The bodeplot package
version 3.1

Rushikesh Kamalapurkar
rlkamalapurkar@gmail.com

May 19, 2026

Contents

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.

All phase plots use degrees as units. Use the `rad` package option or the optional argument `phase unit=rad` to generate plots in radians. The `phase unit` key accepts either `rad` or `deg` as inputs.

Frequency inputs and outputs are in radians per second. Use the `Hz` package option or the optional argument `frequency unit=Hz` to generate plots in Hertz. The `frequency unit` key accepts either `rad` or `Hz` as inputs.

1.1 External Dependencies

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

1.2 Directory Structure

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

1.3 Limitations

- In `pgf` mode, Bode phase plots and Nichols charts in TF form wrap angles so that they are always between -180 and 180° or $-\pi$ and $-\pi$ 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.

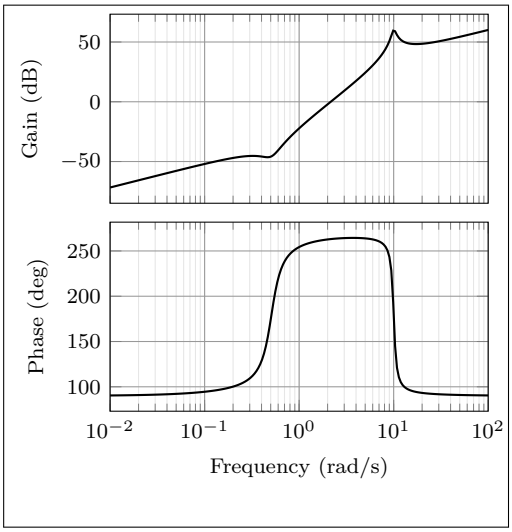
2 TL;DR

All Bode plots in this section are for the transfer function

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

Some of the examples include a transport delay. The examples use PGF-style key-value options and natural syntax for complex numbers in zeros and poles introduced in v3.0. For examples that show the legacy syntax (version 2.1.1 and earlier), please refer to Section A.

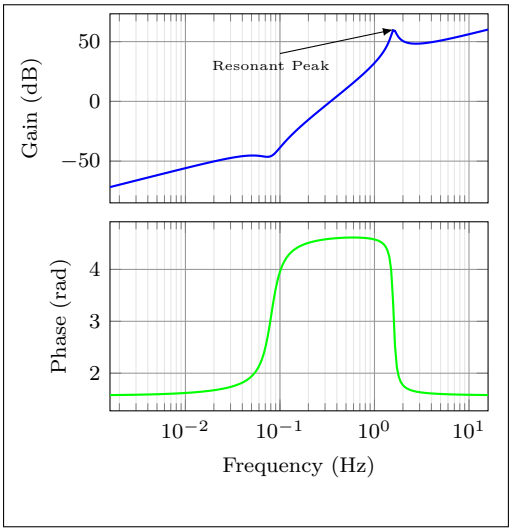
Bode plot in ZPK format



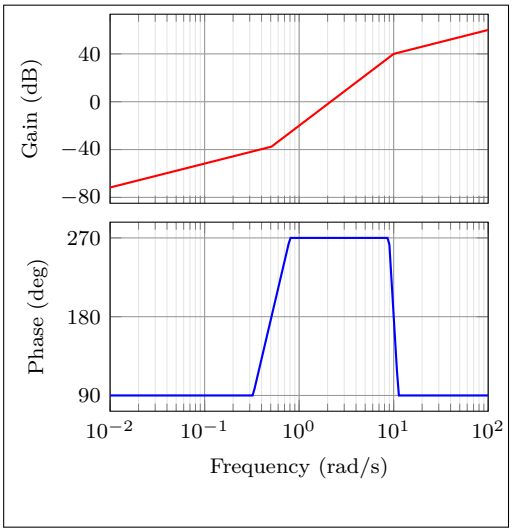

```
\BodeZPK[domain=0.01:100]{%
  zeros={0,-0.1-0.5i,-0.1+0.5i},
  poles={-0.5-10i,-0.5+10i},
  gain=10%
}
```

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



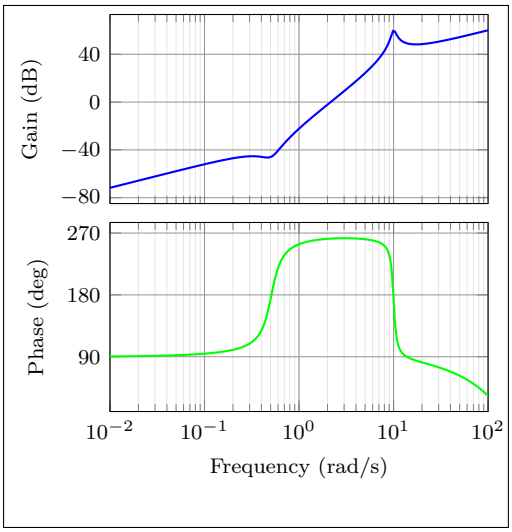
Linear approximation with customization



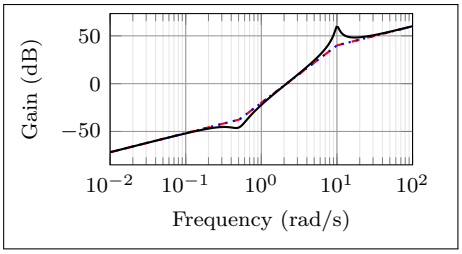

```
\BodeZPK[%
  domain=0.01:100,
  mag plot={red,thick},
  ph plot={blue,thick},
  mag axes={ytick distance=40},
  ph axes={ytick distance=90},
  approx=linear%
]{%
  zeros={0,-0.1-0.5i,-0.1+0.5i},
  poles={-0.5-10i,-0.5+10i},
  gain=10%
}
```

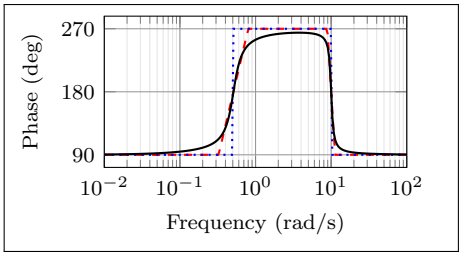
Plot with delay and customization

```
\BodeZPK[%
  domain=0.01:100,
  mag plot={blue,thick},
  ph plot={green,thick},
  mag axes={ytick distance=40},
  ph axes={ytick distance=90}%
]{%
  zeros={0,-0.1-0.5i,-0.1+0.5i},
  poles={-0.5-10i,-0.5+10i},
  gain=10,
  delay=0.01%
}
```



Individual gain and phase plots with more customization



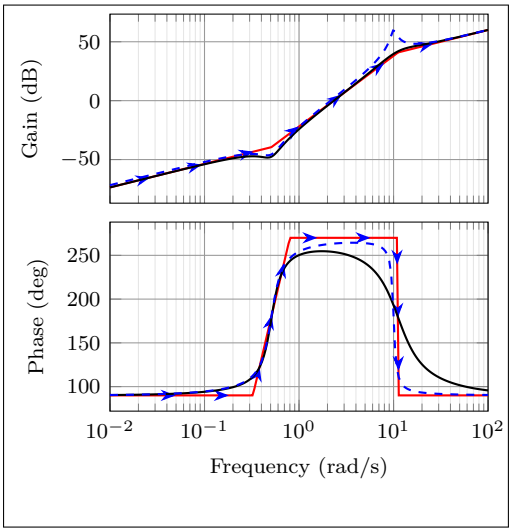



```
\begin{BodeMagPlot}[%
  axes={height=2cm,
  width=4cm},
  domain=0.01:100%
]
  \addBodeZPKPlots[%
    true={black,thick},
    linear={red,dashed,thick},
    asymptotic={blue,dotted,thick}%
  ]
  {magnitude}
  {%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-0.5-10i,-0.5+10i},
    gain=10%
  }
\end{BodeMagPlot}
```

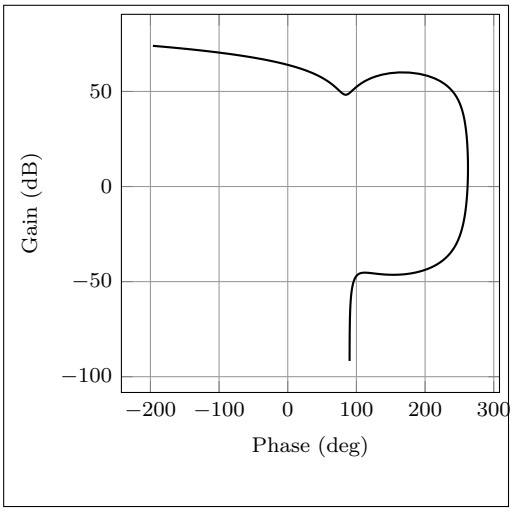
```
\begin{BodePhPlot}[%
  height=2cm,
  width=4cm,
  ytick distance=90,
  domain=0.01:100%
]
  \addBodeZPKPlots[%
    true={black,thick},
    linear={red,dashed,thick},
    asymptotic={blue,dotted,thick}%
  ]
  {phase}
  {%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-0.5-10i,-0.5+10i},
    gain=10%
  }
\end{BodePhPlot}
```

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

```
\begin{BodePlot}[domain=0.01:100]
  \addBodePlot[red,postaction=decorate,
    decoration={%
      markings,
      mark=between positions 0.1 and 0.9 step 2em with {%
        \arrow{Stealth [length=2mm, blue]}
      }
    },linear]{%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-5-10i,-5+10i},
    gain=10%
  }
  \addBodePlot[black,thick]{%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-5-10i,-5+10i},
    gain=10%
  }%
  \addBodePlot[blue,dashed]{%
    numerator={10,2,2.6,0},
    denominator={1,1,100.25}%
  }
\end{BodePlot}
```



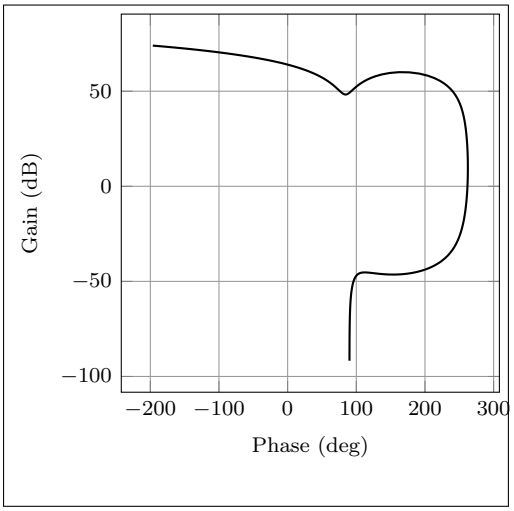
Nichols chart



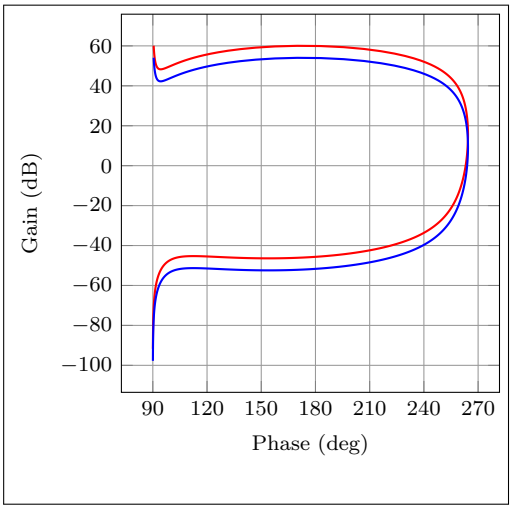

```
\NicholsZPK[domain=0.001:500,samples=1000]
{%
  zeros={0,-0.1-0.5i,-0.1+0.5i},
  poles={-0.5-10i,-0.5+10i},
  gain=10,
  delay=0.01%
}
```

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

```
\NicholsTF[domain=0.001:500,samples=1000]
{%
  numerator={10,2,2.6,0},
  denominator={1,1,100.25},
  delay=0.01%
}
```



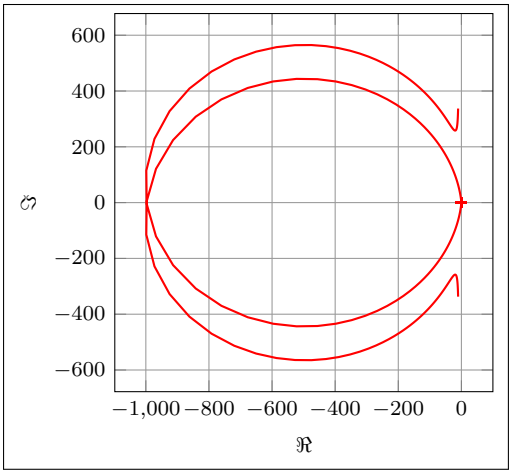
Multiple Nichols charts with customization



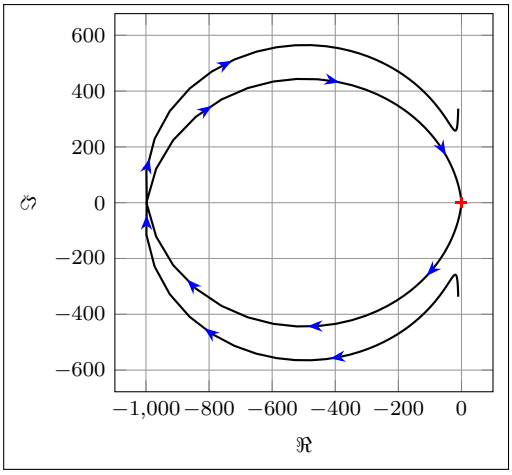

```
\begin{NicholsChart}[%
  ytick distance=20,
  xtick distance=30,
  domain=0.001:100%
]
  \addNicholsZPKChart [red,samples=1000] {%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-0.5-10i,-0.5+10i},
    gain=10%
  }
  \addNicholsZPKChart [blue,samples=1000] {%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-0.5-10i,-0.5+10i},
    gain=5%
  }
\end{NicholsChart}
```

Nyquist plot

```
\NyquistZPK[domain=-30:30,plot={red,thick,samples=1000}]
{%
  zeros={0,-0.1-0.5i,-0.1+0.5i},
  poles={-0.5-10i,-0.5+10i},
  gain=10%
}
```



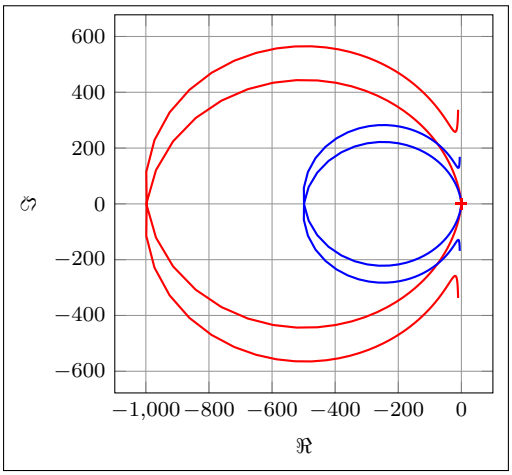
Nyquist plot in TF format with arrows



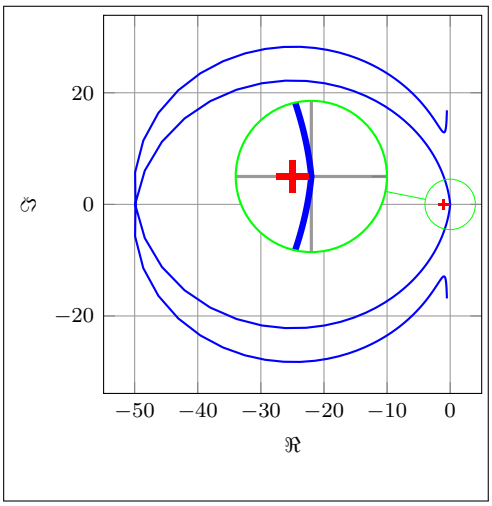

```
\NyquistTF[%
  domain=-30:30,
  plot={%
    samples=1000,
    postaction=decorate,
    decoration={%
      markings,
      mark=between positions 0.1 and 0.9 step 5em with {%
        \arrow{Stealth [length=2mm, blue]}
      }
    }
  }%
]
{%
  numerator={10,2,2.6,0},
  denominator={1,1,100.25}%
}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

Multiple Nyquist plots with customization

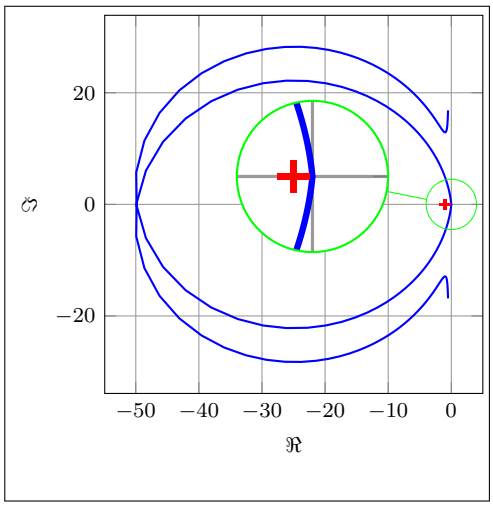
```
\begin{NyquistPlot}[domain=-30:30]
  \addNyquistZPKPlot [red,samples=1000] {%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-0.5-10i,-0.5+10i},
    gain=10%
  }
  \addNyquistZPKPlot [blue,samples=1000] {%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-0.5-10i,-0.5+10i},
    gain=5%
  }
\end{NyquistPlot}
```



Nyquist plots with additional commands, using two different macros

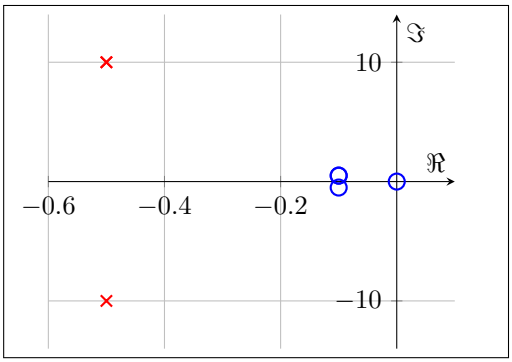



```
\begin{NyquistPlot}[%
  tikz={
    spy using outlines={%
      circle,
      magnification=3,
      connect spies,
      size=2cm
    }
  },
  domain=-30:30%
]
  \addNyquistZPKPlot [blue,samples=1000] {%
    zeros={0,-0.1-0.5i,-0.1+0.5i},
    poles={-0.5-10i,-0.5+10i},
    gain=0.5%
  }
  \coordinate (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);
  },
  domain=-30:30%
]
{%
  zeros={0,-0.1-0.5i,-0.1+0.5i},
  poles={-0.5-10i,-0.5+10i},
  gain=0.5%
}
```

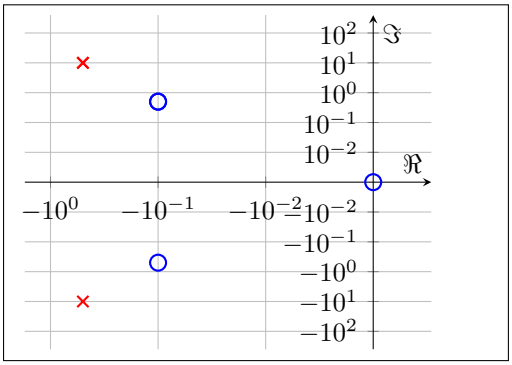
Pole-zero map




```
\PoleZeroMapZPK
{%
  zeros={0,-0.1-0.5i,-0.1+0.5i},
  poles={-0.5-10i,-0.5+10i},
  gain=10%
}
```

Pole-zero map (symmetric log scale)

```
\PoleZeroMapZPK[scale={log}]
{%
  zeros={0,-0.1-0.5i,-0.1+0.5i},
  poles={-0.5-10i,-0.5+10i},
  gain=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 `frequency unit=Hz` is supplied by the user. All phase plots are generated in degrees unless either the `rad` package option is used or the optional argument `phase unit=rad` is supplied by the user.

3.1 Bode plots

`\BodeZPK` `\BodeZPK[`⟨*options*⟩`]{`⟨*zpk-spec*⟩`}`

Plots the Bode plot of a transfer function given in ZPK format using the `groupplot` environment.

zpk-spec: A pgfkeys-style specification using keys `zeros={...}`, `poles={...}`, `gain=...`, and `delay=...` within the `/bodeplot/zpk/` family. Zeros and poles are comma-separated lists with complex poles and zeros entered using the format `a+bi`, where `a` and `b` can be arbitrary pgfmath/gnuplot expressions. The parser starts by detecting the trailing `i` and then finds the first top-level `+` or `-`. Everything between the `i` and the `+` or `-` is treated as the imaginary part, and everything before that is treated as the real part. If no `+` or `-` is found, the entire string except the trailing `i` is treated as the imaginary part and the real part is assumed to be zero. If the `gain` key is not specified, it is assumed to be 1. If the `delay` key is not specified, it is assumed to be zero.

options: The optional argument supports a comma-separated list of options in the form `key=value`. The following keys are supported:

- `domain=min:max`: Sets the frequency range, assumed to be `0.01:100` if not specified.
- `plot={opt}`: Options passed to `\addplot` for both magnitude and phase plots.
- `mag plot={opt}`: Options passed only to the magnitude plot.
- `ph plot={opt}`: Options passed only to the phase plot.
- `axes={opt}`: Options passed to `\nextgroupplot` for both plots.
- `mag axes={opt}`: Options passed only to magnitude axis.
- `ph axes={opt}`: Options passed only to phase axis.
- `group={opt}`: Options passed to the `groupplot` environment.
- `tikz={opt}`: Options passed to the `tikzpicture` environment.
- `commands={opt}`: TikZ commands added after both plots.
- `mag commands={opt}`: TikZ commands added after magnitude plot.
- `ph commands={opt}`: TikZ commands added after phase plot.
- `approx=`⟨*true*/*linear*/*asymptotic*⟩: Approximation type for the plot.
- `prefix={name}`: Prefix for gnuplot temporary files.
- `phase unit=`⟨*deg*/*rad*⟩: Phase axis units.
- `frequency unit=`⟨*rad*/*Hz*⟩: Frequency axis units.
- Any other key-value pair is passed as plot options to both plots.

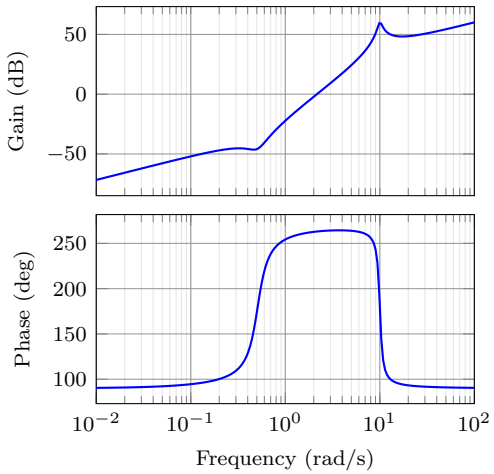


Figure 1: Output of the `\BodeZPK` macro.

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 [domain=0.01:100, blue, thick]
  {zeros={0, -0.1-0.5i, -0.1+0.5i},
   poles={-0.5-10i, -0.5+10i},
   gain=10}
```

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[%
  mag plot={red,thick},
  ph plot={blue,thick},
  mag axes={ytick distance=40,xmajorticks=true,xlabel={Frequency (rad/s)}},
  ph axes={ytick distance=90},
  group={group style={group size=2 by 1,horizontal sep=2cm,width=4cm,height=2cm}},
  approx=linear]
  {zeros={0,-0.1-0.5i,-0.1+0.5i},poles={-0.5-10i,-0.5+10i},gain=10}
```

which generates the plot in Figure 2.

