

The bodeplot package*

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

July 6, 2022

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	12
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.	19
4.3	Second order systems.	20
4.4	Commands for Bode plots	21
4.4.1	User macros	21
4.4.2	Internal macros	25
4.5	Nyquist plots	29
4.5.1	User macros	29
4.5.2	Internal commands	31
4.6	Nichols charts	31

*This document corresponds to bodeplot v1.0.8, dated July 06, 2022.

1 Introduction

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

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

and the zero-pole-gain (ZPK) form

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

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

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.

1.2 Directory Structure

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

1.3 Limitations

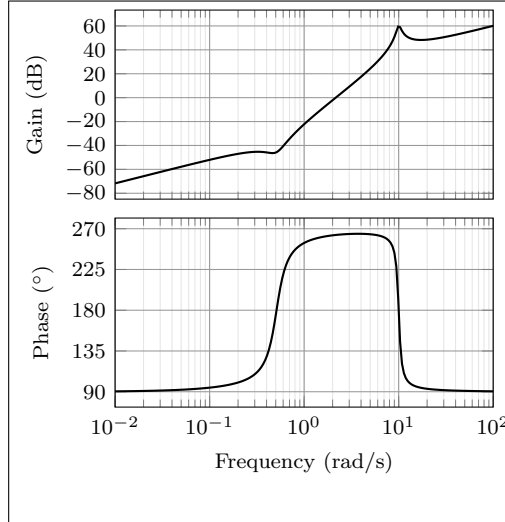
- In TF form, the phase angles are always between 0 and 360°. As such, the Bode phase plots and the Nyquist and Nichols plots will have phase wrapping discontinuities. I do not know how this can be rectified, pull requests are welcome!
- Use of the **declutter** option with other directory management tools such as a **tikzexternalize** prefix is not recommended.

2 TL;DR

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

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

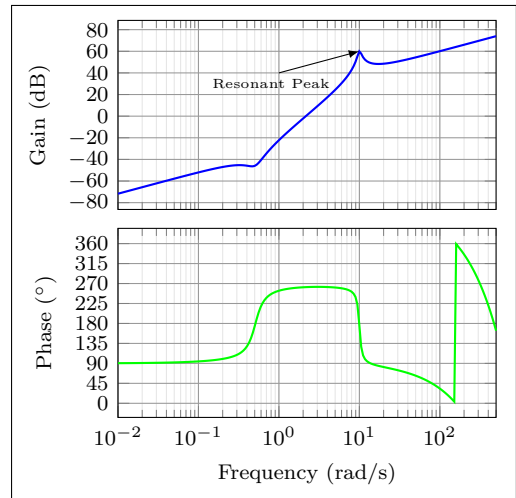
Bode plot in ZPK format



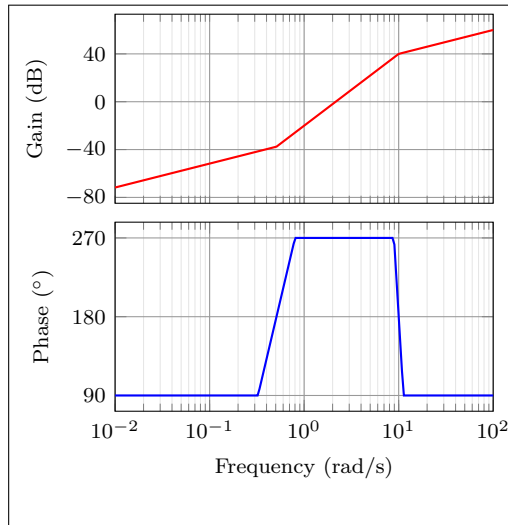
```
\BodeZPK{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10
}
{0.01}
{100}
```

Bode plot in TF format with arrow decoration, transport delay, and color customization (note the phase wrapping)

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



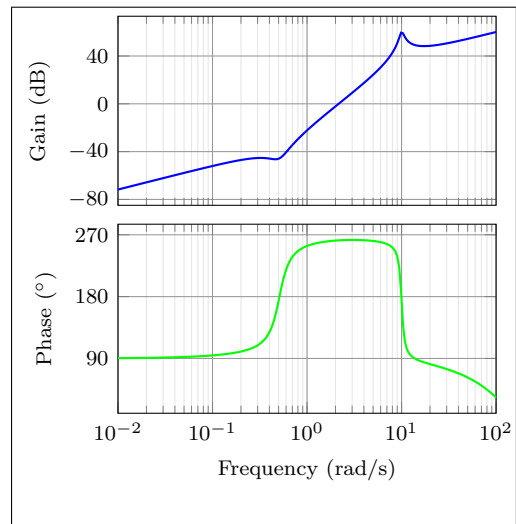
Linear approximation with customization



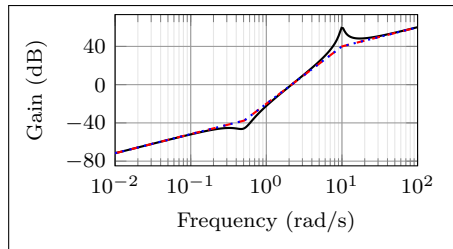
```
\BodeZPK[%
plot/mag/{red,thick},
plot/ph/{blue,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90},
approx/linear%
]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10
}
{0.01}
{100}
```

Plot with delay and customization

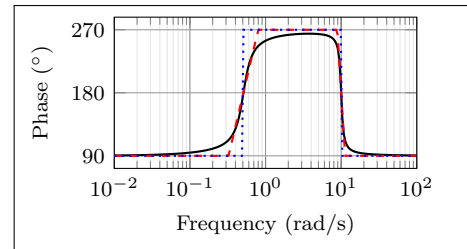
```
\BodeZPK[
plot/mag/{blue,thick},
plot/ph/{green,thick},
axes/mag/{ytick distance=40},
axes/ph/{ytick distance=90}
]{%
z/{0,{-0.1,-0.5},{-0.1,0.5}},
p/{{-0.5,-10},{-0.5,10}},
k/10,
d/0.01
}
{0.01}
{100}
```



Individual gain and phase plots with more customization

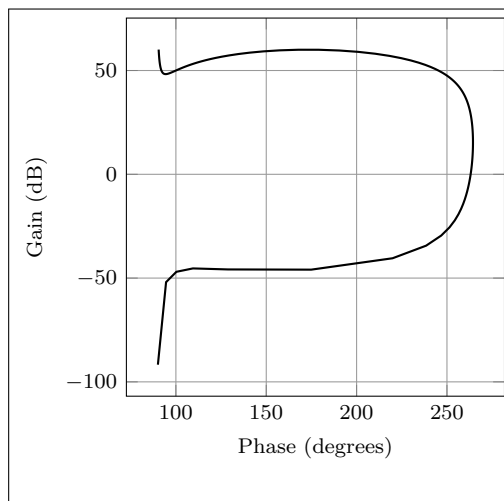


```
\begin{BodePlot}[%
  axes/{ylabel={Gain (dB)},
  ytick distance=40,
  height=2cm,
  width=4cm}
]
{0.01}
{100}
\addBodeZPKPlots[%
  true/{black,thick},
  linear/{red,dashed,thick},
  asymptotic/{blue,dotted,thick}%
]
{magnitude}
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10
}
}\end{BodePlot}
```



```
\begin{BodePlot}[%
  ylabel={Phase ($^\circ$)},
  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{BodePlot}
```

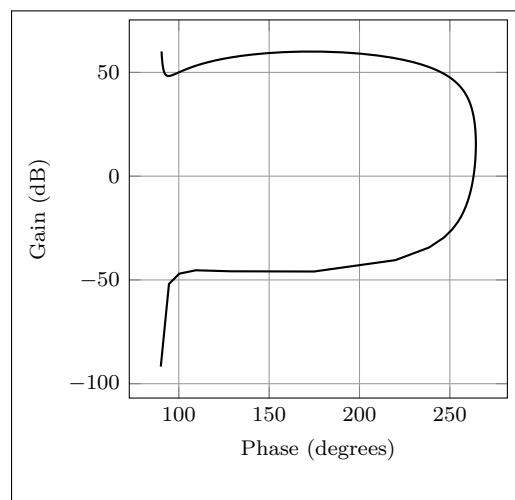
Nichols chart



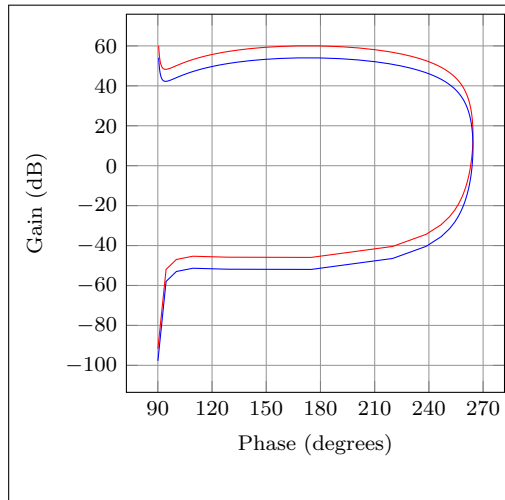
```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10
}
{0.001}
{100}
```

