pytransport Documentation

Release 2.0.0

Royal Holloway

Mar 19, 2023

CONTENTS

pytransport is a set of classes and functions to load MADX output as well as prepare Transport models. The package overall functions as a holder for any code required to load and manipulate Transport output data.

CHAPTER

ONE

LICENCE & DISCLAIMER

CHAPTER

TWO

AUTHORSHIP

The following people have contributed to pytransport:

- William Shields
- Jochem Snuverink
- Laurie Nevay
- Stuart Walker

CHAPTER

THREE

INSTALLATION

pytransport is developed for Python 3. The developers use 3.8 to 3.11. It can be install through pip (with internet access):

```
pip install pytransport
```

3.1 Requirements

pytransport depends on the following Python packages available through pip:

- matplotlib
- numpy
- scipy
- pymadx
- pybdsim

3.2 Local Installation

Although on pip, for development purposes you may wish to use pytransport from a copy of the source code. It is possible to clone the git repository and use pip to *point* at the local set of files, or generally install that set of files as a once off.

We have provided a simple Makefile in the main pybdsim directory that has a small set of 'rules' that should help with the relevant pip commands. pip is used even though pybdsim is found from the local set of files.

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

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

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

CHAPTER

FOUR

CONVERSION

pytransport can convert a TRANSPORT "FOR001" file to BDSIM's *gmad* format:

```
>>> from pytransport import Convert
>>> import pybdsim.Builder
>>> file = 'FOR001.DAT'
>>> Convert.Convert(file, machine=pybdsim.Builder.Machine(), options=pybdsim.Options.
↪Options())

Writing to file: bdsim/FOR001.gmad
Lattice written to:
FOR001_components.gmad
FOR001_sequence.gmad
FOR001_beam.gmad
FOR001_options.gmad
All included in main file:
FOR001.gmad
```

CHAPTER

FIVE

OPTICS

pytransport can read a TRANSPORT *FOR002* output file that has sigma matrices.

```
>>> import pytransport
>>> optics = pytransport.Reader.GetOptics('FOR002.DAT')
>>> import matplotlib.pyplot as plt
>>> plt.plot(optics.S(), optics.Sigma_X())
```

Also with *pybdsim* the TRANSPORT optics can be directly compared with BDSIM:

```
>>> pybdsim.Compare.TransportVsBDSIM('FOR002.DAT', 'bdsim_optics.root')
```

CHAPTER

SIX

MODULE CONTENTS

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

pytransport - Royal Holloway utility to manipulate TRANSPORT data and models.

Authors:

- William Shields
- Jochem Snuverink

6.1 pytransport.Convert module

Wrapper function for automatically converting to bdsim and/or madx

Class containing functions to help convert elements from Transport to gmad/madx. Class instantiated in pybdsim.Convert.Transport2Gmad and pymadx.Convert.Transport2Madx.

Classes: _Convert - a class used to convert different element types.

pytransport.Convert.**Convert**(*inputfile*, *particle='proton'*, *distrType='gauss'*, *output='bdsim'*, *outputDir=''*, *debug=False*, *dontSplit=False*, *keepName=False*, *combineDrifts=False*, *options=None*, *machine=None*)

Convert convert a Transport input or output file into an input file for bdsim, madx, or both

inputfile	dtype = string path to the input file
particle	dtype = string. Optional, default = "proton" the particle species
distrType	dtype = string. Optional, Default = "gauss". the beam distribution type. Can be either gauss or gausstwiss.
output	dtype=string. Optional, default = "bdsim" the output type, can be "bdsim", "madx", or "both"
outputDir	dtype=string. Optional, default = "" the output directory where the files will be written if no input supplied, defaults to output type. if outputDir is supplied and output is "both", outputDir is appended with the output type, e.g. outputDir_bdsim
debug	dtype = bool. Optional, default = False output a log file (inputfile_conversion.log) detailing the conversion process, element by element
dontSp	dtype = bool. Optional, default = False the converter splits the machine into multiple parts when a beam is redefined in a Transport lattice. dontSplit overrides this and forces the machine to be written to a single file
keepName	dtype = bool. Optional, default = False keep the names of elements as defined in the Transport inputfile. Appends element name with _N where N is an integer if the element name has already been used
combineDr	dtype = bool. Optional, default = False combine multiple consecutive drifts into a single drift
machine	dtype = pybdsim.Builder.Machine or pymadx.Builder.Machine required, default = None machine instance required for conversion.
options	dtype = pybdsim.Options.Options. Optional, default = None options instance required to write options in bdsim conversion. Ignored if converting to "madx" format.

Example:

```
>>> Convert(inputfile, machine=pybdsim.Builder.Machine(), options=pybdsim.Options.
↪Options())
>>> Convert(inputfile, output="madx", machine=pymadx.Builder.Machine())
```

Writes converted machine to disk. Reader automatically detects if the supplied input file is a Transport input file or Transport output file.

6.2 pytransport.Data module

Data

Data containers used in converting from Transport to gmad or madx.

Classes: BDSData - a list of data read from Transport files. ConversionData - a class for holding data during conversion.

class pytransport.Data.**BDSData**(**args*, ***kwargs*)

Bases: `list`

General class representing simple 2 column data.

Inherits python list. It's a list of tuples with extra columns of 'name' and 'units'.

ConcatenateMachine(**args*)

This is used to concatenate machines.

Filter(*booleanarray*)

Filter the data with a booleanarray. Where true, will return that event in the data.

Return type is BDSData

GetColumn(*columnstring*)

Return a numpy array of the values in columnstring in order as they appear in the beamline

GetItemTuple(*index*)

Get a specific entry in the data as a tuple of values rather than a dictionary.

