廣東工業大學

QG 中期考核详细报告书

题　　目	QG 中期考核--模型复现
学　　院	信息工程学院
专　　业	电子信息类
年级班别	4 班
学　　号	3222002213
学生姓名	陈羽彤

2023 年 4 月 7 日

目录

一．阅读文献

没有什么看全英论文的经验，从头看到第二部分”THE DISCRETE-AGENT MODEL”对于我来说已经很吃力了，看了两天，还有很多公式或者推导有点迷迷糊糊的，但是所幸有边阅读边批注的习惯，之后再翻阅便没那么吃力；还有就是学了好多单词。

二．模型复现

且不论模型复现的成败，先说一下我模型复现的顺序，Fig5、Fig3、Fig4、Fig2

1. Fig5

①test01

(1)数据集：

```
#创建数据集
#data_1 ndarray(251,) 均匀分布在[0,2.5]之间的251个数
#data_2 ndarray(500,) 均匀分布在[2.5,3]之间的500个数
#data   ndarray(751,)

data_1 = np.linspace(0,2.5,251)
data_2 = np.linspace(2.5,3,500)
data = np.concatenate((data_1,data_2),axis=0)
data.shape
```

(2) 智能体更新函数：

```
def opinion_update(X_t):
    """
    输入:
    X_t         ndarray(n,)
    size        scaler

    输出:
    X_          ndarray(n,)
    """
    m = X_t.shape[0]
    X_ = np.zeros(m)

    for i in range(m):
        neighbor = 0
        times = 0
        for j in range(m):
             if(np.abs(X_t[i] - X_t[j]) < 1):
                neighbor += X_t[j]
                times += 1
        X_[i] = neighbor/times
    return X_
```

$$x_i(t+1) = \frac{\sum_{j:|x_i(t)-x_j(t)|<1} x_j(t)}{\sum_{j:|x_i(t)-x_j(t)|<1} 1}.$$

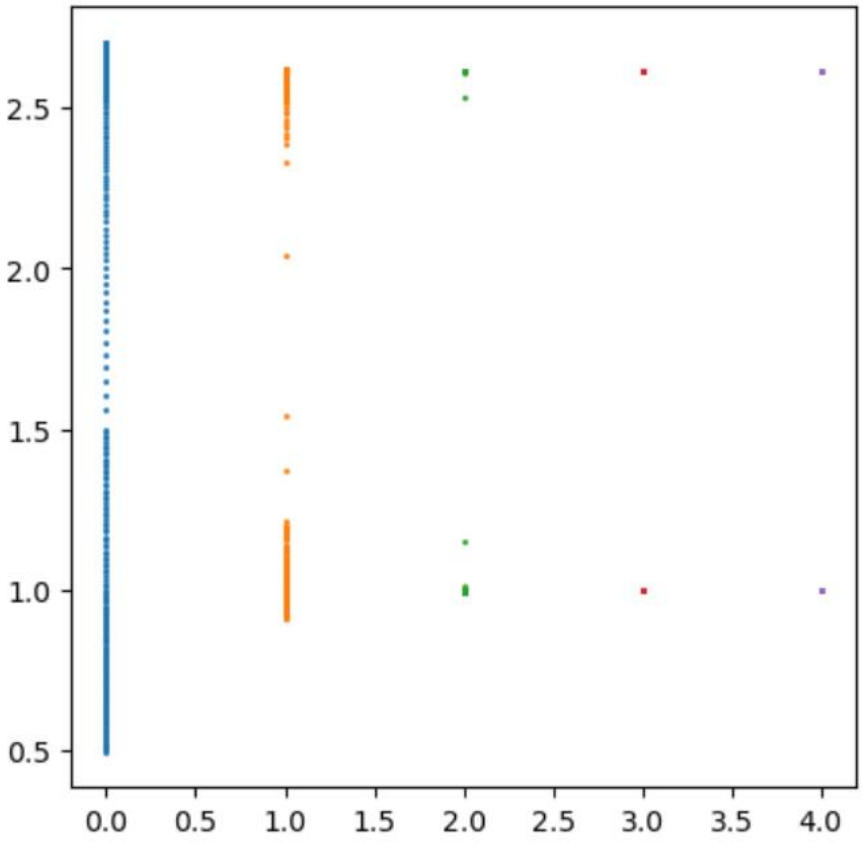
手写上述函数的依据：

(3) 主程序：

每次迭代通过更新并覆盖上一次的数据、在图中画出点，来测试思路和逻辑是否可行

```
plt.figure(figsize=(5,5))
for i in range(5):
    y_ = y_1 + i
    X_function1 = opinion_update(X_function1)
    plt.scatter(y_,X_function1,s=1)
```

