python-control-plotly

vincent choqueuse

Aug 17, 2021

CONTENTS:

CHAPTER

ONE

GETTING STARTED

The *python-control-plotly* library provides several classes and functions to analyse the behavior of continuous and discrete time systems. As its name suggests, this library is based on the *python-control* and *plotly* libraries.

This librairy contains low-level and high-level API.

- Low level API provides several classes that allows in-depth customisation of plots,
- High level API provides several function that mimic the behavior of some Matlab plotting function. This tools relies on the low-level api.

Most of users will only use the high-level API since it allows to plot graph with a mininum of code.

1.1 Installation

```
pip install control-plotly
```

1.2 How to use it

This library requires the *python-control* library. This library can be importer as follows

```
import control as ctl
```

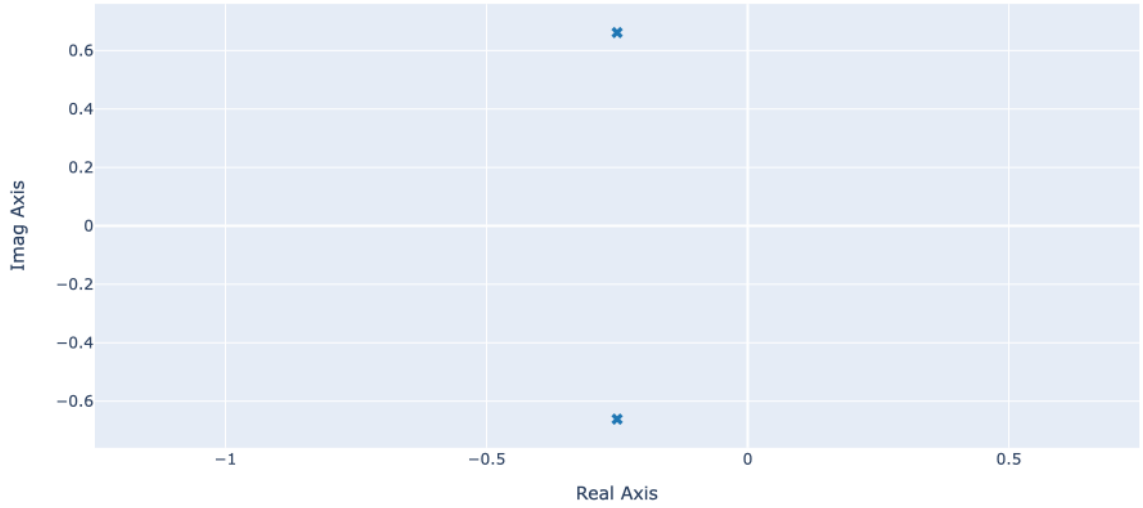
There are basically two ways to use this library.

- Standalone Python code
- Jupyter Notebook

1.2.1 Standalone Python code

For standalone python code, figures are plotted in your default browser. For this use case, you need to explicitely call the *show()* method to show your plot.

```
from control_plotly import pzmap

sys = ctl.tf([1],[2,1,1])
fig = pzmap(sys)
fig.show()
```



1.2.2 Jupyter Notebook

In jupyter notebook, figures are automatically plotted if the plotting function is located in the last line of a code cell.

```
import control as ctl
from control_plotly import pzmap

sys = ctl.tf([1],[2,1,1])
pzmap(sys)
```

