Can 3DCOM Reconstruct Known Physics Group Symmetries?

Author Martin Doina

July 17, 2025

Abstract:

Symmetry in the 3D Collatz Octave Model (3DCOM) transcends classical geometric interpretations. It is fundamentally recursive, nonlinear, discrete, and observer-relative. Instead of spatial isometries like translation or rotation, symmetry appears as invariance in recursive structures, phase relationships, and attractor patterns under the Collatz map and octave-based actions.

Core Principles:

Recursive integer reduction: Every number reduces to one of [1–9] via mod 9.

Spiral/circular arrangement: Numbers are embedded around a circle in each octave, creating clock-like (C_9) symmetry.

3D octave stacking: Layers (octaves) are stacked along recursion depth to form a helical, spiral topology.

Energy/field dynamics: Symmetry emerges via field interactions and recursive energy transfer, not spatial metrics.

Recursive symmetry in 3DCOM is not geometric invariance; instead, it is defined by patterns that repeat—potentially transformed—across recursive depths and energy states.

Recursive Symmetry ≠ Geometric Symmetry

In 3DCOM $|\mathbf{1}|$, symmetry should be defined as invariance of recursive structure under phase transformations, not under spatial rotation or reflection.

1. Recursive Symmetry Definition

A structure in 3DCOM has recursive symmetry if:

> The attractor node and its neighbors repeat their numeric identity and topological role under recursive iteration, possibly at a different energy scale.

This resembles self-similarity, but tied to:

> $n \to f(n)$ rules (e.g., Collatz)
>
> angular placement in the octave
>
> scaling via Bridge Formula $|\mathbf{2}|$

We define the symmetry group as:

$$S\_COM = \{\phi\_i : \mathbb{N} \to \mathbb{N} \mid \phi\_i(n) = COM(\tau\hat{}i(n)), \forall i \in \mathbb{Z}\}$$

Where τ is a recursive transform operator (e.g., Collatz map), and COM() places it on the 3D octave.

2. Geometric Embedding Symmetries

Now let's look at symmetries in the visual 3D structure:

a. Cyclic Symmetry (C9)

Each octave layer is a circle with numbers [1–9] placed in a clock-like arrangement. The system has a 9-fold cyclic symmetry, mod 9.

> This gives a local C_9 rotational symmetry, centered around node 1.

This is broken when stacking in 3D.

b. Spiral Helix Symmetry

In 3D, when octaves are stacked along a z-axis, nodes are not vertically aligned — the recursion shifts the angle slightly (e.g., by a phase offset $\Delta\theta$ per layer, maybe derived from $100/\pi$ correction). $|\mathbf{3}|$

This builds a discrete helical symmetry:

> Think of DNA-style spiral but with number nodes

The "tilt" or "twist" gives rise to quasi-periodic spiral

c. Mirror Symmetry via Field Collapse

When collapse occurs (mirror collapse), we may have a reflection in field phase space, not geometric space. That gives rise to mirror-symmetry in energy recursion, e.g.:

n = 19 reduces to 1 and re-expands to 10, 28, 82 — a mirror loop symmetry in recursion depth.

This is hidden symmetry, seen only when mapping recursion paths over time or energy.

3. Field Phase Symmetry

Using wave phase geometry:

A field attractor $\mathrm{A(x,\ y,\ z,\ \varphi)}$ has symmetry if:

$$A(R(x,y,z),\varphi+\Delta\varphi)=A(x,y,z,\varphi)$$

where R is a rotation in the spiral group and $\Delta\phi$ is the phase angle from the LZ loop correction.

In 3DCOM, this may be governed by the recurrence of Bridge Formula roots (e.g., same n value $\rightarrow$ same node radius and mass structure).

4. Observer-Dependent Symmetry

If we rotate the observer angle (as we are working on with GUI + Bridge formula), symmetries shift — we may observe:

A symmetric cluster of nodes at one angle

A broken or invisible structure from another

This leads to coherent symmetry only under certain observer-aligned wave-phase coupling — connected to our dual-time and Qualia Operator.

