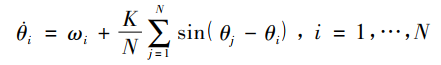
Kuramoto模型在Python和MATLAB中的简单实现

事情是这样的，我最近在研究团队编组及内部模式对发挥团队能力的影响，以及如何正确编组让团队能力发挥实现最大化，别问我为什么研究这个，反正稀里糊涂就研究上了。我发现在描述团队编组间及内部同步能力的时候，人们对Kuramoto模型（藏本模型）作了大量的研究,其中包括模型达到完全相位同步的充分条件、耦合强度对于同步的影响、一定条件下振子的收敛速率等。但具体实现一般都在MATLAB中，且网上代码过于复杂（我运行了一遍一堆报错），这里我使用Python和MATLAB对Kuramoto模型简单模拟。

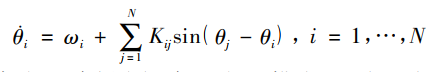
模拟的话还是一遍举个栗子，边分析边测试效果最好，百度学术上有一篇关于Kuramoto模型的简单论文，我们就用它来实现模拟。空间信息支援力量编组模式分析，上连接: <http://xueshu.baidu.com/usercenter/paper/show?paperid=11b7c2ef69b3ac7f6e114afeb75f8083&site=xueshu_se>

好吧，那我们开始，首先是Python实现：

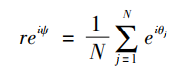
N维Kuramoto模型的数学描述如下：

（1）

没错，是讨厌的数学公式，没事，它可以改写成这样：

（2）

好像还是有点长，那我们在改写一下：

（3）

看着好多了，那我就来说说式子中参数的意义，Kij为耦合矩阵，是为了便于描述不同振子间耦合程度不同的情形。最下面那个式子的r就是我们的目标，反应振子间的相关性，这个相关性就可以描述我们想要的编组内部同步能力。

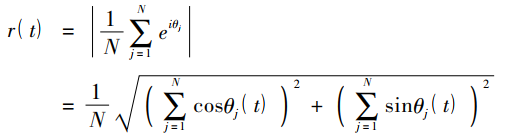
哎呦，这个式子看起来好简单，这里要补充一下知识点：同步能力可不是一下子各组该怎么同步直接确定的了，它是一个从开始到稳定的阶段，也就是说随时间变化，最终反映在各组的同步能力才会确定，那么最后图像是什么样子才算同步能力好呢？

同步能力好，是指随着时间的推移，各组的同步能力r逐渐稳定，波动现象消失或固定在某一个小范围内。需要注意的是这和各组r值之间的差距没有关系，我们要的是一个平稳的状态。那怎么办找r和t的关系呢？

注意看最上面那两个式子，相位（第一项，等号左边那个）上面有个点，这样他可就不简简单单是个相位了，它代表的是相位的变化值，是一个微小的微分值，好吧具体意思就是，那个式子左边展开之后是这样的：