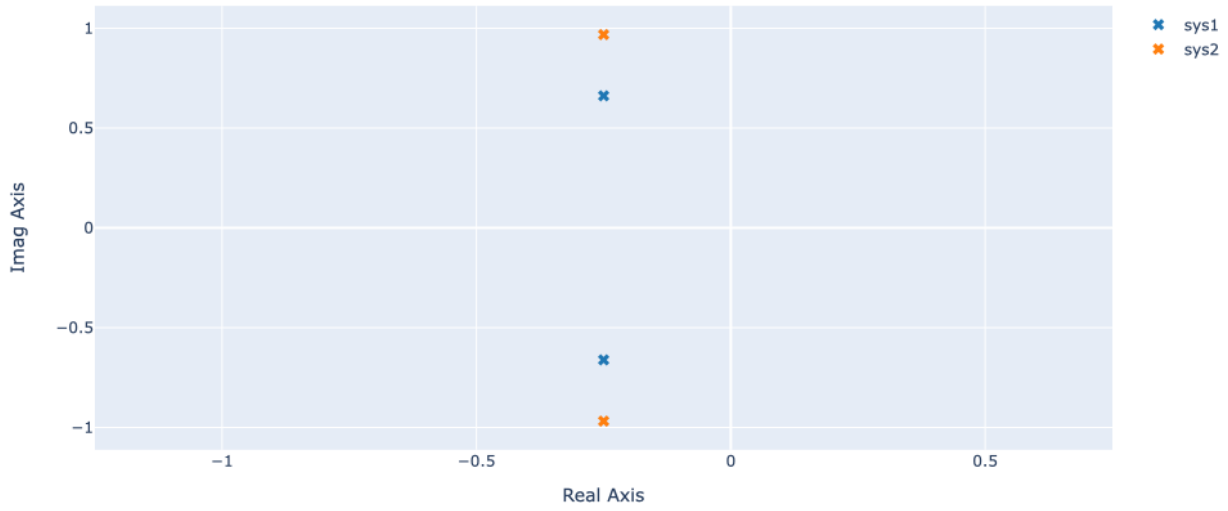
CHAPTER

TWO

GALLERY

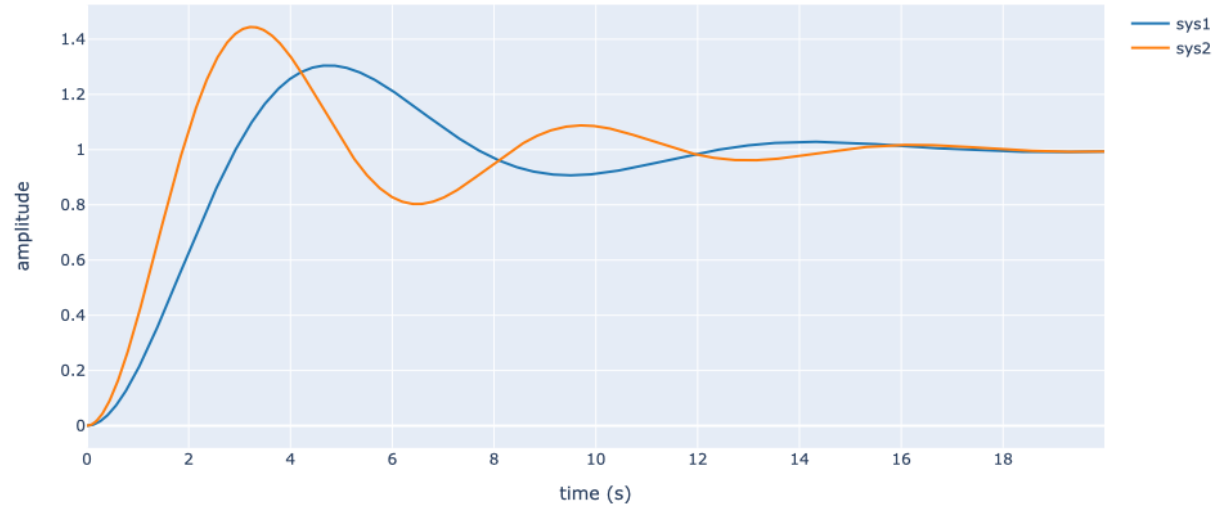
2.1 Poles and Zeros

The function *control_plotly.pzmap()* creates a pole-zero plot of the continuous or discrete-time system *sys*.



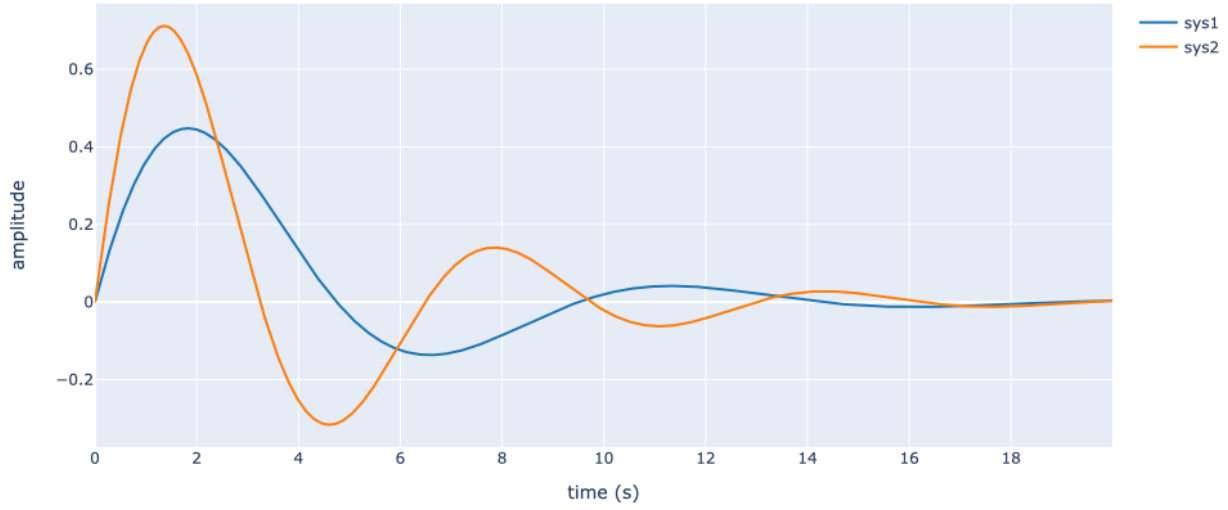
2.2 Step Reponse

The function *control_plotly.step()* create a step response plot for the continuous or discrete-time system *sys*.



