

# The bodeplot package\*

Rushikesh Kamalapurkar  
rlkamalapurkar@gmail.com

July 20, 2022

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	External Dependencies . . . . .	2
1.2	Directory Structure . . . . .	2
1.3	Limitations . . . . .	2
<b>2</b>	<b>TL;DR</b>	<b>3</b>
<b>3</b>	<b>Usage</b>	<b>8</b>
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
<b>4</b>	<b>Implementation</b>	<b>18</b>
4.1	Initialization . . . . .	18
4.2	Parametric function generators for poles, zeros, gains, and delays. . . . .	19
4.3	Second order systems. . . . .	20
4.4	Commands for Bode plots . . . . .	22
4.4.1	User macros . . . . .	22
4.4.2	Internal macros . . . . .	26
4.5	Nyquist plots . . . . .	30
4.5.1	User macros . . . . .	30
4.5.2	Internal commands . . . . .	32
4.6	Nichols charts . . . . .	33

---

\*This document corresponds to bodeplot v1.1.0, dated July 20, 2022.

# 1 Introduction

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

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

and the zero-pole-gain (ZPK) form

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

In the equations above,  $b_m, \dots, b_0$  and  $a_n, \dots, a_0$  are real coefficients,  $T \geq 0$  is the loop delay,  $z_1, \dots, z_m$  and  $p_1, \dots, p_n$  are complex zeros and poles of the transfer function, respectively, and  $K \in \Re$  is the loop gain. For transfer functions in the ZPK format in (2) *with zero delay*, this package also supports linear and asymptotic approximation of Bode plots. By default, all phase plots use degrees as units. Use the **rad** package option to generate plots in radians.

## 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 format, 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

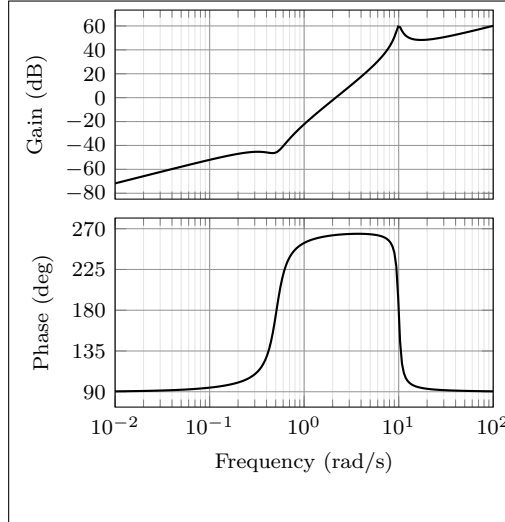
- When plotting Nichols charts in TF form, the phase angles are wrapped between 0 and 360°. As such, the Nichols charts will have phase wrapping discontinuities. Phase wrapping in Bode plots was fixed in v1.1.0 using **gnuplot**. In **pgf** mode, Bode phase plots, plotted using the TF form, will also show phase wrapping discontinuities.
- 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)

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

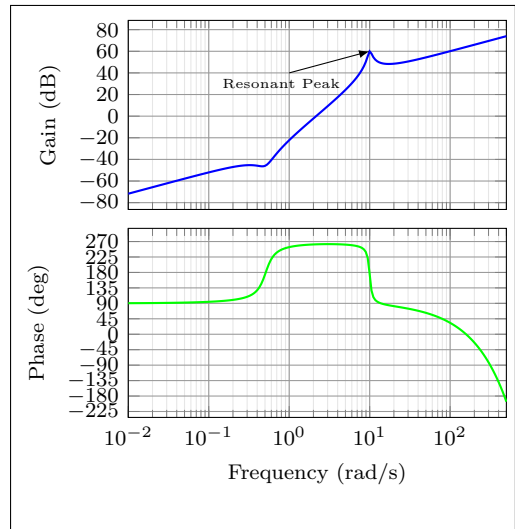
Bode plot in ZPK format



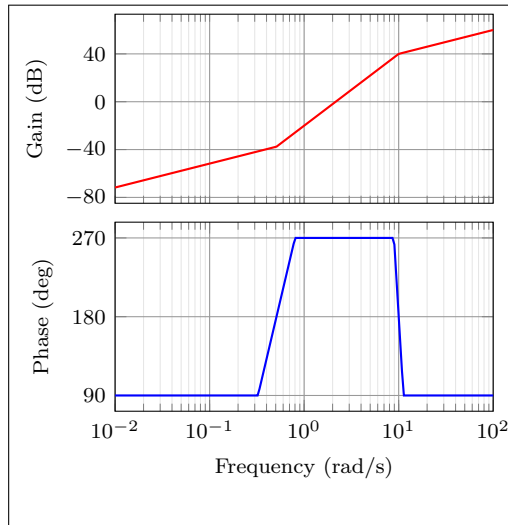
```
\BodeZPK{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10
}
{0.01}
{100}
```

Bode plot in TF format with arrow decoration, transport delay, and color customization (the phase plot will show wrapping if the **pgf** package option is used)

```
\BodeTF[%
  samples=1000,
  plot/mag/{blue,thick},
  plot/ph/{green,thick},
  tikz/>=latex,
  commands/mag/{
    \draw[->](axis cs:1,40) -- (axis cs:10,60);
    \node at (axis cs: 0.8,30) {\tiny Resonant Peak};
  }%
]
{num/{10,2,2.6,0},den/{1,1,100.25},d/0.01}
{0.01}
{500}
```



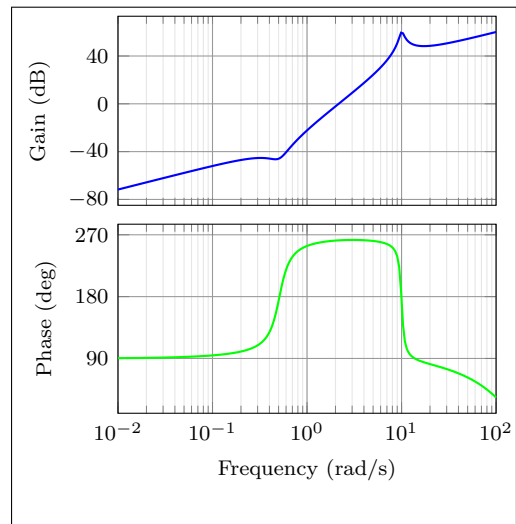
## Linear approximation with customization



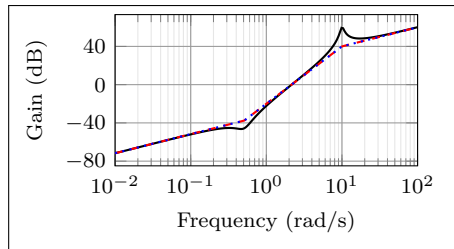
```
\BodeZPK[%
plot/mag/{red,thick},
plot/ph/{blue,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90},
approx/linear%
]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10
}
{0.01}
{100}
```

## Plot with delay and customization

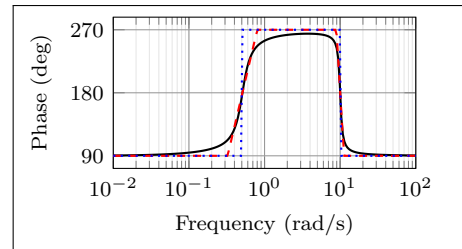
```
\BodeZPK[
plot/mag/{blue,thick},
plot/ph/{green,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90}
]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10,
d/0.01
}
{0.01}
{100}
```



## Individual gain and phase plots with more customization

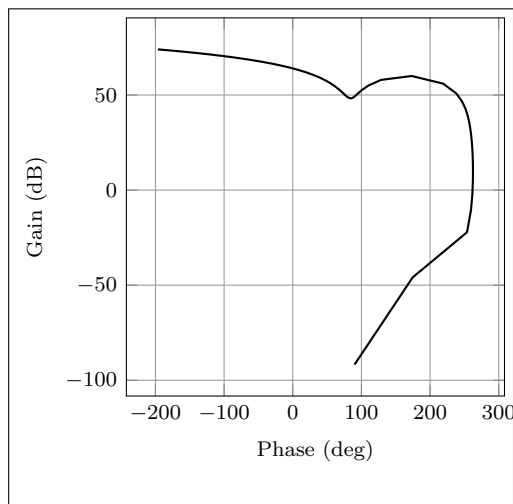


```
\begin{BodeMagPlot}{%
  axes/{height=2cm,
    width=4cm}
}
{0.01}
{100}
\addBodeZPKPlots{%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
}
{magnitude}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10
}
}\end{BodeMagPlot}
```



```
\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}
```

