OPERATION KOMOREBI

REALIZATION CALCULUS



RESEARCH LIBRARY

Certified Bayesian Inference

One-observation Gaussian updates, exact packets, and outward-rounded evidence

Traditional paper

Komorebi P.

Standardized publication edition / 8 October 2026

REALIZATION CALCULUS

Reading scope

A complete finite-dimensional, one-observation inference construction
Based on Certified Inference Core 1.0.0

Abstract

We present a certified finite-dimensional Gaussian inference construction that connects an executable packet solver to a posterior measure and to an explicit arithmetic output contract. Model data are exact rational factors. The prior covariance may be singular, and the observation operator may be rectangular. A positive noise floor makes the innovation covariance invertible. The canonical solver maintains an exact cache of orthogonal output-packet residuals, selects a largest-energy packet, and updates only that packet. Its trajectory agrees with the recomputed greedy trajectory. A geometric residual bound supplies computable finite fuel. After an exact stopping test establishes a strict verification margin, an outward-rounded dyadic verifier certifies a Wasserstein tolerance. The final result also contains enclosing mean and covariance intervals for the Joseph Gaussian at the returned gain, with prescribed maximum radii. We give the underlying algebra, proof composition, worked executions, arithmetic distinctions, and verification boundaries. A checked scalar counterexample shows that independently rounding the gain and cached residual centers does not preserve the exact-cache invariant. The integrated result is complete for one observation update under its declared hypotheses. Sequential filtering, a rounded selective cache, executable sampling, and a universally refined host implementation are separate extensions.

Reading contract

This traditional paper develops the mathematics and the exact program contract. The extended visual guide follows two concrete models through the same chain, with twenty figures generated by Python from exact data or explicit mathematical relationships. The Lean companion gives actual source excerpts, hypotheses, dependency maps, archived evidence, and pinned reproduction instructions. Editable LaTeX, figure programs, exact example outputs, audit reports, and the intact formal release accompany all three documents.

The formal baseline is `FD_Certified_Inference_Core_v1.0.0_Checked.zip`. Its archived build belongs to 12 September 2026. This publication edition separates that evidence from the fresh source and Python checks performed while preparing the papers. The compiler and pinned dependency runtime were not available for a fresh Lean build here. No finite test, source inventory, or rendered page is described as a new kernel check.

Contents

1 The result and its role **4**
1.1 Why the packet representation matters . . . 5
1.2 Arithmetic is part of the statement . . . 6
1.3 Relation to established methods . . . 6
1.4 Evidence and reproduction . . . 6

2 A finite Gaussian model with an inspectable input contract **7**
2.1 Factor-coded rational inputs . . . 7
2.2 Why the noise floor is more than an input check . . . 8

3 The explicit posterior is a genuine conditional Gaussian law **9**
3.1 A positive covariance without an inverse prior . . . 9
3.2 Forward and reverse constructions of the same joint law . . . 9
3.3 Pointwise posterior version and almost-everywhere uniqueness . . . 10

4 An approximate gain defines a valid Joseph Gaussian **11**
4.1 Residual algebra separates the target from its approximation . . . 11
4.2 A shared-noise construction of the two laws . . . 12
4.3 Singular support is compatible with candidate certification . . . 12

5 An inverse-free residual-energy certificate **13**
5.1 What the certificate depends on . . . 14
5.2 The two output tolerances have different meanings . . . 14

6 Packets resolve a complete residual energy **15**
6.1 Orthogonality gives exact accounting . . . 15
6.2 Majorants compose with the posterior certificate . . . 16
6.3 Why partial packet accounting is insufficient for stopping . . . 17

7 From finite rational checks to a real-law guarantee **17**
7.1 Validity and tolerance are separate decisions . . . 17
7.2 The certificate is independent of proposal quality . . . 18

8 A singular-prior calculation through the full semantic chain **19**

9 An exact selective iteration with a preserved cache **20**

10 Selection, exact trajectory equality, and finite fuel **21**

11 Independent final verification and precision fallback **23**

12 Parameter arrays and the meaning of a complete result **25**

13 A complete coordinate execution **26**

14 A singular rotated execution **27**

15 The rounded-cache counterexample and the separate rounded solver **28**

16 Measured work, execution boundaries, and extensions **30**

17 Assurance, reproducibility, and the completed scope **31**

17.1 What was rechecked for this publication 32

17.2 Interpreting cost evidence 32

17.3 Extensions that preserve the completed result 33

18 Conclusion **34**

A Source and reproduction map **34**

1 The result and its role

An approximate inference program needs a conclusion about the object it returns. A small internal residual, a visually plausible covariance, and a successful numerical routine address different obligations. In this construction, a single result joins them through exact statements. The gain returned by the permitted iteration defines a Gaussian approximation. A certificate controls its distance from a proved posterior version. Parameter intervals enclose the mean and covariance of that same approximation. The result ties its reported arrays to the actual outward-rounded evaluator.

The problem is one Gaussian observation update. Write the prior as

$$X \sim \mathcal{N}(m, P), \qquad Y = HX + \eta,$$

where the centered Gaussian noise η is independent of X. The representation uses rational factors

$$P = LL^{\mathsf{T}}, \qquad R = NN^{\mathsf{T}} + \sigma I, \qquad \sigma > 0.$$

All data supplied to the executable construction, including m, H, L, N, σ and the observation y, are rational. Finite state and observation index types determine the matrix shapes. There is no requirement that H be square or that P be invertible. The innovation matrix is

$$S = HPH^{\mathsf{T}} + R.$$

Its positive noise floor supports the inverse used in the exact posterior semantics. The executable iterative solver avoids computing that inverse to construct its returned gain.

For a rational gain B, the Joseph approximation has mean and covariance

$$\begin{aligned} m_B &= m + B(y - Hm), \\ P_B &= (I - BH)P(I - BH)^{\mathsf{T}} + BRB^{\mathsf{T}}. \end{aligned}$$

Its covariance is positive semidefinite by construction. The approximation is a measure, rather than a collection of unqualified decimal entries. The exact gain $K = PH^{\mathsf{T}}S^{-1}$ identifies the familiar Gaussian posterior parameters. Later sections prove that this explicit Gaussian kernel disintegrates the actual forward joint law. This gives a posterior version defined at every supplied observation. Agreement with Mathlib's canonical conditional kernel is an evidence-almost-everywhere statement, as is appropriate for conditional distributions.

Theorem 1.1 (Publication contract of the canonical solver)**.** *Fix a finite-dimensional rational factor-coded model with $\sigma > 0$, and complete, mutually orthogonal self-adjoint rational output and input packet families. Supply an output-packet seed and a finite scan containing every output label. Let $\epsilon, \delta > 0$ be rational. An arbitrary rational initial gain and an arbitrary finite list of requested precision probes are permitted.*

The canonical program `Komorebi.CachedPosteriorRunCached` returns a result r. If B is its candidate gain and Π_y is the explicitly proved Gaussian posterior version, then

$$W_2(\Pi_y, \mathcal{N}(m_B, P_B)) \le \epsilon.$$

The candidate and tolerance predicates hold. The reported mean and covariance intervals enclose m_B and P_B, respectively, and every reported radius is at most δ. Its parameter precision is the program's computed output precision, and its arrays equal the declared evaluator at that precision.

Any successful return of the same program is sound without caller-supplied validity proofs: its internal admission and verification steps imply the same result predicate. The positive-input hypotheses above establish guaranteed successful termination.

Proof composition. The source declaration `Komorebi.CachedPosterior.runCached_terminates` obtains a successful certified gain from `cached_solver_reaches_posterior_tolerance`. That theorem combines initialization, cache preservation, greedy-trajectory equivalence, finite fuel, and a sufficient fallback verification precision. The Gaussian semantic bridge converts its residual certificate into the Wasserstein conclusion. The output theorem `represent_spec` provides parameter enclosure and the radius bound at positive δ. The definition of the result record supplies both reported-evaluator equalities. The separate source declaration `runCached_sound` derives these properties from an actual successful return. The mathematical steps are developed in the following chapters and traced to their exact declarations in the companion. □

The two tolerances have different purposes. The parameter tolerance δ is an enclosure radius for the parameters of the returned approximation. It is not an entrywise error against the exact posterior covariance. The law tolerance ϵ compares the whole approximation with the posterior in W_2. Neither conclusion certifies a Gaussian formed by discarding every interval radius and using its centers as exact parameters. Such a replacement would need a separate perturbation argument, including a positive-semidefinite covariance obligation.

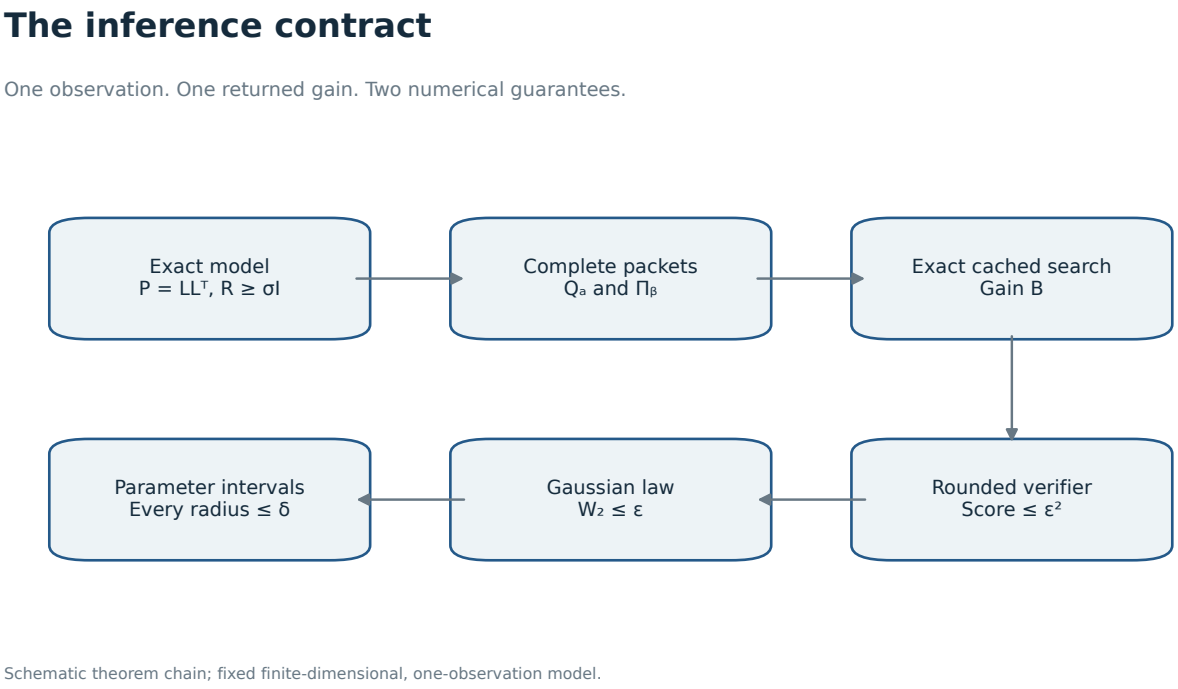


Figure 1: The complete contract links an exact rational model, a permitted cached iterate, rounded verification, a posterior-measure guarantee, and enclosing parameter arrays. The two output tolerances belong to different conclusions.

1.1 Why the packet representation matters

The residual is $F(B) = PH^{\mathsf{T}} - BS$. Orthogonal output projectors Q_a split it into components $G_a = Q_a F(B)$. Completeness makes the sum of their squared Frobenius energies equal the energy of the full residual. Selecting a largest-energy component therefore chooses a block containing a definite fraction of

the remaining error.

The exact cache retains these components, their energies, and their total. For a selected label a, the update changes B by τG_a, changes G_a by right multiplication with $I - \tau S$, and leaves every other cached component unchanged. The proof of that last statement uses orthogonality of the output projectors. It is an algebraic locality result. Keeping unrelated cache entries untouched is justified by their semantic invariant.

The fixed packet family is part of the model of execution. It can represent coordinate blocks or rotated subspaces. The theorem is not restricted to a preferred coordinate basis. Changing the family during the run, however, changes the state invariant and is outside this solver's proved trajectory theorem. The singular rotated worked example demonstrates a non-coordinate packet family without making a claim about changing hierarchies.

