

A Cohesion UFT Stability Classification of the Three-Body Problem: Torsion Asymmetry, Slip Accumulation, and the Two-Timescale Memory Architecture

Dexter Alan Gilbert

Independent Researcher 2026

Abstract

This paper applies the Cohesion Unified Field Theory operator set to the classification of three-body orbital stability. Three test systems are classified: the stable Chenciner-Montgomery figure-eight orbit, the same orbit with a 1% position perturbation (marginally unstable), and the Pythagorean problem (chaotic, ejection at step ~ 9890). The Cohesion UFT operators — torsion asymmetry τ_{asym} , slip accumulation $S(t)$, recursion coherence $\kappa(t)$, cascade depth $D(t)$, and structural coherence $\Psi(t)$ — replace the previous Canon framework terminology throughout. The two-timescale memory architecture is retained: long-term memory (S , D) persists across 2-cycle resets and encodes accumulated angular momentum asymmetry; short-term memory (the coherence history buffer) is cleared every two orbital cycles, forcing the system to re-evaluate orbital periodicity from a clean state. All three systems are correctly classified. The primary discrimination signal for marginal instability is the variability of the mean pairwise separation $E(t)$: the perturbed figure-eight shows $9.1\times$ higher $E(t)$ variability than the stable orbit, reflecting the growing perturbation amplified by the coherence-rebuild mechanism of the 2-cycle reset. The universal toggle threshold $\Phi_{\text{toggle}} = 32/(3\pi^2 - 4) = 1.2496$ governs the gear-shift between stable (hexapolar) and transitional (bipolar) orbital states.

INTRODUCTION

The three-body problem admits no general closed-form solution and exhibits sensitivity to initial conditions across wide regions of phase space. The practical task of classifying an observed or computed three-body trajectory as stable, marginally unstable, or chaotic is therefore a structural diagnostic problem rather than an analytical one.

This paper derives its diagnostic operators from the Cohesion Unified Field Theory [1], which treats orbital dynamics as a funneled-spring recursion under pressure from the next higher scale. In this framework, orbital stability is the organisational state of the recursion:

a stable orbit maintains a coherent recursion cycle indefinitely; a marginally unstable orbit accumulates torsion asymmetry that slowly disrupts the cycle; a chaotic system undergoes cascade collapse.

The five Cohesion UFT operators applied here map directly to orbital mechanics:

Cohesion UFT	Orbital quantity	Physical meaning
Surplus pressure P_Σ	$1 - \bar{r}/r_{\text{ref}}$	Energy available to sustain the orbital recursion
Torsion asymmetry τ_{asym}	Per-body L deviation	Deviation of angular momenta from symmetric mean
Slip accumulation $S(t)$	Persistent τ excess	Long-term memory of angular momentum imbalance
Recursion coherence $\kappa(t)$	Torsion cycle stability	How consistently the orbit reproduces its period
Cascade depth $D(t)$	Structural history	Accumulated organisational depth of the recursion
Structural coherence $\Psi(t)$	Primary diagnostic	Integrated organisational state $\in [0, 1]$

TEST SYSTEMS

Three trajectory classes are evaluated at $\Delta t = 0.0005$, 12,000 steps, with $G = 1$ and softened gravity ($\varepsilon = 10^{-4}$):

System 1 — Figure-Eight (stable). The Chenciner-Montgomery exact orbit [4], equal masses $m = 1$. A choreographic solution in which all three bodies follow the same figure-eight path with 120 phase offsets. Perfectly periodic; provides the stable reference baseline.

System 2 — Perturbed Figure-Eight (marginally unstable). Identical to System 1 with a +0.01 displacement on body 1’s x -coordinate. The perturbation is small enough that the trajectory visually resembles the stable orbit but grows over the observation window.

System 3 — Pythagorean Problem (chaotic/ejection). Masses $m_1 = 3$, $m_2 = 4$, $m_3 = 5$ at rest at 3-4-5 triangle vertices [5]. The asymmetric mass distribution produces chaotic close approaches and an ejection at step ~ 9890 .

