Using PyZgoubi

Kiel Hock[1]

[1]Collider-Accelerator Department
Brookhaven National Laboratory

APHY 689, February 14, 2018

Table of Contents

Installation

Install Prerequisites

Option 1 (Install piecemeal)

sudo apt-get install python-numpy python-scipy python-matplotlib python-pip

Option 2 (Install Anaconda)

https://docs.anaconda.com/anaconda/install/linux

Install PyZgoubi

(http://www.hep.manchester.ac.uk/u/samt/pyzgoubi/doc/trunk/install.html)

pip install --user pyzgoubi

Edit settings in /.pyzgoubi/settings.ini

change zgoubi_path= to zgoubi directory

Headers

```
'OBJET'
1.0013389933e+04
2
1 1
0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.000e+00 ' '
1
'PARTICUL'
2.8083914800e+03 3.2043529740e-19 -4.1841538200e+00 0.0e+0 0
```

```
bstr=Line('Booster')
ob=OBJET2()
ob.set(BORO=BRHO)
ob.add(Y=0.,D=1.)
bstr.add(ob)
bstr.add(HELION())
```

OBJET

BORO
(Reference rigidity)

KOBJ=2
(Particle coordinates entered explicitly)

IMAX, IDMAX
(Number of Particles, Number of Momenta)

Y,T,Z,P,S,D,LET
(Initial coordinates (x,xp,y,yp,s,sp), label)

IEX
(ray trace on/off (1/-9), IMAX times)

PARTICUL

M, Q, G, τ, s
(Particle parameters)

Reference Zgoubi User's guide and compare with PyZgoubi source code (installation directory/lib/python/site-packages/zgoubi), or "pyzgoubi -help ELEMENT" in terminal, to check how to declare parameters.

Adding Elements

Set parameters

```
ld1=ld1=99.61422        #length of first drift in cell
ld2=28.9875             #length of second drift in cell
QS=49.3                 #Length of focusing quadrupole
QL=50.4                 #Length of defocusing quadrupole
R0=8.25                 #Bore radius
BQH=4.4972998591        #Focusing quadrupole field component at poletip
BQV= -4.2757116541      #Defocusing quadrupole field component at poletip
xpas=1                  #Integration Step
```

Prepare elements

```
QF =QUADRUPO('QF scal',XL=QS, R_0=BR, B_0=BQH, XPAS=xpas, KPOS=1)
QD =QUADRUPO('QD scal',XL=QL, R_0=BR, B_0=BQV, XPAS=xpas, KPOS=1)
dr1 =DRIFT('Drift 1', XL=ld1)
dr2 =DRIFT('Drift 2', XL=ld2)
```

Add elements to lattice

```
cell={}
for i in range(1,9):
cell[i]=Line('cell%s'%i)

cell[1].add(dr1,QD,dr2,DIPO)
```

Completing the lattice

Input file (ex. eRHIC.py)

```
bstr=Line('Booster')
bstr.add(ob)
bstr.add(HELION())
bstr.add(scal)
bstr.add(FAISTORE(FNAME='zgoubi.fai',IP=1))
eRHICarc.add(FAISCEAU())
cell[2].add(dr1,QF,dr2,DIPO)
cell[3].add(dr1,QD,dr2,dr3)
cell[4].add(dr1,QF,dr2,DIPO)
cell[5].add(dr1,QD,dr2,DIPO)
cell[6].add(dr1,QF,dr2,dr3)
cell[7].add(dr1,QD,dr2,DIPO)
cell[8].add(dr1,QF,dr2,DIPO)
SP=Line('SP');[SP.add(cell[i]) for i in cell]
ring=Line('Ring');[ring.add(SP) for i in range(0,6)]
bstr.add(ring) bstr.add(END())
r=bstr.run(xterm=False)
r.save_res("bstr.res")
r.save_fai("bstr.fai")
```

From your terminal
pyzgoubi Booster.py

Notes

Line.run is the segment of code that executes zgoubi.

xterm=True/False opens an xterm for interacting with files in the temporary directory (including use of zpop).

The .res file, and any other files requested (.fai via FAISTORE, for example), are cached in a temporary directory (that is deleted after the program terminates). Using line_object.save_res('filename.res'), tells pyzgoubi that you want the file saved in your current directory.

Notes

Although PyZgoubi provides an efficient interface with Zgoubi, there are drawbacks.

- Everything has an added step of going through python which adds computing time.
- In lines 109-115 there are some timers used to calculate added times (will vary by computer).