Legacy interface (v2.1.1 and earlier):
`\BodeZPK [`⟨*options*⟩`]{`⟨*zpk-spec*⟩`}{`⟨*min-freq*⟩`}{`⟨*max-freq*⟩`}`

If both frequency arguments are provided, the macro falls back to the legacy format. This format is supported for backward compatibility with versions 2.1.1 and earlier:
zpk-spec (v2.1.1 and earlier): `z/{zeros},p/{poles},k/{gain},d/{delay}` where zeros and poles are comma-separated lists of complex numbers 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. This format requires the frequency arguments `min-freq` and `max-freq` to be provided.
options (v2.1.1 and earlier): A comma-separated list of options that modify the plots. The following options are supported:

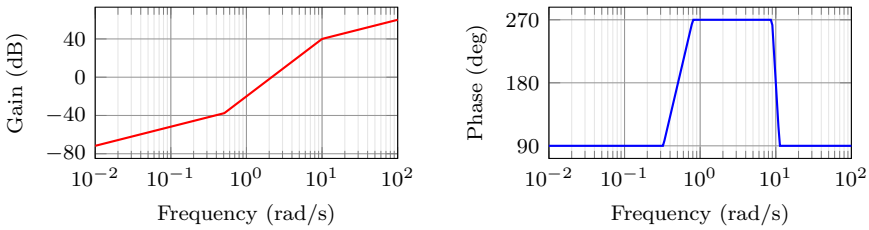


Figure 2: Customization of the default `\BodeZPK` macro.

- `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 to the magnitude plot if `typ` is `mag` and the phase plot if `typ` is `ph`.
- `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.
- `prefix/{opt}`: adds the string `opt` as a prefix to all temporary files generated by `gnuplot` for this plot.
- `{opt}` (bare options): add all of the supplied options to `\addplot` macros for both the magnitude and the phase plots.

Example of the legacy interface:

```
\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}
```

`\BodeTF` `\BodeTF[`⟨*options*⟩`]{`⟨*tf-spec*⟩`}`

Plots the Bode plot of a transfer function given in TF format.

tf-spec: The mandatory argument contains the pgf keys `numerator={coeffs}`, `denominator={coeffs}`, and optionally `delay=value`. The coefficients are entered as a comma-separated list, in order from the highest degree of s to the lowest, with zeros for missing degrees.

options: The frequency range is specified in the first optional argument using `domain=min:max`. The optional argument can also contain any of the keys supported by the `\BodeZPK` macro except for the `approx` key since linear/asymptotic approximation is not supported for TF format.

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

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

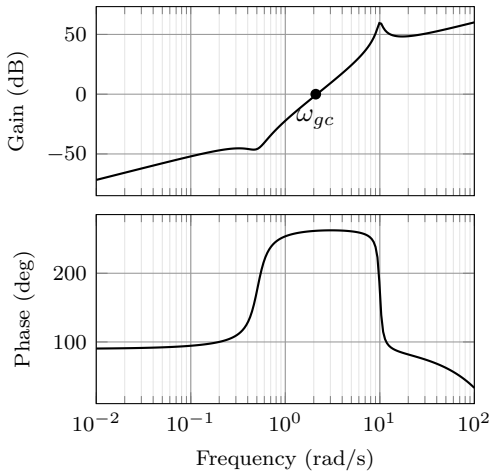


Figure 3: Output of the `\BodeTF` macro with an optional TikZ command used to mark the gain crossover frequency.

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

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

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.

Legacy interface (v2.1.1 and earlier):

`\BodeTF [`⟨*options*⟩`]{`⟨*tf-spec*⟩`}{`⟨*min-freq*⟩`}{`⟨*max-freq*⟩`}`

If both frequency arguments are provided, the macro falls back to the legacy 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 (format: `num/{coeffs},den/{coeffs},d/{delay}`), (2) the lower end of the frequency range, and (3) the upper end of the frequency range. The coefficients are entered as a comma-separated list, in order from the highest degree of s to the lowest, with zeros for missing degrees. The optional arguments are the same as `\bodeZPK` except that linear/asymptotic approximation is not supported, so `approx/...` is ignored. Example of the legacy interface:

```
\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}
```

`BodeMagPlot` (*env.*) `\begin{BodeMagPlot}[`⟨*options*⟩`]`

```
    \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 first optional argument can contain keys from the `/bodeplot/env/` family, including `domain=min:max` to set the frequency range, `axes={...}` for axis options, `tikz={...}` for tikzpicture options, and `prefix={...}` for gnuplot file prefixes. The `prefix` option adds the provided string as a prefix to all temporary files generated by `gnuplot` for all plots in this environment. This option is useful if data generated by gnuplot for a specific plot needs to be uniquely

identified. All other key-value pairs are passed as options to the `semilogaxis` environment. For examples, see the description of `\addBodeZPKPlots`, `\addBodeTFPlot`, and `\addBodeComponentPlot`.

Legacy interface (v2.1.1 and earlier):

`\begin{BodeMagPlot}[`⟨*options*⟩`]{`⟨*min-freq*⟩`}{`⟨*max-freq*⟩`}`
    `\addBode...`
`\end{BodeMagPlot}`

If both frequency arguments are provided, the environment uses the legacy format. The first 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.
  - `prefix/{opt}`: adds the string `opt` as a prefix to all temporary files generated by `gnuplot` for all plots in this environment. This option is useful if data generated by gnuplot for a specific plot needs to be uniquely identified.
- 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.

`BodePhPlot` (*env.*) `\begin{BodePhPlot}[`⟨*options*⟩`]`
    `\addBode...`
`\end{BodePhPlot}`

Same as `BodeMagPlot`, but intended to be used for phase plots.

`\addBodeZPKPlots` `\addBodeZPKPlots[`⟨*options*⟩`]{`⟨*plot-type*⟩`}{`⟨*zpk-spec*⟩`}`

Generates the appropriate parametric functions and supplies them to multiple `\addplot` macros, one for each approximation type specified in the optional argument. If no optional argument is supplied, then a single `\addplot` command corresponding to a thick true Bode plot is generated. The `options` argument accepts keys `true={`⟨*plot-opts*⟩`}`, `linear={`⟨*plot-opts*⟩`}`, and `asymptotic={`⟨*plot-opts*⟩`}`, where ⟨*plot-opts*⟩ are TikZ/pgfplots styling options. Multiple approximations can be specified as `true={black,thick},linear={red,dashed}`, etc. This macro can be used inside any `semilogaxis` environment as long as a domain for the x-axis is supplied in the environment or parent axis options. Use with the `BodeMagPlot` or `BodePhPlot` environments 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 third mandatory argument, `zpk-spec` is the same as that of the `\BodeZPK` macro described earlier.

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,domain=0.01:100,xlabel={}]
  \addBodeZPKPlots[true={black,thick},
    linear={red,dashed,thick},
    asymptotic={blue,dotted,thick}]
  {magnitude}
  {zeros={0,-0.1-0.5i,-0.1+0.5i},poles={-0.5-10i,-0.5+10i},gain=10}
\end{BodeMagPlot}
\begin{BodePhPlot}[height=2cm,width=4cm,ytick distance=90,domain=0.01:100]
  \addBodeZPKPlots[true={black,thick},
    linear={red,dashed,thick},
    asymptotic={blue,dotted,thick}]
  {phase}
  {zeros={0,-0.1-0.5i,-0.1+0.5i},poles={-0.5-10i,-0.5+10i},gain=10}
\end{BodePhPlot}
```

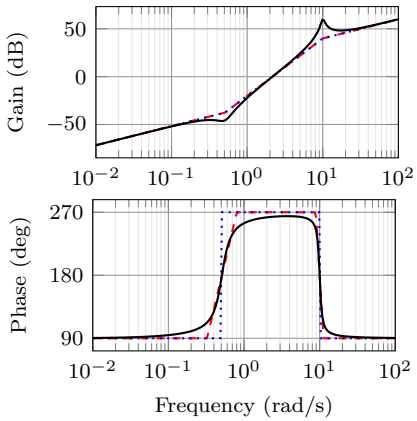


Figure 4: Superimposed approximate and true Bode plots using the `BodeMagPlot` and `BodePhPlot` environments and the `\addBodeZPKPlots` macro.

which generates the plot in Figure 4.
For backward compatibility, the legacy format with explicit frequency arguments is also supported.

`\addBodeZPKPlots[`⟨*options*⟩`]{`⟨*plot-type*⟩`}{`⟨*zpk-spec*⟩`}`

If an equals sign is not detected in the zpk-spec, then the macro falls back to the legacy format. In the legacy format, the macro 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}`. For example, the plots in Figure 4 can also be generated 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}
```

`\addBodeTFPlot` `\addBodeTFPlot[`⟨*options*⟩`]{`⟨*plot-type*⟩`}{`⟨*tf-spec*⟩`}`

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 the `options` argument or in the optional argument of the container `semilogaxis` environment. Use with the `BodeMagPlot` and `BodePhPlot` environments supplied with this package is recommended. The `options` can include any options that are accepted by the `\addplot` macro. The 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 supports both legacy format `num/{coeffs},den/{coeffs},d/{delay}` and new format (v3.0) using pgfkeys like

`numerator={...},denominator={...},delay=...`. The frequency range is specified using `domain=min:max` in the `options` argument.

`\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-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.** For ZPK systems, the `system-data` has the same format `zeros={...},poles={...},gain=..., delay=...` as `\addBodeZPKPlots`. For TF systems, it has the same format `numerator={...},denominator={...},gain=...,delay=...` 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. The macro automatically detects the system representation based on the presence of `zeros` or `poles` keys (ZPK) versus `numerator` or `denominator` keys (TF) in the `system-data` argument.

Legacy format (v2.1.1 and earlier):

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

If invoked using two mandatory arguments, the command falls back to the legacy format. In the legacy format, the command does not automatically detect the system representation. The second mandatory argument `system-type` should be either `zpk` or `tf` to specify the system representation.

`BodePlot` (*env.*)

```
\begin{BodePlot}[⟨options⟩]
    \addBodePlot...
\end{BodePlot}
```

Starting from version 2.1, the deprecated `BodePlot` environment is revived exclusively to host multiple instances of the unified `\addBodePlot` macro. The `\addBodeZPKPlots` and `\addBodeTFPlot` macros **will not function inside this environment.** The purpose of this environment is to plot a Bode plot that contains both phase and magnitude plots for several different transfer functions.

The environment uses the pgfkeys interface from the `/bodeplot/combinedenv/` family, supporting keys like `domain=min:max`, `group={...}`, `approx=...`, `mag axes={...}`, `ph axes={...}`, `mag commands={...}`, `ph commands={...}`, `tikz={...}`, and `prefix={...}` similar to the `\BodeZPK` macro.

For example, the Bode plot of two different transfer functions, one in ZPK form and the other in TF form, can be generated using:

```
\begin{BodePlot}[domain=0.01:100,group={group style={horizontal sep=2cm}}]
  \addBodePlot[true={black,thick},linear={red,dashed,thick},asymptotic={blue,dotted,thick}]
    {zeros={0,-0.1-0.5i,-0.1+0.5i},poles={-0.5-10i,-0.5+10i},gain=10}
  \addBodePlot[blue,thick]
    {numerator={10,2,2.6,0}, denominator={1,1,100.25}, delay=0.01}
\end{BodePlot}
```

which generates the plot in Figure 5.

Legacy interface (v2.1.1 and earlier):

```
\begin{BodePlot}[⟨options⟩]{⟨min-freq⟩}{⟨max-freq⟩}
    \addBodePlot...
\end{BodePlot}
```

If both frequency arguments are provided, the environment falls back to the legacy interface, where the optional arguments are the same as the legacy interface for the

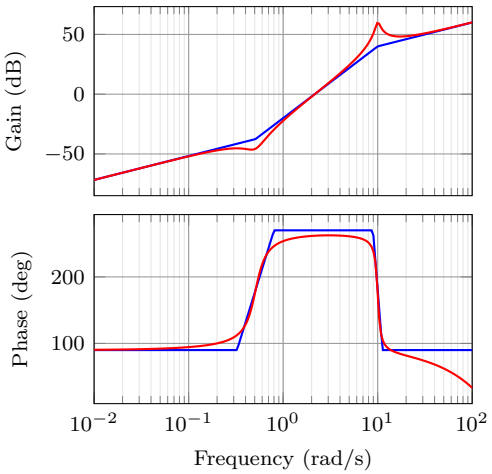


Figure 5: Bode plot combining ZPK and TF systems using the `BodePlot` environment and the unified `\addBodePlot` macro.

`\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 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 $G(s) = \frac{1}{s-a-\mathrm{i}b}$.
- `\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,domain=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] {%
```

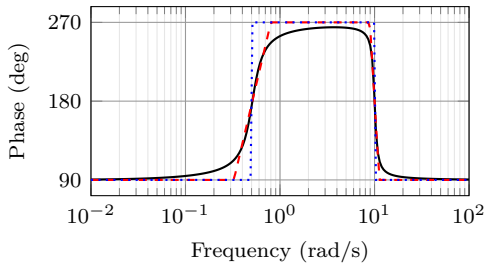


Figure 6: Superimposed approximate and true Bode Phase plot using the `BodePhPlot` environment, the `\addBodeComponentPlot` macro, and several macros of the `\TypeFeatureApprox` form.

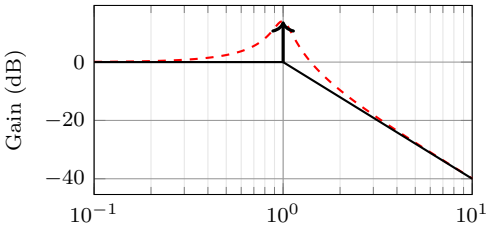


Figure 7: Resonant peak in asymptotic Bode plot using `\MagSOPolesPeak`.

```
    \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 6.

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*⟩`}`

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={},domain=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 7.

`\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 [`⟨*options*⟩`]{`⟨*zpk-spec*⟩`}`
Generates the Nyquist diagram of a transfer function given in ZPK form and marks the critical point $(-1, 0)$ with a thick red $+$.
zpk-spec: Same as the `\BodeZPK` macro. Provide zeros, poles, gain, and optional delay with the pgfkeys-style format `zeros={...},poles={...},gain=...,delay=...`. Complex numbers are written as `a+bi`, where `a` and `b` can be arbitrary pgfmath/gnuplot expressions. The macro assumes unit gain and zero delay when those keys are omitted.
options:

- `domain=min:max`: Sets the frequency sweep for the contour. The default sweep is `-30:30` when the key is not provided.
- `plot={opt}`: Appends `opt` to the `\addplot` options that draw the Nyquist curve, e.g., `samples=2000`.
- `axes={opt}`: Adds `opt` to the surrounding `axis` environment.
- `commands={opt}`: Inserts `opt` immediately after the contour inside the `axis`, which is useful for annotations or additional plotting commands.
- `tikz={opt}`: Adds `opt` to the enclosing `tikzpicture`.
- `prefix={name}`: Overrides the prefix used for auxiliary gnuplot tables.

Any additional `key=value` entries are forwarded to `\addplot`, so native `pgfplots` options may be supplied without wrapping them in `plot={...}`. Linear or asymptotic approximations are not available for Nyquist plots. To indicate the direction of increasing frequency, load `decorations.markings` and `arrows.meta` and add `plot={postaction=decorate,decoration={...}}`. Very large sample counts in combination with decorations can trigger the warning `! Dimension too big`.
For example, the command

```
\NyquistZPK[
  domain=-30:30,
  plot={red,thick,samples=2000},
  axes={blue,thick}]
  {zeros={0,-0.1-0.5i,-0.1+0.5i},
   poles={-0.5-10i,-0.5+10i},
   gain=10}
```

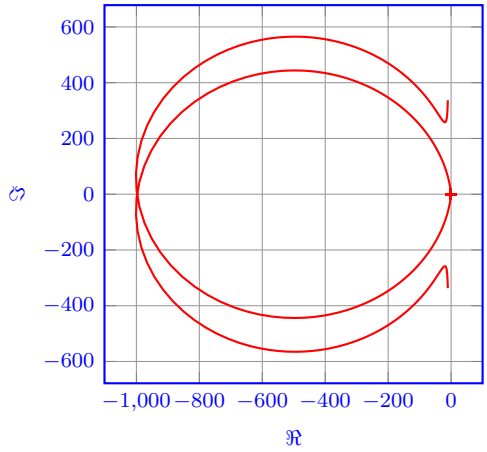


Figure 8: Output of the `\NyquistZPK` macro.

generates the Nyquist plot in Figure 8.

Legacy interface (v2.1.1 and earlier):

`\NyquistZPK[`⟨*options*⟩`]{`⟨*zpk-spec*⟩`}{`⟨*min-freq*⟩`}{`⟨*max-freq*⟩`}`

The optional argument accepts comma-separated tuples `plot/{opt}`, `axes/{opt}`, `commands/{opt}`, `tikz/{opt}`, and `prefix/{opt}`, while a bare `{opt}` is treated as `plot/{opt}`. The `zpk-spec` uses the legacy format `z/{...},p/{...},k/...,d/...`, where complex numbers are expressed as `{a,b}`. Supplying the frequency arguments auto-selects the legacy behaviour and is intended only for maintaining older documents. Example of legacy usage:

```
\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}
```

`\NyquistTF` `\NyquistTF[`⟨*options*⟩`]{`⟨*tf-spec*⟩`}`

Generates the Nyquist diagram of a transfer function expressed in polynomial form. The `tf-spec` is the same as that of the `\BodeTF` macro described earlier and the `options` argument uses the same pgfkeys interface as `\NyquistZPK`. A legacy interface similar to the one in `\NyquistZPK` is also supported when both frequency arguments are provided. For example,

```
\NyquistTF[
  domain=-30:30,
  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]}}}}]
  {numerator={10,2,2.6,0}, denominator={1,1,100.25}}
```

yields the Nyquist plot in Figure 9. As with the ZPK variant, combining dense sampling with `decorations.markings` may trigger the warning `! Dimension too big`. Legacy syntax remains available:

```
\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}
```

`NyquistPlot` (*env.*) `\begin{NyquistPlot}[`⟨*options*⟩`]`

```
    \addNyquist...
\end{NyquistPlot}
```

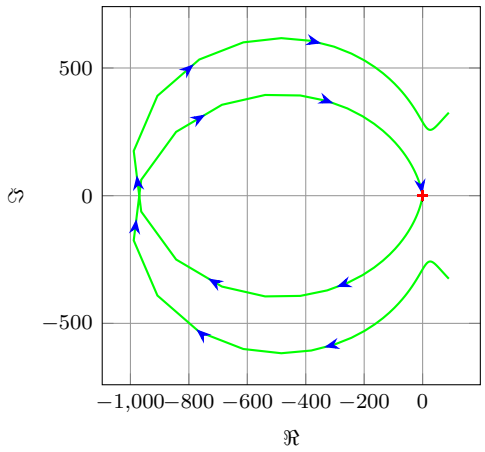


Figure 9: Output of the `\NyquistTF` macro with direction arrows. Increasing the number of samples can cause `decorations.markings` to throw errors.

Wraps a `tikzpicture`/`axis` pair tailored for Nyquist diagrams and hosts one or more `\addNyquist...` commands.

options:

- `domain=min:max`: Sets the frequency sweep that is forwarded to the enclosed `\addplot` commands. The default sweep is `0.01:100`; choose a symmetric interval such as `-30:30` for traditional Nyquist diagrams.
- `axes={opt}`: Appends `opt` to the `axis` options (for example, `axis equal` or `grid=both`).
- `tikz={opt}`: Adds `opt` to the surrounding `tikzpicture`.
- `prefix={name}`: Overrides the prefix used for auxiliary gnuplot tables created inside the environment.
- `phase unit=deg/rad` and `frequency unit=rad/Hz`: Synchronises the environment with the package-wide unit switches.
- Additional `key=value` pairs are passed straight to the `axis` environment, so native `pgfplots` options can be provided without extra wrapping.

The environment marks the critical point $(-1, 0)$ with a red `+` by default. Pair it with `\addNyquistZPKPlot` or `\addNyquistTFPlot` to draw the actual contour.

Legacy interface (v2.1.1 and earlier):

`\begin{NyquistPlot}[`⟨*options*⟩`]{`⟨*min-freq*⟩`}{`⟨*max-freq*⟩`}`
    `\addNyquist...`
`\end{NyquistPlot}`

The legacy optional argument accepts comma-separated tuples `obj/{opt}` or bare `{opt}`. Valid `obj` tokens are `tikz`, `axes`, and `prefix`; unqualified entries are appended to the `axis` options. The mandatory frequency limits fix the axis and plot domain.

`\addNyquistZPKPlot` `\addNyquistZPKPlot[`⟨*plot-options*⟩`]{`⟨*zpk-spec*⟩`}`

Adds a Nyquist contour for a transfer function supplied in ZPK form.

zpk-spec: Same as the `\NyquistZPK` macro.

plot-options: Appended verbatim to `\addplot`, enabling control over sampling density, decorations, colors, and so on. Ensure that the surrounding `axis` (typically provided by `\begin{NyquistPlot}`) defines a suitable `domain` so that the generated parametric function matches the frequency sweep.

Legacy interface (v2.1.1 and earlier):

`\addNyquistZPKPlot[`⟨*plot-options*⟩`]{`⟨*z/{…},p/{…},k/…,d/…*⟩`}`

The legacy format is preserved for backward compatibility; the optional argument continues to be forwarded directly to `\addplot`.

`\addNyquistTFPlot` `\addNyquistTFPlot[`⟨*plot-options*⟩`]{`⟨*tf-spec*⟩`}`
Identical to `\addNyquistZPKPlot` but expects `tf-spec` in the same polynomial format accepted by `\BodeTF`.

3.3 Nichols charts

`\NicholsZPK` `\NicholsZPK [`⟨*options*⟩`]{`⟨*zpk-spec*⟩`}`
Generates a Nichols chart (phase versus gain) for a transfer function in ZPK form.

The `zpk-spec` and `options` arguments are the same as those of the `\BodeZPK` macro described earlier except that the `domain=min:max` key now sets the frequency sweep for the Nichols contour (default `0.01:100`).

Supplying both frequency arguments activates the legacy behaviour for backward compatibility.
`\NicholsZPK [`⟨*options*⟩`]{`⟨*zpk-spec*⟩`}{`⟨*min-freq*⟩`}{`⟨*max-freq*⟩`}`
Example:

```
\NicholsZPK[
  domain=0.001:100,
  plot={red,thick,samples=2000},
  axes={ytick distance=20}]
  {zeros={0,-0.1-0.5i,-0.1+0.5i},
   poles={-0.5-10i,-0.5+10i},
   gain=10}
```

Legacy usage remains available:

```
\NicholsZPK[plot/{red,thick,samples=2000}]
  {z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{{-0.5,-10},{-0.5,10}},k/10}
  {0.001}{100}
```

`\NicholsTF` `\NicholsTF [`⟨*options*⟩`]{`⟨*tf-spec*⟩`}`
Produces a Nichols chart for transfer functions expressed in polynomial form. Same as `\NicholsZPK` except that `tf-spec` is akin to `\BodeTF`. Supplying both frequency arguments reverts the macro to the legacy interface.
Example:

```
\NicholsTF[
  domain=0.001:100,
  plot={green,thick,samples=2000}]
  {numerator={10,2,2.6,0},
   denominator={1,1,100.25},
   delay=0.01}
```

Legacy usage remains available:

