OPERATION KOMOREBI

REALIZATION CALCULUS



RESEARCH LIBRARY

Quantum–Arithmetic Bridge

Projectors, Fourier packets, and unitary realization

Lean and evidence guide

Komorebi P.

Standardized publication edition / 8 October 2026

REALIZATION CALCULUS

Reading scope

A source concordance for the checked bridge

The later five-module checkpoint, its 96 named theorems, archived compiler and kernel evidence, and the exact correspondence with the publication claims.

One exact example. Arithmetic chooses the coprime residues modulo twelve. A rational projector selects the same four modes. A normalized Fourier transform realizes the corresponding coherent state. Every comparison is an explicit equality with its conventions stated.

Contents

1 What this guide certifies 3

2 The typed mathematical target 3

3 Revision reconciliation 4

4 Architecture of the proof 4

5 The rational signal and its filter 4

6 The exact phase arithmetic 6

7 Finite indices and normalized Fourier certificates 7

8 Why the coordinate certificates mean complex identities 8

9 Matrix transport and actual unitarity 9

10 The exponential sign and square-root normalization 10

11 Hermitian projection and phase-sensitive states 12

12 Two exact information-loss statements 13

13 The publication theorem in full 14

14 Manuscript-to-Lean concordance 15

15 Trust boundary and current verification 16

16 Controls, failed attempts, and reproducibility 17

17 Complete active theorem inventory **17**

17.1 `QuantumArithmeticBridge`: 22 theorems 17

17.2 `FiniteFourier`: 17 theorems 18

17.3 `SemanticBridge`: 30 theorems 19

17.4 `Conventions`: 17 theorems 19

17.5 `Publication`: 10 theorems 20

18 What completion means here **20**

1 What this guide certifies

The completed fixed example is a bridge between exact arithmetic and ordinary complex linear algebra. It certifies a twelve-dimensional Fourier matrix, a Ramanujan projector, two normalized states with the same initial probabilities, and their opposite outcomes after a basis change. This guide maps those claims to actual Lean declarations rather than treating numerical agreement as a proof.

The authoritative formal checkpoint is the later five-module version of the September 21, 2026 development. It contains **96 named mathematical theorems**. Their source and archived execution records are supplied unchanged. The present release has independently reconciled the source and log hashes and the complete theorem inventory. A new compiler run was blocked by a runtime transfer error. The distinction between an archived checked proof and a fresh replay is retained throughout.

Production module	Named theorems
`QuantumArithmeticBridge`	22
`FiniteFourier`	17
`SemanticBridge`	30
`Conventions`	17
`Publication`	10
Total	96

These counts include supporting lemmas. They count named source declarations, not new discoveries, compiler jobs, or declarations generated internally by Lean. The namespace is `Komorebi.QuantumArithmetic` throughout.

2 The typed mathematical target

The complex matrices act on vectors indexed by $\mathrm{Fin}(12)$. The Fourier convention is

$$F_{nk} = \frac{1}{\sqrt{12}} \exp\left(\frac{2\pi i n k}{12}\right), \qquad n, k \in \{0, \ldots, 11\}.$$

The adjoint F^* therefore carries the negative exponential sign. The selector D is diagonal with value one precisely at the residues $U = \{1, 5, 7, 11\}$. The transported operator is $P = FDF^*$.

Write

$$c = (4, 0, 2, 0, -2, 0, -4, 0, -2, 0, 2, 0), \qquad \psi = \frac{c}{\sqrt{48}}, \qquad \chi = \frac{|c|}{\sqrt{48}}.$$

Both vectors have norm one. Their initial probability histograms are identical, but their inner product is zero. The vector u has entry $1/2$ on U and zero elsewhere. Lean proves the actual complex identities $Fu = \psi$ and $F^*\psi = u$, as well as transformed coprime probability one for ψ and zero for χ.

The statements are finite and exact. No parameterized arbitrary-modulus theorem, general circuit construction, physical phase-removal process, or quantum speedup is included in these formal claims.

3 Revision reconciliation

Two retrieved archives carry version `0.2.0`, but they do not have the same active source. The earlier four-module checkpoint has 59 mathematical theorems and four separately checked refutation controls. The later five-module checkpoint has 96 mathematical theorems and three rejected control files. Its proof receipt is the later `verification/final/REPORT.json`.

The original 22 finite-table statements are retained inside both checkpoints. They are not counted a second time. Likewise, $59 + 96$ is not a meaningful theorem total: these are overlapping revisions of one formal development.

The earlier proof boundary disclosed that explicit projector transport and normalized-state matrix actions had not yet been supplied. The later source supplies both, connects the Fourier matrix to the usual complex exponential, and proves a histogram nonreconstruction theorem. The earlier boundary remains a record of its own revision, rather than a gap list for the current publication.

The entire later checkpoint is preserved under `lean/checked_checkpoint`. The source files are not edited to match this guide.

4 Architecture of the proof

The proof crosses two representations. First, rational coordinates support exact finite calculations. Second, a semantics map transfers those calculations to ordinary complex matrices and vectors.

Layer	Role
Rational tables	Signal, projector entries, filtering, and energies.
Exact coordinates	Algebra in the basis $1, \sqrt{3}, i, i\sqrt{3}$.
Finite matrices	All entries indexed by $\mathrm{Fin}(12)$, with exact finite sums.
Complex semantics	Preservation of multiplication, adjoint, matrix action, and squared norm.
Conventions	Identification of roots and normalization with standard exponentials and square roots.
Publication theorem	Bundling of the actual complex identities and reconstruction obstructions.

