KOMOREBI

Special Functions through Packet Realization

A consolidated mathematical core

Projector Generated Special Functions I–VIII

Arithmetic packets and positive spectral models

Continuous shape packets and orthogonal refinement

Racah recoupling and q-deformation

Komorebi P.

Working edition 0.1 / 7 October 2026

Editorial scope

This edition gives the eight special-function papers one mathematical entry point. It consolidates repeated definitions and theorem chains into three connected chapters, then supplies a selected visual atlas, a correction register, and an evidence map. The complete reader appends the eight original papers in order. Their original page numbers and statements remain visible, so the correction register must be read alongside them.

This is a first consolidated working edition. It is not a claim that all eight papers have undergone a new exhaustive proof audit or a complete Lean build. The mathematical chapters give concise proofs for elementary identities and explicitly distinguish source results, classical inputs, conditional extensions, and open arithmetic estimates. The correction register records the points exposed by the present bounded review.

The source package preserves the eight release archives, their code, tables, and figures. It also includes all 173 special-function records from the retrieved claim ledger, fresh test results, and the review notes. Claim records are provenance data. They are not interchangeable with Lean proof terms.

Reading route. Read Chapters 1–3 for the unified mathematics. Chapter 4 presents selected figures from the source papers. Chapters 5–6 identify corrections and the exact evidence scope. For full derivations and implementations, follow the source-volume references and the bookmarks in the complete reader.

Contents

Editorial scope **i**

1 An exact packet spine for special functions **1**

1.1 Coefficient packets and ordered differential transport 1

1.2 Bernoulli–Hurwitz refinement 2

1.3 Signed arithmetic transfer and positive energy 3

1.4 Energy asymptotics and the all-moment bridge 4

1.5 Laguerre and finite spectral realizations 5

1.6 The principal cusp from the energy coefficients 6

1.7 The remaining signed arithmetic estimate 7

2 From arithmetic depth to a Gamma shape calculus **8**

2.1 The arithmetic seed and its positive lift 8

2.2 One window and its exact phase-space kernel 9

2.3 Fourier density and additive shape 10

2.4 Polynomial refinement and the common ladder 10

2.5 Shape transfer and its exact contraction spectrum 11

2.6 Coherent smoothing and its Gaussian limit 12

2.7 The signed branch and its retained boundary 12

3 Recoupling packets and their q-deformation **14**

3.1 A single ladder and coproduct convention 14

3.2 Excitation reduction and the two-factor seed 15

3.3 Atlases, recoupling, and recurrence gauges 16

3.4 An exact two-channel deformation 17

3.5 Measurement, entropy, and deformation transport 17

3.6 Tree overlaps, loop closure, and compilation 18

3.7 Scalar bridge and its remaining comparison 19

4 A visual route through the consolidated core **20**

1. Exact coefficient packets . . . 21
2. The arithmetic Gamma scale . . . 22
3. A common Laguerre fixed point . . . 23
4. The exact Gamma–sech window . . . 24
5. Shape refinement . . . 25
6. Finite recoupling . . . 26
7. A deformable two-channel packet . . . 27
8. A retained negative control . . . 28

5 Corrections and proof dependencies **29**

5.1 Endpoint and character conventions . . . 29
5.2 Positive spectral models . . . 29
5.3 Signed rational models . . . 30
5.4 Recoupling and deformation . . . 30
5.5 Retained open statements and review limits . . . 31

6 Source crosswalk and evidence **32**

6.1 The consolidation map . . . 32
6.2 Fresh test run . . . 33
6.3 Formalization scope . . . 33
6.4 Primary-source checks . . . 34

Chapter 1

An exact packet spine for special functions

The first three volumes have a common mathematical spine: exact analysis and synthesis, refinement of an analytic seed, arithmetic scale transfer, a positive moment measure, and its spectral realizations. This chapter consolidates that spine while retaining the distinction between proved coordinate identities, classical asymptotic inputs, and the unresolved signed estimate. Source references name the original volume and claim title. They provide provenance rather than a new certification of every source proof.

1.1 Coefficient packets and ordered differential transport

Let $F(z) = \sum_{n\geq 0} a_n z^n$ be an analytic germ at the origin and let $M \geq 2$ be an integer. Its coefficient packets are

$$F_{M,r}(u) = \sum_{m\geq 0} a_{Mm+r} u^m, \qquad 0 \leq r < M.$$

These are analytic germs in u. This construction is Volume I, "Residue-section functions" and "Exact polyphase reconstruction."

Proposition 1.1 (Exact packet reconstruction and refinement)**.** *Where the defining series converge,*

$$F(z) = \sum_{r=0}^{M-1} z^r F_{M,r}(z^M).$$

For an integer $L \geq 1$, refinement satisfies

$$F_{M,r}(u) = \sum_{\ell=0}^{L-1} u^\ell F_{LM,r+M\ell}(u^L).$$

Proof. The first identity partitions each exponent uniquely as $n = Mm + r$. In the second identity, partition the packet index as $m = Lj + \ell$. Then $Mm + r = LMj + r + M\ell$, and the corresponding term is $a_{LMj+r+M\ell} u^{Lj+\ell}$. Both identities preserve every coefficient. □

Write $\boldsymbol{F}_M = (F_{M,0}, \ldots, F_{M,M-1})^T$. Define the carry matrix by

$$C_M(u)_{0,M-1} = u, \qquad C_M(u)_{r,r-1} = 1 \quad (1 \leq r < M),$$

with all other entries zero, and define

$$\Theta_M = Mu\frac{d}{du}I_M + \mathrm{diag}(0,1,\dots,M-1).$$

The synthesis map $\mathcal{S}_M\boldsymbol{f}(z) = \sum_r z^r f_r(z^M)$ is injective on analytic packet germs. Multiplication and the Euler derivative intertwine as

$$\mathcal{S}_M(C_M\boldsymbol{f}) = z\mathcal{S}_M\boldsymbol{f}, \qquad \mathcal{S}_M(\Theta_M\boldsymbol{f}) = z\frac{d}{dz}\mathcal{S}_M\boldsymbol{f}.$$

Indeed, multiplication advances the residue address and wraps the last address with one factor u; the Euler operator multiplies u^m in address r by $Mm+r$. Consequently

$$[\Theta_M, C_M] = C_M.$$

For every ordered polynomial differential expression $\mathcal{L}(z,\theta)$, where $\theta = z\,d/dz$,

$$\mathcal{L}(z,\theta)F = 0 \quad\Longrightarrow\quad \mathcal{L}(C_M,\Theta_M)\boldsymbol{F}_M = 0.$$

Intertwine each ordered monomial and apply injectivity. This is Volume I, "Intertwining laws," "Weyl commutator," and "Differential lift." The order of operator factors is part of the statement.

1.2 Bernoulli–Hurwitz refinement

For a real translation parameter $a > 0$, define

$$\mathfrak{B}(z;a) = -z\zeta(1-z,a), \qquad \mathfrak{B}_{M,r}(z;a) = -zM^{z-1}\zeta\left(1-z,\frac{a+r}{M}\right).$$

The order variable is $z \in \mathbb{C}$. The removable values are $\mathfrak{B}(0;a) = 1$ and $\mathfrak{B}_{M,r}(0;a) = 1/M$. The pole at $z = 0$ is the only Hurwitz-zeta pole in this variable, so these functions are entire in z.

Proposition 1.2 (Exact Bernoulli reconstruction and refinement)**.** *For integers $M, L \geq 1$ and $0 \leq r < M$,*

$$\mathfrak{B}(z;a) = \sum_{r=0}^{M-1}\mathfrak{B}_{M,r}(z;a), \qquad \mathfrak{B}_{M,r}(z;a) = \sum_{\ell=0}^{L-1}\mathfrak{B}_{LM,r+\ell M}(z;a).$$

For each positive integer n,

$$\mathfrak{B}_{M,r}(n;a) = M^{n-1}B_n\left(\frac{a+r}{M}\right).$$

Proof. For $\mathrm{Re}\,s > 1$, partition the Hurwitz series into residue classes to obtain

$$\sum_{r=0}^{M-1}\zeta\left(s,\frac{a+r}{M}\right) = M^s\zeta(s,a).$$

Meromorphic continuation preserves this identity. Set $s = 1-z$ and multiply by $-zM^{z-1}$. Apply the same identity to the children of a fixed parent to obtain refinement. The integer formula uses the classical special value $\zeta(1-n,x) = -B_n(x)/n$. □

These are Volume I, "Reconstruction," "Parent-child refinement," and "Integer samples." Integer polynomial samples can also be evaluated at $a = 0$, but the entire complex-order family above retains $a > 0$.

Let $b_\ell = \mathfrak{B}_{LM,r+\ell M}(z; a)$ and $\omega_L = e^{2\pi i/L}$. The unitary child transform

$$W_j = \frac{1}{\sqrt{L}} \sum_{\ell=0}^{L-1} \omega_L^{-j\ell} b_\ell$$

has the exact inverse $b_\ell = L^{-1/2} \sum_j \omega_L^{j\ell} W_j$ and preserves $\sum_\ell |b_\ell|^2$. Its DC channel is $W_0 = \mathfrak{B}_{M,r}/\sqrt{L}$. Volume I derives the Lerch closed form, and Volume II derives the local Stieltjes expansion and scalar detail invariants from this same bank. These analytic invariants do not require the positive arithmetic measure introduced next.

1.3 Signed arithmetic transfer and positive energy

Let $q \geq 1$ be an integer and χ a Dirichlet character modulo q. Throughout this section $\beta > 0$ and k is a nonnegative integer. At zero depth, use the moment convention $0^0 = 1$.

Definition 1.3 (Arithmetic transfer and packet energy)**.** For $a > 1$ and $t \in \mathbb{R}$, define the signed transfer

$$C_{a,\beta,\chi}(k, t) = \sum_{n \geq 1} \frac{\mu(n)\chi(n)}{n^{a+it}} (1 - n^{-\beta})^k.$$

For $a > 1/2$, define the positive energy

$$E_{a,\beta,q}(k) = \sum_{\substack{n \geq 1 \\ (n,q)=1}} \frac{\mu(n)^2}{n^{2a}} (1 - n^{-\beta})^{2k}.$$

The signed series is absolutely convergent in its stated range. The energy uses $|\chi(n)|^2 = \mathbf{1}_{(n,q)=1}$ and therefore depends on the modulus rather than the character values.

Proposition 1.4 (Two exact finite-difference coordinates)**.** *For $a > 1$,*

$$C_{a,\beta,\chi}(k, t) = \sum_{j=0}^{k} (-1)^j \binom{k}{j} \frac{1}{L(a + \beta j + it, \chi)}.$$

For $a > 1/2$, set

$$D_q(s) = \frac{\zeta(s)}{\zeta(2s)} \prod_{p \mid q} (1 + p^{-s})^{-1}, \qquad \operatorname{Re} s > 1.$$

Then

$$E_{a,\beta,q}(k) = \sum_{j=0}^{2k} (-1)^j \binom{2k}{j} D_q(2a + \beta j).$$

Proof. The absolutely convergent reciprocal Euler product gives $L(s, \chi)^{-1} = \sum_n \mu(n)\chi(n)n^{-s}$. The squarefree Dirichlet series has Euler factors $1 + p^{-s}$ for $p \nmid q$, so it is $D_q(s)$. Expanding the corresponding powers of $1 - n^{-\beta}$ and interchanging each finite sum with its absolutely convergent series proves both identities. □

The source claims are Volume I, "Möbius form," and Volume II, "Exact zeta-ratio finite difference," repeated in Volume III. Their parallel form does not remove the arithmetic signs from C.

Put

$$w_n = \mu(n)^2 \mathbf{1}_{(n,q)=1} n^{-2a}, \qquad x_n = (1-n^{-\beta})^2, \qquad \nu = \sum_{n\geq 1} w_n \delta_{x_n}.$$

This is a finite positive measure, with atoms in $[0,1)$ and infinitely many distinct positive atoms. In particular the zero atom at $n=1$ is retained. Its moments satisfy $E_k = \int x^k \, d\nu(x)$.

Proposition 1.5 (Positive moment and resolvent laws)**.** *For $m, k \geq 0$,*

$$(-1)^m \Delta^m E_k = \int x^k (1-x)^m \, d\nu(x) \geq 0.$$

Every finite Hankel matrix $(E_{i+j})_{0\leq i,j\leq d}$ is positive definite. The resolvent

$$\mathcal{E}(z) = \int \frac{d\nu(x)}{1-zx}$$

is analytic and zero-free on $\mathbb{C} \setminus [1,\infty)$, and for $|z|<1$ it equals $\sum_{k\geq 0} E_k z^k$.

Proof. Finite differences act on x^k by multiplication by $(x-1)^m$. For a nonzero coefficient vector c, the Hankel quadratic form is

$$\int \left| \sum_{j=0}^{d} c_j x^j \right|^2 d\nu(x) > 0,$$

because a nonzero polynomial cannot vanish at infinitely many distinct atoms. On every compact subset of the slit plane, $|1-zx|$ has a positive lower bound for $x \in [0,1]$, proving analyticity. For real $z<1$ the integrand is positive. For $\operatorname{Im} z > 0$ its imaginary part is

$$\frac{x \operatorname{Im} z}{|1-zx|^2},$$

whose integral is strictly positive; the sign reverses below the real axis. A zero is impossible. □

These are Volume II, "Complete monotonicity," "Strict finite Hankel positivity," and "Zero-free slit plane." Volume III further proves strict positivity of every generalized Hankel minor with strictly increasing nonnegative row and column exponents.

1.4 Energy asymptotics and the all-moment bridge

The classical squarefree counting input is

$$\sum_{\substack{n\leq X \\ (n,q)=1}} \mu(n)^2 = \delta_q X + O_q(\sqrt{X}), \qquad \delta_q = \frac{1}{\zeta(2)} \prod_{p\mid q} \frac{p}{p+1}.$$

The following asymptotic implication is the source result Volume II, "Packet-energy asymptotic," also stated in Volumes I and III. It depends on that classical counting law.

Proposition 1.6 (Source energy law)**.** *Let $a > 1/2$, $\beta > 0$, and $\alpha = (2a-1)/\beta$. Then*

$$E_{a,\beta,q}(k) \sim \frac{\delta_q}{\beta} \Gamma(\alpha) (2k)^{-\alpha}.$$

Proof. The density part of Stieltjes summation is evaluated exactly by

$$\delta_q \int_1^\infty x^{-2a}(1-x^{-\beta})^{2k}\,dx = \frac{\delta_q}{\beta}\mathrm{B}(\alpha, 2k+1).$$

The counting remainder contributes at most a constant times

$$\int_1^\infty \sqrt{x}\left|\frac{d}{dx}\left[x^{-2a}(1-x^{-\beta})^{2k}\right]\right| dx = O_{a,\beta,q}\left(k^{-(2a-1/2)/\beta}\right).$$

This is lower order than $k^{-\alpha}$. The beta-function asymptotic gives the stated leading term. This records the source reduction to the classical counting input, rather than an independent re-audit of that input. □

For $k \geq 1$, normalize the weights $w_n x_n^k / E_k$ into a probability law and put

$$U_k = 2k\, n^{-\beta}, \qquad u_k = k(1-x_n) = U_k - \frac{U_k^2}{4k}.$$

The first coordinate is Volume II's Gamma scaling coordinate, and the second is Volume III's endpoint coordinate.

Proposition 1.7 (All-moment compatibility of the two coordinates)**.** *For every nonnegative integer m,*

$$\mathbb{E}_k[U_k^m] = (2k)^m \frac{E_{a+\beta m/2,\beta,q}(k)}{E_{a,\beta,q}(k)} \longrightarrow \frac{\Gamma(\alpha+m)}{\Gamma(\alpha)}.$$

The moments of u_k have the same limits.

Proof. Multiplication by $n^{-\beta m}$ shifts the exponent $2a$ to $2a+\beta m$, proving the exact identity. Apply the source energy law to numerator and denominator. For $m \geq 1$, $0 \leq u_k \leq U_k$ and

$$0 \leq U_k^m - u_k^m \leq \frac{m}{4k} U_k^{m+1}.$$

The already proved $(m+1)$st moment convergence bounds the expectation on the right by $O(1/k)$. Thus every fixed moment transfers to the endpoint coordinate. □

These limits are the moments of Gamma$(\alpha, 1)$. The classical method of moments, using determinacy of the Gamma law, yields weak convergence as well. The all-moment conclusion is the required bridge to the polynomial and determinant results; bounded-test-function weak convergence alone would not supply it.

1.5 Laguerre and finite spectral realizations

Let $P_m^{(k)}$ be the monic orthogonal polynomial of degree m for the law of u_k. Volume III, "Fixed-degree Laguerre universality," identifies the limit

$$P_m^{(k)}(u) \longrightarrow (-1)^m m! L_m^{(\alpha-1)}(u)$$

coefficientwise and locally uniformly for each fixed m. To see the implication, write orthogonality of $u^m + \sum_{j<m} c_j u^j$ as a linear system in the first $2m$ moments. The limiting Gamma moment matrix is positive definite. The all-moment bridge and continuity of the inverse matrix therefore give convergence

of the coefficients. The classical identification of Gamma orthogonal polynomials supplies the displayed Laguerre formula. Their squared monic norms under the normalized Gamma law are $m!(\alpha)_m$.

At fixed parameters, normalize ν by its mass E_0 and let J_m be the leading $m \times m$ Jacobi matrix of its orthonormal polynomials. The classical Gaussian quadrature theorem gives the source result Volume III, "Moment-exact packet reduction": there are nodes $\xi_j \in (0,1)$ and positive weights λ_j summing to one such that

$$E_r = E_0 \sum_{j=1}^{m} \lambda_j \xi_j^r = E_0 e_1^T J_m^r e_1, \qquad 0 \le r \le 2m-1.$$

The nodes belong to the interval spanned by the measure; they need not equal its discrete atoms. The reduced resolvent

$$\mathcal{E}_m(z) = E_0 \sum_{j=1}^{m} \frac{\lambda_j}{1 - z\xi_j}$$

matches these first $2m$ Taylor coefficients and has poles on $(1, \infty)$. Its spectral weights are positive. In the z coordinate, its residue at $z = 1/\xi_j$ is $-E_0\lambda_j/\xi_j$.

Volume III develops the corresponding fixed-degree Jacobi, kernel, quadrature and shifted-Hankel limits. They depend on the all-moment implication above and classical orthogonal-polynomial identities. The principal cusp has a direct Abelian proof from the energy coefficients, supplied next.

1.6 The principal cusp from the energy coefficients

The following Abelian argument replaces the informal discrete-density passage in Volume III, "Principal edge cusp." It uses only the energy law already stated above.

Lemma 1.8 (An elementary square-root Abelian passage)**.** *Let $c_k \ge 0$ and $c_k \sim Ak^{-3/2}$, where $A > 0$. For $F(z) = \sum_{k\ge 0} c_k z^k$, one has $F(1) < \infty$ and*

$$F(1) - F(z) \sim 2\sqrt{\pi} A \sqrt{1-z} \qquad (z \uparrow 1 \text{ through real values}).$$

Proof. Put $t = -\log z$ and $S(t) = \sum_{k\ge 1} k^{-3/2}(1 - e^{-tk})$. The function

$$f(u) = u^{-3/2}(1 - e^{-u}) = u^{-1/2} \int_0^1 e^{-su}\, ds$$

is positive and decreasing. It is integrable at both endpoints. Thus

$$\int_t^\infty f(u)\, du \le t \sum_{k\ge 1} f(tk) \le \int_0^\infty f(u)\, du = 2\sqrt{\pi},$$

where integration by parts gives the last integral. Since $\int_0^t f(u)du \le 2\sqrt{t}$, it follows that $S(t) = 2\sqrt{\pi}\sqrt{t} + O(t)$.

Write $r_k = c_k - Ak^{-3/2}$ for $k \ge 1$. For each $\varepsilon > 0$, choose K such that $|r_k| \le \varepsilon k^{-3/2}$ for $k \ge K$. Then

$$|F(1) - F(e^{-t}) - AS(t)| \le t \sum_{1\le k<K} k|r_k| + \varepsilon S(t).$$

Divide by $\sqrt{t}$, let $t \downarrow 0$, and then let $\varepsilon \downarrow 0$. Finally, $t \sim 1 - z$. The coefficient hypothesis also guarantees summability. □

For the principal energy $a = \beta = 2$, $q = 1$, the coefficient law gives $A = 3/(2^{5/2}\pi^{3/2})$. Therefore the source cusp is

$$\mathcal{E}(1) - \mathcal{E}(z) \sim \frac{3}{2^{3/2}\pi}\sqrt{1-z}.$$

This proof supplies the missing local implication. The actual coefficient asymptotic warrants an $o(\sqrt{1-z})$ remainder; it does not by itself supply an $O(1-z)$ remainder. The Abelian lemma received a separate neutral review in this consolidation, conditional on the source energy asymptotic.

1.7 The remaining signed arithmetic estimate

For the principal parameters $a = \beta = 2$ and $q = 1$, write

$$c_k = \sum_{n\geq 1} \frac{\mu(n)}{n^2}(1-n^{-2})^k, \qquad E_k = \sum_{n\geq 1} \frac{\mu(n)^2}{n^4}(1-n^{-2})^{2k}.$$

The positive theory gives

$$E_k \sim \frac{3}{2^{5/2}\pi^{3/2}} k^{-3/2}.$$

The classical Báez-Duarte theorem (Theorem 1.1, 2005, checked in the published source) is the external input

$$\text{RH} \quad \Longleftrightarrow \quad c_k = O_\varepsilon(k^{-3/4+\varepsilon}) \quad \text{for every } \varepsilon > 0.$$

It is cited in Volume I, "Coherence form of the Báez-Duarte criterion," and Volume II, "Normalized principal coherence criterion." Consequently, defining $\Gamma_k = |c_k|^2/E_k$,

$$\text{RH} \quad \Longleftrightarrow \quad \Gamma_k = O_\varepsilon(k^{\varepsilon}) \quad \text{for every } \varepsilon > 0.$$

Indeed, square the amplitude bound and divide by the energy law; in the reverse direction multiply by the energy law and take square roots. The arbitrary positive exponent can be renamed in either direction.

This is an equivalence with an open arithmetic estimate. The positive moment measure determines the normalization and supports exact finite models. The estimate still concerns cancellation of the signs $\mu(n)$ inside the principal transfer. Volume II's cross-character decoherence concerns a different Gram entry with coefficients $\mu(n)^2\chi(n)\overline{\psi(n)}$ for distinct characters modulo one fixed modulus. Its cancellation theorem does not estimate c_k.

The common construction therefore separates three achievements precisely: exact packet identities, positive spectral and scaling laws, and a signed coherence target whose truth is equivalent to RH. Consolidation preserves all three without treating the third as a consequence of the first two.

Chapter 2

From arithmetic depth to a Gamma shape calculus

Volumes IV–VI follow one continuous route. A signed number-basis amplitude has a positive time-averaged Gram kernel. Large depth fixes the shape and centering of a log-Gamma window. Exact transforms of that window then produce a phase-space kernel, orthogonal polynomial coordinates, shape refinement, and coherent smoothing. This chapter introduces the shared objects once. Source references identify the original volume and claim title. Elementary identities are rederived here. The arithmetic asymptotic and classical orthogonal-polynomial inputs retain their original hypotheses and evidence scope.

2.1 The arithmetic seed and its positive lift

Retain Chapter 1's signed transfer $C_{a,\beta,\chi}(k,t)$, with $\beta > 0$, $a > 1$, and χ modulo q. Put $d_n = 1 - n^{-\beta}$. The zero-depth convention retains $n = 1$. We now add a two-depth positive kernel. The positive companion is defined already for $a > 1/2$:

$$\mathcal{K}_q(k,\ell;\tau) = \sum_{\substack{n\geq 1\\(n,q)=1}} \mu(n)^2 n^{-2a-i\tau} d_n^{k+\ell}.$$

Proposition 2.1 (Time-average Gram lift)**.** *For $a > 1$, the long-time cross-correlation satisfies*

$$\lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} C_{a,\beta,\chi}(k,t+\tau)\overline{C_{a,\beta,\chi}(\ell,t)}\,dt = \mathcal{K}_q(k,\ell;\tau).$$

For $a > 1/2$, the kernel $\mathcal{K}_q(k,\ell;t-s)$ is positive definite on depth-time pairs.

Proof. Absolute convergence for $a > 1$ permits expansion of the correlation into a double series and passage through the time average. A term with indices n, m contains $e^{-it(\log n - \log m)}$. Its mean is one for $n = m$ and tends to zero otherwise. This leaves the displayed squarefree diagonal. For positivity, set $v_{k,t}(n) = \mu(n)\chi(n)n^{-a}d_n^k n^{-it}$. These vectors are square summable for $a > 1/2$, and the kernel is $\sum_n v_{k,t}(n)\overline{v_{\ell,s}(n)}$. Every finite kernel matrix is a Gram matrix. □

This is Paper IV, "Exact time-average identity" and "Positive-definite packet kernel." Time averaging retains the positive diagonal power and removes the off-diagonal frequency interference. It does not bound the original signed amplitude at a particular time.

Define

$$\alpha = \frac{2a-1}{\beta}, \qquad \delta_q = \frac{1}{\zeta(2)} \prod_{p \mid q} \frac{p}{p+1}.$$

The fixed-modulus squarefree counting input is $\sum_{n \le X,(n,q)=1} \mu(n)^2 = \delta_q X + O_q(X^{1/2})$. Paper IV, "Uniform complex packet asymptotic," uses partial summation to obtain

$$\mathcal{K}_q(k,\ell;\tau) \sim \frac{\delta_q}{\beta} \Gamma(\alpha + i\tau/\beta)(k+\ell)^{-\alpha - i\tau/\beta}.$$

Paper V, "Joint arithmetic phase-space limit," combines the diagonal normalizations. Uniformly when k/ℓ stays in a compact subset of $(0,\infty)$ and τ stays in a compact real interval, with $h = \log(k/\ell)$ and $\eta = \tau/\beta$,

$$e^{i\eta \log(2\sqrt{k\ell})} \frac{\mathcal{K}_q(k,\ell;\tau)}{\sqrt{\mathcal{K}_q(k,k;0)\mathcal{K}_q(\ell,\ell;0)}} \longrightarrow \frac{\Gamma(\alpha+i\eta)}{\Gamma(\alpha)} \operatorname{sech}^{\alpha+i\eta}(h/2). \tag{2.1}$$

The positive domain is $a > 1/2$. The signed series above and its absolute time-average derivation use the stronger condition $a > 1$.

2.2 One window and its exact phase-space kernel

For an independent shape $\alpha > 0$, define

$$g_\alpha(y) = \Gamma(\alpha)^{-1/2} \exp\left(\frac{\alpha y}{2} - \frac{e^y}{2}\right), \qquad y \in \mathbb{R}.$$

The substitution $u = e^y$ gives $\int |g_\alpha|^2 dy = \Gamma(\alpha)^{-1} \int_0^\infty u^{\alpha-1} e^{-u} du = 1$. Thus $|g_\alpha|^2$ is a log-Gamma density. Its shape is fixed by the arithmetic limit in (2.1), but the exact window theory makes sense for every positive shape.

Use a second shape γ to distinguish shape composition from the arithmetic scale exponent. The centered cross-ambiguity convention is

$$A_{\alpha,\gamma}(h,\eta) = \int_{\mathbb{R}} g_\alpha(y + h/2) g_\gamma(y - h/2) e^{i\eta y}\, dy.$$

Theorem 2.2 (Cross-shape Gamma–sech identity)**.** *For $\alpha, \gamma > 0$ and real h, η, with $\lambda = (\alpha+\gamma)/2$,*

$$A_{\alpha,\gamma}(h,\eta) = e^{(\alpha-\gamma)h/4} \frac{\Gamma(\lambda + i\eta)}{\sqrt{\Gamma(\alpha)\Gamma(\gamma)}} \operatorname{sech}^{\lambda+i\eta}(h/2). \tag{2.2}$$

Proof. Multiplication of the shifted windows leaves the constant prefactor $e^{(\alpha-\gamma)h/4}/\sqrt{\Gamma(\alpha)\Gamma(\gamma)}$ and the integrand $\exp((\lambda+i\eta)y - e^y \cosh(h/2))$. The substitution $u = e^y \cosh(h/2)$ evaluates the remaining integral as $\Gamma(\lambda+i\eta)\cosh^{-\lambda-i\eta}(h/2)$. □

This is Paper VI, "Cross-shape Gamma–sech formula." At equal shapes it is Paper V, "Exact ambiguity formula," and precisely the arithmetic limit. The Gamma function has no zeros and $\cosh(h/2) > 0$ on the real line, so this cross-ambiguity is nonzero on real phase space. At $h = \eta = 0$ its value is

$$F(\alpha,\gamma) = \frac{\Gamma((\alpha+\gamma)/2)}{\sqrt{\Gamma(\alpha)\Gamma(\gamma)}}.$$

Cauchy–Schwarz gives $0 < F \le 1$, with equality only for equal shapes. Indeed, the ratio of the two positive windows is a constant times $e^{(\alpha-\gamma)y/2}$ and can be constant only when $\alpha = \gamma$.

