

The **bodeplot** package

version 1.1.5

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

January 11, 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 $K \in \mathbb{R}$ 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

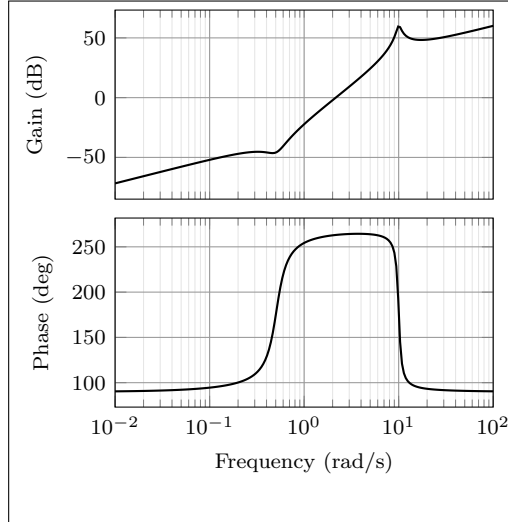
- If French language support is needed through the `babel` package, **the babel package has to be loaded before the bodeplot package**. The options `french` or `main=french` need to be passed to `babel` locally, and not via class options.
- In `pgf` mode, Bode phase plots and Nichols charts in TF form 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)

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

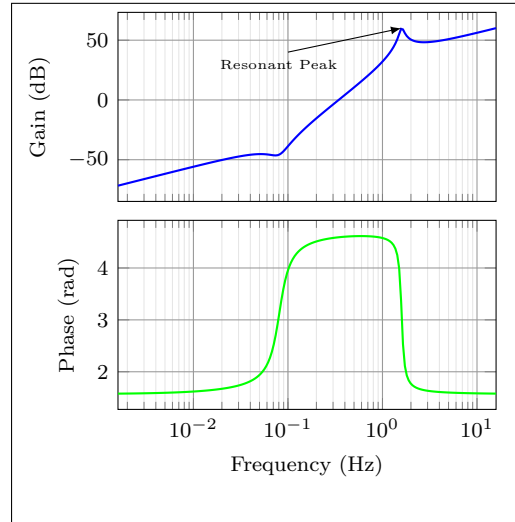
Bode plot in ZPK format



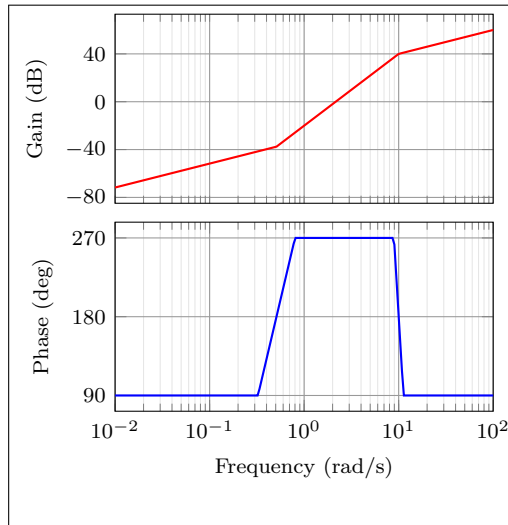
```
\BodeZPK{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
{0.01}
{100}
```

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)}
```



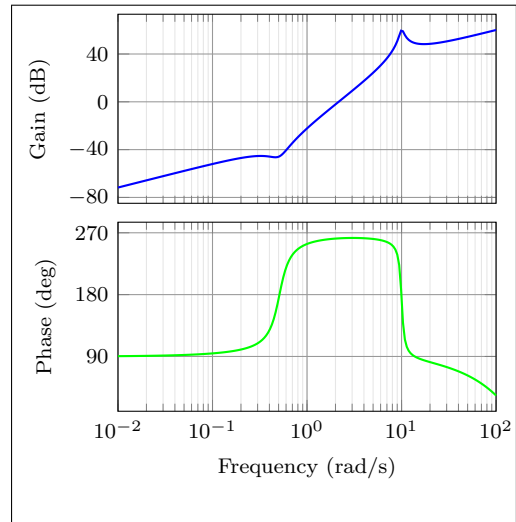
Linear approximation with customization



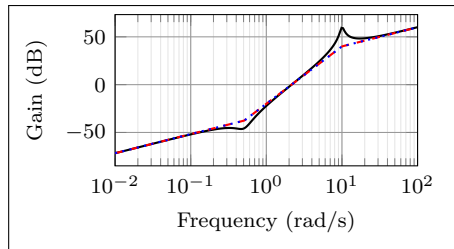
```
\BodeZPK[%
  plot/mag/{red,thick},
  plot/ph/{blue,thick},
  axes/mag/{ytick distance=40},
  axes/ph/{ytick distance=90},
  approx/linear%
]{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
{0.01}
{100}
```

Plot with delay and customization

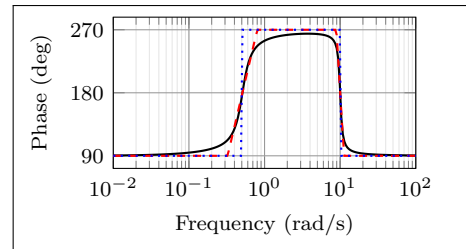
```
\BodeZPK[%
  plot/mag/{blue,thick},
  plot/ph/{green,thick},
  axes/mag/{ytick distance=40},
  axes/ph/{ytick distance=90%}
]{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01%
}
{0.01}
{100}
```



Individual gain and phase plots with more customization

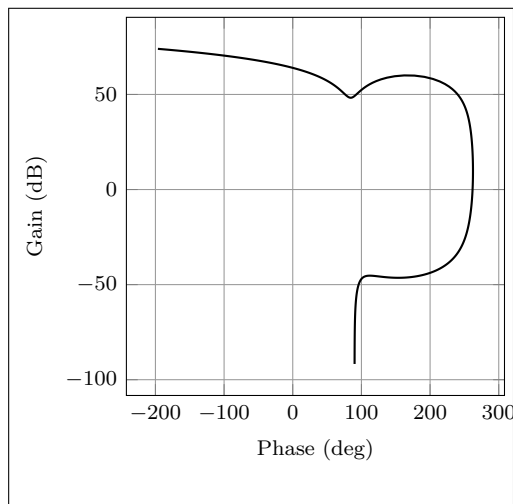


```
\begin{BodeMagPlot}[%
  axes/{height=2cm,
    width=4cm}
]
{0.01}
{100}
\addBodeZPKPlots[%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
]
{magnitude}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\end{BodeMagPlot}
```



```
\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}
```

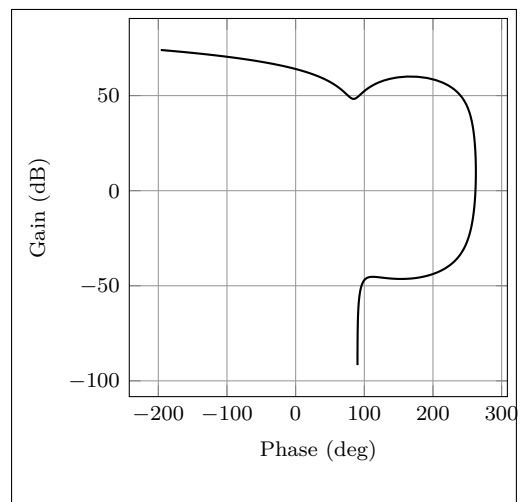
Nichols chart



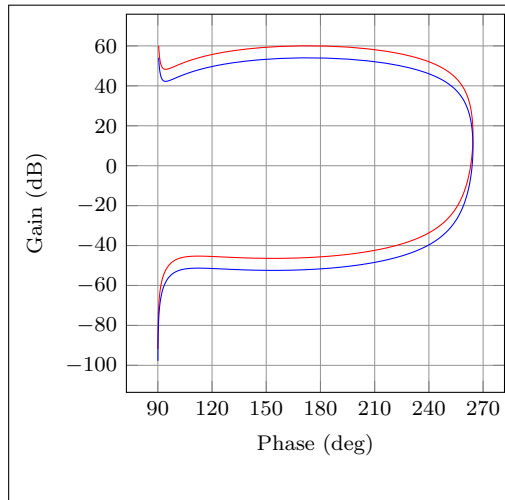
```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01%
}
{0.001}
{500}
```

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}
```



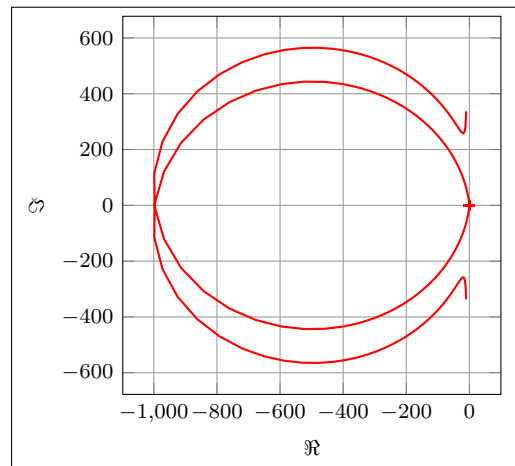
Multiple Nichols charts with customization



