

# Supplementary Material S1: Artifact and Replay Guide for *Certified Golden-Branch Maximality in a Generated Standard-Map Domain*

Surya Tallavarjula and Seyed Ali Rastegar

Release v1.0.1; archive DOI: 10.5281/zenodo.20101820

## Abstract

This supplement records the artifact-level replay protocol, cache layout, validator semantics, negative-control tests, and representative raw support records for the finite certificate used in the main article. The main paper contains the mathematical certificate-implies-theorem statements; this guide documents how the archived release verifies their finite hypotheses.

## S1 Artifact and replay guide

### S1.1 Implementation details for the Theorem-III lower anchor

The body of the paper states Theorem III in the main text as an a posteriori lower-existence theorem at  $K = 0.9716350$ . The repository labels the successful implementation route as **TrackB**; this label is useful for code provenance but is not part of the mathematical statement. The used source object is

`artifacts/final_discharge/stage_cache/theorem_iii.json`.

The public archive records this object as a direct lower-anchor persistence object with `certificate_kind=direct_lower_anchor_persistence_certificate`, `track=TrackB`, `theorem_facing=true`, `promotion_allowed=true`, and `failed_checks=[]` [25]. A full reconstruction run recreates the supporting Phase-1 through Phase-6 files under `artifacts/proof_audit/`; those transient construction directories are not part of the compact release archive.

A natural lower-side concern is that a local a posteriori row away from the threshold would not, by itself, justify a near-critical anchor. The implementation addresses this by using a direct near-critical existence proof rather than a local-to-critical corridor argument. The successful numerical seed was produced by a derivative-weighted polishing objective: a residual may be small in a grid or  $L^\infty$  norm while its derivative is still too large for the reducibility frame, because differentiation weights Fourier mode  $k$  by  $k$ . The implementation therefore penalizes both the invariance residual and its  $\theta$ -derivative before applying the interval/radii check.

The code-level lower-anchor object records the following conservative values:

Repository field	Displayed value	Mathematical role
<code>selected_constants.K</code>	0.9716350	lower anchor used by the final comparison
<code>selected_constants.M</code>	8192	Fourier resolution

<code>selected_constants.nu</code>	1.001	analytic weight
<code>selected_constants.radius</code>	$3.0 \times 10^{-5}$	validation radius $r$
<code>Y_upper</code>	$7.0624 \times 10^{-8}$	residual/Newton defect contribution
<code>Z_upper</code>	0.2289541	linear/reducibility defect contribution
<code>Q_upper</code>	551.0007	nonlinear/quadratic contribution
<code>radii_lhs_upper</code>	$7.4352 \times 10^{-6}$	left side of the radii-polynomial inequality in the main text
<code>radii_margin_lower</code>	$2.2565 \times 10^{-5}$	positive existence margin

The object was allowed into the final proof graph only after the following components were present and replayed: outward-rounded residual proof, small-divisor proof, cohomology-inverse proof, frame/reducibility proof, nonlinear bound proof, tail bound proof, branch/chart compatibility proof, final graph-consumption proof, formal interval backend, and independent replay. The final graph-consumption check restricts downstream use to the direct lower anchor; it does not allow the object to be silently reinterpreted as a full parameter interval or mesh corridor. The strict lower replay reports `known_lower_gap=false`, meaning that the old unresolved lower-corridor gap is no longer part of the final proof path.

## S1.2 Archived release and compact artifact model

The paper-facing release is concrete and archived at Zenodo as `v1.0.1`, DOI `10.5281/zenodo.20101820` [25]. The release is intentionally compact. It distributes the theorem-facing upstream cache needed for ordinary review and leaves the storage-heavy downstream proof layers to be generated by the replay or reconstruction commands. In particular, the compact release includes the following cached theorem objects:

```
artifacts/final_discharge/stage_cache/theorem_i_ii.json
artifacts/final_discharge/stage_cache/theorem_iii.json
artifacts/final_discharge/stage_cache/theorem_iv.json
```

The release does not need to include timestamped full-regeneration logs, proof-audit scratch directories, paper-replay outputs, virtual environments, or cached Theorem V–VIII objects. Those files are either transient run products or storage-heavy downstream products. The downstream verification command and the full reconstruction command recreate them when needed.