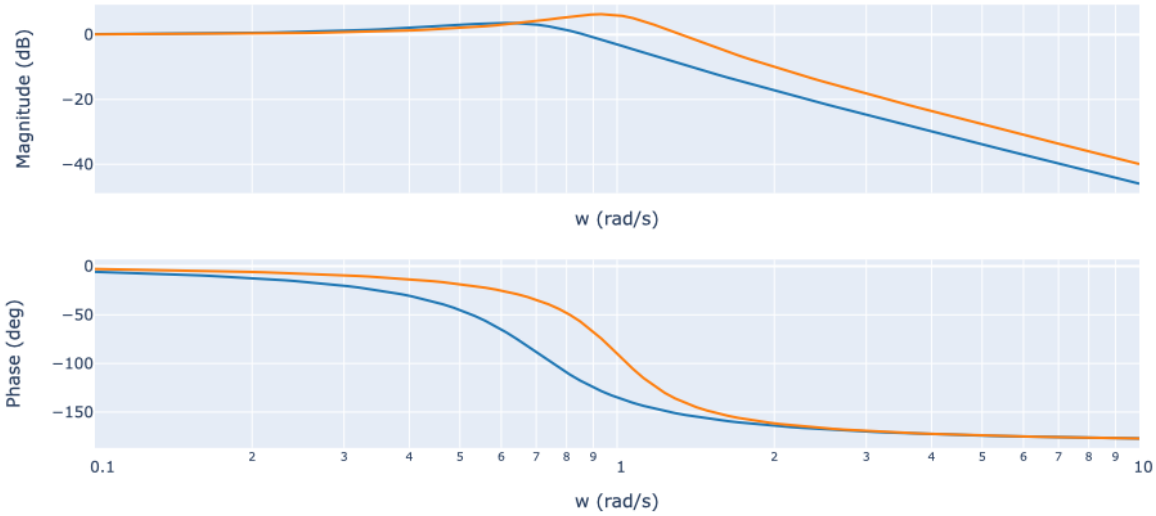
2.3 Impulse Reponse

The function *control_plotly.impulse()* create a step response plot for the continuous or discrete-time system *sys*.



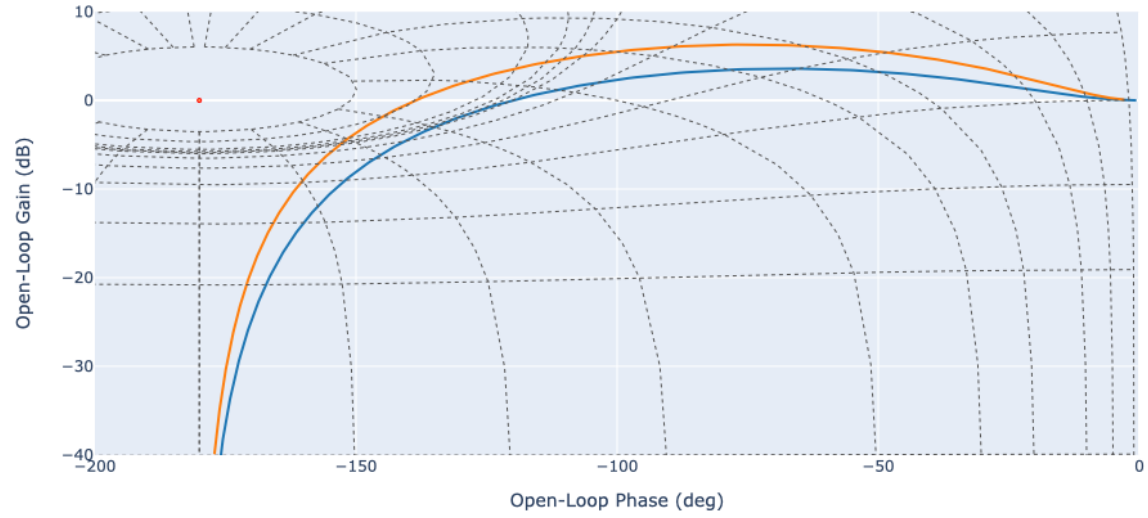
2.4 Bode Plot

The function *control_plotly.bode()* create a bode plot for the continuous or discrete-time system *sys*.



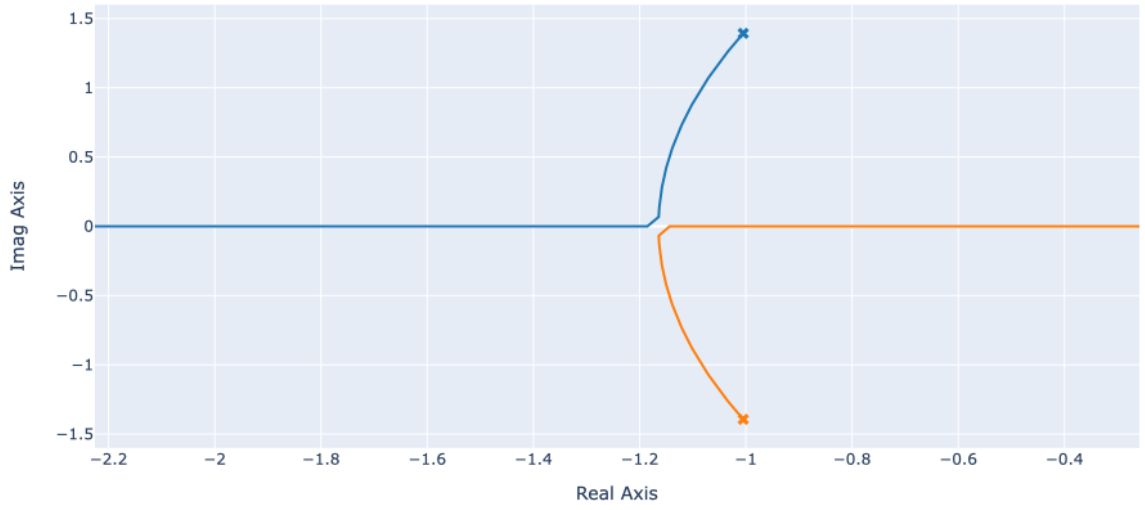
2.5 Nichols Plot

The function *control_plotly.nichols()* create a nichols plot for the continuous or discrete-time system *sys*.



2.6 Root Locus Plot

The function *control_plotly.rlocus()* create a root locus plot for the continuous or discrete-time system *sys*.