1.2 Arithmetic is part of the statement

The cached branch uses exact rational gain updates and exact rational cache scores. It does not independently round the selected gain and cached residual during iteration. Final certificate evaluation and parameter representation use outward-rounded dyadic operations. Those rounded operations have explicit enclosure and inflation bounds. A separately preserved solver uses rounded full-gain updates and has its own convergence theorem. These are distinct verified branches.

The naive rounded-cache counterexample explains the distinction. In a scalar model, exact gain and residual both equal $1/3$ after one step. Rounding both centers independently to zero fractional bits makes both stored values zero. The stored residual score then says zero, although the true residual at the stored gain has score two. The independent final interval verifier rejects that candidate. Sound final verification survives, but the invalid cache cannot supply the exact-cache progress proof.

1.3 Relation to established methods

Maximum-energy packet selection has a direct optimization interpretation. For the quadratic objective

$$\phi(B) = \tfrac{1}{2}\operatorname{tr}(BSB^{\mathsf{T}}) - \langle B, PH^{\mathsf{T}}\rangle_{\mathrm{F}},$$

the gradient is $BS - PH^{\mathsf{T}} = -F(B)$. The selected packet has largest projected-gradient norm. This is a block Gauss–Southwell mechanism, and the residual cache is maintained by exact linear identities. Greedy coordinate and block selection are established methods [2, 3]. Pathwise Gaussian conditioning and Wasserstein approximation analysis also have substantial prior literature [4]. Formally checked interval reasoning for rounded arithmetic is an established methodology [5].

The contribution presented here is the integrated formal construction: a specified executable trajectory, maintained semantic cache, computable fuel, sufficient verification precision, posterior guarantee, and interval output tied to the actual returned record. The paper proves this construction under its declared hypotheses. It makes no global priority claim or competitive-performance claim from the small archived benchmark.

1.4 Evidence and reproduction

The archived release records successful compilation of 117 active modules and axiom reports for 543 named theorem or lemma declarations. Its allowlist is `propext`, `Classical.choice`, and `Quot.sound`. Those counts include support lemmas and benchmark certificates. They count different forms of evidence rather than independent mathematical discoveries.

The original source release contains successful receipts, source snapshots, logs, dependency records, exact diagnostic outputs, and preserved failures. It excludes the portable compiler and compiled dependency runtime. Fresh publication checks can verify retained source and log hashes and reproduce exact Python examples. Archived compiled-output hashes remain recorded values when the corresponding binaries are absent. The Lean companion states precisely which evidence was replayed here and which remains archived.

The reference Python producer accepts exact rational strings and rejects raw JSON floating-point numbers. Its successful status is provisional until the emitted explicit Lean instance is compiled. A checked instance certifies the rational data present in that source. It does not prove a universal JSON-parser or Python-exporter refinement. The resource-capped producer may fail with a resource limit even when the ideal mathematical solver has finite fuel. These distinctions make the delivered object reproducible and its authority inspectable.

2 A finite Gaussian model with an inspectable input contract

The inference problem concerns one observation of a finite-dimensional latent state. Let the state space be $\mathbb{R}^n$ and the observation space be $\mathbb{R}^p$. A state X has prior law $\mathcal{N}(m, P)$, and its observation is

$$Y = HX + V, \qquad V \sim \mathcal{N}(0, R), \qquad V \text{ independent of } X. \tag{1}$$

Here $H \in \mathbb{R}^{p\times n}$ may be rectangular, and $P \in \mathbb{R}^{n\times n}$ may be singular. The assumptions are $P \succeq 0$ and $R \succ 0$. A singular prior is an ordinary Gaussian probability law concentrated on an affine subspace. The argument does not replace that law by a nonsingular approximation or assign an artificial inverse to its covariance.

We use Euclidean vector norms and the squared Frobenius energy

$$E(A) = \|A\|_{\mathrm{F}}^2 = \sum_{i,j} A_{ij}^2. \tag{2}$$

The distinction between these norms matters. A matrix energy is a finite sum of scalar squares that can be evaluated exactly on rational inputs. An operator norm is used in the proof that this energy controls distribution error. The certificate never asks a producer to supply an unverified operator norm or an approximate eigenvalue calculation.

For an observed value y, put

$$d = y - Hm, \qquad S = HPH^\top + R, \qquad K = PH^\top S^{-1}. \tag{3}$$

The matrix S is the innovation covariance. It is positive definite because, for every nonzero $z \in \mathbb{R}^p$,

$$z^\top Sz = (H^\top z)^\top P(H^\top z) + z^\top Rz > 0.$$

Consequently the inverse in the definition of K exists even when P has a nontrivial nullspace or H has deficient rank. Every inverse used by the posterior argument is an inverse on the observation space. This is the structural reason singular priors are supported.

2.1 Factor-coded rational inputs

The completed executable contract represents its model by rational matrices and vectors. Its input consists of H, m, y, a prior factor $L \in \mathbb{Q}^{n\times r}$, a noise factor $N \in \mathbb{Q}^{p\times s}$, and a rational number σ. The

covariance matrices are derived from these inputs:

$$P = LL^\top, \qquad R = NN^\top + \sigma I_p, \qquad \sigma > 0. \tag{4}$$

The factor dimensions r and s are finite and need not equal the state or observation dimension. The factors may have dependent columns or zero entries. The prior factor is allowed to have rank below n. The positive floor is checked rather than inferred from a decimal approximation.

This representation makes the covariance hypotheses concrete. For every x,

$$x^\top LL^\top x = \|L^\top x\|^2 \geq 0, \qquad x^\top Rx = \|N^\top x\|^2 + \sigma\|x\|^2.$$

Thus P is positive semidefinite and $R \succeq \sigma I_p \succ 0$. These conclusions are proved from the factors; they are not additional statistical assumptions hidden inside a covariance record. The real semantics interprets each rational coordinate through the exact inclusion $\mathbb{Q} \hookrightarrow \mathbb{R}$. There is no intermediate conversion through hardware floating point.

Factor coding is a declared restriction of the executable input language. The release does not prove that every real positive semidefinite matrix can be supplied as a rational Gram factor, or that every rational positive semidefinite matrix has a rational factor of the chosen dimensions. A user begins with the represented factors. The theorem then concerns the exact real Gaussian model determined by those factors. If an application first approximates measured real data by rationals, an error analysis for that approximation is a separate obligation.

The record `RationalModel` in `Komorebi/RationalCertificateChecker.lean` stores the factors, the floor, and the observation data. Its methods `prior`, `noise`, `innovation`, `residual`, and `innovationVect or` derive the displayed quantities. `RationalModelSemantics.lean` proves their interpretation in real Euclidean spaces, including `prior_real_posSemidef`, `noise_real_posDef`, `noise_real_floor`, and `real_residual`. These are the connection between the exact input record and the probability measures used later.

2.2 Why the noise floor is more than an input check

The same σ that certifies positive noise also bounds the inverse innovation operator. Adding the positive semidefinite term $HPH^\top$ gives

$$z^\top Sz \geq \sigma\|z\|^2 \qquad \text{for all } z \in \mathbb{R}^p. \tag{5}$$

Let $x = S^{-1}v$. Then $Sx = v$, and the Cauchy–Schwarz inequality yields

$$\sigma\|x\|^2 \leq x^\top Sx = x^\top v \leq \|x\|\|v\|.$$

If $x = 0$ the desired bound is immediate. Otherwise division by $\sigma\|x\|$ gives $\|S^{-1}v\| \leq \sigma^{-1}\|v\|$. Hence

$$\|S^{-1}\|_{\mathrm{op}} \leq \sigma^{-1}. \tag{6}$$

This elementary estimate supplies an explicit certificate constant without computing the inverse. The inverse exists in the semantic proof, while the executable stopping test uses σ, an innovation vector, and a residual. This separation allows an approximate gain to be certified independently of the method that produced it.

The source names these steps `innovation_quadraticLowerBound` and `innovation_inverse_norm_le` in `FDInnovationConditioning.lean`. The latter uses the actual inverse identity, so its conclusion is not an assumption about an arbitrary supplied matrix. A false inverse or an overstated floor cannot be inserted as an accepted semantic shortcut.

3 The explicit posterior is a genuine conditional Gaussian law

The exact posterior formula is

$$\nu_y = \mathcal{N}(m + Kd, C_*), \qquad C_* = P - KHP. \tag{7}$$

Writing this formula does not by itself establish conditional probability semantics. A matrix identity could describe a plausible mean and covariance without proving that its Gaussian is a disintegration of the prior and likelihood. The source constructs that disintegration at the level of probability measures. This section records the argument and the precise meaning of the word posterior.

3.1 A positive covariance without an inverse prior

The covariance C_* admits the Joseph representation

$$C_* = (I_n - KH)P(I_n - KH)^\top + KRK^\top. \tag{8}$$

Each term on the right is positive semidefinite. Therefore $C_* \succeq 0$, including at the singular boundary. This avoids any proof based on P^{-1} or C_*^{-1}.

Lemma 3.1 (Exact Joseph identity)**.** *Under the assumptions of Section 2, the covariance in* (7) *equals* (8)*.*

Proof. The innovation equation gives $KS = PH^\top$. Since P and S are symmetric, transposition gives $SK^\top = HP$. Thus

$$KSK^\top = KHP = PH^\top K^\top.$$

Expanding the right side of (8) and collecting its two quadratic terms gives

$$\begin{aligned}(I_n - KH)P(I_n - KH)^\top + KRK^\top \\ = P - KHP - PH^\top K^\top + KSK^\top \\ = P - KHP.\end{aligned}$$

The last line is C_*. Positivity follows because each unexpanded term is a covariance congruence of a positive semidefinite matrix. □

The source proves `josephCovMatrix_eq_posteriorCovMatrix` and `posteriorCovMatrix_posSemidef` in `FDPosteriorCovPSD.lean`. The posterior kernel is then built as a Gaussian affine kernel, rather than as a formal expression requiring a nonexistent Lebesgue density on a singular state space.

3.2 Forward and reverse constructions of the same joint law

The forward construction samples $X \sim \mathcal{N}(m, P)$ and then samples Y with conditional law $\mathcal{N}(HX, R)$. Its joint mean, in the order (Y, X), is (Hm, m). Its covariance is

$$\begin{pmatrix} S & HP \\ PH^\top & P \end{pmatrix}. \tag{9}$$

This joint law is Gaussian because it is an affine image of independent Gaussian inputs. The argument remains valid when the input covariance has zero eigenvalues.

For the reverse construction, sample $Y \sim \mathcal{N}(Hm, S)$ and an independent $E_* \sim \mathcal{N}(0, C_*)$. Define

$$Z = m + K(Y - Hm) + E_*.$$

Conditional on $Y = y$, the law of Z is precisely ν_y. The joint mean of (Y, Z) is again (Hm, m). Its cross covariance is $SK^\top = HP$, and its state covariance is

$$KSK^\top + C_* = KHP + (P - KHP) = P.$$

It therefore has exactly the covariance in (9). Both constructions yield Gaussian measures with the same mean and covariance, so they yield the same joint probability measure. This is a full law identity, rather than a statement that the two constructions merely agree in their first two moments without Gaussianity.

Theorem 3.2 (Gaussian disintegration)**.** *Let $\ell(x, \cdot) = \mathcal{N}(Hx, R)$ be the observation kernel and let $\eta = \mathcal{N}(Hm, S)$ be the evidence law. Then the explicit kernel $y \mapsto \nu_y$ satisfies*

$$\eta(dy)\,\nu_y(dx) = \mathcal{N}(m, P)(dx)\,\ell(x, dy) \tag{10}$$

after exchanging the order of the two coordinates on the right. In particular, η is the actual observation marginal and ν is a posterior version for this prior and likelihood.

Proof. The forward and reverse constructions just described have the same joint Gaussian mean and covariance. The reverse construction is the product of the evidence measure with its affine Gaussian conditional kernel. The forward construction is the product of the prior measure with the likelihood kernel. Their joint equality is (10). Taking the observation marginal yields the stated evidence law, and equality of the joint measures is the defining disintegration property. □

The raw declaration `fd_joint_disintegration` in `FDGaussianConditioning.lean` is this joint equality. `fd_observation_evidence` derives the evidence marginal by applying the first-coordinate marginal to that equality. These theorems use the actual measure constructions, including the joint packing and covariance identities proved in their imported modules.

