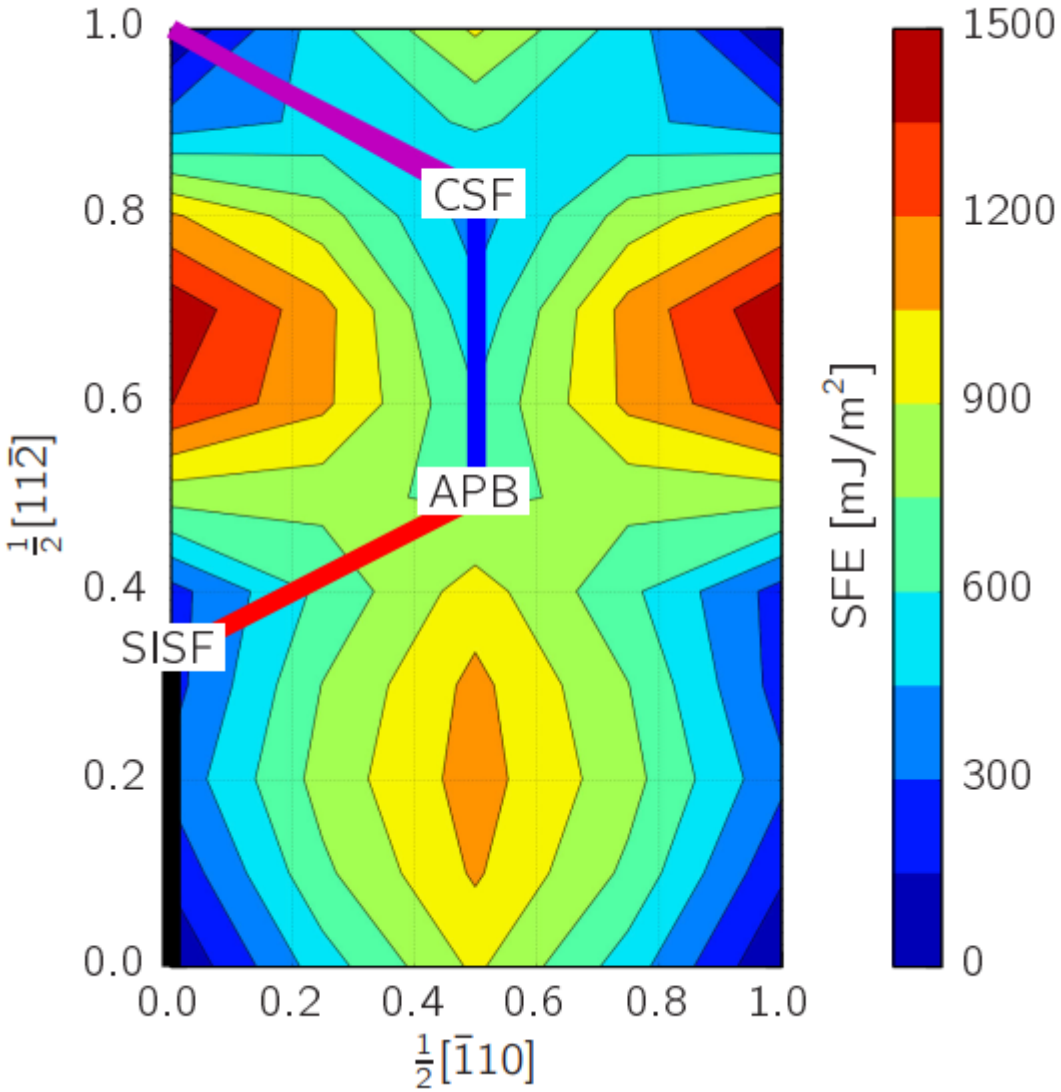
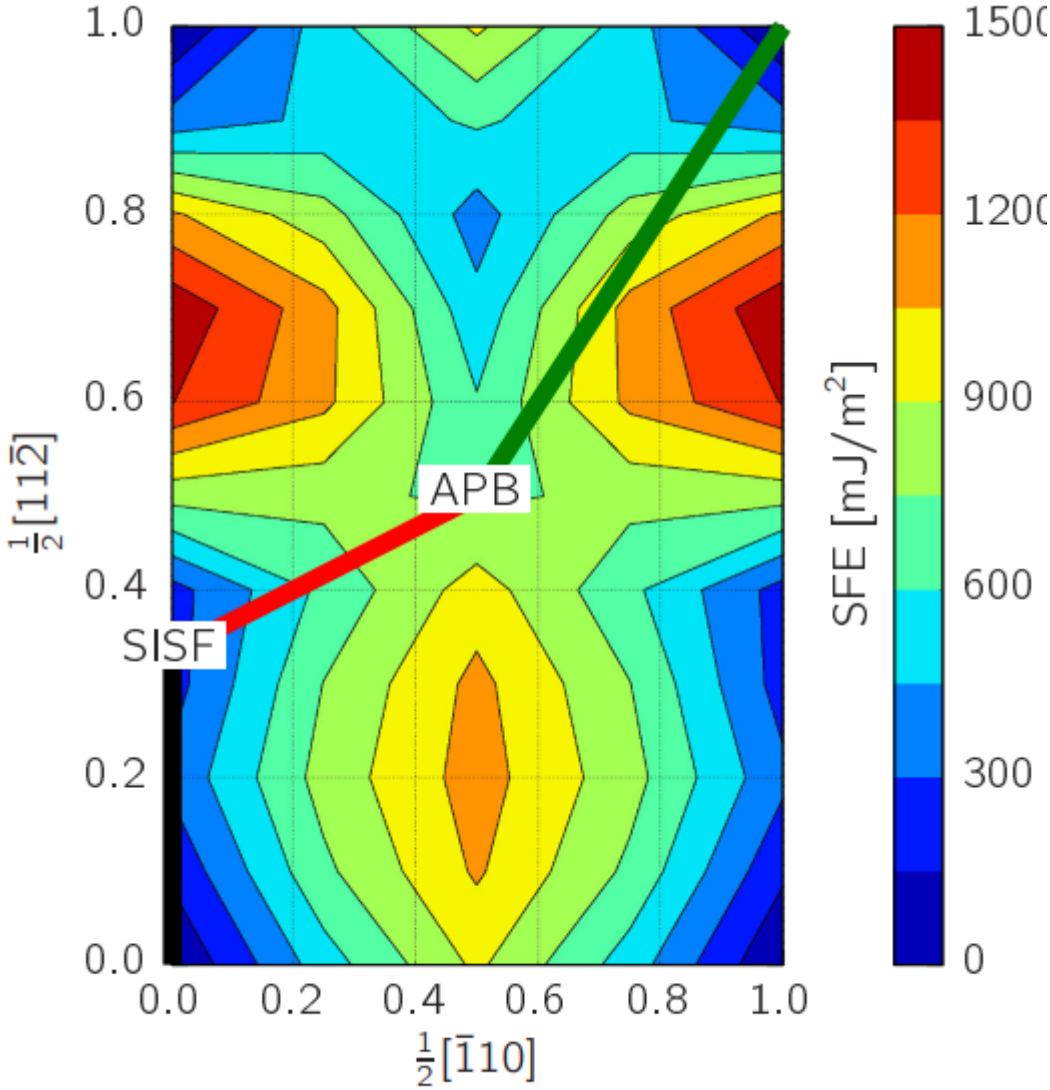
string法自动寻找二维势能面上的MEP

