

The `bodeplot` package

version 1.1.6

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

January 14, 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 | 22 |
| 4.4.1 | User macros | 22 |
| 4.4.2 | Internal macros | 28 |
| 4.5 | Nyquist plots | 32 |
| 4.5.1 | User macros | 32 |
| 4.5.2 | Internal commands | 35 |
| 4.6 | Nichols charts | 36 |

1 Introduction

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

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

and the zero-pole-gain (ZPK) form

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

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

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

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

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

1.1 External Dependencies

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

1.2 Directory Structure

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

1.3 Limitations

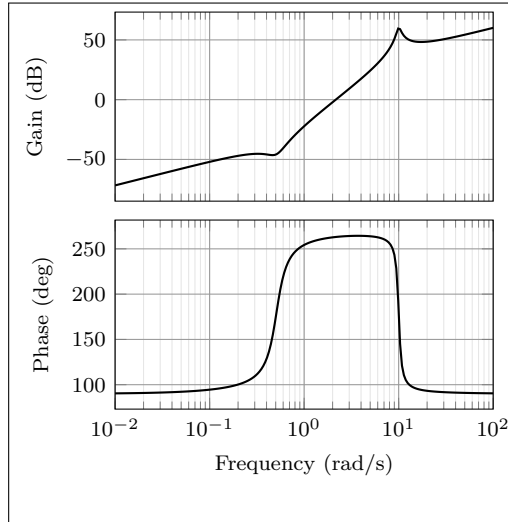
- In `pgf` mode, Bode phase plots and Nichols charts in TF form wrap angles so that they are always between -180° and 180° or $-\pi$ and π radian. As such, these plots will show phase wrapping discontinuities. Since v1.1.1, in `gnuplot` mode, the package uses the `smooth unwrap` filter to correct wrapping discontinuities. As of now, I have not found a way to do this in `pgf` mode, any merge requests or ideas you may have are welcome! Since v1.1.4, you can redefine the `n@mod` macro using the commands `\makeatletter\renewcommand{\n@mod}{\n@mod@p}\makeatother` to wrap the phase between 0° and 360° or 0 and 2π radian. The commands `\makeatletter\renewcommand{\n@mod}{\n@mod@n}\makeatother` will wrap the phase between -360° and 0° or -2π and 0 radian.
- Use of the `declutter` option with other directory management tools such as a `tikzexternalize` prefix is not recommended.

2 TL;DR

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

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

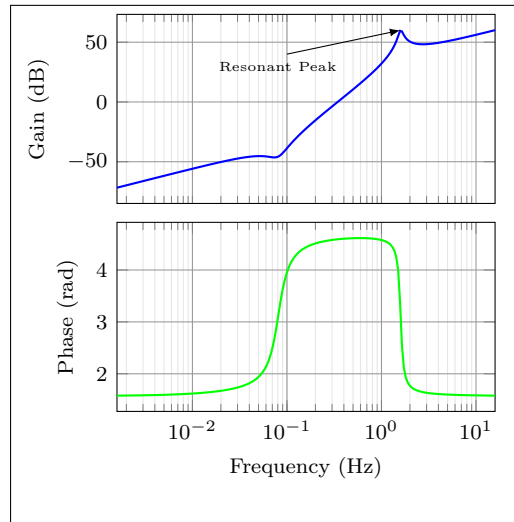
Bode plot in ZPK format



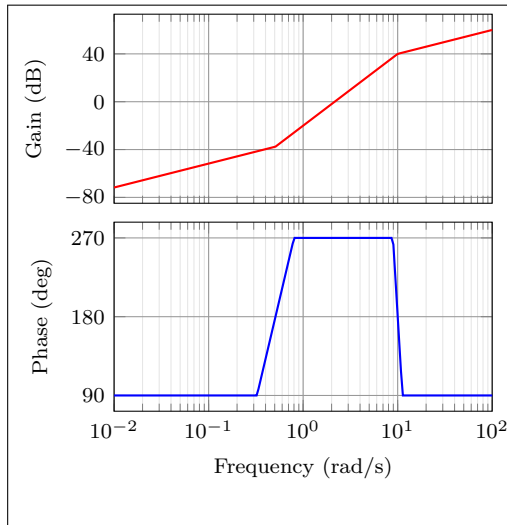
```
\BodeZPK{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

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

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



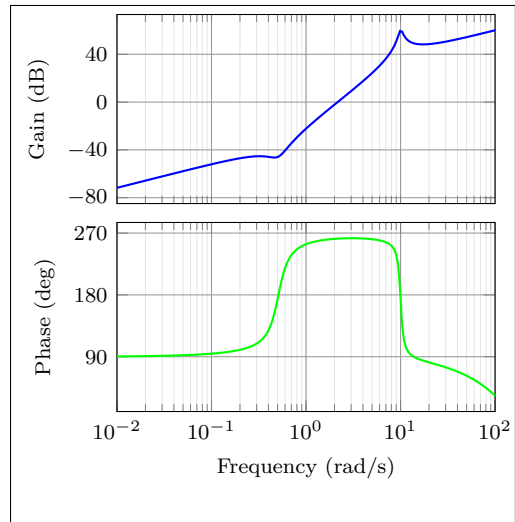
Linear approximation with customization



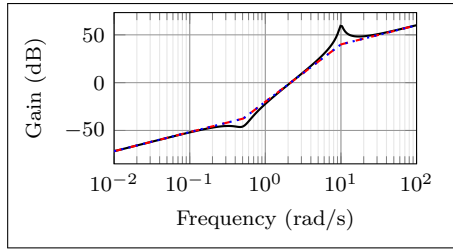
```
\BodeZPK[%
plot/mag/{red,thick},
plot/ph/{blue,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90},
approx/linear%
] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
{0.01}
{100}
```

Plot with delay and customization

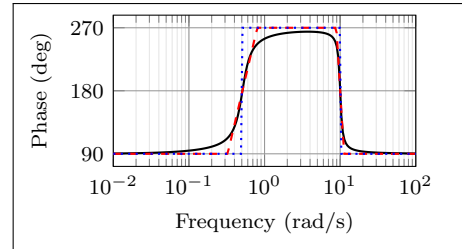
```
\BodeZPK[%
plot/mag/{blue,thick},
plot/ph/{green,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90}
] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10,
d/0.01%
}
{0.01}
{100}
```



Individual gain and phase plots with more customization

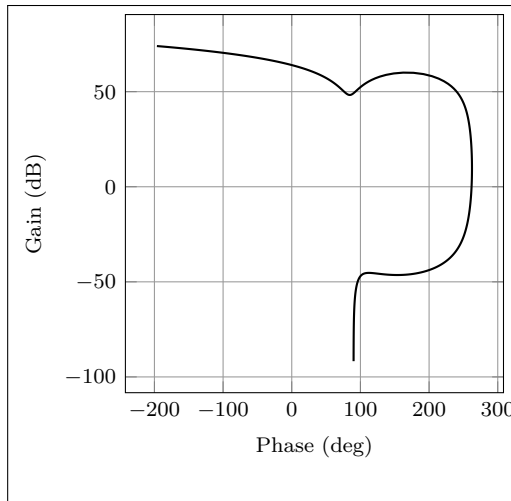


```
\begin{BodeMagPlot}{%
  axes/{height=2cm,
  width=4cm}
}
{0.01}
{100}
\addBodeZPKPlots[%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
]
{magnitude}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
}\end{BodeMagPlot}
```



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

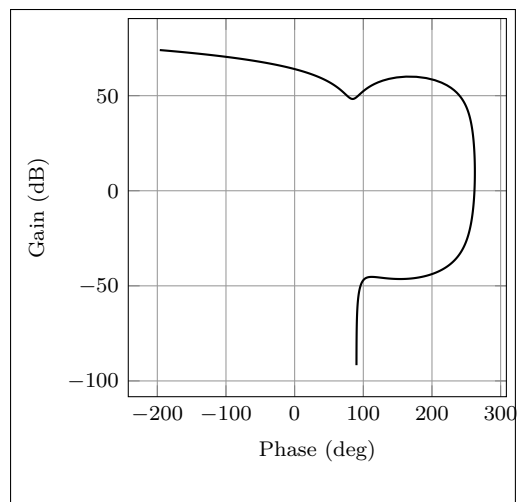
Nichols chart



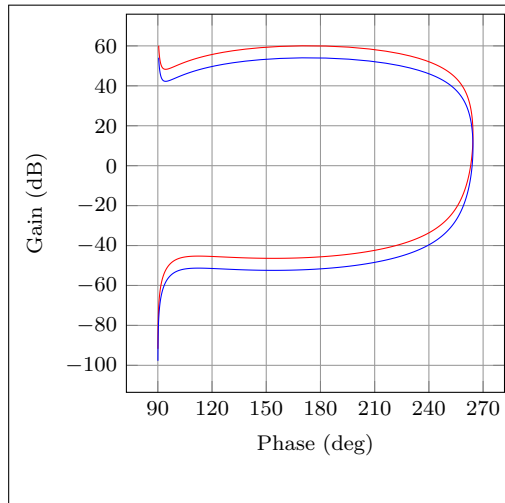
```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01%
}
{0.001}
{500}
```

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

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



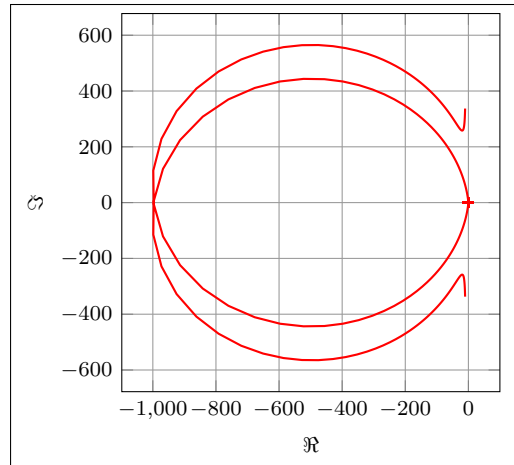
Multiple Nichols charts with customization



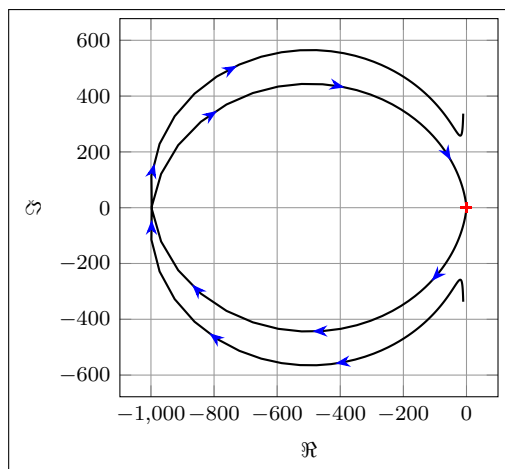
```
\begin{NicholsChart}[%
ytick distance=20,
xtick distance=30
]
{0.001}
{100}
\addNicholsZPKChart [red,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10%
}
\addNicholsZPKChart [blue,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/5%
}
\end{NicholsChart}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