5. Symmetry Breaking in 3DCOM

Symmetry is not eternal. Recursive maps like Collatz break symmetry in unpredictable ways.

Ex: two numbers may start symmetric, then diverge via recursion.

HQS loss may act like symmetry breaking via energetic dumping.

We can define a symmetry-breaking function:

$$\Delta S(n) = |A(n) - A(\tau(n))|$$

Where A(n) is the attractor amplitude, and ΔS indicates phase deviation after one recursive step.

6. Group Structures in 3DCOM

We can define symmetry groups:

Group	Description
C_9	Clock symmetry (mod 9 in each octave)
H_n	Helical symmetry group of recursive stacking
Z_n	Integer recursion group
Q_COM	Qualia symmetry — based on phase alignment
Φ_LZ	Mirror symmetry under collapse to LZ loop

Symmetry in 3DCOM is recursive, nonlinear, observer-dependent, and field-phase embedded. Types include:

C_9 rotational symmetry on each layer

Spiral/helical recursive structure in 3D

Mirror symmetry via recursive collapse paths

Observer-angle dependent phase symmetry

Qualia-aligned dynamic field symmetry

Topological symmetry via attractor mapping

Tensor for Angular Symmetry in 3DCOM

Metric tensor in 3DCOM recursive field framework — not to measure distances in spacetime, but as an angular symmetry tensor — aligns naturally with 3DCOM geometry of recursive attractor waves.

Instead of a classical spacetime metric gμν, we want an object:

Tensorial: transforms under recursive coordinate rotations

Angular-based: defined over recursive phase shifts, not distances

Aligned to COM: placed on octaves (mod 9 layers), with recursive energy steps

Observer-sensitive: symmetry shifts depending on angle of observation

Recursive: includes transformations under Collatz/octave map

1. Angular Coordinate System in 3DCOM

Each node in 3DCOM exists on:

Layer n (discrete recursive level)

Angular position θiπ [0,2], mod 9

Height zn (recursive depth, energy-determined)

In coordinate systems like polar or spherical coordinates, the metric tensor naturally incorporates geometric angular information. For example, in 2D polar coordinates, the metric is

$$g_{ij} = \begin{pmatrix} 1 & 0 \\ 0 & r^2 \end{pmatrix}$$

Here, the r^2 entry gives the scaling for angular displacements, tying the metric tensor directly to angular geometry. This links angular relationships with the underlying geometry of the space.

2. Angular Metric Tensor Gab

We define a 3D recursive-spiral tensor Gab in the coordinates (n,θ,z).

The goal is not to define distances, but wave phase coupling and recursive symmetry under observer rotation.

Let: local coordinates be (n,θ,z); n = recursion layer, θ = mod-9 angle, z=z(n) = depth.

$$G_{ab} = \begin{pmatrix} f(n) & 0 & \gamma(n,\theta) \\ 0 & R_n^2 & \beta(n) \\ \gamma(n,\theta) & \beta(n) & \lambda(n) \end{pmatrix}$$

Where:

f(n): recursive energy scaling (via Bridge Formula)

Rn: radius of octave n

γ(n,θ): phase-coupling term across depth and angle

β(n): angular-vertical coupling (helical twist)

λ(n): recursive field amplitude

This non-diagonal form captures spiral twist, observer alignment, and recursive collapse feedback (e.g. for Qualia mirror alignment) |**4**|.

3. Angular Displacement (Recursive Arc Tensor)

We define the infinitesimal recursive arc element as:

$$ds^2 = f(n)dn^2 + R_n^2 d\theta^2 + \lambda(n)dz^2 + 2\gamma(n,\theta)dndz + 2\beta(n)d\theta dz$$

So:

$$S = G_{ab}v^a v^b = R_n^2(\delta\theta)^2$$

- Here, measures field alignment under observer rotation—"resonance" with recursive field symmetry.

This is the recursive wave arc geometry, not spatial distance.