Nichols chart in TF format

```
\NicholsTF[samples=1000]
{num/{10,2,2.6,0},den/{1,1,100.25}}
{0.001}
{100}
```



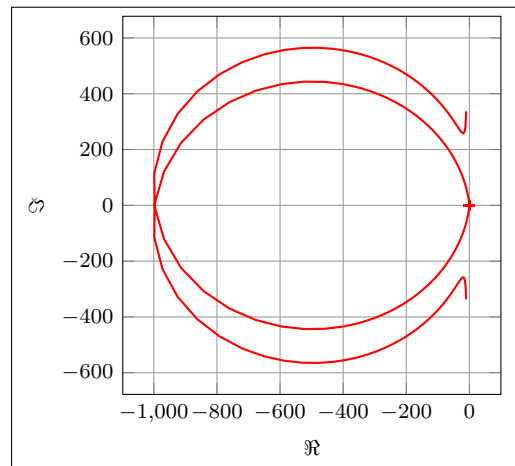
Multiple Nichols charts with customization



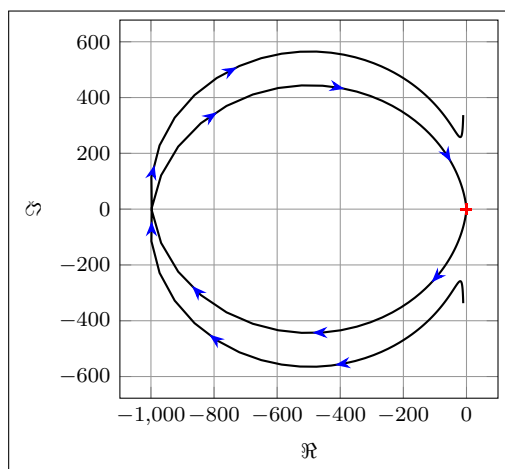
```
\begin{NicholsChart}[%
  ytick distance=20,
  xtick distance=30
]
{0.001}
{100}
\addNicholsZPKChart [red,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10
};
\addNicholsZPKChart [blue,samples=1000] {%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
 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



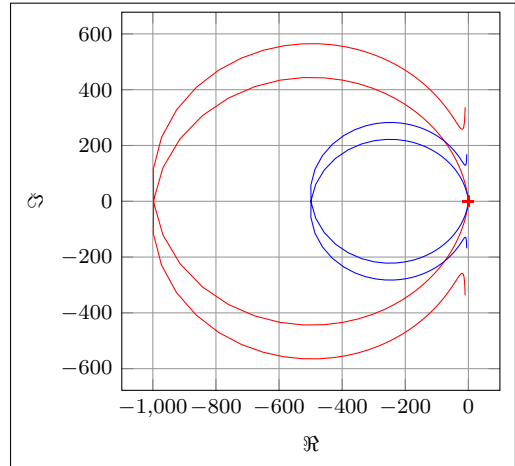
Nyquist plot in TF format with arrows



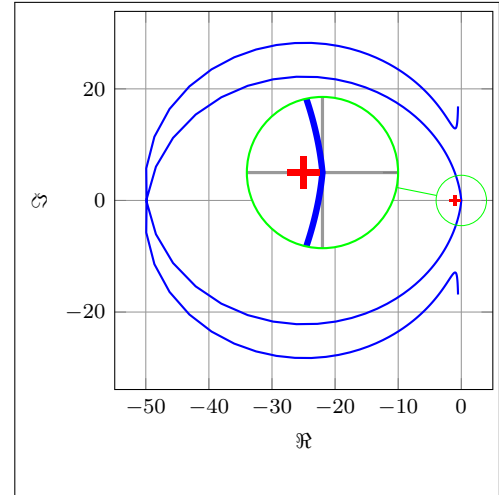
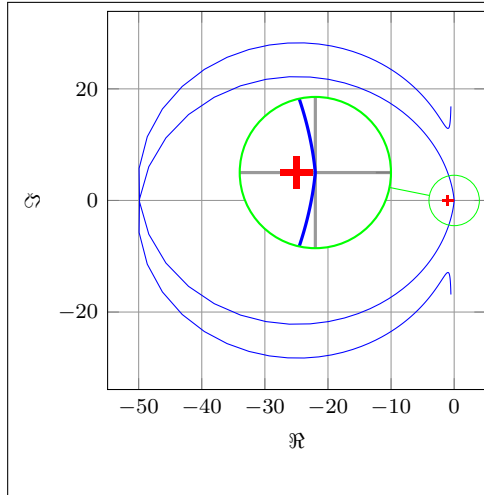
```
\NyquistTF[%
  plot/{%
    samples=1000,
    postaction=decorate,
    decoration={
      markings,
      mark=between positions 0.1 and 0.9 step 5cm with {%
        \arrow{Stealth [length=2mm, blue]}
      }
    }
  }%
]
{num/{10,2,2.6,0},den/{1,1,100.25}}
{-30}
{30}
```

Multiple Nyquist plots with customization

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



Nyquist plots with additional commands, using two different macros



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

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

3 Usage

3.1 Bode plots

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

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

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

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

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

For example, given a transfer function

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

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

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

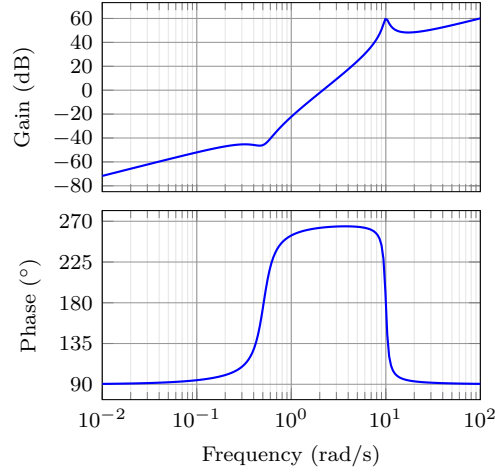


Figure 1: Output of the default `\BodeZPK` macro.

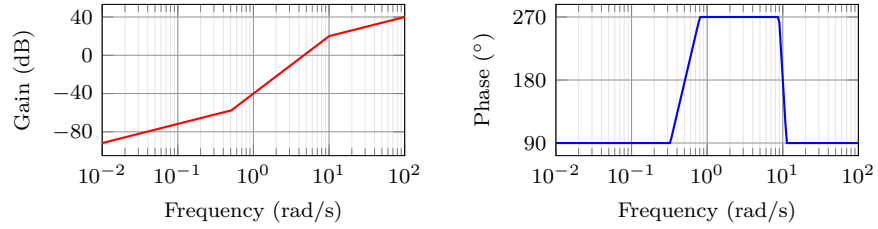


Figure 2: Customization of the default `\BodeZPK` macro.

`{0.01}{100}`

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

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

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

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

which generates the plot in Figure 2.

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

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

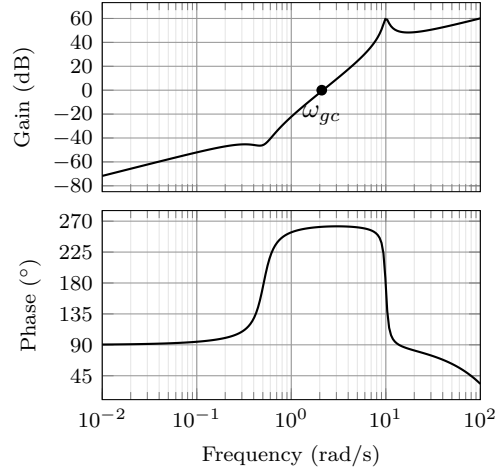


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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.

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

The `BodePlot` environment works in conjunction with the parametric function generator macros `\addBodeZPKPlots`, `\addBodeTFPlot`, and `\addBodeComponentPlot`. 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`.

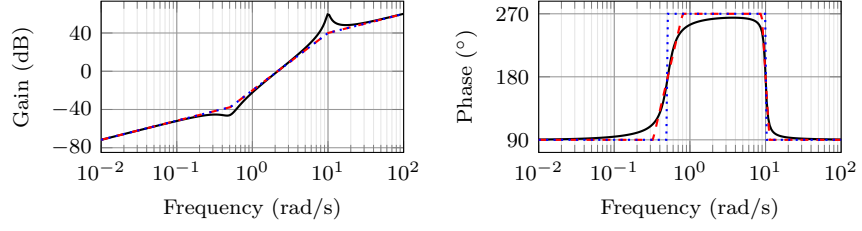


Figure 4: Superimposed approximate and true Bode plots using the **BodePlot** environment and the **\addBodeZPKPlots** macro.

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

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

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

