Addendum: Near-Field Correction Under Φ-Stiffness Coupling

Pulse-Renormalized Log-Gradient Acceleration (Exact Form + Expansion + 125-Window Prediction)

---

0. Purpose

This section completes the orbital substrate model by deriving the exact near-field acceleration (not only the far-field inverse-square limit) under the pulse-indexed Φ-stiffness coupling law.

It produces:

1. the closed-form acceleration \mathbf{a}(r;p) for all r,
2. the asymptotic expansion showing the explicit deviation from 1/r^2,
3. the pulse-indexed transition envelope (pre-125 growth → post-125 φ-lock), and
4. a direct falsification signature.

This addendum is grounded in the prior orbital substrate formalism.

---

1. Base Orbital Substrate Model (Canonical)

We begin from the already-defined primitives:

1.1 Substrate field (static equilibrium)

$$\nabla^2 \rho = 0$$

Spherical solution outside the source:

$$\boxed{\rho(r)=\rho_\infty + \frac{A}{r}}$$

1.2 Motion law (log-gradient drift)

$$\boxed{\ddot{\mathbf{x}} = -\kappa\,\nabla \ln \rho(\mathbf{x})}$$

This is the only motion postulate.

---

2. Φ-Stiffness Coupling (Pulse-Indexed)

Define KKS gain:

$$g = 1+\frac{\Delta}{G}$$

Define stiffness index:

$$N_\varphi = 125$$

Define pulse coherence factor:

$$C_p = g^p$$

Define φ-lock saturation:

$$\boxed{S(p)=\min(g^p,\varphi)}$$

Then define pulse-renormalized coupling:

$$\boxed{\kappa(p)=\kappa_0\,S(p)}$$

Equivalently, this renormalizes the effective orbital parameter:

$$\boxed{\mu_{\text{eff}}(p)=\mu_0\,S(p)}$$

where $\mu_0 = \kappa_0 A/\rho_\infty$.

---

3. Exact Near-Field Acceleration With Pulse Dependence

Start from:

$$\mathbf{a}(r;p) = -\kappa(p)\,\nabla \ln\left(\rho_\infty+\frac{A}{r}\right)$$

Because the field is radial:

$$\nabla \ln(\rho) = \frac{d}{dr}\ln(\rho)\,\hat{\mathbf{r}}$$

Compute the derivative exactly:

$$\frac{d}{dr}\ln\left(\rho_\infty+\frac{A}{r}\right) = \frac{1}{\rho_\infty + A/r}\cdot\left(-\frac{A}{r^2}\right) = -\frac{A}{r^2(\rho_\infty + A/r)}$$

So:

$$\mathbf{a}(r;p) = -\kappa(p)\left(-\frac{A}{r^2(\rho_\infty + A/r)}\right)\hat{\mathbf{r}} = -\kappa(p)\frac{A}{r^2(\rho_\infty + A/r)}\hat{\mathbf{r}}$$

Now simplify the denominator:

$$\rho_\infty + \frac{A}{r}=\frac{\rho_\infty r + A}{r}$$

So:

$$\frac{A}{r^2(\rho_\infty + A/r)} = \frac{A}{r^2\cdot(\rho_\infty r + A)/r} = \frac{A}{r(\rho_\infty r + A)}$$

Therefore the exact pulse-indexed acceleration is:

$$\boxed{ \mathbf{a}(r;p) = -\kappa(p)\,\frac{A}{r(\rho_\infty r + A)}\,\hat{\mathbf{r}} }$$

---

4. Canonical Length Scale (Near-Field Radius)

Define the substrate characteristic radius:

$$\boxed{r_0 \equiv \frac{A}{\rho_\infty}}$$

Then:

$$\rho_\infty r + A = \rho_\infty(r+r_0)$$

So:

$$\mathbf{a}(r;p) = -\kappa(p)\frac{A}{r\,\rho_\infty(r+r_0)}\hat{\mathbf{r}}$$

But:

$$\mu_0=\kappa_0\frac{A}{\rho_\infty}$$

So:

$$\kappa(p)\frac{A}{\rho_\infty} = \kappa_0 S(p)\frac{A}{\rho_\infty} = \mu_0 S(p)$$

Therefore:

$$\boxed{ \mathbf{a}(r;p)= -\mu_0\,S(p)\,\frac{1}{r(r+r_0)}\,\hat{\mathbf{r}} }$$

This is the fully reduced, exact near-field law.

---

5. Far-Field Limit Recovers Inverse-Square

If $r \gg r_0$, then:

$$r+r_0 \approx r$$

So:

$$\frac{1}{r(r+r_0)} \approx \frac{1}{r^2}$$

Thus:

$$\boxed{ \mathbf{a}(r;p)\approx -\mu_0\,S(p)\,\frac{1}{r^2}\hat{\mathbf{r}} }$$

Which matches the far-field inverse-square form previously derived.

