a0>}`

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

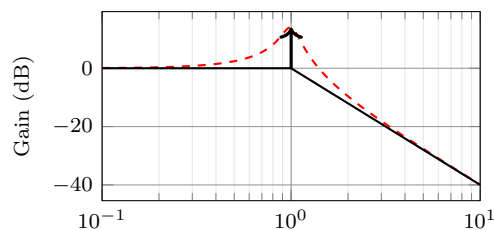


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

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*] $\{\text{num}/\{\text{coeffs}\}, \text{den}/\{\text{coeffs}\}, d/\{\text{delay}\}\}$ $\{\text{min-freq}\}\{\text{max-freq}\}$

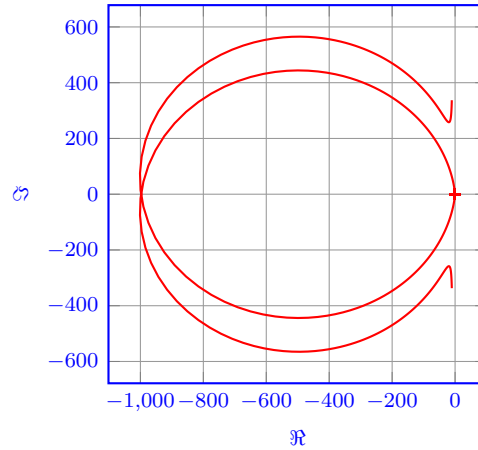


Figure 7: Output of the `\NyquistZPK` macro.

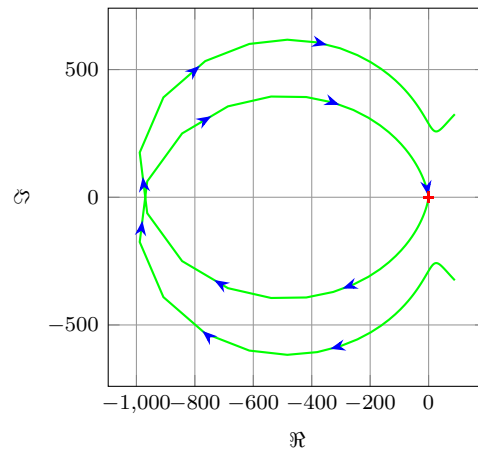


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

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, obj2/\{\langle opt2\}\rangle, \dots]
  \{\langle min-frequency\rangle\}\{\langle max-frequency\rangle\}
  \addNyquist...
\end{NyquistPlot}
```

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

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

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

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

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.

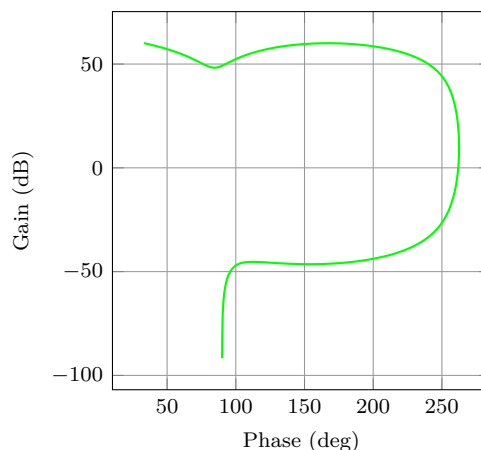


Figure 9: Output of the `\NyquistZPK` macro.

- `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 \else
24 \newcommand{\n@pow}[2]{(#1)**(#2)}
```

Then, we create a counter so that a new data table is generated and for each new plot. If the plot macros have not changed, the tables, once generated, can be reused by `gnuplot`, which reduces compilation time. The `declutter` option is used to enable the `gnuplot` directory to declutter the working directory.

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

get@interval@end 120 \def\get@interval@start#1:#2\@nil{#1}
121 \def\get@interval@end#1:#2\@nil{#2}

```

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

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

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

```

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

```

`\PhKAsymp` True magnitude and phase parametric functions for a pure delay $G(s) = e^{-Ts}$. The
`\PhKLin` macros take two arguments corresponding to real and imaginary part of the gain to
facilitate code reuse between delays, gains, poles, and zeros, but only real gains are
supported. The second argument, if supplied, is ignored.

```

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

```

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

```

\MagPoleLin 130 \newcommand*\MagPole[2]
\PhPole     131 {(-20*log10(sqrt(\n@pow{#1}{2} + \n@pow{t - (#2)}{2})))}

```

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

```

\PhPoleLin 132 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
133   -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :
134   -20*log10(t)
135 )}

```

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

```

136 \newcommand*\MagPoleAsymp{\MagPoleLin}

```

Parametric function for the true phase of a complex pole.