The active import chain is `QuantumArithmeticBridge` into `FiniteFourier`, then `SemanticBridge`, then `Conventions`, then `Publication`. Supporting Mathlib imports enter at the layer that uses them. There is no circular theorem dependency.

5 The rational signal and its filter

The first layer uses literal rational lists. The chosen perturbation is $r_n = 3\cos(\pi n/2) + 2$, whose displayed rational entries can be certified without numerical trigonometric evaluation. Its sum with c is the test signal.

Literal source. `QuantumArithmeticBridge.lean`, physical lines 51–61.

```
/-- Ramanujan sum c_12(n), n = 0,...,11. -/
def ramanujan12 : QVector :=
  [4, 0, 2, 0, -2, 0, -4, 0, -2, 0, 2, 0]

/-- A deliberately off-band perturbation: 3 cos(pi n/2) + 2. -/
def noise12 : QVector :=
  [5, 2, -1, 2, 5, 2, -1, 2, 5, 2, -1, 2]

/-- Signal used in the essay's filter demonstration. -/
def testSignal12 : QVector :=
  [9, 2, 1, 2, 3, 2, -5, 2, 3, 2, 1, 2]
```

The cyclic matrix has entry $c_{(n-m) \bmod 12}/12$. The original layer proves its symmetry, idempotence, preservation of c, rejection of the chosen perturbation, and exact recovery from the displayed contaminated signal. The exact energy contract is $150 = 48 + 102$.

Literal source. `QuantumArithmeticBridge.lean`, physical lines 92–126.

```
theorem projector12_idempotent :
    matMulN 12 projector12 projector12 = projector12 := by
  decide +kernel

/-- The arithmetic signal lies in the selected subspace. -/
theorem projector12_fixes_ramanujan :
    matVec projector12 ramanujan12 = ramanujan12 := by
  decide +kernel

/-- The chosen constant-plus-frequency-3 perturbation lies in the complementary subspace. -/
theorem projector12_kills_noise :
    matVec projector12 noise12 = List.replicate 12 0 := by
  decide +kernel

/-- Therefore the exact filter recovers the arithmetic component. -/
theorem projector12_recovers_test_signal :
    matVec projector12 testSignal12 = ramanujan12 := by
  decide +kernel

/-- The arithmetic amplitude vector has squared norm 48. -/
theorem ramanujan12_energy : sqEnergy ramanujan12 = 48 := by
  decide +kernel

/-- The perturbation has squared norm 102. -/
theorem noise12_energy : sqEnergy noise12 = 102 := by
  decide +kernel

/-- The full signal has squared norm 150. -/
theorem testSignal12_energy : sqEnergy testSignal12 = 150 := by
  decide +kernel

/-- Exact orthogonal energy decomposition 150 = 48 + 102. -/
theorem energy_decomposition :
    sqEnergy testSignal12 = sqEnergy ramanujan12 + sqEnergy noise12 := by
  decide +kernel
```

The list operations use finite lists, including zero defaults and zip-based dot products. These facts are therefore claims about the supplied twelve-entry objects. They should not be silently interpreted as universal matrix theorems for arbitrary ragged lists. The later Fin(12) layer supplies the typed matrix identities used in the publication.

6 The exact phase arithmetic

The structure `Cyclo12` represents

$$(a,b,c,d) \longmapsto a + b\sqrt{3} + i(c + d\sqrt{3}), \qquad a,b,c,d \in \mathbb{Q}.$$

Its multiplication explicitly incorporates $(\sqrt{3})^2 = 3$ and $i^2 = -1$. It is a concrete arithmetic representation. The development does not need or install a global field instance to certify the finite example.

Literal source. `QuantumArithmeticBridge.lean`, physical lines 130–141.

```
/--
An exact coordinate model for Q(sqrt(3), i).  The tuple (a,b,c,d) denotes
  a + b*sqrt(3) + i*(c + d*sqrt(3)).
Only the concrete arithmetic needed for the q=12 certificate is used below;
we do not install a global field instance.
-/
structure Cyclo12 where
  a : ℚ
  b : ℚ
  c : ℚ
  d : ℚ
  deriving DecidableEq, Repr
```

Literal source. `QuantumArithmeticBridge.lean`, physical lines 155–173.

```
/-- Multiplication in Q(sqrt(3), i). -/
def cMul (x y : Cyclo12) : Cyclo12 :=
  ⟨x.a*y.a + 3*x.b*y.b - x.c*y.c - 3*x.d*y.d,
   x.a*y.b + x.b*y.a - x.c*y.d - x.d*y.c,
   x.a*y.c + 3*x.b*y.d + x.c*y.a + 3*x.d*y.b,
   x.a*y.d + x.b*y.c + x.c*y.b + x.d*y.a⟩

def cConj (x : Cyclo12) : Cyclo12 := ⟨x.a, x.b, -x.c, -x.d⟩

def cPow (x : Cyclo12) : Nat → Cyclo12
  | 0 => cOne
  | n + 1 => cMul (cPow x n) x

def cSum : List Cyclo12 → Cyclo12
  | [] => cZero
  | x :: xs => cAdd x (cSum xs)

/-- zeta = exp(2*pi*i/12) = sqrt(3)/2 + i/2. -/
def zeta12 : Cyclo12 := ⟨0, (1/2 : ℚ), (1/2 : ℚ), 0⟩
```

The phase $\zeta = (0, 1/2, 1/2, 0)$ has complex meaning $\sqrt{3}/2 + i/2$. Exact finite certificates establish its twelfth power, inverse, and absence of a smaller positive power below twelve. Later the convention module identifies it with $\exp(\pi i/6)$.

