AETHERMOORE RESEARCH PACKET

GeoSeed Orbital Model

A deterministic geometry analogy connecting Sacred Tongues, orbital shells, and hyperbolic radial spacing.

geometry orbital sacred-tongues hyperbolic code

> **Abstract.** This note preserves an orbital pattern as a GeoSeed model: six tongues as shells, a phi ladder as hyperbolic radial depth, and uniform shell gaps as a deterministic structural invariant.

> **Claim boundary.** Do not present as real electron-orbital computation.

Status: Implemented deterministic analogy, not quantum-chemistry simulation.

Research grade: Research Brief

Review model: Implementation-backed analogy with explicit non-physics boundary.

PDF | Source note

Source Text

GeoSeed Orbital Model

This note preserves the electron-orbital pattern as a GeoSeed geometry model. It is not a claim that the SCBE stack computes real electron orbitals. The useful result is a deterministic structural analogy:

- the six Sacred Tongues map to `s/p/d/f/g/h` shells;
- the tongue phi ladder maps to radial depth in the Poincare ball;
- adjacent shells have a uniform hyperbolic gap of `ln(phi)`;

- the Cassisivadan shell lands at `r = 1/phi`;
- the magnetic sub-state counts form `1 + 3 + 5 + 7 + 9 + 11 = 36`.

The model is implemented in `src/geoseed/orbital_model.py`. It intentionally uses only Python standard-library math so it can run in the CLI and CI without adding NumPy or SciPy.

Interpretation

Electron orbitals are standing-wave solutions to a quantum Hamiltonian. In flat space, the angular part is described by spherical harmonics and the radial part depends on the potential and boundary conditions. In a hyperbolic manifold, the Laplacian becomes the Laplace-Beltrami operator and the geometry changes the spacing, volume growth, and node density.

GeoSeed uses that as a pattern language:

Tongue	Orbital	l	m-states	Phi weight
KO	s	0	1	phi^0
AV	p	1	3	phi^1
RU	d	2	5	phi^2
CA	f	3	7	phi^3
UM	g	4	9	phi^4
DR	h	5	11	phi^5

The radial map is:

```
rho(n) = n * ln(phi)
r(n)   = tanh(rho(n) / 2)
```

That gives the clean invariant:

```
distance(shell_n, shell_n+1) = ln(phi)
```

and the Cassisivadan checkpoint:

```
r(3) = tanh(3 * ln(phi) / 2) = 1 / phi
```

Use

Run:

```
python -m src.geoseed.orbital_model
```

Test:

```
python -m pytest tests/test_geoseed_orbital_model.py -q
```

Next useful step: render the `density_profiles` field into a small SVG or HTML plot so the standing-wave analogy can be inspected visually before wiring it into larger governance or training lanes.

Render:

```
python scripts/research/render_geoseed_orbitals.py
```

Transfer recorder stub

`src/geoseed/transfer_recorder.py` adds an audit scaffold for symbolic atom transfer. It records `(from_tongue, to_tongue, token)` triples, accumulates a 6x6 transfer matrix, and prices each shell hop as `n * ln(phi)`.

The future tokenizer wiring point is intentionally one call:

```
from src.geoseed.transfer_recorder import AtomTransferRecorder

rec = AtomTransferRecorder(session_id="tokenizer-run")
for token, from_tongue, to_tongue in tokenizer.route_tokens(text)
    rec.record(from_tongue, to_tongue, token)
```

The recorder is deterministic and standard-library only. It tracks route provenance without claiming physical atom transfer.

Generated from `research/geoseed_orbital_model.md` by `scripts/research/build_research_library.py`.