
pymad8 Documentation

Release 1.6.1

Royal Holloway

Jun 15, 2021

CONTENTS

1	Licence & Disclaimer	3
2	Authorship	5
3	Installation	7
3.1	Requirements	7
3.2	Installation	7
4	Data Loading	9
5	Converting Models	11
5.1	Mad8 output required	11
6	Plotting	13
7	Support	15
7.1	Feature Request	15
8	Module Contents	17
8.1	Module contents	17
8.2	pymad8.Input module	17
8.3	pymad8.Output module	18
8.4	pymad8.Mad8 module	21
8.5	pymad8.Plot module	24
8.6	pymad8.Sim module	24
8.7	pymad8.Visualisation module	25
9	Indices and tables	27
	Python Module Index	29
	Index	31

pymad8 is a Python package to aid in the preparation, running and validation of BDSIM models.

LICENCE & DISCLAIMER

pymad8 copyright (c) Royal Holloway, University of London, 2018. All rights reserved.

This software is provided “AS IS” and any express or limit warranties, including, but not limited to, implied warranties of merchantability, of satisfactory quality, and fitness for a particular purpose or use are disclaimed. In no event shall Royal Holloway, University of London be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this software, even if advised of the possibility of such damage.

AUTHORSHIP

The following people have contributed to pymad8:

- Stewart Boogert
- Andrey Abramov
- Laurie Nevay
- Will Parker
- William Shields
- Jochem Snuverink
- Stuart Walker

INSTALLATION

3.1 Requirements

- pymad8 is developed for Python 3 but should be Python 2 compatible.
- Requires : fortranformat

3.2 Installation

To install pymad8, simply run `make install` from the root pymad8 directory.:

```
cd /my/path/to/repositories/  
git clone http://bitbucket.org/jairhul/pymad8  
cd pymad8  
make install
```

Alternatively, run `make develop` from the same directory to ensure that any local changes are picked up.

DATA LOADING

Utilies to load pymad8 output data.

CONVERTING MODELS

pymad8 provides converters to allow BDSIM models to be prepared from optical descriptions in MAD8.

5.1 Mad8 output required

To make the TWISS and RMAT output:

```
use, LINE
twiss, beta0=LINE.B0, save, tape=twiss_LINE rtape=RMAT_LINE
```

Envelope:

```
use, LINE
envel, sigma0=LINE.SIGMA0, save, tape=ENVEL_LINE
```

Survey:

```
use, LINE
survey, tape=SURVEY_LINE
```


PLOTTING

SUPPORT

All support issues can be submitted to our [issue tracker](#)

7.1 Feature Request

Feature requests or proposals can be submitted to the issue tracker - select the issue type as proposal or enhancement..

Please have a look at the existing [list of proposals](#) before submitting a new one.

MODULE CONTENTS

This documentation is automatically generated by scanning all the source code. Parts may be incomplete.

8.1 Module contents

pymad8 - python tools for working with MAD8 output and input.

Dependencies:

package - minimum version required

numpy - 1.7.1

matplotlib - 1.3.0

Modules:

Input -

Output -

Plot -

Sim -

Track -

Visualisation -

Copyright Royal Holloway, University of London 2019.

8.2 pymad8.Input module

`pymad8.Input.decodeCollimator(input)`

`pymad8.Input.decodeDecapole(input)`

`pymad8.Input.decodeDrift(input)`

`pymad8.Input.decodeFileLine(input)`

decode line input is a string of a mad8 line

`pymad8.Input.decodeKicker(input)`

`pymad8.Input.decodeLcavity(input)`

`pymad8.Input.decodeLine(input)`

`pymad8.Input.decodeMultipole(input)`

`pymad8.Input.decodeNameAndType(input)`

```

pymad8.Input.decodeNamed(input)
pymad8.Input.decodeOctupole(input)
pymad8.Input.decodeQuadrupole(input)
pymad8.Input.decodeSbend(input)
pymad8.Input.decodeSextupole(input)
pymad8.Input.removeComments(input)
    remove comment lines
pymad8.Input.removeContinuationSymbols(input)
    remove continuation symbols from input input : list of file lines
pymad8.Input.splitKeyValue(t)
pymad8.Input.tidy(input)
    tidy input, remove EOL, remove empty lines input : list of file lines

```

8.3 pymad8.Output module

```

class pymad8.Output.Chrom
    Bases: pymad8.Output.General
    Chromaticity data structure data : numpy array of data keys : key to data
    getData(index)

class pymad8.Output.Common
    Bases: pymad8.Output.General
    containsEnergyVariation()
        Method to determine if the energy is constant in the lattice Required if there is 1) RfCavities
    getApertures(raw=True)
    getColumn(colName)
    getData(index)
    getRowByIndex(index)
    getRowByName(name)

