The bodeplot package
version 2.1

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

September 30, 2025

Contents

1 Introduction **2**
1.1 External Dependencies . . . 2
1.2 Directory Structure . . . 2
1.3 Limitations . . . 2

2 TL;DR **3**

3 Usage **10**
3.1 Bode plots . . . 10
3.1.1 Basic components up to first order . . . 14
3.1.2 Basic components of the second order . . . 15
3.2 Nyquist plots . . . 16
3.3 Nichols charts . . . 18
3.4 Pole-zero maps . . . 20

4 Implementation **21**
4.1 Initialization . . . 21
4.2 Parametric function generators for poles, zeros, gains, and delays. . . . 23
4.3 Second order systems. . . . 24
4.4 Commands for Bode plots . . . 26
4.4.1 User macros . . . 26
4.4.2 Internal macros . . . 32
4.5 Nyquist plots . . . 37
4.5.1 User macros . . . 37
4.5.2 Internal commands . . . 40
4.6 Nichols charts . . . 47

Index **56**

Change History **59**

1 Introduction

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

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

and the zero-pole-gain (ZPK) form

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

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

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

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

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

1.1 External Dependencies

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

1.2 Directory Structure

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

1.3 Limitations

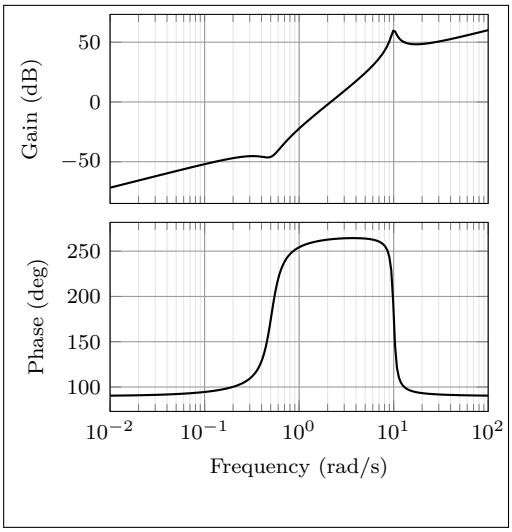
- Before version 1.2, in `pgf` mode, the package set `trig format plots` to `rad` globally. Version 1.2 onwards, this option is passed to each `addplot` command individually so that it does not affect other plots in the document. To roll back to the pre-1.2 behavior, load the package with `\usepackage[pgf]{bodeplot}[=2024-02-06]`.
- In `pgf` mode, Bode phase plots and Nichols charts in TF form wrap angles so that they are always between -180 and 180° or $-\pi$ and $-\pi$ 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)

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

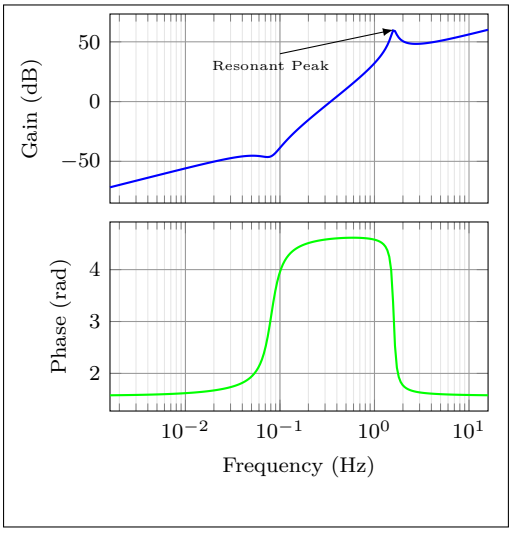
Bode plot in ZPK format



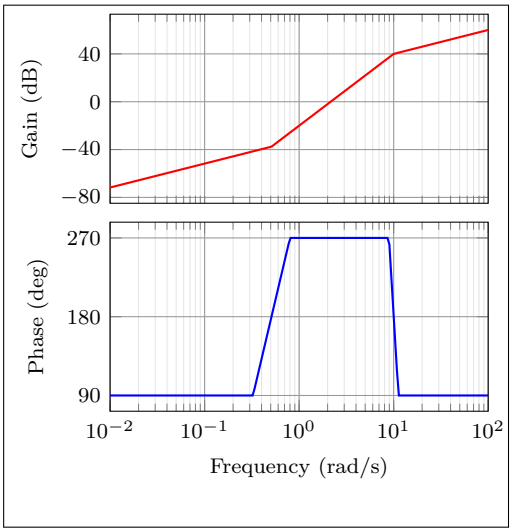

```
\BodeZPK{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
{0.01}
{100}
```

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)}
```



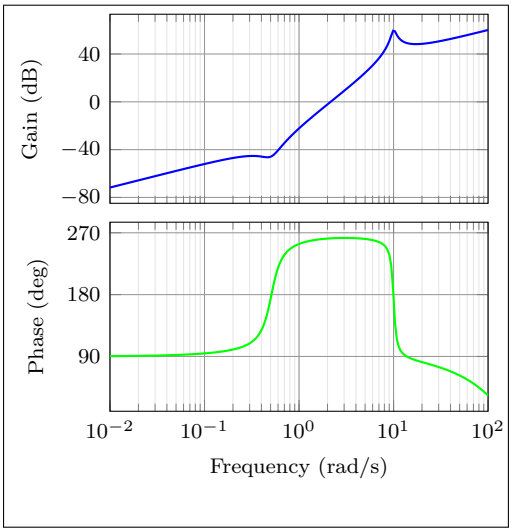
Linear approximation with customization



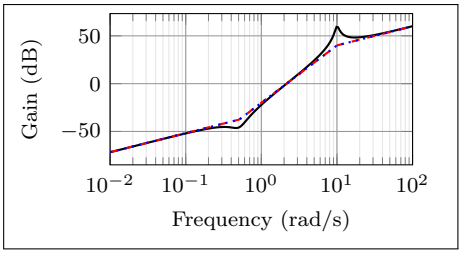

```
\BodeZPK[%
  plot/mag/{red,thick},
  plot/ph/{blue,thick},
  axes/mag/{ytick distance=40},
  axes/ph/{ytick distance=90},
  approx/linear%
]{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
{0.01}
{100}
```

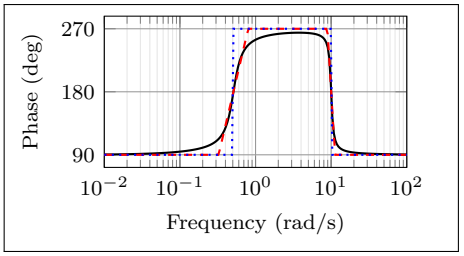
Plot with delay and customization

```
\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



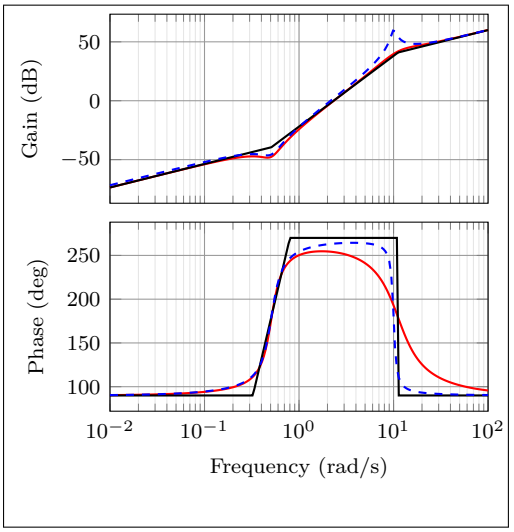



```
\begin{BodeMagPlot}[%
  axes/{height=2cm,
  width=4cm}%
]
{0.01}
{100}
  \addBodeZPKPlots[%
    true/{black,thick},
    linear/{red,dashed,thick},
    asymptotic/{blue,dotted,thick}%
  ]
  {magnitude}
  {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10%
  }
\end{BodeMagPlot}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

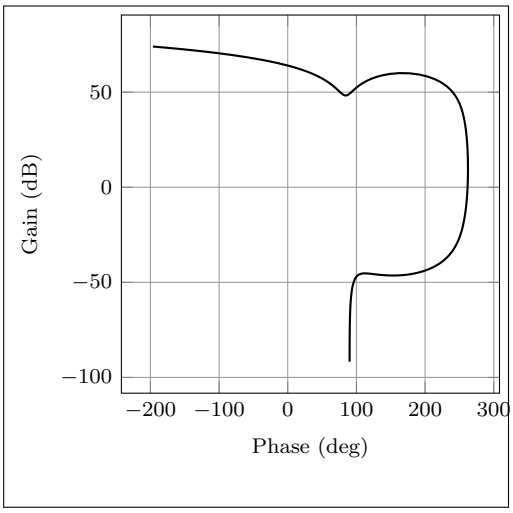
```
\begin{BodePhPlot}[%
  height=2cm,
  width=4cm,
  ytick distance=90
]
{0.01}
{100}
  \addBodeZPKPlots[%
    true/{black,thick},
    linear/{red,dashed,thick},
    asymptotic/{blue,dotted,thick}%
  ]
  {phase}
  {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10%
  }
\end{BodePhPlot}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

Multiple transfer functions in a single Bode plot using the `BodePlot` environment and the `\addBodePlot` macro introduced in v2.1.

```
\begin{BodePlot}{0.01}{100}
  \addBodePlot[red,thick]{zpk}{%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-5,-10},{-5,10}},
    k/10%
  }
  \addBodePlot[black,thick,linear]{zpk}{%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-5,-10},{-5,10}},
    k/10%
  }
  \addBodePlot[blue,dashed]{tf}{%
    num/{10,2,2.6,0},
    den/{1,1,100.25}%
  }
\end{BodePlot}
```



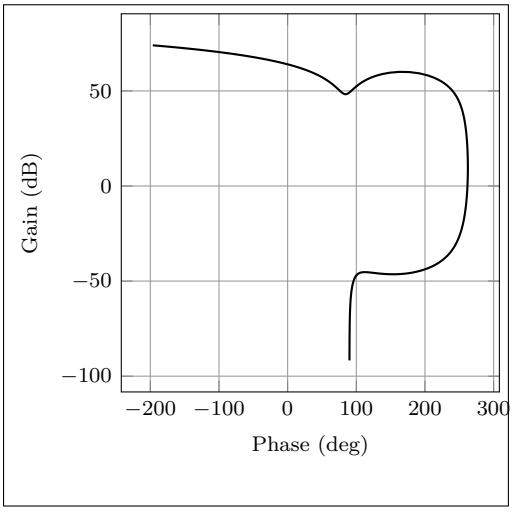
Nichols chart



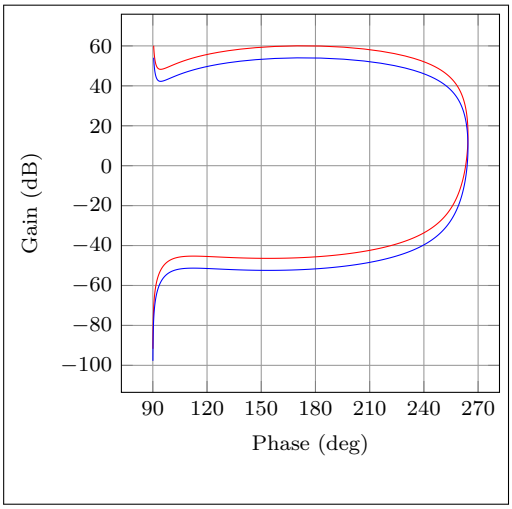

```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01%
}
{0.001}
{500}
```

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

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



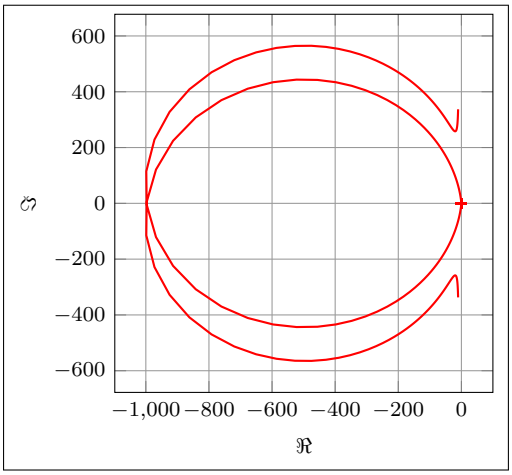
Multiple Nichols charts with customization



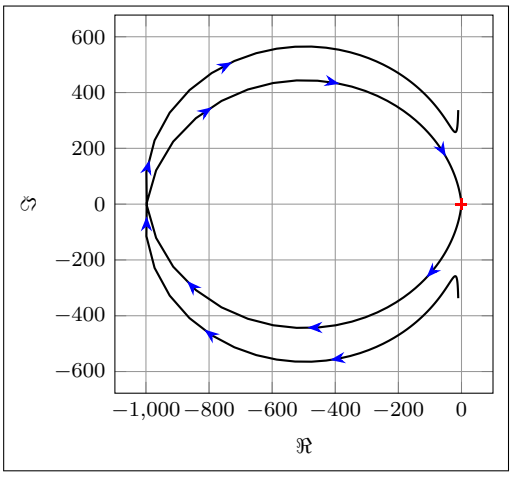

```
\begin{NicholsChart}[%
  ytick distance=20,
  xtick distance=30
]
{0.001}
{100}
  \addNicholsZPKChart [red,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10%
  }
  \addNicholsZPKChart [blue,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},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

Nyquist plot

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



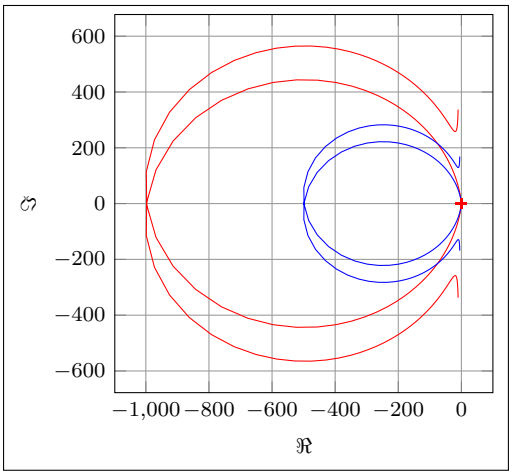
Nyquist plot in TF format with arrows



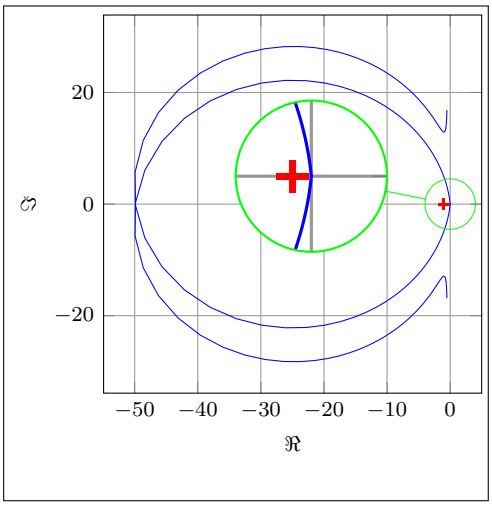

```
\NyquistTF[%
  plot/{%
    samples=1000,
    postaction=decorate,
    decoration={%
      markings,
      mark=between positions 0.1 and 0.9 step 5em with {%
        \arrow{Stealth [length=2mm, blue]}
      }
    }
  }%
]
{%
  num/{10,2,2.6,0},
  den/{1,1,100.25}%
}
{-30}
{30}
```

Multiple Nyquist plots with customization

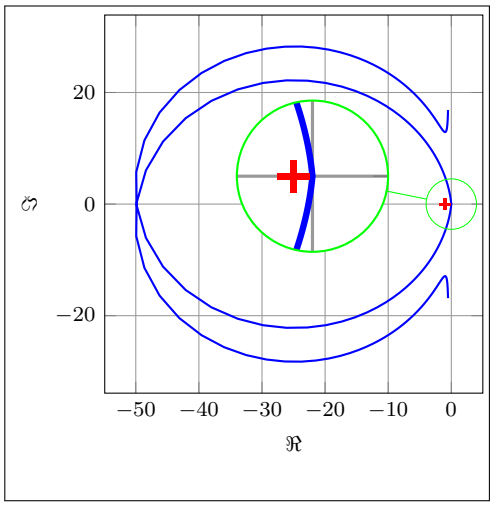
```
\begin{NyquistPlot}{-30}{30}
  \addNyquistZPKPlot [red,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10%
  }
  \addNyquistZPKPlot [blue,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/5%
  }
\end{NyquistPlot}
```



Nyquist plots with additional commands, using two different macros

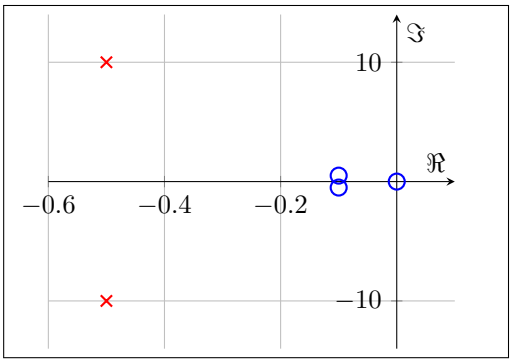



```
\begin{NyquistPlot}[%
  tikz/{
    spy using outlines={%
      circle,
      magnification=3,
      connect spies,
      size=2cm
    }
  }%
]
{-30}{30}
  \addNyquistZPKPlot [blue,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/0.5%
  }
  \coordinate (spyon)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




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

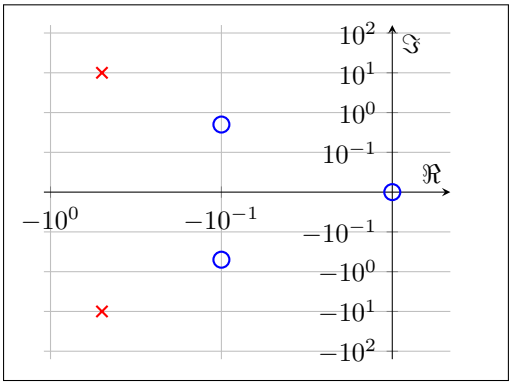
Pole-zero map




```
\PoleZeroMapZPK
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
```

Pole-zero map (symmetric log scale)

```
\PoleZeroMapZPK[scale/{log}]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
```



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 getenrated in degrees unless either the `rad` package option is used or the optional argument `tikz/{frequency unit=rad}` is supplied to the `tikzpicture` environment.

