Paper 60W: Null and Morphology-Preservation Tests for Phase-Compatible High-Order Tail Structure in BAO Residuals

Michael J. Sarnowski

April 28, 2026

Abstract

Paper 60V introduced a phase-coupled decomposition of high-order BAO residual structure. In that model, the transferred $n \geq 7$ tail is not treated as uniformly waste. Instead, high-order modes are sorted into phase-compatible and phase-incoherent components relative to a carrier band $n = 3$–4 and a switch-sideband band $n = 5$–6. Paper 60V found that the real BOSS DR12 Fourier-space monopole residual over $0.015 \leq k \leq 0.145$ produced 1980 strict balanced candidates, preserving band-limited morphology, peak count, spacing, transferred $n = 3$ selection, and low incoherent high-n tail. The present paper tests whether that signature survives adversarial null and morphology-preservation controls.

We construct real-data-derived null and control residuals from the same BOSS DR12 input. The tested families include sign-shuffled residuals, value-shuffled residuals, Fourier-phase-randomized residuals, circular shifts, and block-shuffled residuals. A corrected all-in-one pipeline generates the null inputs, runs the same v3F scan used in Paper 60V, summarizes strict balanced candidate counts, and audits target morphology before model interpretation. The corrected suite evaluates 168 null/control scans using 12 parallel workers, $N_{\mathrm{paths}} = 12000$, $N_{\mathrm{dense}} = 8192$, $n_{\max} = 12$, and the same k-window and baseline convention as Paper 60V.

The destructive null families collapse. Sign shuffle, value shuffle, Fourier phase randomization, block shuffle with block size 2, and block shuffle with block size 3 produce zero strict balanced candidates. This collapse occurs even though several destructive null families still achieve high best-row band-limited cosine similarity, showing that high cosine alone is not the Paper 60V signature. Survival occurs only in morphology-preserving or phase-shifted controls. Circular shifts produce 1284 total strict balanced candidates across three surviving runs, and block shuffle with block size 4 produces one surviving run with 643 strict balanced candidates. The morphology audit shows that the block-size 4 survivor preserves the real target closely, with band cosine 0.9891, shifted cosine 0.9891, autocorrelation similarity 0.999978, harmonic-support similarity 0.999091, spacing similarity 0.9539, peak count 10, and dominant harmonic $n = 10$. The circular-shift survivors preserve spacing, autocorrelation, harmonic support, and peak count despite poor direct phase alignment.

These results reinforce the bounded diagnostic claim of Paper 60V. The v3F strict balanced signature does not arise from arbitrary band-limited noise. It collapses when phase, ordering, or local morphology are destroyed, and survives only when the null/control retains substantial BAO-band waveform structure. The block-size result is treated as an empirical local-morphology threshold in the declared BOSS DR12 window and preprocessing pipeline, not as a universal physical constant. Paper 60W therefore reframes the null result as a morphology-preservation

test: the phase-compatible tail selector is sensitive to retained BAO-band structure, not to generic residual scrambling. The result does not prove a first-principles path-level phase law, but it strengthens the case that such a law has a well-defined empirical target.

Keywords: BAO residual morphology; Holosphere Theory; null tests; surrogate residuals; phase-compatible tail; coherent tail; incoherent tail; admissibility-gated transport; morphology preservation; block shuffle; Fourier phase randomization; BOSS DR12.

1 Introduction

Baryon acoustic oscillation (BAO) measurements are conventionally interpreted as fossil acoustic structure in the large-scale distribution of galaxies and remain a central tool in observational cosmology [14, 15, 16, 17, 18, 19, 20]. The 60-series of Holosphere Theory papers studies a narrower diagnostic question: whether BAO-like residual morphology can be reconstructed from admissibility-gated transport, finite harmonic selection, and corridor-mediated visibility rather than by treating the residual only as an empirical oscillatory fit.

Paper 60V [1] addressed a specific failure mode left open by Paper 60U [2]. Paper 60U showed that a transfer-filtered admissibility ladder can reconstruct band-limited BAO-like residual morphology, but the transferred high-order tail remained too large. Paper 60V replaced blunt high-n suppression with a phase-coupled tail decomposition. Instead of treating all $n \geq 7$ mass as waste, the model decomposed the tail as

$$M_{n\geq 7}^{\mathrm{tail}} = M_{n\geq 7}^{\mathrm{coh}} + M_{n\geq 7}^{\mathrm{incoh}}, \tag{1}$$

where $M_{n\geq 7}^{\mathrm{coh}}$ is the high-order mass classified as phase-compatible with the carrier/sideband scaffold and $M_{n\geq 7}^{\mathrm{incoh}}$ is the phase-incoherent leakage-like component.

The carrier/sideband scaffold in Paper 60V was defined by three interpretive bands:

$$n = 3\text{–}4 \quad \text{carrier band}, \tag{2}$$

$$n = 5\text{–}6 \quad \text{switch-sideband band}, \tag{3}$$

$$n \geq 7 \quad \text{high-order tail}. \tag{4}$$

The main finding was that the high-order tail is not uniformly waste. In the real BOSS DR12 Fourier-space monopole residual over

$$0.015 \leq k \leq 0.145, \tag{5}$$

the v3F phase-coupled model found a large strict balanced candidate class. The real residual produced 1980 strict balanced candidates, while the best scan and extractor rows preserved the $n = 3$ transferred carrier, the 10/10 band peak count, strong band-limited cosine similarity, and low incoherent high-n tail.

Paper 60V was deliberately bounded. It did not claim a full raw-residual reconstruction, a replacement for standard BAO cosmology, or a first-principles derivation of the phase-lock rule. Its claim was diagnostic: if high-order modes are sorted by phase compatibility, the high-n tail can

be decomposed into candidate coherent support and incoherent leakage. The next required test is adversarial. A diagnostic selector that works only on the real residual but also works on arbitrary scrambled residuals would not be physically meaningful.

The present paper, Paper 60W, therefore asks the falsification question:

Does the Paper 60V strict balanced signature survive when the residual's phase, ordering, spacing, or local morphology is destroyed?

This question is sharper than asking whether a null residual can achieve a high band-limited cosine score. High cosine alone is not the Paper 60V signature. The strict balanced signature requires the simultaneous preservation of several conditions:

$$\cos_{\text{band}} \geq 0.88, \tag{6}$$

$$S_{\text{spacing}} \geq 0.92, \tag{7}$$

$$N_{\text{peak}}^{\text{model}} \geq 10, \tag{8}$$

$$n_{\text{transfer}} = 3, \tag{9}$$

$$M_{n\geq 7}^{\text{incoh}} \leq 0.12, \tag{10}$$

$$f_{\text{coh}}^{n\geq 7} \geq 0.65. \tag{11}$$

The null test therefore asks whether scrambled or surrogate residuals can pass the entire joint signature, not merely one metric.

Paper 60W constructs real-data-derived null and control families. The first three are destructive nulls:

- sign-shuffled residuals,
- value-shuffled residuals,
- Fourier-phase-randomized residuals.

These are intended to damage phase organization, peak ordering, or Fourier phase structure while retaining some real-data-derived statistical content.

The fourth family is circular shift. This is not a fully destructive null. It preserves the global waveform but changes its alignment. It is therefore better interpreted as a phase-shifted morphology control.

The fifth family is block shuffle. This family is especially important because it tests how much local morphology is needed for v3F survival. Block shuffling preserves short residual segments while disrupting their global order. Paper 60W tests block sizes 2, 3, and 4, with 24 block-shuffle nulls at each block size.

The corrected all-in-one suite runs 168 total v3F scans:

$$4 \times 24 + 3 \times 24 = 168. \tag{12}$$

The four non-block families are sign shuffle, value shuffle, Fourier phase randomization, and circular shift. The three block families are block shuffle with block sizes 2, 3, and 4. Each null/control input is generated from the same real BOSS DR12 residual and then passed through the same v3F scan settings used in Paper 60V. The run uses 12 parallel workers, $N_{\text{paths}} = 12000$, $N_{\text{dense}} = 8192$, and $n_{\max} = 12$.

The corrected null generator also addresses a concern found during preliminary testing. Earlier block-shuffle controls occasionally produced apparent survivor cases that preserved the real band-limited target exactly. Paper 60W therefore adds a morphology audit before interpreting survival. For every null/control input, the audit compares the null band-limited target to the real band-limited target using direct cosine similarity, shifted cosine similarity, autocorrelation similarity, harmonic-support similarity, peak count, peak-spacing similarity, dominant harmonic, and tail mass. This prevents a morphology-preserving control from being misclassified as a destructive null.

The main result is clean. The destructive nulls collapse:

$$\text{sign shuffle}: \quad 0 \text{ strict balanced candidates}, \tag{13}$$

$$\text{value shuffle}: \quad 0 \text{ strict balanced candidates}, \tag{14}$$

$$\text{Fourier phase randomization}: \quad 0 \text{ strict balanced candidates}, \tag{15}$$

$$\text{block shuffle, size 2}: \quad 0 \text{ strict balanced candidates}, \tag{16}$$

$$\text{block shuffle, size 3}: \quad 0 \text{ strict balanced candidates}. \tag{17}$$

Survival occurs only in morphology-preserving or phase-shifted controls:

$$\text{block shuffle, size 4}: \quad 643 \text{ strict balanced candidates in one run}, \tag{18}$$

$$\text{circular shift}: \quad 1284 \text{ strict balanced candidates across three runs}. \tag{19}$$

This result reinforces Paper 60V rather than disproving it. If the v3F selector were responding to arbitrary band-limited noise, the destructive nulls should regularly produce strict balanced candidates. They do not. If the selector is responding to retained BAO-band morphology, then controls that preserve enough local or global waveform structure should sometimes survive. They do. The null result therefore supports a refined interpretation:

> The v3F strict balanced signature collapses when phase, ordering, spacing, or local morphology are destroyed, and survives only when substantial BAO-band waveform morphology is preserved. The selector is therefore not detecting arbitrary scrambled residual structure; it is sensitive to retained morphology.

The block-size result is interpreted conservatively. The transition between collapse at $b = 2, 3$ and one survivor at $b = 4$ suggests that v3F requires a minimum retained local waveform segment, but Paper 60W does not claim that $b = 4$ is a universal physical constant or derived BAO correlation length. It is an empirical local-morphology threshold in the declared BOSS DR12 window and preprocessing pipeline. A future sampling study should convert block size into physical k-scale using the actual input grid and test whether the transition tracks a fixed fraction of the carrier or sideband period across samples.

This distinction matters for the broader Holosphere program. Paper 60V suggested that high-order tail structure may be phase-compatible support rather than uniform leakage. Paper 60W tests whether that claim is merely a curve-fitting artifact. The result is favorable but bounded. It strengthens the diagnostic mechanism while leaving the first-principles derivation open. A later paper must still derive the phase-lock rule from path-level admissibility sums of the form

$$\mathcal{A}_n = \sum_{\gamma \in \Omega_n^{\mathrm{surv}}} w_\gamma \exp(i\Phi_\gamma). \tag{20}$$

The remainder of the paper is organized as follows. Section 2 defines the destructive nulls and morphology-preserving controls. Section 3 describes the corrected all-in-one null pipeline, v3F scan settings, and morphology audit. Section 4 reports the strict balanced counts, block-size threshold behavior, and survivor morphology. Section 5 interprets the result as support for morphology-sensitive phase-tail selection rather than arbitrary null fitting. Section 6 states the conditions under which the Paper 60W interpretation would fail. Section 7 summarizes the result and identifies the next target: deriving the phase-lock rule from path-level corridor interference.

2 Null Design

Paper 60W tests the diagnostic claim made in Paper 60V by asking whether the v3F strict balanced signature survives when the residual structure is deliberately disrupted. The goal is not to build a new BAO model in this paper. The goal is to separate three possibilities:

1. the v3F selector detects arbitrary band-limited noise,
2. the v3F selector detects generic smooth oscillatory structure,
3. the v3F selector detects retained BAO-band morphology with the correct joint signature.

The distinction matters because Paper 60V did not claim that high band-limited cosine similarity alone is meaningful. The Paper 60V signature was a joint condition: morphology, spacing, peak count, transferred $n = 3$ selection, low incoherent high-n tail, and high coherent-tail fraction must hold together. Paper 60W therefore evaluates null and control residuals using the same strict balanced criteria rather than using a single fit score.

2.1 Strict Balanced Signature

The strict balanced signature is inherited from Paper 60V. A candidate is considered strict balanced only if it satisfies the combined morphology and tail-decomposition requirements:

$$\cos_{\text{band}} \geq 0.88, \tag{21}$$
$$S_{\text{spacing}} \geq 0.92, \tag{22}$$
$$N_{\text{peak}}^{\text{model}} \geq 10, \tag{23}$$
$$n_{\text{transfer}} = 3, \tag{24}$$
$$M_{n\geq 7}^{\text{incoh}} \leq 0.12, \tag{25}$$
$$f_{\text{coh}}^{n\geq 7} \geq 0.65. \tag{26}$$

The real Paper 60V residual produced

$$N_{\text{strict}}^{\text{real}} = 1980 \tag{27}$$

strict balanced candidates. Paper 60W uses this as the comparison target for adversarial null testing.

The primary falsifier is therefore not a high cosine score. The primary falsifier is

$$N_{\text{strict}}^{\text{null}} \approx N_{\text{strict}}^{\text{real}}. \tag{28}$$

If a genuinely destructive null produces a strict balanced class comparable to the real residual, then the Paper 60V diagnostic signature is not specific enough. If destructive nulls collapse while morphology-preserving controls survive, the result supports the interpretation that v3F is sensitive to retained BAO-band morphology.

2.2 Destructive Nulls

A destructive null is designed to damage the residual's phase, ordering, spacing, or local morphology. These nulls should not preserve the full Paper 60V strict balanced signature.

Paper 60W uses three primary destructive null families: sign shuffle, value shuffle, and Fourier phase randomization. It also uses short block shuffles as local-morphology destructive nulls, as described in Section 2.3.

2.2.1 Sign-Shuffled Residuals

The sign-shuffle null preserves the magnitudes of residual values on the same k-grid but randomly flips their signs:

$$R_{\text{null}}(k_i) = s_i R_{\text{raw}}(k_i), \qquad s_i \in \{-1, +1\}. \tag{29}$$

This null disrupts phase coherence while retaining pointwise residual scale. It is intended to destroy organized peak/trough structure without replacing the residual by unrelated synthetic data.

A strong v3F result should collapse under sign shuffle. If sign-shuffled residuals frequently pass

the strict balanced criteria, then the selector may be responding to amplitude scale rather than organized BAO-band morphology.

2.2.2 Value-Shuffled Residuals

The value-shuffle null permutes the residual values across the k-grid:

$$R_{\rm null}(k_i) = R_{\rm raw}(k_{\pi(i)}), \tag{30}$$

where π is a random permutation. This preserves the residual value distribution but destroys the original ordering of peaks and troughs.

A strong v3F result should also collapse under value shuffle. If value-shuffled residuals pass the strict balanced criteria, then the model may be sensitive only to the marginal residual distribution rather than to coherent waveform structure.

2.2.3 Fourier-Phase-Randomized Residuals

The Fourier-phase-randomized null preserves the Fourier amplitudes of the centered residual while randomizing Fourier phases:

$$\widehat{R}_{\rm null}(q) = |\widehat{R}(q)| \exp(i\theta_q), \tag{31}$$

where θ_q is randomized subject to the Hermitian symmetry required for a real-valued inverse transform.

This is an important adversarial control because it preserves the residual's frequency-amplitude inventory while destroying the original phase organization. In the corrected all-in-one suite, no post-hoc standard-deviation normalization is applied after Fourier phase randomization. This avoids artificially modifying the Fourier-amplitude-preserving property of the null.

If the Paper 60V signature survives Fourier phase randomization, then phase compatibility is not specific to the real residual. If it collapses, the result supports the claim that the v3F selector depends on organized phase relations rather than on frequency content alone.

2.3 Block-Shuffle Nulls and Local Morphology

Block shuffling tests whether local waveform fragments are sufficient for v3F survival. The residual is divided into short contiguous blocks and the blocks are randomly permuted. In schematic form,

$$R_{\rm raw} = (B_1, B_2, \ldots, B_m) \quad \longrightarrow \quad R_{\rm null} = (B_{\pi(1)}, B_{\pi(2)}, \ldots, B_{\pi(m)}). \tag{32}$$

Unlike sign shuffle or value shuffle, block shuffle preserves local residual structure inside each block. It therefore interpolates between a destructive null and a morphology-preserving control. Small block sizes should destroy local waveform structure more strongly. Larger block sizes may preserve enough peak/trough morphology for the v3F selector to survive.

Paper 60W tests three block sizes:

$$b \in \{2, 3, 4\}. \tag{33}$$

The expected pattern is:

$$b = 2, 3 \quad \text{should behave as destructive local-morphology nulls,} \tag{34}$$

$$b = 4 \quad \text{may become a partial morphology-preserving control.} \tag{35}$$

This block-size sweep is important because preliminary testing showed that some block-shuffle controls can preserve the real band-limited target too well to be considered destructive. Paper 60W therefore does not treat all block shuffles as equivalent. It asks whether v3F survival depends on preserved local morphology length.

The block-size interpretation is deliberately bounded. The value $b = 4$ is not assumed to be a universal physical BAO length, nor is it claimed to be a derived correlation scale. It is an empirical local-morphology threshold in the declared BOSS DR12 k-window and preprocessing pipeline. A future sampling study should convert block size into physical k-scale using the actual input grid and test whether the transition tracks a stable fraction of the carrier or sideband period across samples.

2.4 Morphology-Preserving Controls

Not every null-like transformation is expected to destroy the v3F signature. Some controls deliberately preserve part of the waveform structure. These controls are useful because they test whether the v3F selector is morphology-sensitive.

2.4.1 Circular-Shift Controls

The circular-shift control shifts the residual along the k-grid:

$$R_{\text{null}}(k_i) = R_{\text{raw}}(k_{i-s}), \tag{36}$$

with indices interpreted cyclically.

This control preserves the global waveform shape but changes its alignment. It is therefore not a fully destructive null. A circular shift may have poor direct phase alignment with the real target while still preserving autocorrelation, peak spacing, harmonic support, and peak count.