```

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

```

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

```

141 \newcommand*\PhPoleLin}[2]{
142 ((abs(#1)+abs(#2) == 0 ? -pi/2 :
143 (t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
144 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))))) ?
145 (-atan2(-(#2), -(#1))) :
146 (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
147 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))))) ?
148 (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) :
149 (-atan2(-(#2), -(#1)) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
150 (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} +
151 \n@pow{#2}{2})))))))*((#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) + atan2(-
152 (#2), -(#1)))/
153 (Log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
154 (\n@pow{#1}{2} + \n@pow{#2}{2})))))))*\ph@scale)}

```

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

```

154 \newcommand*\PhPoleAsymp}[2]{((t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}))) ?
155 (-atan2(-(#2), -(#1))) :
156 (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2))*\ph@scale)}

```

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

```

\MagZeroAsymp \newcommand*\MagZero}{0-\MagPole}
\MagZeroLin \newcommand*\MagZeroLin}{0-\MagPoleLin}
\PhZeroAsymp \newcommand*\MagZeroAsymp}{0-\MagPoleAsymp}
\PhZeroLin \newcommand*\PhZero}{0-\PhPole}
\newcommand*\PhZeroLin}{0-\PhPoleLin}
\newcommand*\PhZeroAsymp}{0-\PhPoleAsymp}

```

4.3 Second order systems.

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

`\MagCSPoles` Consider the canonical second order transfer function $G(s) = \frac{1}{s^2 + 2\zeta w_n s + w_n^2}$. We start with true, linear, and asymptotic magnitude plots for this transfer function.

```

\MagCSPolesAsymp \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\MagCSPolesLin 163 \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles 164 - \n@pow{t}{2}){2} + \n@pow{2*#1*#2*t}{2})))}
\PhCSPolesAsymp 165 \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
\PhCSPolesLin 40*log10(t))}
\MagCSZeros 166 \newcommand*\MagCSPolesAsymp}{\MagCSPolesLin}

```

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

```

\MagCSZerosLin \newcommand*\PhCSPoles}[2]{((-atan2((2*(#1)*(#2)*t), (\n@pow{#2}{2}
\PhCSZerosAsymp 167 \newcommand*\PhCSPoles}[2]{((-atan2((2*(#1)*(#2)*t), (\n@pow{#2}{2}
\PhCSZerosLin 168 - \n@pow{t}{2})))*\ph@scale)}
169 \newcommand*\PhCSPolesLin}[2]{((t < (#2 / (\n@pow{10}{abs(#1)}))) ?
170 0 :
171 (t >= (#2 * (\n@pow{10}{abs(#1)}))) ?
172 (#1>0 ? -pi : pi) :
173 (#1>0 ? (-pi*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
174 (pi*(log10(t*(\n@pow{10}{abs(#1)})/#2))/(2*abs(#1)))))*\ph@scale)}
175 \newcommand*\PhCSPolesAsymp}[2]{((#1>0?(t<#2?0:-
pi):(t<#2?0:pi))*\ph@scale)}

```

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

```

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

```

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

```

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

```

`\MagS0Poles` `\MagS0PolesAsymp` Consider a general second order transfer function $G(s) = \frac{1}{s^2 + as + b}$. We start with true, linear, and asymptotic magnitude plots for this transfer function.

```

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

```

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

```

\PhS0Zeros     195 \newcommand*\PhS0Poles}[2]{((-atan2((#1)*t,((#2) -
\PhS0ZerosAsymp \n@pow{t}{2}))) * \ph@scale)}
\PhS0ZerosLin  196 \newcommand*\PhS0PolesLin}[2]{((#2>0 ?
197   \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
198   (#1>0 ? -pi : pi))}
199 \newcommand*\PhS0PolesAsymp}[2]{((#2>0 ?
200   \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
201   (#1>0 ? -pi : pi))}

```

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

```

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

```

`\MagS0PolesPeak` `\MagS0ZerosPeak` These macros are used to add a resonant peak to linear and asymptotic plots of general second order poles and zeros. Since the plots are parametric, a separate `\draw` command is needed to add a vertical arrow.

```

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

```

4.4 Commands for Bode plots

4.4.1 User macros

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

```
218 \newcommand{\BodeZPK}[4][approx/true]{
```

Most of the work is done by the `\parse@opt` and the `\build@ZPK@plot` macros, described in the 'Internal macros' section. The former is used to parse the optional arguments and the latter to extract poles, zeros, gain, and delay from the first mandatory argument and to generate macros `\func@mag` and `\func@ph` that hold the magnitude and phase parametric functions. The `\noexpand` macros below are needed to so that only the macro `\opt@group` is expanded.

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

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

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

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

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

```

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

```

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

```

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

```

\BodeTF Implementation of this macro is very similar to the **\BodeZPK** macro above. The only difference is the lack of linear and asymptotic plots and slightly different parsing of the mandatory arguments.

```

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

```



```

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

```

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

```

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

```

`\addBodeZPKPlots` This macro is designed to issues multiple `\addplot` macros for the same set of poles, zeros, gain, and delay. All of the work is done by the `\build@ZPK@plot` macro.

```

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

```

```

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

```

`\addBodeTFPlot` This macro is designed to issues a single `\addplot` macros for the set of coefficients and delay. All of the work is done by the `\build@TF@plot` macro.

```

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

```

```

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

```

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

```

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

```

`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`.