```
\begin{BodePlot}[ ylabel={Gain (dB)}, ytick distance=40,
  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{BodePlot}
```

```
\begin{BodePlot}[ylabel={Phase ( $^{\circ}$ )},
  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{BodePlot}
```

which generates the plot in Figure 4.

\addBodeTFPlot **\addBodeTFPlot** [*\langle plot-options \rangle*]
 {\langle plot-type \rangle}
 {\langle num/{\langle coeffs \rangle}, den/{\langle coeffs \rangle}, 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 **\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{BodePlot}[ylabel={Phase (degree)},ytick distance=90]{0.01}{100}
  \addBodeComponentPlot[black,thick]{\PhZero{0}{0} + \PhZero{-0.1}{-0.5} +
    \PhZero{-0.1}{0.5} + \PhPole{-0.5}{-10} + \PhPole{-0.5}{10} +
    \PhK{10}{0}}
  \addBodeComponentPlot[red,dashed,thick]{\PhZeroLin{0}{0} +
    \PhZeroLin{-0.1}{-0.5} + \PhZeroLin{-0.1}{0.5} +
    \PhPoleLin{-0.5}{-10} + \PhPoleLin{-0.5}{10} + \PhKLin{10}{20}}
  \addBodeComponentPlot[blue,dotted,thick]{\PhZeroAsymp{0}{0} +
    \PhZeroAsymp{-0.1}{-0.5} + \PhZeroAsymp{-0.1}{0.5} +
    \PhPoleAsymp{-0.5}{-10} + \PhPoleAsymp{-0.5}{10} + \PhKAsymp{10}{40}}
\end{BodePlot}
```

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

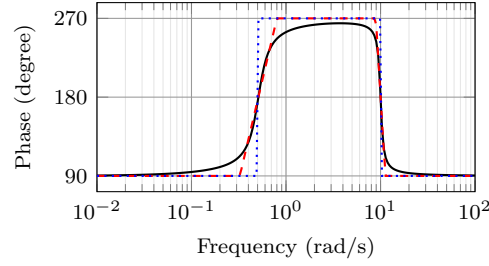


Figure 5: Superimposed approximate and true Bode Phase plot using the **BodePlot** environment, the `\addBodeComponentPlot` macro, and several macros of the `\Type-FeatureApprox` form.

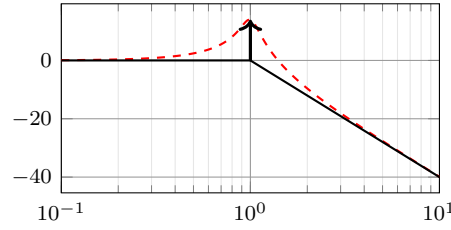


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

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[$\langle draw-options \rangle$]{ $\langle a1 \rangle$ }{ $\langle a0 \rangle$ }`

This entry describes 2 different macros of the form `\MagS0FeaturePeak` that take 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{BodePlot}[xlabel={}] {0.1}{10}
  \addBodeComponentPlot[red,dashed,thick]{\MagSOPoles{0.2}{1}}
  \addBodeComponentPlot[black,thick]{\MagSOPolesLin{0.2}{1}}
  \MagSOPolesPeak[thick]{0.2}{1}
\end{BodePlot}
```

generates the plot in Figure 6.

`\TypeCSFeatureApprox` `\TypeCSFeatureApprox{ $\langle zeta \rangle$ }{ $\langle \omega_n \rangle$ }`

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

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

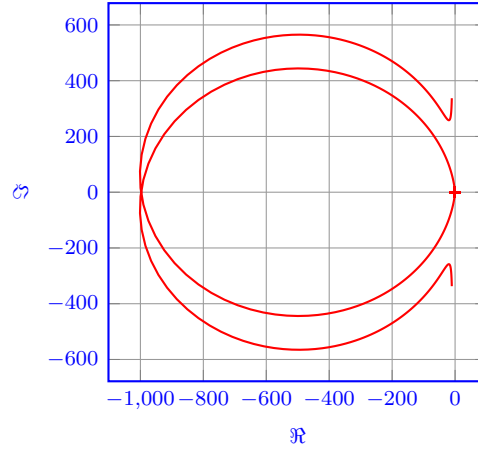


Figure 7: Output of the `\NyquistZPK` macro.

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

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/\langle opt \rangle`, which passes `\langle opt \rangle` to `\addplot`, `axes/\langle opt \rangle`, which passes `\langle opt \rangle` to the `axis` environment, and `tikz/\langle opt \rangle`, which passes `\langle opt \rangle` to the `tikzpicture` environment. Asymptotic/linear approximations are not supported in Nyquist plots. If just `\langle opt \rangle` is provided as the optional argument, it is interpreted as `plot/\langle opt \rangle`. Arrows to indicate the direction of increasing ω can be added by adding `\usetiklibrary{decorations.markings}` and `\usetiklibrary{arrows.meta}` to the preamble and then passing a tuple of the form

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

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,`

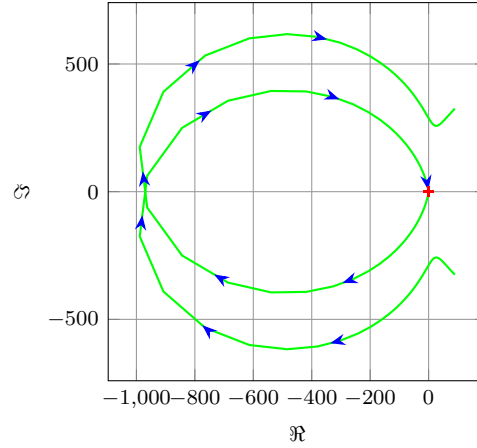


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

```
mark=between positions 0.1 and 0.9 step 5em
with{\arrow{Stealth[length=2mm, blue]}}
{num/{10,2,2.6,0},den/{1,1,100.25}}
{-30}{30}
```

generates the Nyquist plot in Figure 8.

```
NyquistPlot (env.) \begin{NyquistPlot}[\langle obj1/\langle opt1 \rangle \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[\langle plot-options \rangle]
{\langle z/\langle zeros \rangle \rangle, p/\langle poles \rangle, k/\langle gain \rangle, d/\langle delay \rangle}
```

Generates a two parametric functions for the magnitude and the phase a transfer function in ZPK form. The generated magnitude and phase parametric functions are 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[\langle plot-options \rangle]
```

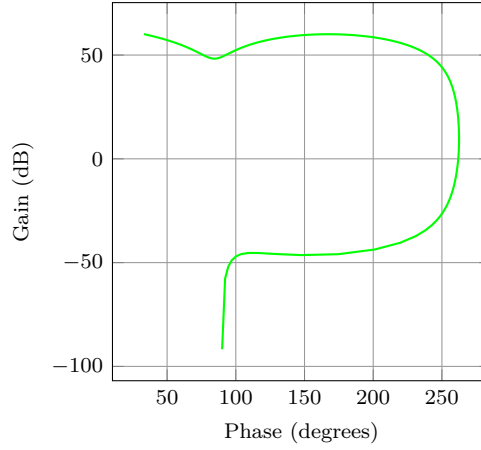


Figure 9: Output of the `\NyquistZPK` macro.

$\{\langle num/\{\langle coeffs \rangle\}, den/\{\langle coeffs \rangle\}, d/\{\langle delay \rangle\}\}$
 Similar to `\addNyquistZPKPlot`, with a transfer function input in the TF form.