CHAPTER

THREE

API DOCUMENTATION

3.1 High-Level API

plots	
pzmap(sys_list[, x_lim, y_lim, x_title, y_title])	Returns a pole-zero plot of the continuous or discrete-time systems *sys_list*.
step(sys_list[, t, x_lim, y_lim, x_title, ...])	Returns the step response plot of the continuous or discrete-time systems *sys_list*.
impulse(sys_list[, t, x_lim, y_lim, ...])	Returns the impulse response of the continuous or discrete-time systems *sys_list*.
bode(sys_list[, w, x_lim, y_lim, dB, Hz, ...])	Returns the impulse response of the continuous or discrete-time systems *sys_list*.
nichols(sys_list[, w, x_lim, y_lim, cm, cp, ...])	Returns the nichols chart of the continuous or discrete-time systems *sys_list*.
rlocus(sys[, k, x_lim, y_lim, show_grid, wn, m])	Returns the root locus chart of the continuous or discrete-time systems *sys_list*.

3.1.1 control_plotly.pzmap

control_plotly.**pzmap**(*sys_list*, *x_lim=None*, *y_lim=None*, *x_title=None*, *y_title=None*)

Returns a pole-zero plot of the continuous or discrete-time systems *sys_list*.

Parameters

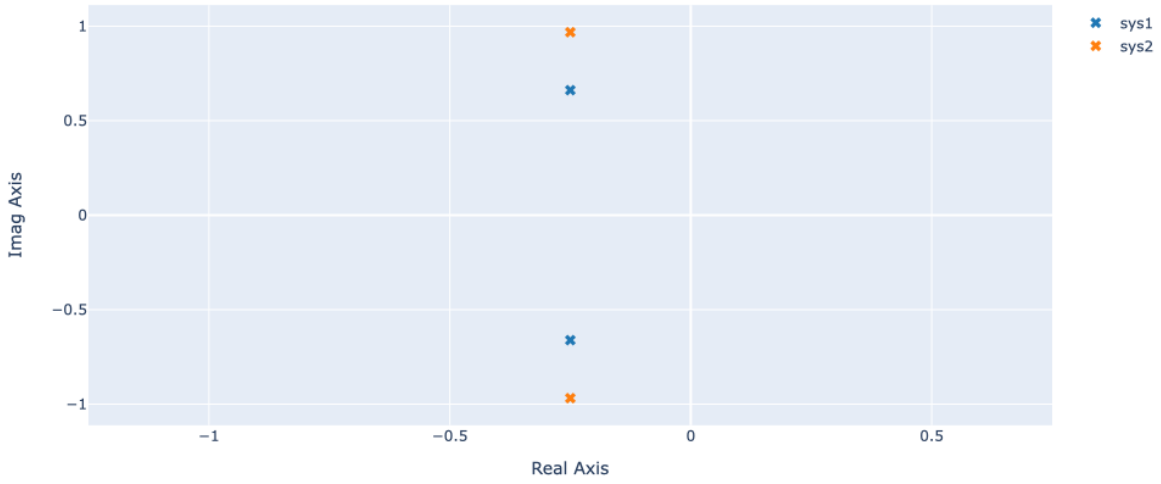
- **sys_list** (*system or list of systems*) – A single system or a list of systems to analyse
- **x_lim** (*list (optional)*) – A list of two element that defines the min and max value for the x axis
- **y_lim** (*list (optional)*) – A list of two element that defines the min and max value for the y axis
- **x_title** (*str (optional)*) – The x axis name
- **y_title** (*str (optional)*) – The y axis name

Returns **fig** – A plotly figure

Return type plotly figure

Example

```
import control as ctl
from control_plotly import pzmap

sys1 = ctl.tf([1],[2,1,1])
sys2 = ctl.tf([1],[1,0.5,1])
pzmap([sys1,sys2])
```



3.1.2 control_plotly.step

control_plotly.**step**(*sys_list*, *t=None*, *x_lim=None*, *y_lim=None*, *x_title=None*, *y_title=None*)

Returns the step response plot of the continuous or discrete-time systems *sys_list*.

Parameters

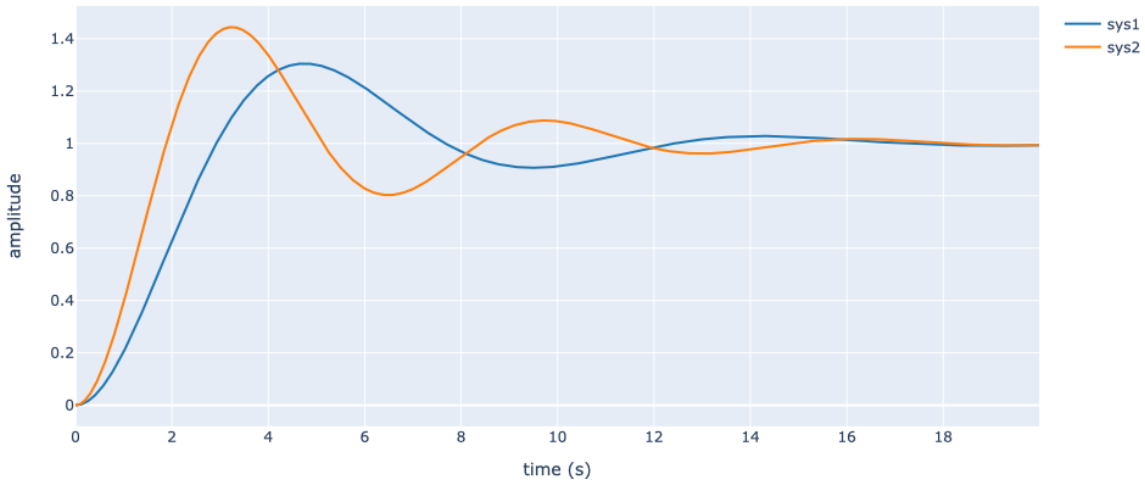
- **sys_list** (*system or list of systems*) – A single system or a list of systems to analyse
- **t** (*numpy vector (optional)*) – The base time vector
- **x_lim** (*list (optional)*) – A list of two element that defines the min and max value for the x axis
- **y_lim** (*list (optional)*) – A list of two element that defines the min and max value for the y axis
- **x_title** (*str (optional)*) – The x axis name
- **y_title** (*str (optional)*) – The y axis name