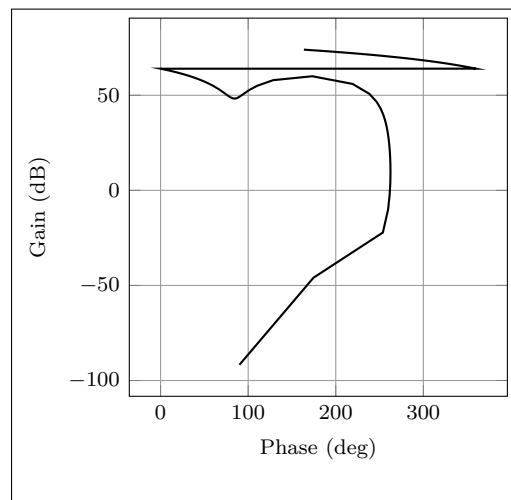
## Nichols chart



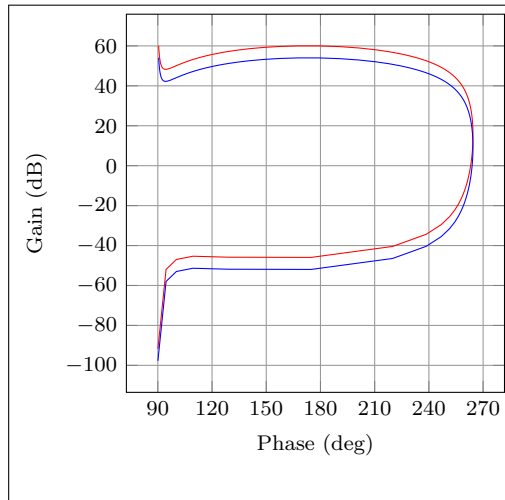
```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01
}
{0.001}
{500}
```

## Same Nichols chart in TF format (shows phase wrapping discontinuity)

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



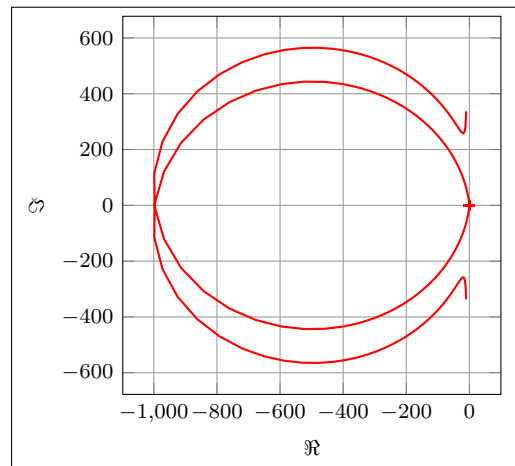
## Multiple Nichols charts with customization



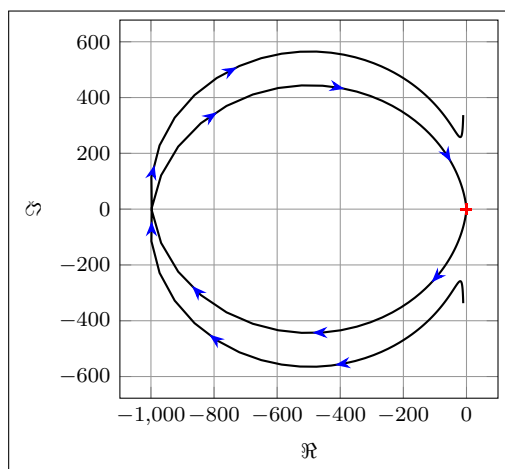
```
\begin{NicholsChart}[%
ytick distance=20,
xtick distance=30
]
{0.001}
{100}
\addNicholsZPKChart [red,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10
};
\addNicholsZPKChart [blue,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/5
};
\end{NicholsChart}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

## 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}
```



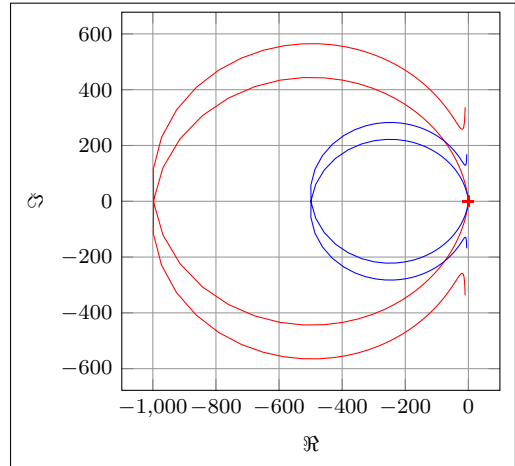
## Nyquist plot in TF format with arrows



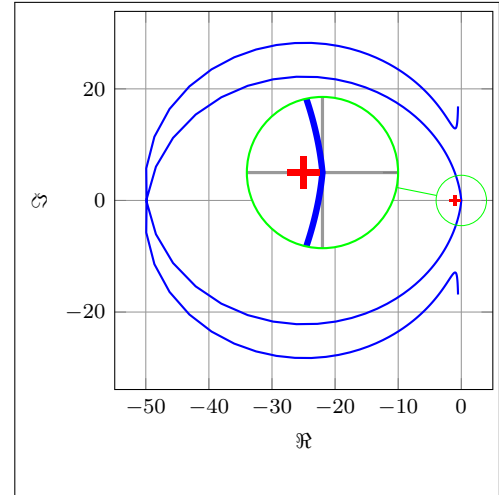
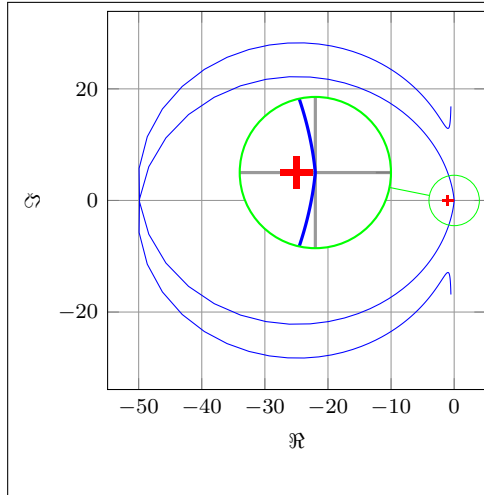
```
\NyquistTF[%
plot/{%
samples=1000,
postaction=decorate,
decoration={
markings,
mark=between positions 0.1 and 0.9 step 5em with {%
\arrow{Stealth [length=2mm, blue]}
}
}%
}%
]{num/{10,2,2.6,0},den/{1,1,100.25}}
{-30}
{30}
```

## 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
}%
}%
}
\addNyquistZPKPlot [blue,samples=1000] {%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/0.5
};
\coordinate (spyon) at (axis cs:0,0);
\coordinate (spyat) at (axis cs:-22,5);
\spy [green] on (spyon) in
node [fill=white] at (spyat);
\end{NyquistPlot}
```

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

## 3 Usage

### 3.1 Bode plots

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

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\}, \dots`. 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 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.

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]`



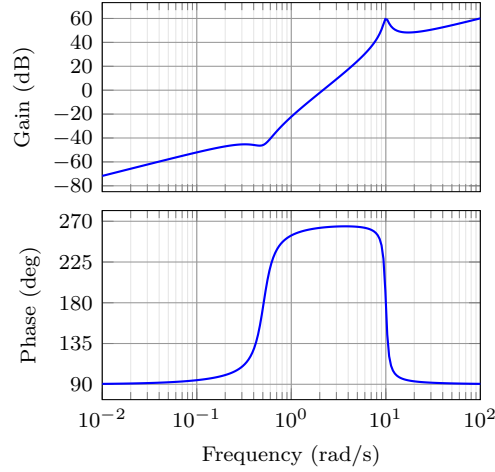


Figure 1: Output of the default `\BodeZPK` macro.

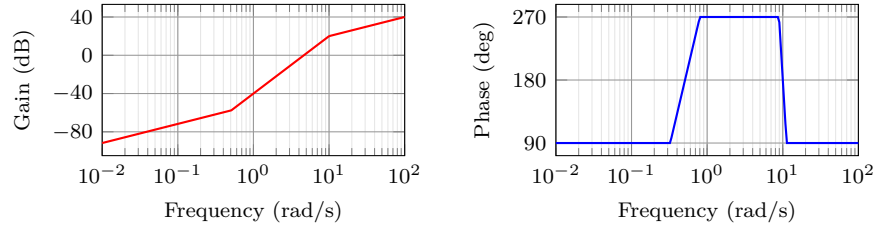


Figure 2: Customization of the default `\BodeZPK` macro.

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

which generates the plot in Figure 1. If a delay is not specified, it is assumed to be zero. If a gain is not specified, it is assumed to be 1. By default, each of the axes, excluding ticks and labels, are 5cm wide and 2.5cm high. The width and the height, along with other properties of the plots, the axes, and the group can be customized using native `pgf` keys as shown in the example below.

As demonstrated in this example, if a single comma-separated list of options is passed, it applies to both the magnitude and the phase plots. Without any optional arguments, we get a thick black Bode plot.

A linear approximation of the Bode plot with customization of the plots, the axes, and the group can be generated using

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

which generates the plot in Figure 2.