```
\NicholsTF[plot/{green,thick,samples=2000}]
  {num/{10,2,2.6,0},den/{1,1,100.25},d/0.01}
  {0.001}{100}
```

Both examples produce the chart in Figure 10.

`NicholsChart` (*env.*) `\begin{NicholsChart}[`⟨*options*⟩`]`
`\addNichols...`
`\end{NicholsChart}`
Provides a ready-made `tikzpicture`/`axis` scaffold for Nichols charts and collects `\addNichols...` commands. Current and legacy interfaces are similar to the `NyquistPlot` environment, except that the domain defaults to `0.01:100` if not supplied.

`\addNicholsZPKChart` `\addNicholsZPKChart[`⟨*plot-options*⟩`]{`⟨*zpk-spec*⟩`}`
Adds a Nichols contour for a transfer function expressed in ZPK form. Current and legacy interfaces are similar to those of `\addNyquistZPKPlot`, with the `domain` key now setting the frequency sweep for the Nichols contour, with a default of `0.01:100`.

`\addNicholsTFChart` `\addNicholsTFChart[`⟨*plot-options*⟩`]{`⟨*tf-spec*⟩`}`
Equivalent to `\addNicholsZPKChart` for transfer functions supplied in polynomial form similar to `\BodeTF`. Phase unwrapping in `gnuplot` mode mirrors the behaviour of `\BodeTF`.

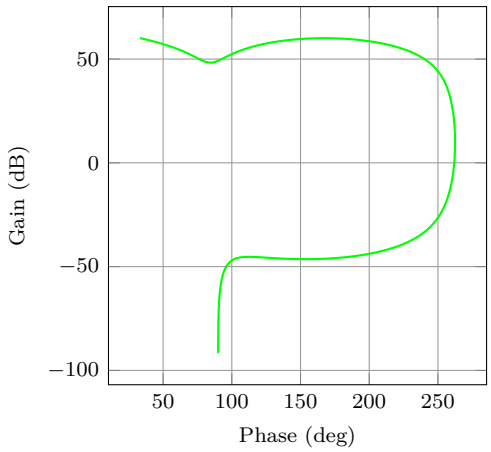


Figure 10: Output of the `\NicholsTF` macro.

3.4 Pole-zero maps

`\PoleZeroMapZPK` `\PoleZeroMapZPK [`⟨*options*⟩`]{`⟨*zpk-spec*⟩`}`

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 supports both the current `zeros={...},poles={...},gain=...` format with complex numbers expressed as `a+bi` (where `a` and `b` can be arbitrary pgfmath/gnuplot expressions) and the legacy format `z/{zeros},p/{poles},k/{gain}` with complex numbers expressed as `{a,b}`. Note that the delay parameter `d` is ignored since delays do not affect pole-zero locations. The optional argument supports the same tuples as `\NyquistZPK`.

The `scale={log}` 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.

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

```
\PoleZeroMapZPK[axes/{grid=major}]
 {zeros={0, -0.1-0.5i, -0.1+0.5i}, poles={-0.5-10i, -0.5+10i}, gain=10}
```

For logarithmic scaling:

```
\PoleZeroMapZPK[scale=log]
 {zeros={0, -0.1-0.5i, -0.1+0.5i}, poles={-0.5-10i, -0.5+10i}, gain=10}
```

The legacy format remains available:

```
\PoleZeroMapZPK[axes/{grid=major}]
  {z/{0,{-0.1,-0.5},{-0.1,0.5}},p/{{-0.5,-10},{-0.5,10}},k/10}
\PoleZeroMapZPK[scale/log]
 {zeros={0, -0.1-0.5i, -0.1+0.5i}, poles={-0.5-10i, -0.5+10i}, gain=10}
```

Both examples produce the pole-zero maps in Figure 11.

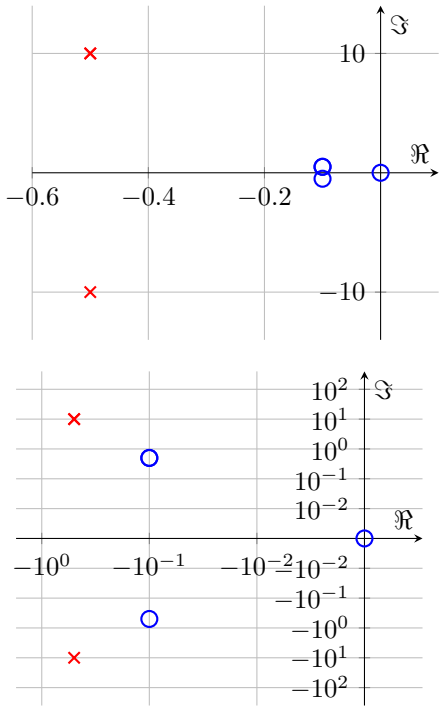


Figure 11: Pole-zero maps generated using the `\PoleZeroMapZPK` macro with linear and logarithmic scaling.

4 Implementation

4.1 Initialization

\n@mod
\n@mod@p
\n@mod@n
\n@pow
bp@gnuplot@id
bp@gnu@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
```

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)}
```

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. The `gnuplot` prefix is set to be the name of the tex file unless the user supplies a prefix through the `prefix` option.

```
  \newcounter{bp@gnuplot@id}
  \setcounter{bp@gnuplot@id}{0}
  \newcommand{\bp@prefix}{%
    \ifx\bp@user@prefix\@empty
      \if@declutterarg
        gnuplot/\jobname
      \else
        \jobname
      \fi
    \else
      \if@declutterarg
        gnuplot/\bp@user@prefix
      \else
        \bp@user@prefix
      \fi
    \fi
  }
  \tikzset{
    bp@gnu@prefix/.style={
      id=\arabic{bp@gnuplot@id},
      prefix=\bp@prefix
    }
  }
```

If the operating system is not Windows, and if the `declutter` option is 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` `\bp@short@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\bp@short@list\@empty
    \def\do#1{%
      \begingroup
        \lccode`\~=`#1\relax
        \lowercase{\ifbabelshorthand~{\g@addto@macro\bp@short@list{~}}{}}
      \endgroup
    }
    \dospecials
  }{}
}
```

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

```
\pgfplotsset{
  bp@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
  }
}
```

`bp@freq@filter` `bp@freq@label` `\bp@freq@scale` `\bp@ph@scale` `bp@ph@x@label` `bp@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{bp@freq@filter/.style = {}}
\def\bp@freq@scale{1}
\pgfplotsset{bp@freq@label/.style = {xlabel = {Frequency (rad/s)}}}
\pgfplotsset{bp@ph@x@label/.style = {xlabel={Phase (deg)}}}
\pgfplotsset{bp@ph@y@label/.style = {ylabel={Phase (deg)}}}
\def\bp@ph@scale{180/pi}
\if@radarg
  \pgfplotsset{bp@ph@y@label/.style = {ylabel={Phase (rad)}}}
  \pgfplotsset{bp@ph@x@label/.style = {xlabel={Phase (rad)}}}
  \def\bp@ph@scale{1}
  \tikzset{
    phase unit/.initial={rad},
    phase unit/.default={rad},
  }
\else
  \tikzset{
    phase unit/.initial={deg},
    phase unit/.default={deg},
  }
```

```
\fi
\if@hzarg
  \def\bp@freq@scale{2*pi}
  \pgfplotsset{bp@freq@label/.style = {xlabel = {Frequency (Hz)}}}
  \if@pgfarg
    \pgfplotsset{bp@freq@filter/.style = {x filter/.expression={x-
  log10(2*pi)}}}
  \fi
  \tikzset{
    frequency unit/.initial={Hz},
    frequency unit/.default={Hz},
  }
\else
  \tikzset{
    frequency unit/.initial={rad},
    frequency unit/.default={rad},
  }
\fi
\tikzset{
  phase unit/.is choice,
    phase unit/deg/.code={
      \pgfkeys{/bodeplot/phase unit=deg}
    },
    phase unit/rad/.code={
      \pgfkeys{/bodeplot/phase unit=rad}
    },
    frequency unit/.is choice,
    frequency unit/Hz/.code={
      \pgfkeys{/bodeplot/frequency unit=Hz}
    },
    frequency unit/rad/.code={
      \pgfkeys{/bodeplot/frequency unit=rad}
    }
}
```

4.2 PGF keys interface

`/bodeplot/env` PGF keys for environment options.

```
\pgfkeys{
  /bodeplot/env/.is family,
  /bodeplot/env/.cd,
  reset/.code={%
    \pgfkeyssetvalue{/bodeplot/env/@tikz}{}
    \pgfkeyssetvalue{/bodeplot/env/@prefix}{}
    \pgfkeyssetvalue{/bodeplot/env/@domain}{}
    \if@hzarg
      \pgfkeys{/bodeplot/env/frequency unit=Hz}
    \else
      \pgfkeys{/bodeplot/env/frequency unit=rad}
    \fi
    \if@radarg
      \pgfkeys{/bodeplot/env/phase unit=rad}
    \else
      \pgfkeys{/bodeplot/env/phase unit=deg}
    \fi
    \gdef\bp@domain@start{0.01}
    \gdef\bp@domain@end{100}
    \gdef\bp@axes{}
  },
  axes/.code={
    \ifx\bp@axes\@empty
      \xdef\bp@axes{\unexpanded\expandafter{#1}}
    \else
```

```
      \xdef\bp@axes{\unexpanded\expandafter{\bp@axes},
      \unexpanded\expandafter{#1}}
    \fi
  },
  axes/.value required,
  tikz/.code={\pgfkeyssetvalue{/bodeplot/env/@tikz}{#1}},
  tikz/.value required,
  prefix/.code={\pgfkeyssetvalue{/bodeplot/env/@prefix}{#1}},
  prefix/.value required,
  domain/.code args={#1:#2}{%
    \gdef\bp@domain@start{#1}%
    \gdef\bp@domain@end{#2}%
  },
  domain/.value required,
  phase unit/.initial={deg},
  phase unit/.default={deg},
  phase unit/.is choice,
  phase unit/deg/.code={
    \renewcommand{\bp@ph@scale}{180/pi}
    \pgfplotsset{bp@ph@x@label/.style = {xlabel={Phase (deg)}}}
    \pgfplotsset{bp@ph@y@label/.style = {ylabel={Phase (deg)}}}
  },
  phase unit/rad/.code={
    \renewcommand{\bp@ph@scale}{1}
    \pgfplotsset{bp@ph@y@label/.style = {ylabel={Phase (rad)}}}
    \pgfplotsset{bp@ph@x@label/.style = {xlabel={Phase (rad)}}}
  },
  frequency unit/.initial={rad},
  frequency unit/.default={rad},
  frequency unit/.is choice,
  frequency unit/Hz/.code={
    \renewcommand{\bp@freq@scale}{2*pi}
    \pgfplotsset{bp@freq@label/.style = {xlabel = {Frequency (Hz)}}}
    \if@pgfarg
      \pgfplotsset{bp@freq@filter/.style = {x filter/.expression={x-
log10(2*pi)}}}
    \fi
  },
  frequency unit/rad/.code={
    \renewcommand{\bp@freq@scale}{1}
    \pgfplotsset{bp@freq@label/.style = {xlabel = {Frequency (rad/s)}}}
    \if@pgfarg
      \pgfplotsset{bp@freq@filter/.style = {x filter/.expression={x}}}
    \fi
  },
  .unknown/.code={%
    \edef\bp@full{\pgfkeyscurrentkey}%
    \def\stripbodeprefix##1/bodeplot/env/##2\relax{##2}%
    \edef\bp@short{\expandafter\stripbodeprefix\bp@full\relax}%
    \edef\bp@checkfull{\bp@full}%
    \edef\bp@checkshort{\bp@short}%
    \ifx\bp@checkfull\bp@checkshort
      \def\removeslash##1/##2\relax{##2}%
      \edef\bp@short{\expandafter\removeslash\bp@full\relax}%
    \fi
    \ifx\pgfkeyscurrentvalue\pgfkeysnovalue
      \edef\bp@new{\bp@short}%
    \else
      \edef\bp@new{\bp@short=
      {\unexpanded\expandafter{\pgfkeyscurrentvalue}}}%
    \fi
    \ifx\bp@new\@empty\else
      \ifx\bp@axes\@empty
```

```
        \xdef\bp@axes{\unexpanded\expandafter{\bp@new}}
      \else
        \xdef\bp@axes{\unexpanded\expandafter{\bp@axes},%
        \unexpanded\expandafter{\bp@new}}
      \fi
    \fi
  }
}
```

`/bodeplot PGF` keys for bode command options.

```
\pgfkeys{
  /bodeplot/.is family,
  /bodeplot/.cd,
  reset/.code={%
    \pgfkeyssetvalue{/bodeplot/@axes/mag}{}
    \pgfkeyssetvalue{/bodeplot/@axes/ph}{}
    \pgfkeyssetvalue{/bodeplot/@group}{}
    \pgfkeyssetvalue{/bodeplot/@approx}{true}
    \pgfkeyssetvalue{/bodeplot/@commands/mag}{}
    \pgfkeyssetvalue{/bodeplot/@commands/ph}{}
    \pgfkeyssetvalue{/bodeplot/@tikz}{}
    \pgfkeyssetvalue{/bodeplot/@prefix}{}
    \gdef\bp@domain@start{0.01}
    \gdef\bp@domain@end{100}
    \if@hzarg
      \pgfkeys{/bodeplot/frequency unit=Hz}
    \else
      \pgfkeys{/bodeplot/frequency unit=rad}
    \fi
    \if@radarg
      \pgfkeys{/bodeplot/phase unit=rad}
    \else
      \pgfkeys{/bodeplot/phase unit=deg}
    \fi
    \gdef\bp@mag@plot{}
    \gdef\bp@ph@plot{}
  },
  plot/.code={%
    \ifx\bp@mag@plot\@empty
      \xdef\bp@mag@plot{\unexpanded\expandafter{#1}}
    \else
      \xdef\bp@mag@plot{\unexpanded\expandafter{\bp@mag@plot,#1}}
    \fi
    \ifx\bp@ph@plot\@empty
      \xdef\bp@ph@plot{\unexpanded\expandafter{#1}}
    \else
      \xdef\bp@ph@plot{\unexpanded\expandafter{\bp@ph@plot,#1}}
    \fi
  },
  plot/.value required,
  mag plot/.code={%
    \ifx\bp@mag@plot\@empty
      \xdef\bp@mag@plot{\unexpanded\expandafter{#1}}
    \else
      \xdef\bp@mag@plot{\unexpanded\expandafter{\bp@mag@plot},
      \unexpanded\expandafter{#1}}
    \fi
  },
  mag plot/.value required,
  ph plot/.code={%
    \ifx\bp@ph@plot\@empty
      \xdef\bp@ph@plot{\unexpanded\expandafter{#1}}
    \else
```

```
      \xdef\bp@ph@plot{\unexpanded\expandafter{\bp@ph@plot},
      \unexpanded\expandafter{#1}}
    \fi
  },
  ph plot/.value required,
  axes/.code={%
    \pgfkeysalso{/bodeplot/mag axes={#1}}
    \pgfkeysalso{/bodeplot/ph axes={#1}}%
  },
  axes/.value required,
  mag axes/.code={\pgfkeyssetvalue{/bodeplot/@axes/mag}{#1}},
  mag axes/.value required,
  ph axes/.code={\pgfkeyssetvalue{/bodeplot/@axes/ph}{#1}},
  ph axes/.value required,
  group/.code={\pgfkeyssetvalue{/bodeplot/@group}{#1}},
  group/.value required,
  approx/.code={\pgfkeyssetvalue{/bodeplot/@approx}{#1}},
  approx/.value required,
  commands/.code={%
    \pgfkeysalso{/bodeplot/mag commands={#1}}
    \pgfkeysalso{/bodeplot/ph commands={#1}}%
  },
  commands/.value required,
  mag commands/.code={\pgfkeyssetvalue{/bodeplot/@commands/mag}{#1}},
  mag commands/.value required,
  ph commands/.code={\pgfkeyssetvalue{/bodeplot/@commands/ph}{#1}},
  ph commands/.value required,
  tikz/.code={\pgfkeyssetvalue{/bodeplot/@tikz}{#1}},
  tikz/.value required,
  prefix/.code={\pgfkeyssetvalue{/bodeplot/@prefix}{#1}},
  prefix/.value required,
  domain/.code args={#1:#2}{%
    \gdef\bp@domain@start{#1}%
    \gdef\bp@domain@end{#2}%
  },
  domain/.value required,
  phase unit/.initial={deg},
  phase unit/.default={deg},
  phase unit/.is choice,
  phase unit/deg/.code={
    \renewcommand{\bp@ph@scale}{180/pi}
    \pgfplotsset{bp@ph@x@label/.style = {xlabel={Phase (deg)}}}
    \pgfplotsset{bp@ph@y@label/.style = {ylabel={Phase (deg)}}}
  },
  phase unit/rad/.code={
    \renewcommand{\bp@ph@scale}{1}
    \pgfplotsset{bp@ph@y@label/.style = {ylabel={Phase (rad)}}}
    \pgfplotsset{bp@ph@x@label/.style = {xlabel={Phase (rad)}}}
  },
  frequency unit/.initial={rad},
  frequency unit/.default={rad},
  frequency unit/.is choice,
  frequency unit/Hz/.code={
    \renewcommand{\bp@freq@scale}{2*pi}
    \pgfplotsset{bp@freq@label/.style = {xlabel = {Frequency (Hz)}}}
    \if@pgfarg
      \pgfplotsset{bp@freq@filter/.style = {x filter/.expression={x-
log10(2*pi)}}}
    \fi
  },
  frequency unit/rad/.code={
    \renewcommand{\bp@freq@scale}{1}
    \pgfplotsset{bp@freq@label/.style = {xlabel = {Frequency (rad/s)}}}
```

```
    \if@pgfarg
      \pgfplotsset{bp@freq@filter/.style = {x filter/.expression={x}}}
    \fi
  },
  .unknown/.code={%
    \edef\bp@full{\pgfkeyscurrentkey}%
    \def\stripbodeprefix##1/bodeplot/##2\relax{##2}%
    \edef\bp@short{\expandafter\stripbodeprefix\bp@full\relax}%
    \edef\bp@checkfull{\bp@full}%
    \edef\bp@checkshort{\bp@short}%
    \ifx\bp@checkfull\bp@checkshort
      \def\removeslash##1/##2\relax{##2}%
      \edef\bp@short{\expandafter\removeslash\bp@full\relax}%
    \fi
    \ifx\pgfkeyscurrentvalue\pgfkeysnovalue
      \edef\bp@new{\bp@short}%
    \else
      \edef\bp@new{\bp@short=
      {\unexpanded\expandafter{\pgfkeyscurrentvalue}}}%
    \fi
    \ifx\bp@new\@empty\else
      \ifx\bp@mag@plot\@empty
        \xdef\bp@mag@plot{\unexpanded\expandafter{\bp@new}}
      \else
        \xdef\bp@mag@plot{\unexpanded\expandafter{\bp@mag@plot},%
        \unexpanded\expandafter{\bp@new}}
      \fi
    \fi
    \ifx\bp@new\@empty\else
      \ifx\bp@ph@plot\@empty
        \xdef\bp@ph@plot{\unexpanded\expandafter{\bp@new}}
      \else
        \xdef\bp@ph@plot{\unexpanded\expandafter{\bp@ph@plot},%
        \unexpanded\expandafter{\bp@new}}
      \fi
    \fi
  }
}
```

/bodeplot/combinedenv PGF keys for combined Bode environment options.

```
\pgfkeys{
  /bodeplot/combinedenv/.is family,
  /bodeplot/combinedenv/.cd,
  reset/.code={%
    \pgfkeyssetvalue{/bodeplot/combinedenv/@group}{}
    \pgfkeyssetvalue{/bodeplot/combinedenv/@approx}{true}
    \pgfkeyssetvalue{/bodeplot/combinedenv/@commands/mag}{}
    \pgfkeyssetvalue{/bodeplot/combinedenv/@commands/ph}{}
    \pgfkeyssetvalue{/bodeplot/combinedenv/@tikz}{}
    \pgfkeyssetvalue{/bodeplot/combinedenv/@prefix}{}
    \gdef\bp@domain@start{0.01}
    \gdef\bp@domain@end{100}
    \if@hzarg
      \pgfkeys{/bodeplot/frequency unit=Hz}
    \else
      \pgfkeys{/bodeplot/frequency unit=rad}
    \fi
    \if@radarg
      \pgfkeys{/bodeplot/phase unit=rad}
    \else
      \pgfkeys{/bodeplot/phase unit=deg}
    \fi
    \gdef\bp@mag@axes{}
```

```
    \gdef\bp@ph@axes{}
  },
  axes/.code={%
    \ifx\bp@mag@axes\@empty
      \xdef\bp@mag@axes{\unexpanded\expandafter{#1}}
    \else
      \xdef\bp@mag@axes{\unexpanded\expandafter{\bp@mag@axes,#1}}
    \fi
    \ifx\bp@ph@axes\@empty
      \xdef\bp@ph@axes{\unexpanded\expandafter{#1}}
    \else
      \xdef\bp@ph@axes{\unexpanded\expandafter{\bp@ph@axes,#1}}
    \fi
  },
  axes/.value required,
  mag axes/.code={%
    \ifx\bp@mag@axes\@empty
      \xdef\bp@mag@axes{\unexpanded\expandafter{#1}}
    \else
      \xdef\bp@mag@axes{\unexpanded\expandafter{\bp@mag@axes,#1}}
    \fi
  },
  mag axes/.value required,
  ph axes/.code={%
    \ifx\bp@ph@axes\@empty
      \xdef\bp@ph@axes{\unexpanded\expandafter{#1}}
    \else
      \xdef\bp@ph@axes{\unexpanded\expandafter{\bp@ph@axes,#1}}
    \fi
  },
  ph axes/.value required,
  group/.code={\pgfkeyssetvalue{/bodeplot/combinedenv/@group}{#1}},
  group/.value required,
  approx/.code={\pgfkeyssetvalue{/bodeplot/combinedenv/@approx}{#1}},
  approx/.value required,
  commands/.code={%
    \pgfkeysalso{/bodeplot/combinedenv/mag commands={#1}}
    \pgfkeysalso{/bodeplot/combinedenv/ph commands={#1}}%
  },
  commands/.value required,
  mag commands/.code={\pgfkeyssetvalue{/bodeplot/combinedenv/@commands/mag}{#1}},
  mag commands/.value required,
  ph commands/.code={\pgfkeyssetvalue{/bodeplot/combinedenv/@commands/ph}{#1}},
  ph commands/.value required,
  tikz/.code={\pgfkeyssetvalue{/bodeplot/combinedenv/@tikz}{#1}},
  tikz/.value required,
  prefix/.code={\pgfkeyssetvalue{/bodeplot/combinedenv/@prefix}{#1}},
  prefix/.value required,
  domain/.code args={#1:#2}{%
    \gdef\bp@domain@start{#1}%
    \gdef\bp@domain@end{#2}%
  },
  domain/.value required,
  phase unit/.initial={deg},
  phase unit/.default={deg},
  phase unit/.is choice,
  phase unit/deg/.code={
    \renewcommand{\bp@ph@scale}{180/pi}
    \pgfplotsset{bp@ph@x@label/.style = {xlabel={Phase (deg)}}}
    \pgfplotsset{bp@ph@y@label/.style = {ylabel={Phase (deg)}}}
  },
  phase unit/rad/.code={
    \renewcommand{\bp@ph@scale}{1}
```

```
    \pgfplotsset{bp@ph@y@label/.style = {ylabel={Phase (rad)}}}
    \pgfplotsset{bp@ph@x@label/.style = {xlabel={Phase (rad)}}}
  },
  frequency unit/.initial={rad},
  frequency unit/.default={rad},
  frequency unit/.is choice,
  frequency unit/Hz/.code={
    \renewcommand{\bp@freq@scale}{2*pi}
    \pgfplotsset{bp@freq@label/.style = {xlabel = {Frequency (Hz)}}}
    \if@pgfarg
      \pgfplotsset{bp@freq@filter/.style = {x filter/.expression={x-
log10(2*pi)}}}
    \fi
  },
  frequency unit/rad/.code={
    \renewcommand{\bp@freq@scale}{1}
    \pgfplotsset{bp@freq@label/.style = {xlabel = {Frequency (rad/s)}}}
    \if@pgfarg
      \pgfplotsset{bp@freq@filter/.style = {x filter/.expression={x}}}
    \fi
  },
  .unknown/.code={%
    \edef\bp@full{\pgfkeyscurrentkey}%
    \def\stripbodeprefix##1/bodeplot/combinedenv/##2\relax{##2}%
    \edef\bp@short{\expandafter\stripbodeprefix\bp@full\relax}%
    \edef\bp@checkfull{\bp@full}%
    \edef\bp@checkshort{\bp@short}%
    \ifx\bp@checkfull\bp@checkshort
      \def\removeslash##1/##2\relax{##2}%
      \edef\bp@short{\expandafter\removeslash\bp@full\relax}%
    \fi
    \ifx\pgfkeyscurrentvalue\pgfkeysnovalue
      \edef\bp@new{\bp@short}%
    \else
      \edef\bp@new{\bp@short=
      {\unexpanded\expandafter{\pgfkeyscurrentvalue}}}%
    \fi
    \ifx\bp@new\@empty\else
      \ifx\bp@mag@axes\@empty
        \xdef\bp@mag@axes{\unexpanded\expandafter{\bp@new}}
      \else
        \xdef\bp@mag@axes{\unexpanded\expandafter{\bp@mag@axes},%
        \unexpanded\expandafter{\bp@new}}
      \fi
    \fi
    \ifx\bp@new\@empty\else
      \ifx\bp@ph@axes\@empty
        \xdef\bp@ph@axes{\unexpanded\expandafter{\bp@new}}
      \else
        \xdef\bp@ph@axes{\unexpanded\expandafter{\bp@ph@axes},%
        \unexpanded\expandafter{\bp@new}}
      \fi
    \fi
  }
}
```

`/bodeplot/zpk` PGF keys for supplying zero-pole-gain-delay (ZPK) representations.