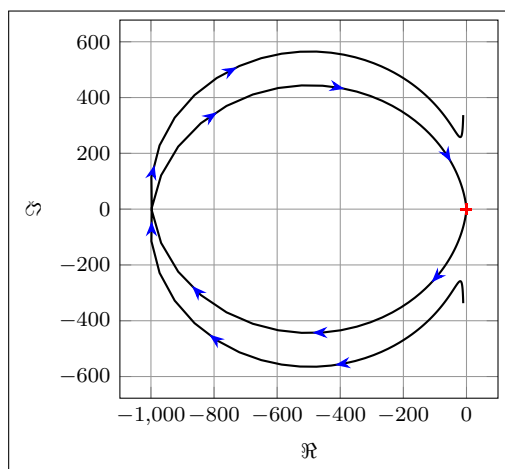
```
\begin{NicholsChart}[%
  ytick distance=20,
  xtick distance=30
]
{0.001}
{100}
\addNicholsZPKChart [red,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
\addNicholsZPKChart [blue,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
 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



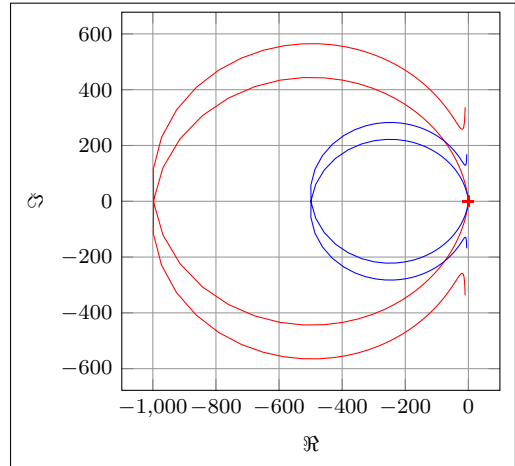
Nyquist plot in TF format with arrows



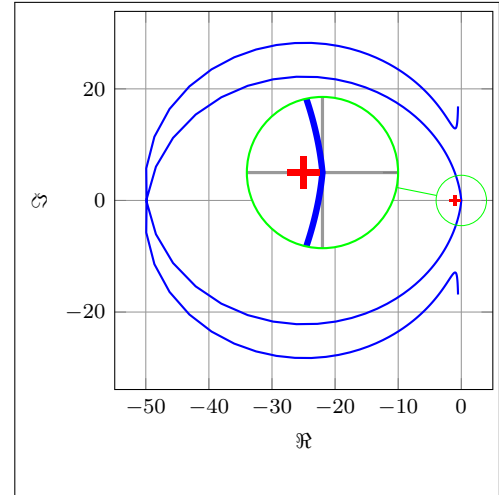
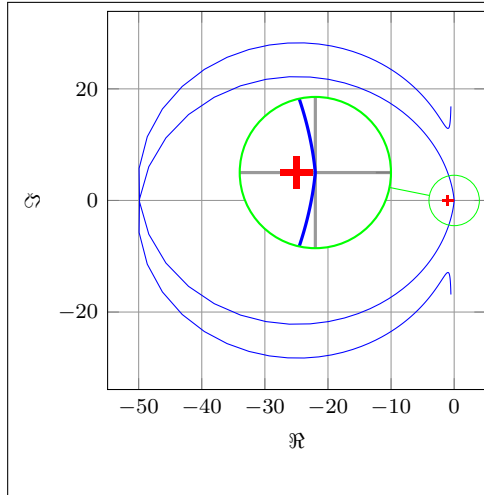
```
\NyquistTF[%
  plot/{%
    samples=1000,
    postaction=decorate,
    decoration={%
      markings,
      mark=between positions 0.1 and 0.9 step 5mm with {%
        \arrow{Stealth [length=2mm, blue]}
      }
    }
  }%
]
{%
  num/{10,2,2.6,0},
  den/{1,1,100.25}%
}
{-30}
{30}
```

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

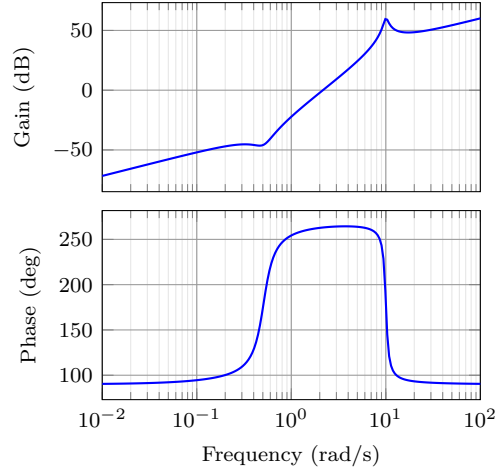


Figure 1: Output of the `\BodeZPK` macro.

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

For example, given a transfer function

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

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

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

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

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

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

which generates the plot in Figure 2.

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

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

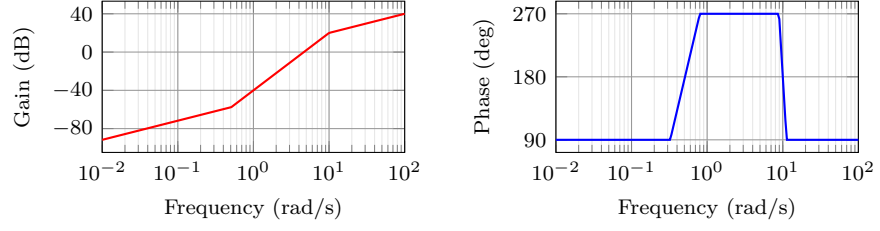


Figure 2: Customization of the default `\BodeZPK` macro.

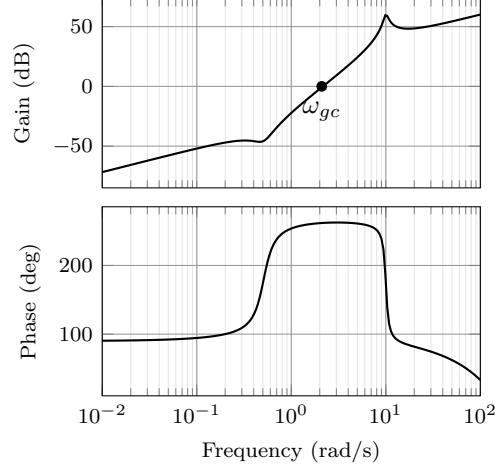


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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 \rangle, \langle obj2/\langle opt2 \rangle \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, ...]
                   {\langle min-frequency \rangle} {\langle max-frequency \rangle}
                   \addBode...
                   \end{BodePhPlot}
```

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

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

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

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

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

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

which generates the plot in Figure 4.

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

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

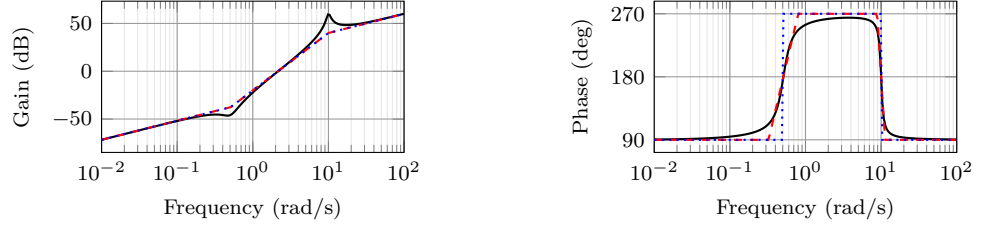


Figure 4: Superimposed approximate and true Bode plots using the **BodeMagPlot** and **BodePhPlot** environments and the **\addBodeZPKPlots** macro.

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

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

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

3.1.1 Basic components up to first order

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

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

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

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

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

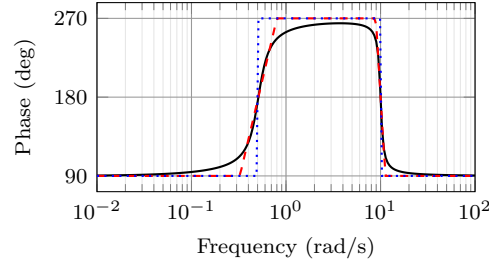


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePh-Plot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

```

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

```

which gives us the plot in Figure 5.

3.1.2 Basic components of the second order

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

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

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

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

```

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

```

generates the plot in Figure 6.

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

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

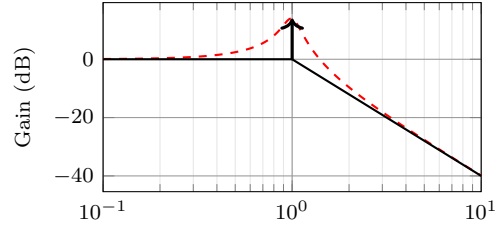


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

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

`\MagCSFeaturePeak` `\MagCSFeaturePeak[$\langle draw-options \rangle$]{ $\langle zeta \rangle$ }{ $\langle \omega_n \rangle$ }`

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[$\langle draw-options \rangle$]{ $\langle real-part \rangle$ }{ $\langle imaginary-part \rangle$ }`

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

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

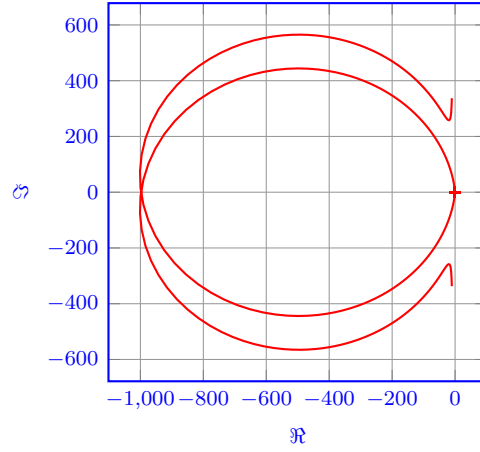


Figure 7: Output of the `\NyquistZPK` macro.