Release field	Value or policy
Paper-facing release	<code>v1.0.1</code>
Archive DOI	<code>10.5281/zenodo.20101820</code>
Source repository	<a href="#">GitHub release page</a>
Cached theorem artifacts distributed	Theorem I–II, Theorem III, and Theorem IV objects in <code>artifacts/final_discharge/stage_cache/</code>
Storage-heavy downstream artifacts	Regenerated locally by downstream replay or full reconstruction; not distributed in the compact release
Release checksum ledger	<code>HASHES.sha256</code> , used to validate the archived static files
Run-specific reconstruction ledger	<code>artifacts/full_regeneration/&lt;stamp&gt;/REGENERATED_HASHES.sha256</code>
Primary downstream verification	<code>python scripts/replay_downstream_from_cache.py</code>
Primary full reconstruction	<code>python scripts/regenerate_all_theorems_from_scratch.py</code>

This release model separates three notions that should not be conflated. First, the static release archive validates the theorem-facing upstream cache through its checksum ledger. Second, downstream verification consumes the cached upstream objects and regenerates Theorem V through the

final theorem-facing discharge. Third, full reconstruction reruns the construction pipeline before repeating the proof-facing checks and writing a new run-specific manifest. The theorem claim depends on the accepted proof objects and interval inequalities, not on timestamped local logs.

### S1.3 Minimal verification

The minimal verification route checks the compact final theorem ledger and is useful as a quick smoke test:

```
python scripts/replay_minimal.py --out artifacts/paper_replay/minimal
```

The expected terminal result is `status=ok` with empty active-assumption, open-hypothesis, and remaining-burden fields. This command is intentionally lightweight; it does not rebuild Theorem III, Theorem IV, or the storage-heavy downstream cache.

### S1.4 Downstream verification from cached Theorems I–IV

The standard referee-facing verification route starts from the cached Theorem I–IV artifacts distributed in the compact release and regenerates the downstream proof layers:

```
sha256sum -c HASHES.sha256
python scripts/replay_downstream_from_cache.py \
  --out artifacts/paper_replay/downstream_from_cache
python scripts/validate_proof_payloads.py
```

This route verifies the archived upstream objects, rebuilds Theorem V, the identification layer, Theorem VI, Theorem VII, Theorem VIII, and the final theorem-facing discharge object, and then checks the final payload fields. It is the ordinary way to review the release without rerunning the expensive construction stages.

If the same script is run after a local fresh reconstruction, the regenerated bytes may differ from the frozen release ledger because run metadata and generated ledgers can change. In that situation the downstream replay can be run with release-hash checking disabled while still enforcing the theorem-facing validation fields:

```
python scripts/replay_downstream_from_cache.py \
  --no-hash-check \
  --out artifacts/paper_replay/downstream_from_cache
```

The frozen `HASHES.sha256` ledger validates the published archive. The regenerated ledger validates a local reconstruction run.

### S1.5 Command-line reconstruction from construction scripts

Artifact evaluators who want to reconstruct the theorem artifacts rather than only replay the compact release can run the top-level construction driver from the repository root:

```
python scripts/regenerate_all_theorems_from_scratch.py \
  --from-scratch \
  --force \
  --workers 64 \
  --run-focused-tests \
  --fail-on-acceptance
```

The command rebuilds the Theorem I–II workstream, reconstructs the Theorem-IV obstruction stack, invokes the Track-B Theorem-III reconstruction harness, installs the regenerated lower-anchor certificate into

```
artifacts/final_discharge/stage_cache/theorem_iii.json
```

and then rebuilds Theorem V, the identification layer, Theorem VI, Theorem VII, Theorem VIII, and the final theorem-facing discharge objects. With strict acceptance checking enabled, any missing artifact, failed replay, malformed proof payload, nonempty burden list, stale provenance record, or failed focused test is fatal.

The Track-B Theorem-III reconstruction can also be invoked directly for targeted auditing:

```
python scripts/regenerate_theorem_iii_trackb_from_scratch.py \
  --from-scratch \
  --force \
  --workers 64 \
  --copy-to-stage-cache \
  --theorem-i-artifact artifacts/final_discharge/stage_cache/theorem_i_ii.json \
  --theorem-ii-artifact artifacts/final_discharge/stage_cache/theorem_i_ii.json \
  --theorem-iv-artifact artifacts/final_discharge/stage_cache/theorem_iv.json
```

The top-level reconstruction command calls this route automatically; the separate command is documented only for targeted Theorem-III audits.