```
\pgfkeys{
  /bodeplot/zpk/.is family,
  /bodeplot/zpk/.cd,
  reset/.code={%
    \pgfkeyssetvalue{/bodeplot/zpk/@zeros}{}
    \pgfkeyssetvalue{/bodeplot/zpk/@poles}{}
```

```
    \pgfkeyssetvalue{/bodeplot/zpk/@gain}{}
    \pgfkeyssetvalue{/bodeplot/zpk/@delay}{}
  },
  zeros/.code={\pgfkeyssetvalue{/bodeplot/zpk/@zeros}{#1}},
  zeros/.value required,
  poles/.code={\pgfkeyssetvalue{/bodeplot/zpk/@poles}{#1}},
  poles/.value required,
  gain/.code={\pgfkeyssetvalue{/bodeplot/zpk/@gain}{#1}},
  gain/.value required,
  delay/.code={\pgfkeyssetvalue{/bodeplot/zpk/@delay}{#1}},
  delay/.value required,
}
```

`/bodeplot/tf` PGF keys for supplying transfer function (TF) representations.

```
\pgfkeys{
  /bodeplot/tf/.is family,
  /bodeplot/tf/.cd,
  reset/.code={%
    \pgfkeyssetvalue{/bodeplot/tf/@numerator}{}
    \pgfkeyssetvalue{/bodeplot/tf/@denominator}{}
    \pgfkeyssetvalue{/bodeplot/tf/@delay}{}
  },
  numerator/.code={\pgfkeyssetvalue{/bodeplot/tf/@numerator}{#1}},
  numerator/.value required,
  denominator/.code={\pgfkeyssetvalue{/bodeplot/tf/@denominator}{#1}},
  denominator/.value required,
  delay/.code={\pgfkeyssetvalue{/bodeplot/tf/@delay}{#1}},
  delay/.value required
}
```

`/bodeplot/add` PGF keys for adding asymptotic and linear plots.

```
\pgfkeys{
  /bodeplot/add/.is family,
  /bodeplot/add/.cd,
  reset/.code={%
    \gdef\bp@add@O{}%
  },
  true/.default={},
  true/.code={%
    \pgfutil@ifempty{#1}{
      \ifx\bp@add@O\@empty
        \g@addto@macro\bp@add@O{true}%
      \else
        \g@addto@macro\bp@add@O{,true}%
      \fi
    }{%
      \ifx\bp@add@O\@empty
        \g@addto@macro\bp@add@O{true/{#1}}%
      \else
        \g@addto@macro\bp@add@O{,true/{#1}}%
      \fi
    }
  },
  linear/.default={},
  linear/.code={%
    \pgfutil@ifempty{#1}{
      \ifx\bp@add@O\@empty
        \g@addto@macro\bp@add@O{linear}%
      \else
        \g@addto@macro\bp@add@O{,linear}%
      \fi
    }{%
      \ifx\bp@add@O\@empty
```

```
        \g@addto@macro\bp@add@O{linear/{#1}}%
      \else
        \g@addto@macro\bp@add@O{,linear/{#1}}%
      \fi
    }
  },
  asymptotic/.default={},
  asymptotic/.code={%
    \pgfutil@ifempty{#1}{
      \ifx\bp@add@O\@empty
        \g@addto@macro\bp@add@O{asymptotic}%
      \else
        \g@addto@macro\bp@add@O{,asymptotic}%
      \fi
    }{%
      \ifx\bp@add@O\@empty
        \g@addto@macro\bp@add@O{asymptotic/{#1}}%
      \else
        \g@addto@macro\bp@add@O{,asymptotic/{#1}}%
      \fi
    }%
  },
  .unknown/.code={%
    \edef\bp@full{\pgfkeyscurrentkey}%
    \def\stripbodeprefix##1/bodeplot/add/##2\relax{##2}%
    \edef\bp@short{\expandafter\stripbodeprefix\bp@full\relax}%
    \edef\bp@checkfull{\bp@full}%
    \edef\bp@checkshort{\bp@short}%
    \ifx\bp@checkfull\bp@checkshort
      \def\removeslash##1/##2\relax{##2}%
      \edef\bp@short{\expandafter\removeslash\bp@full\relax}%
    \fi
    \ifx\pgfkeyscurrentvalue\pgfkeysnovalue
      \edef\bp@new{\bp@short}%
    \else
      \edef\bp@new{\bp@short=
      {\unexpanded\expandafter{\pgfkeyscurrentvalue}}}%
    \fi
    \ifx\bp@new\@empty\else
      \ifx\bp@add@O\@empty
        \xdef\bp@add@O{\unexpanded\expandafter{\bp@new}}
      \else
        \xdef\bp@add@O{\unexpanded\expandafter{\bp@add@O},%
        \unexpanded\expandafter{\bp@new}}
      \fi
    \fi
  }
}
```

`/bodeplot/nyquist` PGF keys for Nyquist plot options.

```
\pgfkeys{
  /bodeplot/nyquist/.is family,
  /bodeplot/nyquist/.cd,
  reset/.code={
    \pgfkeyssetvalue{/bodeplot/nyquist/@axes}{}
    \pgfkeyssetvalue{/bodeplot/nyquist/@plot}{}
    \pgfkeyssetvalue{/bodeplot/nyquist/@commands}{}
    \pgfkeyssetvalue{/bodeplot/nyquist/@tikz}{}
    \pgfkeyssetvalue{/bodeplot/nyquist/@prefix}{}
    \gdef\bp@domain@start{-30}
    \gdef\bp@domain@end{30}
    \gdef\bp@plot{}
  },
```

```
  axes/.code={\pgfkeyssetvalue{/bodeplot/nyquist/@axes}{#1}},
  axes/.value required,
  plot/.code={%
    \xdef\bp@plot{\unexpanded\expandafter{#1}}
  },
  plot/.value required,
  commands/.code={\pgfkeyssetvalue{/bodeplot/nyquist/@commands}{#1}},
  commands/.value required,
  tikz/.code={\pgfkeyssetvalue{/bodeplot/nyquist/@tikz}{#1}},
  tikz/.value required,
  prefix/.code={\pgfkeyssetvalue{/bodeplot/nyquist/@prefix}{#1}},
  prefix/.value required,
  domain/.code args={#1:#2}{\gdef\bp@domain@start{#1}\gdef\bp@domain@end{#2}},
  domain/.value required,
  .unknown/.code={%
    \edef\bp@full{\pgfkeyscurrentkey}%
    \def\stripbodeprefix##1/bodeplot/nyquist/##2\relax{##2}%
    \edef\bp@short{\expandafter\stripbodeprefix\bp@full\relax}%
    \edef\bp@checkfull{\bp@full}%
    \edef\bp@checkshort{\bp@short}%
    \ifx\bp@checkfull\bp@checkshort
      \def\removeslash##1/##2\relax{##2}%
      \edef\bp@short{\expandafter\removeslash\bp@full\relax}%
    \fi
    \ifx\pgfkeyscurrentvalue\pgfkeysnovalue
      \edef\bp@new{\bp@short}%
    \else
      \edef\bp@new{\bp@short=
      {\unexpanded\expandafter{\pgfkeyscurrentvalue}}}%
    \fi
    \ifx\bp@new\@empty\else
      \ifx\bp@plot\@empty
        \xdef\bp@plot{\unexpanded\expandafter{\bp@new}}
      \else
        \xdef\bp@plot{\unexpanded\expandafter{\bp@plot},%
        \unexpanded\expandafter{\bp@new}}
      \fi
    \fi
  }
}
```

`/bodeplot/nichols` PGF keys for Nichols plot options.

```
\pgfkeys{
  /bodeplot/nichols/.is family,
  /bodeplot/nichols/.cd,
  reset/.code={
    \pgfkeyssetvalue{/bodeplot/nichols/@axes}{}
    \pgfkeyssetvalue{/bodeplot/nichols/@plot}{}
    \pgfkeyssetvalue{/bodeplot/nichols/@commands}{}
    \pgfkeyssetvalue{/bodeplot/nichols/@tikz}{}
    \pgfkeyssetvalue{/bodeplot/nichols/@prefix}{}
    \gdef\bp@domain@start{0.01}
    \gdef\bp@domain@end{100}
    \gdef\bp@plot{}
  },
  axes/.code={\pgfkeyssetvalue{/bodeplot/nichols/@axes}{#1}},
  axes/.value required,
  plot/.code={%
    \xdef\bp@plot{\unexpanded\expandafter{#1}}
  },
  plot/.value required,
  commands/.code={\pgfkeyssetvalue{/bodeplot/nichols/@commands}{#1}},
  commands/.value required,
```

```
  tikz/.code={\pgfkeyssetvalue{/bodeplot/nichols/@tikz}{#1}},
  tikz/.value required,
  prefix/.code={\pgfkeyssetvalue{/bodeplot/nichols/@prefix}{#1}},
  prefix/.value required,
  domain/.code args={#1:#2}{\gdef\bp@domain@start{#1}\gdef\bp@domain@end{#2}},
  domain/.value required,
  .unknown/.code={%
    \edef\bp@full{\pgfkeyscurrentkey}%
    \def\stripbodeprefix##1/bodeplot/nichols/##2\relax{##2}%
    \edef\bp@short{\expandafter\stripbodeprefix\bp@full\relax}%
    \edef\bp@checkfull{\bp@full}%
    \edef\bp@checkshort{\bp@short}%
    \ifx\bp@checkfull\bp@checkshort
      \def\removeslash##1/##2\relax{##2}%
      \edef\bp@short{\expandafter\removeslash\bp@full\relax}%
    \fi
    \ifx\pgfkeyscurrentvalue\pgfkeysnovalue
      \edef\bp@new{\bp@short}%
    \else
      \edef\bp@new{\bp@short=
      {\unexpanded\expandafter{\pgfkeyscurrentvalue}}}%
    \fi
    \ifx\bp@new\@empty\else
      \ifx\bp@plot\@empty
        \xdef\bp@plot{\unexpanded\expandafter{\bp@new}}
      \else
        \xdef\bp@plot{\unexpanded\expandafter{\bp@plot},%
        \unexpanded\expandafter{\bp@new}}
      \fi
    \fi
  }
}
```

`/bodeplot/pzmap` PGF keys for pole-zero map options.

```
\pgfkeys{
  /bodeplot/pzmap/.is family,
  /bodeplot/pzmap/.cd,
  reset/.code={
    \pgfkeyssetvalue{/bodeplot/pzmap/@axes}{}
    \pgfkeyssetvalue{/bodeplot/pzmap/@plot}{}
    \pgfkeyssetvalue{/bodeplot/pzmap/@commands}{}
    \pgfkeyssetvalue{/bodeplot/pzmap/@tikz}{}
    \pgfkeyssetvalue{/bodeplot/pzmap/@prefix}{}
    \pgfkeyssetvalue{/bodeplot/pzmap/@scale}{linear}
  },
  axes/.code={\pgfkeyssetvalue{/bodeplot/pzmap/@axes}{#1}},
  axes/.value required,
  plot/.code={\pgfkeyssetvalue{/bodeplot/pzmap/@plot}{#1}},
  plot/.value required,
  commands/.code={\pgfkeyssetvalue{/bodeplot/pzmap/@commands}{#1}},
  commands/.value required,
  tikz/.code={\pgfkeyssetvalue{/bodeplot/pzmap/@tikz}{#1}},
  tikz/.value required,
  prefix/.code={\pgfkeyssetvalue{/bodeplot/pzmap/@prefix}{#1}},
  prefix/.value required,
  scale/.initial=linear,
  scale/.default=linear,
  scale/.is choice,
  scale/linear/.code={\pgfkeyssetvalue{/bodeplot/pzmap/@scale}{linear}},
  scale/log/.code={\pgfkeyssetvalue{/bodeplot/pzmap/@scale}{log}},
}
```

4.3 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 `\bp@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)*\bp@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*\bp@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))))*\bp@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}))})))))))*\bp@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))*\bp@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}
```

4.4 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

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}
```

\MagCSZeros
\MagCSZerosAsymp
\MagCSZerosLin
\PhCSZeros
\PhCSZerosAsymp
\PhCSZerosLin

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})))*\bp@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))))))*\bp@ph@scale)}
\newcommand*{\PhCSPolesAsymp}[2]{((#1>0?(t<#2?0:-
  pi):(t<#2?0:pi))*\bp@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

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}
```

True, linear, and asymptotic phase plots for the general second order transfer function.

```
\newcommand*{\PhSOPoles}[2]{((-atan2((#1)*t,((#2) - \n@pow{t}{2})))*\bp@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))))});
}
```

4.5 Commands for Bode plots

4.5.1 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`.

```
\NewDocumentCommand{\BodeZPK}{ O{} m G{} G{} }{%
```

The macro now accepts an optional argument followed by a mandatory ZPK specification and two optional frequency arguments. If the frequency arguments are not provided, the pgfkeys interface introduced in v3.0 is used. Otherwise, the legacy interface is used for backward compatibility.

```
  \pgfutil@ifempty{#3}{%
    \pgfkeys{/bodeplot/.cd, reset}
    \pgfkeys{/bodeplot/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/@axes/mag}{\bp@mag@axes}
    \pgfkeysgetvalue{/bodeplot/@axes/ph}{\bp@ph@axes}
    \pgfkeysgetvalue{/bodeplot/@group}{\bp@group}
    \pgfkeysgetvalue{/bodeplot/@approx}{\bp@approx}
    \pgfkeysgetvalue{/bodeplot/@commands/mag}{\bp@mag@commands}
    \pgfkeysgetvalue{/bodeplot/@commands/ph}{\bp@ph@commands}
    \pgfkeysgetvalue{/bodeplot/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/@prefix}{\bp@user@prefix}
```

```
    \bp@zpk@new@to@legacy{#2}
  }{%
    \if@radarg
      \pgfkeys{/bodeplot/phase unit=rad}
    \else
      \pgfkeys{/bodeplot/phase unit=deg}
    \fi
    \if@hzarg
      \pgfkeys{/bodeplot/frequency unit=Hz}
    \else
      \pgfkeys{/bodeplot/frequency unit=rad}
    \fi
    \bp@parse@opt{#1}
    \edef\bp@legacy{#2}
    \edef\bp@domain@start{#3}
    \edef\bp@domain@end{#4}
  }%

    \edef\bp@tmp{\noexpand\begin{tikzpicture}
      [\unexpanded\expandafter{\bp@tikz}]}
    \bp@tmp
    \gdef\bp@mag{}
    \gdef\bp@ph{}
    \bp@ZPK@plot{\bp@mag}{\bp@ph}{\bp@approx}{\bp@legacy}
    \edef\bp@tmp{\noexpand\begin{groupplot}[
      bp@style,
      xmin=\bp@domain@start,
      xmax=\bp@domain@end,
      domain=\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale,
      height=2.5cm,
      xmode=log,
      group style = {group size = 1 by 2,vertical sep=0.25cm},
      \unexpanded\expandafter{\bp@group}
    ]}
    \bp@tmp
      \edef\bp@tmp@mag{\noexpand\nextgroupplot
        [ylabel={Gain (dB)}, xmajorticks=false, \bp@mag@axes]
      \noexpand\addplot [bp@freq@filter, variable=t, thick,
        \unexpanded\expandafter{\bp@mag@plot}]}
      \edef\bp@tmp@ph{\noexpand\nextgroupplot
        [bp@ph@y@label, bp@freq@label, \bp@ph@axes]
      \noexpand\addplot [bp@freq@filter, variable=t, thick, trig format plots=rad,
        \unexpanded\expandafter{\bp@ph@plot}]}
      \if@pgfarg
        \bp@tmp@mag {\bp@mag};
        \bp@mag@commands
        \bp@tmp@ph {\bp@ph};
        \bp@ph@commands
      \else
        \stepcounter{bp@gnuplot@id}
        \edef\gnu@id{\arabic{bp@gnuplot@id}}
        \bp@gnu@plot{\bp@mag}{\gnu@id}
        \expandafter\bp@tmp@mag\bp@gnu@cmd
        \bp@mag@commands
        \stepcounter{bp@gnuplot@id}
        \edef\gnu@id{\arabic{bp@gnuplot@id}}
        \bp@gnu@plot{\bp@ph}{\gnu@id}
        \expandafter\bp@tmp@ph\bp@gnu@cmd
        \bp@ph@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{\bp@short@list}
    \BodeZPK@Shorthandoff
  }
  \NewDocumentCommand{\BodeZPK@Shorthandoff}{ O{} m G{} G{} }{%
    \Orig@BodeZPK[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\bp@short@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.

```
\NewDocumentCommand{\BodeTF}{ O{} m G{} G{} }{%
  \if@pgfarg
    \bp@tf@to@zpk{#2}{\bp@tmp@zpk}{\bp@tmp@status}
    \ifnum\pdf@strcmp{\bp@tmp@status}{true}=0
      \pgfutil@ifempty{#3}{%
        \BodeZPK[#1]{\bp@tmp@zpk}%
      }{%
        \BodeZPK[domain=#3:#4,#1]{\bp@tmp@zpk}%
      }%
      \def\bp@fallback{false}
    \else
      \def\bp@fallback{true}
    \fi
  \else
    \def\bp@fallback{true}
  \fi
  \ifnum\pdf@strcmp{\bp@fallback}{true}=0
    \pgfutil@ifempty{#3}{%
      \pgfkeys{/bodeplot/.cd, reset}
      \pgfkeys{/bodeplot/.cd, #1}
      \pgfkeysgetvalue{/bodeplot/@axes/mag}{\bp@mag@axes}
      \pgfkeysgetvalue{/bodeplot/@axes/ph}{\bp@ph@axes}
      \pgfkeysgetvalue{/bodeplot/@group}{\bp@group}
      \pgfkeysgetvalue{/bodeplot/@approx}{\bp@approx}
      \pgfkeysgetvalue{/bodeplot/@commands/mag}{\bp@mag@commands}
      \pgfkeysgetvalue{/bodeplot/@commands/ph}{\bp@ph@commands}
      \pgfkeysgetvalue{/bodeplot/@tikz}{\bp@tikz}
      \pgfkeysgetvalue{/bodeplot/@prefix}{\bp@user@prefix}
      \bp@tf@new@to@legacy{#2}
    }{%
      \if@radarg
        \pgfkeys{/bodeplot/phase unit=rad}
      \else
        \pgfkeys{/bodeplot/phase unit=deg}
      \fi
      \if@hzarg
        \pgfkeys{/bodeplot/frequency unit=Hz}
      \else
        \pgfkeys{/bodeplot/frequency unit=rad}
      \fi
      \bp@parse@opt{#1}
      \edef\bp@legacy{#2}
      \edef\bp@domain@start{#3}
      \edef\bp@domain@end{#4}
    }%
```

```
    \edef\bp@cmd{\noexpand\begin{tikzpicture}
      [\unexpanded\expandafter{\bp@tikz}]}
    \bp@cmd
      \gdef\bp@mag{}
      \gdef\bp@ph{}
      \bp@TF@plot{\bp@mag}{\bp@ph}{\bp@legacy}
      \edef\bp@cmd{\noexpand\begin{groupplot}[
        bp@style,
        xmin=\bp@domain@start,
        xmax=\bp@domain@end,
        domain=\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale,
        height=2.5cm,
        xmode=log,
        group style = {group size = 1 by 2,vertical sep=0.25cm},
        \unexpanded\expandafter{\bp@group}
      ]}
      \bp@cmd
        \edef\bp@mag@cmd{\noexpand\nextgroupplot
          [ylabel={Gain (dB)}, xmajorticks=false, \bp@mag@axes]
        \noexpand\addplot [bp@freq@filter, variable=t, thick,
          \unexpanded\expandafter{\bp@mag@plot}]}
        \edef\bp@ph@cmd{\noexpand\nextgroupplot
          [bp@ph@y@label, bp@freq@label, \bp@ph@axes]
        \noexpand\addplot [bp@freq@filter, variable=t, thick, trig format plots=rad,
          \unexpanded\expandafter{\bp@ph@plot}]}
        \if@pgfarg
          \bp@mag@cmd {\bp@mag};
          \bp@mag@commands
          \bp@ph@cmd {\n@mod{\bp@ph}{2*pi*\bp@ph@scale}};
          \bp@ph@commands
        \else
          \stepcounter{bp@gnuplot@id}
          \edef\gnu@id{\arabic{bp@gnuplot@id}}
          \bp@gnu@plot{\bp@mag}{\gnu@id}
          \expandafter\bp@mag@cmd\bp@gnu@cmd
          \bp@mag@commands
          \stepcounter{bp@gnuplot@id}
          \edef\gnu@id{\arabic{bp@gnuplot@id}}
          \bp@gnu@unwrap@plot{\bp@ph}{\gnu@id}
          \expandafter\bp@ph@cmd\bp@gnu@cmd
          \bp@ph@commands
        \fi
      \end{groupplot}
    \end{tikzpicture}
  \fi
}
```

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

```
\AtBeginDocument{
  \if@babel
  \let\Orig@BodeTF\BodeTF
  \renewcommand{\BodeTF}{%
    \expandafter\shorthandoff\expandafter{\bp@short@list}
    \BodeTF@Shorthandoff
  }
  \NewDocumentCommand{\BodeTF@Shorthandoff}{ O{} m G{} G{} }{%
    \Orig@BodeTF[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\bp@short@list}
  }
  \fi
}
```

`\addBodeZPKPlots` This macro is designed to issue multiple `\addplot` macros for the same set of poles, zeros, gain, and delay. All of the work is done by the `\bp@ZPK@plot` macro.