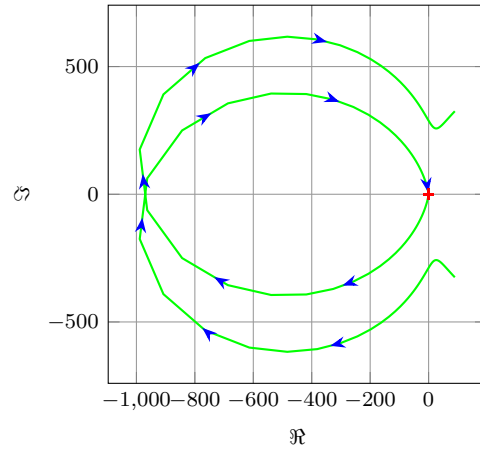


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

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

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

generates the Nyquist plot in Figure 8.

```
NyquistPlot (env.) \begin{NyquistPlot}[\langle obj1 \rangle/{\langle opt1 \rangle},obj2/{\langle opt2 \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.
- **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:

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

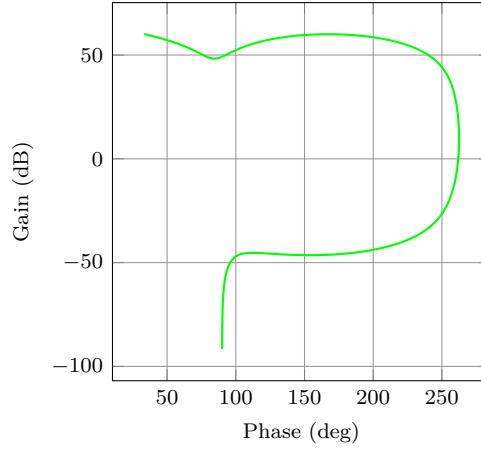


Figure 9: Output of the `\NyquistZPK` macro.

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

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

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

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

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

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

4 Implementation

4.1 Initialization

```
\n@mod We start by processing the class options.
\n@mod@p 1 \newif\if@pgfarg\@pgfargfalse
\n@mod@n 2 \DeclareOption{pgf}{
\n@pow 3 \@pgfargtrue
gnuplot@id 4 }
gnuplot@prefix 5 \newif\if@declutterarg\@declutterargfalse
6 \DeclareOption{declutter}{
7 \@declutterargtrue
8 }
9 \newif\if@radarg\@radargfalse
10 \DeclareOption{rad}{
11 \@radargtrue
12 }
13 \newif\if@hzarg\@hzargfalse
14 \DeclareOption{Hz}{
15 \@hzargtrue
16 }
17 \ProcessOptions\relax
```

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

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

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

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

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

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

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

```

47 \newif\if@babel@french\@babel@frenchfalse
48 \@ifpackagewith{babel}{french}\@babel@frenchtrue{}
49 \@ifpackagewith{babel}{main=french}\@babel@frenchtrue{}

```

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

```

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

```

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 65 \pgfplotsset{freq@filter/.style = {}}
ph@x@label 66 \def\freq@scale{1}
ph@y@label 67 \pgfplotsset{freq@label/.style = {xlabel = {Frequency (rad/s)}}}
68 \pgfplotsset{ph@x@label/.style = {xlabel={Phase (deg)}}}
69 \pgfplotsset{ph@y@label/.style = {ylabel={Phase (deg)}}}
70 \def\ph@scale{180/pi}
71 \if@radarg
72   \pgfplotsset{ph@y@label/.style = {ylabel={Phase (rad)}}}
73   \pgfplotsset{ph@x@label/.style = {xlabel={Phase (rad)}}}
74   \def\ph@scale{1}
75 \fi
76 \if@hzarg
77   \def\freq@scale{2*pi}
78   \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}
79   \if@pgfarg
80     \pgfplotsset{freq@filter/.style = {x filter/.expression={x-
      log10(2*pi)}}}
81   \fi
82 \fi
83 \tikzset{
84   phase unit/.initial={deg},
85   phase unit/.default={deg},
86   phase unit/.is choice,
87   phase unit/deg/.code={
88     \renewcommand{\ph@scale}{180/pi}
89     \pgfplotsset{ph@x@label/.style = {xlabel={Phase (deg)}}}
90     \pgfplotsset{ph@y@label/.style = {ylabel={Phase (deg)}}}
91   },
92   phase unit/rad/.code={
93     \renewcommand{\ph@scale}{1}
94     \pgfplotsset{ph@y@label/.style = {ylabel={Phase (rad)}}}
95     \pgfplotsset{ph@x@label/.style = {xlabel={Phase (rad)}}}
96   },
97   frequency unit/.initial={rad},
98   frequency unit/.default={rad},
99   frequency unit/.is choice,
100  frequency unit/Hz/.code={
101    \renewcommand{\freq@scale}{2*pi}
102    \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}

```

```

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

```

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

```

get@interval@end 112 \def\get@interval@start#1:#2\@nil{#1}
113 \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 114 \newcommand*\MagK[2]{(20*log10(abs(#1)))}
\PhKLin 115 \newcommand*\MagKAsymp{\MagK}
116 \newcommand*\MagKLin{\MagK}
117 \newcommand*\PhK[2]{((#1<0?-pi:0)*\ph@scale)}
118 \newcommand*\PhKAsymp{\PhK}
119 \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.

```

120 \newcommand*\MagDel[2]{0}
121 \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 122 \newcommand*\MagPole[2]
\PhPole 123 {(-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 124 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
125 -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :
126 -20*log10(t)
127 )}

```

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

```

128 \newcommand*\MagPoleAsymp{\MagPoleLin}

```

Parametric function for the true phase of a complex pole.

```

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

```

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

```

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

```

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

```

146 \newcommand*\PhPoleAsymp}[2]{((t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) ?
147   (-atan2(-(#2),-(#1))) :
148   (#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 149 \newcommand*\MagZero{0-\MagPole}
\PhZero     150 \newcommand*\MagZeroLin{0-\MagPoleLin}
\PhZeroAsymp 151 \newcommand*\MagZeroAsymp{0-\MagPoleAsymp}
\PhZeroLin  152 \newcommand*\PhZero{0-\PhPole}
            153 \newcommand*\PhZeroLin{0-\PhPoleLin}
            154 \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 155 \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles     - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2})))}
\PhCSPolesAsymp 157 \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
\PhCSPolesLin  40*log10(t))}
\MagCSZeros    158 \newcommand*\MagCSPolesAsymp{\MagCSPolesLin}

```

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

```

\MagCSZerosLin 159 \newcommand*\PhCSPoles}[2]{((-atan2((2*(#1)*(#2)*t),(\n@pow{#2}{2}
\PhCSZeros     - \n@pow{t}{2}))*\ph@scale)}
\PhCSZerosAsymp 160 \newcommand*\PhCSPolesLin}[2]{((t < (#2 / (\n@pow{10}{abs(#1)})) ?
\PhCSZerosLin  161 0 :
162 (t >= (#2 * (\n@pow{10}{abs(#1)})) ?
163 (#1>0 ? -pi : pi) :
164 (#1>0 ? (-pi*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
165 (pi*(log10(t*(\n@pow{10}{abs(#1)})/#2))/(2*abs(#1)))))*\ph@scale)}
166 \newcommand*\PhCSPolesAsymp}[2]{((#1>0?(t<#2?0:-
167 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).

```

168 \newcommand*\MagCSZeros{0-\MagCSPoles}
169 \newcommand*\MagCSZerosLin{0-\MagCSPolesLin}
170 \newcommand*\MagCSZerosAsymp{0-\MagCSPolesAsymp}
171 \newcommand*\PhCSZeros{0-\PhCSPoles}

```

```

172 \newcommand*\PhCSZerosLin}{0-\PhCSPolesLin}
173 \newcommand*\PhCSZerosAsymp}{0-\PhCSPolesAsymp}

```

\MagCSPolesPeak 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.

```

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

```

\MagS0Poles 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.

```

\MagS0PolesAsymp
\MagS0PolesLin 182 \newcommand*\MagS0Poles}[2]{
\PhS0Poles      183   (-20*log10(sqrt(\n@pow{#2 - \n@pow{t}{2}}{2} + \n@pow{#1*t}{2})))}
\PhS0PolesAsymp 184 \newcommand*\MagS0PolesLin}[2]{
\PhS0PolesLin   185   (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t))}
\MagS0Zeros     186 \newcommand*\MagS0PolesAsymp}{\MagS0PolesLin}

```

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

```

\MagS0ZerosLin
\PhS0Zeros      187 \newcommand*\PhS0Poles}[2]{((-atan2((#1)*t,((#2) -
\PhS0ZerosAsymp \n@pow{t}{2}))) * \ph@scale)}
\PhS0ZerosLin   188 \newcommand*\PhS0PolesLin}[2]{((#2>0 ?
189   \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
190   (#1>0 ? -pi : pi))}
191 \newcommand*\PhS0PolesAsymp}[2]{((#2>0 ?
192   \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
193   (#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).

