Proven-stable modulation: automatic stability certificates for user-written filters

Naman Soni

Infinite · infinitedevices.org

pre-registered 2026-11-23 · published 2026-11-23 · data and code at commit ccf102e1

Abstract

A filter that is stable at every fixed knob setting can still blow up when the knobs move: a linear Moog ladder whose resonance alternates between 0 and 2.89 every sample grows without bound, though it is stable at every fixed resonance up to 4. Infinite's compiler checks stability at one fixed setting only, and Field users and agents write their own filters. We asked whether a compiler can prove, from a kernel's code alone, that it cannot blow up under any modulation inside its knob ranges. Before measuring, we pre-registered four hypotheses with kill criteria. The pass reads each kernel's state matrix from Infinite's interpreter, finds one quadratic energy function that falls at every knob setting (a linear matrix inequality), re-checks it in float64 and covers the whole knob box with an adaptive cell split. On 416 kernels (16 from the literature, 200 generated, 200 blind), the instability oracle finds a witness for 174, and the pass certifies 0 of them (H1). It matches all four of McClellan's hand proofs, and for the Moog ladder it proves $k \le 2.8848$, past the published 2.88, against an oracle witness at 2.8856 (H2). It certifies 92.5% of the stable generated kernels and 88.7% of the blind set, against a bar of 80% (H3). Today's frozen check, at the default knobs, calls 77 of 85 blind kernels that can blow up safe. Slew certificates (H4), which let the energy function depend on the knob when the knob moves slowly, add nothing as registered: while the cutoff can reach zero, the filter cannot decay fast enough to pay for any change in the energy function. With the cutoff held at 20 Hz or above, they widen the certified range by 21% (Moog) and 41% (diode), an unregistered result. The pass runs in Python, as a research tool; it is not in the app.

Keywords: time-varying filters, stability, modulation, Lyapunov, LMI, switched systems, DSP compiler, Field, virtual analog

1 Introduction

Musical modulation moves filter knobs fast: envelopes, LFOs and audio-rate cables change a resonance or a cutoff every sample. A recursive filter whose poles sit inside the unit circle at every fixed setting can still blow up under that motion. The best-known case takes two lines to show [1]: a linear Moog ladder at cutoff $g = 1.3764$, with its resonance k switched between 0 and 2.89 every sample, grows without bound, although the same filter is stable for every fixed k below 4 (Figure 1).

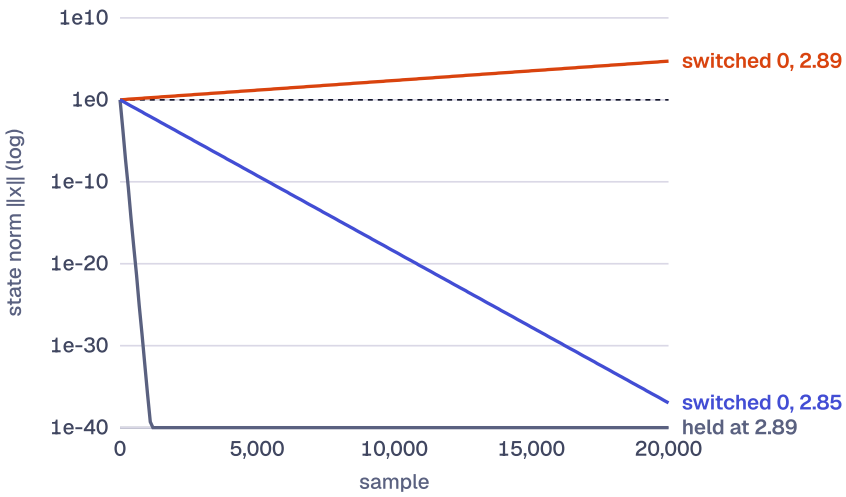

Figure 1. The trap. The dashed line is the start, ‖x‖ = 1. A linear Moog ladder at cutoff g = 1.3764 (McClellan's counterexample) from a unit start. With the resonance k switched between 0 and 2.89 every sample, the state grows to 52,567 in 20,000 samples, though the filter is stable at every fixed k up to 4: held at 2.89 it decays. Switched between 0 and 2.85, inside the range the certifier proves, it cannot grow (‖x‖ = 1e-37 at the end). Data: phase5-trap.json.