```
\NewDocumentCommand{\addBodeZPKPlots}{ O{} m m }{%
  \edef\bp@spec@arg{#3}%
  \expandafter\bp@contains@equal\expandafter{\bp@spec@arg}{
    \pgfkeys{/bodeplot/add/.cd, reset}
    \pgfkeys{/bodeplot/add/.cd, #1}
    \bp@zpk@new@to@legacy{\bp@spec@arg}
    \expandafter\bp@fix@add@opt\expandafter{\bp@add@O}
  }{
    \bp@fix@add@opt{#1}
    \edef\bp@legacy{\bp@spec@arg}
  }
  \foreach \approx/\opt in \bp@add@O {
    \ifx\approx\@empty\else
      \gdef\bp@plot{}
      \gdef\bp@tmp{}
      \ifnum\pdf@strcmp{#2}{phase}=0
        \bp@ZPK@plot{\bp@tmp}{\bp@plot}{\approx}{\bp@legacy}
      \else
        \bp@ZPK@plot{\bp@plot}{\bp@tmp}{\approx}{\bp@legacy}
      \fi
      \if@pgfarg
        \edef\bp@cmd{\noexpand\addplot [bp@freq@filter,
        domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale,
        variable=t, thick, trig format plots=rad, \unexpanded\expandafter{\opt}]}
        \bp@cmd {\bp@plot};
      \else
        \stepcounter{bp@gnuplot@id}
        \edef\gnu@id{\arabic{bp@gnuplot@id}}
        \bp@gnu@plot[\bp@freq@scale]{\bp@plot}{\gnu@id}
        \edef\bp@cmd{\noexpand\addplot [variable=t, thick,
        \unexpanded\expandafter{\opt}]}
        \expandafter\bp@cmd\bp@gnu@cmd
      \fi
    \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 `\bp@TF@plot` macro.

```
\NewDocumentCommand{\addBodeTFPlot}{ O{} m m }{%
  \edef\bp@spec@arg{#3}%
  \expandafter\bp@contains@equal\expandafter{\bp@spec@arg}{
    \bp@tf@new@to@legacy{\bp@spec@arg}
  }{
    \edef\bp@legacy{\bp@spec@arg}
  }
  \gdef\bp@plot{}
  \gdef\bp@tmp{}
  \ifnum\pdf@strcmp{#2}{phase}=0
    \bp@TF@plot{\bp@tmp}{\bp@plot}{\bp@legacy}
  \else
    \bp@TF@plot{\bp@plot}{\bp@tmp}{\bp@legacy}
  \fi
  \if@pgfarg
    \ifnum\pdf@strcmp{#2}{phase}=0
      \edef\bp@cmd{\noexpand\addplot [bp@freq@filter,
      domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale,
      variable=t, thick, trig format plots=rad,
      \unexpanded\expandafter{#1}]}
      \bp@cmd {\n@mod{\bp@plot}{2*pi}};
    \else
      \edef\bp@cmd{\noexpand\addplot [bp@freq@filter,
      domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale,
```

```
      variable=t, thick, \unexpanded\expandafter{#1}]}
      \bp@cmd {\bp@plot};
    \fi
  \else
    \stepcounter{bp@gnuplot@id}
    \edef\gnu@id{\arabic{bp@gnuplot@id}}
    \edef\bp@cmd{\noexpand\addplot [variable=t, thick,
    \unexpanded\expandafter{#1}]}
    \ifnum\pdf@strcmp{#2}{phase}=0
      \bp@gnu@unwrap@plot[\bp@freq@scale]{\bp@plot}{\gnu@id}
      \expandafter\bp@cmd\bp@gnu@cmd
    \else
      \bp@gnu@plot[\bp@freq@scale]{\bp@plot}{\gnu@id}
      \expandafter\bp@cmd\bp@gnu@cmd
    \fi
  \fi
}
```

`\addBodeTFPlots` Similar to `\addBodeZPKPlots` but for systems specified in the TF format. This macro converts the TF specification to ZPK format internally using Python via `--shell-escape` and then calls `\addBodeZPKPlots` to generate the plots. If Python with the `scipy` and `numpy` libraries is not available on the shell, the package returns an error message.

```
\NewDocumentCommand{\addBodeTFPlots}{ O{} m m }{%
  \bp@tf@to@zpk{#3}{\bp@tmp@zpk}{\bp@tmp@status}
  \ifnum\pdf@strcmp{\bp@tmp@status}{true}=0
    \addBodeZPKPlots[#1]{#2}{\bp@tmp@zpk}
  \else
    \PackageError{bodeplot}{The |\addBodeTFPlots| macro enables linear and asymptotic app
shell-escape. If Python is not available, use |\addBodeTFPlot| to generate plots without
  \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][]{
  \if@pgfarg
    \edef\bp@cmd{\noexpand\addplot [bp@freq@filter,
    domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale,
    variable=t, thick, trig format plots=rad, #1]}
    \bp@cmd {#2};
  \else
    \stepcounter{bp@gnuplot@id}
    \edef\gnu@id{\arabic{bp@gnuplot@id}}
    \edef\bp@cmd{\noexpand\addplot [variable=t, thick, #1]}
    \bp@gnu@plot[\bp@freq@scale]{#2}{\gnu@id}
    \expandafter\bp@cmd\bp@gnu@cmd
  \fi
}
```

`BodePhPlot` (*env.*) An environment to host phase plot macros that pass parametric functions to `\addplot` macros. Uses the defaults specified in `bp@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{\bp@short@list}
    }
    \AddToHook{env/BodePhPlot/end}{%
      \expandafter\shorthandon\expandafter{\bp@short@list}
```

```
    }
  \fi
}
\NewDocumentEnvironment{BodePhPlot}{ O{} G{} G{} +b }{
  \pgfutil@ifempty{#2}{%
    \pgfkeys{/bodeplot/env/.cd, reset}
    \pgfkeys{/bodeplot/env/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/env/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/env/@prefix}{\bp@user@prefix}
  }{%
    \bp@parse@env@opt{#1}
    \edef\bp@domain@start{#2}
    \edef\bp@domain@end{#3}
  }
  \edef\bp@cmd{\noexpand\begin{tikzpicture}
  [\unexpanded\expandafter{\bp@tikz}]}
  \bp@cmd
    \edef\bp@cmd{\noexpand\begin{semilogxaxis}[
      bp@ph@y@label,
      bp@freq@label,
      bp@style,
      xmin={\bp@domain@start},
      xmax={\bp@domain@end},
      domain=\bp@domain@start:\bp@domain@end,
      height=2.5cm,
      \unexpanded\expandafter{\bp@axes}
    ]}
    \bp@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 `bp@style` to create a shortcut that includes the `tikzpicture` and `semilogaxis` environments.

```
\AtBeginDocument{%
  \if@babel
    \AddToHook{env/BodeMagPlot/begin}{%
    \expandafter\shorthandoff\expandafter{\bp@short@list}
  }
    \AddToHook{env/BodeMagPlot/end}{%
    \expandafter\shorthandon\expandafter{\bp@short@list}
  }
  \fi
}
\NewDocumentEnvironment{BodeMagPlot}{ O{} G{} G{} +b }{
  \pgfutil@ifempty{#2}{%
    \pgfkeys{/bodeplot/env/.cd, reset}
    \pgfkeys{/bodeplot/env/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/env/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/env/@prefix}{\bp@user@prefix}
  }{%
    \bp@parse@env@opt{#1}
    \edef\bp@domain@start{#2}
    \edef\bp@domain@end{#3}
  }
  \edef\bp@cmd{\noexpand\begin{tikzpicture}
  [\unexpanded\expandafter{\bp@tikz}]}
  \bp@cmd
    \edef\bp@cmd{\noexpand\begin{semilogxaxis}[
      bp@style,
      bp@freq@label,
```

```
      xmin={\bp@domain@start},
      xmax={\bp@domain@end},
      domain=\bp@domain@start:\bp@domain@end,
      height=2.5cm,
      ylabel={Gain (dB)},
      \unexpanded\expandafter{\bp@axes}
    ]}
    \bp@cmd
      #4
    \end{semilogxaxis}
  \end{tikzpicture}
}{}
```

`\addBodePlot` Unified macro to add Bode plots for both ZPK and TF system representations inside a BodePlot environment. Supports both pgfkeys interface introduced in v3.0 and the legacy interface.

```
\NewDocumentCommand{\addBodePlot}{ O{} m G{} }{%
  \gdef\bp@mag{}
  \gdef\bp@ph{}
  \pgfutil@ifempty{#3}{%
    \pgfkeys{/bodeplot/add/.cd, reset}
    \pgfkeys{/bodeplot/add/.cd, #1}
    \expandafter\bp@parse@add@Bode@opt\expandafter{\bp@add@O}
    \bp@contains@num{#2}{%
      \bp@tf@new@to@legacy{#2}
      \bp@TF@plot{\bp@mag}{\bp@ph}{\bp@legacy}
      \edef\bp@mode{tf}
    }{
      \bp@zpk@new@to@legacy{#2}
      \bp@ZPK@plot{\bp@mag}{\bp@ph}{\bp@approx}{\bp@legacy}
      \edef\bp@mode{zpk}
    }
  }{
    \bp@parse@add@Bode@opt{#1}
    \ifnum\pdf@strcmp{#2}{zpk}=0
      \edef\bp@mode{zpk}
      \bp@ZPK@plot{\bp@mag}{\bp@ph}{\bp@approx}{#3}
    \else
      \ifnum\pdf@strcmp{#2}{tf}=0
        \edef\bp@mode{tf}
        \bp@TF@plot{\bp@mag}{\bp@ph}{#3}
      \else
        \PackageError {bodeplot} {Unknown system representation `#2'.}
        {Supported representations are `zpk' and `tf'.}
      \fi
    \fi
  }
  \if@pgfarg
    \xdef\bp@mag@cmd{\unexpanded\expandafter{\bp@mag@cmd}\noexpand\addplot
    [bp@freq@filter, domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain},
    variable=t, thick, trig format plots=rad,
    \unexpanded\expandafter{\bp@plot}] {\bp@mag};}
    \xdef\bp@ph@cmd{\unexpanded\expandafter{\bp@ph@cmd}\noexpand\addplot
    [bp@freq@filter, domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain},
    variable=t, thick, trig format plots=rad,
    \unexpanded\expandafter{\bp@plot}] {\bp@ph};}
  \else
    \stepcounter{bp@gnuplot@id}
    \edef\gnu@id{\arabic{bp@gnuplot@id}}
    \bp@gnu@plot{\bp@mag}{\gnu@id}
    \xdef\bp@mag@cmd{\unexpanded\expandafter{\bp@mag@cmd}\noexpand\addplot
    [variable=t, thick, \unexpanded\expandafter{\bp@plot}]\bp@gnu@cmd}
    \stepcounter{bp@gnuplot@id}
```

```
    \edef\gnu@id{\arabic{bp@gnuplot@id}}
    \ifnum\pdf@strcmp{\bp@mode}{zpk}=0
      \bp@gnu@plot{\bp@ph}{\gnu@id}
      \else
        \ifnum\pdf@strcmp{\bp@mode}{tf}=0
          \bp@gnu@unwrap@plot{\bp@ph}{\gnu@id}
        \fi
      \fi
    \xdef\bp@ph@cmd{\unexpanded\expandafter{\bp@ph@cmd}\noexpand\addplot
    [variable=t, thick, \unexpanded\expandafter{\bp@plot}]\bp@gnu@cmd}
  \fi
}
```

`BodePlot` (*env.*) An 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{} G{} G{} G{} +b }{
  \pgfutil@ifempty{#2}{%
    \pgfkeys{/bodeplot/combinedenv/.cd, reset}
    \pgfkeys{/bodeplot/combinedenv/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/combinedenv/@group}{\bp@group}
    \pgfkeysgetvalue{/bodeplot/combinedenv/@approx}{\bp@approx}
    \pgfkeysgetvalue{/bodeplot/combinedenv/@commands/mag}{\bp@mag@commands}
    \pgfkeysgetvalue{/bodeplot/combinedenv/@commands/ph}{\bp@ph@commands}
    \pgfkeysgetvalue{/bodeplot/combinedenv/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/combinedenv/@prefix}{\bp@user@prefix}
  }{%
    \bp@parse@opt{#1}
    \edef\bp@domain@start{#2}
    \edef\bp@domain@end{#3}
  }
  \gdef\bp@mag@cmd{}
  \gdef\bp@ph@cmd{}
  \edef\bp@cmd{\noexpand\begin{tikzpicture}
  [\unexpanded\expandafter{\bp@tikz}]}
  \bp@cmd
    \edef\bp@cmd{\noexpand\begin{groupplot}[
      bp@style,
      xmin=\bp@domain@start,
      xmax=\bp@domain@end,
      domain=\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale,
      height=2.5cm,
      xmode=log,
      group style = {group size = 1 by 2,vertical sep=0.25cm},
      \unexpanded\expandafter{\bp@group}
    ]}
    \bp@cmd
      #4
      \edef\temp@mag@cmd{\noexpand\nextgroupplot
      [ylabel={Gain (dB)}, xmajorticks=false,
      \unexpanded\expandafter{\bp@mag@axes}]}
      \edef\temp@ph@cmd{\noexpand\nextgroupplot
      [bp@ph@y@label, bp@freq@label,
      \unexpanded\expandafter{\bp@ph@axes}]}
      \temp@mag@cmd
      \bp@mag@cmd
      \temp@ph@cmd
      \bp@ph@cmd
    \end{groupplot}
  \end{tikzpicture}
}{}
```

4.5.2 Internal macros

`\bp@parse@complex` Parses complex numbers in the format `a+bi` where `a` and `b` can be arbitrary pgfmath/gnuplot expressions. The scans to the left starting from the trailing `i` and finds the first top-level `+` or `-`, i.e., the first such operator not inside braces or parentheses. Everything between the `i` and the `+` or `-` is treated as the imaginary part, and everything else is treated as the real part. Sets `\bp@realpart`, `\bp@imagpart`, and `\bp@complex`.

```
\ExplSyntaxOn
\tl_new:N \l_real_tl
\tl_new:N \l_imag_tl
\tl_new:N \bp@complex
\tl_new:N \bp@realpart
\tl_new:N \bp@imagpart

% Variables for the complex number parser
\tl_new:N \l__bp_input_tl
\tl_new:N \l__bp_before_tl
\tl_new:N \l__bp_after_tl
\tl_new:N \l__bp_sep_tl
\int_new:N \l__bp_pos_int
\int_new:N \l__bp_seppos_int
\int_new:N \l__bp_paren_int
\bool_new:N \l__bp_found_bool

% Generate V variant for tl_if_single_token
\prg_generate_conditional_variant:Nnn \tl_if_single_token:n { V } { TF, T, F }

\NewDocumentCommand{\bp@parse@complex}{m}{
  \tl_set:Nx \l__bp_input_tl {#1}
  \tl_remove_all:Nn \l__bp_input_tl { ~ }

  % Check if it ends with 'i'
  \regex_match:nVTF { i$ } \l__bp_input_tl
    { \__bp_parse_complex_with_i: }
    { \__bp_parse_real_only: }

  \tl_gset_eq:NN \bp@realpart \l_real_tl
  \tl_gset_eq:NN \bp@imagpart \l_imag_tl
  \tl_gset:Nx \bp@complex { { { \l_real_tl } , { \l_imag_tl } } }
}

% Parse a pure real number (no trailing i)
\cs_new_protected:Npn \__bp_parse_real_only:
  {
    \__bp_unwrap_braces:N \l__bp_input_tl
    \tl_set_eq:NN \l_real_tl \l__bp_input_tl
    \tl_set:Nn \l_imag_tl {0}
  }

% Parse a complex number (has trailing i)
\cs_new_protected:Npn \__bp_parse_complex_with_i:
  {
    % Remove trailing 'i'
    \regex_replace_once:nnN { i$ } {} \l__bp_input_tl

    % Find the last + or - at depth 0 (but not at position 0)
    \__bp_find_separator:

    \bool_if:NTF \l__bp_found_bool
      { \__bp_split_at_separator: }
      { \__bp_parse_pure_imaginary: }
  }
```

```
1393
1394 % Parse pure imaginary (no separator found)
1395 \cs_new_protected:Npn \__bp_parse_pure_imaginary:
1396   {
1397     \tl_set:Nn \l_real_tl {0}
1398     \__bp_set_signed_value:NN \l_imag_tl \l__bp_input_tl
1399   }
1400
1401 % Set target to value with sign handling
1402 \cs_new_protected:Npn \__bp_set_signed_value:NN #1#2
1403   {
1404     \tl_if_empty:NTF #2
1405       { \tl_set:Nn #1 {1} }
1406       {
1407         \tl_set:Nx \l_tmpa_tl { \tl_head:N #2 }
1408         \str_if_eq:VnTF \l_tmpa_tl {-}
1409           {
1410             \tl_set:Nx \l_tmpb_tl { \tl_tail:N #2 }
1411             \tl_if_empty:NTF \l_tmpb_tl
1412               { \tl_set:Nn #1 {-1} }
1413               {
1414                 \__bp_unwrap_braces:N \l_tmpb_tl
1415                 \tl_set:Nx #1 { -( \l_tmpb_tl ) }
1416               }
1417           }
1418           {
1419             \str_if_eq:VnTF \l_tmpa_tl {+}
1420               {
1421                 \tl_set:Nx \l_tmpb_tl { \tl_tail:N #2 }
1422                 \tl_if_empty:NTF \l_tmpb_tl
1423                   { \tl_set:Nn #1 {1} }
1424                   {
1425                     \__bp_unwrap_braces:N \l_tmpb_tl
1426                     \tl_set_eq:NN #1 \l_tmpb_tl
1427                   }
1428               }
1429               {
1430                 \__bp_unwrap_braces:N #2
1431                 \tl_set_eq:NN #1 #2
1432               }
1433           }
1434       }
1435   }
1436
1437 % Helper to unwrap a single braced group
1438 \cs_new_protected:Npn \__bp_unwrap_braces:N #1
1439   {
1440     \tl_if_single_token:VF #1
1441       {
1442         \tl_set:Nx \l_tmpc_tl { \tl_count:N #1 }
1443         \int_compare:nNnT { \l_tmpc_tl } = {1}
1444           { \tl_set:Nx #1 { \exp_after:wN \use:n #1 } }
1445       }
1446   }
1447
1448 % Process one token/group to find separator
1449 \cs_new_protected:Npn \__bp_process_item:n #1
1450   {
1451     \tl_if_single_token:nTF {#1}
1452       {
1453         % Track parenthesis depth
1454         \token_if_eq_charcode:NNT #1 ( { \int_incr:N \l__bp_paren_int }
1455         \token_if_eq_charcode:NNT #1 ) { \int_decr:N \l__bp_paren_int }
```

```
        % Only consider separators at position > 0 and paren depth 0
        \int_compare:nNnT { \l__bp_pos_int } > {0}
          {
            \int_compare:nNnT { \l__bp_paren_int } = {0}
              {
                \bool_lazy_or:nnT
                  { \token_if_eq_charcode_p:NN #1 + }
                  { \token_if_eq_charcode_p:NN #1 - }
                  {
                    \bool_set_true:N \l__bp_found_bool
                    \int_set_eq:NN \l__bp_seppos_int \l__bp_pos_int
                    \tl_set:Nn \l__bp_sep_tl {#1}
                  }
              }
          }
      }
      { }
    \int_incr:N \l__bp_pos_int
  }

% Find the last + or - at depth 0
\cs_new_protected:Npn \__bp_find_separator:
  {
    \bool_set_false:N \l__bp_found_bool
    \int_zero:N \l__bp_pos_int
    \int_zero:N \l__bp_paren_int
    \tl_map_function:NN \l__bp_input_tl \__bp_process_item:n
  }

% Collect items for splitting
\cs_new_protected:Npn \__bp_collect_item:n #1
  {
    \int_compare:nNnTF { \l__bp_pos_int } < { \l__bp_seppos_int }
      { \tl_put_right:Nn \l__bp_before_tl {#1} }
      {
        \int_compare:nNnF { \l__bp_pos_int } = { \l__bp_seppos_int }
          { \tl_put_right:Nn \l__bp_after_tl {#1} }
      }
    \int_incr:N \l__bp_pos_int
  }

% Split the input at the separator position
\cs_new_protected:Npn \__bp_split_at_separator:
  {
    \tl_clear:N \l__bp_before_tl
    \tl_clear:N \l__bp_after_tl
    \int_zero:N \l__bp_pos_int

    \tl_map_function:NN \l__bp_input_tl \__bp_collect_item:n

    \__bp_unwrap_braces:N \l__bp_before_tl
    \__bp_unwrap_braces:N \l__bp_after_tl

    \tl_if_empty:NTF \l__bp_before_tl
      { \tl_set:Nn \l_real_tl {0} }
      { \tl_set_eq:NN \l_real_tl \l__bp_before_tl }

    \tl_if_empty:NTF \l__bp_after_tl
      {
        \str_if_eq:VnTF \l__bp_sep_tl {-}
          { \tl_set:Nn \l_imag_tl {-1} }
          { \tl_set:Nn \l_imag_tl {1} }
      }
```

```
      {
        \str_if_eq:VnTF \l__bp_sep_tl {-}
          { \tl_set:Nx \l_imag_tl { -( \l__bp_after_tl ) } }
          { \tl_set_eq:NN \l_imag_tl \l__bp_after_tl }
      }
  }
```

`\bp_if_contains:nnTF` Helper function to check if a token list contains a substring.

```
\tl_new:N \l__bode_check_tl
\cs_new_protected:Npn \bp_if_contains:nnTF #1#2#3#4 {
  \tl_set:Nx \l__bode_check_tl {#1}
  \tl_if_in:VnTF \l__bode_check_tl {#2} {#3} {#4}
}
```

`\bp@contains@equal` Checks if argument contains '=' to distinguish new from legacy interface.

```
\NewDocumentCommand{\bp@contains@equal}{m m m}{%
  \bp_if_contains:nnTF {#1} {=} {#2} {#3}%
}
```

`\bp@contains@num` Checks if argument contains 'numerator' keyword.

```
\NewDocumentCommand{\bp@contains@num}{m m m}{%
  \bp_if_contains:nnTF {#1} {numerator} {#2} {#3}%
}
\ExplSyntaxOff
```

`\bp@fix@add@opt` Processes options for addplot commands, organizing by approximation type.

```
\NewDocumentCommand{\bp@fix@add@opt}{ m }{%
  \gdef\bp@add@O{}
  \gdef\bp@add@tmp{}
  \foreach \approx/\opt in {#1} {
    \ifx\approx\@empty\else
    \ifnum\pdf@strcmp{\approx}{linear}=0
      \xdef\bp@add@O{\unexpanded\expandafter{\bp@add@O}%
      linear/{\unexpanded\expandafter{\opt}},}
    \else
      \ifnum\pdf@strcmp{\approx}{asymptotic}=0
        \xdef\bp@add@O{\unexpanded\expandafter{\bp@add@O}%
        asymptotic/{\unexpanded\expandafter{\opt}},}
      \else
        \ifnum\pdf@strcmp{\approx}{true}=0
          \xdef\bp@add@tmp{\unexpanded\expandafter{\bp@add@tmp}%
          \unexpanded\expandafter{\opt},}
        \else
          \xdef\bp@add@tmp{\unexpanded\expandafter{\bp@add@tmp}%
          \unexpanded\expandafter{\approx},}
        \fi
      \fi
    \fi
    \fi
  }
  \ifx\bp@add@tmp\@empty\else
    \xdef\bp@add@O{\unexpanded\expandafter{\bp@add@O}%
    true/{\unexpanded\expandafter{\bp@add@tmp}}}
  \fi
}
```

`\bp@add` Helper macro to build up magnitude and phase expressions by adding contributions from zeros, poles, gain, and delay.

```
\newcommand*{\bp@add}[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
}
```

`\bp@ZPK@plot` Builds magnitude and phase plot expressions from zero-pole-gain representation. Handles linear, asymptotic, and true Bode plot approximations.

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

`\bp@gnu@plot` Generates gnuplot commands for computing Bode plot data.