3.3 Nichols charts

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

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

`\NicholsTF` `\NicholsTF` [$\langle plot/\{\langle opt \rangle\}, axes/\{\langle opt \rangle\} \rangle$]
 $\{\langle num/\{\langle coeffs \rangle\}, den/\{\langle coeffs \rangle\}, d/\{\langle delay \rangle\} \}$
 $\{\langle min-freq \rangle\} \{\langle max-freq \rangle\}$
`\NicholsTF`[`plot`/`{green,thick,samples=2000}`]
 $\{\langle num/\{10,2,2.6,0\}, den/\{1,1,100.25\}, d/0.01 \}$
 $\{0.001\} \{100\}$

generates the Nichols chart in Figure 9.

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

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

- Tuples of the form `obj/ $\{\langle opt \rangle\}$` :
 - `tikz/ $\{\langle opt \rangle\}$` : modify picture properties by adding options $\{\langle opt \rangle\}$ to the `tikzpicture` environment.
 - `axes/ $\{\langle opt \rangle\}$` : modify axis properties by adding options $\{\langle opt \rangle\}$ to the `axis` environment.
 - `commands/ $\{\langle opt \rangle\}$` : 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 $\{\langle opt \rangle\}$ are passed directly to the `axis` environment.

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

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

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

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

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

4 Implementation

4.1 Initialization

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

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

`\n@mod` This code is needed to support both `pgfplots` and `gnuplot` simultaneously. New

`\n@pow` macros are defined for the `pow` and `mod` functions to address differences between the

`gnuplot@id` two math engines. We start by processing the `pgf` and `declutter` class options.

```
gnuplot@prefix 5 \newif\if@pgfarg\@pgfargfalse
gnuplot@degrees 6 \DeclareOption{pgf}{%
7   \@pgfargtrue
8 }
9 \newif\if@declutterarg\@declutterargfalse
10 \DeclareOption{declutter}{%
11   \@declutterargtrue
12 }
13 \ProcessOptions\relax
```

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

```
14 \if@pgfarg
15   \newcommand{\n@pow}[2]{(#1)^(#2)}%
16   \newcommand{\n@mod}[2]{mod((#1),(#2))}%
17 \else
18   \newcommand{\n@pow}[2]{(#1)**(#2)}%
19   \newcommand{\n@mod}[2]{(#1)-(floor((#1)/(#2))*(#2))}%
20
```

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.

```
20 \newcounter{gnuplot@id}%
21 \setcounter{gnuplot@id}{0}%
22 \if@declutterarg
23   \tikzset{%
24     gnuplot@prefix/.style={%
25       id=\arabic{gnuplot@id},
26       prefix=gnuplot/\jobname
27     }%
28   }
29 \else
30   \tikzset{%
31     gnuplot@prefix/.style={%
32       id=\arabic{gnuplot@id},
33       prefix=\jobname
34     }%
35   }
36 \fi
```

Then, we add `set angles degrees` to all `gnuplot` macros to avoid having to convert from degrees to radians everywhere.

```
37 \pgfplotsset{%
38   gnuplot@degrees/.code={%
39     \ifnum\value{gnuplot@id}=1
40       \xdef\pgfplots@gnuplot@format{\pgfplots@gnuplot@format set angles de-
41         grees;}%
42     }%
43 }
```

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.

```

44 \ifwindows\else
45 \if@declutterarg
46 \immediate\write18{mkdir -p gnuplot}%
47 \fi
48 \fi
49 \fi

```

`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 }%
63 }

```

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

`\MagK` True, linear, and asymptotic magnitude and phase parametric functions for a pure gain $G(s) = k + 0i$. The 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.

```

\PhKAsymp 64 \newcommand*\MagK[2]{(20*log10(abs(#1)))}
\PhKLin 65 \newcommand*\MagKAsymp{}\MagK
66 \newcommand*\MagKLin{}\MagK
67 \newcommand*\PhK[2]{(#1<0?-180:0)}
68 \newcommand*\PhKAsymp{}\PhK
69 \newcommand*\PhKLin{}\PhK

```

`\PhKAsymp` True magnitude and phase parametric functions for a pure delay $G(s) = e^{-Ts}$. The 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.

```

70 \newcommand*\MagDel[2]{0}
71 \newcommand*\PhDel[2]{-#1*180*t/pi}

```

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

```

\MagPoleLin 72 \newcommand*\MagPole[2]
\PhPole 73 {((-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 74 \newcommand*\MagPoleLin[2]{(t < sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) ?
75 -20*log10(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) :
76 -20*log10(t)
77 )}

```

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

```

78 \newcommand*\MagPoleAsymp{}\MagPoleLin

```

Parametric function for the true phase of a complex pole.

```
79 \newcommand*\PhPole}[2]{(#1 > 0 ? (#2 > 0 ?
80   (\n@mod{-atan2((t - (#2)), -(#1))+360}{360}) :
81   (-atan2((t - (#2)), -(#1)))) :
82   (-atan2((t - (#2)), -(#1))))}
```

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

```
83 \newcommand*\PhPoleLin}[2]{%
84   (abs(#1)+abs(#2) == 0 ? -90 :
85   (t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
86     (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))) ?
87     (-atan2(-(#2), -(#1))) :
88     (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
89       (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))) ?
90       (#2>0?(#1>0?270:-90):-90) :
91       (-atan2(-(#2), -(#1)) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
92         (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2})))) *
93         (\n@pow{#2}{2})))))) * ((#2>0?(#1>0?270:-90):-90) + atan2(-(#2), -(#1)))/
94       (log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
95         (\n@pow{#1}{2} + \n@pow{#2}{2}))))))))))}
```

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

```
96 \newcommand*\PhPoleAsymp}[2]{(t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2})) ?
97   (-atan2(-(#2), -(#1))) :
98   (#2>0?(#1>0?270:-90):-90))}
```

\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 99 \newcommand*\MagZero{0-\MagPole}
\PhZero 100 \newcommand*\MagZeroLin{0-\MagPoleLin}
\PhZeroAsymp 101 \newcommand*\MagZeroAsymp{0-\MagPoleAsymp}
\PhZeroLin 102 \newcommand*\PhZero{0-\PhPole}
103 \newcommand*\PhZeroLin{0-\PhPoleLin}
104 \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 105 \newcommand*\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
\PhCSPoles 106   - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2}))))}
\PhCSPolesAsymp 107 \newcommand*\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : - 40*log10(t))}
\PhCSPolesLin 108 \newcommand*\MagCSPolesAsymp{\MagCSPolesLin}
```

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

```
\MagCSZerosAsymp
\MagCSZerosLin 109 \newcommand*\PhCSPoles}[2]{(-atan2((2*(#1)*(#2)*t), (\n@pow{#2}{2}
\PhCSZeros 110   - \n@pow{t}{2}))))}
\PhCSZerosAsymp 111 \newcommand*\PhCSPolesLin}[2]{(t < (#2 / (\n@pow{10}{abs(#1)})) ?
\PhCSZerosLin 112   0 :
113   (t >= (#2 * (\n@pow{10}{abs(#1)})) ?
114     (#1>0 ? -180 : 180) :
115     (#1>0 ? (-180*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
116       (180*(log10(t*(\n@pow{10}{abs(#1})/#2))/(2*abs(#1))))))}
117 \newcommand*\PhCSPolesAsymp}[2]{(#1>0?(t<#2?0:-180):(t<#2?0:180))}
```

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).

```
118 \newcommand*\MagCSZeros{0-\MagCSPoles}
119 \newcommand*\MagCSZerosLin{0-\MagCSPolesLin}
```

```

120 \newcommand*\MagCSZerosAsymp}{0-\MagCSPolesAsymp}
121 \newcommand*\PhCSZeros}{0-\PhCSPoles}
122 \newcommand*\PhCSZerosLin}{0-\PhCSPolesLin}
123 \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.

```

124 \newcommand*\MagCSPolesPeak}[3][[]]{%
125   \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
126     (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
127 }
128 \newcommand*\MagCSZerosPeak}[3][[]]{%
129   \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
130     (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
131 }

```

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

```

\MagSOPolesAsymp
\MagSOPolesLin 132 \newcommand*\MagSOPoles}[2]{%
\PhSOPoles 133   (-20*log10(sqrt(\n@pow{#2} - \n@pow{t}{2}){2} + \n@pow{#1*t}{2})))}
\PhSOPolesAsymp 134 \newcommand*\MagSOPolesLin}[2]{%
\PhSOPolesLin 135   (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t))}
\MagS0Zeros 136 \newcommand*\MagSOPolesAsymp}{\MagSOPolesLin}

```

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

```

\MagS0ZerosLin
\PhS0Zeros 137 \newcommand*\PhSOPoles}[2]{(-atan2((#1)*t,((#2) - \n@pow{t}{2})))}
\PhS0ZerosAsymp 138 \newcommand*\PhSOPolesLin}[2]{(#2>0 ?
\PhS0ZerosLin 139   \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
140   (#1>0 ? -180 : 180))}
141 \newcommand*\PhSOPolesAsymp}[2]{(#2>0 ?
142   \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
143   (#1>0 ? -180 : 180))}

```

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).