势能面扫描在催化，功能等领域常常会被用到，其中尤以二维势能面用量最多(废话，搞成三维的咋看)。在之前文章中,tamas分享了如何使用python自动扫描势能面(PES)，非常实用。不过我们好不容易绘制好势能面之后，寻找最小能量路径(MEP)往往是一个很头疼的问题。如果我们做的是NEB之类的话，最后得到的路径本身就是高维势能面上的MEP。尽管我们可以直接肉眼看出二维PES上的MEP，并且用PS等软件给他描上去。比如在(Dumitraschkewitz, Phillip, et al. "Impact of Alloying on Stacking Fault Energies in γ-TiAl." *Applied Sciences* 7.11 (2017): 1193.)这篇文章中，作者进行了γ -TiAl {111}面的广义层错能势能面扫描。





上下两幅图是同一个PES，代表不同的形变路径。从上图的MEP可以很快联想到，伴随位错分解，SISF转变为APB以后生成CSF是一种更优的形变机制。但是我们必须认识到，和很多文章一样，这种MEP往往是靠手直接加上，甚至直接用直接表示。对于直观的二维势能面来说，这很OK，完全不会影响文章结果。但是既然各种确定反应路径的方法如此好用，那为什么不干脆将这些手段用到二维PES上呢，就算多此一举，我们也不能将就。

结果我还没做就发现xin chen(https://github.com/chenxin199261/MEPSearcher#opennewwindow)三年前就做到了。好吧，既然都做了，那我又何必从零开始。我对xin chen的code使用python3重写，对核心模块进行了语句优化以增加以后的可扩展性，添加了CG算法,目前测试来说CG算法略微表现优异于原来的SD算法。同时我增加了draw.py提供快速查看收敛结果，增加了3个例子。经测试三次样条插值已经非常好使，所以我目前没有添加其他诸如B样条插值等手段。我将在以后有心情的时候添加NEB,dimer等算法。