```
\BodeTF \BodeTF [{obj1/typ1/{opt1}},obj2/typ2/{opt2}],...]
{<num/{coeffs}>,den/{coeffs}>,d/{delay}>}
{<min-freq>}{<max-freq>}
```

Plots the Bode plot of a transfer function given in TF format. The three mandatory arguments include: (1) a list of tuples comprised of the coefficients in the numerator and the denominator of the transfer function and the transport delay, (2) the lower end of the frequency range for the  $x$ -axis, and (3) the higher end of the frequency range for the  $x$ -axis. The coefficients are entered as a comma-separated list, in order

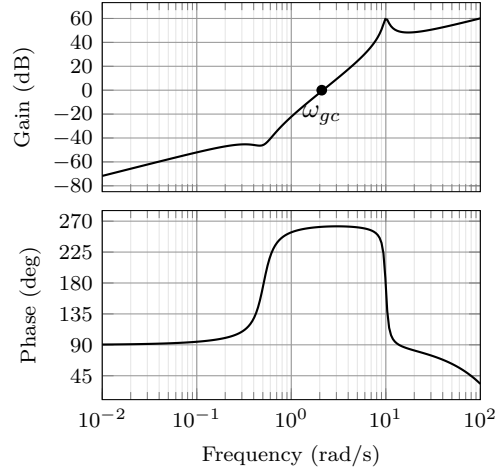


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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

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

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

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

```
\BodeTF[commands/mag/{\node at (axis cs: 2.1,0)
[circle,fill,inner sep=0.05cm,label=below:{$\omega_{gc}$}]}]
{num/{10,2,2.6,0},den/{1,1,100.25},d/0.01}
{0.01}{100}
```

which generates the plot in Figure 3. Note the 0 added to the numerator coefficients to account for the fact that the numerator does not have a constant term in it. Note the semicolon after the TikZ command passed to the `\commands` option.

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

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

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

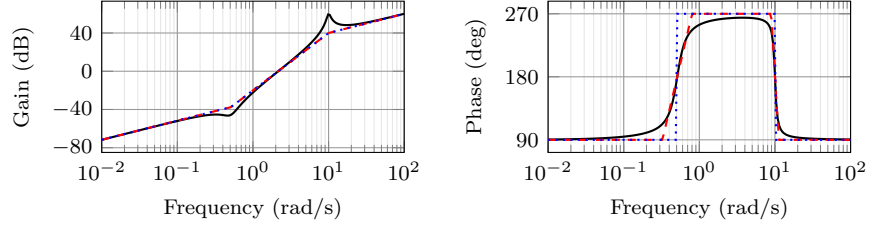


Figure 4: Superimposed approximate and true Bode plots using the **BodeMagPlot** and **BodePhPlot** environments and the **\addBodeZPKPlots** macro.

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

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

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

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

Generates the appropriate parametric functions and supplies them to multiple **\addplot** macros, one for each **approx/\langle opt\rangle** 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/\langle opt\rangle**, **linear/\langle opt\rangle**, or **asymptotic/\langle opt\rangle**. This macro can be used inside any **semilogaxis** environment as long as a domain for the x-axis is supplied through either the **approx/\langle opt\rangle** 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\}
                    \{num/\langle coeffs\rangle\},den/\langle coeffs\rangle\},d/\langle delay\rangle\}}
```

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

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

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

### 3.1.1 Basic components up to first order

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

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

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

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

```
\begin{BodePhPlot}[ytick distance=90]{0.01}{100}
  \addBodeComponentPlot[black,thick]{\PhZero{0}{0} + \PhZero{-0.1}{-
0.5} +
    \PhZero{-0.1}{0.5} + \PhPole{-0.5}{-10} + \PhPole{-0.5}{10} +
    \PhK{10}{0}}
  \addBodeComponentPlot[red,dashed,thick]{\PhZeroLin{0}{0} +
    \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.

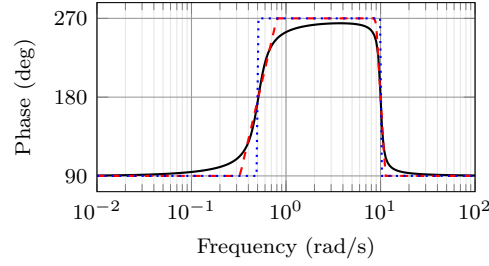


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePh-Plot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

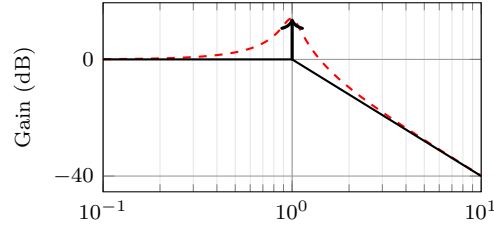


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

### 3.1.2 Basic components of the second order

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

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

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

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

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

generates the plot in Figure 6.

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

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

replaced by **Lin** or **Asymp** to generate the true Bode plot, the linear approximation, or the asymptotic approximation, respectively.

**\MagCSFeaturePeak**      **\MagCSFeaturePeak**[\draw-options]{\zeta}{\omega-n}

This entry describes 2 different macros of the form **\MagCSFeaturePeak** that take the damping ratio,  $\zeta$ , and the natural frequency,  $\omega_n$  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  $\omega$  can be added by adding **\usetikzlibrary{decorations.markings}** and **\usetikzlibrary{arrows.meta}** to the preamble and then passing a tuple of the form

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

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

For example, the command

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

generates the Nyquist plot in Figure 7.

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

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

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

generates the Nyquist plot in Figure 8.

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

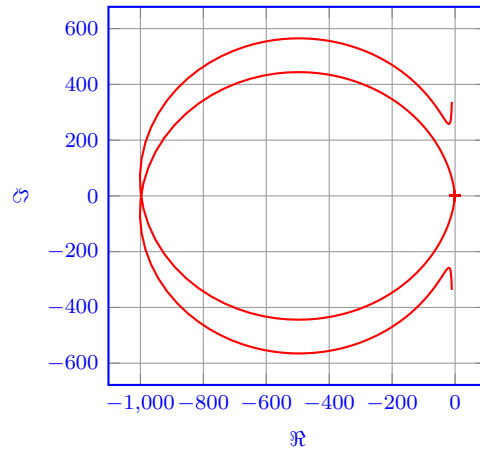


Figure 7: Output of the `\NyquistZPK` macro.

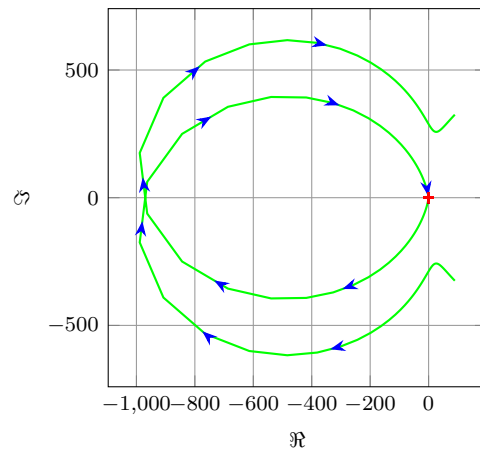


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

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

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

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

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

Generates a 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}>,...]`  
                                 `{<min-frequency>}{<max-frequency>}`  
                                 `\addNichols...`  
                                 `\end{NicholsChart}`

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



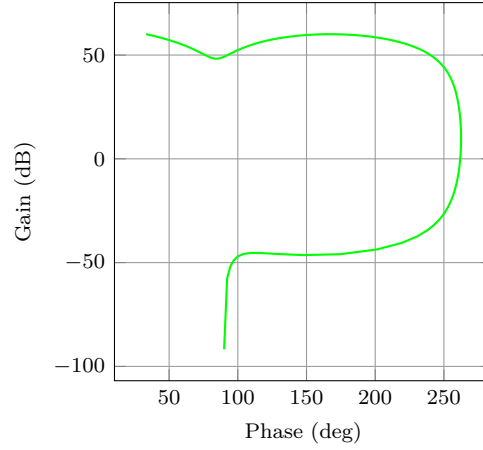


Figure 9: Output of the `\NyquistZPK` macro.

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

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

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

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

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

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

## 4 Implementation

### 4.1 Initialization

`\pdfstrcmp` The package makes extensive use of the `\pdfstrcmp` macro to parse options. Since that macro is not available in `lualatex`, this code is needed.

```
1 \RequirePackage{ifluatex}%
2 \ifluatex
3   \let\pdfstrcmp\pdf@strcmp
4 \fi
```

`\n@mod` This code is needed to support both `pgfplots` and `gnuplot` simultaneously. New  
`\n@pow` macros are defined for the `pow` and `mod` functions to address differences between the  
`gnuplot@id` two math engines. We start by processing the class options.

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

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

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

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

```
26 \newcounter{gnuplot@id}
27 \setcounter{gnuplot@id}{0}
28 \if@declutterarg
29   \tikzset{%
30     gnuplot@prefix/.style={%
31       id=\arabic{gnuplot@id},
32       prefix=gnuplot/\jobname
33     }%
34   }
35 \else
36   \tikzset{%
37     gnuplot@prefix/.style={%
38       id=\arabic{gnuplot@id},
39       prefix=\jobname
40     }%
41   }
42 \fi
```

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.