(4)画出的图



发现此图的收敛点的数值在 2.7 和 1.0 附近，与论文中 Fig5 的图示的收敛值大致相同，于是确定了继续在这个基础上，针对作图进行了 test02

②test02

数据集、智能体更新函数同 test01

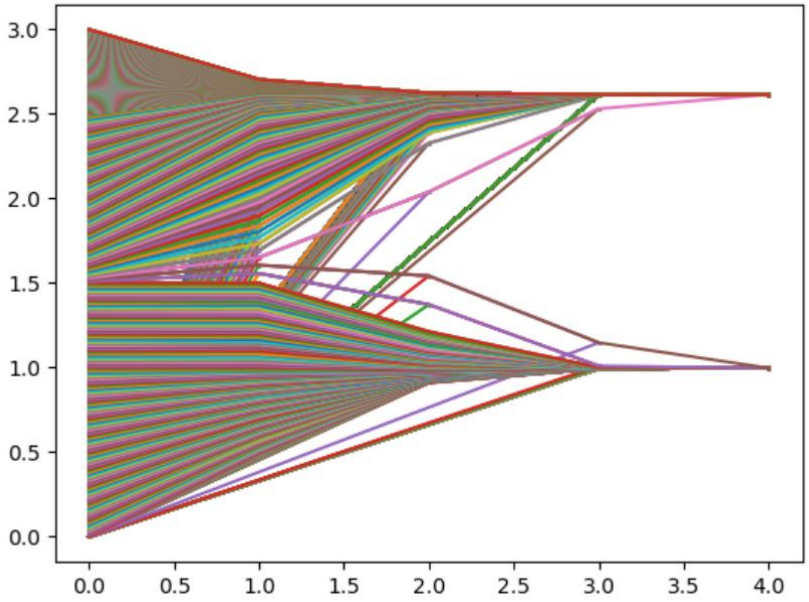
(1) 主函数中在 test01 基础上更改了智能体状态更新后对数据的处理

```
for i in range(1,6):
    y_0[:,i] = y_0[:,i] + i
    k = i-1
    for j in range(751):
        X_fuc[:,i] = opinion_update(X_fuc[:,i-1], 751)
        plt.scatter(y_0[j][i],X_fuc[j][i],s=1)
        plt.plot(y_0[j],X_fuc[j])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_25404\3176829691.py in <module>
      1 for i in range(1,6):
----> 2     y_0[:,i] = y_0[:,i] + i
      3     k = i-1
      4     for j in range(751):
      5         X_fuc[:,i] = opinion_update(X_fuc[:,i-1], 751)

IndexError: index 5 is out of bounds for axis 1 with size 5
```

```
IndexError: index 5 is out of bounds for axis 1 with size 5
```



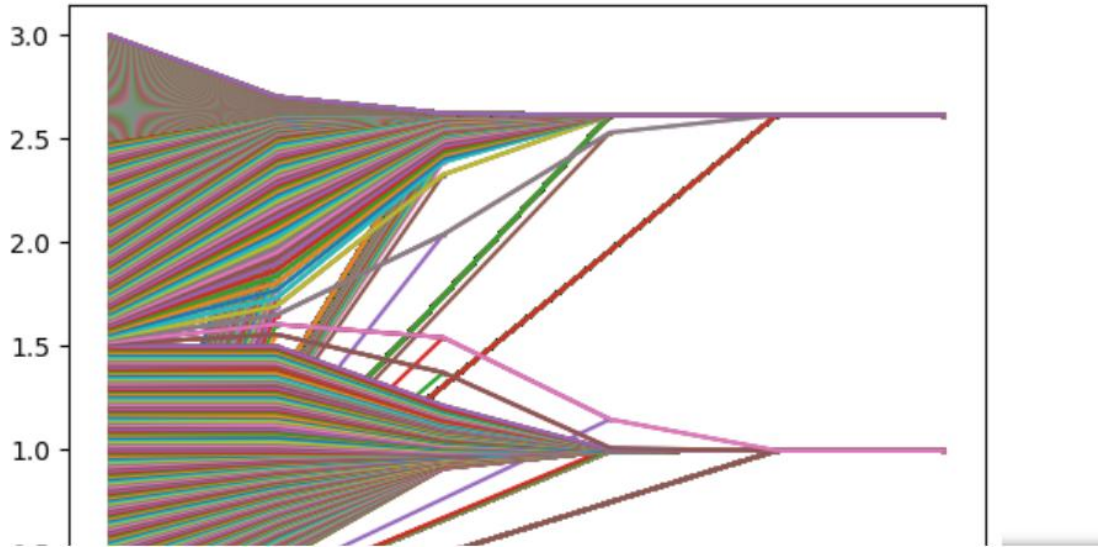
虽然跑完了才知道它报错了...
可见 test02 是不成功的，但是初现雏形，报错是因为没有设置好边界