Returns **fig** – A plotly figure

Return type plotly figure

Example

```
import control as ctl
from control_plotly import step

sys1 = ctl.tf([1],[2,1,1])
sys2 = ctl.tf([1],[1,0.5,1])
t = np.arange(0,20,0.01)

step([sys1,sys2],t=t)
```



3.1.3 control_plotly.impulse

control_plotly.**impulse**(*sys_list*, *t=None*, *x_lim=None*, *y_lim=None*, *x_title=None*, *y_title=None*)

Returns the impulse response of the continuous or discrete-time systems *sys_list*.

Parameters

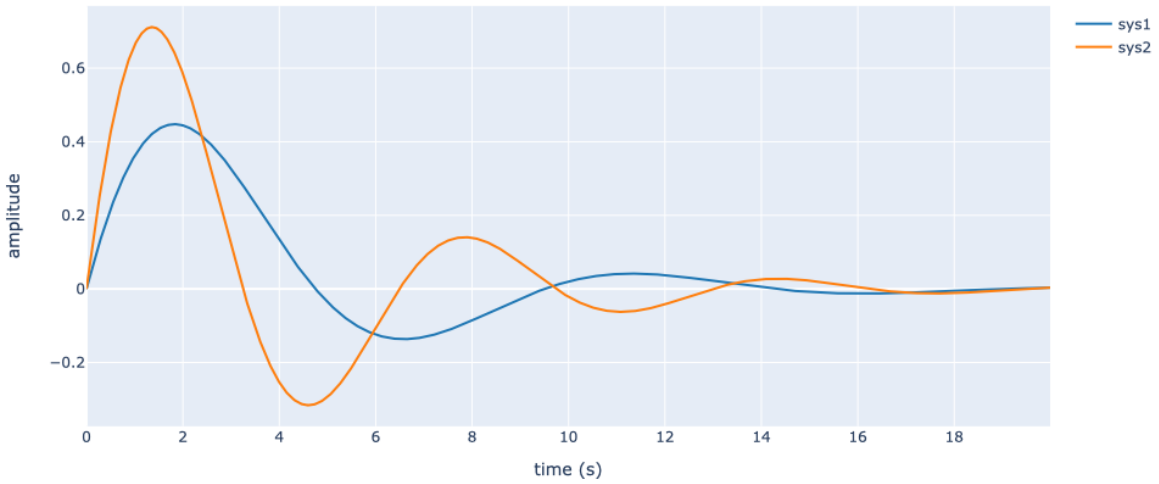
- **sys_list** (*system or list of systems*) – A single system or a list of systems to analyse
- **t** (*numpy vector (optional)*) – The base time vector
- **x_lim** (*list (optional)*) – A list of two element that defines the min and max value for the x axis
- **y_lim** (*list (optional)*) – A list of two element that defines the min and max value for the y axis
- **x_title** (*str (optional)*) – The x axis name
- **y_title** (*str (optional)*) – The y axis name

Returns **fig** – A plotly figure

Return type plotly figure

Example

```
import control as ctl
from control_plotly import impulse

sys1 = ctl.tf([1],[2,1,1])
sys2 = ctl.tf([1],[1,0.5,1])
t = np.arange(0,20,0.01)

impulse([sys1,sys2],t=t)
```



3.1.4 control_plotly.bode

control_plotly.**bode**(*sys_list*, *w=None*, *x_lim=None*, *y_lim=None*, *dB=True*, *Hz=False*, *deg=True*, *log_x=True*)

Returns the impulse response of the continuous or discrete-time systems *sys_list*.

Parameters

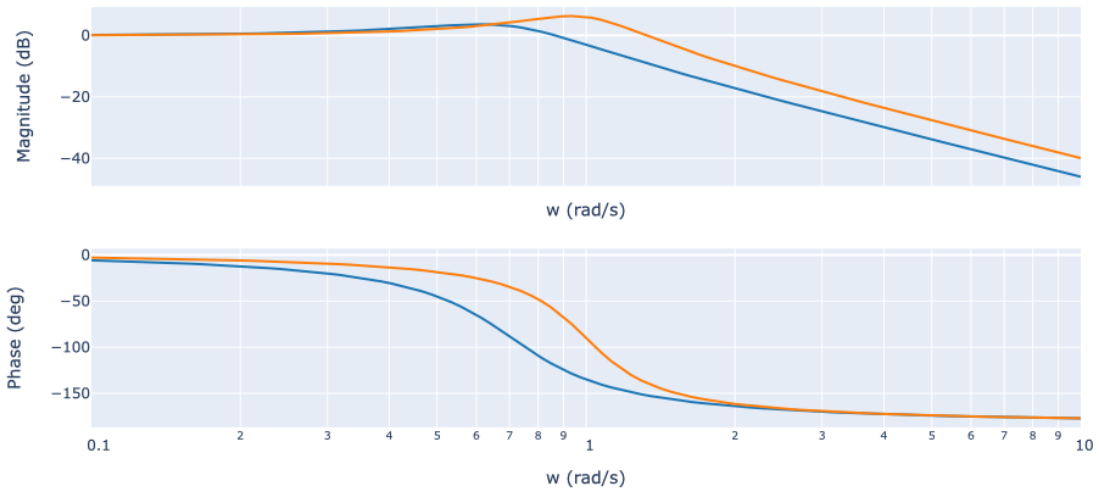
- **sys_list** (*system or list of systems*) – A single system or a list of systems to analyse
- **w** (*numpy vector (optional)*) – The base angular frequency vector (in rad/s)
- **x_lim** (*list (optional)*) – A list of two element that defines the min and max value for the x axis
- **y_lim** (*list (optional)*) – A list of two element that defines the min and max value for the y axis
- **dB** (*boolean (optional)*) – Use a logarithmic scale for the magnitude plot
- **Hz** (*boolean (optional)*) – Use frequency in Hz for the x axis
- **deg** (*boolean (optional)*) – Use angle in degree for the phase plot.