2.3 Fourier density and additive shape

Under the unitary convention $\widehat{f}(\omega) = (2\pi)^{-1/2} \int f(y) e^{-i\omega y} dy$, the same Gamma integral gives

$$\widehat{g}_\alpha(\omega) = \frac{2^{\alpha/2 - i\omega} \Gamma(\alpha/2 - i\omega)}{\sqrt{2\pi \Gamma(\alpha)}}.$$

Its squared modulus is the symmetric Meixner density

$$\rho_\alpha(\omega) = \frac{2^{\alpha-1}}{\pi \Gamma(\alpha)} |\Gamma(\alpha/2 + i\omega)|^2.$$

The Fourier autocorrelation identity and the zero-modulation ambiguity slice give

$$\int_{\mathbb{R}} \rho_\alpha(\omega) e^{ih\omega} d\omega = \operatorname{sech}^\alpha(h/2).$$

Consequently Fourier uniqueness proves the shape convolution law

$$\rho_{s+t} = \rho_s * \rho_t, \qquad s, t > 0.$$

These are Paper V, "Fourier transform of the log-Gamma packet" and "Exact characteristic function," followed by Paper VI, "Meixner shape semigroup."

The cross-ambiguity energy also separates exactly:

$$\frac{|A_{\alpha,\gamma}(h,\eta)|^2}{2\pi} = p_{\alpha,\gamma}(h) \rho_{\alpha+\gamma}(\eta), \qquad p_{\alpha,\gamma}(h) = \frac{\Gamma(\alpha+\gamma) e^{(\alpha-\gamma)h/2}}{2^{\alpha+\gamma} \Gamma(\alpha)\Gamma(\gamma)} \operatorname{sech}^{\alpha+\gamma}(h/2). \tag{2.3}$$

Direct cancellation of constants proves the formula. Setting $u = e^h/(1+e^h)$ transforms $p_{\alpha,\gamma}(h)dh$ into the Beta(α,γ) probability law. Both factors are normalized. This is Paper VI, "Exact cross-spectrogram factorization."

2.4 Polynomial refinement and the common ladder

The classical Meixner–Pollaczek polynomial input is specified by its generating function, rather than by an implicit choice of sign or phase:

$$\sum_{n \geq 0} P_{n,s}(x) z^n = (1 - iz)^{-s/2 + ix} (1 + iz)^{-s/2 - ix}, \qquad |z| < 1.$$

Let $(s)_n = \Gamma(s+n)/\Gamma(s)$ and put

$$p_{n,s}(x) = \sqrt{\frac{n!}{(s)_n}} P_{n,s}(x).$$

Classical orthogonality makes these a normalized polynomial basis in $L^2(\rho_s(x)dx)$. The degree n and shape s are separate coordinates.

Theorem 2.3 (Normalized two-child refinement)**.** *For $\alpha, \gamma > 0$,*

$$p_{n,\alpha+\gamma}(x+y) = \sum_{r=0}^{n} \sqrt{\binom{n}{r} \frac{(\alpha)_r (\gamma)_{n-r}}{(\alpha+\gamma)_n}} p_{r,\alpha}(x) p_{n-r,\gamma}(y).$$

The squared coefficients sum to one.

Proof. Multiplying the generating functions with shapes α, γ produces the one with shape $\alpha+\gamma$ and argument $x+y$. Coefficient comparison gives the unnormalized addition formula, and the displayed normalizations give the square roots. The coefficient identity $\sum_r \binom{n}{r}(\alpha)_r(\gamma)_{n-r} = (\alpha+\gamma)_n$ follows by comparing coefficients in $(1-z)^{-\alpha}(1-z)^{-\gamma} = (1-z)^{-\alpha-\gamma}$. □

Paper VI, "Two-child packet refinement," interprets the coefficient squares as a beta-binomial law. For many children they become the Dirichlet-multinomial coefficients

$$C^{\mathbf{s}}_{n,\mathbf{r}} = \sqrt{\frac{n!\prod_j (s_j)_{r_j}}{(\sum_j s_j)_n \prod_j r_j!}}, \qquad \sum_j r_j = n.$$

Multiplication of generating functions proves this extension. Intermediate Pochhammer factors cancel along every binary refinement tree, establishing the coassociativity recorded in Paper VI.

There is a complementary position basis. Classical Laguerre orthogonality gives

$$L^{\mathrm{packet}}_{n,\alpha}(y) = \sqrt{\frac{n!}{\Gamma(n+\alpha)}} e^{\alpha y/2 - e^y/2} L_n^{(\alpha-1)}(e^y).$$

Paper V, "Laguerre–Meixner Fourier duality," evaluates its transform as

$$\widehat{L^{\mathrm{packet}}_{n,\alpha}}(\omega) = i^n \widehat{g}_\alpha(\omega) p_{n,\alpha}(\omega).$$

The phase i^n is part of the identity. The polynomial frequency basis and the Fourier-transformed position basis therefore differ by degree-dependent phases.

On an abstract orthonormal basis e_n, define

$$K_0 e_n = (n+\alpha/2)e_n, \quad K_+ e_n = \sqrt{(n+1)(n+\alpha)}e_{n+1}, \quad K_- e_n = \sqrt{n(n+\alpha-1)}e_{n-1}.$$

On its finite linear span, direct calculation gives

$$[K_0, K_+] = K_+, \quad [K_0, K_-] = -K_-, \quad [K_-, K_+] = 2K_0,$$

and the Casimir is $\frac{\alpha}{2}(\frac{\alpha}{2}-1)I$. The Laguerre multiplication recurrence has matrix $2K_0 - K_+ - K_-$; the real polynomial Meixner frequency recurrence has matrix $(K_+ + K_-)/2$. These are two coordinate realizations of one ladder. A Fourier comparison must retain the degree phases above. This is Paper V, "$\mathfrak{su}(1,1)$ relations."

2.5 Shape transfer and its exact contraction spectrum

Let X_s have law ρ_s and let Y_t be an independent increment with law ρ_t. Shape convolution gives $X_s + Y_t \sim \rho_{s+t}$. Conditioning the refinement identity on X_s removes all positive-degree increment polynomials by orthogonality. Thus

$$\mathbb{E}[p_{n,s+t}(X_s+Y_t) \mid X_s] = \sigma_n(s,t) p_{n,s}(X_s), \qquad \sigma_n(s,t) = \sqrt{\frac{(s)_n}{(s+t)_n}}.$$

For the transfer $T_{s,t} : L^2(\rho_{s+t}) \to L^2(\rho_s)$ given by $T_{s,t}f(x) = \int f(x+y)\rho_t(y)dy$, these are its singular values in the classical complete polynomial bases. They satisfy

$$\sigma_n(s,t)\sigma_n(s+t,u) = \sigma_n(s,t+u), \qquad \sigma_n(s,t) \sim \sqrt{\frac{\Gamma(s+t)}{\Gamma(s)}} n^{-t/2}.$$

The transfer is compact for every $t > 0$ and belongs to S_p precisely when $pt > 2$. Its mean-zero operator norm is $\sqrt{s/(s+t)}$. These are Paper VI, "Polynomial martingale law" and "Cocycle and compactness." The completeness input matters for a statement about the entire operator, rather than only its polynomial restriction.

2.6 Coherent smoothing and its Gaussian limit

Take the inner product conjugate-linear in its first argument. The centered Weyl states are

$$\psi_{\alpha,u,\eta}(y) = e^{i\eta(y-u/2)} g_\alpha(y-u).$$

Their exact overlap is

$$\langle \psi_{\alpha,u,\eta}, \psi_{\gamma,v,\nu} \rangle = e^{i(\nu u - \eta v)/2} A_{\alpha,\gamma}(v-u, \nu-\eta).$$

The Moyal identity for the normalized window gives the continuous tight frame

$$\frac{1}{2\pi} \int_{\mathbb{R}^2} |\langle \psi_{\alpha,u,\eta}, f \rangle|^2 du\, d\eta = \|f\|_2^2.$$

Its coherent transform has a reproducing kernel equal to this overlap. These are Paper VI, "Cross-shape coherent overlap" and "Continuous tight frame."

The lower symbol of the associated anti-Wick operator is convolution with

$$B_\alpha(h,\eta) = \frac{|A_{\alpha,\alpha}(h,\eta)|^2}{2\pi} = p_\alpha(h)\rho_{2\alpha}(\eta), \qquad p_\alpha(h) = \frac{\Gamma(\alpha+1/2)}{2\sqrt{\pi}\Gamma(\alpha)} \operatorname{sech}^{2\alpha}(h/2).$$

Its ordinary two-dimensional Fourier multiplier is

$$\widehat{B}_\alpha(\xi, x) = \frac{|\Gamma(\alpha + i\xi)|^2}{\Gamma(\alpha)^2} \operatorname{sech}^{2\alpha}(x/2).$$

It is strictly positive on real dual space and tends to zero at infinity. Therefore the Berezin convolution is injective on L^2 but has an unbounded inverse on its range, as Paper VI states in "Injectivity and ill-posed inversion." Exact recoverability is not a stability bound.

For independent $H_\alpha \sim p_\alpha$ and $\Omega_\alpha \sim \rho_{2\alpha}$, the standardized variables

$$Z_\alpha = \sqrt{\alpha/2}\, H_\alpha, \qquad W_\alpha = \sqrt{2/\alpha}\, \Omega_\alpha$$

converge to independent standard Gaussians. For position this follows from the Gamma-ratio characteristic function with argument $\xi\sqrt{\alpha/2}$; for frequency it follows from $\operatorname{sech}^{2\alpha}(x/\sqrt{2\alpha}) \to e^{-x^2/2}$. Paper VI, "Gaussian marginal limits," proves this distributional limit. Its stronger "Heat-operator limit" also uses controlled moments and multiplier derivatives to assert Schwartz-seminorm convergence of the rescaled convolution to $e^{\Delta/2}$. That stronger source claim is not inferred here solely from convergence in distribution or a finite-grid plot.

2.7 The signed branch and its retained boundary

The positive shape calculus does not replace the signed arithmetic problem. For $a > 1$ the signed generating function is

$$\mathcal{C}_{a,\beta,\chi}(z,t) = \sum_{n \geq 1} \frac{\mu(n)\chi(n) n^{-a-it}}{1 - z(1-n^{-\beta})}.$$

Paper IV continues it off the physical cut and proves

$$(1-z)\mathcal{C}_{a,\beta,\chi}(z,t)+z\mathcal{C}_{a+\beta,\beta,\chi}(z,t)=L(a+it,\chi)^{-1}.$$

Its nonzero Hankel determinant supplies a unique rational Padé matching function with degree bounds $\deg P \le m-1$, $\deg Q \le m$ and $Q(0)=1$, preserving the first $2m$ Taylor coefficients. An ordinary m-term simple-packet decomposition additionally follows when $\deg Q = m$ and Q has m simple zeros. This chapter makes no universal claim of full denominator degree, nonbreakdown, physical-interval signed nodes, or stable inversion.

The principal signed coherence target and its exact RH equivalence are stated once in Chapter 1. The shape windows, convolution laws, refinement coefficients, and coherent frames provide positive structure; that open target still concerns cancellation of the original Möbius signs.

Chapter 3

Recoupling packets and their q-deformation

The recoupling papers share one mathematical chain: a tensor excitation space is reduced to a lowest-weight multiplicity space, intermediate Casimirs select coordinate atlases, and their overlap supplies both reversible amplitudes and a measured stochastic channel. Four-factor tree atlases add a loop identity, and every finite overlap admits a rotation compiler. This chapter presents that chain once, with the ordinary theory as the specialization $q = 1$.

The direct finite identities are separated from the classical representation-theoretic inputs. The name Racah records the classical identification of a recoupling matrix. An explicit comparison with a scalar polynomial requires additional parameter, normalization, and phase data.

3.1 A single ladder and coproduct convention

The ordinary ladder in Chapter 2 is recovered with $\kappa = \alpha/2$ and $q = 1$. In this chapter, q denotes the real deformation coordinate rather than the integer character modulus of Chapters 1–2. Let $0 < q \leq 1$ and let every shape κ_r be positive. Define

$$[x]_q = \frac{q^x - q^{-x}}{q - q^{-1}} \quad (q \neq 1), \qquad [x]_1 = x.$$

Equivalently, for $q = e^{-\theta}$, $[x]_q = \sinh(x\theta)/\sinh\theta$. In particular, $[x]_q > 0$ for $x > 0$.

Definition 3.1 (Positive shape ladder)**.** On the algebraic finite-support span of an orthonormal basis $\{|n\rangle : n \geq 0\}$, set

$$K_0|n\rangle = (n + \kappa)|n\rangle,$$

$$K_+|n\rangle = \sqrt{[n+1]_q[2\kappa + n]_q}\,|n+1\rangle, \qquad K_-|n\rangle = \sqrt{[n]_q[2\kappa + n - 1]_q}\,|n-1\rangle,$$

where $K_-|0\rangle = 0$. Tensor factors retain their declared order. Their coproduct is

$$\Delta K_0 = K_0 \otimes I + I \otimes K_0, \qquad \Delta K_\pm = K_\pm \otimes q^{K_0} + q^{-K_0} \otimes K_\pm.$$

Source: VII, "Positive shape packet"; VIII, "Local q-algebra" and "Coassociative packet routing." Evidence: direct algebraic identities.

The ladder obeys $[K_0, K_\pm] = \pm K_\pm$ and $[K_-, K_+] = [2K_0]_q$. The second identity follows from

$$[x+1]_q[y]_q - [x]_q[y-1]_q = [x+y]_q.$$

The coproduct preserves these relations: the mixed commutators cancel using $q^{K_0} K_\pm q^{-K_0} = q^{\pm 1} K_\pm$, and the diagonal terms use $[x+y]_q = [x]_q q^y + q^{-x}[y]_q$. Since $q^{\Delta K_0} = q^{K_0} \otimes q^{K_0}$, either order of a three-factor iteration gives

$$K_\pm \otimes q^{K_0} \otimes q^{K_0} + q^{-K_0} \otimes K_\pm \otimes q^{K_0} + q^{-K_0} \otimes q^{-K_0} \otimes K_\pm.$$

Thus the displayed routing is coassociative.

The Casimir is

$$C_q = [K_0]_q[K_0 - 1]_q - K_+ K_-.$$

Direct substitution on $|n\rangle$ yields $C_q = [\kappa]_q[\kappa - 1]_q I$. The identity $[x+1]_q[x]_q - [x]_q[x-1]_q = [2x]_q$ also shows that C_q commutes with the ladder generators. For an intermediate contiguous tensor block, this centrality supplies a real symmetric block Casimir commuting with the total lowering map. Contiguous blocks are sufficient for every planar tree used below.

3.2 Excitation reduction and the two-factor seed

For $m \geq 2$ ordered factors and $N \geq 0$, define

$$H_N^{(m)} = \operatorname{span}\{|n_1, \ldots, n_m\rangle : \textstyle\sum_r n_r = N\}, \qquad M_N^{(m)} = \ker(K_-^{[m]}|_{H_N^{(m)}}).$$

These are finite spaces even though the local ladders are infinite.

Proposition 3.2 (Multiplicity dimension)**.**

$$\dim H_N^{(m)} = \binom{N+m-1}{m-1}, \qquad \dim M_N^{(m)} = \binom{N+m-2}{m-2}.$$

Proof. The first count is the weak-composition count. For $N > 0$, associate to each target composition t of $N-1$ the source $t + e_m$. Its last-factor lowering coefficient is nonzero. Every earlier-factor lowering from that source has a larger last coordinate than t. Ordering by this coordinate gives a triangular square submatrix with nonzero diagonal. The lowering map is onto, and rank-nullity gives the second count. At $N = 0$, the lowering map has zero target and both dimensions are one. □

Source: VII, "Multiplicity dimension"; VIII, "Multiplicity dimension." Evidence: direct finite proof, using VIII's explicit rank argument.

Let Q_N be an orthonormal frame for $M_N^{(m)}$. A Casimir acting on a contiguous intermediate block restricts to $Q_N^{\mathsf{T}} C_A Q_N$. The ratio of full to reduced dimension is $(N+m-1)/(m-1)$. This is an exact dimensional reduction. It does not assert a universal floating-point accuracy or runtime bound.

The two-factor kernel has dimension one. For $|\Omega_N\rangle = \sum_{n=0}^{N} c_n |n, N-n\rangle$, the lowering equation gives

$$\frac{c_{n+1}}{c_n} = -q^{-(N-1+\kappa_1+\kappa_2)} \sqrt{\frac{[N-n]_q[2\kappa_2 + N - n - 1]_q}{[n+1]_q[2\kappa_1 + n]_q}}.$$

For example, writing $\lambda = q^{-(N-1+\kappa_1+\kappa_2)}$, a normalized solution is

$$c_n = Z_N^{-1/2}(-\lambda)^n \sqrt{\frac{[N]_q!}{[n]_q![N-n]_q!}\frac{\prod_{r=0}^{n-1}[2\kappa_2+N-n+r]_q}{\prod_{r=0}^{n-1}[2\kappa_1+r]_q}}.$$

Here empty products and $[0]_q!$ are one, and Z_N is the positive sum of the displayed squared unnormalized coefficients. The ratio proves the formula. The scalar q-Hahn identification of general coupled coefficients is a classical representation-theoretic input.

Source: VIII, "Explicit q-Hahn lowest packet." Evidence: direct finite recurrence and normalization.

3.3 Atlases, recoupling, and recurrence gauges

For three factors, $\dim M_N^{(3)} = N+1$. Choose orthonormal eigenbases L_N and R_N of the restricted pair Casimirs C_{12} and C_{23}, with columns expressed in the same multiplicity frame. Define the signed overlap

$$\mathcal{R}_N = L_N^{\mathsf{T}} R_N.$$

Theorem 3.3 (Finite reconstruction)**.**

$$\mathcal{R}_N^{\mathsf{T}}\mathcal{R}_N = \mathcal{R}_N\mathcal{R}_N^{\mathsf{T}} = I_{N+1}.$$

Consequently, either atlas analyzes and synthesizes every vector in the same multiplicity space exactly.

Proof. Both basis matrices are orthogonal in the common frame. Multiplying their overlap and its transpose gives the identity. □

Source: VII, "Perfect recoupling reconstruction"; VIII, "Exact analysis and synthesis." Evidence: direct linear algebra.

The usual positive-series decomposition gives the intermediate pair eigenvalues

$$\lambda_j^{ab} = [\kappa_a+\kappa_b+j]_q[\kappa_a+\kappa_b+j-1]_q, \qquad 0 \le j \le N.$$

Their occurrence and the Racah/q-Racah identification are classical representation-theoretic inputs cited by VII and VIII. The release reconstructs the matrices and checks their spectra in bounded examples. It does not supply the full scalar parameter and normalization map in the symmetric q convention.

Proposition 3.4 (Recurrence with an explicit gauge)**.** *Suppose the opposing Casimir*

$$J = \mathcal{R}_N \operatorname{diag}(\lambda_m^{23})\mathcal{R}_N^{\mathsf{T}}$$

is irreducibly tridiagonal in its prescribed label order. Its entries give

$$J_{j,j-1}\mathcal{R}_N(j-1,m) + J_{jj}\mathcal{R}_N(j,m) + J_{j,j+1}\mathcal{R}_N(j+1,m) = \lambda_m^{23}\mathcal{R}_N(j,m),$$

with outside endpoint terms omitted. A row-sign gauge can make every adjacent coefficient positive.

Proof. The recurrence is the matrix eigen-equation. Choose $s_0 = 1$ and $s_{j+1} = s_j \operatorname{sign}(J_{j,j+1})$. For $S = \operatorname{diag}(s_j)$, the matrix SJS has adjacent entries $|J_{j,j+1}|$, and the rephased overlap is $S\mathcal{R}_N$. □

Source: VII, "Finite three-term packet recurrence"; VIII, "Two intermediate q-Casimir atlases." Evidence: direct implication conditional on the cited irreducible-tridiagonal input.

The positivity gauge need not equal the gauge making the first row and column of $\mathcal{R}_N$ positive. The signed coefficients in an explicitly printed matrix must therefore be retained unless its basis is rephased.

3.4 An exact two-channel deformation

Take $\kappa_1 = \kappa_2 = \kappa_3 = 1/2$ and $N = 1$. In the ordered basis $|100\rangle, |010\rangle, |001\rangle$, the lowest-weight plane is

$$qx_1 + x_2 + q^{-1}x_3 = 0.$$

Set $S = 1 + q^2$ and $T = 1 + q^2 + q^4$. The following normalized columns give two pair-Casimir atlases:

$$\ell_0 = \frac{(q, 1, -qS)^{\mathsf{T}}}{\sqrt{ST}}, \quad \ell_1 = \frac{(1, -q, 0)^{\mathsf{T}}}{\sqrt{S}},$$

$$r_0 = \frac{(S, -q^3, -q^2)^{\mathsf{T}}}{\sqrt{ST}}, \quad r_1 = \frac{(0, 1, -q)^{\mathsf{T}}}{\sqrt{S}}.$$

Every column lies in the plane. Within each pair the columns are orthonormal. Direct substitution into the pair Casimirs gives the ordered eigenvalues $0, q + q^{-1}$. Their overlap is

$$\mathcal{R}_q = \begin{pmatrix} c_q & s_q \\ s_q & -c_q \end{pmatrix}, \qquad c_q = \frac{q}{1+q^2}, \qquad s_q = \frac{\sqrt{1+q^2+q^4}}{1+q^2}.$$

Thus the displayed transform has both a matrix derivation and an exact orthogonality check. It tends to

$$\mathcal{R}_1 = \begin{pmatrix} 1/2 & \sqrt{3}/2 \\ \sqrt{3}/2 & -1/2 \end{pmatrix} \quad \text{as } q \to 1, \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{as } q \downarrow 0.$$

At $q = 1$, its opposing Casimir has adjacent entry $-\sqrt{3}/2$, illustrating why the recurrence gauge matters.

Source: VIII, "Closed two-channel q-Racah transform"; VII, "One excitation as a shape beam splitter." Evidence: direct finite calculation.

3.5 Measurement, entropy, and deformation transport

Theorem 3.5 (Measured recoupling channel)**.** *For any real orthogonal overlap $\mathcal{R}$, the matrix $P_{jm} = |\mathcal{R}_{jm}|^2$ is doubly stochastic. For every probability row vector p, its measured output pP satisfies*

$$H(pP) \geq H(p), \qquad H(p) = -\sum_j p_j \log p_j,$$

with $0 \log 0 = 0$.

Proof. Row and column sums are the squared norms of the orthogonal rows and columns. For each column m, concavity of $h(x) = -x \log x$ gives $h(\sum_j P_{jm} p_j) \geq \sum_j P_{jm} h(p_j)$, because that column sums to one. Sum over m and use the row sums. □

Source: VII, "Bistochastic Racah kernel" and "Entropy monotonicity"; VIII, "Bistochastic q-Racah channel." Evidence: direct finite proof.

The channel describes a distribution diagonal in one atlas, followed by measurement in the other. The coherent overlap retains its inverse $\mathcal{R}^{\mathsf{T}}$, while the measured distribution has discarded phase information.

For the exact two-channel example, the smaller row probability is $p(q) = q^2/(1+q^2)^2$, increasing from zero to $1/4$. Its binary row entropy increases with q. The channel eigenvalues are 1 and $2p(q) - 1$, so its

absolute spectral mixing gap is $2p(q)$. These statements concern this model. VIII's higher-excitation negative control prevents a universal monotonicity claim as q varies.

For a differentiable, continuously gauged overlap $\mathcal{R}(\theta)$, define

$$A(\theta) = \mathcal{R}(\theta)^{\mathsf{T}} \partial_\theta \mathcal{R}(\theta).$$

Differentiating $\mathcal{R}^{\mathsf{T}}\mathcal{R} = I$ proves

$$A^{\mathsf{T}} = -A, \qquad \partial_\theta \mathcal{R} = \mathcal{R}A.$$

The coefficient $g(\theta) = \frac{1}{2}\|A(\theta)\|_F^2$ records squared speed in the declared parameter and is invariant under constant orthogonal atlas changes. A parameter-dependent gauge adds its own derivative terms. Discontinuous sign choices must be aligned before numerical differentiation.

Source: VIII, "Monotone one-excitation mixing," "No universal entropy monotonicity in q," and "Skew deformation generator." Evidence: direct two-channel and tangent identities; bounded numerical negative control.

3.6 Tree overlaps, loop closure, and compilation

Four factors have $\binom{N+2}{2}$ lowest-weight coordinates. The five planar trees are

$$(((12)3)4), \quad ((1(23))4), \quad (1((23)4)), \quad (1(2(34))), \quad ((12)(34)).$$

Their intermediate contiguous blocks are nested or disjoint. The corresponding Casimirs commute, so each tree admits an orthonormal joint eigenbasis in the same multiplicity space.

Theorem 3.6 (Common-space loop identity)**.** *For complete orthonormal tree frames $B_0, \dots, B_4$, define $U_{r,r+1} = B_r^{\mathsf{T}} B_{r+1}$, with indices modulo five. Then*

$$U_{01}U_{12}U_{23}U_{34}U_{40} = I.$$

Proof. Each $B_r B_r^{\mathsf{T}}$ is the identity on their common space. Substitution cancels every intermediate frame and leaves $B_0^{\mathsf{T}} B_0 = I$. □

Source: VII, "Pentagon closure"; VIII, "Finite q-pentagon certificate." Evidence: direct overlap identity.

This loop closes for any complete orthonormal atlases of a common space. Consistent phases and label permutations cancel, and a common wrong subspace can also pass. Closure therefore checks overlap composition. Verifying the intended Casimir construction separately requires lowering-kernel residuals, dimensions, spectra, eigen-equations, and joint diagonalization. Comparing independently constructed local Racah edge blocks would provide a stronger associator implementation check.

Theorem 3.7 (Finite rotation compiler)**.** *Every real orthogonal $d \times d$ overlap has a factorization*

$$\mathcal{R} = G_1^{\mathsf{T}} \cdots G_L^{\mathsf{T}} D, \qquad L \le d(d-1)/2,$$

where each G_r is a two-coordinate Givens rotation and D is diagonal with entries ± 1.

Proof. Row rotations eliminate successive entries below the diagonal. An upper triangular orthogonal matrix is diagonal with unit-magnitude diagonal entries. Reverse the rotations. □

Source: VII, "Givens packet compilation"; VIII, "Finite rotation compiler." Evidence: direct finite linear algebra.

The compiler acts on the multiplicity coordinate dimension. This supplies a finite reconstruction circuit without asserting hardware locality or gate optimality.

3.7 Scalar bridge and its remaining comparison

For $0 < q < 1$, define $(a;q)_k = \prod_{r=0}^{k-1}(1-aq^r)$. For nonnegative integers n, y and parameters for which every denominator in the finite sum is nonzero, the scalar q-Racah expression is

$$R_n(\mu(y);\alpha,\beta,\gamma,\delta \mid q) = \sum_{k=0}^{n} \frac{(q^{-n};q)_k(\alpha\beta q^{n+1};q)_k(q^{-y};q)_k(\gamma\delta q^{y+1};q)_k}{(\alpha q;q)_k(\beta\delta q;q)_k(\gamma q;q)_k(q;q)_k}\, q^k,$$

where $\mu(y) = q^{-y} + \gamma\delta q^{y+1}$. Under these conditions, swapping (n,α,β) with (y,γ,δ) leaves the finite expression unchanged. The sums may be restricted to $k \le \min(n,y)$ because the additional numerator terms vanish.