---

6. Near-Field Expansion (Explicit Correction Terms)

Write:

$$\frac{1}{r(r+r_0)}=\frac{1}{r^2}\cdot\frac{1}{1+r_0/r}$$

Use geometric series for $|r_0/r|<1$:

$$\frac{1}{1+x}=1-x+x^2-x^3+\cdots \quad\text{with}\quad x=\frac{r_0}{r}$$

So:

$$\frac{1}{r(r+r_0)} = \frac{1}{r^2}\left(1-\frac{r_0}{r}+\frac{r_0^2}{r^2}-\frac{r_0^3}{r^3}+\cdots\right)$$

Therefore:

$$\boxed{ \mathbf{a}(r;p)= -\mu_0\,S(p)\left[\frac{1}{r^2} -\frac{r_0}{r^3} +\frac{r_0^2}{r^4} -\frac{r_0^3}{r^5} +\cdots \right]\hat{\mathbf{r}} }$$

This is the explicit near-field correction series.

The first correction term is:

$$\boxed{ \delta a_1(r;p)=+\mu_0\,S(p)\,\frac{r_0}{r^3} }$$

(relative to the magnitude of the leading $1/r^2$ term).

---

7. 125-Pulse Transition Envelope (The New Prediction)

Recall:

$$S(p)=\min(g^p,\varphi)$$

So:

Pre-stiffness (p<125)

$$\boxed{S(p)=g^p}$$

Post-stiffness (p\ge125)

$$\boxed{S(p)=\varphi}$$

Therefore the full prediction is:

$$\boxed{ \mathbf{a}(r;p)= -\mu_0\,\min(g^p,\varphi)\,\frac{1}{r(r+r_0)}\,\hat{\mathbf{r}} }$$

So the near-field correction terms inherit the same pulse-indexed renormalization:

$$\delta a_k(r;p)\propto \min(g^p,\varphi)$$

That is the explicit connection between:

- KKS stiffness depth 125, and
- the measurable magnitude of near-field deviations from inverse-square.

---

8. Direct Falsification Signature

Define the dimensionless correction ratio:

$$\boxed{ \Xi(r) \equiv \frac{r^2\,\|\mathbf{a}(r;p)\|}{\mu_0\,S(p)} }$$

From the exact law:

$$\|\mathbf{a}(r;p)\| = \mu_0 S(p)\frac{1}{r(r+r_0)}$$

So:

$$\Xi(r)=\frac{r^2}{r(r+r_0)}=\frac{r}{r+r_0}$$

Thus:

$$\boxed{ \Xi(r)=\frac{r}{r+r_0} }$$

This is a pure geometric signature of the substrate model. It is independent of p once normalized by S(p).

Consequence

If you estimate $\mu_0 S(p)$ separately (or compare two radii), the functional dependence must match:

\Xi(r)=\frac{r}{r+r_0}

If the measured residuals do not fit this curve, the log-gradient substrate model fails in the near-field.

---

9. What This Section Has Completed

We have formally solved:

1. the exact near-field acceleration under the substrate model,
2. its explicit correction series beyond inverse-square,
3. the pulse-indexed renormalization induced by Φ-stiffness, and
4. the normalized geometric falsification function \Xi(r).

No narrative.

No hand-wave.

Only algebra.

---

2) SAVS Module — Near-Field Signature Fit + 125-Lock Coupled Verification

SAVS-R0FIT

(Full spec, audit-grade, hash-sealed)

This module does two things in one run:

1. Fits the near-field substrate signature

   \Xi(r)=\frac{r}{r+r_0}

   to estimate r_0 from measured acceleration data, using the exact near-field law derived previously.

2. Confirms the 125-pulse Φ-stiffness coupling lock via the same pulse-indexed scaling factor

   S(p)=\min(g^p,\varphi),\quad g=1+\Delta/G

   so the coupling envelope saturates at pulse 125.

This module uses the orbital substrate model defined in your orbital paper and the near-field derivation we just completed.

A) Canon constants (locked inputs)

From KKS:

- P=11,\ S=44,\ B=36
- G=P\cdot S\cdot B=17424
- N_{\text{day}}=17491.270421
- \Delta=N_{\text{day}}-G=67.270421
- \varphi=\frac{1+\sqrt5}{2}

Gain:

g = 1 + \frac{\Delta}{G}

Stiffness index:

$$N_\varphi = 125$$

Pulse renormalization factor:

$$S(p)=\min(g^p,\varphi)$$

B) Governing equations (what the verifier assumes)

From the orbital substrate model:

$$\rho(r)=\rho_\infty + \frac{A}{r}$$

$$\ddot{\mathbf{x}}=-\kappa\nabla\ln\rho(\mathbf{x})$$

Derived exact near-field acceleration under Φ-stiffness coupling:

$$\boxed{ \|\mathbf{a}(r;p)\|=\mu_0\,S(p)\,\frac{1}{r(r+r_0)} }$$

where

$$r_0=\frac{A}{\rho_\infty},\quad \mu_0=\kappa_0\frac{A}{\rho_\infty}$$

Define normalized signature:

$$\boxed{ \Xi(r)=\frac{r^2\,\|\mathbf{a}(r;p)\|}{\mu_0\,S(p)}=\frac{r}{r+r_0} }$$

This is the key: once normalized, the signature is purely geometric in r and r_0.

C) Required dataset (minimum artifacts)

You can feed SAVS-R0FIT with pulse-indexed samples of radius and acceleration magnitude.