## S1.6 Resume mode and run-specific outputs

Long construction runs can be resumed without deleting completed stage-cache objects:

```
python scripts/regenerate_all_theorems_from_scratch.py \
  --use-cache \
  --workers 64 \
  --run-focused-tests \
  --fail-on-acceptance
```

When Theorem III has already been regenerated and installed into the final-discharge stage cache, the resume path can skip that heavy step explicitly:

```
python scripts/regenerate_all_theorems_from_scratch.py \
  --use-cache \
  --skip-theorem-iii \
  --workers 64 \
  --run-focused-tests \
  --fail-on-acceptance
```

A successful construction or resume run writes a run directory under

```
artifacts/full_regeneration/<stamp>/
```

containing `FULL_REGENERATION_MANIFEST.json`, `acceptance_checks.json`, `live_theorem_program_discharge_report.json`, replay logs, and `REGENERATED_HASHES.sha256`. The successful release-audit run for v1.0.1 completed minimal replay, downstream replay from cache, proof-payload validation, focused regression tests, manifest generation, regenerated hash-ledger creation, and final acceptance checks with zero failed acceptance checks. The transient `artifacts/full_regeneration/` directory is not part of the compact release archive.

## S1.7 Large-artifact policy

The compact archive keeps the upstream theorem cache through Theorem IV and excludes storage-heavy downstream cache objects. In particular, cached Theorem VIII objects can be multi-gigabyte files. Distributing those objects is unnecessary because they are regenerated by downstream verification and full reconstruction. The reconstruction driver records large cached Theorem VIII files by path, size, hash, and accepted replay fields, and the compact final report is built from the downstream replay object rather than by embedding the large JSON payloads again. This prevents the final report from duplicating multi-gigabyte proof objects while preserving an auditable route back to the generated files.

For a clean release, the following directories are treated as generated outputs and should not be included unless a separate archival policy explicitly requires them:

```
artifacts/full_regeneration/  
artifacts/proof_audit/  
artifacts/paper_replay/  
artifacts/final_discharge/stage_cache/theorem_v*.json  
artifacts/final_discharge/stage_cache/theorem_vi*.json  
artifacts/final_discharge/stage_cache/theorem_vii*.json  
artifacts/final_discharge/stage_cache/theorem_viii*.json  
artifacts/final_discharge/stage_cache/live_theorem_program_discharge_report*.json
```

The omitted downstream files are not hidden proof dependencies: they are generated by the documented commands above.

## S1.8 Artifact manifest and checksum protocol

The repository contains `ARTIFACT_MANIFEST.tsv` and `HASHES.sha256` [25]. The manifest records the release artifacts and their theorem-facing role. The checksum ledger validates the static release archive:

```
sha256sum -c HASHES.sha256
```

After a local full reconstruction, the correct ledger for the regenerated bytes is the run-specific file:

```
sha256sum -c artifacts/full_regeneration/<stamp>/REGENERATED_HASHES.sha256
```

The two ledgers serve different purposes. The release ledger validates what was published; the regenerated ledger validates what was produced by a new local run.

## S1.9 Repository-to-manuscript traceability matrix

The following matrix records how the manuscript-level theorem claims are connected to release artifacts and commands.

Manuscript claim	Release artifact or command	Verification role
Theorems I–II	<code>stage_cache/theorem_i_ii.json</code> ; full reconstruction driver	lower-side workstream and shared constants
Theorem III lower anchor	<code>stage_cache/theorem_iii.json</code> ; <code>regenerate_theorem_iii_trackb_from_scratch.py</code>	direct near-critical persistence certificate at $K_G^- = 0.9716350$
Theorem IV upper obstruction	<code>stage_cache/theorem_iv.json</code> ; full reconstruction driver	branch-scoped nongolden upper obstruction package

Theorem V through final reduction	<code>replay_downstream_from_cache.py</code> ; full reconstruction driver	regenerated downstream proof layers and endpoint comparison
Final theorem object	<code>replay_minimal.py</code> ; downstream replay; reconstruction manifest	terminal closed-discharge fields and final acceptance status
Release integrity	<code>HASHES.sha256</code>	validates the compact published archive
Fresh reconstruction integrity	<code>REGENERATED_HASHES.sha256</code>	validates locally rebuilt artifacts and reports
Negative controls	focused tests in <code>tests/</code>	verify rejection of corrupted margins, burdens, branch labels, and provenance fields

---

## S1.10 Expected statuses and acceptance fields

The final accepted proof path returns the terminal status

```
{
  "theorem_viii_discharge_status": "golden-universal-theorem-final-strong",
  "active_assumptions": [],
  "open_hypotheses": [],
  "remaining_true_mathematical_burden": [],
  "final_golden_maximality_discharge": true
}
```

The status string is not used as an axiom. The replay checks the associated interval endpoints, branch labels, generated-domain routing records, empty failure lists, and final-discharge fields before accepting the theorem object.

