pymad8 Documentation

Release 1.6.1

Royal Holloway

Jun 15, 2021

CONTENTS

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

CHAPTER

ONE

LICENCE & DISCLAIMER

CHAPTER TWO

AUTHORSHIP

The following people have contributed to pymad8:

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

CHAPTER THREE

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.

CHAPTER FOUR

DATA LOADING

Utilies to load pymad8 output data.

CHAPTER

FIVE

CONVERTING MODELS

pymad8 provdies converters to allow BDSIM models to 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
```

CHAPTER

SIX

PLOTTING

CHAPTER

SEVEN

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.

CHAPTER

EIGHT

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, 'suml': 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, 'suml': 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, 'suml': 36}

class pymad8.Mad8.**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.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, 'suml': 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, 'suml': 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, 'suml': 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='ebs1'*)

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*, *xyc*, *theta*)

pymad8.Visualisation.**transformedRect**(*xyc*, *dx*, *dy*, *theta*)

CHAPTER NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX