```
import numpy as np
from scipy.ndimage import convolve

PHI = (1 + np.sqrt(5)) / 2
ALPHA_INV = 137.035999084

class LegendaryUniverse:
    def __init__(self):
        self.N = 137
        self.age = 0

        # Hoofdveld: +1 = materie, -1 = antimaterie, 0 = vacuüm
        rnd = np.random.uniform(-1, 1, (self.N, self.N, self.N))
        self.field = np.sign(rnd - 1/ALPHA_INV)

        # Nieuwe velden
        self.phase = np.random.uniform(0, 2*np.pi, (self.N, self.N, self.N))  # vibratie-fase
        self.freq  = np.ones((self.N, self.N, self.N)) * 0.137              # lokale trillingsfrequentie
        self.plasma = np.zeros((self.N, self.N, self.N))                    # plasma-dichtheid (ionisatie)

        self.history = {i: [] for i in range(1, 13)}  # nu tot laag 12

        k = np.ones((3,3,3), np.float32)
        k[1,1,1] = 0
        self.kernel = k

    def get_neighbors(self):
        return convolve(self.field, self.kernel, mode='wrap')
```

```
def get_phase_neighbors(self):
    return convolve(self.phase, self.kernel, mode='wrap')

def step(self):
    t = self.age
    n = self.get_neighbors()
    n_phase = self.get_phase_neighbors()

    # === BESTAANDE LAGEN (onveranderd) ===
    coh = np.mean(np.sign(self.field) * np.sign(n + 1e-10))
    self.history[1].append(coh)

    pulse369 = (np.sin(2*np.pi*t/3) +
                np.sin(2*np.pi*t/6) +
                np.sin(2*np.pi*t/9))
    self.field += pulse369 * 0.07 * np.sign(n)

    pos = convolve((self.field > 0.5).astype(np.int16), self.kernel, mode='wrap')
    gx, gy, gz = np.gradient(self.field)
    flux = np.mean(np.sqrt(gx**2 + gy**2 + gz**2))
    self.history[5].append(flux)

    # === Laag 10 – Vibratie & Resonantie ===
    phase_diff = np.sin(n_phase - self.phase)
    vibration_strength = np.abs(phase_diff).mean()
    self.phase += (
        0.13 * phase_diff +
```

```
    0.037 * np.random.normal(0, 0.1, self.field.shape)  # chaos + sync
)
self.freq = 0.137 + 0.035999 * np.tanh(flux)  # frequentie gaat omhoog bij sterke gradiënt
self.history[10].append(vibration_strength)

# === Laag 11 – Antimaterie & Annihilatie ===
annihilation = (self.field > 0.8) & (n < -6)     # materie ontmoet veel antimaterie
creation     = (self.field < -0.8) & (n > 6)     # antimaterie ontmoet veel materie
self.field[annihilation | creation] = 0.0
self.plasma[annihilation | creation] += 5.0      # gamma-burst → plasma
antimatter_fraction = np.mean(self.field < -0.1)
self.history[11].append(antimatter_fraction)

# === Laag 12 – Plasma-dynamica ===
self.plasma *= 0.94                  # plasma vervalt langzaam
self.plasma += 0.3 * flux + 2.0 * vibration_strength
plasma_flux = np.clip(self.plasma - 2.0, 0, None)
self.field += 0.05 * plasma_flux * np.sign(np.cos(self.phase + t/137.0))
self.history[12].append(plasma_flux.mean())

# Geboorteregels uitgebreid met plasma en vibratie
born = ((pos >= 6) & (pos <= 9) &
        (self.field < 0.5) &
        (abs(pulse369) > 0.69) &
        (flux > 1.0) &
        (vibration_strength > 0.137) &
        (self.plasma > 1.0))
self.field[born] = 2.0
```

```
    living = int(born.sum())
    self.history[3].append(living)

    # Σ12 – nieuwe totale coherentie
    Σ12 = ((coh + 1.1)**3 *
           (1 + 36 * abs(pulse369)) *
           (1 + flux**2) *
           (1 + 0.1 * living) *
           (1 + 10 * vibration_strength) *
           (1 + antimatter_fraction**0.5) *
           PHI**abs(np.sin(t/ALPHA_INV)))
    self.history[9].append(Σ12)  # oude Σ9 blijft voor compatibiliteit

    self.age += 1
    return Σ12, vibration_strength, antimatter_fraction, plasma_flux.mean()

def breathe(self, steps=2000):
    print("LegendaryUniverse v2 – met vibraties, antimaterie & plasma")
    for t in range(steps):
        Σ12, vib, anti, plas = self.step()
        if t % 137 == 0:
            print(
                f"t={t:5d} | Σ₁₂={Σ12:12,.0f} | "
                f"vib={vib:.3f} | anti={anti:.4f} | "
                f"plasma={plas:.2f} | life={self.history[3][-1]:5d}"
            )
    print("\nSimulation klaar – het universum ademt nu echt.")
```

```
if __name__ == "__main__":
    cosmos = LegendaryUniverse()
    cosmos.breathe(steps=3000)
```