If $\alpha = q^a, \beta = q^b, \gamma = q^c, \delta = q^d$, termwise passage to $q \to 1$ gives

$${}_4F_3\left(\begin{matrix}-n, n+a+b+1, -y, y+c+d+1 \\ a+1, b+d+1, c+1\end{matrix}; 1\right),$$

provided its finite denominator Pochhammer factors are nonzero, or any exceptional cancellation is separately justified.

Source: VIII, "q-Racah duality" and "Termwise classical limit." Evidence: direct finite identities on the nonsingular domain.

A condition such as $\gamma q = q^{-N}$ selects finite support when $0 \le n, y \le N$. Positive orthogonality additionally requires an admissible nondegenerate lattice, nonzero norms, and positive weights. Termination alone does not establish those requirements. At $N = 1$, $q = 1/2$, $\alpha = \beta = 1/2$, $\gamma = 4$, $\delta = 1$, one has $\gamma q = q^{-1}$ but $\mu(0) = \mu(1) = 3$ and $R_0 = R_1 = (1,1)$.

The remaining explicit comparison is a shape-to-scalar formula with the correct basic-hypergeometric base, parameters, weights, norms, label orientation, and phase factors. VII and VIII cite the classical identification but do not provide this full comparison in their convention. Their separately generated scalar tables do not establish it.

VIII also evaluates the normalized terminating Askey–Wilson polynomial and its classical q-difference eigenfunction identity. This provides a scalar continuous-variable bridge. A continuous-series packet transform, its spectral measure, and an exact lift from the finite matrix construction remain separate work. The consolidated operational object is already exact at the finite matrix level: the reduced space, observables, atlases, signed overlaps, measured kernels, deformation tangent, tree graph, and finite compiler.

Chapter 4

A visual route through the consolidated core

These eight vector figures were generated by the original Python programs. Each remains associated with its source volume and caption. Curves at finite parameters are illustrations or bounded numerical evidence; the mathematical conclusions and their qualifications are stated in Chapters 1–3.

1. Exact coefficient packets

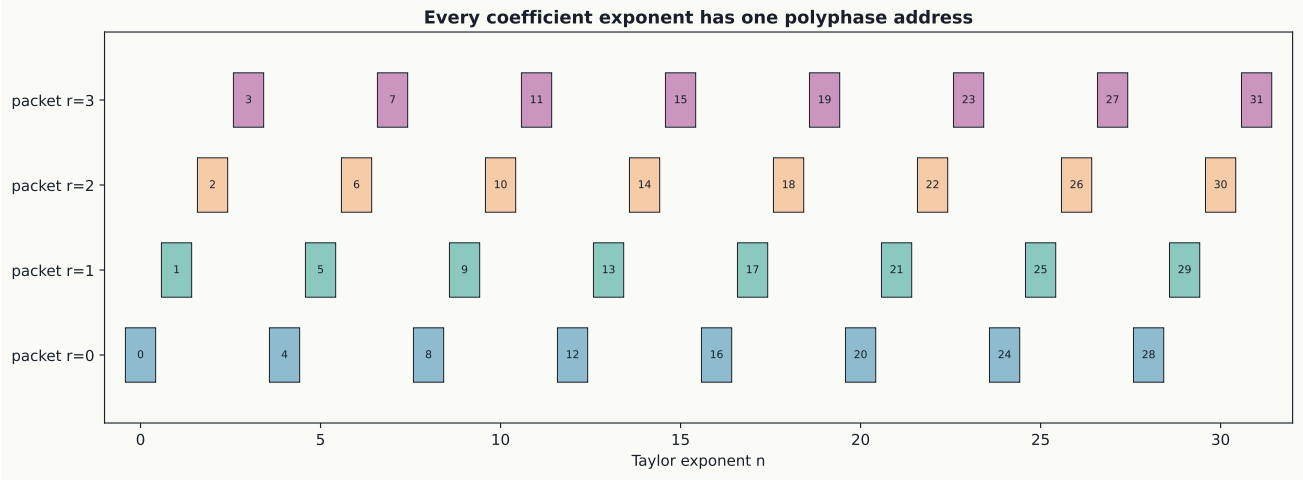


Figure 4.1: Residue addresses partition the coefficients and retain exact reconstruction. This is the algebraic entry point of Chapter 1. Source: Volume I, figure 01.

2. The arithmetic Gamma scale

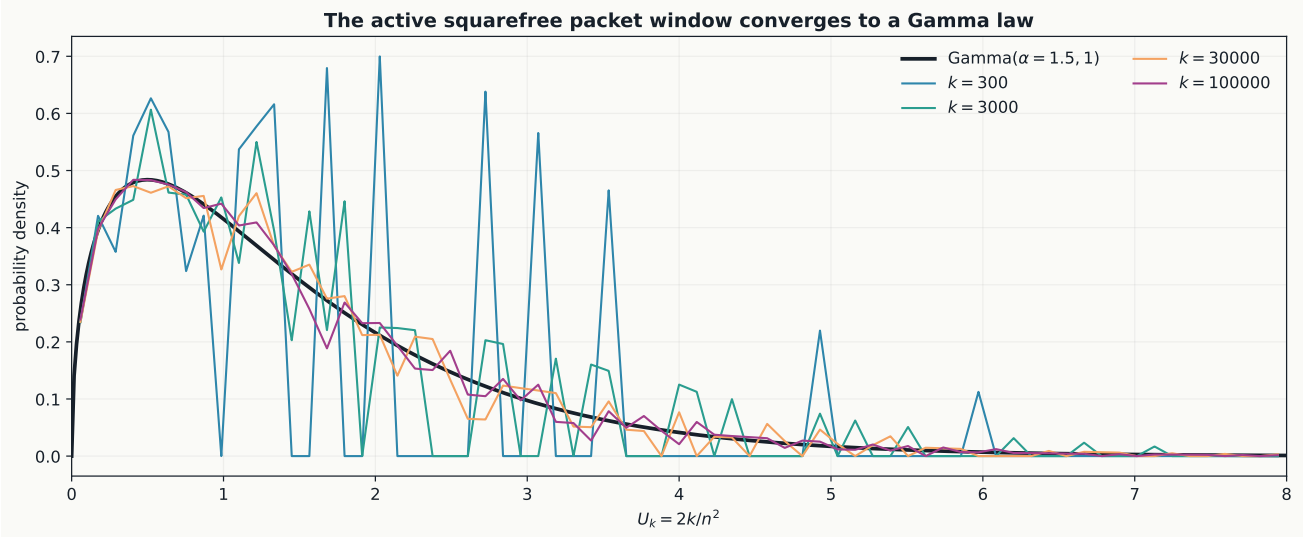


Figure 4.2: The finite-depth distributions approach the Gamma coordinate law. Chapter 1 separates the visual evidence from the all-moment proof. Source: Volume II, figure 14.

3. A common Laguerre fixed point

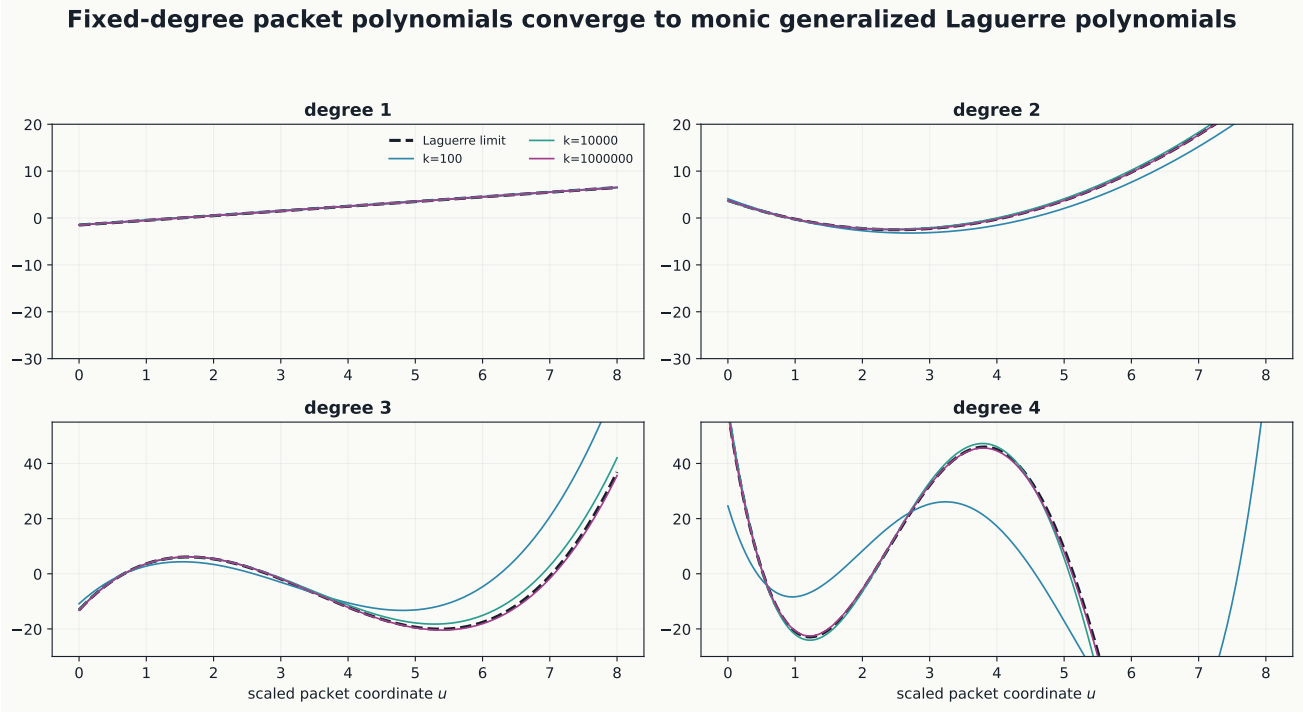


Figure 4.3: Fixed-degree orthogonal polynomials converge after endpoint rescaling. The limiting moment matrix supplies the coefficient argument. Source: Volume III, figure 13.

4. The exact Gamma–sech window

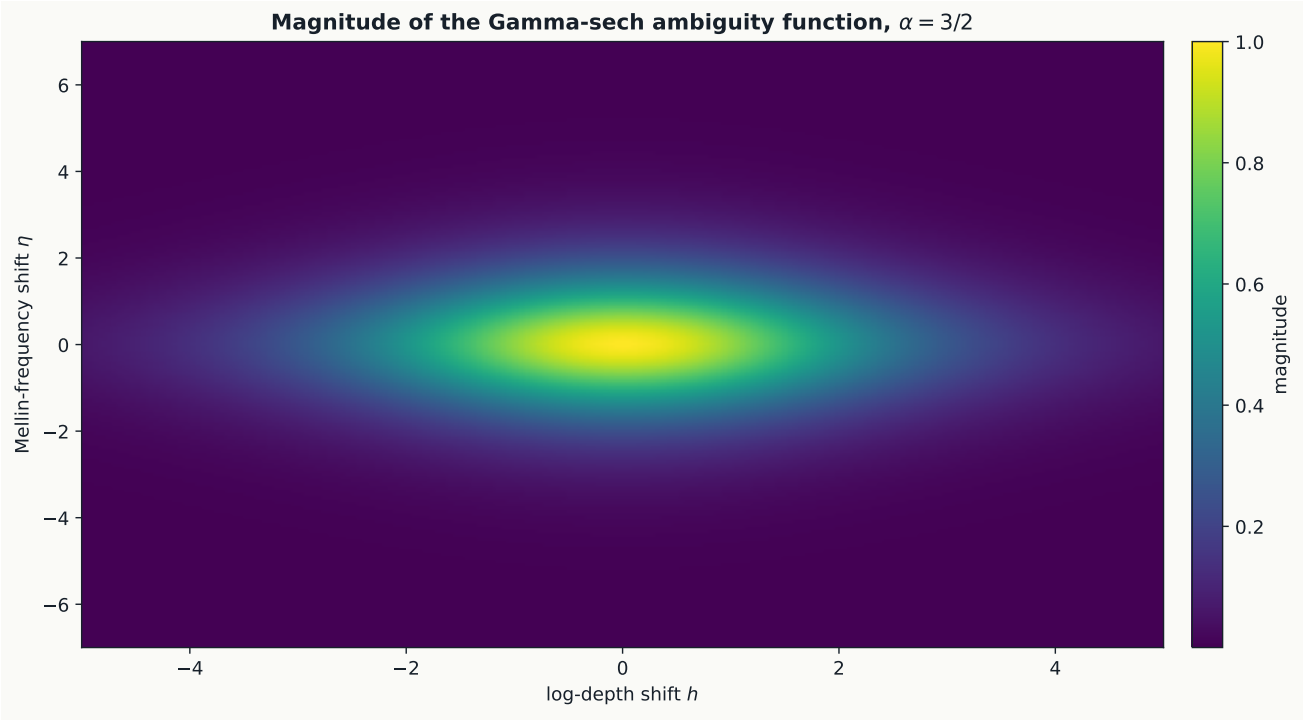


Figure 4.4: The phase-space magnitude is generated from the normalized log-Gamma window. The corresponding cross-shape formula is proved in Chapter 2. Source: Volume V, figure 06.

5. Shape refinement

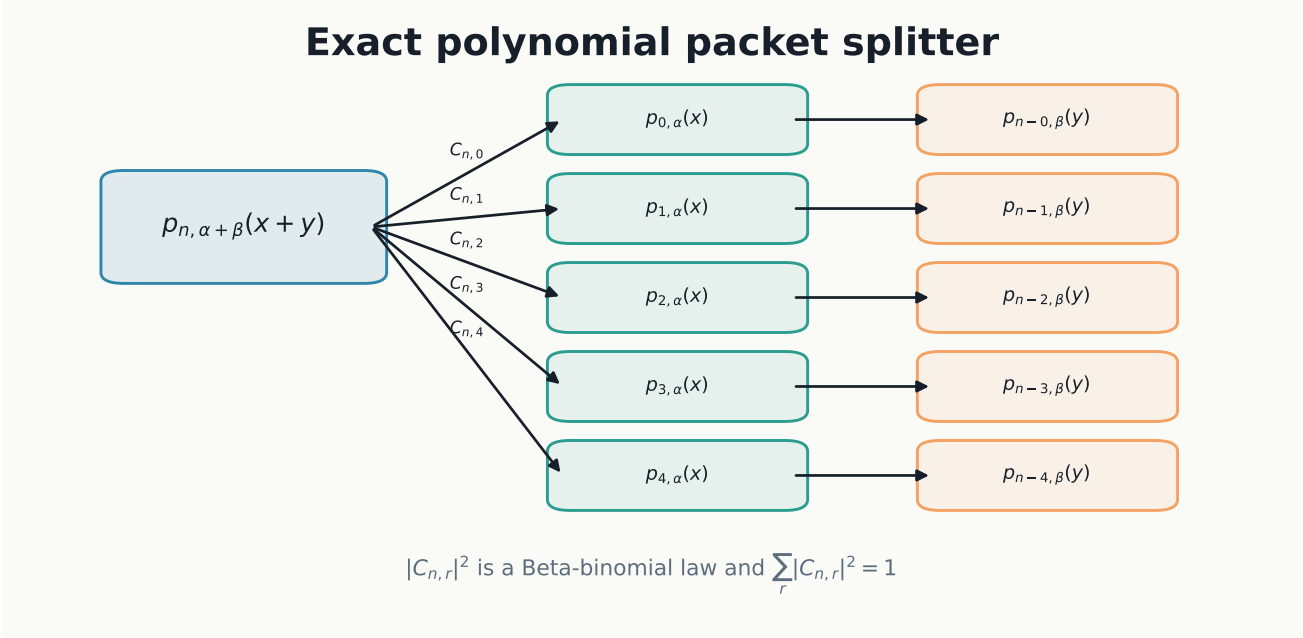


Figure 4.5: The normalized polynomial refinement distributes one total degree across two child shapes. Its squared coefficients form a beta-binomial law. Source: Volume VI, figure 12.

6. Finite recoupling

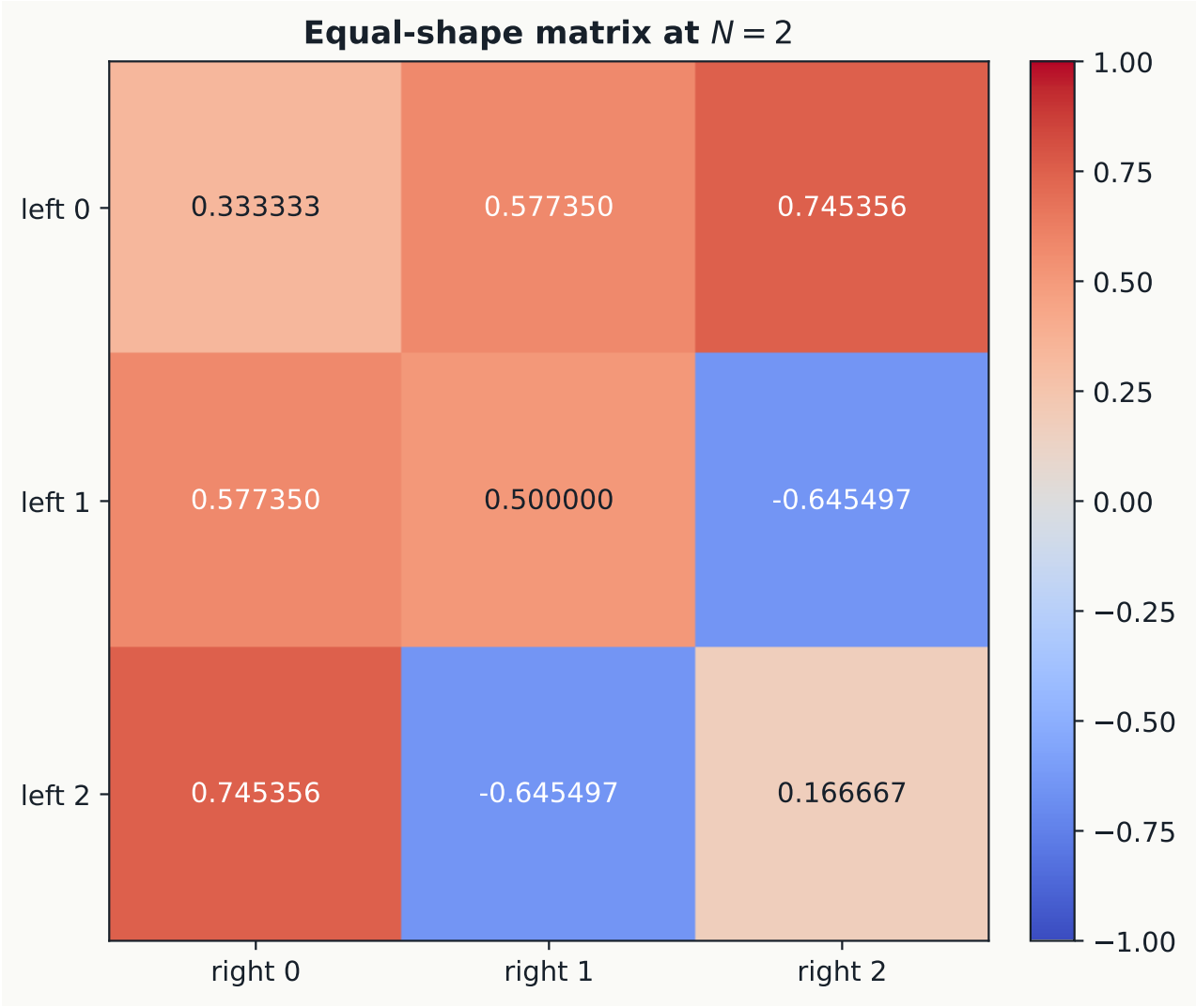


Figure 4.6: Signed amplitudes and their squared probabilities are different outputs of one finite transform. The gauge affects amplitude signs while preserving the measured kernel. Source: Volume VII, figure 08.

7. A deformable two-channel packet

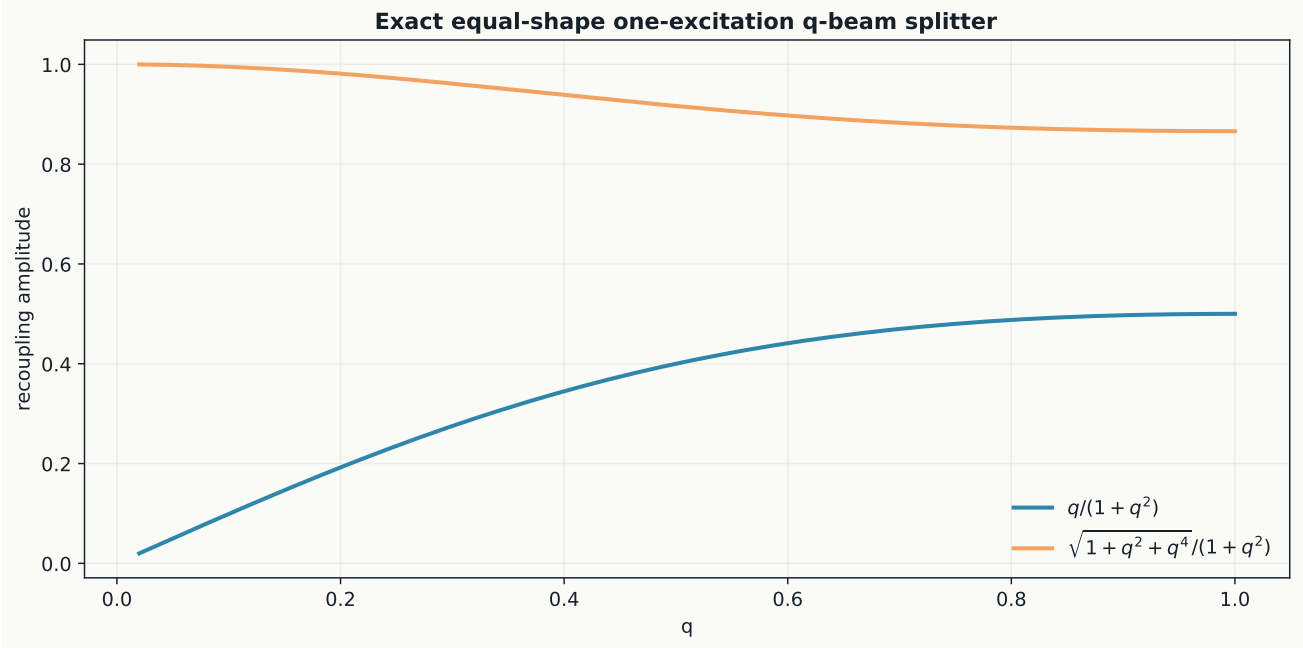


Figure 4.7: The exactly derived one-excitation overlap connects classical mixing with a sparse routing limit. Higher-degree behavior requires its own analysis. Source: Volume VIII, figure 07.

8. A retained negative control

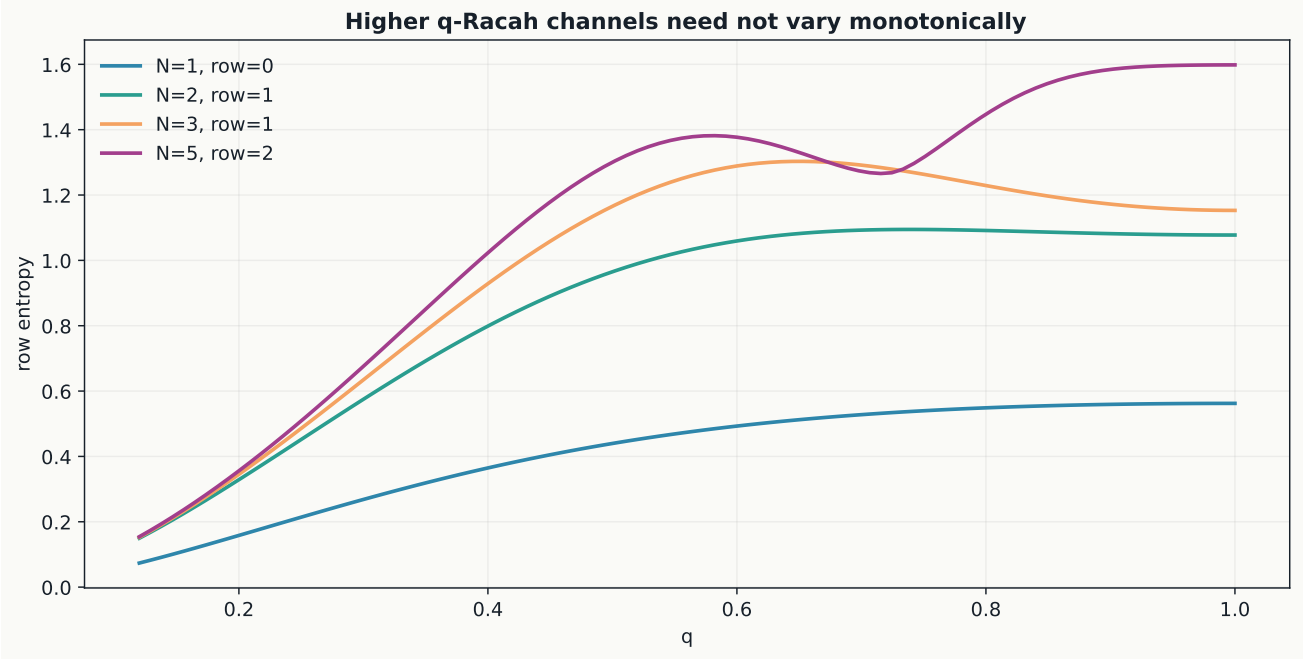


Figure 4.8: The source figure exhibits a higher-excitation entropy decrement as q varies. Fixed-kernel entropy nondecrease after measurement does not imply monotonicity along deformation. Source: Volume VIII, figure 21.

Chapter 5

Corrections and proof dependencies

The following register governs the preserved source papers in the complete reader. It separates false wording, missing hypotheses, and incomplete proof implications. The review was bounded to the current I–VIII manuscripts and selected decisive checks; it does not certify every theorem. The full review notes retain exact source locators.

5.1 Endpoint and character conventions

C01 / Volume II, Proposition 13.1: zero depth. The number-basis matrix element on $\ell^2(\{2,3,\dots\})$ omits the $n=1$ term at $k=0$. Its contribution is one under the moment convention $0^0=1$. State the displayed identities for $k\geq 1$, or retain the separate zero-eigenvalue atom of the positive contraction. The logarithmic scale generator is used only on $n\geq 2$. This is a correction after a stated restriction, not a failure of the positive measure.

C02 / Volumes III–V: modulus and conductor. For a character modulo q, $|\chi(n)|^2=\mathbf{1}_{(n,q)=1}$. An induced character can have a smaller conductor without changing its actual zero set. The principal character modulo four has conductor one but vanishes at two. Thus the coprimality parameter and squarefree density use the modulus; a conductor formulation requires the character to be primitive modulo that conductor. The consolidated chapters use the modulus convention.

C03 / Volume I, Chapter 14: wavelet mean. The zero-integral Mertens-wavelet sentence requires $k\geq 1$. At $k=0$, its stated integral is one. The subsequent source Mellin theorem already has the needed positive-depth restriction.

5.2 Positive spectral models

C04 / Volume III, abstract and claim map: quadrature support. Gaussian quadrature nodes lie in the interval spanned by a positive measure; they need not be original discrete spectral points. For $a=\beta=2$, $q=1$, and one quadrature node, the node is E_1/E_0. The original spectrum has zero and its first positive point $9/16$. Directly,

$$0<\frac{E_1}{E_0}<\sum_{n\geq 2}n^{-4}<\frac{1}{16}+\int_2^\infty x^{-4}\,dx=\frac{5}{48}<\frac{9}{16}.$$

The node lies in the spectral gap. The actual moment-exact theorem survives this correction. Exact Gaussian nodes here are in $(0,1)$; a tiny floating-point node is not an exact representation of the atom

at zero.

C05 / Volume III, Corollary 5.2: residue sign. The weights in $E_0\lambda_j/(1-z\xi_j)$ are positive. The complex residue at $z = 1/\xi_j$ is

$$\operatorname*{Res}_{z=1/\xi_j} \frac{E_0\lambda_j}{1-z\xi_j} = -\frac{E_0\lambda_j}{\xi_j} < 0.$$

The corrected statement is "positive spectral weights and poles on $(1,\infty)$."