3.3 Pointwise posterior version and almost-everywhere uniqueness

A regular conditional distribution is determined by its joint law only up to sets of evidence measure zero. The source accordingly proves that the explicit Gaussian kernel agrees with Mathlib's canonical Bayesian posterior evidence-almost everywhere. The relevant declarations are `fdPosteriorKernel_eq_canonical`, `canonicalPosterior_eq_multivariateGaussian`, and `canonicalPosterior_eq_josephGaussian`. Their almost-everywhere quantifier is part of the theorem and must remain visible when translating it to prose.

For a supplied rational observation y, the executable contract uses the explicit version ν_y. That version is defined at every such y and is already proved to disintegrate the model. The solver's distance theorem is pointwise against this version. It does not prove pointwise equality to every possible modification of a canonical posterior kernel on evidence-null sets. The distinction is standard for conditional laws and does not weaken the law-level certificate that the program actually returns.

In particular, observing a single real vector in a continuous model does not require the event $Y = y$ to have positive probability. Conditioning is encoded by a kernel whose integration reconstructs the joint law. The proof cannot be replaced by dividing the probability of a zero-probability event. This is one reason the measure-level construction is essential when the inference contract claims a posterior distribution guarantee.

4 An approximate gain defines a valid Joseph Gaussian

Let $B \in \mathbb{R}^{n\times p}$ be an arbitrary gain candidate. It need not solve the innovation equation and need not have been produced by the packet algorithm. Define

$$m_B = m + Bd, \tag{11}$$

$$C_B = (I_n - BH)P(I_n - BH)^\top + BRB^\top, \tag{12}$$

$$\mu_B = \mathcal{N}(m_B, C_B). \tag{13}$$

Every C_B is positive semidefinite, since it is a sum of covariance congruences. Thus every candidate gain determines a genuine Gaussian law. This is useful for certification: an imperfect candidate is a valid distribution with a measurable error, rather than a covariance formula that can become negative merely because B is inexact.

One should not replace (12) by $P - BHP$. The latter is the exact posterior covariance only when the required innovation identities hold. For an arbitrary candidate it need not even be symmetric. The Joseph form makes the mathematical object independent of those identities, and the final verifier certifies this same object.

4.1 Residual algebra separates the target from its approximation

Define the gain-equation residual

$$F(B) = PH^\top - BS. \tag{14}$$

It can be computed from the model and B without using K. The exact target satisfies $KS = PH^\top$, so

$$F(B) = (K - B)S, \qquad K - B = F(B)S^{-1}. \tag{15}$$

These are exact identities, not perturbative approximations. They explain how a computable defect in a linear equation controls an uncomputed target gain.

Proposition 4.1 (Excess covariance)**.** *For every gain B,*

$$C_B - C_* = (K - B)S(K - B)^\top = F(B)S^{-1}F(B)^\top \succeq 0. \tag{16}$$

Proof. Expansion of the Joseph form gives

$$C_B = P - BHP - PH^\top B^\top + BSB^\top.$$

Subtract $C_* = P - KHP$. Expanding $(K - B)S(K - B)^\top$ and using $KS = PH^\top$, $SK^\top = HP$, and $KSK^\top = KHP$ gives the same expression. The second equality follows from (15) and the symmetry of S. Since $S \succ 0$, its covariance congruence by $K - B$ is positive semidefinite. □

This identity is recorded as `gainJoseph_sub_posterior_eq_gap` and `gainJoseph_sub_posterior_eq_residual`; positivity is `gainJoseph_excess_posSemidef`, all in `FDInexactGainAlgebra.lean`. It shows that the approximate Joseph law has additional covariance relative to the exact posterior. It does not imply that its mean is exact, or that it is the Bayesian posterior of a second model chosen by the program.

4.2 A shared-noise construction of the two laws

Take independent centered Gaussian variables $U \sim \mathcal{N}(0, P)$ and $V \sim \mathcal{N}(0, R)$. For each gain B, define the mathematical affine random variable

$$T_B = m + Bd + (I_n - BH)U - BV. \tag{17}$$

Its mean is m_B and its covariance is C_B, so its law is μ_B. At $B = K$, the exact Joseph identity identifies its law with ν_y. The pair (T_K, T_B), using the same inputs (U, V), is therefore an actual transport plan between the exact posterior version and the approximate law.

This is a mathematical random-variable construction used to prove a coupling bound. The release does not include an executable random sampler or a refinement theorem for a pseudorandom implementation. The raw name `gainSample` denotes the affine map in this construction. The finite executable program returns a represented gain and certified parameter enclosures, rather than samples from a Gaussian distribution.

Let $\Delta = K - B$. Subtraction in (17) gives

$$T_K - T_B = \Delta d - \Delta(HU + V).$$

The random vector $HU + V$ is centered and has covariance S. The difference therefore has mean Δd and covariance $\Delta S \Delta^\top$. Its expected squared Euclidean norm is

$$\mathbb{E}\|T_K - T_B\|^2 = \|\Delta d\|^2 + \operatorname{tr}(\Delta S \Delta^\top). \tag{18}$$

The source proves both marginals and this cost. It does not assume that a producer has supplied a coupling with the desired properties. The load-bearing declarations are `gainSample_law`, `gainSample_exact_posterior_law`, and `inexactGainCoupling_isTransportPlan` in `FDGainSampler.lean`, followed by `inexactGainCoupling_cost_gap` in `FDInexactGainCoupling.lean`.

The common-noise plan need not be optimal. Its value is that it is explicit and its cost has an exact residual expression. The Wasserstein distance takes an infimum over all transport plans, so the cost of any valid plan supplies an upper bound. No closed formula for the optimal transport between two covariance matrices is required.

4.3 Singular support is compatible with candidate certification

The target posterior can be supported on a proper affine subspace while an arbitrary candidate law is supported on a larger set. The construction does not require the candidate to preserve the target's support exactly. It requires a bound on the distance between the two actual measures. The explicit coupling remains meaningful because both measures are obtained from the same independent Gaussian inputs, even if the resulting affine maps have different ranks.

The extreme case $P = 0$ makes this especially clear. The prior is the point mass at m, and the model's observation carries no uncertainty about the state. The exact gain is $K = 0$, and the posterior version is still the point mass at m. For an arbitrary candidate B, however, the Joseph approximation has mean $m + Bd$ and covariance $BRB^\top$. These are valid Gaussian parameters, but they need not describe a point mass. Its residual is $F(B) = -BR$, so the residual certificate measures the error introduced by that candidate rather than declaring every valid Gaussian approximation exact.

For a candidate $B = 0$, both its mean and covariance are exactly those of the point mass, and its residual energy is zero. This boundary calculation shows why the proof separates covariance validity from

posterior accuracy. The Joseph formula supplies validity for every gain. The residual supplies accuracy for the selected gain. Neither property is inferred from the other.

These observations use only the same general matrix and coupling identities. They do not require a positive prior eigenvalue, an inverse covariance on the target support, or an auxiliary regularization parameter. The declared positive floor regularizes the observation covariance, which is precisely the space where the innovation equation must be solved.

5 An inverse-free residual-energy certificate

For probability laws with finite second moment, write

$$W_2(\nu,\mu)^2 = \inf_{\pi\in\Pi(\nu,\mu)} \int \|x-z\|^2 \, d\pi(x,z),$$

where $\Pi(\nu,\mu)$ is the set of transport plans with the stated marginals. The formal source defines the squared quantity as a nonnegative extended-real infimum and obtains the distance by a square-root operation. The explicit finite Gaussian coupling proves that the quantities considered here are finite. The notation W_2 in the paper is the corresponding finite distance.

Using (15) in (18) yields the more informative, but not directly inverse-free, bound

$$W_2(\nu_y,\mu_B)^2 \le \|F(B)S^{-1}d\|^2 + \operatorname{tr}\big(F(B)S^{-1}F(B)^\top\big). \tag{19}$$

The source declaration `posterior_wasserstein2Sq_le_gain_residual` proves this inequality from the exhibited transport plan. The displayed right side is its exact cost. It is not asserted to equal the optimal squared Wasserstein distance.

Lemma 5.1 (Frobenius energy controls the residual cost)**.** *If $F \in \mathbb{R}^{n\times p}$ and $S \succeq \sigma I_p \succ 0$, then*

$$\|FS^{-1}d\|^2 + \operatorname{tr}(FS^{-1}F^\top) \le E(F)\left(\sigma^{-1} + \sigma^{-2}\|d\|^2\right). \tag{20}$$

Proof. For a vector v, rowwise Cauchy–Schwarz gives

$$\|Fv\|^2 = \sum_i \left(\sum_j F_{ij}v_j\right)^2 \le \sum_i \left(\sum_j F_{ij}^2\right)\|v\|^2 = E(F)\|v\|^2.$$

Apply this with $v = S^{-1}d$ and use (6) to obtain

$$\|FS^{-1}d\|^2 \le E(F)\sigma^{-2}\|d\|^2.$$

If f_i is the transpose of the ith row of F, then

$$\operatorname{tr}(FS^{-1}F^\top) = \sum_i f_i^\top S^{-1} f_i \le \sum_i \|f_i\|\|S^{-1}f_i\| \le \sigma^{-1}\sum_i \|f_i\|^2.$$

The last sum is $E(F)$. Adding the two estimates proves the result. □

The formal estimate `quadratic_residual_cost_le_energy` in `FDResidualEnergyCertificate.lean` is slightly more general: it accepts an auxiliary operator Q with a proved bound $\|Q\|_{\mathrm{op}} \le c$. Its application to $Q = S^{-1}$ uses the established noise-floor bound and gives the formula above.

Theorem 5.2 (Residual-energy posterior certificate)**.** *For the model in* (1)*, with $R \succeq \sigma I_p$ and $\sigma > 0$, every gain B satisfies*

$$W_2(\nu_y, \mu_B)^2 \le a(B), \qquad a(B) = E(F(B))\frac{\sigma + \|d\|^2}{\sigma^2}. \tag{21}$$

In particular, for $\varepsilon \ge 0$, the stopping condition

$$E(F(B))(\sigma + \|d\|^2) \le \varepsilon^2\sigma^2 \tag{22}$$

implies $W_2(\nu_y, \mu_B) \le \varepsilon$.

Proof. The common-noise transport plan gives (19). Lemma 5.1 bounds its cost by $a(B)$, establishing the first statement. Since $\sigma^2 > 0$, (22) is equivalent to $a(B) \le \varepsilon^2$. Taking the nonnegative square root gives the asserted distance bound. □

The raw declarations are `posterior_wasserstein2Sq_le_residual_energy` and `posterior_wasserstein2_le_of_residual_tolerance`. Their hypotheses include the covariance assumptions, positivity of the floor, and the floor's quadratic interpretation. In the factor-coded program, those hypotheses are discharged by the represented factors and the checked positive scalar. The user does not supply the conclusion $a(B) \ge W_2^2$ as an input promise.

5.1 What the certificate depends on

The score depends on the proposed gain through $F(B)$, on the observation through d, and on conditioning through σ. The same residual energy can lead to different certified errors for different observations. A large innovation can amplify a gain discrepancy in the posterior mean. A small noise floor makes a residual less informative about the gain error. These are visible features of the bound, rather than implementation effects hidden in a tolerance parameter.

The stopping test is sufficient and can be conservative. A score above ε^2 does not prove that the true distance exceeds ε. The coupling may have a nonoptimal cost, the inverse norm bound may be loose, and later interval evaluation may further enlarge an upper bound. Conversely, a passing verified score does prove the tolerance. The solver's convergence theorem concerns this explicit sufficient test, so its correctness does not depend on predicting how tight it will be on a particular application.

If $F(B) = 0$, invertibility of S gives $B = K$, and the approximation is exactly the posterior version. The source proves this as `gainPosteriorLaw_eq_of_residual_zero`. No inference from approximately zero floating point entries is used in that theorem. For a nonzero tolerance the theorem uses the whole squared energy, allowing a candidate to pass before the gain equation is solved exactly.