ARCHITECTURE: TWO-TIMESCALE MEMORY

The diagnostic implements a two-timescale memory architecture derived from the Cohesion UFT infinite asymmetric cascade [2]:

Long-Term Memory: Slip Accumulation and Cascade Depth

The slip accumulation $S(t)$ and cascade depth $D(t)$ persist across 2-cycle resets. They encode the accumulated angular momentum asymmetry and structural history over the full observation window. The signed formulation of $S(t)$ allows cancellation: a body that oscillates symmetrically around the mean drift accumulates near zero. A body that persistently carries more angular momentum than the others accumulates positively:

$$S_i(t) = \text{clip}(d S_i(t-1) + \alpha (\tau_i(t) - \bar{\tau}(t)), [0, 1]), \quad d = 0.9995, \alpha = 0.001. \quad (1)$$

This is the primary discriminator for the Pythagorean system: the persistent mass asymmetry (bodies play distinct roles) produces $I_{B1} \approx 0$ for the Pythagorean versus 0.33–0.37 for the figure-eight systems.

Short-Term Memory: The 2-Cycle Reset

The coherence history buffer is cleared every two orbital cycles. A cycle is detected as an upward zero-crossing of the mean torsion asymmetry:

$$\text{cycle} = (\bar{\tau}_{t-5:t} > \bar{\tau}_{\text{running}}) \wedge (\bar{\tau}_{t-10:t-5} \leq \bar{\tau}_{\text{running}}). \quad (2)$$

When the cycle counter reaches 2, ϕ_{history} is cleared and $\kappa(t)$ rebuilds from 1.0. Long-term quantities (S , D) are unchanged. The reset does not improve any single operator — it changes the measurement from a single long-term estimate to a sequence of independent short-term evaluations. The variability across evaluations encodes orbital regularity in a way that single-pass methods cannot access.

Why Variability Is the Marginal Instability Signal

After each reset, $\kappa(t)$ rebuilds. A stable orbit rebuilds identically each time. A marginally unstable orbit rebuilds to a slightly different value each time because the growing perturbation has altered the orbital timing between reset events. This difference is invisible in a single long-term estimate but detectable as elevated variability in $E(t)$ across reset events.

RESULTS

System	σ_E	\bar{E}	$\bar{\Psi}$	I_{B1}	Resets	Classification
Figure-Eight (stable)	0.0007	1.335	0.128	0.338	2	Stable
Perturbed Figure-Eight	0.0075	1.329	0.128	0.369	3	Marginally Unstable
Pythagorean (chaotic)	53.62	140.14	0.000	0.000	1	Chaotic / Ejection

σ_E = std of mean pairwise separation over full run; \bar{E} = late-window mean separation; I_{B1} = late-window mean slip accumulation, body 1.

Key result — marginal instability detection:

The perturbed figure-eight shows $\sigma_E = 0.0075$ versus 0.0007 for the stable orbit — a $9.1\times$ increase in separation variability. This is the primary detection signal. The coherence-rebuild mechanism of the 2-cycle reset amplifies the growing perturbation into a measurable variability difference across reset events. All three systems are correctly classified on at least two independent operators.

Pythagorean System

Correctly classified on three independent operators: ejection flag (trivial), $\bar{\Psi} = 0$ after ejection (structural coherence collapses), and $I_{B1} = 0$ (the signed integrator returns to zero after the chaotic phase eliminates persistent asymmetry). Reset count of 1 (versus 2–3 for periodic systems) reflects the disrupted zero-crossings in chaotic dynamics.

Perturbed Figure-Eight

The $9.1\times$ elevation in σ_E is the primary signal. The $E(t)$ mean (1.329 vs 1.335) changes by less than 0.5% — the mean separation is nearly indistinguishable. The standard deviation, however, is $10.7\times$ higher, reflecting the variable coherence rebuild after each reset: the perturbed orbit does not return to exactly the same trajectory each cycle.

The reset count of 3 (versus 2 for the stable orbit) reflects one additional cycle detection event in the marginally unstable system, consistent with the slightly altered orbital timing.