C06 / Volume III, Theorems 6.1–7.1: all moments. Weak convergence against bounded continuous functions does not by itself imply convergence of polynomial moments. Chapter 1 supplies the missing implication from Volume II's shifted-energy identity. It also transfers the moments between $U_k = 2kn^{-\beta}$ and $u_k = U_k - U_k^2/(4k)$. This closes the proof linkage needed for the fixed-degree Laguerre and determinant arguments.

C07 / Volume III, Theorem 11.1: endpoint cusp. The original proof informally replaces a discrete measure by a density. Chapter 1 supplies an Abelian proof directly from $E_k \sim Ak^{-3/2}$, with no density replacement. The resulting coefficient is the stated $3/(2^{3/2}\pi)$. This repairs the local proof passage conditional on the source energy law and its counting input. It does not provide a new signed arithmetic estimate.

5.3 Signed rational models

C08 / Volume IV, Theorem 11.1: denominator degree. Hankel nonbreakdown establishes unique normalized rational matching with $\deg P \le m-1$ and $\deg Q \le m$. An ordinary m-term simple-packet decomposition additionally follows when $\deg Q = m$ and its m zeros are simple. The source proof does not establish that full-degree implication for every arithmetic packet.

Generic signed moments $(0,1,0,0)$ at $m = 2$ have a nonzero Hankel determinant but matching rational function z and denominator one. This illustrates the missing general implication; it is not a counterexample within the Möbius arithmetic family. Preserve the rational matching result and qualify its simple-packet extension. Positivity, real support, and universal numerical stability are not implied.

5.4 Recoupling and deformation

C09 / Volume VII, finite recurrence: sign gauge. An irreducible real tridiagonal matrix has nonzero signed off-diagonal entries. Positive entries require a diagonal sign gauge. For the displayed equal-half-shape one-excitation matrix,

$$R = \begin{pmatrix} 1/2 & \sqrt{3}/2 \\ \sqrt{3}/2 & -1/2 \end{pmatrix}, \qquad R\,\mathrm{diag}(0,2)R^T = \begin{pmatrix} 3/2 & -\sqrt{3}/2 \\ -\sqrt{3}/2 & 1/2 \end{pmatrix}.$$

The displayed matrix therefore cannot satisfy the asserted positive-off-diagonal recurrence in that same gauge. Chapter 3 states the gauge transformation explicitly.

C10 / Volumes VII–VIII, pentagon interpretation. The product of overlap matrices telescopes for any complete orthonormal bases of one common space. Consistent sign changes, permutations, or a shared incorrect subspace can still give identity. The theorem is valid; the inference that every wrong phase, ordering, or projector must be detected by closure is false. Representation relations, lowest-weight annihilation, pair spectra, and tree eigen-equations must be checked separately. A small closure residual is one consistency check.

C11 / Volume VIII, scalar q-Racah family. Termination through a denominator parameter q^{-N} alone does not guarantee a positive orthogonal family. Let $N = 1$, $q = 1/2$, and

$$\alpha = \beta = 1/2, \qquad \gamma = 4, \qquad \delta = 1.$$

Then $\gamma q = q^{-N} = 2$, all terms needed for $n, y \in \{0, 1\}$ have nonzero denominators, yet the two lattice values coincide:

$$\mu(0) = \mu(1) = 3, \qquad R_0 = (1, 1), \quad R_1 = (1, 1).$$

Nondegenerate lattice and admissible positive-weight hypotheses are necessary. Chapter 3 distinguishes a well-defined terminating scalar formula from an orthogonal family.

C12 / Volume VIII: scalar-to-matrix comparison. The implemented positive Casimir-overlap matrix and the standalone scalar q-Racah fixture are separate constructions. Duality and classical-limit checks of the latter do not establish equality with the former. The exact parameter dictionary, weight normalization, and phase convention remain a comparison obligation. Generic Askey–Wilson eigenfunction identities likewise do not confer positive orthogonality without parameter hypotheses.

5.5 Retained open statements and review limits

The principal Möbius coherence estimate remains open. Its three ledger conjecture entries are provenance for one recurring target, not three independent conjectures. The higher-excitation q-entropy monotonicity claim is excluded by Volume VIII's own negative control; fixed-kernel entropy nondecrease after measurement remains valid.

Classical recoupling/tridiagonal identifications, strong Schwartz convergence in the heat limit, and full bibliography normalization were not all rederived in this consolidation. Their source dependencies remain explicit. The supplied elementary repairs were self-reviewed; the cusp lemma also received a separate neutral subagent audit. Agreement between reviewers is not a kernel certificate.

Chapter 6

Source crosswalk and evidence

6.1 The consolidation map

Sources	Shared block	Retained mathematical work
I–IV	Arithmetic packets	Coefficient and Bernoulli refinement; cyclotomic germs and invariants; reciprocal L-value transfers; positive squarefree measure, moments, resolvent, finite Jacobi models, Gamma/Laguerre scaling; signed continuation.
IV–VI	Shape packets	Depth-time Gram lift; joint Gamma–sech limit; exact log-Gamma window; Fourier/Wigner coordinates; orthogonal bases; shape convolution, refinement, transfers, coherent smoothing.
VI–VIII	Recoupling	Positive discrete-series ladder; lowest-weight projectors; Casimir atlases; ordinary and q-deformed overlaps; measurement kernels; deformation generator; pentagon and finite compilation.
I, II, V	Open arithmetic boundary	One principal signed coherence target, retaining the three original conjecture-record links.

This manuscript selects a mathematical spine rather than reproducing every invariant or example. Volume I's Wronskian negative control, Volume II's Stieltjes/resultant details, the full Wigner formulas in V, and the complete transfer/frame estimates in VI remain available in the preserved papers. The source crosswalk groups related material; it does not certify that differently stated claims are equivalent or count distinct results.

The eight originals contain 431 PDF pages. The source package also preserves the earlier special-function registry as context. It is not silently merged with the numbered series. The complete reader's machine-readable index gives each paper's physical page range, which differs from its original printed page labels.

6.2 Fresh test run

Volume	Fresh pytest passed	Archived built-in passed	Source PDF pages
I	13	51	60
II	18	61	50
III	19	34	38
IV	18	18	55
V	19	19	52
VI	21	24	54
VII	20	13	60
VIII	20	23	62
Total	148	243	431

All 148 collected pytest cases passed, with zero failures, errors, or skips. Each suite ran from a clean temporary source copy. SHA256 comparisons confirmed that the original extracted source trees remained unchanged. Tests include recomputed numerical and symbolic identities as well as archival-ledger, figure, and manuscript checks.

The archived built-in ledgers report 243 successful checks. Only the 64 built-in checks in Papers I and VII were recomputed by the current pytest run. The remaining built-in totals are historical records. Neither test count is a universal proof count. The JSON report preserves exact commands, environment versions, timings, and case-level results. Small exact correction witnesses are recorded separately.

6.3 Formalization scope

The retrieved projector-calculus ledger has 173 special-function source objects: 125 theorem, proposition, lemma, or corollary records; 15 definitions; 22 context, example, or algorithm records; eight boundary records; and three conjecture records.

Volume	Source objects	Theorem/lemma/proposition/corollary records
I	32	22
II	23	15
III	24	16
IV	16	10
V	21	17
VI	22	20
VII	15	10
VIII	20	15
Total	173	125

The statuses are 88 phase-1-queued, 51 queued, one Lean-draft-present, 22 context/example, eight boundary, and three conjectural. No special-function claim in that retrieved ledger has a kernel-checked source-claim status. The actual special-function Lean tranche inspected contains one coefficient-subsequence refinement theorem. It does not formalize the complete analytic series, Gamma limits, orthogonal-polynomial analysis, or signed coherence estimate.

The whole-project v1.1.0 static figures of 594 claim records and 104 theorem-like Lean declarations are not special-function proof counts. That release explicitly reports that a kernel build was not attempted. The later finite browser-safe core does not establish analytic completion of this series. These findings are bounded by the targeted files and searches recorded in the scope report.

6.4 Primary-source checks

The following exact external inputs received bounded primary-source checks in this edition. Other source-paper bibliographies remain preserved for subsequent normalization.

1. L. Báez-Duarte, *A sequential Riesz-like criterion for the Riemann hypothesis*, International Journal of Mathematics and Mathematical Sciences 2005, 3527–3537, Theorem 1.1. DOI: 10.1155/IJMMS.2005.3527. Its binomial coefficients agree with Chapter 1's c_k by the exact reciprocal-zeta identity; its arbitrary-positive-exponent bound gives the stated coherence equivalence.
2. NIST DLMF, Section 18.28, equations 18.28.1_5 and 18.28.6_1–6_2. These match the normalized scalar Askey–Wilson function and the q-difference operator in VIII. The separate orthogonality parameter restrictions are not inferred from the eigenfunction identity.
3. W. Gautschi, *Orthogonal polynomials: applications and computation*, Acta Numerica 5 (1996), 45–119, opening Gaussian-quadrature discussion and support-interval statements. The quadrature support is an interval containing the measure's support, not an assertion that quadrature nodes are original atoms. The decisive counterexample in Chapter 5 is proved directly from the packet measure.

The accompanying literature notes state exactly what was checked and what was not. The consolidation makes no global novelty claim. Classical special functions retain their established names; the unified organization records how reconstruction, refinement, positive models, shape transfer, and recoupling are composed within the packet calculus.

Projector-Generated Special Functions I

Polyphase Differential Lifts, Bernoulli-Lerch Wavelets, Cyclotomic Detail Functions, and Möbius Coherence

Komorebi P.

Abstract

This paper develops a special-function theory generated by exact projector and polyphase operations. For an analytic scalar function, residue sections are assembled into a vector-valued packet function. Multiplication and the Euler derivative become a carry-shift matrix and a diagonal differential operator that satisfy the same Weyl commutator as the scalar pair. Every ordered scalar differential equation therefore lifts to an exact matrix differential system.

The construction is applied to exponentials, Bessel functions, Airy functions, generalized hypergeometric equations, Bernoulli interpolation, Hurwitz zeta, Lerch transcendent, and Möbius-weighted reciprocal L-function transfers. A complex-order Bernoulli-Hurwitz packet family is proved to obey exact reconstruction, exact parent-child refinement, an order-lowering differential ladder, a translation law, and integer interpolation of the terms in Raabe multiplication. A unitary transform across the child address yields cyclotomic Bernoulli-Lerch details with an inverse transform and Parseval identity. The two-channel case identifies Bernoulli and Euler polynomials as the low-pass and high-pass outputs of one exact refinement bank.

Packet Wronskians provide a direct test for ratio monotonicity, zero interlacing, and Hermite-Biehler compatibility. Computation also supplies a useful negative result: the first Bernoulli-Euler Wronskian family is not universally real-rooted. The final part introduces a generalized Möbius-Bernoulli transfer function. Its binomial, Dirichlet, exponential-generating, Mellin, wavelet, polyphase, character, and quantum forms are proved. The diagonal packet energy has a closed asymptotic law. In the zeta case, the Riemann hypothesis is equivalent to subpolynomial growth of a normalized packet-coherence ratio. This equivalence is a theorem target, not a proof of the hypothesis.

All figures were generated from the accompanying Python program. The release contains 26 vector figures, matching PNG files, exact tables, 51 algebraic or high-precision checks, and the full source listing.

Status and claim boundary

The paper proves the algebraic lift, the Bernoulli-Hurwitz packet laws, the cyclotomic transform, the Bernoulli-Euler channel identity, the packet-Wronskian formulas, the generalized transfer identities, the Mellin formula, the long-time power identity, and the packet-energy asymptotic under the stated hypotheses.

Classical inputs include the Hurwitz zeta function, the Lerch transcendent, Raabe multiplication, Bernoulli and Euler polynomial identities, inverse Euler products, the Riesz and Báez-Duarte criteria, and standard squarefree-density estimates.

The paper does not claim that every generated function is historically new. The contribution is the common reconstruction, refinement, differential, transform, and certificate structure. The Bernoulli interpolation and its distinction from the integer Bernoulli-value sequence were developed in the preceding transfer-block note [8]. Historical novelty of individual specializations requires a separate literature study. The packet-coherence condition in the zeta case is equivalent to RH and remains open.

Contents

Status and claim boundary **1**

I The general polyphase lift **5**

1 From a scalar germ to a vector special function **5**

2 The carry-shift representation of the Weyl algebra **7**

3 Worked lifts of classical functions **9**

3.1 The exponential 9

3.2 Bessel functions 11

3.3 Airy functions 12

3.4 Generalized hypergeometric functions 12

II Bernoulli-Hurwitz packets and cyclotomic details **13**

4 Complex-order Bernoulli interpolation **13**

5 Exact refinement, differentiation, and translation **14**

6 Cyclotomic Bernoulli-Lerch details **16**

7 Exponential generating function of the detail bank **18**

8 Bernoulli and Euler as low-pass and high-pass outputs **18**

9 Packet Wronskians and zero diagnostics **19**

10 A worked cyclotomic norm over the third roots of unity **21**

III Möbius-Bernoulli transfer functions 23

11 A generalized reciprocal L-function transfer family 23

12 Exponential generating and Mellin transforms 23

13 The zeta specialization and the Bernoulli seed 24

14 Scale localization and the Mertens wavelet 25

15 Diagonal packet energy 27

16 Möbius packet coherence and RH 29

17 Long-time mean power and a quantum correlation 31

18 Arithmetic polyphase and character packets 33

IV Worked computational atlas and research program 35

19 Worked example: complete three-channel exponential lift 35

20 Worked example: Bernoulli packet refinement 36

21 Worked example: third-root cyclotomic norm 36

22 Worked example: the principal transfer at order 1200 37

23 Algorithmic workflow 37

24 Open problems generated by the calculus 38

24.1 Near-term problems 38

24.2 Major-conjecture interfaces 38

25 Verification 39

26 Conclusion 40

A Formula registry 40

B Complete Python source 41

C Bibliographic note 57

List of Figures

1.1 A coefficient sequence separated into four residue packets. Each Taylor exponent appears in exactly one row, so the analysis map loses no coefficient. 6

2.1 The carry-shift action for five packets. Multiplication advances the packet address, while the last packet wraps around with the factor u. 8

2.2 The operator-level lift. The scalar differential equation and the packet matrix system are two coordinate realizations of the same ordered algebra. 9

3.1 Exponential packets for two and three channels. The section functions are ordinary generalized hypergeometric functions arranged as one exact vector object. 10

3.2 Numerical reconstruction residuals for three packet counts. The residual remains at floating-point roundoff across the displayed interval. 11

3.3 Three residue sections of J_0. The scalar oscillation is divided by exponent class, while exact synthesis recovers the original Bessel function. 12

4.1 Four Bernoulli-Hurwitz packets and their synthesis. The packets may be individually large or oscillatory, but their complete sum equals the direct interpolation. 14

5.1 A Bernoulli packet refined from modulus one to modulus eight. Every parent is the exact sum of its children. The refinement law works at complex order, not only at integer samples. 15

6.1 Four cyclotomic Bernoulli-Lerch channels as the translation parameter varies. The non-DC details carry phase and magnitude information that disappears in the parent sum. 17

6.2 Energy before and after the cyclotomic transform. The child and detail energies agree to numerical precision, so the transform is a genuine perfect-reconstruction bank. 17

8.1 Two Bernoulli children transformed into Bernoulli and Euler channels. The transform changes coordinates without changing the information carried by the pair. 19

9.1 Four low-order packet Wronskians. The zeros can be read directly from the ratio derivative and provide a concrete interlacing diagnostic. 20

9.2 Real and nonreal roots for the first fifteen packet Wronskians. The finite computation rejects the naive conjecture that this entire Wronskian family is real-rooted. 21

10.1 Growth of the exact $L = 3$ cyclotomic detail norm. The parity-dependent square pattern follows from Bernoulli reflection symmetry. 22

14.1 Rescaled principal transfer kernels. The curves collapse near one common profile after the coordinate is divided by $\sqrt{k}$. 25

14.2 The Mertens wavelet at three scales. The sign change remains fixed in the normalized coordinate, making the scale selection explicit. 26

15.1 The scaled energy $k^{3/2}E_k$. The numerical curve approaches the exact constant predicted by the theorem. 28

15.2 Active squarefree mass and effective dimension. The number of packets contributing at one scale grows like a constant multiple of $\sqrt{k}$. 29

16.1 The principal transfer amplitude, diagonal energy scale, and coherence ratio. The numerical values are finite diagnostics only. They do not establish the required asymptotic bound. . 30

16.2 The normalized sequence $k^{3/4}c_k$ in logarithmic time. Zeros off the critical line would appear as expanding modes in this coordinate. 31

17.1 One finite Möbius-Bernoulli correlation and its running mean power. The dashed line is the exact diagonal energy of the truncated packet family. 32

17.2 Operator realization of the generalized transfer function. Scale evolution and real-time logarithmic evolution commute because both are diagonal in the integer basis. 33

18.1 Residue packets of the principal Möbius transfer at three scales. Positive and negative packets shift with k, while their exact sum remains c_k. 34

18.2 Principal and quadratic-character transfer amplitudes. The character coordinate can strongly alter cancellation even when the same scale filter is used. 35

23.1 The generated function families and their main transfer routes. The same packet operations produce differential systems, refinable Bernoulli packets, cyclotomic details, Wronskians, and arithmetic coherence signals. 38

24.1 Status map for the paper. The map separates proved identities, classical inputs, finite computations, open special-function questions, and major conjecture interfaces. 39

Part I

The general polyphase lift

Chapter 1

From a scalar germ to a vector special function

Let

$$F(z) = \sum_{n=0}^{\infty} a_n z^n$$

be analytic in a neighborhood of the origin. Fix an integer $M \geq 2$.

Definition 1.1 (Residue-section functions)**.** For $0 \leq r < M$, define

$$\boxed{F_{M,r}(u) = \sum_{m=0}^{\infty} a_{Mm+r} u^m.}$$

The packet vector is

$$\mathbf{F}_M(u) = \begin{pmatrix} F_{M,0}(u) \\ F_{M,1}(u) \\ \vdots \\ F_{M,M-1}(u) \end{pmatrix}.$$

Theorem 1.2 (Exact polyphase reconstruction)**.** *Wherever the defining series converge,*

$$\boxed{F(z) = \sum_{r=0}^{M-1} z^r F_{M,r}(z^M).}$$

Proof. Every exponent has a unique representation $n = Mm + r$ with $0 \le r < M$. Therefore

$$\sum_{n\ge 0} a_n z^n = \sum_{r=0}^{M-1} \sum_{m\ge 0} a_{Mm+r} z^{Mm+r} = \sum_{r=0}^{M-1} z^r F_{M,r}(z^M).$$

□

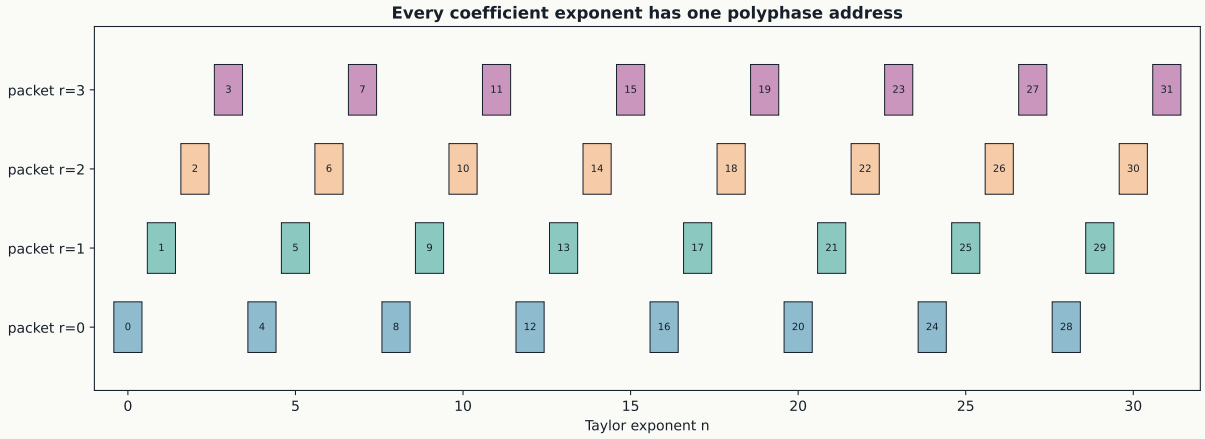


Figure 1.1: A coefficient sequence separated into four residue packets. Each Taylor exponent appears in exactly one row, so the analysis map loses no coefficient.

This elementary identity becomes more useful when the scalar differential operators are also transported into packet coordinates.

Chapter 2

The carry-shift representation of the Weyl algebra

Define the carry-shift matrix

$$C_M(u) = \begin{pmatrix} 0 & 0 & \cdots & 0 & u \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

Define the packet Euler operator

$$\Theta_M = Mu\frac{\mathrm{d}}{\mathrm{d}u}I_M + \mathrm{diag}(0, 1, \ldots, M-1).$$

Proposition 2.1 (Intertwining laws)**.** *Let*

$$\mathcal{S}_M\mathbf{f}(z) = \sum_{r=0}^{M-1} z^r f_r(z^M).$$

Then

$$\boxed{\mathcal{S}_M(C_M\mathbf{f}) = z\,\mathcal{S}_M\mathbf{f}}$$

and

$$\boxed{\mathcal{S}_M(\Theta_M\mathbf{f}) = \left(z\frac{\mathrm{d}}{\mathrm{d}z}\right)\mathcal{S}_M\mathbf{f}.}$$

Proof. Multiplication by z sends packet r to packet $r+1$. The last packet wraps to packet 0 and contributes a factor $z^M = u$. This is exactly $C_M(u)$. For the differential identity, a monomial u^m in packet r reconstructs to z^{Mm+r}. The operator $Mu\frac{\mathrm{d}}{\mathrm{d}u} + r$ multiplies that monomial by $Mm+r$, which is the eigenvalue of $z\frac{\mathrm{d}}{\mathrm{d}z}$ on the reconstructed monomial. □

Corollary 2.2 (Weyl commutator)**.**

$$\boxed{[\Theta_M, C_M] = C_M.}$$

Proof. The scalar operators satisfy

$$\left[z\frac{\mathrm{d}}{\mathrm{d}z}, z\right] = z.$$

Apply the two intertwining laws. The synthesis map is injective because the residue decomposition is unique. □

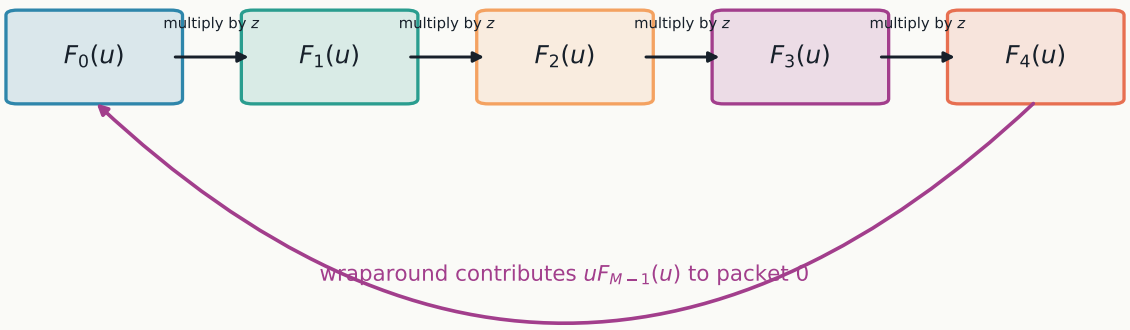


Figure 2.1: The carry-shift action for five packets. Multiplication advances the packet address, while the last packet wraps around with the factor u.

Theorem 2.3 (Differential lift)**.** *Let $\mathcal{L}(z,\theta)$ be an ordered polynomial in the noncommuting variables z and θ, where $\theta = z\frac{\mathrm{d}}{\mathrm{d}z}$. If*

$$\mathcal{L}(z,\theta)F = 0,$$

then

$$\boxed{\mathcal{L}(C_M,\Theta_M)\mathbf{F}_M = 0.}$$

The order of the factors in every monomial of $\mathcal{L}$ is preserved.

Proof. The assignment

$$z \longmapsto C_M, \qquad \theta \longmapsto \Theta_M$$

preserves the defining commutator. It therefore gives a representation of the same ordered operator algebra. Intertwining each ordered monomial with $\mathcal{S}_M$ gives

$$\mathcal{S}_M\big(\mathcal{L}(C_M,\Theta_M)\mathbf{F}_M\big) = \mathcal{L}(z,\theta)F = 0.$$

Injectivity of $\mathcal{S}_M$ gives the result. □

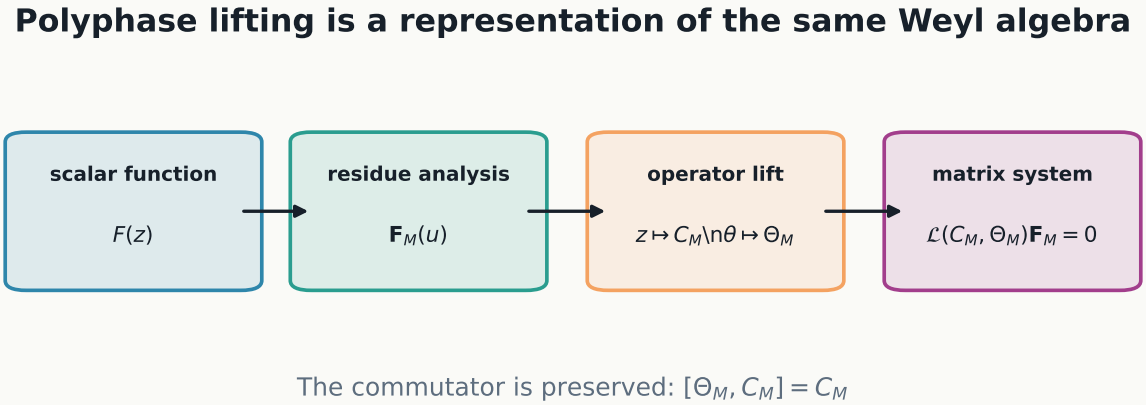


Figure 2.2: The operator-level lift. The scalar differential equation and the packet matrix system are two coordinate realizations of the same ordered algebra.

Chapter 3

Worked lifts of classical functions

3.1 The exponential

The scalar equation

$$\theta e^z = z e^z$$

lifts to

$$\boxed{\Theta_M \mathbf{E}_M = C_M \mathbf{E}_M.}$$

The residue sections are

$$E_{M,r}(u) = \sum_{m=0}^{\infty} \frac{u^m}{(Mm+r)!}.$$

Gauss multiplication gives

$$\boxed{E_{M,r}(u) = \frac{1}{r!}{}_0F_{M-1}\left(\left.\begin{matrix}-\\ \frac{r+1}{M}, \ldots, \widehat{1}, \ldots, \frac{r+M}{M}\end{matrix}\right| \frac{u}{M^M}\right),}$$

where the denominator parameter equal to 1 is omitted.