```

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

```

```

467   \end{semilogxaxis}
468 \end{tikzpicture}
469 }{}

```

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.

```

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

```

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

```

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

```

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](#).

```

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

```

`\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.

```

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

```

```

565     }
566   \fi
567   \ifnum\pdf@strcmp{\feature}{k}=0
568     \ifnum\pdf@strcmp{#3}{linear}=0
569       \add@feature{#2}{\PhKLin}{\values}
570       \add@feature{#1}{\MagKLin}{\values}
571     \else
572       \ifnum\pdf@strcmp{#3}{asymptotic}=0
573         \add@feature{#2}{\PhKAsymp}{\values}
574         \add@feature{#1}{\MagKAsymp}{\values}
575       \else
576         \add@feature{#2}{\PhK}{\values}
577         \add@feature{#1}{\MagK}{\values}
578       \fi
579     \fi
580   \fi
581   \ifnum\pdf@strcmp{\feature}{d}=0
582     \ifnum\pdf@strcmp{#3}{linear}=0
583       \PackageError {bodeplot} {Linear approximation for pure de-
584         lays is not
585         supported.} {Plot the true Bode plot using 'true' in-
586         stead of 'linear'.}
587     \else
588       \ifnum\pdf@strcmp{#3}{asymptotic}=0
589         \PackageError {bodeplot} {Asymptotic approxima-
590         tion for pure delays is not
591         supported.} {Plot the true Bode plot using 'true' in-
592         stead of 'asymptotic'.}
593       \else
594         \ifdim\values pt < 0pt
595           \PackageError {bodeplot} {Delay needs to be a posi-
596           tive number.}
597         \fi
598       \add@feature{#2}{\PhDel}{\values}
599       \add@feature{#1}{\MagDel}{\values}
600     \fi
601   \fi
602 }
603 }

```

`\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).

```

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

```

```

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

```

`\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.

```

654 \newcommand{\parse@opt}[1]{
655 \gdef\optmag@axes{}
656 \gdef\optph@axes{}
657 \gdef\optph@plot{}
658 \gdef\optmag@plot{}
659 \gdef\opt@group{}
660 \gdef\opt@approx{}
661 \gdef\optph@commands{}
662 \gdef\optmag@commands{}
663 \gdef\opt@tikz{}
664 \foreach \obj/\typ/\opt in {#1} {
665 \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
666 \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0

```

```

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

```

`\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.

```

717 \newcommand{\parse@env@opt}[1]{
718   \gdef\opt@axes{}
719   \gdef\opt@tikz{}
720   \foreach \obj/\opt in {#1} {
721     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
722     \xdef\opt@axes{\unexpanded\expandafter{\opt}}
723   \else

```



```

724     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
725     \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
726   \else
727     \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
728     \unexpanded\expandafter{\obj}}
729   \fi
730 \fi
731 }
732 }

```

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!

```

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

```

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

```

767 \AtBeginDocument{%
768   \if@babel

```

```

769 \let\Orig@NyquistZPK\NyquistZPK
770 \renewcommand{\NyquistZPK}{%
771   \expandafter\shorthandoff\expandafter{\shorthand@list}
772   \NyquistZPK@Shorthandoff
773 }
774 \newcommand{\NyquistZPK@Shorthandoff}[4][]{%
775   \Orig@NyquistZPK[#1]{#2}{#3}{#4}
776   \expandafter\shorthandon\expandafter{\shorthand@list}
777 }
778 \fi
779 }

```

\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.

```

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

```

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

```

814 \AtBeginDocument{%
815   \if@babel
816   \let\Orig@NyquistTF\NyquistTF
817   \renewcommand{\NyquistTF}{%
818     \expandafter\shorthandoff\expandafter{\shorthand@list}
819     \NyquistTF@Shorthandoff
820   }
821   \newcommand{\NyquistTF@Shorthandoff}[4][]{%
822     \Orig@NyquistTF[#1]{#2}{#3}{#4}
823     \expandafter\shorthandon\expandafter{\shorthand@list}
824   }

```

```

825 \fi
826 }

```

\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.

```

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

```

\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.

```

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

```

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.

```

861 \AtBeginDocument{%
862   \if@babel
863     \AddToHook{env/NyquistPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand
864     \AddToHook{env/NyquistPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@
865   \fi
866 }
867 \NewDocumentEnvironment{NyquistPlot}{0}{mm+b}{
868   \parse@env@opt{#1}

```

```

869 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
870 \temp@cmd
871 \edef\temp@cmd{\noexpand\begin{axis}[
872     bode@style,
873     height=5cm,
874     domain=#2:#3,
875     xlabel={\Re$},
876     ylabel={\Im$},
877     \unexpanded\expandafter{\opt@axes}
878 ]}
879 \temp@cmd
880 \addplot [only marks,mark=+,thick,red] (-1 , 0);
881 #4
882 \end{axis}
883 \end{tikzpicture}
884 }{}

```

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.