3.1 Bode plots

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

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

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

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

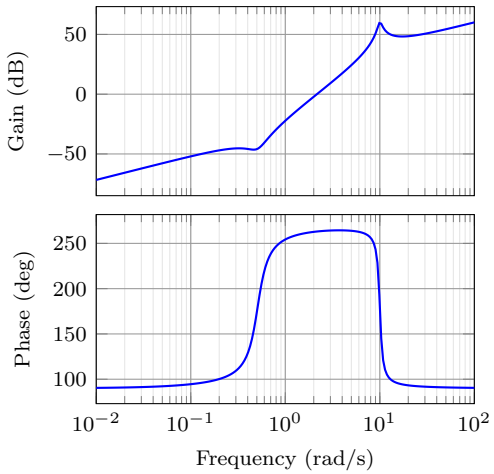


Figure 1: Output of the `\BodeZPK` macro.

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

For example, given a transfer function

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

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

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

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

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

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

which generates the plot in Figure 2.

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

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

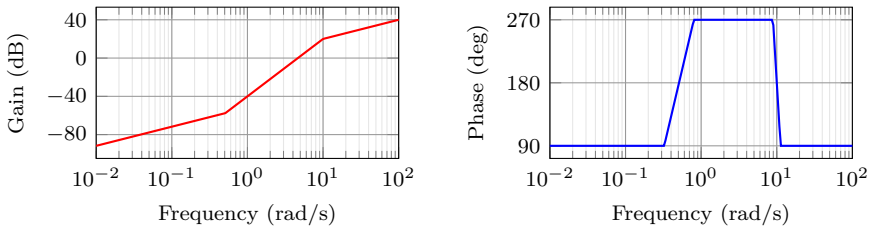


Figure 2: Customization of the default `\BodeZPK` macro.

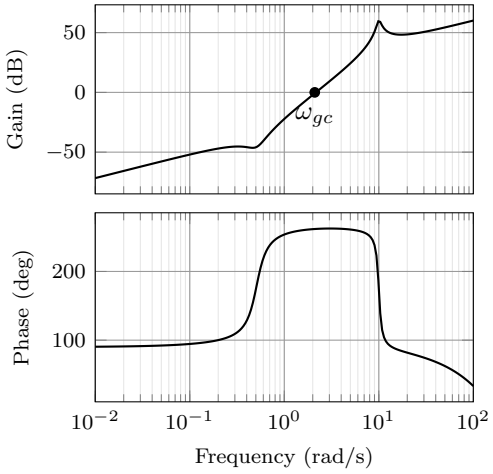


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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

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

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

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

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

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

`BodeMagPlot` (*env.*)

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

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

- Tuples of the form `obj/{opt}`:
  - `tikz/{opt}`: modify picture properties by adding options `{opt}` to the `tikzpicture` environment.
  - `axes/{opt}`: modify axis properties by adding options `{opt}` to the `semilogaxis` environment.
- 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}[⟨obj1/{⟨opt1⟩},obj2/{⟨opt2⟩},...⟩]
   {⟨min-frequency⟩}{⟨max-frequency⟩}
 \addBode...
\end{BodePhPlot}
```

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

`\addBodeZPKPlots`

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

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[⟨plot-options⟩]
 {⟨plot-type⟩}
 {⟨num/{⟨coeffs⟩},den/{⟨coeffs⟩},d/{⟨delay⟩}⟩}
```

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

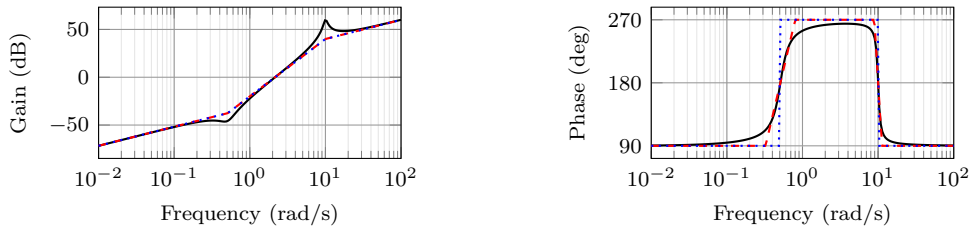


Figure 4: Superimposed approximate and true Bode plots using the `BodeMagPlot` and `BodePhPlot` environments and the `\addBodeZPKPlots` macro.

with the `BodePlot` environment supplied with this package is recommended. The second mandatory argument, `plot-type` is either magnitude or `phase`. If it is not equal to `phase`, it is assumed to be `magnitude`. The last mandatory argument is the same as `\BodeTF`.

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

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

`\addBodePlot` `\addBodePlot[`⟨*plot-options*⟩`]{`⟨*system-type*⟩`}{`⟨*system-data*⟩`}`

Unified macro to add Bode plots for both ZPK and TF system representations. Unlike the `\addBodeZPKPlots` and `\addBodeTFPlots` macros, this macro can only be used inside a `BodePlot` environment. **It will not function if used in a generic `semilogxaxis` environment or in the `BodeMagPlot` and `BodePhPlot` environments.** The second mandatory argument `system-type` should be either `zpk` or `tf` to specify the system representation. For ZPK systems, the `system-data` has the same format as `\addBodeZPKPlots`. For TF systems, it has the same format as `\addBodeTFPlot`. The optional arguments are the same as the `\addBodeTfPlot` macro, with two additions. Passing `linear` or `asymptotic` options will generate linear or asymptotic approximations if the system representation is ZPK. These two options are ignored for TF systems.

`BodePlot` (*env.*)

```
\begin{BodePlot}[⟨obj1/{⟨opt1⟩},obj2/{⟨opt2⟩},...⟩]
   {⟨min-frequency⟩}{⟨max-frequency⟩}
 \addBodePlot...
\end{BodePlot}
```

Version 2.1 revives the the deprecated `BodePlot` environment exclusively to host multiple instances of the new unified `\addBodePlot` macro. The `\addBodeZPKPlots` and `\addBodeTFPlots` macros **will not function inside this environment.** The purpose is to plot a Bode plot that contains several different transfer functions. The optional argument is the same as the original `\BodeZPK` macro, supporting `plot/`, `axes/`, `group/`, `tikz/` options with `mag` and `ph` sub-types.

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

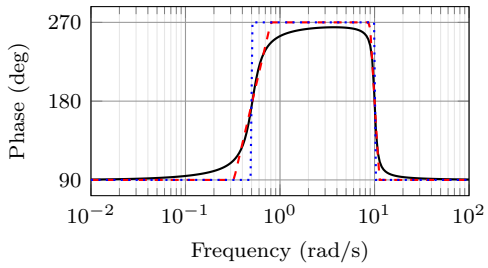


Figure 5: Superimposed approximate and true Bode Phase plot using the `BodePhPlot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

replaced by `Lin` or `Asymp` to generate the true Bode plot, the linear approximation, or the asymptotic approximation, respectively. If the `Feature` is set to `Del`, then `Approx` has to be removed. For example,

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

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

```
\begin{BodePhPlot}[ytick distance=90]{0.01}{100}
  \addBodeComponentPlot[black,thick]{%
    \PhZero{0}{0} + \PhZero{-0.1}{-0.5} + \PhZero{-0.1}{0.5} +
    \PhPole{-0.5}{-10} + \PhPole{-0.5}{10} + \PhK{10}{0}}
  \addBodeComponentPlot[red,dashed,thick] {%
    \PhZeroLin{0}{0} + \PhZeroLin{-0.1}{-0.5} + \PhZeroLin{-0.1}{0.5} +
    \PhPoleLin{-0.5}{-10} + \PhPoleLin{-0.5}{10} + \PhKLin{10}{20}}
  \addBodeComponentPlot[blue,dotted,thick] {%
    \PhZeroAsymp{0}{0} + \PhZeroAsymp{-0.1}{-0.5} + \PhZeroAsymp{-
0.1}{0.5} +
    \PhPoleAsymp{-0.5}{-10} + \PhPoleAsymp{-0.5}{10} + \PhKAsymp{10}{40}}
\end{BodePhPlot}
```

which gives us the plot in Figure 5.

3.1.2 Basic components of the second order

`\TypeSOFeatureApprox` `\TypeSOFeatureApprox{`⟨*a1*⟩`}{`⟨*a0*⟩`}`

This entry describes 12 different macros of the form `\TypeSOFeatureApprox` 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 $G(s) = \frac{1}{s^2+a_1 s+a_0}$ or $G(s) = s^2 + a_1 s + a_0$, 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.

`\MagSOFeaturePeak` `\MagSOFeaturePeak[`⟨*draw-options*⟩`]{`⟨*a1*⟩`}{`⟨*a0*⟩`}`

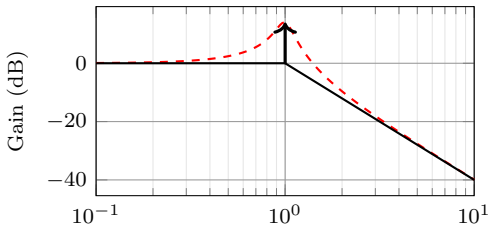


Figure 6: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

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

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

generates the plot in Figure 6.

`\TypeCSFeatureApprox` `\TypeCSFeatureApprox{`⟨*zeta*⟩`}{`⟨*omega-n*⟩`}`

This entry describes 12 different macros of the form `\TypeCSFeatureApprox` that take the damping ratio, 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[`⟨*draw-options*⟩`]{`⟨*zeta*⟩`}{`⟨*omega-n*⟩`}`

This entry describes 2 different macros of the form `\MagCSFeaturePeak` that take the damping ratio, ζ, and the natural frequency, ω_n of a canonical second order system as inputs, and draw a resonant peak using the `\draw` TikZ macro. The `Feature` in the macro name should be replaced by either `Poles` or `Zeros` to generate a peak for poles and a valley for zeros, respectively.

`\MagCCFeaturePeak` `\MagCCFeaturePeak[`⟨*draw-options*⟩`]{`⟨*real-part*⟩`}{`⟨*imaginary-part*⟩`}`

This entry describes 2 different macros of the form `\MagCCFeaturePeak` that take the real and imaginary parts of a pair of complex conjugate poles or zeros as inputs, and draw a resonant peak using the `\draw` TikZ macro. The `Feature` in the macro name should be replaced by either `Poles` or `Zeros` to generate a peak for poles and a valley for zeros, respectively.

3.2 Nyquist plots

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

Plots the Nyquist plot of a transfer function given in ZPK format with a thick red + marking the critical point (-1,0). The mandatory arguments are the same as `\BodeZPK`. Since there is only one plot in a Nyquist diagram, the `\typ` specifier in the optional argument tuples is not needed. As such, the supported optional argument tuples are `plot/{opt}`, which passes `{opt}` to `\addplot`, `axes/{opt}`, which passes `{\opt}` to the `axis` environment, and `tikz/{opt}`, which passes `{\opt}` to the `tikzpicture` environment. Asymptotic/linear approximations are

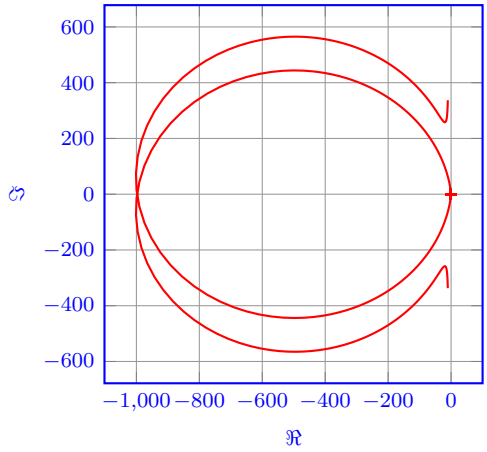


Figure 7: Output of the `\NyquistZPK` macro.

not supported in Nyquist plots. If just `{opt}` is provided as the optional argument, it is interpreted as `plot/{opt}`. Arrows to indicate the direction of increasing ω can be added by adding `\usetikzlibrary{decorations.markings}` and `\usetikzlibrary{arrows.meta}` to the preamble and then passing a tuple of the form

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

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

For example, the command

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

generates the Nyquist plot in Figure 7.

`\NyquistTF`

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

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

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

generates the Nyquist plot in Figure 8.

`NyquistPlot` (*env.*)

```
\begin{NyquistPlot}[⟨obj1/{⟨opt1⟩},obj2/{⟨opt2⟩},...⟩]
  {⟨min-frequency⟩}{⟨max-frequency⟩}
  \addNyquist...
\end{NyquistPlot}
```

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}`:

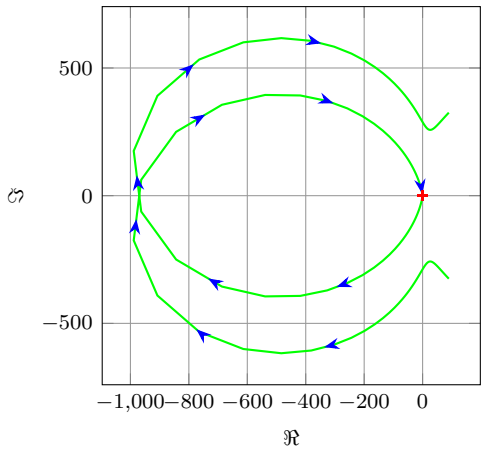


Figure 8: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

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

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

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

`\addNyquistZPKPlot` `\addNyquistZPKPlot[`⟨*plot-options*⟩`]`
`{`⟨*z/{*⟨*zeros*⟩*},p/{*⟨*poles*⟩*},k/{*⟨*gain*⟩*},d/{*⟨*delay*⟩*}*⟩`}`
Generates a twp parametric functions for the magnitude and the phase a transfer function in ZPK form. The generated magnitude and phase parametric functions are converted to real and imaginary part parametric functions and passed to the `\addplot` macro. This macro can be used inside any `axis` environment as long as a domain for the x-axis is supplied through either the `plot-options` interface or directly in the optional argument of the container `axis` environment. Use with the `NyquistPlot` environment supplied with this package is recommended. The mandatory argument is the same as `\BodeZPK`.

`\addNyquistTFPlot` `\addNyquistTFPlot[`⟨*plot-options*⟩`]`
`{`⟨*num/{*⟨*coeffs*⟩*},den/{*⟨*coeffs*⟩*},d/{*⟨*delay*⟩*}*⟩`}`
Similar to `\addNyquistZPKPlot`, with a transfer function input in the TF form.

