trifilar-mass-prop Documentation

Release 1.0

Timo Horstschäfer

June 29, 2014

CONTENTS

Contents: All functions related to the center of gravity.

`inc.cog.`**`GetCoG2D`**(*G*, *free_arm*, *m*)

Calculate 2D center of gravity from a single measurement on two arms.

Parameters **G** : ndarray

The 2 g values for the measured arms.

free_arm : int

Number indicating the free arm, may be one of 1, 2 and 3.

m : float

The mass of the system.

Returns **ndarray** :

The 2D CoG of the system.

`inc.cog.`**`GetCoG2DSeries`**(*name*, *free_arm*, *S*, *M*, *Mt*, *R*)

Returns the series of 2D CoGs for a given measurement series.

Parameters **name** : str

Name for the measurement, may contain LaTeX.

free_arm : int

Number indicating the free arm, may be one of 1, 2 and 3.

M : float

System mass

Mt : float

Test mass

D : array_like

A 3D CoG measurement series as read from the JSON file, see examples.

R : ndarray

The measurements positions in m.

Returns **R** : ndarray

Array containing x, y coordinates of the 2D CoG. Each entry contains n values for the complete series.

Examples

```
>>> import json
>>> S = json.loads('''
{
        "name": "R_x",
        "free arm": 2,
        "series": [
                {
                        "name": "g_{x,1}",
                        "data": [ 4.95, 6.03, 7.08, 8.14, 9.26 ],
                        "platform": [ 2.11, 3.09, 4.13, 5.20, 6.27 ]
                },
```

```
                {
                        "name": "g_{x,3}",
                        "data": [ 7.35, 8.46, 9.48, 10.54, 11.60 ],
                        "platform": [ 6.49, 7.54, 8.61, 9.67, 10.70 ]
                }
        ]
}
'''')
>>> pprint(S)
[{u'data': [4.95, 6.03, 7.08, 8.14, 9.26],
  u'name': u'g_{x,1}',
  u'platform': [2.11, 3.09, 4.13, 5.2, 6.27]},
 {u'data': [7.35, 8.46, 9.48, 10.54, 11.6],
  u'name': u'g_{x,3}',
  u'platform': [6.49, 7.54, 8.61, 9.67, 10.7]}]
```

inc.cog.**GetCoG3D**(*axes*, *G*)

Returns the 3D CoG for 2 different 2D CoG measurements in the given axis configuration.

Parameters **axes** : str

String describing the two axes, in which the measurements were taken. Can be one of 'xy', 'xz' and 'yz'.

G : ndarray

A 2x2 array giving two 2D CoG coordinates. G[0] is the first CoG, G[1] the second.

Returns **R** : ndarray

The 3D center of gravity

inc.cog.**GetCoG3DSeries**(*axes*, *D*, *M*, *Mt*, *R*)

Return the full 3D CoG series for a given measurements series.

Parameters **axes** : str

String describing the two axes, in which the measurements were taken. Can be one of 'xy', 'xz' and 'yz'.

D : array_like

A 3D CoG measurement series as read from the JSON file, see examples.

Returns **R** : ndarray

Array containing x, y and z coordinate of the 3D CoG. Each entry contains n values for the complete series.

Examples

```
>>> import json
>>> D = json.loads('''
{
        "cog": {
                "axes": "xz",
                "data": [
                        {
                                "name": "R_x",
                                "free arm": 2,
                                "series": [
```

```
                                        {
                                                "name": "g_{x,1}",
                                                "data": [ 4.95, 6.03, 7.08, 8.14, 9.26 ],
                                                "platform": [ 2.11, 3.09, 4.13, 5.20, 6.27 ]
                                        },
                                        {
                                                "name": "g_{x,3}",
                                                "data": [ 7.35, 8.46, 9.48, 10.54, 11.60 ],
                                                "platform": [ 6.49, 7.54, 8.61, 9.67, 10.70 ]
                                        }
                                ]
                        },
                        {
                                "name": "R_z",
                                "free arm": 1,
                                "series": [
                                        {
                                                "name": "g_{z,2}",
                                                "data": [ 2.92, 3.99, 5.05, 6.14, 7.18 ],
                                                "platform": [ 2.19, 3.25, 4.29, 5.38, 6.52 ]
                                        },
                                        {
                                                "name": "g_{z,3}",
                                                "data": [ 5.50, 6.61, 7.74, 8.84, 10.07 ],
                                                "platform": [ 6.45, 7.53, 8.59, 9.67, 10.72 ]
                                        }
                                ]
                        }
                ]
        }
    }
    '''')
    >>> pprint(D)
    [{u'free arm': 2,
      u'name': u'R_x',
      u'series': [{u'data': [4.95, 6.03, 7.08, 8.14, 9.26],
                             u'name': u'g_{x,1}',
                             u'platform': [2.11, 3.09, 4.13, 5.2, 6.27]},
                            {u'data': [7.35, 8.46, 9.48, 10.54, 11.6],
                             u'name': u'g_{x,3}',
                             u'platform': [6.49, 7.54, 8.61, 9.67, 10.7]}]},
     {u'free arm': 1,
      u'name': u'R_z',
      u'series': [{u'data': [2.92, 3.99, 5.05, 6.14, 7.18],
                             u'name': u'g_{z,2}',
                             u'platform': [2.19, 3.25, 4.29, 5.38, 6.52]},
                            {u'data': [5.5, 6.61, 7.74, 8.84, 10.07],
                             u'name': u'g_{z,3}',
                             u'platform': [6.45, 7.53, 8.59, 9.67, 10.72]}]}]
    '''
```

inc.cog.**GetG**(*data*, *platform*, *Mt*, *R*, *name='g'*)

Return the g value of a measurements.