For this reason, circular-shift survivors are not automatically false positives. They are interpreted as phase-shifted morphology controls. If circular-shift controls survive, the correct conclusion is not that the selector fits arbitrary noise, but that the selector remains sensitive to preserved oscillatory structure even when direct alignment changes.

2.5 Morphology Audit Requirement

A key lesson from the preliminary null tests is that strict balanced count alone is not enough. A null/control input must also be audited for how much real morphology it preserves before its survival can be interpreted.

For each null/control input, Paper 60W compares the null band-limited target $B_{\mathrm{null}}(k)$ to the real band-limited target $B_{\mathrm{real}}(k)$. The audit computes:

- direct band cosine similarity,
- maximum shifted band cosine similarity,
- autocorrelation similarity,
- harmonic-support similarity,
- peak count,
- peak-spacing similarity,
- dominant harmonic,
- high-n tail mass.

This audit separates dangerous null survivors from benign morphology-preserving survivors. A dangerous null survivor would satisfy:

$$N_{\mathrm{strict}}^{\mathrm{null}} > 0, \tag{37}$$

$$\cos(B_{\mathrm{null}}, B_{\mathrm{real}}) \ll 1, \tag{38}$$

$$\max_{s} \cos(B_{\mathrm{null}}(k+s), B_{\mathrm{real}}(k)) \ll 1, \tag{39}$$

while also failing to preserve peak spacing, autocorrelation, or harmonic support.

By contrast, a morphology-preserving survivor is expected to have high similarity in one or more structure metrics. Such a survivor does not falsify the Paper 60V mechanism. It shows that the v3F selector responds to retained BAO-band waveform structure.

2.6 Interpretive Categories

Paper 60W classifies null/control families into three practical categories.

First, a family is *morphology-destroying* if it destroys direct band similarity, shifted similarity, peak spacing, and harmonic support relative to the real target. Such a family should produce zero or near-zero strict balanced candidates.

Second, a family is *partly preserving* if it damages direct alignment but preserves some combination of spacing, autocorrelation, or harmonic support. Such a family may occasionally pass some v3F

criteria but should not produce a broad strict balanced class unless it preserves substantial BAO-band structure.

Third, a family is *structure-preserving* if it retains near-real morphology under the audit metrics. Survival in this category is not a failure of the v3F selector. It is expected behavior for a morphology-sensitive diagnostic.

The paper-facing logic is therefore:

> A destructive null should collapse. A morphology-preserving control may survive. The v3F selector is supported if survival occurs only when the null/control preserves substantial BAO-band waveform structure.

2.7 Null Design Summary

The null design of Paper 60W is deliberately adversarial but not indiscriminate. The destructive nulls test whether the Paper 60V signature survives phase, spacing, and ordering destruction. The block-size sweep tests how much local morphology is required for survival. The circular-shift controls test whether the selector responds to preserved oscillatory structure under phase displacement.

The expected reinforcing outcome is:

$$N_{\text{strict}}^{\text{destroying null}} \approx 0, \tag{40}$$

while

$$N_{\text{strict}}^{\text{structure preserving control}} \quad \text{may be nonzero.} \tag{41}$$

Thus, Paper 60W does not ask whether every null-like transformation eliminates the v3F signature. It asks the more precise question: does the signature collapse when morphology is destroyed, and survive only when morphology is retained?

3 Methods

3.1 Overview

Paper 60W uses a corrected all-in-one pipeline to generate null/control inputs, run the Paper 60V v3F scan on each input, summarize strict balanced candidate counts, and audit the morphology of each null/control target before interpretation. The purpose is to test whether the v3F strict balanced signature survives arbitrary scrambling or only survives when BAO-band morphology is retained.

The workflow has six stages:

1. load the real BOSS DR12 Fourier-space monopole data,
2. construct the real-data broadband baseline and residual,

3. generate real-data-derived null/control residuals,
4. write each null/control as a BAO-like input file,
5. run the Paper 60V v3F phase-coupled tail scan on each file,
6. audit each null/control target against the real band-limited target.

The pipeline is real-data-derived throughout. No synthetic fallback data are used. The null/control inputs are transformations of the measured BOSS DR12 residual, not substitutes for missing data. If the real BOSS DR12 input file is unavailable, the script stops rather than generating substitute data.

3.2 Input Data and Analysis Window

The real input dataset is the BOSS DR12 Fourier-space monopole file

`Beutleretal_pk_monopole_DR12_NGC_z1_postrecon_120.dat`.

The analysis window is the same as Paper 60V:

$$0.015 \leq k \leq 0.145. \tag{42}$$

The all-in-one suite records the resolved real input path, output directory, Python version, null configuration, v3F scan configuration, and generated artifacts in JSON and text receipt files. The production run used:

$$N_{\mathrm{dense}} = 8192, \tag{43}$$
$$n_{\max} = 12, \tag{44}$$
$$N_{\mathrm{paths}} = 12000. \tag{45}$$

The run was executed with 12 parallel workers. Each worker ran one v3F scan at a time while process-level numerical thread counts were held to one worker-local thread where applicable.

3.3 Broadband Baseline and Residual Construction

Let the loaded real power-spectrum samples be

$$\{k_i, P(k_i)\}.$$

The broadband baseline follows the Paper 60V convention: a fourth-degree polynomial is fit to $\log P(k)$ over the selected k-window,

$$\log P(k) \approx q_4(k), \tag{46}$$

where q_4 is a degree-four polynomial in a standardized k-coordinate.

The broadband baseline is

$$P_{\text{base}}(k) = \exp[q_4(k)]. \tag{47}$$

The real residual is

$$R_{\text{real}}(k) = P(k) - P_{\text{base}}(k). \tag{48}$$

Each null/control residual $R_{\text{null}}(k)$ is generated from this real residual. A null/control power spectrum is then written as

$$P_{\text{null}}(k) = P_{\text{base}}(k) + R_{\text{null}}(k). \tag{49}$$

For comparability with Paper 60V, each generated null/control file is then passed into the same v3F script as if it were an input monopole file. The v3F script refits its declared fourth-degree log-polynomial baseline and reconstructs its own residual target from that file. This makes the null/control scans operationally comparable to the original Paper 60V run.

This choice also means that the null suite tests the full operational pipeline, not only an isolated residual vector. If the baseline refit or target construction makes a null/control resemble the real band-limited target, that resemblance is detected by the morphology audit before the v3F survivor is interpreted.

3.4 Corrected Null Generation

The all-in-one suite differs from the preliminary null generator in two important ways.

First, Fourier phase randomization preserves the Fourier amplitudes of the centered residual and does not apply post-hoc standard-deviation normalization after inverse transformation. This avoids altering the Fourier-amplitude-preserving property of the null.

Second, null residuals are not globally rescaled to match the real residual standard deviation by default. The transformations are allowed to carry their natural variance consequences. Mean alignment is retained where appropriate so that the null residual remains compatible with the broadband baseline construction.

The corrected null suite therefore avoids making all nulls artificially comparable by amplitude. This makes the null tests more transparent: if a null survives, it should survive because it retains relevant morphology, not because post-processing forced it into a favorable amplitude envelope.

The corrected generator also records diagnostics intended to catch code artifacts, including block-permutation status, unchanged block positions, residual cosine relative to the real residual before file writing, maximum absolute residual difference, and positivity-clipping counts.

3.5 Null and Control Families

The production all-in-one suite evaluates 168 null/control scans. The family counts are:

$$
\begin{aligned}
&24 \quad \text{sign-shuffled residuals,} && (50)\\
&24 \quad \text{value-shuffled residuals,} && (51)\\
&24 \quad \text{Fourier-phase-randomized residuals,} && (52)\\
&24 \quad \text{circular-shift controls,} && (53)\\
&24 \quad \text{block-shuffle residuals with block size 2,} && (54)\\
&24 \quad \text{block-shuffle residuals with block size 3,} && (55)\\
&24 \quad \text{block-shuffle residuals with block size 4.} && (56)
\end{aligned}
$$

Thus,

$$4 \times 24 + 3 \times 24 = 168 \tag{57}$$

v3F scans are performed.

Each generated null/control file records the null family, seed, residual statistics, clipping diagnostics, and any family-specific information such as circular-shift offset or block-shuffle permutation diagnostics.

3.6 Sign-Shuffle Generation

For sign shuffle, the real residual values are multiplied by random signs:

$$R_{\text{null}}(k_i) = s_i R_{\text{real}}(k_i), \qquad s_i \in \{-1, +1\}. \tag{58}$$

This preserves residual magnitudes on the same grid but disrupts coherent phase organization. The null is real-data-derived because it uses the real residual values and only changes their sign structure.

3.7 Value-Shuffle Generation

For value shuffle, the residual values are randomly permuted:

$$R_{\text{null}}(k_i) = R_{\text{real}}(k_{\pi(i)}), \tag{59}$$

where π is a random permutation. This preserves the residual value distribution but destroys the original ordering along k.

3.8 Fourier-Phase-Randomized Generation

For Fourier phase randomization, the centered real residual

$$R_0(k_i) = R_{\text{real}}(k_i) - \langle R_{\text{real}} \rangle$$

is transformed into Fourier space:

$$\widehat{R}_0(q) = \mathcal{F}[R_0](q). \tag{60}$$

The null Fourier coefficients are constructed as

$$\widehat{R}_{\text{null}}(q) = |\widehat{R}_0(q)| \exp(i\theta_q), \tag{61}$$

where θ_q is randomized subject to the Hermitian symmetry required for a real-valued inverse transform.

The inverse transform produces the null residual:

$$R_{\text{null}}(k_i) = \mathcal{F}^{-1}[\widehat{R}_{\text{null}}](k_i) + \langle R_{\text{real}} \rangle. \tag{62}$$

No post-hoc standard-deviation matching is applied after inverse transformation.

3.9 Circular-Shift Generation

For circular shift, the residual sequence is shifted cyclically:

$$R_{\text{null}}(k_i) = R_{\text{real}}(k_{i-s}), \tag{63}$$

where s is a randomly selected shift and indices are interpreted cyclically.

This is not treated as a fully destructive null. It preserves the global waveform, residual value distribution, autocorrelation, and local peak/trough pattern, while altering direct phase alignment with the real target. Circular shift is therefore interpreted as a phase-shifted morphology control.

3.10 Block-Shuffle Generation

For block shuffle, the real residual is divided into contiguous blocks of length b:

$$R_{\text{real}} = (B_1, B_2, \ldots, B_m). \tag{64}$$

The blocks are then randomly permuted:

$$R_{\text{null}} = (B_{\pi(1)}, B_{\pi(2)}, \ldots, B_{\pi(m)}). \tag{65}$$

Paper 60W tests:

$$b \in \{2, 3, 4\}. \tag{66}$$

The all-in-one suite records whether a block permutation is the identity, how many block positions remain unchanged, the residual cosine relative to the real residual before file writing, and the maximum absolute residual difference. Identity permutations are avoided when possible.

This diagnostic matters because block shuffle can become a morphology-preserving control rather than a destructive null if the blocks are long enough to preserve peak/trough structure.

The block size b refers to adjacent samples on the original input k-grid inside the selected analysis window, not to points on the dense interpolation grid. Therefore, $b = 4$ should not be interpreted directly as a universal physical length. It is a pipeline-level local-morphology scale that must be converted to physical k-extent and retested across samples before it can be given a stronger physical interpretation.

3.11 v3F Scan Settings

Each null/control input is passed to the same Paper 60V v3F scan script. The key settings are:

$$\epsilon_{\text{step}} \in \{0.08\}, \tag{67}$$
$$\delta_{\text{budget}} \in \{0.26, 0.30, 0.34\}, \tag{68}$$
$$w_{\mathcal{H}} \in \{3\}, \tag{69}$$
$$N_{\text{paths}} = 12000, \tag{70}$$
$$n_{\max} = 12. \tag{71}$$

The transfer settings are:

$$p_T \in \{1.75\}, \tag{72}$$
$$m_B \in \{0.75\}, \tag{73}$$
$$T_{\text{floor}} \in \{0.02\}. \tag{74}$$

The tested transfer modes are:

$$\{\texttt{observer\_product}, \texttt{phase\_tail\_sink}, \texttt{phase\_tail\_blend}, \texttt{phase\_tail\_only}, \texttt{phase\_tail\_boost\_only}\}. \tag{75}$$

The tested phase-reference modes are:

$$\{\texttt{carrier\_multiple}, \texttt{sideband\_multiple}, \texttt{nearest\_carrier\_or\_sideband}, \texttt{parity\_pair}\}. \tag{76}$$

The phase-lock grid is:

$$\lambda_{\text{lock}} \in \{0.5, 1.0, 2.0, 4.0\}, \tag{77}$$
$$L_{\text{floor}} \in \{0.05, 0.15, 0.30, 0.50\}, \tag{78}$$
$$\lambda_{\text{incoh}} \in \{0.5, 1.0, 2.0\}, \tag{79}$$
$$\beta_{\text{coh}} \in \{0.0, 0.15, 0.30\}. \tag{80}$$

The v3F scan records the best row, balanced candidates, transfer-mode summaries, phase-reference summaries, ladder files, spectra files, and diagnostic figures for each null/control input.

3.12 Strict Balanced Count

For each null/control scan, the all-in-one suite records the number of strict balanced candidates:

$$N_{\text{strict}}^{\text{null}}. \tag{81}$$

The real Paper 60V comparison value is

$$N_{\text{strict}}^{\text{real}} = 1980. \tag{82}$$

The principal diagnostic is whether each null/control family produces:

$$N_{\text{strict}}^{\text{null}} \approx 0 \tag{83}$$

or a nonzero survivor class.

A nonzero survivor class is not interpreted by itself. It is interpreted only after morphology auditing.

3.13 Morphology Audit

For every null/control input, the all-in-one suite reconstructs the null target using the same residual preprocessing and harmonic projection convention as the real target. It then compares the null band-limited target B_{null} to the real band-limited target B_{real}.

The audit computes direct band cosine:

$$C_{\text{band}} = \cos(B_{\text{null}}, B_{\text{real}}). \tag{84}$$

It also computes a shifted cosine:

$$C_{\text{shift}} = \max_{s} \cos(B_{\text{null}}(k+s), B_{\text{real}}(k)), \tag{85}$$

where s ranges over allowed circular shifts.

The autocorrelation similarity is computed by comparing the autocorrelation profiles of the real and null band-limited targets:

$$C_{\text{AC}} = \cos(\text{AC}_{\text{null}}, \text{AC}_{\text{real}}). \tag{86}$$

The harmonic-support similarity is computed by comparing the normalized harmonic support vectors:

$$C_{\text{harm}} = \cos(\mathbf{S}_{\text{null}}, \mathbf{S}_{\text{real}}), \tag{87}$$

where $\mathbf{S}$ contains the normalized harmonic amplitudes through $n_{\max} = 12$.

The audit also records:

- target peak count,

- median peak spacing,
- spacing similarity,
- dominant harmonic,
- carrier mass $3 \le n \le 4$,
- sideband mass $5 \le n \le 6$,
- high-n tail mass $n \ge 7$.

This audit is required because a null/control that preserves real morphology should not be interpreted as a destructive null.

3.14 Morphology Classes

The all-in-one suite classifies each null/control target into one of three morphology classes.

A target is classified as *structure-preserving* if it has high direct or shifted similarity to the real band-limited target:

$$C_{\text{band}} \ge 0.80 \quad \text{or} \quad C_{\text{shift}} \ge 0.90. \tag{88}$$

A target is classified as *partly preserving* if it has moderate direct similarity, good spacing similarity, and near-real peak count:

$$C_{\text{band}} \ge 0.60, \qquad S_{\text{spacing}} \ge 0.80, \qquad |N_{\text{peak}}^{\text{null}} - N_{\text{peak}}^{\text{real}}| \le 2. \tag{89}$$

All other valid targets are classified as *morphology-destroying*.

These classes are interpretive aids, not additional model-fitting criteria. The main use is to distinguish dangerous null survivors from benign morphology-preserving controls.

3.15 Generated Artifacts

The all-in-one production run writes the following primary artifacts:

- `paper60W_all_command.txt`
- `paper60W_all_config.json`
- `paper60W_all_null_manifest.csv`
- `paper60W_all_null_manifest.json`
- `paper60W_all_v3F_final_summary.csv`
- `paper60W_all_v3F_by_family_summary.csv`

- `paper60W_all_morphology_audit.csv`
- `paper60W_all_morphology_by_family_summary.csv`
- `paper60W_all_survivors.csv`
- `paper60W_all_interpretation.txt`
- `paper60W_all_receipt.json`

Each individual v3F null/control run also writes its own scan summary, balanced-candidate file, best-gate summary, ladder files, spectra files, and figures in its run-specific output directory.

Tables in the manuscript are custom paper tables constructed from the generated CSV and text artifacts after review and formatting for clarity.

3.16 Reproducibility Scope

The all-in-one run was executed on Windows with Python 3.11.9. The run configuration records:

- real BOSS DR12 input path,
- output root directory,
- number of workers,
- null family counts,
- block sizes,
- k-window,
- baseline degree,
- dense grid size,
- maximum harmonic order,
- path count,
- seed,
- generated artifacts.

The production run generated and scanned 168 null/control inputs. The run completed with 168 v3F results and 4 v3F survivor rows with nonzero strict balanced counts.

The method is reproducible from the declared real BOSS DR12 input file and the all-in-one script included in the appendix.

4 Results

4.1 Overview of the Corrected Null Suite

The corrected all-in-one suite evaluated 168 real-data-derived null/control scans. Each null/control input was passed through the same Paper 60V v3F phase-coupled tail scan and then audited for morphology preservation before interpretation.

The real Paper 60V residual produced

$$N_{\text{strict}}^{\text{real}} = 1980$$

strict balanced candidates. This value is the comparison target for the null suite.

The corrected suite produced only 4 survivor runs with nonzero strict balanced counts. These survivors occurred only in morphology-preserving or phase-shifted control families:

- one block-shuffle run with block size 4,
- three circular-shift runs.

All destructive null families collapsed to zero strict balanced candidates.

4.2 Strict Balanced Counts by Null Family

Table 1 summarizes the v3F strict balanced counts by null/control family.