Nyquist plot

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



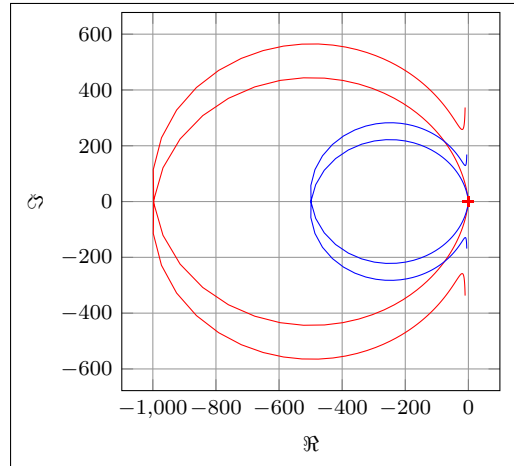
Nyquist plot in TF format with arrows



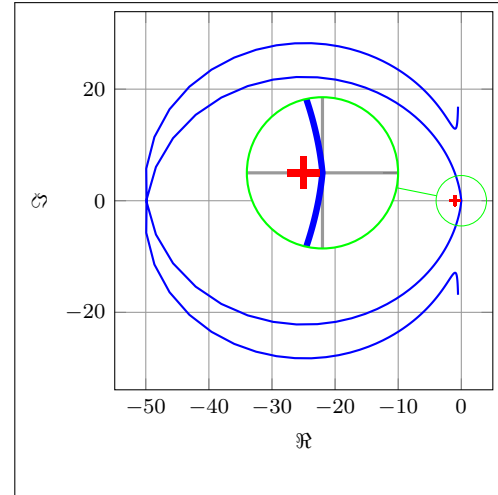
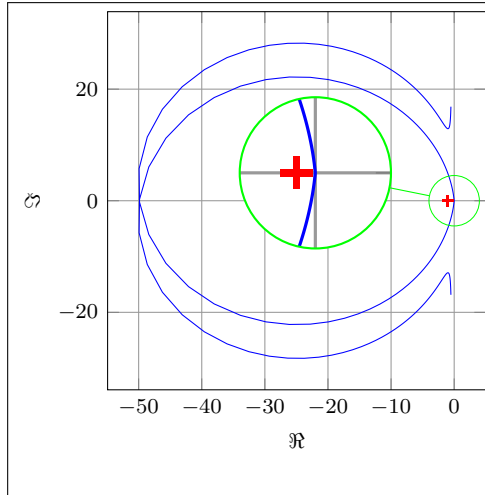
```
\NyquistTF[%
plot/{%
samples=1000,
postaction=decorate,
decoration={%
markings,
mark=between positions 0.1 and 0.9 step 5em with {%
\arrow{Stealth [length=2mm, blue]}
}
}
}%
]
{%
num/{10,2,2.6,0},
den/{1,1,100.25}%
}
{-30}
{30}
```

Multiple Nyquist plots with customization

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



Nyquist plots with additional commands, using two different macros



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

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

3 Usage

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

3.1 Bode plots

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

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

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

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

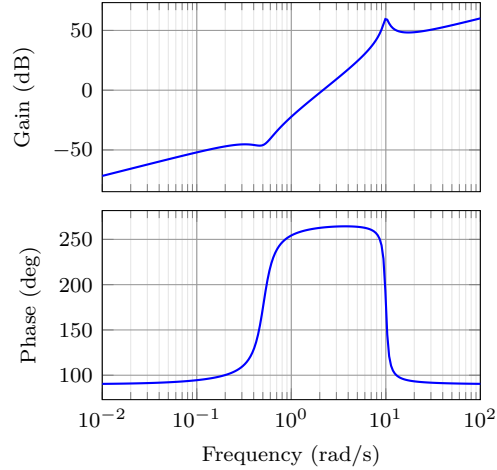


Figure 1: Output of the `\BodeZPK` macro.

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

For example, given a transfer function

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

its Bode plot over the frequency range `[0.01, 100]` can be generated using

```
\BodeZPK [blue,thick]
  {z/{0,{-0.1,-0.5}},{-0.1,0.5}},p/{{-0.5,-10},{-0.5,10}},k/10}
  {0.01}{100}
```

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

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

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

which generates the plot in Figure 2.

```
\BodeTF [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

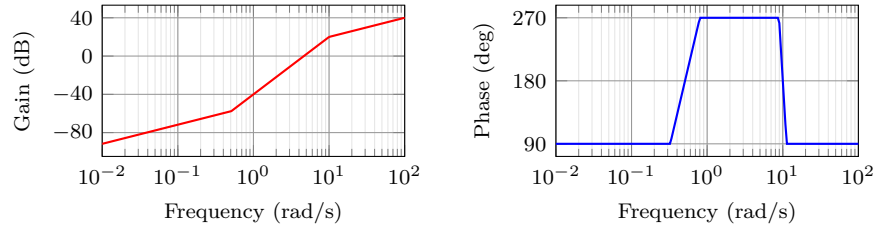


Figure 2: Customization of the default `\BodeZPK` macro.

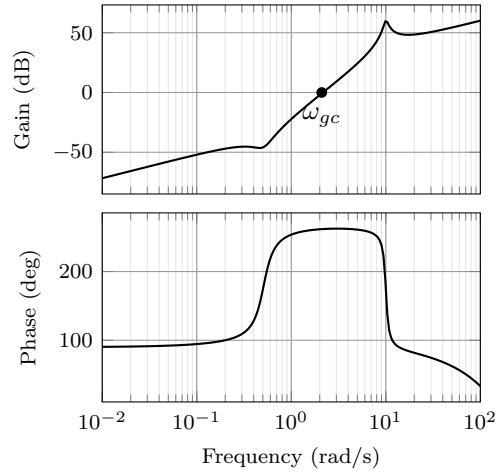


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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]
  \{\langle min-frequency\rangle\}\{\langle max-frequency\rangle\}
  \addBode...
  \end{BodeMagPlot}
```

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

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

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

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

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

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

Generates the appropriate parametric functions and supplies them to multiple `\addplot` macros, one for each `approx/{opt}` pair in the optional argument. If no optional argument is supplied, then a single `\addplot` command corresponding to a thick true Bode plot is generated. If an optional argument is supplied, it needs to be one of `true/{opt}`, `linear/{opt}`, or `asymptotic/{opt}`. This macro can be used inside any `semilogaxis` environment as long as a domain for the x-axis is supplied through either the `approx/{opt}` interface or directly in the optional argument of the `semilogaxis` environment. Use with the `BodePlot` environment supplied with this package is recommended. The second mandatory argument, `plot-type` is either `magnitude` or `phase`. If it is not equal to `phase`, it is assumed to be `magnitude`. The last mandatory argument is the same as `\BodeZPK`.

For example, given the transfer function in (4), its linear, asymptotic, and true Bode plots can be superimposed using

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

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

which generates the plot in Figure 4.

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

Generates a single parametric function for either Bode magnitude or phase plot of a transfer function in TF form. The generated parametric function is passed to the

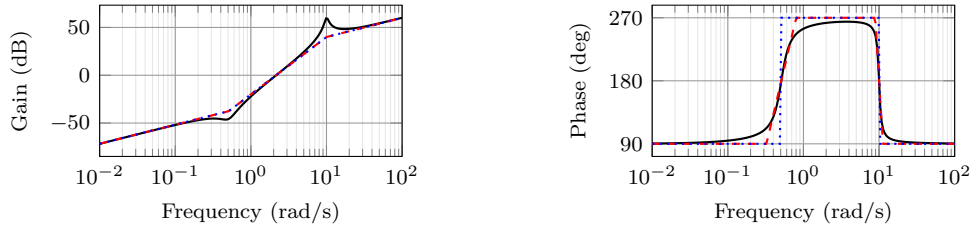


Figure 4: Superimposed approximate and true Bode plots using the `BodeMagPlot` and `BodePhPlot` environments and the `\addBodeZPKPlots` macro.

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

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

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

3.1.1 Basic components up to first order

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

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

- `\MagK{k}{0}` or `\MagK{k}{400}` generates a parametric function for the true Bode magnitude of $G(s) = k$
- `\PhPoleLin{a}{b}` generates a parametric function for the linear approximation of the Bode phase of $G(s) = \frac{1}{s-a-ib}$.
- `\PhDel{T}{200}` or `\PhDel{T}{0}` generates a parametric function for the Bode phase of $G(s) = e^{-Ts}$.

All 20 of the macros defined by combinations of `Type`, `Feature`, and `Approx`, and any `gnuplot` (or `pgfplot` if the `pgf` class option is loaded) compatible function of the 20 macros can be used as `plot-command` in the `addBodeComponentPlot` macro. This is sufficient to generate the Bode plot of any rational transfer function with delay. For example, the Bode phase plot in Figure 4 can also be generated using:

```
\begin{BodePhPlot}[ytick distance=90]{0.01}{100}
  \addBodeComponentPlot[black,thick]{%
    \PhZero{0}{0} + \PhZero{-0.1}{-0.5} + \PhZero{-0.1}{0.5} +
```

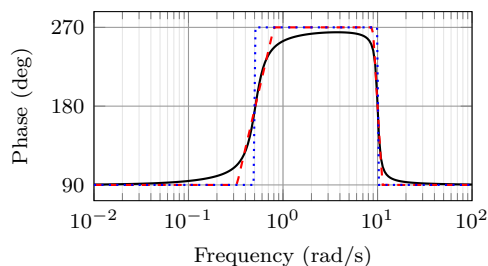


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePhPlot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

```

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

```

which gives us the plot in Figure 5.

3.1.2 Basic components of the second order

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

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

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

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

```

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

```

generates the plot in Figure 6.

