**阶段性实验结果**

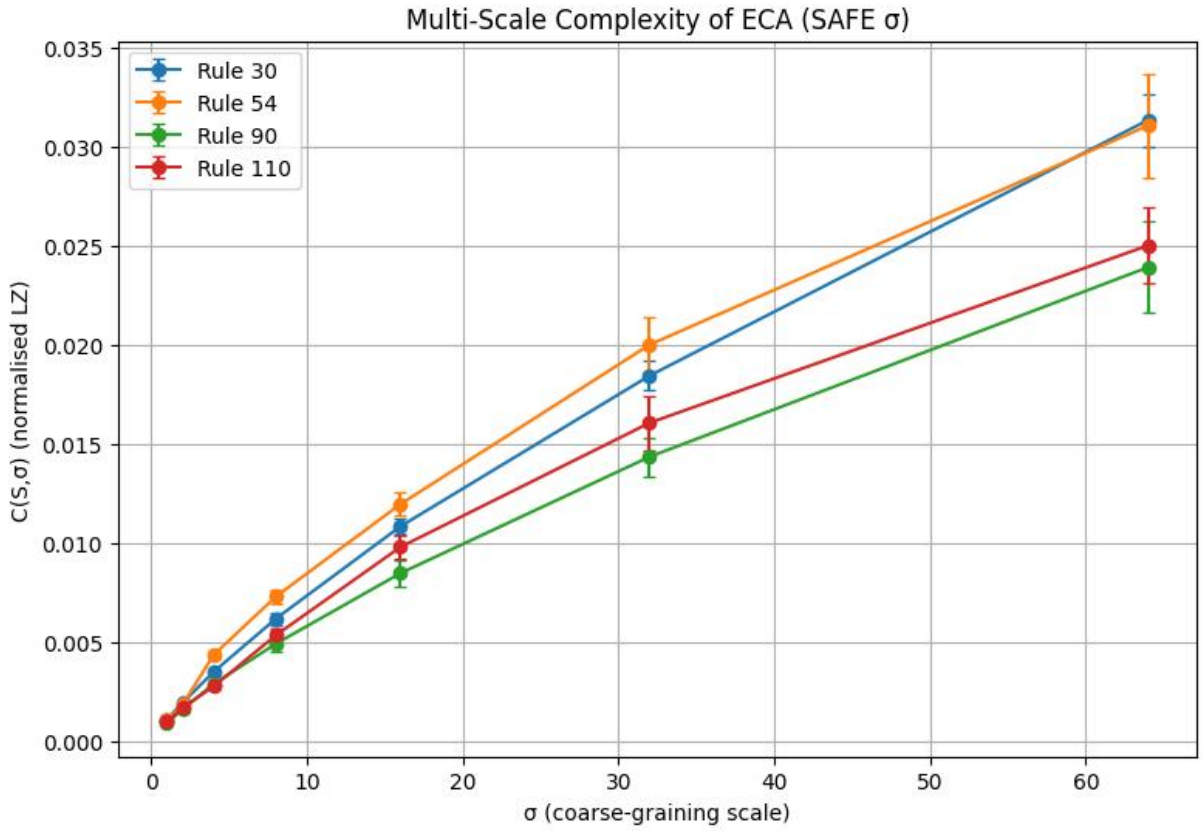
**Preliminary Experimental Results**

本结果由 MSC-CA（Full Verification Script） 生成，源代码：Multiscale_Complexity_Framework_ECA.py，运行平台：Google Colab / Python 3。结果处于实验性、阶段性阶段，文本与结论欢迎探讨与复现交流：kaifanxieve@gmail.com

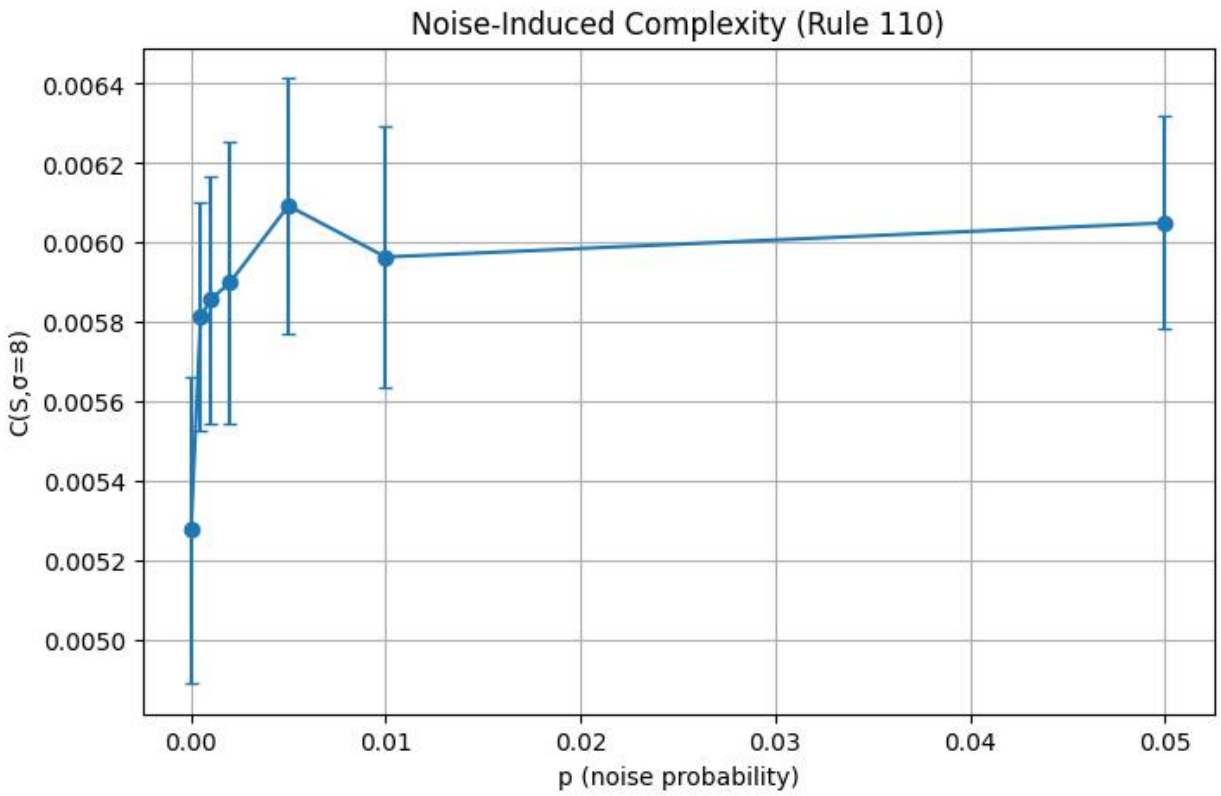
。

These results were generated by MSC-CA (Full Verification Script), source code: **Multiscale_Complexity_Framework_ECA.py**, executed on Google Colab / Python 3. The findings are experimental and preliminary; discussion and reproducibility feedback are welcome: kaifanxieve@gmail.com.