```

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

```

4.6 Nichols charts

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

```

911 \newcommand{\NicholsZPK}[4][1]{
912   \parse@N@opt{#1}
913   \gdef\func@mag{}
914   \gdef\func@ph{}
915   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}

```

```

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

```

```

977     \unexpanded\expandafter{\opt@axes}
978   ]}
979   \temp@cmd
980   \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig for-
mat plots=rad, \opt@plot]}
981   \if@pgfarg
982     \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
983   \opt@commands
984   \else
985     \stepcounter{gnuplot@id}
986     \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
987       { set table $metal;
988         set logscale x 10;
989         set dummy t;
990         set samples \pgfkeysvalueof{/pgfplots/samples};
991         set trange [#3*\freq@scale:#4*\freq@scale];
992         plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
993         unset logscale x;
994         set table $meta2;
995         plot "$metal" using ($1):($2) smooth unwrap;
996         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
997         plot "$meta2" using ($2*\ph@scale):($1);
998       };
999   \opt@commands
1000  \fi
1001  \end{axis}
1002  \end{tikzpicture}
1003 }
1004 \AtBeginDocument{
1005   \if@babel
1006     \let\Orig@NicholsTF\NicholsTF
1007     \renewcommand{\NicholsTF}{%
1008       \expandafter\shorthandoff\expandafter{\shorthand@list}
1009       \NicholsTF@Shorthandoff
1010     }
1011     \newcommand{\NicholsTF@Shorthandoff}[4][[]]{%
1012       \Orig@NicholsTF[#1]{#2}{#3}{#4}
1013       \expandafter\shorthandon\expandafter{\shorthand@list}
1014     }
1015     \AddToHook{env/NicholsChart/begin}{\expandafter\shorthandoff\expandafter{\shorthand@list}}
1016     \AddToHook{env/NicholsChart/end}{\expandafter\shorthandon\expandafter{\shorthand@list}}
1017   \fi
1018 }
1019 \NewDocumentEnvironment{NicholsChart}{0}{mm+b}{
1020   \parse@env@opt{#1}
1021   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
1022   \temp@cmd
1023   \edef\temp@cmd{\noexpand\begin{axis}[
1024     ph@x@label,
1025     bode@style,
1026     domain=#2:#3,
1027     height=5cm,
1028     ylabel={Gain (dB)},
1029     \unexpanded\expandafter{\opt@axes}
1030   ]}
1031   \temp@cmd
1032   #4
1033   \end{axis}
1034   \end{tikzpicture}
1035 }{}
1036 \newcommand{\addNicholsZPKChart}[2][[]]{
1037   \gdef\func@mag{

```