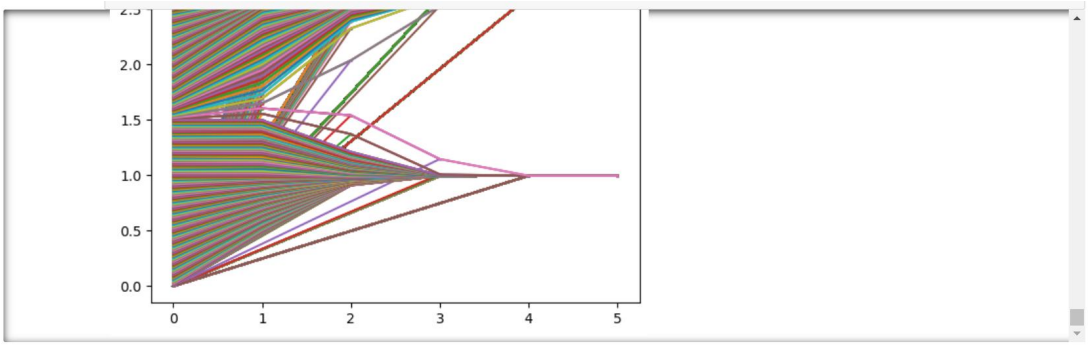
③done(最后版本)

(1) 主程序更改部分

```
for i in range(1,6):
    y_0[:,i] = y_0[:,i] + i
    for j in range(751):
        print(i,j)
        X_fuc[:,i] = opinion_update(X_fuc[:,i-1], 751)
        plt.scatter(y_0[j][i],X_fuc[j][i],s=1)
        plt.plot(y_0[j],X_fuc[j])
```

放两张图是因为它把结果放在滑动条下了.....一次截不完

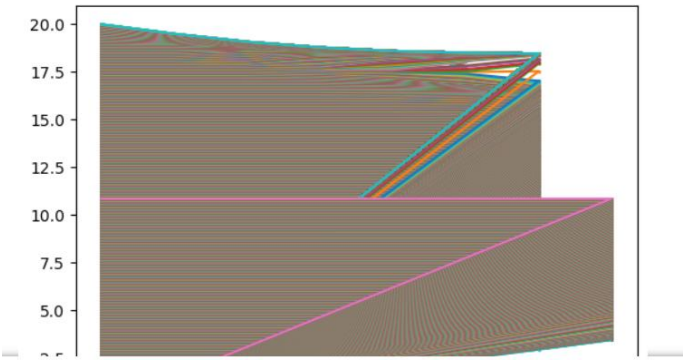


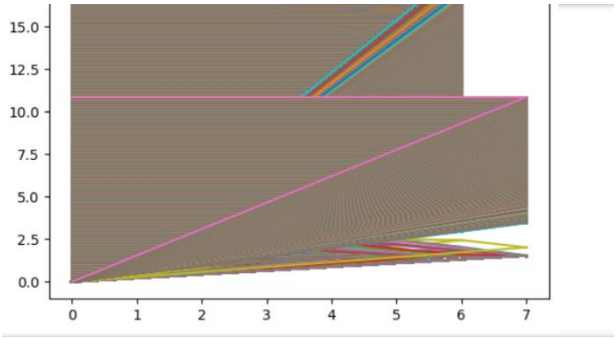


虽说是有点样子，但是还是不知道为什么中间有一堆“错乱”的数字...

2. Fig3

...跑了快一天了，画出来...





也有可能是设置数据的时候出问题了，2000 个智能体迭代 15 次....