IndexFromNearestS(*S*)

IndexFromNearestS(S)

return the index of the beamline element clostest to S

Only works if "SStart" column exists in data

MatchValue(*parametername*, *matchvalue*, *tolerance*)

This is used to filter the instance of the class based on matching a parameter withing a certain tolerance.

```
>>> a = pytransport.Data.Load("myfile.txt")
>>> a.MatchValue("S",0.3,0.0004)
```

this will match the "S" variable in instance "a" to the value of 0.3 within +- 0.0004.

You can therefore used to match any parameter.

Return type is BDSAsciiData

MergeDuplicatesAtSameS()

Merge duplicate entries at the same s position. This is to prevent having multiple entries at the same s in situations such as poleface rotations. Will merge zero-length items into finite length items.

NameFromNearestS(*S*)

class pytransport.Data.**ConversionData**(*inputfile*, *machine*, *options=None*, *particle='proton'*, *debug=False*, *distrType='gauss'*, *gmad=True*, *gmadDir='gmad'*, *madx=False*, *madxDir='madx'*, *auto=True*, *dontSplit=False*, *keepName=False*, *combineDrifts=False*, *outlog=True*)

Bases: object

Class used as data container object in Transport2Gmad / Transport2Madx conversion. Required input: - inputfile: string, inputfile name - machine: either pybdsim.Builder.Machine or pymadx.Builder.Machine instance.

Note: if used as a holder for conversion to gmad, options must be supplied a pybdsim.Options.Options instance.

This class will hold ALL conversion related data, some stored in member variables which are separate containers for: conversion related properties (user input arguments), beam properties, and machine properties.

AddBeam()

Function to prepare the beam and add to the machine.

AddOptions()

Function to set the Options for a BDSIM machine.

ResetMachine()

Delete the machine and set to be the empty machine copied at class instantiation.

6.3 pytransport.Reader module

Reader

Readers for loading Transport input files and output files. Can extract the individual lattice, optics, and fitting sections.

Classes: Reader - a list of data read from Transport files. ConversionData - a class for holding data during conversion.

pytransport.Reader.**GetFitsSection**(*inputFile*)

Function to get the fit routine data from the standard transport output. Returns two lists, the first with the direct output from the fitting data, the second with the first line of each element in the output data, which contains the element parameters with their fitted values.

pytransport.Reader.**GetLattice**(*inputFile*)

Function to extract the lattice from a standard output file.

pytransport.Reader.**GetOptics**(*inputFile*, *inputType=None*)

Extract the optics from a Transport output file.

pytransport.Reader.**GetResultsFromFitting**(*inputFile*)

6.4 pytransport._General module

General utilities for day to day housekeeping

Classes: _Writer - a class used for writing any output during conversion.

pytransport._General.**CheckDirExists**(*directory*)

pytransport._General.**CheckIsAddition**(*line*, *filetype='input'*)

Function to check if a BEAM line of TRANSPORT code is a beam definition or r.m.s addition.

pytransport._General.**CheckIsOutput**(*inputfile*)

Function to check if a file is a standard TRANSPORT output file. Based upon existence of the lines:

```
"0  XXX"
```

being present, which represents the TRANSPORT indicator card line. X can be 0, 1, 2. Default is 0.

pytransport._General.**CheckIsSentinel**(*line*)

pytransport._General.**CheckSingleLineOutputApplied**(*inputfile*)

Function to check if the control element that print element output in a single line was successfully applied. Check needed as not all versions of TRANSPORT can run this type code.

pytransport._General.**ConvertBunchLength**(*transport*, *bunch_length*)

Function to convert bunch length unit in TRANSPORT into seconds.

pytransport._General.**FindEndOfLine**(*line*)

pytransport._General.**GetComment**(*line*)

Function to extract a comment from a line. Will only find one comment per line.

pytransport._General.**GetElementData**(*line*)

pytransport._General.**GetFaceRotationAngles**(*data*, *linenum*)

pytransport._General.**GetIndicator**(*data*)

Function to read the indicator number. Must be 0, 1, or 2, where:
0 is a new lattice 1 is for fitting with the first lattice (from a 0 indicator file) 2 if for a second fitting which suppresses the first fitting.

pytransport._General.**GetLabel**(*line*)

Function to get element label from code line.

pytransport._General.**GetPreamble**(*data*)

Function to read any preamble at the start of the TRANSPORT file.

pytransport._General.**GetTypeNum**(*line*)

Function to extract the element type number (type code). Written because element types can contain alphabetical characters when fits are used, e.g: 5.0A. Only the number is required, the use of fitting does not need to be known.

pytransport._General.**JoinSplitLines**(*linenum*, *lattice*)

pytransport._General.**OutputFitsToRegistry**(*transport*, *outputdata*)

pytransport._General.**ProcessFits**(*fits*)

pytransport._General.**RemoveFileExt**(*inputfile*)

Remove the file extension from the input file name. Only works on known extensions.

pytransport._General.**RemoveIllegals**(*line*)

Function to remove ‘’ and stray characters from lines.

pytransport._General.**RemoveLabel**(*line*)

Function to remove the label from a line.

pytransport._General.**RemoveSpaces**(*line*)

pytransport._General.**ScaleToMeters**(*transport*, *quantity*)

Function to scale quantity (string) to meters, returns conversion factor.

pytransport._General.**UpdateEnergyFromMomentum**(*transport*, *momentum*)

Function to calculate (from momentum):
Total Energy Kinetic Energy Momentum Lorentz factor (gamma) Velocity (beta) Magnetic rigidity (brho)

pytransport._General.**UpdateMomentumFromEnergy**(*transport*, *k_energy*)

Function to calculate (from kinetic energy):
Total Energy Kinetic Energy Momentum Lorentz factor (gamma) Velocity (beta) Magnetic rigidity (brho)

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

R

S

U