3.3 Nichols charts

`\NicholsZPK` `\NicholsZPK [`⟨*plot/{*⟨*opt*⟩*},axes/{*⟨*opt*⟩*}*⟩`]`
`{`⟨*z/{*⟨*zeros*⟩*},p/{*⟨*poles*⟩*},k/{*⟨*gain*⟩*},d/{*⟨*delay*⟩*}*⟩`}`
`{`⟨*min-freq*⟩`}{`⟨*max-freq*⟩`}`
Nichols chart of a transfer function given in ZPK format. Same arguments as `\NyquistZPK`.

`\NicholsTF` `\NicholsTF [`⟨*plot/{*⟨*opt*⟩*},axes/{*⟨*opt*⟩*}*⟩`]`
`{`⟨*num/{*⟨*coeffs*⟩*},den/{*⟨*coeffs*⟩*},d/{*⟨*delay*⟩*}*⟩`}`
`{`⟨*min-freq*⟩`}{`⟨*max-freq*⟩`}`
Nichols chart of a transfer function given in TF format. Same arguments as `\NyquistTF`. For example, the command

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

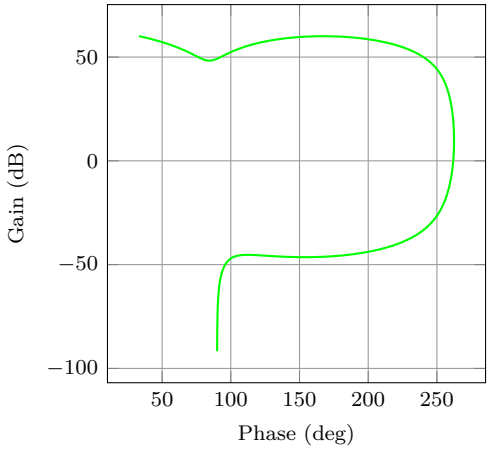


Figure 9: Output of the `\NyquistZPK` macro.

```
{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.
  - `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 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[`⟨*plot-options*⟩`]`
`{`⟨*num/{*⟨*coeffs*⟩*},den/{*⟨*coeffs*⟩*},d/{*⟨*delay*⟩*}*⟩`}`

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

3.4 Pole-zero maps

`\PoleZeroMapZPK` `\PoleZeroMapZPK [`⟨*plot/{*⟨*opt*⟩*},axes/{*⟨*opt*⟩*},scale/{*⟨*log*⟩*}*⟩`]`
`{`⟨*z/{*⟨*zeros*⟩*},p/{*⟨*poles*⟩*},k/{*⟨*gain*⟩*}*⟩`}`

Plots the pole-zero map of a transfer function given in ZPK format, similar to MATLAB's `pzmap` function. The poles are marked with red 'x' symbols and the zeros are marked with blue 'o' symbols. The mandatory argument contains the same ZPK specification as `\BodeZPK`, but the delay parameter `d` is ignored since delays do not affect pole-zero locations. The optional argument supports the same tuples as `\NyquistZPK`: `plot/{opt}` passes options to `\addplot`, `axes/{opt}` passes options to the `axis` environment, `scale/{opt}` sets the scaling type, and `tikz/{opt}` passes options to the `tikzpicture` environment. If just `{opt}` is provided, it is interpreted as `axes/{opt}`.

By default, the axes use linear scaling with unequal scaling to better distribute poles and zeros across the available plot area. This provides improved readability when poles and zeros have different ranges in real and imaginary parts. To force equal axes (square aspect ratio), use `axes/{axis equal}`.

The `scale/{opt}` option enables symmetric logarithmic (symlog) scaling for both axes. This is particularly useful for systems with poles and zeros spanning multiple decades. The symlog scaling preserves the sign of coordinates while applying logarithmic scaling to their magnitude, allowing visualization of both positive and negative values on the same plot.

For example, the command

```
\PoleZeroMapZPK[axes/{grid=major,xlabel={Real Part},ylabel={Imaginary Part}}]
  {z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{{-0.5,-10},{-0.5,10}},k/10}
```

generates a standard linear pole-zero map with the zeros at the origin and at $-0.1 \pm 0.5j$, and poles at $-0.5 \pm 10j$.

```
\PoleZeroMapZPK[scale/log]{z/{-1000,-10,-0.1},p/{-100000,-1000,-100,-
1},k/1}
\PoleZeroMapZPK[scale/log]{z/{{-1,2},{-1,-2}},p/{{-10,5},{-10,-5},-
0.01},k/1}
```

The first example creates a symlog pole-zero map with real poles and zeros; the threshold will be automatically set to the smallest real part. The second example includes complex poles and zeros; the threshold will be set to the smallest real part.

4 Implementation

4.1 Initialization

\n@mod
\n@mod@p
\n@mod@n
\n@pow
gnuplot@id
gnuplot@prefix

We start by processing the class options.

```
\newif\if@pgfarg\@pgfargfalse
\DeclareOption{pgf}{
  \@pgfargtrue
}
\newif\if@declutterarg\@declutterargfalse
\DeclareOption{declutter}{
  \@declutterargtrue
}
\newif\if@radarg\@radargfalse
\DeclareOption{rad}{
  \@radargtrue
}
\newif\if@hzarg\@hzargfalse
\DeclareOption{Hz}{
  \@hzargtrue
}
\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.

```
\newcommand{\n@mod}[2]{(#1)-((round((#1)/(#2)))*(#2))}
\newcommand{\n@mod@p}[2]{(#1)-((floor((#1)/(#2)))*(#2))}
\newcommand{\n@mod@n}[2]{(#1)-((floor((#1)/(#2))+1)*(#2))}
\if@pgfarg
  \newcommand{\n@pow}[2]{(#1)^(#2)}
\else
  \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.

```
  \newcounter{gnuplot@id}
  \setcounter{gnuplot@id}{0}
  \if@declutterarg
    \edef\bodeplot@prefix{gnuplot/\jobname}
  \else
    \edef\bodeplot@prefix{\jobname}
  \fi
  \tikzset{
    gnuplot@prefix/.style={
      id=\arabic{gnuplot@id},
      prefix=\bodeplot@prefix
    }
  }
```

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.

```
  \ifwindows\else
    \if@declutterarg
      \immediate\write18{mkdir -p gnuplot}
    \fi
  \fi
\fi
```

\if@babel
\shorthand@list

Check if the `babel` package is loaded and generate a list of shorthands if it is. The code is based on this stackexchange answer.

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

```
  \@ifpackageloaded{babel}{%
    \@babeltrue
    \let\shorthand@list\@empty
    \def\do#1{%
      \begingroup
        \lccode`\~=`#1\relax
        \lowercase{\ifbabelshorthand~{\g@addto@macro\shorthand@list{~}}{}}
      \endgroup
    }
    \dospecials
  }{}
}
```

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

```
\pgfplotsset{
  bode@style/.style = {
    label style={font=\footnotesize},
    tick label style={font=\footnotesize},
    grid=both,
    major grid style={color=gray!80},
    minor grid style={color=gray!20},
    x label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
    y label style={at={(ticklabel cs:0.5)},anchor=near ticklabel},
    scale only axis,
    samples=200,
    width=5cm,
    log basis x=10
  }
}
```

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

```
\pgfplotsset{freq@filter/.style = {}}
\def\freq@scale{1}
\pgfplotsset{freq@label/.style = {xlabel = {Frequency (rad/s)}}}
\pgfplotsset{ph@x@label/.style = {xlabel={Phase (deg)}}}
\pgfplotsset{ph@y@label/.style = {ylabel={Phase (deg)}}}
\def\ph@scale{180/pi}
\if@radarg
  \pgfplotsset{ph@y@label/.style = {ylabel={Phase (rad)}}}
  \pgfplotsset{ph@x@label/.style = {xlabel={Phase (rad)}}}
  \def\ph@scale{1}
\fi
\if@hzarg
  \def\freq@scale{2*pi}
  \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}
  \if@pgfarg
    \pgfplotsset{freq@filter/.style = {x filter/.expression={x-
  log10(2*pi)}}}
  \fi
\fi
\tikzset{
  phase unit/.initial={deg},
  phase unit/.default={deg},
  phase unit/.is choice,
  phase unit/deg/.code={
    \renewcommand{\ph@scale}{180/pi}
    \pgfplotsset{ph@x@label/.style = {xlabel={Phase (deg)}}}
    \pgfplotsset{ph@y@label/.style = {ylabel={Phase (deg)}}}
  },
  phase unit/rad/.code={
    \renewcommand{\ph@scale}{1}
```

```
    \pgfplotsset{ph@y@label/.style = {ylabel={Phase (rad)}}}
    \pgfplotsset{ph@x@label/.style = {xlabel={Phase (rad)}}}
  },
  frequency unit/.initial={rad},
  frequency unit/.default={rad},
  frequency unit/.is choice,
  frequency unit/Hz/.code={
    \renewcommand{\freq@scale}{2*pi}
    \pgfplotsset{freq@label/.style = {xlabel = {Frequency (Hz)}}}
    \if@pgfarg
      \pgfplotsset{freq@filter/.style = {x filter/.expression={x-
  log10(2*pi)}}}
    \fi
  },
  frequency unit/rad/.code={
    \renewcommand{\freq@scale}{1}
    \pgfplotsset{freq@label/.style = {xlabel = {Frequency (rad/s)}}}
  }
}
```

get@interval@start
get@interval@end

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

```
\def\get@interval@start#1:#2\@nil{#1}
\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 frequeny 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
\MagKAsymp
\MagKLin
\PhK
\PhKAsymp
\PhKLin

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

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

\PhKAsymp
\PhKLin

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.

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

\MagPole
\MagPoleAsymp
\MagPoleLin
\PhPole
\PhPoleAsymp
\PhPoleLin

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.

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

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

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

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

```
\newcommand*{\MagPoleAsymp}{\MagPoleLin}
```

Parametric function for the true phase of a complex pole.

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

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

```
\newcommand*{\PhPoleLin}[2]{
  ((abs(#1)+abs(#2) == 0 ? -pi/2 :
  (t < (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
    (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2}))})) ?
  (-atan2(-(#2),-(#1))) :
  (t >= (sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) *
    (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} + \n@pow{#2}{2}))})) ?
  (#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) :
  (-atan2(-(#2),-(#1)) + (log10(t/(sqrt(\n@pow{#1}{2} + \n@pow{#2}{2}) /
    (\n@pow{10}{sqrt(\n@pow{#1}{2}/(\n@pow{#1}{2} +
    \n@pow{#2}{2}))})))))*((#2>0?(#1>0?3*pi/2:-pi/2):-pi/2) + atan2(-
  (#2),-(#1)))/
    (log10(\n@pow{10}{sqrt((4*\n@pow{#1}{2})/
    (\n@pow{#1}{2} + \n@pow{#2}{2}))})))))))*\ph@scale)}
```

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

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

\MagZero
\MagZeroAsymp
\MagZeroLin
\PhZero
\PhZeroAsymp
\PhZeroLin

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).

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

4.3 Second order systems.

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

\MagCSPoles
\MagCSPolesAsymp
\MagCSPolesLin
\PhCSPoles
\PhCSPolesAsymp
\PhCSPolesLin
\MagCSZeros
\MagCSZerosAsymp
\MagCSZerosLin
\PhCSZeros
\PhCSZerosAsymp
\PhCSZerosLin

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

```
\newcommand*{\MagCSPoles}[2]{(-20*log10(sqrt(\n@pow{\n@pow{#2}{2}
    - \n@pow{t}{2}}{2} + \n@pow{2*#1*#2*t}{2})))}
\newcommand*{\MagCSPolesLin}[2]{(t < #2 ? -40*log10(#2) : -
  40*log10(t))}
\newcommand*{\MagCSPolesAsymp}{\MagCSPolesLin}
```

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

```
\newcommand*{\PhCSPoles}[2]{((-atan2((2*(#1)*(#2)*t),(\n@pow{#2}{2}
  - \n@pow{t}{2})))*\ph@scale)}
\newcommand*{\PhCSPolesLin}[2]{((t < (#2 / (\n@pow{10}{abs(#1)}))) ?
  0 :
  (t >= (#2 * (\n@pow{10}{abs(#1)}))) ?
  (#1>0 ? -pi : pi) :
  (#1>0 ? (-pi*(log10(t*(\n@pow{10}{#1})/#2))/(2*#1)) :
    (pi*(log10(t*(\n@pow{10}{abs(#1)})/#2))/(2*abs(#1))))))*\ph@scale)}
\newcommand*{\PhCSPolesAsymp}[2]{((#1>0?(t<#2?0:-
  pi):(t<#2?0:pi))*\ph@scale)}
```

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

```
\newcommand*{\MagCSZeros}{0-\MagCSPoles}
\newcommand*{\MagCSZerosLin}{0-\MagCSPolesLin}
\newcommand*{\MagCSZerosAsymp}{0-\MagCSPolesAsymp}
\newcommand*{\PhCSZeros}{0-\PhCSPoles}
\newcommand*{\PhCSZerosLin}{0-\PhCSPolesLin}
\newcommand*{\PhCSZerosAsymp}{0-\PhCSPolesAsymp}
```

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

```
\newcommand*{\MagCSPolesPeak}[3][]{
  \draw[#1,->] (axis cs:{#3},{-40*log10(#3)}) --
  (axis cs:{#3},{-40*log10(#3)-20*log10(2*abs(#2))})
}
\newcommand*{\MagCSZerosPeak}[3][]{
  \draw[#1,->] (axis cs:{#3},{40*log10(#3)}) --
  (axis cs:{#3},{40*log10(#3)+20*log10(2*abs(#2))})
}
```

`\MagSOPoles` `\MagSOPolesAsymp` `\MagSOPolesLin` `\PhSOPoles` `\PhSOPolesAsymp` `\PhSOPolesLin` `\MagSOZeros` `\MagSOZerosAsymp` `\MagSOZerosLin` `\PhSOZeros` `\PhSOZerosAsymp` `\PhSOZerosLin` 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.

```
\newcommand*{\MagSOPoles}[2]{
  (-20*log10(sqrt(\n@pow{#2 - \n@pow{t}{2}}{2} + \n@pow{#1*t}{2})))}
\newcommand*{\MagSOPolesLin}[2]{
  (t < sqrt(abs(#2)) ? -20*log10(abs(#2)) : - 40*log10(t))}
\newcommand*{\MagSOPolesAsymp}{\MagSOPolesLin}
```

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

```
\newcommand*{\PhSOPoles}[2]{((-atan2((#1)*t,((#2) -
  \n@pow{t}{2})))*\ph@scale)}
\newcommand*{\PhSOPolesLin}[2]{((#2>0 ?
  \PhCSPolesLin{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
  (#1>0 ? -pi : pi)))}
\newcommand*{\PhSOPolesAsymp}[2]{((#2>0 ?
  \PhCSPolesAsymp{(#1/(2*sqrt(#2)))}{(sqrt(#2))} :
  (#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).

```
\newcommand*{\MagSOZeros}{0-\MagSOPoles}
\newcommand*{\MagSOZerosLin}{0-\MagSOPolesLin}
\newcommand*{\MagSOZerosAsymp}{0-\MagSOPolesAsymp}
\newcommand*{\PhSOZeros}{0-\PhSOPoles}
\newcommand*{\PhSOZerosLin}{0-\PhSOPolesLin}
\newcommand*{\PhSOZerosAsymp}{0-\PhSOPolesAsymp}
```

`\MagSOPolesPeak` `\MagSOZerosPeak` 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.