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

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

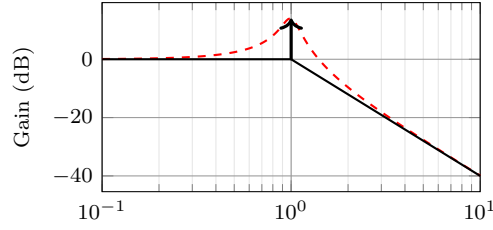


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

- `\MagCSFeaturePeak` `\MagCSFeaturePeak` [*draw-options*] [*zeta*] [*omega-n*]
This entry describes 2 different macros of the form `\MagCSFeaturePeak` that take the damping ratio, ζ , and the natural frequency, ω_n of a canonical second order system as inputs, and draw a resonant peak using the `\draw TikZ` macro. The **Feature** in the macro name should be replaced by either **Poles** or **Zeros** to generate a peak for poles and a valley for zeros, respectively.
- `\MagCCFeaturePeak` `\MagCCFeaturePeak` [*draw-options*] [*real-part*] [*imaginary-part*]
This entry describes 2 different macros of the form `\MagCCFeaturePeak` that take the real and imaginary parts of a pair of complex conjugate poles or zeros as inputs, and draw a resonant peak using the `\draw TikZ` macro. The **Feature** in the macro name should be replaced by either **Poles** or **Zeros** to generate a peak for poles and a valley for zeros, respectively.

3.2 Nyquist plots

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

Plots the Nyquist plot of a transfer function given in ZPK format with a thick red + marking the critical point (-1,0). The mandatory arguments are the same as `\BodeZPK`. Since there is only one plot in a Nyquist diagram, the `\typ` specifier in the optional argument tuples is not needed. As such, the supported optional argument tuples are `plot/{opt}`, which passes `{opt}` to `\addplot`, `axes/{opt}`, which passes `\opt` to the `axis` environment, and `tikz/{opt}`, which passes `\opt` to the `tikzpicture` environment. Asymptotic/linear approximations are not supported in Nyquist plots. If just `{opt}` is provided as the optional argument, it is interpreted as `plot/{opt}`. Arrows to indicate the direction of increasing ω can be added by adding `\usetikzlibrary{decorations.markings}` and `\usetikzlibrary{arrows.meta}` to the preamble and then passing a tuple of the form

```
plot/{postaction=decorate,decoration={markings,
mark=between positions 0.1 and 0.9 step 5em with {%
\arrow{Stealth|} [|length=2mm, blue]}}
```

Caution: with a high number of samples, adding arrows in this way may cause the error message ! **Dimension too big**.

For example, the command

```
\NyquistZPK[plot/{red,thick,samples=2000},axes/{blue,thick}]
{z/{0,-0.1,-0.5,-0.1,0.5}},p/{-0.5,-10},{-0.5,10}},k/10}
{-30}{30}
```

generates the Nyquist plot in Figure 7.

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

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

```
\NyquistTF[plot/{green,thick,samples=500,postaction=decorate,
```

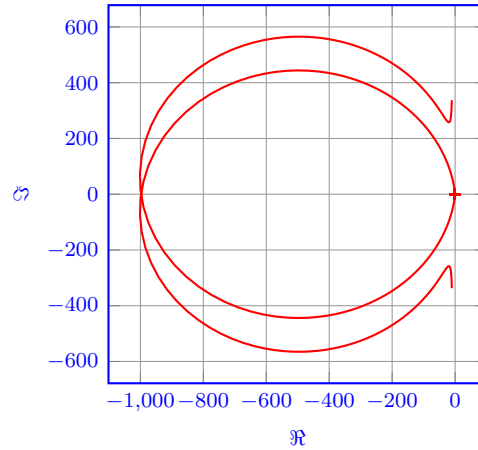


Figure 7: Output of the `\NyquistZPK` macro.

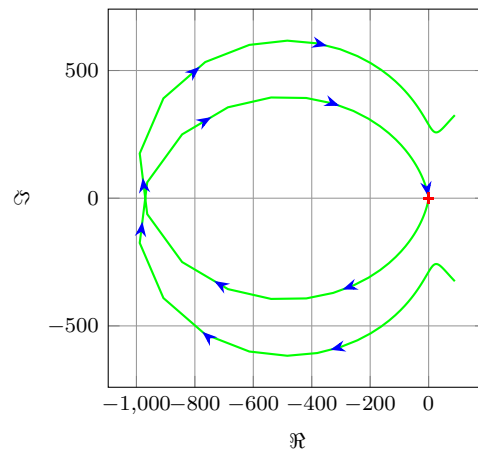


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

```

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 \rangle]
\langle min-frequency \rangle \langle max-frequency \rangle
\addNyquist...
\end{NyquistPlot}

```

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

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

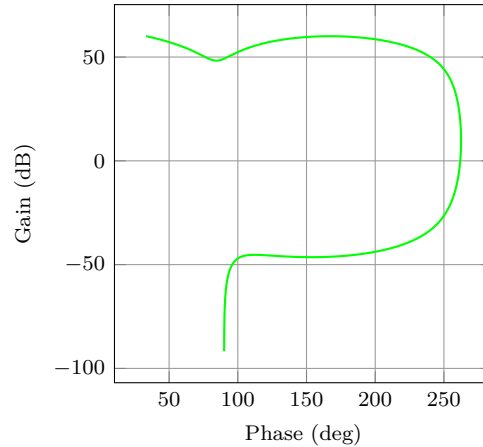


Figure 9: Output of the `\NyquistZPK` macro.

- `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.

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

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

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

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

3.3 Nichols charts

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

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

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

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

```
\NicholsTF[plot/{green,thick,samples=2000}]
           {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},...]
```



```

    {\langle min-frequency \rangle}{\langle max-frequency \rangle}
  \addNichols...
\end{NicholsChart}

```

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

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

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

```

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

```

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

```

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

Then, we define new macros to unify `pgfplots` and `gnuplot`. New macros are defined for the `pow` and `mod` functions to address differences between the two math engines.

```
18 \newcommand{\n@mod}[2]{(#1)-((round((#1)/(#2)))*(#2))}
19 \newcommand{\n@mod@p}[2]{(#1)-((floor((#1)/(#2)))*(#2))}
20 \newcommand{\n@mod@n}[2]{(#1)-((floor((#1)/(#2))+1)*(#2))}
21 \if@pgfarg
22 \newcommand{\n@pow}[2]{(#1)^(#2)}
23 \pgfplotsset{
24 trig format plots=rad
25 }
26 \else
27 \newcommand{\n@pow}[2]{(#1)**(#2)}
```

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

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

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

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

`\if@babel@french` Check if the `babel` package is loaded with French language option.

```

47 \newif\if@babel@french\@babel@frenchfalse
48 \AtBeginDocument{
49   \ifdefined\frenchbsetup
50     \@babel@frenchtrue
51   \fi
52 }

```

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

```

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

```

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

```

```

103 frequency unit/Hz/.code={
104   \renewcommand{\freq@scale}{2*pi}
105   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}
106   \ifpgfarg
107     \pgfplotsset{freq@filter/.style = {x filter/.expression={x-
log10(2*pi)}}}
108   \fi
109 },
110 frequency unit/rad/.code={
111   \renewcommand{\freq@scale}{1}
112   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (rad/s)}}}
113 }
114 }

```

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

```

get@interval@end 115 \def\get@interval@start#1:#2\@nil{#1}
116 \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 117 \newcommand*\MagK[2]{(20*log10(abs(#1)))}
\PhKLin 118 \newcommand*\MagKAsymp{\MagK}
119 \newcommand*\MagKLin{\MagK}
120 \newcommand*\PhK[2]{((#1<0?-pi:0)*\ph@scale)}
121 \newcommand*\PhKAsymp{\PhK}
122 \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.

```

123 \newcommand*\MagDel[2]{0}
124 \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 125 \newcommand*\MagPole[2]
\PhPole 126 {(-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 127 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
128 -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :
129 -20*log10(t)
130 )}

```

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

```

131 \newcommand*\MagPoleAsymp{\MagPoleLin}

```

Parametric function for the true phase of a complex pole.

```

132 \newcommand*\PhPole[2]{((#1 > 0 ? (#2 > 0 ?
133 (\n@mod@p{-atan2((t - (#2)), -(#1))}{2*pi}) :
134 (-atan2((t - (#2)), -(#1)))) :
135 (-atan2((t - (#2)), -(#1))))*\ph@scale)}

```

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

```

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

```

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

```

149 \newcommand*\PhPoleAsymp}[2]{((t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) ?
150 (-atan2(-(#2), -(#1))) :
151 (#2>0? (#1>0?3*pi/2:-pi/2):-pi/2))*\ph@scale)}

```

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

```

\MagZeroAsymp
\MagZeroLin 152 \newcommand*\MagZero}{0-\MagPole}
\PhZero 153 \newcommand*\MagZeroLin}{0-\MagPoleLin}
\PhZeroAsymp 154 \newcommand*\MagZeroAsymp}{0-\MagPoleAsymp}
\PhZeroLin 155 \newcommand*\PhZero}{0-\PhPole}
156 \newcommand*\PhZeroLin}{0-\PhPoleLin}
157 \newcommand*\PhZeroAsymp}{0-\PhPoleAsymp}

```

4.3 Second order systems.

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

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

```

\MagCSPolesAsymp
\MagCSPolesLin 158 \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles 159 - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2}))))}
\PhCSPolesAsymp 160 \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : - 40*log10(t))}
\PhCSPolesLin 161 \newcommand*\MagCSPolesAsymp}{\MagCSPolesLin}

```

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

```

\MagCSZerosAsymp
\MagCSZerosLin 162 \newcommand*\PhCSPoles}[2]{((-atan2((2*(#1)*(#2)*t), (\n@pow{#2}{2}
\PhCSZeros 163 - \n@pow{t}{2}))))*\ph@scale)}
\PhCSZerosAsymp 164 \newcommand*\PhCSPolesLin}[2]{((t < (#2 / (\n@pow{10}{abs(#1)})) ?
\PhCSZerosLin 165 0 :
166 (t >= (#2 * (\n@pow{10}{abs(#1)})) ?
167 (#1>0 ? -pi : pi) :
168 (#1>0 ? (-pi*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
169 (pi*(log10(t*(\n@pow{10}{abs(#1)})/#2))/(2*abs(#1)))))*\ph@scale)}
170 \newcommand*\PhCSPolesAsymp}[2]{((#1>0?(t<#2?0:-
pi):(t<#2?0:pi))*\ph@scale)}

```

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