（4）

word公式编辑器编出来的还凑合，我们把r单提出来，再把那个指数消掉，方便求解：

（5）

哎呀，t出现了，其实与t有关，这里你可能有点绕，因为它们之间的关系是一一对应的，就是说每个时间的t对应了一个，我下面带入具体数值的时候你就知道了。

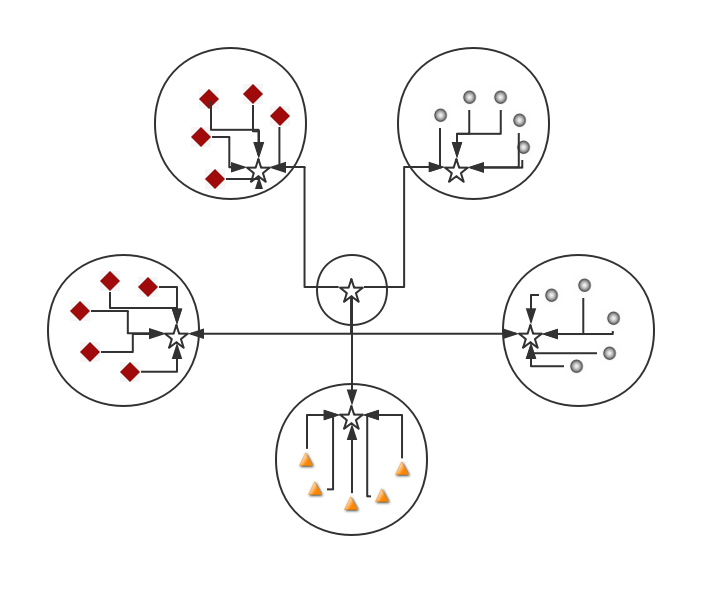
组间同步能力与时间t的关系出现了！

也就是说我先用上面的那个公式4计算出来的值，在带入到公式5，那么t-r关系就可以明确下来了，那现在我们再回过头来看看文章中已经给的例子，看看还有没有未知量。

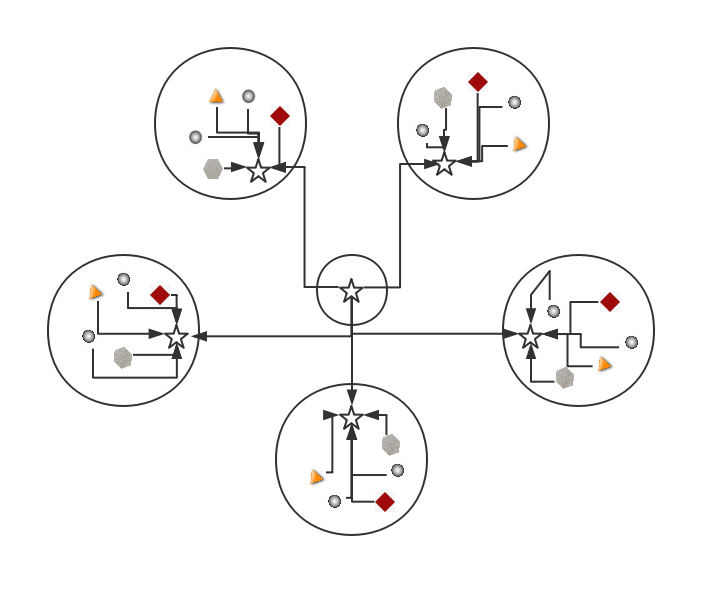
栗子是这样的栗子：

假设某机构内部有 4 个编组，每个编组包括 5 个节点( 其中 1 个节点为领导节点) 。另外，将上级领导作为一个独立的编组，且只包含一个节点。假设 在领导机关增加4名信息传递人员。当以独立编组模式编组时，指定1名信息传递人员为指挥者， 其指挥关系与其他编组一致; 当分散编组时，信息传递力量节点的关系与所在编组其他节点指挥关系一 致。其中，完全分散编组模式时，各信息传递力量节点之间无信息共享通道; 不完全分散编组时，在各信息传递人员节点之间建立一条信息共享通道。各编组模式及其拓扑结构如下图所示。

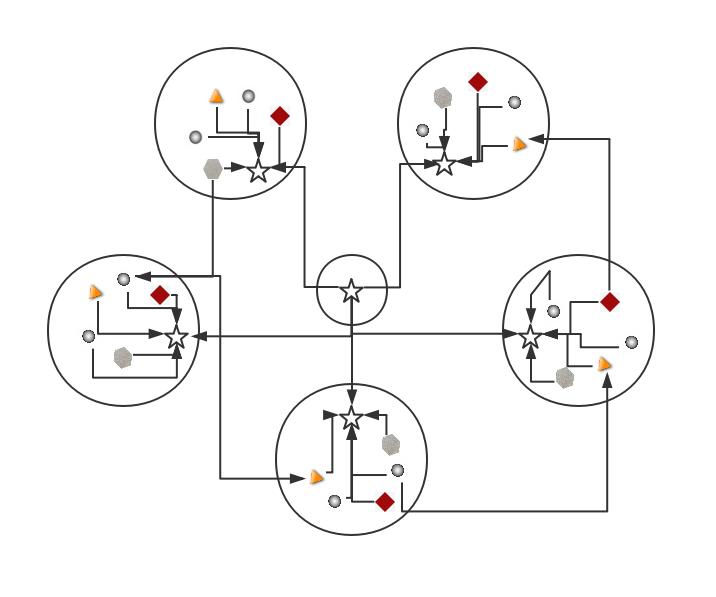
独立编组模式：



分散编组模式：



不完全分散编组模式：



参数数据：



参数确定一下有没有未知量：

首先N，数据数目已知，这个有了。

K值是分组内的连接强度，这个是看实际情况，由甲方提供或者自己看着给的，这里就是甲方给的编组图，i与j点的链接强度一目了然，这个有了。

是振子i的固有频率，也称自然频率，甲方会给，没法自己估计，这个有了。

，怎么办，初始的会给，自己也能测的出来，但那么多得多少不知道啊，这里通过翻看文章，我发现其实文章是有一个特殊条件的，不然的话是需要研究耦合因子求三种约束条件解情况的，特殊条件就在这：