```
\newcommand*{\MagSOPolesPeak}[3][]{
  \draw[#1,->] (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3))}) --
  (axis cs:{sqrt(abs(#3))},{-20*log10(abs(#3)) -
    20*log10(abs(#2/sqrt(abs(#3))))});
}
\newcommand*{\MagSOZerosPeak}[3][]{
  \draw[#1,->] (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3))}) --
  (axis cs:{sqrt(abs(#3))},{20*log10(abs(#3)) +
    20*log10(abs(#2/sqrt(abs(#3))))});
}
```

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.

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

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

```
  \parse@opt{#1}
  \gdef\func@mag{}
  \gdef\func@ph{}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
  \build@ZPK@plot{\func@mag}{\func@ph}{\opt@approx}{#2}
    \edef\temp@cmd{\noexpand\begin{groupplot}[
      bode@style,
      xmin=#3,
      xmax=#4,
      domain=#3*\freq@scale:#4*\freq@scale,
      height=2.5cm,
      xmode=log,
      group style = {group size = 1 by 2,vertical sep=0.25cm},
      \opt@group
    ]}
    \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.

```
      \edef\temp@mag@cmd{\noexpand\nextgroupplot [ylabel={Gain (dB)}, xmajorticks=false, \optmag@axes]
      \noexpand\addplot [freq@filter, variable=t, thick, \optmag@plot]}
      \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
      \noexpand\addplot [freq@filter, variable=t, thick, trig format plots=rad, \optph@plot]}
      \if@pgfarg
        \temp@mag@cmd {\func@mag};
        \optmag@commands
        \temp@ph@cmd {\func@ph};
        \optph@commands
      \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.

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

```
          set xrange [#3*\freq@scale:#4*\freq@scale];
          set samples \pgfkeysvalueof{/pgfplots/samples};
          plot \func@mag;
          set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
          plot "$meta" using ($1/(\freq@scale)):($2);
        };
        \optmag@commands
        \stepcounter{gnuplot@id}
        \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
        { set table $meta;
          set dummy t;
          set logscale x 10;
          set xrange [#3*\freq@scale:#4*\freq@scale];
          set samples \pgfkeysvalueof{/pgfplots/samples};
          plot \func@ph;
          set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
          plot "$meta" using ($1/(\freq@scale)):($2);
        };
        \optph@commands
      \fi
    \end{groupplot}
  \end{tikzpicture}
}
```

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

```
\AtBeginDocument{%
  \if@babel
  \let\Orig@BodeZPK\BodeZPK
  \renewcommand{\BodeZPK}{%
    \expandafter\shorthandoff\expandafter{\shorthand@list}
    \BodeZPK@Shorthandoff
  }
  \newcommand{\BodeZPK@Shorthandoff}[4][]{%
    \Orig@BodeZPK[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\shorthand@list}
  }
  \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.

```
\newcommand{\BodeTF}[4][]{
  \parse@opt{#1}
  \gdef\func@mag{}
  \gdef\func@ph{}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
  \build@TF@plot{\func@mag}{\func@ph}{#2}
    \edef\temp@cmd{\noexpand\begin{groupplot}[
      bode@style,
      xmin=#3,
      xmax=#4,
      domain=#3*\freq@scale:#4*\freq@scale,
      height=2.5cm,
      xmode=log,
      group style = {group size = 1 by 2,vertical sep=0.25cm},
      \opt@group
    ]}
    \temp@cmd
      \edef\temp@mag@cmd{\noexpand\nextgroupplot [ylabel={Gain (dB)}, xmajorticks=false, \optmag@axes]
```

```
      \noexpand\addplot [freq@filter, variable=t, thick, \opt-
mag@plot]}
      \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes
      \noexpand\addplot [freq@filter, variable=t, thick, trig for-
mat plots=rad, \optph@plot]}
      \if@pgfarg
        \temp@mag@cmd {\func@mag};
        \optmag@commands
        \temp@ph@cmd {\n@mod{\func@ph}{2*pi*\ph@scale}};
        \optph@commands
      \else
        \stepcounter{gnuplot@id}
        \temp@mag@cmd gnuplot [raw gnuplot, gnuplot@prefix]
        { set table $meta;
          set dummy t;
          set logscale x 10;
          set xrange [#3*\freq@scale:#4*\freq@scale];
          set samples \pgfkeysvalueof{/pgfplots/samples};
          plot \func@mag;
          set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
          plot "$meta" using ($1/(\freq@scale)):($2);
        };
        \optmag@commands
        \stepcounter{gnuplot@id}
        \temp@ph@cmd gnuplot [raw gnuplot, gnuplot@prefix]
        { set table $meta;
          set dummy t;
          set logscale x 10;
          set trange [#3*\freq@scale:#4*\freq@scale];
          set samples \pgfkeysvalueof{/pgfplots/samples};
          plot '+' using (t) : ((\func@ph)/(\ph@scale)) smooth unwrap;
          set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
          plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
        };
        \optph@commands
      \fi
    \end{groupplot}
  \end{tikzpicture}
}
```

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

```
\AtBeginDocument{
  \if@babel
  \let\Orig@BodeTF\BodeTF
  \renewcommand{\BodeTF}{%
    \expandafter\shorthandoff\expandafter{\shorthand@list}
    \BodeTF@Shorthandoff
  }
  \newcommand{\BodeTF@Shorthandoff}[4][]{%
    \Orig@BodeTF[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\shorthand@list}
  }
  \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.

```
\newcommand{\addBodeZPKPlots}[3][true/{}]{
  \foreach \approx/\opt in {#1} {
    \gdef\plot@macro{}
    \gdef\temp@macro{}
    \ifnum\pdf@strcmp{#2}{phase}=0
      \build@ZPK@plot{\temp@macro}{\plot@macro}{\approx}{#3}
    \else
```

```
      \build@ZPK@plot{\plot@macro}{\temp@macro}{\approx}{#3}
    \fi
    \if@pgfarg
      \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, thick, trig format plots=rad, \opt]}
      \temp@cmd {\plot@macro};
    \else
      \stepcounter{gnuplot@id}
      \edef\temp@cmd{\noexpand\addplot [variable=t, thick, \opt]}
      \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
      { set table $meta;
        set dummy t;
        set logscale x 10;
        set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
        set samples \pgfkeysvalueof{/pgfplots/samples};
        plot \plot@macro;
        set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
        plot "$meta" using ($1/(\freq@scale)):($2);
      };
    \fi
  }
}
```

`\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.

```
\newcommand{\addBodeTFPlot}[3][thick]{
  \gdef\plot@macro{}
  \gdef\temp@macro{}
  \ifnum\pdf@strcmp{#2}{phase}=0
    \build@TF@plot{\temp@macro}{\plot@macro}{#3}
  \else
    \build@TF@plot{\plot@macro}{\temp@macro}{#3}
  \fi
  \if@pgfarg
    \ifnum\pdf@strcmp{#2}{phase}=0
      \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, trig format plots=rad, #1]}
      \temp@cmd {\n@mod{\plot@macro}{2*pi}};
    \else
      \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
able=t, #1]}
      \temp@cmd {\plot@macro};
    \fi
  \else
    \stepcounter{gnuplot@id}
    \ifnum\pdf@strcmp{#2}{phase}=0
      \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
      { set table $meta;
        set dummy t;
        set logscale x 10;
        set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
        set samples \pgfkeysvalueof{/pgfplots/samples};
        plot '+' using (t) : ((\plot@macro)/(\ph@scale)) smooth un-
wrap;
        set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
        plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
      };
    \else
      \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
        { set table $meta;
```

```
          set dummy t;
          set logscale x 10;
          set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
          set samples \pgfkeysvalueof{/pgfplots/samples};
          plot \plot@macro;
          set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
          plot "$meta" using ($1/(\freq@scale)):($2);
        };
    \fi
  \fi
}
```

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

```
\newcommand{\addBodeComponentPlot}[2][thick]{
  \if@pgfarg
    \edef\temp@cmd{\noexpand\addplot [freq@filter, do-
  main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale, vari-
  able=t, trig format plots=rad, #1]}
    \temp@cmd {#2};
  \else
    \stepcounter{gnuplot@id}
    \addplot [variable=t, #1] gnuplot [raw gnuplot, gnuplot@prefix]
    { set table $meta;
      set dummy t;
      set logscale x 10;
      set xrange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
      set samples \pgfkeysvalueof{/pgfplots/samples};
      plot #2;
      set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
      plot "$meta" using ($1/(\freq@scale)):($2);
    };
  \fi
}
```

`\addBodePlot` Unified macro to add Bode plots supporting both ZPK and TF system representations. Automatically generates both magnitude and phase plot commands for later insertion.

```
\newcommand{\addBodePlot}[3][]{
  \parse@add@Bode@opt{#1}
  \gdef\plot@macro{}
  \gdef\temp@macro{}
  \ifnum\pdf@strcmp{#2}{zpk}=0
    \build@ZPK@plot{\plot@macro}{\temp@macro}{\opt@approx}{#3}
  \else
    \ifnum\pdf@strcmp{#2}{tf}=0
      \build@TF@plot{\plot@macro}{\temp@macro}{#3}
    \else
      \PackageError {bodeplot} {Unknown system representation ‘#2’.}
      {Supported representations are ‘zpk’ and ‘tf’.}
    \fi
  \fi
  \if@pgfarg
    \xdef\plot@a@cmd{\unexpanded\expandafter{\plot@a@cmd}\noexpand\addplot [freq@filt
  main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}, vari-
  able=t, thick, trig format plots=rad, \opt@plot] {\plot@macro};}
    \xdef\plot@b@cmd{\unexpanded\expandafter{\plot@b@cmd}\noexpand\addplot [freq@filt
  main=\freq@scale*\pgfkeysvalueof{/pgfplots/domain}, vari-
  able=t, thick, trig format plots=rad, \opt@plot] {\temp@macro};}
  \else
    \stepcounter{gnuplot@id}
    \edef\gnu@id{\arabic{gnuplot@id}}
    \build@gnu@plot{\plot@macro}{\gnu@id}
```

```
    \xdef\plot@a@cmd{\unexpanded\expandafter{\plot@a@cmd}\noexpand\addplot [variable=t, thick, \opt@plot]\gnu@cmd}
    \stepcounter{gnuplot@id}
    \edef\gnu@id{\arabic{gnuplot@id}}
    \ifnum\pdf@strcmp{#2}{zpk}=0
      \build@gnu@plot{\temp@macro}{\gnu@id}
      \else
        \ifnum\pdf@strcmp{#2}{tf}=0
          \build@gnu@unwrap@plot{\temp@macro}{\gnu@id}
        \else
          \PackageError {bodeplot} {Unknown system representation '#2'.}
          {Supported representations are 'zpk' and 'tf'.}
        \fi
      \fi
    \xdef\plot@b@cmd{\unexpanded\expandafter{\plot@b@cmd}\noexpand\addplot [variable=t, thick, \opt@plot]\gnu@cmd}
  \fi
}
```

`BodePhPlot` (*env.*) An environment to host phase plot macros that pass parametric functions to `\addplot` macros. Uses the defaults specified in `bode@style` to create a shortcut that includes the `tikzpicture` and `semilogaxis` environments. The body of the environment is grabbed as a macro to maintain compatibility with externalization in `tikz`.

```
\AtBeginDocument{%
  \if@babel
    \AddToHook{env/BodePhPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthand
    \AddToHook{env/BodePhPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@li
  \fi
}
\NewDocumentEnvironment{BodePhPlot}{O{}mm+b}{
  \parse@env@opt{#1}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
    \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
      ph@y@label,
      freq@label,
      bode@style,
      xmin={#2},
      xmax={#3},
      domain=#2:#3,
      height=2.5cm,
      \unexpanded\expandafter{\opt@axes}
    ]}
    \temp@cmd
      #4
    \end{semilogxaxis}
  \end{tikzpicture}
}{}
```

`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.

```
\AtBeginDocument{%
  \if@babel
    \AddToHook{env/BodeMagPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthan
    \AddToHook{env/BodeMagPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@l
  \fi
}
\NewDocumentEnvironment{BodeMagPlot}{O{}mm+b}{
  \parse@env@opt{#1}
```

```
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
    \edef\temp@cmd{\noexpand\begin{semilogxaxis}[
      bode@style,
      freq@label,
      xmin={#2},
      xmax={#3},
      domain=#2:#3,
      height=2.5cm,
      ylabel={Gain (dB)},
      \unexpanded\expandafter{\opt@axes}
    ]}
    \temp@cmd
      #4
    \end{semilogxaxis}
  \end{tikzpicture}
}{}
```

`BodePlot` (*env.*) Enhanced version of the `BodePlot` environment that works with the unified `\addBodePlot` macro. Creates a grouped plot with magnitude on top and phase on bottom, automatically collecting and inserting plot commands generated by `\addBodePlot` calls within the environment body.

```
\NewDocumentEnvironment{BodePlot}{O{}mm+b}{
  \gdef\plot@a@cmd{}
  \gdef\plot@b@cmd{}
  \parse@opt{#1}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
    \edef\temp@cmd{\noexpand\begin{groupplot}[
      bode@style,
      xmin=#2,
      xmax=#3,
      domain=#2*\freq@scale:#3*\freq@scale,
      height=2.5cm,
      xmode=log,
      group style = {group size = 1 by 2,vertical sep=0.25cm},
      \opt@group
    ]}
    \temp@cmd
      #4
      \edef\temp@mag@cmd{\noexpand\nextgroupplot [ylabel={Gain (dB)}, xmajorticks=false, \optmag@axes]}
      \edef\temp@ph@cmd{\noexpand\nextgroupplot [ph@y@label, freq@label, \optph@axes]
      \temp@mag@cmd
      \plot@a@cmd
      \temp@ph@cmd
      \plot@b@cmd
    \end{groupplot}
  \end{tikzpicture}
}{}
```

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.

```
\newcommand*{\add@feature}[3]{
  \ifcat$\detokenize\expandafter{#1}$
```

```
    \xdef#1{\unexpanded\expandafter{#1 0+#2}}
  \else
    \xdef#1{\unexpanded\expandafter{#1+#2}}
  \fi
  \foreach \y [count=\n] in #3 {
    \xdef#1{\unexpanded\expandafter{#1}{\y}}
    \xdef\Last@LoopValue{\n}
  }
  \ifnum\Last@LoopValue=1
    \xdef#1{\unexpanded\expandafter{#1}{0}}
  \fi
}
```

`\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.

```
\newcommand{\build@ZPK@plot}[4]{
  \foreach \feature/\values in {#4} {
    \ifnum\pdf@strcmp{\feature}{z}=0
      \foreach \z in \values {
        \ifnum\pdf@strcmp{#3}{linear}=0
          \add@feature{#2}{\PhZeroLin}{\z}
          \add@feature{#1}{\MagZeroLin}{\z}
        \else
          \ifnum\pdf@strcmp{#3}{asymptotic}=0
            \add@feature{#2}{\PhZeroAsymp}{\z}
            \add@feature{#1}{\MagZeroAsymp}{\z}
          \else
            \add@feature{#2}{\PhZero}{\z}
            \add@feature{#1}{\MagZero}{\z}
          \fi
        \fi
      }
    \fi
    \ifnum\pdf@strcmp{\feature}{p}=0
      \foreach \p in \values {
        \ifnum\pdf@strcmp{#3}{linear}=0
          \add@feature{#2}{\PhPoleLin}{\p}
          \add@feature{#1}{\MagPoleLin}{\p}
        \else
          \ifnum\pdf@strcmp{#3}{asymptotic}=0
            \add@feature{#2}{\PhPoleAsymp}{\p}
            \add@feature{#1}{\MagPoleAsymp}{\p}
          \else
            \add@feature{#2}{\PhPole}{\p}
            \add@feature{#1}{\MagPole}{\p}
          \fi
        \fi
      }
    \fi
    \ifnum\pdf@strcmp{\feature}{k}=0
      \ifnum\pdf@strcmp{#3}{linear}=0
        \add@feature{#2}{\PhKLin}{\values}
        \add@feature{#1}{\MagKLin}{\values}
      \else
        \ifnum\pdf@strcmp{#3}{asymptotic}=0
          \add@feature{#2}{\PhKAsymp}{\values}
          \add@feature{#1}{\MagKAsymp}{\values}
```

```
        \else
          \add@feature{#2}{\PhK}{\values}
          \add@feature{#1}{\MagK}{\values}
        \fi
      \fi
    \fi
    \ifnum\pdf@strcmp{\feature}{d}=0
      \ifnum\pdf@strcmp{#3}{linear}=0
        \PackageError {bodeplot} {Linear approximation for pure delays is not
        supported.} {Plot the true Bode plot using 'true' instead of 'linear'.}
      \else
        \ifnum\pdf@strcmp{#3}{asymptotic}=0
          \PackageError {bodeplot} {Asymptotic approximation for pure delays is not
          supported.} {Plot the true Bode plot using 'true' instead of 'asymptotic'.}
        \else
          \ifdim\values pt < 0pt
            \PackageError {bodeplot} {Delay needs to be a positive number.}
          \fi
          \add@feature{#2}{\PhDel}{\values}
          \add@feature{#1}{\MagDel}{\values}
        \fi
      \fi
    \fi
  }
}
```

`\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).