```

171 \newcommand*\MagCSZeros}{0-\MagCSPoles}
172 \newcommand*\MagCSZerosLin}{0-\MagCSPolesLin}
173 \newcommand*\MagCSZerosAsymp}{0-\MagCSPolesAsymp}
174 \newcommand*\PhCSZeros}{0-\PhCSPoles}
175 \newcommand*\PhCSZerosLin}{0-\PhCSPolesLin}
176 \newcommand*\PhCSZerosAsymp}{0-\PhCSPolesAsymp}

```

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

```

177 \newcommand*\MagCSPolesPeak}[3][]{
178   \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
179   (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
180 }
181 \newcommand*\MagCSZerosPeak}[3][]{
182   \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
183   (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
184 }

```

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

```

\MagSOPolesLin 185 \newcommand*\MagSOPoles}[2]{
\PhSOPoles     186   (-20*log10(sqrt(\n@pow{#2} - \n@pow{t}{2}){2} + \n@pow{#1*t}{2})))}
\PhSOPolesAsymp 187 \newcommand*\MagSOPolesLin}[2]{
\PhSOPolesLin  188   (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : -40*log10(t))}
\MagSOPolesLin 189 \newcommand*\MagSOPolesAsymp}{\MagSOPolesLin}

```

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

```

\PhSOPolesLin 190 \newcommand*\PhSOPoles}[2]{((-atan2((#1)*t,((#2) - \n@pow{t}{2}))) * \ph@scale)}
\PhSOPolesAsymp 191 \newcommand*\PhSOPolesLin}[2]{((#2>0 ?
\PhSOPolesLin  192   \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
193   (#1>0 ? -pi : pi))}
194 \newcommand*\PhSOPolesAsymp}[2]{((#2>0 ?
195   \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
196   (#1>0 ? -pi : pi))}

```

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

```

197 \newcommand*\MagSOPoles}{0-\MagSOPoles}
198 \newcommand*\MagSOPolesLin}{0-\MagSOPolesLin}
199 \newcommand*\MagSOPolesAsymp}{0-\MagSOPolesAsymp}
200 \newcommand*\PhSOPoles}{0-\PhSOPoles}
201 \newcommand*\PhSOPolesLin}{0-\PhSOPolesLin}
202 \newcommand*\PhSOPolesAsymp}{0-\PhSOPolesAsymp}

```

`\MagSOPolesPeak` `\MagSOPolesZerosPeak` 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.

```

203 \newcommand*\MagSOPolesPeak}[3][]{
204   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
205   (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
206     20*log10(abs(#2/sqrt(abs(#3))))});
207 }
208 \newcommand*\MagSOPolesZerosPeak}[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 }

```

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`.

```
213 \newcommand{\BodeZPK}[4][approx>true]{
```

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

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

```
231 \edef\temp@mag@cmd{\noexpand\nextgroupplot [ylabel={Gain (dB)}, xmajor ticks=false,
232 \noexpand\addplot [freq@filter, variable=t, thick, \optmag@plot]}
233 \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
234 \noexpand\addplot [freq@filter, variable=t, thick, \optph@plot]}
235 \if@pgfarg
236 \temp@mag@cmd {\func@mag};
237 \optmag@commands
238 \temp@ph@cmd {\func@ph};
239 \optph@commands
240 \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.

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

```

258         set xrange [#3*\freq@scale:#4*\freq@scale];
259         set samples \pgfkeysvalueof{/pgfplots/samples};
260         plot \func@ph;
261         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
262         plot "$meta" using ($1/(\freq@scale)):($2);
263     };
264     \optph@commands
265     \fi
266 \end{groupplot}
267 \end{tikzpicture}
268 }

```

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

```

269 \AtBeginDocument{
270   \if@babel@french
271     \let\Orig@BodeZPK\BodeZPK
272     \renewcommand{\BodeZPK}{%
273       \shorthandoff{;:!?}%
274       \BodeZPK@Shorthandoff
275     }
276     \newcommand{\BodeZPK@Shorthandoff}[4][]{%
277       \Orig@BodeZPK[#1]{#2}{#3}{#4}%
278       \shorthandon{;:!?}%
279     }
280   \fi
281 }

```

\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.

```

282 \newcommand{\BodeTF}[4][]{
283   \parse@opt{#1}
284   \gdef\func@mag{}
285   \gdef\func@ph{}
286   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
287   \temp@cmd
288   \build@TF@plot{\func@mag}{\func@ph}{#2}
289   \edef\temp@cmd{\noexpand\begin{groupplot}[
290     bode@style,
291     xmin=#3,
292     xmax=#4,
293     domain=#3*\freq@scale:#4*\freq@scale,
294     height=2.5cm,
295     xmode=log,
296     group style = {group size = 1 by 2,vertical sep=0.25cm},
297     \opt@group
298   ]}
299   \temp@cmd
300   \edef\temp@mag@cmd{\noexpand\nextgroupplot [ylabel={Gain (dB)}, xmajor ticks=false,
301     \noexpand\addplot [freq@filter, variable=t, thick, \optmag@plot]}
302   \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
303     \noexpand\addplot [freq@filter, variable=t, thick, \optph@plot]}
304   \if@pgfarg
305     \temp@mag@cmd {\func@mag};
306     \optmag@commands
307     \temp@ph@cmd {\n@mod{\func@ph}{2*pi*\ph@scale}};
308     \optph@commands
309   \else
310     \stepcounter{gnuplot@id}
311     \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
312     { set table $meta;
313       set dummy t;
314       set logscale x 10;
315       set xrange [#3*\freq@scale:#4*\freq@scale];

```



```

316         set samples \pgfkeysvalueof{/pgfplots/samples};
317         plot \func@mag;
318         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
319         plot "$meta" using ($1/(\freq@scale)):($2);
320     };
321     \optmag@commands
322     \stepcounter{gnuplot@id}
323     \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
324     { set table $meta;
325       set dummy t;
326       set logscale x 10;
327       set trange [#3*\freq@scale:#4*\freq@scale];
328       set samples \pgfkeysvalueof{/pgfplots/samples};
329       plot '+' using (t) : ((\func@ph)/(\ph@scale)) smooth unwrap;
330       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
331       plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
332     };
333     \optph@commands
334     \fi
335     \end{groupplot}
336 \end{tikzpicture}
337 }

```

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

```

338 \AtBeginDocument{
339   \if@babel@french
340     \let\Orig@BodeTF\BodeTF
341     \renewcommand{\BodeTF}{%
342       \shorthandoff{;:!?}%
343       \BodeTF@Shorthandoff
344     }
345     \newcommand{\BodeTF@Shorthandoff}[4][[]]{%
346       \Orig@BodeTF[#1]{#2}{#3}{#4}%
347       \shorthandon{;:!?}%
348     }
349   \fi
350 }

```

`\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.

```

351 \newcommand{\addBodeZPKPlots}[3][true/{}]{
352   \foreach \approx/\opt in {#1} {
353     \gdef\plot@macro{
354       \gdef\temp@macro{
355         \ifnum\pdf@strcmp{#2}{phase}=0
356           \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}
357         \else
358           \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}
359         \fi
360         \if@pgfarg
361           \edef\temp@cmd{\noexpand\addplot [freq@filter, domain=\freq@scale*\pgfkeysvalueof{/
362             \temp@cmd {\plot@macro};
363         \else
364           \stepcounter{gnuplot@id}
365           \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \opt]}
366           \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
367           { set table $meta;
368             set dummy t;
369             set logscale x 10;
370             set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
371             set samples \pgfkeysvalueof{/pgfplots/samples};
372             plot \plot@macro;
373             set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
374             plot "$meta" using ($1/(\freq@scale)):($2);

```

```

375     };
376     \fi
377 }
378 }

```

`\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.

```

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

```

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

```

422 \newcommand{\addBodeComponentPlot}[2][thick]{
423   \if@pgfarg
424     \edef\temp@cmd{\noexpand\addplot [freq@filter, domain=\freq@scale*\pgfkeysvalueof{/pg
425     \temp@cmd {#2}};
426   \else
427     \stepcounter{gnuplot@id}
428     \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
429     { set table $meta;
430       set dummy t;

```

```

431     set logscale x 10;
432     set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
433     set samples \pgfkeysvalueof{/pgfplots/samples};
434     plot #2;
435     set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
436     plot "$meta" using ($1/(\freq@scale)):($2);
437   };
438 \fi
439 }

```

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.

```

440 \AtBeginDocument{
441   \if@babel@french
442     \AddToHook{env/BodePhPlot/begin}{\shorthandoff{;:;!}}
443   \fi
444 }
445 \NewDocumentEnvironment{BodePhPlot}{0}{mm+b}{
446   \parse@env@opt{#1}
447   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
448   \temp@cmd
449   \edef\temp@cmd{\noexpand\begin{semilogaxis}[
450     ph@y@label,
451     freq@label,
452     bode@style,
453     xmin={#2},
454     xmax={#3},
455     domain=#2:#3,
456     height=2.5cm,
457     \unexpanded\expandafter{\opt@axes}
458   ]}
459   \temp@cmd
460   #4
461   \end{semilogaxis}
462 \end{tikzpicture}
463 }{}

```

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.

```

464 \AtBeginDocument{
465   \if@babel@french
466     \AddToHook{env/BodeMagPlot/begin}{\shorthandoff{;:;!}}
467   \fi
468 }
469 \NewDocumentEnvironment{BodeMagPlot}{0}{mm+b}{
470   \parse@env@opt{#1}
471   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
472   \temp@cmd
473   \edef\temp@cmd{\noexpand\begin{semilogaxis}[
474     bode@style,
475     freq@label,
476     xmin={#2},
477     xmax={#3},
478     domain=#2:#3,
479     height=2.5cm,
480     ylabel={Gain (dB)},
481     \unexpanded\expandafter{\opt@axes}
482   ]}
483   \temp@cmd
484   #4
485   \end{semilogaxis}