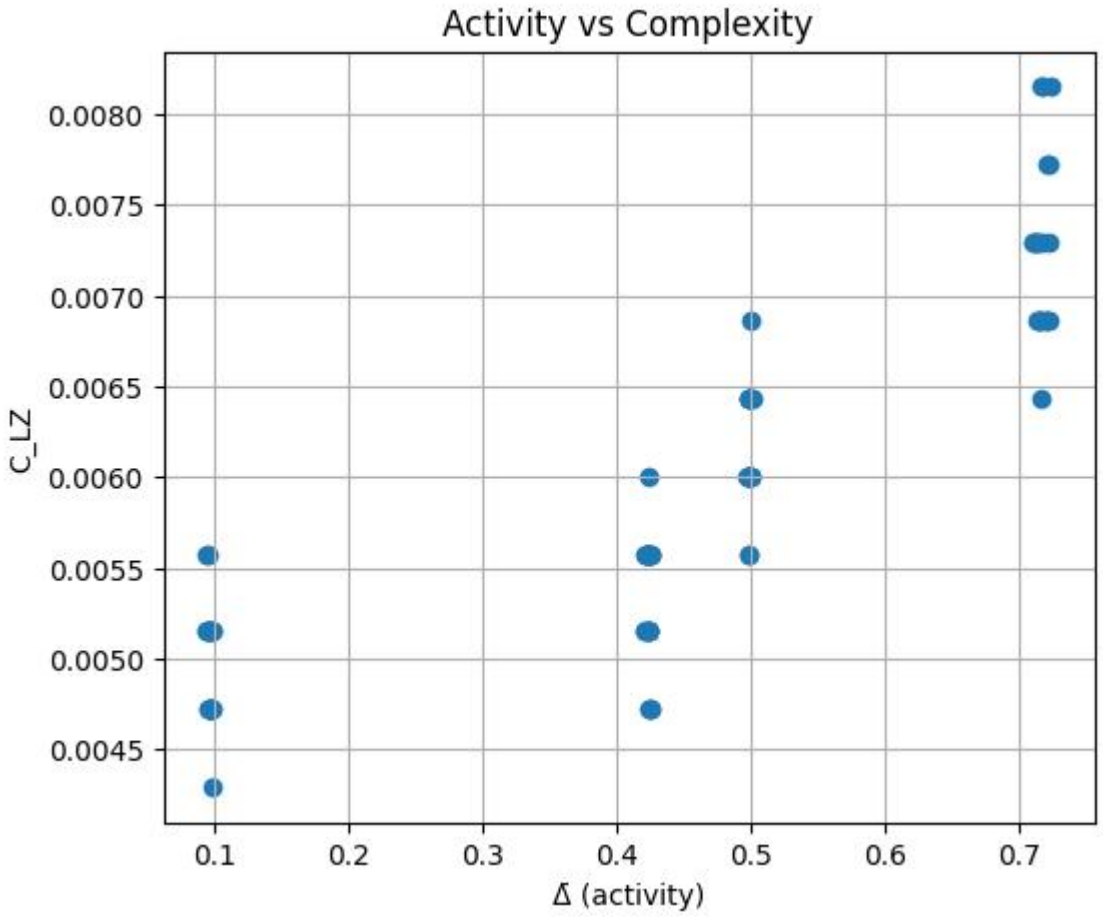
没必要多说了。
懂的人自然懂，不懂的也没必要多说了。
算了，还是不说话吧，免得造成误解。
There's no need to say more.
Those who understand will understand; those who don't won't, no matter how much is said.
Better to remain silent, to avoid misunderstanding.




Ranking by $\sigma$:
σ=1: [30, 54, 110, 90]
σ=2: [30, 54, 110, 90]
σ=4: [54, 30, 90, 110]
σ=8: [54, 30, 110, 90]
σ=16: [54, 30, 110, 90]
σ=32: [54, 30, 110, 90]
σ=64: [30, 54, 110, 90]