3. Fig4

①test01

(1)数据集：

```
#创建数据集

data_1 = np.linspace(0,1,300)
data_2 = np.linspace(3,4,300)
data_3 = np.linspace(1,3,50)
data = np.concatenate((data_1,data_3,data_2),axis=0)
```

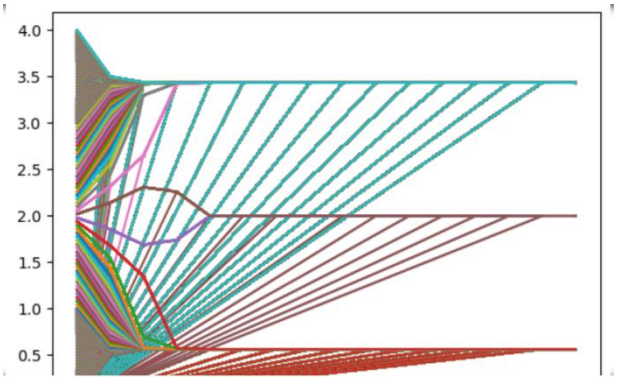
(2) 智能体更新函数:同 Fig5

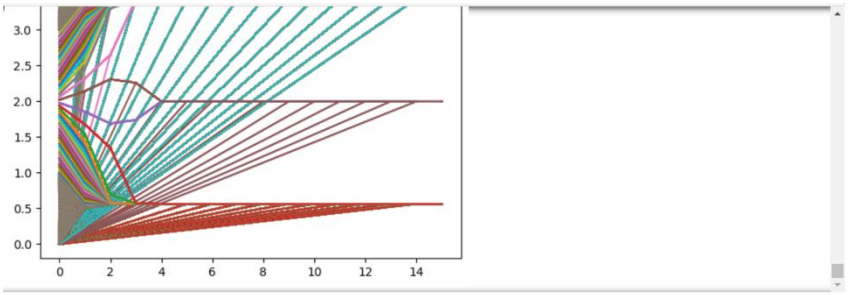
(3) 主函数：

```
for i in range(1,16):
    y_0[:,i] = y_0[:,i] + i
    for j in range(650):
        print(i,j)
        X_fuc[:,i] = opinion_update(X_fuc[:,i-1], 650)
        plt.scatter(y_0[j][i],X_fuc[j][i],s=1)
        plt.plot(y_0[j],X_fuc[j])
```