```

```

486 \end{tikzpicture}
487 }{}

```

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

```

488 \AtBeginDocument{
489   \if@babel@french
490     \AddToHook{env/BodePlot/begin}{\shorthandoff{;:!?}}
491   \fi
492 }
493 \NewDocumentEnvironment{BodePlot}{0}{mm+b}{
494   \parse@env@opt{#1}
495   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
496   \temp@cmd
497   \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
498     bode@style,
499     freq@label,
500     xmin={#2},
501     xmax={#3},
502     domain=#2:#3,
503     height=2.5cm,
504     \unexpanded\expandafter{\opt@axes}
505   ]}
506   \temp@cmd
507   #4
508   \end{semilogxaxis}
509   \end{tikzpicture}
510 }{}

```

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](#).

```

511 \newcommand*\add@feature}[3]{
512   \ifcat$\detokenize\expandafter{#1}$
513     \xdef#1{\unexpanded\expandafter{#1 0+#2}}
514   \else
515     \xdef#1{\unexpanded\expandafter{#1+#2}}
516   \fi
517   \foreach \y [count=\n] in #3 {
518     \xdef#1{\unexpanded\expandafter{#1}{\y}}
519     \xdef\Last@LoopValue{\n}
520   }
521   \ifnum\Last@LoopValue=1
522     \xdef#1{\unexpanded\expandafter{#1}{0}}
523   \fi
524 }

```

\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.

```

525 \newcommand{\build@ZPK@plot}[4]{
526   \foreach \feature/\values in {#4} {
527     \ifnum\pdf@strcmp{\feature}{z}=0

```

```

528     \foreach \z in \values {
529         \ifnum\pdf@strcmp{#3}{linear}=0
530             \add@feature{#2}{\PhZeroLin}{\z}
531             \add@feature{#1}{\MagZeroLin}{\z}
532         \else
533             \ifnum\pdf@strcmp{#3}{asymptotic}=0
534                 \add@feature{#2}{\PhZeroAsymp}{\z}
535                 \add@feature{#1}{\MagZeroAsymp}{\z}
536             \else
537                 \add@feature{#2}{\PhZero}{\z}
538                 \add@feature{#1}{\MagZero}{\z}
539             \fi
540         \fi
541     }
542 \fi
543 \ifnum\pdf@strcmp{\feature}{p}=0
544     \foreach \p in \values {
545         \ifnum\pdf@strcmp{#3}{linear}=0
546             \add@feature{#2}{\PhPoleLin}{\p}
547             \add@feature{#1}{\MagPoleLin}{\p}
548         \else
549             \ifnum\pdf@strcmp{#3}{asymptotic}=0
550                 \add@feature{#2}{\PhPoleAsymp}{\p}
551                 \add@feature{#1}{\MagPoleAsymp}{\p}
552             \else
553                 \add@feature{#2}{\PhPole}{\p}
554                 \add@feature{#1}{\MagPole}{\p}
555             \fi
556         \fi
557     }
558 \fi
559 \ifnum\pdf@strcmp{\feature}{k}=0
560     \ifnum\pdf@strcmp{#3}{linear}=0
561         \add@feature{#2}{\PhKLin}{\values}
562         \add@feature{#1}{\MagKLin}{\values}
563     \else
564         \ifnum\pdf@strcmp{#3}{asymptotic}=0
565             \add@feature{#2}{\PhKAsymp}{\values}
566             \add@feature{#1}{\MagKAsymp}{\values}
567         \else
568             \add@feature{#2}{\PhK}{\values}
569             \add@feature{#1}{\MagK}{\values}
570         \fi
571     \fi
572 \fi
573 \ifnum\pdf@strcmp{\feature}{d}=0
574     \ifnum\pdf@strcmp{#3}{linear}=0
575         \PackageError {bodeplot} {Linear approximation for pure delays is not
576             supported.} {Plot the true Bode plot using `true' instead of `linear'.}
577     \else
578         \ifnum\pdf@strcmp{#3}{asymptotic}=0
579             \PackageError {bodeplot} {Asymptotic approximation for pure delays is not
580             supported.} {Plot the true Bode plot using `true' instead of `asymptotic'.}
581         \else
582             \ifdim\values pt < 0pt
583                 \PackageError {bodeplot} {Delay needs to be a positive number.}
584             \fi
585             \add@feature{#2}{\PhDel}{\values}
586             \add@feature{#1}{\MagDel}{\values}
587         \fi
588     \fi
589 \fi
590 }

```

591 }

`\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).

```
592 \newcommand{\build@TF@plot}[3]{
593   \gdef\num@real{0}
594   \gdef\num@im{0}
595   \gdef\den@real{0}
596   \gdef\den@im{0}
597   \gdef\loop@delay{0}
598   \foreach \feature/\values in {#3} {
599     \ifnum\pdf@strcmp{\feature}{num}=0
600       \foreach \numcoeff [count=\numpow] in \values {
601         \xdef\num@degree{\numpow}
602       }
603       \foreach \numcoeff [count=\numpow] in \values {
604         \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}
605         \ifnum\currentdegree = 0
606           \xdef\num@real{\num@real+\numcoeff}
607         \else
608           \ifodd\currentdegree
609             \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-1}{(\currentdegree-
610 1)/2})*)%
611               (\n@pow{t}{\currentdegree}))}
612           \else
613             \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-
614 1}{(\currentdegree)/2})*)%
615               (\n@pow{t}{\currentdegree}))}
616           \fi
617         \fi
618       }
619       \ifnum\pdf@strcmp{\feature}{den}=0
620         \foreach \dencoeff [count=\denpow] in \values {
621           \xdef\den@degree{\denpow}
622         }
623         \foreach \dencoeff [count=\denpow] in \values {
624           \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}
625           \ifnum\currentdegree = 0
626             \xdef\den@real{\den@real+\dencoeff}
627           \else
628             \ifodd\currentdegree
629               \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-1}{(\currentdegree-
630 1)/2})*)%
631                 (\n@pow{t}{\currentdegree}))}
632             \else
633               \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-
634 1}{(\currentdegree)/2})*)%
635                 (\n@pow{t}{\currentdegree}))}
636             \fi
637           \fi
638         }
639       \xdef\loop@delay{\values}
640     }
641   \xdef#2{((atan2((\num@im),(\num@real))-atan2((\den@im),%
642     (\den@real))-\loop@delay*t)*(\ph@scale))}
643   \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2})))-%
644     20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2})))}
```

`\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.

```

646 \newcommand{\parse@opt}[1]{
647   \gdef\optmag@axes{}
648   \gdef\optph@axes{}
649   \gdef\optph@plot{}
650   \gdef\optmag@plot{}
651   \gdef\opt@group{}
652   \gdef\opt@approx{}
653   \gdef\optph@commands{}
654   \gdef\optmag@commands{}
655   \gdef\opt@tikz{}
656   \foreach \obj/\typ/\opt in {#1} {
657     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
658       \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
659         \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
660       \else
661         \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
662           \xdef\optph@plot{\unexpanded\expandafter{\opt}}
663         \else
664           \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
665           \xdef\optph@plot{\unexpanded\expandafter{\opt}}
666         \fi
667       \fi
668     \else
669       \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
670         \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
671           \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
672         \else
673           \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
674             \xdef\optph@axes{\unexpanded\expandafter{\opt}}
675           \else
676             \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
677             \xdef\optph@axes{\unexpanded\expandafter{\opt}}
678           \fi
679         \fi
680       \else
681         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{group}=0
682           \xdef\opt@group{\unexpanded\expandafter{\opt}}
683         \else
684           \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{approx}=0
685             \xdef\opt@approx{\unexpanded\expandafter{\opt}}
686           \else
687             \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
688               \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
689                 \xdef\optph@commands{\unexpanded\expandafter{\opt}}
690               \else
691                 \xdef\optmag@commands{\unexpanded\expandafter{\opt}}
692               \fi
693             \else
694               \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
695                 \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
696               \else
697                 \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
698                   \unexpanded\expandafter{\obj}}

```

```

699             \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
700             \unexpanded\expandafter{\obj}}
701         \fi
702     \fi
703 \fi
704 \fi
705 \fi
706 \fi
707 }
708 }

```

`\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.

```

709 \newcommand{\parse@env@opt}[1]{
710   \gdef\opt@axes{}
711   \gdef\opt@tikz{}
712   \foreach \obj/\opt in {#1} {
713     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
714       \xdef\opt@axes{\unexpanded\expandafter{\opt}}
715     \else
716       \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
717         \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
718       \else
719         \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
720         \unexpanded\expandafter{\obj}}
721       \fi
722     \fi
723   }
724 }

```

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!

```

725 \newcommand{\NyquistZPK}[4][ ]{
726   \parse@N@opt{#1}
727   \gdef\func@mag{}
728   \gdef\func@ph{}
729   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
730   \temp@cmd
731   \build@ZPK@plot{\func@mag}{\func@ph}{#2}
732   \edef\temp@cmd{\noexpand\begin{axis}[
733     bode@style,
734     domain=#3*\freq@scale:#4*\freq@scale,
735     height=5cm,
736     xlabel={\Re$},
737     ylabel={\Im$},
738     samples=500,
739     \unexpanded\expandafter{\opt@axes}
740   ]}

```



```

741 \temp@cmd
742 \addplot [only marks,mark=+,thick,red] (-1 , 0);
743 \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \unexpanded\expandafter{\opt@p
744 \if@pgfarg
745 \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
746 {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
747 \opt@commands
748 \else
749 \stepcounter{gnuplot@id}
750 \temp@cmd gnuplot [parametric, gnuplot@prefix] {
751 \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
752 \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
753 };
754 \opt@commands
755 \fi
756 \end{axis}
757 \end{tikzpicture}
758 }

```

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

```

759 \AtBeginDocument{
760 \if@babel@french
761 \let\Orig@NyquistZPK\NyquistZPK
762 \renewcommand{\NyquistZPK}{%
763 \shorthandoff{;:!?}%
764 \NyquistZPK@Shorthandoff
765 }
766 \newcommand{\NyquistZPK@Shorthandoff}[4][]{%
767 \Orig@NyquistZPK[#1]{#2}{#3}{#4}%
768 \shorthandon{;:!?}%
769 }
770 \fi
771 }