The Ramanujan vector is obtained by summing powers of this phase over $1, 5, 7, 11$. The negative-sign unnormalized DFT has value 12 at these four frequencies and zero elsewhere. The absolute-value vector has Fourier support on the even frequencies instead. These are exact phase calculations over rational coordinates.

7 Finite indices and normalized Fourier certificates

The finite layer uses `Index := Fin 12`. Its normalized Fourier scale is represented exactly as $\sqrt{3}/6 = 1/\sqrt{12}$, and its state scale as $\sqrt{3}/12 = 1/\sqrt{48}$. The use of a finite index type makes every row and column well defined without list-length side conditions.

Literal source. `FiniteFourier.lean`, physical lines 58–93.

```
/-- The stored arithmetic selector is exactly the coprimality predicate. -/
theorem coprime_residues_exact :
    (List.range 12).filter (fun a => Nat.gcd a 12 == 1) = unitResidues12 := by
  decide +kernel

theorem root_exact_order :
    ∀ n : Fin 12, 0 < n.val → cPow zeta12 n.val ≠ cOne := by
  decide +kernel

theorem normalized_fourier_left :
    ∀ i j, kMatMul (kAdjoint exactFourier) exactFourier i j = kIdentity i j := by
  decide +kernel

theorem normalized_fourier_right :
    ∀ i j, kMatMul exactFourier (kAdjoint exactFourier) i j = kIdentity i j := by
  decide +kernel

def selectedKernel : KMatrix := fun i j =>
  kSum (fun k => if Nat.gcd k.val 12 = 1 then
    cMul (exactFourier i k) (cConj (exactFourier j k)) else cZero)

theorem exact_projector_transport :
    ∀ i j, selectedKernel i j = exactProjector i j := by
  decide +kernel

theorem exact_projector_matches_rational_table :
    ∀ i j, exactProjector i j = cRat ((projector12.getD i.val []).getD j.val 0) := by
  decide +kernel

theorem exact_state_preparation :
    ∀ i, kMatVec exactFourier exactUnitState i = exactPsi i := by
  decide +kernel

theorem exact_state_recovery :
    ∀ i, kMatVec (kAdjoint exactFourier) exactPsi i = exactUnitState i := by
  decide +kernel
```

The two Fourier identities above quantify over all 12×12 entries. Kernel reduction proves these concrete universal statements over a finite type. The selected-kernel equality and state actions are also complete finite certificates, rather than floating-point matrix comparisons.

8 Why the coordinate certificates mean complex identities

The semantics map is explicitly defined with the positive real square root. Soundness for addition, multiplication, conjugation, powers, and finite sums prevents the exact coordinate layer from being a detached calculation with an unproved interpretation.

Literal source. `SemanticBridge.lean`, physical lines 14–56.

```
/-- The exact coordinate arithmetic is interpreted in the ordinary complex numbers. -/
noncomputable def interpret (x : Cyclo12) : ℂ :=
  ⟨(x.a : ℝ) + (x.b : ℝ) * Real.sqrt 3,
   (x.c : ℝ) + (x.d : ℝ) * Real.sqrt 3⟩

@[simp] theorem interpret_zero : interpret cZero = 0 := by
  apply Complex.ext <;> simp [interpret, cZero]

@[simp] theorem interpret_one : interpret cOne = 1 := by
  apply Complex.ext <;> simp [interpret, cOne]

@[simp] theorem interpret_rat (r : ℚ) : interpret (cRat r) = (r : ℂ) := by
  apply Complex.ext <;> simp [interpret, cRat]

@[simp] theorem interpret_add (x y : Cyclo12) :
    interpret (cAdd x y) = interpret x + interpret y := by
  apply Complex.ext <;> simp [interpret, cAdd] <;> ring

@[simp] theorem interpret_mul (x y : Cyclo12) :
    interpret (cMul x y) = interpret x * interpret y := by
  apply Complex.ext <;> simp [interpret, cMul, Complex.mul_re, Complex.mul_im] <;>
    ring_nf <;> norm_num [Real.sq_sqrt]

@[simp] theorem interpret_conj (x : Cyclo12) :
    interpret (cConj x) = star (interpret x) := by
  apply Complex.ext <;> simp [interpret, cConj]
  ring

@[simp] theorem interpret_pow (x : Cyclo12) (n : Nat) :
    interpret (cPow x n) = interpret x ^ n := by
  induction n with
  | zero => simp [cPow]
  | succ n ih => simp [cPow, ih, pow_succ]

@[simp] theorem interpret_sum (xs : List Cyclo12) :
    interpret (cSum xs) = (xs.map interpret).sum := by
  induction xs with
  | nil => simp [cSum]
  | cons x xs ih => simp [cSum, ih]

@[simp] theorem interpret_kSum (f : Index → Cyclo12) :
    interpret (kSum f) = ∑ i, interpret (f i) := by
  simp only [kSum, interpret_sum, List.map_ofFn, List.sum_ofFn]
```

The multiplication proof uses complex extensionality, real polynomial normalization, and the square-root identity. It proves preservation for arbitrary coordinates, not merely for the handful of roots that occur in the example. The power and finite-sum proofs use induction.