```

```

keys = {'blmo': {'E': 11, 'l': 0, 'note': 10}, 'drif': {'E': 11, 'aper': 9,
'l': 0, 'note': 10}, 'ecol': {'E': 11, 'aper': 9, 'l': 0, 'note': 10,
'xsize': 4, 'ysize': 5}, 'elseparator': {'E': 11, 'aper': 9, 'efield': 5,
'l': 0, 'note': 10, 'tilt': 4}, 'hkic': {'E': 11, 'aper': 9, 'hkick': 4,
'l': 0, 'note': 10}, 'hmonitor': {'E': 11, 'aper': 9, 'l': 0, 'note': 10},
'imon': {'E': 11, 'l': 0, 'note': 10}, 'inst': {'E': 11, 'l': 0, 'note':
10}, 'kick': {'E': 11, 'aper': 9, 'hkick': 4, 'l': 0, 'note': 10, 'vkick':
5}, 'lcav': {'E': 11, 'aper': 9, 'freq': 5, 'l': 0, 'lag': 7, 'note': 10,
'volt': 6}, 'mark': {'E': 11, 'l': 0, 'note': 10}, 'matr': {'E': 11, 'aper':
9, 'l': 0}, 'moni': {'E': 11, 'aper': 9, 'l': 0, 'note': 10}, 'mult': {'E':
11, 'aper': 9, 'k0l': 1, 'k1l': 2, 'k2l': 3, 'k3l': 5, 'note': 10, 't0':
4, 't1': 6, 't2': 7, 't3': 8}, 'octu': {'E': 11, 'aper': 9, 'k3': 5, 'l':
0, 'note': 10, 'tilt': 4}, 'prof': {'E': 11, 'l': 0, 'note': 10}, 'quad':
{'E': 11, 'aper': 9, 'k1': 2, 'l': 0, 'note': 10, 'tilt': 4}, 'rben': {'E':
11, 'angle': 1, 'aper': 9, 'e1': 5, 'e2': 6, 'h1': 7, 'h2': 8, 'k1': 2,
'k2': 3, 'l': 0, 'note': 10, 'tilt': 4}, 'rcol': {'E': 11, 'aper': 9, 'l':
0, 'note': 10, 'xsize': 4, 'ysize': 5}, 'rfcavity': {'E': 11, 'aper': 9,
'freq': 5, 'l': 0, 'lag': 7, 'note': 10, 'volt': 6}, 'sben': {'E': 11,
'angle': 1, 'aper': 9, 'e1': 5, 'e2': 6, 'h1': 7, 'h2': 8, 'k1': 2, 'k2':
3, 'l': 0, 'note': 10, 'tilt': 4}, 'sext': {'E': 11, 'aper': 9, 'k2': 3,
'l': 0, 'note': 10, 'tilt': 4}, 'sole': {'E': 11, 'aper': 9, 'ks': 5, 'l':
0, 'note': 10}, 'srot': {'E': 11, 'angle': 5, 'aper': 9, 'l': 0, 'note':
10}, 'vkic': {'E': 11, 'aper': 9, 'l': 0, 'note': 10, 'vkick': 5},
'vmonitor': {'E': 11, 'aper': 9, 'l': 0, 'note': 10}, 'wire': {'E': 11, 'l':
0, 'note': 10}, 'yrot': {'E': 11, 'angle': 5, 'aper': 9, 'l': 0, 'note':
10}}

```

```
makeLocationList(elementNames=[])
```

```
class pymad8.Output.EchoValue(echoFileName)
    Bases: object
```

```
loadMarkedValues()
```

```
loadValues()
```

```
class pymad8.Output.Envelope
    Bases: pymad8.Output.General
```

Beam envelope data structure data : numpy array of data keys : key to data

```
getData(index)
```

```

keys = {'s11': 0, 's12': 1, 's13': 2, 's14': 3, 's15': 4, 's16': 5, 's21':
6, 's22': 7, 's23': 8, 's24': 9, 's25': 10, 's26': 11, 's31': 12, 's32':
13, 's33': 14, 's34': 15, 's35': 16, 's36': 17, 's41': 18, 's42': 19,
's43': 20, 's44': 21, 's45': 22, 's46': 23, 's51': 24, 's52': 25, 's53':
26, 's54': 27, 's55': 28, 's56': 29, 's61': 30, 's62': 31, 's63': 32,
's64': 33, 's65': 34, 's66': 35, 'sum1': 36}