```

194 \newcommand*\MagS0Zeros}{0-\MagS0Poles}
195 \newcommand*\MagS0ZerosLin}{0-\MagS0PolesLin}
196 \newcommand*\MagS0ZerosAsymp}{0-\MagS0PolesAsymp}
197 \newcommand*\PhS0Zeros}{0-\PhS0Poles}
198 \newcommand*\PhS0ZerosLin}{0-\PhS0PolesLin}
199 \newcommand*\PhS0ZerosAsymp}{0-\PhS0PolesAsymp}

```

\MagS0PolesPeak 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.

```

200 \newcommand*\MagS0PolesPeak}[3][]{
201   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
202     (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
203       20*log10(abs(#2/sqrt(abs(#3))))});
204 }
205 \newcommand*\MagS0ZerosPeak}[3][]{
206   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
207     (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
208       20*log10(abs(#2/sqrt(abs(#3))))});
209 }

```

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`.

```
210 \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.

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

```
228 \edef\temp@mag@cmd{\noexpand\nextgroupplot [yla-
    bel={Gain (dB)}, xmajorticks=false, \optmag@axes]
229 \noexpand\addplot [freq@filter, variable=t, thick, \opt-
    mag@plot]}
230 \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
231 \noexpand\addplot [freq@filter, variable=t, thick, \optph@plot]}
232 \if@pgfarg
233 \temp@mag@cmd {\func@mag};
234 \optmag@commands
235 \temp@ph@cmd {\func@ph};
236 \optph@commands
237 \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.

```
238 \stepcounter{gnuplot@id}
239 \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
240 { set table $meta;
241   set dummy t;
242   set logscale x 10;
243   set xrange [#3*\freq@scale:#4*\freq@scale];
244   set samples \pgfkeysvalueof{/pgfplots/samples};
245   plot \func@mag;
246   set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
247   plot "$meta" using ($1/(\freq@scale)):($2);
248 };
```

```

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

```

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

```

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

```

\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.

```

277 \newcommand{\BodeTF}[4][]{
278   \parse@opt{#1}
279   \gdef\func@mag{}
280   \gdef\func@ph{}
281   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
282     panded\expandafter{\opt@tikz}]}
283   \temp@cmd
284   \build@TF@plot{\func@mag}{\func@ph}{#2}
285   \edef\temp@cmd{\noexpand\begin{groupplot}[
286     bode@style,
287     xmin=#3,
288     xmax=#4,
289     domain=#3*\freq@scale:#4*\freq@scale,
290     height=2.5cm,
291     xmode=log,
292     group style = {group size = 1 by 2,vertical sep=0.25cm},
293     \opt@group
294   ]}
295   \temp@cmd
296   \edef\temp@mag@cmd{\noexpand\nextgroupplot [yla-
297     bel={Gain (dB)}, xmajor ticks=false, \optmag@axes]
298     \noexpand\addplot [freq@filter, variable=t, thick, \opt-
299     mag@plot]}
300   \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
301     \noexpand\addplot [freq@filter, variable=t, thick, \optph@plot]}
302   \if@pgfarg
303     \temp@mag@cmd {\func@mag};
304     \optmag@commands
305     \temp@ph@cmd {\n@mod{\func@ph}{2*pi*\ph@scale}};
306     \optph@commands

```



```

304     \else
305         \stepcounter{gnuplot@id}
306         \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
307         { set table $meta;
308           set dummy t;
309           set logscale x 10;
310           set xrange [#3*\freq@scale:#4*\freq@scale];
311           set samples \pgfkeysvalueof{/pgfplots/samples};
312           plot \func@mag;
313           set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
314           plot "$meta" using ($1/(\freq@scale)):($2);
315         };
316         \optmag@commands
317         \stepcounter{gnuplot@id}
318         \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
319         { set table $meta;
320           set dummy t;
321           set logscale x 10;
322           set trange [#3*\freq@scale:#4*\freq@scale];
323           set samples \pgfkeysvalueof{/pgfplots/samples};
324           plot '+' using (t) : ((\func@ph)/(\ph@scale)) smooth unwrap;
325           set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
326           plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
327         };
328         \optph@commands
329     \fi
330 \end{groupplot}
331 \end{tikzpicture}
332 }

```

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

```

333 \if@babel@french
334 \let\Orig@BodeTF\BodeTF
335 \renewcommand{\BodeTF}{%
336   \shorthandoff{;:!?}%
337   \BodeTF@Shorthandoff
338 }
339 \newcommand{\BodeTF@Shorthandoff}[4][[]]{%
340   \Orig@BodeTF[#1]{#2}{#3}{#4}%
341   \shorthandon{;:!?}%
342 }
343 \fi

```

\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.

```

344 \newcommand{\addBodeZPKPlots}[3][true/{}]{
345   \foreach \approx/\opt in {#1} {
346     \gdef\plot@macro{
347       \gdef\temp@macro{
348         \ifnum\pdf@strcmp{#2}{phase}=0
349           \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}
350         \else
351           \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}
352         \fi
353         \if@pgfarg
354           \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, thick, \opt]}
355           \temp@cmd {\plot@macro};
356         \else
357           \stepcounter{gnuplot@id}
358           \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \opt]}
359           \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
360           { set table $meta;

```

```

361         set dummy t;
362         set logscale x 10;
363         set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
364         set samples \pgfkeysvalueof{/pgfplots/samples};
365         plot \plot@macro;
366         set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
367         plot "$meta" using ($1/(\freq@scale)):(($2));
368     };
369 \fi
370 }
371 }

```

\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.

```

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

```

\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.

```

415 \newcommand{\addBodeComponentPlot}[2][thick]{
416   \if@pgfarg
417     \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, #1]}
418     \temp@cmd {#2};
419   \else
420     \stepcounter{gnuplot@id}
421     \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
422     { set table $meta;
423       set dummy t;
424       set logscale x 10;
425       set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
426       set samples \pgfkeysvalueof{/pgfplots/samples};
427       plot #2;
428       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
429       plot "$meta" using ($1/(\freq@scale)):(#2);
430     };
431   \fi
432 }

```

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.

```

433 \if@babel@french
434   \AddToHook{env/BodePhPlot/begin}{\shorthandoff{;:!?}}
435 \fi
436 \NewDocumentEnvironment{BodePhPlot}{0}{mm+b}{
437   \parse@env@opt{#1}
438   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
439   \temp@cmd
440   \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
441     ph@y@label,
442     freq@label,
443     bode@style,
444     xmin={#2},
445     xmax={#3},
446     domain=#2:#3,
447     height=2.5cm,
448     \unexpanded\expandafter{\opt@axes}
449   ]}
450   \temp@cmd
451   #4
452   \end{semilogxaxis}
453 \end{tikzpicture}
454 }{}

```

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.

```

455 \if@babel@french
456   \AddToHook{env/BodeMagPlot/begin}{\shorthandoff{;:!?}}
457 \fi
458 \NewDocumentEnvironment{BodeMagPlot}{0}{mm+b}{
459   \parse@env@opt{#1}
460   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
461   \temp@cmd
462   \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
463     bode@style,

```

```

464     freq@label,
465     xmin={#2},
466     xmax={#3},
467     domain=#2:#3,
468     height=2.5cm,
469     ylabel={Gain (dB)},
470     \unexpanded\expandafter{\opt@axes}
471   ]]
472   \temp@cmd
473   #4
474   \end{semilogxaxis}
475 \end{tikzpicture}
476 }{}

```

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

```

477 \if@babel@french
478   \AddToHook{env/BodePlot/begin}{\shorthandoff{;:!?}}
479 \fi
480 \NewDocumentEnvironment{BodePlot}{0}{mm+b}{
481   \parse@env@opt{#1}
482   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
483     panded\expandafter{\opt@tikz}]]
484   \temp@cmd
485   \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
486     bode@style,
487     freq@label,
488     xmin={#2},
489     xmax={#3},
490     domain=#2:#3,
491     height=2.5cm,
492     \unexpanded\expandafter{\opt@axes}
493   ]}
494   \temp@cmd
495   #4
496   \end{semilogxaxis}
497 \end{tikzpicture}
498 }{}

```

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](#).

```

498 \newcommand*\add@feature}[3]{
499   \ifcat$\detokenize\expandafter{#1}$
500     \xdef#1{\unexpanded\expandafter{#1 0+#2}}
501   \else
502     \xdef#1{\unexpanded\expandafter{#1+#2}}
503   \fi
504   \foreach \y [count=\n] in #3 {
505     \xdef#1{\unexpanded\expandafter{#1}{\y}}
506     \xdef\Last@LoopValue{\n}
507   }
508   \ifnum\Last@LoopValue=1
509     \xdef#1{\unexpanded\expandafter{#1}{0}}
510   \fi
511 }

```

\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.