Pearson corr(Δ̄, C_LZ): 0.8527358492688043

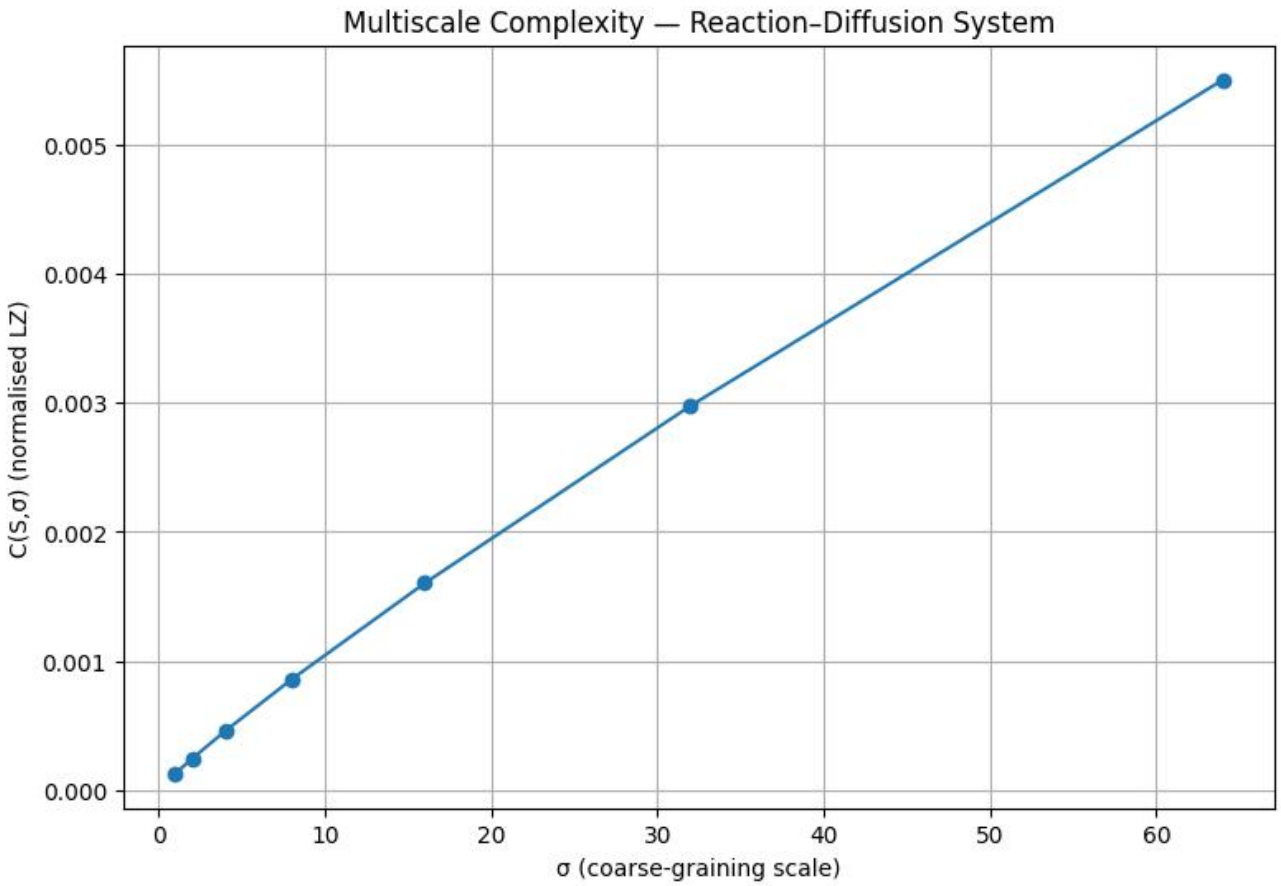


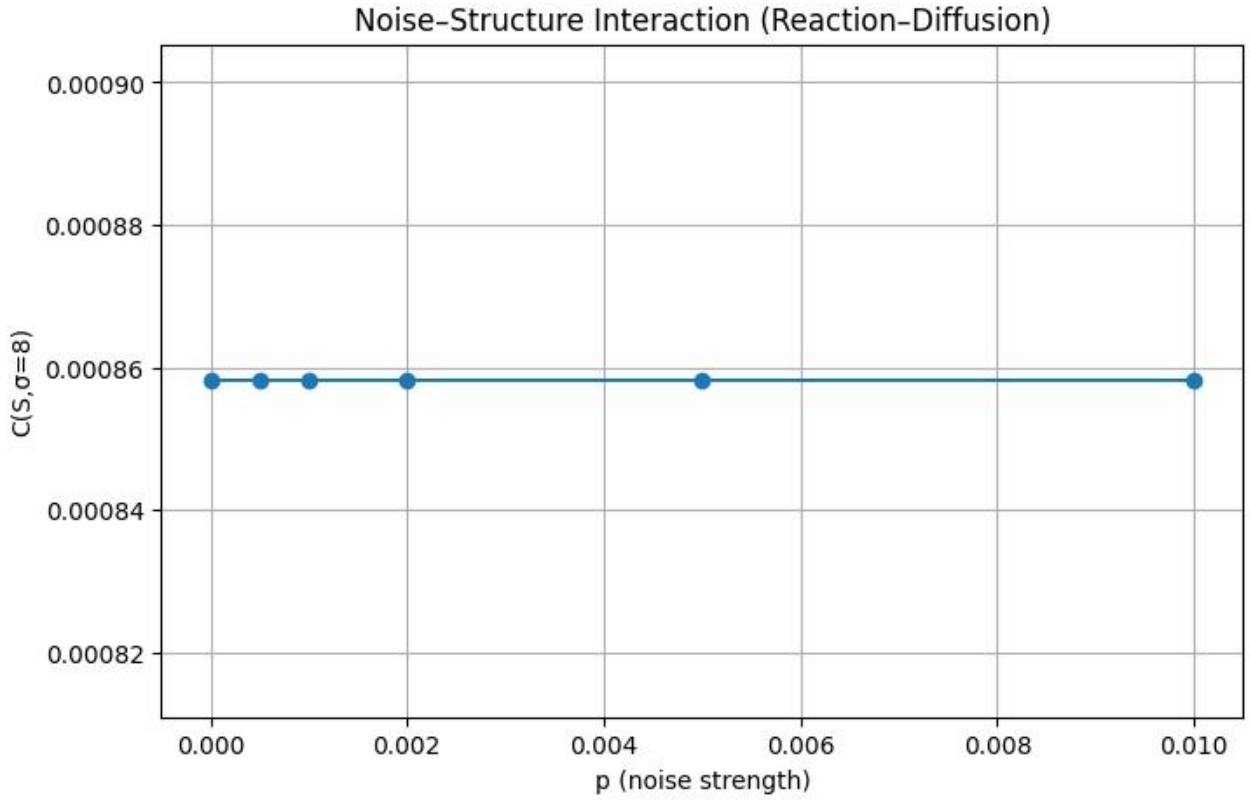

---

源代码：Multiscale_Complexity_Framework_RD.py
运行平台：Google Colab / Python 3
结果处于实验性、阶段性阶段。

Source code: Multiscale_Complexity_Framework_RD.py
Runtime environment: Google Colab / Python 3
The results are experimental

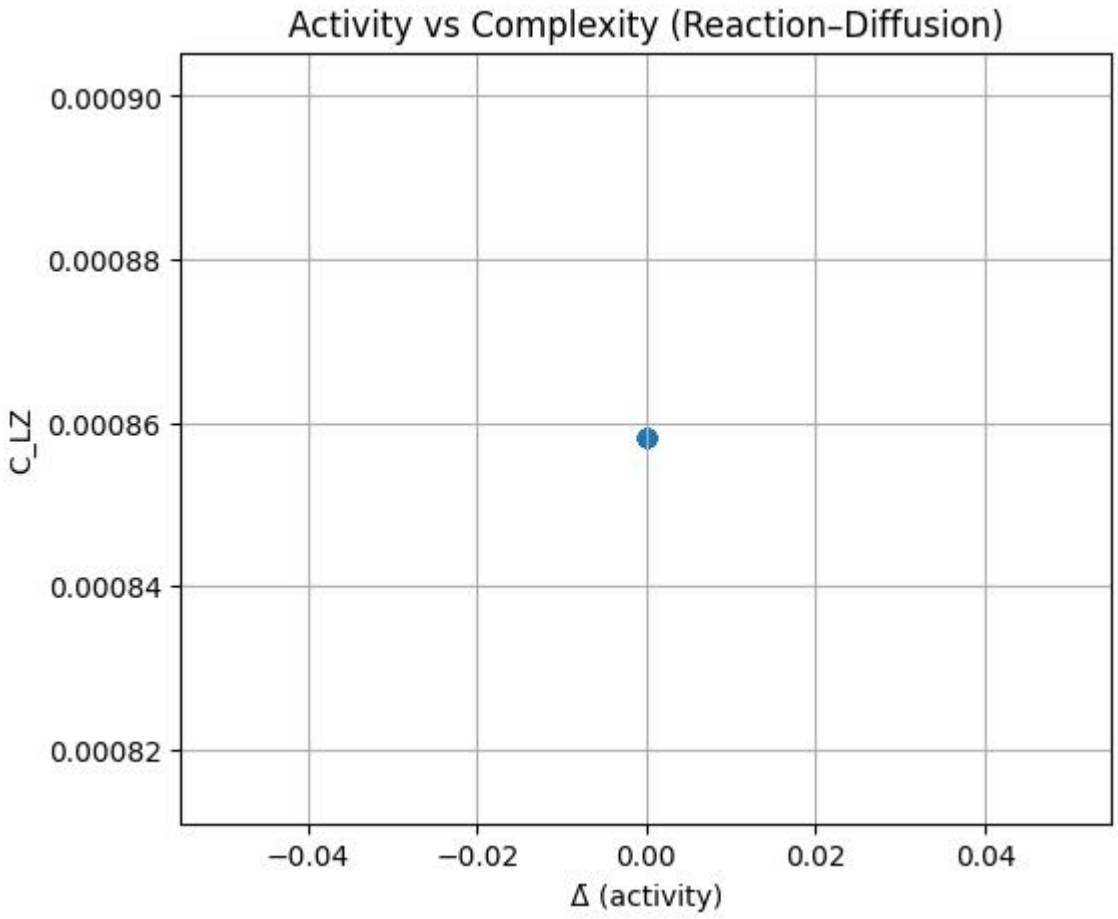
**In smooth diffusion-dominated systems, multiscale complexity may degenerate into a monotonic scale function, indicating the absence of a resolvable scale window.**




Pearson corr(Δ̄, C_LZ): nan

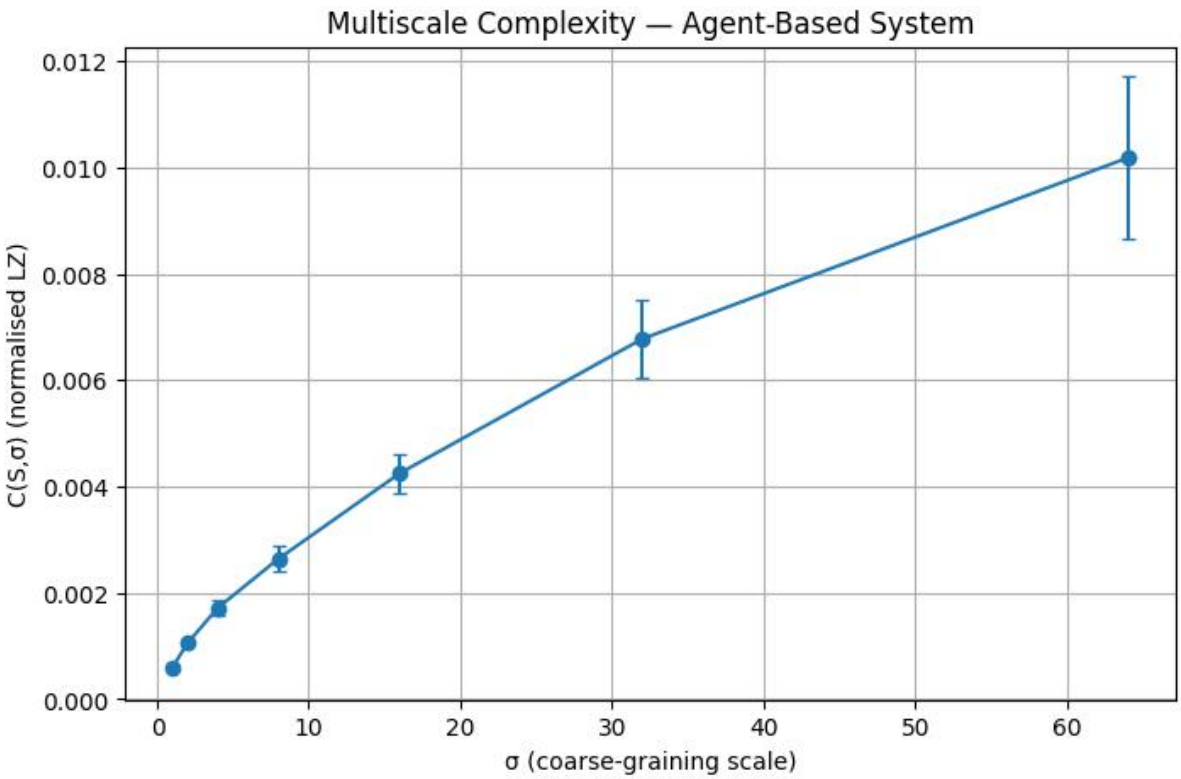
/tmp/ipython-input-3382747823.py:204: RuntimeWarning: invalid value encountered in scalar divide
  return np.sum(x*y) / np.sqrt(np.sum(x*x)*np.sum(y*y))