与 Fig5 大致相同，把范围和迭代次数更改了一下

(4) 画出来的图





肉眼可见的奇怪....有很多异常的数值...以为是数据集设置的不好，于是有了 test02

②test02

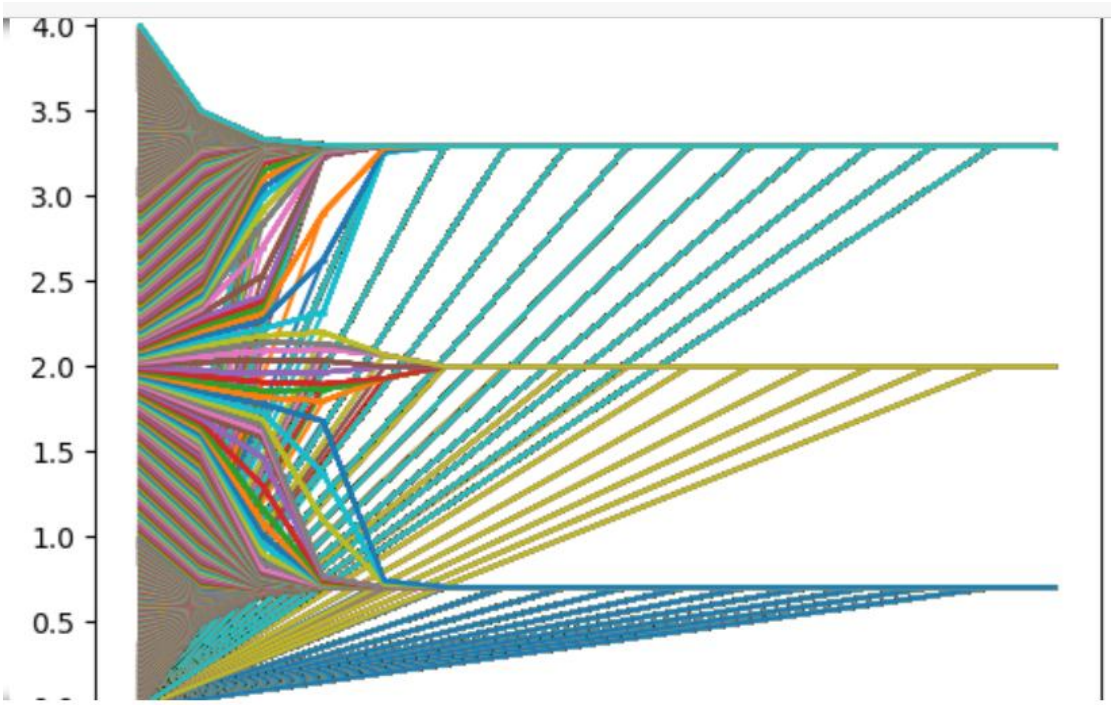
在 test01 的基础上仅对数据集做出改动

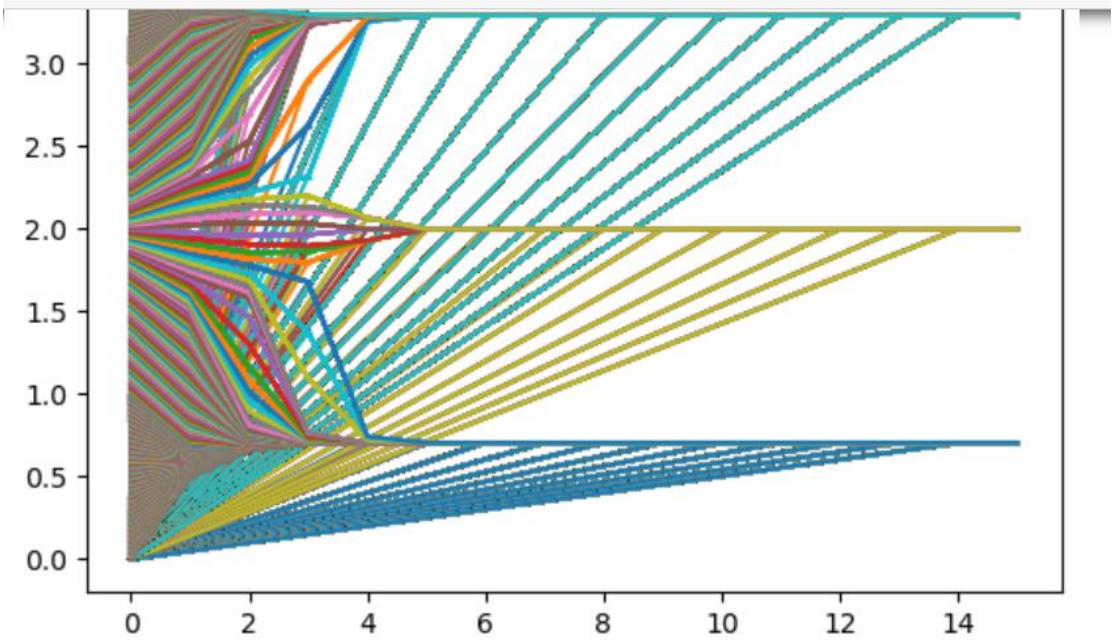
```
#创建数据集

data_1 = np.linspace(0,1,300)
data_2 = np.linspace(3,4,300)
data_3 = np.linspace(1,3,150)
data = np.concatenate((data_1,data_3,data_2),axis=0)
```

把中间零星的智能体的比例加大

画出来的图：





总感觉是有一组数据与部分数据格格不入.....可惜已没有再多的时间去 debug 了....

4.Fig2

①test01

(1) 函数

1)创建数据集函数

```
def creat_agents(L):
    """
    输入:
    L scaler 范围

    输出:
    X nparray(n,)
    """
    X = np.linspace(0,L,100*L)
    return X
```

2)单个智能体更新函数(同 Fig5)

```
def opinion_update(X_t):
    """
    输入:
    X_t         ndarray(n,)

    输出:
    X_          ndarray(n,)
    """
    m = X_t.shape[0]
    X_ = np.zeros(m)

    for i in range(m):
        neighbor = 0
        times = 0
        for j in range(m):
             if(np.abs(X_t[i] - X_t[j]) < 1):
                neighbor += X_t[j]
                times += 1
        X_[i] = neighbor/times
    return X_
```