本code使用string法(Weinan, E., Weiqing Ren, and Eric Vanden-Eijnden. "String method for the study of rare events." *Physical Review B* 66.5 (2002): 052301.)。我首先简要介绍一下这个方法。string法是一种跟NEB同属于chain of states方法的优秀的过渡态搜索算法。我们可以一句话概括这个算法，我们如果将一串很密的珠子放在PES上，让这些珠子自由朝能谷跑，最后稳定的状态就是MEP。它的本质公式如下

$$\varphi_t = -[\nabla V(\varphi)]^{\perp} + r\hat{t}.$$

前项表示扣除平行于串珠分量的切线力，后项是一个约束用的拉格朗日乘子，在本code中，采用珠子均匀化作为一种最简单的约束。

下面我讲一下怎么使用该code.首先你需要安装ALGlib库，安装很简单，自己百度一下。然后你需要准备PES数据PES.data. 我这里用的都是example里面的数据。

```
 1 0.0     0.0     -0.8390715290764524
 2 0.060606060606606061    0.0     -0.8375310061591744
 3 0.121212121212122    0.0     -0.832915093496508
 4 0.181818181818182    0.0     -0.8252407052219515
 5 0.242424242424243    0.0     -0.8145354958740859
 6 0.303030303030304    0.0     -0.8008349676384359
 7 0.363636363636365    0.0     -0.7841719417458072
 8 0.424242424242425    0.0     -0.7645488782097121
 9 0.484848484848486    0.0     -0.7418811509758331
10 0.5454545454545454    0.0     -0.7159024468855784
11 0.6060606060606061    0.0     -0.6860343886705959
12 0.6666666666666667    0.0     -0.6512407202005478
13 0.7272727272727273    0.0     -0.6099074624219966
14 0.7878787878787878    0.0     -0.5598066349402844
15 0.8484848484848485    0.0     -0.49820376909388425
```

第一栏为x轴，第二栏为y轴，第三栏为势能值。

然后你需要准备guess.data表示你建立的初猜路径。

```
File  Edit  View  Search  Terminal  Help
 1 41
 2 3 3
 3 2 1
 4
```

第一行表示珠子个数，一般30到40个为宜。第二行为起点的x,y值，第三行为终点的x,y值。这个需要你对着势能面确定一下，不过不需要很精确，只要确保它们不跑到其他能谷就可以。你如果测试过就会发现string的稳定性有多强大。程序会根据你的参数建立一条线性插值的初始路径。

随后我们需要在MEPsearcher.py中修改某些参数。

```
33 ##### SETTINGS ######
34 Max_iter = 100000      #maximum iretation steps
35 Tol = 1e-5           #converge tolerance
36 h = 0.05           #step size, 0.5 is okay for most case
37 o = 'sd'             #'cg' or 'sd'
```