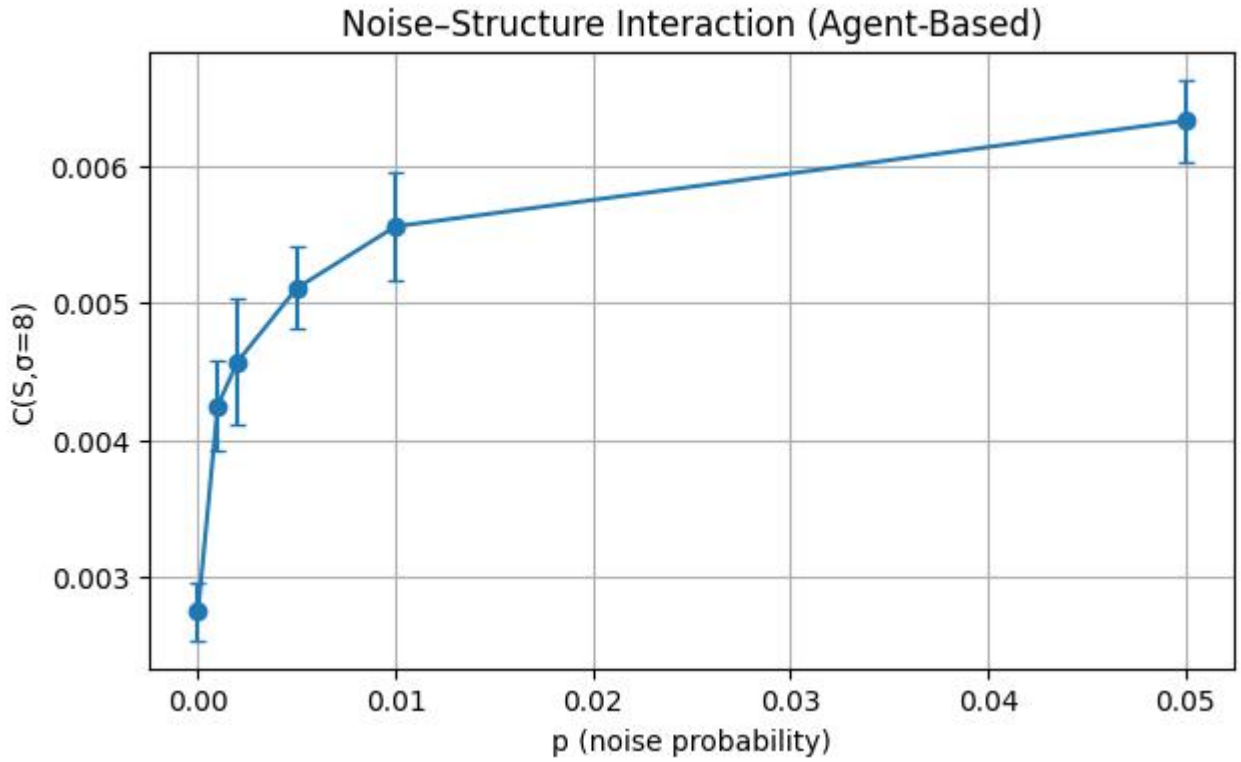


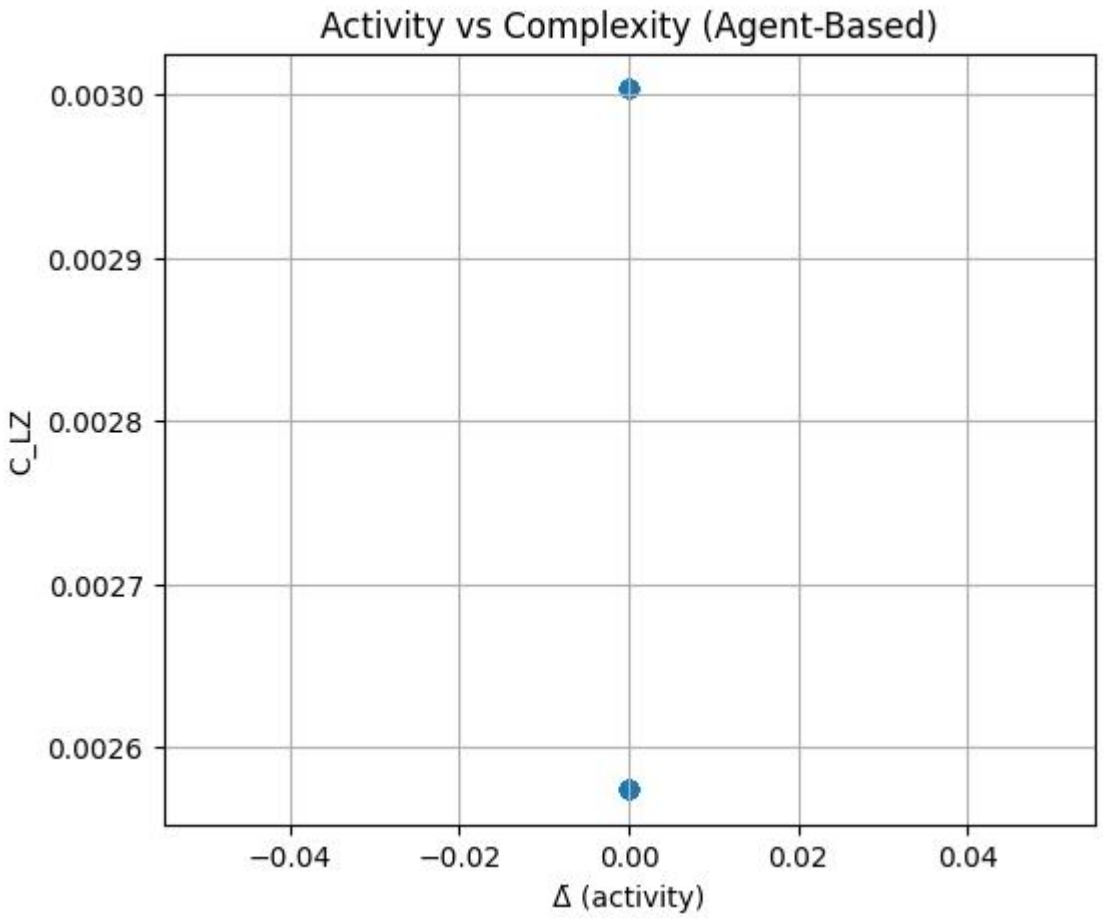

---

源代码：Multiscale_Complexity_Framework_AgentBased.py
运行平台：Google Colab / Python 3
结果处于实验性、阶段性阶段。

Source code: Multiscale_Complexity_Framework_AgentBased.py
Runtime environment: Google Colab / Python 3
The results are experimental


Multiscale Complexity — Agent-Based System
C(S,σ) (normalised LZ)
σ (coarse-graining scale)
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0
10
20
30
40
50
60