Injectivity of this interpretation is not needed to transfer an equality: if two coordinate expressions are proved equal, applying the interpretation gives equal complex expressions. No claim of a globally packaged field equivalence is required by the theorem chain.

9 Matrix transport and actual unitarity

The semantics layer proves that coordinate matrix multiplication, adjoint, identity, and matrix-vector action have their ordinary complex meanings. The matrices F, P, D and the normalized states are then defined directly in that complex representation.

Literal source. `SemanticBridge.lean`, physical lines 65–91.

```
@[simp] theorem complexMatrix_mul (A B : KMatrix) :
    complexMatrix (kMatMul A B) = complexMatrix A * complexMatrix B := by
  ext i j
  simp [complexMatrix, kMatMul, Matrix.mul_apply]

@[simp] theorem complexMatrix_adjoint (A : KMatrix) :
    complexMatrix (kAdjoint A) = (complexMatrix A)ᴴ := by
  ext i j
  simp [complexMatrix, kAdjoint, Matrix.conjTranspose_apply]

@[simp] theorem complexMatrix_identity :
    complexMatrix kIdentity = 1 := by
  ext i j
  by_cases h : i = j <;> simp [complexMatrix, kIdentity, h, Matrix.one_apply]

@[simp] theorem complexVector_mul (A : KMatrix) (v : KVector) :
    complexVector (kMatVec A v) = (complexMatrix A).mulVec (complexVector v) := by
  ext i
  simp [complexVector, complexMatrix, kMatVec, Matrix.mulVec, dotProduct]

noncomputable def F : Matrix Index Index ℂ := complexMatrix exactFourier
noncomputable def P : Matrix Index Index ℂ := complexMatrix exactProjector
noncomputable def D : Matrix Index Index ℂ :=
  Matrix.diagonal (fun i => if Nat.gcd i.val 12 = 1 then 1 else 0)
noncomputable def psi : Index → ℂ := complexVector exactPsi
noncomputable def chi : Index → ℂ := complexVector exactChi
noncomputable def unitState : Index → ℂ := complexVector exactUnitState
```

Consequently, the finite certificates produce actual complex unitarity and projector transport. The following excerpt contains the full proofs of both unitary identities, the cross-representation projector identity, and the two state-action identities.

Literal source. `SemanticBridge.lean`, physical lines 93–120.

```
/-- Actual complex unitarity, not an assertion only about the coordinate checker. -/
theorem fourier_unitary_left : Fᴴ * F = 1 := by
  have h : kMatMul (kAdjoint exactFourier) exactFourier = kIdentity :=
    funext (fun i => funext (normalized_fourier_left i))
  simpa [F] using congrArg complexMatrix h

theorem fourier_unitary_right : F * Fᴴ = 1 := by
  have h : kMatMul exactFourier (kAdjoint exactFourier) = kIdentity :=
```

```
    funext (fun i => funext (normalized_fourier_right i))
  simpa [F] using congrArg complexMatrix h

theorem projector_transport : F * D * Fᴴ = P := by
  ext i j
  have h := congrArg interpret (exact_projector_transport i j)
  simp only [selectedKernel, interpret_kSum] at h
  rw [Matrix.mul_apply]
  simpa [D, P, F, complexMatrix, Matrix.mul_diagonal, Matrix.conjTranspose_apply,
    apply_ite interpret, mul_ite] using h

theorem state_preparation : F.mulVec unitState = psi := by
  have h : kMatVec exactFourier exactUnitState = exactPsi :=
    funext exact_state_preparation
  simpa [F, unitState, psi] using congrArg complexVector h

theorem state_recovery : Fᴴ.mulVec psi = unitState := by
  have h : kMatVec (kAdjoint exactFourier) exactPsi = exactUnitState :=
    funext exact_state_recovery
  simpa [F, unitState, psi] using congrArg complexVector h
```

This is the key strengthening over the earlier checkpoint. The rational matrix, the exact coordinate kernel, and the complex matrix FDF^* are explicitly connected. State preparation and recovery are equalities of actual complex vectors. The theorem name `state_preparation` describes a matrix identity, not an efficient hardware preparation algorithm.

10 The exponential sign and square-root normalization

The usual Fourier matrix is not left as an informal interpretation. The convention module proves that the coordinate root equals the positive exponential root, including the selected square-root branch.

Literal source. `Conventions.lean`, physical lines 10–24.

```
/-- The coordinate root is exactly the usual exponential twelfth root. -/
theorem root_is_exponential :
    interpret zeta12 = Complex.exp (((Real.pi / 6 : ℝ) : ℂ) * Complex.I) := by
  rw [Complex.exp_mul_I]
  rw [← Complex.ofReal_cos, ← Complex.ofReal_sin, Real.cos_pi_div_six, Real.sin_pi_div_six]
  apply Complex.ext <;> simp [interpret, zeta12]
  ring

theorem root_power_twelve : interpret zeta12 ^ 12 = 1 := by
  simpa using congrArg interpret zeta12_order_certificate

theorem rootPower_interpret (n : Nat) :
    interpret (rootPower n) = interpret zeta12 ^ n := by
  simp only [rootPower, interpret_pow]
  exact (pow_eq_pow_mod n root_power_twelve).symm
```

Its normalization lemmas prove $\sqrt{12} = 2\sqrt{3}$ and $\sqrt{48} = 4\sqrt{3}$, then identify the exact scales with their ordinary reciprocals. The matrix entry formula and state formula below are therefore literal equalities in $\mathbb{C}$.