5.2 The two output tolerances have different meanings

The completed solver uses a posterior-law tolerance $\varepsilon > 0$ and a parameter-representation radius tolerance $\delta > 0$. The first bounds $W_2(\nu_y, \mu_B)$. The second bounds the radii of outward-rounded intervals containing entries of m_B and C_B. The latter arrays are descriptions of the approximation already certified by the former theorem.

An interval enclosing an entry of C_B is not, by itself, an interval enclosing the corresponding entry of C_*. Nor does a radius at most δ certify a Gaussian formed by treating all interval centers as exact parameters. The represented gain and its Joseph law remain the semantic object. Keeping that object fixed is what permits the solver, the final certificate, and the output enclosure proof to compose without changing the meaning of the returned distribution midway through the computation.

6 Packets resolve a complete residual energy

A packet family in this release is a finite collection of self-adjoint, pairwise disjoint idempotent matrices. Let $\{Q_a : a \in \mathcal{A}\}$ act on the state space and $\{D_b : b \in \mathcal{B}\}$ act on the observation space. For each family, the structural conditions are

$$Q_a^\top = Q_a, \qquad Q_a^2 = Q_a, \qquad Q_a Q_{a'} = 0 \quad (a \neq a'), \tag{23}$$

$$D_b^\top = D_b, \qquad D_b^2 = D_b, \qquad D_b D_{b'} = 0 \quad (b \neq b'). \tag{24}$$

Completeness is a separate condition:

$$\sum_{a \in \mathcal{A}} Q_a = I_n, \qquad \sum_{b \in \mathcal{B}} D_b = I_p. \tag{25}$$

The self-adjoint condition makes these Euclidean orthogonal projectors. Idempotence alone would not give the required energy geometry. Completeness prevents a certificate from forgetting a residual direction outside the selected packet spans.

The two-sided residual packet is

$$F_{ab}(B) = Q_a F(B) D_b. \tag{26}$$

Each packet remains an $n \times p$ matrix, although it is supported on the output and input subspaces selected by its projectors. The notation does not assume a coordinate partition or require projectors to be diagonal. Rational rotated projectors are allowed when their exact entries satisfy the declared identities.

6.1 Orthogonality gives exact accounting

Use the Frobenius inner product $\langle A, C \rangle_{\mathrm{F}} = \operatorname{tr}(A^\top C)$. For $a \neq a'$, self-adjointness and disjointness give

$$\langle Q_a A, Q_{a'} A \rangle_{\mathrm{F}} = \operatorname{tr}(A^\top Q_a Q_{a'} A) = 0.$$

Consequently the energy of a finite sum of output packets is the sum of their energies. The input analogue follows by transposing the matrices. Applying the two decompositions successively gives complete two-sided accounting.

Theorem 6.1 (Complete packet decomposition)**.** *If both families satisfy* (23)–(25)*, then every* $A \in \mathbb{R}^{n \times p}$ *satisfies*

$$E(A) = \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} E(Q_a A D_b). \tag{27}$$

More generally, for subsets $U \subseteq \mathcal{A}$ *and* $V \subseteq \mathcal{B}$*,*

$$E\left(\left(\sum_{a \in U} Q_a\right) A \left(\sum_{b \in V} D_b\right)\right) = \sum_{a \in U} \sum_{b \in V} E(Q_a A D_b). \tag{28}$$

Proof. The sum of any subfamily of the Q_a is again self-adjoint and idempotent: expansion of its square removes the off-diagonal products, and the remaining diagonal products equal the original projectors. For a self-adjoint idempotent Q,

$$E(QA) = \operatorname{tr}((QA)^\top QA) = \operatorname{tr}(A^\top QA).$$

The trace is linear, so replacing Q by a sum over U proves the output decomposition. Transposition preserves energy and turns right multiplication by the sum over V into left multiplication, proving the input decomposition. Combining these proves (28). Setting $U = \mathcal{A}$ and $V = \mathcal{B}$, completeness reduces the matrix inside its energy to A, proving (27). □

The source formalizes the packet structure in `SelfAdjointPackets.lean`. Its exact energy declarations are `matrixEnergy_sumOn_left`, `matrixEnergy_sumOn_right`, `matrixEnergy_sumOn_both`, and `matrixEnergy_eq_sum_blocks` in `PacketEnergyDecomposition.lean`. The local parent identity does not require global completeness. The final whole-residual identity does. This separation allows local algebra to be used without silently granting full-space coverage.

Equation (28) describes aggregation within a fixed family. It does not by itself certify an algorithm that changes its hierarchy while running. The completed solver fixes its packet families before iteration. Dynamic hierarchy changes belong to a separate extension and require their own state and convergence argument.

6.2 Majorants compose with the posterior certificate

Suppose a producer supplies values u_{ab} and v with proved inequalities

$$E(F_{ab}(B)) \le u_{ab}, \qquad \|d\|^2 \le v. \tag{29}$$

Completeness and nonnegative energy imply

$$E(F(B)) \le U, \qquad U = \sum_{a,b} u_{ab}.$$

Each u_{ab} is nonnegative because it bounds a nonnegative quantity. Likewise $v \ge 0$. The score

$$\widehat{a}(B) = U\frac{\sigma + v}{\sigma^2} \tag{30}$$

therefore bounds $a(B)$ and hence the squared posterior distance. If

$$U(\sigma + v) \le \varepsilon^2\sigma^2, \qquad \varepsilon \ge 0, \tag{31}$$

the same law tolerance follows.

Corollary 6.2 (Packet-majorant tolerance)**.** *Under the model and complete packet hypotheses,* (29) *and* (31) *imply* $W_2(\nu_y, \mu_B) \le \varepsilon$.

Proof. Sum the packet inequalities and apply Theorem 6.1 to obtain $E(F(B)) \le U$. Since $\sigma + \|d\|^2 \ge 0$, and $U \ge 0$,

$$E(F(B))(\sigma + \|d\|^2) \le U(\sigma + \|d\|^2) \le U(\sigma + v) \le \varepsilon^2\sigma^2.$$

Theorem 5.2 now applies. □

The source proves the residual bound from local majorants as `frobeniusSq_le_packet_majorants` and the tolerance result as `posterior_wasserstein2_le_of_packet_majorants` in `FDResidualPacketCertificate.lean`. The innovation-majorant combination is used by the rational checker through `posterior_wasserstein2Sq_le_energy_majorants` in `PosteriorEnclosureBounds.lean`.

Majorants allow different arithmetic representations to feed the same semantic theorem. An exact rational evaluator can choose $u_{ab} = E(F_{ab}(B))$. An outward-rounded evaluator can instead supply a certified upper endpoint. The law proof needs the inequality, not an assumption that every local value was calculated with unlimited precision. The arithmetic chapters explain how the final dyadic evaluator proves those inequalities for its own operations.

6.3 Why partial packet accounting is insufficient for stopping

For a fixed residual, a sum over only some packets is no larger than the full energy. The declaration `sum_block_energies_le` proves this direction. It is a useful local diagnostic, but it cannot replace the whole-energy upper bound in a stopping test. A candidate can have zero residual in the inspected subspaces while retaining a large residual elsewhere.

For example, take the coordinate projectors onto the two axes of $\mathbb{R}^2$. A residual matrix with a nonzero second row has zero energy in the first output packet. Certifying that first packet alone says nothing about the omitted row. Completeness and a bound for every packet are what change local information into a global posterior guarantee. The exact checker tests those structural and local obligations; it does not accept a producer's assertion that a chosen subset happened to be representative.

The canonical selective loop uses the output-packet energies $E(Q_a F(B))$ to choose which gain component to update. Its final residual certificate may use both output and input packets. These roles are compatible but different: output packets guide the iterative correction, while complete two-sided packets organize the final verification. Neither role makes the final distribution error a collection of independent per-packet posterior distances.

7 From finite rational checks to a real-law guarantee

The rational checker turns the previous inequalities into an inspectable decision boundary. A candidate contains a rational gain B, rational packet bounds u_{ab}, and a rational innovation bound v. In the source these are the fields `gain`, `packetUpper`, and `innovationUpper` of `RationalCandidate`. Its validity proposition requires five groups of conditions: a positive noise floor, valid complete output packets, valid complete input packets, every local residual-energy inequality, and the innovation-energy inequality.

The packet check is exact. It checks symmetry, idempotence, pairwise disjointness, and the sum-to-identity condition by finite rational equalities. These identities are not inferred from near-equality of floating point arrays. Exact inclusion into $\mathbb{R}$ preserves matrix products, transposes, finite sums, and order. A rational family that passes therefore constructs a real self-adjoint complete packet family with the geometry required in Section 6.

The energy checks are also exact finite sums. The real interpretation of rational matrix energy is the squared real Frobenius energy of the interpreted matrix. The declaration `rationalEnergy_cast` proves this fact, and `rational_packet_majorant` transfers a rational local upper bound into the real packet inequality. Likewise `rationalVectorEnergy_cast` identifies the rational innovation energy with the squared Euclidean norm used in the probability theorem.

7.1 Validity and tolerance are separate decisions

The first Boolean function is

$$\texttt{checkCandidate}(M, Q, D, c) = \texttt{decide}(\texttt{ValidCandidate}(M, Q, D, c)).$$

It recomputes the finite obligations for the explicit model, packets, and candidate. A claimed successful producer run is not one of its inputs. The equivalence `checkCandidate_eq_true` states that its true result is exactly candidate validity.

The second Boolean function checks $\varepsilon \geq 0$ and

$$\left(\sum_{a,b} u_{ab}\right)(\sigma + v) \leq \varepsilon^2 \sigma^2.$$

This function is `checkTolerance`. The displayed cross-multiplied form avoids evaluating square roots and does not require division in the tolerance decision. Positivity of σ, obtained through candidate validity, justifies interpreting this test as a score bound. A tolerance check alone is not a substitute for the structural and local validity checks.

Theorem 7.1 (Sound exact candidate checking)**.** *Let M be a factor-coded rational model, Q, D rational packet families, and c a rational candidate. If both* `checkCandidateMQDc` *and* `checkToleranceMcepsilon` *return true, then the exact real posterior version represented by M and the exact real Joseph Gaussian represented by the gain of c satisfy*

$$W_2(M.\text{posterior}, M.\text{approximation}(c.\text{gain})) \leq \varepsilon. \tag{32}$$

Here ε is the real interpretation of the rational tolerance.

Proof. The first check gives positivity of the floor and the four finite packet identities for each family. The factors prove the covariance and noise-floor hypotheses. The rational-to-real interpretation lemmas turn every checked local inequality into a bound on the corresponding real packet energy, and turn the innovation inequality into a bound on $\|d\|^2$. Thus the real majorant score in (30) equals the interpretation of the rational candidate score. Corollary 6.2, applied to the second check, gives the law tolerance. The model semantics identifies the two laws in its conclusion with those in (32). □

The exact raw theorem is `checked_tolerance_bounds_posterior` in `RationalPosteriorCertificate.lean`. Its squared precursor, `checked_score_bounds_posterior`, derives a bound by the interpreted rational candidate score. The route passes through `valid_residual_energy_bound` and `valid_innovation_energy_bound`, which explicitly connect the accepted finite data to the analytic quantities. These connections are why the checker certifies a distribution law rather than merely certifying internal consistency of a table of numbers.

7.2 The certificate is independent of proposal quality

A candidate proposal can come from an exact iterative solver, an external optimizer, or a finite list of speculative alternatives. The general candidate theorem requires a passing checker, rather than a theorem about that proposal process. The source's validity-preserving finite acceptance policy keeps a checked incumbent and replaces it only when its finite tests permit the replacement. The declarations `selected_batch_bounds_posterior` and `selected_batch_preserves_posterior_tolerance` show that a finite batch of untrusted proposals preserves the relevant law bound under that policy.

This does not certify the parser or arbitrary code that prepares the explicit data. An exported Lean instance certifies the rational model, packets, gain, and arrays written into that instance. Connecting an external data file to those constants is an additional implementation relation. The first release supplies useful Python producers and exports, but it does not claim a universal refinement of that parser and implementation to the Lean program.