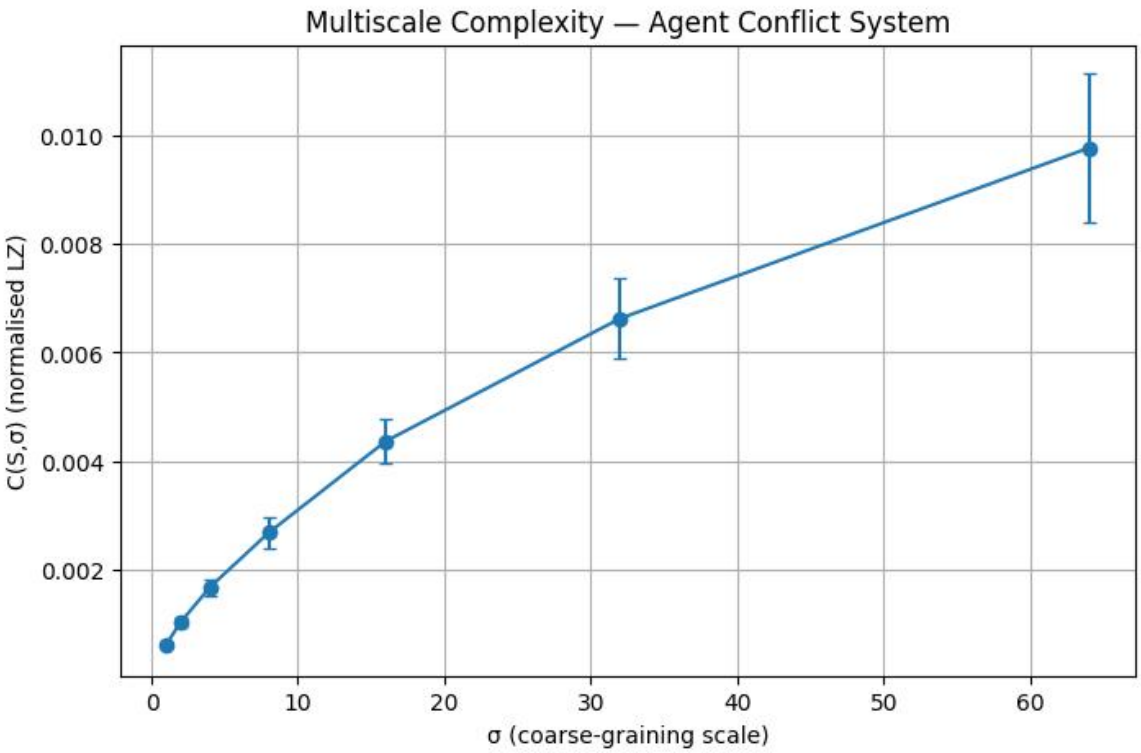

Noise–Structure Interaction (Agent-Based)
C(S,σ=8)
p (noise probability)
0.003
0.004
0.005
0.006
0.00
0.01
0.02
0.03
0.04
0.05

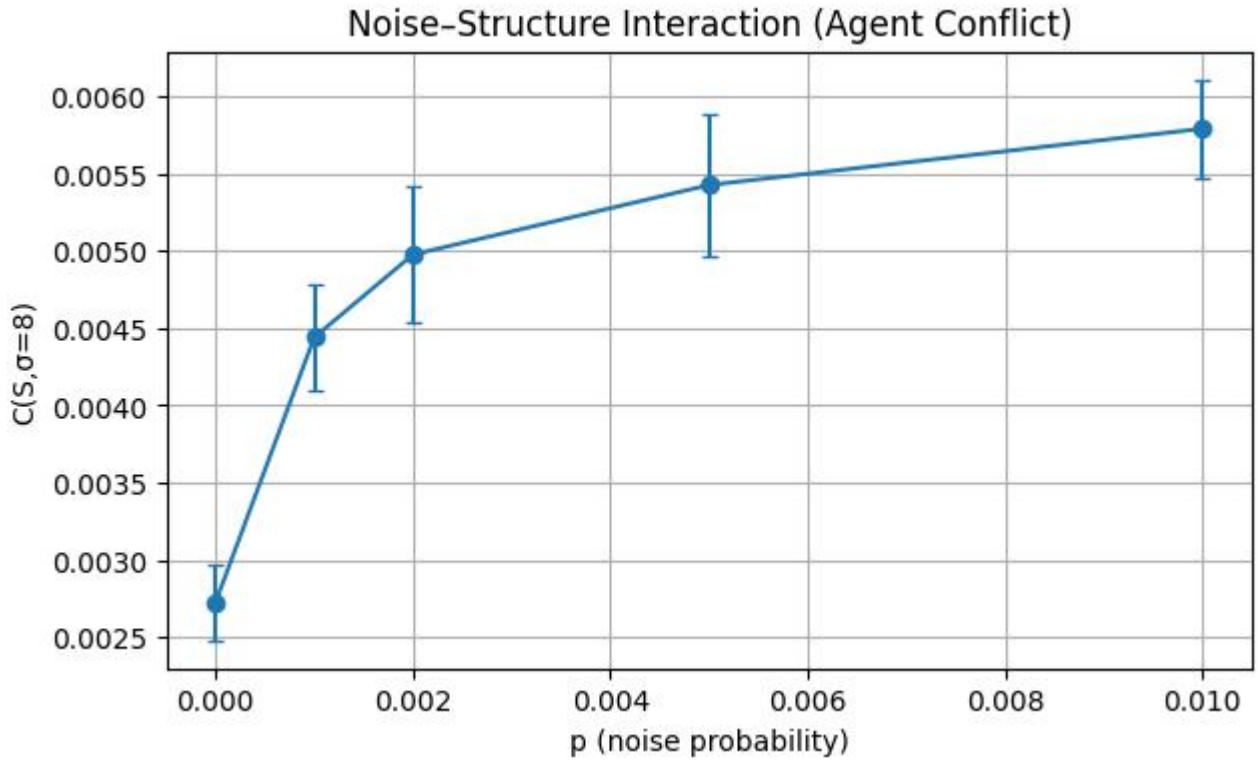



---

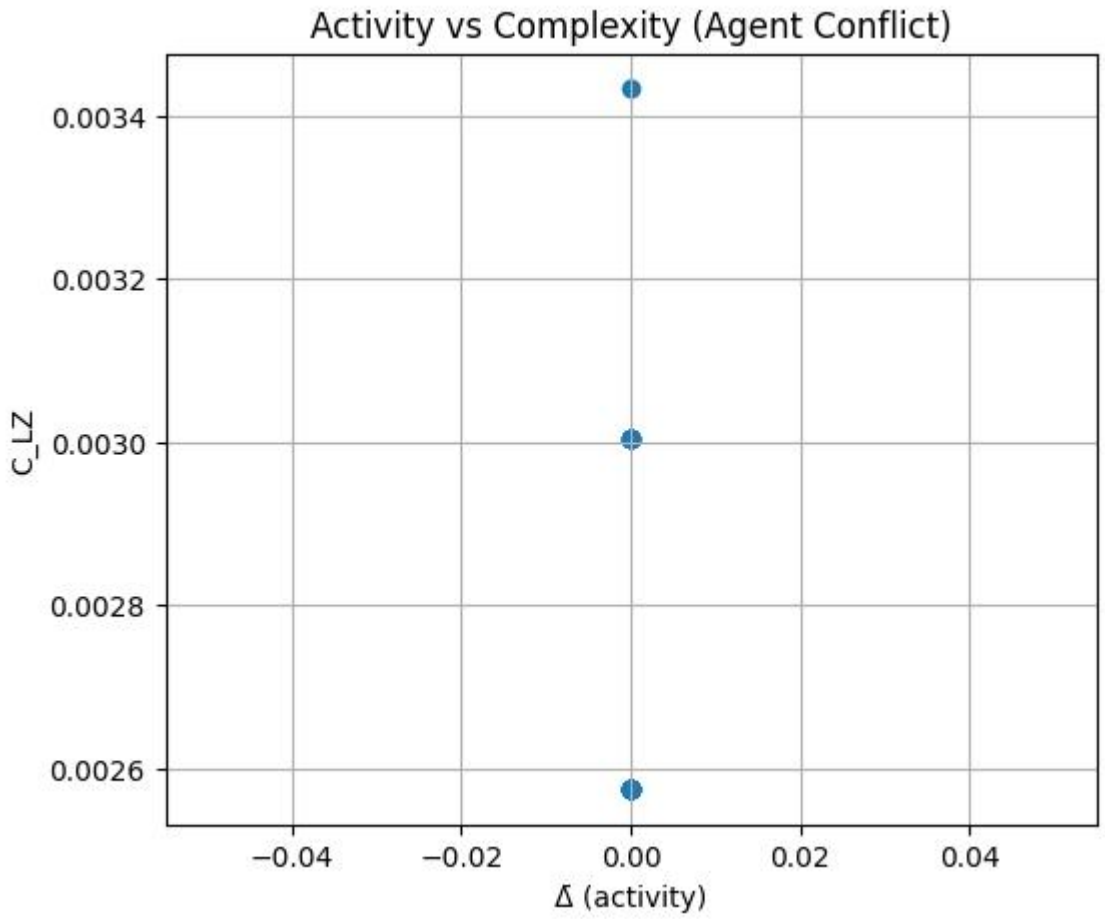
源代码：Multiscale_Complexity_Framework_AgentConflict.py
运行平台：Google Colab / Python 3
结果处于实验性、阶段性阶段。