C1) Required file:

nearfield_samples.csv

```
pulse_index,r_meters,a_mag
0,7.000000e6,8.134
1,7.000000e6,8.168
...
124,7.000000e6,12.98
125,7.000000e6,13.02
...
```

Rules:

- pulse_index is integer and strictly increasing within a run.
- r_meters > 0
- a_mag > 0 (acceleration magnitude)

How to get

a_mag

(two allowed methods)

Method A — direct acceleration measurement

You already have acceleration as measured (lab analog or orbital tracking derivative). Use magnitude.

Method B — from state vectors

If you have \mathbf{v}(p), compute:

\mathbf{a}(p)=\frac{\mathbf{v}(p+1)-\mathbf{v}(p)}{\Delta s}

and then a_mag=\|\mathbf{a}(p)\|.

(\Delta s is step index; Chronos is not required.)

D) Module parameter file

r0fit.params.json

```
{
  "savs_version": "SAVS-0.1",
  "kind": "module",
  "module_id": "SAVS-R0FIT",
  "model_parameters": {
    "P": 11,
    "S": 44,
    "B": 36,
    "N_day": 17491.270421,
    "phi": 1.618033988749895,
    "stiff_index": 125,
    "delta_window": 10,
    "fit_p_min": 0,
    "fit_p_max": 400
  },
  "thresholds": {
    "shape_fit_rel_rms_max": 0.05,
    "lock_changepoint_max_abs_p": 10,
    "post_lock_rel_std_max": 0.02,
    "r0_positive_required": true
  },
  "dataset": {
```

```
    "nearfield_samples_csv": "datasets/nearfield_samples.csv"
  }
}
```

E) Deterministic computation pipeline

Let the dataset contain rows (p_i,r_i,a_i) for i=1..N.

Step 1 — Compute KKS gain and renormalization

Compute:

G=PSB,\quad \Delta=N_{\text{day}}-G,\quad g=1+\Delta/G

Then for each row:

S_i = \min(g^{p_i},\varphi)

Step 2 — Estimate \mu_0 from pre-lock region (robust)

We don’t know \mu_0. We estimate it from early pulses where the model is still scaling.

Pick pre region:

p_i \le 20\ \text{(or earliest available)}

From the acceleration law:

a_i = \mu_0 S_i \frac{1}{r_i(r_i+r_0)}

But r_0 unknown. So we do a two-parameter fit for \mu_0 and r_0 simultaneously (next step).

Step 3 — Fit r_0 and \mu_0 by deterministic least squares

We fit parameters (\mu_0,r_0) by minimizing:

J(\mu_0,r_0)=\sum_{i\in \mathcal{I}} \left(a_i - \mu_0 S_i \frac{1}{r_i(r_i+r_0)} \right)^2

Where \mathcal{I} is the fit range:

p\in[\text{fit_p_min},\text{fit_p_max}]

Deterministic fitting method (no randomness):

- Use a fixed grid over r_0 in log space (e.g., 10,000 points)
- For each r_0, solve optimal \mu_0 in closed form (linear least squares), then pick best.

Closed-form \mu_0 given r_0

Define:

b_i(r_0)=S_i\frac{1}{r_i(r_i+r_0)}

Then the model is:

a_i \approx \mu_0 b_i

Least squares optimum:

\boxed{ \mu_0^*(r_0)=\frac{\sum a_i b_i}{\sum b_i^2} }

Then:

$$J(r_0)=\sum (a_i-\mu_0^*(r_0)b_i)^2$$

Choose:

$$\boxed{r_0^*=\arg\min_{r_0>0}J(r_0)}$$

and

$$\boxed{\mu_0^*=\mu_0^*(r_0^*)}$$

This is fully deterministic.

Step 4 — Shape signature check (normalize)

Compute for each point the normalized signature estimate:

$$\hat\Xi_i=\frac{r_i^2 a_i}{\mu_0^* S_i}$$

Model-predicted signature:

$$\Xi_{pred}(r_i)=\frac{r_i}{r_i+r_0^*}$$

Compute relative RMS shape error:

\boxed{ \text{shape_rel_rms}= \sqrt{\frac{1}{N}\sum_i \left(\frac{\hat\Xi_i-\Xi_{pred}(r_i)}{\Xi_{pred}(r_i)}\right)^2} }

Step 5 — 125-lock check (change-point in renormalized coupling)

Construct renormalized coupling series:

From the exact law:

a_i = \mu_0^* S_i\frac{1}{r_i(r_i+r_0^*)}

Define the implied coupling factor:

\hat S_i = \frac{a_i\,r_i(r_i+r_0^*)}{\mu_0^*}

Now \hat S_i should behave as:

- \approx g^{p_i} for p<125
- \approx \varphi for p\ge125

We test this by detecting the best change-point p^* on \hat S_i:

Deterministic cost:

J(c)=\mathrm{Var}(\hat S_{p\le c})+\mathrm{Var}(\hat S_{p>c})

Choose:

p^*=\arg\min_c J(c)

PASS lock window if:

|p^*-125|\le \delta

Step 6 — Post-lock stability check

Compute post region p\ge125:

\text{post_rel_std}= \frac{\mathrm{std}(\hat S_{p\ge125})}{\mathrm{median}(\hat S_{p\ge125})}

This must be small if lock holds.