## S1.11 Recommended audit protocol

A reviewer or artifact evaluator can audit the release at two levels.

1. **Ordinary release audit.** Verify the static archive and regenerate the downstream proof layers from cached Theorems I–IV:

```
python -m venv .venv
source .venv/bin/activate
python -m pip install -U pip setuptools wheel
python -m pip install -e "[dev]"
sha256sum -c HASHES.sha256
python scripts/replay_minimal.py --out artifacts/paper_replay/minimal
python scripts/replay_downstream_from_cache.py \
  --out artifacts/paper_replay/downstream_from_cache
python scripts/validate_proof_payloads.py
python -m pytest -q \
  tests/test_trackb_phase6_final_integration.py \
  tests/test_proof_payload_negative_controls.py \
  tests/test_replay_heavy_audit_protocol.py \
  tests/test_theorem_iv_cache_inventory.py \
  tests/test_upper_obstruction_audit.py
```

2. **Full construction audit.** Rebuild the theorem artifacts and rerun the proof-facing checks:

```
python scripts/regenerate_all_theorems_from_scratch.py \
  --from-scratch \
  --force \
  --workers 64 \
  --run-focused-tests \
  --fail-on-acceptance
```

For interrupted long runs, use the resume command in Section S1.6. The reconstruction command is an audit route, not a claim that the Theorem-III and Theorem-IV construction stages are lightweight.

## S1.12 Robustness and negative-control checks

The release includes positive replay checks and negative-control checks. The negative controls modify theorem-facing payloads in ways that would make the proof invalid—for example by eroding the endpoint margin, changing branch labels, adding a nonempty burden list, replacing a compressed transport object with a diagnostic shell, removing the direct-anchor formal flag, or perturbing a provenance/hash field. The validator is expected to reject these corrupted objects. These tests support the audit argument that the proof system is not merely reading a success string.

Check type	Representative test or command	Expected behavior
Positive minimal replay	<code>python scripts/replay_minimal.py</code>	accepted terminal theorem ledger
Positive downstream replay	<code>python scripts/replay_downstream_from_cache.py</code>	accepted downstream final theorem object
Proof-payload validation	<code>python scripts/validate_proof_payloads.py</code>	accepted payload schema and closure fields
Direct-anchor negative controls	<code>tests/test_proof_payload_negative_controls.py</code>	fail closed when lower-anchor evidence is weakened
Theorem-IV cache inventory	<code>tests/test_theorem_iv_cache_inventory.py</code>	fail closed when obstruction inventory is incomplete
Upper-obstruction audit	<code>tests/test_upper_obstruction_audit.py</code>	fail closed when obstruction fields are missing or inconsistent
Heavy-audit protocol	<code>tests/test_replay_heavy_audit_protocol.py</code>	verifies construction/replay boundary semantics

## S1.13 Minimal theorem-object schemas

The following schema summary is not a replacement for the JSON artifacts. It records the minimum fields that a reviewer should expect in proof-level objects used by the final replay.

Object	Required fields	Meaning	Failure condition
the direct lower anchor	<code>certificate_kind</code> , <code>selected_constants.K</code> , radii/flag fields, burden lists	persistence-side lower anchor	failed formal flag, nonpositive radii margin, or nonempty burden

Upper obstruction	obstruction interval, common-support, tail-coherence, analytic-incompatibility fields	proof-level upper front	missing analytic lift or unstable tail/support
Compressed V	<code>target_interval</code> , compatibility, branch identity, gap-preservation fields	rational-to-irrational transport contract	raw shell used or gap not preserved
Identification	overlap window, branch witness, chart/label fields	common threshold branch	chart mismatch or second branch witness
VII support	six support certificates and empty failure lists	generated-domain nongolden discharge	any nonempty failure field
VIII reduction	final implication, normalization, theorem-universe matching	reduction to main theorem	scope mismatch or equality ambiguity
Final replay	upstream statuses, readiness flags, failed checks	terminal theorem object	failed check or nonempty burden list