Source code: Multiscale_Complexity_Framework_AgentConflict.py
Runtime environment: Google Colab / Python 3
The results are experimental




Pearson corr($\bar{\Delta}$, C_LZ): nan




---

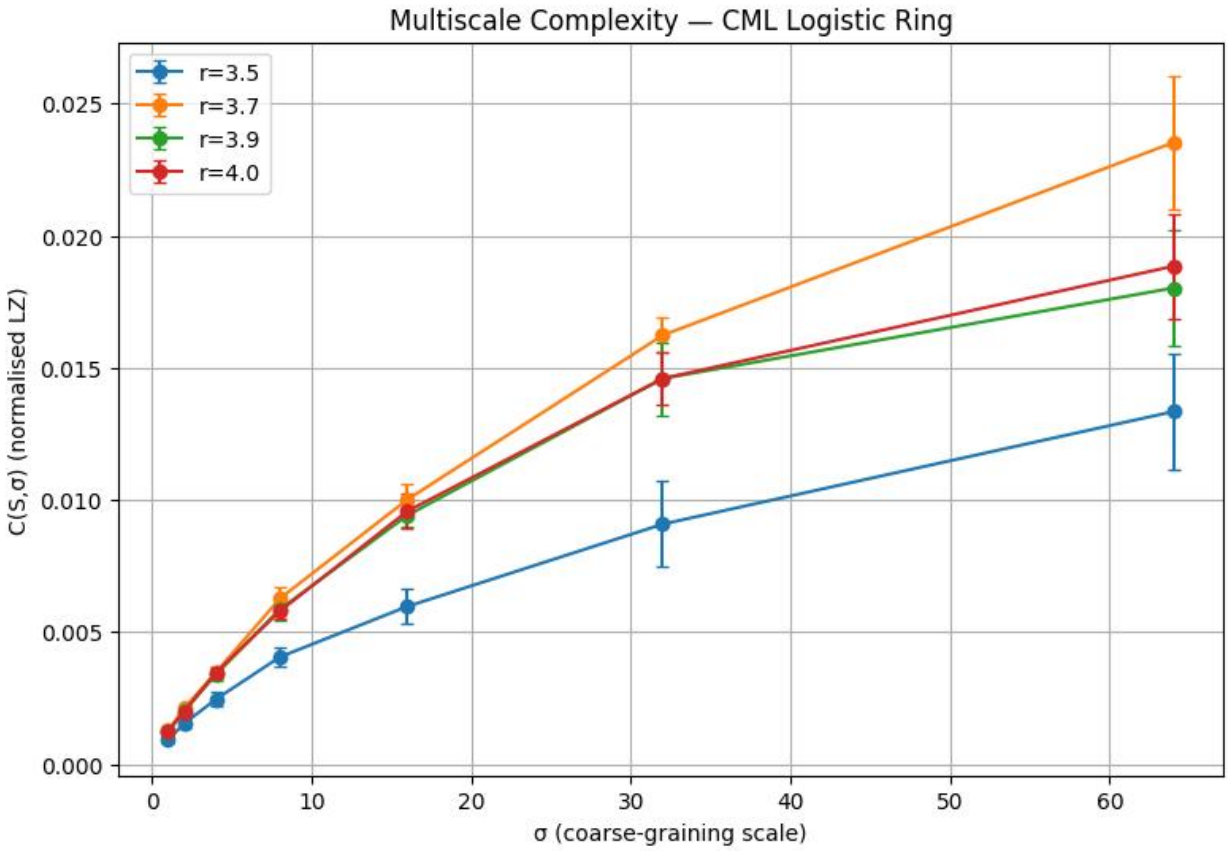
源代码：Coupled Map Lattice  - Logistic Map Ring.py
运行平台：Google Colab / Python 3
结果处于实验性、阶段性阶段。

Source code: Coupled Map Lattice  - Logistic Map Ring.py
Runtime environment: Google Colab / Python 3
The results are experimental

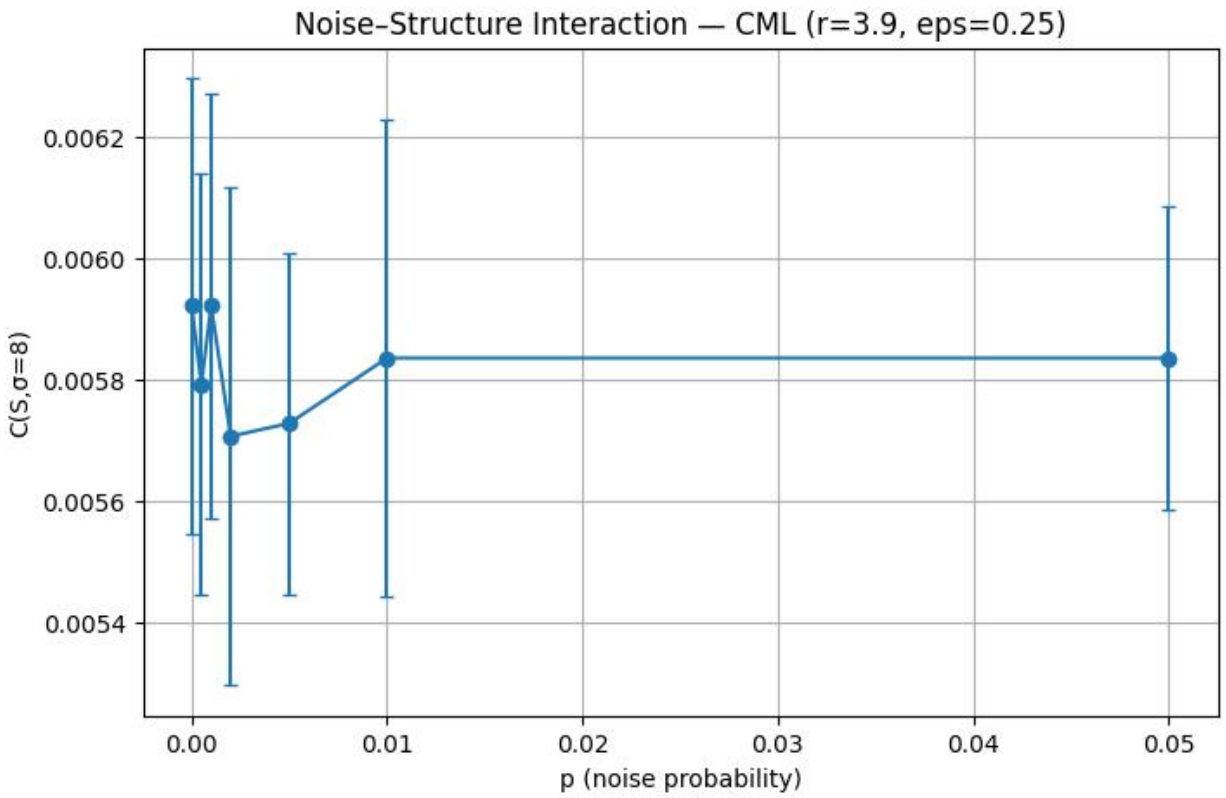
在多类动力系统（反应扩散、ABM、CML）中，复杂性随尺度呈单调行为，未出现尺度窗口；而在特定离散规则临界系统（ECA）中，复杂性作为尺度函数表现出中间尺度峰值与排序翻转。该结果表明，复杂性尺度窗口并非动力系统的普遍性质，而是高度依赖于规则级离散临界结构。