Returns **fig** – A plotly figure

Return type plotly figure

Example

```
import control as ctl
from control_plotly import bode

sys1 = ctl.tf([1],[2,1,1])
sys2 = ctl.tf([1],[1,0.5,1])
w = np.logspace(-1,1,100)

bode([sys1,sys2],w=w)
```



3.1.5 control_plotly.nichols

control_plotly.**nichols**(*sys_list*, *w=None*, *x_lim=None*, *y_lim=None*, *cm=array([6.0, 3.0, 1.0, 0.5, 0.25, 0.0, - 1.0, - 3.0, - 6.0, - 12.0, - 20.0, - 40.0])*, *cp=array([1, 5, 10, 20, 30, 50, 90, 120, 150, 180])*, *show_grid=True*, *show_mag=True*, *show_phase=True*)

Returns the nichols chart of the continuous or discrete-time systems *sys_list*.

Parameters

- **sys_list** (*system or list of systems*) – A single system or a list of systems to analyse
- **w** (*numpy vector (optional)*) – The base angular frequency vector (in rad/s)
- **x_lim** (*list (optional)*) – A list of two element that defines the min and max value for the x axis
- **y_lim** (*list (optional)*) – A list of two element that defines the min and max value for the y axis

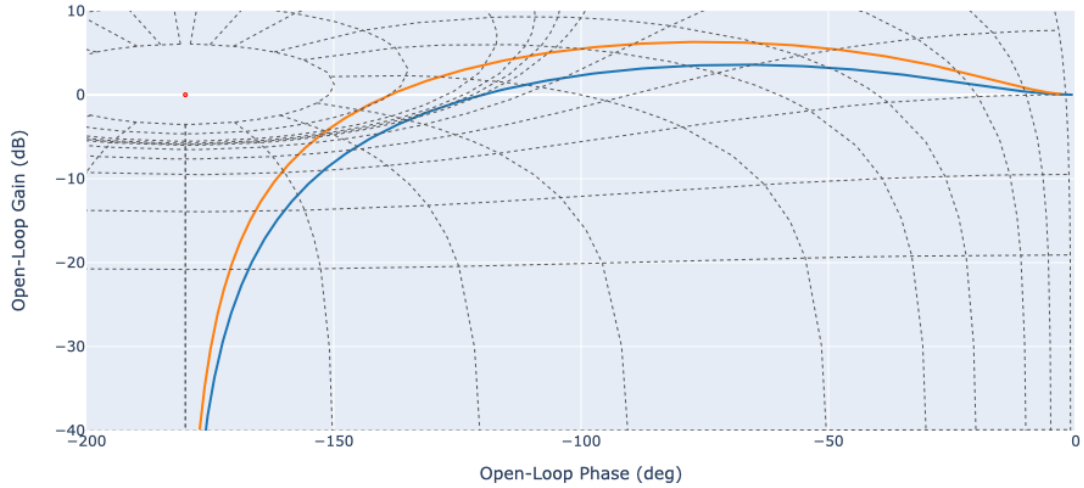
- **cm** (*numpy vector (optional)*) – A numpy vector containing the list of contour gain (in dB)
- **cp** (*numpy vector (optional)*) – A numpy vector containing the list of contour phase (in deg)
- **show_grid** (*boolean (optional)*) – Add the nichols grid
- **show_mag** (*boolean (optional)*) – Show the nichols magnitude grid
- **show_phase** (*boolean (optional)*) – Show the nichols phase grid