The canonical cached solver is stronger in a different direction. It constructs its own exact cache from the model, produces actual greedy iterates, and proves finite success for positive tolerances under its input hypotheses. The general checker theorem remains useful because the final verification is independent of

any internal claim that the iteration has converged. The subsequent chapters establish how that canonical execution reaches the certificate boundary without assuming candidate success in advance.

8 A singular-prior calculation through the full semantic chain

Consider the rational model

$$P = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad H = \begin{pmatrix} 1 & 1 \end{pmatrix}, \qquad R = 1, \qquad m = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad y = 1.$$

It is factor-coded by $L = (1,0)^\top$, a zero noise factor, and $\sigma = 1$. The second state coordinate is deterministic under the prior. Nevertheless, the observation operator includes that coordinate, and all model formulas remain well typed. The innovation covariance is $S = 2$, the innovation is $d = 1$, and

$$K = \begin{pmatrix} 1/2 \\ 0 \end{pmatrix}, \qquad C_* = \begin{pmatrix} 1/2 & 0 \\ 0 & 0 \end{pmatrix}.$$

Thus the explicit posterior is concentrated on the first coordinate axis with mean $(1/2,0)^\top$.

Choose $B = (1/3,0)^\top$. The approximate mean and Joseph covariance are

$$m_B = \begin{pmatrix} 1/3 \\ 0 \end{pmatrix}, \qquad C_B = \begin{pmatrix} 5/9 & 0 \\ 0 & 0 \end{pmatrix}.$$

The residual is $F(B) = (1/3,0)^\top$, with energy $1/9$. Its exact common-noise coupling cost is

$$\|F(B)S^{-1}d\|^2 + \operatorname{tr}(F(B)S^{-1}F(B)^\top) = \frac{1}{36} + \frac{1}{18} = \frac{1}{12}.$$

The inverse-free energy score is larger:

$$a(B) = \frac{1}{9}\frac{1+1}{1^2} = \frac{2}{9}.$$

Both are valid upper bounds on the squared law distance. The difference illustrates the cost of replacing the exact inverse expression by an explicit noise-floor majorant. No inverse prior or inverse posterior covariance was used at any stage.

For coordinate output packets and the single identity input packet, the local residual energies are $1/9$ and 0. Their complete sum is exactly the full energy. If the rational tolerance is $\varepsilon = 1/2$, the squared tolerance is $1/4$, and $2/9 \le 1/4$. The exact certificate therefore proves the posterior-law tolerance for this candidate. This is a worked arithmetic specialization of the general theorems, not a claim that a fresh Lean instance of these particular constants has been compiled in this publication run.

The example also separates three objects that can otherwise be confused: the target conditional Gaussian ν_y, the gain-defined Joseph approximation μ_B, and any finite interval arrays used to describe the latter. The posterior theorem relates the first two probability laws. The output theorem encloses parameters of the second. Their composition provides a usable represented approximation whose law error and representation error are stated separately.

9 An exact selective iteration with a preserved cache

The posterior certificate becomes an algorithm when it is paired with a gain construction that is guaranteed to reach a passing certificate. The canonical construction in Core 1.0.0 performs exact rational selected-packet updates. It retains enough residual information to choose the next update without rebuilding the complete residual after every step. Its final certificate and parameter representation use outward-rounded dyadic arithmetic. These are separate stages with separate correctness arguments.

Throughout this section, the state dimension and observation dimension are finite. The supplied rational model has

$$P = LL^{\mathsf{T}}, \qquad R = NN^{\mathsf{T}} + \sigma I, \qquad \sigma > 0,$$

and observation operator H. Put

$$S = HPH^{\mathsf{T}} + R, \quad v = y - Hm, \quad F(B) = PH^{\mathsf{T}} - BS, \quad E(X) = \|X\|_{\mathrm{F}}^2.$$

The output packet family $(Q_a)_{a\in A}$ consists of rational self-adjoint, mutually orthogonal projectors with sum I. The input family $(D_b)_{b\in D}$ has the same properties on the observation space. The output family controls selection and cache maintenance. The input family participates in the independent final certificate. No commutation relation between S and the input projectors is assumed.

Preprocessing forms three model-dependent constants:

$$\tau = \frac{1}{\sigma + \operatorname{tr} S}, \qquad T = I - \tau S, \qquad w = \frac{\sigma + \|v\|^2}{\sigma^2}. \tag{33}$$

All are rational. They are the fields of the record `Komorebi.CachedPosterior.Kernel`, produced by `Komorebi.CachedPosterior.prepare`. The correctness theorems use this actual preparation, rather than an arbitrary record whose fields happen to have the same names. A wrong transition matrix or weight would invalidate the correspondence between the stored score and the posterior certificate.

Definition 9.1 (Cache validity)**.** A cache state is a tuple $c = (B, (G_a), (e_a), t)$ satisfying

$$G_a = Q_a F(B), \qquad e_a = E(G_a), \qquad t = \sum_{a\in A} e_a. \tag{34}$$

The cached score is $s(c) = tw$.

Validity is an exact equality invariant. It does not mean that a stored residual is approximately near a true residual, nor that its energy is merely an upper bound. Initialization evaluates $F(B_0)$ once, projects it onto every output packet, computes every energy, and sums them. The theorem `initialState_valid` follows directly from the definitions. It requires no prediction about future iterates.

Orthogonal complete packets account for the whole energy:

$$\sum_{a\in A} E(Q_a F) = E(F).$$

Consequently a valid cache has $t = E(F(B))$ and

$$s(c) = E(F(B))\frac{\sigma + \|v\|^2}{\sigma^2} = a(B). \tag{35}$$

The right-hand side is the exact residual score used by the posterior error certificate. The declarations `total_eq_residual_energy` and `score_eq` establish these two identifications. The total is therefore a meaningful global stopping signal even though only one of its summands changes in an iteration.

For a selected label a, the executed update is

$$\begin{aligned} B' &= B + \tau G_a, & G'_a &= G_a T, & e'_a &= E(G'_a), \\ G'_b &= G_b \quad (b \neq a), & e'_b &= e_b \quad (b \neq a), & t' &= t - e_a + e'_a. \end{aligned} \tag{36}$$

This is the definition `Komorebi.CachedPosterior.step`. It computes the new selected residual from the old selected residual and the precomputed transition. It retains unselected entries and updates the scalar total by a difference. The implementation does not define its new state by recomputing $F(B')$ and projecting that matrix onto every packet.

Proposition 9.2 (Preservation by the executed update)**.** *If the output packets satisfy the stated projector assumptions, c is valid, and the kernel is* `prepared`*, then the state in* (36) *is valid.*

Proof. By cache validity, $G_a = Q_a F(B)$, so the gain update is the exact packet step $B' = B + \tau Q_a F(B)$. Expand its residual:

$$F(B') = F(B) - \tau Q_a F(B) S.$$

Multiplication on the left by the selected projector gives

$$Q_a F(B') = Q_a F(B) - \tau Q_a^2 F(B) S = G_a(I - \tau S) = G_a T.$$

For $b \neq a$, mutual orthogonality gives $Q_b Q_a = 0$ and hence $Q_b F(B') = Q_b F(B) = G_b$. This proves every residual equality in the new cache. Its energy equalities hold because the selected energy is evaluated from its new residual and the remaining residual-energy pairs are retained. Finally, replacing one term in a finite sum changes that sum by exactly $-e_a + e'_a$. Thus $t' = \sum_b e'_b$. All three clauses of validity are preserved. □

This proof is realized by `step_valid`, using the algebraic identities `selected_residual` and `unselected_residual`. The declarations `step_other_residual` and `step_other_energy` expose the unchanged entries independently. The distinction between proving an invariant for an executed update and postulating it for a desired state is decisive: the later termination theorem uses the invariant at every state visited by the program.

The locality is on the output side. For an input projector D_b,

$$F(B')D_b = F(B)D_b - \tau Q_a F(B) S D_b.$$

The factor SD_b can mix input directions. The source declaration `input_packet_effect` makes this effect explicit. Complete input packets can still account for certificate energy, but an input packet decomposition does not by itself make the iteration independently local in both indices.

10 Selection, exact trajectory equality, and finite fuel

The algorithm chooses a largest stored output energy. Its finite scan has a seed label and an ordered list. It replaces the incumbent label only when a scanned label has strictly greater energy. On an equality it preserves the incumbent. This tie rule appears in the definition `winner`, whose comparison is `ea<eb`. The scan may contain repeated labels. Completeness requires that every output label occurs in the scan, while the seed supplies an inhabitant of the label type. The proof counts the cardinality of the finite label type, not the length of a possibly repeated scan.

`winner_ge_initial` and `winner_ge_member` prove that the final winner has energy at least that of the seed and of every scanned label. Under complete coverage, `selectPacket_max` therefore supplies a genuine maximum over all output packet energies. The argument does not prescribe a fixed schedule. The maximum is selected again from the current state after every update.

Proposition 10.1 (Exact selection and gain equality)**.** *For a valid cache, cached selection returns the same label as a scan of the recomputed energies $E(Q_a F(B))$. Starting from an initialized cache and using the prepared kernel, its gain after n updates equals the recomputed greedy gain after n updates, for every $n \in \mathbb{N}$.*

Proof. Each stored energy equals the corresponding recomputed energy by (34). Both scans therefore receive the same energy function, the same seed, the same list, and the same strict comparison. Their outputs are equal, including ties. This is `chosen_eq`. The cache gain update then equals the recomputed greedy step by `advance_gain`.

The initial gains agree at $n = 0$. Suppose they agree at n. The cache at that index is valid by repeated application of `step_valid`. Equality of selection and equality of the gain step give equality at $n + 1$. Induction proves the entire trajectory identity. The formal statements are `trajectory_valid` and `trajectory_gain`. □

This is stronger than convergence to the same limit. At a fixed index, the two rational gain matrices are identical. As a result, the convergence proof for the recomputed greedy algorithm transfers directly to the cached execution.

Write $A_{\mathrm{count}} = |A|$ for the positive number of output labels, and set

$$\rho = 1 - \frac{\sigma}{\sigma + \operatorname{tr} S}, \qquad q = \rho^2, \qquad r = 1 - \frac{1 - q}{A_{\mathrm{count}}}. \tag{37}$$

The model factorization and positive noise floor imply that S is positive semidefinite with quadratic lower bound σ. Its trace is nonnegative, and $\sigma + \operatorname{tr} S$ is a valid positive spectral cap. Thus $0 \le \rho < 1$, $0 \le q < 1$, and $0 \le r < 1$. No eigenvalue oracle is needed to construct these rational constants.

The transition contracts every residual matrix:

$$E(XT) \le qE(X).$$

If label a is updated, its residual energy contracts by this inequality and all other output residuals remain unchanged. Orthogonal energy accounting gives

$$E(F(B')) \le E(F(B)) - (1 - q)e_a.$$

Because a is a maximum, $e_a \ge E(F(B))/A_{\mathrm{count}}$. It follows that

$$E(F(B')) \le rE(F(B)), \qquad a(B_n) \le r^n a(B_0). \tag{38}$$

The energy share is `selected_energy_share`; one-step contraction is `greedy_energy_contract`; and the score estimate is `greedy_score_geometric`. The estimate allows singular P, arbitrary initial rational B_0, rectangular H, and rational rotated packets under the same projector assumptions.

A convergent sequence alone would not define a terminating program with an explicit natural-number budget. The source supplies a logarithm-free fuel:

$$N = \left\lceil \frac{a(B_0)}{(\varepsilon/2)^2(1 - r)} \right\rceil_{\mathbb{N}}, \qquad \varepsilon > 0. \tag{39}$$

Here $\lceil\cdot\rceil_{\mathbb{N}}$ is the natural-number ceiling used by `toleranceFuel`. The canonical `adaptiveFuel` evaluates this expression from the actual model, initial gain, tolerance, and number of labels.