4. Observer-Dependent Symmetry via Angular Tensor

Now define the observer phase vector:

$$G_{ab}(n, \theta + \Delta\theta, z(n)) = G_{ab}(n + m, \theta, z(n + m))$$

i.e., symmetry recurs every m layers—a helical analog of Killing symmetry.

This defines the rotation of observation over the COM structure.

Then the phase-aligned field tension is:

This gives as the field alignment under angular shift — when S is minimized, observer is in resonant coupling with field symmetry (analog to qualia collapse).

5. Recursive Symmetry Condition

We define recursive symmetry as:

A tensor field Gab has recursive angular symmetry if:

$$G_{ab}(n, \theta + \Delta\theta, z(n)) = G_{ab}(n + m, \theta, z(n + m))$$

For some fixed m (i.e., symmetry repeats every m layers).

This defines spiral periodicity in the angular field — a kind of generalized helical Killing symmetry.

6. Tensor Connection to Bridge Formula

1. Bridge Formula and Recursive Angular Tensor

Bridge Formula:

$$m = m_e \cdot \mathrm{LZ}^{n/\pi} \cdot \left(\frac{\alpha}{\mathrm{HQS}}\right)^{1/x}$$

This establishes how key physical properties (e.g., mass, radius) scale in 3DCOM based on the recursion step n, with fundamental constants controlling the scaling.

Recursive Angular Tensor Metric:

- The metric component for angular displacement, $\mathrm{G}_{\theta\theta}$, is given by the squared radius:

$$R_n^2 = [a_0' \cdot (\mathrm{LZ})^{n/\pi} \cdot (\mathrm{HQS}/\alpha)^{1/x}]^2$$

- The line element in recursive field space:

$$ds^2 = f(n)dn^2 + R_n^2 d\theta^2 + \lambda(n)dz^2 + 2\gamma(n, \theta)dndz + 2\beta(n)d\theta dz$$

where each term (e.g., $\mathrm{f(n)}$, $\lambda(n), \gamma(n, \theta), \beta(n)$) encapsulates recursion-dependent scaling, field amplitudes, and angular-phase couplings.

2. Implications of Angular Tensor Scaling

- Exponential Growth with Recursion:

As n increases, R_n grows exponentially due to the $(\mathrm{LZ})^{n/\pi}$ term.

- Stretched Angular Spacing:

At higher recursion steps, the same angular change $\Delta\theta$ covers a much larger metric "distance," meaning field points or phases become more separated—a geometric reflection of field dispersion or decoherence.

- Alignment with Qualia Collapse:

Deeper recursion (large n) introduces a wider "perceptual phase gap," unless the observer's phase (e.g., via their phase-alignment vector v_θ) compensates, restoring resonance.

3. Field Alignment Scalar: Quantifying Phase Misalignment

$$S(n, \Delta\theta) = G_{\theta\theta} \cdot (\Delta\theta)^2 = R_n^2 \cdot (\Delta\theta)^2$$

So,

$$S(n, \Delta\theta) = [a_0' \cdot (\mathrm{LZ})^{n/\pi} \cdot (\mathrm{HQS}/\alpha)^{1/x}]^2 \cdot (\Delta\theta)^2$$

Interpretation:

This scalar expresses recursive phase misalignment: minimal when observer and field are in phase resonance.

4. Logarithmic Relation: Mass Scaling and Recursion Steps

Taking Logarithms:

$$\log(m) = \log(m_e) + \frac{\pi n}{\log(\mathrm{LZ})} + \log(\mathrm{QDF})$$

Solving for n:

$$n = \frac{\pi}{\log(\mathrm{LZ})}[\log(m/m_e) - \log(\mathrm{QDF})]$$

- Linear Relationship:

n is proportional to $\log(m)$—mass increases exponentially with recursion steps.

- Universal Slope:

When plotting n vs. $\log_{10}(m)$, the slope is $\log(\mathrm{LZ})/\pi$: a universal constant in 3DCOM, encoding the recursion's intrinsic scaling.