假设编组内节点的初始相位差为π/2，且编号最小者为0，随编号增大而增大。

哦，初始相位差知道了，你还告诉了我各个初始相位，那么的值就在一个范围内的几个固定值里面啊！

好的，没有未知量了，就是找K的时候麻烦点，没办法，这个决定了编组的不同，写脚本算一下吧：

1. #codingutf-8
2. ##ScriptName:KuramotoSimulation.py
3. **import** matplotlib.pyplot as plt
4. **from** pylab **import**
5. **from** sympy **import**
6. **from** matplotlib.ticker **import** MultipleLocator, FormatStrFormatter
7. **import** math
8. **import** numpy as np
10. N = 31      #总节点数
12. c=[0,0,math.pi  2,math.pi,3  math.pi 2,0,0,0,math.pi  2,math.pi,3  math.pi 2,0,0,0,math.pi  2,math.pi,3  math.pi 2,0,0,0,math.pi  2,math.pi,3  math.pi 2,0,0,0,math.pi  2,math.pi,3  math.pi 2,0,0]
13. w = [4,3,3,2,2,1,4,3,3,2,2,1,4,3,3,2,2,1,4,3,3,2,2,1,4,3,3,2,2,1,5]
15. k1 = [
16. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
17. [1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2],
18. [1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
19. [1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
20. [1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
21. [1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
22. [1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
23. [0,0,0,0,0,0,1,0.9,0.9,0.9,0.9,0.9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2],
24. [0,0,0,0,0,0,0.9,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
25. [0,0,0,0,0,0,0.9,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
26. [0,0,0,0,0,0,0.9,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
27. [0,0,0,0,0,0,0.9,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
28. [0,0,0,0,0,0,0.9,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
29. [0,0,0,0,0,0,0,0,0,0,0,0,1,0.8,0.8,0.8,0.8,0.8,0,0,0,0,0,0,0,0,0,0,0,0,2],
30. [0,0,0,0,0,0,0,0,0,0,0,0,0.8,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
31. [0,0,0,0,0,0,0,0,0,0,0,0,0.8,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
32. [0,0,0,0,0,0,0,0,0,0,0,0,0.8,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
33. [0,0,0,0,0,0,0,0,0,0,0,0,0.8,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
34. [0,0,0,0,0,0,0,0,0,0,0,0,0.8,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
35. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0.7,0.7,0.7,0.7,0.7,0,0,0,0,0,0,2],
36. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.7,1,0,0,0,0,0,0,0,0,0,0,0],
37. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.7,0,1,0,0,0,0,0,0,0,0,0,0],
38. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.7,0,0,1,0,0,0,0,0,0,0,0,0],
39. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.7,0,0,0,1,0,0,0,0,0,0,0,0],
40. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.7,0,0,0,0,1,0,0,0,0,0,0,0],
41. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0.6,0.6,0.6,0.6,0.6,2],
42. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.6,1,0,0,0,0,0],
43. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.6,0,1,0,0,0,0],
44. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.6,0,0,1,0,0,0],
45. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.6,0,0,0,1,0,0],
46. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.6,0,0,0,0,1,0],
47. [2,0,0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,0,1]]

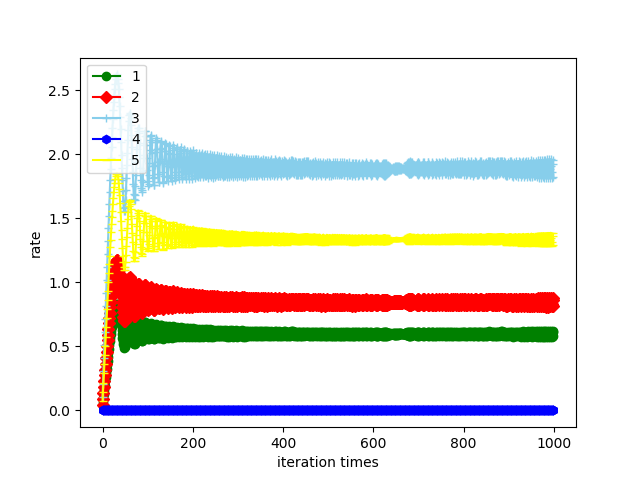
50. n = [i + 1 **for** i **in** range(22)]                #目标划分,24个值,1-24
51. t = [j  **for** j **in** range(1000)]
52. ci = 0