```

144 \newcommand*\MagS0Zeros}{0-\MagSOPoles}
145 \newcommand*\MagS0ZerosLin}{0-\MagSOPolesLin}
146 \newcommand*\MagS0ZerosAsymp}{0-\MagSOPolesAsymp}
147 \newcommand*\PhS0Zeros}{0-\PhSOPoles}
148 \newcommand*\PhS0ZerosLin}{0-\PhSOPolesLin}
149 \newcommand*\PhS0ZerosAsymp}{0-\PhSOPolesAsymp}

```

\MagSOPolesPeak 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.

```

150 \newcommand*\MagSOPolesPeak}[3][[]]{%
151   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
152     (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
153       20*log10(abs(#2/sqrt(abs(#3))))});
154 }
155 \newcommand*\MagS0ZerosPeak}[3][[]]{%
156   \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
157     (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
158       20*log10(abs(#2/sqrt(abs(#3))))});
159 }

```

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`.

```
160 \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.

```
161 \parse@opt{#1}%
162 \gdef\func@mag{}%
163 \gdef\func@ph{}%
164 \build@ZPK@plot{\func@mag}{\func@ph}{\opt@approx}{#2}%
```

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

```
165 \edef\temp@cmd{\noexpand\begin{tikzpicture}}{\unexpanded\expandafter{\opt@tikz}}%
166 \noexpand\begin{groupplot}%
167     bode@style,
168     xmin={#3},
169     xmax={#4},
170     domain=#3:#4,
171     height=2.5cm,
172     xmode=log,
173     group style = {group size = 1 by 2,vertical sep=0.25cm},
174     \opt@group
175 ]%
176 }%
177 \temp@cmd
```

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

```
178 \if@pgfarg
179 \expandafter\nextgroupplot\expandafter[ytick distance=20,
180 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
181 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optmag@plot]}%
182 \temp@cmd {\func@mag};
183 \optmag@commands
184 \expandafter\nextgroupplot\expandafter[ytick distance=45,
185 ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
186 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optph@plot]}%
187 \temp@cmd {\func@ph};
188 \optph@commands
189 \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.

```
190 \stepcounter{gnuplot@id}
191 \expandafter\nextgroupplot\expandafter[ytick distance=20,
192 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
193 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optmag@plot]}%
194 \temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\func@mag};
195 \optmag@commands
196 \stepcounter{gnuplot@id}
197 \expandafter\nextgroupplot\expandafter[ytick distance=45,
198 ylabel={Phase ( $\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
199 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optph@plot]}%
200 \temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\func@ph};
201 \optph@commands
202 \fi
```

```

203 \end{groupplot}
204 \end{tikzpicture}
205 }

```

\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.

```

206 \newcommand{\BodeTF}[4][]{%
207 \parse@opt{#1}%
208 \gdef\func@mag{}%
209 \gdef\func@ph{}%
210 \build@TF@plot{\func@mag}{\func@ph}{#2}%
211 \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}}%
212 \noexpand\begin{groupplot}%
213 bode@style,
214 xmin={#3},
215 xmax={#4},
216 domain=#3:#4,
217 height=2.5cm,
218 xmode=log,
219 group style = {group size = 1 by 2,vertical sep=0.25cm},
220 \opt@group
221 ]%
222 }%
223 \temp@cmd
224 \if@pgfarg
225 \expandafter\nextgroupplot\expandafter[ytick distance=20,
226 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
227 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optmag@plot]}%
228 \temp@cmd {\func@mag};
229 \optmag@commands
230 \expandafter\nextgroupplot\expandafter[ytick distance=45,
231 ylabel={Phase ( $^\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
232 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optph@plot]}%
233 \temp@cmd {\func@ph};
234 \optph@commands
235 \else
236 \stepcounter{gnuplot@id}%
237 \expandafter\nextgroupplot\expandafter[ytick distance=20,
238 ylabel={Gain (dB)},xmajorticks=false,\optmag@axes]
239 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optmag@plot]}%
240 \temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\func@mag};
241 \optmag@commands
242 \stepcounter{gnuplot@id}%
243 \expandafter\nextgroupplot\expandafter[ytick distance=45,
244 ylabel={Phase ( $^\circ$ )},xlabel={Frequency (rad/s)},\optph@axes]
245 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\optph@plot]}%
246 \temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\func@ph};
247 \optph@commands
248 \fi
249 \end{groupplot}
250 \end{tikzpicture}
251 }

```

\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.

```

252 \newcommand{\addBodeZPKPlots}[3][true/{}]{%
253 \foreach \approx/\opt in {#1} {%
254 \gdef\plot@macro{}%
255 \gdef\temp@macro{}%
256 \ifnum\pdfstrcmp{#2}{phase}=0
257 \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}%
258 \else

```

```

259     \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}%
260 \fi
261 \if@pgfarg
262     \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt]}%
263     \temp@cmd {\plot@macro};
264 \else
265     \stepcounter{gnuplot@id}%
266     \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt]}
267     \temp@cmd gnuplot[gnuplot@degrees,gnuplot@prefix] {\plot@macro};
268 \fi
269 }%
270 }

```

\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.

```

271 \newcommand{\addBodeTFPlot}[3][thick]{%
272 \gdef\plot@macro{}%
273 \gdef\temp@macro{}%
274 \ifnum\pdfstrcmp{#2}{phase}=0
275     \build@TF@plot{\temp@macro}{\plot@macro}{#3}%
276 \else
277     \build@TF@plot{\plot@macro}{\temp@macro}{#3}%
278 \fi
279 \if@pgfarg
280     \addplot[variable=t,#1]{\plot@macro};
281 \else
282     \stepcounter{gnuplot@id}%
283     \addplot[variable=t,#1] gnuplot[gnuplot@degrees, gnuplot@prefix] {\plot@macro};
284 \fi
285 }

```

\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.

```

286 \newcommand{\addBodeComponentPlot}[2][thick]{%
287 \if@pgfarg
288     \addplot[variable=t,#1]{#2};
289 \else
290     \stepcounter{gnuplot@id}%
291     \addplot[variable=t,#1] gnuplot[gnuplot@degrees,gnuplot@prefix] {#2};
292 \fi
293 }

```

BodePlot (*env.*) An environment to host 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.

```

294 \newenvironment{BodePlot}[3][{}]{%
295 \parse@env@opt{#1}%
296 \edef\temp@cmd{\noexpand\begin{tikzpicture}}[\unexpanded\expandafter{\opt@tikz}]
297 \noexpand\begin{semilogaxis}[%
298     bode@style,
299     xmin={#2},
300     xmax={#3},
301     domain=#2:#3,
302     height=2.5cm,
303     xlabel={Frequency (rad/s)},
304     \unexpanded\expandafter{\opt@axes}
305 ]%
306 }%
307 \temp@cmd
308 }{
309     \end{semilogaxis}
310 \end{tikzpicture}

```



```
311 }
```

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](#).

```
312 \newcommand*\add@feature}[3]{%
313   \ifcat$\detokenize\expandafter{#1}$%
314     \xdef#1{\unexpanded\expandafter{#1 0+#2}}%
315   \else
316     \xdef#1{\unexpanded\expandafter{#1+#2}}%
317   \fi
318   \foreach \y [count=\n] in #3 {%
319     \xdef#1{\unexpanded\expandafter{#1}{\y}}%
320     \xdef\Last@LoopValue{\n}%
321   }%
322   \ifnum\Last@LoopValue=1%
323     \xdef#1{\unexpanded\expandafter{#1}{0}}%
324   \fi
325 }
```

\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.