```

512 \newcommand{\build@ZPK@plot}[4]{
513   \foreach \feature/\values in {#4} {
514     \ifnum\pdf@strcmp{\feature}{z}=0
515       \foreach \z in \values {
516         \ifnum\pdf@strcmp{#3}{linear}=0
517           \add@feature{#2}{\PhZeroLin}{\z}
518           \add@feature{#1}{\MagZeroLin}{\z}
519         \else
520           \ifnum\pdf@strcmp{#3}{asymptotic}=0
521             \add@feature{#2}{\PhZeroAsymp}{\z}
522             \add@feature{#1}{\MagZeroAsymp}{\z}
523           \else
524             \add@feature{#2}{\PhZero}{\z}
525             \add@feature{#1}{\MagZero}{\z}
526           \fi
527         \fi
528       }
529     \fi
530     \ifnum\pdf@strcmp{\feature}{p}=0
531       \foreach \p in \values {
532         \ifnum\pdf@strcmp{#3}{linear}=0
533           \add@feature{#2}{\PhPoleLin}{\p}
534           \add@feature{#1}{\MagPoleLin}{\p}
535         \else
536           \ifnum\pdf@strcmp{#3}{asymptotic}=0
537             \add@feature{#2}{\PhPoleAsymp}{\p}
538             \add@feature{#1}{\MagPoleAsymp}{\p}
539           \else
540             \add@feature{#2}{\PhPole}{\p}
541             \add@feature{#1}{\MagPole}{\p}
542           \fi
543         \fi
544       }
545     \fi
546     \ifnum\pdf@strcmp{\feature}{k}=0
547       \ifnum\pdf@strcmp{#3}{linear}=0
548         \add@feature{#2}{\PhKLin}{\values}
549         \add@feature{#1}{\MagKLin}{\values}
550       \else
551         \ifnum\pdf@strcmp{#3}{asymptotic}=0
552           \add@feature{#2}{\PhKAsymp}{\values}
553           \add@feature{#1}{\MagKAsymp}{\values}
554         \else
555           \add@feature{#2}{\PhK}{\values}
556           \add@feature{#1}{\MagK}{\values}
557         \fi
558       \fi
559     \fi
560     \ifnum\pdf@strcmp{\feature}{d}=0
561       \ifnum\pdf@strcmp{#3}{linear}=0
562         \PackageError {bodeplot} {Linear approximation for pure de-
563           lays is not
564             supported.} {Plot the true Bode plot using 'true' in-
565             stead of 'linear'.}
566       \else
567         \ifnum\pdf@strcmp{#3}{asymptotic}=0
568           \PackageError {bodeplot} {Asymptotic approxima-

```

```

tion for pure delays is not
567     supported.} {Plot the true Bode plot using 'true' in-
stead of 'asymptotic'.}
568     \else
569         \ifdim\values pt < 0pt
570             \PackageError {bodeplot} {Delay needs to be a posi-
tive number.}
571         \fi
572         \add@feature{#2}{\PhDel}{\values}
573         \add@feature{#1}{\MagDel}{\values}
574     \fi
575 \fi
576 \fi
577 }
578 }

```

`\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).

```

579 \newcommand{\build@TF@plot}[3]{
580     \gdef\num@real{0}
581     \gdef\num@im{0}
582     \gdef\den@real{0}
583     \gdef\den@im{0}
584     \gdef\loop@delay{0}
585     \foreach \feature/\values in {#3} {
586         \ifnum\pdf@strcmp{\feature}{num}=0
587             \foreach \numcoeff [count=\numpow] in \values {
588                 \xdef\num@degree{\numpow}
589             }
590             \foreach \numcoeff [count=\numpow] in \values {
591                 \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}
592                 \ifnum\currentdegree = 0
593                     \xdef\num@real{\num@real+\numcoeff}
594                 \else
595                     \ifodd\currentdegree
596                         \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-
1}}{(\currentdegree-1)/2}))*%
597                         (\n@pow{t}{\currentdegree})))}
598                     \else
599                         \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-
1}}{(\currentdegree)/2}))*%
600                         (\n@pow{t}{\currentdegree})))}
601                     \fi
602                 \fi
603             }
604         \fi
605         \ifnum\pdf@strcmp{\feature}{den}=0
606             \foreach \dencoeff [count=\denpow] in \values {
607                 \xdef\den@degree{\denpow}
608             }
609             \foreach \dencoeff [count=\denpow] in \values {
610                 \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}
611                 \ifnum\currentdegree = 0
612                     \xdef\den@real{\den@real+\dencoeff}
613                 \else
614                     \ifodd\currentdegree
615                         \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-
1}}{(\currentdegree-1)/2}))*%
616                         (\n@pow{t}{\currentdegree})))}
617                     \else
618                         \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-

```

```

1}{(\currentdegree)/2})*%
619      (\n@pow{t}{\currentdegree}}))}
620      \fi
621      \fi
622    }
623    \fi
624    \ifnum\pdf@strcmp{\feature}{d}=0
625      \xdef\loop@delay{\values}
626    \fi
627  }
628  \xdef#2{((atan2((\num@im),(\num@real))-atan2((\den@im),%
629    (\den@real))-\loop@delay*t)*(\ph@scale))}
630  \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2}))) -
%
631    20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2}))))}
632 }

```

\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.

```

633 \newcommand{\parse@opt}[1]{
634   \gdef\optmag@axes{}
635   \gdef\optph@axes{}
636   \gdef\optph@plot{}
637   \gdef\optmag@plot{}
638   \gdef\opt@group{}
639   \gdef\opt@approx{}
640   \gdef\optph@commands{}
641   \gdef\optmag@commands{}
642   \gdef\opt@tikz{}
643   \foreach \obj/\typ/\opt in {#1} {
644     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
645       \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
646         \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
647       \else
648         \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
649           \xdef\optph@plot{\unexpanded\expandafter{\opt}}
650         \else
651           \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
652           \xdef\optph@plot{\unexpanded\expandafter{\opt}}
653         \fi
654       \fi
655     \else
656       \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
657         \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
658           \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
659         \else
660           \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
661             \xdef\optph@axes{\unexpanded\expandafter{\opt}}
662           \else
663             \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
664             \xdef\optph@axes{\unexpanded\expandafter{\opt}}
665           \fi
666         \fi
667       \else
668         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{group}=0
669           \xdef\opt@group{\unexpanded\expandafter{\opt}}
670         \else

```

```

671         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{approx}=0
672         \xdef\opt@approx{\unexpanded\expandafter{\opt}}
673     \else
674         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
675         \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
676         \xdef\optph@commands{\unexpanded\expandafter{\opt}}
677         \else
678         \xdef\optmag@commands{\unexpanded\expandafter{\opt}}
679         \fi
680     \else
681         \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
682         \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
683     \else
684         \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
685         \unexpanded\expandafter{\obj}}
686         \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
687         \unexpanded\expandafter{\obj}}
688     \fi
689 \fi
690 \fi
691 \fi
692 \fi
693 \fi
694 }
695 }

```

`\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.

```

696 \newcommand{\parse@env@opt}[1]{
697   \gdef\opt@axes{}
698   \gdef\opt@tikz{}
699   \foreach \obj/\opt in {#1} {
700     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
701     \xdef\opt@axes{\unexpanded\expandafter{\opt}}
702   \else
703     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
704     \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
705   \else
706     \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
707     \unexpanded\expandafter{\obj}}
708   \fi
709 \fi
710 }
711 }

```

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!

```

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

```



```

713 \parse@N@opt{#1}
714 \gdef\func@mag{}
715 \gdef\func@ph{}
716 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
717 \temp@cmd
718 \build@ZPK@plot{\func@mag}{\func@ph}{{#2}}
719 \edef\temp@cmd{\noexpand\begin{axis}[
720     bode@style,
721     domain=#3*\freq@scale:#4*\freq@scale,
722     height=5cm,
723     xlabel={\$Re\$},
724     ylabel={\$Im\$},
725     samples=500,
726     \unexpanded\expandafter{\opt@axes}
727 ]}
728 \temp@cmd
729 \addplot [only marks,mark=+,thick,red] (-1 , 0);
730 \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \unex-
panded\expandafter{\opt@plot}]}
731 \ifpgfarg
732 \temp@cmd ( {\n@pow{10}}{((\func@mag)/20))*cos((\func@ph)/(\ph@scale))},
733 {\n@pow{10}}{((\func@mag)/20))*sin((\func@ph)/(\ph@scale))} );
734 \opt@commands
735 \else
736 \stepcounter{gnuplot@id}
737 \temp@cmd gnuplot [parametric, gnuplot@prefix] {
738     \n@pow{10}}{((\func@mag)/20))*cos((\func@ph)/(\ph@scale)),
739     \n@pow{10}}{((\func@mag)/20))*sin((\func@ph)/(\ph@scale))
740 };
741 \opt@commands
742 \fi
743 \end{axis}
744 \end{tikzpicture}
745 }

```

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

```

746 \if@babel@french
747 \let\Orig@NyquistZPK\NyquistZPK
748 \renewcommand{\NyquistZPK}{%
749     \shorthandoff{;:!?}%
750     \NyquistZPK@Shorthandoff
751 }
752 \newcommand{\NyquistZPK@Shorthandoff}[4][]{%
753     \Orig@NyquistZPK[#1]{#2}{#3}{#4}%
754     \shorthandon{;:!?}%
755 }
756 \fi

```

\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**.