55. C = []
56. C.append(c)
58. **for** d **in** range(1100)
59. **for** i **in** n
60. **for** j **in** range(22)
61. cj = c[j + 1] - c[i]
62. ci += k1[i][j + 1]  math.sin(cj)
64. cii = ci + w[i]
65. h = [0.01  cii + z **for** z **in** c]
66. C.append(h)
67. c = h
69. **def** r\_function(u)
70. y1 = 0
71. y2 = 0
72. y12 = 0
73. y22 = 0
74. r = []
75. **for** x **in** range(1000)
76. **for** ul **in** u
77. y1 += math.cos(C[x][ul])
78. y2 += math.sin(C[x][ul])
79. y12 = y1  2
80. y22 = y2  2
81. r.append((float(1)  N )  ((y12 + y22)  0.5))
82. **return** r

85. r1 = r\_function([1,2,3,4,5,6])
86. r2 = r\_function([7,8,9,10,11,12])
87. r3 = r\_function([13,14,15,16,17,18])
88. r4 = r\_function([19,20,21,22,23,24])
89. r5 = r\_function([25,26,27,28,29,30])
90. #r6 = r\_function([31])

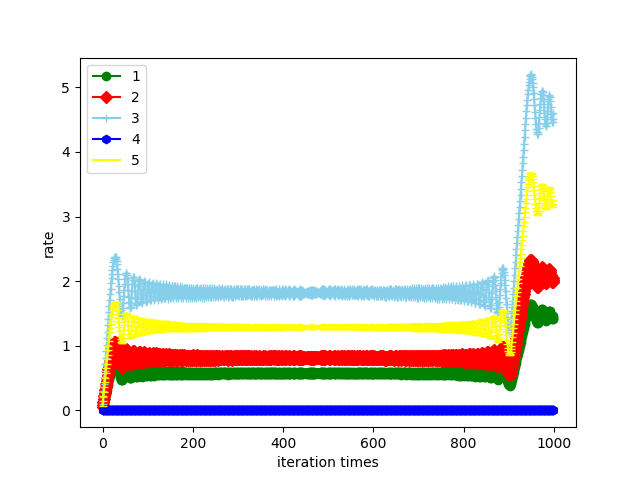
93. ax = subplot(111) #注意一般都在ax中设置,不再plot中设置
94. ymajorLocator  = MultipleLocator(0.1) #将y轴主刻度标签设置为0.5的倍数
95. ax.yaxis.set\_major\_locator(ymajorLocator)
96. plt.plot(t, r1, marker='o', color='green', label='1')
97. plt.plot(t, r2, marker='D',color='red', label='2')
98. plt.plot(t, r3, marker='+',color='skyblue', label='3')
99. plt.plot(t, r4, marker='h', color='blue', label='4')
100. plt.plot(t, r5, marker='\_',color='yellow', label='5')
102. #plt.plot(t, r6, color='red', label='6')
104. plt.legend() # 显示图例
105. plt.xlabel('iteration times')
106. plt.ylabel('r')
107. plt.show()

独立编组结果如图：



好的，从图像我们来看看Kuramoto模型在描述这个编组的时候，5组最终稳定，我们说这个团队编组还算科学，但我们改变一下K的值，换成分散编组：

1. k2 = [
2. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
3. [1,1,0.9,0.8,0.7,0.6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2],
4. [1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
5. [0.9,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
6. [0.8,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
7. [0.7,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
8. [0.6,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
9. [0,0,0,0,0,0,1,1,0.9,0.8,0.7,0.6,0,0,0,0,0,0,0,0,0,0,0,2],
10. [0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
11. [0,0,0,0,0,0,0.9,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
12. [0,0,0,0,0,0,0.8,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
13. [0,0,0,0,0,0,0.7,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
14. [0,0,0,0,0,0,0.6,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
15. [0,0,0,0,0,0,0,0,0,0,0,0,1,0.9,0.8,0.7,0.6,0,0,0,0,0,0,2],
16. [0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0],
17. [0,0,0,0,0,0,0,0,0,0,0,0,0.9,0,1,0,0,0,0,0,0,0,0,0],
18. [0,0,0,0,0,0,0,0,0,0,0,0,0.8,0,0,1,0,0,0,0,0,0,0,0,0],
19. [0,0,0,0,0,0,0,0,0,0,0,0,0.7,0,0,0,1,0,0,0,0,0,0,0,0],
20. [0,0,0,0,0,0,0,0,0,0,0,0,0.6,0,0,0,0,1,0,0,0,0,0,0,0],
21. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.9,0.8,0.7,0.6,2],
22. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0],
23. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.9,0,1,0,0,0,0],
24. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.8,0,0,1,0,0,0],
25. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.7,0,0,0,1,0,0],
26. [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.6,0,0,0,0,1,0],
27. [2,0,0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,0,1]