```
326 \newcommand*\build@ZPK@plot}[4]{%
327   \foreach \feature/\values in {#4} {%
328     \ifnum\pdfstrcmp{\feature}{z}=0
329       \foreach \z in \values {%
330         \ifnum\pdfstrcmp{#3}{linear}=0
331           \add@feature{#2}{\PhZeroLin}{\z}%
332           \add@feature{#1}{\MagZeroLin}{\z}%
333         \else
334           \ifnum\pdfstrcmp{#3}{asymptotic}=0
335             \add@feature{#2}{\PhZeroAsymp}{\z}%
336             \add@feature{#1}{\MagZeroAsymp}{\z}%
337           \else
338             \add@feature{#2}{\PhZero}{\z}%
339             \add@feature{#1}{\MagZero}{\z}%
340           \fi
341         \fi
342       }%
343     \fi
344     \ifnum\pdfstrcmp{\feature}{p}=0
345       \foreach \p in \values {%
346         \ifnum\pdfstrcmp{#3}{linear}=0
347           \add@feature{#2}{\PhPoleLin}{\p}%
348           \add@feature{#1}{\MagPoleLin}{\p}%
349         \else
350           \ifnum\pdfstrcmp{#3}{asymptotic}=0
351             \add@feature{#2}{\PhPoleAsymp}{\p}%
352             \add@feature{#1}{\MagPoleAsymp}{\p}%
353           \else
354             \add@feature{#2}{\PhPole}{\p}%
355             \add@feature{#1}{\MagPole}{\p}%

```

```

356         \fi
357     \fi
358 }%
359 \fi
360 \ifnum\pdfstrcmp{\feature}{k}=0
361     \ifnum\pdfstrcmp{#3}{linear}=0
362         \add@feature{#2}{\PhKLin}{\values}%
363         \add@feature{#1}{\MagKLin}{\values}%
364     \else
365         \ifnum\pdfstrcmp{#3}{asymptotic}=0
366             \add@feature{#2}{\PhKAsymp}{\values}%
367             \add@feature{#1}{\MagKAsymp}{\values}%
368         \else
369             \add@feature{#2}{\PhK}{\values}%
370             \add@feature{#1}{\MagK}{\values}%
371         \fi
372     \fi
373 \fi
374 \ifnum\pdfstrcmp{\feature}{d}=0
375     \ifnum\pdfstrcmp{#3}{linear}=0
376         \PackageError {bodeplot} {Linear approximation for pure delays is not
377             supported.} {Plot the true Bode plot using 'true' instead of 'linear'.}
378     \else
379         \ifnum\pdfstrcmp{#3}{asymptotic}=0
380             \PackageError {bodeplot} {Asymptotic approximation for pure de-
381             lays is not
382             supported.} {Plot the true Bode plot using 'true' instead of 'asymptotic'.}
383         \else
384             \ifdim\values pt < 0pt
385                 \PackageError {bodeplot} {Delay needs to be a positive number.}
386             \fi
387             \add@feature{#2}{\PhDel}{\values}%
388             \add@feature{#1}{\MagDel}{\values}%
389         \fi
390     \fi
391 }%
392 }

```

\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**).

```

393 \newcommand{\build@TF@plot}[3]{%
394     \gdef\num@real{0}%
395     \gdef\num@im{0}%
396     \gdef\den@real{0}%
397     \gdef\den@im{0}%
398     \gdef\loop@delay{0}%
399     \foreach \feature/\values in {#3} {%
400         \ifnum\pdfstrcmp{\feature}{num}=0
401             \foreach \numcoeff [count=\numpow] in \values {%
402                 \xdef\num@degree{\numpow}%
403             }%
404             \foreach \numcoeff [count=\numpow] in \values {%
405                 \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}%
406                 \ifnum\currentdegree = 0
407                     \xdef\num@real{\num@real+\numcoeff}%
408                 \else
409                     \ifodd\currentdegree
410                         \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-1}){(\currentdegree-
411 1)/2})*%

```

```

412         \else
413         \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-1}{(\currentdegree)/2})*%
414         (\n@pow{t}{\currentdegree}))}%
415         \fi
416     \fi
417 }%
418 \fi
419 \ifnum\pdfstrcmp{\feature}{den}=0
420     \foreach \dencoeff [count=\denpow] in \values {%
421         \xdef\den@degree{\denpow}%
422     }%
423     \foreach \dencoeff [count=\denpow] in \values {%
424         \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}%
425         \ifnum\currentdegree = 0
426             \xdef\den@real{\den@real+\dencoeff}%
427         \else
428             \ifodd\currentdegree
429                 \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-1}{(\currentdegree-
430                 1)/2})*%
431                 (\n@pow{t}{\currentdegree}))}%
432             \else
433                 \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-1}{(\currentdegree)/2})*%
434                 (\n@pow{t}{\currentdegree}))}%
435             \fi
436         \fi
437     }%
438 \ifnum\pdfstrcmp{\feature}{d}=0
439     \xdef\loop@delay{\values}%
440 \fi
441 }%
442 \xdef#2{(\n@mod{atan2((\num@im),(\num@real))-atan2((\den@im),%
443 (\den@real))-\loop@delay*180*t/pi+360}{360})}%
444 \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2})))-%
445 20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2}))))}%
446 }

```

\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.

```

447 \newcommand{\parse@opt}[1]{%
448     \gdef\optmag@axes{}%
449     \gdef\optph@axes{}%
450     \gdef\optph@plot{}%
451     \gdef\optmag@plot{}%
452     \gdef\opt@group{}%
453     \gdef\opt@approx{}%
454     \gdef\optph@commands{}%
455     \gdef\optmag@commands{}%
456     \gdef\opt@tikz{}%
457     \foreach \obj/\typ/\opt in {#1} {%
458         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{plot}=0
459             \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{mag}=0
460                 \xdef\optmag@plot{\unexpanded\expandafter{\opt}}%
461             \else
462                 \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
463                     \xdef\optph@plot{\unexpanded\expandafter{\opt}}%
464                 \else

```

```

465         \xdef\optmag@plot{\unexpanded\expandafter{\opt}}%
466         \xdef\optph@plot{\unexpanded\expandafter{\opt}}%
467     \fi
468 \fi
469 \else
470     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0
471         \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{mag}=0
472             \xdef\optmag@axes{\unexpanded\expandafter{\opt}}%
473         \else
474             \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
475                 \xdef\optph@axes{\unexpanded\expandafter{\opt}}%
476             \else
477                 \xdef\optmag@axes{\unexpanded\expandafter{\opt}}%
478                 \xdef\optph@axes{\unexpanded\expandafter{\opt}}%
479             \fi
480         \fi
481 \else
482     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{group}=0
483         \xdef\opt@group{\unexpanded\expandafter{\opt}}%
484     \else
485         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{approx}=0
486             \xdef\opt@approx{\unexpanded\expandafter{\opt}}%
487         \else
488             \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{commands}=0
489                 \ifnum\pdfstrcmp{\unexpanded\expandafter{\typ}}{ph}=0
490                     \xdef\optph@commands{\unexpanded\expandafter{\opt}}%
491                 \else
492                     \xdef\optmag@commands{\unexpanded\expandafter{\opt}}%
493                 \fi
494             \else
495                 \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{tikz}=0
496                     \xdef\opt@tikz{\unexpanded\expandafter{\opt}}%
497                 \else
498                     \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
499                     \unexpanded\expandafter{\obj}}%
500                     \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
501                     \unexpanded\expandafter{\obj}}%
502                 \fi
503             \fi
504         \fi
505     \fi
506 \fi
507 \fi
508 }%
509 }

```

`\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.

```

510 \newcommand{\parse@env@opt}[1]{%
511     \gdef\opt@axes{}%
512     \gdef\opt@tikz{}%
513     \foreach \obj/\opt in {#1} {%
514         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0
515             \xdef\opt@axes{\unexpanded\expandafter{\opt}}%
516         \else
517             \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{tikz}=0
518                 \xdef\opt@tikz{\unexpanded\expandafter{\opt}}%
519             \else
520                 \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
521                 \unexpanded\expandafter{\obj}}%

```

```

522     \fi
523     \fi
524 }%
525 }

```

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!

```

526 \newcommand{\NyquistZPK}[4][[]]{%
527   \parse@N@opt{#1}%
528   \gdef\func@mag{}%
529   \gdef\func@ph{}%
530   \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
531   \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]{%
532     \noexpand\begin{axis}%
533       bode@style,
534       domain=#3:#4,
535       height=5cm,
536       xlabel={\Re$},
537       ylabel={\Im$},
538       samples=500,
539       \unexpanded\expandafter{\opt@axes}
540     ]%
541   }%
542   \temp@cmd
543     \addplot [only marks,mark=+,thick,red] (-1 , 0);
544     \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\unexpanded\expandafter{\opt@
545     \ifpgfarg
546       \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}}*cos(\func@ph)},
547       {\n@pow{10}{((\func@mag)/20)}}*sin(\func@ph)} );
548     \opt@commands
549   \else
550     \stepcounter{gnuplot@id}%
551     \temp@cmd gnuplot[parametric,gnuplot@degrees,gnuplot@prefix] {%
552       \n@pow{10}{((\func@mag)/20)}}*cos(\func@ph),
553       \n@pow{10}{((\func@mag)/20)}}*sin(\func@ph)};
554     \opt@commands
555     \fi
556   \end{axis}
557 \end{tikzpicture}
558 }

```

\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**.