Table 1: v3F strict balanced counts by null/control family. The real Paper 60V residual produced 1980 strict balanced candidates. Destructive nulls collapse to zero; survival occurs only in block-size 4 and circular-shift controls.

Family	Runs	Total strict rows	Max strict rows	Runs with strict rows
Sign shuffle	24	0	0	0
Value shuffle	24	0	0	0
Fourier phase randomized	24	0	0	0
Block shuffle, $b = 2$	24	0	0	0
Block shuffle, $b = 3$	24	0	0	0
Block shuffle, $b = 4$	24	643	643	1
Circular shift	24	1284	428	3

The most important result is the complete collapse of the destructive null families:

$$N_{\text{strict}}^{\text{sign}} = 0, \tag{90}$$
$$N_{\text{strict}}^{\text{value}} = 0, \tag{91}$$
$$N_{\text{strict}}^{\text{Fourier}} = 0, \tag{92}$$
$$N_{\text{strict}}^{\text{block},\, b=2} = 0, \tag{93}$$
$$N_{\text{strict}}^{\text{block},\, b=3} = 0. \tag{94}$$

These collapses occur even though the best scan rows within those null families can still achieve high band-limited cosine similarity. Therefore, high cosine alone is not sufficient to reproduce the Paper 60V signature. The full joint condition is more selective.

4.3 Best-Row Scores Are Not Enough

Several null families produced high best-row band-limited cosine similarity despite having zero strict balanced candidates. This is important because it shows why Paper 60W does not use cosine similarity alone as the falsification metric.

Table 2 reports the maximum and mean best-row band cosine by family.

Table 2: Best-row band-limited cosine similarity by null/control family. High best-row cosine can occur even when the strict balanced class collapses. This confirms that the Paper 60V signature is not equivalent to high cosine similarity alone.

Family	Max best cosine	Mean best cosine	Total strict rows
Sign shuffle	0.993226	0.974541	0
Value shuffle	0.995540	0.975807	0
Fourier phase randomized	0.989297	0.970799	0
Block shuffle, $b = 2$	0.997290	0.968297	0
Block shuffle, $b = 3$	0.988727	0.968523	0
Block shuffle, $b = 4$	0.987859	0.957730	643
Circular shift	0.991992	0.972456	1284

The destructive null families can produce high best cosine but still fail the strict balanced criteria. The failure therefore comes from the joint signature: peak count, spacing, transferred $n = 3$ selection, low incoherent tail, and high coherent-tail fraction must all hold simultaneously.

This is one of the strongest safeguards against the curve-fitting objection. Several destructive nulls achieve high best-row cosine values, in some cases above 0.99, yet still produce zero strict balanced candidates. Therefore, the v3F signature is not equivalent to minimizing an L^2-type waveform error or maximizing a single cosine score. It is a joint topological/morphological condition involving peak count, spacing, transferred carrier selection, and coherent/incoherent tail structure. In this sense, Paper 60W supports the view that morphology carries information not captured by amplitude agreement alone.

This confirms the central methodological point of Paper 60W:

The null test must evaluate the full strict balanced signature, not only waveform similarity.

4.4 Block-Size Threshold

The block-shuffle family gives the cleanest local-morphology result. Block sizes 2 and 3 collapse completely, while block size 4 produces one survivor.

Table 3: Block-shuffle threshold result. Short block sizes destroy enough local morphology that the strict balanced class collapses. Block size 4 admits one survivor, which the morphology audit identifies as structure-preserving.

Block size	Runs	Runs with strict rows	Total strict rows
2	24	0	0
3	24	0	0
4	24	1	643

This result supports a local morphology threshold interpretation. The v3F signature does not survive arbitrary block scrambling. It survives only when the block shuffle preserves enough local waveform structure to remain close to the real band-limited target.

The block-size 4 survivor is therefore not interpreted as a destructive-null failure. It is interpreted as a morphology-preserving control.

4.5 Sampling Interpretation of the Block-Size Threshold

The block-size threshold should be interpreted carefully. In this suite, block shuffling is applied to the residual sampled on the original input k-grid within the selected analysis window, not to the dense interpolation grid used later for band-limited reconstruction. Thus, a block size b represents b adjacent original residual samples.

The observed transition,

$$b = 2, 3 \quad \Rightarrow \quad N_{\text{strict}} = 0,$$

while

$$b = 4 \quad \Rightarrow \quad N_{\text{strict}} > 0 \quad \text{in one morphology-preserving survivor,}$$

suggests that v3F requires more than isolated pointwise fluctuations or very short local fragments. It begins to survive only when the shuffled blocks preserve a longer local segment of the real waveform.

This is consistent with a sampling-theoretic interpretation: a selector that depends on carrier/sideband morphology should fail when blocks are too short to preserve a recognizable fraction of a peak–trough pattern, and should begin to survive when a block retains enough local phase shape to preserve peak count, spacing, harmonic support, or autocorrelation. However, Paper 60W

does not claim that $b = 4$ is a universal physical constant or a derived BAO correlation length. It is an empirical threshold in the declared BOSS DR12 window and preprocessing pipeline.

4.6 Morphology Audit by Family

The morphology audit compares each null/control target against the real band-limited target before interpreting the v3F result. Table 4 summarizes the audit by family.

Table 4: Morphology audit by null/control family. Destructive families are classified as morphology-destroying. Survival occurs only in families that preserve either strong local morphology or phase-shifted global structure.

Family	Runs	Destroying	Partly preserving	Structure preserving	Strict survivors
Sign shuffle	24	23	1	0	0
Value shuffle	24	23	0	1	0
Fourier phase randomized	24	24	0	0	0
Block shuffle, $b = 2$	24	24	0	0	0
Block shuffle, $b = 3$	24	24	0	0	0
Block shuffle, $b = 4$	24	23	0	1	1
Circular shift	24	24	0	0	3

The block-size 4 result is especially clear: the only v3F survivor is also the only structure-preserving block-size 4 target. This supports the claim that block-shuffle survival depends on retained morphology.

The circular-shift family is different. Its survivors are classified as morphology-destroying by direct/shifted band-cosine thresholds, but they preserve strong autocorrelation, harmonic support, spacing, and peak count. For this reason, circular shift is interpreted separately as a phase-shifted morphology control rather than a purely destructive null.

4.7 Morphology Similarity Metrics

Table 5 reports the mean morphology-audit metrics by family. These metrics explain why the surviving families should not be interpreted as arbitrary false positives.

Table 5: Mean morphology-audit metrics by null/control family. The block-size 4 and circular-shift survivors occur in families with high autocorrelation, harmonic-support, or spacing preservation relative to destructive nulls.

Family	Mean band cos.	Mean shifted cos.	Mean autocorr. cos.	Mean harmonic cos.	Mean spacing sim.
Sign shuffle	−0.124878	0.474074	0.988565	0.726724	0.531460
Value shuffle	0.128032	0.439977	0.987246	0.694202	0.474186
Fourier phase randomized	−0.051647	0.438745	0.988694	0.737517	0.562759
Block shuffle, $b = 2$	0.051273	0.392805	0.989702	0.729540	0.594659
Block shuffle, $b = 3$	−0.067851	0.459029	0.991533	0.785180	0.640191
Block shuffle, $b = 4$	−0.182093	0.516084	0.993537	0.823222	0.732190
Circular shift	−0.077103	0.397820	0.987191	0.732461	0.820383

The table shows that direct band cosine alone is not the whole morphology story. Circular-shift controls have low mean direct band cosine but high spacing preservation. Block-size 4 has stronger harmonic-support and spacing preservation than block sizes 2 and 3, consistent with the observed survival threshold.

4.8 Surviving Controls

Only four null/control runs produced nonzero strict balanced counts. They are shown in Table 6.

Table 6: Surviving null/control runs. The block-size 4 survivor preserves the real morphology closely. The circular-shift survivors preserve spacing, autocorrelation, harmonic support, and peak count despite poor direct phase alignment.

Run	Family	Strict rows	Band cos.	Harmonic cos.	Spacing sim.
150	Block shuffle, $b = 4$	643	0.989109	0.999091	0.953869
74	Circular shift	428	−0.267662	0.889042	0.996856
85	Circular shift	428	−0.267662	0.889042	0.996856
93	Circular shift	428	−0.267662	0.889042	0.996856

The block-size 4 survivor is the cleanest example of morphology-preserving survival:

$$\cos(B_{\text{null}}, B_{\text{real}}) = 0.989109, \tag{95}$$

$$\max_s \cos(B_{\text{null}}(k+s), B_{\text{real}}(k)) = 0.989136, \tag{96}$$

$$C_{\text{AC}} = 0.999978, \tag{97}$$

$$C_{\text{harm}} = 0.999091, \tag{98}$$

$$S_{\text{spacing}} = 0.953869, \tag{99}$$

$$N_{\text{peak}}^{\text{null}} = 10, \tag{100}$$

$$n_{\text{dom}}^{\text{null}} = 10. \tag{101}$$

This target is not a destructive null. It retains nearly the same band-limited morphology as the real target. Its survival therefore reinforces the morphology-sensitive interpretation of v3F.

The circular-shift survivors are also not arbitrary. They have poor direct band alignment:

$$\cos(B_{\text{null}}, B_{\text{real}}) = -0.267662,$$

but preserve:

$$C_{\text{AC}} = 0.994706, \tag{102}$$

$$C_{\text{harm}} = 0.889042, \tag{103}$$

$$S_{\text{spacing}} = 0.996856, \tag{104}$$

$$N_{\text{peak}}^{\text{null}} = 10. \tag{105}$$

These are phase-shifted morphology controls, not random residuals.

4.9 No Dangerous Destructive-Null Survivors

A dangerous survivor would be a null/control input that satisfies two conditions:

$$N_{\text{strict}}^{\text{null}} > 0, \tag{106}$$

$$\text{morphology preservation} \approx 0. \tag{107}$$

The corrected suite does not show such a case. Every nonzero strict balanced survivor either preserves near-real band-limited morphology directly or preserves strong phase-shifted morphology through autocorrelation, harmonic support, spacing, and peak count.

The destructive nulls produce zero strict balanced candidates. This includes:

- all 24 sign-shuffle nulls,
- all 24 value-shuffle nulls,
- all 24 Fourier-phase-randomized nulls,

- all 24 block-size 2 shuffles,
- all 24 block-size 3 shuffles.

This is the central empirical result of Paper 60W.

4.10 Results Summary

The corrected null suite supports the diagnostic claim of Paper 60V.

First, the destructive null families collapse completely:

$$N_{\text{strict}}^{\text{destroying null}} = 0.$$

Second, high best-row band cosine is not enough. Several null families achieve high best-row cosine values while producing zero strict balanced candidates.

Third, block-shuffle survival depends on local morphology length. Block sizes 2 and 3 collapse, while block size 4 produces one survivor that the morphology audit identifies as structure-preserving.

Fourth, circular-shift controls can survive because they preserve spacing, autocorrelation, harmonic support, and peak count despite poor direct phase alignment.

The main result is therefore:

> The v3F strict balanced signature does not survive arbitrary residual scrambling. It collapses when phase, ordering, or local morphology are destroyed, and survives only when the null/control preserves substantial BAO-band waveform structure. Paper 60W therefore reinforces Paper 60V's diagnostic claim: the phase-compatible high-order tail selector is morphology-sensitive, not a generic band-limited noise fitter.

5 Discussion

5.1 What the Null Suite Establishes

Paper 60W addresses the main vulnerability of Paper 60V: whether the v3F strict balanced signature could be reproduced by arbitrary band-limited residual scrambling. The corrected null suite shows that it cannot. Destructive null families collapse, while survival occurs only when substantial BAO-band waveform structure is preserved.

The result is therefore not simply:

all nulls fail.

The more accurate statement is:

destructive nulls fail, while morphology-preserving controls may survive.

This distinction is important. A diagnostic designed to detect morphology should not survive when phase, ordering, and local waveform structure are destroyed. But it should remain sensitive when a control retains the morphology that the diagnostic is designed to detect. That is the pattern observed here.

The result therefore supports a bounded interpretation: the v3F selector is not responding to arbitrary scrambled residual structure. It is responding to retained BAO-band waveform morphology under the declared diagnostic protocol.

5.2 Why This Reinforces Paper 60V

Paper 60V argued that the high-order BAO residual tail is not uniformly waste. Its key claim was that high-order structure must be decomposed into phase-compatible and phase-incoherent components:

$$M_{n\geq 7}^{\mathrm{tail}} = M_{n\geq 7}^{\mathrm{coh}} + M_{n\geq 7}^{\mathrm{incoh}}.$$

A natural criticism is that this decomposition might be too flexible. If the phase-compatible tail selector produced strict balanced candidates for sign-shuffled, value-shuffled, Fourier-phase-randomized, or short-block-shuffled residuals, then the Paper 60V result would be weakened.

The corrected suite does not show that failure. The destructive nulls produce zero strict balanced candidates. This means the v3F strict signature is not equivalent to a flexible high-cosine curve fit. It requires a joint morphology/tail condition: band-limited morphology, peak spacing, peak count, transferred $n = 3$ selection, low incoherent high-n tail, and high coherent-tail fraction must hold together.

This reinforces the central diagnostic claim of Paper 60V. The high-order tail split is not merely an after-the-fact labeling of arbitrary high-n mass. Under destructive nulls, the joint signature collapses.

5.3 High Cosine Is Not the Signal

Several destructive null families achieve high best-row band-limited cosine similarity, but still produce zero strict balanced candidates. This is one of the strongest safeguards against the curve-fitting objection.

The result shows that

$$\mathrm{cos}_{\mathrm{band}}$$

is not the decisive signal by itself. The decisive object is the joint strict balanced signature:

$$\mathrm{cos}_{\mathrm{band}} + S_{\mathrm{spacing}} + N_{\mathrm{peak}}^{\mathrm{model}} + n_{\mathrm{transfer}} + M_{n\geq 7}^{\mathrm{incoh}} + f_{\mathrm{coh}}^{n\geq 7}.$$

The destructive nulls can sometimes satisfy one part of this structure, but they do not satisfy the full signature.

Thus, Paper 60W supports the interpretation that morphology and harmonic organization carry information not captured by a single amplitude-agreement score. In practical terms, the v3F

diagnostic is closer to a morphology-and-selection test than to an L^2-style curve-matching score.

This is important for the broader Holosphere interpretation. The relevant observable is not just a smooth waveform resemblance. It is the simultaneous survival of carrier selection, peak topology, spacing, and coherent/incoherent high-order tail sorting.

5.4 Block-Size Threshold as a Local-Morphology Result

The block-shuffle result is one of the most useful outputs of the paper. Block sizes 2 and 3 collapse completely, while block size 4 produces one survivor. The morphology audit shows that this survivor is not a dangerous false positive; it is a structure-preserving control with near-real band-limited morphology.

This supports a local-morphology threshold interpretation. The v3F selector does not survive isolated point shuffling or very short local fragments. It begins to survive only when a block preserves enough local peak/trough context for the band-limited waveform to remain recognizable.

This interpretation should remain bounded. Paper 60W does not claim that $b = 4$ is a universal physical constant or a derived BAO correlation length. In this paper, $b = 4$ is an empirical threshold in one BOSS DR12 input, one k-window, and one preprocessing pipeline. A future sampling study should convert block size into physical k-scale using the actual input grid and test whether the survival transition tracks a stable fraction of the carrier or sideband period.

5.5 Relation to Sampling and Carrier-Scale Morphology

The block-size threshold provides a useful bridge between the numerical null test and the physical interpretation of the v3F selector. The collapse at $b = 2$ and $b = 3$ shows that preserving isolated amplitudes or very short local fragments is not enough. The single survivor at $b = 4$ appears only when the morphology audit shows near-real band-limited structure.

This suggests that the selector depends on local phase-shape information across multiple neighboring k-samples. In ordinary signal language, a peak or trough cannot be recognized from a single point. It requires a short run of points carrying curvature, spacing, and phase context. The v3F selector appears to behave similarly: it fails when local waveform context is destroyed and survives only when enough local context remains.

It is tempting to interpret the $b = 4$ transition as a sampling threshold associated with the $n = 3$–4 carrier or the $n = 5$–6 sideband. That interpretation is plausible but not yet established. The present run uses one input file, one k-window, and one preprocessing convention. Therefore, Paper 60W treats the block threshold as an empirical local-morphology threshold rather than a derived physical scale.

A stronger future test would measure the effective k-extent of each block and compare it to the carrier and sideband periods in the band-limited target. If the survival transition consistently occurs when blocks contain a fixed fraction of the carrier/sideband cycle, then the block-size result would become a direct sampling-theoretic bridge between the null controls and the underlying transport morphology.

5.6 Circular Shift as a Phase-Shifted Morphology Control

Circular shift is not a fully destructive null. It preserves the global waveform but changes alignment on the k-grid. The circular-shift survivors therefore do not show arbitrary null fitting. They show that the v3F selector remains sensitive when oscillatory structure, spacing, autocorrelation, harmonic support, and peak count are preserved under a phase displacement.

This is why circular-shift survival must be interpreted separately from sign shuffle, value shuffle, Fourier phase randomization, and short block shuffle. The circular-shift controls are phase-shifted morphology controls, not noise controls.

The circular-shift result also clarifies the role of direct band cosine. A shifted waveform can have poor direct alignment with the real target while retaining strong autocorrelation, harmonic support, spacing, and peak count. For a morphology-sensitive diagnostic, this partial survival is expected and informative.

5.7 Target-Informed Status

The result remains diagnostic. The v3F scan uses target-derived band-limited structure and phase information as part of the test. Paper 60W therefore does not prove that the phase-compatible tail rule is an independent first-principles prediction.

The value of the result is narrower and cleaner: under the same diagnostic protocol used in Paper 60V, destructive nulls collapse while morphology-preserving controls may survive. This shows that the strict balanced signature is not produced by arbitrary scrambling. It gives a stronger empirical target for a future path-level derivation.

The next theoretical step remains a derivation from path-level admissibility sums:

$$\mathcal{A}_n = \sum_{\gamma \in \Omega_n^{\mathrm{surv}}} w_\gamma \exp(i\Phi_\gamma).$$

Such a derivation should explain why some high-order path families remain aligned with the carrier or switch-sideband scaffold while others become incoherent leakage.