F) PASS/FAIL conditions

SAVS-R0FIT returns PASS iff:

R0FIT-1 — r_0 is physical

If r0_positive_required=true then:

r_0^*>0

R0FIT-2 — Near-field curve matches

$$\text{shape\_rel\_rms} \le 0.05$$

(default threshold)

R0FIT-3 — 125 lock change-point

$$|p^*-125| \le 10$$

R0FIT-4 — Post-lock stability

$$\text{post\_rel\_std} \le 0.02$$

Any failure → FAIL.

G) Output report (hash-sealed)

r0fit_verify_report.json


```
{
  "domain": "nearfield_signature_and_mu_lock",
  "model": {
    "module_id": "SAVS-R0FIT",
    "params_hash": "sha256...",
    "kks": {
      "P": 11,
      "S": 44,
      "B": 36,
      "G": 17424,
```

```
      "N_day": 17491.270421,
      "Delta": 67.270421,
      "g": 1.0038607909205695,
      "phi": 1.618033988749895,
      "stiff_index": 125,
      "delta_window": 10
    }
  },
  "dataset": {
    "nearfield_samples_sha256": "sha256...",
    "N": 1200,
    "p_min": 0,
    "p_max": 400
  },
  "fit": {
    "r0_hat": 12345.6789,
    "mu0_hat": 3.986e14
  },
  "metrics": {
    "shape_rel_rms": 0.0123,
    "changepoint_p_star": 126,
    "post_rel_std": 0.0091
  },
  "passes": {
    "R0FIT_1_r0_positive": true,
    "R0FIT_2_shape_fit": true,
    "R0FIT_3_lock_125": true,
    "R0FIT_4_post_stability": true
  },
  "result": "PASS"
}
```


And:

- r0fit_verify_report.sha256

SAVS Companion Module — State Vectors → Near-Field Samples

SAVS-MU-EXTRACT

(Full spec, deterministic, hash-sealed)

This module takes existing measured state vectors and produces the exact dataset needed by SAVS-R0FIT:

- nearfield_samples.csv with columns:
  - pulse_index
  - r_meters
  - a_mag

It does no fitting and no assumptions about gravity vs substrate. It only computes kinematics deterministically and emits an artifact.

It is compatible with your orbital substrate model because it produces the acceleration magnitude needed to evaluate:

\|\mathbf{a}(r;p)\|=\mu_0 S(p)\frac{1}{r(r+r_0)}

A) Inputs

A1) Required file:

state_vectors.csv

```
pulse_index,rx,ry,rz,vx,vy,vz
0, 7000000.0,0.0,0.0, 0.0,7500.0,0.0
1, 6999990.0,357.0,0.0, -0.38,7499.99,0.0
...
```

Rules:

- pulse_index must be strictly increasing integers.
- Position units must be meters.
- Velocity units must be meters/second.
- Sample spacing is assumed uniform in index step (pulse index is the canonical index).

B) Canon constants (optional, only for metadata)

KKS constants are not required for extraction. This module is purely kinematic.

But for downstream integration, the report will include:

- pulse index range
- inferred sample step assumption ($\Delta s = 1$ by index)

C) Deterministic computation pipeline

Step 1 — Parse state vectors

For each row p:

- $\mathbf{r}_p = (x,y,z)$
- $\mathbf{v}_p = (v_x,v_y,v_z)$

Compute radius:

$r_p = \|\mathbf{r}_p\| = \sqrt{x^2+y^2+z^2}$

Step 2 — Compute acceleration by finite difference

Use the canonical forward difference:

$$\mathbf{a}_p = \mathbf{v}_{p+1} - \mathbf{v}_p$$

Because we are not using Chronos, this is “per index step” acceleration.

If you want acceleration in m/s², you must supply a known Δt per step. We do not do that here. We output an index-acceleration magnitude, and downstream modules can scale if desired.

Compute magnitude:

$$a^{(idx)}_p = \|\mathbf{a}_p\|$$

Step 3 — Output near-field samples

For each p from min to max-1, emit:

- pulse_index = p
- r_meters = r_p
- a_mag = a^{(idx)}_p

That produces nearfield_samples.csv.

D) Output files

D1)

nearfield_samples.csv

```
pulse_index,r_meters,a_mag
0,7000000.000000,0.412345
1,7000000.000009,0.412346
...
```

D2)

mu_extract_report.json

(hash-sealed)

```
{
  "domain": "state_vector_to_nearfield_samples",
  "model": {
    "module_id": "SAVS-MU-EXTRACT",
    "params_hash": "sha256..."
  },
  "dataset": {
    "state_vectors_sha256": "sha256...",
    "N_vectors": 10001,
    "p_min": 0,
    "p_max": 10000
  },
  "metrics": {
    "N_samples": 10000,
    "r_min": 6990000.12,
    "r_max": 7050000.34,
    "a_idx_min": 0.0012,
    "a_idx_max": 0.9321
  },
  "passes": {
    "E1_monotone_pulse_index": true,
    "E2_finite_values": true
  },
  "result": "PASS"
}
```

Also:

- mu_extract_report.sha256 = SHA-256(canonical JSON)

E) PASS/FAIL for extraction (strict but simple)

E1 — monotone index

FAIL if any pulse_index is missing or non-increasing.