```
43 \ifwindows\else
44   \if@declutterarg
```

```

45 \immediate\write18{mkdir -p gnuplot}
46 \fi
47 \fi
48 \fi

```

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

```

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

```

**ph@filter** These macros create **pgf** filters to convert plots from radians to degrees.

```

ph@x@filter 63 \if@radarg
64   \pgfplotsset{ph@filter/.style = {ytick distance=pi/4, yla-
        bel={Phase (rad)}}}%
65   \pgfplotsset{ph@x@filter/.style = {xlabel={Phase (rad)}}}%
66 \else
67   \pgfplotsset{ph@filter/.style = {y filter/.expression={y*180/pi}, yt-
        ick distance=45, ylabel={Phase (deg)}}}%
68   \pgfplotsset{ph@x@filter/.style = {x filter/.expression={x*180/pi}, xla-
        bel={Phase (deg)}}}%
69 \fi

```

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

**\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 70 \newcommand*\MagK[2]{(20*log10(abs(#1)))}
\PhKLin 71 \newcommand*\MagKAsymp{\MagK}
72 \newcommand*\MagKLin{\MagK}
73 \newcommand*\PhK[2]{(#1<0?-pi:0)}
74 \newcommand*\PhKAsymp{\PhK}
75 \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.

```

76 \newcommand*\MagDel[2]{0}
77 \newcommand*\PhDel[2]{-#1*t}

```

**\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 78 \newcommand*\MagPole[2]
\PhPole 79 {( -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 80 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
81   -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :

```

```

82 -20*log10(t)
83 }}

```

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

```

84 \newcommand*\MagPoleAsymp{\MagPoleLin}

```

Parametric function for the true phase of a complex pole.

```

85 \newcommand*\PhPole}[2]{(#1 > 0 ? (#2 > 0 ?
86   (\n@mod{-atan2((t - (#2)), -(#1))}{2*pi}) :
87   (-atan2((t - (#2)), -(#1)))) :
88   (-atan2((t - (#2)), -(#1))))}

```

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

```

89 \newcommand*\PhPoleLin}[2]{%
90   (abs(#1)+abs(#2) == 0 ? -pi/2 :
91   (t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
92     (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))) ?
93     (-atan2(-(#2), -(#1))) :
94     (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
95       (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))) ?
96       (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) :
97       (-atan2(-(#2), -(#1)) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) /
98         (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} +
99         \n@pow{#2}{2})))))))*((#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) + atan2(-
100       (#2), -(#1)))/
101       (log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
102       (\n@pow{#1}{2} + \n@pow{#2}{2})))))))))}

```

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

```

102 \newcommand*\PhPoleAsymp}[2]{(t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) ?
103   (-atan2(-(#2), -(#1))) :
104   (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2))}

```

**\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 105 \newcommand*\MagZero{0-\MagPole}
\PhZero 106 \newcommand*\MagZeroLin{0-\MagPoleLin}
\PhZeroAsymp 107 \newcommand*\MagZeroAsymp{0-\MagPoleAsymp}
\PhZeroLin 108 \newcommand*\PhZero{0-\PhPole}
109 \newcommand*\PhZeroLin{0-\PhPoleLin}
110 \newcommand*\PhZeroAsymp{0-\PhPoleAsymp}

```

### 4.3 Second order systems.

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

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

```

\MagCSPolesAsymp
\MagCSPolesLin 111 \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles 112   - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2}))))}
\PhCSPolesAsymp 113 \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
\PhCSPolesLin 40*log10(t))}
\MagCSZeros 114 \newcommand*\MagCSPolesAsymp{\MagCSPolesLin}

```

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

```

\PhCSZeros 115 \newcommand*\PhCSPoles}[2]{(-atan2((2*(#1)*(#2)*t), (\n@pow{#2}{2}
\PhCSZerosAsymp 116   - \n@pow{t}{2}))))}
\PhCSZerosLin 117 \newcommand*\PhCSPolesLin}[2]{(t < (#2 / (\n@pow{10}{abs(#1)))) ?
118   0 :
119   (t >= (#2 * (\n@pow{10}{abs(#1)))) ?
120   (#1>0 ? -pi : pi) :

```

```

121  (#1>0 ? (-pi*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
122  (pi*(log10(t*(\n@pow{10}{abs(#1)})/#2))/(2*abs(#1))))))}
123 \newcommand*\PhCSPolesAsymp}[2]{(#1>0?(t<#2?0:-pi):(t<#2?0:pi))}

```

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).

```

124 \newcommand*\MagCSZeros}{0-\MagCSPoles}
125 \newcommand*\MagCSZerosLin}{0-\MagCSPolesLin}
126 \newcommand*\MagCSZerosAsymp}{0-\MagCSPolesAsymp}
127 \newcommand*\PhCSZeros}{0-\PhCSPoles}
128 \newcommand*\PhCSZerosLin}{0-\PhCSPolesLin}
129 \newcommand*\PhCSZerosAsymp}{0-\PhCSPolesAsymp}

```

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

```

130 \newcommand*\MagCSPolesPeak}[3][[%
131  \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
132  (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
133 }
134 \newcommand*\MagCSZerosPeak}[3][[%
135  \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
136  (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
137 }

```

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

```

138 \newcommand*\MagS0Poles}[2]{%
139  \PhS0Poles 139 (-20*log10(sqrt(\n@pow{#2} - \n@pow{t}{2}){2} + \n@pow{#1*t}{2})))}
140 \newcommand*\MagS0PolesLin}[2]{%
141  \PhS0PolesLin 141 (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t))}
142 \newcommand*\MagS0PolesAsymp}{\MagS0PolesLin}

```

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

```

143 \newcommand*\PhS0Poles}[2]{(-atan2((#1)*t,((#2) - \n@pow{t}{2})))}
144 \newcommand*\PhS0PolesLin}[2]{(#2>0 ?
145  \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
146  (#1>0 ? -pi : pi))}
147 \newcommand*\PhS0PolesAsymp}[2]{(#2>0 ?
148  \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
149  (#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).

```

150 \newcommand*\MagS0Zeros}{0-\MagS0Poles}
151 \newcommand*\MagS0ZerosLin}{0-\MagS0PolesLin}
152 \newcommand*\MagS0ZerosAsymp}{0-\MagS0PolesAsymp}
153 \newcommand*\PhS0Zeros}{0-\PhS0Poles}
154 \newcommand*\PhS0ZerosLin}{0-\PhS0PolesLin}
155 \newcommand*\PhS0ZerosAsymp}{0-\PhS0PolesAsymp}

```

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

```

156 \newcommand*\MagS0PolesPeak}[3][[%
157  \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
158  (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
159  20*log10(abs(#2/sqrt(abs(#3))))});
160 }
161 \newcommand*\MagS0ZerosPeak}[3][[%
162  \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --

```

```

163   (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
164     20*log10(abs(#2/sqrt(abs(#3))))});
165 }

```

## 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`.

```

166 \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.

```

167   \parse@opt{#1}%
168   \gdef\func@mag{}%
169   \gdef\func@ph{}%
170   \build@ZPK@plot{\func@mag}{\func@ph}{\opt@approx}{#2}%

```

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

```

171   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
    panded\expandafter{\opt@tikz}]%
172     \noexpand\begin{groupplot}%
173       bode@style,
174       xmin={#3},
175       xmax={#4},
176       domain=#3:#4,
177       height=2.5cm,
178       xmode=log,
179       group style = {group size = 1 by 2,vertical sep=0.25cm},
180       \opt@group
181     ]%
182   }%
183   \temp@cmd

```

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

```

184     \edef\temp@mag@cmd{\noexpand\nextgroupplot[ytick dis-
    tance=20, ylabel={Gain (dB)}, xmajor ticks=false, \optmag@axes]
185     \noexpand\addplot[variable=t, thick, \optmag@plot]}
186     \edef\temp@ph@cmd{\noexpand\nextgroupplot[ph@filter, xla-
    bel={Frequency (rad/s)}, \optph@axes]
187     \noexpand\addplot[variable=t, thick, \optph@plot]}

```

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.

```

188     \if@pgfarg\else
189       \edef\temp@mag@cmd{\noexpand\stepcounter{gnuplot@id} \unex-
    panded\expandafter{\temp@mag@cmd} gnuplot[gnuplot@prefix]}
190       \edef\temp@ph@cmd{\noexpand\stepcounter{gnuplot@id} \unex-
    panded\expandafter{\temp@ph@cmd} gnuplot[gnuplot@prefix]}
191     \fi
192     \temp@mag@cmd {\func@mag};
193     \optmag@commands

```

```

194     \temp@ph@cmd {\func@ph};
195     \optph@commands
196   \end{groupplot}
197 \end{tikzpicture}
198 }

```

**\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.