Proofs that a filter survives modulation exist for named filters, written by hand [2, 3, 4, 1]. Shipped tools rely on structure (topology-preserving transforms) and on ranges written in comments; the Faust library documents ladder feedback "between 0 and 4" [5]. In Infinite, the only check is a frozen one: the compiler's analysis computes the spectral radius of the state Jacobian at the default knobs and calls a value below 1 stable. That is the check the trap defeats. And Field, Infinite's language for user-written kernels, lets users and agents write their own recurrences, which no hand proof covers.

This report asks whether a compiler can read a kernel and prove, from its code alone, that the kernel cannot blow up under any modulation of its knobs inside their declared ranges, or say that it can't. The maths exists in control theory: one quadratic energy function that falls at every knob setting, found by a linear matrix inequality (LMI) [6]. It has not been joined to an audio language's compiler. Our contributions are an automatic certifier for Field kernels that are linear in their state (Section 3), its evaluation against an instability oracle, four published hand proofs and a blind set (Section 4), and a negative result for slew-limited certificates with its cause (Section 4.5). The data, scripts and every figure's source are public (Section 9).

2 Pre-registered hypotheses

All five hypotheses, with the numbers that would kill them, were committed to the repository (commit `5de4551b`) before the first measurement.

	Claim	Metric and bar	Kill criterion
H1	Sound: the pass never certifies a kernel that can blow up.	0 certificates on kernels with an instability witness, on L, G and the blind set B.	any such certificate (after a rig-bug check)
H2	It matches the hand proofs automatically.	≥ 99% of McClellan's published range for SVF, Sallen-Key, Moog and diode ladder; never past the published counterexamples.	any filter below 95%, or a certificate past 2.89 / 8.91
H3	It covers most stable user-style kernels.	≥ 80% of generated kernels without a witness certified, then the same on the blind set.	< 80%
H4	Slew certificates widen the range for smooth modulation.	With k moving at most its full range in 20 ms, the certified k range of the Moog and diode ladders is ≥ 25% wider than the any-speed range.	< 25% on either
H5	It is cheap enough for a compiler.	Median ≤ 50 ms per kernel in C++.	scored only if an in-app port is approved

The study ships as research if H1 to H3 pass. Because the author wrote the analysis and the first two corpora, "confirmed" requires the blind set B, generated after the scorer was locked (commit `f9180894`) and scored once.

3 Method

3.1 Kernels and their matrices

A Field kernel with `state` cells whose update is linear in the state is a linear parameter-varying system: x[n+1] = A(θ[n]) x[n] + B(θ[n]) u[n], where θ are its knobs inside their declared ranges. The coefficients may be any function of θ (`sin`, `pow`, `exp`, a `tan` prewarp, or `if` on a knob, which makes the system switched). Nonlinear loops and delay lines are out of scope.

A(θ) is read from the real interpreter, not from a second evaluator. A self-test in a dev build runs one sample step with the state set to each unit vector and the input at zero; column i of A is the next state. A linearity gate checks superposition on random state pairs, and a kernel that fails it leaves scope. Across 232 compiled kernels and 32 112 knob settings, the interpreter's matrices match a Python reference within 1.7e-6 (Float32 rounding).

Three corpora were scored. L holds 16 filters ported from published equations: McClellan's four, in the paper's state form, and twelve textbook filters (one-pole, Chamberlin and TPT state-variable filters, three RBJ biquad forms with modulated cutoff and Q, a coupled resonator, an allpass and a 4-stage phaser, a Stilson ladder, a DC blocker). G holds 200 generated kernels (seed 20261121): random dense state updates of size 1 to 6 with 1 to 3 knobs, coefficients built from affine, `sin`, `pow`, `exp` and `if` terms, half of them built to be stable at every fixed setting. B, the blind set, uses the same generator with seed 20261123, sizes up to 8 and twice the share of `if` terms. Field's sample backend caps a kernel at 128 registers, and a dense update of size n costs about $3n^2$ of them, so 184 of the 400 generated kernels don't compile. Those were scored on the Python matrices and are reported separately.