```
\newcommand{\bp@gnu@plot}[3][1]{
  \xdef\bp@gnu@cmd{ gnuplot [raw gnuplot, id=#3, prefix=\bp@prefix]
    { set table $meta;
      set dummy t;
      set logscale x 10;
      set xrange [#1*\pgfkeysvalueof{/pgfplots/domain}*#1];
      set samples \pgfkeysvalueof{/pgfplots/samples};
      plot #2;
      set table "\bp@prefix#3.table";
      plot "$meta" using ($1/(\bp@freq@scale)):($2);
    };}
}
```

`\bp@zpk@new@to@legacy` Converts the pgfkeys interface format to the legacy ZPK format. Parses complex zeros and poles using `\bp@parse@complex`.

```
\NewDocumentCommand{\bp@zpk@new@to@legacy}{ m }{%
  % Pre-expand #1 so pgfkeys sees character tokens (not a macro token).
  % This allows passing a macro whose expansion is a key=value list.
  \edef\bp@zpknl@spec{/bodeplot/zpk/.cd, reset, #1}%
  \expandafter\pgfkeys\expandafter{\bp@zpknl@spec}
  \pgfkeysgetvalue{/bodeplot/zpk/@zeros}{\bp@z}
  \edef\bp@z{\bp@z}%
  \gdef\bp@z@list{}
  \ifx\bp@z\@empty\else
    \foreach \z in \bp@z {
      \bp@parse@complex{\z}
      \xdef\bp@z@list{\unexpanded\expandafter{\bp@z@list} \bp@complex,}
    }
  \fi
  \pgfkeysgetvalue{/bodeplot/zpk/@poles}{\bp@p}
  \edef\bp@p{\bp@p}%
  \gdef\bp@p@list{}
  \ifx\bp@p\@empty\else
    \foreach \p in \bp@p {
      \bp@parse@complex{\p}
      \xdef\bp@p@list{\unexpanded\expandafter{\bp@p@list} \bp@complex,}
```

```
    }
  \fi
  \pgfkeysgetvalue{/bodeplot/zpk/@gain}{\bp@k}
  \pgfkeysgetvalue{/bodeplot/zpk/@delay}{\bp@d}
  \xdef\bp@legacy{z/{\bp@z@list},p/{\bp@p@list},k/\bp@k,d/\bp@d}
}
```

`\bp@tf@new@to@legacy` Converts the pgfkeys interface format to the legacy TF format.

```
\NewDocumentCommand{\bp@tf@new@to@legacy}{ m }{%
  % Pre-expand #1 so pgfkeys sees character tokens (not a macro token).
  \edef\bp@tfnl@spec{/bodeplot/tf/.cd, reset, #1}%
  \expandafter\pgfkeys\expandafter{\bp@tfnl@spec}
  \pgfkeysgetvalue{/bodeplot/tf/@numerator}{\bp@num}
  \pgfkeysgetvalue{/bodeplot/tf/@denominator}{\bp@den}
  \pgfkeysgetvalue{/bodeplot/tf/@delay}{\bp@d}
  \gdef\bp@num@list{}
  \ifx\bp@num\@empty\else
    \foreach \n in \bp@num {
      \xdef\bp@num@list{\unexpanded\expandafter{\bp@num@list} \n,}
    }
  \fi
  \gdef\bp@den@list{}
  \ifx\bp@den\@empty\else
    \foreach \d in \bp@den {
      \xdef\bp@den@list{\unexpanded\expandafter{\bp@den@list} \d,}
    }
  \fi
  \xdef\bp@legacy{num/{\bp@num@list},den/{\bp@den@list},d/\bp@d}
}
```

`\bp@tztpk@outfile` `\bp@tf@to@zpk` Converts a transfer-function specification to zero-pole-gain (ZPK) form by invoking a `python3` helper script via `--shell-escape`.

At the start of the document the package registers an `\AtBeginDocument` hook that writes a six-line Python driver to `\jobname-tf2zpk.py`. The script is intentionally written as a set of zero-indented one-liners so that TeX's space-collapsing rules inside `\write` arguments do not corrupt Python indentation. Characters that would require a backslash in TeX (`\`, `{`, `}`) and the newline character are obtained at runtime using `chr(92)`, `chr(123)`, `chr(125)`, and `chr(10)`, so that no `\detokenize` wrapper is needed for any line.

`\bp@tf@to@zpk` accepts a TF specification in either the new pgfkeys format (e.g. `numerator={2,10},denominator={1,3,2},delay=0`) or the legacy slash format (e.g. `num/{2,10},den/{1,3,2},d/0`), mirroring the two interfaces accepted by `\BodeTF`. The numerator and denominator coefficient lists are extracted and passed to the Python helper as comma-separated strings.

On success the macro `\edef`s into the caller-supplied output control sequence a ZPK specification string in the new pgfkeys format, e.g. `zeros={-5.0},poles={-2.0,-1.0},gain=2.000000,delay=`. The delay value is taken from the TF spec and passed through unchanged; if no delay was specified it is the empty string, which the ZPK pgfkeys family treats as zero. The result can be passed to `\BodeZPK` via `\expandafter`:

> `\expandafter\BodeZPK\expandafter[`⟨*opts*⟩`]\expandafter{`⟨*output-cs*⟩`}`

and directly to `\bp@zpk@new@to@legacy` or `\bp@ZPK@plot` via the same `\expandafter` idiom for internal use.

Usage:

> `\bp@tf@to@zpk{`⟨*tf-spec*⟩`}{`⟨`\`*zpk-cs*⟩`} {`⟨`\`*status*⟩`}`

tf-spec Transfer-function specification in either the pgfkeys format accepted by `\BodeTF` (new interface) or the legacy slash format (legacy interface).

`\zpk-cs` Control sequence `\edef`-bound on success to a ZPK spec string in the format accepted by `\BodeZPK`: `zeros={…},poles={…},gain=…,delay=….`

`\status` Control sequence `\def`-bound to `true` on success, or `false` if the Python script did not produce an output file (`python3` unavailable, missing dependencies, or no `--shell-escape`).

```
\newcommand*{\bp@tztpk@outfile}{\jobname-tf2zpk-out.tex}
\AtBeginDocument{%
  \newwrite\bp@tztpk@file
  \immediate\openout\bp@tztpk@file=\jobname-tf2zpk.py
  \immediate\write\bp@tztpk@file{import sys,numpy as np}
  \immediate\write\bp@tztpk@file{from scipy.signal import tf2zpk}
  \immediate\write\bp@tztpk@file{bs,lb,rb=chr(92),chr(123),chr(125)}
  \immediate\write\bp@tztpk@file{fmt=lambda arr:",".join([str(round(c.real,6)) if abs(rou
    5 else str(round(c.real,6))+("+"+str(round(c.imag,6)) if round(c.imag,6)>=0 else str(roun
  \immediate\write\bp@tztpk@file{try: num=[float(x) for x in sys.argv[1].split(",")]; den=
  \immediate\write\bp@tztpk@file{except Exception as e: sys.stderr.write(str(e))}
  \immediate\closeout\bp@tztpk@file
}
\NewDocumentCommand{\bp@tf@to@zpk}{ m m m }{%
  % Step 1: Extract numerator, denominator, and delay from the TF spec.
  % Supports both the new pgfkeys format (numerator={...},denominator={...},
  % delay=...) and the legacy slash format (num/{...},den/{...},d/...).
  \bp@contains@equal{#1}{%
    % New pgfkeys format
    \pgfkeys{/bodeplot/tf/.cd, reset}%
    \pgfkeys{/bodeplot/tf/.cd, #1}%
    \pgfkeysgetvalue{/bodeplot/tf/@numerator}{\bp@tztpk@num}%
    \pgfkeysgetvalue{/bodeplot/tf/@denominator}{\bp@tztpk@den}%
    \pgfkeysgetvalue{/bodeplot/tf/@delay}{\bp@tztpk@d}%
  }{%
    % Legacy slash format
    \gdef\bp@tztpk@num{}\gdef\bp@tztpk@den{}\gdef\bp@tztpk@d{}%
    \edef\bp@tztpk@list{#1}%
    \foreach \bp@tztpk@feat/\bp@tztpk@val in \bp@tztpk@list {%
      \ifnum\pdf@strcmp{\bp@tztpk@feat}{num}=0
        \xdef\bp@tztpk@num{\bp@tztpk@val}%
      \fi
      \ifnum\pdf@strcmp{\bp@tztpk@feat}{den}=0
        \xdef\bp@tztpk@den{\bp@tztpk@val}%
      \fi
      \ifnum\pdf@strcmp{\bp@tztpk@feat}{d}=0
        \xdef\bp@tztpk@d{\bp@tztpk@val}%
      \fi
    }%
  }%
  % Step 2: Invoke the Python helper.
  \ifwindows
    \immediate\write18{if exist "\bp@tztpk@outfile" del /f "\bp@tztpk@outfile"}%
  \else
    \immediate\write18{rm -f "\bp@tztpk@outfile"}%
  \fi
  \immediate\write18{python3 "\jobname-tf2zpk.py" "\bp@tztpk@num" "\bp@tztpk@den" "\bp@tz
  % Step 3: Build the ZPK spec and set status.
  \IfFileExists{\bp@tztpk@outfile}{%
    \input{\bp@tztpk@outfile}%
    \edef#2{zeros={\zpkZeros},poles={\zpkPoles},gain=\zpkGain,delay=\bp@tztpk@d}%
    \def#3{true}%
  }{%
    \def#3{false}%
  }%
}
```

\bp@TF@plot Builds magnitude and phase plot expressions from transfer function representation.

```
\newcommand{\bp@TF@plot}[3]{
  \gdef\bp@num@re{0}
  \gdef\bp@num@im{0}
  \gdef\bp@den@re{0}
  \gdef\bp@den@im{0}
  \gdef\bp@loop@delay{0}
  \edef\bp@list{#3}
  \foreach \feature/\values in \bp@list {
    \ifx\values\empty\else
      \ifnum\pdf@strcmp{\feature}{num}=0
        \foreach \numcoeff [count=\numpow] in \values {
          \ifx\numcoeff\empty\else
            \xdef\bp@num@deg{\numpow}
          \fi
        }
        \foreach \numcoeff [count=\numpow] in \values {
          \ifx\numcoeff\empty\else
            \pgfmathtruncatemacro{\currentdegree}{\bp@num@deg-\numpow}
            \ifnum\currentdegree = 0
              \xdef\bp@num@re{\bp@num@re+\numcoeff}
            \else
              \ifodd\currentdegree
                \xdef\bp@num@im{\bp@num@im+(\numcoeff*(\n@pow{-
1}{(\currentdegree-1)/2})*%
                  (\n@pow{t}{\currentdegree}))}
              \else
                \xdef\bp@num@re{\bp@num@re+(\numcoeff*(\n@pow{-
1}{(\currentdegree)/2})*%
                  (\n@pow{t}{\currentdegree}))}
              \fi
            \fi
          \fi
        }
      \fi
      \ifnum\pdf@strcmp{\feature}{den}=0
        \foreach \dencoeff [count=\denpow] in \values {
          \ifx\dencoeff\empty\else
            \xdef\bp@den@deg{\denpow}
          \fi
        }
        \foreach \dencoeff [count=\denpow] in \values {
          \ifx\dencoeff\empty\else
            \pgfmathtruncatemacro{\currentdegree}{\bp@den@deg-\denpow}
            \ifnum\currentdegree = 0
              \xdef\bp@den@re{\bp@den@re+\dencoeff}
            \else
              \ifodd\currentdegree
                \xdef\bp@den@im{\bp@den@im+(\dencoeff*(\n@pow{-
1}{(\currentdegree-1)/2})*%
                  (\n@pow{t}{\currentdegree}))}
              \else
                \xdef\bp@den@re{\bp@den@re+(\dencoeff*(\n@pow{-
1}{(\currentdegree)/2})*%
                  (\n@pow{t}{\currentdegree}))}
              \fi
            \fi
          \fi
        }
      \fi
      \ifnum\pdf@strcmp{\feature}{d}=0
        \xdef\bp@loop@delay{\values}
      \fi
```

```
    \fi
  }
  \xdef#2{((atan2((\bp@num@im),(\bp@num@re))-atan2((\bp@den@im),%
    (\bp@den@re))-\bp@loop@delay*t)*(\bp@ph@scale))}
  \xdef#1{(20*log10(sqrt((\n@pow{\bp@num@re}{2})+(\n@pow{\bp@num@im}{2})))-
  %
    20*log10(sqrt((\n@pow{\bp@den@re}{2})+(\n@pow{\bp@den@im}{2}))))}
}
```

`\bp@gnu@unwrap@plot` Generates gnuplot commands with phase unwrapping for TF plots.

```
\newcommand{\bp@gnu@unwrap@plot}[3][1]{
  \xdef\bp@gnu@cmd{ gnuplot [raw gnuplot, id=#3, prefix=\bp@prefix]
    { set table $meta;
      set dummy t;
      set logscale x 10;
      set trange [#1*\pgfkeysvalueof{/pgfplots/domain}*#1];
      set samples \pgfkeysvalueof{/pgfplots/samples};
      plot '+' using (t) : ((#2)/(\bp@ph@scale)) smooth unwrap;
      set table "\bp@prefix#3.table";
      plot "$meta" using ($1/(\bp@freq@scale)):($2*\bp@ph@scale);
    };}
}
```

`\bp@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 `\bp@ph@commands` and `\bp@mag@commands`, to be executed in appropriate `axis` environments.

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

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

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

```
  }
}
```

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

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

4.6 Nyquist plots

4.6.1 User macros

`\NyquistZPK` Converts magnitude and phase parametric functions built using `\bp@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!

```
\NewDocumentCommand{\NyquistZPK}{ O{} m G{} G{} }{%
  \pgfutil@ifempty{#3}{%
    \pgfkeys{/bodeplot/nyquist/.cd, reset}
    \pgfkeys{/bodeplot/nyquist/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/nyquist/@axes}{\bp@axes}
    \pgfkeysgetvalue{/bodeplot/nyquist/@commands}{\bp@commands}
    \pgfkeysgetvalue{/bodeplot/nyquist/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/nyquist/@prefix}{\bp@user@prefix}
    \bp@zpk@new@to@legacy{#2}
  }{%
    \bp@parse@N@opt{#1}
    \edef\bp@legacy{#2}
    \edef\bp@domain@start{#3}
    \edef\bp@domain@end{#4}
  }%
  \gdef\bp@mag{}\gdef\bp@ph{}%
  \edef\bp@cmd{\noexpand\begin{tikzpicture}
  [\unexpanded\expandafter{\bp@tikz}]}\bp@cmd
  \bp@ZPK@plot{\bp@mag}{\bp@ph}{}{\bp@legacy}%
  \edef\bp@cmd{\noexpand\begin{axis}[
      bp@style,
      domain=\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale,
      height=5cm,
      xlabel={$\Re$},
      ylabel={$\Im$},
```

```
      samples=500,
      \unexpanded\expandafter{\bp@axes}
    ]}
    \bp@cmd
      \addplot [only marks,mark=+,thick,red] (-1 , 0);
      \edef\bp@cmd{\noexpand\addplot
      [variable=t, thick, trig format plots=rad,
      \unexpanded\expandafter{\bp@plot}]}
      \if@pgfarg
        \bp@cmd
        ( {\n@pow{10}{((\bp@mag)/20)}*cos((\bp@ph)/(\bp@ph@scale))},
          {\n@pow{10}{((\bp@mag)/20)}*sin((\bp@ph)/(\bp@ph@scale))} );
        \bp@commands
      \else
        \stepcounter{bp@gnuplot@id}
        \bp@cmd gnuplot [parametric, bp@gnu@prefix] {
          \n@pow{10}{((\bp@mag)/20)}*cos((\bp@ph)/(\bp@ph@scale)),
          \n@pow{10}{((\bp@mag)/20)}*sin((\bp@ph)/(\bp@ph@scale))
        };
        \bp@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{\bp@short@list}
    \NyquistZPK@Shorthandoff
  }
  \NewDocumentCommand{\NyquistZPK@Shorthandoff}{ O{} m G{} G{} }{%
    \Orig@NyquistZPK[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\bp@short@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 `\bp@TF@plot`.

```
\NewDocumentCommand{\NyquistTF}{ O{} m G{} G{} }{%
  \pgfutil@ifempty{#3}{%
    \pgfkeys{/bodeplot/nyquist/.cd, reset}
    \pgfkeys{/bodeplot/nyquist/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/nyquist/@axes}{\bp@axes}
    \pgfkeysgetvalue{/bodeplot/nyquist/@commands}{\bp@commands}
    \pgfkeysgetvalue{/bodeplot/nyquist/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/nyquist/@prefix}{\bp@user@prefix}
    \bp@tf@new@to@legacy{#2}
  }{%
    \bp@parse@N@opt{#1}
    \edef\bp@legacy{#2}
    \edef\bp@domain@start{#3}
    \edef\bp@domain@end{#4}
  }%
  \gdef\bp@mag{}\gdef\bp@ph{}%
  \edef\bp@cmd{\noexpand\begin{tikzpicture}
  [\unexpanded\expandafter{\bp@tikz}]}\bp@cmd
  \bp@TF@plot{\bp@mag}{\bp@ph}{\bp@legacy}%
  \edef\bp@cmd{\noexpand\begin{axis}[
      bp@style,
      domain=\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale,
```

```
      height=5cm,
      xlabel={$\Re$},
      ylabel={$\Im$},
      samples=500,
      \unexpanded\expandafter{\bp@axes}
    ]}
    \bp@cmd
      \addplot [only marks, mark=+, thick, red] (-1 , 0);
      \edef\bp@cmd{\noexpand\addplot
      [variable=t, thick, trig format plots=rad,
      \unexpanded\expandafter{\bp@plot}]}
      \if@pgfarg
        \bp@cmd
        ( {\n@pow{10}{((\bp@mag)/20)}*cos((\bp@ph)/(\bp@ph@scale))},
          {\n@pow{10}{((\bp@mag)/20)}*sin((\bp@ph)/(\bp@ph@scale))} );
        \bp@commands
      \else
        \stepcounter{bp@gnuplot@id}
        \bp@cmd gnuplot [parametric, bp@gnu@prefix] {
          \n@pow{10}{((\bp@mag)/20)}*cos((\bp@ph)/(\bp@ph@scale)),
          \n@pow{10}{((\bp@mag)/20)}*sin((\bp@ph)/(\bp@ph@scale))
        };
        \bp@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{\bp@short@list}
    \NyquistTF@Shorthandoff
  }
  \NewDocumentCommand{\NyquistTF@Shorthandoff}{ O{} m G{} G{} }{%
    \Orig@NyquistTF[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\bp@short@list}
  }
  \fi
}
```

`\addNyquistZPKPlot` Adds Nyquist plot of a transfer function in ZPK form with dual interface support. Converts magnitude and phase to real and imaginary parts for parametric plotting.

```
\NewDocumentCommand{\addNyquistZPKPlot}{ O{} m }{%
  \edef\bp@spec@arg{#2}%
  \expandafter\bp@contains@equal\expandafter{\bp@spec@arg}{%
    \bp@zpk@new@to@legacy{\bp@spec@arg}}{\edef\bp@legacy{\bp@spec@arg}}%
  \gdef\bp@mag{}\gdef\bp@ph{}%
  \bp@ZPK@plot{\bp@mag}{\bp@ph}{}{\bp@legacy}%
  \if@pgfarg
    \edef\bp@cmd{\noexpand\addplot
      [domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale,
       variable=t, thick, trig format plots=rad, #1]}%
    \bp@cmd ( {\n@pow{10}{((\bp@mag)/20)}*cos((\bp@ph)/(\bp@ph@scale))},
              {\n@pow{10}{((\bp@mag)/20)}*sin((\bp@ph)/(\bp@ph@scale))} );
  \else
    \stepcounter{bp@gnuplot@id}%
    \edef\bp@cmd{\noexpand\addplot
      [domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale, thick, #1]
    \bp@cmd gnuplot [parametric, bp@gnu@prefix] {%
      \n@pow{10}{((\bp@mag)/20)}*cos((\bp@ph)/(\bp@ph@scale)),
      \n@pow{10}{((\bp@mag)/20)}*sin((\bp@ph)/(\bp@ph@scale))
```

```
    };
  \fi
}
```

`\addNyquistTFPlot` Adds Nyquist plot of a transfer function in TF form with dual interface support. Converts magnitude and phase to real and imaginary parts for parametric plotting.

```
\NewDocumentCommand{\addNyquistTFPlot}{ O{} m }{%
  \edef\bp@spec@arg{#2}%
  \expandafter\bp@contains@equal\expandafter{\bp@spec@arg}{%
    \bp@tf@new@to@legacy{\bp@spec@arg}}{\edef\bp@legacy{\bp@spec@arg}}%
  \gdef\bp@mag{}\gdef\bp@ph{}%
  \bp@TF@plot{\bp@mag}{\bp@ph}{\bp@legacy}%
  \if@pgfarg
    \edef\bp@cmd{\noexpand\addplot
      [domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale,
       variable=t, thick, trig format plots=rad, #1]}%
    \bp@cmd ( {\n@pow{10}{((\bp@mag)/20)}*cos((\bp@ph)/(\bp@ph@scale))},
              {\n@pow{10}{((\bp@mag)/20)}*sin((\bp@ph)/(\bp@ph@scale))} );
  \else
    \stepcounter{bp@gnuplot@id}%
    \edef\bp@cmd{\noexpand\addplot
      [domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale, thick, #1]
    \bp@cmd gnuplot [parametric, bp@gnu@prefix] {%
      \n@pow{10}{((\bp@mag)/20)}*cos((\bp@ph)/(\bp@ph@scale)),
      \n@pow{10}{((\bp@mag)/20)}*sin((\bp@ph)/(\bp@ph@scale))
    };
  \fi
}
```

`NyquistPlot` An environment to host `\addNyquist...` macros that pass parametric functions to `\addplot`. Uses the defaults specified in `bp@style` to create a shortcut that includes the `tikzpicture` and `axis` environments.

```
\AtBeginDocument{%
  \if@babel
    \AddToHook{env/NyquistPlot/begin}{%
      \expandafter\shorthandoff\expandafter{\bp@short@list}
    }
    \AddToHook{env/NyquistPlot/end}{%
      \expandafter\shorthandon\expandafter{\bp@short@list}
    }
  \fi
}
\NewDocumentEnvironment{NyquistPlot}{ O{} G{} G{} +b }{
  \pgfutil@ifempty{#2}{%
    \pgfkeys{/bodeplot/env/.cd, reset}
    \pgfkeys{/bodeplot/env/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/env/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/env/@prefix}{\bp@user@prefix}
  }{%
    \bp@parse@env@opt{#1}
    \edef\bp@domain@start{#2}%
    \edef\bp@domain@end{#3}%
  }
  \edef\bp@cmd{\noexpand\begin{tikzpicture}
  [\unexpanded\expandafter{\bp@tikz}]}\bp@cmd
  \edef\bp@cmd{\noexpand\begin{axis}[
    bp@style,
    height=5cm,
    domain=\bp@domain@start:\bp@domain@end,
    xlabel={$\Re$},
    ylabel={$\Im$},
    \unexpanded\expandafter{\bp@axes}%
  ]}\bp@cmd
```

```
    \addplot [only marks,mark=+,thick,red] (-1 , 0);
    #4
  \end{axis}
  \end{tikzpicture}
}{}
```