```

`\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`.

```

772 \newcommand{\NyquistTF}[4][]{
773 \parse@N@opt{#1}
774 \gdef\func@mag{}
775 \gdef\func@ph{}
776 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
777 \temp@cmd
778 \build@TF@plot{\func@mag}{\func@ph}{#2}
779 \edef\temp@cmd{\noexpand\begin{axis}[
780 bode@style,
781 domain=#3*\freq@scale:#4*\freq@scale,
782 height=5cm,
783 xlabel={\Re$},
784 ylabel={\Im$},
785 samples=500,
786 \unexpanded\expandafter{\opt@axes}
787 ]}
788 \temp@cmd
789 \addplot [only marks, mark=+, thick, red] (-1 , 0);
790 \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \unexpanded\expandafter{\opt@p
791 \if@pgfarg
792 \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
793 {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
794 \opt@commands
795 \else
796 \stepcounter{gnuplot@id}
797 \temp@cmd gnuplot [parametric, gnuplot@prefix] {
798 \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),

```

```

799         \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
800     };
801     \opt@commands
802     \fi
803     \end{axis}
804 \end{tikzpicture}
805 }

```

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

```

806 \AtBeginDocument{
807   \if@babel@french
808     \let\Orig@NyquistTF\NyquistTF
809     \renewcommand{\NyquistTF}{%
810       \shorthandoff{;:!?}%
811       \NyquistTF@Shorthandoff
812     }
813     \newcommand{\NyquistTF@Shorthandoff}[4][]{%
814       \Orig@NyquistTF[#1]{#2}{#3}{#4}%
815       \shorthandon{;:!?}%
816     }
817   \fi
818 }

```

\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.

```

819 \newcommand{\addNyquistZPKPlot}[2][]{
820   \gdef\func@mag{}
821   \gdef\func@ph{}
822   \build@ZPK@plot{\func@mag}{\func@ph}{#2}
823   \if@pgfarg
824     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
825       \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
826         {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
827   \else
828     \stepcounter{gnuplot@id}
829     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
830       \temp@cmd gnuplot [parametric, gnuplot@prefix] {
831         \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
832         \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
833       };
834   \fi
835 }

```

\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.

```

836 \newcommand{\addNyquistTFPlot}[2][]{
837   \gdef\func@mag{}
838   \gdef\func@ph{}
839   \build@TF@plot{\func@mag}{\func@ph}{#2}
840   \if@pgfarg
841     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
842       \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
843         {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
844   \else
845     \stepcounter{gnuplot@id}
846     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
847       \temp@cmd gnuplot [parametric, gnuplot@prefix]{
848         \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
849         \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))

```

```

850 };
851 \fi
852 }

```

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.

```

853 \AtBeginDocument{
854   \if@babel@french
855     \AddToHook{env/NyquistPlot/begin}{\shorthandoff{;:;!}}
856   \fi
857 }
858 \NewDocumentEnvironment{NyquistPlot}{0}{mm+b}{
859   \parse@env@opt{#1}
860   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
861   \temp@cmd
862   \edef\temp@cmd{\noexpand\begin{axis}[
863     bode@style,
864     height=5cm,
865     domain=#2:#3,
866     xlabel={\$Re\$},
867     ylabel={\$Im\$},
868     \unexpanded\expandafter{\opt@axes}
869   ]}
870   \temp@cmd
871   \addplot [only marks,mark=+,thick,red] (-1 , 0);
872   #4
873   \end{axis}
874   \end{tikzpicture}
875 }{}

```

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.

```

876 \newcommand{\parse@N@opt}[1]{
877   \gdef\opt@axes{}
878   \gdef\opt@plot{}
879   \gdef\opt@commands{}
880   \gdef\opt@tikz{}
881   \foreach \obj/\opt in {#1} {
882     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
883       \xdef\opt@axes{\unexpanded\expandafter{\opt}}
884     \else
885       \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
886         \xdef\opt@plot{\unexpanded\expandafter{\opt}}
887       \else
888         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
889           \xdef\opt@commands{\unexpanded\expandafter{\opt}}
890         \else
891           \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
892             \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
893           \else
894             \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
895               \unexpanded\expandafter{\obj}}
896           \fi
897         \fi
898       \fi
899     \fi
900   }

```

901 }

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 902 \newcommand{\NicholsZPK}[4][[]]{
\addNicholsZPKChart 903 \parse@N@opt{#1}
\addNicholsTFChart 904 \gdef\func@mag{}
905 \gdef\func@ph{}
906 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
907 \temp@cmd
908 \build@ZPK@plot{\func@mag}{\func@ph}{{#2}}
909 \edef\temp@cmd{\noexpand\begin{axis}[
910 ph@x@label,
911 bode@style,
912 domain=#3*\freq@scale:#4*\freq@scale,
913 height=5cm,
914 ylabel={Gain (dB)},
915 samples=500,
916 \unexpanded\expandafter{\opt@axes}
917 ]}
918 \temp@cmd
919 \edef\temp@cmd{\noexpand\addplot [variable=t,thick,\opt@plot]}
920 \if@pgfarg
921 \temp@cmd ( {\func@ph} , {\func@mag} );
922 \opt@commands
923 \else
924 \stepcounter{gnuplot@id}
925 \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
926 { set table $meta;
927 set logscale x 10;
928 set dummy t;
929 set samples \pgfkeysvalueof{/pgfplots/samples};
930 set trange [#3*\freq@scale:#4*\freq@scale];
931 plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
932 unset logscale x;
933 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
934 plot "$meta" using ($2*\ph@scale):($1);
935 };
936 \opt@commands
937 \fi
938 \end{axis}
939 \end{tikzpicture}
940 }
941 \AtBeginDocument{
942 \if@babel@french
943 \let\Orig@NicholsZPK\NicholsZPK
944 \renewcommand{\NicholsZPK}{%
945 \shorthandoff{;:!?}%
946 \NicholsZPK@Shorthandoff
947 }
948 \newcommand{\NicholsZPK@Shorthandoff}[4][[]]{%
949 \Orig@NicholsZPK[#1]{#2}{#3}{#4}%
950 \shorthandon{;:!?}%
951 }
952 \fi
953 }
954 \newcommand{\NicholsTF}[4][[]]{
955 \parse@N@opt{#1}
956 \gdef\func@mag{}
957 \gdef\func@ph{}
958 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}

```

```

959 \temp@cmd
960   \build@TF@plot{\func@mag}{\func@ph}{#2}
961   \edef\temp@cmd{\noexpand\begin{axis}[
962     ph@x@label,
963     bode@style,
964     domain=#3*\freq@scale:#4*\freq@scale,
965     height=5cm,
966     ylabel={Gain (dB)},
967     samples=500,
968     \unexpanded\expandafter{\opt@axes}
969   ]}
970 \temp@cmd
971   \edef\temp@cmd{\noexpand\addplot [variable=t,thick, \opt@plot]}
972   \if@pgfarg
973     \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
974     \opt@commands
975   \else
976     \stepcounter{gnuplot@id}
977     \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
978     { set table $metal;
979       set logscale x 10;
980       set dummy t;
981       set samples \pgfkeysvalueof{/pgfplots/samples};
982       set trange [#3*\freq@scale:#4*\freq@scale];
983       plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
984       unset logscale x;
985       set table $meta2;
986       plot "$metal" using ($1):($2) smooth unwrap;
987       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
988       plot "$meta2" using ($2*\ph@scale):($1);
989     };
990     \opt@commands
991   \fi
992 \end{axis}
993 \end{tikzpicture}
994 }
995 \AtBeginDocument{
996   \if@babel@french
997     \let\Orig@NicholsTF\NicholsTF
998     \renewcommand{\NicholsTF}{%
999       \shorthandoff{;:!?}%
1000     \NicholsTF@Shorthandoff
1001   }
1002   \newcommand{\NicholsTF@Shorthandoff}[4][[%
1003     \Orig@NicholsTF[#1]{#2}{#3}{#4}%
1004     \shorthandon{;:!?}%
1005   ]}
1006   \AddToHook{env/NicholsChart/begin}{\shorthandoff{;:!?}}
1007 \fi
1008 }
1009 \NewDocumentEnvironment{NicholsChart}{0}{mm+b}{
1010   \parse@env@opt{#1}
1011   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
1012   \temp@cmd
1013     \edef\temp@cmd{\noexpand\begin{axis}[
1014       ph@x@label,
1015       bode@style,
1016       domain=#2:#3,
1017       height=5cm,
1018       ylabel={Gain (dB)},
1019       \unexpanded\expandafter{\opt@axes}
1020     ]}
1021   \temp@cmd

```

