#!/usr/bin/env python

import os

import numpy as np

from mtuq import read, open\_db, download\_greens\_tensors

from mtuq.event import Origin

from mtuq.graphics import plot\_data\_greens2, plot\_beachball, plot\_misfit\_dc

from mtuq.grid import DoubleCoupleGridRegular

from mtuq.grid\_search import grid\_search

from mtuq.misfit import Misfit

from mtuq.process\_data import ProcessData

from mtuq.util import fullpath, merge\_dicts, save\_json

from mtuq.util.cap import parse\_station\_codes, Trapezoid

if \_\_name\_\_=='\_\_main\_\_':

#

# Carries out grid search over 64,000 double couple moment tensors

#

# USAGE

# python SerialGridSearch.DoubleCouple.py

#

# A typical runtime is about 60 seconds. For faster results try

# GridSearch.DoubleCouple.py, which runs the same inversion in parallel

#

#

# We will investigate the source process of an Mw~4 earthquake using data

# from a regional seismic array

#

path\_data=fullpath('/home/aayushi/pysep/pysep/configs/mtuq\_workshop\_2022/2016-12-12T220444\_SOUTHERN\_XINJIANG\_CHINA/SAC/\*[ZRT].sac')

path\_weights= fullpath('/home/aayushi/pysep/pysep/configs/mtuq\_workshop\_2022/2016-12-12T220444\_SOUTHERN\_XINJIANG\_CHINA/weights.dat')

event\_id= '2016-12-12T220444\_SOUTHERN\_XINJIANG\_CHINA'

model= 'ak135'

#

# Body and surface wave measurements will be made separately

#

process\_bw = ProcessData(

filter\_type='Bandpass',

freq\_min= 0.1,

freq\_max= 0.333,

pick\_type='taup',

taup\_model=model,

window\_type='body\_wave',

window\_length=15.,

capuaf\_file=path\_weights,

)

process\_sw = ProcessData(

filter\_type='Bandpass',

freq\_min=0.025,

freq\_max=0.0625,

pick\_type='taup',

taup\_model=model,

window\_type='surface\_wave',

window\_length=150.,

capuaf\_file=path\_weights,

)

#

# For our objective function, we will use a sum of body and surface wave

# contributions

#

misfit\_bw = Misfit(

norm='L2',

time\_shift\_min=-2.,

time\_shift\_max=+2.,

time\_shift\_groups=['ZR'],

)

misfit\_sw = Misfit(

norm='L2',

time\_shift\_min=-10.,

time\_shift\_max=+10.,

time\_shift\_groups=['ZR','T'],

)

#

# User-supplied weights control how much each station contributes to the

# objective function

#

station\_id\_list = parse\_station\_codes(path\_weights)

#

# Next, we specify the moment tensor grid and source-time function

#

grid = DoubleCoupleGridRegular(

npts\_per\_axis=40,

magnitudes=[4.2])

wavelet = Trapezoid(

magnitude=4.2)

#

# Origin time and location will be fixed. For an example in which they

# vary, see examples/GridSearch.DoubleCouple+Magnitude+Depth.py

#

# See also Dataset.get\_origins(), which attempts to create Origin objects

# from waveform metadata

#

origin = Origin({

'time': '2016-12-12T22:04:44.13',

'latitude': 39.2856,

'longitude': 74.4344,

'depth\_in\_m': 10000,

})

#

# The main I/O work starts now

#

print('Reading data...\n')

data = read(path\_data, format='sac',

event\_id=event\_id,

station\_id\_list=station\_id\_list,

tags=['units:m', 'type:velocity'])

data.sort\_by\_distance()

stations = data.get\_stations()

print('Processing data...\n')

data\_bw = data.map(process\_bw)

data\_sw = data.map(process\_sw)

print('Reading Greens functions...\n')

greens = download\_greens\_tensors(stations, origin, model)

print('Processing Greens functions...\n')

greens.convolve(wavelet)

greens\_bw = greens.map(process\_bw)

greens\_sw = greens.map(process\_sw)

#

# The main computational work starts now

#

print('Evaluating body wave misfit...\n')

results\_bw = grid\_search(data\_bw, greens\_bw, misfit\_bw, origin, grid)

print('Evaluating surface wave misfit...\n')

results\_sw = grid\_search(data\_sw, greens\_sw, misfit\_sw, origin, grid)

results = results\_bw + results\_sw

#

# Collect information about best-fitting source

#

# index of best-fitting moment tensor

idx = results.source\_idxmin()

# MomentTensor object

best\_mt = grid.get(idx)

# dictionary of lune parameters

lune\_dict = grid.get\_dict(idx)

# dictionary of Mij parameters

mt\_dict = best\_mt.as\_dict()

merged\_dict = merge\_dicts(

mt\_dict, lune\_dict, {'M0': best\_mt.moment()},

{'Mw': best\_mt.magnitude()}, origin)

#

# Generate figures and save results

#

print('Generating figures...\n')

plot\_data\_greens2(event\_id+'DC\_waveforms.png',

data\_bw, data\_sw, greens\_bw, greens\_sw, process\_bw, process\_sw,

misfit\_bw, misfit\_sw, stations, origin, best\_mt, lune\_dict)

plot\_beachball(event\_id+'DC\_beachball.png',

best\_mt, stations, origin)

plot\_misfit\_dc(event\_id+'DC\_misfit.png', results)

print('Saving results...\n')

# save best-fitting source

save\_json(event\_id+'DC\_solution.json', merged\_dict)

# save misfit surface

results.save(event\_id+'DC\_misfit.nc')

print('\nFinished\n')