```
\newcommand{\build@TF@plot}[3]{
  \gdef\num@real{0}
  \gdef\num@im{0}
  \gdef\den@real{0}
  \gdef\den@im{0}
  \gdef\loop@delay{0}
  \foreach \feature/\values in {#3} {
    \ifnum\pdf@strcmp{\feature}{num}=0
      \foreach \numcoeff [count=\numpow] in \values {
        \xdef\num@degree{\numpow}
      }
      \foreach \numcoeff [count=\numpow] in \values {
        \pgfmathtruncatemacro{\currentdegree}{\num@degree-\numpow}
        \ifnum\currentdegree = 0
          \xdef\num@real{\num@real+\numcoeff}
        \else
          \ifodd\currentdegree
            \xdef\num@im{\num@im+(\numcoeff*(\n@pow{-1}{(\currentdegree-1)/2})*%
              (\n@pow{t}{\currentdegree}))}
          \else
            \xdef\num@real{\num@real+(\numcoeff*(\n@pow{-1}{(\currentdegree)/2})*%
              (\n@pow{t}{\currentdegree}))}
          \fi
        \fi
      }
    \fi
```

```
    \ifnum\pdf@strcmp{\feature}{den}=0
      \foreach \dencoeff [count=\denpow] in \values {
        \xdef\den@degree{\denpow}
      }
      \foreach \dencoeff [count=\denpow] in \values {
        \pgfmathtruncatemacro{\currentdegree}{\den@degree-\denpow}
        \ifnum\currentdegree = 0
          \xdef\den@real{\den@real+\dencoeff}
        \else
          \ifodd\currentdegree
            \xdef\den@im{\den@im+(\dencoeff*(\n@pow{-
1}{(\currentdegree-1)/2})*%
              (\n@pow{t}{\currentdegree}))}
          \else
            \xdef\den@real{\den@real+(\dencoeff*(\n@pow{-
1}{(\currentdegree)/2})*%
              (\n@pow{t}{\currentdegree}))}
          \fi
        \fi
      }
    \fi
    \ifnum\pdf@strcmp{\feature}{d}=0
      \xdef\loop@delay{\values}
    \fi
  }
  \xdef#2{((atan2((\num@im),(\num@real))-atan2((\den@im),%
    (\den@real))-\loop@delay*t)*(\ph@scale))}
  \xdef#1{(20*log10(sqrt((\n@pow{\num@real}{2})+(\n@pow{\num@im}{2})))-
%
    20*log10(sqrt((\n@pow{\den@real}{2})+(\n@pow{\den@im}{2}))))}
}
```

`\parse@opt` Parses options supplied to the main Bode macros. A `for` loop over tuples of the form `\obj/\typ/\opt` with a long list of nested if-else statements does the job. If the input `\obj` is `plot`, `axes`, `group`, `approx`, or `tikz` the corresponding `\opt` are passed, unexpanded, to the `\addplot` macro, the `\nextgroupplot` macro, the `groupplot` environment, the `\build@ZPK@plot` macro, and the `tikzpicture` environment, respectively. If `\obj` is `commands`, the corresponding `\opt` are stored, unexpanded, in the macros `\optph@commands` and `\optmag@commands`, to be executed in appropriate `axis` environments.

```
\newcommand{\parse@opt}[1]{
  \gdef\optmag@axes{}
  \gdef\optph@axes{}
  \gdef\optph@plot{}
  \gdef\optmag@plot{}
  \gdef\opt@group{}
  \gdef\opt@approx{}
  \gdef\optph@commands{}
  \gdef\optmag@commands{}
  \gdef\opt@tikz{}
  \foreach \obj/\typ/\opt in {#1} {
    \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
        \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
      \else
        \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
          \xdef\optph@plot{\unexpanded\expandafter{\opt}}
        \else
          \xdef\optmag@plot{\unexpanded\expandafter{\opt}}
          \xdef\optph@plot{\unexpanded\expandafter{\opt}}
        \fi
      \fi
    \else
```

```
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
        \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{mag}=0
          \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
        \else
          \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
            \xdef\optph@axes{\unexpanded\expandafter{\opt}}
          \else
            \xdef\optmag@axes{\unexpanded\expandafter{\opt}}
            \xdef\optph@axes{\unexpanded\expandafter{\opt}}
          \fi
        \fi
      \else
        \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{group}=0
          \xdef\opt@group{\unexpanded\expandafter{\opt}}
        \else
          \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{approx}=0
            \xdef\opt@approx{\unexpanded\expandafter{\opt}}
          \else
            \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
              \ifnum\pdf@strcmp{\unexpanded\expandafter{\typ}}{ph}=0
                \xdef\optph@commands{\unexpanded\expandafter{\opt}}
              \else
                \xdef\optmag@commands{\unexpanded\expandafter{\opt}}
              \fi
            \else
              \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
                \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
              \else
                \xdef\optmag@plot{\unexpanded\expandafter{\optmag@plot},
                  \unexpanded\expandafter{\obj}}
                \xdef\optph@plot{\unexpanded\expandafter{\optph@plot},
                  \unexpanded\expandafter{\obj}}
              \fi
            \fi
          \fi
        \fi
      \fi
    \fi
  }
}
```

`\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.

```
\newcommand{\parse@env@opt}[1]{
  \gdef\opt@axes{}
  \gdef\opt@tikz{}
  \foreach \obj/\opt in {#1} {
    \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
      \xdef\opt@axes{\unexpanded\expandafter{\opt}}
    \else
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
        \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
      \else
        \xdef\opt@axes{\unexpanded\expandafter{\opt@axes},
          \unexpanded\expandafter{\obj}}
      \fi
    \fi
  }
}
```

\parse@add@Bode@opt Parses options for the unified \addBodePlot macro. Handles linear and asymptotic approximation options plus general plot styling options.

```
\newcommand{\parse@add@Bode@opt}[1]{
  \gdef\opt@plot{}
  \gdef\opt@approx{}
  \foreach \obj/\opt in {#1} {
    \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{linear}=0
      \xdef\opt@approx{\unexpanded\expandafter{\opt}}
    \else
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{asymptotic}=0
        \xdef\opt@approx{\unexpanded\expandafter{\opt}}
      \else
        \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
          \unexpanded\expandafter{\obj}}
      \fi
    \fi
  }
}
```

\build@gnu@plot Helper macro to assemble a general gnuplot invocation for parametric plots.

```
\newcommand{\build@gnu@plot}[2]{
  \xdef\gnu@cmd{ gnuplot [raw gnuplot, id=#2, prefix=\bodeplot@prefix]
    { set table $meta;
      set dummy t;
      set logscale x 10;
      set xrange [\pgfkeysvalueof{/pgfplots/domain}];
      set samples \pgfkeysvalueof{/pgfplots/samples};
      plot #1;
      set table "\bodeplot@prefix#2.table";
      plot "$meta" using ($1/(\freq@scale)):($2);
    };}
}
```

\build@gnu@unwrap@plot Helper macro to assemble a gnuplot invocation with phase unwrapping for TF phase plots.

```
\newcommand{\build@gnu@unwrap@plot}[2]{
  \xdef\gnu@cmd{ gnuplot [raw gnuplot, id=#2, prefix=\bodeplot@prefix]
    { set table $meta;
      set dummy t;
      set logscale x 10;
      set trange [\pgfkeysvalueof{/pgfplots/domain}];
      set samples \pgfkeysvalueof{/pgfplots/samples};
      plot '+' using (t) : ((#1)/(\ph@scale)) smooth unwrap;
      set table "\bodeplot@prefix#2.table";
      plot "$meta" using ($1/(\freq@scale)):($2*\ph@scale);
    };}
}
```

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!

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

```
  \parse@N@opt{#1}
  \gdef\func@mag{}
  \gdef\func@ph{}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
  \build@ZPK@plot{\func@mag}{\func@ph}{}{#2}
    \edef\temp@cmd{\noexpand\begin{axis}[
      bode@style,
      domain=#3*\freq@scale:#4*\freq@scale,
      height=5cm,
      xlabel={$\Re$},
      ylabel={$\Im$},
      samples=500,
      \unexpanded\expandafter{\opt@axes}
    ]}
    \temp@cmd
      \addplot [only marks,mark=+,thick,red] (-1 , 0);
      \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig format plots=rad, \unexpanded\expandafter{\opt@plot}]}
      \if@pgfarg
        \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
          {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
        \opt@commands
      \else
        \stepcounter{gnuplot@id}
        \temp@cmd gnuplot [parametric, gnuplot@prefix] {
          \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
          \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
        };
        \opt@commands
      \fi
    \end{axis}
  \end{tikzpicture}
}
```

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

```
\AtBeginDocument{%
  \if@babel
  \let\Orig@NyquistZPK\NyquistZPK
  \renewcommand{\NyquistZPK}{%
    \expandafter\shorthandoff\expandafter{\shorthand@list}
    \NyquistZPK@Shorthandoff
  }
  \newcommand{\NyquistZPK@Shorthandoff}[4][]{%
    \Orig@NyquistZPK[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\shorthand@list}
  }
  \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`.

```
\newcommand{\NyquistTF}[4][]{
  \parse@N@opt{#1}
  \gdef\func@mag{}
  \gdef\func@ph{}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
    \build@TF@plot{\func@mag}{\func@ph}{#2}
    \edef\temp@cmd{\noexpand\begin{axis}[
      bode@style,
```

```
      domain=#3*\freq@scale:#4*\freq@scale,
      height=5cm,
      xlabel={$\Re$},
      ylabel={$\Im$},
      samples=500,
      \unexpanded\expandafter{\opt@axes}
    ]}
    \temp@cmd
      \addplot [only marks, mark=+, thick, red] (-1 , 0);
      \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig for-
mat plots=rad, \unexpanded\expandafter{\opt@plot}]}
      \if@pgfarg
        \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
          {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
        \opt@commands
      \else
        \stepcounter{gnuplot@id}
        \temp@cmd gnuplot [parametric, gnuplot@prefix] {
          \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
          \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
        };
        \opt@commands
      \fi
    \end{axis}
  \end{tikzpicture}
}
```

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

```
\AtBeginDocument{%
  \if@babel
  \let\Orig@NyquistTF\NyquistTF
  \renewcommand{\NyquistTF}{%
    \expandafter\shorthandoff\expandafter{\shorthand@list}
    \NyquistTF@Shorthandoff
  }
  \newcommand{\NyquistTF@Shorthandoff}[4][]{%
    \Orig@NyquistTF[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\shorthand@list}
  }
  \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.

```
\newcommand{\addNyquistZPKPlot}[2][]{
  \gdef\func@mag{}
  \gdef\func@ph{}
  \build@ZPK@plot{\func@mag}{\func@ph}{}{#2}
  \if@pgfarg
    \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do
mable=t, trig format plots=rad, #1]}
    \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
      {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
  \else
    \stepcounter{gnuplot@id}
    \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do
mable=t, #1]}
    \temp@cmd gnuplot [parametric, gnuplot@prefix] {
      \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
      \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
    };
  \fi
```

```
}
```

`\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.

```
\newcommand{\addNyquistTFPlot}[2][]{
  \gdef\func@mag{}
  \gdef\func@ph{}
  \build@TF@plot{\func@mag}{\func@ph}{#2}
  \if@pgfarg
    \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do
able=t, trig format plots=rad, #1]}
    \temp@cmd ( {\n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale))},
      {\n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))} );
  \else
    \stepcounter{gnuplot@id}
    \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do
able=t, #1]}
    \temp@cmd gnuplot [parametric, gnuplot@prefix]{
      \n@pow{10}{((\func@mag)/20)}*cos((\func@ph)/(\ph@scale)),
      \n@pow{10}{((\func@mag)/20)}*sin((\func@ph)/(\ph@scale))
    };
  \fi
}
```

`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.

```
\AtBeginDocument{%
  \if@babel
    \AddToHook{env/NyquistPlot/begin}{\expandafter\shorthandoff\expandafter{\shorthar
    \AddToHook{env/NyquistPlot/end}{\expandafter\shorthandon\expandafter{\shorthand@
  \fi
}
\NewDocumentEnvironment{NyquistPlot}{O{}mm+b}{
  \parse@env@opt{#1}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
  \temp@cmd
    \edef\temp@cmd{\noexpand\begin{axis}[
      bode@style,
      height=5cm,
      domain=#2:#3,
      xlabel={$\Re$},
      ylabel={$\Im$},
      \unexpanded\expandafter{\opt@axes}
    ]}
    \temp@cmd
      \addplot [only marks,mark=+,thick,red] (-1 , 0);
      #4
    \end{axis}
  \end{tikzpicture}
}{}
```

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`, `scale`, or `tikz` then the corresponding `\opt` are passed, unexpanded, to the `\addplot` macro, the `axis` environment, the scaling option, and the `tikzpicture` environment, respectively.

```
\newcommand{\parse@N@opt}[1]{
  \gdef\opt@axes{}
  \gdef\opt@plot{}
  \gdef\opt@commands{}
  \gdef\opt@tikz{}
  \gdef\opt@scale{linear}
  \foreach \obj/\opt in {#1} {
    \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{axes}=0
      \xdef\opt@axes{\unexpanded\expandafter{\opt}}
    \else
      \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{plot}=0
        \xdef\opt@plot{\unexpanded\expandafter{\opt}}
      \else
        \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{commands}=0
          \xdef\opt@commands{\unexpanded\expandafter{\opt}}
        \else
          \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{tikz}=0
            \xdef\opt@tikz{\unexpanded\expandafter{\opt}}
          \else
            \ifnum\pdf@strcmp{\unexpanded\expandafter{\obj}}{scale}=0
              \xdef\opt@scale{\unexpanded\expandafter{\opt}}
            \else
              \xdef\opt@plot{\unexpanded\expandafter{\opt@plot},
                \unexpanded\expandafter{\obj}}
            \fi
          \fi
        \fi
      \fi
    \fi
  }
}
```

`\min@real@ZPK` Computes the minimum nonzero absolute value of real parts from all poles and zeros in ZPK format. This is used for automatically setting the threshold in logarithmic pole-zero maps. The result is stored in `\min@re@threshold@result`.

```
\newcommand{\min@real@ZPK}[1]{
  % Initialize with large default value
  \gdef\min@re@threshold@result{1000}
  \def\@min@false{false}
  \gdef\min@threshold@found{false}
  % Keep a float version in FPU format for safe comparisons under LuaLaTeX
  \global\let\min@thresh@float\relax
  % Track maximum absolute value for axis sizing
  \pgfkeys{/pgf/fpu=true}
  \pgfmathparse{0}
  \global\let\max@re@float=\pgfmathresult
  % Track positive and negative sides separately
  \pgfmathparse{0}
  \global\let\max@re@pos@float=\pgfmathresult
  \pgfmathparse{0}
  \global\let\max@re@neg@float=\pgfmathresult
  \pgfkeys{/pgf/fpu=false}
  \gdef\max@re@value{0}
  \gdef\has@positive@values{false}
  \gdef\has@negative@values{false}
  \foreach \feature/\values in {#1} {
    \ifnum\pdf@strcmp{\feature}{z}=0
      % Process zeros
      \foreach \z in \values {
        \foreach \y [count=\zcnt] in \z {
          \ifnum\zcnt=1
            % Compute absolute value using PGF FPU to avoid TeX di-
```

```
men overflows
            \pgfkeys{/pgf/fpu=true}
            \pgfmathparse{abs(\y)}
            \let\abs@valuefloat=\pgfmathresult
            \pgfmathfloattofixed{\abs@valuefloat}
            \edef\abs@value{\pgfmathresult}
            \pgfkeys{/pgf/fpu=false}
            % Skip if zero (string compare avoids numeric parser)
            \ifnum\pdf@strcmp{\abs@value}{0}=0\else
              \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
                % Check if value is positive or nega-
tive and track separately
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\y >= 0 ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  % Positive value
                  \gdef\has@positive@values{true}
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat > \max@re@pos@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \global\let\max@re@pos@float=\abs@valuefloat
                  \fi
                \else
                  % Negative value
                  \gdef\has@negative@values{true}
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat > \max@re@neg@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \global\let\max@re@neg@float=\abs@valuefloat
                  \fi
                \fi
                % Update overall maximum tracker
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\abs@valuefloat > \max@re@float ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  \global\let\max@re@float=\abs@valuefloat
                  \xdef\max@re@value{\abs@value}
                \fi
                \ifx\min@threshold@found\@min@false
                  % First valid nonzero value
                  \xdef\min@re@threshold@result{\abs@value}
                  \global\let\min@thresh@float=\abs@valuefloat
                  \gdef\min@threshold@found{true}
                \else
                  % Compare floats safely with FPU; then trun-
cate boolean to an int
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat < \min@thresh@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \xdef\min@re@threshold@result{\abs@value}
                    \global\let\min@thresh@float=\abs@valuefloat
                  \fi
                \fi
```