```

```
class pymad8.Output.General
    Bases: object
```

General list of accelerator component infomation

```
addElement(type, name, data)
```

```
findByName(name)
```

```
findByType(type)
```

```
getColumn(key)
```

```
getIndex(name)
```

```
getNElements()
```

```
getNames(ind)
getRowByIndex(index)
getRowByName(name)
makeArray()
plotXY(xkey, ykey)
subline(start, end)
class pymad8.Output.Mad8(filename)
    Bases: object
    readFile(filename)
class pymad8.Output.OutputReader
    Bases: object

    Class to load different Mad8 output files Usage : o = Mad8.OutputReader() [c,
s] = o.readFile('./survey.tape','survey') [c, r] = o.readFile('./rmat.tape','rmat') [c, t]
= o.readFile('./twiss.tape','twiss') [c, c] = o.readFile('./chrom.tape','chrom') [c, e] =
o.readFile('./envelope.tape','envel')

c : Common data r : Rmat object t : Twiss object c : Chrom object e : Envelope object

readChromFile(f=None)
readEnvelopeFile(f=None)
readFile(fileName='', type='twiss')
    read mad8 output file
readRmatFile(f=None)
readSurveyFile()
readTwissFile(f=None)
class pymad8.Output.Rmat
    Bases: pymad8.Output.General

    Rmatrix data structure data : numpy array of data keys : key to data

    getData(index)

    keys = {'r11': 0, 'r12': 1, 'r13': 2, 'r14': 3, 'r15': 4, 'r16': 5, 'r21':
6, 'r22': 7, 'r23': 8, 'r24': 9, 'r25': 10, 'r26': 11, 'r31': 12, 'r32':
13, 'r33': 14, 'r34': 15, 'r35': 16, 'r36': 17, 'r41': 18, 'r42': 19,
'r43': 20, 'r44': 21, 'r45': 22, 'r46': 23, 'r51': 24, 'r52': 25, 'r53':
26, 'r54': 27, 'r55': 28, 'r56': 29, 'r61': 30, 'r62': 31, 'r63': 32,
'r64': 33, 'r65': 34, 'r66': 35, 'sum1': 36}
class pymad8.Output.Saveline(fileName, lineName='EBDS')
    Bases: object

    expandLine()
    findNamedDict(name)
    findNamedIndex(name)
    findRenamedNamedDict(name)
    findRenamedNamedIndex(name)
    makeSubLines()
    parseFile()
    readFile(fileName)
```



```

    removeDuplicates()
    removeReplacements()
    writeRenamed(filename)
class pymad8.Output.Survey
    Bases: pymad8.Output.General
    Survey data structure data : numpy array of data keys : key to data
    keys = {'phi': 5, 'psi': 6, 'suml': 3, 'theta': 4, 'x': 0, 'y': 1, 'z': 2}
class pymad8.Output.Track(folderpath, filemapname, twissname)
    Bases: object
    appendDir(folderpath)
        Loop over all mad8 track output files in the target directory and append the data to the existing data
        structure.
    readDir()
        Loop over all mad8 track output files in the target directory and build a dictionary of the data. File map
        is used to match data from track files to observation plane in the twiss file.
class pymad8.Output.Twiss
    Bases: pymad8.Output.General
    Twiss data structure data : numpy array of data keys : key to data
    keys = {'alfx': 0, 'alfy': 5, 'betx': 1, 'bety': 6, 'dpx': 4, 'dpy': 9,
    'dx': 3, 'dy': 8, 'mux': 2, 'muy': 7, 'px': 11, 'py': 13, 'suml': 14, 'x':
    10, 'y': 12}
    nameFromNearestS(s)
    plotAlf()
    plotBeta()
    plotEta()
    plotEtaPrime()
    plotMu()
pymad8.Output.getValueByName(name, key, common, table)
pymad8.Output.writeContinuation(f, l)

```

8.4 pymad8.Mad8 module

```

class pymad8.Mad8.Chrom
    Bases: pymad8.Mad8.General
    Chromaticity data structure data : numpy array of data keys : key to data
    getData(index)
class pymad8.Mad8.Common
    Bases: pymad8.Mad8.General
    containsEnergyVariation()
        Method to determine if the energy is constant in the lattice Required if there is 1) RfCavities
    getApertures(raw=True)
    getColumn(colName)
    getData(index)