```

199 \newcommand{\BodeTF}[4][[]]{%
200   \parse@opt{#1}%
201   \gdef\func@mag{%
202     \gdef\func@ph{%
203       \build@TF@plot{\func@mag}{\func@ph}{#2}%
204       \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
205         panded\expandafter{\opt@tikz}]{%
206         \noexpand\begin{groupplot}{%
207           bode@style,
208           xmin={#3},
209           xmax={#4},
210           domain=#3:#4,
211           height=2.5cm,
212           xmode=log,
213           group style = {group size = 1 by 2,vertical sep=0.25cm},
214           \opt@group
215         }%
216       \temp@cmd
217       \edef\temp@mag@cmd{\noexpand\nextgroupplot[ytick dis-
218         tance=20, ylabel={Gain (dB)}, xmajor ticks=false, \optmag@axes]
219       \noexpand\addplot[variable=t, thick, \optmag@plot]}
220       \edef\temp@ph@cmd{\noexpand\nextgroupplot[ph@filter, xla-
221         bel={Frequency (rad/s)}, \optph@axes]
222       \noexpand\addplot[variable=t, thick, \optph@plot]}
223       \ifpgfarg\else
224         \edef\temp@mag@cmd{\noexpand\stepcounter{gnuplot@id} \unex-
225         panded\expandafter{\temp@mag@cmd} gnuplot[gnuplot@prefix]}
226         \edef\temp@ph@cmd{\noexpand\stepcounter{gnuplot@id} \unex-
227         panded\expandafter{\temp@ph@cmd} gnuplot[gnuplot@prefix]}
228       \fi
229       \temp@mag@cmd {\func@mag};
230       \optmag@commands
231       \ifpgfarg
232         \temp@ph@cmd {\n@mod{\func@ph}{2*pi}};
233       \else
234         \temp@ph@cmd {'+' using (t):\func@ph smooth unwrap};
235       \fi
236       \optph@commands
237     \end{groupplot}
238   \end{tikzpicture}
239 }

```

**\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.

```

236 \newcommand{\addBodeZPKPlots}[3][true/{}]{%
237   \foreach \approx/\opt in {#1} {%
238     \gdef\plot@macro{%
239       \gdef\temp@macro{%
240         \ifnum\pdfstrcmp{#2}{phase}=0
241           \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}%
242         \else
243           \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}%
244         \fi

```

```

245 \ifpgfarg
246 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt]}%
247 \temp@cmd {\plot@macro};
248 \else
249 \stepcounter{gnuplot@id}%
250 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt]}
251 \temp@cmd gnuplot[gnuplot@prefix] {\plot@macro};
252 \fi
253 }%
254 }

```

**\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.

```

255 \newcommand{\addBodeTFPlot}[3][thick]{%
256 \gdef\plot@macro{}%
257 \gdef\temp@macro{}%
258 \ifnum\pdfstrcmp{#2}{phase}=0
259 \build@TF@plot{\temp@macro}{\plot@macro}{#3}%
260 \else
261 \build@TF@plot{\plot@macro}{\temp@macro}{#3}%
262 \fi
263 \ifpgfarg
264 \ifnum\pdfstrcmp{#2}{phase}=0
265 \addplot[variable=t,#1]{\n@mod{\plot@macro}{2*pi}};
266 \else
267 \addplot[variable=t,#1]{\plot@macro};
268 \fi
269 \else
270 \stepcounter{gnuplot@id}%
271 \ifnum\pdfstrcmp{#2}{phase}=0
272 \addplot[variable=t,#1] gnuplot[gnuplot@prefix] {'+' us-
ing (t):\plot@macro smooth unwrap}
273 \else
274 \addplot[variable=t,#1] gnuplot[gnuplot@prefix] {\plot@macro};
275 \fi
276 \fi
277 }

```

**\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.

```

278 \newcommand{\addBodeComponentPlot}[2][thick]{%
279 \ifpgfarg
280 \addplot[variable=t,#1]{#2};
281 \else
282 \stepcounter{gnuplot@id}%
283 \addplot[variable=t,#1] gnuplot[gnuplot@prefix] {#2};
284 \fi
285 }

```

**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.

```

286 \newenvironment{BodePhPlot}[3][]{%
287 \parse@env@opt{#1}%
288 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}}
289 \noexpand\begin{semilogaxis}{%
290 ph@filter,
291 bode@style,
292 xmin={#2},
293 xmax={#3},
294 domain=#2:#3,

```



```

295     height=2.5cm,
296     xlabel={Frequency (rad/s)},
297     \unexpanded\expandafter{\opt@axes}
298   ]%
299 }
300 \temp@cmd
301 }{
302   \end{semilogxaxis}
303 \end{tikzpicture}
304 }

```

**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.

```

305 \newenvironment{BodeMagPlot}[3][{}]{%
306   \parse@env@opt{#1}%
307   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
308     panded\expandafter{\opt@tikz}]
309     \noexpand\begin{semilogxaxis}[%
310       bode@style,
311       xmin={#2},
312       xmax={#3},
313       domain=#2:#3,
314       height=2.5cm,
315       xlabel={Frequency (rad/s)},
316       ylabel={Gain (dB)},
317       ytick distance=40,
318       \unexpanded\expandafter{\opt@axes}
319     ]%
320   }
321 }{
322   \end{semilogxaxis}
323 \end{tikzpicture}
324 }

```

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

```

325 \newenvironment{BodePlot}[3][{}]{%
326   \PackageWarning{bodeplot}{Since v1.1.0, the BodePlot environment re-
327     turns phase plots in radian units only. Please use the BodePhPlot envi-
328     ronment if degree units are needed.}%
329   \parse@env@opt{#1}%
330   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
331     panded\expandafter{\opt@tikz}]
332     \noexpand\begin{semilogxaxis}[%
333       bode@style,
334       xmin={#2},
335       xmax={#3},
336       domain=#2:#3,
337       height=2.5cm,
338       xlabel={Frequency (rad/s)},
339       \unexpanded\expandafter{\opt@axes}
340     ]%
341   }
342 }{
343   \end{semilogxaxis}
344 \end{tikzpicture}
345 }

```

#### 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](#).

```

344 \newcommand*\add@feature}[3]{%
345   \ifcat$\detokenize\expandafter{#1}$%
346   \xdef#1{\unexpanded\expandafter{#1 0+#2}}%
347   \else
348   \xdef#1{\unexpanded\expandafter{#1+#2}}%
349   \fi
350   \foreach \y [count=\n] in #3 {%
351     \xdef#1{\unexpanded\expandafter{#1}{\y}}%
352     \xdef\Last@LoopValue{\n}%
353   }%
354   \ifnum\Last@LoopValue=1%
355     \xdef#1{\unexpanded\expandafter{#1}{0}}%
356   \fi
357 }
```

**\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.

```

358 \newcommand{\build@ZPK@plot}[4]{%
359   \foreach \feature/\values in {#4} {%
360     \ifnum\pdfstrcmp{\feature}{z}=0
361       \foreach \z in \values {%
362         \ifnum\pdfstrcmp{#3}{linear}=0
363           \add@feature{#2}{\PhZeroLin}{\z}%
364           \add@feature{#1}{\MagZeroLin}{\z}%
365         \else
366           \ifnum\pdfstrcmp{#3}{asymptotic}=0
367             \add@feature{#2}{\PhZeroAsymp}{\z}%
368             \add@feature{#1}{\MagZeroAsymp}{\z}%
369           \else
370             \add@feature{#2}{\PhZero}{\z}%
371             \add@feature{#1}{\MagZero}{\z}%
372           \fi
373         \fi
374       }%
375     \fi
376     \ifnum\pdfstrcmp{\feature}{p}=0
377       \foreach \p in \values {%
378         \ifnum\pdfstrcmp{#3}{linear}=0
379           \add@feature{#2}{\PhPoleLin}{\p}%
380           \add@feature{#1}{\MagPoleLin}{\p}%
381         \else
382           \ifnum\pdfstrcmp{#3}{asymptotic}=0
383             \add@feature{#2}{\PhPoleAsymp}{\p}%
384             \add@feature{#1}{\MagPoleAsymp}{\p}%
385           \else
386             \add@feature{#2}{\PhPole}{\p}%
387             \add@feature{#1}{\MagPole}{\p}%
388           \fi
389         \fi

```

```

390     }%
391   \fi
392   \ifnum\pdfstrcmp{\feature}{k}=0
393     \ifnum\pdfstrcmp{#3}{linear}=0
394       \add@feature{#2}{\PhKLin}{\values}%
395       \add@feature{#1}{\MagKLin}{\values}%
396     \else
397       \ifnum\pdfstrcmp{#3}{asymptotic}=0
398         \add@feature{#2}{\PhKAsymp}{\values}%
399         \add@feature{#1}{\MagKAsymp}{\values}%
400       \else
401         \add@feature{#2}{\PhK}{\values}%
402         \add@feature{#1}{\MagK}{\values}%
403       \fi
404     \fi
405   \fi
406   \ifnum\pdfstrcmp{\feature}{d}=0
407     \ifnum\pdfstrcmp{#3}{linear}=0
408       \PackageError {bodeplot} {Linear approximation for pure de-
409       lays is not
410       supported.} {Plot the true Bode plot using 'true' in-
411       stead of 'linear'.}
412     \else
413       \ifnum\pdfstrcmp{#3}{asymptotic}=0
414         \PackageError {bodeplot} {Asymptotic approxima-
415         tion for pure delays is not
416         supported.} {Plot the true Bode plot using 'true' in-
417         stead of 'asymptotic'.}
418       \else
419         \ifdim\values pt < 0pt
420           \PackageError {bodeplot} {Delay needs to be a positive num-
421           ber.}
422         \fi
423       \fi
424     \fi
425     \add@feature{#2}{\PhDel}{\values}%
426     \add@feature{#1}{\MagDel}{\values}%
427   \fi
428 \fi
429 }%
430 }

```

**\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).