3.2 Ground truth: an instability oracle

A product of state matrices $A(\theta_1)\ldots A(\theta_k)$ with spectral radius above 1 is a proof of instability: repeating that switching sequence grows the state without bound. The oracle searches for such a witness (radius > 1 + 1e-6, length ≤ 8): every product of length ≤ 4 over the box corners, a dense frozen scan, a greedy search over periodic products, and continuous tuning of the best candidates. A witness proves instability; the absence of one proves nothing, so H1 is scored on kernels with a witness and H3's denominator is the kernels without one. The oracle was checked against the paper's counterexamples first: it finds the Moog ladder unstable from k = 2.8856 and the diode ladder from 8.9009, both inside the published (2.88, 2.89] and (8.90, 8.91].

3.3 The certifier

The pass looks for one matrix P such that the energy x^TPx falls at every knob setting: $A(\theta)^T P A(\theta) \prec P$ for all θ in the box. If such a P exists, the kernel cannot blow up under any sequence of knob values, at any speed. On a grid of settings this is an LMI, solved with Clarabel through CVXPY [7, 8], with each point rescaled by its own decay so that settings that barely decay (cutoff near zero) are as well conditioned as the rest. The solver's P is never trusted: the norm ‖P^½ A P^−½‖ is re-checked in float64 at every point. Between grid points the box is covered by cells. On a cell, A(θ) lies within a measured margin of the blend of its corner matrices, and the P-norm is convex, so a cell closes when the largest corner norm plus 1.5 times the largest blend error measured at the cell centre and edge midpoints is below 1. A cell that fails is split; a corner whose own norm reaches 1 is added to the LMI as a cutting plane. Cells start split at every `if` threshold, so each sees one side of a jump.

Slew certificates (H4) let P depend on the knob, P(k), with the energy required to fall only for knob steps no larger than the slew limit. This is the standard rate-bounded parameter-dependent Lyapunov function [6]; it reduces to the common P when the slew is unbounded.

4 Results

4.1 The hand proofs, automatically (H2)

Table 1. H2: the certifier against McClellan's four hand proofs, the cutoff over g ∈ [1e-5, 1e3] and k bisected to 0.001. “Oracle: unstable from” is the smallest upper end of k at which the instability oracle finds a witness. Data: phase1-h2.json, phase0-b3.json.

filter	paper, by hand	bar	certified automatically	oracle: unstable from	pass
Moog ladder	$k \in [0, 2.88]$	≥ 2.85	**$k \in [0, 2.8848]$**	2.8856	yes
Diode ladder	$k \in [0, 8.90]$	≥ 8.81	**$k \in [0, 8.8965]$**	8.9009	yes
Sallen-Key	$k < 2$	≥ 1.98	**$k \in [0, 1.9990]$**	2 (frozen)	yes
SVF	any $R \geq R_{min} > 0$	$R \in [R_{min}, 10]$, $R_{min} = 0.01, 0.1$	**both boxes**	—	yes

H2 passes on all four filters with the same code and no hand work (Table 1). For the Moog ladder the certificate goes past the published proof. Together with the oracle's witness, which moves the cutoff along with the resonance, the any-speed limit of the linear Moog ladder is now pinned to $k \in [2.8848, 2.8856)$, where the paper left [2.88, 2.89]. On 200 000 random settings the certifier never saw, no point has a P-norm of A at or above 1.

Both ends of these brackets are exact proofs (version 1.1). The four filters are trapezoidal, $A = M^{-1}N$ with $M = I - g \cdot Ac$ and $N = I + g \cdot Ac$. M and N commute, which gives the identity $M^T(A^TPA - P)M = 2g \cdot (Ac^TP + P \cdot Ac)$, a standard property of the bilinear transform. So the certificate holds at every cutoff $g > 0$ exactly when $Ac^TP + P \cdot Ac$ is negative definite. Ac is affine in the resonance, so checking the two ends of its range is enough. We converted the certifier's own float64 P to rationals exactly and checked both ends by exact LDL^T factorisation; all five certificates (Moog, diode, Sallen-Key, and the SVF at R_min = 0.01 and 0.1) pass, for every $g > 0$, with no grid. The two witnesses are proved too: each switching product is computed exactly in rationals, and Arb ball arithmetic encloses its eigenvalues, giving spectral radius ≥ 1.0000399 (Moog, k = 2.8856) and ≥ 1.0000619 (diode, k = 8.9009). Negative controls behave: the same Moog P fails the exact check at k = 2.8856, and a sequence the oracle rejected encloses below 1. The identity also explains why the certificate's margin shrinks like g as the cutoff falls to zero. It applies only to trapezoidal filters with a knob that enters Ac affinely; the generated and blind sets still rest on the cover below.