3)找收敛点函数

```
def get_clusters(L,X):
    """
    输入:
    L scaler
    X ndarray(n,)
    输出:
    L_result ndarray
    """
    m = X.shape[0]
    L_all =[]
    L_set = set()
    L_result = []
    index_num = L + 1
    for i in range(index_num):
        index = (int)(m/L)*i - 1
        L_all.append(X[index])
    for num in L_all:
        if num not in L_result:
            L_set.add(num)
            L_result.append(num)
    L_result = np.array(L_result)

    return L_result
```

4)多个智能体更新并作图函数

```
def update_and_plot(X, iteration):
    m = X.shape[0]
    y_0 = np.zeros(m)
    plt.figure(figsize=(10,10))
    plt.scatter(y_0,X,s=1)
    for i in range(iteration):
        y = y_0 + i
        X = opinion_update(X)
        plt.scatter(y,X,s=1)
    return X
```

5)得到最终收敛点并作图函数

```
def get_and_draw(L,X_clusters):
    m = X_clusters.shape[0]
    distance = np.zeros(m)
    distance = distance + L/2
    y = X_clusters - distance
    x = np.zeros(m)
    x = x + L
    for i in range(m):
        plt.scatter(x[i],y[i],s=1)
    return y
```

(2) 思路

1) 先按 L 的大小创建数据集，论文中是 5000L 个智能体，但是大概率是跑不完的，于是我设置的是 100L 智能体

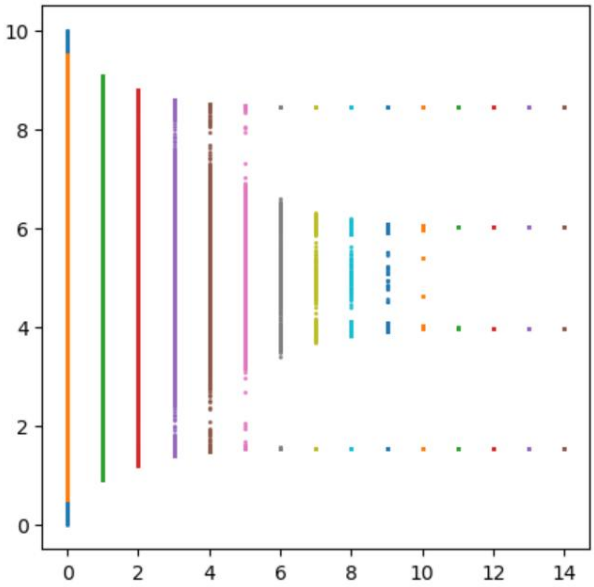
2) 智能体更新完毕后，接下来就是要找 clusters 了，论文中提到(下图)，根据 L 的大小可以得出在有限时间内收敛后有[L]+1 个簇团，于是设置取最后一次迭代的[L]+1 个平均分布在数据集中的值，按理来说应该是不会有对 clusters 的缺漏的(吧)

Theorem 1 states that opinions converge to clusters separated by at least 1. Since the smallest and largest opinions are nondecreasing and nonincreasing, respectively, it follows that opinions initially confined to an interval of length L can converge to at most $\lceil L \rceil + 1$ clusters. It has however been observed in the literature that the distances between clusters are usually significantly larger than 1 (see [21], [24], and Fig. 1), resulting in a number of clusters that is significantly smaller than the upper bound of $\lceil L \rceil + 1$. To further study this phenomenon, we analyze below different experimental results, similar to those in [24].

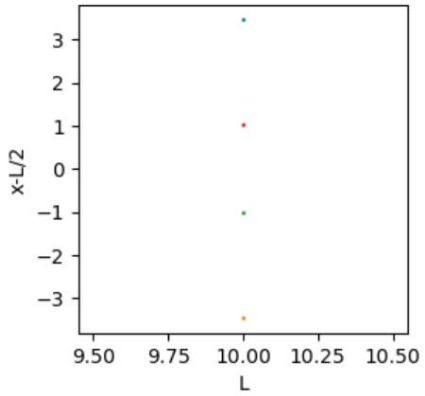
3) 然后再对簇团中重复的值进行处理，得出每个 L 下对应 clusters 距离 L/2 的值