```

425 \newcommand{\build@TF@plot}[3]{%
426   \gdef\num@real{0}%
427   \gdef\num@im{0}%
428   \gdef\den@real{0}%
429   \gdef\den@im{0}%
430   \gdef\loop@delay{0}%
431   \foreach \feature/\values in {#3} {%
432     \ifnum\pdfstrcmp{\feature}{num}=0
433       \foreach \numcoeff [count=\numpow] in \values {%
434         \xdef\num@degree{\numpow}%
435       }%
436       \foreach \numcoeff [count=\numpow] in \values {%
437         \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}%
438         \ifnum\currentdegree = 0
439           \xdef\num@real{\num@real+\numcoeff}%
440         \else
441           \ifodd\currentdegree
442             \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-

```

```

1}{(\currentdegree-1)/2})*%
443      (\n@pow{t}{\currentdegree}}))}%
444      \else
445      \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-
1}{(\currentdegree)/2})*%
446      (\n@pow{t}{\currentdegree}}))}%
447      \fi
448      \fi
449    }%
450  \fi
451  \ifnum\pdfstrcmp{\feature}{den}=0
452    \foreach \dencoeff [count=\denpow] in \values {%
453      \xdef\den@degree{\denpow}%
454    }%
455    \foreach \dencoeff [count=\denpow] in \values {%
456      \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}%
457      \ifnum\currentdegree = 0
458        \xdef\den@real{\den@real+\dencoeff}%
459      \else
460        \ifodd\currentdegree
461          \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-
1}{(\currentdegree-1)/2})*%
462          (\n@pow{t}{\currentdegree}}))}%
463        \else
464          \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-
1}{(\currentdegree)/2})*%
465          (\n@pow{t}{\currentdegree}}))}%
466        \fi
467      \fi
468    }%
469  \fi
470  \ifnum\pdfstrcmp{\feature}{d}=0
471    \xdef\loop@delay{\values}%
472  \fi
473 }%
474 \xdef#2{(atan2((\num@im),(\num@real))-atan2((\den@im),%
475 (\den@real))-\loop@delay*t)}%
476 \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2}))) -
%
477 20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2}))))}%
478 }

```

`\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.

```

479 \newcommand{\parse@opt}[1]{%
480   \gdef\optmag@axes{}%
481   \gdef\optph@axes{}%
482   \gdef\optph@plot{}%
483   \gdef\optmag@plot{}%
484   \gdef\opt@group{}%
485   \gdef\opt@approx{}%
486   \gdef\optph@commands{}%
487   \gdef\optmag@commands{}%
488   \gdef\opt@tikz{}%
489   \foreach \obj/\typ/\opt in {#1} {%
490     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{plot}=0
491       \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{mag}=0

```

```

492     \xdef\optmag@plot{\unexpanded\expandafter{\opt}}%
493 \else
494     \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
495     \xdef\optph@plot{\unexpanded\expandafter{\opt}}%
496 \else
497     \xdef\optmag@plot{\unexpanded\expandafter{\opt}}%
498     \xdef\optph@plot{\unexpanded\expandafter{\opt}}%
499 \fi
500 \fi
501 \else
502     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0
503     \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{mag}=0
504     \xdef\optmag@axes{\unexpanded\expandafter{\opt}}%
505 \else
506     \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
507     \xdef\optph@axes{\unexpanded\expandafter{\opt}}%
508 \else
509     \xdef\optmag@axes{\unexpanded\expandafter{\opt}}%
510     \xdef\optph@axes{\unexpanded\expandafter{\opt}}%
511 \fi
512 \fi
513 \else
514     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{group}=0
515     \xdef\opt@group{\unexpanded\expandafter{\opt}}%
516 \else
517     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{approx}=0
518     \xdef\opt@approx{\unexpanded\expandafter{\opt}}%
519 \else
520     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{commands}=0
521     \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
522     \xdef\optph@commands{\unexpanded\expandafter{\opt}}%
523 \else
524     \xdef\optmag@commands{\unexpanded\expandafter{\opt}}%
525 \fi
526 \else
527     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{tikz}=0
528     \xdef\opt@tikz{\unexpanded\expandafter{\opt}}%
529 \else
530     \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
531     \unexpanded\expandafter{\obj}}%
532     \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
533     \unexpanded\expandafter{\obj}}%
534 \fi
535 \fi
536 \fi
537 \fi
538 \fi
539 \fi
540 }%
541 }

```

**\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.

```

542 \newcommand{\parse@env@opt}[1]{%
543 \gdef\opt@axes{}%
544 \gdef\opt@tikz{}%
545 \foreach \obj/\opt in {#1} {%
546     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0
547     \xdef\opt@axes{\unexpanded\expandafter{\opt}}%
548 \else

```

```

549     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{tikz}=0
550     \xdef\opt@tikz{\unexpanded\expandafter{\opt}}%
551     \else
552     \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
553     \unexpanded\expandafter{\obj}}%
554     \fi
555     \fi
556 }%
557 }

```

## 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  $\omega$ . Logarithmic sampling makes it worse, perhaps inverse logarithmic sampling will help, pull requests to fix that are welcome!

```

558 \newcommand{\NyquistZPK}[4][{}]{%
559   \parse@N@opt{#1}%
560   \gdef\func@mag{}%
561   \gdef\func@ph{}%
562   \build@ZPK@plot{\func@mag}{\func@ph}{{#2}}%
563   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}}%
564     \noexpand\begin{axis}[%
565       bode@style,
566       domain=#3:#4,
567       height=5cm,
568       xlabel={\Re$},
569       ylabel={\Im$},
570       samples=500,
571       \unexpanded\expandafter{\opt@axes}
572     ]%
573   }%
574   \temp@cmd
575     \addplot [only marks,mark=+,thick,red] (-1 , 0);
576     \edef\temp@cmd{\noexpand\addplot[variable=t, thick, \unex-
panded\expandafter{\opt@plot}}}%
577     \if@pgfarg
578       \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}}*cos(\func@ph)},
579       {\n@pow{10}{((\func@mag)/20)}}*sin(\func@ph)} );
580     \opt@commands
581     \else
582       \stepcounter{gnuplot@id}%
583       \temp@cmd gnuplot[parametric,gnuplot@prefix] {%
584         \n@pow{10}{((\func@mag)/20)}}*cos(\func@ph),
585         \n@pow{10}{((\func@mag)/20)}}*sin(\func@ph)};
586       \opt@commands
587     \fi
588     \end{axis}
589   \end{tikzpicture}
590 }

```

**\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**.

```

591 \newcommand{\NyquistTF}[4][[]]{%
592   \parse@N@opt{#1}%
593   \gdef\func@mag{}%
594   \gdef\func@ph{}%
595   \build@TF@plot{\func@mag}{\func@ph}{#2}%
596   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]%
597     \noexpand\begin{axis}%
598       bode@style,
599       domain=#3:#4,
600       height=5cm,
601       xlabel={\Re$},
602       ylabel={\Im$},
603       samples=500,
604       \unexpanded\expandafter{\opt@axes}
605     ]%
606   }%
607   \temp@cmd
608     \addplot [only marks,mark=+,thick,red] (-1 , 0);
609     \edef\temp@cmd{\noexpand\addplot[variable=t, thick, \unex-
panded\expandafter{\opt@plot}]%
610       \if@pgfarg
611         \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}}*cos(\func@ph)},
612         {\n@pow{10}{((\func@mag)/20)}}*sin(\func@ph)} );
613       \opt@commands
614     \else
615       \stepcounter{gnuplot@id}%
616       \temp@cmd gnuplot[parametric,gnuplot@prefix]{%
617         \n@pow{10}{((\func@mag)/20)}}*cos(\func@ph),
618         \n@pow{10}{((\func@mag)/20)}}*sin(\func@ph)};
619       \opt@commands
620     \fi
621   \end{axis}
622 \end{tikzpicture}
623 }

```

**\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.