```

757 \newcommand{\NyquistTF}[4][]{
758     \parse@N@opt{#1}
759     \gdef\func@mag{}
760     \gdef\func@ph{}
761     \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
762     \temp@cmd
763     \build@TF@plot{\func@mag}{\func@ph}{{#2}}
764     \edef\temp@cmd{\noexpand\begin{axis}[
765         bode@style,
766         domain=#3*\freq@scale:#4*\freq@scale,
767         height=5cm,

```

```

768     xlabel={\$Re\$},
769     ylabel={\$Im\$},
770     samples=500,
771     \unexpanded\expandafter{\opt@axes}
772   }}
773   \temp@cmd
774     \addplot [only marks, mark=+, thick, red] (-1 , 0);
775   \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \unex-
panded\expandafter{\opt@plot}]}
776   \ifpgfarg
777     \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
778       {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
779   \opt@commands
780   \else
781     \stepcounter{gnuplot@id}
782     \temp@cmd gnuplot [parametric, gnuplot@prefix] {
783       \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
784       \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
785     };
786   \opt@commands
787   \fi
788   \end{axis}
789 \end{tikzpicture}
790 }

```

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

```

791 \if@babel@french
792   \let\Orig@NyquistTF\NyquistTF
793   \renewcommand{\NyquistTF}{%
794     \shorthandoff{;:!?}%
795     \NyquistTF@Shorthandoff
796   }
797   \newcommand{\NyquistTF@Shorthandoff}[4][]{%
798     \Orig@NyquistTF[#1]{#2}{#3}{#4}%
799     \shorthandon{;:!?}%
800   }
801 \fi

```

\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.

```

802 \newcommand{\addNyquistZPKPlot}[2][]{
803   \gdef\func@mag{}
804   \gdef\func@ph{}
805   \build@ZPK@plot{\func@mag}{\func@ph}{#2}
806   \ifpgfarg
807     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
808       \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
809       {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
810   \else
811     \stepcounter{gnuplot@id}
812     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
813       \temp@cmd gnuplot [parametric, gnuplot@prefix] {
814         \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
815         \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
816       };
817   \fi
818 }

```

\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.

```

819 \newcommand{\addNyquistTFPlot}[2][]{
820   \gdef\func@mag{}
821   \gdef\func@ph{}
822   \build@TF@plot{\func@mag}{\func@ph}{#2}
823   \if@pgfarg
824     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
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/d
able=t, #1]}
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 }

```

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.

```

836 \if@babel@french
837   \AddToHook{env/NyquistPlot/begin}{\shorthandoff{;:!?}}
838 \fi
839 \NewDocumentEnvironment{NyquistPlot}{0}{mm+b}{
840   \parse@env@opt{#1}
841   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
842   \temp@cmd
843   \edef\temp@cmd{\noexpand\begin{axis}[
844     bode@style,
845     height=5cm,
846     domain=#2:#3,
847     xlabel={\$Re\$},
848     ylabel={\$Im\$},
849     \unexpanded\expandafter{\opt@axes}
850   ]}
851   \temp@cmd
852   \addplot [only marks,mark=+,thick,red] (-1 , 0);
853   #4
854   \end{axis}
855   \end{tikzpicture}
856 }{}

```

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.

```

857 \newcommand{\parse@N@opt}[1]{
858   \gdef\opt@axes{}
859   \gdef\opt@plot{}
860   \gdef\opt@commands{}
861   \gdef\opt@tikz{}
862   \foreach \obj/\opt in {#1} {
863     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0

```

```

864     \xdef\opt@axes{\unexpanded\expandafter{\opt}}
865   \else
866     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
867     \xdef\opt@plot{\unexpanded\expandafter{\opt}}
868   \else
869     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
870     \xdef\opt@commands{\unexpanded\expandafter{\opt}}
871   \else
872     \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
873     \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
874   \else
875     \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
876       \unexpanded\expandafter{\obj}}
877   \fi
878 \fi
879 \fi
880 \fi
881 }
882 }

```

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 883 \newcommand{\NicholsZPK}[4][]{
\addNicholsZPKChart 884 \parse@N@opt{#1}
\addNicholsTFChart 885 \gdef\func@mag{}
886 \gdef\func@ph{}
887 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
888 \temp@cmd
889 \build@ZPK@plot{\func@mag}{\func@ph}{#2}
890 \edef\temp@cmd{\noexpand\begin{axis}[
891   ph@x@label,
892   bode@style,
893   domain=#3*\freq@scale:#4*\freq@scale,
894   height=5cm,
895   ylabel={Gain (dB)},
896   samples=500,
897   \unexpanded\expandafter{\opt@axes}
898 ]}
899 \temp@cmd
900 \edef\temp@cmd{\noexpand\addplot [variable=t,thick,\opt@plot]}
901 \ifpgfarg
902   \temp@cmd ( {\func@ph} , {\func@mag} );
903   \opt@commands
904 \else
905   \stepcounter{gnuplot@id}
906   \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
907   { set table $meta;
908     set logscale x 10;
909     set dummy t;
910     set samples \pgfkeysvalueof{/pgfplots/samples};
911     set trange [#3*\freq@scale:#4*\freq@scale];
912     plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
913     unset logscale x;
914     set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
915     plot "$meta" using ($2*\ph@scale):($1);
916   };
917   \opt@commands
918 \fi
919 \end{axis}
920 \end{tikzpicture}

```

```

921 }
922 \if@babel@french
923 \let\Orig@NicholsZPK\NicholsZPK
924 \renewcommand{\NicholsZPK}{%
925 \shorthandoff{;:!?}%
926 \NicholsZPK@Shorthandoff
927 }
928 \newcommand{\NicholsZPK@Shorthandoff}[4][]{%
929 \Orig@NicholsZPK[#1]{#2}{#3}{#4}%
930 \shorthandon{;:!?}%
931 }
932 \fi
933 \newcommand{\NicholsTF}[4][]{
934 \parse@N@opt{#1}
935 \gdef\func@mag{}
936 \gdef\func@ph{}
937 \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
938 \temp@cmd
939 \build@TF@plot{\func@mag}{\func@ph}{#2}
940 \edef\temp@cmd{\noexpand\begin{axis}[
941 ph@x@label,
942 bode@style,
943 domain=#3*\freq@scale:#4*\freq@scale,
944 height=5cm,
945 ylabel={Gain (dB)},
946 samples=500,
947 \unexpanded\expandafter{\opt@axes}
948 ]}
949 \temp@cmd
950 \edef\temp@cmd{\noexpand\addplot [variable=t,thick, \opt@plot]}
951 \ifpgfarg
952 \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
953 \opt@commands
954 \else
955 \stepcounter{gnuplot@id}
956 \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
957 { set table $meta1;
958 set logscale x 10;
959 set dummy t;
960 set samples \pgfkeysvalueof{/pgfplots/samples};
961 set trange [#3*\freq@scale:#4*\freq@scale];
962 plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
963 unset logscale x;
964 set table $meta2;
965 plot "$meta1" using ($1):($2) smooth unwrap;
966 set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
967 plot "$meta2" using ($2*\ph@scale):($1);
968 };
969 \opt@commands
970 \fi
971 \end{axis}
972 \end{tikzpicture}
973 }
974 \if@babel@french
975 \let\Orig@NicholsTF\NicholsTF
976 \renewcommand{\NicholsTF}{%
977 \shorthandoff{;:!?}%
978 \NicholsTF@Shorthandoff
979 }
980 \newcommand{\NicholsTF@Shorthandoff}[4][]{%
981 \Orig@NicholsTF[#1]{#2}{#3}{#4}%
982 \shorthandon{;:!?}%