```

`getRowByIndex(index)`

`getRowByName(name)`

```
keys = {'blmo': {'E': 11, 'l': 0, 'note': 10}, 'drif': {'E': 11, 'aper': 9,
'l': 0, 'note': 10}, 'ecol': {'E': 11, 'aper': 9, 'l': 0, 'note': 10,
'xsize': 4, 'ysize': 5}, 'elseparator': {'E': 11, 'aper': 9, 'efield': 5,
'l': 0, 'note': 10, 'tilt': 4}, 'hkic': {'E': 11, 'aper': 9, 'hkick': 4,
'l': 0, 'note': 10}, 'hmonitor': {'E': 11, 'aper': 9, 'l': 0, 'note': 10},
'imon': {'E': 11, 'l': 0, 'note': 10}, 'inst': {'E': 11, 'l': 0, 'note':
10}, 'kicker': {'E': 11, 'aper': 9, 'hkick': 4, 'l': 0, 'note': 10, 'vkick':
5}, 'lcav': {'E': 11, 'aper': 9, 'freq': 5, 'l': 0, 'lag': 7, 'note': 10,
'volt': 6}, 'mark': {'E': 11, 'l': 0, 'note': 10}, 'moni': {'E': 11, 'aper':
9, 'l': 0, 'note': 10}, 'mult': {'E': 11, 'aper': 9, 'k0l': 1, 'k1l': 2,
'k2l': 3, 'k3l': 5, 'note': 10, 't0': 4, 't1': 6, 't2': 7, 't3': 8},
'octu': {'E': 11, 'aper': 9, 'k3': 5, 'l': 0, 'note': 10, 'tilt': 4},
'prof': {'E': 11, 'l': 0, 'note': 10}, 'quad': {'E': 11, 'aper': 9, 'k1':
2, 'l': 0, 'note': 10, 'tilt': 4}, 'rben': {'E': 11, 'angle': 1, 'aper': 9,
'e1': 5, 'e2': 6, 'h1': 7, 'h2': 8, 'k1': 2, 'k2': 3, 'l': 0, 'note': 10,
'tilt': 4}, 'rcol': {'E': 11, 'aper': 9, 'l': 0, 'note': 10, 'xsize': 4,
'ysize': 5}, 'rfcavity': {'E': 11, 'aper': 9, 'freq': 5, 'l': 0, 'lag': 7,
'note': 10, 'volt': 6}, 'sben': {'E': 11, 'angle': 1, 'aper': 9, 'e1': 5,
'e2': 6, 'h1': 7, 'h2': 8, 'k1': 2, 'k2': 3, 'l': 0, 'note': 10, 'tilt':
4}, 'sext': {'E': 11, 'aper': 9, 'k2': 3, 'l': 0, 'note': 10, 'tilt': 4},
'solenoid': {'E': 11, 'aper': 9, 'ks': 5, 'l': 0, 'note': 10}, 'srot':
{'E': 11, 'angle': 5, 'aper': 9, 'l': 0, 'note': 10}, 'vkic': {'E': 11,
'aper': 9, 'l': 0, 'note': 10, 'vkick': 5}, 'vmonitor': {'E': 11, 'aper':
9, 'l': 0, 'note': 10}, 'wire': {'E': 11, 'l': 0, 'note': 10}, 'yrot':
{'E': 11, 'angle': 5, 'aper': 9, 'l': 0, 'note': 10}}
```

`makeLocationList(elementNames=[])`

`class pymad8.Mad8.EchoValue(echoFileName)`

Bases: object

`loadValues()`

`class pymad8.Mad8.Envelope`

Bases: [pymad8.Mad8.General](#)

Beam envelope data structure data : numpy array of data keys : key to data

`getData(index)`

```
keys = {'s11': 0, 's12': 1, 's13': 2, 's14': 3, 's15': 4, 's16': 5, 's21':
6, 's22': 7, 's23': 8, 's24': 9, 's25': 10, 's26': 11, 's31': 12, 's32':
13, 's33': 14, 's34': 15, 's35': 16, 's36': 17, 's41': 18, 's42': 19,
's43': 20, 's44': 21, 's45': 22, 's46': 23, 's51': 24, 's52': 25, 's53':
26, 's54': 27, 's55': 28, 's56': 29, 's61': 30, 's62': 31, 's63': 32,
's64': 33, 's65': 34, 's66': 35, 'sum1': 36}
```