	Time Used [s]	Added Time[s]
Defining Lattice Parameters	0.0009	0.0009
PyZgoubi Processing time	0.0233	0.0233
Zgoubi Execution time	0.1480	0.0
Saving zgoubi.res file	0.0012	0.0012

- The Defining Lattice Parameters, and associated calculations, adds a small amount of time that would likely be calculated by a program external to zgoubi.
- PyZgoubi Processing time includes starting and executing Zgoubi. This can add up significantly if performing parameter optimization with python, requiring many zgoubi startups.
- Zgoubi Execution time is unavoidable.
- Saving zgoubi.res file is moving the .res file from the temporary directory to the working directory. This is faster than cp or mv.

FIT

PyZgoubi relies on parameter optimization done by python which can add a significant amount to computing time.

Let us add routine to PyZgoubi that uses Zgoubi to find the closed orbit.

Zgoubi syntax for using FIT to find the closed orbit.

```
'FIT'
2
1 30 0 [-2., 2.]
1 31 0 [-10., 10.]
2 1e-10
3.1 1 2 #End 0. 1. 0
3.1 1 3 #End 0. 1. 0
```

It works.

Edit static_defs.py pyzgoubi directory /lib/python2.7/site-packages/zgoubi/ and add

```
class FCO(zgoubi_element):
  def output(self):
      out = "'FIT'" + nl
      out += "2" + nl
      out += "1 30 0 [-2., 2.]" + nl
      out += "1 31 0 [-10., 10.]" + nl
      out += "2 1e-10" + nl
      out += "3.1 1 2 #End 0. 1. 0" + nl
      out += "3.1 1 3 #End 0. 1. 0" + nl
      return out
```

Twiss and Matrix

Matrix and TWISS require OBJET5 instead of OBJET2.

```
ob5=OBJET5()
ob5.set(BORO=BRHO)
ob5.set(YR=0.,TR=0., ZR=0., PR=0., DR=1.0)
ob5.set(PY=1.e-3, PT=1.e-3, PZ=1.e-3, PX=0., PD=1.e-4)
bstr.replace(ob, ob5)
matrix=MATRIX(IORD=1, IFOC=11)
bstr.replace(FAISCEAU(), matrix)
bstr.add(END())
r=bstr.run()
tune=r.get_tune()
print tune
twss=TWISS(KTW1=2, FAC_D=1, FAC_A=1)
bstr.replace(matrix, twss)
r=bstr.run()
r.save_twiss('zgoubi.twiss')
```

You should get an error when you try to run the TWISS calculation. Let's add a more sophisticated routine into PyZgoubi.

```
class TWISS(zgoubi_element):
        _class_name='TWISS'
        _zgoubi_name='TWISS'
         def   __init__(self, label1='', label2='', **settings):
                self._params=
                self._types=
                self._params['label1'] = label1
                self._params['label2'] = label2
                self._params['KTW1']=0
                self._types['KTW1']='I'
                self._params['FAC_D']=0
                self._types['FAC_D']='E'
                self._params['FAC_A']=0
                self._types['FAC_A']='E'
                self.set(settings)
         def   output(self):
                I=self.i2s
                E=self.f2s
                A=str
                X=self.x2s
                out = ''
                nl = '/n'
                sq = ''''
                out += sq + 'TWISS' + sq + ' ' + self._params['label1'][:8] + ' ' + self._params['label2'][:8] + nl
                out += I(self._params['KTW1']) + ' ' + E(self._params['FAC_D']) + ' ' + E(self._params['FAC_A']) +nl
                return out
```

Multi-turn Tracking

Adding RF

```
Circ=3833.
Harm=360.
CavV=100.e3
sigsS=math.asin(0.5)
cav=CAVITE(IOPT=2,L=Circ,h=Harm,V=cavV,sig_s=sigS)
```

REBELOTE

```
Nturns=1000
reb=REBELOTE(NPASS=Nturns,KWRIT=0.4,K=99)
```

Spin Tracking

```
st=SPNTRK()
st.set(KSO=3)
reb=REBELOTE(NPASS=Nturns,KWRIT=0.4,K=99)
```

Setup the Line

```
bstr=Line('Booster')
bstr.add(ob)
bstr.add(HELION())
bstr.add(FAISTORE(FNAME='zgoubi.fai',IP=1))
bstr.add(st)
bstr.add(ring)
bstr.add(cav)
bstr.add(reb)
bstr.add(END())
r=bstr.run()
r.save_fai('bstr.fai')
```

Notes

Cavite should be added at the end of the lattice but before end of input file items(like REBELOTE/END)

KWRIT tells zgoubi when to write to the console (0 prints first and final turn to .res file, the 4 prints every 1000 turns to the terminal)

K=99 uses the coordinates at the end of pass #n for start of pass #n+1