Lemma 10.2 (Rational fuel sufficiency)**.** *For $0 \le r < 1$ and $n \ge N$, the geometric envelope obeys $r^n a(B_0) \le \varepsilon^2/4$.*

Proof. The source first proves

$$r^n\big(1 + n(1-r)\big) \le 1.$$

At the induction step, the inequality $r(1 + (n+1)(1-r)) \le 1 + n(1-r)$ reduces to nonnegativity of $(n+1)(1-r)^2$. The natural ceiling in (39) gives $a(B_0) \le n(\varepsilon/2)^2(1-r)$. Multiplying by the nonnegative r^n and applying the preceding bound yields the required squared tolerance. The proof remains valid when the contraction factor is zero. The source statements are `geometric_bernoulli`, `geometric_fuel_sufficient`, and `greedy_score_at_fuel`. □

This fuel is conservative. The geometric rate is substantially sharper than the reciprocal envelope used to obtain a simple rational fuel, and the executed search can stop much earlier. Neither the large natural fuel nor its existence is a claim about a practical time or memory budget.

`search` tests the exact cached score against $\varepsilon^2/4$ at the current state. If the test fails and fuel remains, it advances once and decrements fuel. At fuel zero it still tests the last permitted state. This convention permits success after exactly N updates. `search_score` proves that every returned state passes the score condition. `search_origin` proves that a returned state is an actual trajectory state at some $k \le N$. Finally, `search_complete` proves that a passing score at the last permitted state forces the search to return some state, possibly earlier.

Combining preserved validity, exact trajectory equality, and fuel sufficiency therefore proves a finite successful cached search. There is no caller-supplied claim that an iteration has converged. The computed budget and proved invariant supply the necessary conclusion internally.

11 Independent final verification and precision fallback

The exact cached score is used to select a gain and establish termination. The successful gain is still sent to the independent certificate evaluator. That evaluator reconstructs rounded packet residual obligations from the actual model and gain. It does not accept the stored cache energies as certificate evidence. This separation protects successful returns from an incorrect internal stopping signal. The canonical route checks setup, constructs the outward-rounded candidate, and executes its tolerance test. Construction and setup imply that `checkCandidate` is true by theorem. A separate untrusted-candidate checker can recompute exact residual obligations; that is distinct from the canonical rounded construction route.

The definition `solveCached` first checks a positive posterior tolerance, valid model and packet setup, and complete scan coverage. It prepares the kernel, initializes the cache, and runs the exact-score search with its computed fuel. Only the returned gain is passed to `certifyGain`. Thus certificate evaluation is outside the selective update loop. The definition makes one call to this final verification procedure, although that procedure may try several precisions.

The precision schedule consists of the caller's finite list of probes followed by one computed fallback. The probes can be empty, repeated, or unordered. Successful verification records the precision actually used. The appended fallback makes finite success independent of a lucky externally chosen probe.

Let $h_p = 2^{-p}$. The dyadic evaluator propagates rational centers and outward radii through its arithmetic graph. Its budget calculation supplies a nonnegative coefficient $C_{\mathrm{ver}}(M, Q, D, B)$ independent of p, with

$$\widehat{a}_p(B) \le a(B) + C_{\mathrm{ver}}(M, Q, D, B)h_p. \tag{40}$$

Here $\widehat{a}_p$ is the actual rounded candidate score. The full expression for the coefficient is `scoreRoundCoeff`, built from packet residual energy and innovation-vector magnitude and radius budgets. It bounds the entire score graph, rather than rounding only its final exact answer. The theorem `rounded_score_inflation` proves (40).

The common precision selector is the computable integer expression

$$\mathrm{precisionFor}(C, d) = \log_2^{\mathbb{N}}\left(\lceil C/d \rceil_{\mathbb{N}}\right) + 1, \qquad d > 0, \tag{41}$$

where $\log_2^{\mathbb{N}}$ denotes the natural-number logarithm `Nat.log`, including its specified value at zero. It is not a floating-point logarithm. For every precision at least this selected count, `coefficient_mesh_le` gives $C2^{-p} \le d$. The proof uses integer power and ceiling inequalities. The additional bit is a sufficient choice, not a claim that the bit count is minimal.

For final verification, the solver appends

$$p_* = \mathrm{sufficientPrecision}(M, Q, D, B, \varepsilon^2/4).$$

At its returned search state, $a(B) \le \varepsilon^2/4$. The fallback guarantees inflation at most another $\varepsilon^2/4$, so

$$\widehat{a}_{p_*}(B) \le \varepsilon^2/2 \le \varepsilon^2.$$

Positive ε gives a positive inflation allowance. Valid model and packet setup give the candidate check, while this inequality gives the tolerance check. The declarations `certifyRounded_of_slack` and `certifyGain_complete` establish that the fallback must succeed.

The stricter search threshold supplies room for rounded verification. Searching only until an exact score happened to equal ε^2 would leave no positive allowance for an enclosing evaluator. The construction deliberately separates progress from verification slack. Its inherited geometric fuel is computed for $\varepsilon/2$, which is exactly the square-root tolerance corresponding to $\varepsilon^2/4$.

Theorem 11.1 (Successful and complete cached gain construction)**.** *For rational factor-coded model data with $\sigma > 0$, valid complete rational output and input packets, a seed and complete output scan, any positive rational ε, any finite precision probe list, and any rational initial gain, `solveCached` returns a candidate and verification precision. The candidate and tolerance checks pass, and its gain is an actual greedy iterate at an index within the computed fuel. Its Joseph Gaussian has Wasserstein distance at most ε from the explicit posterior version established by the Bayesian semantics.*

The source assembles these conclusions through `solveCached_terminates`, `cached_solver_gain_origin`, and `cached_solver_reaches_posterior_tolerance`. The semantic posterior used here is the proved explicit Gaussian kernel version. Its agreement with Bayesian conditioning is established through disintegration and almost-everywhere agreement with canonical conditional versions, as discussed in the foundations. The theorem at the supplied rational observation does not assert uniqueness of every possible pointwise conditional version at an event of probability zero.

The independent statement `solveCached_sound` is logically different from termination. It says that any successful program return satisfies both Boolean checks without requiring the caller to supply setup-validity proofs. A returned candidate has passed the evaluator's checks. Valid input hypotheses are needed to guarantee that a return exists; they are not external assumptions attached to the meaning of an already successful return.

12 Parameter arrays and the meaning of a complete result

A useful posterior output contains more than a passing gain. For the returned rational gain B, the parameter stage represents

$$\mu_B = m + B(y - Hm),$$
$$V_B = (I - BH)P(I - BH)^{\mathsf{T}} + BRB^{\mathsf{T}}. \tag{42}$$

These are the mean and Joseph covariance of the same Gaussian law certified by the Wasserstein statement. The covariance is positive semidefinite under the model hypotheses even when the prior is singular. The declaration `approximation_eq_output_gaussian` identifies this parameterized Gaussian with the approximation law used in the posterior theorem.

The evaluator returns a ball for each mean coordinate and each covariance entry. For exact rational entry x, a ball (c, r) means $|x - c| \leq r$. Its radius is an arithmetic enclosure error for that parameter of the returned approximation. The output tolerance $\delta > 0$ bounds every such radius. It is independent of the posterior Wasserstein tolerance ε.

Each output arithmetic expression has a precision-independent budget coefficient. The source sums all mean coefficients and all covariance coefficients to obtain C_{out}, defined by `outputCoefficient`. Every individual coefficient is nonnegative and no greater than this sum. It chooses

$$p_{\text{out}} = \text{precisionFor}(C_{\text{out}}, \delta)$$

and evaluates the output graph at that precision. Since every entry radius is bounded by its coefficient times $2^{-p_{\text{out}}}$, the same integer precision bounds all radii by δ. The declarations `mean_coeff_le`, `covariance_coeff_le`, and `evaluate_radii_le` establish this argument. `represent_spec` joins entrywise enclosure and the requested radius bound.

The result record stores verification precision, the checked candidate, parameter precision, and the evaluated arrays. Its contract `ResultValid` contains seven clauses: the candidate check passes, the tolerance check passes, the Wasserstein bound holds, the arrays enclose the returned approximation's parameters, all radii are at most δ, the reported parameter precision equals the actual precision selector, and the arrays equal the actual evaluator at that precision. The final two clauses make the reported representation part of the contract, rather than allowing arbitrary arrays that happen to satisfy a looser description.

Theorem 12.1 (Complete first-release output)**.** *Under the valid-input hypotheses above and $\delta > 0$, `Komorebi.CachedPosterior.runCached` returns a result satisfying `ResultValid`. Every successful return is sound even without setup-validity proofs supplied by the caller. A nonpositive output-radius tolerance is rejected.*

These are precisely `runCached_terminates`, `runCached_sound`, and `runCached_rejects_nonpositive_output_tolerance`. The complete contract is finite-dimensional inference for one observation update. It includes singular priors, rectangular observation operators, gain origin, computable iteration fuel, verification precision, and parameter precision.

Two interpretations must remain separate. The intervals enclose the parameters of the certified Joseph Gaussian. They do not certify another Gaussian obtained by treating interval centers as exact mean and covariance. In particular, rounded covariance centers are not independently proved positive semidefinite. Likewise, δ is not a coordinate error bound against the exact posterior. The Wasserstein guarantee compares laws; the interval guarantee encloses the parameters of one of those laws. A theorem combining either guarantee with a new operation must account for that operation explicitly.

The program is mathematically finite for every positive rational pair (ε, δ) under its declared assumptions. A reference implementation can still encounter a time, memory, or user-supplied resource cap. The Python tools report `RESOURCE_LIMIT` when their caps are reached. Such a cap is a limit of the execution environment, not a counterexample to the finite-success theorem. The result contract also contains no sampler and no guarantee for an external floating-point implementation.

13 A complete coordinate execution

The coordinate fixture has a two-dimensional state and one-dimensional observation:

$$P = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}, \quad H = \begin{pmatrix} 1 & 1 \end{pmatrix}, \quad R = 1, \quad m = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad y = 1.$$

It is represented by $L = \operatorname{diag}(2, 1)$, $N = 0$, and $\sigma = 1$. The two output projectors select the coordinate axes, and the single input projector is 1. Start with $B_0 = 0$, seed 0, scan $[0, 1]$, probes $[4, 12]$, $\varepsilon = 1/20$, and $\delta = 1/1000$.

Here $S = 6$, $\tau = 1/7$, $T = 1/7$, and $w = 2$. The initial residual is $(4, 1)^{\mathsf{T}}$, with packet energies 16 and 1. Updating one coordinate multiplies its residual by $1/7$ and its energy by $1/49$. The other coordinate remains exactly unchanged. The maximum-energy schedule therefore alternates between labels 0 and 1. The complete pre-update trace is

Updates already made	Selected label	e_0	e_1	t
0	0	16	1	17
1	1	16/49	1	65/49
2	0	16/49	1/49	17/49
3	1	16/2401	1/49	65/2401
4	0	16/2401	1/2401	17/2401
5	1	16/117649	1/2401	65/117649

After six updates, each coordinate has been selected three times. The geometric sum in its gain gives

$$B_6 = \begin{pmatrix} 228/343 \\ 57/343 \end{pmatrix}, \qquad F(B_6) = \begin{pmatrix} 4/343 \\ 1/343 \end{pmatrix}, \qquad a(B_6) = \frac{34}{117649}.$$

The strict search threshold is $\varepsilon^2/4 = 1/1600$. At the previous state, $a(B_5) = 130/117649 > 1/1600$, while $a(B_6) < 1/1600$. Thus six is the actual update count for this stopping rule. The formal fuel is 410093; the program uses only six of these allowed updates. This large difference illustrates the conservative fuel construction, not an ambiguity in the observed trace.

At verification precision 4, the rounded score is $25/16$ and the attempt fails. At precision 12, it is $481/1048576$, which is below the required final threshold $\varepsilon^2 = 1/400$. The solver therefore returns verification precision 12. A failed coarse enclosure is a failed certificate attempt, not evidence that the gain itself is inaccurate. Sufficiently high precision bounds the enclosure inflation.