`class pymad8.Mad8.General`

Bases: object

General list of accelerator component information

`addElement(type, name, data)`

`findByName(name)`

`findByType(type)`

`getColumn(key)`

`getIndex(name)`

```

getNElements()
getNames(ind)
getRowByIndex(index)
getRowByName(name)
makeArray()
plotXY(xkey, ykey)
subline(start, end)
class pymad8.Mad8.Mad8(filename)
    Bases: object
    readFile(filename)
class pymad8.Mad8.OutputReader
    Bases: object

    Class to load different Mad8 output files Usage : o = Mad8.OutputReader() [c,
    s] = o.readFile('./survey.tape','survey') [c, r] = o.readFile('./rmat.tape','rmat') [c, t]
    = o.readFile('./twiss.tape','twiss') [c, c] = o.readFile('./chrom.tape','chrom') [c, e] =
    o.readFile('./envelope.tape','envel')

    c : Common data r : Rmat object t : Twiss object c : Chrom object e : Envelope object

    readChromFile(f=None)
    readEnvelopeFile(f=None)
    readFile(fileName='', type='twiss')
        read mad8 output file
    readRmatFile(f=None)
    readSurveyFile()
    readTwissFile(f=None)
class pymad8.Mad8.Rmat
    Bases: pymad8.Mad8.General

    Rmatrix data structure data : numpy array of data keys : key to data

    getData(index)

    keys = {'r11': 0, 'r12': 1, 'r13': 2, 'r14': 3, 'r15': 4, 'r16': 5, 'r21':
    6, 'r22': 7, 'r23': 8, 'r24': 9, 'r25': 10, 'r26': 11, 'r31': 12, 'r32':
    13, 'r33': 14, 'r34': 15, 'r35': 16, 'r36': 17, 'r41': 18, 'r42': 19,
    'r43': 20, 'r44': 21, 'r45': 22, 'r46': 23, 'r51': 24, 'r52': 25, 'r53':
    26, 'r54': 27, 'r55': 28, 'r56': 29, 'r61': 30, 'r62': 31, 'r63': 32,
    'r64': 33, 'r65': 34, 'r66': 35, 'sum1': 36}
class pymad8.Mad8.Survey
    Bases: pymad8.Mad8.General

    Survey data structure data : numpy array of data keys : key to data

    keys = {'phi': 5, 'psi': 6, 'sum1': 3, 'theta': 4, 'x': 0, 'y': 1, 'z': 2}
class pymad8.Mad8.Twiss
    Bases: pymad8.Mad8.General

    Twiss data structure data : numpy array of data keys : key to data

    keys = {'alfx': 0, 'alfy': 5, 'betx': 1, 'bety': 6, 'dpx': 4, 'dpy': 9,
    'dx': 3, 'dy': 8, 'mux': 2, 'muy': 7, 'px': 11, 'py': 13, 'sum1': 14, 'x':
    10, 'y': 12}

```

```
nameFromNearestS(s)
plotAlf()
plotBeta()
plotEta()
plotEtaPrime()
plotMu()
```

```
pymad8.Mad8.getValueByName(name, key, common, table)
```

8.5 pymad8.Plot module

```
pymad8.Plot.AddMachineLatticeToFigure(figure, mad8opt, tightLayout=True)
```

Add a diagram above the current graph in the figure that represents the accelerator based on a madx twiss file in tfs format.

Note you can use matplotlib's `gcf()` 'get current figure' as an argument.

```
>>> pymadx.Plot.AddMachineLatticeToFigure(gcf(), 'afile.tfs')
```

```
pymad8.Plot.apertures(twissfile='ebds1', envelfile='ebds1')
```

```
pymad8.Plot.beamSize(envelfile='ebds1', twissfile='ebds1')
```

```
pymad8.Plot.dispersion(twissfile='ebds1')
```

```
pymad8.Plot.dispersionPrime(twissfile='ebds1')
```

```
pymad8.Plot.drawMachineLattice(mad8c, mad8t)
```

```
pymad8.Plot.energy(twissfile='ebds1')
```

```
pymad8.Plot.linearOptics(twissfile='ebds1')
```

```
pymad8.Plot.phaseAdvance(twissfile='ebds1')
```

```
pymad8.Plot.setCallbacks(figure, axm, axplot, twiss)
```

```
pymad8.Plot.survey(surveyfile='ebds1')
```

8.6 pymad8.Sim module

```
class pymad8.Sim.Track(common, rmat)
```

Bases: object

```
generate()
```

```
trackParticle(p)
```

```
trackParticles(nparticle)
```

```
pymad8.Sim.testTrack(rmatFile, nparticle=10)
```

8.7 pymad8.Visualisation module

`pymad8.Visualisation.MakeCombinedSurveyPlot`(*name*, *QUAD=True*, *RBEN=True*, *SBEN=True*,
MONI=True, *MARK=True*)

Takes a list of Survey filenames, plots them all on the same 2D plot. For branching machines or segmented models. Elements selectable via booleans, default to true

class `pymad8.Visualisation.OneDim`(*common*, *survey*, *debug*)

Bases: `object`

`drawBend`(*c*, *s*, *suml*, *colour=True*)

`drawElement`(*elem*, *colour=True*)

`drawElements`(*type*, *colour=True*)

`drawHkic`(*c*, *s*, *suml*, *colour=True*)

`drawInst`(*c*, *s*, *suml*, *colour=True*)

`drawMark`(*c*, *s*, *suml*, *colour=True*)

`drawMoni`(*c*, *s*, *suml*, *colour=True*)

`drawMult`(*c*, *s*, *suml*, *colour=True*)

`drawProf`(*c*, *s*, *suml*, *colour=True*)