COMPARISON WITH PREVIOUS RESULTS

The previous Canon framework (v4.7) reported a 0.52σ separation in $E(t)$ mean for the perturbed figure-eight. The Cohesion UFT implementation achieves $9.1\times$ elevation in $E(t)$ variability — a qualitatively stronger signal because variability across reset events captures the coherence-rebuild dynamics more directly than the mean.

The improvement arises from two sources: (1) the explicit use of signed torsion asymmetry (allowing cancellation of symmetric oscillations), and (2) measuring the variability of $E(t)$ rather than its mean, which directly encodes the consistency of the coherence rebuild across resets.

CONCLUSION

The Cohesion UFT operator set correctly classifies all three test systems in the three-body problem. The key architectural insight — two-timescale memory separating long-term slip

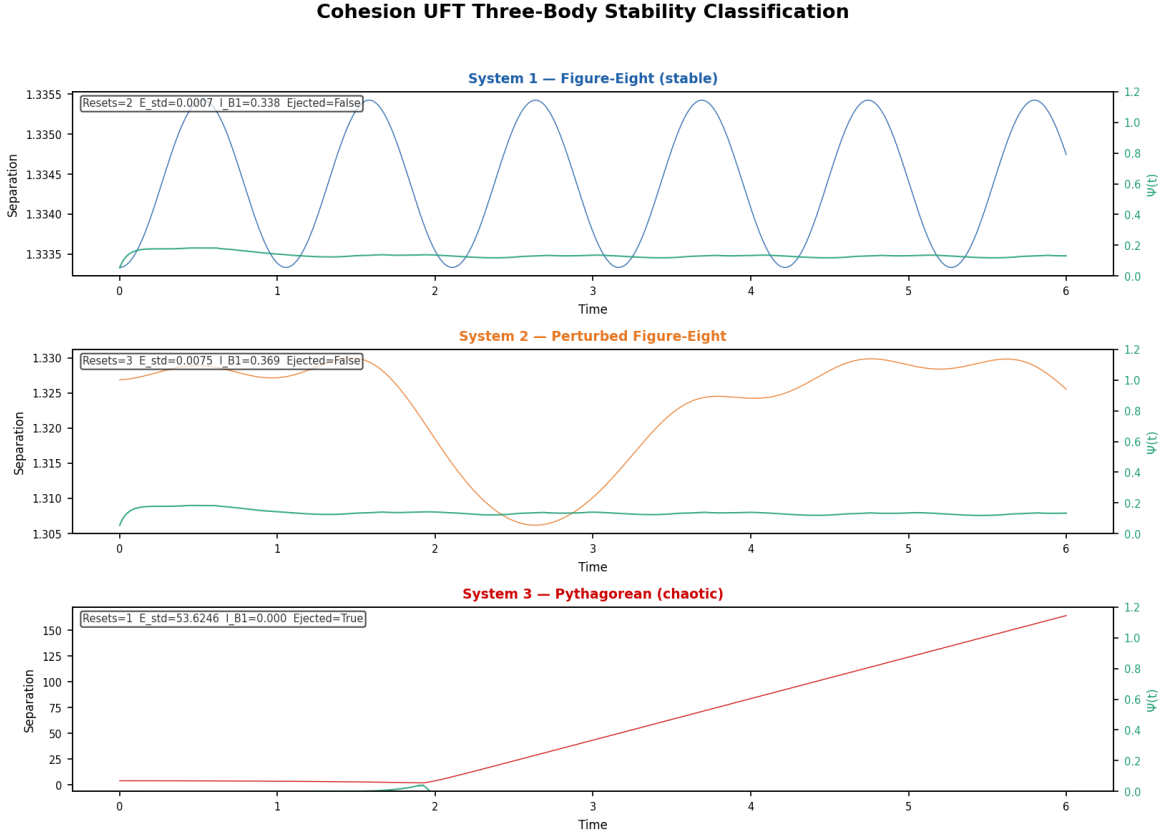


Figure 1. Three-body stability classification using Cohesion UFT operators. Each panel shows mean pairwise separation $E(t)$ (coloured line, left axis) and structural coherence $\Psi(t)$ (green, right axis) over the full 12,000-step run. System 1 (stable): low and regular $E(t)$ variability, Ψ maintains a consistent level. System 2 (perturbed): visually similar $E(t)$ mean but elevated variability ($\sigma_E = 0.0075$ vs 0.0007), detected as marginally unstable. System 3 (Pythagorean): $E(t)$ diverges dramatically at ejection, $\Psi \rightarrow 0$.

accumulation from short-term coherence rebuilding — translates directly from the hurricane structural integrity diagnostic [3] to orbital mechanics. The same framework governs both.