```

559 \newcommand{\NyquistTF}[4][[]]{%
560   \parse@N@opt{#1}%
561   \gdef\func@mag{}%
562   \gdef\func@ph{}%
563   \build@TF@plot{\func@mag}{\func@ph}{#2}%
564   \edef\temp@cmd{\noexpand\begin{tikzpicture}[\unexpanded\expandafter{\opt@tikz}]{%
565     \noexpand\begin{axis}%

```

```

566     bode@style,
567     domain=#3:#4,
568     height=5cm,
569     xlabel={\$Re\$},
570     ylabel={\$Im\$},
571     samples=500,
572     \unexpanded\expandafter{\opt@axes}
573 ]%
574 }%
575 \temp@cmd
576 \addplot [only marks,mark=+,thick,red] (-1 , 0);
577 \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\unexpanded\expandafter{\opt@
578 \if@pgfarg
579     \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
580               {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
581     \opt@commands
582 \else
583     \stepcounter{gnuplot@id}%
584     \temp@cmd gnuplot[parametric,gnuplot@degrees,gnuplot@prefix]{%
585         \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
586         \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
587     \opt@commands
588 \fi
589 \end{axis}
590 \end{tikzpicture}
591 }

```

\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.

```

592 \newcommand{\addNyquistZPKPlot}[2][]{%
593     \gdef\func@mag{%
594     \gdef\func@ph{%
595     \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
596     \if@pgfarg
597         \addplot[variable=t,#1] ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
598                                   {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
599     \else
600         \stepcounter{gnuplot@id}%
601         \addplot[variable=t,#1] gnuplot[parametric,gnuplot@degrees,gnuplot@prefix]{%
602             \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
603             \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
604     \fi
605 }

```

\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.

```

606 \newcommand{\addNyquistTFPlot}[2][]{%
607     \gdef\func@mag{%
608     \gdef\func@ph{%
609     \build@TF@plot{\func@mag}{\func@ph}{#2}%
610     \if@pgfarg
611         \addplot[variable=t,#1] ( {\n@pow{10}{((\func@mag)/20)}*cos(\func@ph)},
612                                   {\n@pow{10}{((\func@mag)/20)}*sin(\func@ph)} );
613     \else
614         \stepcounter{gnuplot@id}%
615         \addplot[variable=t,#1] gnuplot[parametric,gnuplot@degrees,gnuplot@prefix]{%
616             \n@pow{10}{((\func@mag)/20)}*cos(\func@ph),
617             \n@pow{10}{((\func@mag)/20)}*sin(\func@ph)};
618     \fi

```

619 }

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.

```

620 \newenvironment{NyquistPlot}[3][{}]{%
621   \parse@env@opt{#1}%
622   \edef\temp@cmd{\noexpand\begin{tikzpicture}}\unexpanded\expandafter{\opt@tikz}}%
623   \noexpand\begin{axis}%
624     bode@style,
625     height=5cm,
626     domain=#2:#3,
627     xlabel={\Re$},
628     ylabel={\Im$},
629     \unexpanded\expandafter{\opt@axes}
630   ]%
631 }%
632 \temp@cmd
633 \addplot [only marks,mark=+,thick,red] (-1 , 0);
634 }{%
635   \end{axis}
636   \end{tikzpicture}
637 }
```

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.

```

638 \newcommand{\parse@N@opt}[1]{%
639   \gdef\opt@axes{}%
640   \gdef\opt@plot{}%
641   \gdef\opt@commands{}%
642   \gdef\opt@tikz{}
643   \foreach \obj/\opt in {#1} {%
644     \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{axes}=0
645       \xdef\opt@axes{\unexpanded\expandafter{\opt}}%
646     \else
647       \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{plot}=0
648         \xdef\opt@plot{\unexpanded\expandafter{\opt}}%
649       \else
650         \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{commands}=0
651           \xdef\opt@commands{\unexpanded\expandafter{\opt}}%
652         \else
653           \ifnum\pdfstrcmp{\unexpanded\expandafter{\obj}}{tikz}=0
654             \xdef\opt@tikz{\unexpanded\expandafter{\opt}}%
655           \else
656             \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
657               \unexpanded\expandafter{\obj}}%
658           \fi
659         \fi
660       \fi
661     \fi
662   }%
663 }
```

4.6 Nichols charts

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

`\NicholsTF`

`NicholsChart`

`\addNicholsZPKChart`

`\addNicholsTFChart`

```

664 \newcommand{\NicholsZPK}[4][[]]{%
665   \parse@N@opt{#1}%
666   \gdef\func@mag{}%
667   \gdef\func@ph{}%
668   \build@ZPK@plot{\func@mag}{\func@ph}{#2}%
669   \edef\temp@cmd{\noexpand\begin{tikzpicture}}\unexpanded\expandafter{\opt@tikz}}%
670   \noexpand\begin{axis}%
671     bode@style,
672     domain=#3:#4,
673     height=5cm,
674     xlabel={Phase (degrees)},
675     ylabel={Gain (dB)},
676     samples=500,
677     \unexpanded\expandafter{\opt@axes}
678   ]%
679 }%
680 \temp@cmd
681   \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt@plot]}%
682   \ifpgfarg
683     \temp@cmd ( {\func@ph} , {\func@mag} );
684     \opt@commands
685   \else
686     \stepcounter{gnuplot@id}%
687     \temp@cmd gnuplot[parametric, gnuplot@degrees, gnuplot@prefix]
688       { \func@ph , \func@mag };
689     \opt@commands
690   \fi
691   \end{axis}
692 \end{tikzpicture}
693 }
694 \newcommand{\NicholsTF}[4][[]]{%
695   \parse@N@opt{#1}%
696   \gdef\func@mag{}%
697   \gdef\func@ph{}%
698   \build@TF@plot{\func@mag}{\func@ph}{#2}%
699   \edef\temp@cmd{\noexpand\begin{tikzpicture}}\unexpanded\expandafter{\opt@tikz}}%
700   \noexpand\begin{axis}%
701     bode@style,
702     domain=#3:#4,
703     height=5cm,
704     xlabel={Phase (degrees)},
705     ylabel={Gain (dB)},
706     samples=500,
707     \unexpanded\expandafter{\opt@axes}
708   ]%
709 }%
710 \temp@cmd
711   \edef\temp@cmd{\noexpand\addplot[variable=t,thick,\opt@plot]}%
712   \ifpgfarg
713     \temp@cmd ( {\func@ph} , {\func@mag} );
714     \opt@commands
715   \else
716     \stepcounter{gnuplot@id}%
717     \temp@cmd gnuplot[parametric, gnuplot@degrees, gnuplot@prefix]
718       { \func@ph , \func@mag };
719     \opt@commands
720   \fi
721   \end{axis}
722 \end{tikzpicture}
723 }
724 \newenvironment{NicholsChart}[3][[]]{%
725   \parse@env@opt{#1}%
726   \edef\temp@cmd{\noexpand\begin{tikzpicture}}\unexpanded\expandafter{\opt@tikz}}%

```