Returns **fig** – A plotly figure

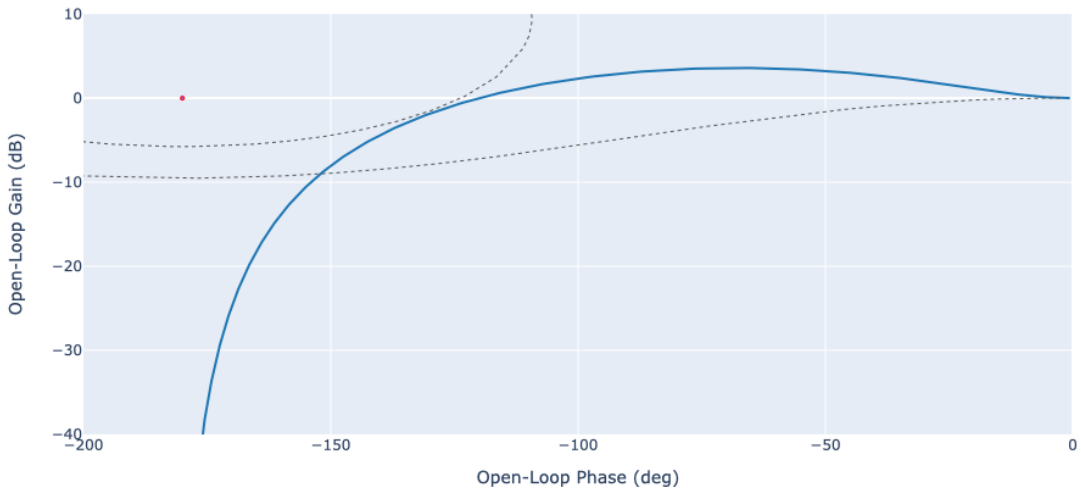
Return type plotly figure

Example

```
import control as ctl
from control_plotly import nichols

sys1 = ctl.tf([1],[2,1,1])
sys2 = ctl.tf([1],[1,0.5,1])

nichols([sys1,sys2])
```




```
import control as ctl
import numpy as np
from control_plotly import nichols

sys = ctl.tf([1],[2,1,1])

nichols(sys,show_phase=False,cm=np.array([0.5,-6]),x_lim=[-200,0],y_lim=[-40,10])
```



3.1.6 control_plotly.rlocus

control_plotly.**rlocus**(*sys*, *k=None*, *x_lim=None*, *y_lim=None*, *show_grid=False*, *wn=array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])*, *m=array([0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])*)

Returns the root locus chart of the continuous or discrete-time systems *sys_list*.

Parameters

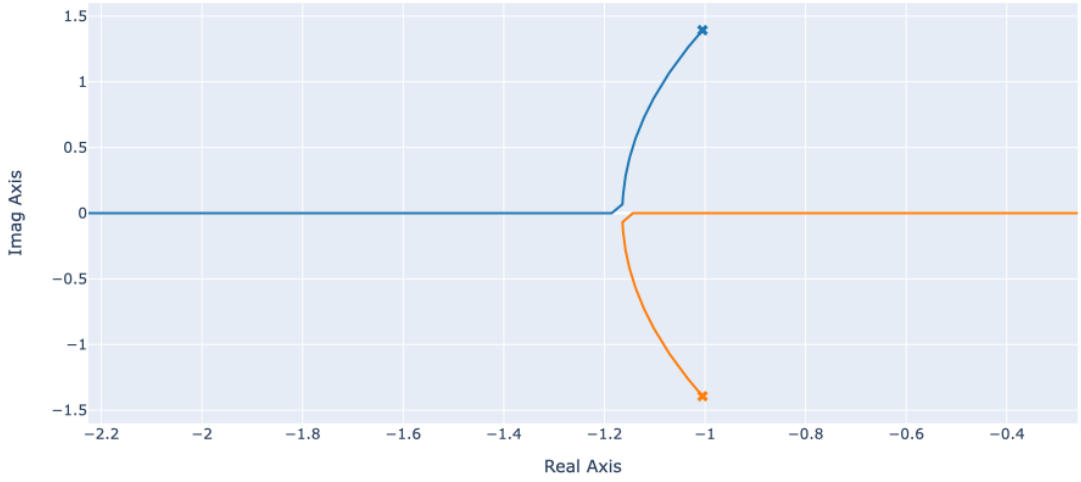
- **sys** (*system*) – A single system
- **k** (*numpy vector (optional)*) – The vector of feedback gains
- **x_lim** (*list (optional)*) – A list of two element that defines the min and max value for the x axis
- **y_lim** (*list (optional)*) – A list of two element that defines the min and max value for the y axis
- **show_grid** (*boolean (optional)*) – Add the discrete to continuous pole grid

Returns **fig** – A plotly figure

Return type plotly figure

Example

```
import control as ctl
from control_plotly import rlocus

sys = ctl.tf([2,5,1],[1,2,3])
rlocus(sys)
```



3.2 Low-Level API

figures
Time_Figure([log_x, log_y])
Step_Figure([log_x, log_y])
Impulse_Figure([log_x, log_y])
Bode_Figure([dB, Hz, deg, log_x])
Nichols_Figure(show_mag, show_phase)

3.2.1 control_plotly.Time_Figure

class control_plotly.**Time_Figure**(*log_x=False*, *log_y=False*)

__init__(*log_x=False*, *log_y=False*)

Methods

__init__([log_x, log_y])	
add_plot(tf[, type, T, label])	Add a new plot for the system *tf*.
clear()	
clear_extra()	
clear_grid()	
clear_plot()	
get_grid_line()	Returns the plotly grid line
get_line_shape(tf)	Returns the line shape for a particular transfer function (linear for continuous systems and hv for discrete systems)
get_next_color()	Returns the next curve color
get_response(tf[, type, T])	Return the response of the system *tf*.
get_x_title()	
get_x_type()	Returns the type for the xaxis (log or linear)
get_y_title()	
get_y_type()	Returns the type for the yaxis (log or linear)
json()	Returns a json representation of the plotly data
set_x_lim(range)	Specify the xlim range
set_x_title(title)	
set_y_lim(range)	Specify the ylim range
set_y_title(title)	
show([show_data, show_grid, show_extra])	Constructs the figure and return the plotly figure instance