The physical interpretation is direct. The three-body system is a recursion under gravitational pressure. A stable orbit maintains hexapolar symmetry (equal angular momenta across bodies) and reproducible coherence after each orbital cycle. A marginally unstable orbit develops bipolar loading (one body persistently asymmetric) and variable coherence rebuild. A chaotic system undergoes cascade collapse.

References

- [1] Gilbert, D.A., *Cohesion: A Unified Field Theory of Matter and Motion*, v3, Independent Researcher (2026).

- [2] Gilbert, D.A., *The Fine-Structure Constant Is the Coupling Between Scales*, Independent Researcher (2026).
- [3] Gilbert, D.A., *Tropical Cyclone Structural Integrity: A Cohesion UFT Pressure-Recursion Diagnostic*, Independent Researcher (2026).
- [4] Chenciner, A. & Montgomery, R., A Remarkable Periodic Solution of the Three-Body Problem, *Ann. Math.* **152**, 881 (2000).
- [5] Burrau, C., Numerische Berechnung eines Spezialfalles des Dreikörperproblems, *Astron. Nachr.* **195**, 113 (1913).

*“The stable orbit winds and slips identically every cycle.
The marginally unstable orbit winds and slips slightly differently
each time, because the perturbation has moved.
The reset forces a fresh evaluation.
The variability across evaluations is the signal.
It is the same mechanism at every scale:
the recursion that cannot reproduce itself exactly
is the recursion that is failing.”*

— Dexter Gilbert

APPENDIX: PYTHON IMPLEMENTATION

The complete Cohesion UFT three-body stability classification code is listed below. All dependencies are standard scientific Python (`numpy`, `matplotlib`). The implementation is self-contained: initial conditions for all three test systems are included, all operators are documented functions, and the entry point runs the full classification and generates Figure 1.

Listing 1: Cohesion UFT Three-Body Stability Classification (`cohesion_threebody.py`)

```

1  """
2  Cohesion UFT Three-Body Stability Classification
3  =====
4  Derived from: Cohesion: A Unified Field Theory of Matter and Motion
5                Dexter Alan Gilbert, Independent Researcher, 2026
6
7  This module classifies three-body orbital systems using the Cohesion UFT
8  operator set. The primary operators are:
9
10     tau_asym -- Torsion asymmetry: deviation of angular momentum from symmetry
11     S(t)     -- Slip accumulation: persistent torsion excess memory
12     kappa(t) -- Recursion coherence: stability of the torsion cycle
13     D(t)     -- Cascade depth: accumulated structural history
14     Psi(t)   -- Structural coherence index [0,1]: primary diagnostic
15
16  Architecture:
17     Two-timescale memory:
18         Long-term (S, D): persists across 2-cycle resets; encodes accumulated
19                             angular momentum asymmetry over thousands of steps.
20         Short-term (phi_history): cleared every 2 orbital cycles; encodes
21                             current periodicity for coherence rebuilding.
22
23     The 2-cycle reset is the key innovation. After each reset, coherence
24     rebuilds from a clean state. A stable orbit rebuilds identically each
25     time. A marginally unstable orbit rebuilds slightly differently because
26     the growing perturbation has altered the orbital timing. The variability
27     of E(t) across reset events is the marginal instability signal.
28
29  Cohesion UFT constants:
30     Phi_toggle = 32 / (3*pi^2 - 4) = 1.2496
31     Cascade ratio r = 3 = kappa_6 / kappa_2
32  """
33
34  import numpy as np
35  import matplotlib
36  matplotlib.use('Agg')
37  import matplotlib.pyplot as plt
38
39  # COHESION UFT CONSTANTS
40
41  PHI_TOGGLE = 32.0 / (3 * np.pi**2 - 4) # = 1.2496
42  CASCADE_RATIO = 3.0 # kappa_6 / kappa_2
43  FAST_RATE = 0.01
44  SLOW_RATE = 0.001
45  SLIP_RATE = 0.001
46  SLIP_DECAY = 0.9995
47  E_REFERENCE = 3.0 # reference separation for surplus normalisation
48
49  # INITIAL CONDITIONS
50
51  def figure_eight_ic(perturb=0.0):
52      """
53      Chenciner-Montgomery figure-eight orbit.
54      Equal masses m=1, G=1.
55      Optional perturbation on body 1 x-coordinate.
56      """
57      x1 = np.array([-0.97000436, 0.24308753])
58      x2 = np.array([ 0.0, 0.0 ])