```
              \fi
            \fi
          \fi
        }
      }
    \fi
    \ifnum\pdf@strcmp{\feature}{p}=0
      % Process poles
      \foreach \p in \values {
        \foreach \y [count=\pcnt] in \p {
          \ifnum\pcnt=1
            % Compute absolute value using PGF FPU to avoid TeX di-
men overflows
            \pgfkeys{/pgf/fpu=true}
            \pgfmathparse{abs(\y)}
            \let\abs@valuefloat=\pgfmathresult
            \pgfmathfloattofixed{\abs@valuefloat}
            \edef\abs@value{\pgfmathresult}
            \pgfkeys{/pgf/fpu=false}
            % Skip if zero (string compare avoids numeric parser)
            \ifnum\pdf@strcmp{\abs@value}{0}=0\else
              \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
                % Check if value is positive or nega-
tive and track separately
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\y >= 0 ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  % Positive value
                  \gdef\has@positive@values{true}
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat > \max@re@pos@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \global\let\max@re@pos@float=\abs@valuefloat
                  \fi
                \else
                  % Negative value
                  \gdef\has@negative@values{true}
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat > \max@re@neg@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \global\let\max@re@neg@float=\abs@valuefloat
                  \fi
                \fi
                \ifx\min@threshold@found\@min@false
                  % First valid nonzero value
                  \xdef\min@re@threshold@result{\abs@value}
                  \global\let\min@thresh@float=\abs@valuefloat
                  \gdef\min@threshold@found{true}
                \else
                  % Compare floats safely with FPU; then trun-
cate boolean to an int
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat < \min@thresh@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \xdef\min@re@threshold@result{\abs@value}
```

```
                    \global\let\min@thresh@float=\abs@valuefloat
                  \fi
                \fi
              \fi
            \fi
          \fi
        }
      }
    \fi
  }
  % If no valid values found, use default
  \ifx\min@threshold@found\@min@false
    \gdef\min@re@threshold@result{0.01}
  \fi
  \xdef\min@threshold@result{\min@re@threshold@result}
  % Compute \min@re@pow@10 such that 10^{\min@re@pow@10} is the clos-
est power of 10 smaller than or equal to \min@re@threshold@result
  \pgfkeys{/pgf/fpu=true}
  \pgfmathparse{log10(\min@re@threshold@result)}
  \let\log@result=\pgfmathresult
  % Add small epsilon to handle floating-point precision is-
sues with exact powers of 10
  \pgfmathparse{\log@result + 1e-5}
  \let\log@adjusted=\pgfmathresult
  \pgfmathparse{floor(\log@adjusted)}
  \pgfmathfloattofixed{\pgfmathresult}
  \xdef\min@re@pow@10{\pgfmathresult}
  \xdef\min@pow@10{\min@re@pow@10}
  % Compute separate maximum exponents for positive and negative sides
  % Positive side
  \ifx\has@positive@values\@min@false
    \xdef\max@re@pos@pow@10{\min@re@pow@10}
  \else
    \pgfmathparse{log10(max(\max@re@pos@float,1e-100))}
    \let\log@max@re@pos=\pgfmathresult
    \pgfmathparse{\log@max@re@pos + 1e-5}
    \let\log@max@re@pos@adjusted=\pgfmathresult
    \pgfmathparse{ceil(\log@max@re@pos@adjusted)}
    \pgfmathfloattoint{\pgfmathresult}
    \xdef\max@re@pos@pow@10{\pgfmathresult}
  \fi
  % Negative side
  \ifx\has@negative@values\@min@false
    \xdef\max@re@neg@pow@10{\min@re@pow@10}
  \else
    \pgfmathparse{log10(max(\max@re@neg@float,1e-100))}
    \let\log@max@re@neg=\pgfmathresult
    \pgfmathparse{\log@max@re@neg + 1e-5}
    \let\log@max@re@neg@adjusted=\pgfmathresult
    \pgfmathparse{ceil(\log@max@re@neg@adjusted)}
    \pgfmathfloattoint{\pgfmathresult}
    \xdef\max@re@neg@pow@10{\pgfmathresult}
  \fi
  % Keep overall maximum for backward compatibility
  \pgfmathparse{max(\max@re@pos@float > 0 ? \max@re@pos@float : 0, \max@re@neg@float
  \let\max@re@valuefloat=\pgfmathresult
  \pgfmathparse{\max@re@valuefloat > 0 ? \max@re@valuefloat : \min@re@threshold@resul
  \let\max@re@valuefloat=\pgfmathresult
  \pgfmathparse{log10(max(\max@re@valuefloat,1e-100))}
  \let\log@max@re=\pgfmathresult
  \pgfmathparse{\log@max@re + 1e-5}
  \let\log@max@re@adjusted=\pgfmathresult
  \pgfmathparse{ceil(\log@max@re@adjusted)}
```

```
  \pgfmathfloattoint{\pgfmathresult}
  \xdef\max@re@pow@10{\pgfmathresult}
  \pgfkeys{/pgf/fpu=false}
}
```

`\min@im@ZPK` Computes the minimum nonzero absolute value of imaginary parts from all poles and zeros in ZPK format. This is used for automatically setting thresholds in logarithmic pole-zero maps for imaginary axis scaling. The result is stored in `\min@im@threshold@result` and the corresponding power of 10 is stored in `\min@im@pow@10`.

```
\newcommand{\min@im@ZPK}[1]{
  % Initialize with large default value
  \gdef\min@im@threshold@result{1000}
  \def\@min@false{false}
  \gdef\min@im@threshold@found{false}
  % Keep a float version in FPU format for safe comparisons under LuaLaTeX
  \global\let\min@im@thresh@float\relax
  % Track maximum absolute value for axis sizing
  \pgfkeys{/pgf/fpu=true}
  \pgfmathparse{0}
  \global\let\max@im@float=\pgfmathresult
  \pgfkeys{/pgf/fpu=false}
  \gdef\max@im@value{0}
  \foreach \feature/\values in {#1} {
    \ifnum\pdf@strcmp{\feature}{z}=0
      % Process zeros
      \foreach \z in \values {
        \foreach \y [count=\zcnt] in \z {
          \ifnum\zcnt=2
            % Second element is imaginary part - compute absolute value using PGF FPU
            \pgfkeys{/pgf/fpu=true}
            \pgfmathparse{abs(\y)}
            \let\abs@valuefloat=\pgfmathresult
            \pgfmathfloattofixed{\abs@valuefloat}
            \edef\abs@value{\pgfmathresult}
            \pgfkeys{/pgf/fpu=false}
            % Skip if zero (string compare avoids numeric parser)
            \ifnum\pdf@strcmp{\abs@value}{0}=0\else
              \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
                % Update maximum tracker
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\abs@valuefloat > \max@im@float ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  \global\let\max@im@float=\abs@valuefloat
                  \xdef\max@im@value{\abs@value}
                \fi
                \ifx\min@im@threshold@found\@min@false
                  % First valid nonzero value
                  \xdef\min@im@threshold@result{\abs@value}
                  \global\let\min@im@thresh@float=\abs@valuefloat
                  \gdef\min@im@threshold@found{true}
                \else
                  % Compare floats safely with FPU; then truncate boolean to an int
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat < \min@im@thresh@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
```

```
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \xdef\min@im@threshold@result{\abs@value}
                    \global\let\min@im@thresh@float=\abs@valuefloat
                  \fi
                \fi
              \fi
            \fi
          \fi
        }
      }
    \fi
    \ifnum\pdf@strcmp{\feature}{p}=0
      % Process poles
      \foreach \p in \values {
        \foreach \y [count=\pcnt] in \p {
          \ifnum\pcnt=2
            % Second element is imaginary part - compute abso-
lute value using PGF FPU
            \pgfkeys{/pgf/fpu=true}
            \pgfmathparse{abs(\y)}
            \let\abs@valuefloat=\pgfmathresult
            \pgfmathfloattofixed{\abs@valuefloat}
            \edef\abs@value{\pgfmathresult}
            \pgfkeys{/pgf/fpu=false}
            % Skip if zero (string compare avoids numeric parser)
            \ifnum\pdf@strcmp{\abs@value}{0}=0\else
              \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
                \ifx\min@im@threshold@found\@min@false
                  % First valid nonzero value
                  \xdef\min@im@threshold@result{\abs@value}
                  \global\let\min@im@thresh@float=\abs@valuefloat
                  \gdef\min@im@threshold@found{true}
                \else
                  % Compare floats safely with FPU; then trun-
cate boolean to an int
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat < \min@im@thresh@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \xdef\min@im@threshold@result{\abs@value}
                    \global\let\min@im@thresh@float=\abs@valuefloat
                  \fi
                \fi
              \fi
            \fi
          \fi
        }
      }
    \fi
  }
  % If no valid values found, use default
  \ifx\min@im@threshold@found\@min@false
    \gdef\min@im@threshold@result{0.01}
  \fi
  % Compute \min@im@pow@10 such that 10^{\min@im@pow@10} is the clos-
est power of 10 smaller than or equal to \min@im@threshold@result
  \pgfkeys{/pgf/fpu=true}
  \pgfmathparse{log10(\min@im@threshold@result)}
  \let\log@result=\pgfmathresult
  % Add small epsilon to handle floating-point precision is-
sues with exact powers of 10
```

```
  \pgfmathparse{\log@result + 1e-5}
  \let\log@adjusted=\pgfmathresult
  \pgfmathparse{floor(\log@adjusted)}
  \pgfmathfloattofixed{\pgfmathresult}
  \xdef\min@im@pow@10{\pgfmathresult}
  \xdef\min@Im@pow@10{\min@im@pow@10}
  % Compute maximum exponent to determine axis extent for imaginary parts
  \pgfmathparse{\max@im@float > 0 ? \max@im@float : \min@im@threshold@result}
  \let\max@im@valuefloat=\pgfmathresult
  \pgfmathparse{log10(max(\max@im@valuefloat,1e-100))}
  \let\log@max@im=\pgfmathresult
  \pgfmathparse{\log@max@im + 1e-5}
  \let\log@max@im@adjusted=\pgfmathresult
  \pgfmathparse{ceil(\log@max@im@adjusted)}
  \pgfmathfloattoint{\pgfmathresult}
  \xdef\max@im@pow@10{\pgfmathresult}
  \pgfkeys{/pgf/fpu=false}
}
```

4.6 Nichols charts

\NicholsZPK
\NicholsTF
NicholsChart
\addNicholsZPKChart
\addNicholsTFChart

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

```
\newcommand{\NicholsZPK}[4][]{
  \parse@N@opt{#1}
  \gdef\func@mag{}
  \gdef\func@ph{}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
  \build@ZPK@plot{\func@mag}{\func@ph}{}{#2}
    \edef\temp@cmd{\noexpand\begin{axis}[
      ph@x@label,
      bode@style,
      domain=#3*\freq@scale:#4*\freq@scale,
      height=5cm,
      ylabel={Gain (dB)},
      samples=500,
      \unexpanded\expandafter{\opt@axes}
    ]}
    \temp@cmd
      \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig format plots=rad, \opt@plot]}
      \if@pgfarg
        \temp@cmd ( {\func@ph} , {\func@mag} );
        \opt@commands
      \else
        \stepcounter{gnuplot@id}
        \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
        { set table $meta;
          set logscale x 10;
          set dummy t;
          set samples \pgfkeysvalueof{/pgfplots/samples};
          set trange [#3*\freq@scale:#4*\freq@scale];
          plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
          unset logscale x;
          set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
          plot "$meta" using ($2*\ph@scale):($1);
        };
        \opt@commands
```

```
      \fi
    \end{axis}
  \end{tikzpicture}
}
\AtBeginDocument{%
  \if@babel
  \let\Orig@NicholsZPK\NicholsZPK
  \renewcommand{\NicholsZPK}{%
    \expandafter\shorthandoff\expandafter{\shorthand@list}
    \NicholsZPK@Shorthandoff
  }
  \newcommand{\NicholsZPK@Shorthandoff}[4][]{%
    \Orig@NicholsZPK[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\shorthand@list}
  }
  \fi
}
\newcommand{\NicholsTF}[4][]{
  \parse@N@opt{#1}
  \gdef\func@mag{}
  \gdef\func@ph{}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unexpanded\expandafter{\opt@tikz}]}
  \temp@cmd
    \build@TF@plot{\func@mag}{\func@ph}{#2}
    \edef\temp@cmd{\noexpand\begin{axis}[
      ph@x@label,
      bode@style,
      domain=#3*\freq@scale:#4*\freq@scale,
      height=5cm,
      ylabel={Gain (dB)},
      samples=500,
      \unexpanded\expandafter{\opt@axes}
    ]}
    \temp@cmd
      \edef\temp@cmd{\noexpand\addplot [variable=t, thick, trig format plots=rad, \opt@plot]}
      \if@pgfarg
        \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
        \opt@commands
      \else
        \stepcounter{gnuplot@id}
        \temp@cmd gnuplot [raw gnuplot, gnuplot@prefix]
          { set table $meta1;
            set logscale x 10;
            set dummy t;
            set samples \pgfkeysvalueof{/pgfplots/samples};
            set trange [#3*\freq@scale:#4*\freq@scale];
            plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
            unset logscale x;
            set table $meta2;
            plot "$meta1" using ($1):($2) smooth unwrap;
            set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
            plot "$meta2" using ($2*\ph@scale):($1);
          };
        \opt@commands
      \fi
    \end{axis}
  \end{tikzpicture}
}
\AtBeginDocument{
  \if@babel
    \let\Orig@NicholsTF\NicholsTF
```

```
    \renewcommand{\NicholsTF}{%
      \expandafter\shorthandoff\expandafter{\shorthand@list}
      \NicholsTF@Shorthandoff
    }
    \newcommand{\NicholsTF@Shorthandoff}[4][]{%
      \Orig@NicholsTF[#1]{#2}{#3}{#4}
      \expandafter\shorthandon\expandafter{\shorthand@list}
    }
    \AddToHook{env/NicholsChart/begin}{\expandafter\shorthandoff\expandafter{\shortha
    \AddToHook{env/NicholsChart/end}{\expandafter\shorthandon\expandafter{\shorthand@
  \fi
}
\NewDocumentEnvironment{NicholsChart}{O{}mm+b}{
  \parse@env@opt{#1}
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
  \temp@cmd
    \edef\temp@cmd{\noexpand\begin{axis}[
      ph@x@label,
      bode@style,
      domain=#2:#3,
      height=5cm,
      ylabel={Gain (dB)},
      \unexpanded\expandafter{\opt@axes}
    ]}
    \temp@cmd
      #4
    \end{axis}
  \end{tikzpicture}
}{}
\newcommand{\addNicholsZPKChart}[2][]{
  \gdef\func@mag{}
  \gdef\func@ph{}
  \build@ZPK@plot{\func@mag}{\func@ph}{}{#2}
  \if@pgfarg
    \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do
able=t, trig format plots=rad, #1]}
    \temp@cmd ( {\func@ph} , {\func@mag} );
  \else
    \stepcounter{gnuplot@id}
    \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
    { set table $meta;
      set logscale x 10;
      set dummy t;
      set samples \pgfkeysvalueof{/pgfplots/samples};
      set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
      plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
      unset logscale x;
      set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
      plot "$meta" using ($2*\ph@scale):($1);
    };
  \fi
}
\newcommand{\addNicholsTFChart}[2][]{
  \gdef\func@mag{}
  \gdef\func@ph{}
  \build@TF@plot{\func@mag}{\func@ph}{#2}
  \if@pgfarg
    \edef\temp@cmd{\noexpand\addplot [domain=\freq@scale*\pgfkeysvalueof{/pgfplots/do
able=t, trig format plots=rad, #1]}
    \temp@cmd ( {\n@mod{\func@ph}{2*pi*\ph@scale}} , {\func@mag} );
  \else
    \stepcounter{gnuplot@id}
```

```
    \addplot [#1] gnuplot [raw gnuplot, gnuplot@prefix]
    { set table $meta1;
      set logscale x 10;
      set dummy t;
      set samples \pgfkeysvalueof{/pgfplots/samples};
      set trange [\freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\freq@scale];
      plot '+' using (\func@mag) : ((\func@ph)/(\ph@scale));
      unset logscale x;
      set table $meta2;
      plot "$meta1" using ($1):($2) smooth unwrap;
      set table "\bodeplot@prefix\arabic{gnuplot@id}.table";
      plot "$meta2" using ($2*\ph@scale):($1);
    };
  \fi
}
```

\PoleZeroMapZPK Creates a pole-zero map similar to MATLAB's `pzmap` function. Poles are plotted as 'x' markers and zeros as 'o' markers on the complex s-plane. The gain parameter is ignored since it does not affect pole-zero locations, and delay is also ignored since it does not add poles or zeros.