4.2 Today's check is fooled

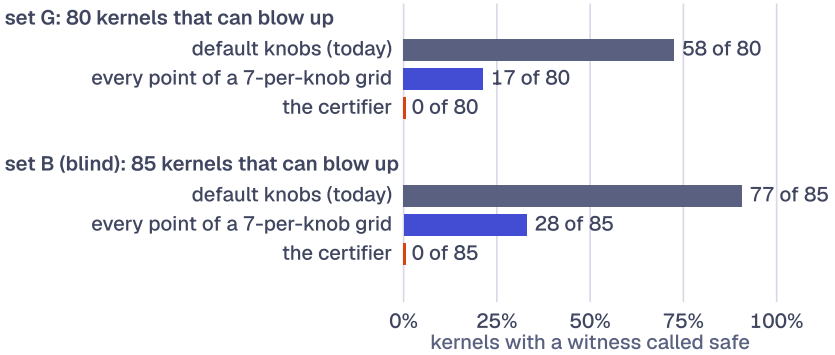

Figure 2. Kernels that can blow up (they have a witness) but that a frozen-knob check calls stable. “Default knobs” is what Infinite's analysis does today: spectral radius below 1 at one setting. The grid check is far stricter than today's and still misses 17 of 80 on G and 28 of 85 on B, because their witnesses need the knobs to move. The certifier calls none of them safe. Data: phase5-frozen.json.

On the generated set, 80 of 200 kernels can blow up, and the frozen check at the default knobs calls 58 of 80 of them safe; on the blind set it calls 77 of 85 safe (Figure 2). A far stricter frozen check, at every point of a 7-per-knob grid, still misses 17 of 80 and 28 of 85: their witnesses need the knobs to move, and no frozen check can see that. The worst of the missed kernels grow by 17% per sample under their witness, a million times louder in about 90 samples (2 ms at 48 kHz).

4.3 Soundness and coverage (H1, H3)

Table 2. H1 and H3 on every scored kernel. A witness is a product of up to 8 state matrices with spectral radius above 1 + 1e-6: a proof that the kernel can blow up. H1 counts certificates on such kernels (bar: 0). H3 is the share of kernels without a witness that are certified (bar: 80%, on G and then on the blind set B). “Field matrices” were measured on Infinite's interpreter; kernels over Field's 128-register cap use the Python reference. Data: phase3-L-G.json, phase4-B.json.

set	kernels	with a witness	certified with a witness (H1)	no witness	certified (H3)	share
L: literature filters	16	9	**0**	7	5	71.4%
G, Field matrices	125	43	**0**	82	77	93.9%
G, Python matrices	75	37	**0**	38	34	89.5%
B (blind), Field matrices	91	34	**0**	57	55	96.5%
B (blind), Python matrices	109	51	**0**	58	47	81.0%
all of G	**200**	**80**	**0**	**120**	**111**	**92.5%**
all of B	**200**	**85**	**0**	**115**	**102**	**88.7%**

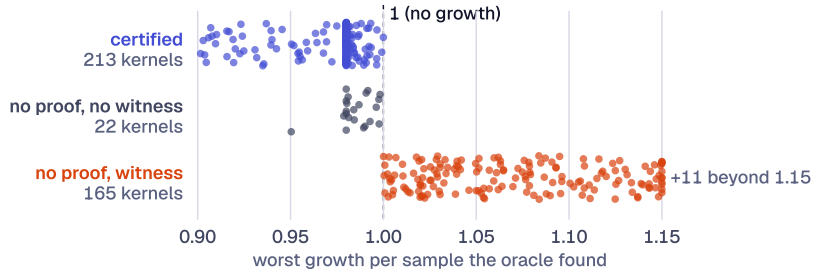

Figure 3. Every G and B kernel at the worst growth per sample the instability oracle found (the spectral radius of its worst switching product, per step). Right of 1, the kernel can blow up. No certified kernel sits there (H1). The grey row is what H3 misses: kernels with no witness and no proof, all close to 1. Points beyond 1.15 are counted at the edge. On the web, hover a point for its kernel. Data: phase3-L-G.json, phase4-B.json.