Attributes

color_list	
data	
layout	Returns the plotly figure layout
x_scaleanchor	
x_scaleratio	
x_title	

continues on next page

Table 4 – continued from previous page

y_scaleanchor	
y_scaleratio	
y_title	

3.2.2 control_plotly.Step_Figure

class control_plotly.**Step_Figure**(*log_x=False*, *log_y=False*)

__init__(*log_x=False*, *log_y=False*)

Methods

__init__([log_x, log_y])	
add_plot(tf[, type, T, label])	Add a new plot for the system *tf*.
clear()	
clear_extra()	
clear_grid()	
clear_plot()	
get_grid_line()	Returns the plotly grid line
get_line_shape(tf)	Returns the line shape for a particular transfer function (linear for continuous systems and hv for discrete systems)
get_next_color()	Returns the next curve color
get_response(tf[, type, T])	Return the step response
get_x_title()	
get_x_type()	Returns the type for the xaxis (log or linear)
get_y_title()	
get_y_type()	Returns the type for the yaxis (log or linear)
json()	Returns a json representation of the plotly data
set_x_lim(range)	Specify the xlim range
set_x_title(title)	
set_y_lim(range)	Specify the ylim range
set_y_title(title)	
show([show_data, show_grid, show_extra])	Constructs the figure and return the plotly figure instance

Attributes

color_list	
data	
layout	Returns the plotly figure layout
x_scaleanchor	
x_scaleratio	
x_title	
y_scaleanchor	
y_scaleratio	
y_title	

3.2.3 control_plotly.Impulse_Figure

class control_plotly.**Impulse_Figure**(*log_x=False*, *log_y=False*)

__init__(*log_x=False*, *log_y=False*)

Methods

__init__([log_x, log_y])	
add_plot(tf[, type, T, label])	Add a new plot for the system *tf*.
clear()	
clear_extra()	
clear_grid()	
clear_plot()	
get_grid_line()	Returns the plotly grid line
get_line_shape(tf)	Returns the line shape for a particular transfer function (linear for continuous systems and hv for discrete systems)
get_next_color()	Returns the next curve color
get_response(tf[, type, T])	Return the impulse response
get_x_title()	
get_x_type()	Returns the type for the xaxis (log or linear)

continues on next page

Table 7 – continued from previous page

get_y_title()	
get_y_type()	Returns the type for the yaxis (log or linear)
json()	Returns a json representation of the plotly data
set_x_lim(range)	Specify the xlim range
set_x_title(title)	
set_y_lim(range)	Specify the ylim range
set_y_title(title)	
show([show_data, show_grid, show_extra])	Constructs the figure and return the plotly figure instance

Attributes

color_list	
data	
layout	Returns the plotly figure layout
x_scaleanchor	
x_scaleratio	
x_title	
y_scaleanchor	
y_scaleratio	
y_title	

3.2.4 control_plotly.Bode_Figure

class control_plotly.**Bode_Figure**(*dB=False*, *Hz=False*, *deg=True*, *log_x=True*)

__init__(*dB=False*, *Hz=False*, *deg=True*, *log_x=True*)

Methods

`__init__`([dB, Hz, deg, log_x])	
`add_plot`(tf[, w, label])	
`clear`()	
`clear_extra`()	
`clear_grid`()	
`clear_plot`()	
`get_grid_line`()	Returns the plotly grid line
`get_line_shape`(tf)	Returns the line shape for a particular transfer function (linear for continuous systems and hv for discrete systems)
`get_next_color`()	Returns the next curve color
`get_x_title`()	
`get_x_type`()	Returns the type for the xaxis (log or linear)
`get_y1_title`()	
`get_y2_title`()	
`get_y_title`()	
`get_y_type`()	Returns the type for the yaxis (log or linear)
`json`()	Returns a json representation of the plotly data
`set_x_lim`(range)	Specify the xlim range
`set_x_title`(title)	
`set_y_lim`(range)	Specify the ylim range
`set_y_title`(title)	
`show`()	Constructs the figure and return the plotly figure instance

Attributes

`color_list`	
`data`	
`layout`	Returns the plotly figure layout
`x_scaleanchor`	
`x_scaleratio`	

continues on next page

Table 10 – continued from previous page

x_title	
y_scaleanchor	
y_scaleratio	
y_title	

3.2.5 control_plotly.Nichols_Figure

class control_plotly.**Nichols_Figure**(*show_mag*, *show_phase*)

__init__(*show_mag*, *show_phase*)

Methods

__init__(show_mag, show_phase)	
add_grid([cm, cp])	
add_plot(tf[, w, label])	
clear()	
clear_extra()	
clear_grid()	
clear_plot()	
get_grid_line()	Returns the plotly grid line
get_line_shape(tf)	Returns the line shape for a particular transfer function (linear for continuous systems and hv for discrete systems)
get_next_color()	Returns the next curve color
get_x_title()	
get_x_type()	Returns the type for the xaxis (log or linear)
get_y_title()	
get_y_type()	Returns the type for the yaxis (log or linear)
json()	Returns a json representation of the plotly data
set_x_lim(range)	Specify the xlim range
set_x_title(title)	
set_y_lim(range)	Specify the ylim range

continues on next page

Table 11 – continued from previous page

set_y_title(title)	
show([show_data, show_grid, show_extra])	Constructs the figure and return the plotly figure instance
update_min_max(mag, phase)	

Attributes

color_list	
data	
layout	Returns the plotly figure layout
x_scaleanchor	
x_scaleratio	
x_title	
y_scaleanchor	
y_scaleratio	
y_title	

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

B

I

N

P

R

S

T