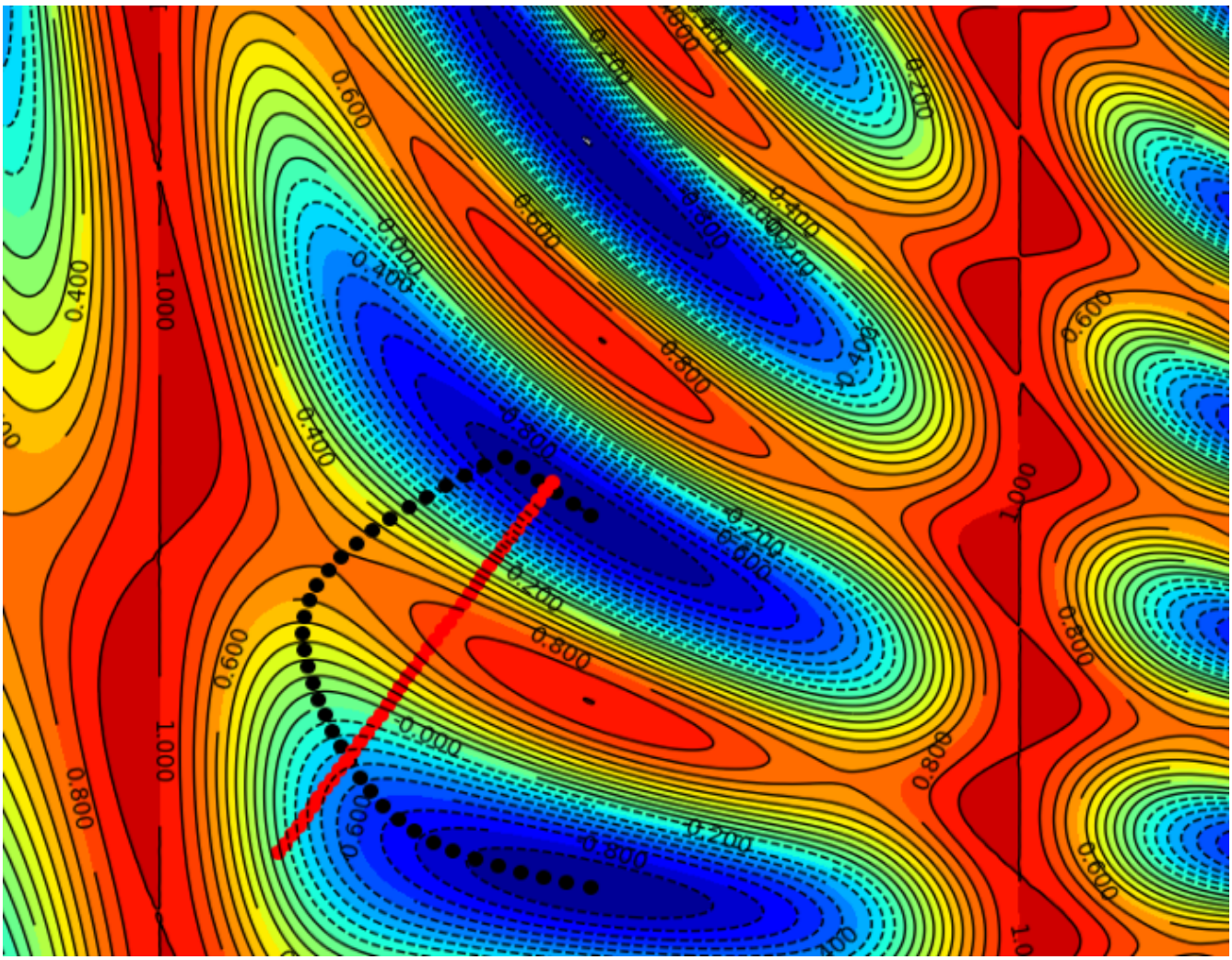
这里主要说一下h。h表示每次位移的步长，这个你用默认也行。o表示使用sd或者cg算法，默认使用sd算法。基本上这四个参数你其实是不用改的。如果你觉的收敛不行，你就试试cg算法吧。

```
draw.py  example.py  iio.py          optimize.py  redistribute.py
example  guess.data  MEPSearcher.py  PES.data
```

现在将两个输入文件放到MEPSearcher所在的主文件夹，run it。

```
cjchen@cjchen:~/test/MEPsearcherv2.0_alpha$ ./MEPSearcher.py
STEP:   30 ;    diff  =  0.174691491555139
STEP:   60 ;    diff  =  0.044279868551973474
STEP:   90 ;    diff  =  0.013698186512046542
STEP:   120 ;    diff  =  0.004594220856770038
STEP:   150 ;    diff  =  0.0017473055892967349
STEP:   180 ;    diff  =  0.0007436885744482607
STEP:   210 ;    diff  =  0.00033422318436236335
STEP:   240 ;    diff  =  0.00015158993316333365
STEP:   270 ;    diff  =  6.829576665367205e-05
STEP:   300 ;    diff  =  3.0549469536151355e-05
STEP:   330 ;    diff  =  1.3596919807946428e-05
Successful !
Final steps =  342
```

程序每隔30步发送一次消息。成功后会告诉你收敛的总步数。这里面diff表示当前收敛差值。产生的MEP路径会保存在out.data中，它的格式和PES.data是一样的。我们可以运行draw.py观察结果。



红线表示初始路径，黑线表示MEP。效果还是很不错的。

我们也可以发现，string相对于NEB强大的优势在于，它的初态和末态并不是固定的，这就方便我们搭建初猜路径。但是这个优势仅限于二维的情况，道理我就不多说了。

写博文的这天我才知道，pymatgen也集成了类似的画MEP的功能，还是3D，还是动画，这就很尴尬了。无论如何string法还是更准确的嘛。我也只是把这个当作一个模型，以后可以简单地用来测试各种过渡态搜索算法。

ps：最近收到很多之前我发的脚本的询问和反馈，我会尽快回复，并不是没有看到哈。同时本文之后我就要闭关写文章了，下次更新应该比较后面了。

ponychen

20190606