```

1038 \gdef\func@ph{}
1039 \build@ZPK@plot{\func@mag}{\func@ph}{\#2}
1040 \if@pgfarg
1041 \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, trig format plots=rad, #1]}
1042 \temp@cmd ( {\func@ph} , {\func@mag} );
1043 \else
1044 \stepcounter{gnuplot@id}
1045 \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1046 { set table $meta;
1047 set logscale x 10;
1048 set dummy t;
1049 set samples \pgfkeysvalueof{/pgfplots/samples};
1050 set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1051 plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1052 unset logscale x;
1053 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1054 plot "$meta" using ($2*\ph@scale):($1);
1055 };
1056 \fi
1057 }
1058 \newcommand{\addNicholsTFChart}[2][]{
1059 \gdef\func@mag{}
1060 \gdef\func@ph{}
1061 \build@TF@plot{\func@mag}{\func@ph}{\#2}
1062 \if@pgfarg
1063 \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, trig format plots=rad, #1]}
1064 \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
1065 \else
1066 \stepcounter{gnuplot@id}
1067 \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1068 { set table $meta1;
1069 set logscale x 10;
1070 set dummy t;
1071 set samples \pgfkeysvalueof{/pgfplots/samples};
1072 set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1073 plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1074 unset logscale x;
1075 set table $meta2;
1076 plot "$meta1" using ($1):($2) smooth unwrap;
1077 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1078 plot "$meta2" using ($2*\ph@scale):($1);
1079 };
1080 \fi
1081 }

```

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	\ifpackageloaded .. 46	A
\@babelfalse	44	\add@feature
\@babeltrue	47	<u>519</u> , 538, 539, 542,
\@declutterargfalse ..	5	543, 545, 546, 554,
\@declutterargtrue ..	7	555, 558, 559, 561,
\@empty	48	562, 569, 570, 573,
\@hzargfalse	13	574, 576, 577, 593, 594
\@hzargtrue	15	\addBodeComponentPlot
		\@nil
		120, 121
		\@pgfargfalse
		1
		\@pgfargtrue
		3
		\@radargfalse
		9
		\@radargtrue
		11
		\~
		51

.....	427	F	695, 696, 702, 721,
<code>\addBodeTFPlot</code>	384	<code>\freq@filter</code>	724, 891, 894, 897, 900
<code>\addBodeZPKPlots</code> ..	356	<code>\freq@label</code>	73
<code>\addNicholsTFChart</code>	911	<code>\freq@scale</code>	73, 74, 85, 109, 116,
<code>\addNicholsZPKChart</code>	911		229, 251, 255, 263,
<code>\addNyquistTFPlot</code> .	844		267, 298, 320, 324,
<code>\addNyquistZPKPlot</code>	827		332, 336, 366, 375,
<code>\AddToHook</code> .	447, 448,		379, 394, 397, 407,
	472, 473, 497, 498,		411, 418, 422, 429,
	863, 864, 1015, 1016		437, 441, 742, 789,
<code>\AtBeginDocument</code> ..			832, 837, 849, 854,
... ..	45, 274, 343,		921, 939, 973, 991,
	445, 470, 495, 767,		1041, 1050, 1063, 1072
	814, 861, 950, 1004	<code>\func@mag</code> ..	220, 224,
B			241, 253, 289, 293,
<code>\beginngroup</code>	50		310, 322, 735, 739,
<code>\bode@style</code>	58		753, 754, 759, 760,
<code>BodeMagPlot</code> (env.) ..	470		782, 786, 800, 801,
<code>BodePhPlot</code> (env.) ...	445		806, 807, 828, 830,
<code>BodePlot</code> (env.)	495		833, 834, 839, 840,
<code>\bodeplot@prefix</code> ..			845, 847, 850, 851,
.....	28, 30, 35,		856, 857, 913, 917,
	254, 266, 323, 335,		930, 940, 965, 969,
	378, 410, 421, 440,		982, 992, 1037,
	942, 996, 1053, 1077		1039, 1042, 1051,
<code>\BodeTF</code>	287		1059, 1061, 1064, 1073
<code>\BodeTF@Shorthandoff</code>		<code>\func@ph</code> ...	221, 224,
.....	348, 350		243, 265, 290, 293,
<code>\BodeZPK</code>	218		312, 334, 736, 739,
<code>\BodeZPK@Shorthandoff</code>			753, 754, 759, 760,
.....	279, 281		783, 786, 800, 801,
<code>\build@TF@plot</code>			806, 807, 829, 830,
	293, 388, 390, 600,		833, 834, 839, 840,
	786, 847, 969, 1061		846, 847, 850, 851,
<code>\build@ZPK@plot</code> ...			856, 857, 914, 917,
	224, 361, 363, 533,		930, 940, 966, 969,
	739, 830, 917, 1039		982, 992, 1038,
			1039, 1042, 1051,
			1060, 1061, 1064, 1073
C		G	
<code>\currentdegree</code> .	612,	<code>\g@addto@macro</code>	52
	613, 616, 617, 618,	<code>\get@interval@end</code> .	
	620, 621, 631, 632,	120, 121
	635, 636, 637, 639, 640	<code>\get@interval@start</code>	
		120, 120
<code>\den@degree</code>	628, 631	<code>\gnuplot@id</code>	1
<code>\den@im</code>	604, 636, 649, 652	<code>\gnuplot@prefix</code>	1
<code>\den@real</code>	603,		
	633, 639, 650, 652	I	
<code>\dencoeff</code>	627,	<code>\if@babel</code>	44
	630, 633, 636, 639	<code>\if@hzarg</code>	13, 84
<code>\denpow</code>	627, 628, 630, 631	<code>\ifbabelshorthand</code> ..	52
<code>\do</code>	49	<code>\ifcat</code>	520
<code>\dospecials</code>	55	<code>\ifnum</code>	360, 387,
			393, 402, 529, 535,