The exact posterior mean is $(2/3, 1/6)^{\mathsf{T}}$, while the returned Joseph mean is the gain B_6 because $m = 0$ and $y = 1$. Parameter output uses precision 32. Its mean centers are

$$\begin{pmatrix} 2854963683/4294967296 \\ 89217615/536870912 \end{pmatrix}$$

with radius $1/4294967296$ in each coordinate. The covariance centers are

$$\begin{pmatrix} 5726720411/4294967296 & -1431643597/2147483648 \\ -1431643597/2147483648 & 3579145495/4294967296 \end{pmatrix},$$

and the corresponding radii are

$$\begin{pmatrix} 1/536870912 & 7/4294967296 \\ 7/4294967296 & 1/536870912 \end{pmatrix}.$$

Every radius is smaller than $1/1000$. These fractions describe output arithmetic, not uncertainty intervals for model estimation. They enclose exactly the Joseph mean and covariance at B_6.

The exact rational matrix enclosed by those covariance balls is

$$V_{B_6} = \frac{1}{117649}\begin{pmatrix} 156868 & -78432 \\ -78432 & 98041 \end{pmatrix}.$$

It can be checked directly by substituting B_6 into (42). Another useful check starts from the exact posterior covariance $V_* = \left(\begin{smallmatrix} 4/3 & -2/3 \\ -2/3 & 5/6 \end{smallmatrix}\right)$. If $B_* = (2/3, 1/6)^\mathsf{T}$ is the exact gain, expansion of the Joseph expression gives

$$V_{B_6} = V_* + 6(B_6 - B_*)(B_6 - B_*)^\mathsf{T}.$$

The gain discrepancy is $B_* - B_6 = (2/1029, 1/2058)^\mathsf{T}$, which produces the displayed matrix. This second calculation checks the sign of the off-diagonal entries and explains why the approximate covariance has an additional positive semidefinite term. The posterior accuracy certificate concerns the complete mean-covariance law, while the dyadic output stage faithfully represents these rational parameters. The larger output budget can still choose many more bits than the smallest possible entrywise enclosure would require. Its purpose is a guaranteed common precision, not an optimal encoding of this particular covariance.

The module `GeneratedCachedCoordinate.lean` defines this actual model, initial cache, six-step cache trajectory, final gain, certificate, and result. Its declarations check the computed fuel, final gain, exact and rounded scores, each of the six chosen labels, solver return, parameter precision, every displayed center and radius, and the full output return. Its theorem `output_valid` applies `runCached_sound` to the proved return. This is a checked finite program example, while the universal termination theorem covers all valid inputs in the declared class.

An earlier recomputed selective solver verified after each update and could stop at five updates with a passing rounded certificate. That is a different stopping policy. The canonical cached solver first waits for the stricter exact-score margin and then performs final verification. A recomputed baseline using this same strict margin returns the same six-step gain. Comparisons between cached and recomputed execution must align their stopping policy before comparing work.

14 A singular rotated execution

The rotated fixture makes the prior singular and the packets non-coordinate:

$$P = J = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad H = I, \quad R = I, \quad m = 0, \quad y = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

It uses the rectangular prior factor $L = (1,1)^{\mathsf{T}}$, zero noise factor, and $\sigma = 1$. Both output and input packet families are

$$Q_+ = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad Q_- = \frac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

These project onto the common and antisymmetric directions. They are rational, self-adjoint, orthogonal, and complete. The prior has rank one, but the innovation matrix $S = I + J$ is positive definite because of the positive noise floor.

Take $B_0 = 0$, seed 0, scan $[0,1]$, probes $[4,12]$, $\varepsilon = 1/4$, and $\delta = 1/1000$. Here $\operatorname{tr} S = 4$, $\tau = 1/5$, and $T = I - (I+J)/5$. The initial residual is J, entirely in the Q_+ output packet. Since $JS = 3J$, each selected update multiplies this residual by $2/5$. The Q_- residual remains zero. Thus every selected label is 0 and

$$(e_+, e_-) = (4(4/25)^n, 0), \qquad B_n = \frac{1-(2/5)^n}{3} J.$$

The first four pre-update energy pairs are $(4,0)$, $(16/25,0)$, $(64/625,0)$, and $(256/15625,0)$.

After four updates,

$$B_4 = \frac{203}{625} J, \qquad F(B_4) = \frac{16}{625} J, \qquad a(B_4) = \frac{3072}{390625}.$$

The score weight is $w = 3$, because $\|y\|^2 = 2$. The strict margin is $\varepsilon^2/4 = 1/64$. At the previous state the score is $768/15625 > 1/64$; after the fourth it is below $1/64$. The computed fuel is 4267. Precision 4 produces score $435/64$ and fails; precision 12 produces $9111/1048576$ and passes the final threshold $1/16$.

The exact posterior mean is zero because $Jy = 0$, and the returned approximation's mean is also zero. Its Joseph covariance is

$$V_{B_4} = \frac{130379}{390625} J.$$

The exact posterior covariance is $J/3$. Both laws remain supported in the prior's one-dimensional common-direction subspace. This example therefore combines a singular prior, rotated packet selection, a nontrivial covariance approximation, and a zero mean without using an inverse of the singular prior covariance.

At output precision 32, each mean center is zero with radius $1/2147483648$. Every covariance entry has center $179191533/536870912$ and radius $7/4294967296$. All output radii satisfy $\delta = 1/1000$. The equality of these centers reflects the exact common-direction structure of the returned Joseph covariance. It does not turn the rounded center matrix into the object of the posterior theorem.

`GeneratedCachedRotated.lean` checks the same kinds of finite obligations as the coordinate module: fuel, chosen labels, gain, scores, actual return, parameter precision, arrays, and `ResultValid`. The example is particularly useful for testing whether a presentation has accidentally assumed an invertible prior or coordinate-only packets. Neither assumption belongs to the solver contract.

15 The rounded-cache counterexample and the separate rounded solver

Exact cache validity cannot be reused unchanged after independently rounding its components. The release contains a scalar counterexample with $P = H = R = 1$, $\sigma = 1$, $m = 0$, $y = 1$, and $B_0 = 0$. Then $S = 2$, $\tau = 1/3$, $w = 2$, and the first exact update is

$$B_1 = 1/3, \qquad F(B_1) = 1/3.$$

The naive route separately rounds the gain center and residual center downward to zero fractional bits. Both become zero. It then computes the stored energy from the rounded residual, obtaining a stored score of zero.

The actual residual at its stored gain is instead

$$F(0) = 1, \qquad a(0) = 2.$$

Thus the state claims exact internal convergence while its true residual is nonzero. The equality $G = QF(B)$ has been destroyed by rounding independently related quantities. The source module `CachedRoundingBoundary.lean` proves `naive_score_zero`, `true_score_two`, and `naive_cache_invalid`. For $\varepsilon = 1/10$ and probe [12], it also proves `naive_final_verification_rejected`: `certifyGain` returns no candidate.

This result identifies both a failure and a protection. The naive rounded cache cannot support the exact-cache termination proof. The independent final verifier still rejects its false stopping signal, preserving the meaning of successful returns. The example does not rule out a rounded selective cache with propagated error bounds. Such a construction needs an enclosure relation, an error budget for each selected update, and a selection argument that tolerates uncertain energies. Those obligations are a separate extension.

Core 1.0.0 already retains a different, completed rounded solver. Its update is a genuinely rounded full-gain Richardson step. It neither claims exact selective cache invariants nor selects a largest cached packet. Its declarations expose the whole-update defect instead. If $\widetilde{B}_{n+1}$ is the rounded update and B^{ideal}_{n+1} is the exact full step at the current gain, then

$$F(\widetilde{B}_{n+1}) = F(\widetilde{B}_n)T - (\widetilde{B}_{n+1} - B^{\mathrm{ideal}}_{n+1})S.$$

This is `rounded_residual_step`. The defect is explicitly included in the tracking and termination argument.

Writing d for the number of gain entries and $h_t = 2^{-t}$, the source establishes an all-iteration tracking bound

$$E(\widetilde{B}_n - B^{\mathrm{exact}}_n) \le \frac{6dh_t^2}{(1-q)^2}.$$

Its residual-score comparison has the form

$$a(\widetilde{B}_n) \le 2a(B^{\mathrm{exact}}_n) + C_{\mathrm{round}}h_t^2, \qquad C_{\mathrm{round}} = \frac{12dE(S)w}{(1-q)^2}.$$

The source names are `roundedGain_tracking`, `roundoffScoreCoeff`, and `rounded_score_compare`. The first term is ideal progress. The second is a precision-dependent floor. Both are needed.

The chosen update precision `updateAccuracy` places the floor below $\varepsilon^2/8$. The rounded fuel `roundedFuel` drives the exact full-step score below $(\varepsilon/4)^2 = \varepsilon^2/16$. The factor of two in the comparison therefore gives another $\varepsilon^2/8$, so the actual rounded iterate reaches score at most $\varepsilon^2/4$. The same independent final precision fallback can certify it. `solveRounded_terminates` proves finite successful return, and `rounded_solver_gain_origin` identifies the result as an actual rounded iterate within its declared fuel.

There are consequently two completed branches: exact selective updates with an exact cache, and genuinely rounded full-gain updates with an explicit perturbation bound. A rounded selective cache combining their advantages remains an extension. No proof for one branch is presented as a theorem about the other.

16 Measured work, execution boundaries, and extensions

The cached representation reduces the work needed to maintain packet residuals relative to recomputing every packet at every selected update. It stores the projected residual matrices and energies, retains unselected objects, and updates the total by one scalar difference. Selection still scans the stored energies. The amount of storage depends on the number and shape of the represented packet matrices. Rational numerator and denominator growth also matters in execution. The algebraic locality theorem alone is not a formal bit-complexity estimate.

The archived bounded benchmark compares cached selective, recomputed selective, and full Richardson reference implementations on seven cases of dimensions at most eight. All use exact fraction arithmetic, the same zero-skipping matrix backend, the same strict exact-score stopping margin, and common requested posterior and output tolerances. All include setup, final rounded certification, and parameter output. The recomputed and cached selective branches return identical candidates. The full baseline maintains a full residual by recurrence and avoids unnecessary packet scans.

One warmup precedes five repetitions with rotating method order. Archived median producer times and selective-iteration ratios are

Case	Cached ms	Recomputed ms	Full ms	Iteration ratio
coordinate	2.38	2.46	2.33	1.94
rotated	4.38	4.45	4.28	2.16
sparse4	13.84	14.52	13.79	2.51
dense4	16.34	17.96	16.34	3.11
sparse8	171.98	174.30	169.87	4.62
dense8	200.69	235.15	195.44	5.61
conditioned4	19.16	21.36	17.46	3.08

The final column is recomputed-selective iteration time divided by cached iteration time. It excludes setup, verification, and output. The producer columns include these stages and exclude export and Lean checking. These are archived reference measurements, rather than fresh timing measurements from this publication session.

The cache improves selective iteration in these cases, yet these measurements establish no overall advantage over the efficient full baseline. Several cases require substantially more selective updates than full updates. The dense8 case, for example, takes 67 cached selective updates and 9 full updates. Final certification and output account for a large part of total production cost. The bounded result supports a narrower conclusion: preserving a packet cache avoids unnecessary selective recomputation, while end-to-end performance depends on more than the selective update loop.

Every method-case pair also exported an explicit final candidate certificate, for 21 checked benchmark outputs. Their archived Lean checks certify those explicit rational gains and posterior tolerances. They do not certify the Python timing trace, counters, or numeric parameter arrays from the benchmark runs. The two generated coordinate and rotated modules have the stronger finite-program and output-array checks described above. A universal theorem about the mathematical output evaluator is also distinct from a universal refinement theorem for a Python implementation of that evaluator.

The archived per-instance Lean checking measurements range from seconds to tens of seconds and dominate millisecond producer times. Each includes a fresh pinned Lean process, import, elaboration, and kernel checking with common dependencies already compiled. The dependency build is accounted

for separately. One checking time per export cannot reliably rank small differences. The source reports preserve these distinctions, rather than combining an iteration ratio with an unrelated claim of total acceleration.

The producer/exporter boundary is similarly precise. Python parses user data, executes a reference algorithm, and emits a Lean instance containing explicit rational data. Successful checking establishes the statement about that emitted instance. A universal theorem identifying arbitrary parser input with the emitted model is outside this release. The output record is a mathematical representation with checked arithmetic semantics, rather than a claim about every external file format or hardware arithmetic instruction.