4.6.2 Internal macros

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

4.7 Nichols charts

`\NicholsZPK`

```
\NewDocumentCommand{\NicholsZPK}{ O{} m G{} G{} }{%
  \pgfutil@ifempty{#3}{%
    \pgfkeys{/bodeplot/nichols/.cd, reset}
    \pgfkeys{/bodeplot/nichols/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/nichols/@axes}{\bp@axes}
    \pgfkeysgetvalue{/bodeplot/nichols/@commands}{\bp@commands}
    \pgfkeysgetvalue{/bodeplot/nichols/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/nichols/@prefix}{\bp@user@prefix}
```

```
    \bp@zpk@new@to@legacy{#2}
  }{%
    \bp@parse@N@opt{#1}
    \edef\bp@legacy{#2}
    \edef\bp@domain@start{#3}
    \edef\bp@domain@end{#4}
  }%
  \gdef\bp@mag{}\gdef\bp@ph{}%
  \edef\bp@cmd{\noexpand\begin{tikzpicture}
  [\unexpanded\expandafter{\bp@tikz}]}\bp@cmd
  \bp@ZPK@plot{\bp@mag}{\bp@ph}{}{\bp@legacy}%
    \edef\bp@cmd{\noexpand\begin{axis}[
      bp@ph@x@label,
      bp@style,
      domain=\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale,
      height=5cm,
      ylabel={Gain (dB)},
      samples=500,
      \unexpanded\expandafter{\bp@axes}
    ]}
    \bp@cmd
      \edef\bp@cmd{\noexpand\addplot
      [variable=t, thick, trig format plots=rad,
      \unexpanded\expandafter{\bp@plot}]}
      \if@pgfarg
        \bp@cmd ( {\bp@ph} , {\bp@mag} );
        \bp@commands
      \else
        \stepcounter{bp@gnuplot@id}
        \bp@cmd gnuplot [raw gnuplot, bp@gnu@prefix]
        { set table $meta;
          set logscale x 10;
          set dummy t;
          set samples \pgfkeysvalueof{/pgfplots/samples};
          set trange [\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale];
          plot '+' using (\bp@mag) : ((\bp@ph)/(\bp@ph@scale));
          unset logscale x;
          set table "\bp@prefix\arabic{bp@gnuplot@id}.table";
          plot "$meta" using ($2*\bp@ph@scale):($1);
        };
        \bp@commands
      \fi
    \end{axis}
  \end{tikzpicture}
}
\AtBeginDocument{%
  \if@babel
  \let\Orig@NicholsZPK\NicholsZPK
  \renewcommand{\NicholsZPK}{%
    \expandafter\shorthandoff\expandafter{\bp@short@list}
    \NicholsZPK@Shorthandoff
  }
  \NewDocumentCommand{\NicholsZPK@Shorthandoff}{ O{} m G{} G{} }{%
    \Orig@NicholsZPK[#1]{#2}{#3}{#4}
    \expandafter\shorthandon\expandafter{\bp@short@list}
  }
  \fi
}
```

\NicholsTF

```
\NewDocumentCommand{\NicholsTF}{ O{} m G{} G{} }{%
  \if@pgfarg
    \bp@tf@to@zpk{#2}{\bp@tmp@zpk}{\bp@tmp@status}
```

```
    \ifnum\pdf@strcmp{\bp@tmp@status}{true}=0
      \pgfutil@ifempty{#3}{%
        \NicholsZPK[#1]{\bp@tmp@zpk}%
      }{%
        \NicholsZPK[domain=#3:#4,#1]{\bp@tmp@zpk}%
      }%
      \def\bp@fallback{false}
    \else
      \def\bp@fallback{true}
    \fi
  \else
    \def\bp@fallback{true}
  \fi
  \ifnum\pdf@strcmp{\bp@fallback}{true}=0
    \pgfutil@ifempty{#3}{%
      \pgfkeys{/bodeplot/nichols/.cd, reset}
      \pgfkeys{/bodeplot/nichols/.cd, #1}
      \pgfkeysgetvalue{/bodeplot/nichols/@axes}{\bp@axes}
      \pgfkeysgetvalue{/bodeplot/nichols/@commands}{\bp@commands}
      \pgfkeysgetvalue{/bodeplot/nichols/@tikz}{\bp@tikz}
      \pgfkeysgetvalue{/bodeplot/nichols/@prefix}{\bp@user@prefix}
      \bp@tf@new@to@legacy{#2}
    }{%
      \bp@parse@N@opt{#1}
      \edef\bp@legacy{#2}
      \edef\bp@domain@start{#3}
      \edef\bp@domain@end{#4}
    }%
    \gdef\bp@mag{}\gdef\bp@ph{}%
    \edef\bp@cmd{\noexpand\begin{tikzpicture}
    [\unexpanded\expandafter{\bp@tikz}]}\bp@cmd
    \bp@TF@plot{\bp@mag}{\bp@ph}{\bp@legacy}%
    \edef\bp@cmd{\noexpand\begin{axis}[
        bp@ph@x@label,
        bp@style,
        domain=\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale,
        height=5cm,
        ylabel={Gain (dB)},
        samples=500,
        \unexpanded\expandafter{\bp@axes}
      ]}
      \bp@cmd
        \edef\bp@cmd{\noexpand\addplot
        [variable=t, thick, trig format plots=rad,
        \unexpanded\expandafter{\bp@plot}]}%
        \if@pgfarg
          \bp@cmd ( {\n@mod{\bp@ph}{2*pi*\bp@ph@scale}} , {\bp@mag} );
          \bp@commands
        \else
          \stepcounter{bp@gnuplot@id}
          \bp@cmd gnuplot [raw gnuplot, bp@gnu@prefix]
            { set table $meta1;
              set logscale x 10;
              set dummy t;
              set samples \pgfkeysvalueof{/pgfplots/samples};
              set trange [\bp@domain@start*\bp@freq@scale:\bp@domain@end*\bp@freq@scale];
              plot '+' using (\bp@mag) : ((\bp@ph)/(\bp@ph@scale));
              unset logscale x;
              set table $meta2;
              plot "$meta1" using ($1):($2) smooth unwrap;
              set table "\bp@prefix\arabic{bp@gnuplot@id}.table";
              plot "$meta2" using ($2*\bp@ph@scale):($1);
            };
```

```
          \bp@commands
        \fi
      \end{axis}
    \end{tikzpicture}
  \fi
}
```

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

```
\AtBeginDocument{
  \if@babel
    \let\Orig@NicholsTF\NicholsTF
    \renewcommand{\NicholsTF}{%
      \expandafter\shorthandoff\expandafter{\bp@short@list}
      \NicholsTF@Shorthandoff
    }
    \NewDocumentCommand{\NicholsTF@Shorthandoff}{ O{} m G{} G{} }{%
      \Orig@NicholsTF[#1]{#2}{#3}{#4}
      \expandafter\shorthandon\expandafter{\bp@short@list}
    }
    \AddToHook{env/NicholsChart/begin}{%
      \expandafter\shorthandoff\expandafter{\bp@short@list}
    }
    \AddToHook{env/NicholsChart/end}{%
      \expandafter\shorthandon\expandafter{\bp@short@list}
    }
  \fi
}
```

NicholsChart

```
\NewDocumentEnvironment{NicholsChart}{ O{} G{} G{} +b }{
  \pgfutil@ifempty{#2}{%
    \pgfkeys{/bodeplot/env/.cd, reset}
    \pgfkeys{/bodeplot/env/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/env/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/env/@prefix}{\bp@user@prefix}
  }{%
    \bp@parse@env@opt{#1}
    \edef\bp@domain@start{#2}
    \edef\bp@domain@end{#3}
  }
  \edef\bp@cmd{\noexpand\begin{tikzpicture}
  [\unexpanded\expandafter{\bp@tikz}]}\bp@cmd
  \edef\bp@cmd{\noexpand\begin{axis}[
    bp@ph@x@label,
    bp@style,
    domain=\bp@domain@start:\bp@domain@end,
    height=5cm,
    ylabel={Gain (dB)},
    \unexpanded\expandafter{\bp@axes}%
  ]}\bp@cmd
    #4
  \end{axis}
  \end{tikzpicture}
}{}
```

\addNicholsZPKChart Generates Nichols chart for ZPK system with dual interface support. Plots phase vs magnitude.

```
\NewDocumentCommand{\addNicholsZPKChart}{ O{} m }{%
  \edef\bp@spec@arg{#2}%
  \expandafter\bp@contains@equal\expandafter{\bp@spec@arg}{%
    \bp@zpk@new@to@legacy{\bp@spec@arg}}{\edef\bp@legacy{\bp@spec@arg}}%
  \gdef\bp@mag{}\gdef\bp@ph{}%
  \bp@ZPK@plot{\bp@mag}{\bp@ph}{}{\bp@legacy}%
  \if@pgfarg
```

```
    \edef\bp@cmd{\noexpand\addplot
      [domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale,
       variable=t, thick, trig format plots=rad, #1]}%
    \bp@cmd ( {\bp@ph} , {\bp@mag} );
  \else
    \stepcounter{bp@gnuplot@id}%
    \addplot [thick, #1] gnuplot [raw gnuplot, bp@gnu@prefix] {%
      set table $meta;
      set logscale x 10;
      set dummy t;
      set samples \pgfkeysvalueof{/pgfplots/samples};
      set trange [\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale];
      plot '+' using (\bp@mag) : ((\bp@ph)/(\bp@ph@scale));
      unset logscale x;
      set table "\bp@prefix\arabic{bp@gnuplot@id}.table";
      plot "$meta" using ($2*\bp@ph@scale):($1);
    };
  \fi
}
```

\addNicholsTFChart Generates Nichols chart for TF system with dual interface support. Plots phase vs magnitude with phase unwrapping.

```
\NewDocumentCommand{\addNicholsTFChart}{ O{} m }{%
  \edef\bp@spec@arg{#2}%
  \expandafter\bp@contains@equal\expandafter{\bp@spec@arg}{%
    \bp@tf@new@to@legacy{\bp@spec@arg}}{\edef\bp@legacy{\bp@spec@arg}}%
  \gdef\bp@mag{}\gdef\bp@ph{}%
  \bp@TF@plot{\bp@mag}{\bp@ph}{\bp@legacy}%
  \if@pgfarg
    \edef\bp@cmd{\noexpand\addplot
      [domain=\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale,
       variable=t, thick, trig format plots=rad, #1]}%
    \bp@cmd ( {\n@mod{\bp@ph}{2*pi*\bp@ph@scale}} , {\bp@mag} );
  \else
    \stepcounter{bp@gnuplot@id}%
    \addplot [thick, #1] gnuplot [raw gnuplot, bp@gnu@prefix] {%
      set table $meta1;
      set logscale x 10;
      set dummy t;
      set samples \pgfkeysvalueof{/pgfplots/samples};
      set trange [\bp@freq@scale*\pgfkeysvalueof{/pgfplots/domain}*\bp@freq@scale];
      plot '+' using (\bp@mag) : ((\bp@ph)/(\bp@ph@scale));
      unset logscale x;
      set table $meta2;
      plot "$meta1" using ($1):($2) smooth unwrap;
      set table "\bp@prefix\arabic{bp@gnuplot@id}.table";
      plot "$meta2" using ($2*\bp@ph@scale):($1);
    };
  \fi
}
```

4.8 Pole-zero maps

4.8.1 User macros

\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.

```
\NewDocumentCommand{\PoleZeroMapZPK}{ O{} m }{%
\edef\bp@spec@arg{#2}%
\expandafter\bp@contains@equal\expandafter{\bp@spec@arg}{%
    \pgfkeys{/bodeplot/pzmap/.cd, reset}
```

```
    \pgfkeys{/bodeplot/pzmap/.cd, #1}
    \pgfkeysgetvalue{/bodeplot/pzmap/@axes}{\bp@axes}
    \pgfkeysgetvalue{/bodeplot/pzmap/@plot}{\bp@plot}
    \pgfkeysgetvalue{/bodeplot/pzmap/@commands}{\bp@commands}
    \pgfkeysgetvalue{/bodeplot/pzmap/@tikz}{\bp@tikz}
    \pgfkeysgetvalue{/bodeplot/pzmap/@prefix}{\bp@user@prefix}
    \pgfkeysgetvalue{/bodeplot/pzmap/@scale}{\bp@scale}
    \bp@zpk@new@to@legacy{\bp@spec@arg}
  }{%
    \bp@parse@N@opt{#1}
    \edef\bp@legacy{\bp@spec@arg}
  }%
  \ifnum\pdf@strcmp{\bp@scale}{log}=0
    \bp@min@real@ZPK{\bp@legacy}
    \bp@min@im@ZPK{\bp@legacy}
    \pgfkeys{/pgf/fpu=true}
    \ifx\bp@has@positive@values\@bp@min@false
      \xdef\bp@PZZPK@ticksXPos{0}
    \else
      \pgfmathparse{max(\bp@max@re@pos@pow@10 - \bp@min@re@pow@10 + 1, 1)}
      \pgfmathfloattoint{\pgfmathresult}
      \xdef\bp@PZZPK@ticksXPos{\pgfmathresult}
    \fi
    \ifx\bp@has@negative@values\@bp@min@false
      \xdef\bp@PZZPK@ticksXNeg{0}
    \else
      \pgfmathparse{max(\bp@max@re@neg@pow@10 - \bp@min@re@pow@10 + 1, 1)}
      \pgfmathfloattoint{\pgfmathresult}
      \xdef\bp@PZZPK@ticksXNeg{\pgfmathresult}
    \fi
    \pgfkeys{/pgf/fpu=false}
    \def\PoleZeroMapZPK@formatXTick##1{%
      \pgfmathtruncatemacro{\PoleZeroMapZPK@tick}{##1}%
      \ifnum\PoleZeroMapZPK@tick=0
        $0$
      \else
        \pgfmathtruncatemacro{\PoleZeroMapZPK@exp}
        {\bp@min@re@pow@10 + abs(\PoleZeroMapZPK@tick) - 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}
        {\bp@min@im@pow@10 + abs(\PoleZeroMapZPK@tick) - 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\bp@cmd{\noexpand\begin{tikzpicture}
```

```
    [\unexpanded\expandafter{\bp@tikz}]}
  \bp@cmd
    \ifnum\pdf@strcmp{\bp@scale}{log}=0
      \edef\bp@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) < \bp@min@re@threshold@result ?
        0 : (x >= 0 ? (log10(max(min(x, 1e100), 1e-
100)) - \bp@min@re@pow@10 + 1) :
        (-log10(max(min(-x, 1e100), 1e-100)) + \bp@min@re@pow@10 - 1))},
        y filter/.expression={abs(y) < \bp@min@im@threshold@result ?
        0 : (y >= 0 ? (log10(max(min(y, 1e100), 1e-
100)) - \bp@min@im@pow@10 + 1) :
        (-log10(max(min(-y, 1e100), 1e-100)) + \bp@min@im@pow@10 - 1))},
        \unexpanded\expandafter{\bp@axes}
      ]}
    \else
      \edef\bp@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{\bp@axes}
      ]}
    \fi
    \bp@cmd
      \foreach \feature/\values in \bp@legacy {
        \ifnum\pdf@strcmp{\feature}{z}=0
          \foreach \z in \values {
            \foreach \y [count=\zcnt] in \z {
              \ifnum\zcnt=1
                \xdef\bp@zre{\y}
              \fi
              \ifnum\zcnt=2
                \xdef\bp@zim{\y}
              \fi
              \xdef\bp@Last@Loop@z{\zcnt}
            }
            \ifnum\bp@Last@Loop@z=1
              \xdef\bp@zim{0}
            \fi
            \edef\bp@cmd{\noexpand\addplot
            [only marks, mark=o, mark size=3pt, thick, blue,
            \unexpanded\expandafter{\bp@plot}]}
            \bp@cmd coordinates {(\bp@zre,\bp@zim)};
          }
        \fi
        \ifnum\pdf@strcmp{\feature}{p}=0
          \foreach \p in \values {
            \foreach \y [count=\pcnt] in \p {
              \ifnum\pcnt=1
```

```
                \xdef\bp@pre{\y}
              \fi
              \ifnum\pcnt=2
                \xdef\bp@pim{\y}
              \fi
              \xdef\bp@Last@Loop@p{\pcnt}
            }
            \ifnum\bp@Last@Loop@p=1
              \xdef\bp@pim{0}
            \fi
            \edef\bp@cmd{\noexpand\addplot
            [only marks, mark=x, mark size=3pt, thick, red,
            \unexpanded\expandafter{\bp@plot}]}
            \bp@cmd coordinates {(\bp@pre,\bp@pim)};
          }
        \fi
      }
      \bp@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{\bp@short@list}
    \PoleZeroMapZPK@Shorthandoff
  }
  \newcommand{\PoleZeroMapZPK@Shorthandoff}[2][]{%
    \Orig@PoleZeroMapZPK[#1]{#2}
    \expandafter\shorthandon\expandafter{\bp@short@list}
  }
  \fi
}
```

`\PoleZeroMapTF` Similar to `\PoleZeroMapZPK` but for the system specified in the TF format. This macro converts the TF specification to ZPK format internally using Python via `--shell-escape` and then calls `\PoleZeroMapZPK` to generate the plot. If Python with the `scipy` and `numpy` libraries is not available on the shell, the package returns an error message.

```
\NewDocumentCommand{\PoleZeroMapTF}{ O{} m }{%
  \bp@tf@to@zpk{#2}{\bp@tmp@zpk}{\bp@tmp@status}
  \ifnum\pdf@strcmp{\bp@tmp@status}{true}=0
    \PoleZeroMapZPK[#1]{\bp@tmp@zpk}
  \else
    \PackageError{bodeplot}{Python with scipy and numpy is required to convert TF to ZPK
  zero map. Please ensure Python is installed and accessible via --shell-
  escape.}{}
  \fi
}
```

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

```
\AtBeginDocument{%
  \if@babel
  \let\Orig@PoleZeroMapTF\PoleZeroMapTF
  \renewcommand{\PoleZeroMapTF}{%
    \expandafter\shorthandoff\expandafter{\bp@short@list}
    \PoleZeroMapTF@Shorthandoff
  }
  \newcommand{\PoleZeroMapTF@Shorthandoff}[2][]{%
    \Orig@PoleZeroMapTF[#1]{#2}
    \expandafter\shorthandon\expandafter{\bp@short@list}
```

```
  }
  \fi
}
```

4.8.2 Internal macros

\bp@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 \bp@min@re@threshold@result and the corresponding power of 10 is stored in \bp@min@re@pow@10.

```
\newcommand{\bp@min@real@ZPK}[1]{
  \gdef\bp@min@re@threshold@result{1000}
  \def\@bp@min@false{false}
  \gdef\bp@min@threshold@found{false}
  \global\let\bp@min@thresh@float\relax
  \pgfkeys{/pgf/fpu=true}
  \pgfmathparse{0}
  \global\let\bp@max@re@float=\pgfmathresult
  \pgfmathparse{0}
  \global\let\bp@max@re@pos@float=\pgfmathresult
  \pgfmathparse{0}
  \global\let\bp@max@re@neg@float=\pgfmathresult
  \pgfkeys{/pgf/fpu=false}
  \gdef\bp@max@re@value{0}
  \gdef\bp@has@positive@values{false}
  \gdef\bp@has@negative@values{false}
  \foreach \feature/\values in {#1} {
    \ifnum\pdf@strcmp{\feature}{z}=0
      \foreach \z in \values {
        \foreach \y [count=\zcnt] in \z {
          \ifnum\zcnt=1
            \pgfkeys{/pgf/fpu=true}
            \pgfmathparse{abs(\y)}
            \let\abs@valuefloat=\pgfmathresult
            \pgfmathfloattofixed{\abs@valuefloat}
            \edef\abs@value{\pgfmathresult}
            \pgfkeys{/pgf/fpu=false}
            \ifnum\pdf@strcmp{\abs@value}{0}=0\else
              \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\y >= 0 ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  \gdef\bp@has@positive@values{true}
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat > \bp@max@re@pos@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \global\let\bp@max@re@pos@float=\abs@valuefloat
                  \fi
                \else
                  \gdef\bp@has@negative@values{true}
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat > \bp@max@re@neg@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \global\let\bp@max@re@neg@float=\abs@valuefloat
                  \fi
                \fi
                \pgfkeys{/pgf/fpu=true}
```

```
              \pgfmathparse{\abs@valuefloat > \bp@max@re@float ? 1 : 0}
              \pgfmathfloattoint{\pgfmathresult}
              \pgfkeys{/pgf/fpu=false}
              \ifnum\pgfmathresult=1
                \global\let\bp@max@re@float=\abs@valuefloat
                \xdef\bp@max@re@value{\abs@value}
              \fi
              \ifx\bp@min@threshold@found\@bp@min@false
                \xdef\bp@min@re@threshold@result{\abs@value}
                \global\let\bp@min@thresh@float=\abs@valuefloat
                \gdef\bp@min@threshold@found{true}
              \else
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\abs@valuefloat < \bp@min@thresh@float ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  \xdef\bp@min@re@threshold@result{\abs@value}
                  \global\let\bp@min@thresh@float=\abs@valuefloat
                \fi
              \fi
            \fi
          \fi
        \fi
      }
    }
  \fi
  \ifnum\pdf@strcmp{\feature}{p}=0
    \foreach \p in \values {
      \foreach \y [count=\pcnt] in \p {
        \ifnum\pcnt=1
          \pgfkeys{/pgf/fpu=true}
          \pgfmathparse{abs(\y)}
          \let\abs@valuefloat=\pgfmathresult
          \pgfmathfloattofixed{\abs@valuefloat}
          \edef\abs@value{\pgfmathresult}
          \pgfkeys{/pgf/fpu=false}
          \ifnum\pdf@strcmp{\abs@value}{0}=0\else
            \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
              \pgfkeys{/pgf/fpu=true}
              \pgfmathparse{\y >= 0 ? 1 : 0}
              \pgfmathfloattoint{\pgfmathresult}
              \pgfkeys{/pgf/fpu=false}
              \ifnum\pgfmathresult=1
                \gdef\bp@has@positive@values{true}
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\abs@valuefloat > \bp@max@re@pos@float ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  \global\let\bp@max@re@pos@float=\abs@valuefloat
                \fi
              \else
                \gdef\bp@has@negative@values{true}
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\abs@valuefloat > \bp@max@re@neg@float ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  \global\let\bp@max@re@neg@float=\abs@valuefloat
                \fi
              \fi
              \ifx\bp@min@threshold@found\@bp@min@false
```

```
                  \xdef\bp@min@re@threshold@result{\abs@value}
                  \global\let\bp@min@thresh@float=\abs@valuefloat
                  \gdef\bp@min@threshold@found{true}
                \else
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat < \bp@min@thresh@float ? 1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \xdef\bp@min@re@threshold@result{\abs@value}
                    \global\let\bp@min@thresh@float=\abs@valuefloat
                  \fi
                \fi
              \fi
            \fi
          \fi
        }
      }
    \fi
  }
  \ifx\bp@min@threshold@found\@bp@min@false
    \gdef\bp@min@re@threshold@result{0.01}
  \fi
  \xdef\bp@min@threshold@result{\bp@min@re@threshold@result}
  \pgfkeys{/pgf/fpu=true}
  \pgfmathparse{log10(\bp@min@re@threshold@result)}
  \let\log@result=\pgfmathresult
  \pgfmathparse{\log@result + 1e-5}
  \let\log@adjusted=\pgfmathresult
  \pgfmathparse{floor(\log@adjusted)}
  \pgfmathfloattofixed{\pgfmathresult}
  \xdef\bp@min@re@pow@10{\pgfmathresult}
  \xdef\bp@min@pow@10{\bp@min@re@pow@10}
  \ifx\bp@has@positive@values\@bp@min@false
    \xdef\bp@max@re@pos@pow@10{\bp@min@re@pow@10}
  \else
    \pgfmathparse{log10(max(\bp@max@re@pos@float,1e-100))}
    \let\log@bp@max@re@pos=\pgfmathresult
    \pgfmathparse{\log@bp@max@re@pos + 1e-5}
    \let\log@bp@max@re@pos@adjusted=\pgfmathresult
    \pgfmathparse{ceil(\log@bp@max@re@pos@adjusted)}
    \pgfmathfloattoint{\pgfmathresult}
    \xdef\bp@max@re@pos@pow@10{\pgfmathresult}
  \fi
  \ifx\bp@has@negative@values\@bp@min@false
    \xdef\bp@max@re@neg@pow@10{\bp@min@re@pow@10}
  \else
    \pgfmathparse{log10(max(\bp@max@re@neg@float,1e-100))}
    \let\log@bp@max@re@neg=\pgfmathresult
    \pgfmathparse{\log@bp@max@re@neg + 1e-5}
    \let\log@bp@max@re@neg@adjusted=\pgfmathresult
    \pgfmathparse{ceil(\log@bp@max@re@neg@adjusted)}
    \pgfmathfloattoint{\pgfmathresult}
    \xdef\bp@max@re@neg@pow@10{\pgfmathresult}
  \fi
  \pgfmathparse{max(\bp@max@re@pos@float > 0 ?
  \bp@max@re@pos@float : 0, \bp@max@re@neg@float > 0 ? \bp@max@re@neg@float : 0)}
  \let\bp@max@re@valuefloat=\pgfmathresult
  \pgfmathparse{\bp@max@re@valuefloat > 0 ?
  \bp@max@re@valuefloat : \bp@min@re@threshold@result}
  \let\bp@max@re@valuefloat=\pgfmathresult
  \pgfmathparse{log10(max(\bp@max@re@valuefloat,1e-100))}
  \let\log@bp@max@re=\pgfmathresult
```

```
  \pgfmathparse{\log@bp@max@re + 1e-5}
  \let\log@bp@max@re@adjusted=\pgfmathresult
  \pgfmathparse{ceil(\log@bp@max@re@adjusted)}
  \pgfmathfloattoint{\pgfmathresult}
  \xdef\bp@max@re@pow@10{\pgfmathresult}
  \pgfkeys{/pgf/fpu=false}
}
```