`drawQuad`(*c*, *s*, *suml*, *colour=True*)

`drawSext`(*c*, *s*, *suml*, *colour=True*)

`drawVkic`(*c*, *s*, *suml*, *colour=True*)

`drawWire`(*c*, *s*, *suml*, *colour=True*)

`plot`(*colour=True*)

class `pymad8.Visualisation.TwoDim`(*common*, *survey*, *debug=False*, *annotate=False*, *fancy=False*)

Bases: `object`

`drawBend`(*c*, *s*, *x*, *y*, *z*)

`drawElement`(*elem*)

`drawElements`(*type*)

`drawMark`(*c*, *s*, *x*, *y*, *z*)

`drawMoni`(*c*, *s*, *x*, *y*, *z*)

`drawQuad`(*c*, *s*, *x*, *y*, *z*)

`plot`(*event=None*)

`plotUpdate`(*event*)

`pymad8.Visualisation.testOneDim`()

`pymad8.Visualisation.testTwoDim`()

`pymad8.Visualisation.transformedPoly`(*xy*, *xyz*, *theta*)

`pymad8.Visualisation.transformedRect`(*xyz*, *dx*, *dy*, *theta*)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

- `pymad8`, [17](#)
- `pymad8.Input`, [17](#)
- `pymad8.Mad8`, [21](#)
- `pymad8.Output`, [18](#)
- `pymad8.Plot`, [24](#)
- `pymad8.Sim`, [24](#)
- `pymad8.Visualisation`, [25](#)

A

addElement() (*pymad8.Mad8.General method*), 22
 addElement() (*pymad8.Output.General method*), 19
 AddMachineLatticeToFigure() (*in module pymad8.Plot*), 24
 apertures() (*in module pymad8.Plot*), 24
 appendDir() (*pymad8.Output.Track method*), 21

B

beamSize() (*in module pymad8.Plot*), 24

C

Chrom (*class in pymad8.Mad8*), 21
 Chrom (*class in pymad8.Output*), 18
 Common (*class in pymad8.Mad8*), 21
 Common (*class in pymad8.Output*), 18
 containsEnergyVariation() (*py-mad8.Mad8.Common method*), 21
 containsEnergyVariation() (*py-mad8.Output.Common method*), 18

D

decodeCollimator() (*in module pymad8.Input*), 17
 decodeDecapole() (*in module pymad8.Input*), 17
 decodeDrift() (*in module pymad8.Input*), 17
 decodeFileLine() (*in module pymad8.Input*), 17
 decodeKicker() (*in module pymad8.Input*), 17
 decodeLcavity() (*in module pymad8.Input*), 17
 decodeLine() (*in module pymad8.Input*), 17
 decodeMultipole() (*in module pymad8.Input*), 17
 decodeNameAndType() (*in module pymad8.Input*), 17
 decodeNamed() (*in module pymad8.Input*), 17
 decodeOctupole() (*in module pymad8.Input*), 18
 decodeQuadrupole() (*in module pymad8.Input*), 18
 decodeSbend() (*in module pymad8.Input*), 18
 decodeSextupole() (*in module pymad8.Input*), 18
 dispersion() (*in module pymad8.Plot*), 24
 dispersionPrime() (*in module pymad8.Plot*), 24
 drawBend() (*pymad8.Visualisation.OneDim method*), 25
 drawBend() (*pymad8.Visualisation.TwoDim method*), 25
 drawElement() (*pymad8.Visualisation.OneDim method*), 25
 drawElement() (*pymad8.Visualisation.TwoDim method*), 25

drawElements() (*pymad8.Visualisation.OneDim method*), 25
 drawElements() (*pymad8.Visualisation.TwoDim method*), 25
 drawHkic() (*pymad8.Visualisation.OneDim method*), 25
 drawInst() (*pymad8.Visualisation.OneDim method*), 25
 drawMachineLattice() (*in module pymad8.Plot*), 24
 drawMark() (*pymad8.Visualisation.OneDim method*), 25
 drawMark() (*pymad8.Visualisation.TwoDim method*), 25
 drawMoni() (*pymad8.Visualisation.OneDim method*), 25
 drawMoni() (*pymad8.Visualisation.TwoDim method*), 25
 drawMult() (*pymad8.Visualisation.OneDim method*), 25
 drawProf() (*pymad8.Visualisation.OneDim method*), 25
 drawQuad() (*pymad8.Visualisation.OneDim method*), 25
 drawQuad() (*pymad8.Visualisation.TwoDim method*), 25
 drawSext() (*pymad8.Visualisation.OneDim method*), 25
 drawVkic() (*pymad8.Visualisation.OneDim method*), 25
 drawWire() (*pymad8.Visualisation.OneDim method*), 25