Worked example 3.1 (Two channels)**.** For $M = 2$,

$$E_{2,0}(u) = \cosh\sqrt{u}, \qquad E_{2,1}(u) = \frac{\sinh\sqrt{u}}{\sqrt{u}}.$$

Hence

$$e^z = E_{2,0}(z^2) + zE_{2,1}(z^2) = \cosh z + \sinh z.$$

Worked example 3.2 (Three channels)**.** For $M = 3$,

$$E_{3,0}(u) = {}_0F_2\left(;\frac{1}{3},\frac{2}{3};\frac{u}{27}\right),$$
$$E_{3,1}(u) = {}_0F_2\left(;\frac{2}{3},\frac{4}{3};\frac{u}{27}\right),$$
$$E_{3,2}(u) = \frac{1}{2}{}_0F_2\left(;\frac{4}{3},\frac{5}{3};\frac{u}{27}\right).$$

The exact synthesis is

$$\boxed{e^z = E_{3,0}(z^3) + zE_{3,1}(z^3) + z^2E_{3,2}(z^3).}$$

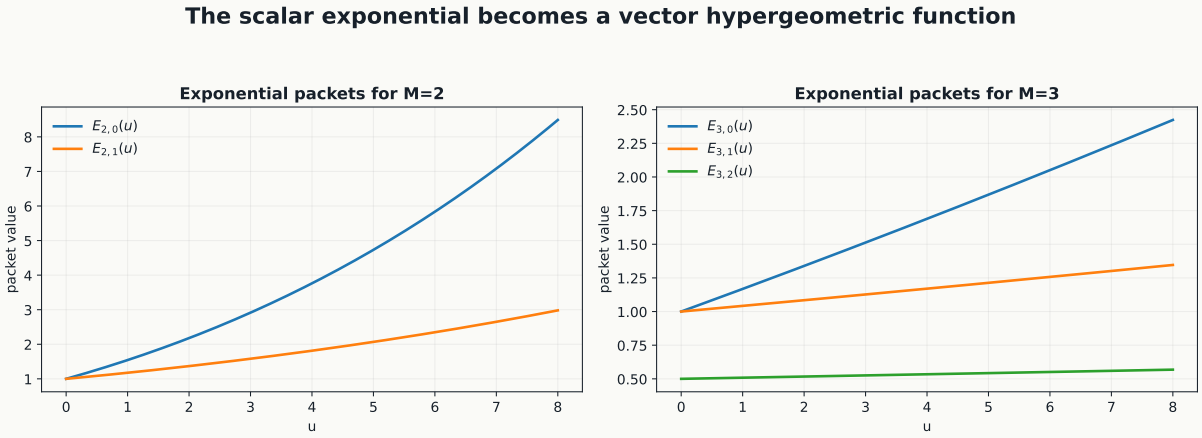


Figure 3.1: Exponential packets for two and three channels. The section functions are ordinary generalized hypergeometric functions arranged as one exact vector object.

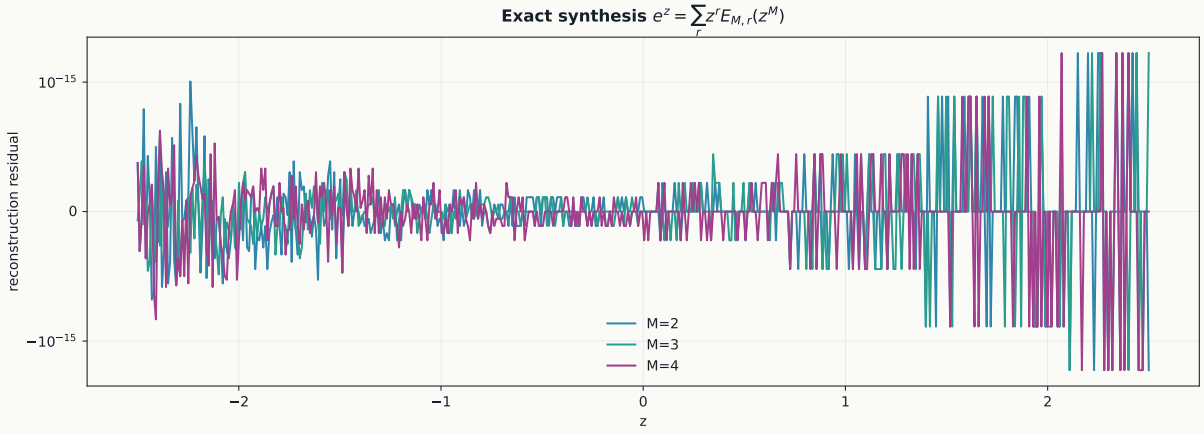


Figure 3.2: Numerical reconstruction residuals for three packet counts. The residual remains at floating-point roundoff across the displayed interval.

3.2 Bessel functions

The Bessel equation for J_ν can be written

$$\left(\theta^2 + z^2 - \nu^2\right) J_\nu(z) = 0.$$

Whenever the chosen branch is analytic in the working neighborhood, the packet vector satisfies

$$\boxed{\left(\Theta_M^2 + C_M^2 - \nu^2 I\right) \mathbf{J}_{\nu,M} = 0.}$$

For integer order, the Taylor expansion at the origin supplies a global entire packet vector. Figure 3.3 shows the three residue sections of J_0.

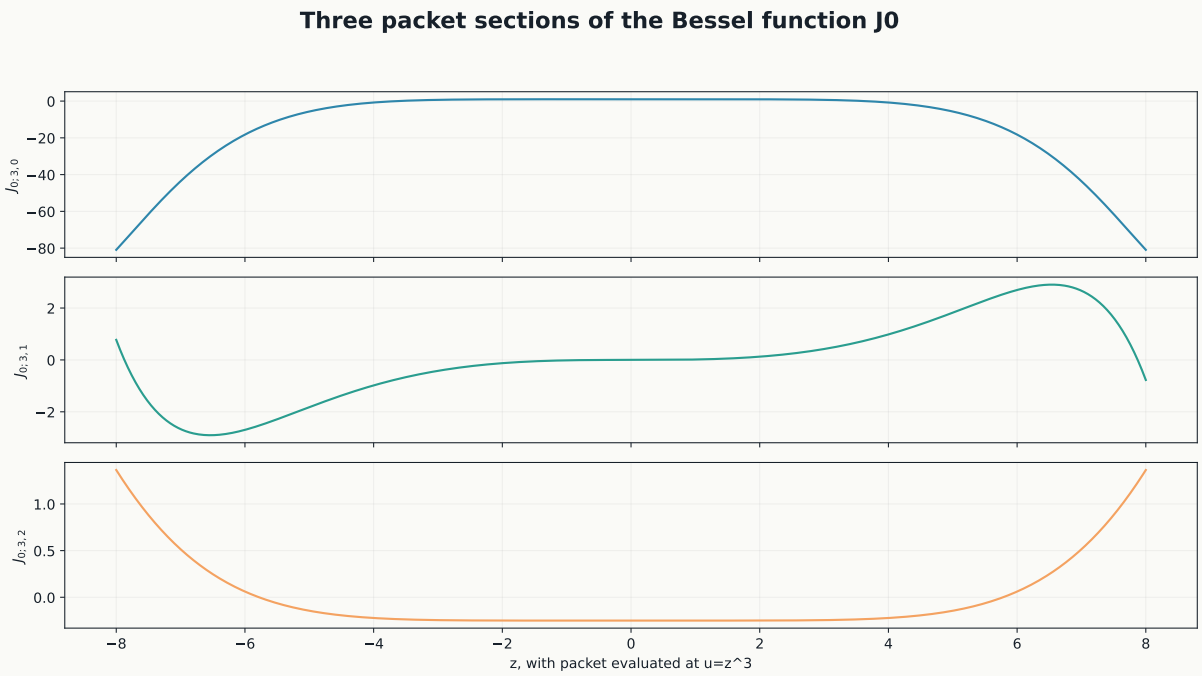


Figure 3.3: Three residue sections of J_0. The scalar oscillation is divided by exponent class, while exact synthesis recovers the original Bessel function.

3.3 Airy functions

The Airy equation $F'' = zF$ is equivalent to

$$\theta(\theta - 1)F = z^3 F.$$

Its packet lift is

$$\boxed{\Theta_M(\Theta_M - I)\mathbf{F}_M = C_M^3 \mathbf{F}_M.}$$

This example is useful because the scalar equation contains a derivative order and a multiplication order that do not commute. The ordered lift preserves both.

3.4 Generalized hypergeometric functions

The generalized hypergeometric function satisfies

$$\left[\theta \prod_{j=1}^{q}(\theta + b_j - 1) - z\prod_{i=1}^{p}(\theta + a_i)\right] F = 0.$$

The packet system is therefore

$$\boxed{\left[\Theta_M \prod_{j=1}^{q}(\Theta_M + (b_j - 1)I) - C_M \prod_{i=1}^{p}(\Theta_M + a_i I)\right] \mathbf{F}_M = 0.}$$

This gives a systematic matrix-valued lift of every scalar generalized hypergeometric equation.

Part II

Bernoulli-Hurwitz packets and cyclotomic details

Chapter 4

Complex-order Bernoulli interpolation

For $a > 0$, define

$$\boxed{\mathfrak{B}(z;a) = -z\zeta(1-z,a).}$$

The pole of the Hurwitz zeta function at $1 - z = 1$ is removed by the factor $-z$, so $\mathfrak{B}(z;a)$ is entire in z and $\mathfrak{B}(0;a) = 1$. The special-value and multiplication formulas used below are classical properties of Bernoulli polynomials and the Hurwitz zeta function [1, 2].

For positive integers n,

$$\mathfrak{B}(n;a) = B_n(a).$$

Thus $\mathfrak{B}$ is a complex-order interpolation of the Bernoulli polynomial sequence.

Definition 4.1 (Bernoulli-Hurwitz packet)**.** For $M \geq 1$ and $0 \leq r < M$, define

$$\boxed{\mathfrak{B}_{M,r}(z;a) = -zM^{z-1}\zeta\left(1-z, \frac{a+r}{M}\right).}$$

At $z = 0$, define $\mathfrak{B}_{M,r}(0;a) = 1/M$ by removal of the singularity.

Theorem 4.2 (Reconstruction)**.**

$$\boxed{\mathfrak{B}(z;a) = \sum_{r=0}^{M-1} \mathfrak{B}_{M,r}(z;a).}$$

Proof. The Hurwitz multiplication formula is

$$\sum_{r=0}^{M-1} \zeta\left(s, \frac{a+r}{M}\right) = M^s\zeta(s,a).$$

Set $s = 1 - z$ and multiply by $-zM^{z-1}$. □

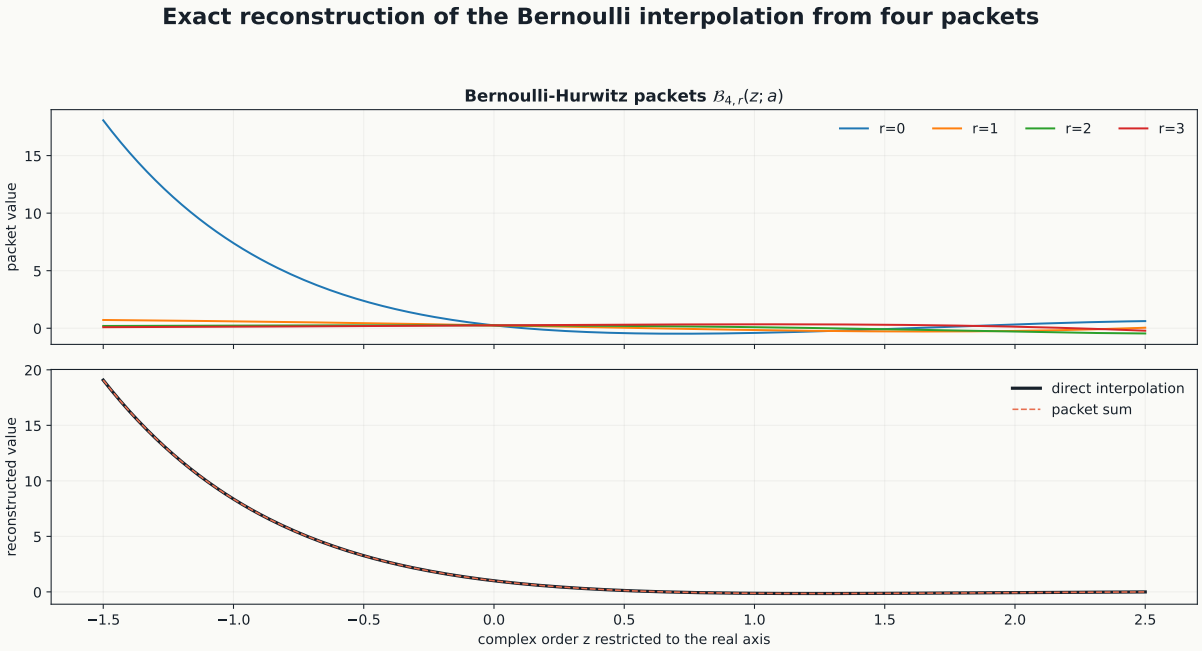


Figure 4.1: Four Bernoulli-Hurwitz packets and their synthesis. The packets may be individually large or oscillatory, but their complete sum equals the direct interpolation.

Chapter 5

Exact refinement, differentiation, and translation

Theorem 5.1 (Parent-child refinement)**.** *For integers $L, M \geq 1$,*

$$\boxed{\mathfrak{B}_{M,r}(z;a) = \sum_{\ell=0}^{L-1} \mathfrak{B}_{LM,r+\ell M}(z;a).}$$

Proof. Set $A = (a+r)/M$. The child arguments are $(A+\ell)/L$. Summing their Hurwitz zeta functions and applying the multiplication formula at modulus L gives the parent. The powers of L cancel exactly. □

Theorem 5.2 (Integer samples)**.** *For every positive integer n,*

$$\boxed{\mathfrak{B}_{M,r}(n;a) = M^{n-1} B_n\left(\frac{a+r}{M}\right).}$$

Proof. Use $\zeta(1-n,x) = -B_n(x)/n$. □

The reconstruction theorem at integer order becomes Raabe multiplication:

$$B_n(a) = M^{n-1} \sum_{r=0}^{M-1} B_n\left(\frac{a+r}{M}\right).$$

Theorem 5.3 (Differential ladder)**.**

$$\boxed{\frac{\partial}{\partial a}\mathfrak{B}_{M,r}(z;a) = z\mathfrak{B}_{M,r}(z-1;a).}$$

Proof. The Hurwitz derivative is

$$\frac{\partial}{\partial x}\zeta(s,x) = -s\zeta(s+1,x).$$

Apply the chain rule to $x = (a+r)/M$ and simplify. □

Theorem 5.4 (Translation law)**.**

$$\boxed{\mathfrak{B}_{M,r}(z;a+M) - \mathfrak{B}_{M,r}(z;a) = z(a+r)^{z-1}.}$$

Proof. Use

$$\zeta(s,x+1) = \zeta(s,x) - x^{-s}.$$

□

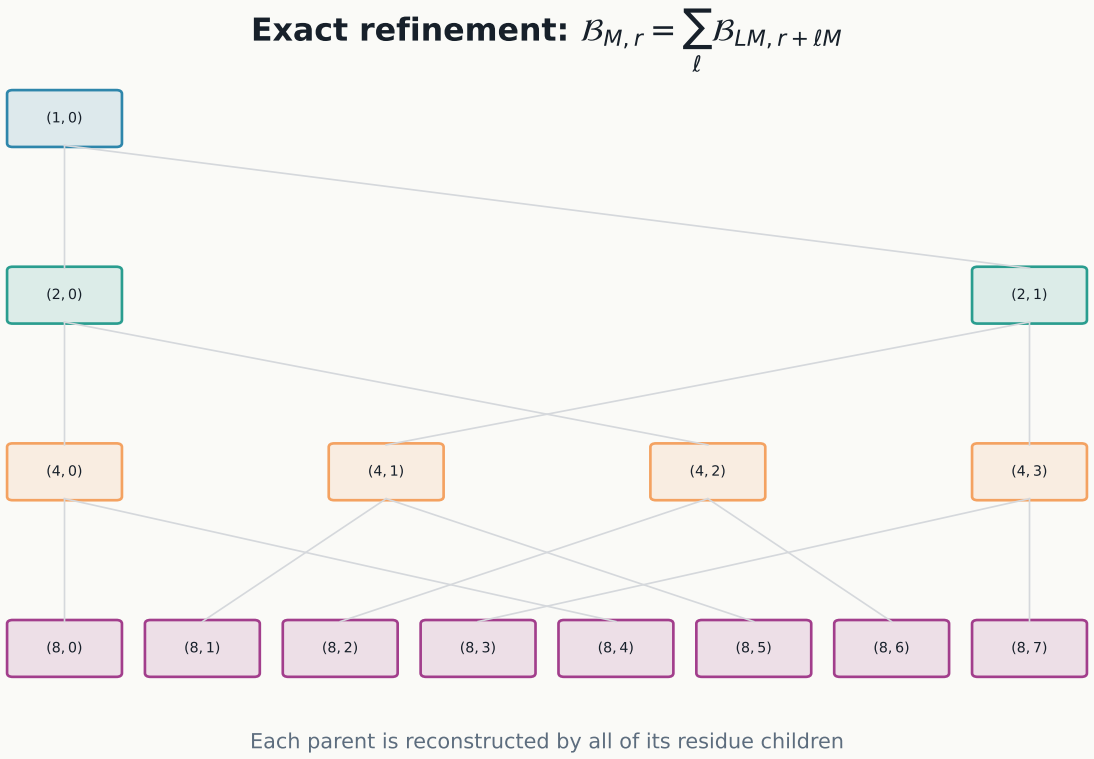


Figure 5.1: A Bernoulli packet refined from modulus one to modulus eight. Every parent is the exact sum of its children. The refinement law works at complex order, not only at integer samples.

Chapter 6

Cyclotomic Bernoulli-Lerch details

Let $\omega_L = \mathrm{e}^{2\pi \mathrm{i}/L}$.

Definition 6.1 (Cyclotomic detail)**.**

$$\boxed{\mathcal{W}_{M,r}^{(L,j)}(z;a) = \frac{1}{\sqrt{L}} \sum_{\ell=0}^{L-1} \omega_L^{-j\ell} \mathfrak{B}_{LM,r+\ell M}(z;a).}$$

Theorem 6.2 (Lerch closed form)**.**

$$\boxed{\mathcal{W}_{M,r}^{(L,j)}(z;a) = -\frac{z}{\sqrt{L}} M^{z-1} \Phi\left(\omega_L^{-j}, 1-z, \frac{a+r}{M}\right).}$$

Proof. Write each child Hurwitz zeta function as a series. The child address ℓ and the internal summation index combine into one integer $n = Lm + \ell$. The Fourier weight $\omega_L^{-j\ell}$ becomes ω_L^{-jn}. The resulting series is the Lerch transcendent. □

The DC channel is the parent:

$$\boxed{\mathcal{W}_{M,r}^{(L,0)} = \frac{1}{\sqrt{L}} \mathfrak{B}_{M,r}.}$$

The inverse transform is

$$\boxed{\mathfrak{B}_{LM,r+\ell M} = \frac{1}{\sqrt{L}} \sum_{j=0}^{L-1} \omega_L^{j\ell} \mathcal{W}_{M,r}^{(L,j)}.}$$

For each fixed (z, a), Parseval gives

$$\boxed{\sum_{\ell=0}^{L-1} \left|\mathfrak{B}_{LM,r+\ell M}\right|^2 = \sum_{j=0}^{L-1} \left|\mathcal{W}_{M,r}^{(L,j)}\right|^2.}$$

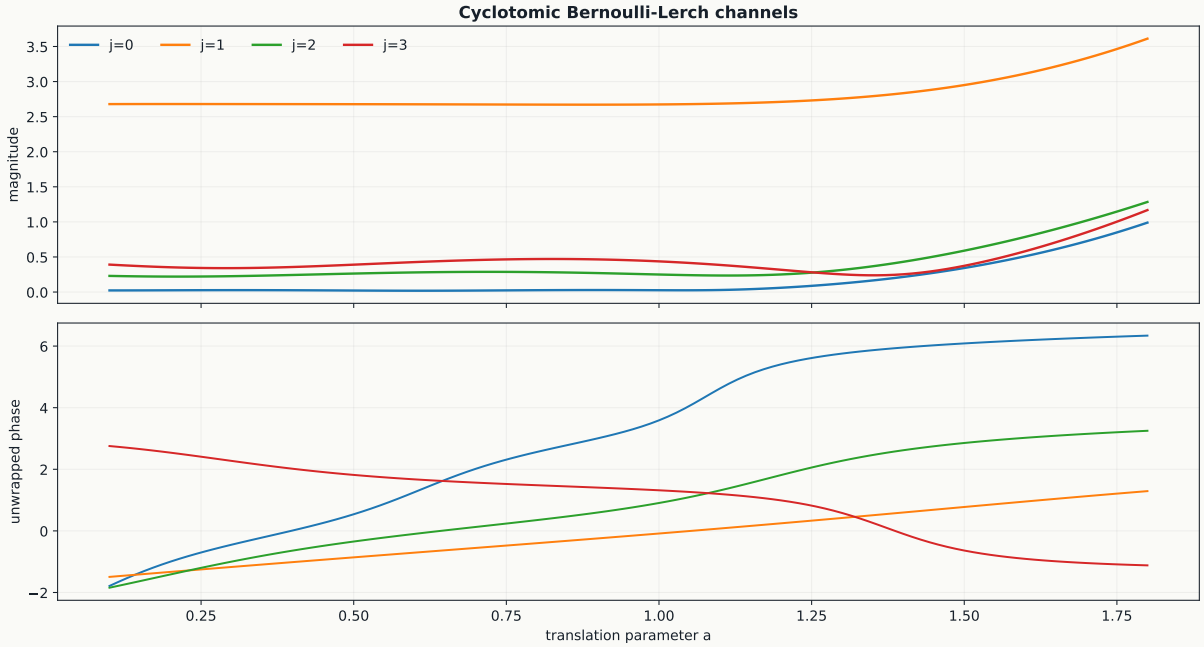


Figure 6.1: Four cyclotomic Bernoulli-Lerch channels as the translation parameter varies. The non-DC details carry phase and magnitude information that disappears in the parent sum.

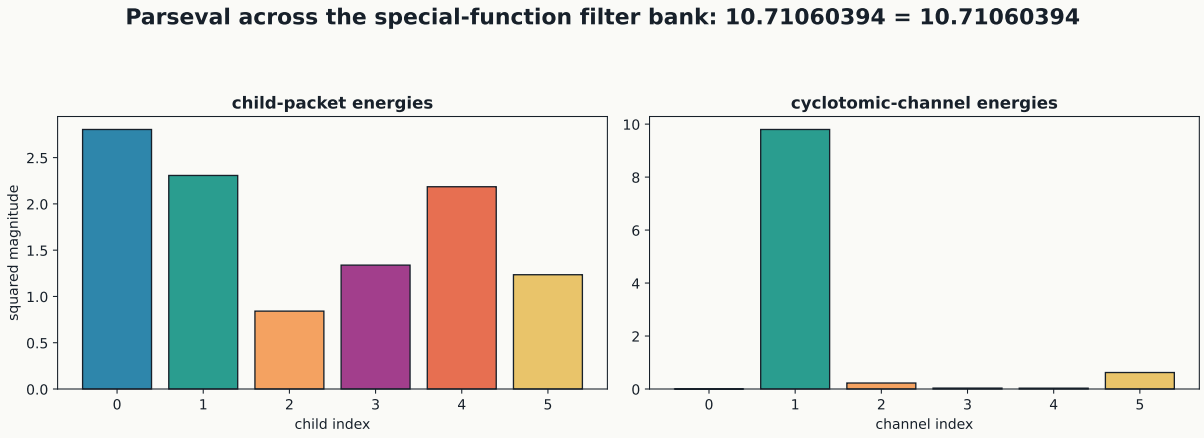


Figure 6.2: Energy before and after the cyclotomic transform. The child and detail energies agree to numerical precision, so the transform is a genuine perfect-reconstruction bank.

Chapter 7

Exponential generating function of the detail bank

At positive integer order, define

$$W_{L,j}(n;a) = \frac{1}{\sqrt{L}} \sum_{\ell=0}^{L-1} \omega_L^{-j\ell} L^{n-1} B_n\left(\frac{a+\ell}{L}\right).$$

Theorem 7.1 (Cyclotomic EGF)**.**

$$\boxed{\sum_{n=0}^{\infty} W_{L,j}(n;a)\frac{t^n}{n!} = \frac{te^{at}}{\sqrt{L}\left(\omega_L^{-j}e^t - 1\right)}.}$$

Proof. The EGF of the ℓ-th child is

$$\sum_{n\geq 0} L^{n-1} B_n\left(\frac{a+\ell}{L}\right)\frac{t^n}{n!} = \frac{te^{(a+\ell)t}}{e^{Lt}-1}.$$

Take the finite Fourier transform over ℓ and sum the geometric progression. The factor $1 - e^{Lt}$ cancels the denominator. □

This is the standard root-of-unity Apostol-Bernoulli generating form, now obtained as one channel of an exact refinement bank. The Lerch representation uses the conventional analytic continuation described in [2].

Chapter 8

Bernoulli and Euler as low-pass and high-pass outputs

Take $L = 2$. The two children are

$$2^{n-1}B_n\left(\frac{a}{2}\right) \quad \text{and} \quad 2^{n-1}B_n\left(\frac{a+1}{2}\right).$$

Theorem 8.1 (Two-channel identity)**.**

$$\boxed{W_{2,0}(n;a) = \frac{B_n(a)}{\sqrt{2}}}$$

and, for $n \geq 1$*,*

$$\boxed{W_{2,1}(n;a) = -\frac{n}{2\sqrt{2}}E_{n-1}(a).}$$

Proof. The first formula is Raabe multiplication. The second follows from

$$E_{n-1}(a) = \frac{2^n}{n}\left[B_n\left(\frac{a+1}{2}\right) - B_n\left(\frac{a}{2}\right)\right].$$

□

Thus Bernoulli and Euler polynomials are the DC and Nyquist channels of one exact Bernoulli refinement.

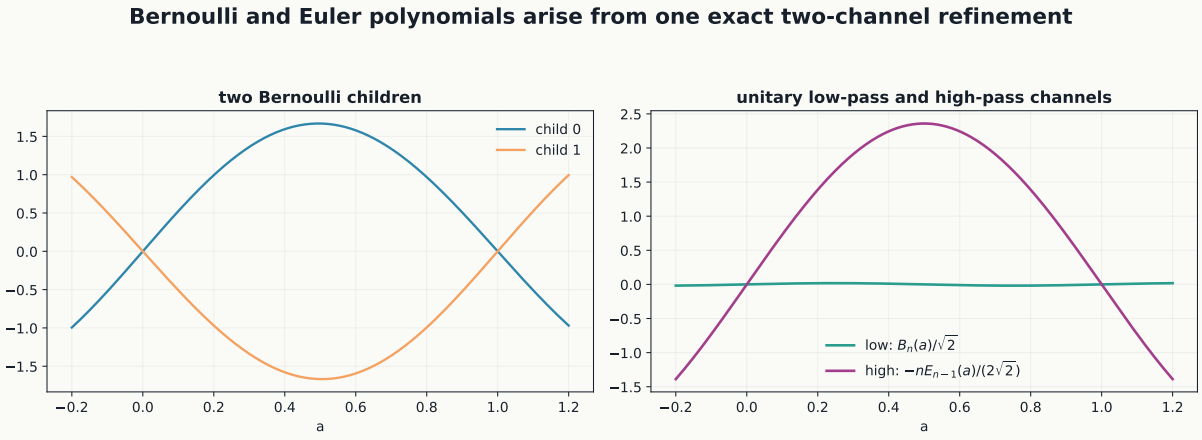


Figure 8.1: Two Bernoulli children transformed into Bernoulli and Euler channels. The transform changes coordinates without changing the information carried by the pair.

Chapter 9

Packet Wronskians and zero diagnostics

Define

$$\boxed{\mathcal{D}_{j,\ell}(z;a) = \mathcal{W}_j(z;a)\mathcal{W}_\ell(z-1;a) - \mathcal{W}_\ell(z;a)\mathcal{W}_j(z-1;a).}$$

Proposition 9.1 (Ratio derivative)**.** *Where* $\mathcal{W}_\ell(z;a) \neq 0$,

$$\boxed{\frac{\partial}{\partial a}\left(\frac{\mathcal{W}_j(z;a)}{\mathcal{W}_\ell(z;a)}\right) = -\frac{z\mathcal{D}_{j,\ell}(z;a)}{\mathcal{W}_\ell(z;a)^2}.}$$

Proof. Differentiate the ratio and use the order-lowering differential ladder for both channels. □

For the two-channel integer family, define

$$\boxed{\mathcal{W}_n(a) = nB_{n-1}(a)E_{n-1}(a) - (n-1)B_n(a)E_{n-2}(a).}$$

This is four times the packet Wronskian of the Bernoulli and Euler channels at consecutive orders.

Worked example 9.2 (Orders two through four)**.**

$$\mathcal{W}_2(a) = \frac{3a^2 - 3a + 1}{3},$$

which has no real zeros. Also,

$$\mathcal{W}_3(a) = a^2(a-1)^2,$$

while

$$\mathcal{W}_4(a) = \frac{a(a-1)}{5}\left(5a^4 - 10a^3 + 5a - 2\right).$$

The fourth-order polynomial has nonreal roots.