```

```

983 }
984 \AddToHook{env/NicholsChart/begin}{\shorthandoff{;:!!?}}
985 \fi
986 \NewDocumentEnvironment{NicholsChart}{0}{mm+b}{
987   \parse@env@opt{#1}
988   \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]]
989   \temp@cmd
990   \edef\temp@cmd{\noexpand\begin{axis}[
991     ph@x@label,
992     bode@style,
993     domain=#2:#3,
994     height=5cm,
995     ylabel={Gain (dB)},
996     \unexpanded\expandafter{\opt@axes}
997   ]}
998   \temp@cmd
999   #4
1000   \end{axis}
1001   \end{tikzpicture}
1002 }{}
1003 \newcommand{\addNicholsZPKChart}[2][] {
1004   \gdef\func@mag{}
1005   \gdef\func@ph{}
1006   \build@ZPK@plot{\func@mag}{\func@ph}{#2}
1007   \if@pgfarg
1008     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
1009     \temp@cmd ( {\func@ph} , {\func@mag} );
1010   \else
1011     \stepcounter{gnuplot@id}
1012     \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1013     { set table $meta;
1014       set logscale x 10;
1015       set dummy t;
1016       set samples \pgfkeysvalueof{/pgfplots/samples};
1017       set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1018       plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1019       unset logscale x;
1020       set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1021       plot "$meta" using ($2*\ph@scale):($1);
1022     };
1023   \fi
1024 }
1025 \newcommand{\addNicholsTFChart}[2][] {
1026   \gdef\func@mag{}
1027   \gdef\func@ph{}
1028   \build@TF@plot{\func@mag}{\func@ph}{#2}
1029   \if@pgfarg
1030     \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/d
able=t, #1]}
1031     \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
1032   \else
1033     \stepcounter{gnuplot@id}
1034     \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
1035     { set table $meta1;
1036       set logscale x 10;
1037       set dummy t;
1038       set samples \pgfkeysvalueof{/pgfplots/samples};
1039       set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
1040       plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
1041       unset logscale x;
1042       set table $meta2;

```

```

1043     plot "$meta1" using ($1):($2) smooth unwrap;
1044     set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
1045     plot "$meta2" using ($2*\ph@scale):($1);
1046 };
1047 \fi
1048 }

```

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	\build@TF@plot	820, 822, 825, 826,
\@babel@frenchfalse 47	283, 376, 378, <u>579</u> ,	831, 832, 885, 889,
\@babel@frenchtrue	763, 822, 939, 1028	902, 912, 935, 939,
..... 48, 49	\build@ZPK@plot . . .	952, 962, 1004,
\@declutterargfalse . 5	216, 349, 351, <u>512</u> ,	1006, 1009, 1018,
\@declutterargtrue . . 7	718, 805, 889, 1006	1026, 1028, 1031, 1040
\@hzargfalse 13	C	\func@ph . . . 213, 216,
\@hzargtrue 15	\currentdegree . 591,	235, 257, 280, 283,
\@ifpackagewith . 48, 49	592, 595, 596, 597,	302, 324, 715, 718,
\@nil 112, 113	599, 600, 610, 611,	732, 733, 738, 739,
\@pgfargfalse 1	614, 615, 616, 618, 619	760, 763, 777, 778,
\@pgfargtrue 3		783, 784, 804, 805,
\@radargfalse 9	D	808, 809, 814, 815,
\@radargtrue 11	\den@degree 607, 610	821, 822, 825, 826,
A	\den@im 583, 615, 628, 631	831, 832, 886, 889,
\add@feature	\den@real 582,	902, 912, 936, 939,
498, 517, 518, 521,	612, 618, 629, 631	952, 962, 1005,
522, 524, 525, 533,	\dencoeff 606,	1006, 1009, 1018,
534, 537, 538, 540,	609, 612, 615, 618	1027, 1028, 1031, 1040
541, 548, 549, 552,	\denpow 606, 607, 609, 610	
553, 555, 556, 572, 573	E	G
\addBodeComponentPlot	environments:	\get@interval@end .
..... <u>415</u>	BodeMagPlot <u>455</u> <u>112</u> , 113
\addBodeTFPlot <u>372</u>	BodePhPlot <u>433</u>	\get@interval@start
\addBodeZPKPlots . . <u>344</u>	BodePlot <u>477</u> <u>112</u> , 112
\addNicholsTFChart <u>883</u>		\gnuplot@id <u>1</u>
\addNicholsZPKChart <u>883</u>	F	\gnuplot@prefix <u>1</u>
\addNyquistTFPlot . <u>819</u>	\freq@filter <u>65</u>	
\addNyquistZPKPlot <u>802</u>	\freq@label <u>65</u>	I
\AddToHook 434,	\freq@scale <u>65</u> ,	\if@babel@french . .
456, 478, 837, 984	66, 77, 101, 108, 47, 266,
B	221, 243, 247, 255,	333, 433, 455, 477,
\bode@style <u>50</u>	259, 288, 310, 314,	746, 791, 836, 922, 974
BodeMagPlot (env.) . . <u>455</u>	322, 326, 354, 363,	\if@hzarg 13, 76
BodePhPlot (env.) . . . <u>433</u>	367, 382, 385, 395,	\ifcat 499
BodePlot (env.) <u>477</u>	399, 406, 410, 417,	\ifnum 348, 375,
\bodeplot@prefix . .	425, 429, 721, 766,	381, 390, 508, 514,
..... 31, 33, 38,	807, 812, 824, 829,	516, 520, 530, 532,
246, 258, 313, 325,	893, 911, 943, 961,	536, 546, 547, 551,
366, 398, 409, 428,	1008, 1017, 1030, 1039	560, 561, 565, 586,
914, 966, 1020, 1044	\func@mag . . 212, 216,	592, 605, 611, 624,
\BodeTF <u>277</u>	233, 245, 279, 283,	644, 645, 648, 656,
\BodeTF@Shorthandoff	300, 312, 714, 718,	657, 660, 668, 671,
..... 337, 339	732, 733, 738, 739,	674, 675, 681, 700,
\BodeZPK <u>210</u>	759, 763, 777, 778,	703, 863, 866, 869, 872
\BodeZPK@Shorthandoff	783, 784, 803, 805,	\ifwindows 41
..... 270, 272	808, 809, 814, 815,	\immediate 43

J		
\jobname	31, 33	
L		
\Last@LoopValue	506, 508	
\loop@delay	584, 625, 629	
M		
\MagCSPoles	155	
\MagCSPolesAsymp	155	
\MagCSPolesLin	155	
\MagCSPolesPeak	174	
\MagCSZeros	155	
\MagCSZerosAsymp	155	
\MagCSZerosLin	155	
\MagCSZerosPeak	174	
\MagDel	120, 573	
\MagK	114, 556	
\MagKAsymp	114, 553	
\MagKLin	114, 549	
\MagPole	122, 149, 541	
\MagPoleAsymp	122, 151, 538	
\MagPoleLin	122, 150, 534	
\MagSOPoles	182	
\MagSOPolesAsymp	182	
\MagSOPolesLin	182	
\MagSOPolesPeak	200	
\MagS0Zeros	182	
\MagS0ZerosAsymp	182	
\MagS0ZerosLin	182	
\MagS0ZerosPeak	200	
\MagZero	149, 525	
\MagZeroAsymp	149, 522	
\MagZeroLin	149, 518	
N		
\n	504, 506	
\n@mod	1, 302, 383, 952, 1031	
\n@mod@n	1	
\n@mod@p	1, 130	
\n@pow	1, 123, 124, 125, 135, 136, 138, 139, 141, 142, 143, 144, 145, 146, 155, 156, 159, 160, 161, 163, 165, 166, 183, 187, 596, 597, 599, 600, 615, 616, 618, 619, 630, 631, 732, 733, 738, 739, 777, 778, 783, 784, 808, 809, 814, 815, 825, 826, 831, 832	
\newcounter	28	
\NewDocumentEnvironment	436, 458, 480, 839, 986	
\NicholsChart	883	
\NicholsTF	883	
\NicholsTF@Shorthandoff	978, 980	
\NicholsZPK	883	
\NicholsZPK@Shorthandoff	390, 514, 516, 520, 530, 532, 536, 546, 547, 551, 560, 561, 565, 586, 605, 624, 644, 645, 648, 656, 657, 660, 668, 671, 674, 675, 681, 700, 703, 863, 866, 869, 872	
\num@degree	588, 591	
\num@im	581, 596, 628, 630	
\num@real	580, 593, 599, 628, 630	
\numcoeff	587, 590, 593, 596, 599	
\numpow	587, 588, 590, 591	
\NyquistPlot	836	
\NyquistTF	757	
\NyquistTF@Shorthandoff	795, 797	
\NyquistZPK	712	
\NyquistZPK@Shorthandoff	750, 752	
O		
\opt@approx	216, 639, 672	
\opt@axes	448, 470, 491, 697, 701, 706, 726, 771, 849, 858, 864, 897, 947, 996	
\opt@commands	734, 741, 779, 786, 860, 870, 903, 917, 953, 969	
\opt@group	225, 292, 638, 669	
\opt@plot	730, 775, 859, 867, 875, 900, 950	
\opt@tikz	214, 281, 438, 460, 482, 642, 682, 698, 704, 716, 761, 841, 861, 873, 887, 937, 988	
\optmag@axes	228, 295, 634, 658, 663	
\optmag@commands	234, 249, 301, 316, 641, 678	
\optmag@plot	229, 296, 637, 646, 651, 684	
\optph@axes	230, 297, 635, 661, 664	
\optph@commands	236, 261, 303, 328, 640, 676	
\optph@plot	231, 298, 636, 649, 652, 686	
\Orig@BodeTF	334, 340	
\Orig@BodeZPK	267, 273	
\Orig@NicholsTF	975, 981	
\Orig@NicholsZPK	923, 929	
\Orig@NyquistTF	792, 798	
\Orig@NyquistZPK	747, 753	
P		
\p	531, 533, 534, 537, 538, 540, 541	
\parse@env@opt	437, 459, 481, 696, 840, 987	
\parse@N@opt	713, 758, 857, 884, 934	
\parse@opt	211, 278, 633	
\pdf@strcmp	348, 375, 381,	
\pgfkeysvalueof	244, 256, 311, 323, 354, 363, 364, 382, 385, 395, 396, 406, 407, 417, 425, 426, 807, 812, 824, 829, 910, 960, 1008, 1016, 1017, 1030, 1038, 1039	
\pgfplotsset	23, 50, 65, 67, 68, 69, 72, 73, 78, 80, 89, 90, 94, 95, 102, 104, 109	
\ph@scale	65, 70, 74, 88, 93, 117, 121, 132, 145, 148, 160, 166, 167, 187, 302, 324, 326, 397, 399, 629, 732, 733, 738, 739, 777, 778, 783, 784, 808, 809, 814, 815, 825, 826, 831, 832, 912, 915, 952, 962, 967, 1018, 1021, 1031, 1040, 1045	
\ph@x@label	65	
\ph@y@label	65	
\PhCSPoles	155	
\PhCSPolesAsymp	155, 192	
\PhCSPolesLin	155, 189	
\PhCSZeros	155	
\PhCSZerosAsymp	155	
\PhCSZerosLin	155	
\PhDel	121, 572	
\PhK	114, 555	
\PhKAsymp	114, 120, 552	
\PhKLin	114, 120, 548	
\PhPole	122, 152, 540	
\PhPoleAsymp	122, 154, 537	
\PhPoleLin	122, 153, 533	
\PhSOPoles	182	
\PhSOPolesAsymp	182	
\PhSOPolesLin	182	
\PhS0Zeros	182	
\PhS0ZerosAsymp	182	
\PhS0ZerosLin	182	
\PhZero	149, 524	
\PhZeroAsymp	149, 521	
\PhZeroLin	149, 517	
\plot@macro	346, 349, 351, 355, 365, 373, 376, 378, 383, 386, 397, 408	
R		
\renewcommand	88,	