The block-size result adds one more constraint on that future derivation. The path-level theory should not merely predict that phase-compatible tail exists. It should also explain why finite local waveform context is required to preserve carrier/sideband identity under destructive scrambling.

5.8 Scope and Limits

Paper 60W strengthens the diagnostic status of Paper 60V, but it does not establish cosmological universality. The present result is based on one BOSS DR12 Fourier-space monopole input, one k-window, and one baseline convention. The null suite shows that the Paper 60V signature is not a generic artifact of arbitrary scrambling in this declared setting. It does not yet show that the same collapse/survival pattern holds across all BAO samples or preprocessing choices.

The most important future robustness tests are:

- independent BOSS DR12 samples and redshift bins,
- pre-reconstruction and post-reconstruction comparisons,
- alternative reasonable broadband baselines,
- nearby k-windows,
- configuration-space analogues where comparable preprocessing is possible,
- a sampling study that converts block-size thresholds into physical k-scale.

These are not added as new claims in the present paper. They are listed to clarify the boundary between the current null-control result and future generalization.

5.9 Discussion Summary

Paper 60W reinforces the bounded diagnostic claim of Paper 60V. The strict balanced signature collapses under destructive nulls and survives only in controls that preserve substantial waveform structure.

The key result is:

$$N_{\text{strict}}^{\text{destroying null}} = 0,$$

while

$$N_{\text{strict}}^{\text{morphology preserving control}} \quad \text{may be nonzero.}$$

This is the expected behavior of a morphology-sensitive diagnostic. It would be a problem if arbitrary scrambled residuals produced large strict balanced classes. They do not. It is not a problem if controls preserving the waveform survive. That is exactly what a morphology-sensitive selector should do.

The main conclusion is therefore:

> Paper 60W supports the diagnostic specificity of Paper 60V. The v3F selector is not merely fitting arbitrary band-limited residuals; it requires retained BAO-band morphology. The result remains bounded, but it provides a stronger empirical target for a future path-level derivation of phase-compatible high-order tail selection.

6 Falsifiability

Paper 60W is a falsification-oriented paper. Its purpose is not to add a new physical mechanism beyond Paper 60V, but to test whether the Paper 60V v3F strict balanced signature survives adversarial null and control residuals. The central claim is bounded:

The v3F strict balanced signature collapses when phase, ordering, or local morphology are destroyed, and survives only when substantial BAO-band waveform morphology is retained.

This claim can fail in several direct ways.

6.1 Falsifier 1: Destructive Nulls Produce a Real-Like Strict Class

The primary falsifier is simple. If a genuinely destructive null produces a strict balanced class comparable to the real Paper 60V residual, then the v3F signature is not specific enough.

The real residual produced

$$N_{\text{strict}}^{\text{real}} = 1980. \tag{108}$$

A destructive null would challenge Paper 60V if

$$N_{\text{strict}}^{\text{null}} \approx N_{\text{strict}}^{\text{real}}. \tag{109}$$

A stronger failure would be

$$N_{\text{strict}}^{\text{null}} \geq N_{\text{strict}}^{\text{real}}. \tag{110}$$

The corrected Paper 60W suite does not show this failure for destructive nulls. Sign shuffle, value shuffle, Fourier phase randomization, block shuffle with block size 2, and block shuffle with block size 3 all produce zero strict balanced candidates.

Thus, the primary null test reinforces Paper 60V. It does not falsify it.

6.2 Falsifier 2: High Cosine Alone Predicts Strict Survival

If high band-limited cosine similarity alone predicted strict balanced survival, then the v3F signature would be too weak. It would mean the selector is mainly fitting smooth band-limited curves rather than detecting the joint carrier/sideband/tail structure.

The falsifying pattern would be:

$$\cos_{\text{band}} \text{ high} \quad \Longrightarrow \quad N_{\text{strict}} > 0 \tag{111}$$

for most null/control families.

The corrected suite does not show this pattern. Several destructive null families achieve high best-row band-limited cosine similarity, but still produce zero strict balanced candidates. Therefore,

$$\cos_{\text{band}} \quad \text{is necessary but not sufficient.} \tag{112}$$

The strict balanced signature requires simultaneous morphology, spacing, peak count, transferred $n = 3$ selection, low incoherent high-n tail, and high coherent-tail fraction. The destructive nulls fail that joint condition.

6.3 Falsifier 3: Phase-Randomized Nulls Survive

Fourier phase randomization is an important adversarial test because it preserves the Fourier-amplitude inventory while destroying the original phase organization. If the phase-compatible tail

selector survived this null broadly, then the claimed phase structure would be weakened.

A direct falsifier is:

$$N_{\text{strict}}^{\text{Fourier phase randomized}} > 0 \tag{113}$$

across many runs, especially if those survivors are not morphology-preserving under the audit.

The corrected suite finds:

$$N_{\text{strict}}^{\text{Fourier phase randomized}} = 0. \tag{114}$$

This strengthens the interpretation that the v3F signature depends on organized phase relations rather than on frequency amplitudes alone.

6.4 Falsifier 4: Short-Block Shuffles Survive

Block shuffling tests local morphology preservation. If very short blocks survive, then the v3F selector may be sensitive to short local fragments rather than to coherent BAO-band morphology.

The critical falsifier is:

$$N_{\text{strict}}^{b=2} > 0 \quad \text{or} \quad N_{\text{strict}}^{b=3} > 0 \tag{115}$$

for block-shuffle nulls that are classified as morphology-destroying.

The corrected suite finds:

$$N_{\text{strict}}^{b=2} = 0, \tag{116}$$

$$N_{\text{strict}}^{b=3} = 0. \tag{117}$$

This supports the local-morphology threshold interpretation. Very short block shuffles destroy enough local waveform structure that the strict balanced class collapses.

6.5 Falsifier 5: Block-Size Survival Occurs Without Morphology Preservation

The block-size 4 family produced one survivor. This would be dangerous if the survivor had low morphology similarity to the real target. In that case, the v3F selector would be detecting a scrambled residual as if it were real.

The falsifying pattern would be:

$$N_{\text{strict}}^{b=4} > 0, \tag{118}$$

$$\cos(B_{\text{null}}, B_{\text{real}}) \ll 1, \tag{119}$$

$$C_{\text{harm}} \ll 1, \tag{120}$$

$$S_{\text{spacing}} \ll 1. \tag{121}$$

The corrected suite does not show this failure. The block-size 4 survivor preserves the real morphology

closely:

$$\cos(B_{\text{null}}, B_{\text{real}}) \approx 0.989, \tag{122}$$
$$C_{\text{AC}} \approx 0.999978, \tag{123}$$
$$C_{\text{harm}} \approx 0.999091, \tag{124}$$
$$S_{\text{spacing}} \approx 0.954, \tag{125}$$
$$N_{\text{peak}}^{\text{null}} = 10. \tag{126}$$

Thus, the block-size 4 survivor is not a destructive-null false positive. It is a morphology-preserving control.

6.6 Falsifier 6: Circular Shifts Behave Like Arbitrary Nulls

Circular shift is a phase-shifted morphology control, not a fully destructive null. It preserves the waveform while changing direct alignment. Survival under circular shift is therefore not automatically problematic.

The result would become problematic if circular-shift survivors lacked internal structure:

$$C_{\text{AC}} \ll 1, \tag{127}$$
$$C_{\text{harm}} \ll 1, \tag{128}$$
$$S_{\text{spacing}} \ll 1, \tag{129}$$
$$N_{\text{peak}}^{\text{null}} \neq 10. \tag{130}$$

The corrected suite does not show this failure. The circular-shift survivors preserve strong autocorrelation, harmonic support, spacing, and peak count:

$$C_{\text{AC}} \approx 0.994706, \tag{131}$$
$$C_{\text{harm}} \approx 0.889042, \tag{132}$$
$$S_{\text{spacing}} \approx 0.996856, \tag{133}$$
$$N_{\text{peak}}^{\text{null}} = 10. \tag{134}$$

Their direct band cosine is poor, but their internal waveform structure remains strong. This is consistent with their interpretation as phase-shifted morphology controls.

6.7 Falsifier 7: Morphology Audit Does Not Predict Survival

A core claim of Paper 60W is that v3F survival tracks morphology preservation. This would be weakened if morphology audit metrics failed to separate survivors from non-survivors.

The falsifying pattern would be:

$$N_{\text{strict}} > 0 \quad \text{for morphology-destroying nulls with low audit scores,} \tag{135}$$

or

$$N_{\text{strict}} = 0 \quad \text{for all morphology-preserving controls.} \tag{136}$$

The corrected suite supports the audit interpretation. The destructive families collapse. The block-size 4 survivor is structure-preserving under the audit. The circular-shift survivors preserve spacing, autocorrelation, harmonic support, and peak count.

Therefore, the morphology audit is not decorative. It is necessary for interpreting null survival correctly.

6.8 Falsifier 8: Results Depend on a Coding Artifact

The preliminary null tests raised a legitimate concern: some block-shuffle controls appeared to preserve the real target too perfectly. Paper 60W addresses this by using a corrected all-in-one suite.

A code-artifact failure would be indicated by any of the following:

- identity block permutations mislabeled as shuffled nulls,
- post-hoc standard-deviation normalization creating artificial null strength,
- Fourier phase randomization failing to preserve Fourier amplitudes,
- generated nulls becoming identical to the real target without being flagged,
- v3F summaries being mismatched to the wrong null files.

The corrected suite reduces these risks by:

- avoiding identity block permutations where possible,
- recording block permutation diagnostics,
- removing post-hoc standard-deviation normalization from Fourier phase randomization,
- recording residual cosine and maximum residual difference before file writing,
- joining v3F results back to null files before morphology interpretation.

This does not prove the absence of every possible implementation error, but it substantially reduces the specific artifacts identified during preliminary testing. The relevant claim is therefore not that the code is immune to all implementation mistakes, but that the specific degeneracies detected during the first null suite were addressed before the reported production run.

6.9 Falsifier 9: Independent BAO Samples Do Not Reproduce the Pattern

Paper 60W tests one BOSS DR12 Fourier-space monopole input. The result would be strengthened if the same collapse/survival pattern appears in independent BAO samples.

A future failure would be:

$$N_{\text{strict}}^{\text{destroying null}} > 0 \tag{137}$$

across independent samples, or

$$N_{\text{strict}}^{\text{real}} \approx 0 \tag{138}$$

for real BAO residuals under comparable preprocessing.

The required next samples include:

- BOSS DR12 SGC samples,
- other redshift bins,
- pre-reconstruction spectra,
- alternative post-reconstruction spectra,
- configuration-space residuals where comparable preprocessing is possible.

Paper 60W does not claim that this independent-sample test is complete. It identifies it as a necessary future robustness target.

6.10 Falsifier 10: Baseline Changes Destroy the Pattern

The present paper uses the same fourth-degree log-polynomial baseline convention as Paper 60V. A robust morphology-sensitive result should not depend entirely on that one baseline choice.

A future baseline falsifier is:

$$N_{\text{strict}}^{\text{real}} \to 0 \quad \text{or} \quad N_{\text{strict}}^{\text{destroying null}} > 0 \tag{139}$$

under nearby reasonable baseline choices.

The most direct follow-up tests are:

$$\deg(q) = 3, \tag{140}$$

$$\deg(q) = 5, \tag{141}$$

with the same k-window and strict balanced criteria.

Paper 60W does not complete that baseline sweep. It preserves comparability with Paper 60V and leaves baseline robustness as a required follow-up.

6.11 Falsifier 11: The Block Threshold Does Not Track Physical Scale

The block-size threshold is currently an empirical result in one BOSS DR12 sample and one preprocessing pipeline. It would become physically stronger if the survival transition tracked a stable fraction of the carrier or sideband period across independent datasets.

A future falsifier is:

$$b_{\mathrm{crit}} \quad \text{varies erratically across samples, baselines, or windows,} \tag{142}$$

with no relation to the local sampling density, carrier period, sideband period, peak spacing, or morphology-audit metrics.

Conversely, the interpretation is strengthened if the transition from collapse to survival occurs when the shuffled block contains a stable fraction of the local carrier/sideband waveform. This would suggest that v3F requires a minimum local morphology length, not merely a favorable random arrangement of residual amplitudes.

Paper 60W does not claim that the observed $b = 4$ threshold is universal. It identifies $b = 4$ as the first tested block length that produced a morphology-preserving survivor in the declared pipeline. Establishing a physical correlation length requires a dedicated follow-up across grids, samples, and windows.

6.12 Falsifier 12: Path-Level Derivation Fails

The strongest future test is theoretical. Paper 60V and Paper 60W use a diagnostic phase-lock rule. A later paper must derive the phase-compatibility selector from path-level admissibility dynamics.

The target form is:

$$\mathcal{A}_n = \sum_{\gamma \in \Omega_n^{\mathrm{surv}}} w_\gamma \exp(i\Phi_\gamma). \tag{143}$$

The diagnostic interpretation is weakened if no path-level mechanism can reproduce:

- finite transferred $n = 3$ selection,
- high band-limited morphology,
- low incoherent high-n tail,
- high coherent-tail fraction,
- collapse under destructive nulls,
- survival under morphology-preserving controls.

Thus, Paper 60W strengthens the empirical target for a derivation, but it does not replace the derivation.

6.13 Summary of Falsifiability Criteria

The Paper 60W interpretation is supported only if the following conditions remain true:

1. Destructive nulls produce zero or near-zero strict balanced candidates.

2. High band-limited cosine alone does not predict strict balanced survival.
3. Fourier-phase-randomized nulls collapse.
4. Short block shuffles collapse.
5. Longer block-shuffle survivors preserve real morphology under audit.
6. Circular-shift survivors preserve internal waveform structure.
7. Morphology audit metrics explain survivor status.
8. Corrected code eliminates the preliminary block-shuffle degeneracy.
9. Independent BAO samples reproduce the collapse/survival pattern.
10. Nearby baseline choices preserve the qualitative result.
11. The block-size threshold tracks a stable morphology scale rather than varying erratically across grids, samples, or windows.
12. A future path-level derivation reproduces the diagnostic phase-lock behavior.

The current Paper 60W result satisfies the initial adversarial test. Destructive nulls collapse, and survival occurs only in controls that retain substantial waveform structure.

The falsifiability status is therefore:

> Paper 60W reinforces Paper 60V because the v3F strict balanced signature collapses under destructive residual scrambling and survives only when BAO-band morphology is retained. The result remains bounded: it does not prove the final physical phase-lock law, and it still requires independent-sample, baseline-robustness, block-threshold, and path-level derivation tests.

7 Conclusion

Paper 60W tested the adversarial null question raised by Paper 60V. Paper 60V showed that the high-order BAO residual tail should not be treated as uniformly waste, but should instead be decomposed into phase-compatible and phase-incoherent components:

$$M_{n\geq 7}^{\mathrm{tail}} = M_{n\geq 7}^{\mathrm{coh}} + M_{n\geq 7}^{\mathrm{incoh}}.$$

That result was strong but vulnerable to a natural criticism: perhaps the v3F phase-compatible tail selector was flexible enough to produce strict balanced candidates for arbitrary band-limited residual structure.

Paper 60W directly tested that concern. A corrected all-in-one null suite generated real-data-derived null/control residuals, ran the same v3F scan used in Paper 60V, summarized strict balanced

candidate counts, and audited morphology before interpreting any survivor. The suite evaluated 168 null/control scans using the same BOSS DR12 Fourier-space monopole input, the same k-window,

$$0.015 \leq k \leq 0.145,$$

the same fourth-degree log-polynomial baseline convention, $N_{\rm dense} = 8192$, $n_{\rm max} = 12$, and $N_{\rm paths} = 12000$.

The main result is clear. The destructive null families collapse completely:

$$N_{\rm strict}^{\rm sign} = 0, \tag{144}$$
$$N_{\rm strict}^{\rm value} = 0, \tag{145}$$
$$N_{\rm strict}^{\rm Fourier} = 0, \tag{146}$$
$$N_{\rm strict}^{{\rm block},\, b=2} = 0, \tag{147}$$
$$N_{\rm strict}^{{\rm block},\, b=3} = 0. \tag{148}$$

These nulls disrupt phase organization, value ordering, Fourier phase structure, or short-block local morphology. None of them reproduce the Paper 60V strict balanced signature.

This collapse is important because several destructive null families still achieve high best-row band-limited cosine similarity. Therefore, high cosine alone is not the Paper 60V signal. The relevant signature is the joint condition: band-limited morphology, peak spacing, peak count, transferred $n = 3$ selection, low incoherent high-n tail, and high coherent-tail fraction must hold together. The destructive nulls fail that joint condition.

Survival occurs only in controls that preserve substantial waveform structure. The block-shuffle family shows a local morphology threshold:

$$b = 2 \Rightarrow N_{\rm strict} = 0, \qquad b = 3 \Rightarrow N_{\rm strict} = 0, \qquad b = 4 \Rightarrow N_{\rm strict} = 643$$

in one survivor run. The morphology audit shows that this block-size 4 survivor is not a dangerous false positive. It preserves the real band-limited target closely:

$$\cos(B_{\rm null}, B_{\rm real}) \approx 0.989, \tag{149}$$
$$C_{\rm AC} \approx 0.999978, \tag{150}$$
$$C_{\rm harm} \approx 0.999091, \tag{151}$$
$$S_{\rm spacing} \approx 0.954, \tag{152}$$
$$N_{\rm peak}^{\rm null} = 10. \tag{153}$$

Its survival is therefore best interpreted as morphology-preserving survival, not arbitrary null survival.

The circular-shift controls also require careful interpretation. They produce three survivor runs and 1284 total strict balanced candidates. Their direct band cosine is poor,

$$\cos(B_{\rm null}, B_{\rm real}) \approx -0.268,$$

but they preserve strong internal structure:

$$C_{\mathrm{AC}} \approx 0.994706, \tag{154}$$

$$C_{\mathrm{harm}} \approx 0.889042, \tag{155}$$

$$S_{\mathrm{spacing}} \approx 0.996856, \tag{156}$$

$$N_{\mathrm{peak}}^{\mathrm{null}} = 10. \tag{157}$$

These are phase-shifted morphology controls. Their survival does not show that v3F fits arbitrary noise; it shows that v3F remains sensitive to preserved oscillatory structure even when direct alignment is shifted.