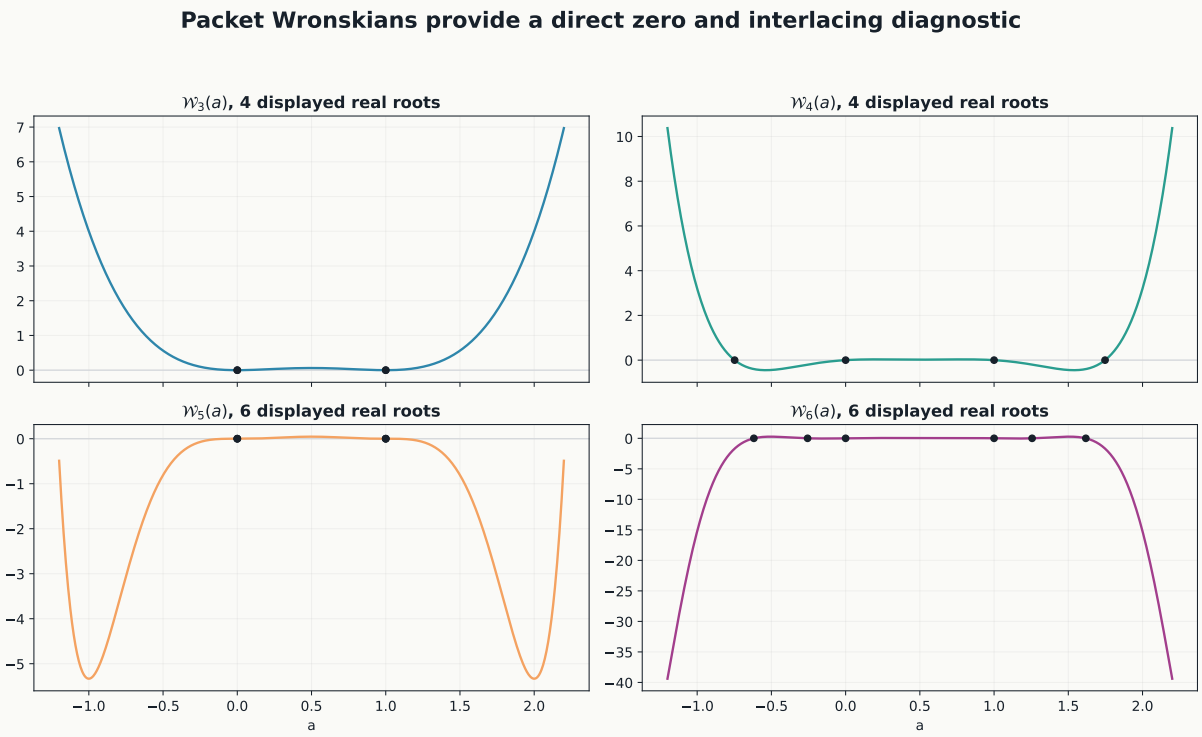


Figure 9.1: Four low-order packet Wronskians. The zeros can be read directly from the ratio derivative and provide a concrete interlacing diagnostic.

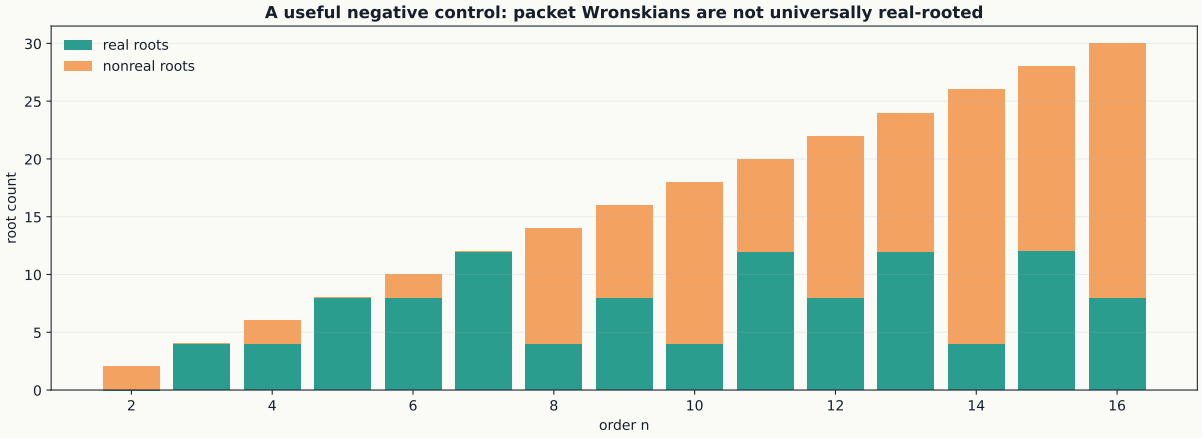


Figure 9.2: Real and nonreal roots for the first fifteen packet Wronskians. The finite computation rejects the naive conjecture that this entire Wronskian family is real-rooted.

Boundary statement 9.3 (What the negative control means). The failure of universal real-rootedness does not remove the Wronskian route. It shows that a successful theorem must impose the right channel, order, parameter, or completed-function normalization. A blanket real-rootedness statement is false.

Chapter 10

A worked cyclotomic norm over the third roots of unity

Use the unnormalized $L = 3$ detail

$$D_1(n) = \sum_{r=0}^{2} \omega_3^{-r} 3^{n-1} B_n\left(\frac{r}{3}\right).$$

Write $D_1 = A_n + B_n\omega_3$. Its field norm is

$$N_n = A_n^2 - A_n B_n + B_n^2.$$

Theorem 10.1 (Parity law for the third-root detail norm)**.** *For every even $n \geq 2$,*

$$D_1(n) = 3^{n-1}\left[B_n - B_n\left(\frac{1}{3}\right)\right] \in \mathbb{Q}$$

and

$$\boxed{N_n = \left\{3^{n-1}\left[B_n - B_n\left(\frac{1}{3}\right)\right]\right\}^2.}$$

For every odd $n \geq 3$,

$$D_1(n) = (\omega_3^2 - \omega_3)3^{n-1} B_n\left(\frac{1}{3}\right)$$

and

$$\boxed{N_n = 3\left[3^{n-1} B_n\left(\frac{1}{3}\right)\right]^2.}$$

Proof. Bernoulli reflection gives $B_n(1-x) = (-1)^n B_n(x)$. For even n, the two child values at $1/3$ and $2/3$ agree, and $\omega_3 + \omega_3^2 = -1$ reduces the detail to the displayed rational number. For odd $n \geq 3$, one has $B_n = 0$ and $B_n(2/3) = -B_n(1/3)$. Since $(\omega_3^2 - \omega_3)(\omega_3 - \omega_3^2) = 3$, the field norm is three times the indicated rational square. □

Table 10.1: Exact first non-DC cyclotomic detail data for $L = 3$ and $a = 0$.

n	A_n	B_n	$N_n = A_n^2 - A_n B_n + B_n^2$	factor pattern
1	$-1/3$	$1/3$	$1/3$	first-order exception
2	$2/3$	0	$4/9$	$9N_n$ is a square
3	$-1/3$	$-2/3$	$1/3$	$3N_n$ is a square
4	$-4/3$	0	$16/9$	$9N_n$ is a square
5	$5/3$	$10/3$	$25/3$	$3N_n$ is a square
6	$26/3$	0	$676/9$	$9N_n$ is a square
7	$-49/3$	$-98/3$	$2401/3$	$3N_n$ is a square
8	$-328/3$	0	$107584/9$	$9N_n$ is a square
9	$809/3$	$1618/3$	$654481/3$	$3N_n$ is a square
10	$6710/3$	0	$45024100/9$	$9N_n$ is a square
11	$-20317/3$	$-40634/3$	$412780489/3$	$3N_n$ is a square
12	$-201772/3$	0	$40711939984/9$	$9N_n$ is a square

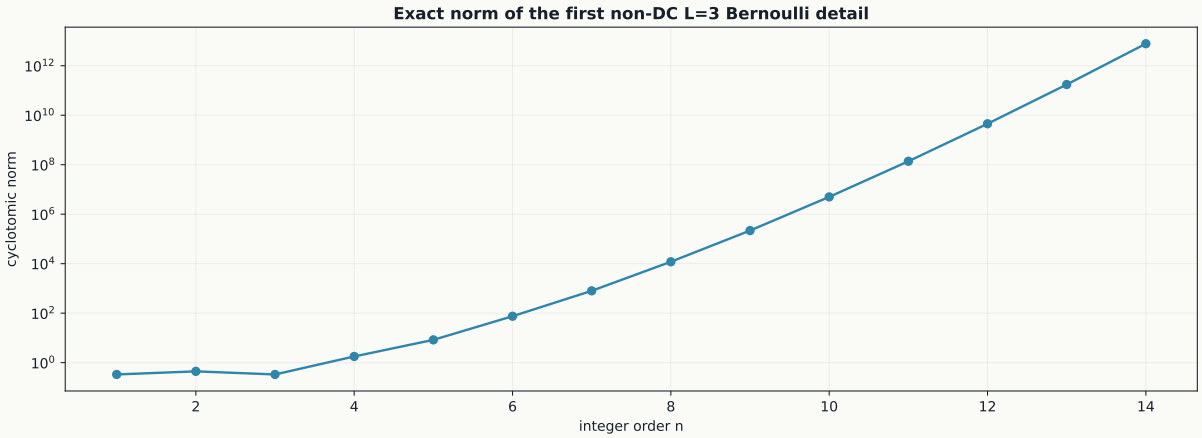


Figure 10.1: Growth of the exact $L = 3$ cyclotomic detail norm. The parity-dependent square pattern follows from Bernoulli reflection symmetry.

Part III

Möbius-Bernoulli transfer functions

Chapter 11

A generalized reciprocal L-function transfer family

Let χ be a Dirichlet character. For parameters $a > 1$, $\beta > 0$, integer $k \geq 0$, and real t, define

$$\boxed{\mathcal{C}_{a,\beta,\chi}(k,t) = \sum_{j=0}^{k} (-1)^j \binom{k}{j} \frac{1}{L(a+\beta j + \mathrm{i}t, \chi)}.}$$

Theorem 11.1 (Möbius form)**.** *In the half-plane of absolute convergence,*

$$\boxed{\mathcal{C}_{a,\beta,\chi}(k,t) = \sum_{n=1}^{\infty} \frac{\mu(n)\chi(n)}{n^{a+\mathrm{i}t}} \left(1 - \frac{1}{n^\beta}\right)^k.}$$

Proof. Use

$$\frac{1}{L(s,\chi)} = \sum_{n \geq 1} \frac{\mu(n)\chi(n)}{n^s}$$

and interchange the finite and absolutely convergent sums. The binomial sum becomes $(1 - n^{-\beta})^k$. □

Proposition 11.2 (Parameter-shift recursion)**.**

$$\boxed{\mathcal{C}_{a,\beta,\chi}(k+1,t) = \mathcal{C}_{a,\beta,\chi}(k,t) - \mathcal{C}_{a+\beta,\beta,\chi}(k,t).}$$

Proof. Apply Pascal's identity to the binomial coefficient and shift the index in the second sum. □

Chapter 12

Exponential generating and Mellin transforms

Define

$$\boxed{\mathcal{R}_{a,\beta,\chi}(x,t) = x \sum_{n=1}^{\infty} \frac{\mu(n)\chi(n)}{n^{a+\mathrm{i}t}} \mathrm{e}^{-x/n^\beta}.}$$

Theorem 12.1 (Poissonization)**.**

$$\boxed{\sum_{k=0}^{\infty} \mathcal{C}_{a,\beta,\chi}(k,t) \frac{x^k}{k!} = \frac{\mathrm{e}^x}{x} \mathcal{R}_{a,\beta,\chi}(x,t).}$$

Proof. For each integer n,

$$\sum_{k \geq 0} \left(1 - \frac{1}{n^\beta}\right)^k \frac{x^k}{k!} = \exp\left[x\left(1 - \frac{1}{n^\beta}\right)\right].$$

Sum over n. □

Theorem 12.2 (Mellin transform)**.** *For absolute convergence, assume*

$$-1 < \Re s < \frac{a-1}{\beta} - 1.$$

The resulting identity extends meromorphically wherever the reciprocal L-function side is defined.

$$\boxed{\int_0^\infty \mathcal{R}_{a,\beta,\chi}(x,t)x^{s-1}\,\mathrm{d}x = \frac{\Gamma(s+1)}{L(a+\mathrm{i}t-\beta(s+1),\chi)}.}$$

Proof. Integrate term by term. The change of variable $x = n^\beta y$ gives

$$\int_0^\infty x^s \mathrm{e}^{-x/n^\beta}\,\mathrm{d}x = n^{\beta(s+1)}\Gamma(s+1).$$

The remaining Dirichlet series is the reciprocal L-function. □

Chapter 13

The zeta specialization and the Bernoulli seed

Take the principal character, $a = 2$, $\beta = 2$, and $t = 0$. Then

$$\boxed{c_k = \mathcal{C}_{2,2}(k,0) = \sum_{j=0}^{k}(-1)^j\binom{k}{j}\frac{1}{\zeta(2j+2)}.}$$

The even zeta values satisfy

$$\frac{1}{\zeta(2j+2)} = \frac{2(2j+2)!}{(2\pi)^{2j+2}\,|B_{2j+2}|}.$$

If

$$q = (2\pi)^{-2}, \qquad A_j = \frac{2(2j+2)!}{|B_{2j+2}|},$$

then

$$\boxed{c_k = q(I-qE)^k A_0.}$$

The Báez-Duarte theorem [3, 4] states that RH is equivalent to

$$\boxed{c_k = O_\varepsilon(k^{-3/4+\varepsilon}) \quad \text{for every } \varepsilon > 0.}$$

Thus RH is a stability statement for repeated weighted finite differences of one normalized Bernoulli seed.

Chapter 14

Scale localization and the Mertens wavelet

Write

$$f_k(x) = x^{-2}(1 - x^{-2})^k.$$

Then

$$c_k = \sum_{n \geq 1} \mu(n) f_k(n).$$

The maximum of f_k occurs at

$$x = \sqrt{k+1}.$$

More generally, the kernel

$$x^{-a}(1 - x^{-\beta})^k$$

is centered at

$$\boxed{x_*(k) = \left(1 + \frac{\beta k}{a}\right)^{1/\beta}.}$$

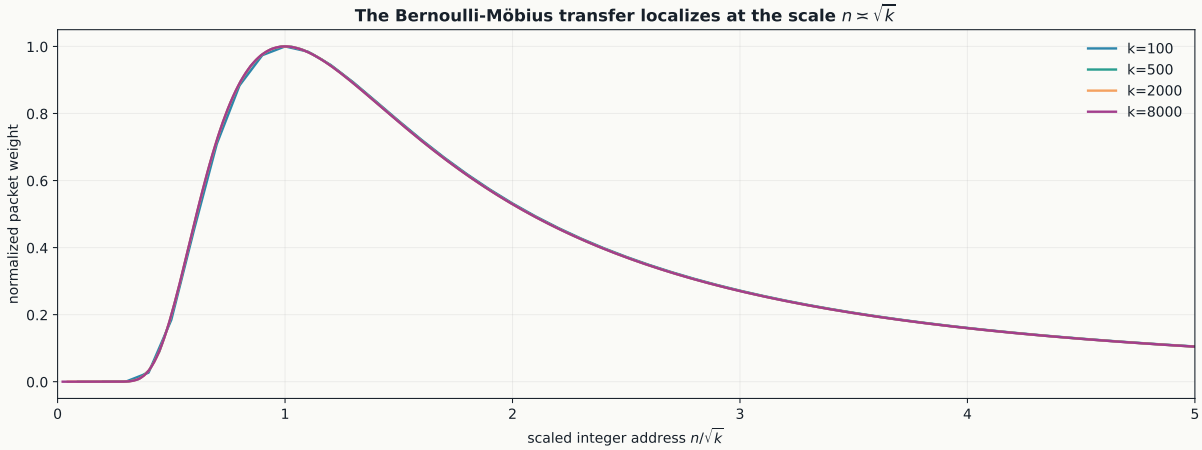


Figure 14.1: Rescaled principal transfer kernels. The curves collapse near one common profile after the coordinate is divided by $\sqrt{k}$.

Let

$$M(x) = \sum_{n \leq x} \mu(n).$$

Discrete summation by parts gives

$$\boxed{c_k = \sum_{n=1}^{\infty} M(n) \left[f_k(n) - f_k(n+1) \right].}$$

The continuous wavelet is

$$\boxed{\psi_k(x) = -f_k'(x) = 2x^{-5}(1 - x^{-2})^{k-1}(x^2 - k - 1).}$$

It has one sign change at $x = \sqrt{k+1}$ and zero integral.

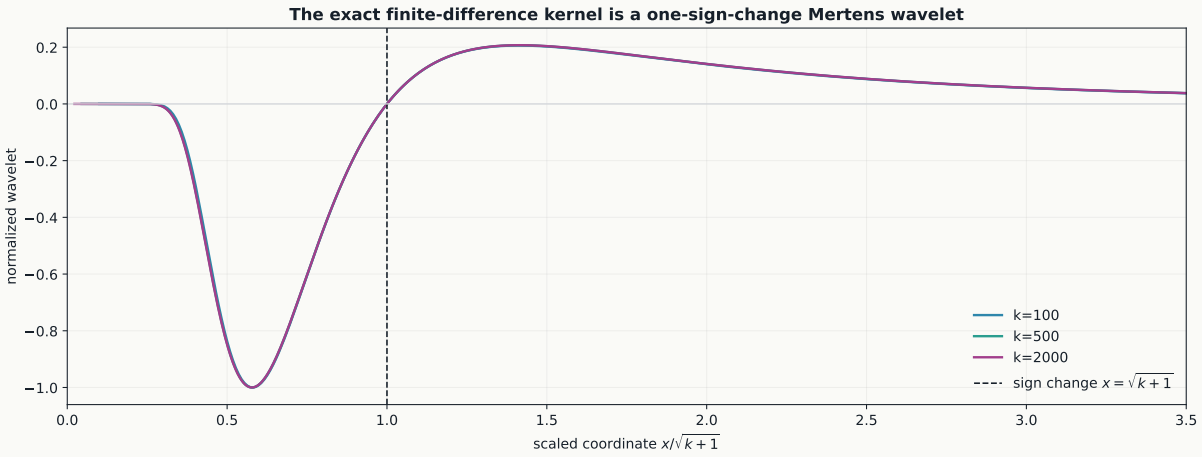


Figure 14.2: The Mertens wavelet at three scales. The sign change remains fixed in the normalized coordinate, making the scale selection explicit.

Theorem 14.1 (Mellin response of the wavelet)**.** *For $k \geq 1$ and $\Re s < 3$,*

$$\boxed{\int_1^\infty \psi_k(x) x^{s-1}\,\mathrm{d}x = \frac{s-1}{2}\mathrm{B}\left(\frac{3-s}{2}, k+1\right).}$$

Proof. First compute

$$\int_1^\infty f_k(x) x^{s-1}\,\mathrm{d}x = \frac{1}{2}\mathrm{B}\left(1-\frac{s}{2}, k+1\right)$$

by the substitution $y = x^{-2}$. Integration by parts transfers the derivative from f_k to x^{s-1}. □

Mellin inversion gives the exact Nörlund-Rice form

$$\boxed{c_k = \frac{1}{4\pi\mathrm{i}}\int_{\sigma-\mathrm{i}\infty}^{\sigma+\mathrm{i}\infty} \frac{\mathrm{B}(1-s/2, k+1)}{\zeta(s)}\,\mathrm{d}s, \qquad 1<\sigma<2.}$$

A simple zero ρ contributes asymptotically

$$\boxed{\frac{\Gamma(1-\rho/2)}{2\zeta'(\rho)} k^{\rho/2-1}.}$$

In logarithmic time $u = \log k$, an off-critical zero creates an exponentially growing normalized mode.

Chapter 15

Diagonal packet energy

Define

$$E_{a,\beta,\chi}(k) = \sum_{n=1}^{\infty} \frac{\mu(n)^2 \left|\chi(n)\right|^2}{n^{2a}} \left(1 - \frac{1}{n^{\beta}}\right)^{2k}.$$

For the asymptotic theorem below, assume that χ is primitive with conductor q, and set

$$\delta_{\chi} = \frac{1}{\zeta(2)} \prod_{p \mid q} \frac{p}{p+1}.$$

Theorem 15.1 (Packet-energy asymptotic)**.** *For $a > 1/2$,*

$$\boxed{E_{a,\beta,\chi}(k) \sim \frac{\delta_{\chi}}{\beta} \Gamma\left(\frac{2a-1}{\beta}\right) (2k)^{-(2a-1)/\beta}.}$$

Proof. The summatory density of squarefree integers coprime to q is

$$\sum_{n \le x} \mu(n)^2 \left|\chi(n)\right|^2 = \delta_{\chi} x + O_q(\sqrt{x}).$$

Use partial summation. The kernel is concentrated at $n \asymp k^{1/\beta}$. On this scale,

$$\left(1 - n^{-\beta}\right)^{2k} = \exp(-2kn^{-\beta})(1 + o(1)).$$

After the scaling $n = (2k)^{1/\beta} y$, the main term becomes

$$\delta_{\chi} (2k)^{-(2a-1)/\beta} \int_0^{\infty} y^{-2a} \mathrm{e}^{-y^{-\beta}} \, \mathrm{d}y.$$

The substitution $v = y^{-\beta}$ evaluates the integral as

$$\frac{1}{\beta} \Gamma\left(\frac{2a-1}{\beta}\right).$$

The square-root error in the summatory function is lower order after the same localization. □

For $a = \beta = 2$,

$$\boxed{E_k \sim \frac{3}{2^{5/2} \pi^{3/2}} k^{-3/2}.}$$

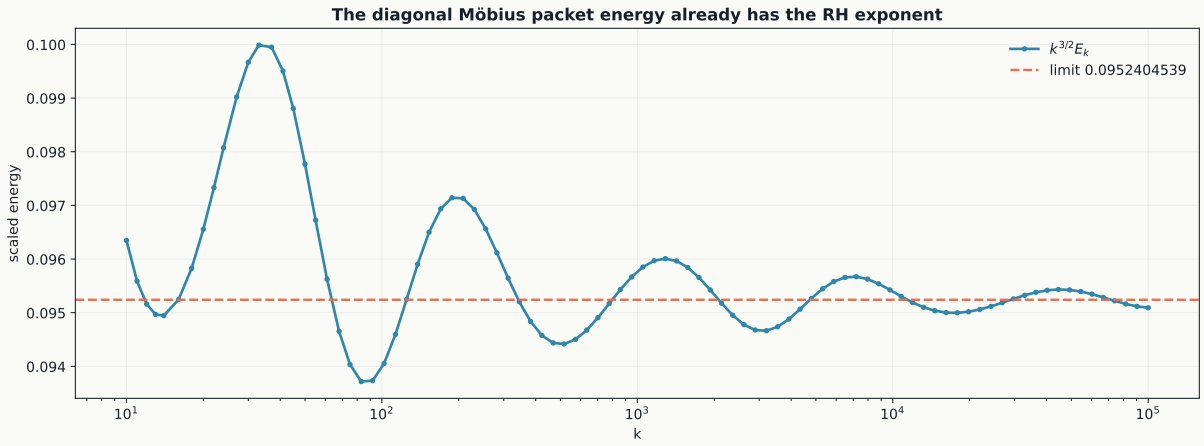


Figure 15.1: The scaled energy $k^{3/2}E_k$. The numerical curve approaches the exact constant predicted by the theorem.

Define also the active squarefree mass

$$S_k = \sum_{n\geq 1} \frac{\mu(n)^2}{n^2}\left(1-\frac{1}{n^2}\right)^k.$$

The same argument gives

$$S_k \sim \frac{3}{\pi^{3/2}}k^{-1/2}.$$

Therefore

$$\boxed{\frac{S_k^2}{E_k} \sim \frac{3\cdot 2^{5/2}}{\pi^{3/2}}\sqrt{k}.}$$

This is the effective number of active packets.

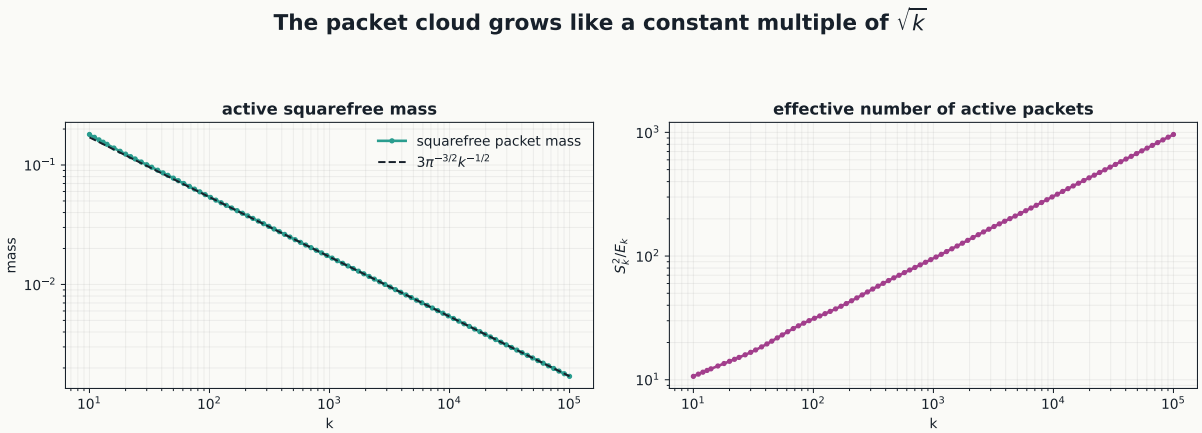


Figure 15.2: Active squarefree mass and effective dimension. The number of packets contributing at one scale grows like a constant multiple of $\sqrt{k}$.

Chapter 16

Möbius packet coherence and RH

Define

$$\boxed{\Gamma_k = \frac{|c_k|^2}{E_k}.}$$

Theorem 16.1 (Coherence form of the Báez-Duarte criterion)**.** *The Riemann hypothesis is equivalent to*

$$\boxed{\Gamma_k = O_\varepsilon(k^\varepsilon) \quad \textit{for every } \varepsilon > 0.}$$

Proof. The energy theorem gives $E_k \asymp k^{-3/2}$. If RH holds, the Báez-Duarte criterion gives $|c_k| \ll_\varepsilon k^{-3/4+\varepsilon}$. Hence $\Gamma_k \ll_\varepsilon k^{2\varepsilon}$, which is subpolynomial after renaming 2ε. Conversely, subpolynomial coherence gives

$$|c_k|^2 \ll_\varepsilon k^{-3/2+\varepsilon},$$

so $|c_k| \ll_\varepsilon k^{-3/4+\varepsilon}$ after another harmless renaming. □

The theorem separates the problem into two parts. The diagonal energy decay is already proved. The remaining RH content is the off-diagonal arithmetic coherence of the Möbius signs.

Table 16.1: Selected numerical values of the principal Möbius-Bernoulli transfer.

k	c_k	E_k	Γ_k	S_k^2/E_k
10	-6.91391e-02	3.04674e-03	1.56896e+00	1.06483e+01
50	-5.36535e-03	2.76532e-04	1.04100e-01	2.17944e+01
102	-1.42289e-03	9.13011e-05	2.21752e-02	3.14049e+01
517	-5.96831e-05	8.03172e-06	4.43501e-04	6.99561e+01
950	-1.84349e-05	3.26717e-06	1.04018e-04	9.34893e+01
4800	-8.04548e-07	2.86464e-07	2.25961e-06	2.10639e+02
9750	-1.33337e-07	9.91180e-08	1.79370e-07	3.00488e+02
49238	1.69527e-08	8.73393e-09	3.29055e-08	6.73612e+02
100000	1.82845e-09	3.00705e-09	1.11179e-09	9.61804e+02

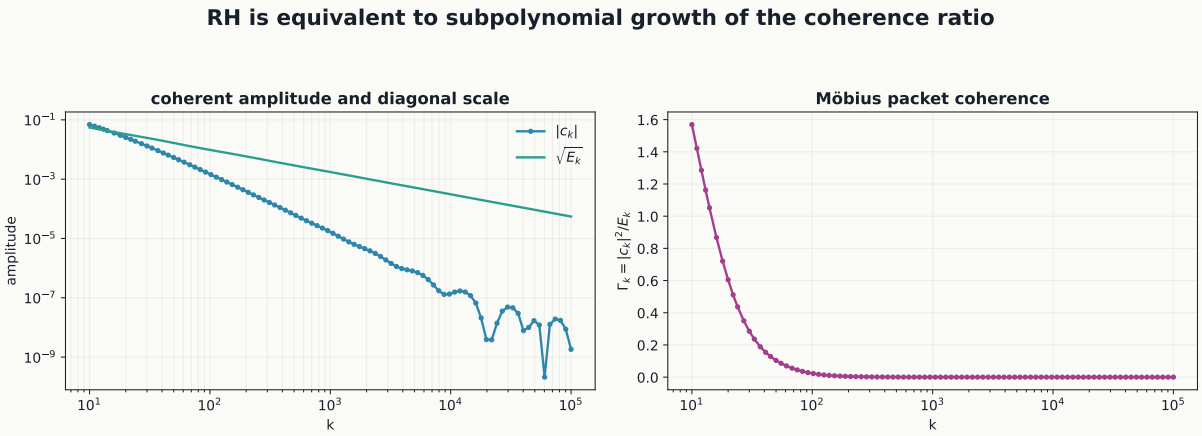


Figure 16.1: The principal transfer amplitude, diagonal energy scale, and coherence ratio. The numerical values are finite diagnostics only. They do not establish the required asymptotic bound.