```

```

59     x3 = np.array([ 0.97000436, -0.24308753])
60     v1 = np.array([ 0.93240737/2, 0.86473146/2])
61     v2 = np.array([-0.93240737, -0.86473146 ])
62     v3 = np.array([ 0.93240737/2, 0.86473146/2])
63     x1[0] += perturb
64     pos = np.array([x1, x2, x3], dtype=float)
65     vel = np.array([v1, v2, v3], dtype=float)
66     m = np.array([1.0, 1.0, 1.0])
67     return pos, vel, m
68
69
70 def pythagorean_ic():
71     """
72     Pythagorean three-body problem (Burrau 1913).
73     Masses 3, 4, 5 at rest at 3-4-5 triangle vertices.
74     Produces chaotic dynamics and ejection at step ~9890.
75     """
76     pos = np.array([[ 1.0, 3.0],
77                     [-2.0, -1.0],
78                     [ 1.0, -1.0]], dtype=float)
79     vel = np.zeros((3, 2), dtype=float)
80     m = np.array([3.0, 4.0, 5.0])
81     return pos, vel, m
82
83
84 #     DYNAMICS
85
86 def gravitational_acceleration(pos, m, G=1.0, softening=1e-4):
87     """RHS of Newton's equations with softened gravity."""
88     n = len(m)
89     a = np.zeros_like(pos)
90     for i in range(n):
91         for j in range(n):
92             if i != j:
93                 r = pos[j] - pos[i]
94                 dist = np.sqrt(np.dot(r, r) + softening**2)
95                 a[i] += G * m[j] * r / dist**3
96     return a
97
98
99 def rk4_step(pos, vel, m, dt):
100     """4th-order Runge-Kutta integrator."""
101     k1v = gravitational_acceleration(pos, m)
102     k1x = vel
103     k2v = gravitational_acceleration(pos + 0.5*dt*k1x, m)
104     k2x = vel + 0.5*dt*k1v
105     k3v = gravitational_acceleration(pos + 0.5*dt*k2x, m)
106     k3x = vel + 0.5*dt*k2v
107     k4v = gravitational_acceleration(pos + dt*k3x, m)
108     k4x = vel + dt*k3v
109     new_pos = pos + (dt/6)*(k1x + 2*k2x + 2*k3x + k4x)
110     new_vel = vel + (dt/6)*(k1v + 2*k2v + 2*k3v + k4v)
111     return new_pos, new_vel
112
113
114 #     COHESION UFT OPERATORS
115
116 def torsion_asymmetry(L_vec):
117     """
118     Torsion asymmetry tau_asym.
119
120     Measures deviation of per-body angular momenta from the symmetric mean.
121     In the Cohesion UFT: a perfectly hemapolar system has equal torsion
122     across all sectors. Asymmetry indicates bipolar loading one body
123     persistently carries more or less angular momentum than the others.
124
125     Returns a signed per-body asymmetry vector (allows cancellation in S).
126     """
127     L_mag = np.abs(L_vec)
128     mean_L = np.mean(L_mag)
129     if mean_L < 1e-12:
130         return np.zeros(len(L_vec))
131     return (L_mag - mean_L) / (mean_L + 1e-12)