```

624 \newcommand{\addNyquistZPKPlot}[2][[]]{%
625   \gdef\func@mag{}%
626   \gdef\func@ph{}%
627   \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
628   \if@pgfarg
629     \addplot[variable=t,#1] ( {\n@pow{10}{((\func@mag)/20)}}*cos(\func@ph)},
630     {\n@pow{10}{((\func@mag)/20)}}*sin(\func@ph)} );
631   \else
632     \stepcounter{gnuplot@id}%
633     \addplot[variable=t,#1] gnuplot[parametric,gnuplot@prefix]{%
634       \n@pow{10}{((\func@mag)/20)}}*cos(\func@ph),
635       \n@pow{10}{((\func@mag)/20)}}*sin(\func@ph)};
636   \fi
637 }

```

**\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.

```

638 \newcommand{\addNyquistTFPlot}[2][[]]{%
639   \gdef\func@mag{}%
640   \gdef\func@ph{}%
641   \build@TF@plot{\func@mag}{\func@ph}{#2}%

```

```

642 \ifpgfarg
643   \addplot[variable=t,#1] ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
644     {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
645 \else
646   \stepcounter{gnuplot@id}%
647   \addplot[variable=t,#1] gnuplot[parametric,gnuplot@prefix]{%
648     \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
649     \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
650 \fi
651 }

```

**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.

```

652 \newenvironment{NyquistPlot}[3][]{%
653   \parse@env@opt{#1}%
654   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
655     panded\expandafter{\opt@tikz}]}%
656   \noexpand\begin{axis}{%
657     bode@style,
658     height=5cm,
659     domain=#2:#3,
660     xlabel={\Re$},
661     ylabel={\Im$},
662     \unexpanded\expandafter{\opt@axes}
663   }%
664   \temp@cmd
665   \addplot [only marks,mark=+,thick,red] (-1 , 0);
666 }{%
667   \end{axis}
668   \end{tikzpicture}
669 }

```

#### 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.

```

670 \newcommand{\parse@N@opt}[1]{%
671   \gdef\opt@axes{}%
672   \gdef\opt@plot{}%
673   \gdef\opt@commands{}%
674   \gdef\opt@tikz{}
675   \foreach \obj/\opt in {#1} {%
676     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0
677       \xdef\opt@axes{\unexpanded\expandafter{\opt}}%
678     \else
679       \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{plot}=0
680         \xdef\opt@plot{\unexpanded\expandafter{\opt}}%
681       \else
682         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{commands}=0
683           \xdef\opt@commands{\unexpanded\expandafter{\opt}}%
684         \else
685           \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{tikz}=0
686             \xdef\opt@tikz{\unexpanded\expandafter{\opt}}%
687           \else
688             \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
689               \unexpanded\expandafter{\obj}}%
690           \fi
691         \fi

```



```

692     \fi
693     \fi
694 }%
695 }

```

## 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 696 \newcommand{\NicholsZPK}[4][]{%
\addNicholsZPKChart 697   \parse@N@opt{#1}%
\addNicholsTFChart 698   \gdef\func@mag{}%
699   \gdef\func@ph{}%
700   \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
701   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]%
702     \noexpand\begin{axis}%
703     ph@x@filter,
704     bode@style,
705     domain=#3:#4,
706     height=5cm,
707     ylabel={Gain (dB)},
708     samples=500,
709     \unexpanded\expandafter{\opt@axes}
710   ]%
711 }%
712   \temp@cmd
713   \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt@plot]}%
714   \ifpgfarg
715     \temp@cmd ( {\func@ph} , {\func@mag} );
716     \opt@commands
717   \else
718     \stepcounter{gnuplot@id}%
719     \temp@cmd gnuplot[parametric,gnuplot@prefix]
720     { \func@ph , \func@mag };
721     \opt@commands
722   \fi
723   \end{axis}
724 \end{tikzpicture}
725 }
726 \newcommand{\NicholsTF}[4][]{%
727   \parse@N@opt{#1}%
728   \gdef\func@mag{}%
729   \gdef\func@ph{}%
730   \build@TF@plot{\func@mag}{\func@ph}{#2}%
731   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]%
732     \noexpand\begin{axis}%
733     ph@x@filter,
734     bode@style,
735     domain=#3:#4,
736     height=5cm,
737     ylabel={Gain (dB)},
738     samples=500,
739     \unexpanded\expandafter{\opt@axes}
740   ]%
741 }%
742   \temp@cmd
743   \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt@plot]}%
744   \ifpgfarg
745     \temp@cmd ( {\n@mod{\func@ph}{2*pi}} , {\func@mag} );
746     \opt@commands
747   \else

```

```

748         \stepcounter{gnuplot@id}%
749         \temp@cmd gnuplot[parametric,gnuplot@prefix]
750         { \n@mod{\func@ph}{2*pi} , \func@mag };
751         \opt@commands
752         \fi
753     \end{axis}
754 \end{tikzpicture}
755 }
756 \newenvironment{NicholsChart}[3][]{%
757     \parse@env@opt{#1}%
758     \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
759         panded\expandafter{\opt@tikz}]%
760         \noexpand\begin{axis}%
761         ph@x@filter,
762         bode@style,
763         domain=#2:#3,
764         height=5cm,
765         ylabel={Gain (dB)},
766         \unexpanded\expandafter{\opt@axes}
767     ]%
768     }%
769     \temp@cmd
770 }{
771     \end{axis}
772 \end{tikzpicture}
773 }
774 \newcommand{\addNicholsZPKChart}[2][]{%
775     \gdef\func@mag{}%
776     \gdef\func@ph{}%
777     \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
778     \ifpgfarg
779         \addplot[variable=t,#1] ( {\func@ph} , {\func@mag} );
780     \else
781         \stepcounter{gnuplot@id}%
782         \addplot[variable=t,#1] gnuplot[parametric,gnuplot@prefix]
783         {\func@ph , \func@mag};
784     \fi
785 }
786 \newcommand{\addNicholsTFChart}[2][]{%
787     \gdef\func@mag{}%
788     \gdef\func@ph{}%
789     \build@TF@plot{\func@mag}{\func@ph}{#2}%
790     \ifpgfarg
791         \addplot[variable=t,#1] ( {\n@mod{\func@ph}{2*pi}} , {\func@mag} );
792     \else
793         \stepcounter{gnuplot@id}%
794         \addplot[variable=t,#1] gnuplot[gnuplot@prefix]
795         {\n@mod{\func@ph}{2*pi} , \func@mag};
796     \fi
797 }

```

## 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	\@pgfargtrue	7	A
\@declutterargfalse	. 9		
\@declutterargtrue	. 11	\@radargfalse	13
\@pgfargfalse	5	\@radargtrue	15
		\add@feature	340, 359, 360, 363, 364, 366, 367, 375,

376, 379, 380, 382,  
 383, 390, 391, 394,  
 395, 397, 398, 414, 415  
 \addNodeComponentPlot ..... 274  
 \addNodeTFPlot .... 251  
 \addNodeZPKPlots .. 232  
 \addNicholsTFChart 692  
 \addNicholsZPKChart 692  
 \addNyquistTFPlot . 634  
 \addNyquistZPKPlot 620

**B**

\bode@style ..... 49  
 BodeMagPlot (env.) .. 301  
 BodePhPlot (env.) ... 282  
 BodePlot (env.) ..... 321  
 \BodeTF ..... 197  
 \BodeZPK ..... 166  
 \build@TF@plot ....  
 .. 201, 255, 257,  
 421, 591, 637, 726, 784  
 \build@ZPK@plot ...  
 .. 170, 237, 239,  
 354, 558, 623, 696, 772

**C**

\currentdegree . 433,  
 434, 437, 438, 439,  
 441, 442, 452, 453,  
 456, 457, 458, 460, 461

**D**

\den@degree .... 449, 452  
 \den@im 425, 457, 470, 473  
 \den@real ..... 424,  
 454, 460, 471, 473  
 \dencoeff ..... 448,  
 451, 454, 457, 460  
 \denpow 448, 449, 451, 452

**E**

environments:  
 BodeMagPlot .... 301  
 BodePhPlot ..... 282  
 BodePlot ..... 321

**F**

\func@mag ..... 168,  
 170, 190, 199, 201,  
 221, 556, 558, 574,  
 575, 580, 581, 589,  
 591, 607, 608, 613,  
 614, 621, 623, 625,  
 626, 630, 631, 635,  
 637, 639, 640, 644,  
 645, 694, 696, 711,  
 716, 724, 726, 741,  
 746, 770, 772, 774,  
 778, 782, 784, 786, 790  
 \func@ph ... 169, 170,  
 192, 200, 201, 224,  
 226, 557, 558, 574,  
 575, 580, 581, 590,

591, 607, 608, 613,  
 614, 622, 623, 625,  
 626, 630, 631, 636,  
 637, 639, 640, 644,  
 645, 695, 696, 711,  
 716, 725, 726, 741,  
 746, 771, 772, 774,  
 778, 783, 784, 786, 790