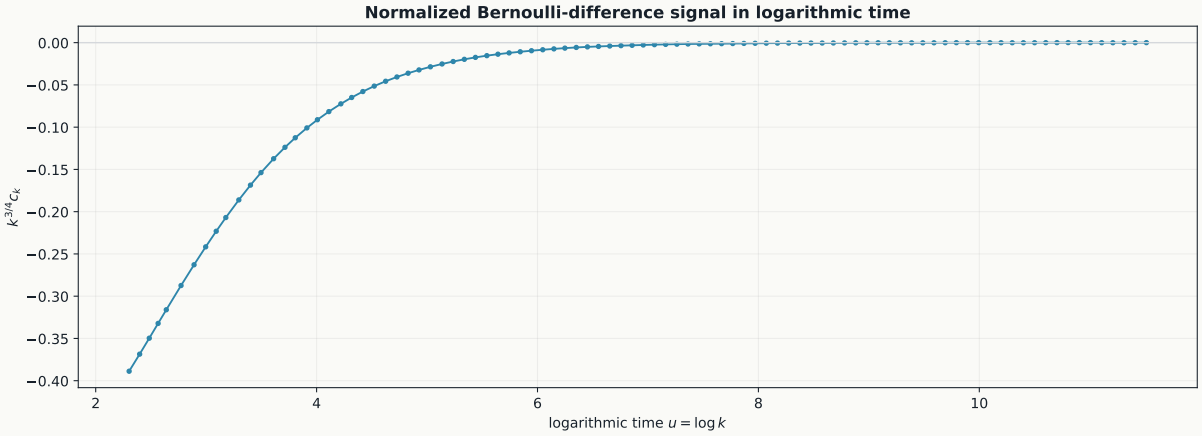


Figure 16.2: The normalized sequence $k^{3/4}c_k$ in logarithmic time. Zeros off the critical line would appear as expanding modes in this coordinate.

Boundary statement 16.2 (Why a generic frame estimate cannot prove RH). If every active packet had the same sign, the coherent-to-diagonal ratio would grow like $\sqrt{k}$. The missing amplitude factor is $k^{1/4}$. Therefore Cauchy-Schwarz, Parseval, positivity, or a sign-free frame bound cannot prove RH. Any successful estimate must use arithmetic structure of $\mu(n)$.

Chapter 17

Long-time mean power and a quantum correlation

Define

$$F_k(t) = \sum_{n=1}^{\infty} \frac{\mu(n)}{n^{2+\mathrm{i}t}} \left(1 - \frac{1}{n^2}\right)^k.$$

Then

$$F_k(0) = c_k.$$

Theorem 17.1 (Long-time diagonalization)**.**

$$\boxed{\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |F_k(t)|^2 \, \mathrm{d}t = E_k.}$$

Proof. The frequencies $\log n$ are distinct. For a finite truncation, every off-diagonal time average tends to zero. The coefficient sequence is square summable, so the truncation tail is controlled in mean square. Pass to the limit. □

Therefore

$$\boxed{\Gamma_k = \frac{|F_k(0)|^2}{\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |F_k(t)|^2 \, \mathrm{d}t}.}$$

RH says that the zero-time recurrence is only subpolynomially larger than the long-time mean power.

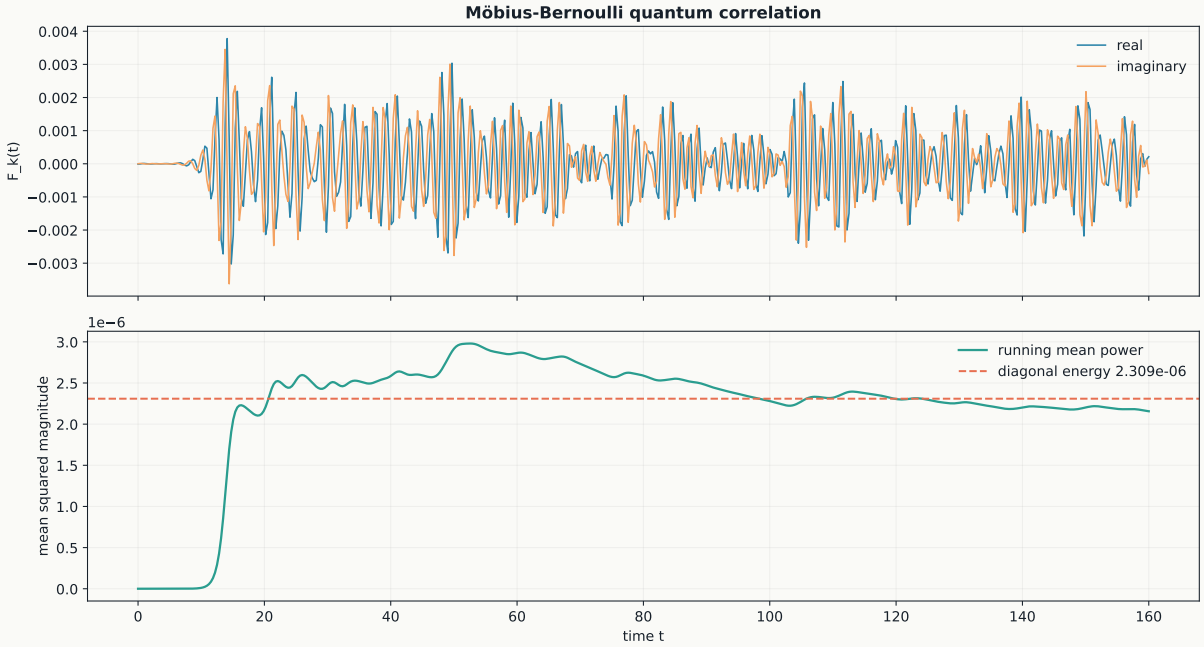


Figure 17.1: One finite Möbius-Bernoulli correlation and its running mean power. The dashed line is the exact diagonal energy of the truncated packet family.

For $a > 1$ and $k \geq 1$, work in $\ell^2(\{2, 3, \ldots\})$. Define

$$|\phi_a\rangle = \sum_{n \geq 2} n^{-a/2} |n\rangle,$$

$$H|n\rangle = \log n |n\rangle,$$
$$K_\beta |n\rangle = -\log(1 - n^{-\beta})|n\rangle,$$

and

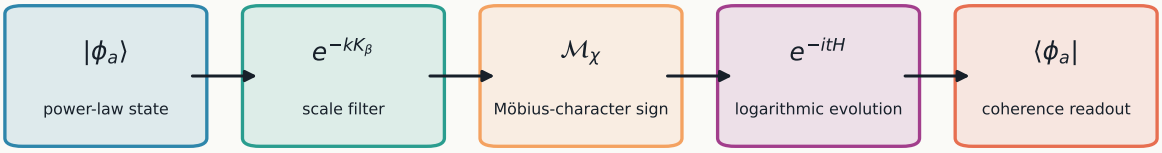
$$\mathcal{M}_\chi |n\rangle = \mu(n)\chi(n)|n\rangle.$$

Then

$$\boxed{\mathcal{C}_{a,\beta,\chi}(k,t) = \langle \phi_a | \mathcal{M}_\chi \mathrm{e}^{-kK_\beta} \mathrm{e}^{-\mathrm{i}tH} | \phi_a \rangle.}$$

The logarithmic Hamiltonian H is the same spectral object used in the recent zeta-governed quantum-dynamics construction of Wei and collaborators [7]. That work builds thermal weights and phases from the spectrum $\log n$. The new ingredients here are the Möbius-character observable and the scale filter e^{-kK_β}. A complete gate construction would require a reversible Möbius or squarefree-phase oracle, which remains an engineering and complexity question.

$$C_{a,\beta,\chi}(k,t) = \langle \phi_a | \mathcal{M}_\chi e^{-kK_\beta} e^{-itH} | \phi_a \rangle$$



$H|n\rangle = \log n|n\rangle$ and $K_\beta = -\log(I - e^{-\beta H})$

Figure 17.2: Operator realization of the generalized transfer function. Scale evolution and real-time logarithmic evolution commute because both are diagonal in the integer basis.

Chapter 18

Arithmetic polyphase and character packets

For a dyadic scale shell $I_J = [2^J, 2^{J+1})$ and a residue class $r \pmod M$, define

$$\boxed{F_{k;J,M,r}(t) = \sum_{\substack{n \in I_J \\ n \equiv r \pmod M}} \frac{\mu(n)}{n^{2+it}} \left(1 - \frac{1}{n^2}\right)^k.}$$

Then

$$F_k(t) = \sum_J \sum_{r=0}^{M-1} F_{k;J,M,r}(t).$$

Arithmetic refinement is exact:

$$\boxed{F_{k;J,M,r} = \sum_{\ell=0}^{L-1} F_{k;J,LM,r+\ell M}.}$$

Scale refinement is exact because each shell splits into two half-shells. The diagonal energy also separates exactly over every complete packet family.

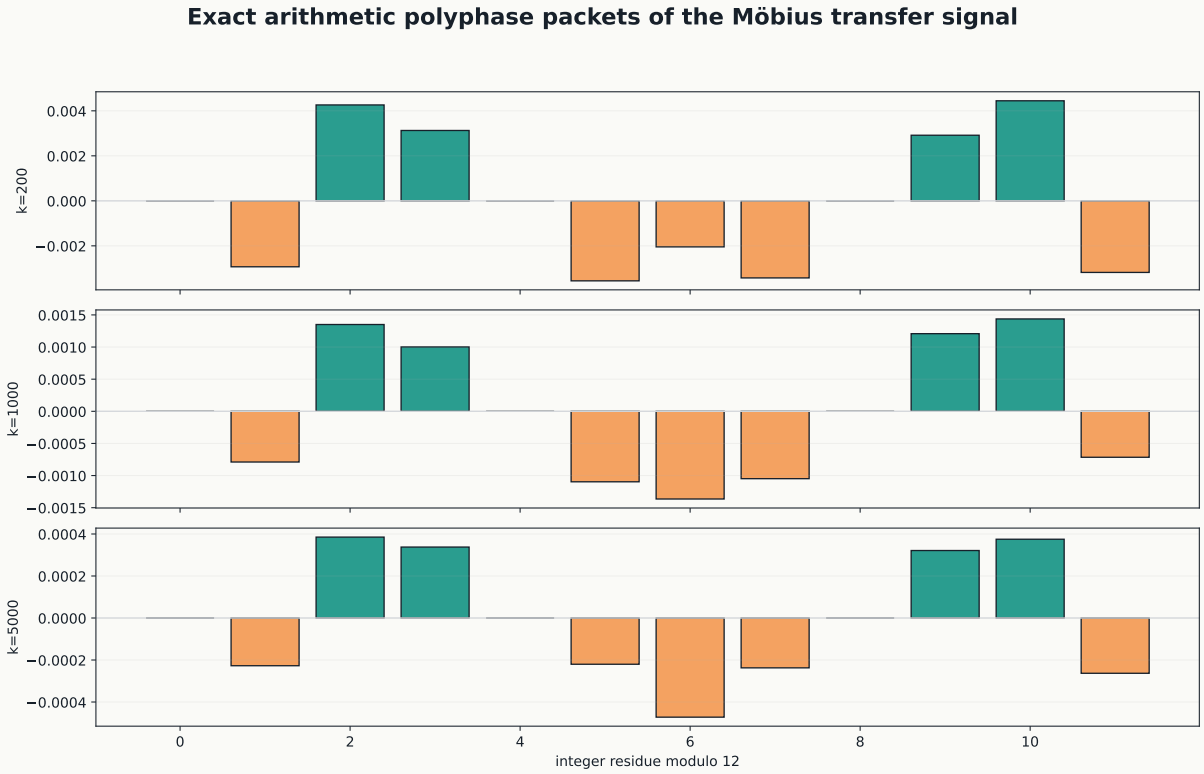


Figure 18.1: Residue packets of the principal Möbius transfer at three scales. Positive and negative packets shift with k, while their exact sum remains c_k.

For a Dirichlet character χ, the character transfer

$$\mathcal{C}_{a,\beta,\chi}(k,t) = \sum_{n\geq 1} \frac{\mu(n)\chi(n)}{n^{a+it}}(1-n^{-\beta})^k$$

provides a multiplicative-frequency coordinate. Known Riesz-type theorems give GRH criteria for continuous exponential kernels and related L-function classes [5, 6]. The corresponding discrete coherence criterion is a natural target, but no character-uniform theorem is claimed here.

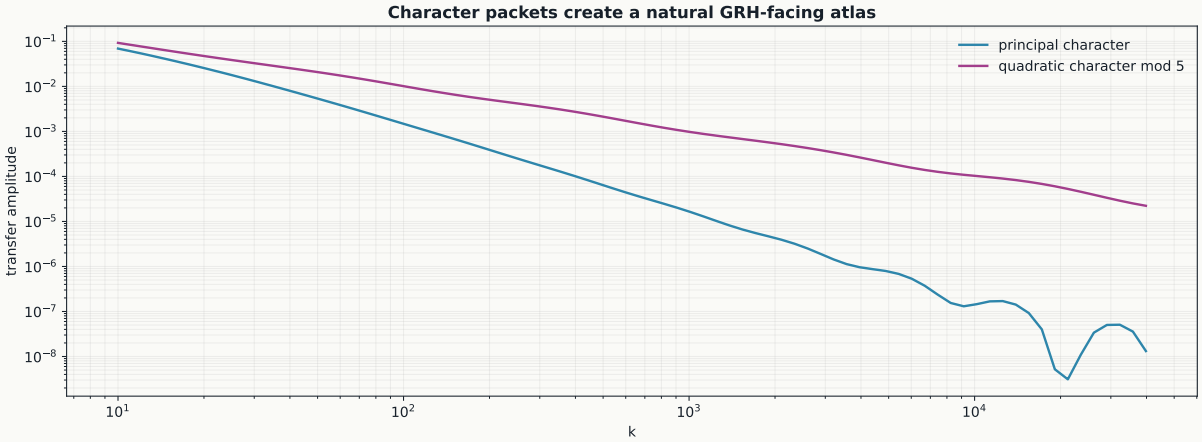


Figure 18.2: Principal and quadratic-character transfer amplitudes. The character coordinate can strongly alter cancellation even when the same scale filter is used.

Part IV

Worked computational atlas and research program

Chapter 19

Worked example: complete three-channel exponential lift

Start from

$$e^z = \sum_{n\geq 0} \frac{z^n}{n!}.$$

Separate the exponents modulo three:

$$E_{3,0}(u) = 1 + \frac{u}{3!} + \frac{u^2}{6!} + \cdots,$$
$$E_{3,1}(u) = 1 + \frac{u}{4!} + \frac{u^2}{7!} + \cdots,$$
$$E_{3,2}(u) = \frac{1}{2!} + \frac{u}{5!} + \frac{u^2}{8!} + \cdots.$$

The packet system is

$$\Theta_3 \begin{pmatrix} E_{3,0} \\ E_{3,1} \\ E_{3,2} \end{pmatrix} = \begin{pmatrix} 0 & 0 & u \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} E_{3,0} \\ E_{3,1} \\ E_{3,2} \end{pmatrix}.$$

Writing the rows explicitly gives

$$3uE'_{3,0} = uE_{3,2},$$
$$(3u\partial_u + 1)E_{3,1} = E_{3,0},$$
$$(3u\partial_u + 2)E_{3,2} = E_{3,1}.$$

Reconstruction yields

$$e^z = E_{3,0}(z^3) + zE_{3,1}(z^3) + z^2E_{3,2}(z^3).$$

Every equation can be checked coefficient by coefficient. The matrix system is not a numerical approximation. It is the scalar equation in an exact residue coordinate system.

Chapter 20

Worked example: Bernoulli packet refinement

Take $a = 1/3$, $M = 2$, and refine by $L = 2$. The parent packet is

$$\mathfrak{B}_{2,1}(z; 1/3) = -z2^{z-1}\zeta\left(1 - z, \frac{2}{3}\right).$$

Its two children are

$$\mathfrak{B}_{4,1}(z; 1/3) = -z4^{z-1}\zeta\left(1 - z, \frac{1}{3}\right)$$

and

$$\mathfrak{B}_{4,3}(z; 1/3) = -z4^{z-1}\zeta\left(1 - z, \frac{5}{6}\right).$$

The multiplication formula at modulus two gives

$$\zeta\left(1 - z, \frac{1}{3}\right) + \zeta\left(1 - z, \frac{5}{6}\right) = 2^{1-z}\zeta\left(1 - z, \frac{2}{3}\right).$$

Multiplication by $-z4^{z-1}$ gives

$$\boxed{\mathfrak{B}_{2,1}(z; 1/3) = \mathfrak{B}_{4,1}(z; 1/3) + \mathfrak{B}_{4,3}(z; 1/3).}$$

At $z = 6$, this becomes an exact identity among rational Bernoulli-polynomial values.

Chapter 21

Worked example: third-root cyclotomic norm

For $n = 8$, the three integer child packets at $a = 0$ are

$$-\frac{729}{10}, \qquad \frac{1093}{30}, \qquad \frac{1093}{30}.$$

Let $\omega^2 + \omega + 1 = 0$. The first non-DC detail is

$$D_1 = -\frac{729}{10} + \omega^2\frac{1093}{30} + \omega\frac{1093}{30} = -\frac{328}{3}.$$

Its cyclotomic norm is

$$\boxed{N_8 = D_1\overline{D_1} = \frac{107584}{9} = \left(\frac{328}{3}\right)^2.}$$

For odd $n \geq 3$, the same calculation produces three times a rational square. The table in table 10.1 records the first values.

Chapter 22

Worked example: the principal transfer at order 1200

For $k = 1200$, define

$$w_n = \frac{\mu(n)}{n^2}\left(1 - \frac{1}{n^2}\right)^{1200}.$$

The finite computation to $n = 200000$ gives a zero-time amplitude of approximately

$$|F_{1200}(0)| = 1.1153 \times 10^{-5}.$$

The diagonal packet power is approximately

$$E_{1200} = 2.3091 \times 10^{-6}.$$

Hence

$$\Gamma_{1200} \approx 5.3868 \times 10^{-5}.$$

This is one finite numerical value. It is not an asymptotic theorem. Its purpose is to show how the exact definitions become a measurable signal-to-energy ratio.

Chapter 23

Algorithmic workflow

Algorithm 23.1 (Generating a polyphase special-function system)**.** Given a scalar power series or differential equation:

1. Choose a modulus M and separate Taylor exponents by residue.
2. Form $\mathbf{F}_M(u)$.
3. Replace multiplication by z with $C_M(u)$.
4. Replace $z\frac{\mathrm{d}}{\mathrm{d}z}$ with Θ_M.
5. Preserve operator order and derive the matrix system.
6. If a parent-child identity is available, refine packet addresses.
7. Apply a unitary transform to create low-pass and detail channels.
8. Derive inverse synthesis and energy identities.
9. Build Wronskians, recurrences, arithmetic traces, or coherence ratios suited to the question.
10. Test simple conjectures against exact and high-precision counterexamples before promoting them.

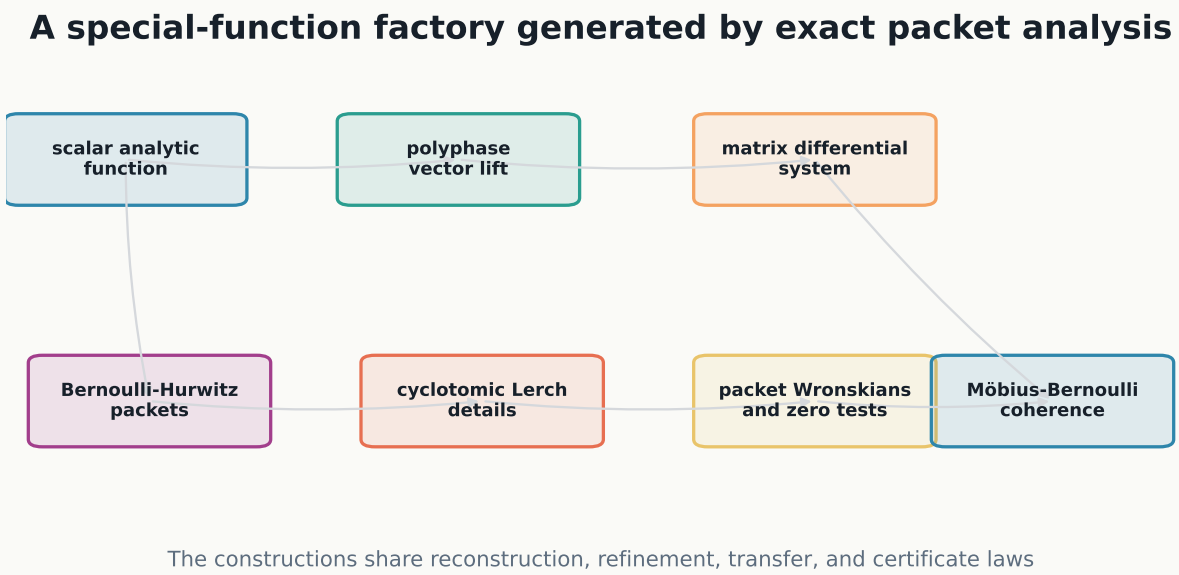


Figure 23.1: The generated function families and their main transfer routes. The same packet operations produce differential systems, refinable Bernoulli packets, cyclotomic details, Wronskians, and arithmetic coherence signals.

Chapter 24

Open problems generated by the calculus

24.1 Near-term problems

P1. Determine sharp denominator and local valuation laws for cyclotomic Bernoulli details at rational a.

P2. Classify the orders and parameter intervals for which selected packet Wronskians have one sign.

P3. Determine when Bernoulli and Euler channel zeros interlace after completion or weighting.

P4. Develop dominant-pole asymptotics for reciprocal and logarithmic derivatives of non-DC detail functions.

P5. Find minimal matrix differential systems after symmetry reduction of the general polyphase lift.

P6. Identify transforms for which packet children form a positive or totally positive kernel.

24.2 Major-conjecture interfaces

M1. Prove subpolynomial Möbius packet coherence in the principal zeta case.

M2. Establish a character-uniform version compatible with GRH criteria.

M3. Construct positive canonical systems from packet Gram matrices.

M4. Connect local p-adic packet estimates to Archimedean coherence through a quantitative global height theorem.

Conjecture 24.1 (Möbius packet coherence)**.** *For every $\varepsilon > 0$,*

$$\boxed{\left|\sum_{n=1}^{\infty} \frac{\mu(n)}{n^2}\left(1-\frac{1}{n^2}\right)^k\right|^2 \le C_\varepsilon k^\varepsilon \sum_{n=1}^{\infty} \frac{\mu(n)^2}{n^4}\left(1-\frac{1}{n^2}\right)^{2k}.}$$

By the proved energy asymptotic and the Báez-Duarte criterion, this conjecture is equivalent to RH. It is included as the sharp theorem target, not as a claimed result.

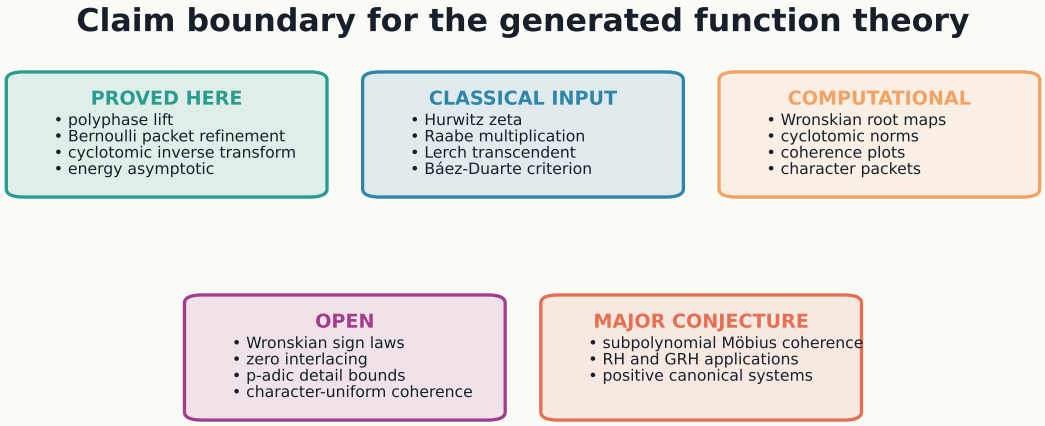


Figure 24.1: Status map for the paper. The map separates proved identities, classical inputs, finite computations, open special-function questions, and major conjecture interfaces.

Chapter 25

Verification

Every figure and numerical table was generated by the accompanying Python source. The symbolic suite verifies coefficient reconstruction, the Weyl commutator, hypergeometric exponential packets, Bernoulli and Euler channel identities, complex-order Bernoulli reconstruction and refinement, Lerch closed forms, and packet Wronskian formulas.

The asymptotic plots use a Möbius sieve through 200000. The largest plotted transfer order is 100000. These computations illustrate the proved formulas and expose possible failure modes. They do not replace the asymptotic theorems or the open coherence estimate.

Table 25.1: Exact and high-precision verification summary.

Check family	checks	passed
Bernoulli	15	15
Euler	9	9
Lerch	3	3
Weyl	4	4
exponential	9	9
packet	7	7
polynomial	4	4
Total	51	51

Chapter 26

Conclusion

The calculus generates a coherent special-function theory rather than a list of renamed scalar functions. The defining structure is:

analysis $\longrightarrow$ matrix transfer $\longrightarrow$ refinement $\longrightarrow$ unitary detail transform $\longrightarrow$ synthesis $\longrightarrow$ certificate.

The general polyphase lift transports scalar differential equations into exact matrix systems. Bernoulli-Hurwitz packets extend Raabe multiplication to complex order. Their cyclotomic transform produces Lerch details with perfect reconstruction. The two-channel case unifies Bernoulli and Euler polynomials. Packet Wronskians create direct zero diagnostics and also reveal false broad conjectures. The Möbius-Bernoulli transfer family turns reciprocal L-values into scale-localized signals whose diagonal energy is explicit. In the zeta specialization, RH is precisely the remaining coherence estimate.

The practical next step is not to introduce more notation. It is to prove one nontrivial property of a generated family that is not automatic from the scalar parent. The most accessible targets are denominator laws, Wronskian signs, symmetry reductions, and reciprocal-detail recurrences. The most difficult target is the Möbius coherence inequality.