---

## S1.14 Representative raw replay payloads

The main text states the mathematical consequences of the validation payloads. For reproducibility, this supplement records a representative raw support object for Theorem VII. Some archived field names retain the older word “exhaustion”; in the manuscript narrative these fields are interpreted as generated-domain discharge records. The proof does not depend on the spelling of the status summaries alone; the validator checks the associated Boolean fields, empty failure lists, manifest provenance, and interval margins.

```
{
  "status": "theorem-vii-generated-domain-exhaustion-support-certified",
  "all_vii_assumptions_dischargeable": true,
  "undischarged_vii_assumptions": [],
  "screened_panel_labels": ["silver", "bronze"],
  "support_certificates": {
    "exact_near_top_lagrange_spectrum_ranking_certificate": {
      "proves_exact_near_top_lagrange_spectrum_ranking": true,
      "unranked_labels": []
    },
    "theorem_level_pruning_certificate": {
      "proves_theorem_level_pruning_of_dominated_regions": true,
      "unproved_pruning_labels": []
    },
    "screened_panel_global_completeness_certificate": {
      "screened_panel_globally_complete": true,
      "missing_completion_labels": []
    },
    "deferred_retired_domination_certificate": {
      "proves_deferred_or_retired_classes_are_globally_dominated": true,
      "uncontrolled_deferred_labels": [],
      "uncontrolled_retired_labels": []
    },
    "termination_search_exclusion_certificate": {
```



```

    "proves_termination_search_promotes_to_theorem_exclusion": true,
    "unpromoted_candidate_labels": []
  },
  "omitted_class_global_control_certificate": {
    "omitted_classes_globally_controlled": true,
    "uncontrolled_omitted_labels": []
  }
}
}
}

```

### S1.15 Detailed transport ledger and endpoint fields

The main text summarizes the transport budget by the strict inequality  $\Delta_{\text{available}} > \delta_{\text{tot}}$ . The following detailed tables record the charge decomposition and the stored endpoint fields used by the validator.

The public replay exposes the detailed transport fields through the compressed-contract payload and the manifest-backed replay logs. The entries below are organized by mathematical role: rational-window transport, branch/chart drift, tail transport, and interval rounding/widening.

Error source	Symbol	Certified bound	Artifact field / source
rational approximation	$\delta_{\text{rat}}$	$6.99999999812711 \times 10^{-7}$	transport-budget ledger
branch/chart drift	$\delta_{\text{branch}}$	$4.99999999866223 \times 10^{-7}$	transport-budget ledger
tail transport	$\delta_{\text{tail}}$	$5.99999999839467 \times 10^{-7}$	transport-budget ledger
rounding/widening	$\delta_{\text{round}}$	$1.0 \times 10^{-12}$	outward-rounding floor
total majorant	$\delta_{\text{tot}}$	$1.80000099995184 \times 10^{-6}$	sum of the four displayed charges
available transport gap	$\Delta_{\text{available}}$	$1.0 \times 10^{-5}$	VI top-gap scale used by the transport contract
transport remaining margin	$\Delta_{\text{available}} - \delta_{\text{tot}}$	$8.19999900004816 \times 10^{-6}$	transport gap-preservation margin
pass condition	—	$\Delta_{\text{available}} > \delta_{\text{tot}}$	gap preservation verified by the replay ledger

The final endpoint comparison is a separate strict interval comparison:

Endpoint or window	Displayed value	Stored field	Proof use
Golden lower anchor $L_G$	0.9716350	lower-anchor object / lower shell	lower endpoint in final sign check
Near-top upper ceiling $U_{\text{nt}}$	0.9716347	Theorem-VII near-top upper bound	challenger ceiling
Endpoint separation $L_G - U_{\text{nt}}$	$3.0 \times 10^{-7}$	final-reduction margin field	final strict comparison
Compressed-V target	[0.9716350, 0.9716370]	compressed target interval	transport compatibility
Identification overlap	[0.971634, 0.971638]	identification overlap window	branch seam
Native-tail witness	[0.9716352, 0.9716355]	identification native-tail witness	witness inclusion

Witness-width/top-gap slack	$9.7 \times 10^{-6}$	final-reduction geometry field	$10^{-5}$ top-gap scale minus $3 \times 10^{-7}$ witness width; not the transport charge
--------------------------------	----------------------	--------------------------------	---

---

The proof uses the stored fields named in the third column, not the formatted decimal strings in the manuscript. The transport charge  $1.80000099995184 \times 10^{-6}$ , the unused transport budget  $8.19999900004816 \times 10^{-6}$ , the witness-width/top-gap slack  $9.7 \times 10^{-6}$ , and the final endpoint separation  $3.0 \times 10^{-7}$  are distinct quantities. If any stored upper endpoint or accumulated charge is widened enough to make a required margin nonpositive, the margin-erosion negative control is expected to fail before final reduction. Conversely, a displayed decimal equality or apparent near-equality is not accepted unless the stored outward-rounded inequality remains strict. The small theorem-level endpoint margin is therefore protected by two independent positive checks: the transport budget has  $8.19999900004816 \times 10^{-6}$  of slack, and the final lower-versus-upper endpoint comparison has  $3.0 \times 10^{-7}$  of slack.

