testscaling

July 10, 2023

[103]:
```
import at
from math import pi
import numpy as np
delta=0.01
scaling=1.0+delta
orbit0=np.zeros((6,))
orbit1=np.array([0.0, 0.0, 0.0, 0.0, delta, 0.0])
```

0.0.1 Simple FODO lattice

[104]:
```
dr0=at.Drift('dr0', 0.5)
qf0=at.Quadrupole('qf0', 1.0, 0.5)
qd0=at.Quadrupole('qd0', 1.0, -0.5)
dip0=at.Dipole('dip0', 1.0, 2*pi/128, 0)
# build the nominal lattice
fodo0=at.Lattice([dr0, dip0, dr0, qf0, dr0, dip0, dr0, qd0], energy=6.0E9,␣
 ↪name='fodo0')
# Compute optics
el0, rd0, _ = at.get_optics(fodo0)
```

0.0.2 Off-momentum test

To check the effect of field scaling, we build different scaled lattices and run them off-momentum. The optics should be restored and be identical to the nominal one.

1. Using scaled elements + B[0] trick (present behaviour of tapering in the master branch)

[105]:
```
qfa=at.Quadrupole('qfa', 1.0, 0.5*scaling)
qda=at.Quadrupole('qda', 1.0, -0.5*scaling)
dipa=at.Dipole('dipa', 1.0, 2*pi/128, 0, PolynomB=[delta*2*pi/128/1.0,0])
# lattice with scaled magnets
fodoa=at.Lattice([dr0, dipa, dr0, qfa, dr0, dipa, dr0, qda], energy=6.0E9,␣
 ↪name='fodoa')
# Compute off-momentum optics
ela, rda, _ = at.get_optics(fodoa, dp=delta)
```

2. Using the new `FieldScaling` attribute

[106]:
```
qfb=at.Quadrupole('qfa', 1.0, 0.5, FieldScaling=scaling)
qdb=at.Quadrupole('qda', 1.0, -0.5, FieldScaling=scaling)
dipb=at.Dipole('dipd', 1.0, 2*pi/128, 0, FieldScaling=scaling)
# lattice with with new attribute
fodob=at.Lattice([dr0, dipb, dr0, qfb, dr0, dipb, dr0, qdb], energy=6.0E9,␣
 ↪name='fodob')
# Compute off-momentum optics
elb, rdb, _ = at.get_optics(fodob, dp=delta)
```

3. Using the new "Reference momentum scaling" elements

[107]:
```
scin=at.Element('scin', PassMethod='ChangePRefPass', FieldScaling=scaling)
scout=at.Element('scout', PassMethod='ChangePRefPass', FieldScaling=1.0/scaling)
```

by changing globally the whole lattice

[108]:
```
fodoc=at.Lattice([scin, dr0, dip0, dr0, qf0, dr0, dip0, dr0, qd0, scout],␣
 ↪energy=6.0E9, name='fodoc')
# Compute off-momentum optics
elc, rdc, _ = at.get_optics(fodoc, dp=delta)
```

by changing change of reference in dipoles only and using scaled quadrupoles

[109]:
```
fodod=at.Lattice([dr0, scin, dip0, scout, dr0, qfa, dr0, scin, dip0, scout,␣
 ↪dr0, qda], energy=6.0E9, name='fodod')
# Compute off-momentum optics
eld, rdd, _ = at.get_optics(fodod, dp=delta)
```

Compare tunes: all optics should be identical

[110]:
```
print('               nominal:', rd0.tune)
print('       scaled elements:', rda.tune)
print('FieldScaling attribute:', rdb.tune)
print('      changed all refs:', rdc.tune)
print('     changed dip. refs:', rdd.tune)
```

```
               nominal: [0.23164082 0.22976227]
       scaled elements: [0.23162228 0.22976227]
FieldScaling attribute: [0.23164082 0.22976227]
      changed all refs: [0.23164082 0.22976227]
     changed dip. refs: [0.23164082 0.22976227]
```

The optics using scaled magnets and the B[0] trick is slightly different, because the horizontal intrinsic focusing of the dipoles is not taken into account.

Using the `FieldScaling` attribute restores perfectly the optics

Compare closed orbit

[111]:
```
print('          nominal:', el0.closed_orbit[:2])
print('          scaling:', ela.closed_orbit[:2])
```

```
print(' changed all refs:', elb.closed_orbit[:2])
print('changed dip. refs:', elc.closed_orbit[:2])
print('   new PassMethod:', eld.closed_orbit[:2])
```

```
          nominal: [0. 0.]
          scaling: [0. 0.]
 changed all refs: [0. 0.]
changed dip. refs: [0. 0.]
   new PassMethod: [0. 0.]
```

All is correct