E2 — finite values

FAIL if any position/velocity is NaN/Inf, or computed radius/accel is NaN/Inf.

This module will not invent values.

F) Parameter file + schema

F1)

mu_extract.params.json

```
{
  "savs_version": "SAVS-0.1",
  "kind": "module",
  "module_id": "SAVS-MU-EXTRACT",
  "model_parameters": {
    "difference_method": "forward",
    "units": {
      "position": "meters",
      "velocity": "meters_per_second",
      "accel": "per_index_step"
    }
  },
  "dataset": {
    "state_vectors_csv": "datasets/state_vectors.csv"
  }
}
```

F2) Schema:

src/modules/mu_extract/mu_extract.schema.json

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://phi.network/savs/schema/mu_extract.schema.json",
  "title": "SAVS-MU-EXTRACT Params",
  "type": "object",
  "additionalProperties": false,
  "required": ["savs_version", "kind", "module_id", "model_parameters",
"dataset"],
  "properties": {
    "savs_version": { "type": "string", "const": "SAVS-0.1" },
    "kind": { "type": "string", "const": "module" },
    "module_id": { "type": "string", "const": "SAVS-MU-EXTRACT" },
    "model_parameters": {
      "type": "object",
      "additionalProperties": false,
      "required": ["difference_method", "units"],
      "properties": {
        "difference_method": { "type": "string", "enum": ["forward"] },
        "units": {
          "type": "object",
          "additionalProperties": false,
          "required": ["position", "velocity", "accel"],
          "properties": {
            "position": { "type": "string", "const": "meters" },
            "velocity": { "type": "string", "const": "meters_per_second" },
            "accel": { "type": "string", "const": "per_index_step" }
          }
        }
      }
    },
    "dataset": {
      "type": "object",
      "additionalProperties": false,
      "required": ["state_vectors_csv"],
      "properties": {
        "state_vectors_csv": { "type": "string", "minLength": 1, "maxLength":
512 }
      }
    }
  }
}
```

F3) Report schema:

src/modules/mu_extract/mu_extract.report.schema.json

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
```

```
  "$id": "https://phi.network/savs/schema/mu_extract.report.schema.json",
  "title": "SAVS-MU-EXTRACT Report",
  "type": "object",
  "additionalProperties": false,
  "required": ["domain", "model", "dataset", "metrics", "passes", "result"],
  "properties": {
    "domain": { "type": "string", "const": "state_vector_to_nearfield_samples"
},
    "model": {
      "type": "object",
      "additionalProperties": false,
      "required": ["module_id", "params_hash"],
      "properties": {
        "module_id": { "type": "string", "const": "SAVS-MU-EXTRACT" },
        "params_hash": { "type": "string", "pattern": "^[a-f0-9]{64}$" }
      }
    },
    "dataset": {
      "type": "object",
      "additionalProperties": false,
      "required": ["state_vectors_sha256", "N_vectors", "p_min", "p_max"],
      "properties": {
        "state_vectors_sha256": { "type": "string", "pattern": "^[a-f0-9]{64}$"
},
        "N_vectors": { "type": "integer", "minimum": 2 },
        "p_min": { "type": "integer" },
        "p_max": { "type": "integer" }
      }
    },
    "metrics": {
      "type": "object",
      "additionalProperties": false,
      "required": ["N_samples", "r_min", "r_max", "a_idx_min", "a_idx_max"],
      "properties": {
        "N_samples": { "type": "integer", "minimum": 1 },
        "r_min": { "type": "number", "exclusiveMinimum": 0 },
        "r_max": { "type": "number", "exclusiveMinimum": 0 },
        "a_idx_min": { "type": "number", "minimum": 0 },
        "a_idx_max": { "type": "number", "minimum": 0 }
      }
    },
    "passes": {
      "type": "object",
      "additionalProperties": false,
      "required": ["E1_monotone_pulse_index", "E2_finite_values"],
      "properties": {
        "E1_monotone_pulse_index": { "type": "boolean" },
        "E2_finite_values": { "type": "boolean" }
      }
    },
    "result": { "type": "string", "enum": ["PASS", "FAIL"] }
  }
}
```

G) How the full chain runs (end-to-end)

1. SAVS-MU-EXTRACT

   Input: state_vectors.csv

   Output: nearfield_samples.csv + report

2. SAVS-R0FIT

   Input: nearfield_samples.csv

   Output: r_0 fit + 125-lock report

3. SAVS-MU125 (optional but recommended)

   Input: mu_series.csv (or derived)

   Output: change-point at 125 verification

SAVS Companion Module — Nearfield Samples → μ Series

SAVS-MU-SERIES

(Full spec, deterministic, hash-sealed)

This module takes the output of SAVS-MU-EXTRACT (or any pulse-indexed (r, a_mag) samples) and produces the μ-hat series needed by SAVS-MU125:

\hat\mu(p) \equiv a(p)\,r(p)^2

This comes directly from the far-field inverse-square form of your orbital substrate model:

\|\mathbf{a}(r)\|\approx \frac{\mu_{\text{eff}}}{r^2} \;\Rightarrow\; \mu_{\text{eff}}\approx \|\mathbf{a}\|r^2

(consistent with your inverse-square emergence section).