Literal source. `Conventions.lean`, physical lines 42–54.

```
theorem fourierScale_interpret :
    interpret fourierScale = ((1 / Real.sqrt 12 : ℝ) : ℂ) := by
  apply Complex.ext <;> simp [interpret, fourierScale, sqrt_twelve]
  have h : Real.sqrt 3 ≠ 0 := by positivity
  field_simp
  nlinarith [Real.sq_sqrt (show (0 : ℝ) ≤ 3 by norm_num)]

theorem stateScale_interpret :
    interpret stateScale = ((1 / Real.sqrt 48 : ℝ) : ℂ) := by
  apply Complex.ext <;> simp [interpret, stateScale, sqrt_fortyeight]
  have h : Real.sqrt 3 ≠ 0 := by positivity
  field_simp
  nlinarith [Real.sq_sqrt (show (0 : ℝ) ≤ 3 by norm_num)]
```

Literal source. `Conventions.lean`, physical lines 56–65.

```
/-- The matrix checked in the certificate is the standard positive sign Fourier matrix. -/
theorem fourier_entry_formula (i j : Index) :
    F i j = ((1 / Real.sqrt 12 : ℝ) : ℂ) *
      Complex.exp (2 * (Real.pi : ℂ) * Complex.I * (i.val : ℂ) * (j.val : ℂ) / 12) := by
  simp only [F, complexMatrix, exactFourier, interpret_mul, fourierScale_interpret,
    rootPower_interpret, root_is_exponential]
  rw [← Complex.exp_nat_mul]
  congr 2
  push_cast
  ring
```

Literal source. `Conventions.lean`, physical lines 67–82.

```
theorem psi_entry_formula (i : Index) :
    psi i = ((1 / Real.sqrt 48 : ℝ) : ℂ) * (ramanujan12.getD i.val 0 : ℂ) := by
  simp [psi, complexVector, exactPsi, stateScale_interpret]

theorem chi_entry_formula (i : Index) :
    chi i = ((1 / Real.sqrt 48 : ℝ) : ℂ) * (absRamanujan12.getD i.val 0 : ℂ) := by
  simp [chi, complexVector, exactChi, stateScale_interpret]

theorem unitState_entry_formula (i : Index) :
    unitState i = if Nat.gcd i.val 12 = 1 then (1/2 : ℂ) else 0 := by
  by_cases h : Nat.gcd i.val 12 = 1 <;>
    simp [unitState, complexVector, exactUnitState, h]

theorem projector_entry_formula (i j : Index) :
    P i j = (ramanujan12.getD ((i.val + 12 - j.val) % 12) 0 : ℂ) / 12 := by
  simp [P, complexMatrix, exactProjector]
```

The convention is positive sign for F and negative sign for F^*. The earlier unnormalized `dft12` uses the negative sign. The normalization and adjoint are the precise link between those notations.

11 Hermitian projection and phase-sensitive states

The selector is Hermitian and idempotent. Transport through a unitary matrix then proves the corresponding facts for P. They are ordinary matrix identities, not claims that every rational list operation is an orthogonal projection.

Literal source. `Conventions.lean`, physical lines 84–108.

```
theorem selector_idempotent : D * D = D := by
  unfold D
  rw [Matrix.diagonal_mul_diagonal]
  congr 1
  funext i
  by_cases h : Nat.gcd i.val 12 = 1 <;> simp [h]

theorem selector_selfAdjoint : Dᴴ = D := by
  ext i j
  by_cases h : i = j
  · subst j
    simp [D, Matrix.conjTranspose_apply, apply_ite star]
  · simp [D, Matrix.conjTranspose_apply, h, Ne.symm h]

theorem projector_idempotent_complex : P * P = P := by
  rw [← projector_transport]
  calc
    (F * D * Fᴴ) * (F * D * Fᴴ) = F * D * (Fᴴ * F) * D * Fᴴ := by
      simp only [Matrix.mul_assoc]
    _ = F * D * Fᴴ := by rw [fourier_unitary_left, Matrix.mul_one, Matrix.mul_assoc F D D,
      selector_idempotent]

theorem projector_selfAdjoint_complex : Pᴴ = P := by
  rw [← projector_transport]
  simp [Matrix.conjTranspose_mul, selector_selfAdjoint, Matrix.mul_assoc]
```

The vectors ψ and χ are separately defined states. One is the signed arithmetic amplitude vector and the other its entrywise absolute value. They have the same squared amplitudes at each initial index but retain different relative phase. The development neither claims nor requires a universal quantum operation implementing the nonlinear map $v \mapsto |v|$.