```
\newcommand{\PoleZeroMapZPK}[2][]{
  \parse@N@opt{#1}
  % Set threshold for log scale if needed
  \ifnum\pdf@strcmp{\opt@scale}{log}=0
  % Compute automatic threshold as the minimum nonzero abso-
lute real part
  % from the provided ZPK data.
    \min@real@ZPK{#2}
    \min@im@ZPK{#2}
    % Prepare axis scaling helpers
    \pgfkeys{/pgf/fpu=true}
    % Compute separate tick counts for positive and negative x-axis
    \ifx\has@positive@values\@min@false
      \xdef\PoleZeroMapZPK@ticksXPos{0}
    \else
      \pgfmathparse{max(\max@re@pos@pow@10 - \min@re@pow@10 + 1, 1)}
      \pgfmathfloattoint{\pgfmathresult}
      \xdef\PoleZeroMapZPK@ticksXPos{\pgfmathresult}
    \fi
    \ifx\has@negative@values\@min@false
      \xdef\PoleZeroMapZPK@ticksXNeg{0}
    \else
      \pgfmathparse{max(\max@re@neg@pow@10 - \min@re@pow@10 + 1, 1)}
      \pgfmathfloattoint{\pgfmathresult}
      \xdef\PoleZeroMapZPK@ticksXNeg{\pgfmathresult}
    \fi
    \pgfkeys{/pgf/fpu=false}
    \def\PoleZeroMapZPK@formatXTick##1{%
      \pgfmathtruncatemacro{\PoleZeroMapZPK@tick}{##1}%
      \ifnum\PoleZeroMapZPK@tick=0
        $0$
      \else
        \pgfmathtruncatemacro{\PoleZeroMapZPK@exp}{\min@re@pow@10 + abs(\PoleZeroMapZ
1}%
        \ifnum\PoleZeroMapZPK@tick>0
          $10^{\PoleZeroMapZPK@exp}$%
        \else
          $-10^{\PoleZeroMapZPK@exp}$%
        \fi
      \fi
    }
    \def\PoleZeroMapZPK@formatYTick##1{%
      \pgfmathtruncatemacro{\PoleZeroMapZPK@tick}{##1}%
```

```
      \ifnum\PoleZeroMapZPK@tick=0
        $0$
      \else
        \pgfmathtruncatemacro{\PoleZeroMapZPK@exp}{\min@im@pow@10 + abs(\PoleZeroMapZ
1}%
        \ifnum\PoleZeroMapZPK@tick>0
          $10^{\PoleZeroMapZPK@exp}$%
        \else
          $-10^{\PoleZeroMapZPK@exp}$%
        \fi
      \fi
    }
    \def\PoleZeroMapZPK@xticklabel{\PoleZeroMapZPK@formatXTick{\tick}}
    \def\PoleZeroMapZPK@yticklabel{\PoleZeroMapZPK@formatYTick{\tick}}
  \fi
  \edef\temp@cmd{\noexpand\begin{tikzpicture} [\unex-
panded\expandafter{\opt@tikz}]}
  \temp@cmd
    % Set up axis based on scale option
    \ifnum\pdf@strcmp{\opt@scale}{log}=0
      \edef\temp@cmd{\noexpand\begin{axis}[
        xlabel={$\Re$},
        ylabel={$\Im$},
        axis lines=center,
        grid=major,
        height=6cm,
        enlarge x limits=0.2,
        enlarge y limits=0.2,
        xtick distance=1,
        ytick distance=1,
        xticklabel=\noexpand\PoleZeroMapZPK@xticklabel,
        yticklabel=\noexpand\PoleZeroMapZPK@yticklabel,
        x filter/.expression={abs(x) < \min@re@threshold@result ? 0 : (x >= 0 ? (log1
100)) - \min@re@pow@10 + 1) : (-log10(max(min(-x, 1e100), 1e-
100)) + \min@re@pow@10 - 1))},
        y filter/.expression={abs(y) < \min@im@threshold@result ? 0 : (y >= 0 ? (log1
100)) - \min@im@pow@10 + 1) : (-log10(max(min(-y, 1e100), 1e-
100)) + \min@im@pow@10 - 1))},
        \unexpanded\expandafter{\opt@axes}
      ]}
    \else
      \edef\temp@cmd{\noexpand\begin{axis}[
        xlabel={$\Re$},
        ylabel={$\Im$},
        axis lines=center,
        grid=major,
        height=6cm,
        enlarge x limits=0.2,
        enlarge y limits=0.2,
        \unexpanded\expandafter{\opt@axes}
      ]}
    \fi
    \temp@cmd
      % Plot poles and zeros from ZPK data
      \foreach \feature/\values in {#2} {
        \ifnum\pdf@strcmp{\feature}{z}=0
          \foreach \z in \values {
            \foreach \y [count=\zcnt] in \z {
              \ifnum\zcnt=1
                \xdef\zreal{\y}
              \fi
              \ifnum\zcnt=2
                \xdef\zimag{\y}
```

```
              \fi
              \xdef\Last@LoopValue@z{\zcnt}
            }
            \ifnum\Last@LoopValue@z=1
              \xdef\zimag{0}
            \fi
            \edef\temp@cmd{\noexpand\addplot [only marks, mark=o, mark size=3pt, thic
expanded\expandafter{\opt@plot}]}
            \temp@cmd coordinates {(\zreal,\zimag)};
          }
        \fi
        \ifnum\pdf@strcmp{\feature}{p}=0
          \foreach \p in \values {
            \foreach \y [count=\pcnt] in \p {
              \ifnum\pcnt=1
                \xdef\preal{\y}
              \fi
              \ifnum\pcnt=2
                \xdef\pimag{\y}
              \fi
              \xdef\Last@LoopValue@p{\pcnt}
            }
            \ifnum\Last@LoopValue@p=1
              \xdef\pimag{0}
            \fi
            \edef\temp@cmd{\noexpand\addplot [only marks, mark=x, mark size=3pt, thic
expanded\expandafter{\opt@plot}]}
            \temp@cmd coordinates {(\preal,\pimag)};
          }
        \fi
      }
      \opt@commands
    \end{axis}
  \end{tikzpicture}
}
```

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