```

1022     #4
1023     \end{axis}
1024 \end{tikzpicture}
1025 }{}
1026 \newcommand{\addNicholsZPKChart}[2][]{
1027   \gdef\func@mag{}
1028   \gdef\func@ph{}
1029   \build@ZPK@plot{\func@mag}{\func@ph}{#2}
1030   \if@pgfarg
1031     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
1032       \temp@cmd ( {\func@ph} , {\func@mag} );
1033   \else
1034     \stepcounter{gnuplot@id}
1035     \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1036     { set table $meta;
1037       set logscale x 10;
1038       set dummy t;
1039       set samples \pgfkeysvalueof{/pgfplots/samples};
1040       set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1041       plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1042       unset logscale x;
1043       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1044       plot "$meta" using ($2*\ph@scale):($1);
1045     };
1046   \fi
1047 }
1048 \newcommand{\addNicholsTFChart}[2][]{
1049   \gdef\func@mag{}
1050   \gdef\func@ph{}
1051   \build@TF@plot{\func@mag}{\func@ph}{#2}
1052   \if@pgfarg
1053     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}
1054       \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
1055   \else
1056     \stepcounter{gnuplot@id}
1057     \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1058     { set table $meta1;
1059       set logscale x 10;
1060       set dummy t;
1061       set samples \pgfkeysvalueof{/pgfplots/samples};
1062       set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1063       plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1064       unset logscale x;
1065       set table $meta2;
1066       plot "$meta1" using ($1):($2) smooth unwrap;
1067       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1068       plot "$meta2" using ($2*\ph@scale):($1);
1069     };
1070   \fi
1071 }

```

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 | <code>\@declutterargtrue</code> ... 7 | <code>\@pgfargfalse</code> 1 |
| <code>\@babel@frenchfalse</code> .. | 47 | <code>\@hzargfalse</code> 13 | <code>\@pgfargtrue</code> 3 |
| <code>\@babel@frenchtrue</code> .. | 50 | <code>\@hzargtrue</code> 15 | <code>\@radargfalse</code> 9 |
| <code>\@declutterargfalse</code> .. | 5 | <code>\@nil</code> 115, 116 | <code>\@radargtrue</code> 11 |

| | | | |
|---|--------------------|---------------------|--|
| A | BodePlot | 488 | 687 , 688 , 694 , 713 , 716 , 882 , 885 , 888 , 891 |
| \add@feature | | | |
| 511 , 530 , 531 , 534 , | | | |
| 535 , 537 , 538 , 546 , | | | |
| 547 , 550 , 551 , 553 , | | | |
| 554 , 561 , 562 , 565 , | | | |
| 566 , 568 , 569 , 585 , 586 | | | |
| \addBodeComponentPlot | | | |
| 422 | | | |
| \addBodeTFPlot | | | |
| 379 | | | |
| \addBodeZPKPlots | | | |
| 351 | | | |
| \addNicholsTFChart | | | |
| 902 | | | |
| \addNicholsZPKChart | | | |
| 902 | | | |
| \addNyquistTFPlot | | | |
| 836 | | | |
| \addNyquistZPKPlot | | | |
| 819 | | | |
| \AddToHook | | | |
| 442 , | | | |
| 466 , 490 , 855 , 1006 | | | |
| \AtBeginDocument | | | |
| 48 , 269 , | | | |
| 338 , 440 , 464 , 488 , | | | |
| 759 , 806 , 853 , 941 , 995 | | | |
| B | | | |
| \bode@style | | | |
| 53 | | | |
| BodeMagPlot (env.) | | | |
| 464 | | | |
| BodePhPlot (env.) | | | |
| 440 | | | |
| BodePlot (env.) | | | |
| 488 | | | |
| \bodeplot@prefix | | | |
| 31 , 33 , 38 , | | | |
| 249 , 261 , 318 , 330 , | | | |
| 373 , 405 , 416 , 435 , | | | |
| 933 , 987 , 1043 , 1067 | | | |
| \BodeTF | | | |
| 282 | | | |
| \BodeTF@Shorthandoff | | | |
| 343 , 345 | | | |
| \BodeZPK | | | |
| 213 | | | |
| \BodeZPK@Shorthandoff | | | |
| 274 , 276 | | | |
| \build@TF@plot | | | |
| 288 , 383 , 385 , 592 , | | | |
| 778 , 839 , 960 , 1051 | | | |
| \build@ZPK@plot | | | |
| 219 , 356 , 358 , 525 , | | | |
| 731 , 822 , 908 , 1029 | | | |
| C | | | |
| \currentdegree | | | |
| 604 , | | | |
| 605 , 608 , 609 , 610 , | | | |
| 612 , 613 , 623 , 624 , | | | |
| 627 , 628 , 629 , 631 , 632 | | | |
| D | | | |
| \den@degree | | | |
| 620 , 623 | | | |
| \den@im | | | |
| 596 , 628 , 641 , 644 | | | |
| \den@real | | | |
| 595 , | | | |
| 625 , 631 , 642 , 644 | | | |
| \dencoeff | | | |
| 619 , | | | |
| 622 , 625 , 628 , 631 | | | |
| \denpow | | | |
| 619 , 620 , 622 , 623 | | | |
| E | | | |
| environments: | | | |
| BodeMagPlot | | | |
| 464 | | | |
| BodePhPlot | | | |
| 440 | | | |
| F | | | |
| \frenchbsetup | | | |
| 49 | | | |
| \freq@filter | | | |
| 68 | | | |
| \freq@label | | | |
| 68 | | | |
| \freq@scale | | | |
| 68 , | | | |
| 69 , 80 , 104 , 111 , | | | |
| 224 , 246 , 250 , 258 , | | | |
| 262 , 293 , 315 , 319 , | | | |
| 327 , 331 , 361 , 370 , | | | |
| 374 , 389 , 392 , 402 , | | | |
| 406 , 413 , 417 , 424 , | | | |
| 432 , 436 , 734 , 781 , | | | |
| 824 , 829 , 841 , 846 , | | | |
| 912 , 930 , 964 , 982 , | | | |
| 1031 , 1040 , 1053 , 1062 | | | |
| \func@mag | | | |
| 215 , 219 , | | | |
| 236 , 248 , 284 , 288 , | | | |
| 305 , 317 , 727 , 731 , | | | |
| 745 , 746 , 751 , 752 , | | | |
| 774 , 778 , 792 , 793 , | | | |
| 798 , 799 , 820 , 822 , | | | |
| 825 , 826 , 831 , 832 , | | | |
| 837 , 839 , 842 , 843 , | | | |
| 848 , 849 , 904 , 908 , | | | |
| 921 , 931 , 956 , 960 , | | | |
| 973 , 983 , 1027 , | | | |
| 1029 , 1032 , 1041 , | | | |
| 1049 , 1051 , 1054 , 1063 | | | |
| \func@ph | | | |
| 216 , 219 , | | | |
| 238 , 260 , 285 , 288 , | | | |
| 307 , 329 , 728 , 731 , | | | |
| 745 , 746 , 751 , 752 , | | | |
| 775 , 778 , 792 , 793 , | | | |
| 798 , 799 , 821 , 822 , | | | |
| 825 , 826 , 831 , 832 , | | | |
| 838 , 839 , 842 , 843 , | | | |
| 848 , 849 , 905 , 908 , | | | |
| 921 , 931 , 957 , 960 , | | | |
| 973 , 983 , 1028 , | | | |
| 1029 , 1032 , 1041 , | | | |
| 1050 , 1051 , 1054 , 1063 | | | |
| G | | | |
| \get@interval@end 115 , 116 | | | |
| \get@interval@start | | | |
| 115 , 115 | | | |
| \gnuplot@id | | | |
| 1 | | | |
| \gnuplot@prefix | | | |
| 1 | | | |
| I | | | |
| \if@babel@french | | | |
| 47 | | | |
| \if@hzarg | | | |
| 13 , 79 | | | |
| \ifcat | | | |
| 512 | | | |
| \ifdefined | | | |
| 49 | | | |
| \ifnum | | | |
| 355 , 382 , | | | |
| 388 , 397 , 521 , 527 , | | | |
| 529 , 533 , 543 , 545 , | | | |
| 549 , 559 , 560 , 564 , | | | |
| 573 , 574 , 578 , 599 , | | | |
| 605 , 618 , 624 , 637 , | | | |
| 657 , 658 , 661 , 669 , | | | |
| 670 , 673 , 681 , 684 , | | | |
| J | | | |
| \jobname | | | |
| 31 , 33 | | | |
| L | | | |
| \Last@LoopValue | | | |
| 519 , 521 | | | |
| \loop@delay | | | |
| 597 , 638 , 642 | | | |
| M | | | |
| \MagCSPoles | | | |
| 158 | | | |
| \MagCSPolesAsymp | | | |
| 158 | | | |
| \MagCSPolesLin | | | |
| 158 | | | |
| \MagCSPolesPeak | | | |
| 177 | | | |
| \MagCSZeros | | | |
| 158 | | | |
| \MagCSZerosAsymp | | | |
| 158 | | | |
| \MagCSZerosLin | | | |
| 158 | | | |
| \MagCSZerosPeak | | | |
| 177 | | | |
| \MagDel | | | |
| 123 , 586 | | | |
| \MagK | | | |
| 117 , 569 | | | |
| \MagKAsymp | | | |
| 117 , 566 | | | |
| \MagKLin | | | |
| 117 , 562 | | | |
| \MagPole | | | |
| 125 , 152 , 554 | | | |
| \MagPoleAsymp 125 , 154 , 551 | | | |
| \MagPoleLin 125 , 153 , 547 | | | |
| \MagSOPoles | | | |
| 185 | | | |
| \MagSOPolesAsymp | | | |
| 185 | | | |
| \MagSOPolesLin | | | |
| 185 | | | |
| \MagSOPolesPeak | | | |
| 203 | | | |
| \MagSOPolesAsymp | | | |
| 185 | | | |
| \MagSOPolesLin | | | |
| 185 | | | |
| \MagSOPolesPeak | | | |
| 203 | | | |
| \MagZero | | | |
| 152 , 538 | | | |
| \MagZeroAsymp | | | |
| 152 , 535 | | | |
| \MagZeroLin | | | |
| 152 , 531 | | | |
| N | | | |
| \n | | | |
| 517 , 519 | | | |
| \n@mod 1 , 307 , 390 , 973 , 1054 | | | |
| \n@mod@n | | | |
| 1 | | | |
| \n@mod@p | | | |
| 1 , 133 | | | |
| \n@pow | | | |
| 1 , | | | |
| 126 , 127 , 128 , 138 , | | | |
| 139 , 141 , 142 , 144 , | | | |
| 145 , 146 , 147 , 148 , | | | |
| 149 , 158 , 159 , 162 , | | | |
| 163 , 164 , 166 , 168 , | | | |
| 169 , 186 , 190 , 609 , | | | |
| 610 , 612 , 613 , 628 , | | | |
| 629 , 631 , 632 , 643 , | | | |
| 644 , 745 , 746 , 751 , | | | |
| 752 , 792 , 793 , 798 , | | | |
| 799 , 825 , 826 , 831 , | | | |
| 832 , 842 , 843 , 848 , 849 | | | |
| \newcounter | | | |
| 28 | | | |
| \NewDocumentEnvironment | | | |
| 445 , | | | |
| 469 , 493 , 858 , 1009 | | | |
| \NicholsChart | | | |
| 902 | | | |
| \NicholsTF | | | |
| 902 | | | |

| | | | | |
|---------------------------------------|-----------------------------|------------------------------|------------------------------------|-----------------------------------|
| <code>\NicholsTF@Shorthandoff</code> | 1000, 1002 | <code>\parse@opt</code> | . 214, 283, <u>646</u> | R |
| <code>\NicholsZPK</code> | <u>902</u> | <code>\pdf@strcmp</code> | | <code>\renewcommand</code> |