`\bp@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 `\bp@min@im@threshold@result` and the corresponding power of 10 is stored in `\bp@min@im@pow@10`.

```
\newcommand{\bp@min@im@ZPK}[1]{
  \gdef\bp@min@im@threshold@result{1000}
  \def\@bp@min@false{false}
  \gdef\bp@min@im@threshold@found{false}
  \global\let\bp@min@im@thresh@float\relax
  \pgfkeys{/pgf/fpu=true}
  \pgfmathparse{0}
  \global\let\bp@max@im@float=\pgfmathresult
  \pgfkeys{/pgf/fpu=false}
  \gdef\bp@max@im@value{0}
  \foreach \feature/\values in {#1} {
    \ifnum\pdf@strcmp{\feature}{z}=0
      \foreach \z in \values {
        \foreach \y [count=\zcnt] in \z {
          \ifnum\zcnt=2
            \pgfkeys{/pgf/fpu=true}
            \pgfmathparse{abs(\y)}
            \let\abs@valuefloat=\pgfmathresult
            \pgfmathfloattofixed{\abs@valuefloat}
            \edef\abs@value{\pgfmathresult}
            \pgfkeys{/pgf/fpu=false}
            \ifnum\pdf@strcmp{\abs@value}{0}=0\else
              \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
                \pgfkeys{/pgf/fpu=true}
                \pgfmathparse{\abs@valuefloat > \bp@max@im@float ? 1 : 0}
                \pgfmathfloattoint{\pgfmathresult}
                \pgfkeys{/pgf/fpu=false}
                \ifnum\pgfmathresult=1
                  \global\let\bp@max@im@float=\abs@valuefloat
                  \xdef\bp@max@im@value{\abs@value}
                \fi
                \ifx\bp@min@im@threshold@found\@bp@min@false
                  \xdef\bp@min@im@threshold@result{\abs@value}
                  \global\let\bp@min@im@thresh@float=\abs@valuefloat
                  \gdef\bp@min@im@threshold@found{true}
                \else
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat < \bp@min@im@thresh@float ?
                  1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \xdef\bp@min@im@threshold@result{\abs@value}
                    \global\let\bp@min@im@thresh@float=\abs@valuefloat
                  \fi
                \fi
              \fi
            \fi
          \fi
        }
```

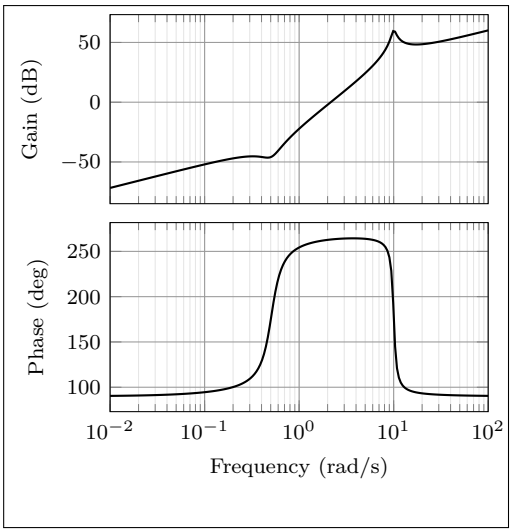
```
      }
    \fi
    \ifnum\pdf@strcmp{\feature}{p}=0
      \foreach \p in \values {
        \foreach \y [count=\pcnt] in \p {
          \ifnum\pcnt=2
            \pgfkeys{/pgf/fpu=true}
            \pgfmathparse{abs(\y)}
            \let\abs@valuefloat=\pgfmathresult
            \pgfmathfloattofixed{\abs@valuefloat}
            \edef\abs@value{\pgfmathresult}
            \pgfkeys{/pgf/fpu=false}
            \ifnum\pdf@strcmp{\abs@value}{0}=0\else
              \ifnum\pdf@strcmp{\abs@value}{0.0}=0\else
                \ifx\bp@min@im@threshold@found\@bp@min@false
                  \xdef\bp@min@im@threshold@result{\abs@value}
                  \global\let\bp@min@im@thresh@float=\abs@valuefloat
                  \gdef\bp@min@im@threshold@found{true}
                \else
                  \pgfkeys{/pgf/fpu=true}
                  \pgfmathparse{\abs@valuefloat < \bp@min@im@thresh@float ?
                  1 : 0}
                  \pgfmathfloattoint{\pgfmathresult}
                  \pgfkeys{/pgf/fpu=false}
                  \ifnum\pgfmathresult=1
                    \xdef\bp@min@im@threshold@result{\abs@value}
                    \global\let\bp@min@im@thresh@float=\abs@valuefloat
                  \fi
                \fi
              \fi
            \fi
          \fi
        }
      }
    \fi
  }
  \ifx\bp@min@im@threshold@found\@bp@min@false
    \gdef\bp@min@im@threshold@result{0.01}
  \fi
  \pgfkeys{/pgf/fpu=true}
  \pgfmathparse{log10(\bp@min@im@threshold@result)}
  \let\log@result=\pgfmathresult
  \pgfmathparse{\log@result + 1e-5}
  \let\log@adjusted=\pgfmathresult
  \pgfmathparse{floor(\log@adjusted)}
  \pgfmathfloattofixed{\pgfmathresult}
  \xdef\bp@min@im@pow@10{\pgfmathresult}
  \xdef\bp@min@Im@pow@10{\bp@min@im@pow@10}
  \pgfmathparse{\bp@max@im@float > 0 ?
  \bp@max@im@float : \bp@min@im@threshold@result}
  \let\bp@max@im@valuefloat=\pgfmathresult
  \pgfmathparse{log10(max(\bp@max@im@valuefloat,1e-100))}
  \let\log@bp@max@im=\pgfmathresult
  \pgfmathparse{\log@bp@max@im + 1e-5}
  \let\log@bp@max@im@adjusted=\pgfmathresult
  \pgfmathparse{ceil(\log@bp@max@im@adjusted)}
  \pgfmathfloattoint{\pgfmathresult}
  \xdef\bp@max@im@pow@10{\pgfmathresult}
  \pgfkeys{/pgf/fpu=false}
}
```

A TL;DR for package version 2.1.1 and earlier

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

$$G(s) = 10\frac{s(s+0.1+0.5\mathrm{i})(s+0.1-0.5\mathrm{i})}{(s+0.5+10\mathrm{i})(s+0.5-10\mathrm{i})} = \frac{s(10s^2+2s+2.6)}{(s^2+s+100.25)}. \tag{6}$$

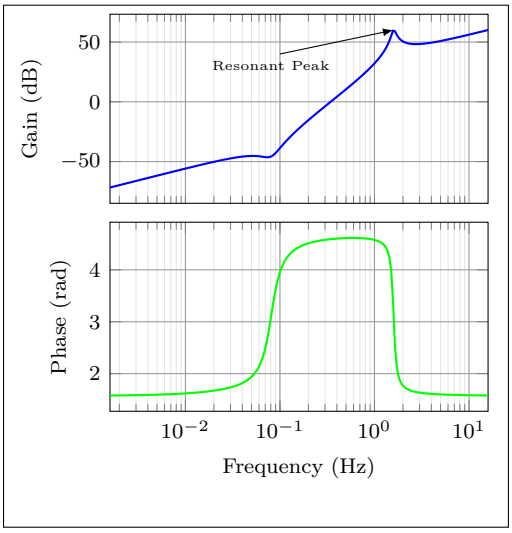
Bode plot in ZPK format



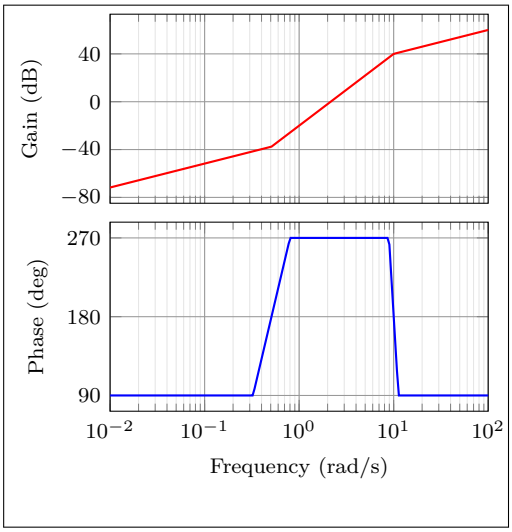

```
\BodeZPK{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
{0.01}
{100}
```

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)}
```



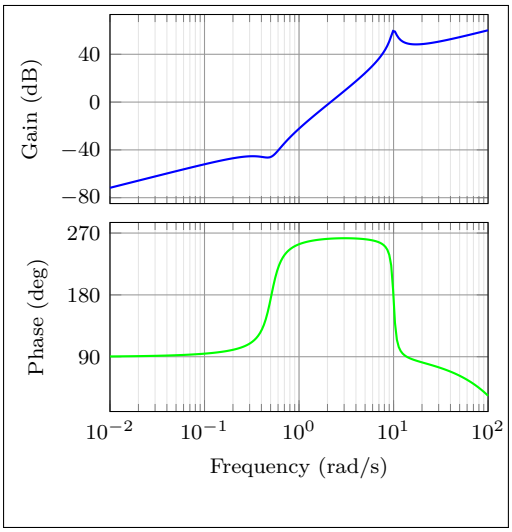
Linear approximation with customization



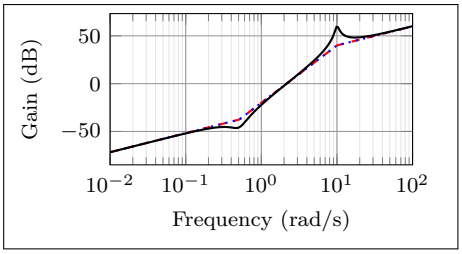

```
\BodeZPK[%
  plot/mag/{red,thick},
  plot/ph/{blue,thick},
  axes/mag/{ytick distance=40},
  axes/ph/{ytick distance=90},
  approx/linear%
]{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10%
}
{0.01}
{100}
```

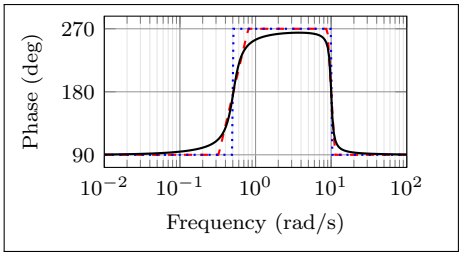
Plot with delay and customization

```
\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



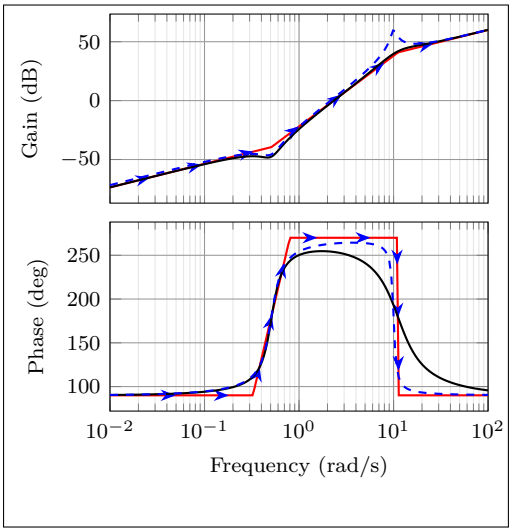



```
\begin{BodeMagPlot}[%
  axes/{height=2cm,
  width=4cm}%
]
{0.01}
{100}
  \addBodeZPKPlots[%
    true/{black,thick},
    linear/{red,dashed,thick},
    asymptotic/{blue,dotted,thick}%
  ]
  {magnitude}
  {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10%
  }
\end{BodeMagPlot}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

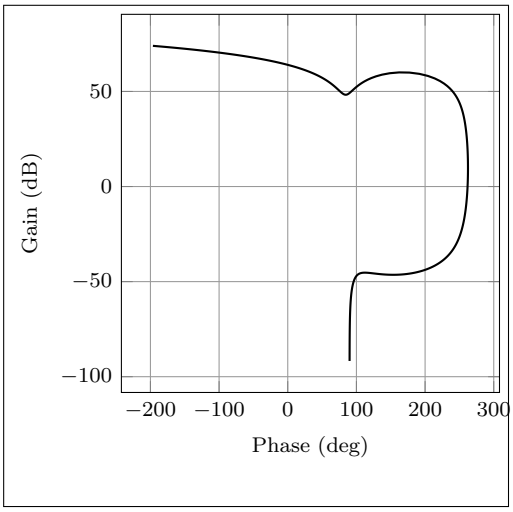
```
\begin{BodePhPlot}[%
  height=2cm,
  width=4cm,
  ytick distance=90
]
{0.01}
{100}
  \addBodeZPKPlots[%
    true/{black,thick},
    linear/{red,dashed,thick},
    asymptotic/{blue,dotted,thick}%
  ]
  {phase}
  {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10%
  }
\end{BodePhPlot}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

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,postaction=decorate,
    decoration={%
      markings,
      mark=between positions 0.1 and 0.9 step 2em with {%
        \arrow{Stealth [length=2mm, blue]}
      }
    },linear]{zpk}{%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-5,-10},{-5,10}},
    k/10%
  }
  \addBodePlot[black,thick]{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}
```



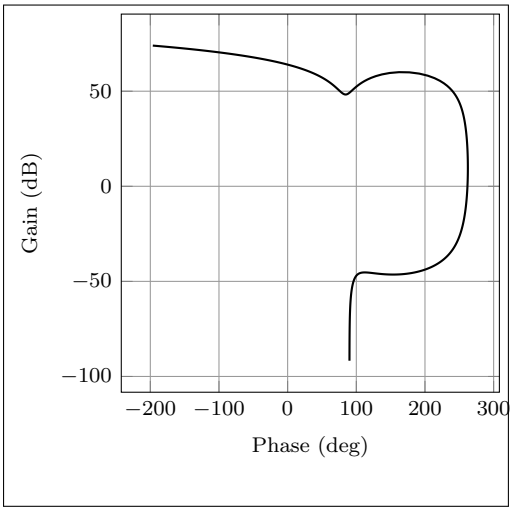
Nichols chart



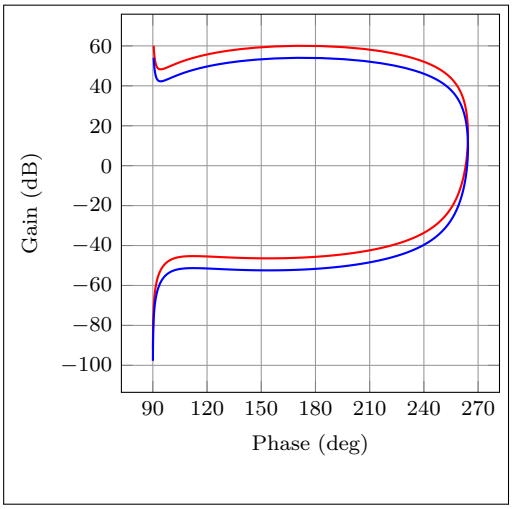

```
\NicholsZPK[samples=1000]
{%
  z/{0,{-0.1,-0.5},{-0.1,0.5}},
  p/{{-0.5,-10},{-0.5,10}},
  k/10,
  d/0.01%
}
{0.001}
{500}
```

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}
```



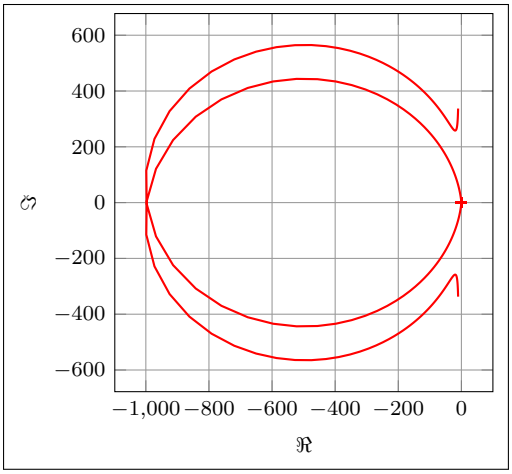
Multiple Nichols charts with customization



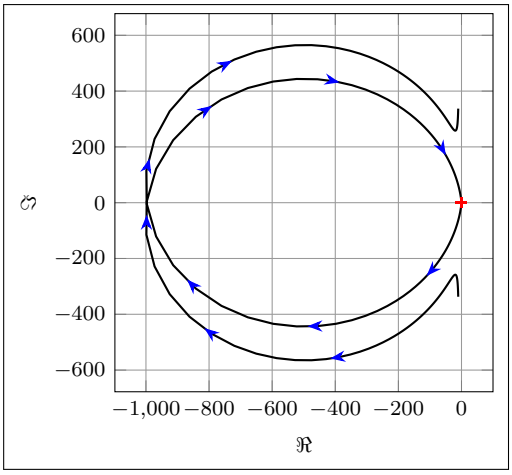

```
\begin{NicholsChart}[%
  ytick distance=20,
  xtick distance=30
]
{0.001}
{100}
  \addNicholsZPKChart [red,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10%
  }
  \addNicholsZPKChart [blue,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},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



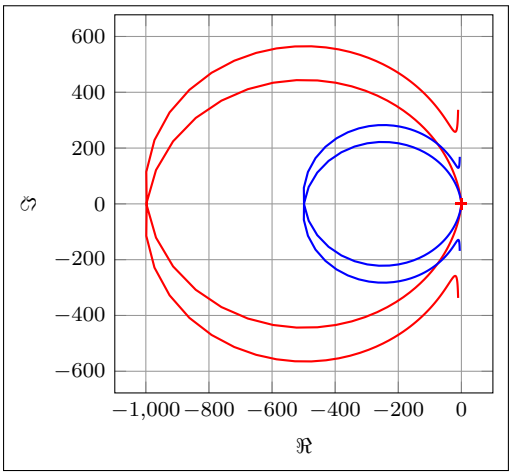
Nyquist plot in TF format with arrows



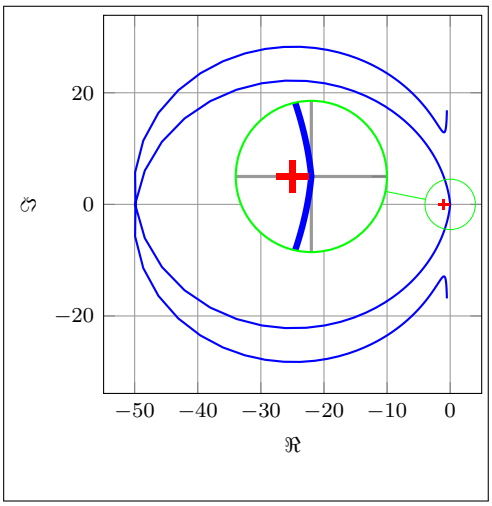

```
\NyquistTF[%
  plot/{%
    samples=1000,
    postaction=decorate,
    decoration={%
      markings,
      mark=between positions 0.1 and 0.9 step 5em with {%
        \arrow{Stealth [length=2mm, blue]}
      }
    }
  }%
]
{%
  num/{10,2,2.6,0},
  den/{1,1,100.25}%
}
{-30}
{30}
```

Multiple Nyquist plots with customization

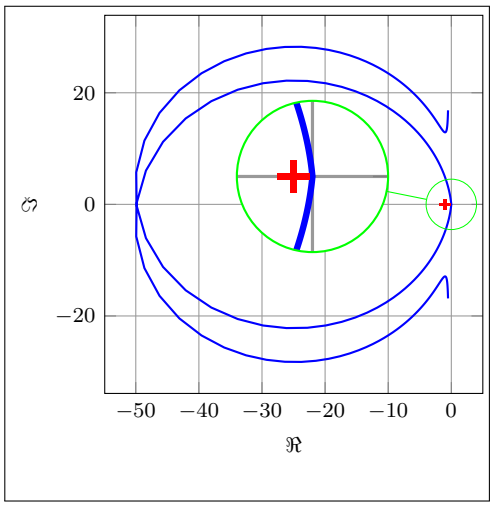
```
\begin{NyquistPlot}{-30}{30}
  \addNyquistZPKPlot [red,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/10%
  }
  \addNyquistZPKPlot [blue,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/5%
  }
\end{NyquistPlot}
```



Nyquist plots with additional commands, using two different macros

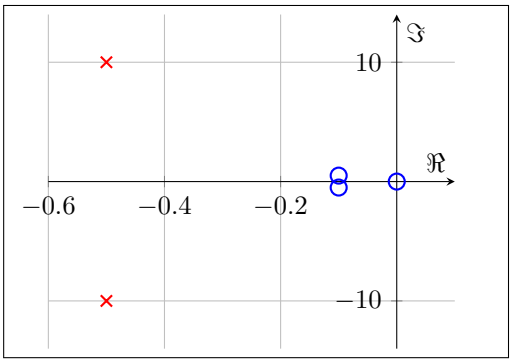



```
\begin{NyquistPlot}[%
  tikz/{
    spy using outlines={%
      circle,
      magification=3,
      connect spies,
      size=2cm
    }
  }%
]
{-30}{30}
  \addNyquistZPKPlot [blue,samples=1000] {%
    z/{0,{-0.1,-0.5},{-0.1,0.5}},
    p/{{-0.5,-10},{-0.5,10}},
    k/0.5%
  }
  \coordinate (spyon) 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,
      magification=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}
```

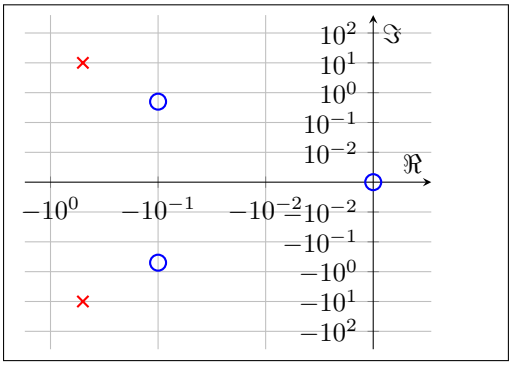
Pole-zero map




```
\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%
}
```



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.

C

D

E

G

I

L

M

Change History