```
\AtBeginDocument{%
  \if@babel
  \let\Orig@PoleZeroMapZPK\PoleZeroMapZPK
  \renewcommand{\PoleZeroMapZPK}{%
    \expandafter\shorthandoff\expandafter{\shorthand@list}
    \PoleZeroMapZPK@Shorthandoff
  }
  \newcommand{\PoleZeroMapZPK@Shorthandoff}[2][]{%
    \Orig@PoleZeroMapZPK[#1]{#2}
    \expandafter\shorthandon\expandafter{\shorthand@list}
  }
  \fi
}
```

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

\@babelfalse 44
\@babeltrue 47
\@declutterargfalse . 5
\@declutterargtrue . . 7
\@empty 48
\@hzargfalse 13
\@hzargtrue 15
\@ifpackageloaded . . 46
\@min@false 999, 1067, 1130, 1154, 1171, 1183, 1213, 1248,

1285, 1309, 1519, 1526
\@nil 120, 121
\@pgfargfalse 1
\@pgfargtrue 3
\@radargfalse 9
\@radargtrue 11
\~ 51

A

\abs@value 1027, 1030, 1031, 1065, 1069, 1079, 1099, 1102, 1103, 1132, 1142, 1234, 1237, 1238, 1246, 1250, 1260, 1280, 1283, 1284, 1287, 1297
\abs@valuefloat 1025, 1026, 1041, 1045, 1051, 1055, 1060, 1064, 1070, 1075, 1080, 1097, 1098, 1113, 1117, 1123, 1127, 1133, 1138, 1143, 1232, 1233, 1241, 1245, 1251, 1256, 1261, 1278, 1279, 1288, 1293, 1298
\add@feature 559, 578, 579, 582, 583, 585, 586, 594, 595, 598, 599, 601, 602, 609, 610, 613, 614, 616, 617, 633, 634
\addBodeComponentPlot 427
\addBodePlot 445
\addBodeTFPlot 384
\addBodeZPKPlots . . 356
\addNicholsTFChart 1337
\addNicholsZPKChart 1337
\addNyquistTFPlot . 924
\addNyquistZPKPlot 907
\AddToHook 484, 485, 509, 510, 943, 944, 1441, 1442
\AtBeginDocument 45, 274, 343, 482, 507, 847, 894, 941, 1376, 1430, 1639

B

\begingroup 50
\bode@style 58
BodeMagPlot (env.) . . 507
BodePhPlot (env.) . . . 482
BodePlot (env.) 532
\bodeplot@prefix 28, 30, 35, 254, 266, 323, 335, 378, 410, 421, 440, 790, 797, 802, 809, 1368, 1422, 1479, 1503
\BodeTF 287
\BodeTF@Shorthandoff 348, 350
\BodeZPK 218
\BodeZPK@Shorthandoff 279, 281
\build@gnu@plot 465, 470, 789
\build@gnu@unwrap@plot 473, 801
\build@TF@plot . 293, 388, 390, 453, 640, 866, 927, 1395, 1487
\build@ZPK@plot 224, 361, 363, 450, 573, 819, 910, 1343, 1465

C

\currentdegree . 652, 653, 656, 657, 658, 660, 661, 671, 672, 675, 676, 677, 679, 680

D

\den@degree 668, 671
\den@im 644, 676, 689, 692
\den@real 643, 673, 679, 690, 692
\dencoeff 667, 670, 673, 676, 679
\denpow 667, 668, 670, 671
\do 49
\dospecials 55

E

\endgroup 53
environments:
BodeMagPlot 507
BodePhPlot 482
BodePlot 532

F

\freq@filter 73
\freq@label 73
\freq@scale 73, 74, 85, 109, 116, 229, 251, 255, 263, 267, 298, 320, 324, 332, 336, 366, 375, 379, 394, 397, 407, 411, 418, 422, 429, 437, 441, 460, 461, 542, 798, 810, 822, 869, 912, 917, 929, 934, 1347, 1365, 1399, 1417, 1467, 1476, 1489, 1498
\func@mag 220, 224, 241, 253, 289, 293, 310, 322, 815, 819, 833, 834, 839, 840, 862, 866, 880, 881, 886, 887, 908, 910, 913, 914, 919, 920, 925, 927, 930, 931, 936, 937, 1339, 1343, 1356, 1366, 1391, 1395, 1408, 1418, 1463, 1465, 1468, 1477, 1485, 1487, 1490, 1499
\func@ph 221, 224, 243, 265, 290, 293, 312, 334, 816, 819, 833, 834, 839, 840, 863, 866, 880, 881, 886, 887, 909, 910, 913, 914, 919, 920, 926, 927, 930, 931, 936, 937, 1340, 1343, 1356, 1366, 1392, 1395, 1408, 1418, 1464, 1465, 1468, 1477, 1486, 1487, 1490, 1499

G

\g@addto@macro 52
\get@interval@end 120, 121
\get@interval@start 120, 120
\global 1002, 1006, 1009, 1011, 1045, 1055, 1064, 1070, 1080, 1117, 1127, 1133, 1143, 1216, 1220, 1245, 1251, 1261, 1288, 1298
\gnu@cmd 466, 479, 790, 802
\gnu@id 464, 465, 468, 470, 473
\gnuplot@id 1
\gnuplot@prefix 1

H

\has@negative@values 1015, 1049, 1121, 1183, 1526
\has@positive@values 1014, 1039, 1111, 1171, 1519

I

\if@babel 44
\if@hzarg 13, 84
\ifbabelshorthand . . 52
\ifcat 560
\ifnum 360, 387, 393, 402, 449, 452, 469, 472, 569, 575, 577, 581, 591, 593, 597, 607, 608, 612, 621, 622, 626, 647, 653, 666, 672, 685, 705, 706, 709, 717, 718, 721, 729, 732, 735, 736, 742,

761, 764, 777, 780, 972, 975, 978, 981, 984, 1017, 1021, 1030, 1031, 1037, 1044, 1054, 1063, 1078, 1089, 1093, 1102, 1103, 1109, 1116, 1126, 1141, 1224, 1228, 1237, 1238, 1244, 1259, 1270, 1274, 1283, 1284, 1296, 1511, 1536, 1540, 1549, 1553, 1566, 1598, 1601, 1604, 1609, 1616, 1619, 1622, 1627
\ifwindows 38
\ifx 1067, 1130, 1154, 1171, 1183, 1248, 1285, 1309, 1519, 1526
\immediate 40

J

\jobname 28, 30

L

\Last@LoopValue 567, 569
\Last@LoopValue@p 1625, 1627
\Last@LoopValue@z 1607, 1609
\lccode 51
\log@adjusted . 1164, 1165, 1318, 1319
\log@max@im . . 1327, 1328
\log@max@im@adjusted 1329, 1330
\log@max@re . . 1200, 1201
\log@max@re@adjusted 1202, 1203
\log@max@re@neg 1187, 1188
\log@max@re@neg@adjusted 1189, 1190
\log@max@re@pos 1175, 1176
\log@max@re@pos@adjusted 1177, 1178
\log@result . . . 1161, 1163, 1315, 1317
\loop@delay 645, 686, 690
\lowercase 52

M

\MagCSPoles 163
\MagCSPolesAsymp . . 163
\MagCSPolesLin 163
\MagCSPolesPeak . . . 182
\MagCSZeros 163
\MagCSZerosAsymp . . 163
\MagCSZerosLin 163
\MagCSZerosPeak . . . 182
\MagDel 128, 634
\MagK 122, 617
\MagKAsymp 122, 614
\MagKLin 122, 610
\MagPole . . . 130, 157, 602
\MagPoleAsymp 130, 159, 599
\MagPoleLin 130, 158, 595
\MagSOPoles 190
\MagSOPolesAsymp . . 190
\MagSOPolesLin 190
\MagSOPolesPeak . . . 208
\MagSOZeros 190
\MagSOZerosAsymp . . 190
\MagSOZerosLin 190
\MagSOZerosPeak . . . 208
\MagZero 157, 586
\MagZeroAsymp . . 157, 583
\MagZeroLin 157, 579
\max@im@float . 1220, 1241, 1245, 1324
\max@im@pow@ 1332
\max@im@value 1222, 1246
\max@im@valuefloat 1325, 1326
\max@re@float 1006, 1060, 1064
\max@re@neg@float . 1011, 1051, 1055, 1123, 1127, 1186, 1195
\max@re@neg@pow@ 1184, 1192, 1529
\max@re@pos@float . 1009, 1041, 1045, 1113, 1117, 1174, 1195
\max@re@pos@pow@ 1172, 1180, 1522
\max@re@pow@ 1205
\max@re@value 1013, 1065
\max@re@valuefloat 1196, 1197, 1198, 1199
\min@Im@pow@ 1322
\min@im@pow@ . . 1312, 1321, 1322, 1552, 1580
\min@im@thresh@float 1216, 1251, 1256, 1261, 1288, 1293, 1298
\min@im@threshold@found 1214, 1248, 1252, 1285, 1289, 1309
\min@im@threshold@result 1212, 1250, 1260, 1287, 1297, 1310, 1312, 1314, 1324, 1580
\min@im@ZPK . . 1210, 1515
\min@pow@ 1168
\min@re@pow@ 1158, 1167, 1168, 1172, 1184, 1522, 1529, 1539, 1579
\min@re@threshold@result 998, 1069, 1079, 1132, 1142, 1155, 1157, 1158, 1160, 1197, 1579
\min@real@ZPK . 996, 1514
\min@thresh@float . 1002, 1070, 1075, 1080, 1133, 1138, 1143
\min@threshold@found 1000, 1067, 1071, 1130, 1134, 1154
\min@threshold@result 1157

N

\n 565, 567
\n@mod 1, 312, 395, 1408, 1490
\n@mod@n 1
\n@mod@p 1, 138
\n@pow 1, 131, 132, 133, 143, 144, 146, 147, 149, 150, 151, 152, 153, 154, 163, 164, 167, 168, 169, 171, 173, 174, 191, 195, 657, 658, 660, 661, 676, 677, 679, 680, 691, 692, 833, 834, 839, 840, 880, 881, 886, 887, 913, 914, 919, 920, 930, 931, 936, 937
\newcounter 25
\NewDocumentEnvironment 488, 513, 532, 947, 1445
\NicholsChart 1337
\NicholsTF 1337
\NicholsTF@Shorthandoff 1435, 1437
\NicholsZPK 1337
\NicholsZPK@Shorthandoff 1381, 1383
\num@degree 649, 652
\num@im 642, 657, 689, 691
\num@real 641, 654, 660, 689, 691
\numcoeff 648, 651, 654, 657, 660
\numpow 648, 649, 651, 652
\NyquistPlot 941
\NyquistTF 860
\NyquistTF@Shorthandoff 899, 901
\NyquistZPK 813
\NyquistZPK@Shorthandoff 852, 854

O

\opt@approx 224, 450, 700, 733, 775, 778, 781
\opt@axes 500, 525, 758, 762, 767, 827, 874, 957,

966, 973, 1351, 1403, 1455, 1581, 1592
\opt@commands 835, 842, 882, 889, 968, 979, 1357, 1371, 1409, 1425, 1635
\opt@group 233, 302, 546, 699, 730
\opt@plot 460, 461, 466, 479, 774, 783, 831, 878, 967, 976, 987, 1354, 1406, 1612, 1630
\opt@scale 970, 985, 1511, 1566
\opt@tikz . . 222, 291, 490, 515, 536, 703, 743, 759, 765, 817, 864, 949, 969, 982, 1341, 1393, 1447, 1563
\optmag@axes . . . 236, 305, 550, 695, 719, 724
\optmag@commands 242, 257, 311, 326, 702, 739
\optmag@plot . . . 237, 306, 698, 707, 712, 745
\optph@axes 238, 307, 551, 696, 722, 725
\optph@commands 244, 269, 313, 338, 701, 737
\optph@plot 239, 308, 697, 710, 713, 747
\Orig@BodeTF . . . 345, 351
\Orig@BodeZPK . . 276, 282
\Orig@NicholsTF 1432, 1438
\Orig@NicholsZPK 1378, 1384
\Orig@NyquistTF 896, 902
\Orig@NyquistZPK 849, 855
\Orig@PoleZeroMapZPK 1641, 1647

P

\p 592, 594, 595, 598, 599, 601, 602, 1091, 1092, 1272, 1273, 1617, 1618
\parse@add@Bode@opt 446, 773
\parse@env@opt . 489, 514, 757, 948, 1446
\parse@N@opt 814, 861, 965, 1338, 1390, 1509
\parse@opt 219, 288, 535, 694
\pcnt 1092, 1093, 1273, 1274, 1618, 1619, 1622, 1625
\pdf@strcmp 360, 387, 393, 402, 449, 452, 469, 472, 575, 577, 581, 591, 593, 597, 607, 608, 612, 621, 622, 626, 647, 666, 685, 705, 706, 709, 717, 718, 721, 729, 732, 735, 736, 742, 761, 764, 777, 780, 972, 975, 978, 981, 984, 1017, 1030, 1031, 1089, 1102, 1103, 1224, 1237, 1238, 1270, 1283, 1284, 1511, 1566, 1598, 1616
\pgfkeys . . 1004, 1012, 1023, 1028, 1033, 1036, 1040, 1043, 1050, 1053, 1059, 1062, 1074, 1077, 1095, 1100, 1105, 1108, 1112, 1115, 1122, 1125, 1137, 1140, 1159, 1206, 1218, 1221, 1230, 1235, 1240, 1243, 1255, 1258, 1276, 1281, 1292, 1295, 1313, 1333, 1517, 1533
\pgfkeysvalueof . . . 252, 264, 321, 333, 366, 375, 376, 394, 397, 407, 408, 418, 419, 429, 437, 438, 460, 461, 794, 795, 806, 807, 912, 917, 929, 934, 1364, 1416, 1467, 1475, 1476, 1489, 1497, 1498
\pgfmathfloattofixed 1026, 1098, 1166, 1233, 1279, 1320
\pgfmathfloattoint 1035, 1042, 1052, 1061, 1076, 1107, 1114, 1124, 1139, 1179, 1191, 1204, 1242, 1257, 1294, 1331, 1523, 1530
\pgfmathparse 1005, 1008, 1010, 1024, 1034, 1041, 1051, 1060, 1075, 1096, 1106, 1113, 1123, 1138, 1160, 1163, 1165, 1174, 1176, 1178, 1186, 1188, 1190, 1195, 1197, 1199, 1201, 1203, 1219, 1231, 1241, 1256, 1277, 1293, 1314, 1317, 1319, 1324, 1326, 1328, 1330, 1522, 1529
\pgfmathresult 1006, 1009, 1011, 1025, 1027, 1035, 1037, 1042, 1044, 1052, 1054, 1061, 1063, 1076, 1078, 1097, 1099, 1107, 1109, 1114, 1116, 1124, 1126, 1139, 1141, 1161, 1164, 1166, 1167, 1175, 1177, 1179, 1180, 1187, 1189, 1191, 1192, 1196, 1198, 1200, 1202, 1204, 1205, 1220, 1232, 1234, 1242, 1244, 1257, 1259, 1278, 1280, 1294, 1296, 1315, 1318, 1320, 1321, 1325, 1327, 1329, 1331, 1332, 1523, 1524, 1530, 1531
\pgfplotsset 58, 73, 75, 76, 77, 80, 81, 86, 88, 97, 98, 102, 103, 110, 112, 117
\ph@scale 73, 78, 82, 96, 101, 125, 129, 140, 153, 156, 168, 174, 175, 195, 312, 334, 336, 409, 411, 690, 808, 810, 833, 834, 839, 840, 880, 881, 886, 887, 913, 914, 919, 920, 930, 931, 936, 937, 1366, 1369, 1408, 1418, 1423, 1477, 1480, 1490, 1499, 1504
\ph@x@label 73
\ph@y@label 73
\PhCSPoles 163
\PhCSPolesAsymp 163, 200
\PhCSPolesLin . . 163, 197
\PhCSZeros 163
\PhCSZerosAsymp . . . 163
\PhCSZerosLin 163
\PhDel 129, 633
\PhK 122, 616
\PhKAsymp . . 122, 128, 613
\PhKLin . . . 122, 128, 609
\PhPole . . . 130, 160, 601
\PhPoleAsymp 130, 162, 598
\PhPoleLin . 130, 161, 594
\PhSOPoles 190
\PhSOPolesAsymp . . . 190
\PhSOPolesLin 190
\PhSOZeros 190
\PhSOZerosAsymp . . . 190
\PhSOZerosLin 190
\PhZero 157, 585
\PhZeroAsymp . . . 157, 582
\PhZeroLin 157, 578
\pimag . . 1623, 1628, 1631

\plot@a@cmd 460, 466, 533, 553
\plot@b@cmd 461, 479, 534, 555
\plot@macro 358, 361, 363, 367, 377, 385, 388, 390, 395, 398, 409, 420, 447, 450, 453, 460, 465
\PoleZeroMapZPK . . 1508
\PoleZeroMapZPK@exp 1539, 1541, 1543, 1552, 1554, 1556
\PoleZeroMapZPK@formatXTick 1534, 1560
\PoleZeroMapZPK@formatYTick 1547, 1561
\PoleZeroMapZPK@Shorthandoff 1644, 1646
\PoleZeroMapZPK@tick 1535, 1536, 1539, 1540, 1548, 1549, 1552, 1553
\PoleZeroMapZPK@ticksXNeg 1527, 1531
\PoleZeroMapZPK@ticksXPos 1520, 1524
\PoleZeroMapZPK@xticklabel 1560, 1577
\PoleZeroMapZPK@yticklabel 1561, 1578
\preal 1620, 1631

R

\renewcommand 96, 101, 109, 116, 277, 346, 850, 897, 1379, 1433, 1642

S

\setcounter 26
\shorthand@list . 44, 278, 283, 347, 352, 484, 485, 509, 510, 851, 856, 898, 903, 943, 944, 1380, 1385, 1434, 1439, 1441, 1442, 1643, 1648
\shorthandoff 278, 347, 484, 509, 851, 898, 943, 1380, 1434, 1441, 1643
\shorthandon 283, 352, 485, 510, 856, 903, 944, 1385, 1439, 1442, 1648
\stepcounter . . . 246, 258, 315, 327, 369, 401, 432, 463, 467, 837, 884, 916, 933, 1359, 1411, 1470, 1492

T

\temp@cmd 222, 223, 225, 235, 291, 292, 294, 304, 366, 367, 370, 371, 394, 395, 397, 398, 429, 430, 490, 491, 492, 502, 515, 516, 517, 527, 536, 537, 538, 548, 817, 818, 820, 829, 831, 833, 838, 864, 865, 867, 876, 878, 880, 885, 912, 913, 917, 918, 929, 930, 934, 935, 949, 950, 951, 959, 1341, 1342, 1344, 1353, 1354, 1356, 1360, 1393, 1394, 1396, 1405, 1406, 1408, 1412, 1447, 1448, 1449, 1457, 1467, 1468, 1489, 1490, 1563, 1564, 1567, 1584, 1595, 1612, 1613, 1630, 1631
\temp@macro 359, 361, 363, 386, 388, 390, 448, 450, 453, 461, 470, 473
\temp@mag@cmd 236, 241, 247, 305, 310, 316, 550, 552
\temp@ph@cmd 238, 243, 259, 307, 312, 328, 551, 554
\tick 1560, 1561

V

\values 574, 576, 592, 609, 610, 613, 614, 616, 617, 630, 633, 634, 646, 648, 651, 667, 670, 686, 1016, 1019, 1091, 1223, 1226, 1272, 1597, 1599, 1617

W

\write 40

Y

\y 565, 566, 1020, 1024, 1034, 1092, 1096, 1106, 1227, 1231, 1273, 1277, 1600, 1602, 1605, 1618, 1620, 1623

Z

\z 576, 578, 579, 582, 583, 585, 586, 1019, 1020, 1226, 1227, 1599, 1600
\zcnt 1020, 1021, 1227, 1228, 1600, 1601, 1604, 1607
\zimag . . 1605, 1610, 1613
\zreal 1602, 1613

Change History

v1.0
General: Initial release 1
v1.0.1
\addBodeZPKPlots: Improved optional argument handling. . . . 28
\BodeZPK: Pass arbitrary TikZ commands as options. 26
v1.0.2
gnuplot@prefix: Fixed issue #1 . . 21
v1.0.3
BodePlot: Added tikz option to environments 32
\BodeTF: Added Tikz option 27
\BodeZPK: Added Tikz option 26
NicholsChart: Added tikz option to environments 47
\NicholsTF: Added commands and tikz options 47
\NicholsZPK: Added commands and tikz options 47
gnuplot@prefix: Added jobname to gnuplot prefix 21
\NyquistTF: Added commands and tikz options 38
\NyquistZPK: Added commands and tikz options 37
\parse@env@opt: Added tikz option to environments 36
\parse@N@opt: Added commands and tikz options 40
\parse@opt: Added Tikz option . . . 35
NyquistPlot: Added tikz option to environments 40
v1.0.4
General: Fixed unintended optional argument macro expansion 1
v1.0.5
\parse@opt: Fixed a bug 35
v1.0.6
General: Fixed issue #3 1
v1.0.7
General: Removed unnecessary semicolons 1
Updated documentation 1
v1.0.8
General: Added a new class option 'declutter' 1
\build@TF@plot: Included phase due to delay in wrapping. 34
gnuplot@prefix: Fixed issue #6 . . 21
v1.1.0
General: Fixed phase wrapping in gnuplot mode 1
\addBodeTFPlot: Fixed phase wrapping in gnuplot mode 29
BodeMagPlot: Added separate environments for phase and magnitude plots 31
BodePhPlot: Added separate environments for phase and magnitude plots 31
BodePlot: Deprecated BodePlot environment 32
\BodeTF: Fixed phase wrapping in gnuplot mode 27
v1.1.1
General: Enable Hz and rad units 1
\addBodeComponentPlot: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 30
\addBodeTFPlot: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 29
\addBodeZPKPlots: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 28
\addNicholsTFChart: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 47
\addNyquistTFPlot: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 40
\addNyquistZPKPlot: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 39
BodeMagPlot: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 31
BodePhPlot: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 31
BodePlot: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 32
\BodeTF: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 27
\BodeZPK: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 26
\build@TF@plot: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 34
get@interval@end: New macros to enable 'Hz' and 'rad' units for frequency and phase, respectively 23
ph@y@label: New macros to enable 'Hz' and 'rad' units for frequency and phase, respectively 22
\NyquistTF: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 38
\NyquistZPK: Enabled 'Hz' and 'rad' units for frequency and phase, respectively 37
v1.1.2
BodeMagPlot: Defined using the 'NewEnviron' command from the

'environ' package to fix conflicts with externalization 31
BodePhPlot: Defined using the 'NewEnviron' command from the 'environ' package to fix conflicts with externalization 31
BodePlot: Defined using the 'NewEnviron' command from the 'environ' package to fix conflicts with externalization 32
NicholsChart: Defined using the 'NewEniron' command from the 'environ' package to fix conflicts with externalization 47
\PhSOZerosLin: Fix scaling bug introduced in v1.1.1 25
NyquistPlot: Defined using the 'NewEniron' command from the 'environ' package to fix conflicts with externalization 40
v1.1.3
\addBodeComponentPlot: Changed implementation to respect user-supplied domain 30
\addBodeTFPlot: Changed implementation to respect user-supplied domain 29
\addBodeZPKPlots: Changed implementation to respect user-supplied domain 28
\addNicholsTFChart: Changed implementation to respect user-supplied domain 47
\addNicholsZPKChart: Changed implementation to respect user-supplied domain 47
\addNyquistTFPlot: Changed implementation to respect user-supplied domain 40
\addNyquistZPKPlot: Changed implementation to respect user-supplied domain 39
v1.1.4
\addBodeTFPlot: Changed phase wrapping in pgf mode 29
\addNicholsTFChart: Changed phase wrapping in pgf mode . . . 47
\BodeTF: Changed phase wrapping in pgf mode 27
gnuplot@prefix: Changed phase wrapping in pgf mode 21
v1.1.5
BodeMagPlot: Defined using the 'NewDocumentEnvironment' command from the 'xparse' package and added a hook to handle active characters 31
BodePhPlot: Defined using the 'NewDocumentEnvironment' command from the 'xparse' package and added a hook to handle active characters 31
BodePlot: Defined using the 'NewDocumentEnvironment' command from the 'xparse' package and added a hook to handle active characters 32
\BodeTF: Added code to handle active characters 28
\BodeZPK: Added code to handle active characters 27
NicholsChart: Defined using the 'NewDocumentEnvironment' command from the 'xparse' package and added a hook to handle active characters 47
\NicholsTF: Added code to handle active characters 47
\NicholsZPK: Added code to handle active characters 47
\NyquistTF: Added code to handle active characters 39
\NyquistZPK: Added code to handle active characters 38
NyquistPlot: Defined using the 'NewDocumentEnvironment' command from the 'xparse' package and added a hook to handle active characters 40
v1.1.6
\shorthand@list: Detect 'babel-french' using 'frenchbsetup' 21
v1.1.7
General: Detect and turn off shorthands to improve 'babel' compatibility 1
BodeMagPlot: Use auto-generated list of active characters instead of manually entering them. 31
BodePhPlot: Use auto-generated list of active characters instead of manually entering them. 31
BodePlot: Use auto-generated list of active characters instead of manually entering them 32
\BodeTF: Use auto-generated list of shorthands instead of manually specifying them 28
\BodeZPK: Use auto-generated list of shorthands instead of manually specifying them 27
NicholsChart: Use auto-generated list of active characters instead of manually entering them. 47
\NicholsTF: Use auto-generated list of shorthands instead of manually specifying them 47
\NicholsZPK: Use auto-generated list of shorthands instead of manually specifying them 47
\NyquistTF: Use auto-generated list of shorthands instead of manually specifying them 39
\NyquistZPK: Use auto-generated list of shorthands instead of manually

specifying them 38
\shorthand@list: Directly detect shorthands instead of detecting the language. 21
NyquistPlot: Use auto-generated list of active characters instead of manually entering them. 40
v1.2
General: Removed global option to process pgf commands in radians . 1
gnuplot@prefix: Removed global option to process pgf commands in radians 21
v2.0
General: Added pole zero maps 1
Added pole-zero map functionality 20
\PoleZeroMapZPK: Added pole-zero map functionality 50
v2.1
General: Added an environment and macros to add multiple transfer functions to a single Bode plot . . . 1
Added unified Bode plot macro supporting both ZPK and TF systems 14
Enhanced BodePlot environment with unified plot command collection 14
\addBodePlot: Added unified Bode plot macro supporting both ZPK and TF systems 30
BodePlot: Enhanced BodePlot environment with unified plot command collection 32
\build@gnu@plot: Added general gnuplot helper macro 37
\build@gnu@unwrap@plot: Added gnuplot helper macro with phase unwrapping 37
\parse@add@Bode@opt: Added option parser for unified Bode plot macro 37