The central conclusion of Paper 60W is therefore:

> The v3F strict balanced signature collapses when phase, ordering, or local morphology are destroyed, and survives only when substantial BAO-band waveform structure is retained. This reinforces Paper 60V: the phase-compatible high-order tail selector is morphology-sensitive, not a generic band-limited noise fitter.

This result strengthens the bounded diagnostic interpretation of Paper 60V. It shows that the phase-compatible tail decomposition is not merely a flexible post-hoc sorting rule that can succeed on arbitrary scrambled residuals. The selector requires retained structure. When destructive nulls erase that structure, the strict balanced class disappears.

At the same time, the conclusion remains bounded. Paper 60W does not prove a final physical phase-lock law. It does not claim to reconstruct the full raw BAO residual. It does not replace standard BAO cosmology. It does not yet establish robustness across independent BAO samples, alternative baseline choices, or different k-windows. The result is a null-control result for the Paper 60V diagnostic mechanism.

The next technical step is now better defined. A future paper should derive the phase-compatible tail selector from path-level admissibility dynamics rather than from the band-limited harmonic projection. The natural target remains a complex path sum:

$$\mathcal{A}_n = \sum_{\gamma \in \Omega_n^{\mathrm{surv}}} w_\gamma \exp(i\Phi_\gamma).$$

Such a derivation should explain why some high-order path families remain aligned with the carrier or switch-sideband scaffold while others become incoherent leakage.

The block-size result also gives the next derivation a concrete clue. The corrected null suite suggests that v3F survival requires a minimum amount of retained local morphology. Blocks of size 2 and 3 collapse, while one block-size 4 control survives only because it remains close to the real band-limited morphology under the audit. This suggests that the eventual path-level theory should not only predict phase compatibility in the abstract; it should also explain why a finite local segment of the waveform is required to preserve carrier/sideband identity.

Paper 60W therefore advances the 60-series by converting an important vulnerability into a controlled result. Paper 60V proposed that the high-order BAO tail is not uniformly waste. Paper 60W

shows that this diagnostic signature collapses under destructive nulls and survives only under morphology-preserving controls. That is the behavior expected of a real morphology-sensitive selector. The high-order tail remains a clue, not a closure; but it is now a much better-tested clue.

References

[1] M. J. Sarnowski, *Paper 60V: Phase-Compatible High-Order Tail Structure in BAO Residual Morphology*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19824977. Latest version DOI: https://doi.org/10.5281/zenodo.19824978.

[2] M. J. Sarnowski, *Paper 60U: Hybrid Transfer-Operator Reconstruction of BAO-Like Residual Structure from Admissibility-Gated Transport*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19742421.

[3] M. J. Sarnowski, *Paper 60T: Forward Generation of BAO-Like Spectra from Admissibility-Gated Transport*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19478148.

[4] M. J. Sarnowski, *Paper 60S: Microscopic Derivation of Harmonic Selection in Admissibility-Gated Transport*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19420957.

[5] M. J. Sarnowski, *Paper 60R: Horizon-Selected Harmonics in BAO-Like Spectra*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19411041.

[6] M. J. Sarnowski, *Paper 60Q: Harmonic Selection Tests for BAO-Like Modes in Transport Spectra*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19411699.

[7] M. J. Sarnowski, *Paper 60P: A Minimal Geometric Corridor Model for the BAO Spacing Spectrum*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19411389.

[8] M. J. Sarnowski, *Paper 60N: Blind Harmonic Spacing Structure in BAO Residuals*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19402234.

[9] M. J. Sarnowski, *Paper 138: Coherence-Dependent Spectral Filtering in Admissibility-Constrained Transport*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.19723149.

[10] M. J. Sarnowski, *Paper T7: How Light Really Moves: Angular Coherence, Bidirectional Phase Propagation, and Double-Slit Gating in the Holosphere Lattice*, Zenodo (2025), Cite all versions DOI: https://doi.org/10.5281/zenodo.15696985.

[11] M. J. Sarnowski, *Paper 101: Derivation of Coherence Constants in the Holosphere Lattice*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.17050872.

[12] M. J. Sarnowski, *Paper 27: Curvature-Regulated Coherence Membranes*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.16408230.

[13] M. J. Sarnowski, *Paper 32 (Math): Real-Valued S^3 Overlap: Mathematical Structure and Spectral Consistency Framework*, Zenodo (2026), Cite all versions DOI: https://doi.org/10.5281/zenodo.15624423.

[14] D. J. Eisenstein et al., *Detection of the Baryon Acoustic Peak in the Large-Scale Correlation Function of SDSS Luminous Red Galaxies*, Astrophys. J. **633** (2005), 560–574.

[15] S. Cole et al., *The 2dF Galaxy Redshift Survey: Power-Spectrum Analysis of the Final Dataset and Cosmological Implications*, Mon. Not. R. Astron. Soc. **362** (2005), 505–534.

[16] H.-J. Seo and D. J. Eisenstein, *Improved Forecasts for the Baryon Acoustic Oscillations and Cosmological Distance Scale*, Astrophys. J. **665** (2007), 14–24.

[17] W. J. Percival et al., *Baryon Acoustic Oscillations in the Sloan Digital Sky Survey Data Release 7 Galaxy Sample*, Mon. Not. R. Astron. Soc. **401** (2010), 2148–2168.

[18] L. Anderson et al., *The Clustering of Galaxies in the SDSS-III Baryon Oscillation Spectroscopic Survey: Baryon Acoustic Oscillations in Data Releases 10 and 11*, Mon. Not. R. Astron. Soc. **441** (2014), 24–62.

[19] S. Alam et al., *The Clustering of Galaxies in the Completed SDSS-III Baryon Oscillation Spectroscopic Survey: Cosmological Analysis of the DR12 Galaxy Sample*, Mon. Not. R. Astron. Soc. **470** (2017), 2617–2652.

[20] F. Beutler et al., *The Clustering of Galaxies in the Completed SDSS-III BOSS Survey: Fourier-Space Clustering Analysis of the Final DR12 Sample*, Mon. Not. R. Astron. Soc. **464** (2017), 3409–3430.

[21] F. Beutler et al., *The Clustering of Galaxies in the Completed SDSS-III BOSS Survey: Anisotropic Clustering in Configuration Space*, Mon. Not. R. Astron. Soc. **466** (2017), 2242–2260.

A Glossary

A.1 Keywords and Definitions

Admissibility A rule that decides whether a path or signal remains allowed to contribute. In the 60-series, admissibility is tied to local and cumulative phase compatibility.

Admissibility-Gated Transport A transport process in which possible paths are filtered by compatibility rules before they can become visible in an observable residual.

BAO Baryon acoustic oscillation. In standard cosmology, BAO refers to a characteristic large-scale clustering feature in the galaxy distribution. In this paper, BAO data provide the residual morphology being tested.

BAO-Like Residual The oscillatory wiggle-like structure that remains after a smooth broadband trend is removed from a galaxy power spectrum.

Band-Limited Residual The smoother oscillatory target extracted from the raw residual. This is the primary comparison target in Papers 60V and 60W.

Block Shuffle A null operation that divides the real residual into short contiguous blocks and randomly reorders those blocks. It preserves local fragments while disrupting global order.

Carrier Band The main low/intermediate harmonic scaffold of the BAO-like residual. In Paper 60V, this is associated with

$$n = 3\text{–}4.$$

Circular Shift A control operation that shifts the residual cyclically along the k-grid. It changes direct alignment but preserves the global waveform shape, autocorrelation, spacing, and residual distribution.

Coherent Tail The part of the high-order tail classified by the diagnostic selector as phase-compatible with the carrier or sideband scaffold.

Coherent Tail Fraction The fraction of the high-order tail classified as coherent:

$$f_{\mathrm{coh}}^{n\geq 7} = \frac{M_{n\geq 7}^{\mathrm{coh}}}{M_{n\geq 7}^{\mathrm{tail}}}.$$

Destructive Null A null residual designed to destroy phase, ordering, spacing, or local morphology. In this paper, sign shuffle, value shuffle, Fourier phase randomization, and short block shuffles are treated as destructive nulls.

Fourier Phase Randomization A null operation that preserves Fourier amplitudes while randomizing Fourier phases. It tests whether the v3F signature depends on organized phase structure rather than frequency amplitudes alone.

High-n Tail The high-order harmonic portion of the model, especially

$$n \geq 7.$$

Paper 60V showed that this tail should be decomposed before being called waste.

Holosphere Theory The broader framework in which observable structure is interpreted through angular coherence, admissibility-gated transport, vacancy scaffolds, resonator structure, and corridor-mediated visibility.

Incoherent Tail The part of the high-order tail classified as phase-incompatible with the carrier or sideband scaffold. This is the leakage-like component.

Morphology Audit A pre-interpretation check that compares a null/control target against the real band-limited target. It measures direct band cosine, shifted cosine, autocorrelation similarity, harmonic-support similarity, peak count, spacing, and dominant harmonic.

Morphology-Preserving Control A null-like transformation that preserves substantial waveform structure. Such controls are not expected to collapse fully because they retain the structure the selector is designed to detect.

Nearest Carrier-or-Sideband Rule The phase-reference rule from Paper 60V that allows a high-order mode to remain compatible if it aligns with either the carrier band or the sideband band.

Null Residual A real-data-derived surrogate residual used to test whether the v3F signature survives after phase, spacing, ordering, or morphology are disrupted.

Phase Compatibility The condition that a high-order mode remains aligned with the carrier or sideband scaffold closely enough to be classified as coherent by the diagnostic selector.

Phase-Compatible Tail Selector The v3F mechanism that separates high-order modes into coherent and incoherent components according to phase compatibility.

Phase-Incoherent Leakage High-order structure that fails the phase-compatibility test and is treated as leakage rather than useful morphology.

Raw Residual The direct residual after subtracting the broadband baseline from the observed power spectrum. It contains localized features and baseline-sensitive variation, so it is secondary in this paper.

Sideband Band The transition or switch band associated with

$$n = 5\text{–}6.$$

In the 60-series, it is interpreted as shoulder-bearing or corridor-switching structure.

Sign Shuffle A destructive null that randomly flips the signs of real residual values while preserving their magnitudes.

Strict Balanced Candidate A v3F scan row satisfying the combined criteria for morphology, spacing, peak count, transferred $n = 3$ selection, low incoherent high-n tail, and high coherent-tail fraction.

Structure-Preserving Survivor A null/control run that produces strict balanced candidates because it preserves substantial real-like waveform morphology. This is not treated as an arbitrary false positive.

Tail Decomposition The split of high-order tail mass into coherent and incoherent components:

$$M_{n\geq 7}^{\text{tail}} = M_{n\geq 7}^{\text{coh}} + M_{n\geq 7}^{\text{incoh}}.$$

Transferred Harmonic The harmonic order that dominates after transfer filtering and phase-tail selection:

$$n_{\text{transfer}} = \arg\max_n |A_n^{\text{phase}}|.$$

Value Shuffle A destructive null that randomly permutes residual values across the k-grid. It preserves the value distribution but destroys the original ordering.

v3F The Paper 60V phase-coupled tail scan framework used to test whether high-order tail mass is coherent support or incoherent leakage.

A.2 Symbols and Meanings

k Fourier-space wavenumber.

$P(k)$ Observed galaxy power spectrum.

$P_{\text{base}}(k)$ Smooth broadband baseline removed from $P(k)$.

$R_{\text{raw}}(k)$ Raw residual after subtracting the broadband baseline.

$R_{\text{real}}(k)$ Real residual used to generate null/control residuals.

$R_{\text{null}}(k)$ Null or control residual generated from the real residual.

$B_{\text{real}}(k)$ Real band-limited residual target.

$B_{\text{null}}(k)$ Null/control band-limited residual target.

$M(k)$ Model reconstructed band-limited residual.

n Harmonic order.

$n_{\max}$ Maximum harmonic order included in the scan.

n_{transfer} Dominant transferred harmonic after v3F filtering.

N_{strict} Number of strict balanced candidates.

$N_{\text{strict}}^{\text{real}}$ Number of strict balanced candidates for the real Paper 60V residual:

$$N_{\text{strict}}^{\text{real}} = 1980.$$

$N_{\text{strict}}^{\text{null}}$ Number of strict balanced candidates for a null/control residual.

N_{peak} Detected peak count in the band-limited target or model.

$\cos_{\text{band}}$ Band-limited cosine similarity between the model and the target.

S_{spacing} Peak-spacing similarity score.

C_{AC} Autocorrelation similarity between real and null/control targets.

C_{harm} Harmonic-support similarity between real and null/control targets.

D_n Geometric engagement factor.

S_n Admissible survivor fraction.

C_n Coherence-retention factor.

T_n Transfer-visibility factor.

A_n^{obs} Observable transferred harmonic amplitude before phase-tail selection.

A_n^{phase} Observable transferred harmonic amplitude after phase-tail selection.

$M_{n\geq 7}^{\text{tail}}$ Total high-order tail mass.

$M_{n\geq 7}^{\text{coh}}$ High-order mass classified as phase-compatible.

$M_{n\geq 7}^{\text{incoh}}$ High-order mass classified as phase-incoherent leakage.

$f_{\text{coh}}^{n\geq 7}$ Coherent fraction of the high-order tail.

b Block size used in block-shuffle nulls.

π A random permutation used in value-shuffle or block-shuffle nulls.

s_i Random sign used in sign-shuffle nulls.

θ_q Randomized Fourier phase for Fourier-phase-randomized nulls.

Ω_n^{surv} Surviving admissible path ensemble at harmonic order n.

$\mathcal{A}_n$ Future path-level complex amplitude:

$$\mathcal{A}_n = \sum_{\gamma \in \Omega_n^{\text{surv}}} w_\gamma \exp(i\Phi_\gamma).$$

Φ_γ Cumulative phase carried by path γ.

w_γ Weight assigned to path γ.

B Plain-Language Summary

B.1 Plain-Language Summary

Paper 60W checks whether the result from Paper 60V is real or whether it could happen just by scrambling the data.

Paper 60V found that the high-order tail in the BAO residual is not all bad. Some of the tail seems to line up with the main pattern. That part may help shape the wiggles. The part that does not line up acts more like leakage.

But there was an important question. What if the model is just flexible enough to find patterns in almost anything? Paper 60W tests that question.

The paper makes several scrambled versions of the real galaxy residual. These are called nulls. Some nulls flip signs. Some shuffle values. Some randomize Fourier phases. Some shuffle small blocks of the signal. Some shift the whole signal around.

Then each null is passed through the same v3F test used in Paper 60V. The key question is whether the nulls can produce the same kind of strong result as the real data.

The answer is mostly no. When the nulls destroy phase, order, or local shape, the strong v3F result disappears. Sign shuffle, value shuffle, Fourier phase randomization, and short block shuffles all produce zero strict balanced candidates.

But some controls survive. These are not random-noise successes. They survive only when they preserve enough of the real waveform. One block shuffle with block size 4 survived because it kept a shape very close to the real band-limited pattern. Some circular shifts survived because they kept the same internal spacing and rhythm, even though the pattern was shifted.

This means Paper 60W does not show that the model works on arbitrary scrambled data. It shows the opposite. The model fails when the important morphology is destroyed. It can survive when the morphology is preserved.

The main lesson is simple: the v3F selector is sensitive to retained BAO-band shape. It is not just fitting random noise.

Paper 60W supports Paper 60V, but it does not finish the theory. The phase rule still needs a deeper derivation from path-level Holosphere transport. But Paper 60W shows that the target for that derivation is worth taking seriously.

B.2 More Intuitive Analogy

Imagine you are testing a song-recognition app.

First, you play the real song. The app recognizes it.

Then you scramble the song in different ways. You flip notes upside down. You shuffle notes randomly. You randomize the timing. When you do that, the app no longer recognizes the song.

That is good. It means the app is not just reacting to random sound.

But then you try two different tests. You move the whole song forward in time, or you shuffle bigger chunks that still contain recognizable pieces of the melody. Sometimes the app recognizes those. That is not a failure. It means the app is detecting the song's structure.

Paper 60W works the same way. The real BAO residual is like the song. The destructive nulls scramble the song so badly that v3F cannot recognize it. The morphology-preserving controls keep enough of the melody that v3F sometimes still recognizes it.

So the test tells us something important: v3F is not hearing random noise. It is responding to preserved structure.

B.3 Thirty-Word Analogy

The BAO pattern is like a song. Randomly scrambled notes are not recognized, but shifted or partly preserved melodies can be. Paper 60W tests that difference.

B.4 Sixth-Grade Summary

This paper checks whether our earlier result was too easy to get.

In Paper 60V, we found that the high-order tail in the BAO pattern was not all noise. Some of it seemed to line up with the main pattern and help shape it. That was exciting, but we needed to test it.

So in Paper 60W, we made scrambled versions of the real data. These scrambled versions are called nulls. If the model still worked on the scrambled data, that would be bad. It would mean the model might be finding patterns where there are none.

We tried several kinds of scrambling. We flipped signs. We shuffled values. We randomized Fourier phases. We shuffled small blocks. These tests damaged the real pattern.

When the pattern was damaged, the strong result disappeared. That supports Paper 60V.

But not every control was fully destructive. A circular shift keeps the same pattern but moves it. A larger block shuffle can keep some local pieces of the pattern. These controls sometimes survived. That is not bad. It means the model can still find the pattern when enough of the pattern remains.

The main point is this: the model does not work on random scrambled data. It works when the BAO-band shape is still preserved.

This makes Paper 60V stronger. It shows the high-order tail result is not just random curve fitting. But there is still more work to do. The next step is to derive the phase rule from the deeper Holosphere path model.

C Script

C.1 Appendix Introduction

This appendix provides the full all-in-one Python script used for the Paper 60W null/control suite. The script generates corrected real-data-derived null inputs, runs the Paper 60V v3F scan on each null/control file, summarizes strict balanced candidate counts, performs the morphology audit, and writes the paper-facing summary artifacts.