H1 holds on all three sets: of 174 kernels with a witness, the certifier certifies 0 (Table 2). H3 passes on G (111 of 120, 92.5%) and is confirmed on the blind set (102 of 115, 88.7%). Every cover the certifier started closed; no kernel ended with only a grid proof. Figure 3 shows the shape of the result: certified kernels all sit left of 1, witnesses all sit right of it, and the kernels the pass leaves unproved without a witness sit just below 1 (worst growth 0.95 to 0.998), where every LMI is clearly infeasible. Each of those is either stable without a common quadratic energy function, or unstable in a way the oracle did not find. Coverage falls with size: on B, 73 of 76 kernels of size up to 5 are certified, and 29 of 39 of size 6 to 8. Two literature filters are left without a proof for the right reason: the one-pole at a = 0 and the coupled resonator at r = 1 reach the edge of stability inside their own boxes. Certification takes 0.36 s per kernel at the median and 6.2 s at p95, in Python on an M2.

4.4 Float32

Table 3. Does each certificate survive the interpreter's Float32 arithmetic? At up to 5000 check points per certificate (every LMI point and the cover points with the largest P-norm) the interpreter's own matrix was measured. “Holds”: every $\|A_{f32}\|_P < 1$. “Proved”: the largest rise over float64 is also below the cover's slack (1 − worst cell bound). Kernels over the register cap can't run on the interpreter, so a bound ($\sqrt{\kappa} \cdot n \cdot 10 \times$ the measured Float32 error) stands in for the measurement. Data: phase3-L-G.json, phase4-B.json.

certified kernels	count	holds at every check point	proved in Float32	largest rise over float64
L, Field	5	4	3	2.5e-07
G, Field	77	77	77	1.7e-07
G, Python (bound)	34	n/a	31	8.7e-05 (bound)
B, Field	55	55	55	1.3e-07
B, Python (bound)	47	n/a	41	9.5e-05 (bound)

A certificate is a float64 statement, while the interpreter computes in Float32. At up to 5000 check points per certificate, the interpreter's own matrices were measured (Table 3). Every measured G and B certificate holds and is proved, with a largest rise of 1.7e-7. Two literature filters show the limit: the paper's SVF and the TPT SVF are certified with a margin smaller than Float32 can resolve, because their boxes reach the edge of stability (R = R_min). A shipped check must report them as not proved in Float32, or shrink the box. For the kernels that don't compile, a loose bound stands in for the measurement, and 9 of 81 certificates are not proved by it.

4.5 Slew certificates (H4)



Figure 4. H4: the largest resonance k the certifier proves when k may move by at most its full range in 20 ms (Δk ≤ range / 960 per sample), against the any-speed range and the frozen limit. With the cutoff free down to 0.15 Hz, as registered, the slew limit adds nothing (the registered bar was +25% on both filters; the diode's slew LMI fails even at its any-speed 8.90). With the cutoff held at 20 Hz or above it does, but that run was not registered. Data: phase3-h4.json.

H4 fails as registered (Figure 4). With the cutoff free down to 1e-5 (about 0.15 Hz), a slew limit adds nothing to either filter. The cause is structural, not a tuning problem. As the cutoff goes to zero, the filter stops moving: A → I, and the energy can fall by only about g per sample. Hold the state fixed and step k up or down: for the energy never to rise, V(x, k ± Δk) ≤ V(x, k) must hold both ways, so near g = 0 any change of P with k must be paid for by a decay that is not there. Any energy function V(x, k) that ignores the cutoff is forced flat in k, and so collapses to the any-speed certificate. Affine and piecewise-linear P(k), with 9 and 17 nodes, both fail past the any-speed limit.