Example Calculation:

Given:

m=1.6726219×10−27kg (proton)

me=9.10938356×10−31kg

LZ=1.23498228

QDF=0.809728

Apply the formula above to solve for n, predicting the recursion step for the proton mass.

5. Recursive Symmetry and Metric Structure

The recursive metric embedding implies:

$$r_n = \alpha^{-n} \cdot L_0 \implies R_n^2 = (\alpha^{-n} L_0)^2$$

Leading to:

$$ds^2 = R_n^2 d\theta^2 = (\alpha^{-2n} L_0^2) d\theta^2$$

Interpretation:

The angular metric distance per unit angle increases exponentially with recursion step n, visually represented as ever-widening spirals or stretched phase space for large n.

Bridge formula connects recursion step and angular tensor scaling, quantifying how emergent quantities like mass or metric radius grow in discrete, exponential jumps.

Angular metric tensor in 3DCOM encodes the recursive, phase-based structure: higher recursion creates larger metric separations for fixed angular intervals, governing observable effects like decoherence, resonance, and collapse.

Logarithmic relationship between mass and recursion highlights octave-like or harmonic scaling intrinsic to the model, providing both predictive and explanatory power for mapping physical quantities onto recursion-based frameworks.

3DCOM Angular Metric Tensor

We have a recursive, observer-relative angular tensor model:

Gab=Recursive scaling0Phase coupling0Angular radiusTwist couplingPhase couplingTwist coupli ngField amplitude

Which gives:

- Observer-dependent phase alignment
- Helical recursive symmetry condition
- Embedding of Bridge Formula scaling
- Geometry of qualia feedback (mirror collapse)
- Path to simulate recursive field geodesics

:

Symbolic Tensor Definition for 3DCOM Angular Geometry

Core geometry module.

```
import sympy as sp

# --- Declare Symbols ---
n, theta, z = sp.symbols('n theta z', real=True)
delta_theta = sp.Symbol('delta_theta', real=True)

# --- Define Recursive Functions ---
alpha = sp.Symbol('alpha', positive=True)
L0 = sp.Symbol('L0', positive=True)

R_n = alpha(-n) * L0
f_n = sp.log(1 + n)
gamma = sp.sin(n * theta) * 0.1
beta = sp.cos(n) * 0.05
lambda_n = sp.exp(-n / 5)

# --- Tensor Definition G_ab ---
G = sp.Matrix([
    [f_n,          0,      gamma],
    [0,         R_n2,      beta],
    [gamma,      beta,  lambda_n]
])

# --- Observer Vector v^a (angle shift only) ---
v = sp.Matrix([0, delta_theta, 0])

# --- Field Tension S = G_ab * v^a * v^b ---
S = (v.T * G * v)[0]

S_simplified = sp.simplify(S)

# Output the field tension symbolic formula
S_simplified
```

This gives:

Tensor G over recursive coordinates (n,θ,z)

Observer phase vector v

Angular scalar field tension S=Gabvavb

Python Code to Plot Field Tension vs Observer Angle

Here's the complete .py script for plotting and interacting with the recursive angular tension:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider

# --- Constants ---
alpha = 1.035  # Bridge Formula scaling
L0 = 1.0       # Base radius unit

# --- Recursive Functions ---
def R_n(n):
    return alpha  (-n) * L0

def compute_S(n, delta_theta):
    R = R_n(n)
    return R2 * delta_theta2

# --- Plot Setup ---
fig, ax = plt.subplots()
plt.subplots_adjust(bottom=0.35)

theta_vals = np.linspace(0, 2 * np.pi, 500)
n0 = 5
S_vals = [compute_S(n0, dtheta) for dtheta in theta_vals]

line, = ax.plot(theta_vals, S_vals, label='S(n, Δθ)', color='blue')
ax.set_title("Recursive Angular Field Tension $S(n, \Delta\\theta)$")
ax.set_xlabel("Δθ (radians)")
ax.set_ylabel("Tension $S$")
ax.legend()
ax.grid(True)