Literal source. `SemanticBridge.lean`, physical lines 145–174.

```
theorem psi_normalized : totalProbability psi = 1 := by
  have h := congrArg interpret exact_psi_normalized
  simpa [psi] using congrArg Complex.re h

theorem chi_normalized : totalProbability chi = 1 := by
  have h := congrArg interpret exact_chi_normalized
  simpa [chi] using congrArg Complex.re h

theorem unitState_normalized : totalProbability unitState = 1 := by
  have h := congrArg interpret exact_unit_normalized
  simpa [unitState] using congrArg Complex.re h

theorem same_initial_probabilities (i : Index) :
    Complex.normSq (psi i) = Complex.normSq (chi i) := by
```

```
  have h := congrArg interpret (exact_same_initial_probabilities i)
  simpa [psi, chi, complexVector] using congrArg Complex.re h

theorem states_orthogonal : ∑ i, star (psi i) * chi i = 0 := by
  have h := congrArg interpret exact_orthogonal_states
  simpa [psi, chi, complexVector] using h

theorem signed_state_recovers_arithmetic :
    coprimeProbability (Fᴴ.mulVec psi) = 1 := by
  have h := congrArg interpret exact_coprime_mass_psi
  simpa [F, psi] using congrArg Complex.re h

theorem unsigned_state_erases_arithmetic :
    coprimeProbability (Fᴴ.mulVec chi) = 0 := by
  have h := congrArg interpret exact_coprime_mass_chi
  simpa [F, chi] using congrArg Complex.re h
```

The transformed probability statement is exact. The measurement event is the coprime-frequency subset. After applying F^*, its probability is one for ψ and zero for χ. This basis change reveals information that their shared initial histogram does not contain.

12 Two exact information-loss statements

The formal development isolates distinct obstructions. First, the initial histogram cannot be inverted on both states by any reconstruction function. Second, the projector itself is noninjective because it annihilates the nonzero normalized state χ.

Literal source. `Publication.lean`, physical lines 39–50.

```
/-- No state reconstruction function can invert the initial probability histogram on both states. -/
theorem no_reconstruction_from_initial_probabilities :
    ¬ ∃ R : (Index → ℝ) → (Index → ℂ),
      R (fun i => Complex.normSq (psi i)) = psi ∧
      R (fun i => Complex.normSq (chi i)) = chi := by
  rintro ⟨R, hpsi, hchi⟩
  have heq : (fun i => Complex.normSq (psi i)) =
      (fun i => Complex.normSq (chi i)) := funext same_initial_probabilities
  have hstate : psi = chi := by rw [← hpsi, ← hchi, heq]
  have h := signed_state_recovers_arithmetic
  rw [hstate, unsigned_state_erases_arithmetic] at h
  norm_num at h
```

Literal source. `Publication.lean`, physical lines 52–57.

```
/-- Exact probabilities in the recovered basis, connected to the normalized state. -/
theorem recovered_probability_at (i : Index) :
    Complex.normSq (Fᴴ.mulVec psi i) =
      if Nat.gcd i.val 12 = 1 then (1/4 : ℝ) else 0 := by
  rw [state_recovery, unitState_entry_formula]
  by_cases h : Nat.gcd i.val 12 = 1 <;> norm_num [h, Complex.normSq_apply]
```

Literal source. `Publication.lean`, physical lines 59–65.

```
/-- The operator is not injective. Its nonzero normalized kernel state is explicit. -/
theorem projector_not_injective : ¬ Function.Injective P.mulVec := by
  intro h
  have hz : chi = 0 := h (by simp [projector_kills_chi])
  have hp := chi_normalized
  rw [hz] at hp
  norm_num [totalProbability] at hp
```

The nonreconstruction theorem quantifies over every function from real-valued histograms to complex-valued vectors. It uses equality of the two histograms and the incompatible transformed probabilities. This is a mathematical obstruction to a specified inverse, not a statistical claim about estimating a state from many measurements in several bases.

The traditional paper also proves that the two states cannot both be recovered up to an overall phase from this histogram. That strengthening is a prose corollary of their orthogonality and normalization. It is not separately exported as a quotient-by-global-phase Lean theorem.

Projection becomes reconstructive when its residual is retained. The next identity holds for every twelve-dimensional complex vector, including vectors outside the arithmetic signal subspace.

Literal source. `Publication.lean`, physical lines 33–37.

```
/-- Every signal is reconstructible from the retained and discarded components together. -/
theorem projector_residual_reconstructs (v : Index → ℂ) :
    P.mulVec v + (1 - P).mulVec v = v := by
  rw [Matrix.sub_mulVec, Matrix.one_mulVec]
  exact add_sub_cancel _ _
```

Its proof is algebraic. The identity $Pv + (I - P)v = v$ itself does not require idempotence. Hermitian idempotence supplies the stronger interpretation as orthogonal retained and discarded components. This separates the reconstruction identity from the additional geometric structure.

13 The publication theorem in full

The final declaration bundles the formal core of the worked example. Every conjunct is supported by a previously proved declaration rather than by a new numerical test or an implicit physical assumption.