With the cutoff held at 20 Hz or above, an unregistered exploratory run, the slew limit widens the certified range to k ≤ 3.50 for the Moog ladder (+21%, short of the bar) and to 12.50 for the diode ladder (+41%). A control run shows the gain comes from the slew limit, not from the floor: with the floor and a constant P, the LMI is already infeasible at Moog 2.9 and diode 8.9. Simulations at those limits stay bounded.

5 Discussion

A compiler can prove that a user-written linear filter survives any modulation of its knobs, from its code alone, and the proof costs well under a second for most kernels. On the blind set the pass is never wrong where it can be checked, and it certifies nearly nine in ten of the kernels with no known instability, against today's frozen check, which waves through nine in ten of the kernels that do blow up. For hand-written filters it is at least as good as a person: it reproduces four published proofs and tightens one.

What it does not do is equally clear. Slow modulation does not buy a wider range unless the certificate also knows how fast the filter itself decays; a fixed 20 ms slew limit means nothing to a filter whose cutoff can be 0.15 Hz. The useful form of the slew claim is relative: a step in k no larger than some multiple of the cutoff, per sample. That is a new hypothesis, not a rescue of this one.

6 Threats to validity

- **What "certified" means.** The LMI and its float64 re-check are exact at their points. Between points, the cover's margin is a measured heuristic (1.5 × the blend error at the cell centre and edge midpoints), not a

bound. A certificate is "verified on a dense adaptive cover", not a formal proof over the continuum. Evidence that the margin holds: 200 000 unseen random points in the hand-proof check, up to 5000 more per certificate in Float32, and no point anywhere with a P-norm of A at or above 1. An interval bound on A(θ) per cell would make it a proof; it was not done. For the four hand-proof filters this is closed in version 1.1: their certificates and witnesses are proved exactly (Section 4.1), with no cover.

- **The oracle is one-sided.** An unstable kernel the oracle misses would sit in the H3 denominator, and a false certificate on it would count as a success. The float64 cover makes that unlikely; the oracle can't rule it out. It had one rig bug (a grid too coarse to see four frozen-unstable kernels), found and fixed before any verdict.
- **Who wrote the kernels.** L was ported by the author; G and B come from one generator the author wrote. No real user kernels or shipped presets were scored. Generated kernels are dense random systems, a harder shape than typical sparse filters, so coverage on real filters may be higher; it is not measured.
- **Matrices from Python.** 184 of 400 generated kernels exceed Field's register cap and were scored on Python matrices, with a bound in place of the Float32 measurement.
- **Linear only.** Nonlinear loops (`tanh`, `clamp`) and delay lines, common in real patches, are out of scope.
- **H4's bar was unreachable by design.** The registered claim fails for every energy function that ignores the cutoff; that should have been caught before sign-off.

7 Related work

Laroche gave two state-matrix criteria for time-varying recursive filters, applied per topology [2]; Rabenstein and Czarnach showed that stability depends on the state-space structure as well as the coefficients [9]. Wishnick proved the state-variable filter stable under modulation [3]. Werner and McClellan extended Laroche's norm criterion to products of state matrices over several steps [4]. McClellan used common quadratic Lyapunov functions for trapezoidally discretised filters, with hand-found or randomly searched P for four filters and counterexamples at the edge, and named slowly varying parameters as future work [1]. The LMI and parameter-dependent Lyapunov machinery is standard in control [6]. We found no compiler, DSL or audio tool that certifies stability of user-written filters under modulation; absence in a search is not proof of absence. Infinite's first report studied a different compile-time analysis on the same language [10].

8 Conclusion

Stability under modulation can be checked by a compiler, automatically and soundly in every case we could test, for any Field kernel that is linear in its state. Today's frozen check cannot do this, and on generated kernels it misses most of the ones that blow up. Slew-limited certificates do not help while the cutoff can reach zero; a rate-aware form is the next question. The pass is a Python research tool; putting it in Infinite's compiler is a separate decision.

9 Reproducibility

The study is in the Infinite repository under `docs/research/stable-modulation/`. `scripts/kick-the-tires.sh` reruns the oracle against the paper's counterexamples, the certifier at the published bounds and the locked scorer on 40 kernels in about two minutes, and compares each verdict with the committed data. `scripts/run_all.sh` reruns every phase into `data-rerun/` (the Float32 step needs a dev build of Infinite). `ARTIFACT.md` links each claim to its command, data file and tolerance. Raw data is never edited; every figure and table here is generated from it.