E

EchoValue (*class in pymad8.Mad8*), 22
 EchoValue (*class in pymad8.Output*), 19
 energy() (*in module pymad8.Plot*), 24
 Envelope (*class in pymad8.Mad8*), 22
 Envelope (*class in pymad8.Output*), 19
 expandLine() (*pymad8.Output.Saveline method*), 20

F

findByName() (*pymad8.Mad8.General method*), 22
 findByName() (*pymad8.Output.General method*), 19
 findByType() (*pymad8.Mad8.General method*), 22
 findByType() (*pymad8.Output.General method*), 19
 findNamedDict() (*pymad8.Output.Saveline method*), 20

`findNamedIndex()` (*pymad8.Output.Saveline method*), 20
`findRenamedNamedDict()` (*pymad8.Output.Saveline method*), 20
`findRenamedNamedIndex()` (*pymad8.Output.Saveline method*), 20

G

`General` (*class in pymad8.Mad8*), 22
`General` (*class in pymad8.Output*), 19
`generate()` (*pymad8.Sim.Track method*), 24
`getApertures()` (*pymad8.Mad8.Common method*), 21
`getApertures()` (*pymad8.Output.Common method*), 18
`getColumn()` (*pymad8.Mad8.Common method*), 21
`getColumn()` (*pymad8.Mad8.General method*), 22
`getColumn()` (*pymad8.Output.Common method*), 18
`getColumn()` (*pymad8.Output.General method*), 19
`getData()` (*pymad8.Mad8.Chrom method*), 21
`getData()` (*pymad8.Mad8.Common method*), 21
`getData()` (*pymad8.Mad8.Envelope method*), 22
`getData()` (*pymad8.Mad8.Rmat method*), 23
`getData()` (*pymad8.Output.Chrom method*), 18
`getData()` (*pymad8.Output.Common method*), 18
`getData()` (*pymad8.Output.Envelope method*), 19
`getData()` (*pymad8.Output.Rmat method*), 20
`getIndex()` (*pymad8.Mad8.General method*), 22
`getIndex()` (*pymad8.Output.General method*), 19
`getNames()` (*pymad8.Mad8.General method*), 23
`getNames()` (*pymad8.Output.General method*), 19
`getNElements()` (*pymad8.Mad8.General method*), 22
`getNElements()` (*pymad8.Output.General method*), 19
`getRowByIndex()` (*pymad8.Mad8.Common method*), 21
`getRowByIndex()` (*pymad8.Mad8.General method*), 23
`getRowByIndex()` (*pymad8.Output.Common method*), 18
`getRowByIndex()` (*pymad8.Output.General method*), 20
`getRowByName()` (*pymad8.Mad8.Common method*), 22
`getRowByName()` (*pymad8.Mad8.General method*), 23
`getRowByName()` (*pymad8.Output.Common method*), 18
`getRowByName()` (*pymad8.Output.General method*), 20
`getValueByName()` (*in module pymad8.Mad8*), 24
`getValueByName()` (*in module pymad8.Output*), 21

K

`keys` (*pymad8.Mad8.Common attribute*), 22
`keys` (*pymad8.Mad8.Envelope attribute*), 22
`keys` (*pymad8.Mad8.Rmat attribute*), 23
`keys` (*pymad8.Mad8.Survey attribute*), 23
`keys` (*pymad8.Mad8.Twiss attribute*), 23

`keys` (*pymad8.Output.Common attribute*), 18
`keys` (*pymad8.Output.Envelope attribute*), 19
`keys` (*pymad8.Output.Rmat attribute*), 20
`keys` (*pymad8.Output.Survey attribute*), 21
`keys` (*pymad8.Output.Twiss attribute*), 21

L

`linearOptics()` (*in module pymad8.Plot*), 24
`loadMarkedValues()` (*pymad8.Output.EchoValue method*), 19
`loadValues()` (*pymad8.Mad8.EchoValue method*), 22
`loadValues()` (*pymad8.Output.EchoValue method*), 19

M

`Mad8` (*class in pymad8.Mad8*), 23
`Mad8` (*class in pymad8.Output*), 20
`makeArray()` (*pymad8.Mad8.General method*), 23
`makeArray()` (*pymad8.Output.General method*), 20
`MakeCombinedSurveyPlot()` (*in module pymad8.Visualisation*), 25
`makeLocationList()` (*pymad8.Mad8.Common method*), 22
`makeLocationList()` (*pymad8.Output.Common method*), 19
`makeSubLines()` (*pymad8.Output.Saveline method*), 20
`module`
 pymad8, 17
 pymad8.Input, 17
 pymad8.Mad8, 21
 pymad8.Output, 18
 pymad8.Plot, 24
 pymad8.Sim, 24
 pymad8.Visualisation, 25

