Check the braid at the crossings.

The most useful test is whether the strands remain separate, identifiable and continuous where they actually interlace. These checks apply to the displayed geometric model.

The representation

Each configuration has a checked profile in `models/braid-profile-N.json`. One rotor revolution contains 128 output vertices per carrier. The measured carrier permutation joins that cell to the next; it repeats after a complete carrier circuit. Industrial ends are separate radial offsets from each carrier's centreline, spaced 0.20 mm across the bank. The two wire diameters are 0.45 mm for tabletop and 0.16 mm for industrial.

The formation table contains 128 tabletop poses. The industrial tables contain 142 poses for 48 carriers and 159 for 72 carriers, with extra poses inserted at the tighter transitions on a 256-step phase grid. It fixes the real guide at the upstream boundary and the output at the downstream boundary. A piecewise linear interpolation supplies intermediate centreline nodes; radial bank offsets and guide motion are evaluated continuously. Four rotational prototypes reduce equivalent geometry. Symmetry is checked against the actual guide equations and the deposited profile before using it in the clearance bound.

`tools/relax_braid.py` and `tools/relax_formation.py` generate compact candidates using path length, bend regularisation and finite-segment clearance penalties. Their weights are numerical construction choices. An optimiser reaching its iteration limit does not count as validation or physical equilibrium: the independent geometry checks must still accept the candidate.

What each check tests

Check	Method and limit
Individual topology	`check_wire_topology.py` evaluates the actual guide equations independently and compares the cyclic partner and outside/inside sequence for every individual wire with the deposited profile. All wires are checked, not just the four prototypes. This is a sampled crossing-sequence comparison, not a general braid-group theorem.
Deposited output	`validate_braid_profile.py` checks exact distances between the rendered straight segments, including neighbouring periodic cells and the carrier permutation. Core clearance uses the closest radial point on each chord. Refinement 1 corresponds to the actual static mesh centreline edges.
Continuous formation clearance	`certify_formation.py` splits time at guide handoffs, profile vertices and formation keyframes. It bounds pair distance from midpoint distance minus relative endpoint travel and analytic chord-error bounds. Unresolved intervals are subdivided. A separate bound covers the core. Accepted lower bounds must exceed the 0.0003 mm numerical reserve.
Sampled formation	`validate_formation.py` checks four times as many phases as the table and records close pairs and bend diagnostics. This independently samples the moving approach and the exact downstream segments near its join.

Supply routing	`export_supply_paths.mjs` exports the viewer's complete source routes. `check_supply_clearance.py` compares all carriers and ends at 256 phases using exact segment distances after a spatial broad phase. It also checks nonlocal portions of the same wire, excluding pairs less than two wire diameters apart along the path. This is sampled clearance; it is not a continuous-time supply certificate or a complete self-contact test.
Tube surfaces	`check_strand_surfaces.mjs` builds the actual supply and formation tubes with the shared output joining ring. It tests every rendered triangle against its outward normals over 512 phases for all ends of the four rotational prototypes. The static deposited output is checked too. This catches folded rings; it is not a complete mesh self-intersection theorem.
Material and motion	`check_yarn.mjs` checks every carrier's crossing repeat, advected material, non-reversing spool payout, visible loop storage, actual neutral-line route lengths and handoff continuity. `check_animation.mjs` checks all carrier transfer and full-circuit seams.
Rendered identity	`check_output_identity.mjs` compares the actual GPU instance attributes and identity texture with each named wire's material-history coordinates across every output cell. An explicit integer cell attribute avoids Float32 pitch-division errors.
Tabletop source solids	`export_supply_solids.mjs` and `check_supply_solids.py` check conservative capsules around the actual source route against the exported guide post, arm and bobbin at maximum stored slack. FreeCAD is required. This excludes other carriers and the new viewer-only guide hardware.
Browser	`check_viewer.cjs` checks controls, count/end switching, exposed crossings, speed ratios, offline loading, mobile layout and console errors. `check_rendering.cjs` also checks pixel continuity at loop boundaries, solid close-up tubes, orbit/cutaway, pulley picking and paused rendering. Screenshots support visual review.

Reproduce the numerical checks

Run from the project directory. Python requires NumPy and SciPy; the JavaScript checks require Node. The checked assets are included, so validation does not require repeating the optimisation. Substitute 12, 16, 48 or 72 and use `--ends 1`, `--ends 4` or `--ends 8` for industrial checks.

```
node tools/export_braid_source.mjs 8 /tmp/braid-source-8.json
python3 tools/check_wire_topology.py models/braid-profile-8.json
python3 tools/validate_braid_profile.py /tmp/braid-source-8.json models/braid-profile-8.jso
python3 tools/certify_formation.py models/braid-profile-8.json
python3 tools/validate_formation.py /tmp/braid-source-8.json models/braid-profile-8.json
node tools/export_supply_paths.mjs models/braid-profile-8.json /tmp/supply-8.jsonl 256
python3 tools/check_supply_clearance.py /tmp/supply-8.jsonl
node tools/check_yarn.mjs
node tools/check_animation.mjs
```

The optional `--report PATH.json` arguments write machine-readable evidence. Industrial supply exports accept the end count as a fourth positional argument. `OPENBLAS_NUM_THREADS=1` keeps concurrent numerical runs from competing for unnecessary BLAS workers.

The surface checker imports the vendored Three.js through an esbuild alias. Supply the path to an installed esbuild executable:

```
esbuild tools/check_strand_surfaces.mjs --bundle --platform=node --format=esm --alias:three
node /tmp/braid-check-surfaces.mjs models/braid-profile-8.json /tmp/surfaces-8.json
```

Build and review offline

```
python3 tools/build_offline.py /path/to/esbuild
./start.sh
node tools/check_viewer.cjs
node tools/check_rendering.cjs
python3 tools/summarize_geometry.py
python3 tools/package.py
```

Set `PLAYWRIGHT_MODULE`, `CHROMIUM_PATH` and optionally `BRAIDBENCH_URL` for the local browser installation. The offline HTML embeds the five profiles and the model geometry; it requires no CDN or network fetch. Packaging checks document links and the ZIP CRC and creates a SHA-256 file manifest.

Where the evidence stops

The finite-diameter model prevents demonstrated different-wire overlaps through formation. It does not predict calibrated tension, material stress, friction, torsion, complete self-contact behaviour, winding depletion or force-driven carrier transfer. Multi-wire payout uses the bundle’s neutral centreline; per-end elasticity and slip are absent. The guide-bank swivel, return pulley and dancer are visible geometric mechanisms, with fabrication details still outstanding.

A physical machine needs a constrained carrier shoe and transfer mechanism, detailed brake and guide hardware, a full assembly interference study, and hand-built samples before its behaviour can be qualified. Those are separate from these numerical and browser results.

Reports apply to their hashed assets and stated scopes. Publication makes the geometric study accessible; it does not qualify the physical machine.