### S1.16 Supplementary replay record

The archival release is accompanied by machine-readable theorem objects, replay scripts, checksum ledgers, validator documentation, a certificate glossary, and negative-control fixtures, following standard reproducibility norms for computational scientific results [17, 18, 19, 20, 25]. The supplementary reproducibility record contains the certified-universe file, artifact manifest and hashes, minimal replay transcript, downstream replay transcript, negative-control test transcripts, and hardware/replay-boundary notes. These materials are part of the public proof package cited in the data and code availability statement.

## S2 Scope and artifact clarifications

This supplement collects scope and artifact clarifications that are useful for readers auditing the proof package. It is not a substitute for the mathematical proof sections above; its purpose is to make the relation between the manuscript theorem, the code package, the replay path, and the scoped claim explicit.

### S2.1 The theorem is scoped to the generated comparison domain

The generated arithmetic domain is not the set of all irrational rotation numbers. It is the generated domain grammar recorded in `CERTIFIED_UNIVERSE.json`: screened-panel labels, exact near-top ranking records, theorem-pruned regions, inherited-domination labels, terminal candidates with exclusion estimates, and omitted-tail records. The theorem ranges only over this generated domain. This is why the abstract, main theorem, and scope paragraphs say that the result is not a proof of the unrestricted Greene conjecture. Greene’s residue criterion, MacKay’s renormalization program, and converse-KAM methods provide the external dynamical context [4, 5, 23, 22, 24]; the present theorem is a replayable certified-domain theorem.

### S2.2 The lower side uses a direct near-critical anchor

A central lower-side concern is whether the near-critical lower anchor is supported directly, rather than inferred from a local validation row away from the threshold. The current release resolves this by using a direct lower-anchor certificate at  $K = 0.9716350$  [25]. The source of truth is

`artifacts/final_discharge/stage_cache/theorem_iii.json`; the proof-audit directory records the  $H^1$ -polished seed, radii payload, formal attachments, promotion gate, independent replay, and final integration. The manuscript therefore does not claim that a  $K = 0.2$  local row proves the near-critical anchor. It claims the direct a posteriori existence statement in the main Theorem III lower-anchor statement.

The strict audit path checks that the lower certificate is direct-anchor mode, that `known_lower_gap=false`, and that the lower object is accepted only after the radii, small-divisor, cohomology-inverse, frame, nonlinear, tail, branch/chart, graph-consumption, backend, and independent-replay flags pass. This is the lower-side answer to the original 95+ implementation objection: the final lower anchor is now proof-carrying and mechanically audited.

### S2.3 Upper analytic incompatibility is branch-scoped

The upper obstruction is stronger than a residue plot but narrower than a general converse-KAM theorem. The code first proves local rational-branch enclosures, then assembles an adaptive obstruction atlas, then checks a coherent upper window, coherent hyperbolic barrier, common-support neighborhood, tail-aware neighborhood, tail stability, bridge-profile robustness, and final analytic-incompatibility payload. The theorem is only over the identified branch and theorem universe; it is not presented as a new unrestricted converse-KAM theorem for every twist-map invariant circle. The manuscript states the obstruction-incompatibility principle in the branch-order obstruction lemma in the main text; the final replay accepts the upper object only when the proof-level support, tail, branch, and incompatibility fields are present and hash-matched.

### S2.4 Transport and final margin are interval comparisons, not midpoint arguments

The final comparison margin is small enough that midpoint arguments would be unacceptable under standard interval-arithmetic validation practice [14, 16]. The proof therefore uses interval semantics throughout: a challenger upper ceiling is accepted only if its upper endpoint is strictly below the lower golden anchor. The compact replay exposes the conservative near-top upper bound 0.9716347, the direct lower anchor 0.9716350, and margin  $3.0 \times 10^{-7}$ . The compressed transport contract is accepted downstream only if the uniform majorant preserves the available gap and the raw transport shell is excluded. The detailed transport ledger in Section S1.15 records the majorant decomposition and endpoint fields.