| <code>\NicholsZPK@Shorthandoff</code> | 946, 948 | | ... 355, 382, 388, | ... 91, |
| <code>\num@degree</code> | ... 601, 604 | | 397, 527, 529, 533, | 96, 104, 111, 272, |
| <code>\num@im</code> | . 594, 609, 641, 643 | | 543, 545, 549, 559, | 341, 762, 809, 944, 998 |
| <code>\num@real</code> | 593, | | 560, 564, 573, 574, | S |
| | 606, 612, 641, 643 | | 578, 599, 618, 637, | <code>\setcounter</code> |
| <code>\numcoeff</code> | 600, | | 657, 658, 661, 669, | <code>\shorthandoff</code> |
| | 603, 606, 609, 612 | | 670, 673, 681, 684, | ... 273, 342, 442, |
| <code>\numpow</code> | . 600, 601, 603, 604 | <code>\pgfkeysvalueof</code> | 247, | 466, 490, 763, 810, |
| <code>\NyquistPlot</code> | <u>853</u> | | 259, 316, 328, 361, | 855, 945, 999, 1006 |
| <code>\NyquistTF</code> | <u>772</u> | | 370, 371, 389, 392, | <code>\shorthandon</code> |
| <code>\NyquistTF@Shorthandoff</code> | 811, 813 | | 402, 403, 413, 414, | 278, 347, |
| <code>\NyquistZPK</code> | <u>725</u> | | 424, 432, 433, 824, | 768, 815, 950, 1004 |
| <code>\NyquistZPK@Shorthandoff</code> | 764, 766 | | 829, 841, 846, 929, | <code>\stepcounter</code> |
| | | | 981, 1031, 1039, | ... 241, 253, 310, |
| O | | | 1040, 1053, 1061, 1062 | 322, 364, 396, 427, |
| <code>\opt@approx</code> | 219, 652, 685 | <code>\pgfplotsset</code> | . 23, 53, | 749, 796, 828, 845, |
| <code>\opt@axes</code> | ... 457, 481, | | 68, 70, 71, 72, 75, | 924, 976, 1034, 1056 |
| | 504, 710, 714, 719, | | 76, 81, 83, 92, 93, | T |
| | 739, 786, 868, 877, | | 97, 98, 105, 107, 112 | <code>\temp@cmd</code> |
| | 883, 916, 968, 1019 | <code>\ph@scale</code> | .. <u>68</u> , 73, 77, | |
| <code>\opt@commands</code> | .. 747, | | 91, 96, 120, 124, | 217, 218, 220, 230, |
| | 754, 794, 801, 879, | | 135, 148, 151, 163, | 286, 287, 289, 299, |
| | 889, 922, 936, 974, 990 | | 169, 170, 190, 307, | 361, 362, 365, 366, |
| <code>\opt@group</code> | | | 329, 331, 404, 406, | 389, 390, 392, 393, |
| | .. 228, 297, 651, 682 | | 642, 745, 746, 751, | 424, 425, 447, 448, |
| <code>\opt@plot</code> | ... 743, 790, | | 752, 792, 793, 798, | 449, 459, 471, 472, |
| | 878, 886, 894, 919, 971 | | 799, 825, 826, 831, | 473, 483, 495, 496, |
| <code>\opt@tikz</code> | 217, | | 832, 842, 843, 848, | 497, 506, 729, 730, |
| | 286, 447, 471, 495, | | 849, 931, 934, 973, | 732, 741, 743, 745, |
| | 655, 695, 711, 717, | | 983, 988, 1041, | 750, 776, 777, 779, |
| | 729, 776, 860, 880, | | 1044, 1054, 1063, 1068 | 788, 790, 792, 797, |
| | 892, 906, 958, 1011 | <code>\ph@x@label</code> | <u>68</u> | 824, 825, 829, 830, |
| <code>\optmag@axes</code> | ... 231, | | <code>\ph@y@label</code> | 841, 842, 846, 847, |
| | 300, 647, 671, 676 | | <u>68</u> | 860, 861, 862, 870, |
| <code>\optmag@commands</code> | 237, | <code>\PhCSPoles</code> | <u>158</u> | 906, 907, 909, 918, |
| | 252, 306, 321, 654, 691 | | <code>\PhCSPolesAsymp</code> | 919, 921, 925, 958, |
| <code>\optmag@plot</code> | ... 232, | | <u>158</u> , 195 | 959, 961, 970, 971, |
| | 301, 650, 659, 664, 697 | | <code>\PhCSPolesLin</code> | 973, 977, 1011, |
| <code>\optph@axes</code> | ... 233, | | .. <u>158</u> , 192 | 1012, 1013, 1021, |
| | 302, 648, 674, 677 | | <code>\PhCSZeros</code> | 1031, 1032, 1053, 1054 |
| <code>\optph@commands</code> | 239, | | <u>158</u> | <code>\temp@macro</code> |
| | 264, 308, 333, 653, 689 | | <code>\PhCSZerosAsymp</code> | ... 354, |
| <code>\optph@plot</code> | ... 234, | | <u>158</u> | 356, 358, 381, 383, 385 |
| | 303, 649, 662, 665, 699 | | <code>\PhCSZerosLin</code> | .. 231, |
| <code>\Orig@BodeTF</code> | ... 340, 346 | | <u>158</u> | 236, 242, 300, 305, 311 |
| <code>\Orig@BodeZPK</code> | .. 271, 277 | | <code>\PhDel</code> | 124, 585 |
| <code>\Orig@NicholsTF</code> | 997, 1003 | | <code>\PhK</code> | <u>117</u> , 568 |
| <code>\Orig@NicholsZPK</code> | 943, 949 | | .. <u>117</u> , <u>123</u> , 565 | <code>\PhKAsymp</code> |
| <code>\Orig@NyquistTF</code> | 808, 814 | | <u>117</u> , <u>123</u> , 561 | ... <u>117</u> , <u>123</u> , 565 |
| <code>\Orig@NyquistZPK</code> | 761, 767 | | <code>\PhPole</code> | <u>125</u> , 155, 553 |
| | | | <u>125</u> , 157, 550 | <code>\PhPoleAsymp</code> |
| P | | | <code>\PhPoleLin</code> | . <u>125</u> , 156, 546 |
| <code>\p</code> | 544, 546, | | <u>185</u> | <code>\PhSOPoles</code> |
| | 547, 550, 551, 553, 554 | | <code>\PhSOPolesAsymp</code> | <u>185</u> |
| <code>\parse@env@opt</code> | | | <u>185</u> | <code>\PhSOPolesLin</code> |
| | 446, 470, | | <u>185</u> | <u>185</u> |
| | 494, <u>709</u> , 859, 1010 | | <code>\PhS0Zeros</code> | <u>185</u> |
| <code>\parse@N@opt</code> | ... 726, | | <u>185</u> | <code>\PhS0ZerosAsymp</code> |
| | 773, <u>876</u> , 903, 955 | | <u>185</u> | <u>185</u> |
| | | | <code>\PhS0ZerosLin</code> | <u>185</u> |
| | | | <u>152</u> , 537 | <code>\PhZero</code> |
| | | | <u>152</u> , 534 | <u>152</u> , 534 |
| | | | <u>152</u> , 530 | <code>\PhZeroLin</code> |
| | | | <code>\plot@macro</code> | |
| | | | ... 353, 356, 358, | |
| | | | 362, 372, 380, 383, | Z |
| | | | 385, 390, 393, 404, 415 | <code>\z</code> |
| | | | | 528, 530, |
| | | | | 531, 534, 535, 537, 538 |

Change History

| | | | |
|---|-------|----|--|
| v1.0 | | | |
| General: Initial release | | 1 | |
| v1.0.1 | | | |
| \addBodeZPKPlots : Improved optional argument handling. | | 25 | |
| \BodeZPK : Pass arbitrary TikZ commands as options. | | 23 | |
| v1.0.2 | | | |
| gnuplot@prefix : Fixed issue #1 | ... | 18 | |
| v1.0.3 | | | |
| BodePlot : Added tikz option to environments | | 28 | |
| \BodeTF : Added Tikz option | | 24 | |
| \BodeZPK : Added Tikz option | | 23 | |
| NicholsChart : Added tikz option to environments | | 36 | |
| \NicholsTF : Added commands and tikz options | | 36 | |
| \NicholsZPK : Added commands and tikz options | | 36 | |
| gnuplot@prefix : Added jobname to gnuplot prefix | | 18 | |
| \NyquistTF : Added commands and tikz options | | 33 | |
| \NyquistZPK : Added commands and tikz options | | 32 | |
| \parse@env@opt : Added tikz option to environments | | 32 | |
| \parse@N@opt : Added commands and tikz options | | 35 | |
| \parse@opt : Added Tikz option | | 31 | |
| NyquistPlot : Added tikz option to environments | | 35 | |
| v1.0.4 | | | |
| General: Fixed unintended optional argument macro expansion | | 1 | |
| v1.0.5 | | | |
| \parse@opt : Fixed a bug | | 31 | |
| v1.0.6 | | | |
| General: Fixed issue #3 | | 1 | |
| v1.0.7 | | | |
| General: Removed unnecessary semicolons | | 1 | |
| Updated documentation | | 1 | |
| v1.0.8 | | | |
| General: Added a new class option 'declutter' | | 1 | |
| \build@TF@plot : Included phase due to delay in wrapping. | | 30 | |
| gnuplot@prefix : Fixed issue #6 | ... | 18 | |
| v1.1.0 | | | |
| General: Fixed phase wrapping in gnuplot mode | | 1 | |
| \addBodeTFPlot : Fixed phase wrapping in gnuplot mode | | 26 | |
| BodeMagPlot : Added separate environments for phase and magnitude plots | | 27 | |
| BodePhPlot : Added separate environments for phase and magnitude plots | | 27 | |
| BodePlot : Deprecated BodePlot environment | | 28 | |
| \BodeTF : Fixed phase wrapping in gnuplot mode | | 24 | |
| v1.1.1 | | | |
| General: Enable Hz and rad units | ... | 1 | |
| \addBodeComponentPlot : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 26 | |
| \addBodeTFPlot : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 26 | |
| \addBodeZPKPlots : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 25 | |
| \addNicholsTFChart : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 36 | |
| \addNyquistTFPlot : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 34 | |
| \addNyquistZPKPlot : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 34 | |
| BodeMagPlot : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 27 | |
| BodePhPlot : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 27 | |
| BodePlot : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 28 | |
| \BodeTF : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 24 | |
| \BodeZPK : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 23 | |
| \build@TF@plot : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 30 | |
| get@interval@end : New macros to enable 'Hz' and 'rad' units for frequency and phase, respectively | | 20 | |
| ph@y@label : New macros to enable 'Hz' and 'rad' units for frequency and phase, respectively | | 19 | |
| \NyquistTF : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 33 | |
| \NyquistZPK : Enabled 'Hz' and 'rad' units for frequency and phase, respectively | | 32 | |
| v1.1.2 | | | |
| BodeMagPlot : Defined using the 'NewEnviron' command from the | | | |

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