Across multiple classes of dynamical systems (reaction–diffusion systems, ABM, and CML), complexity exhibits monotonic behaviour with respect to scale, and no scale window is observed. By contrast, in specific discrete rule–critical systems (ECA), complexity as a function of scale displays an intermediate-scale peak accompanied by a reversal in ordering. These results indicate that complexity scale windows are not a

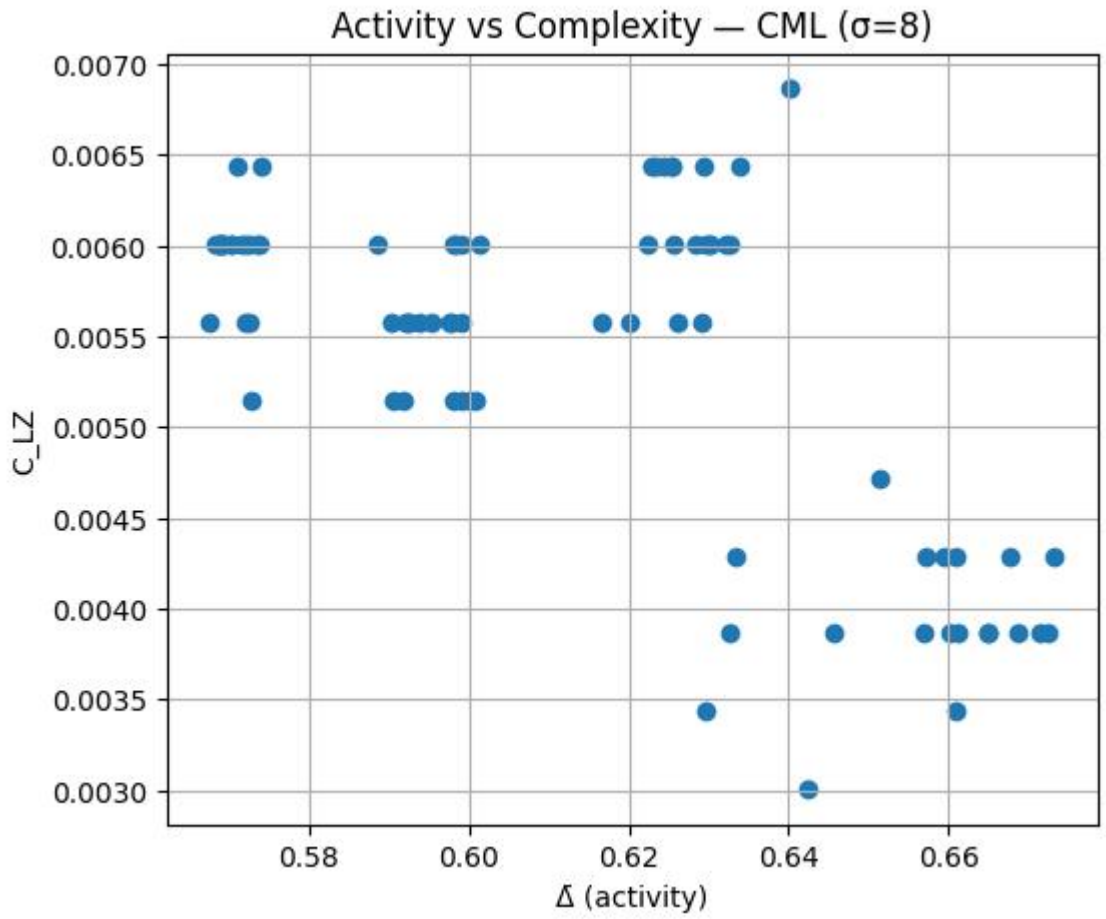
universal property of dynamical systems, but instead depend critically on discrete, rule-level critical structures.




Ranking by σ (higher C_LZ first):
σ=1: [3.7, 4.0, 3.9, 3.5]
σ=2: [3.7, 3.9, 4.0, 3.5]
σ=4: [3.7, 4.0, 3.9, 3.5]
σ=8: [3.7, 3.9, 4.0, 3.5]
σ=16: [3.7, 4.0, 3.9, 3.5]
σ=32: [3.7, 3.9, 4.0, 3.5]
σ=64: [3.7, 4.0, 3.9, 3.5]




Pearson corr($\bar{\Delta}$, C_LZ): -0.6300336809503791




---

源代码：msc_fast_ranking_flip.py
运行平台：www.kaggle.com/ Python 3
结果处于实验性、阶段性阶段。

Source code: msc_fast_ranking_flip.py
Runtime environment: www.kaggle.com/ / Python 3
The results are experimental

=== FAST RANKING-FLIP TEST START ===

--- Coarse-grain mode: majority ---
σ= 1  order=[30, 54, 110, 90]  C=[np.float64(0.001907), np.float64(0.001869), np.float64(0.00172), np.float64(0.001645)]
σ= 2  order=[30, 54, 90, 110]  C=[np.float64(0.00324), np.float64(0.00317), np.float64(0.003029), np.float64(0.002818)]
σ= 4  order=[54, 30, 90, 110]  C=[np.float64(0.00661), np.float64(0.005552), np.float64(0.005552), np.float64(0.004759)]
σ= 8  order=[54, 30, 90, 110]  C=[np.float64(0.010869), np.float64(0.010128), np.float64(0.00914), np.float64(0.00914)]

σ=16 order=[54, 30, 110, 90] C=[np.float64(0.017455), np.float64(0.016536), np.float64(0.016077), np.float64(0.015158)]
σ=32 order=[54, 30, 90, 110] C=[np.float64(0.029723), np.float64(0.028874), np.float64(0.027175), np.float64(0.024627)]
σ=64 order=[54, 30, 90, 110] C=[np.float64(0.049906), np.float64(0.048346), np.float64(0.046787), np.float64(0.038989)]

[mode=majority] RANKING FLIP: True

--- Coarse-grain mode: parity ---
σ= 1 order=[30, 54, 110, 90] C=[np.float64(0.001907), np.float64(0.001869), np.float64(0.00172), np.float64(0.001645)]
σ= 2 order=[30, 90, 110, 54] C=[np.float64(0.003311), np.float64(0.003029), np.float64(0.002818), np.float64(0.002254)]
σ= 4 order=[54, 30, 90, 110] C=[np.float64(0.006081), np.float64(0.005684), np.float64(0.00542), np.float64(0.005023)]
σ= 8 order=[110, 30, 54, 90] C=[np.float64(0.010375), np.float64(0.010128), np.float64(0.010128), np.float64(0.009634)]
σ=16 order=[54, 110, 30, 90] C=[np.float64(0.017914), np.float64(0.016995), np.float64(0.016536), np.float64(0.016077)]
σ=32 order=[110, 30, 54, 90] C=[np.float64(0.029723), np.float64(0.028024), np.float64(0.028024), np.float64(0.027175)]
σ=64 order=[54, 110, 30, 90] C=[np.float64(0.049906), np.float64(0.049906), np.float64(0.048346), np.float64(0.040549)]

[mode=parity] RANKING FLIP: True

=== FAST RANKING-FLIP TEST END ===

---

源代码：**msc_breaktest_pipeline_artifact.py**
运行平台：Google Colab / Python 3
结果处于实验性、阶段性阶段。

Source code: **msc_breaktest_pipeline_artifact.py**
Runtime environment: Google Colab / Python 3
The results are experimental