The most direct extensions now have concrete obligations. A rounded selective cache must propagate residual and energy enclosures and prove progress under its actual selection rule. Approximate maximum selection needs a quantitative energy share bound. A changing packet hierarchy must preserve global accounting across refinement and coarsening. Uncertain model inputs need their own error budget in both the certificate and the posterior semantics. Sequential filtering must account for repeated updates and transport of earlier approximation error. Large-scale performance work must measure preprocessing, final certification, output representation, rational bit growth, memory, and checking cost as well as iteration kernels.

These extensions do not leave the first-release theorem unfinished. The closed contract remains exact rational, finite-dimensional, single-update Gaussian inference with complete fixed packets, positive tolerances, a terminating gain construction, independent dyadic verification, and enclosed parameters of the certified Joseph law. The rounded-cache counterexample and bounded performance result make the limits of that contract inspectable and give specific targets for the next stage of development.

17 Assurance, reproducibility, and the completed scope

The result has a mathematical input boundary, a formal program boundary, and an implementation boundary. These boundaries can be read directly from the declared functions and predicates. Rational factor-coded data determine the Gaussian model. Complete rational packet families determine the permitted decomposition. A successful formal program return determines the candidate gain and parameter arrays. The Python producer and a physical execution environment require additional correspondence evidence.

Object	Established conclusion	Evidence role
Factor-coded model	Prior covariance PSD, innovation invertible at positive noise floor	Internal algebra and Gaussian semantics
Posterior kernel	Explicit Gaussian version disintegrates the forward law; canonical agreement evidence-a.e.	Internal measure-level proof
Joseph gain	A valid Gaussian approximation and residual-based W_2 bound	Common-noise coupling and matrix bounds
Exact packet cache	Correct residuals and energies at every iterate; same gains as recomputation	Initialization and preservation induction
Computed search fuel	Successful search under the positive-input hypotheses	Geometric contraction and computable sufficient fuel

Object	Established conclusion	Evidence role
Rounded verifier	Candidate validity and posterior tolerance follow from successful checking	Outward enclosure and inflation proofs
Parameter arrays	Enclose this approximation's parameters, with radii at most δ	Declared evaluator and precision theorem
Python instance	Exact emitted rational data can receive a Lean certificate	Compilation needed for that instance
Fresh publication runs	Specified examples and diagnostic assertions reproduced	Bounded executable evidence
Archived Lean release	117 successful module receipts, 543 axiom-audited declarations	Preserved historical kernel evidence

17.1 What was rechecked for this publication

The standalone Core 1.0.0 archive was recovered and its SHA-256 compared with its separately delivered checksum. The source release and its history were retained intact. The fresh inventory checks the archived manifest, current source bytes, retained source snapshots, logs, dependency-source records, declaration locations, and axiom-report coverage. An independent parser finds the same 543 theorem and lemma declarations. Verbatim excerpts in the Lean companion are matched back to their actual source intervals.

Fresh exact computations reproduce the coordinate and rotated returns, including their gains, score values and every output interval. Independent rational matrix calculations reconstruct the exact posterior and the Joseph covariance. They check the coupling cost, residual bound, cache identities, stopping thresholds, and the rounded-cache obstruction. The computation record states its finite domain and distinguishes producer checks from formal program theorems. The source archive remains byte-for-byte unchanged.

The figures use exact input data and explicit formulas. Floating evaluation is used to draw curves and contours, with that visual role identified in their captions. No sampled scatter cloud is used as a proof of a Gaussian identity. Archived benchmark timings are marked as archived. Fresh timings of publication checks are not substituted into the original performance table.

All three documents are compiled and rendered for page review. The audit records track any mathematical transcription corrections and the inspected page coverage. Build warnings and PDF structure checks address layout, references, links and glyphs. These checks verify the publications as artifacts; they do not extend the solver's formal scope.

The original release records a clean rebuild after removing active project caches while reusing pinned Mathlib caches. It preserves successful source and log receipts, corrected failures, and recorded compiled-output hashes. Compiled output payloads and the portable runtime are not included in the source archive. A fresh source audit therefore cannot recompute absent compiled-output hashes. Fresh Lean compilation was unavailable for this publication edition. The companion provides the exact pinned reproduction route and keeps this limitation explicit.

17.2 Interpreting cost evidence

The controlled benchmark contains seven models with dimensions at most eight, using five measured repetitions after warmup. Cached and recomputed selective runs use matching tolerances, backend and

stopping conditions and return identical gains. Their iterative work differs because the cache updates only the selected packet. Precomputation, final interval verification, parameter representation, export and proof checking remain parts of a full execution.

The archived measurements support a reduction in selective iteration cost. They do not establish overall superiority to the efficient full Richardson baseline. Operation counters describe the tested implementation's scalar arithmetic and comparisons. They do not constitute a formal bit-complexity theorem or a prediction for large matrices. Exact rational numerator and denominator growth, memory use, model preparation and certificate evaluation would need to be included in a practical scale study.

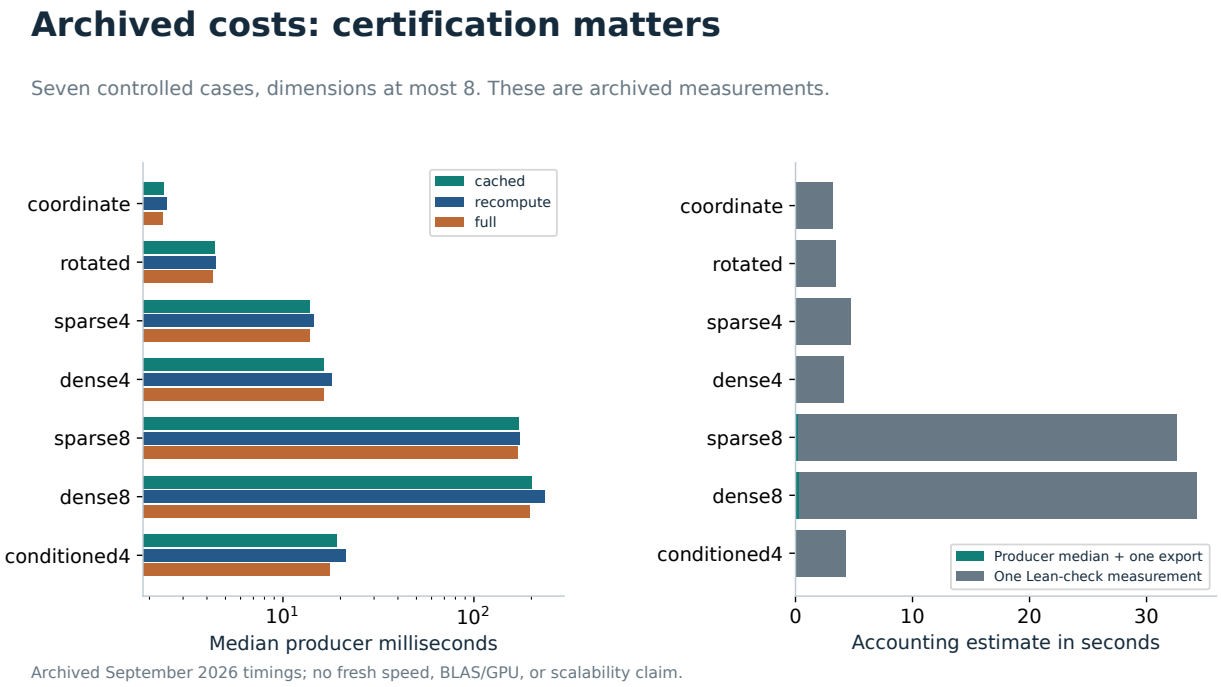


Figure 2: Archived bounded cost comparison. The plotted measurements concern the seven declared small models and their original reference environment. Iterative cache savings and whole-producer cost are different quantities.

17.3 Extensions that preserve the completed result

The first extension is a rounded selective cache. Its invariant must carry uncertainty about the semantic residual of the stored gain. A center-only update cannot reuse the exact equality $G_a = Q_a F(B)$ after independent rounding. The checked counterexample identifies the failed implication. A valid replacement could track residual enclosures and allocate update, selection and final-verification error budgets. That construction would require a new progress proof and a new origin statement.

Sequential filtering is another extension. A repeated update needs a contract for how the returned representation becomes the next prior, including propagation of approximation error and data uncertainty. The present output arrays represent the returned Joseph law. Replacing them by centers or repeatedly treating an approximate prior as exact would add unstated obligations. The single-update theorem remains a reusable component for such a future composition.

Changing packet geometry also requires a transition contract. The exact invariant uses one fixed complete orthogonal family. A new family can represent the same residual, but its cache must be reconstructed or transported with a proved relationship. Adaptive choice of a label within the fixed family is already covered. Adaptive replacement of the family is a separate operation.

A universally refined host implementation would connect parsing, rational representation, code generation and execution to the defined Lean program. The current exporter provides explicit candidate data and checkable instance source. Its bounded diagnostics support that route without supplying a universal refinement theorem. Executable sampling and hardware floating-point implementations similarly need their own input, arithmetic and output contracts.

18 Conclusion

Certified Inference Core 1.0.0 closes a precise finite-dimensional inference construction. Its exact cached packet computation has a maintained semantic invariant, a proved greedy trajectory, and computed finite fuel. Its final outward-rounded verifier connects the returned gain to a posterior-measure tolerance. Its parameter evaluator returns enclosing arrays for the same Gaussian approximation, with a prescribed radius bound. These are composed conclusions about one actual program result.

The worked examples show the construction operating with rectangular observations and a singular prior. The rounded-cache counterexample supplies an equally useful boundary: the proof identifies which state equality an attractive alternative loses. Together, the positive result and the obstruction make the arithmetic contract inspectable. The extensive visual and Lean companions provide routes from the picture to the exact formula, and from the formula to the source declaration.

A Source and reproduction map

The authoritative archive is `inputs/FD_Certified_Inference_Core_v1.0.0_Checked.zip`. Its SHA-256 is

`844ae280ea4be7e7c25906e00733f0130996b32aa110c53951df00f8c18fa488`.

Extract it under `source`. The publication package includes all editable LaTeX, twenty figures, their common generator, fresh exact results, source maps, computation provenance, independent audit and page-review records. The intact archive contains the underlying formal modules, Python reference tools, historical receipts, failed attempts, benchmark records and runtime restoration metadata.

The formal pins are Lean 4.34.0-rc2, compiler commit `6a10ac8c22beadecabdbb0919c2b50214762f91d`, and Mathlib commit `aeb72f1e9981b6d35b426b16dca7759be3e7ba06`. The dependency manifest supplies the remaining pins. Read the original `README.md` and `RELEASE_SCOPE.md` before rebuilding. Original historical receipts should remain separate from newly generated evidence. The main theorem is in `Komorebi/CachedPacketOutput.lean`; its result predicate is in `Komorebi/PosteriorOutputSolver.lean`.

References

[1] Komorebi P. *Certified Inference Core 1.0.0*. Checked source release, 12 September 2026. FD_Certified_Inference_Core_v1.0.0_Checked.zip.

[2] J. Nutini, M. Schmidt, I. H. Laradji, M. Friedlander, and H. Koepke. Coordinate Descent Converges Faster with the Gauss-Southwell Rule Than Random Selection. *Proceedings of Machine Learning Research* 37:1632–1641, 2015. https://proceedings.mlr.press/v37/nutini15.html.

[3] J. Nutini, I. Laradji, and M. Schmidt. Let's Make Block Coordinate Descent Converge Faster: Faster Greedy Rules, Message-Passing, Active-Set Complexity, and Superlinear Convergence. *Journal of Machine Learning Research* 23(131):1–74, 2022. https://jmlr.org/papers/v23/18-045.html.

[4] J. T. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth. Pathwise Conditioning of Gaussian Processes. *Journal of Machine Learning Research* 22(105):1–47, 2021. https://jmlr.org/papers/v22/20-1260.html.

[5] M. Daumas and G. Melquiond. Certification of bounds on expressions involving rounded operators. *arXiv:cs/0701186v2*, 24 May 2007. https://arxiv.org/abs/cs/0701186v2.