Literal source. `Publication.lean`, physical lines 67–80.

```
/-- Publication theorem gathering the complex identities stated in the worked example. -/
theorem publication_bridge :
    Fᴴ * F = 1 ∧ F * Fᴴ = 1 ∧ F * D * Fᴴ = P ∧
    P * P = P ∧ Pᴴ = P ∧ F.mulVec unitState = psi ∧
    Fᴴ.mulVec psi = unitState ∧ totalProbability psi = 1 ∧
    totalProbability chi = 1 ∧
    (∀ i, Complex.normSq (psi i) = Complex.normSq (chi i)) ∧
    (∑ i, star (psi i) * chi i) = 0 ∧
    coprimeProbability (Fᴴ.mulVec psi) = 1 ∧
    coprimeProbability (Fᴴ.mulVec chi) = 0 := by
  exact (fourier_unitary_left, fourier_unitary_right, projector_transport,
    projector_idempotent_complex, projector_selfAdjoint_complex, state_preparation,
    state_recovery, psi_normalized, chi_normalized, same_initial_probabilities,
    states_orthogonal, signed_state_recovers_arithmetic, unsigned_state_erases_arithmetic)
```

The bundled theorem has a fixed dimension and explicitly named operators and states. It establishes the bridge used in the paper. Broader arbitrary-modulus results remain separate prose theorems, with their own proofs and evidence.

The manuscript's QA18 additionally includes filtering and energy, normalization of u, the projector actions on both states, and noninjectivity. Those clauses use separately exported declarations, including `projector12_recovers_test_signal`, `energy_decomposition`, `unitState_normalized`, `projector_fixes_psi`, `projector_kills_chi`, and `projector_not_injective`. The complete machine concordance lists these supporting results. The manuscript bundle is not identified with a literal statement that omits its extra clauses.

14 Manuscript-to-Lean concordance

The main paper labels its results QA01 through QA20. The table below distinguishes the exact fixed-example formal coverage from the more general mathematical development. "Not formalized here" is a scope label, not a claim that the prose result is false.

Claim	Subject	Formal coverage
QA01	Primitive root	Exact order 12 and standard exponential root.
QA02	Character orthogonality	Fixed-example content through both exact unitary identities; general theorem not separately exported.
QA03	Fourier theory	Both ordinary complex unitary identities and standard entry formula; general Parseval theorem not separately exported.
QA04	Möbius formula	General modulus formula not formalized; exact integer table for 12 certified.
QA05	Ramanujan support	Exact signed and unsigned twelve-frequency DFT tables and arithmetic energy 48.
QA06	Divisor projectors	Primitive-order projector at 12 certified; the full divisor family not a general Lean family.
QA07	Projector geometry	Hermitian idempotence, displayed energies, explicit noninjectivity, and all-vector residual reconstruction.
QA08	Residue recombination	No theorem in this checkpoint.

Claim	Subject	Formal coverage
QA09	Polyphase aliasing	No theorem in this checkpoint.
QA10	Arithmetic example	Phase sum connected to the exact rational sample table.
QA11	Exact filtering	Displayed contaminated signal and orthogonal energy decomposition.
QA12	State preparation	Ordinary complex normalized vector actions and recovery.
QA13	Shared histogram	Entrywise probability equality and state orthogonality.
QA14	Phase separation	Actual transformed coprime probabilities 1 versus 0.
QA15	Reconstruction barrier	No vector-valued initial-histogram inverse works on both specified states; the global-phase corollary is prose only.
QA16	Measurement law	Relevant projector algebra and state probabilities; no general conditional-state-update API.
QA17	Arithmetic semantics	Soundness for scalar, matrix, vector, adjoint, and mass operations.
QA18	Central bridge	Publication bundle plus separately exported filter, energy, normalization, projector-action, and noninjectivity results.
QA19	Relative-phase family	Prose extension $\sin^2(\theta/2)$; not an inherited Lean theorem.
QA20	Dephased probability	Prose density-matrix calculation $1/3$; not an inherited Lean theorem.

The machine concordance includes all theorem statements verbatim, per-theorem axiom records, source hashes, and exact physical line locators. Its mapping records formal scope; it does not upgrade general manuscript claims merely because a nearby finite special case is checked.

15 Trust boundary and current verification

The archived run used Lean 4.34.0-rc2, compiler commit `6a10ac8c22beadecabdbb0919c2b50214762f91d`, and Mathlib revision `aeb72f1e9981b6d35b426b16dca7759be3e7ba06`. It built the five source modules with warnings treated as errors, printed axiom dependencies for all 96 named theorems, and replayed each module with `leanchecker`. The compiler and imported dependency cache were not rebuilt from their source. Kernel replay uses Lean's own kernel, not a second independent proof checker.

The axiom union is exactly `propext`, `Classical.choice`, and `Quot.sound`. This is the standard foundation used by the audited proof terms, not an axiom-free claim. The active source contains no proof placeholders, custom axiom declarations, native decision proofs, or unsafe declarations. Historical failed or native-decision drafts are excluded from the active build.

Current source reconciliation checked all five source hashes and all eleven archived check-log hashes against the archived report. All sixteen match. The current inventory contains exactly 96 unique names and 96 archived axiom records. All line numbers agree with the archived source index.

The present environment had no `lean`, `lake`, or `leanchecker` on PATH. The exact pinned five-part portable runtime was resolved from its documented existing files, but three materialization attempts failed with HTTP status 502. No runtime bytes landed and no compiler process ran. The current `FRESH_LEAN_STATUS.json` therefore says **NOT_RUN**. This transfer failure is preserved; the release does not claim a fresh Lean pass.

16 Controls, failed attempts, and reproducibility

The latest archive rejected a deliberately false signal sign and a proof using `sorry` under warnings as errors. Its false Fourier-mode control also failed, but the compiler stopped at opaque `Decidable` reduction. That failure alone is not a direct mathematical certificate that the proposition is false. The positive `ramanujan_dft_table` theorem separately establishes the correct value 12.