### S2.5 Generated-domain discharge is a finite grammar audit

The generated-domain part of the proof is not a finite ad hoc comparison among a few rotation numbers. Theorem VII uses a generated grammar and assigns every generated nongolden record to one of six closed mechanisms: screened-panel completion, exact near-top ranking, theorem-level pruning, routing by inherited domination estimates, terminal candidates carrying explicit exclusion estimates, or omitted-tail control. A zero omitted-tail count in the compact replay means an empty complement inside the generated grammar, not a statement about all external irrational rotation numbers. Any future nonempty omitted-tail release would require a nonvacuous envelope-control certificate.

## S2.6 $GL(2, \mathbb{Z})$ language is representative selection

The  $GL(2, \mathbb{Z})$  clause is a representative-normalization statement inside the certified arithmetic grammar. It is not an analytic conjugacy classification of all projectively equivalent rotation numbers and not a proof that every noble representative has the same physical threshold in the unscoped standard map. The theorem compares normalized arithmetic classes as recorded by the final reduction certificate.

## S2.7 Replay, reconstruction, and fail-closed behavior

The minimal replay builds compact proof-level shells and verifies the final discharge ledger. The downstream replay requires the compact upstream cache `theorem_i_ii.json`, `theorem_iii.json`, and `theorem_iv.json`, verifies the archived inputs against `HASHES.sha256` when run in release mode, builds the V-and-beyond proof-level shells, and then runs the final discharge. Fresh construction is routed through `scripts/regenerate_all_theorems_from_scratch.py`, which rebuilds the theorem artifacts, reruns the proof-facing checks, and writes a run-specific manifest and regenerated hash ledger. Storage-heavy downstream Theorem V–VIII cache files and timestamped reconstruction logs are generated outputs, not required components of the compact release. This separation prevents a smoke replay, a cached downstream replay, and a full construction audit from being confused with one another.

## S2.8 Why the validator is not merely a status-field reader

The paper-facing validator checks numerical and symbolic conditions before final replay: the direct lower anchor is present and accepted by the direct-anchor audit; Theorem IV exposes the required obstruction/common-support/tail/incompatibility fields; the compressed Theorem-V contract is strong, has an ordered target interval, does not consume the raw shell, and preserves the gap; the identification witness lies strictly inside the overlap window with matching branch/chart labels; the VI challenger upper and VII near-top upper bound both lie below the lower golden anchor; the near-top pending count is zero; and every VII failure field is empty. The negative-control tests then perturb these fields and verify fail-closed rejection.

Potential objection	Resolution in the archived release	Where checked or stated
The result overclaims Greene’s conjecture	The theorem is explicitly scoped to $\mathcal{D}_{\text{cert}}$ and the standard-sine theorem universe	Abstract, the introduction and Main Theorem
The lower anchor is unsupported	direct lower-anchor certificate at $K = 0.9716350$ is used as the source of truth	the main Theorem III section; <code>replay_heavy_lower.py</code>
Scalar residual does not imply graph persistence	The certificate uses derivative-weighted polishing plus radii, cohomology, frame, nonlinear, and tail checks	the main Theorem III section and this supplement
Upper obstruction is just a label	The upper theorem requires common-support, tail-coherence, tail-stability, and analytic-incompatibility fields on the identified branch	the main Theorem IV section
Transport uses a tiny unexamined margin	The transport ledger decomposes the available gap, charged errors, and remaining positive margin	Section <a href="#">S1.15</a>

Domain discharge is hand-picked	The generated grammar has route counts and empty failure lists for all six mechanisms	the main Theorem VII discharge section
$GL(2, \mathbb{Z})$ sounds like analytic conjugacy	The final theorem uses it only as representative normalization	the final normalization section of the main text
Hash checks replace mathematics	Hashes provide provenance; mathematical acceptance uses interval, symbolic, routing, and branch checks	the finite-check soundness section of the main text
Status strings are trusted	Negative controls perturb raw fields and require fail-closed rejection	Section <a href="#">S1.12</a>
Full reconstruction is confused with ordinary replay	ordinary replay checks archived theorem-facing objects; the construction audit reruns expensive stages through the reconstruction command and writes a separate manifest	Section <a href="#">S1</a>

---

## S2.9 Boundary of the release

For clarity, the release does not claim any of the following: a solution of Greene’s conjecture over all irrational rotation numbers; a lightweight runtime for the heavy construction stages; a universal converse-KAM theorem outside the certified branch convention; or a theorem for arbitrary analytic twist families without regenerated family/domain/normalization artifacts. The compact archive also does not claim to store every downstream generated cache object; Theorem V–VIII objects and full-regeneration logs are generated by the documented commands. These exclusions are not weaknesses of the stated theorem; they are part of the certified-universe and artifact-distribution boundary that makes the theorem auditable.