E			537, 541, 551, 553,
<code>\endgroup</code>	53		557, 567, 568, 572,
environments:			581, 582, 586, 607,
<code>BodeMagPlot</code>	470		613, 626, 632, 645,
<code>BodePhPlot</code>	445		665, 666, 669, 677,
<code>BodePlot</code>	495		678, 681, 689, 692,
			695, 696, 702, 721,
			724, 891, 894, 897, 900
			<code>\ifwindows</code>
			38
			<code>\immediate</code>
			40
		J	
		<code>\jobname</code>	28, 30
		L	
		<code>\Last@LoopValue</code>	527, 529
		<code>\lccode</code>	51
		<code>\loop@delay</code>	605, 646, 650
		<code>\lowercase</code>	52
		M	
		<code>\MagCSPoles</code>	163
		<code>\MagCSPolesAsymp</code> ..	163
		<code>\MagCSPolesLin</code>	163
		<code>\MagCSPolesPeak</code> ...	182
		<code>\MagCSZeros</code>	163
		<code>\MagCSZerosAsymp</code> ..	163
		<code>\MagCSZerosLin</code>	163
		<code>\MagCSZerosPeak</code> ...	182
		<code>\MagDel</code>	128, 594
		<code>\MagK</code>	122, 577
		<code>\MagKAsymp</code>	122, 574
		<code>\MagKLin</code>	122, 570
		<code>\MagPole</code> ...	130, 157, 562
		<code>\MagPoleAsymp</code>	130, 159, 559
		<code>\MagPoleLin</code>	130, 158, 555
		<code>\MagSOPoles</code>	190
		<code>\MagSOPolesAsymp</code> ..	190
		<code>\MagSOPolesLin</code>	190
		<code>\MagSOPolesPeak</code> ...	208
		<code>\MagSOZeros</code>	190
		<code>\MagSOZerosAsymp</code> ..	190
		<code>\MagSOZerosLin</code>	190
		<code>\MagSOZerosPeak</code> ...	208
		<code>\MagZero</code>	157, 546
		<code>\MagZeroAsymp</code> ..	157, 543
		<code>\MagZeroLin</code>	157, 539
		N	
		<code>\n</code>	525, 527
		<code>\n@mod</code>	1, 312, 395, 982, 1064
		<code>\n@mod@n</code>	1
		<code>\n@mod@p</code>	1, 138
		<code>\n@pow</code>	1,
			131, 132, 133, 143,
			144, 146, 147, 149,
			150, 151, 152, 153,
			154, 163, 164, 167,
			168, 169, 171, 173,
			174, 191, 195, 617,
			618, 620, 621, 636,
			637, 639, 640, 651,
			652, 753, 754, 759,
			760, 800, 801, 806,
			807, 833, 834, 839,
			840, 850, 851, 856, 857
		<code>\newcounter</code>	25

<code>\NewDocumentEnvironment</code> 451, 476, 501, 867, 1019	P	<code>\p</code>	552, 554, 555, 558, 559, 561, 562	<code>\PhS0ZerosLin</code>	190	
<code>\NicholsChart</code>	911	<code>\parse@env@opt</code>	452, 477, 502, 717, 868, 1020	<code>\PhZero</code>	157, 545		
<code>\NicholsTF</code>	911	<code>\parse@N@opt</code> ...	734, 781, 885, 912, 964	<code>\PhZeroAsymp</code> ...	157, 542		
<code>\NicholsTF@Shorthandoff</code> 1009, 1011	<code>\parse@opt</code> .	219, 288, 654	<code>\PhZeroLin</code>	157, 538		
<code>\NicholsZPK</code>	911	<code>\pdf@strcmp</code> 360, 387, 393, 402, 535, 537, 541, 551, 553, 557, 567, 568, 572, 581, 582, 586, 607, 626, 645, 665, 666, 669, 677, 678, 681, 689, 692, 695, 696, 702, 721, 724, 891, 894, 897, 900	<code>\plot@macro</code> 358, 361, 363, 367, 377, 385, 388, 390, 395, 398, 409, 420		
<code>\NicholsZPK@Shorthandoff</code> 955, 957	<code>\pgfkeysvalueof</code>	252, 264, 321, 333, 366, 375, 376, 394, 397, 407, 408, 418, 419, 429, 437, 438, 832, 837, 849, 854, 938, 990, 1041, 1049, 1050, 1063, 1071, 1072	..	358, 361, 363, 367, 377, 385, 388, 390, 395, 398, 409, 420		
<code>\num@degree</code>	609, 612	<code>\pgfplotsset</code>	58, 73, 75, 76, 77, 80, 81, 86, 88, 97, 98, 102, 103, 110, 112, 117	R			
<code>\num@im</code> 602, 617, 649, 651		<code>\ph@scale</code> .	73, 78, 82, 96, 101, 125, 129, 140, 153, 156, 168, 174, 175, 195, 312, 334, 336, 409, 411, 650, 753, 754, 759, 760, 800, 801, 806, 807, 833, 834, 839, 840, 850, 851, 856, 857, 940, 943, 982, 992, 997, 1051, 1054, 1064, 1073, 1078	<code>\renewcommand</code>	96, 101, 109, 116, 277, 346, 770, 817, 953, 1007		
<code>\num@real</code>	601, 614, 620, 649, 651	<code>\ph@x@label</code>	73	S			
<code>\numcoeff</code>	608, 611, 614, 617, 620	<code>\ph@y@label</code>	73	<code>\setcounter</code>	26		
<code>\numpow</code> 608, 609, 611, 612		<code>\PhCSPoles</code>	163	<code>\shorthand@list</code> 44, 278, 283, 347, 352, 447, 448, 472, 473, 497, 498, 771, 776, 818, 823, 863, 864, 954, 959, 1008, 1013, 1015, 1016		
<code>\NyquistPlot</code>	861	<code>\PhCSPolesAsymp</code>	163, 200	<code>\shorthandoff</code> 278, 347, 447, 472, 497, 771, 818, 863, 954, 1008, 1015		
<code>\NyquistTF</code>	780	<code>\PhCSPolesLin</code> ..	163, 197	<code>\shorthandon</code> 283, 352, 448, 473, 498, 776, 823, 864, 959, 1013, 1016		
<code>\NyquistTF@Shorthandoff</code> 819, 821	<code>\PhCSZeros</code>	163	<code>\stepcounter</code> 246, 258, 315, 327, 369, 401, 432, 757, 804, 836, 853, 933, 985, 1044, 1066		
<code>\NyquistZPK</code>	733	<code>\PhCSZerosAsymp</code> ...	163	T			