The script is included to make the null test transparent and reproducible. It is designed to be run directly on Windows using Python 3.11.9 and the real BOSS DR12 Fourier-space monopole file declared in the reproducibility manifest. No synthetic fallback data are generated. If the real input file or the v3F scan script is missing, the program stops rather than substituting artificial data.

The script evaluates the same null/control families reported in the paper:

- sign-shuffled residuals,
- value-shuffled residuals,
- Fourier-phase-randomized residuals,

- circular-shift controls,
- block-shuffle residuals with block sizes 2, 3, and 4.

The production configuration uses 24 runs per non-block null family and 24 runs per block size, for a total of 168 v3F scans. It uses 12 parallel workers, $N_{\text{paths}} = 12000$, $N_{\text{dense}} = 8192$, $n_{\max} = 12$, and the same k-window and baseline convention used in Paper 60V.

C.2 Reproducibility Statement

The primary reproduction target for Paper 60W is the corrected all-in-one null suite. Running the script produces the following paper-facing artifacts:

- `paper60W_all_command.txt`
- `paper60W_all_config.json`
- `paper60W_all_null_manifest.csv`
- `paper60W_all_null_manifest.json`
- `paper60W_all_v3F_final_summary.csv`
- `paper60W_all_v3F_by_family_summary.csv`
- `paper60W_all_morphology_audit.csv`
- `paper60W_all_morphology_by_family_summary.csv`
- `paper60W_all_survivors.csv`
- `paper60W_all_interpretation.txt`
- `paper60W_all_receipt.json`

The generated receipt records the output root, null count, v3F result count, survivor count, elapsed runtime, and artifact paths. In the production run reported in this paper, the script generated 168 null/control inputs, completed 168 v3F scan results, and identified 4 survivor runs with nonzero strict balanced counts.

Tables in the manuscript are constructed from the generated CSV and text artifacts after review and formatting for clarity. The purpose of including the full script is to document the exact computational procedure used to produce the null/control suite, including null generation, v3F execution, strict-count summarization, and morphology auditing.

C.3 All-in-One Null Suite Script

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
r"""
Paper 60W all-in-one null suite.

Purpose
-------
Generate corrected real-data-derived null inputs, run Paper 60V v3F scans
on all nulls using 12 parallel workers, summarize strict balanced counts,
and audit target morphology before model fitting.

This replaces the separate generator / runner / analyzer / morphology audit
workflow with one Python script.

Null design
-----------
Non-block nulls:
    sign_shuffle                 24 runs
    value_shuffle                24 runs
    fourier_phase_randomized      24 runs
    circular_shift               24 runs

Block-shuffle nulls:
    block_size = 2               24 runs
    block_size = 3               24 runs
    block_size = 4               24 runs

Total v3F scans:
    96 non-block + 72 block = 168 total scans

Correctness fixes compared with the earlier generator
-----------------------------------------------------
1. Fourier phase randomization preserves Fourier amplitudes and does not
   force time-domain standard deviation afterward.
2. Null residuals are not globally re-normalized by default.
3. Block shuffle records permutation diagnostics and identity/degen flags.
4. Clipping diagnostics are recorded if positivity clipping is needed.
5. Morphology audit is run before interpreting strict balanced survival.
6. Outputs include both strict-count summaries and morphology summaries.

Required existing script
------------------------
C:\Users\thile\Downloads\paper60V_v3F_phase_coupled_tail.py

Outputs
-------
Under:
C:\Users\thile\Downloads\Paper60W_AllInOne_NullSuite

Primary outputs:
    paper60W_all_null_manifest.csv
    paper60W_all_v3F_final_summary.csv
    paper60W_all_v3F_by_family_summary.csv
    paper60W_all_morphology_audit.csv
    paper60W_all_morphology_by_family_summary.csv
    paper60W_all_survivors.csv
    paper60W_all_interpretation.txt
    paper60W_all_receipt.json
"""

from __future__ import annotations

import csv
import json
import math
import os
import platform
import shutil
import subprocess
import sys
import time
from concurrent.futures import ProcessPoolExecutor, as_completed
from pathlib import Path
from statistics import mean, median
from typing import Any, Dict, Iterable, List, Optional, Tuple

import numpy as np
import pandas as pd


# ============================================================
# User configuration
```

```
# ============================================================

PYTHON = Path(r"C:\Users\thile\AppData\Local\Programs\Python\Python311\python.exe")
V3F_SCRIPT = Path(r"C:\Users\thile\Downloads\paper60V_v3F_phase_coupled_tail.py")

BAO_DATA = Path(
    r"C:\Users\thile\BOSS_DR12_BAO_Bundles"
    r"\Beutler_etal_DR12COMBINED_BAO_powspec.tar.gz_EXTRACTED"
    r"\public_material_BAO"
    r"\Beutleretal_pk_monopole_DR12_NGC_z1_postrecon_120.dat"
)

ROOT = Path(r"C:\Users\thile\Downloads\Paper60W_AllInOne_NullSuite")
INPUT_DIR = ROOT / "null_inputs"
RUN_ROOT = ROOT / "v3F_runs"

MAX_WORKERS = 12

NONBLOCK_NULLS_PER_TYPE = 24
BLOCK_NULLS_PER_BLOCK_SIZE = 24
BLOCK_SIZES = [2, 3, 4]

SEED = 60060
REAL_STRICT_COUNT = 1980

KMIN = 0.015
KMAX = 0.145
BASELINE_DEGREE = 4
USE_LOGPOLY = True
DENSE_N = 8192
CENTER_WIN = 129
N_MAX = 12
N_PATHS = 12000

CLIP_POSITIVE = True

# If True, this would repeat all five null types for every block size.
# For the requested design, leave False: block size only affects block_shuffle.
RUN_ALL_TYPES_PER_BLOCK_SIZE = False


# ============================================================
# General utilities
# ============================================================

def ensure_exists(path: Path, label: str) -> None:
    if not path.exists():
        raise FileNotFoundError(f"{label} not found: {path}")


def reset_dir(path: Path) -> None:
    if path.exists():
        print(f"Removing old folder:\n{path}")
        shutil.rmtree(path)
    path.mkdir(parents=True, exist_ok=True)


def ensure_dir(path: Path) -> Path:
    path.mkdir(parents=True, exist_ok=True)
    return path


def to_int(value: Any, default: int = 0) -> int:
    try:
        if value is None or value == "":
            return default
        return int(float(str(value)))
    except Exception:
        return default


def to_float(value: Any, default: float = float("nan")) -> float:
    try:
        if value is None or value == "":
            return default
        return float(str(value))
    except Exception:
        return default


def write_csv(path: Path, rows: List[Dict[str, Any]], fieldnames: Optional[List[str]] = None) -> None:
    path.parent.mkdir(parents=True, exist_ok=True)

    if fieldnames is None:
        keys: List[str] = []
```

```
        for row in rows:
            for key in row.keys():
                if key not in keys:
                    keys.append(key)
        fieldnames = keys

    with path.open("w", encoding="utf-8", newline="") as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames, extrasaction="ignore")
        writer.writeheader()
        for row in rows:
            writer.writerow(row)


def read_csv(path: Path) -> List[Dict[str, str]]:
    if not path.exists():
        return []
    with path.open("r", encoding="utf-8-sig", newline="") as f:
        return list(csv.DictReader(f))


def run_command(cmd: List[str], stdout_path: Path) -> int:
    env = os.environ.copy()
    env["OMP_NUM_THREADS"] = "1"
    env["OPENBLAS_NUM_THREADS"] = "1"
    env["MKL_NUM_THREADS"] = "1"
    env["NUMEXPR_NUM_THREADS"] = "1"
    env["PYTHONUNBUFFERED"] = "1"

    stdout_path.parent.mkdir(parents=True, exist_ok=True)

    with stdout_path.open("w", encoding="utf-8", errors="replace") as f:
        f.write("COMMAND:\n")
        f.write(" ".join(cmd))
        f.write("\n\n")
        f.flush()

        proc = subprocess.Popen(
            cmd,
            stdout=subprocess.PIPE,
            stderr=subprocess.STDOUT,
            text=True,
            encoding="utf-8",
            errors="replace",
            bufsize=1,
            env=env,
        )

        assert proc.stdout is not None
        for line in proc.stdout:
            f.write(line)
            f.flush()

        proc.wait()
        f.write(f"\nEXIT_CODE={proc.returncode}\n")
        return int(proc.returncode)


# ============================================================
# Data and target construction
# ============================================================

def robust_load_bao(path: Path, kmin: Optional[float], kmax: Optional[float]) -> pd.DataFrame:
    """
    Explicit two-column loader for whitespace or comma separated files.
    Keeps the first two numeric columns only.
    """
    rows: List[Tuple[float, float]] = []

    with path.open("r", encoding="utf-8", errors="ignore") as f:
        for line in f:
            s = line.strip()
            if not s or s.startswith("#") or s.startswith("%") or s.startswith("//"):
                continue

            parts = s.replace(",", " ").split()
            nums: List[float] = []
            for tok in parts:
                try:
                    nums.append(float(tok))
                except ValueError:
                    pass

            if len(nums) < 2:
                continue
```

```
            k, pk = nums[0], nums[1]

            if not np.isfinite(k) or not np.isfinite(pk):
                continue
            if k <= 0.0 or k > 1.0:
                continue
            if kmin is not None and k < kmin:
                continue
            if kmax is not None and k > kmax:
                continue

            rows.append((k, pk))

    if not rows:
        raise ValueError(f"No usable BAO rows parsed from: {path}")

    return pd.DataFrame(rows, columns=["k", "pk"]).sort_values("k").reset_index(drop=True)


def fit_poly_baseline(k: np.ndarray, pk: np.ndarray, degree: int, log_mode: bool) -> np.ndarray:
    x = (k - np.mean(k)) / max(np.std(k), 1e-12)

    if log_mode:
        if np.any(pk <= 0):
            raise ValueError("log-polynomial baseline requires positive P(k); got non-positive values.")
        y = np.log(pk)
        coeff = np.polyfit(x, y, deg=degree)
        return np.exp(np.polyval(coeff, x))

    coeff = np.polyfit(x, pk, deg=degree)
    return np.polyval(coeff, x)


def normalize(y: np.ndarray) -> np.ndarray:
    y = np.asarray(y, dtype=float)
    sd = float(np.std(y))
    if sd < 1e-12:
        return y - float(np.mean(y))
    return (y - float(np.mean(y))) / sd


def smooth_moving_average(y: np.ndarray, win: int) -> np.ndarray:
    y = np.asarray(y, dtype=float)
    if win <= 1:
        return y.copy()
    if win % 2 == 0:
        win += 1

    pad = win // 2
    yp = np.pad(y, (pad, pad), mode="edge")
    ker = np.ones(win, dtype=float) / float(win)
    return np.convolve(yp, ker, mode="valid")


def center_signal(y: np.ndarray, center_win: int) -> np.ndarray:
    yc = np.asarray(y, dtype=float) - float(np.mean(y))
    if center_win > 1:
        yc = yc - smooth_moving_average(yc, center_win)
    return yc


def dense_interp(x: np.ndarray, y: np.ndarray, n: int) -> Tuple[np.ndarray, np.ndarray]:
    xd = np.linspace(float(np.min(x)), float(np.max(x)), int(n))
    yd = np.interp(xd, x, y)
    return xd, yd


def cosine_similarity(a: np.ndarray, b: np.ndarray) -> float:
    a = np.asarray(a, dtype=float)
    b = np.asarray(b, dtype=float)
    da = float(np.linalg.norm(a))
    db = float(np.linalg.norm(b))
    if da <= 1e-12 or db <= 1e-12:
        return 0.0
    return float(np.dot(a, b) / (da * db))


def rmse(a: np.ndarray, b: np.ndarray) -> float:
    return float(np.sqrt(np.mean((np.asarray(a) - np.asarray(b)) ** 2)))


def harmonic_projection(y: np.ndarray, n_max: int) -> Dict[str, np.ndarray]:
    y = normalize(np.asarray(y, dtype=float))
    N = len(y)
    x = np.linspace(0.0, 1.0, N, endpoint=False)
```

```
    n_arr = np.arange(1, n_max + 1, dtype=int)

    cos_coef = np.zeros(n_max)
    sin_coef = np.zeros(n_max)
    amp = np.zeros(n_max)
    phase = np.zeros(n_max)

    for i, n in enumerate(n_arr):
        c = np.cos(2.0 * np.pi * n * x)
        s = np.sin(2.0 * np.pi * n * x)
        a = 2.0 * float(np.mean(y * c))
        b = 2.0 * float(np.mean(y * s))
        cos_coef[i] = a
        sin_coef[i] = b
        amp[i] = math.sqrt(a * a + b * b)
        phase[i] = math.atan2(-b, a)

    support = amp / max(float(np.max(amp)), 1e-12)

    return {
        "n": n_arr,
        "cos": cos_coef,
        "sin": sin_coef,
        "amp": amp,
        "phase": phase,
        "support": support,
    }


def reconstruct_from_projection(proj: Dict[str, np.ndarray], dense_n: int) -> np.ndarray:
    amps = np.asarray(proj["amp"], dtype=float)
    phases = np.asarray(proj["phase"], dtype=float)
    x = np.linspace(0.0, 1.0, dense_n, endpoint=False)
    y = np.zeros(dense_n, dtype=float)

    for i, A in enumerate(amps):
        n = i + 1
        if abs(A) < 1e-15:
            continue
        y += float(A) * np.cos(2.0 * np.pi * n * x + phases[i])

    return normalize(y)


def make_targets_from_file(path: Path) -> Dict[str, Any]:
    df = robust_load_bao(path, KMIN, KMAX)
    k = df["k"].to_numpy(dtype=float)
    pk = df["pk"].to_numpy(dtype=float)
    baseline = fit_poly_baseline(k, pk, BASELINE_DEGREE, USE_LOGPOLY)
    residual = normalize(pk - baseline)

    k_dense, residual_dense = dense_interp(k, residual, DENSE_N)
    raw_centered = normalize(center_signal(residual_dense, CENTER_WIN))
    proj = harmonic_projection(raw_centered, N_MAX)
    band = reconstruct_from_projection(proj, DENSE_N)

    return {
        "k": k,
        "pk": pk,
        "baseline": baseline,
        "residual": residual,
        "k_dense": k_dense,
        "raw_centered": raw_centered,
        "projection": proj,
        "band": band,
    }


# ============================================================
# Null generation
# ============================================================

def match_mean_only(source: np.ndarray, target: np.ndarray) -> np.ndarray:
    """
    Preserve target's natural variance/distribution but match source mean.
    """
    return np.asarray(target, dtype=float) - float(np.mean(target)) + float(np.mean(source))


def null_sign_shuffle(resid: np.ndarray, rng: np.random.Generator) -> np.ndarray:
    signs = rng.choice([-1.0, 1.0], size=len(resid))
    return match_mean_only(resid, resid * signs)


def null_value_shuffle(resid: np.ndarray, rng: np.random.Generator) -> np.ndarray:
```

```
    out = np.asarray(resid, dtype=float).copy()
    rng.shuffle(out)
    return match_mean_only(resid, out)


def null_fourier_phase_randomized(resid: np.ndarray, rng: np.random.Generator) -> np.ndarray:
    """
    Preserve Fourier amplitudes of the centered residual while randomizing phases.
    No post-hoc standard-deviation normalization is applied.
    """
    x = np.asarray(resid, dtype=float)
    x_mean = float(np.mean(x))
    x0 = x - x_mean

    fft = np.fft.rfft(x0)
    amp = np.abs(fft)

    phases = rng.uniform(0.0, 2.0 * np.pi, size=len(fft))
    phases[0] = 0.0

    if len(x0) % 2 == 0 and len(phases) > 1:
        phases[-1] = 0.0

    randomized = amp * np.exp(1j * phases)
    out0 = np.fft.irfft(randomized, n=len(x0))

    return out0 + x_mean


def block_shuffle_once(
    resid: np.ndarray,
    rng: np.random.Generator,
    block_size: int,
) -> Tuple[np.ndarray, Dict[str, Any]]:
    x = np.asarray(resid, dtype=float)
    n = len(x)

    if block_size < 2:
        out = null_value_shuffle(x, rng)
        return out, {
            "block_size": block_size,
            "n_blocks": n,
            "permutation_is_identity": False,
            "unchanged_block_positions": "",
            "raw_residual_cos_vs_real": cosine_similarity(x, out),
            "raw_max_abs_diff_vs_real": float(np.max(np.abs(out - x))),
        }

    blocks: List[np.ndarray] = []
    for start in range(0, n, block_size):
        blocks.append(x[start:min(start + block_size, n)].copy())

    n_blocks = len(blocks)
    order = np.arange(n_blocks)

    # Avoid identity permutation where possible.
    for _ in range(100):
        perm = order.copy()
        rng.shuffle(perm)
        if not np.array_equal(perm, order):
            break
    else:
        perm = order.copy()

    shuffled = [blocks[int(i)] for i in perm]
    out = np.concatenate(shuffled)
    out = out[:n]

    if len(out) < n:
        out = np.pad(out, (0, n - len(out)), mode="edge")

    out = match_mean_only(x, out)

    unchanged = int(np.sum(perm == order))
    max_abs = float(np.max(np.abs(out - x)))
    raw_cos = cosine_similarity(x, out)

    return out, {
        "block_size": block_size,
        "n_blocks": n_blocks,
        "permutation_is_identity": bool(np.array_equal(perm, order)),
        "unchanged_block_positions": unchanged,
        "raw_residual_cos_vs_real": raw_cos,
        "raw_max_abs_diff_vs_real": max_abs,
    }
```