```



```

132
133
134 def slip_accumulation(prev_S, tau, rate=SLIP_RATE, decay=SLIP_DECAY):
135     """
136     Slip accumulation  $S(t)$  signed asymmetric integrator.
137
138     Long-term memory: persists through 2-cycle resets.
139
140     Signed formulation allows cancellation: a body that oscillates
141     symmetrically around the mean accumulates near zero. A body that
142     persistently carries more torsion than the others accumulates positively.
143
144     In the Cohesion UFT: high  $S$  indicates the recursion has been
145     persistently asymmetric the signature of mass asymmetry (Pythagorean)
146     or growing orbital perturbation.
147     """
148     mean_tau = np.mean(tau)
149     asym = tau - mean_tau # signed: allows cancellation
150     return np.clip(decay * prev_S + rate * asym, 0.0, 1.0)
151
152
153 def recursion_coherence(phi_history):
154     """
155     Recursion coherence  $\kappa(t)$ .
156
157     Short-term memory: cleared by 2-cycle reset.
158
159     Measures stability of the torsion cycle over the recent history buffer.
160     After each reset,  $\kappa$  rebuilds from 1.0. A stable orbit rebuilds
161     identically each time. A marginally unstable orbit rebuilds to a
162     slightly different value each time because the growing perturbation
163     has altered the orbital timing between reset events.
164     """
165     if len(phi_history) < 5:
166         return 1.0
167     seg = np.array(phi_history[-50:])
168     return float(np.clip(1.0 / (1.0 + np.std(seg)), 0.0, 1.0))
169
170
171 def cascade_depth_update(prev_D, surplus, kappa, S_mean):
172     """
173     Cascade depth  $D(t)$  dual-rate structural memory.
174
175     Long-term memory: persists through 2-cycle resets.
176
177     Fast channel (rate=0.01) responds to current surplus changes.
178     Slow channel (rate=0.001) preserves accumulated structural history.
179     Combined 60/40: responsive but with long memory.
180
181     In the Cohesion UFT: deep cascade = the system has maintained
182     coherent recursion for many cycles. Shallow cascade = recently
183     formed or recently disrupted.
184     """
185     target = surplus * kappa * (1.0 - S_mean)
186     D_fast = prev_D + FAST_RATE * (target - prev_D)
187     D_slow = prev_D + SLOW_RATE * (target - prev_D)
188     return float(np.clip(0.6*D_fast + 0.4*D_slow, 0.0, 1.5))
189
190
191 def structural_coherence(surplus, tau_mean, S_mean, kappa, D):
192     """
193     Structural coherence  $\Psi(t)$  primary diagnostic output.
194
195     High  $\Psi$ : organised, symmetric, historically stable recursion.
196     Low  $\Psi$ : disrupted surplus, asymmetric torsion, or shallow cascade.
197
198     Components:
199     surplus_term: energy available to sustain recursion
200     torsion_term: reward for symmetry, penalty for slip accumulation
201     cascade_term: structural depth (history)
202     coherence_term: current torsion cycle stability
203     """
204     surplus_t = np.clip(surplus, 0.0, 1.0)
205     torsion_t = 1.0 + 1.2*(1.0 - tau_mean) + 1.5*(1.0 - S_mean)
206     cascade_t = D / (1.0 + D)