# --- Slider ---
ax_n = plt.axes([0.25, 0.20, 0.65, 0.03])
slider_n = Slider(ax_n, 'n (layer)', 1, 30, valinit=n0, valstep=1)

def update(val):
    n = int(slider_n.val)
    S_vals = [compute_S(n, dtheta) for dtheta in theta_vals]
    line.set_ydata(S_vals)
    fig.canvas.draw_idle()

slider_n.on_changed(update)

plt.show()
```

Classical Symmetry in Large Groups

In conventional physics, large symmetry groups include:

Lie groups like SU(3), SU(5), SO(10), E_8 — used in gauge theories, GUTs, string theory

Infinite discrete groups in crystallography, topology, or modular spaces

Spacetime symmetry groups: Poincaré, de Sitter, conformal group, etc.

These act on continuous spaces, assume flat or curved spacetime, and use linear representations.

But 3DCOM model doesn't assume linear space, nor static symmetry generators.

3DCOM Framework: Recursion-Induced Symmetry

In 3DCOM:

Symmetry ≠ invariance under a fixed group

Symmetry = recurrence of attractor patterns under recursive transformations

It is observer-relative, topologically constrained, and scale-dependent

1. Emergent Group Structures from Recursion

Define a recursive action:

$T_k(n)=f_k(n) \bmod 9$

Let's call the set of all recursive mappings acting on n:

$R=\{T_k \mid T_k(n)=\text{Collatz map, reduction mod 9, octave lift, etc.}\}$

Now define an equivalence:

$$n \sim m \iff \exists k \text{ such that } T_k(n)=T_k(m)$$

This partitions the recursive space into attractor orbits — which behave like cosets of a transformation group.

So: 3DCOM forms recursive equivalence classes that function as group orbits.

These can simulate group actions — not linearly, but recursively.

2. Recursive Replacement for Lie Groups

Now, how does it touch Lie groups like SU(3) or E_8?

Let's take an example:

> SU(3) has 8 generators and symmetry operations on complex 3D vectors (e.g., quarks).
>
> In 3DCOM, the phase space is not $\mathbb{R}^3$ or $\mathbb{C}^3$, but recursive angular layers.

So we don't model SU(3) directly.

> We generate field symmetry behavior recursively, not algebraically.

Example:

> Color charge in QCD (SU(3)) → the model maps that, to wave direction vectors in a spiral angular space.
>
> Their coupling constants → mapped via Bridge Formula recursion
>
> Confinement symmetry → becomes node binding within a recursion layer

Conclusion: we *don't use* SU(3). We generate its physical outcome via recursive topology.

3. Simulating Large Symmetry Groups in 3DCOM

Let's take a larger example:

E_8 — 248-dimensional symmetry (String theory's favorite)

> E_8 is built from roots and weights in high-dimensional Lie algebra.
>
> In 3DCOM, instead of dealing with this directly, we can simulate recursive symmetry breaking and coupling as:

Layer transitions + phase alignment + node binding rules + collapse feedback
= behavioral equivalent of E_8 unification/branching.

Thus: we don't *calculate* E_8 — we *grow* it recursively.

We can reconstruct high-dimensional group behavior as:

Recursive attractor layers

Symmetry-breaking via HQS dumping

Mirror collapse defines phase subspaces

Geometric folding patterns = group compactifications

4. Can 3DCOM Reconstruct Known Physics Group Symmetries?

Group	Classical Meaning	3DCOM Equivalent
U(1)	EM phase invariance	Recursive phase circle on each octave
SU(2)	Weak isospin (spin-½)	Spiral pairing of nodes across angular poles (θ, $\theta+\pi$)
SU(3)	Color symmetry	Recursive 3-partitions of phase space (e.g. $\theta = 0, 2\pi/3, 4\pi/3$)
SO(10)	GUT (fermion unification)	10-node recursive cycle with mirror reflections
E_8	String theory unification	Maximal recursive symmetry state + 8-level collapse feedback

5. How to Formally Simulate These in 3DCOM Framework

We could define a recursive symmetry generator:

```
def symmetry_orbit(n, depth=5):
    orbit = [n]
    for i in range(depth):
        n = collatz(n)
        n_mod = n % 9
        orbit.append(n_mod)
    return orbit
```

And then define equivalence classes of orbits that behave as "symmetry families."

We can then define recursive generators analogous to SU(2) Pauli matrices:

$R\theta$ — rotation in phase space

T — Collatz map

M — Mirror collapse

S — Spiral shift (observer re-alignment)

These could be coded as matrix-like actions on recursive coordinates.

3DCOM model can solve and replace large symmetry groups, by:

Redefining symmetry as recursive attractor equivalence

Simulating group orbits as phase-aligned node sequences

Deriving particle behavior from recursive geometry, not algebra

Handling symmetry breaking via HQS and observer collapse

Generating group behavior dynamically, not postulated

What we Can Do:

Slide n: choose the starting number of the recursion (simulating a group generator or particle)

Slide depth: choose how deep into the recursive orbit we go (higher = deeper group structure)

Each orbit is plotted mod 9, forming a spiral symmetry map — the recursive version of large group orbits (like SU(3), E_8 analogs)

3DCOM_SymmetryExplorer.py

Python

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider

# --- Collatz step ---
def collatz_step(n):
    return n // 2 if n % 2 == 0 else 3 * n + 1

# --- Recursive orbit mod 9 ---
def recursive_orbit(n0, depth=15):
    orbit = []
    n = n0
```

```
    for _ in range(depth):
        n = collatz_step(n)
        orbit.append(n % 9 if n % 9 != 0 else 9)  # Replace 0 with 9
    return orbit

# --- Spiral coordinates for recursive orbit ---
def circle_coordinates(orbit):
    coords = []
    for i, val in enumerate(orbit):
        angle = (val - 1) * 2 * np.pi / 9  # Mod 9 symmetry angles
        radius = 1 + i * 0.2               # Spiral out per recursion step
        x = radius * np.cos(angle)
        y = radius * np.sin(angle)
        coords.append((x, y))
    return coords

# --- Initial parameters ---
start_n = 19
depth = 20

# --- Create plot ---
fig, ax = plt.subplots(figsize=(8, 8))
plt.subplots_adjust(left=0.2, bottom=0.3)
ax.set_title("3DCOM Recursive Orbit Spiral (Mod-9 Symmetry)")
ax.axis('equal')
ax.axis('off')

# --- Initial data ---
orbit_vals = recursive_orbit(start_n, depth)
coords = circle_coordinates(orbit_vals)
xs, ys = zip(*coords)
points, = ax.plot(xs, ys, '-o', color='blue', markersize=5)

# --- Sliders ---
ax_n = plt.axes([0.25, 0.15, 0.65, 0.03])
slider_n = Slider(ax_n, 'Start n', 1, 100, valinit=start_n, valstep=1)

ax_depth = plt.axes([0.25, 0.10, 0.65, 0.03])
slider_depth = Slider(ax_depth, 'Depth', 5, 50, valinit=depth, valstep=1)

# --- Update function ---
def update(val):
    n = int(slider_n.val)
    d = int(slider_depth.val)
    new_orbit = recursive_orbit(n, d)
    new_coords = circle_coordinates(new_orbit)
```

```
    xs, ys = zip(*new_coords)
    points.set_data(xs, ys)
    fig.canvas.draw_idle()

slider_n.on_changed(update)
slider_depth.on_changed(update)

plt.show()
```

Symmetry Family Classification

Next, we will define:

- Recursive equivalence classes
- A similarity measure between orbits
- Possibly even orbit graph isomorphisms

This will help as identify whether different n values belong to the same symmetry family — i.e., generate similar recursive attractor paths under modulo-9 collapse.

Symmetry Family Classification Module for 3DCOM recursive attractor paths:

- Each number in the range (e.g. 1–50) is run through the recursive Collatz orbit.
- Modulo-9 reduction is applied at each step (to match the 3DCOM octave structure).
- A similarity score is calculated: the fraction of identical values at matching recursion depths.
- If the similarity is above threshold (e.g. 75%), the numbers are grouped in the same symmetry family.