## References

- [1] A. N. Kolmogorov. On conservation of conditionally periodic motions for a small change in Hamilton's function. *Doklady Akademii Nauk SSSR*, 98:527–530, 1954.
- [2] V. I. Arnold. Proof of a theorem of A. N. Kolmogorov on the invariance of quasi-periodic motions under small perturbations of the Hamiltonian. *Russian Mathematical Surveys*, 18(5):9–36, 1963.
- [3] J. Moser. On invariant curves of area-preserving mappings of an annulus. *Nachrichten der Akademie der Wissenschaften in Göttingen, Mathematisch-Physikalische Klasse II*, 1962:1–20, 1962.
- [4] J. M. Greene. A method for determining a stochastic transition. *Journal of Mathematical Physics*, 20(6):1183–1201, 1979.
- [5] R. S. MacKay. A renormalisation approach to invariant circles in area-preserving maps. *Physica D: Nonlinear Phenomena*, 7(1–3):283–300, 1983.
- [6] J. N. Mather. Existence of quasi-periodic orbits for twist homeomorphisms of the annulus. *Topology*, 21(4):457–467, 1982.
- [7] S. Aubry and P. Y. Le Daeron. The discrete Frenkel–Kontorova model and its extensions. I. Exact results for the ground-states. *Physica D: Nonlinear Phenomena*, 8(3):381–422, 1983.
- [8] V. Bangert. Mather sets for twist maps and geodesics on tori. In *Dynamics Reported*, volume 1, pages 1–56. Teubner, Stuttgart, 1988.
- [9] O. Perron. Über die Approximation irrationaler Zahlen durch rationale. *Sitzungsberichte der Heidelberger Akademie der Wissenschaften*, 1921.
- [10] T. W. Cusick and M. E. Flahive. *The Markoff and Lagrange Spectra*. Mathematical Surveys and Monographs, vol. 30. American Mathematical Society, 1989.
- [11] R. de la Llave. A tutorial on KAM theory. In *Smooth Ergodic Theory and Its Applications*, Proceedings of Symposia in Pure Mathematics, volume 69, pages 175–292. American Mathematical Society, 2001.
- [12] W. Tucker. A rigorous ODE solver and Smale's 14th problem. *Foundations of Computational Mathematics*, 2(1):53–117, 2002.
- [13] A. Haro, M. Canadell, J.-L. Figueras, A. Luque, and J.-M. Mondelo. *The Parameterization Method for Invariant Manifolds: From Rigorous Results to Effective Computations*. Applied Mathematical Sciences, Springer, 2016.
- [14] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.
- [15] R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–201, 1969.
- [16] S. M. Rump. Verification methods: rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, 2010.

- [17] R. D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.
- [18] V. Stodden, F. Leisch, and R. D. Peng, editors. *Implementing Reproducible Research*. CRC Press, 2014.
- [19] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig. Ten simple rules for reproducible computational research. *PLoS Computational Biology*, 9(10):e1003285, 2013.
- [20] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S. Haddock, K. Huff, I. Mitchell, M. Plumbley, B. Waugh, E. P. White, and P. Wilson. Best practices for scientific computing. *PLoS Biology*, 12(1):e1001745, 2014.
- [21] J.-L. Figueras, A. Haro, and A. Luque. Rigorous computer-assisted application of KAM theory: a modern approach. *Foundations of Computational Mathematics*, 17:1123–1193, 2017.
- [22] D. Jungreis. A method for proving that monotone twist maps have no invariant circles. *Ergodic Theory and Dynamical Systems*, 11:79–84, 1991.
- [23] R. S. MacKay and I. C. Percival. Converse KAM: theory and practice. *Communications in Mathematical Physics*, 98:469–512, 1985.
- [24] C. Falcolini and R. de la Llave. A rigorous partial justification of Greene’s criterion. *Journal of Statistical Physics*, 67:609–643, 1992.
- [25] S. Tallavarjula and S. A. Rastegar. *Certified Golden Maximality for Invariant-Circle Thresholds in Conservative Twist Maps: proof package* [software and certified artifacts]. Zenodo, version v1.0.1, DOI: 10.5281/zenodo.20101820; public GitHub release v1.0.1, 2026.