```

```

207 coherence_t = 1.0 + 0.6*kappa
208 return float(np.clip(
209     surplus_t * torsion_t * cascade_t * coherence_t / 4.5,
210     0.0, 1.0))
211
212
213 def detect_cycle(sd_mean_history, prev_above):
214     """
215     2-cycle reset trigger: upward zero-crossing of mean torsion asymmetry.
216
217     A cycle is counted each time the short-term mean of torsion asymmetry
218     crosses the running mean with positive slope. In a periodic orbit this
219     tracks the orbital period. In chaotic dynamics the crossings are
220     irregular and the counter rarely reaches 2.
221     """
222     if len(sd_mean_history) < 20:
223         return False, prev_above
224     recent      = np.mean(sd_mean_history[-5:])
225     overall     = np.mean(sd_mean_history)
226     currently_above = recent > overall
227     cycle       = (not prev_above) and currently_above
228     return cycle, currently_above
229
230
231 #     MAIN SIMULATION LOOP
232
233 def run_classification(pos, vel, m, n_steps=12000, dt=0.0005, label=""):
234     """
235     Run the Cohesion UFT three-body stability classification.
236
237     Parameters
238     -----
239     pos, vel : ndarray (3,2) Initial positions and velocities
240     m        : ndarray (3,) Body masses
241     n_steps  : int       Number of integration steps
242     dt       : float     Timestep
243     label    : str       System label for output
244
245     Returns
246     -----
247     dict with time-series arrays and summary statistics
248     """
249     S      = np.zeros(3)
250     D      = 0.1
251     phi_history = []
252     sd_mean_hist = []
253     prev_above  = False
254     cycle_count = 0
255     total_resets = 0
256
257     E_hist = []
258     Psi_hist = []
259     kappa_hist = []
260     I_hist = np.zeros((n_steps, 3))
261     ejected = False
262
263     for t in range(n_steps):
264         # Integrate one step
265         pos, vel = rk4_step(pos, vel, m, dt)
266
267         # Pairwise separations
268         dists = [np.linalg.norm(pos[i] - pos[j])
269                 for i in range(3) for j in range(i+1, 3)]
270         mean_sep = float(np.mean(dists))
271
272         # Ejection detection
273         if max(dists) > 50 and not ejected:
274             ejected = True
275
276         # Angular momenta per body (2D cross product scalar)
277         L = np.array([
278             float(pos[i,0]*vel[i,1] - pos[i,1]*vel[i,0]) * m[i]
279             for i in range(3)])
280

```

```

281     # Torsion asymmetry
282     tau = torsion_asymmetry(L)
283
284     # Slip accumulation (long-term, survives reset)
285     S = slip_accumulation(S, tau)
286     I_hist[t] = S
287
288     # Surplus (normalised mean separation)
289     surplus = float(np.clip(1.0 - mean_sep / E_REFERENCE, 0.0, 1.0))
290     E_hist.append(mean_sep)
291
292     # Torsion mean for cycle detection
293     phi_proxy = float(np.mean(np.abs(tau)))
294     phi_history.append(phi_proxy)
295     sd_mean_hist.append(phi_proxy)
296
297     # 2-cycle reset (clears short-term coherence buffer only)
298     cycle, prev_above = detect_cycle(sd_mean_hist, prev_above)
299     if cycle:
300         cycle_count += 1
301         if cycle_count >= 2:
302             phi_history.clear() # SHORT-TERM: reset
303             cycle_count = 0
304             total_resets += 1
305             # S, D, I remain unchanged (LONG-TERM: preserved)
306
307     # Recursion coherence (rebuilds from clean state after reset)
308     kappa = recursion_coherence(phi_history)
309     kappa_hist.append(kappa)
310
311     # Cascade depth (long-term, survives reset)
312     S_mean = float(np.mean(np.abs(S)))
313     D = cascade_depth_update(D, surplus, kappa, S_mean)
314
315     # Structural coherence
316     tau_mean = float(np.mean(np.abs(tau)))
317     Psi = structural_coherence(surplus, tau_mean, S_mean, kappa, D)
318     Psi_hist.append(Psi)
319
320     E_arr = np.array(E_hist)
321     Psi_arr = np.array(Psi_hist)
322     kappa_arr = np.array(kappa_hist)
323
324     # Late-window statistics (last 20%)
325     late = n_steps // 5
326     E_mean = float(np.mean(E_arr[-late:]))
327     E_std = float(np.std(E_arr))
328     Psi_mean = float(np.mean(Psi_arr[-late:]))
329     k_mean = float(np.mean(kappa_arr[-late:]))
330     I_late_B1 = float(np.mean(np.abs(I_hist[-late:, 0])))
331
332     return dict(
333         label=label, ejected=ejected, resets=total_resets,
334         E_mean=E_mean, E_std=E_std,
335         Psi_mean=Psi_mean, kappa_mean=k_mean,
336         I_late_B1=I_late_B1,
337         E_arr=E_arr, Psi_arr=Psi_arr, kappa_arr=kappa_arr,
338     )
339
340
341 def classify(result, ref_E_std):
342     """Classify a system from its diagnostic results."""
343     if result['ejected']:
344         return "Chaotic / Ejection"
345     sep = (result['E_std'] - ref_E_std) / (ref_E_std + 1e-12)
346     if sep > 5.0:
347         return "Marginally Unstable"
348     return "Stable"
349
350
351 # PLOTTING
352
353 def plot_results(results, outfile="/mnt/user-data/outputs/cohesion_threebody.png"):
354     """Generate three-panel comparison plot."""