<code>\NyquistZPK@Shorthandoff</code> 772, 774	<code>\PhCSZerosLin</code>	163	<code>\temp@cmd</code>	222, 223, 225, 235, 291, 292, 294, 304, 366, 367, 370, 371, 394, 395, 397, 398, 429, 430, 453, 454, 455, 465, 478, 479, 480, 490, 503, 504, 505, 514, 737, 738, 740, 749, 751, 753, 758, 784, 785, 787, 796, 798, 800, 805, 832, 833, 837, 838, 849, 850, 854, 855, 869, 870, 871, 879, 915, 916, 918, 927, 928, 930, 934, 967, 968, 970, 979, 980, 982, 986, 1021, 1022, 1023, 1031, 1041, 1042, 1063, 1064		
O		<code>\PhDel</code>	129, 593	<code>\temp@macro</code>	359, 361, 363, 386, 388, 390		
<code>\opt@approx</code> 224, 660, 693		<code>\PhK</code>	122, 576	<code>\temp@mag@cmd</code> ..	236, 241, 247, 305, 310, 316		
<code>\opt@axes</code> ..	463, 488, 512, 718, 722, 727, 747, 794, 877, 886, 892, 925, 977, 1029	<code>\PhKAsymp</code> ..	122, 128, 573				
<code>\opt@commands</code> ..	755, 762, 802, 809, 888, 898, 931, 945, 983, 999	<code>\PhKLin</code> ...	122, 128, 569				
<code>\opt@group</code> 233, 302, 659, 690	<code>\PhPole</code> ...	130, 160, 561				
<code>\opt@plot</code> ..	751, 798, 887, 895, 903, 928, 980	<code>\PhPoleAsymp</code>	130, 162, 558				
<code>\opt@tikz</code>	222, 291, 453, 478, 503, 663, 703, 719, 725, 737, 784, 869, 889, 901, 915, 967, 1021	<code>\PhPoleLin</code> .	130, 161, 554				
<code>\optmag@axes</code> ...	236, 305, 655, 679, 684	<code>\PhS0Poles</code>	190				
<code>\optmag@commands</code>	242, 257, 311, 326, 662, 699	<code>\PhS0PolesAsymp</code> ...	190				
<code>\optmag@plot</code> ...	237, 306, 658, 667, 672, 705	<code>\PhS0PolesLin</code>	190				
<code>\optph@axes</code>	238, 307, 656, 682, 685	<code>\PhS0Zeros</code>	190				
<code>\optph@commands</code>	244, 269, 313, 338, 661, 697	<code>\PhS0ZerosAsymp</code> ...	163				
<code>\optph@plot</code>	239, 308, 657, 670, 673, 707	<code>\PhS0ZerosLin</code>	163				
<code>\Orig@BodeTF</code> ...	345, 351	<code>\PhDel</code>	129, 593				
<code>\Orig@BodeZPK</code> ..	276, 282	<code>\PhK</code>	122, 576				
<code>\Orig@NicholsTF</code> 1006, 1012	<code>\PhKAsymp</code> ..	122, 128, 573				
<code>\Orig@NicholsZPK</code>	952, 958	<code>\PhKLin</code> ...	122, 128, 569				
<code>\Orig@NyquistTF</code>	816, 822	<code>\PhPole</code> ...	130, 160, 561				
<code>\Orig@NyquistZPK</code>	769, 775	<code>\PhPoleAsymp</code>	130, 162, 558				

<code>\temp@ph@cmd</code> . . .	238,	573, 574, 576, 577,	Y
	243, 259, 307, 312, 328	590, 593, 594, 606,	<code>\y</code>
		608, 611, 627, 630, 646	525, 526
V			Z
<code>\values</code>	534,	W	<code>\z</code>
	536, 552, 569, 570,	<code>\write</code>	536, 538,
		40	539, 542, 543, 545, 546

Change History

v1.0	General: Initial release	1	v1.1.0	General: Fixed phase wrapping in gnuplot mode	1
v1.0.1	<code>\addBodeZPKPlots</code> : Improved optional argument handling.	25	<code>\addBodeTFPlot</code> : Fixed phase wrapping in gnuplot mode	26	
	<code>\BodeZPK</code> : Pass arbitrary TikZ commands as options.	23	<code>BodeMagPlot</code> : Added separate environments for phase and magnitude plots	28	
v1.0.2	<code>gnuplot@prefix</code> : Fixed issue #1 . . .	18	<code>BodePhPlot</code> : Added separate environments for phase and magnitude plots	27	
v1.0.3	<code>BodePlot</code> : Added tikz option to environments	28	<code>BodePlot</code> : Deprecated <code>BodePlot</code> environment	28	
	<code>\BodeTF</code> : Added Tikz option	24	<code>\BodeTF</code> : Fixed phase wrapping in gnuplot mode	24	
	<code>\BodeZPK</code> : Added Tikz option	23	v1.1.1	General: Enable Hz and rad units	1
	<code>NicholsChart</code> : Added tikz option to environments	36	<code>\addBodeComponentPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27	
	<code>\NicholsTF</code> : Added commands and tikz options	36	<code>\addBodeTFPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	26	
	<code>\NicholsZPK</code> : Added commands and tikz options	36	<code>\addBodeZPKPlots</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	25	
	<code>gnuplot@prefix</code> : Added jobname to gnuplot prefix	18	<code>\addNicholsTFChart</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	36	
	<code>\NyquistTF</code> : Added commands and tikz options	34	<code>\addNyquistTFPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	35	
	<code>\NyquistZPK</code> : Added commands and tikz options	33	<code>\addNyquistZPKPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	35	
	<code>\parse@env@opt</code> : Added tikz option to environments	32	<code>BodeMagPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	28	
	<code>\parse@N@opt</code> : Added commands and tikz options	36	<code>BodePhPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27	
	<code>\parse@opt</code> : Added Tikz option	31	<code>BodePlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	28	