Parameters **data** : array_like

The measured g values in gram.

platform : array_like

The measured bare platform g values in gram.

Mt : float

Test mass

R : ndarray

The measurements positions in m.

name : str

Name for the measurement, may contain LaTeX.

Returns **g** : ndarray

The bare unit g values.

inc.cog.**PlotGTest**(*g*, *Mt*, *R*, *name*)

Creates a plot for a g value measurements series in the 'out' folder.

Parameters **g** : array_like

The measured g values in gram.

Mt : float

Test mass

R : ndarray

The measurements positions in m.

name : str

Name for the measurement, may contain LaTeX.

All functions related to the moment of inertia and inertia tensor.

inc.inertia.**GetDistanceFromAxis**(*p*, *axis*)

Returns the distance of an arbitrary point from the given axis.

Parameters **p** : ndarray

The point in 3D space.

axis : str

Identifier for the axis, must be one of 'xx', 'yy', 'zz', 'xy', 'xz', 'yz'.

Returns: :

d : float

The distance from the point to the axis.

inc.inertia.**GetISeries**(*workdir*, *name*, *M*, *Mt*, *R*, *p0*, *Ip*)

Same as GetInertiaSeries(), but taking into account the platform inertia. Thus, gives the real unit moment of inertia.

Parameters **workdir** : str

The name of the working directoy

name : str

Name of the measurement

M : float

Mass of the system, e.g. unit mass + platform mass in kg.

R : ndarray

The measurements positions in m.

p0 : array_like

Initial parameters for the least-square fit

Ip : ndarray

The platform inertia series, measured in the same positions and with the same test mass.

Returns **Iu** : ndarray

The bare unit inertia.

inc.inertia.**GetInertiaSeries**(*workdir*, *name*, *M*, *Mt*, *R*, *p0*)

Get the list of moments of inertia for a set of movies or .csv files given in the working directory. The movies/csv files have to be named after the position of the test mass in meters with a presicion of to, e.g. '0.3.csv' or '0.7.MP4'.

Parameters **workdir** : str

The name of the working directoy

name : str

Name for the measurement, appears in the title, may contain LaTeX.

M : float

Mass of the system, e.g. unit mass + platform mass in kg.

R : ndarray

The measurements positions in m.

p0 : array_like

Initial parameters for the least-square fit

Returns **I** : ndarray

The inertia series.

inc.inertia.**GetInertiaTensorSeries**(*workdir*, *Mu*, *Mp*, *Mt*, *R*, *p0*, *CoG3D*)

Computes the series of the inertia tensor a given measurement series and a 3D center of gravity, taking into account the parallel axis theorem.

Parameters **workdir** : str

The name of the working directoy

Mu : float

Unit mass

Mu : float

Platform mass

Mu : float

Test mass

R : ndarray

The measurements positions in m.

p0 : array_like

Initial parameters for the least-square fit

CoG3D : ndarray

The 3 coordinates of the center of gravity.

Returns **I** : dict

The inertia tensor as a dictionary containing entries 'xx', 'yy', etc. Each entry contains the series of results for the given component as an ndarray.

inc.inertia.**PlotITest**(*I*, *Mt*, *R*, *name*)

Create a plot for a inertia series measurements. The plot is written as a pdf file in the 'out' folder.

Parameters **I** : ndarray

The inertia series

Mt : float

The test mass

R : ndarray

The measurements positions in m

name : str

Name for the measurement, appears in the title, may contain LaTeX.

Helper function to analyze csv files from videos.

inc.signal.**GetPeriod**(*name, p0=[100.0, 0.1, 1.0, 0.1, 10.0, 1.0, 1.0, 0.1, 100.0]*)

Returns the period of oscillation from a given filename, corresponding to a csv or movie file. If no csv file exist, the tracker program is called to analyze the video.

Parameters **name** : str

Filename WITHOUT extension.

p0 : array_like

Initial parameters for the least-square fit given in the order A1, tau1, T1, d1, A2, tau2, T2, d2, C

Returns **T** : float

Period of oscillation in seconds.

General tools

inc.tools.**Savefig**(*fig*, *name*)

Saves a figure in the 'out' directory, putting it in seperate a folder for each script.

Parameters **fig** : Figure

The figure object from matplotlib

name : str

Name of the figure, used for the output filename.

inc.tools.**StatPrint**(*name*, *D*, *unit=''*)

Give information about a measurement series.

Parameters **name** : str

Name of measurements to format output.

D : ndarray

Data array

unit : str

Unit of the data

inc.tools.**pol2xy**(*r*, *theta*)

Converts polar coordinates from cartesian.

Parameters **r** : float

Radius

theta : float

Angle

Returns **ndarray** :

x and y coordinates

inc.tools.**un2str**(*x*, *xe*, *precision=2*)

Pretty print nominal value and uncertainty

Parameters **x** : float

Nominal value

xe : float

Uncertainty

precision : int

Number of significant digits in uncertainty

Returns **str** :

Shortest string representation of *x* +- *xe* either as x.xx(ee)e+xx

or as xxx.xx(ee)

i

G

I

P

S

U