(3) 画出来的图

以 L=10、迭代 15 次为例子



上面是粗略的收敛分布图

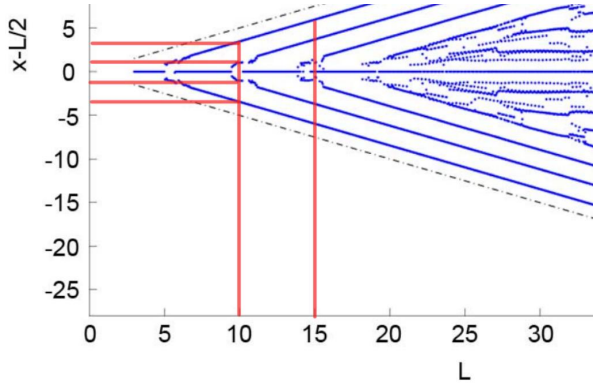


上面是仅在 L=10 下得出的点位

下面是 x-L/2 的具体数值

```
y
```

```
array([ 3.45858825, -3.45858825, -1.0271363 ,  1.0271363 ])
```



感觉是跟论文中的大致相同的

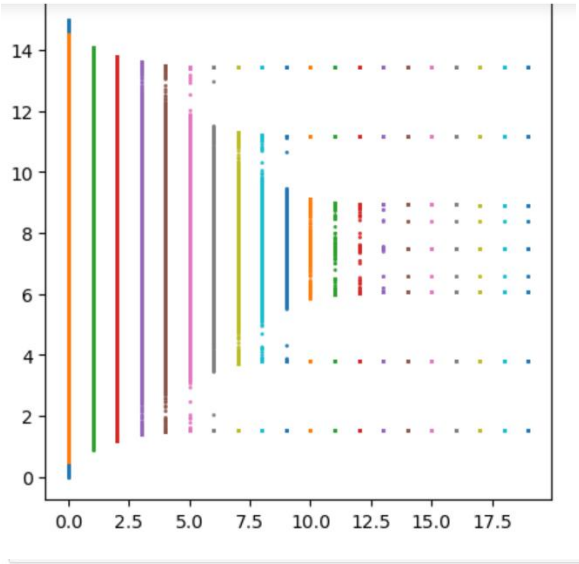
②test02

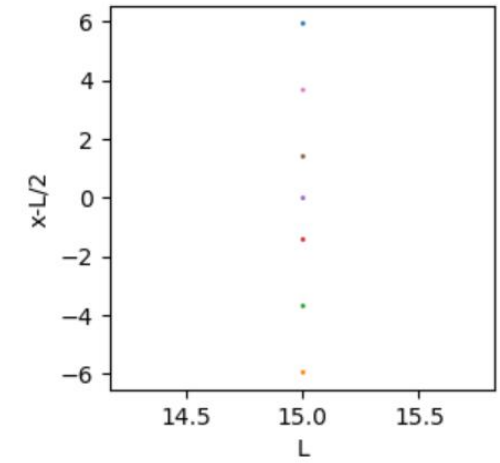
本想着在 test01 基础上写一个总的函数囊括 tes01 中的功能，但是太多报错了...以失败告终

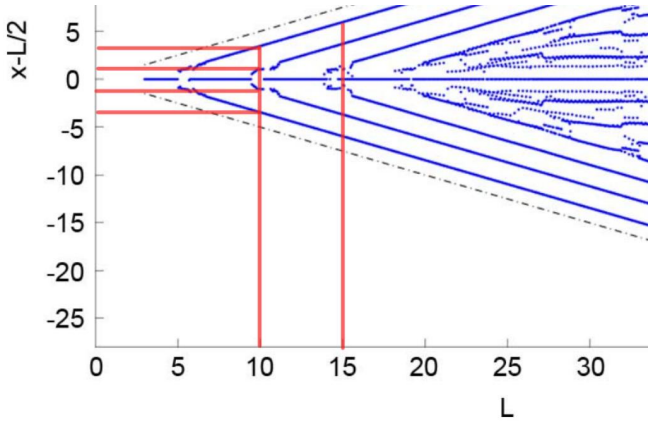
```
def function(creat_func, opinion_update_func, update_func, get_cluster_func, get_and_draw_func, L_max, iteration):
    for L in range(1,L_max):
        X = creat_func(L)
        X_ = update_func(X,L)
        X_get_clusters = get_cluster_func(L,X_)
        get_and_draw_func(L,X_get_clusters)
```

③test03

Fig2 是我觉得相对来说更难的一个复现，放在最后做，也没有时间再去 debug 了，于是再设置了一个 L 去测试了一次，直接放图了(L=15,迭代次数为 20)







从 Fig2 图示来看是大致相同的(吧)

```
y
```

```
array([ 5.95259676, -5.95259676, -3.68262321, -1.41387694,  0.        ,
        1.41387694,  3.68262321])
```