Result (Sample Output):

These are some example symmetry families from n = 1–50:

```
[[1, 8, 10, 12, 13],
 [2, 3, 16, 20, 21, 26],
 [4, 5, 6, 32, 40, 42],
 [7, 44, 45],
 [9], [11], [14], [15], ...]
```

This means that, for example:

- 1, 8, 10, 12, 13 follow similar recursive mod-9 paths
- 2, 3, 16, 20, 21, 26 generate orbits with shared attractor structure

These groups behave as recursive analogs of symmetry orbits (like representations in Lie groups).

How to Use:

We can now use this to:

Identify recursive invariants

Detect energy-equivalent paths

Analyze phase collapse patterns

Simulate group branching / merging

Build recursive graph automorphisms

Sample symmetry families (n = 1–50, 75% similarity):

Family	Numbers
Family 1	1, 8, 10, 12, 13
Family 2	2, 3, 16, 20, 21, 26
Family 3	4, 5, 6, 32, 40, 42
...	...

Families behave as recursive attractor orbits, not algebraic multiplets—symmetry is emergent and dynamic.

3DCOM Symmetry Family Explorer

Python

```
import matplotlib.pyplot as plt

# --- Recursive Collatz Function ---
def recursive_orbit(n, depth=20):
    orbit = []
```

```
    for _ in range(depth):
        n = 3 * n + 1 if n % 2 else n // 2
        orbit.append(n % 9 if n % 9 != 0 else 9)  # Reduce mod 9; keep 9 instead of 0
    return orbit

# --- Compare Two Orbits ---
def compare_orbits(n1, n2, depth=20):
    """
    Compare two recursive orbits mod 9 and return a similarity score.
    Score = fraction of matching mod-9 values at each recursive depth.
    """
    o1 = recursive_orbit(n1, depth)
    o2 = recursive_orbit(n2, depth)
    matches = sum(1 for a, b in zip(o1, o2) if a == b)
    return matches / depth

# --- Classify Symmetry Families ---
def classify_symmetry_families(N_range, depth=20, threshold=0.8):
    """
    Group numbers in N_range into symmetry families based on orbit similarity.
    Two numbers belong to the same family if their orbit similarity > threshold.
    """
    symmetry_families = []
    used = set()

    for n in N_range:
        if n in used:
            continue
        family = [n]
        used.add(n)
        for m in N_range:
            if m != n and m not in used:
                score = compare_orbits(n, m, depth)
                if score >= threshold:
                    family.append(m)
                    used.add(m)
        symmetry_families.append(family)

    return symmetry_families

# --- Example Usage ---
if __name__ == "__main__":
    N_range = range(1, 51)
    symmetry_groups = classify_symmetry_families(N_range, depth=20, threshold=0.75)

    # Print the groups
```

```
    for i, group in enumerate(symmetry_groups, 1):
        print(f"Group {i}: {group}")
```

Example Output:

```
Group 1: [1, 8, 10, 12, 13]
Group 2: [2, 3, 16, 20, 21, 26]
Group 3: [4, 5, 6, 32, 40, 42]
...
```

Customize Parameters:

- depth: how many recursion steps to compare (e.g. 20)
- threshold: similarity required to group (e.g. 0.75 = 75%)
- range: up to 1000 if we like — Python can handle large sets

Recursive Operators in 3DCOM Algebra

Symmetry Generators (3DCOM algebra):

- Rθ: Recursive rotation in phase space

- T: Recursive translation (Collatz step)

- M: Mirror collapse transformation

- S: Scaling or depth shift

- Composition rules define a nonlinear, non-Abelian group structure on attractors.