```
def null_circular_shift(resid: np.ndarray, rng: np.random.Generator, min_frac: float = 0.15) -> Tuple[np.ndarray, Dict[str, Any]]:
    x = np.asarray(resid, dtype=float)
    n = len(x)
    min_shift = max(1, int(min_frac * n))
    max_shift = max(min_shift + 1, n - min_shift)

    shift = int(rng.integers(min_shift, max_shift))
    out = np.roll(x, shift)

    return out, {
        "shift": shift,
        "raw_residual_cos_vs_real": cosine_similarity(x, out),
        "raw_max_abs_diff_vs_real": float(np.max(np.abs(out - x))),
    }


def write_null_dat(path: Path, k: np.ndarray, pk_null: np.ndarray, header: Dict[str, Any]) -> None:
    path.parent.mkdir(parents=True, exist_ok=True)
    with path.open("w", encoding="utf-8") as f:
        f.write("# Paper 60W all-in-one real-data-derived null BAO input\n")
        for key, value in header.items():
            f.write(f"# {key}: {value}\n")
        f.write("# columns: k  P_null(k)\n")
        for ki, pi in zip(k, pk_null):
            f.write(f"{ki:.12e} {pi:.12e}\n")


def generate_all_nulls() -> List[Dict[str, Any]]:
    df = robust_load_bao(BAO_DATA, KMIN, KMAX)
    k = df["k"].to_numpy(dtype=float)
    pk = df["pk"].to_numpy(dtype=float)

    baseline = fit_poly_baseline(k, pk, BASELINE_DEGREE, USE_LOGPOLY)
    resid = pk - baseline

    pk_min_positive = max(1e-12, float(np.min(pk[pk > 0])) if np.any(pk > 0) else 1e-12)

    rng_master = np.random.default_rng(SEED)
    rows: List[Dict[str, Any]] = []

    # Real audit copy.
    write_null_dat(
        INPUT_DIR / "paper60W_real_window_audit.dat",
        k,
        pk,
        {
            "type": "real_audit",
            "source_bao": str(BAO_DATA),
            "kmin": KMIN,
            "kmax": KMAX,
            "baseline_degree": BASELINE_DEGREE,
            "use_logpoly": USE_LOGPOLY,
        },
    )

    # Non-block null types.
    nonblock_types = [
        "sign_shuffle",
        "value_shuffle",
        "fourier_phase_randomized",
        "circular_shift",
    ]

    for null_type in nonblock_types:
        for i in range(NONBLOCK_NULLS_PER_TYPE):
            child_seed = int(rng_master.integers(1, 2_147_483_647))
            rng = np.random.default_rng(child_seed)

            extra: Dict[str, Any] = {}

            if null_type == "sign_shuffle":
                resid_null = null_sign_shuffle(resid, rng)
                extra["raw_residual_cos_vs_real"] = cosine_similarity(resid, resid_null)
                extra["raw_max_abs_diff_vs_real"] = float(np.max(np.abs(resid_null - resid)))

            elif null_type == "value_shuffle":
                resid_null = null_value_shuffle(resid, rng)
                extra["raw_residual_cos_vs_real"] = cosine_similarity(resid, resid_null)
                extra["raw_max_abs_diff_vs_real"] = float(np.max(np.abs(resid_null - resid)))

            elif null_type == "fourier_phase_randomized":
                resid_null = null_fourier_phase_randomized(resid, rng)
                extra["raw_residual_cos_vs_real"] = cosine_similarity(resid, resid_null)
```

```
            extra["raw_max_abs_diff_vs_real"] = float(np.max(np.abs(resid_null - resid)))

        elif null_type == "circular_shift":
            resid_null, extra = null_circular_shift(resid, rng)

        else:
            raise ValueError(null_type)

        pk_before_clip = baseline + resid_null
        n_clipped = int(np.sum(pk_before_clip <= pk_min_positive * 0.05)) if CLIP_POSITIVE else 0
        pk_null = np.clip(pk_before_clip, pk_min_positive * 0.05, None) if CLIP_POSITIVE else pk_before_clip

        family = null_type
        filename = f"paper60W_null_{family}_{i+1:03d}_seed{child_seed}.dat"
        null_path = INPUT_DIR / filename

        header = {
            "type": null_type,
            "family": family,
            "index": i + 1,
            "seed": child_seed,
            "source_bao": str(BAO_DATA),
            "kmin": KMIN,
            "kmax": KMAX,
            "baseline_degree": BASELINE_DEGREE,
            "use_logpoly": USE_LOGPOLY,
            "clip_positive": CLIP_POSITIVE,
            "n_clipped": n_clipped,
            **extra,
        }

        write_null_dat(null_path, k, pk_null, header)

        rows.append(
            {
                "family": family,
                "null_type": null_type,
                "block_size": "",
                "index": i + 1,
                "seed": child_seed,
                "null_file": str(null_path.resolve()),
                "n_rows": len(k),
                "real_residual_mean": float(np.mean(resid)),
                "real_residual_std": float(np.std(resid)),
                "null_residual_mean": float(np.mean(resid_null)),
                "null_residual_std": float(np.std(resid_null)),
                "raw_residual_cos_vs_real": extra.get("raw_residual_cos_vs_real", ""),
                "raw_max_abs_diff_vs_real": extra.get("raw_max_abs_diff_vs_real", ""),
                "shift": extra.get("shift", ""),
                "permutation_is_identity": "",
                "unchanged_block_positions": "",
                "n_clipped": n_clipped,
                "pk_null_min": float(np.min(pk_null)),
                "pk_null_max": float(np.max(pk_null)),
            }
        )

# Block shuffle at block sizes 2, 3, 4.
for block_size in BLOCK_SIZES:
    for i in range(BLOCK_NULLS_PER_BLOCK_SIZE):
        child_seed = int(rng_master.integers(1, 2_147_483_647))
        rng = np.random.default_rng(child_seed)

        resid_null, extra = block_shuffle_once(resid, rng, block_size=block_size)

        pk_before_clip = baseline + resid_null
        n_clipped = int(np.sum(pk_before_clip <= pk_min_positive * 0.05)) if CLIP_POSITIVE else 0
        pk_null = np.clip(pk_before_clip, pk_min_positive * 0.05, None) if CLIP_POSITIVE else pk_before_clip

        family = f"block_shuffle_b{block_size}"
        filename = f"paper60W_null_{family}_{i+1:03d}_seed{child_seed}.dat"
        null_path = INPUT_DIR / filename

        header = {
            "type": "block_shuffle",
            "family": family,
            "index": i + 1,
            "seed": child_seed,
            "block_size": block_size,
            "source_bao": str(BAO_DATA),
            "kmin": KMIN,
            "kmax": KMAX,
            "baseline_degree": BASELINE_DEGREE,
            "use_logpoly": USE_LOGPOLY,
            "clip_positive": CLIP_POSITIVE,
```

```
                "n_clipped": n_clipped,
                **extra,
            }

            write_null_dat(null_path, k, pk_null, header)

            rows.append(
                {
                    "family": family,
                    "null_type": "block_shuffle",
                    "block_size": block_size,
                    "index": i + 1,
                    "seed": child_seed,
                    "null_file": str(null_path.resolve()),
                    "n_rows": len(k),
                    "real_residual_mean": float(np.mean(resid)),
                    "real_residual_std": float(np.std(resid)),
                    "null_residual_mean": float(np.mean(resid_null)),
                    "null_residual_std": float(np.std(resid_null)),
                    "raw_residual_cos_vs_real": extra.get("raw_residual_cos_vs_real", ""),
                    "raw_max_abs_diff_vs_real": extra.get("raw_max_abs_diff_vs_real", ""),
                    "shift": "",
                    "permutation_is_identity": extra.get("permutation_is_identity", ""),
                    "unchanged_block_positions": extra.get("unchanged_block_positions", ""),
                    "n_clipped": n_clipped,
                    "pk_null_min": float(np.min(pk_null)),
                    "pk_null_max": float(np.max(pk_null)),
                }
            )

    manifest_path = ROOT / "paper60W_all_null_manifest.csv"
    write_csv(manifest_path, rows)
    (ROOT / "paper60W_all_null_manifest.json").write_text(json.dumps(rows, indent=2), encoding="utf-8")

    return rows


# ============================================================
# v3F execution
# ============================================================

def summarize_v3f_run(out_dir: Path) -> Dict[str, Any]:
    scan_summary = out_dir / "paper60V_v3F_scan_summary.csv"
    balanced = out_dir / "paper60V_v3F_balanced_candidates.csv"

    balanced_count = ""
    if balanced.exists():
        balanced_count = len(read_csv(balanced))

    metrics = {
        "balanced_count": balanced_count,
        "best_score": "",
        "best_band_cosine": "",
        "best_spacing_score": "",
        "best_peak_count": "",
        "best_transfer_n": "",
        "best_incoherent_tail": "",
        "best_coherent_fraction": "",
        "best_transfer_mode": "",
        "best_phase_reference_mode": "",
    }

    rows = read_csv(scan_summary)
    if rows:
        top = rows[0]
        metrics.update(
            {
                "best_score": top.get("score", ""),
                "best_band_cosine": top.get("band_cosine_similarity", ""),
                "best_spacing_score": top.get("spacing_score", ""),
                "best_peak_count": top.get("model_peak_count_band", ""),
                "best_transfer_n": top.get("peak_n_transfer", ""),
                "best_incoherent_tail": top.get("incoherent_tail_mass_ge_7_transfer", ""),
                "best_coherent_fraction": top.get("coherent_tail_fraction_ge_7", ""),
                "best_transfer_mode": top.get("transfer_mode", ""),
                "best_phase_reference_mode": top.get("phase_reference_mode", ""),
            }
        )

    return metrics


def run_one_v3f(row: Dict[str, Any]) -> Dict[str, Any]:
    run_index = int(row["run_index"])
    family = str(row["family"])
```

```
null_type = str(row["null_type"])
seed = str(row["seed"])
null_file = Path(str(row["null_file"]))
safe_family = family.replace(" ", "_").replace("/", "_")

run_name = f"{run_index:03d}_{safe_family}_seed{seed}"
out_dir = RUN_ROOT / run_name
out_dir.mkdir(parents=True, exist_ok=True)

stdout_path = out_dir / "stdout_combined_live.txt"

cmd = [
    str(PYTHON),
    str(V3F_SCRIPT),
    f"--bao-data={null_file}",
    f"--out-dir={out_dir}",
    f"--kmin={KMIN}",
    f"--kmax={KMAX}",
    f"--dense-n={DENSE_N}",
    f"--baseline-degree={BASELINE_DEGREE}",
    "--use-logpoly=true",
    f"--center-win={CENTER_WIN}",

    "--eps-grid=0.08",
    "--delta-grid=0.26,0.30,0.34",
    "--history-window-grid=3",
    "--alpha-phase-grid=1.0",
    "--alpha-survival-grid=1.0",
    "--alpha-imbalance-grid=0.7",
    "--coherence-mix-grid=0.50",

    "--transfer-history-power-grid=1.75",
    "--band-support-mix-grid=0.75",
    "--transfer-floor-grid=0.02",
    "--transfer-mode-grid=observer_product,phase_tail_sink,phase_tail_blend,phase_tail_only,phase_tail_boost_only",

    "--gate-blend-grid=0.25,0.50,0.75",
    "--gate-eta-omega-grid=0.00,0.10",
    "--gate-epscrit-grid=2.50,3.00",
    "--gate-kappa-grid=4",
    "--gate-lambda-grid=1.00,1.50",
    "--gate-power-grid=1.0",
    "--radial-power-grid=1.0",
    "--angular-power-grid=1.0",

    "--phase-lock-lambda-grid=0.5,1.0,2.0,4.0",
    "--phase-lock-floor-grid=0.05,0.15,0.30,0.50",
    "--phase-reference-mode-grid=carrier_multiple,sideband_multiple,nearest_carrier_or_sideband,parity_pair",
    "--incoherent-tail-lambda-grid=0.5,1.0,2.0",
    "--coherent-tail-boost-grid=0.0,0.15,0.30",

    "--phase-mode=random",
    "--imbalance-mode=phasor",
    "--step-sigma-factor=0.60",
    "--budget-accum-power=1.30",
    f"--n-paths={N_PATHS}",
    f"--n-max={N_MAX}",
    "--seed=60",
    "--top-k=80",
    "--progress-every=0",
]

header = [
    f"RUN_INDEX={run_index}",
    f"FAMILY={family}",
    f"NULL_TYPE={null_type}",
    f"SEED={seed}",
    f"NULL_FILE={null_file}",
    f"OUT_DIR={out_dir}",
    "COMMAND=" + " ".join(cmd),
    "",
]
stdout_path.write_text("\n".join(header), encoding="utf-8")

exit_code = run_command(cmd, stdout_path)
metrics = summarize_v3f_run(out_dir)

return {
    "run_index": run_index,
    "family": family,
    "null_type": null_type,
    "block_size": row.get("block_size", ""),
    "seed": seed,
    "null_file": str(null_file),
    "out_dir": str(out_dir),
```

```
        "exit_code": exit_code,
        **metrics,
        "stdout": str(stdout_path),
    }


# ============================================================
# Morphology audit
# ============================================================

def detect_peak_positions(k: np.ndarray, y: np.ndarray, smooth_win: int = 101, min_rel_height: float = 0.20) -> np.ndarray:
    yy = smooth_moving_average(np.asarray(y, dtype=float), smooth_win)
    if len(yy) < 5:
        return np.array([], dtype=float)

    mx = float(np.max(yy))
    mn = float(np.min(yy))
    thr = mn + min_rel_height * (mx - mn)

    peaks: List[float] = []
    last_i = -999999
    min_sep = max(3, len(yy) // 40)

    for i in range(1, len(yy) - 1):
        if yy[i] > yy[i - 1] and yy[i] >= yy[i + 1] and yy[i] >= thr:
            if i - last_i >= min_sep:
                peaks.append(float(k[i]))
                last_i = i

    return np.asarray(peaks, dtype=float)


def peak_metrics(k: np.ndarray, y: np.ndarray) -> Dict[str, Any]:
    peaks = detect_peak_positions(k, y)
    if len(peaks) >= 2:
        spacing = float(np.median(np.diff(peaks)))
    else:
        spacing = math.nan
    return {"peak_count": int(len(peaks)), "median_peak_spacing": spacing}


def spacing_similarity(real_spacing: float, null_spacing: float) -> float:
    if not np.isfinite(real_spacing) or not np.isfinite(null_spacing) or real_spacing <= 1e-12:
        return 0.0
    return float(math.exp(-2.5 * abs(null_spacing - real_spacing) / real_spacing))


def max_shift_cosine(a: np.ndarray, b: np.ndarray, max_shift_frac: float = 0.25) -> Tuple[float, int]:
    a = normalize(a)
    b = normalize(b)
    n = len(a)
    max_shift = int(max_shift_frac * n)

    best = -2.0
    best_shift = 0

    for shift in range(-max_shift, max_shift + 1):
        c = cosine_similarity(a, np.roll(b, shift))
        if c > best:
            best = c
            best_shift = shift

    return float(best), int(best_shift)


def autocorr(y: np.ndarray, max_lag: int) -> np.ndarray:
    y = normalize(y)
    vals: List[float] = []
    for lag in range(max_lag + 1):
        if lag == 0:
            vals.append(1.0)
        else:
            vals.append(cosine_similarity(y[:-lag], y[lag:]))
    return np.asarray(vals, dtype=float)


def dominant_harmonic(proj: Dict[str, np.ndarray]) -> int:
    n = np.asarray(proj["n"], dtype=int)
    amp = np.asarray(proj["amp"], dtype=float)
    return int(n[int(np.argmax(amp))])


def support_mass(proj: Dict[str, np.ndarray], lo: int, hi: int) -> float:
    n = np.asarray(proj["n"], dtype=int)
    amp = np.abs(np.asarray(proj["amp"], dtype=float))
```

```
    total = max(float(np.sum(amp)), 1e-12)
    return float(np.sum(amp[(n >= lo) & (n <= hi)]) / total)


def support_mass_tail(proj: Dict[str, np.ndarray], lo: int) -> float:
    n = np.asarray(proj["n"], dtype=int)
    amp = np.abs(np.asarray(proj["amp"], dtype=float))
    total = max(float(np.sum(amp)), 1e-12)
    return float(np.sum(amp[n >= lo]) / total)


def classify_morphology(row: Dict[str, Any]) -> str:
    band_cos = to_float(row.get("band_cos_vs_real"))
    shift_cos = to_float(row.get("max_shift_band_cos_vs_real"))
    spacing_score = to_float(row.get("spacing_similarity_vs_real"))
    peak_count_diff = abs(to_int(row.get("peak_count_null")) - to_int(row.get("peak_count_real")))

    if band_cos >= 0.80 or shift_cos >= 0.90:
        return "structure_preserving"

    if band_cos >= 0.60 and spacing_score >= 0.80 and peak_count_diff <= 2:
        return "partly_preserving"

    return "morphology_destroying"


def build_join_map(rows: List[Dict[str, Any]]) -> Dict[str, Dict[str, Any]]:
    out: Dict[str, Dict[str, Any]] = {}
    for row in rows:
        key = str(Path(str(row.get("null_file", ""))).resolve()).lower()
        out[key] = row
    return out


def audit_morphology(null_rows: List[Dict[str, Any]], v3f_results: List[Dict[str, Any]]) -> List[Dict[str, Any]]:
    real_targets = make_targets_from_file(BAO_DATA)
    real_k_dense = np.asarray(real_targets["k_dense"], dtype=float)
    real_raw = np.asarray(real_targets["raw_centered"], dtype=float)
    real_band = np.asarray(real_targets["band"], dtype=float)
    real_proj = real_targets["projection"]

    real_peaks = peak_metrics(real_k_dense, real_band)
    real_ac = autocorr(real_band, max_lag=min(512, DENSE_N // 8))

    real_dom = dominant_harmonic(real_proj)
    real_carrier = support_mass(real_proj, 3, 4)
    real_sideband = support_mass(real_proj, 5, 6)
    real_tail = support_mass_tail(real_proj, 7)
    real_support = np.asarray(real_proj["support"], dtype=float)

    v3f_map = build_join_map(v3f_results)

    audit_rows: List[Dict[str, Any]] = []

    for i, row in enumerate(null_rows, start=1):
        null_file = Path(str(row["null_file"])).resolve()

        try:
            null_targets = make_targets_from_file(null_file)
            null_raw = np.asarray(null_targets["raw_centered"], dtype=float)
            null_band = np.asarray(null_targets["band"], dtype=float)
            null_proj = null_targets["projection"]

            band_cos = cosine_similarity(real_band, null_band)
            raw_cos = cosine_similarity(real_raw, null_raw)
            band_rmse = rmse(real_band, null_band)
            raw_rmse = rmse(real_raw, null_raw)
            max_shift_cos, best_shift = max_shift_cosine(real_band, null_band)

            null_peaks = peak_metrics(real_k_dense, null_band)
            spacing_sim = spacing_similarity(
                float(real_peaks["median_peak_spacing"]),
                float(null_peaks["median_peak_spacing"]),
            )

            null_ac = autocorr(null_band, max_lag=len(real_ac) - 1)
            ac_cos = cosine_similarity(real_ac, null_ac)

            null_support = np.asarray(null_proj["support"], dtype=float)
            support_cos = cosine_similarity(real_support, null_support)

            null_dom = dominant_harmonic(null_proj)
            null_carrier = support_mass(null_proj, 3, 4)
            null_sideband = support_mass(null_proj, 5, 6)
            null_tail = support_mass_tail(null_proj, 7)
```