N

`nameFromNearestS()` (*pymad8.Mad8.Twiss method*), 23
`nameFromNearestS()` (*pymad8.Output.Twiss method*), 21

O

`OneDim` (*class in pymad8.Visualisation*), 25
`OutputReader` (*class in pymad8.Mad8*), 23
`OutputReader` (*class in pymad8.Output*), 20

P

`parseFile()` (*pymad8.Output.Saveline method*), 20
`phaseAdvance()` (*in module pymad8.Plot*), 24
`plot()` (*pymad8.Visualisation.OneDim method*), 25
`plot()` (*pymad8.Visualisation.TwoDim method*), 25
`plotAlf()` (*pymad8.Mad8.Twiss method*), 24
`plotAlf()` (*pymad8.Output.Twiss method*), 21
`plotBeta()` (*pymad8.Mad8.Twiss method*), 24
`plotBeta()` (*pymad8.Output.Twiss method*), 21
`plotEta()` (*pymad8.Mad8.Twiss method*), 24

plotEta() (*pymad8.Output.Twiss method*), 21
 plotEtaPrime() (*pymad8.Mad8.Twiss method*), 24
 plotEtaPrime() (*pymad8.Output.Twiss method*), 21
 plotMu() (*pymad8.Mad8.Twiss method*), 24
 plotMu() (*pymad8.Output.Twiss method*), 21
 plotUpdate() (*pymad8.Visualisation.TwoDim method*), 25
 plotXY() (*pymad8.Mad8.General method*), 23
 plotXY() (*pymad8.Output.General method*), 20
 pymad8
 module, 17
 pymad8.Input
 module, 17
 pymad8.Mad8
 module, 21
 pymad8.Output
 module, 18
 pymad8.Plot
 module, 24
 pymad8.Sim
 module, 24
 pymad8.Visualisation
 module, 25

R

readChromFile() (*pymad8.Mad8.OutputReader method*), 23
 readChromFile() (*pymad8.Output.OutputReader method*), 20
 readDir() (*pymad8.Output.Track method*), 21
 readEnvelopeFile() (*pymad8.Mad8.OutputReader method*), 23
 readEnvelopeFile() (*pymad8.Output.OutputReader method*), 20
 readFile() (*pymad8.Mad8.Mad8 method*), 23
 readFile() (*pymad8.Mad8.OutputReader method*), 23
 readFile() (*pymad8.Output.Mad8 method*), 20
 readFile() (*pymad8.Output.OutputReader method*), 20
 readFile() (*pymad8.Output.Saveline method*), 20
 readRmatFile() (*pymad8.Mad8.OutputReader method*), 23
 readRmatFile() (*pymad8.Output.OutputReader method*), 20
 readSurveyFile() (*pymad8.Mad8.OutputReader method*), 23
 readSurveyFile() (*pymad8.Output.OutputReader method*), 20
 readTwissFile() (*pymad8.Mad8.OutputReader method*), 23
 readTwissFile() (*pymad8.Output.OutputReader method*), 20
 removeComments() (*in module pymad8.Input*), 18
 removeContinuationSymbols() (*in module pymad8.Input*), 18
 removeDuplicates() (*pymad8.Output.Saveline method*), 20

removeReplacements() (*pymad8.Output.Saveline method*), 21
 Rmat (*class in pymad8.Mad8*), 23
 Rmat (*class in pymad8.Output*), 20

S

Saveline (*class in pymad8.Output*), 20
 setCallbacks() (*in module pymad8.Plot*), 24
 splitKeyValue() (*in module pymad8.Input*), 18
 subline() (*pymad8.Mad8.General method*), 23
 subline() (*pymad8.Output.General method*), 20
 Survey (*class in pymad8.Mad8*), 23
 Survey (*class in pymad8.Output*), 21
 survey() (*in module pymad8.Plot*), 24

T

testOneDim() (*in module pymad8.Visualisation*), 25
 testTrack() (*in module pymad8.Sim*), 24
 testTwoDim() (*in module pymad8.Visualisation*), 25
 tidy() (*in module pymad8.Input*), 18
 Track (*class in pymad8.Output*), 21
 Track (*class in pymad8.Sim*), 24
 trackParticle() (*pymad8.Sim.Track method*), 24
 trackParticles() (*pymad8.Sim.Track method*), 24
 transformedPoly() (*in module pymad8.Visualisation*), 25
 transformedRect() (*in module pymad8.Visualisation*), 25
 Twiss (*class in pymad8.Mad8*), 23
 Twiss (*class in pymad8.Output*), 21
 TwoDim (*class in pymad8.Visualisation*), 25

W

writeContinuation() (*in module pymad8.Output*), 21
 writeRenamed() (*pymad8.Output.Saveline method*), 21