=== BREAK SUCCESSES (potential counterexamples / artifact evidence) ===
[RD] thr=0.3,cg=majority,scan=time_major -> {'flip': True, 'peak(default)': False, 'peak_idx': 6, 'rank0': ['F0.022_k0.051', 'F0.040_k0.060', 'F0.030_k0.062', 'F0.035_k0.065'], 'rank_last': ['F0.030_k0.062', 'F0.035_k0.065', 'F0.040_k0.060', 'F0.022_k0.051']}
[RD] thr=0.3,cg=majority,scan=space_major -> {'flip': True, 'peak(default)': False, 'peak_idx': 6, 'rank0': ['F0.040_k0.060', 'F0.022_k0.051', 'F0.030_k0.062',

'F0.035_k0.065'], 'rank_last': ['F0.030_k0.062', 'F0.035_k0.065', 'F0.040_k0.060', 'F0.022_k0.051']}
[RD] thr=0.3,cg=OR,scan=time_major -> {'flip': True, 'peak(default)': False, 'peak_idx': 6, 'rank0': ['F0.022_k0.051', 'F0.040_k0.060', 'F0.030_k0.062', 'F0.035_k0.065'], 'rank_last': ['F0.040_k0.060', 'F0.022_k0.051', 'F0.030_k0.062', 'F0.035_k0.065']}
[RD] thr=0.3,cg=OR,scan=space_major -> {'flip': True, 'peak(default)': False, 'peak_idx': 6, 'rank0': ['F0.040_k0.060', 'F0.022_k0.051', 'F0.030_k0.062', 'F0.035_k0.065'], 'rank_last': ['F0.040_k0.060', 'F0.022_k0.051', 'F0.030_k0.062', 'F0.035_k0.065']}
[RD] thr=0.3,cg=AND,scan=time_major -> {'flip': True, 'peak(default)': False, 'peak_idx': 6, 'rank0': ['F0.022_k0.051', 'F0.040_k0.060', 'F0.030_k0.062', 'F0.035_k0.065'], 'rank_last': ['F0.030_k0.062', 'F0.035_k0.065', 'F0.040_k0.060', 'F0.022_k0.051']}
[RD] thr=0.3,cg=AND,scan=space_major -> {'flip': True, 'peak(default)': False, 'peak_idx': 6, 'rank0': ['F0.040_k0.060', 'F0.022_k0.051', 'F0.030_k0.062', 'F0.035_k0.065'], 'rank_last': ['F0.030_k0.062', 'F0.035_k0.065', 'F0.040_k0.060', 'F0.022_k0.051']}
[RD] thr=0.3,cg=parity,scan=time_major -> {'flip': True, 'peak(default)': False, 'peak_idx': 6, 'rank0': ['F0.022_k0.051', 'F0.040_k0.060', 'F0.030_k0.062', 'F0.035_k0.065'], 'rank_last': ['F0.022_k0.051', 'F0.040_k0.060', 'F0.030_k0.062', 'F0.035_k0.065']}
[RD] thr=0.3,cg=parity,scan=space_major -> {'flip': True, 'peak(default)': False, 'peak_idx': 6, 'rank0': ['F0.040_k0.060', 'F0.022_k0.051', 'F0.030_k0.062', 'F0.035_k0.065'], 'rank_last': ['F0.022_k0.051', 'F0.040_k0.060', 'F0.030_k0.062', 'F0.035_k0.065']}
[CML] thr=0.3,cg=majority,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.3,cg=majority,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.3,cg=OR,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.3,cg=OR,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.3,cg=AND,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.7, 3.9, 4.0, 3.5]}
[CML] thr=0.3,cg=AND,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.7, 3.9, 4.0, 3.5]}
[CML] thr=0.3,cg=parity,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.9, 4.0, 3.7, 3.5]}
[CML] thr=0.3,cg=parity,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.9, 3.7, 4.0, 3.5]}
[CML] thr=0.4,cg=majority,scan=time_major -> {'flip': True, 'peak(r=3.9)': np.False_, 'peak_idx': 4, 'rank0': [4.0, 3.7, 3.9, 3.5], 'rank_last': [3.5, 4.0, 3.7, 3.9]}
[CML] thr=0.4,cg=majority,scan=space_major -> {'flip': True, 'peak(r=3.9)': np.False_, 'peak_idx': 4, 'rank0': [3.9, 4.0, 3.7, 3.5], 'rank_last': [3.5, 4.0, 3.7, 3.9]}
[CML] thr=0.4,cg=OR,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.7, 3.9, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.4,cg=OR,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.9, 4.0, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}

[CML] thr=0.4,cg=AND,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.7, 3.9, 3.5], 'rank_last': [3.7, 3.9, 3.5, 4.0]}
[CML] thr=0.4,cg=AND,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.9, 4.0, 3.7, 3.5], 'rank_last': [3.9, 3.7, 3.5, 4.0]}
[CML] thr=0.4,cg=parity,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.7, 3.9, 3.5], 'rank_last': [3.9, 3.7, 4.0, 3.5]}
[CML] thr=0.4,cg=parity,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.9, 4.0, 3.7, 3.5], 'rank_last': [3.9, 4.0, 3.7, 3.5]}
[CML] thr=0.5,cg=majority,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.7, 4.0, 3.9, 3.5]}
[CML] thr=0.5,cg=majority,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.7, 3.9, 4.0, 3.5]}
[CML] thr=0.5,cg=OR,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.5,cg=OR,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.5,cg=AND,scan=time_major -> {'flip': True, 'peak(r=3.9)': np.False_, 'peak_idx': 4, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.7, 3.9, 3.5, 4.0]}
[CML] thr=0.5,cg=AND,scan=space_major -> {'flip': True, 'peak(r=3.9)': np.False_, 'peak_idx': 4, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.7, 3.9, 3.5, 4.0]}
[CML] thr=0.5,cg=parity,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.7, 3.9, 4.0, 3.5]}
[CML] thr=0.5,cg=parity,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.7, 4.0, 3.9, 3.5]}
[CML] thr=0.6,cg=majority,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.7, 4.0, 3.9, 3.5]}
[CML] thr=0.6,cg=majority,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [4.0, 3.9, 3.7, 3.5]}
[CML] thr=0.6,cg=OR,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.6,cg=OR,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.6,cg=AND,scan=time_major -> {'flip': True, 'peak(r=3.9)': np.False_, 'peak_idx': 4, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.6,cg=AND,scan=space_major -> {'flip': True, 'peak(r=3.9)': np.False_, 'peak_idx': 4, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.6,cg=parity,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [3.7, 4.0, 3.9, 3.5], 'rank_last': [4.0, 3.9, 3.7, 3.5]}
[CML] thr=0.6,cg=parity,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.7, 3.9, 4.0, 3.5]}
[CML] thr=0.7,cg=majority,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [4.0, 3.9, 3.7, 3.5]}
[CML] thr=0.7,cg=majority,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [4.0, 3.9, 3.7, 3.5]}
[CML] thr=0.7,cg=OR,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.7,cg=OR,scan=space_major -> {'flip': True, 'peak(r=3.9)': np.False_, 'peak_idx': 3, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.7,cg=AND,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}

[CML] thr=0.7,cg=AND,scan=space_major -> {'flip': True, 'peak(r=3.9)': np.False_, 'peak_idx': 3, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.5, 3.7, 3.9, 4.0]}
[CML] thr=0.7,cg=parity,scan=time_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [4.0, 3.9, 3.7, 3.5]}
[CML] thr=0.7,cg=parity,scan=space_major -> {'flip': True, 'peak(r=3.9)': False, 'peak_idx': 6, 'rank0': [4.0, 3.9, 3.7, 3.5], 'rank_last': [3.7, 3.9, 4.0, 3.5]}

---

有些错误还没修正。
对精神世界的探索太危险，精神消耗也过大；我担心自己会变得不稳定，
先停一段时间吧。

Some errors haven’t been fixed yet.
Exploring the inner world is risky and mentally draining; I’m concerned it could affect my stability.
I’ll pause for a while.

物理学要是感兴趣，也可以自己建模试试；我这边没有设备，且还有其他限制。

If physics is of interest, one can try modelling it independently; I don’t have the equipment, and there are additional constraints.