```
            v3f = v3f_map.get(str(null_file).lower(), {})

            out = {
                **row,
                "raw_cos_vs_real_after_v3f_preprocess": raw_cos,
                "band_cos_vs_real": band_cos,
                "max_shift_band_cos_vs_real": max_shift_cos,
                "best_shift_samples": best_shift,
                "band_rmse_vs_real": band_rmse,
                "raw_rmse_vs_real": raw_rmse,
                "autocorr_cos_vs_real": ac_cos,
                "harmonic_support_cos_vs_real": support_cos,
                "dominant_n_real": real_dom,
                "dominant_n_null": null_dom,
                "carrier_mass_real_3_4": real_carrier,
                "carrier_mass_null_3_4": null_carrier,
                "sideband_mass_real_5_6": real_sideband,
                "sideband_mass_null_5_6": null_sideband,
                "tail_mass_real_ge7": real_tail,
                "tail_mass_null_ge7": null_tail,
                "peak_count_real": int(real_peaks["peak_count"]),
                "peak_count_null": int(null_peaks["peak_count"]),
                "median_peak_spacing_real": real_peaks["median_peak_spacing"],
                "median_peak_spacing_null": null_peaks["median_peak_spacing"],
                "spacing_similarity_vs_real": spacing_sim,
                "v3f_balanced_count": v3f.get("balanced_count", ""),
                "v3f_best_band_cosine": v3f.get("best_band_cosine", ""),
                "v3f_best_spacing_score": v3f.get("best_spacing_score", ""),
                "v3f_best_peak_count": v3f.get("best_peak_count", ""),
                "v3f_best_transfer_n": v3f.get("best_transfer_n", ""),
                "v3f_best_incoherent_tail": v3f.get("best_incoherent_tail", ""),
                "v3f_best_coherent_fraction": v3f.get("best_coherent_fraction", ""),
                "v3f_best_transfer_mode": v3f.get("best_transfer_mode", ""),
                "v3f_best_phase_reference_mode": v3f.get("best_phase_reference_mode", ""),
            }

            out["morphology_class"] = classify_morphology(out)
            audit_rows.append(out)

            if i % 25 == 0:
                print(f"[AUDIT] processed {i}/{len(null_rows)}")

        except Exception as exc:
            audit_rows.append(
                {
                    **row,
                    "error": f"{type(exc).__name__}: {exc}",
                    "morphology_class": "error",
                }
            )

    return audit_rows


# ============================================================
# Summaries
# ============================================================

def summarize_by_family(rows: List[Dict[str, Any]]) -> List[Dict[str, Any]]:
    groups: Dict[str, List[Dict[str, Any]]] = {}
    for row in rows:
        groups.setdefault(str(row.get("family", "")), []).append(row)

    out: List[Dict[str, Any]] = []

    for family in sorted(groups):
        group = groups[family]
        counts = [to_int(r.get("balanced_count")) for r in group]
        band_vals = [to_float(r.get("best_band_cosine")) for r in group if str(r.get("best_band_cosine", "")) != ""]
        spacing_vals = [to_float(r.get("best_spacing_score")) for r in group if str(r.get("best_spacing_score", "")) != ""]
        incoh_vals = [to_float(r.get("best_incoherent_tail")) for r in group if str(r.get("best_incoherent_tail", "")) != ""]
        coh_vals = [to_float(r.get("best_coherent_fraction")) for r in group if str(r.get("best_coherent_fraction", "")) != ""]

        out.append(
            {
                "family": family,
                "runs": len(group),
                "total_balanced_count": sum(counts),
                "max_balanced_count": max(counts) if counts else 0,
                "mean_balanced_count": round(mean(counts), 4) if counts else 0,
                "median_balanced_count": median(counts) if counts else 0,
                "runs_with_any_balanced": sum(1 for c in counts if c > 0),
                "total_vs_real_fraction": round(sum(counts) / REAL_STRICT_COUNT, 6),
                "max_vs_real_fraction": round((max(counts) if counts else 0) / REAL_STRICT_COUNT, 6),
```

```
                "max_best_band_cosine": max(band_vals) if band_vals else "",
                "mean_best_band_cosine": round(mean(band_vals), 6) if band_vals else "",
                "max_best_spacing": max(spacing_vals) if spacing_vals else "",
                "mean_best_spacing": round(mean(spacing_vals), 6) if spacing_vals else "",
                "min_best_incoherent_tail": min(incoh_vals) if incoh_vals else "",
                "mean_best_incoherent_tail": round(mean(incoh_vals), 6) if incoh_vals else "",
                "max_best_coherent_fraction": max(coh_vals) if coh_vals else "",
                "mean_best_coherent_fraction": round(mean(coh_vals), 6) if coh_vals else "",
            }
        )

    return out


def summarize_morphology_by_family(rows: List[Dict[str, Any]]) -> List[Dict[str, Any]]:
    groups: Dict[str, List[Dict[str, Any]]] = {}
    for row in rows:
        groups.setdefault(str(row.get("family", "")), []).append(row)

    out: List[Dict[str, Any]] = []

    for family in sorted(groups):
        group = [r for r in groups[family] if r.get("morphology_class") != "error"]

        if not group:
            continue

        band = [to_float(r.get("band_cos_vs_real")) for r in group]
        shift = [to_float(r.get("max_shift_band_cos_vs_real")) for r in group]
        ac = [to_float(r.get("autocorr_cos_vs_real")) for r in group]
        support = [to_float(r.get("harmonic_support_cos_vs_real")) for r in group]
        spacing = [to_float(r.get("spacing_similarity_vs_real")) for r in group]
        peak_count = [to_float(r.get("peak_count_null")) for r in group]
        tail = [to_float(r.get("tail_mass_null_ge7")) for r in group]
        balanced = [to_int(r.get("v3f_balanced_count")) for r in group]

        out.append(
            {
                "family": family,
                "runs": len(group),
                "morphology_destroying": sum(1 for r in group if r.get("morphology_class") == "morphology_destroying"),
                "partly_preserving": sum(1 for r in group if r.get("morphology_class") == "partly_preserving"),
                "structure_preserving": sum(1 for r in group if r.get("morphology_class") == "structure_preserving"),
                "v3f_runs_with_balanced": sum(1 for c in balanced if c > 0),
                "v3f_total_balanced": sum(balanced),
                "mean_band_cos_vs_real": round(mean(band), 6),
                "max_band_cos_vs_real": max(band),
                "mean_max_shift_cos_vs_real": round(mean(shift), 6),
                "max_max_shift_cos_vs_real": max(shift),
                "mean_autocorr_cos_vs_real": round(mean(ac), 6),
                "mean_harmonic_support_cos_vs_real": round(mean(support), 6),
                "mean_spacing_similarity_vs_real": round(mean(spacing), 6),
                "mean_peak_count_null": round(mean(peak_count), 6),
                "mean_tail_mass_null_ge7": round(mean(tail), 6),
            }
        )

    return out


def write_interpretation(
    v3f_by_family: List[Dict[str, Any]],
    morph_by_family: List[Dict[str, Any]],
    survivors: List[Dict[str, Any]],
) -> Path:
    path = ROOT / "paper60W_all_interpretation.txt"

    lines: List[str] = []
    lines.append("Paper 60W All-In-One Null Suite Interpretation")
    lines.append("==============================================")
    lines.append("")
    lines.append(f"Total v3F scans: {len(read_csv(ROOT / 'paper60W_all_v3F_final_summary.csv'))}")
    lines.append(f"Real Paper 60V strict balanced count: {REAL_STRICT_COUNT}")
    lines.append("")
    lines.append("v3F by-family summary:")
    lines.append(pd.DataFrame(v3f_by_family).to_string(index=False))
    lines.append("")
    lines.append("Morphology by-family summary:")
    lines.append(pd.DataFrame(morph_by_family).to_string(index=False))
    lines.append("")
    lines.append("Top survivors:")
    if survivors:
        display_cols = [
            "run_index",
            "family",
```

```
            "null_type",
            "block_size",
            "seed",
            "v3f_balanced_count",
            "band_cos_vs_real",
            "max_shift_band_cos_vs_real",
            "autocorr_cos_vs_real",
            "harmonic_support_cos_vs_real",
            "spacing_similarity_vs_real",
            "peak_count_null",
            "dominant_n_null",
            "morphology_class",
        ]
        lines.append(pd.DataFrame(survivors)[display_cols].head(30).to_string(index=False))
    else:
        lines.append("No v3F survivors with balanced_count > 0.")
    lines.append("")
    lines.append("Interpretation guide:")
    lines.append("- Destructive null families should produce zero or near-zero strict balanced candidates.")
    lines.append("- Structure-preserving controls may survive, but only if morphology metrics show retained real-like structure.")
    lines.append("- Block-size results test whether local morphology length controls survival.")
    lines.append("- A dangerous null is one with low morphology similarity and high balanced_count.")
    lines.append("- A benign survivor is one with high morphology similarity and high balanced_count.")
    lines.append("")
    lines.append("Paper-safe framing:")
    lines.append("The v3F strict signature should be interpreted against the morphology audit, not balanced_count alone.")
    lines.append("If phase/spacing-destroying nulls collapse while morphology-preserving controls survive, the selector is responding to retained BAO-band waveform struct

    path.write_text("\n".join(lines), encoding="utf-8")
    return path


# ============================================================
# Main
# ============================================================

def main() -> int:
    t0 = time.time()

    print("")
    print("=== Paper 60W all-in-one null suite ===")
    print(f"Python: {PYTHON}")
    print(f"V3F script: {V3F_SCRIPT}")
    print(f"BAO data: {BAO_DATA}")
    print(f"Root: {ROOT}")
    print(f"Max workers: {MAX_WORKERS}")
    print(f"Non-block nulls per type: {NONBLOCK_NULLS_PER_TYPE}")
    print(f"Block nulls per block size: {BLOCK_NULLS_PER_BLOCK_SIZE}")
    print(f"Block sizes: {BLOCK_SIZES}")
    print("")

    ensure_exists(PYTHON, "Python")
    ensure_exists(V3F_SCRIPT, "Paper 60V v3F script")
    ensure_exists(BAO_DATA, "BAO data")

    reset_dir(ROOT)
    ensure_dir(INPUT_DIR)
    ensure_dir(RUN_ROOT)

    command_path = ROOT / "paper60W_all_command.txt"
    command_path.write_text(" ".join([sys.executable] + sys.argv), encoding="utf-8")

    config = {
        "paper": "Paper 60W",
        "purpose": "All-in-one corrected null generation, v3F scan, summary, and morphology audit.",
        "python": str(PYTHON),
        "v3f_script": str(V3F_SCRIPT),
        "bao_data": str(BAO_DATA),
        "root": str(ROOT),
        "max_workers": MAX_WORKERS,
        "nonblock_nulls_per_type": NONBLOCK_NULLS_PER_TYPE,
        "block_nulls_per_block_size": BLOCK_NULLS_PER_BLOCK_SIZE,
        "block_sizes": BLOCK_SIZES,
        "total_expected_scans": 4 * NONBLOCK_NULLS_PER_TYPE + len(BLOCK_SIZES) * BLOCK_NULLS_PER_BLOCK_SIZE,
        "kmin": KMIN,
        "kmax": KMAX,
        "baseline_degree": BASELINE_DEGREE,
        "use_logpoly": USE_LOGPOLY,
        "dense_n": DENSE_N,
        "center_win": CENTER_WIN,
        "n_max": N_MAX,
        "n_paths": N_PATHS,
        "clip_positive": CLIP_POSITIVE,
        "seed": SEED,
        "platform": platform.platform(),
```

```
    "python_version": sys.version,
}
(ROOT / "paper60W_all_config.json").write_text(json.dumps(config, indent=2), encoding="utf-8")

print("Generating corrected null files...")
null_rows = generate_all_nulls()

for i, row in enumerate(null_rows, start=1):
    row["run_index"] = i

manifest_path = ROOT / "paper60W_all_null_manifest.csv"
write_csv(manifest_path, null_rows)
print(f"[WROTE] {manifest_path}")
print(f"Generated null files: {len(null_rows)}")

results: List[Dict[str, Any]] = []

print("")
print("Launching v3F scans...")
with ProcessPoolExecutor(max_workers=MAX_WORKERS) as executor:
    future_to_row = {executor.submit(run_one_v3f, row): row for row in null_rows}

    completed = 0
    for future in as_completed(future_to_row):
        completed += 1
        row = future_to_row[future]

        try:
            result = future.result()
        except Exception as exc:
            result = {
                "run_index": row.get("run_index", ""),
                "family": row.get("family", ""),
                "null_type": row.get("null_type", ""),
                "block_size": row.get("block_size", ""),
                "seed": row.get("seed", ""),
                "null_file": row.get("null_file", ""),
                "out_dir": "",
                "exit_code": f"EXCEPTION: {type(exc).__name__}: {exc}",
                "balanced_count": "",
                "best_score": "",
                "best_band_cosine": "",
                "best_spacing_score": "",
                "best_peak_count": "",
                "best_transfer_n": "",
                "best_incoherent_tail": "",
                "best_coherent_fraction": "",
                "best_transfer_mode": "",
                "best_phase_reference_mode": "",
                "stdout": "",
            }

        results.append(result)
        results_sorted = sorted(results, key=lambda r: to_int(r.get("run_index")))
        final_summary_path = ROOT / "paper60W_all_v3F_final_summary.csv"
        write_csv(final_summary_path, results_sorted)

        print(
            f"[DONE {completed:03d}/{len(null_rows):03d}] "
            f"family={result.get('family')} "
            f"exit={result.get('exit_code')} "
            f"balanced={result.get('balanced_count')} "
            f"cos={result.get('best_band_cosine')} "
            f"nT={result.get('best_transfer_n')}"
        )

results_sorted = sorted(results, key=lambda r: to_int(r.get("run_index")))
final_summary_path = ROOT / "paper60W_all_v3F_final_summary.csv"
write_csv(final_summary_path, results_sorted)

v3f_by_family = summarize_by_family(results_sorted)
v3f_by_family_path = ROOT / "paper60W_all_v3F_by_family_summary.csv"
write_csv(v3f_by_family_path, v3f_by_family)

print("")
print("Running morphology audit...")
audit_rows = audit_morphology(null_rows, results_sorted)
audit_path = ROOT / "paper60W_all_morphology_audit.csv"
write_csv(audit_path, audit_rows)

morph_by_family = summarize_morphology_by_family(audit_rows)
morph_by_family_path = ROOT / "paper60W_all_morphology_by_family_summary.csv"
write_csv(morph_by_family_path, morph_by_family)

survivors = [
```

```
        r for r in audit_rows
        if to_int(r.get("v3f_balanced_count")) > 0
    ]
    survivors = sorted(
        survivors,
        key=lambda r: (to_int(r.get("v3f_balanced_count")), to_float(r.get("band_cos_vs_real"))),
        reverse=True,
    )
    survivors_path = ROOT / "paper60W_all_survivors.csv"
    write_csv(survivors_path, survivors)

    interpretation_path = write_interpretation(v3f_by_family, morph_by_family, survivors)

    receipt = {
        "paper": "Paper 60W",
        "purpose": "All-in-one corrected null generation, v3F scan, summary, and morphology audit.",
        "root": str(ROOT),
        "null_count": len(null_rows),
        "v3f_result_count": len(results_sorted),
        "survivor_count": len(survivors),
        "elapsed_sec": float(time.time() - t0),
        "artifacts": [
            str(command_path.resolve()),
            str((ROOT / "paper60W_all_config.json").resolve()),
            str(manifest_path.resolve()),
            str(final_summary_path.resolve()),
            str(v3f_by_family_path.resolve()),
            str(audit_path.resolve()),
            str(morph_by_family_path.resolve()),
            str(survivors_path.resolve()),
            str(interpretation_path.resolve()),
        ],
    }
    receipt_path = ROOT / "paper60W_all_receipt.json"
    receipt_path.write_text(json.dumps(receipt, indent=2), encoding="utf-8")

    print("")
    print("=== Paper 60W all-in-one null suite complete ===")
    print("")
    print("v3F by-family summary:")
    print(pd.DataFrame(v3f_by_family).to_string(index=False))
    print("")
    print("Morphology by-family summary:")
    print(pd.DataFrame(morph_by_family).to_string(index=False))
    print("")
    print("Top survivors:")
    if survivors:
        display_cols = [
            "run_index",
            "family",
            "null_type",
            "block_size",
            "seed",
            "v3f_balanced_count",
            "band_cos_vs_real",
            "max_shift_band_cos_vs_real",
            "autocorr_cos_vs_real",
            "harmonic_support_cos_vs_real",
            "spacing_similarity_vs_real",
            "peak_count_null",
            "dominant_n_null",
            "morphology_class",
        ]
        print(pd.DataFrame(survivors)[display_cols].head(30).to_string(index=False))
    else:
        print("No v3F survivors.")
    print("")
    print("[ARTIFACTS]")
    for p in receipt["artifacts"]:
        print(f"[WROTE] {p}")
    print(f"[WROTE] {receipt_path.resolve()}")
    print("")
    print(f"[DONE] elapsed_sec={time.time() - t0:.2f}")

    return 0


if __name__ == "__main__":
    raise SystemExit(main())


C:\Users\thile\AppData\Local\Programs\Python\Python311\python.exe C:\Users\thile\Downloads\paper60W_all_in_one_null_suite.py
```