	<code>NyquistPlot</code> : Added tikz option to environments	35	<code>\BodeTF</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	24	
v1.0.4	General: Fixed unintended optional argument macro expansion	1	<code>\BodeZPK</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	23	
v1.0.5	<code>\parse@opt</code> : 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			
	<code>\build@TF@plot</code> : Included phase due to delay in wrapping.	30			
	<code>gnuplot@prefix</code> : Fixed issue #6 . . .	18			

<code>\build@TF@plot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	30	<code>\addNicholsTFChart</code> : Changed phase wrapping in pgf mode	36
<code>get@interval@end</code> : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	20	<code>\BodeTF</code> : Changed phase wrapping in pgf mode	24
<code>ph@y@label</code> : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	19	<code>gnuplot@prefix</code> : Changed phase wrapping in pgf mode	18
<code>\NyquistTF</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	34	v1.1.5	
<code>\NyquistZPK</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	33	<code>BodeMagPlot</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	28
v1.1.2		<code>BodePhPlot</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	27
<code>BodeMagPlot</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	28	<code>BodePlot</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	28
<code>BodePhPlot</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	27	<code>\BodeTF</code> : Added code to handle active characters	25
<code>BodePlot</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	28	<code>\BodeZPK</code> : Added code to handle active characters	24
<code>NicholsChart</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	36	<code>NicholsChart</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	36
<code>\PhSOZerosLin</code> : Fix scaling bug introduced in v1.1.1	22	<code>\NicholsTF</code> : Added code to handle active characters	36
<code>NyquistPlot</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	35	<code>\NicholsZPK</code> : Added code to handle active characters	36
v1.1.3		<code>\NyquistTF</code> : Added code to handle active characters	34
<code>\addBodeComponentPlot</code> : Changed implementation to respect user-supplied domain	27	<code>\NyquistZPK</code> : Added code to handle active characters	33
<code>\addBodeTFPlot</code> : Changed implementation to respect user-supplied domain	26	<code>NyquistPlot</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	35
<code>\addBodeZPKPlots</code> : Changed implementation to respect user-supplied domain	25	v1.1.6	
<code>\addNicholsTFChart</code> : Changed implementation to respect user-supplied domain	36	<code>\shorthand@list</code> : Detect ‘babel-french’ using ‘frenchbsetup’	18
<code>\addNicholsZPKChart</code> : Changed implementation to respect user-supplied domain	36	v1.1.7	
<code>\addNyquistTFPlot</code> : Changed implementation to respect user-supplied domain	35	General: Detect and turn off shorthands to improve ‘babel’ compatibility	1
<code>\addNyquistZPKPlot</code> : Changed implementation to respect user-supplied domain	35	<code>BodeMagPlot</code> : Use auto-generated list of active characters instead of manually entering them.	28
v1.1.4		<code>BodePhPlot</code> : Use auto-generated list of active characters instead of manually entering them.	27
<code>\addBodeTFPlot</code> : Changed phase wrapping in pgf mode	26	<code>BodePlot</code> : Use auto-generated list of active characters instead of manually entering them.	28
		<code>\BodeTF</code> : Use auto-generated list of shorthands instead of manually specifying them	25

<code>\BodeZPK</code> : Use auto-generated list of shorthands instead of manually specifying them	24	<code>\NyquistZPK</code> : Use auto-generated list of shorthands instead of manually specifying them	33
<code>NicholsChart</code> : Use auto-generated list of active characters instead of manually entering them.	36	<code>\shorthand@list</code> : Directly detect shorthands instead of detecting the language.	18
<code>\NicholsTF</code> : Use auto-generated list of shorthands instead of manually specifying them	36	<code>NyquistPlot</code> : Use auto-generated list of active characters instead of manually entering them.	35
<code>\NicholsZPK</code> : Use auto-generated list of shorthands instead of manually specifying them	36	v1.2	
<code>\NyquistTF</code> : Use auto-generated list of shorthands instead of manually specifying them	34	General: Removed global option to process pgf commands in radians .	1
		<code>gnuplot@prefix</code> : Removed global option to process pgf commands in radians	18