```

727 \noexpand\begin{axis}[%
728     bode@style,
729     domain=#2:#3,
730     height=5cm,
731     xlabel={Phase (degrees)},
732     ylabel={Gain (dB)},
733     \unexpanded\expandafter{\opt@axes}
734 ]%
735 }%
736 \temp@cmd
737 }{
738     \end{axis}
739 \end{tikzpicture}
740 }
741 \newcommand{\addNicholsZPKChart}[2][]{%
742     \gdef\func@mag{}%
743     \gdef\func@ph{}%
744     \build@ZPK@plot{\func@mag}{\func@ph}{\func@ph}{#2}%
745     \ifpgfarg
746         \addplot[variable=t,#1] ( {\func@ph} , {\func@mag} );
747     \else
748         \stepcounter{gnuplot@id}%
749         \addplot[variable=t,#1] gnuplot[parametric,gnuplot@degrees,gnuplot@prefix]
750             {\func@ph , \func@mag};
751     \fi
752 }
753 \newcommand{\addNicholsTFChart}[2][]{%
754     \gdef\func@mag{}%
755     \gdef\func@ph{}%
756     \build@TF@plot{\func@mag}{\func@ph}{#2}%
757     \ifpgfarg
758         \addplot[variable=t,#1] ( {\func@ph} , {\func@mag} );
759     \else
760         \stepcounter{gnuplot@id}%
761         \addplot[variable=t,#1] gnuplot[gnuplot@degrees,gnuplot@prefix]
762             {\func@ph , \func@mag};
763     \fi
764 }

```

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	\addBodeZPKPlots .. 252	326 , 530 , 595 , 668 , 744
\@declutterargfalse . 9	\addNicholsTFChart 664	
\@declutterargtrue . 11	\addNicholsZPKChart 664	C
\@pgfargfalse 5	\addNyquistTFPlot . 606	\circ .. 185 , 198 , 231 , 244
\@pgfargtrue 7	\addNyquistZPKPlot 592	\currentdegree . 405 ,
		406 , 409 , 410 , 411 ,
		413 , 414 , 424 , 425 ,
		428 , 429 , 430 , 432 , 433
A	B	D
\add@feature 312 , 331 , 332 , 335 ,	\bode@style 50	\den@degree 421 , 424
336 , 338 , 339 , 347 ,	BodePlot (env.) 294	\den@im 397 , 429 , 442 , 445
348 , 351 , 352 , 354 ,	\BodeTF 206	\den@real 396 ,
355 , 362 , 363 , 366 ,	\BodeZPK 160	426 , 432 , 443 , 445
367 , 369 , 370 , 386 , 387	\build@TF@plot	\dencoeff 420 ,
\addBodeComponentPlot	.. 210 , 275 , 277 ,	423 , 426 , 429 , 432
..... 286	393 , 563 , 609 , 698 , 756	\denpow 420 , 421 , 423 , 424
\addBodeTFPlot 271	\build@ZPK@plot ...	
	.. 164 , 257 , 259 ,	

E		\MagCSPOlesPeak ... 124	\opt@group 174 , 220 , 452 , 483
environments:		\MagCSZeros 105	\opt@plot .. 544 , 577 , 640 , 648 , 656 , 681 , 711
BodePlot 294		\MagCSZerosAsymp .. 105	\opt@tikz .. 165 , 211 , 296 , 456 , 496 , 512 , 518 , 531 , 564 , 622 , 642 , 654 , 669 , 699 , 726
F		\MagCSZerosPeak ... 124	\optmag@axes 180 , 192 , 226 , 238 , 448 , 472 , 477
\func@mag 162 , 164 , 182 , 194 , 208 , 210 , 228 , 240 , 528 , 530 , 546 , 547 , 552 , 553 , 561 , 563 , 579 , 580 , 585 , 586 , 593 , 595 , 597 , 598 , 602 , 603 , 607 , 609 , 611 , 612 , 616 , 617 , 666 , 668 , 683 , 688 , 696 , 698 , 713 , 718 , 742 , 744 , 746 , 750 , 754 , 756 , 758 , 762		\MagDel 70 , 387	\optmag@commands 183 , 195 , 229 , 241 , 455 , 492
		\MagK 64 , 370	\optmag@plot 181 , 193 , 227 , 239 , 451 , 460 , 465 , 498
		\MagKAsymp 64 , 367	\optph@axes 185 , 198 , 231 , 244 , 449 , 475 , 478
		\MagKLin 64 , 363	\optph@commands 188 , 201 , 234 , 247 , 454 , 490
\func@ph 163 , 164 , 187 , 200 , 209 , 210 , 233 , 246 , 529 , 530 , 546 , 547 , 552 , 553 , 562 , 563 , 579 , 580 , 585 , 586 , 594 , 595 , 597 , 598 , 602 , 603 , 608 , 609 , 611 , 612 , 616 , 617 , 667 , 668 , 683 , 688 , 697 , 698 , 713 , 718 , 743 , 744 , 746 , 750 , 755 , 756 , 758 , 762		\MagPole 72 , 99 , 355	\optph@plot 186 , 199 , 232 , 245 , 450 , 463 , 466 , 500
		\MagPoleAsymp 72 , 101 , 352	
		\MagPoleLin . 72 , 100 , 348	
		\MagSOPoles 132	
		\MagSOPolesAsymp .. 132	
		\MagSOPolesLin 132	
		\MagSOPolesPeak ... 150	
		\MagS0Zeros 132	
		\MagS0ZerosAsymp .. 132	
		\MagS0ZerosLin 132	
		\MagS0ZerosPeak ... 150	
		\MagZero 99 , 339	
		\MagZeroAsymp .. 99 , 336	
		\MagZeroLin 99 , 332	
G		N	
\gnuplot@degrees 5		\n 318 , 320	\p 345 , 347 , 348 , 351 , 352 , 354 , 355
\gnuplot@id 5		\n@mod 5 , 80 , 442	\parse@env@opt 295 , 510 , 621 , 725
\gnuplot@prefix 5		\n@pow 5 , 73 , 74 , 75 , 85 , 86 , 88 , 89 , 91 , 92 , 93 , 94 , 95 , 96 , 105 , 106 , 109 , 110 , 111 , 113 , 115 , 116 , 133 , 137 , 410 , 411 , 413 , 414 , 429 , 430 , 432 , 433 , 444 , 445 , 546 , 547 , 552 , 553 , 579 , 580 , 585 , 586 , 597 , 598 , 602 , 603 , 611 , 612 , 616 , 617	\parse@N@opt ... 527 , 560 , 638 , 665 , 695
I		\newcounter 20	\parse@opt . 161 , 207 , 447
\if@declutterarg 9 , 22 , 45		\newenvironment 294 , 620 , 724	\pdf@strcmp 3
\ifcat 313		\NicholsChart 664	\pdfstrcmp ... 1 , 256 , 274 , 328 , 330 , 334 , 344 , 346 , 350 , 360 , 361 , 365 , 374 , 375 , 379 , 400 , 419 , 438 , 458 , 459 , 462 , 470 , 471 , 474 , 482 , 485 , 488 , 489 , 495 , 514 , 517 , 644 , 647 , 650 , 653
\ifluatex 2		\NicholsTF 664	\pgfplots@gnuplot@format 40
\ifnum 39 , 256 , 274 , 322 , 328 , 330 , 334 , 344 , 346 , 350 , 360 , 361 , 365 , 374 , 375 , 379 , 400 , 406 , 419 , 425 , 438 , 458 , 459 , 462 , 470 , 471 , 474 , 482 , 485 , 488 , 489 , 495 , 514 , 517 , 644 , 647 , 650 , 653		\NicholsZPK 664	\pgfplotsset 37 , 50
\ifwindows 44		\num@degree 402 , 405	\PhCSPOles 105
\immediate 46		\num@im 395 , 410 , 442 , 444	\PhCSPOlesAsymp 105 , 142
J		\num@real 394 , 407 , 413 , 442 , 444	\PhCSPOlesLin .. 105 , 139
\jobname 26 , 33		\numcoeff 401 , 404 , 407 , 410 , 413	\PhCSZeros 105
L		\numpow 401 , 402 , 404 , 405	\PhCSZerosAsymp ... 105
\Last@LoopValue 320 , 322		\NyquistPlot 620	\PhCSZerosLin 105
\loop@delay 398 , 439 , 443		\NyquistTF 559	\PhDel 71 , 386
M		\NyquistZPK 526	\PhK 64 , 369
\MagCSPOles 105		O	
\MagCSPOlesAsymp .. 105		\opt@approx 164 , 453 , 486	\PhKAsymp 64 , 70 , 366
\MagCSPOlesLin 105		\opt@axes 304 , 511 , 515 , 520 , 539 , 572 , 629 , 639 , 645 , 677 , 707 , 733	\PhKLin 64 , 70 , 362
		\opt@commands .. 548 , 554 , 581 , 587 , 641 , 651 , 684 , 689 , 714 , 719	\PhPole 72 , 102 , 354
			\PhPoleAsymp 72 , 104 , 351
			\PhPoleLin .. 72 , 103 , 347
			\PhSOPoles 132
			\PhSOPolesAsymp ... 132
			\PhSOPolesLin 132
			\PhS0Zeros 132
			\PhS0ZerosAsymp ... 132
			\PhS0ZerosLin 132