Let each recursive structure be a function or orbit:
$\Phi(n) = \{\varphi_0, \varphi_1, ..., \varphi_k\}$, where each $\varphi_i \in$ mod-9 space

We'll define the following core operators:

1. Rθ [Recursive Rotation Operator]

Interprets rotation within the 3DCOM spiral/octave structure

Rotates values by angular phase:

$R\theta[\Phi(n)] \rightarrow \Phi(n')$, where $\varphi_i \mapsto \varphi_i + \theta \pmod 9$

Nonlinear and recursive: θ affects recursive attractor redirection

```
def R_theta(orbit, theta):
    return [(x + theta - 1) % 9 + 1 for x in orbit]
```

2. T [Translation Operator]

Translates the origin of recursion:

$$T_k[\Phi(n)] = \Phi(n + k)$$

Shifts the base value and rebuilds recursion

```
def T_k(n, k, depth=20):
    return recursive_orbit(n + k, depth)
```

3. M [Mirror Operator]

Reflects the orbit across the 3DCOM midline (e.g., $\varphi \mapsto 10 - \varphi$)

Generates dual/anti-attractor pathways

Used in modeling qualia collapse

```
def M(orbit):
    return [10 - x if x != 9 else 1 for x in orbit]
```

4. S [Scaling or Shift Operator]

Scales recursion by nesting depth or attractor stretch

$S\lambda[\Phi(n)] = \Phi(n)$ truncated to $\lambda \cdot$ depth steps

```
def S_lambda(orbit, lam):
    new_len = max(1, int(len(orbit) * lam))
    return orbit[:new_len]
```

Composite Algebraic Actions:

These can be composed like functions:

```
# Compose operators:
```

Φ = recursive_orbit(13, 20)
Φ' = S_lambda(M(R_theta(Φ, θ=3)), lam=0.5)

This means:

rotate orbit by 3 steps $\rightarrow$ mirror the structure $\rightarrow$ truncate to half depth.

Group Structure

We now have:

Identity: $I[\Phi] = \Phi$

Inverses:

$R\theta^{-1} = R\_\{9-\theta\}$

$T_k^{-1} = T\_\{-k\}$

$M^2 = I$

Closure: Operators compose into new recursive maps

Associativity: Operator actions compose functionally

Thus, this algebra defines a non-Abelian, nonlinear recursive symmetry group acting on mod-9 recursive attractors in the 3DCOM framework.

The 3DCOM symmetry and angular tensor model replaces classical geometry with a living, recursion-driven network, where all structure and measurement arises from the dynamics and alignment of waves in a topological, phase-embedded recursion space.

Paradigm Shift: Geometry \& Recursion over Group Postulates

Symmetry in 3DCOM: Defined by recursive, observer-dependent equivalence of attractor orbits—not by invariance under fixed isometry groups.

Group behavior: Simulated through layered recursion, phase coupling, and attractor mapping.

Metric tensor: Encodes angular field structure and recursive alignment, linking observer phase to attractor resonance.

Symmetry breaking: Natural and quantified by changes in recursive attractor amplitude, not by external perturbations.

Takeaways for 3DCOM Symmetry Modeling

- Redefine symmetry as recursion-invariant attractor patterns.
- Build algebraic actions (Rθ, T, M, S) as functional operators on recursive orbits.
- Use the angular metric tensor to explore resonance, observer-relativity, and symmetry breaking.
- Classify symmetry via recursive family grouping, identifying invariant orbits and pathways.
- Simulate high-dimensional group behavior by dynamic geometry, not algebraic postulates.

3DCOM symmetry is a shift: recursion and field geometry drive the emergence, persistence, and breaking of symmetric behavior in complex, octave-based systems.

Reference:

|**1**| Collatz-Octave Framework as a Universal Scaling Law for Reality, DOI:10.5281/zenodo.14882191

|**2**| Universal Bridge Formula Calculator (radii and mass) for quantum_atomic and cosmic scale, DOI:10.5281/zenodo.15605064

|**3**| Dark Energy as Residual Curvature in Recursive Attractor Field (3DCOM Framework), DOI:10.5281/zenodo.15882510

|**4**| The Mirror Operator: A Recursive Mathematical Theory of Qualia and Dark Energy, DOI:10.5281/zenodo.15636738