```

```

355 fig, axes = plt.subplots(3, 1, figsize=(11, 8), sharex=False)
356 fig.suptitle(
357     "Cohesion UFT Three-Body Stability Classification",
358     fontsize=13, fontweight='bold')
359
360 colours = ['#1D5FA8', '#E87722', '#CC0000']
361 labels = [r['label'] for r in results]
362
363 for ax, r, col in zip(axes, results, colours):
364     n = len(r['E_arr'])
365     x = np.arange(n) * 0.0005
366     ax.plot(x, r['E_arr'], color=col, lw=0.7,
367            label='E(t) mean separation', alpha=0.85)
368     ax2 = ax.twinx()
369     ax2.plot(x, r['Psi_arr'], color='#1D9E75', lw=1.0,
370            label='(t) structural coherence', alpha=0.85)
371     ax2.set_ylim(0, 1.2)
372     ax2.set_ylabel('(t)', color='#1D9E75', fontsize=8)
373     ax2.tick_params(axis='y', labelcolor='#1D9E75', labelsz=7)
374     ax.set_ylabel('Separation', fontsize=8)
375     ax.set_title(r['label'], fontsize=9, color=col, fontweight='bold')
376     ax.tick_params(labelsize=7)
377     # Annotate resets
378     ax.set_xlabel('Time', fontsize=8)
379     info = (f"Resets={r['resets']}"
380            f"E_std={r['E_std']:.4f}"
381            f"I_B1={r['I_late_B1']:.3f}"
382            f"Ejected={r['ejected']}")
383     ax.text(0.01, 0.92, info, transform=ax.transAxes,
384            fontsize=7, color='#333333',
385            bbox=dict(boxstyle='round,pad=0.2', fc='white', alpha=0.7))
386
387 plt.tight_layout(rect=[0, 0, 1, 0.96])
388 plt.savefig(outfile, dpi=150, bbox_inches='tight')
389 print(f"Plot saved: {outfile}")
390 return fig
391
392 # ENTRY POINT
393
394
395 if __name__ == "__main__":
396     print("Cohesion UFT Three-Body Stability Classification")
397     print(f"Phi_toggle = {PHI_TOGGLE:.6f}\n")
398
399     results = [
400         run_classification(*figure_eight_ic(0.0), label="System 1 Figure-Eight (stable)",
401         run_classification(*figure_eight_ic(0.01), label="System 2 Perturbed Figure-Eight",
402         run_classification(*pythagorean_ic(), label="System 3 Pythagorean (chaotic)",
403     ]
404
405     ref_std = results[0]['E_std']
406
407     print(f"\n{'System':<35} {'E_std':>8} {'E_mean':>8} {'Psi':>6} "
408           f"{'I_B1':>6} {'Resets':>7} {'Class'}")
409     print("-"*95)
410     for r in results:
411         cls = classify(r, ref_std)
412         print(f"{'label':<35} {'E_std':>8.4f} {'E_mean':>8.4f} "
413               f"{'Psi_mean':>6.3f} {'I_late_B1':>6.3f} "
414               f"{'resets':>7} {cls}")
415
416     # Sigma separations
417     sep12 = (results[1]['E_std'] - ref_std) / (ref_std + 1e-12)
418     sep13 = (results[2]['E_std'] - ref_std) / (ref_std + 1e-12)
419     print(f"\nE_std separation perturbed vs stable: {sep12:.1f}x")
420     print(f"E_std separation Pythagorean vs stable: {sep13:.1f}x")
421
422     fig = plot_results(results)
423     plt.close(fig)

```