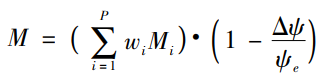
这两幅图像都在开始阶段大幅波动，而后在一定范围内趋于稳定，那么到底哪个分组模式最符合实际，最能突出编组能力呢？

这里还有一个公式，来解决这个问题，编组同步能力的量化：



就可以描述某编组的同步效果，是达到稳定状态后序参量的均值，β∈(0,1)是调节因子。我们可以用来比较编组内部的好坏。那编组间能力的好坏怎样比较呢？

这个Kuramoto模型同样有所考虑，它有一个描述整个系统编组能力的公式：



其中，P是编组的数量，是第i个编组的同步能力，是编组在整个系统中的权重，是各编组平均 相位的均值，是各编组平均相位的标准差。具体的计算不是这篇文章的重点，就不在计算和M的值来比较上述例子独立编组和分散编组的好坏了，本篇文章主要是讲下Kuramoto模型的解决思路，尤其是上面解决值的方法可以套用在其他Kuramoto模型中，做一个目标估计绰绰有余的。

下面是解决Kuramoto模型常用的MATLAB编程方法，具体思路与上述基本一致，这里不再赘述，K的值我们给另一种编组模式：不完全分散编组模式，也是现在实际上最长用的编组方式，直接上代码：

1. clc;
2. clear all;
4. k=[0  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0.5;
5. 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
6. 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
7. 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
8. 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
9. 0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0.5;
10. 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
11. 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
12. 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
13. 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
14. 0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0  0  0.5;
15. 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
16. 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
17. 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
18. 0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0;
19. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0.5;
20. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0;
21. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0;
22. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0;
23. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0;
24. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  2  2  0.5;
25. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0;
26. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0;
27. 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0;
28. 0.5 0 0  0  0  0.5 0 0  0  0 0.5 0  0  0  0  0.5 0 0  0  0  0.5 0 0  0  0;
29. ]
30. N = 25;
31. Step= 10000;
32. Theta=zeros(Step,N);
33. Omega =zeros(Step,N);
34. DeltaT=0.01
35. Theta(1,:)=[0 pi/2 pi 3\*pi/2 0 pi/2 pi 3\*pi/2 0 pi/2 pi 3\*pi/2 0 pi/2 pi 3\*pi/2 0 pi/2 pi 3\*pi/2 0 pi/2 pi 3\*pi/2 0];
36. Omega(1,:)=2\*pi\*[2  3   3  4      4 2    3  3      4 4    2  3      3 4    4  2      3 3    4  4      2 2    3  4      1];
38. %   求解微分方程
39. **for** n = 1:Step
40. **for** i = 1:N
41. Sigma = 0;
42. **for** j = 1:N
43. %%判断k对同步效果的影响
44. = Sigma + k(i,j) \* sin( Theta(n,i) - Theta(n,j) );
45. end
46. Omega(n+1,i) = Omega(n,i) - Sigma;
47. Theta(n+1,i) = Omega(n+1,i)\*DeltaT + Theta(n,i);
48. end
49. end

52. %   求解序参量r(t)
53. groupN = 5;
54. Step = 1000;
55. **for** i = 1:Step
56. sigma1 = 0;
57. sigma2 = 0;
58. sumTheta = 0;
59. **for** j =6:10  %表示相应的群组
60. sigma1=sigma1+cos(Theta(i,j));
61. sigma2=sigma2+sin(Theta(i,j));
62. end
64. r(i)= sqrt( sigma1^2+sigma2^2 )/groupN;
65. end
66. x= 0.01:DeltaT:DeltaT\*Step;
67. plot(x,r);

MATLAB版不完全分散编组结果如下图：