**G**

\gnuplot@id ..... 5  
 \gnuplot@prefix ..... 5

**I**

\if@declutterarg ..  
 ..... 9, 28, 44  
 \if@radarg ..... 13, 63  
 \ifcat ..... 341  
 \ifluatex ..... 2  
 \ifnum .... 236, 254,  
 260, 267, 350, 356,  
 358, 362, 372, 374,  
 378, 388, 389, 393,  
 402, 403, 407, 428,  
 434, 447, 453, 466,  
 486, 487, 490, 498,  
 499, 502, 510, 513,  
 516, 517, 523, 542,  
 545, 672, 675, 678, 681  
 \ifwindows ..... 43  
 \immediate ..... 45

**J**

\jobname ..... 32, 39

**L**

\Last@LoopValue 348, 350  
 \loop@delay 426, 467, 471

**M**

\MagCSPoles ..... 111  
 \MagCSPolesAsymp .. 111  
 \MagCSPolesLin .... 111  
 \MagCSPolesPeak ... 130  
 \MagCSZeros ..... 111  
 \MagCSZerosAsymp .. 111  
 \MagCSZerosLin .... 111  
 \MagCSZerosPeak ... 130  
 \MagDel ..... 76, 415  
 \MagK ..... 70, 398  
 \MagKAsymp ..... 70, 395  
 \MagKLin ..... 70, 391  
 \MagPole .... 78, 105, 383  
 \MagPoleAsymp 78, 107, 380  
 \MagPoleLin . 78, 106, 376  
 \MagSOPoles ..... 138  
 \MagSOPolesAsymp .. 138  
 \MagSOPolesLin .... 138  
 \MagSOPolesPeak ... 156  
 \MagS0Zeros ..... 138  
 \MagS0ZerosAsymp .. 138  
 \MagS0ZerosLin .... 138  
 \MagS0ZerosPeak ... 156  
 \MagZero ..... 105, 367

\MagZeroAsymp .. 105, 364  
 \MagZeroLin .... 105, 360

**N**

\n ..... 346, 348  
 \n@mod ... 5, 86, 224,  
 261, 741, 746, 786, 790  
 \n@pow .. 5, 79, 80, 81,  
 91, 92, 94, 95, 97,  
 98, 99, 100, 101,  
 102, 111, 112, 115,  
 116, 117, 119, 121,  
 122, 139, 143, 438,  
 439, 441, 442, 457,  
 458, 460, 461, 472,  
 473, 574, 575, 580,  
 581, 607, 608, 613,  
 614, 625, 626, 630,  
 631, 639, 640, 644, 645  
 \newcounter ..... 26  
 \newenvironment 282,  
 301, 321, 648, 752  
 \NicholsChart ..... 692  
 \NicholsTF ..... 692  
 \NicholsZPK ..... 692  
 \num@degree .... 430, 433  
 \num@im 423, 438, 470, 472  
 \num@real ..... 422,  
 435, 441, 470, 472  
 \numcoeff ..... 429,  
 432, 435, 438, 441  
 \numpow 429, 430, 432, 433  
 \NyquistPlot ..... 648  
 \NyquistTF ..... 587  
 \NyquistZPK ..... 554

**O**

\opt@approx 170, 481, 514  
 \opt@axes ..... 293,  
 313, 332, 539, 543,  
 548, 567, 600, 657,  
 667, 673, 705, 735, 761  
 \opt@commands .. 576,  
 582, 609, 615, 669,  
 679, 712, 717, 742, 747  
 \opt@group .....  
 .. 180, 211, 480, 511  
 \opt@plot .. 572, 605,  
 668, 676, 684, 709, 739  
 \opt@tikz .....  
 171, 202, 284, 303,  
 324, 484, 524, 540,  
 546, 559, 592, 650,  
 670, 682, 697, 727, 754  
 \optmag@axes ... 184,  
 215, 476, 500, 505  
 \optmag@commands ..  
 .. 191, 222, 483, 520  
 \optmag@plot ... 184,  
 215, 479, 488, 493, 526  
 \optph@axes .... 185,  
 216, 477, 503, 506  
 \optph@commands ...  
 .. 193, 228, 482, 518

<code>\optph@plot</code> . . . . . 185, 216, 478, 491, 494, 528	<code>\PhCSZerosLin</code> . . . . . 111 <code>\PhDel</code> . . . . . 77, 414 <code>\PhK</code> . . . . . 70, 397 <code>\PhKAsymp</code> . . . . . 70, 76, 394 <code>\PhKLin</code> . . . . . 70, 76, 390 <code>\PhPoLe</code> . . . . . 78, 108, 382 <code>\PhPoLeAsymp</code> 78, 110, 379 <code>\PhPoLeLin</code> . . 78, 109, 375 <code>\PhSOPoles</code> . . . . . 138 <code>\PhSOPolesAsymp</code> . . . 138 <code>\PhSOPolesLin</code> . . . . . 138 <code>\PhS0Zeros</code> . . . . . 138 <code>\PhS0ZerosAsymp</code> . . . 138 <code>\PhS0ZerosLin</code> . . . . . 138 <code>\PhZero</code> . . . . . 105, 366 <code>\PhZeroAsymp</code> . . . 105, 363 <code>\PhZeroLin</code> . . . . . 105, 359 <code>\plot@macro</code> . . . . . .. 234, 237, 239, 243, 247, 252, 255, 257, 261, 263, 268, 270	243, 246, 247, 284, 296, 303, 316, 324, 335, 559, 570, 572, 574, 579, 592, 603, 605, 607, 612, 650, 660, 697, 708, 709, 711, 715, 727, 738, 739, 741, 745, 754, 764 <code>\temp@macro</code> . . . . 235, 237, 239, 253, 255, 257 <code>\temp@mag@cmd</code> . . 184, 187, 190, 215, 218, 221 <code>\temp@ph@cmd</code> 185, 188, 192, 216, 219, 224, 226
<b>P</b>	<b>S</b>	<b>W</b>
<code>\p</code> . . . . . 373, 375, 376, 379, 380, 382, 383	<code>\setcounter</code> . . . . . 27 <code>\stepcounter</code> 187, 188, 218, 219, 245, 266, 278, 578, 611, 628, 642, 714, 744, 776, 788	<code>\write</code> . . . . . 45
<code>\parse@env@opt</code> . 283, 302, 323, 538, 649, 753	<code>\ph@filter</code> . . . . . 63 <code>\ph@x@filter</code> . . . . . 63 <code>\PhCSPOles</code> . . . . . 111 <code>\PhCSPOlesAsymp</code> 111, 148 <code>\PhCSPOlesLin</code> . . 111, 145	<b>Y</b>
<code>\parse@N@opt</code> . . . 555, 588, 666, 693, 723	<b>T</b>	<b>Z</b>
<code>\parse@opt</code> . 167, 198, 475	<code>\temp@cmd</code> . . . . . 171, 183, 202, 214, 242,	<code>\z</code> . . . . . 357, 359, 360, 363, 364, 366, 367
<code>\pdf@stricmp</code> . . . . . 3		
<code>\pdfstricmp</code> . . . . . . 1, 236, 254, 260, 267, 356, 358, 362, 372, 374, 378, 388, 389, 393, 402, 403, 407, 428, 447, 466, 486, 487, 490, 498, 499, 502, 510, 513, 516, 517, 523, 542, 545, 672, 675, 678, 681		

## Change History

v1.0	General: Initial release	1	\parse@env@opt: Added tikz option to environments	29
v1.0.1	\addBodeZPKPlots: Improved optional argument handling.	23	\parse@N@opt: Added commands and tikz options	32
	\BodeZPK: Pass arbitrary TikZ commands as options.	22	\parse@opt: Added Tikz option	28
v1.0.2			NyquistPlot: Added tikz option to environments	31
	gnuplot@prefix: Fixed issue #1	18	v1.0.4	
v1.0.3			General: Fixed unintended optional argument macro expansion	1
	BodePlot: Added tikz option to environments	25	v1.0.5	
	\BodeTF: Added Tikz option	22	\parse@opt: Fixed a bug	28
	\BodeZPK: Added Tikz option	22	v1.0.6	
	NicholsChart: Added tikz option to environments	32	General: Fixed issue #3	1
	\NicholsTF: Added commands and tikz options	32	v1.0.7	
	\NicholsZPK: Added commands and tikz options	32	General: Removed unnecessary semicolons	1
	gnuplot@prefix: Added jobname to gnuplot prefix	18	Updated documentation	1
	\NyquistTF: Added commands and tikz options	30	v1.0.8	
	\NyquistZPK: Added commands and tikz options	29	General: Added a new class option ‘declutter’	1
			\build@TF@plot: Included phase due to delay in wrapping.	27
			gnuplot@prefix: Fixed issue #6	18
			v1.1.0	
			General: Fixed phase wrapping in	

gnuplot mode . . . . .	1	environments for phase and magnitude plots . . . . .	24
<b>\addBodeTFPlot</b> : Fixed phase wrapping in gnuplot mode . . . . .	23	<b>BodePlot</b> : Deprecated BodePlot environment . . . . .	25
<b>BodeMagPlot</b> : Added separate environments for phase and magnitude plots . . . . .	24	<b>\BodeTF</b> : Fixed phase wrapping in gnuplot mode . . . . .	22
<b>BodePhPlot</b> : Added separate			