A) Inputs

A1) Required file:

nearfield_samples.csv

```
pulse_index,r_meters,a_mag
0,7000000.0,0.412345
1,7000000.1,0.412346
...
```

Rules:

- pulse_index strictly increasing integers
- r_meters > 0
- a_mag >= 0 (magnitude per index step or scaled)

B) Outputs

B1)

mu_series.csv

```
pulse_index,mu_hat
0,2.021e13
1,2.021e13
...
```

Where:

\boxed{\mu_{\hat{}}(p)=a(p)\,r(p)^2}

B2)

mu_series_report.json

+

mu_series_report.sha256

C) Parameter file

mu_series.params.json

```
{
  "savs_version": "SAVS-0.1",
  "kind": "module",
  "module_id": "SAVS-MU-SERIES",
  "model_parameters": {
    "mu_definition": "a_mag_times_r_squared",
    "units": {
      "r": "meters",
      "a": "per_index_step",
      "mu": "meters_cubed_per_index_step_squared"
    },
    "clip_negative_a_to_zero": true,
    "drop_rows_with_nonfinite": true
  },
  "dataset": {
    "nearfield_samples_csv": "datasets/nearfield_samples.csv"
  }
}
```

Notes:

- clip_negative_a_to_zero exists only as a safety valve if a numeric pipeline ever produces tiny negative values due to differencing noise. If a_mag is a magnitude, it should already be non-negative.

D) Deterministic computation pipeline

Given rows (p_i,r_i,a_i):

Step 1 — Validate

FAIL if:

- pulse indices not strictly increasing
- any r_i \le 0
- any NaN/Inf (if drop_rows_with_nonfinite=false)

Step 2 — Compute μ series

For each row:

- if clip_negative_a_to_zero=true:

  a_i \leftarrow \max(0,a_i)

- compute:

  \mu_i = a_i r_i^2

Step 3 — Emit CSV

Emit pulse_index,mu_hat.

Step 4 — Metrics

Compute:

- mu_min, mu_max, mu_mean, mu_rel_std

  \mu_{\text{rel_std}} = \frac{\mathrm{std}(\mu)}{\mathrm{mean}(\mu)}

E) PASS/FAIL gates

This module is mostly deterministic arithmetic; it fails only on invalid data.

MU-SERIES-1 — monotone pulse indices

Must be strictly increasing.

MU-SERIES-2 — finite values

All used numeric values must be finite.

MU-SERIES-3 — positive radii

All radii must be >0.

If those pass, result is PASS.

F) Output report format (hash-sealed)

mu_series_report.json

```
{
  "domain": "nearfield_to_mu_series",
  "model": {
    "module_id": "SAVS-MU-SERIES",
    "params_hash": "sha256..."
  },
  "dataset": {
    "nearfield_samples_sha256": "sha256...",
    "N": 10000,
    "p_min": 0,
    "p_max": 9999
  },
  "metrics": {
    "mu_min": 1.23e12,
    "mu_max": 4.56e12,
    "mu_mean": 2.34e12,
    "mu_rel_std": 0.0123
  },
  "passes": {
    "MU_SERIES_1_monotone_p": true,
    "MU_SERIES_2_finite": true,
    "MU_SERIES_3_r_positive": true
  },
  "result": "PASS"
}
```

Also produce:

- mu_series_report.sha256

G) Schema files

G1) Params schema:

src/modules/mu_series/mu_series.schema.json

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
```

```
  "$id": "https://phi.network/savs/schema/mu_series.schema.json",
  "title": "SAVS-MU-SERIES Params",
  "type": "object",
  "additionalProperties": false,
  "required": ["savs_version", "kind", "module_id", "model_parameters",
"dataset"],
  "properties": {
    "savs_version": { "type": "string", "const": "SAVS-0.1" },
    "kind": { "type": "string", "const": "module" },
    "module_id": { "type": "string", "const": "SAVS-MU-SERIES" },
    "model_parameters": {
      "type": "object",
      "additionalProperties": false,
      "required": ["mu_definition", "units", "clip_negative_a_to_zero",
"drop_rows_with_nonfinite"],
      "properties": {
        "mu_definition": { "type": "string", "const": "a_mag_times_r_squared"
},
        "units": {
          "type": "object",
          "additionalProperties": false,
          "required": ["r", "a", "mu"],
          "properties": {
            "r": { "type": "string", "const": "meters" },
            "a": { "type": "string", "const": "per_index_step" },
            "mu": { "type": "string", "const":
"meters_cubed_per_index_step_squared" }
          }
        },
        "clip_negative_a_to_zero": { "type": "boolean" },
        "drop_rows_with_nonfinite": { "type": "boolean" }
      }
    },
    "dataset": {
      "type": "object",
      "additionalProperties": false,
      "required": ["nearfield_samples_csv"],
      "properties": {
        "nearfield_samples_csv": { "type": "string", "minLength": 1,
"maxLength": 512 }
      }
    }
  }
}
```

G2) Report schema:

src/modules/mu_series/mu_series.report.schema.json

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://phi.network/savs/schema/mu_series.report.schema.json",
  "title": "SAVS-MU-SERIES Report",
  "type": "object",
```

```
  "additionalProperties": false,
  "required": ["domain", "model", "dataset", "metrics", "passes", "result"],
  "properties": {
    "domain": { "type": "string", "const": "nearfield_to_mu_series" },
    "model": {
      "type": "object",
      "additionalProperties": false,
      "required": ["module_id", "params_hash"],
      "properties": {
        "module_id": { "type": "string", "const": "SAVS-MU-SERIES" },
        "params_hash": { "type": "string", "pattern": "^[a-f0-9]{64}$" }
      }
    },
    "dataset": {
      "type": "object",
      "additionalProperties": false,
      "required": ["nearfield_samples_sha256", "N", "p_min", "p_max"],
      "properties": {
        "nearfield_samples_sha256": { "type": "string", "pattern":
"^[a-f0-9]{64}$" },
        "N": { "type": "integer", "minimum": 1 },
        "p_min": { "type": "integer" },
        "p_max": { "type": "integer" }
      }
    },
    "metrics": {
      "type": "object",
      "additionalProperties": false,
      "required": ["mu_min", "mu_max", "mu_mean", "mu_rel_std"],
      "properties": {
        "mu_min": { "type": "number" },
        "mu_max": { "type": "number" },
        "mu_mean": { "type": "number" },
        "mu_rel_std": { "type": "number", "minimum": 0 }
      }
    },
    "passes": {
      "type": "object",
      "additionalProperties": false,
      "required": ["MU_SERIES_1_monotone_p", "MU_SERIES_2_finite",
"MU_SERIES_3_r_positive"],
      "properties": {
        "MU_SERIES_1_monotone_p": { "type": "boolean" },
        "MU_SERIES_2_finite": { "type": "boolean" },
        "MU_SERIES_3_r_positive": { "type": "boolean" }
      }
    },
    "result": { "type": "string", "enum": ["PASS", "FAIL"] }
  }
}
```

H) End-to-end orbital verification chain (now complete)

With these modules, the full pipeline is deterministic:

1. SAVS-MU-EXTRACT

   state_vectors.csv → nearfield_samples.csv

2. SAVS-MU-SERIES

   nearfield_samples.csv → mu_series.csv

3. SAVS-MU125

   mu_series.csv → verifies lock at 125

4. SAVS-R0FIT

   nearfield_samples.csv → fits r_0 and verifies near-field curve + lock

This is now a complete test harness for the addendum.

Orbital Proof Bundle v1.0

One-command, hash-sealed end-to-end bundle that verifies:

- μ-lock at 125 pulses (Φ-stiffness coupling)
- near-field curve fit \Xi(r)=\frac{r}{r+r_0}
- fully deterministic artifacts and manifests

This bundle wires together, in order:

1. SAVS-MU-EXTRACT: state vectors → nearfield samples

2. SAVS-MU-SERIES: nearfield samples → μ series
3. SAVS-MU125: μ series → 125 lock verifier
4. SAVS-R0FIT: nearfield samples → r_0 fit + 125 lock + curve fit

The physics basis is your orbital substrate paper (harmonic scalar + log-gradient) .

---

1) Bundle directory layout

```
Orbital_Proof_Bundle_v1/
  bundle_index.json

  params/
    mu_extract.params.json
    mu_series.params.json
    mu125.params.json
    r0fit.params.json

  datasets/
    state_vectors.csv

  out/                          (generated)
    datasets/
      nearfield_samples.csv
      mu_series.csv

    reports/
      mu_extract_report.json
      mu_series_report.json
      mu125_verify_report.json
      r0fit_verify_report.json

    hashes/
      bundle_hashes.json
      bundle_hashes.sha256
      reports_hashes.json
      reports_hashes.sha256

    sav_manifest.json
    sav_manifest.sha256
```

Only required human-provided dataset: datasets/state_vectors.csv

Everything else is generated deterministically.

---

2) bundle_index.json (wires the run)

```
{
  "savs_version": "SAVS-0.1",
  "bundle_name": "Orbital_Proof_Bundle_v1",
  "modules": ["SAVS-MU-EXTRACT", "SAVS-MU-SERIES", "SAVS-MU125", "SAVS-R0FIT"],
  "params": {
    "global": "params/global.json",
    "per_module": {
      "SAVS-MU-EXTRACT": "params/mu_extract.params.json",
      "SAVS-MU-SERIES": "params/mu_series.params.json",
      "SAVS-MU125": "params/mu125.params.json",
      "SAVS-R0FIT": "params/r0fit.params.json"
    }
  },
  "datasets": {
    "SAVS-MU-EXTRACT": {
      "state_vectors_csv": "datasets/state_vectors.csv"
    },
    "SAVS-MU-SERIES": {
      "nearfield_samples_csv": "out/datasets/nearfield_samples.csv"
    },
    "SAVS-MU125": {
      "mu_series_csv": "out/datasets/mu_series.csv"
    },
    "SAVS-R0FIT": {
      "nearfield_samples_csv": "out/datasets/nearfield_samples.csv"
    }
  },
  "notes": "End-to-end orbital substrate verification: state-vectors →
mu_hat(p) → 125 lock + near-field signature fit."
}
```