93, 101, 108, 268, 335, 748, 793, 924, 976	358, 359, 382, 383, 385, 386, 417, 418, 438, 439, 440, 450, 460, 461, 462, 472, 482, 483, 484, 493, 716, 717, 719, 728, 730, 732, 737, 761, 762, 764, 773, 775, 777, 782, 807, 808, 812, 813, 824, 825, 829, 830, 841, 842, 843, 851, 887, 888, 890, 899, 900, 902, 906, 937, 938, 940, 949, 950, 952, 956, 988, 989, 990, 998, 1008, 1009, 1030, 1031	233, 239, 295, 300, 306 <code>\temp@ph@cmd</code> ... 230, 235, 251, 297, 302, 318
S <code>\setcounter</code> 29 <code>\shorthandoff</code> 269, 336, 434, 456, 478, 749, 794, 837, 925, 977, 984 <code>\shorthandon</code> ... 274, 341, 754, 799, 930, 982 <code>\stepcounter</code> 238, 250, 305, 317, 357, 389, 420, 736, 781, 811, 828, 905, 955, 1011, 1033	V <code>\values</code> 513, 515, 531, 548, 549, 552, 553, 555, 556, 569, 572, 573, 585, 587, 590, 606, 609, 625	W <code>\write</code> 43
T <code>\temp@cmd</code> .. 214, 215, 217, 227, 281, 282, 284, 294, 354, 355,	<code>\temp@macro</code> 347, 349, 351, 374, 376, 378 <code>\temp@mag@cmd</code> .. 228,	Y <code>\y</code> 504, 505
	Z <code>\z</code> 515, 517, 518, 521, 522, 524, 525	

Change History

v1.0 General: Initial release 1	v1.0.5 <code>\parse@opt</code> : Fixed a bug 31
v1.0.1 <code>\addBodeZPKPlots</code> : Improved optional argument handling. 25 <code>\BodeZPK</code> : Pass arbitrary TikZ commands as options. 23	v1.0.6 General: Fixed issue #3 1
v1.0.2 <code>gnuplot@prefix</code> : Fixed issue #1 .. 18	v1.0.7 General: Removed unnecessary semicolons 1 Updated documentation 1
v1.0.3 <code>BodePlot</code> : Added tikz option to environments 28 <code>\BodeTF</code> : Added Tikz option 24 <code>\BodeZPK</code> : Added Tikz option 23 <code>NicholsChart</code> : Added tikz option to environments 36 <code>\NicholsTF</code> : Added commands and tikz options 36 <code>\NicholsZPK</code> : Added commands and tikz options 36 <code>gnuplot@prefix</code> : Added jobname to gnuplot prefix 18 <code>\NyquistTF</code> : Added commands and tikz options 33 <code>\NyquistZPK</code> : Added commands and tikz options 32 <code>\parse@env@opt</code> : Added tikz option to environments 32 <code>\parse@N@opt</code> : Added commands and tikz options 35 <code>\parse@opt</code> : Added Tikz option ... 31 <code>NyquistPlot</code> : Added tikz option to environments 35	v1.0.8 General: Added a new class option ‘declutter’ 1 <code>\build@TF@plot</code> : Included phase due to delay in wrapping. 30 <code>gnuplot@prefix</code> : Fixed issue #6 .. 18
v1.0.4 General: Fixed unintended optional argument macro expansion 1	v1.1.0 General: Fixed phase wrapping in gnuplot mode 1 <code>\addBodeTFPlot</code> : Fixed phase wrapping in gnuplot mode 26 <code>BodeMagPlot</code> : Added separate environments for phase and magnitude plots 27 <code>BodePhPlot</code> : Added separate environments for phase and magnitude plots 27 <code>BodePlot</code> : Deprecated <code>BodePlot</code> environment 28 <code>\BodeTF</code> : Fixed phase wrapping in gnuplot mode 24
	v1.1.1 General: Enable Hz and rad units 1 <code>\addBodeComponentPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively 27 <code>\addBodeTFPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively 26

<code>\addBodeZPKPlots</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	25	v1.1.3	‘environ’ package to fix conflicts with externalization	35
<code>\addNicholsTFChart</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	36		<code>\addBodeComponentPlot</code> : Changed implementation to respect user-supplied domain	27
<code>\addNyquistTFPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	35		<code>\addBodeTFPlot</code> : Changed implementation to respect user-supplied domain	26
<code>\addNyquistZPKPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	34		<code>\addBodeZPKPlots</code> : Changed implementation to respect user-supplied domain	25
<code>BodeMagPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27		<code>\addNicholsTFChart</code> : Changed implementation to respect user-supplied domain	36
<code>BodePhPlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	27		<code>\addNicholsZPKChart</code> : Changed implementation to respect user-supplied domain	36
<code>BodePlot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	28		<code>\addNyquistTFPlot</code> : Changed implementation to respect user-supplied domain	35
<code>\BodeTF</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	24		<code>\addNyquistZPKPlot</code> : Changed implementation to respect user-supplied domain	34
<code>\BodeZPK</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	23	v1.1.4		
<code>\build@TF@plot</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	30		<code>\addBodeTFPlot</code> : Changed phase wrapping in pgf mode	26
<code>get@interval@end</code> : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	20		<code>\addNicholsTFChart</code> : Changed phase wrapping in pgf mode	36
<code>ph@y@label</code> : New macros to enable ‘Hz’ and ‘rad’ units for frequency and phase, respectively	19		<code>\BodeTF</code> : Changed phase wrapping in pgf mode	24
<code>\NyquistTF</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	33		<code>gnuplot@prefix</code> : Changed phase wrapping in pgf mode	18
<code>\NyquistZPK</code> : Enabled ‘Hz’ and ‘rad’ units for frequency and phase, respectively	32	v1.1.5		
v1.1.2			General: Detect ‘babel-french’ to handle active characters	1
<code>BodeMagPlot</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	27		<code>BodeMagPlot</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	27
<code>BodePhPlot</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	27		<code>BodePhPlot</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	27
<code>BodePlot</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	28		<code>BodePlot</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	28
<code>NicholsChart</code> : Defined using the ‘NewEnviron’ command from the ‘environ’ package to fix conflicts with externalization	36		<code>\BodeTF</code> : Added code to handle active characters	25
<code>\PhS0ZerosLin</code> : Fix scaling bug introduced in v1.1.1	22		<code>\BodeZPK</code> : Added code to handle active characters	24
<code>NyquistPlot</code> : Defined using the ‘NewEnviron’ command from the			<code>NicholsChart</code> : Defined using the ‘NewDocumentEnvironment’ command from the ‘xparse’ package and added a hook to handle active characters	36
			<code>\NicholsTF</code> : Added code to handle active characters	36

<code>\NicholsZPK</code> : Added code to handle active characters	36	active characters	33
<code>gnuplot@prefix</code> : Added babel option to handle active characters	18	<code>NyquistPlot</code> : Defined using the 'NewDocumentEnvironment' command from the 'xparse' package and added a hook to handle active characters	35
<code>\NyquistTF</code> : Added code to handle active characters	34		
<code>\NyquistZPK</code> : Added code to handle			