Appendix A. Rig bugs, and changes from the registered method

Oracle, phase 0. The first oracle's grid included the cutoff ends g = 0 and g → ∞, where A = ±I has radius exactly 1; those trivial points outranked the real near misses. Candidates are now interior only. Found before any score.

Oracle, phase 3. A 4-to-9-point grid missed four generated kernels that are unstable at one fixed setting between grid points. The oracle now runs a dense frozen scan with local tuning. The baseline counts moved from 76 to 80 kernels with a witness; no witness was lost and no verdict changed.

Float32 check. The first pass showed rises up to 0.44 on kernels with `if`. Every such point sat within 1e-9 of a threshold, below Float32 resolution, so the interpreter took the other branch. Points are now moved to 1e-6 of the knob range on the cover's side.

Cover margin. The registered first-order Lipschitz margin can't close near g → 0, where decay per sample shrinks like g but the derivative of A doesn't; the second-order blend margin replaced it before H2 was scored.

Version 1.1 (2026-11-23). After a reviewer-style critique, the four hand-proof certificates and the two witnesses were checked exactly (Section 4.1, `scripts/phase6_exact.py`, `data/phase6-exact.json`). No verdict or number changed; the Moog and diode brackets are now proved rather than verified.

H4. Piecewise-linear P(k) (9 and 17 nodes) was added to the registered affine P(k) once affine failed; it fails the same way.

10 References

1. R. McClellan (2026). Stability analysis of time-varying virtual analog filters. *Proc. 29th Int. Conf. on Digital Audio Effects (DAFx26), Cambridge, MA*. www.dafx.de/paper-archive/2026/papers/DAFx26_paper_28.pdf
2. J. Laroche (2007). On the stability of time-varying recursive filters. *Journal of the Audio Engineering Society, vol. 55, no. 6, pp. 460–471*. www.aes.org/e-lib/browse.cfm?elib=14168
3. A. Wishnick (2014). Time-varying filters for musical applications. *Proc. 17th Int. Conf. on Digital Audio Effects (DAFx-14), Erlangen*. dafx.de/paper-archive/details/yfd4LbdXr2Cywsqxfby8ug
4. K. J. Werner and R. McClellan (2022). Time-varying filter stability and state matrix products. *Proc. Int. Conf. on Digital Audio Effects (DAFx20in22), Vienna*. www.dafx.de/paper-archive/details/RdjkWSf80p3vUeP3_STX_w
5. GRAME. Faust libraries: vaeffects. *Faust library documentation*. faustlibraries.grame.fr/libs/vaeffects/
6. S. Boyd, L. El Ghaoui, E. Feron and V. Balakrishnan (1994). Linear Matrix Inequalities in System and Control Theory. *SIAM Studies in Applied Mathematics, vol. 15*. web.stanford.edu/~boyd/lmibook/
7. S. Diamond and S. Boyd (2016). CVXPY: a Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research, vol. 17, no. 83, pp. 1–5 (software version 1.9.3 used here)*. www.cvxpy.org
8. P. J. Goulart and Y. Chen. Clarabel: an interior-point solver for conic programs with quadratic objectives. *Software*. clarabel.org
9. R. Rabenstein and B. Czarnach (1985). Stability of recursive time-varying digital filters by state vector transformation. *Signal Processing*. www.sciencedirect.com/science/article/abs/pii/0165168485900908
10. N. Soni (2026). Aliasing-aware compilation for Field. *Infinite Research, Technical Report 1*. infinitedevices.org/research/aliasing-aware-field.html

---

Provenance. Commit ccf102e151b1ecb00dc1743b2ae4e42ef4b25026 reproduces every number in this report. Machine: Apple M2, 8 cores, 8 GB RAM, macOS 27.2.1 arm64; Apple clang version 21.0.0 (clang-2100.4.1.1), for the dev build that dumps A(theta).

Raw data: https://infinitedevices.org/research/stable-modulation/data/ · Source: https://github.com/n1m21n/Infinite/tree/ccf102e1/docs/research/stable-modulation · Interactive version: https://infinitedevices.org/research/stable-modulation.html