---

3) Parameter files (complete templates)

3.1 params/mu_extract.params.json

```
{
  "savs_version": "SAVS-0.1",
  "kind": "module",
  "module_id": "SAVS-MU-EXTRACT",
  "model_parameters": {
    "difference_method": "forward",
    "units": {
      "position": "meters",
      "velocity": "meters_per_second",
```

```
      "accel": "per_index_step"
    }
  },
  "dataset": {
    "state_vectors_csv": "datasets/state_vectors.csv"
  }
}
```

3.2 params/mu_series.params.json

```
{
  "savs_version": "SAVS-0.1",
  "kind": "module",
  "module_id": "SAVS-MU-SERIES",
  "model_parameters": {
    "mu_definition": "a_mag_times_r_squared",
    "units": {
      "r": "meters",
      "a": "per_index_step",
      "mu": "meters_cubed_per_index_step_squared"
    },
    "clip_negative_a_to_zero": true,
    "drop_rows_with_nonfinite": true
  },
  "dataset": {
    "nearfield_samples_csv": "out/datasets/nearfield_samples.csv"
  }
}
```

3.3 params/mu125.params.json

```
{
  "savs_version": "SAVS-0.1",
  "kind": "module",
  "module_id": "SAVS-MU125",
  "model_parameters": {
    "P": 11,
    "S": 44,
    "B": 36,
    "N_day": 17491.270421,
    "phi": 1.618033988749895,
    "stiff_index": 125,
    "delta_window": 10
  },
  "thresholds": {
    "pre_fit_rel_rms_max": 0.05,
    "post_plateau_rel_std_max": 0.01,
    "changepoint_within_window": true
  },
  "dataset": {
    "mu_series_csv": "out/datasets/mu_series.csv"
  }
}
```

3.4 params/r0fit.params.json

```
{
  "savs_version": "SAVS-0.1",
  "kind": "module",
  "module_id": "SAVS-R0FIT",
  "model_parameters": {
    "P": 11,
    "S": 44,
    "B": 36,
    "N_day": 17491.270421,
    "phi": 1.618033988749895,
    "stiff_index": 125,
    "delta_window": 10,
    "fit_p_min": 0,
    "fit_p_max": 400
  },
  "thresholds": {
    "shape_fit_rel_rms_max": 0.05,
    "lock_changepoint_max_abs_p": 10,
    "post_lock_rel_std_max": 0.02,
    "r0_positive_required": true
  },
  "dataset": {
    "nearfield_samples_csv": "out/datasets/nearfield_samples.csv"
  }
}
```

---

4) Dataset format (single required input)

datasets/state_vectors.csv

```
pulse_index,rx,ry,rz,vx,vy,vz
0,7000000.0,0.0,0.0,0.0,7500.0,0.0
1,6999999.9,357.0,0.0,-0.4,7499.9,0.0
...
```

- pulse_index is your canonical index.
- No timestamps required.
- The extraction defines acceleration “per index step.” This is consistent with Chronos-free evaluation.

---

5) One-command execution (the actual run)

From the SAVS engine root:

```
savs hash --bundle Orbital_Proof_Bundle_v1
savs run  --bundle Orbital_Proof_Bundle_v1 --out Orbital_Proof_Bundle_v1/out
--modules SAVS-MU-EXTRACT,SAVS-MU-SERIES,SAVS-MU125,SAVS-R0FIT
savs verify --out Orbital_Proof_Bundle_v1/out
```

---

6) Manifest outputs (hash sealed)

6.1 out/sav_manifest.json (template)

```
{
  "version": "SAVS-0.1",
  "engine": {
    "name": "savs",
    "engine_version": "0.1.0",
    "lock_hash": "sha256-of-pnpm-lock-or-equivalent"
  },
  "bundle_hashes_sha256": "sha256-of-out/hashes/bundle_hashes.json",
  "modules": {
    "SAVS-MU-EXTRACT": {
      "report_path": "reports/mu_extract_report.json",
      "report_sha256": "sha256...",
      "result": "PASS"
    },
    "SAVS-MU-SERIES": {
      "report_path": "reports/mu_series_report.json",
      "report_sha256": "sha256...",
      "result": "PASS"
    },
    "SAVS-MU125": {
      "report_path": "reports/mu125_verify_report.json",
      "report_sha256": "sha256...",
      "result": "PASS"
    },
    "SAVS-R0FIT": {
      "report_path": "reports/r0fit_verify_report.json",
      "report_sha256": "sha256...",
      "result": "PASS"
    }
  },
  "global_pass": true
}
```

6.2 out/sav_manifest.sha256

- SHA-256 of the canonical JSON bytes of sav_manifest.json

---

7) What “PASS” means for the orbital substrate + Φ-stiffness addendum

If this bundle passes end-to-end, then:

1. The 125-pulse stiffness lock is present in the measured/derived μ-series (SAVS-MU125 PASS).
2. The near-field samples fit the exact substrate signature:

   \Xi(r)=\frac{r}{r+r_0}

   with positive r_0 and low shape error (SAVS-R0FIT PASS).

3. The above is consistent with the inverse-square emergence from your orbital substrate formalism.

That is a concrete, falsifiable, hash-sealed artifact bundle.