The control runner's broad diagnostic expression can match the word "false" in a filename. The guide does not interpret that match as a stronger arithmetic test than the actual log provides. The earlier checkpoint's four explicit refutation proofs are retained as earlier evidence and are not silently counted as current production theorems.

The supplied verifier creates a fresh source project, checks dependency revisions, builds, prints all theorem axiom dependencies, replays all five modules, and runs the controls. It refuses to overwrite an existing fresh build directory and treats timeouts as failures. It makes no network requests or installations. With the pinned runtime restored, the documented command is:

```
python scripts/verify_lean.py \
  --toolchain /path/to/Komorebi_Portable_Lean_4.34.0-rc2/toolchain \
  --packages /path/to/Komorebi_Portable_Lean_4.34.0-rc2/workspace/.lake/packages \
  --out verification/reproduction_new
```

The runner was exercised under Linux x86-64. It uses POSIX process groups and filesystem symlinks. Native Windows behavior is not tested. WSL2 with the pinned Linux runtime is the documented route for a Windows machine.

Full proof source, manifests, archived build and replay logs, the static concordance builder, and the current failed runtime restoration status are included. The runtime itself remains an external dependency.

17 Complete active theorem inventory

All names in the following table have prefix `Komorebi.QuantumArithmetic`. File and physical source line identify the exact declaration. The full literal statements and axiom lists are also available in `Lean_Concordance.json` and `Lean_Concordance.csv`.

17.1 `QuantumArithmeticBridge`: 22 theorems

Declaration	Line
`ramanujan12_length`	80
`projector12_shape`	82
`projector12_symmetric`	88
`projector12_idempotent`	92
`projector12_fixes_ramanujan`	97
`projector12_kills_noise`	102
`projector12_recovers_test_signal`	107
`ramanujan12_energy`	112
`noise12_energy`	116
`testSignal12_energy`	120

Declaration	Line
energy_decomposition	124
zeta12_order_certificate	177
zeta12_inverse_certificate	180
ramanujanC12_table	190
ramanujan_dft_table	205
abs_ramanujan_dft_table	217
same_computational_basis_probabilities	231
abs_ramanujan12_energy	235
recovered_mode_probability	243
recovered_total_probability	248
phase_stripping_coprime_probability	253
arithmetic_signal_quantum_bridge	258

17.2 FiniteFourier: 17 theorems

Declaration	Line
coprime_residues_exact	59
root_exact_order	63
normalized_fourier_left	67
normalized_fourier_right	71
exact_projector_transport	79
exact_projector_matches_rational_table	83
exact_state_preparation	87
exact_state_recovery	91
exact_unit_normalized	95
exact_psi_normalized	98
exact_chi_normalized	101
exact_same_initial_probabilities	104
exact_orthogonal_states	108
exact_coprime_mass_psi	112
exact_coprime_mass_chi	116
exact_projector_fixes_psi	120
exact_projector_kills_chi	124

17.3 `SemanticBridge`: 30 theorems

Declaration	Line
`interpret_zero`	19
`interpret_one`	22
`interpret_rat`	25
`interpret_add`	28
`interpret_mul`	32
`interpret_conj`	37
`interpret_pow`	42
`interpret_sum`	48
`interpret_kSum`	54
`complexMatrix_mul`	65
`complexMatrix_adjoint`	70
`complexMatrix_identity`	75
`complexVector_mul`	80
`fourier_unitary_left`	94
`fourier_unitary_right`	99
`projector_transport`	104
`state_preparation`	112
`state_recovery`	117
`interpret_normSq`	122
`interpret_mass`	133
`interpret_coprimeMass`	137
`psi_normalized`	145
`chi_normalized`	149
`unitState_normalized`	153
`same_initial_probabilities`	157
`states_orthogonal`	162
`signed_state_recovers_arithmetic`	166
`unsigned_state_erases_arithmetic`	171
`projector_fixes_psi`	176
`projector_kills_chi`	181

17.4 `Conventions`: 17 theorems

Declaration	Line
`root_is_exponential`	11
`root_power_twelve`	18
`rootPower_interpret`	21
`sqrt_twelve`	26
`sqrt_fortyeight`	34

Declaration	Line
`fourierScale_interpret`	42
`stateScale_interpret`	49
`fourier_entry_formula`	57
`psi_entry_formula`	67
`chi_entry_formula`	71
`unitState_entry_formula`	75
`projector_entry_formula`	80
`selector_idempotent`	84
`selector_selfAdjoint`	91
`projector_idempotent_complex`	98
`projector_selfAdjoint_complex`	106
`complete_phase_separation`	111

17.5 `Publication`: 10 theorems

Declaration	Line
`exact_phase_signal_lookup`	9
`complex_ramanujan_signal`	14
`absolute_signal_lookup`	22
`retained_discarded_orthogonal`	27
`signal_decomposition`	30
`projector_residual_reconstructs`	34
`no_reconstruction_from_initial_probabilities`	40
`recovered_probability_at`	53
`projector_not_injective`	60
`publication_bridge`	68

18 What completion means here

The fixed twelve-dimensional bridge is formally covered in the preserved later checkpoint. This guide and machine concordance expose its proof and trust boundaries.

General-modulus Möbius and divisor-projector theory, polyphase identities, and a general measurement API remain outside this Lean package. The traditional paper supplies separate proofs. Runtime restoration failed, so no fresh replay is claimed.