Integer and modular flow reconfiguration for moduli four and five

Codex research session, initiated by Jack
4 September 2026 · Research brief, version 1

Abstract

We give an argument that reduction modulo k preserves the connected components of nowhere-zero flow reconfiguration when $k \in \{4, 5\}$. The key is to lift an auxiliary midpoint flow in which zeros are allowed. This addresses small-modulus cases of a published open problem, whose authors explicitly identify $k = 4$ as unresolved. The abstract argument has been checked in Lean; its graph-specific instantiation remains a written proof. This is a candidate original result, not a peer-reviewed or independently certified breakthrough.

1. Problem and statement

Fix an orientation of a finite loopless multigraph G. An integer k-flow is an integer circulation f with $0 < |f(e)| < k$ on every edge. A modular flow is a nowhere-zero $\mathbb{Z}/k\mathbb{Z}$-circulation. In the reconfiguration graphs $\mathcal{I}_k(G)$ and $\mathcal{M}_k(G)$, two states are adjacent when the support of their difference is a simple cycle, including a pair of parallel edges. Problem 2.6 of Esperet et al. [1] asks whether modular connectivity implies integer connectivity; the general equivalence also appears as Conjecture A of Cranston et al. [2].

Theorem. *For $k \in \{4, 5\}$, reduction modulo k induces a bijection*

$$\mathrm{Comp}\big(\mathcal{I}_k(G)\big) \longrightarrow \mathrm{Comp}\big(\mathcal{M}_k(G)\big).$$

This includes empty state spaces. It asserts neither existence of a nowhere-zero flow nor connectivity for every graph.

2. Proof

Write c for the signed unit vector of an oriented simple cycle. We use two standard graph facts: every nonzero edge of an integer circulation lies on a cycle conformal to its signs; and every modular circulation, *including one with zeros*, has an integer lift h with $|h(e)| < k$. The latter is Tutte's lifting theorem [1, Theorem 2.3]. The accompanying full proof supplies elementary arguments for both facts.

Fibers are connected. For two bounded lifts f, g of the same nowhere-zero modular flow, $(g - f)(e) \in \{-k, 0, k\}$. Choose a cycle c conformal to $g - f$ and replace f by $f + kc$. Every changed edge immediately takes its final value $g(e)$, so bounds and nonvanishing are preserved. The number of differing edges strictly decreases. Repetition joins f to g.

Unit moves lift. If $q' = q + c$, any bounded lift f of q gives $f + c$ as a lift of q'. Its values lie in $[-k, k]$. Nonzero residues exclude the only forbidden values in this interval: $-k$, 0, and k.

Double moves lift. If $q' = q+2c$, lift the auxiliary modular circulation $q+c$ to a bounded integer circulation h, allowing zero coordinates. Set

$$a = h - c, \qquad b = h + c.$$

These have residues q, q' and values in $[-k, k]$. Their nonzero residues again exclude $-k, 0, k$, so both are valid integer k-flows. Since $b - a = 2c$, they are adjacent. Join prescribed endpoint lifts to a and b inside their respective fibers. The auxiliary h need not itself be a nowhere-zero reconfiguration state.

On a simple cycle, conservation forces a modular difference to be a constant multiple of c. Every nonzero scalar modulo 4 or 5 has a representative in $\{1, -1, 2, -2\}$. Reversing c covers negative signs, so every modular path lifts. Conversely, integer cycle moves reduce to modular moves or repetitions. Bounded lifting supplies surjectivity on vertices, proving the component bijection. □

3. Verification and qualifications

Machine-checked scope. The file `FlowReconfiguration.lean` uses Lean 4.34.0-rc2 and only its `Std` library. It proves bounded lifting, fiber connectivity, the midpoint lemma and component equivalence for a finite integer circulation system with signed unit cycles. Compilation and a separate kernel replay of the module's declarations passed. The reported axioms are the standard `propext`, `Classical.choice` and `Quot.sound`; there are no `sorry` placeholders or custom axioms.

The structural conformal-cycle property is a hypothesis of that system. The graph-specific proof of this property, and the spanning-forest argument identifying modular graph flows with liftable residue classes, are written in the full note but are *not instantiated from a graph incidence definition in Lean.* Thus this is an abstract formal theorem with a written graph application, not an end-to-end machine proof from a standard graph library. Kernel replay uses Lean's kernel, not an independent implementation of it.

Computational checks. An exact-integer Python implementation constructs lifts and paths. An independent verifier enumerates all edge assignments on 12 small graphs for each modulus, including bridges, parallel edges, disconnected graphs and empty graphs. It checks 3,060 lifted modular moves, 4,092 ordered endpoint pairs in common fibers, and 12,276 returned path steps. It also checks 575 bounded lifts, of which 368 have zero residues, and 243 additional lifts for other moduli. Separate exploratory enumeration found matching component counts in 300 sampled graph/modulus cases (150 per modulus). These are finite checks, not a proof for all graphs.

Status. Targeted searches on 4 September 2026 found no prior proof of the stated result. The current source versions checked were arXiv:2512.17342v4 and arXiv:2606.24685v1. Novelty and correctness still warrant independent expert review. The arbitrary-k conjecture remains outside this result; no resolution of Tutte's flow-existence conjectures is claimed.

4. Process, prompt and effort

The work was produced by Codex in an agentic research session initiated by Jack. The initiating prompt was:

> Find a specific mathematical or programming or similar open problem and solve and formalize it until you make a legitimate breakthrough. you are incredibly intelligent and resilient and resourceful and capable. do not give up easily.

The search began with candidate problems in spectral graph theory. Some apparent targets were rejected after finding existing proofs. The work then focused on the explicit flow-reconfiguration question, developed the fiber argument and midpoint construction, checked the abstract proof in Lean, and tested a constructive implementation against independent enumeration. All mathematical claims here are the output of that AI-assisted session; this report does not imply outside endorsement.

Reported research effort: 35 minutes 11 seconds (2,111 seconds). This elapsed duration was supplied by Jack for the original research session. It is not an independently measured compute-cost estimate. Subsequent report preparation and website publication are excluded.

The full written proof, Lean source, Python implementation, exact check records and reproducibility instructions are available at `https://proofs.mhny.dev/`.

References

[1] L. Esperet, K. Hendrey, A. Lagoutte, M. Marseloo, S. Norin and R. Steiner. *Nowhere-zero flow reconfiguration.* arXiv:2512.17342v4, 3 July 2026. `https://arxiv.org/abs/2512.17342v4`.

[2] D. W. Cranston, J. Li, B. Su, Z. Wang and N. Xu. *Reconfiguration of Nowhere-zero Flows.* arXiv:2606.24685v1, 23 June 2026. `https://arxiv.org/abs/2606.24685v1`.