Chapter A

Formula registry

Object	Defining formula
Polyphase section	$F_{M,r}(u) = \sum_{m\ge 0} a_{Mm+r} u^m$
Carry shift	$C_M(u)_{0,M-1} = u$, $C_M(u)_{r,r-1} = 1$
Packet Euler operator	$\Theta_M = Mu\partial_u I + \operatorname{diag}(0,\dots,M-1)$
Bernoulli interpolation	$\mathfrak{B}(z;a) = -z\zeta(1-z,a)$
Bernoulli packet	$\mathfrak{B}_{M,r}(z;a) = -zM^{z-1}\zeta(1-z,(a+r)/M)$
Cyclotomic detail	$\mathcal{W}_{M,r}^{(L,j)} = L^{-1/2}\sum_\ell \omega_L^{-j\ell}\mathfrak{B}_{LM,r+\ell M}$
Cyclotomic EGF	$\sum_n W_{L,j}(n;a)t^n/n! = te^{at}/[\sqrt{L}(\omega_L^{-j}e^t-1)]$
Packet Wronskian	$\mathcal{D}_{j,\ell} = \mathcal{W}_j(z)\mathcal{W}_\ell(z-1) - \mathcal{W}_\ell(z)\mathcal{W}_j(z-1)$
General transfer	$\mathcal{C}_{a,\beta,\chi}(k,t) = \sum_{j=0}^k (-1)^j \binom{k}{j} / L(a+\beta j+\mathrm{i}t,\chi)$

Object	Defining formula
Möbius form	$\mathcal{C} = \sum_n \mu(n)\chi(n)n^{-a-\mathrm{i}t}(1-n^{-\beta})^k$
Continuous companion	$\mathcal{R} = x\sum_n \mu(n)\chi(n)n^{-a-\mathrm{i}t}\mathrm{e}^{-x/n^\beta}$
Packet energy	$E_{a,\beta,\chi} = \sum_n \mu(n)^2\lvert\chi(n)\rvert^2 n^{-2a}(1-n^{-\beta})^{2k}$
Coherence ratio	$\Gamma = \lvert\mathcal{C}\rvert^2/E$

Chapter B

Complete Python source

The following program generates every figure, table, and numerical record used in this paper.

```
from __future__ import annotations

import argparse
import cmath
import csv
import hashlib
import json
import math
from dataclasses import dataclass
from pathlib import Path
from typing import Iterable, Sequence

import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import FancyArrowPatch, FancyBboxPatch, Rectangle
import mpmath as mp
import numpy as np
import sympy as sp
from scipy.special import gamma

mp.mp.dps = 40

ROOT = Path(__file__).resolve().parent
FIGURES = ROOT / "figures"
RESULTS = ROOT / "results"
TABLES = ROOT / "tables"

for directory in (FIGURES, RESULTS, TABLES):
    directory.mkdir(parents=True, exist_ok=True)

INK = "#17202A"
BLUE = "#2E86AB"
TEAL = "#2A9D8F"
ORANGE = "#F4A261"
MAGENTA = "#A23E8C"
RED = "#E76F51"
YELLOW = "#E9C46A"
MUTED = "#5D6D7E"
PAPER = "#FAFAF7"
GRID = "#D5D8DC"
COLORS = [BLUE, TEAL, ORANGE, MAGENTA, RED, YELLOW]

plt.rcParams.update(
    {
        "figure.facecolor": PAPER,
        "axes.facecolor": PAPER,
        "axes.edgecolor": INK,
        "axes.labelcolor": INK,
        "xtick.color": INK,
        "ytick.color": INK,
```

```
        "text.color": INK,
        "font.size": 10.5,
        "axes.titlesize": 12.5,
        "axes.titleweight": "bold",
        "legend.frameon": False,
        "savefig.bbox": "tight",
        "savefig.facecolor": PAPER,
    }
)


def save_figure(fig: plt.Figure, stem: str) -> None:
    fig.savefig(FIGURES / f"{stem}.pdf")
    fig.savefig(FIGURES / f"{stem}.png", dpi=220)
    plt.close(fig)


def complex_array(values: Iterable[complex]) -> np.ndarray:
    return np.asarray([complex(value) for value in values], dtype=np.complex128)


def mobius_sieve(nmax: int) -> np.ndarray:
    """Linear sieve for the Mobius function on 0..nmax."""
    mu = np.zeros(nmax + 1, dtype=np.int8)
    mu[1] = 1
    primes: list[int] = []
    composite = np.zeros(nmax + 1, dtype=bool)
    for i in range(2, nmax + 1):
        if not composite[i]:
            primes.append(i)
            mu[i] = -1
        for p in primes:
            ip = i * p
            if ip > nmax:
                break
            composite[ip] = True
            if i % p == 0:
                mu[ip] = 0
                break
            mu[ip] = -mu[i]
    return mu


# ---------------------------------------------------------------------------
# General polyphase lift
# ---------------------------------------------------------------------------


def carry_matrix_symbolic(m: int, u: sp.Symbol) -> sp.Matrix:
    matrix = sp.zeros(m)
    matrix[0, m - 1] = u
    for row in range(1, m):
        matrix[row, row - 1] = 1
    return matrix


def polyphase_sections_from_coefficients(coeffs: Sequence[sp.Expr], m: int, u: sp.Symbol) -> sp.Matrix:
    sections: list[sp.Expr] = []
    for r in range(m):
        section = sp.Integer(0)
        q = 0
        index = r
        while index < len(coeffs):
            section += coeffs[index] * u**q
            q += 1
            index += m
        sections.append(sp.expand(section))
    return sp.Matrix(sections)


def reconstruct_sections(sections: Sequence[sp.Expr], m: int, z: sp.Symbol) -> sp.Expr:
```

```
    u_value = z**m
    return sp.expand(sum(z**r * sections[r].subs({sp.Symbol("u"): u_value}) for r in range(m)))


def theta_apply(section: sp.Expr, r: int, m: int, u: sp.Symbol) -> sp.Expr:
    return sp.expand(m * u * sp.diff(section, u) + r * section)


def theta_vector_apply(vector: sp.Matrix, m: int, u: sp.Symbol) -> sp.Matrix:
    return sp.Matrix([theta_apply(vector[r], r, m, u) for r in range(m)])


def exp_section_hypergeometric(m: int, r: int, u: complex) -> complex:
    params = [mp.mpf(r + j) / m for j in range(1, m + 1)]
    params = [value for value in params if abs(value - 1) > mp.mpf("1e-40")]
    return mp.hyper([], params, u / (m**m)) / mp.factorial(r)


def exp_section_series(m: int, r: int, u: complex, terms: int = 80) -> complex:
    return mp.nsum(lambda q: u**q / mp.factorial(m * q + r), [0, mp.inf]) if terms <= 0 else sum(
        mp.mpf(u) ** q / mp.factorial(m * q + r) for q in range(terms)
    )


# ---------------------------------------------------------------------------
# Bernoulli-Hurwitz packet functions and cyclotomic details
# ---------------------------------------------------------------------------


def bernoulli_interpolation(z: complex, a: complex = 1.0) -> complex:
    if abs(z) < 1e-20:
        return 1.0 + 0.0j
    zz = mp.mpc(z)
    return complex(-zz * mp.zeta(1 - zz, a))


def bernoulli_packet(z: complex, a: complex, m: int, r: int) -> complex:
    if abs(z) < 1e-20:
        return 1.0 / m
    argument = (a + r) / m
    zz = mp.mpc(z)
    return complex(-zz * (m ** (zz - 1)) * mp.zeta(1 - zz, argument))


def cyclotomic_detail(z: complex, a: complex, m: int, r: int, ell: int, j: int) -> complex:
    omega = cmath.exp(2j * math.pi / ell)
    values = [bernoulli_packet(z, a, m * ell, r + child * m) for child in range(ell)]
    return sum((omega ** (-j * child)) * values[child] for child in range(ell)) / math.sqrt(ell)


def cyclotomic_detail_lerch(z: complex, a: complex, m: int, r: int, ell: int, j: int) -> complex:
    omega = mp.e ** (2j * mp.pi / ell)
    argument = (a + r) / m
    return complex(
        -mp.mpc(z)
        / mp.sqrt(ell)
        * (m ** (mp.mpc(z) - 1))
        * mp.lerchphi(omega ** (-j), 1 - mp.mpc(z), argument)
    )


def cyclotomic_egf(t: complex, a: complex, ell: int, j: int) -> complex:
    omega = cmath.exp(2j * math.pi / ell)
    denominator = omega ** (-j) * cmath.exp(t) - 1
    if abs(t) < 1e-14 and j == 0:
        return 1 / math.sqrt(ell)
    return t * cmath.exp(a * t) / (math.sqrt(ell) * denominator)


def bernoulli_euler_channels(n: int, a: sp.Expr) -> tuple[sp.Expr, sp.Expr]:
    low = sp.bernoulli(n, a) / sp.sqrt(2)
```

```
    if n == 0:
        high = sp.Integer(0)
    else:
        high = -sp.Rational(n, 2) * sp.euler(n - 1, a) / sp.sqrt(2)
    return sp.simplify(low), sp.simplify(high)


def packet_wronskian_polynomial(n: int, x: sp.Symbol) -> sp.Expr:
    if n < 2:
        raise ValueError("n must be at least 2")
    return sp.expand(
        n * sp.bernoulli(n - 1, x) * sp.euler(n - 1, x)
        - (n - 1) * sp.bernoulli(n, x) * sp.euler(n - 2, x)
    )


# ---------------------------------------------------------------------------
# Mobius-Bernoulli transfer family
# ---------------------------------------------------------------------------


def principal_transfer_values(
    k_values: Sequence[int],
    nmax: int,
    mu: np.ndarray | None = None,
) -> dict[str, np.ndarray]:
    if mu is None:
        mu = mobius_sieve(nmax)
    n = np.arange(2, nmax + 1, dtype=np.float64)
    mu_n = mu[2:].astype(np.float64)
    log_lambda = np.log1p(-1.0 / (n * n))
    base = 1.0 / (n * n)
    squarefree = mu_n * mu_n
    c_values = []
    energy_values = []
    mass_values = []
    for k in k_values:
        weight = base * np.exp(k * log_lambda)
        c_values.append(float(np.dot(mu_n, weight)))
        energy_values.append(float(np.dot(squarefree, weight * weight)))
        mass_values.append(float(np.dot(squarefree, weight)))
    c = np.asarray(c_values)
    energy = np.asarray(energy_values)
    mass = np.asarray(mass_values)
    coherence = np.divide(c * c, energy, out=np.zeros_like(c), where=energy > 0)
    effective_dimension = np.divide(mass * mass, energy, out=np.zeros_like(mass), where=energy > 0)
    return {
        "k": np.asarray(k_values, dtype=int),
        "c": c,
        "energy": energy,
        "mass": mass,
        "coherence": coherence,
        "effective_dimension": effective_dimension,
    }


def generalized_transfer_direct(
    a: float,
    beta: float,
    k: int,
    t: float,
    nmax: int,
    mu: np.ndarray,
    character: np.ndarray | None = None,
) -> complex:
    n = np.arange(2, nmax + 1, dtype=np.float64)
    coefficients = mu[2 : nmax + 1].astype(np.complex128)
    if character is not None:
        coefficients = coefficients * character[2 : nmax + 1]
    damping = np.exp(k * np.log1p(-1.0 / np.power(n, beta)))
    phase = np.exp(-1j * t * np.log(n))
```

```
    total = np.sum(coefficients * phase * damping / np.power(n, a))
    if k == 0:
        total += 1.0
    return complex(total)


def long_time_power(
    a: float,
    beta: float,
    k: int,
    nmax: int,
    mu: np.ndarray,
    character: np.ndarray | None = None,
) -> float:
    n = np.arange(2, nmax + 1, dtype=np.float64)
    coefficients = mu[2 : nmax + 1].astype(np.complex128)
    if character is not None:
        coefficients = coefficients * character[2 : nmax + 1]
    damping = np.exp(k * np.log1p(-1.0 / np.power(n, beta)))
    values = coefficients * damping / np.power(n, a)
    total = np.sum(np.abs(values) ** 2)
    if k == 0:
        total += 1.0
    return float(total)


def quadratic_character_mod5(nmax: int) -> np.ndarray:
    values = np.zeros(nmax + 1, dtype=np.int8)
    residue_value = {0: 0, 1: 1, 2: -1, 3: -1, 4: 1}
    for n in range(1, nmax + 1):
        values[n] = residue_value[n % 5]
    return values.astype(np.complex128)


def polyphase_transfer_packets(k: int, modulus: int, nmax: int, mu: np.ndarray) -> np.ndarray:
    n = np.arange(2, nmax + 1, dtype=np.float64)
    weights = mu[2 : nmax + 1].astype(np.float64) / (n * n) * np.exp(k * np.log1p(-1.0 / (n * n)))
    packets = np.zeros(modulus, dtype=np.float64)
    residues = np.arange(2, nmax + 1) % modulus
    for residue in range(modulus):
        packets[residue] = float(np.sum(weights[residues == residue]))
    return packets


# ---------------------------------------------------------------------------
# Exact checks and tables
# ---------------------------------------------------------------------------


@dataclass
class Check:
    name: str
    passed: bool
    detail: str


def run_symbolic_checks() -> list[Check]:
    checks: list[Check] = []
    z, u = sp.symbols("z u")

    coeffs = [sp.Rational((index + 1) * (-1) ** index, index + 2) for index in range(15)]
    for m in (2, 3, 4, 5):
        sections = polyphase_sections_from_coefficients(coeffs, m, u)
        reconstruction = sp.expand(sum(z**r * sections[r].subs(u, z**m) for r in range(m)))
        original = sp.expand(sum(coeffs[index] * z**index for index in range(len(coeffs))))
        checks.append(Check(f"polynomial reconstruction M={m}", sp.simplify(reconstruction - original) == 0, "exact"
    ))

        carry = carry_matrix_symbolic(m, u)
        theta_vector = theta_vector_apply(sections, m, u)
        theta_carry = theta_vector_apply(carry * sections, m, u)
```

```
        carry_theta = carry * theta_vector
        commutator = sp.simplify(theta_carry - carry_theta - carry * sections)
        checks.append(Check(f"Weyl commutator M={m}", all(value == 0 for value in commutator), "[Theta,C]=C"))

    for m in (2, 3, 4):
        for r in range(m):
            point = mp.mpf("0.37")
            series_value = exp_section_series(m, r, point, 80)
            hyper_value = exp_section_hypergeometric(m, r, point)
            residual = abs(series_value - hyper_value)
            checks.append(Check(f"exponential hypergeometric M={m} r={r}", residual < mp.mpf("1e-35"), f"{float(
     residual):.3e}"))

    a = sp.symbols("a")
    for n in range(1, 10):
        child0 = 2 ** (n - 1) * sp.bernoulli(n, a / 2)
        child1 = 2 ** (n - 1) * sp.bernoulli(n, (a + 1) / 2)
        low, high = bernoulli_euler_channels(n, a)
        low_direct = sp.simplify((child0 + child1) / sp.sqrt(2))
        high_direct = sp.simplify((child0 - child1) / sp.sqrt(2))
        checks.append(Check(f"Bernoulli low channel n={n}", sp.simplify(low_direct - low) == 0, "exact"))
        checks.append(Check(f"Euler high channel n={n}", sp.simplify(high_direct - high) == 0, "exact"))

    z0 = 0.37 + 0.41j
    a0 = 0.62
    for m in (2, 3, 4):
        reconstructed = sum(bernoulli_packet(z0, a0, m, r) for r in range(m))
        target = bernoulli_interpolation(z0, a0)
        residual = abs(reconstructed - target)
        checks.append(Check(f"Bernoulli packet reconstruction M={m}", residual < 1e-11, f"{residual:.3e}"))

    for m, ell, r in ((1, 2, 0), (2, 3, 1), (3, 2, 2)):
        parent = bernoulli_packet(z0, a0, m, r)
        children = sum(bernoulli_packet(z0, a0, m * ell, r + child * m) for child in range(ell))
        residual = abs(parent - children)
        checks.append(Check(f"Bernoulli refinement M={m} L={ell} r={r}", residual < 1e-11, f"{residual:.3e}"))

    for m, r, ell, j in ((1, 0, 3, 1), (2, 1, 4, 2), (3, 0, 2, 1)):
        direct = cyclotomic_detail(z0, a0, m, r, ell, j)
        lerch = cyclotomic_detail_lerch(z0, a0, m, r, ell, j)
        residual = abs(direct - lerch)
        checks.append(Check(f"Lerch detail M={m} L={ell} j={j}", residual < 1e-10, f"{residual:.3e}"))

    x = sp.symbols("x")
    for n in range(2, 9):
        polynomial = packet_wronskian_polynomial(n, x)
        low_n, high_n = bernoulli_euler_channels(n, x)
        low_prev, high_prev = bernoulli_euler_channels(n - 1, x)
        direct = sp.expand(4 * (low_n * high_prev - high_n * low_prev))
        checks.append(Check(f"packet Wronskian n={n}", sp.simplify(polynomial - direct) == 0, "exact"))

    return checks


def write_cyclotomic_norm_table() -> list[dict[str, str]]:
    rows: list[dict[str, str]] = []
    for n in range(1, 15):
        children = [sp.simplify(3 ** (n - 1) * sp.bernoulli(n, sp.Rational(r, 3))) for r in range(3)]
        # D_1 = child0 + omega^2 child1 + omega child2 = A + B omega, omega^2+omega+1=0
        a_part = sp.simplify(children[0] - children[1])
        b_part = sp.simplify(-children[1] + children[2])
        norm = sp.factor(a_part**2 - a_part * b_part + b_part**2)
        rows.append(
            {
                "n": str(n),
                "child_0": str(children[0]),
                "child_1": str(children[1]),
                "child_2": str(children[2]),
                "A": str(a_part),
                "B": str(b_part),
                "norm": str(norm),
```

```
            }
        )
    with (TABLES / "cyclotomic_norms_L3.csv").open("w", newline="", encoding="utf-8") as handle:
        writer = csv.DictWriter(handle, fieldnames=list(rows[0]))
        writer.writeheader()
        writer.writerows(rows)
    return rows


# ---------------------------------------------------------------------------
# Figures
# ---------------------------------------------------------------------------


def figure_01_polyphase_partition() -> None:
    m = 4
    degree = 31
    fig, ax = plt.subplots(figsize=(12.8, 4.8))
    for n in range(degree + 1):
        residue = n % m
        ax.add_patch(Rectangle((n - 0.42, residue - 0.32), 0.84, 0.64, facecolor=COLORS[residue] + "88", edgecolor=INK, lw=0.6))
        ax.text(n, residue, str(n), ha="center", va="center", fontsize=7.5)
    ax.set_xlim(-1, degree + 1)
    ax.set_ylim(-0.8, m - 0.2)
    ax.set_yticks(range(m), [f"packet r={r}" for r in range(m)])
    ax.set_xlabel("Taylor exponent n")
    ax.set_title("Every coefficient exponent has one polyphase address")
    ax.grid(False)
    fig.tight_layout()
    save_figure(fig, "01_polyphase_partition")


def figure_02_carry_shift() -> None:
    m = 5
    fig, ax = plt.subplots(figsize=(12.8, 5.7))
    ax.axis("off")
    y = 0.58
    xs = np.linspace(0.10, 0.90, m)
    for r, xpos in enumerate(xs):
        box = FancyBboxPatch((xpos - 0.065, y - 0.08), 0.13, 0.16, boxstyle="round,pad=.01", facecolor=COLORS[r % len(COLORS)] + "28", edgecolor=COLORS[r % len(COLORS)], lw=2)
        ax.add_patch(box)
        ax.text(xpos, y, rf"$F_{{{r}}}(u)$", ha="center", va="center", fontsize=15)
    for r in range(m - 1):
        arrow = FancyArrowPatch((xs[r] + 0.066, y), (xs[r + 1] - 0.066, y), arrowstyle="-|>", mutation_scale=13, lw=1.8, color=INK)
        ax.add_patch(arrow)
        ax.text((xs[r] + xs[r + 1]) / 2, y + 0.055, r"multiply by $z$", ha="center", fontsize=9)
    arrow = FancyArrowPatch((xs[-1], y - 0.085), (xs[0], y - 0.085), connectionstyle="arc3,rad=-0.47", arrowstyle="-|>", mutation_scale=13, lw=2.2, color=MAGENTA)
    ax.add_patch(arrow)
    ax.text(0.5, 0.15, r"wraparound contributes $uF_{M-1}(u)$ to packet $0$", ha="center", fontsize=13, color=MAGENTA)
    ax.text(0.5, 0.89, r"The carry-shift matrix $C_M(u)$ is multiplication by $z$ in packet coordinates", ha="center", fontsize=18, weight="bold")
    save_figure(fig, "02_carry_shift")


def figure_03_exp_sections() -> None:
    u = np.linspace(0, 8, 350)
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    for m, ax in zip((2, 3), axes, strict=True):
        for r in range(m):
            values = [float(mp.re(exp_section_hypergeometric(m, r, value))) for value in u]
            ax.plot(u, values, lw=2, label=rf"$E_{{{m},{r}}}(u)$")
        ax.set_title(f"Exponential packets for M={m}")
        ax.set_xlabel("u")
        ax.set_ylabel("packet value")
        ax.grid(alpha=0.18)
        ax.legend()
```

```
    fig.suptitle("The scalar exponential becomes a vector hypergeometric function", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.92])
    save_figure(fig, "03_exponential_sections")


def figure_04_exp_reconstruction() -> None:
    z = np.linspace(-2.5, 2.5, 500)
    fig, ax = plt.subplots(figsize=(12.8, 4.8))
    target = np.exp(z)
    for m, color in zip((2, 3, 4), (BLUE, TEAL, MAGENTA), strict=True):
        reconstruction = np.zeros_like(z)
        for r in range(m):
            reconstruction += z**r * np.asarray([float(mp.re(exp_section_hypergeometric(m, r, value**m))) for value in z])
        ax.plot(z, reconstruction - target, color=color, label=f"M={m}")
    ax.set_yscale("symlog", linthresh=1e-15)
    ax.set_xlabel("z")
    ax.set_ylabel("reconstruction residual")
    ax.set_title(r"Exact synthesis $e^z=\sum_r z^r E_{M,r}(z^M)$")
    ax.grid(alpha=0.2)
    ax.legend()
    fig.tight_layout()
    save_figure(fig, "04_exponential_reconstruction")


def figure_05_operator_lift_map() -> None:
    fig, ax = plt.subplots(figsize=(12.8, 6.0))
    ax.axis("off")
    stages = [
        (0.04, "scalar function", r"$F(z)$", BLUE),
        (0.28, "residue analysis", r"$\mathbf{F}_M(u)$", TEAL),
        (0.53, "operator lift", r"$z\mapsto C_M$\n$\theta\mapsto\Theta_M$", ORANGE),
        (0.78, "matrix system", r"$\mathcal{L}(C_M,\Theta_M)\mathbf{F}_M=0$", MAGENTA),
    ]
    for x0, title, formula, color in stages:
        box = FancyBboxPatch((x0, 0.38), 0.18, 0.25, boxstyle="round,pad=.018", facecolor=color + "24", edgecolor=color, lw=2.2)
        ax.add_patch(box)
        ax.text(x0 + 0.09, 0.57, title, ha="center", va="center", fontsize=12, weight="bold")
        ax.text(x0 + 0.09, 0.46, formula, ha="center", va="center", fontsize=13)
    for left, right in zip(stages[:-1], stages[1:], strict=True):
        ax.add_patch(FancyArrowPatch((left[0] + 0.18, 0.505), (right[0], 0.505), arrowstyle="-|>", mutation_scale=15, lw=2, color=INK))
    ax.text(0.5, 0.83, "Polyphase lifting is a representation of the same Weyl algebra", ha="center", fontsize=19, weight="bold")
    ax.text(0.5, 0.17, r"The commutator is preserved: $[\Theta_M,C_M]=C_M$", ha="center", fontsize=15, color=MUTED)
    save_figure(fig, "05_operator_lift_map")


def figure_06_bessel_sections() -> None:
    z_values = np.linspace(-8, 8, 800)
    m = 3
    # J0 Taylor coefficients up to degree 72
    coeffs = [sp.Integer(0)] * 73
    for q in range(37):
        coeffs[2 * q] = sp.Rational((-1) ** q, 4**q * math.factorial(q) ** 2)
    u_symbol = sp.Symbol("u")
    sections = polyphase_sections_from_coefficients(coeffs, m, u_symbol)
    functions = [sp.lambdify(u_symbol, section, "numpy") for section in sections]
    fig, axes = plt.subplots(m, 1, figsize=(12.8, 7.2), sharex=True)
    for r, ax in enumerate(axes):
        u_values = z_values**m
        values = np.real(functions[r](u_values))
        ax.plot(z_values, values, color=COLORS[r], lw=1.6)
        ax.set_ylabel(rf"$J_{{0;3,{r}}}$")
        ax.grid(alpha=0.16)
    axes[-1].set_xlabel("z, with packet evaluated at u=z^3")
    fig.suptitle("Three packet sections of the Bessel function J0", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.94])
    save_figure(fig, "06_bessel_sections")

```

```

def figure_07_bernoulli_packet_reconstruction() -> None:
    a = 0.37
    z_values = np.linspace(-1.5, 2.5, 240)
    m = 4
    packets = np.zeros((m, len(z_values)), dtype=np.float64)
    total = np.zeros(len(z_values), dtype=np.float64)
    for index, z in enumerate(z_values):
        for r in range(m):
            packets[r, index] = bernoulli_packet(z, a, m, r).real
        total[index] = bernoulli_interpolation(z, a).real
    fig, axes = plt.subplots(2, 1, figsize=(12.8, 7.0), sharex=True)
    for r in range(m):
        axes[0].plot(z_values, packets[r], lw=1.5, label=f"r={r}")
    axes[0].set_ylabel("packet value")
    axes[0].set_title(r"Bernoulli-Hurwitz packets $\mathcal{B}_{4,r}(z;a)$")
    axes[0].legend(ncol=4)
    axes[0].grid(alpha=0.16)
    axes[1].plot(z_values, total, color=INK, lw=2.4, label="direct interpolation")
    axes[1].plot(z_values, packets.sum(axis=0), color=RED, lw=1.2, ls="--", label="packet sum")
    axes[1].set_xlabel("complex order z restricted to the real axis")
    axes[1].set_ylabel("reconstructed value")
    axes[1].grid(alpha=0.16)
    axes[1].legend()
    fig.suptitle("Exact reconstruction of the Bernoulli interpolation from four packets", fontsize=17, weight="bold"
    )
    fig.tight_layout(rect=[0, 0, 1, 0.94])
    save_figure(fig, "07_bernoulli_packet_reconstruction")


def figure_08_bernoulli_refinement_tree() -> None:
    fig, ax = plt.subplots(figsize=(12.8, 7.0))
    ax.axis("off")
    levels = [(1, [0]), (2, [0, 1]), (4, list(range(4))), (8, list(range(8)))]
    positions: dict[tuple[int, int], tuple[float, float]] = {}
    for depth, (modulus, residues) in enumerate(levels):
        y = 0.86 - depth * 0.23
        xs = np.linspace(0.08, 0.92, len(residues))
        for residue, xpos in zip(residues, xs, strict=True):
            positions[(modulus, residue)] = (xpos, y)
            color = COLORS[depth % len(COLORS)]
            ax.add_patch(FancyBboxPatch((xpos - 0.045, y - 0.032), 0.09, 0.064, boxstyle="round,pad=.005", facecolor
    =color + "25", edgecolor=color, lw=1.6))
            ax.text(xpos, y, rf"${{modulus}},{residue}}$", ha="center", va="center", fontsize=8.5)
    for (parent_m, residues), (child_m, child_residues) in zip(levels[:-1], levels[1:], strict=True):
        factor = child_m // parent_m
        for parent_r in residues:
            px, py = positions[(parent_m, parent_r)]
            for ell in range(factor):
                child_r = parent_r + ell * parent_m
                cx, cy = positions[(child_m, child_r)]
                ax.plot([px, cx], [py - 0.035, cy + 0.035], color=GRID, lw=1)
    ax.text(0.5, 0.96, r"Exact refinement: $\mathcal{B}_{M,r}=\sum_\ell \mathcal{B}_{LM,r+\ell M}$", ha="center",
    fontsize=19, weight="bold")
    ax.text(0.5, 0.04, "Each parent is reconstructed by all of its residue children", ha="center", fontsize=12.5,
    color=MUTED)
    save_figure(fig, "08_bernoulli_refinement_tree")


def figure_09_cyclotomic_details() -> None:
    z = 3.4 + 0.6j
    a_values = np.linspace(0.1, 1.8, 120)
    ell = 4
    fig, axes = plt.subplots(2, 1, figsize=(12.8, 7.0), sharex=True)
    for j in range(ell):
        values = np.asarray([cyclotomic_detail(z, a, 1, 0, ell, j) for a in a_values])
        axes[0].plot(a_values, np.abs(values), lw=1.8, label=f"j={j}")
        axes[1].plot(a_values, np.unwrap(np.angle(values)), lw=1.5, label=f"j={j}")
    axes[0].set_ylabel("magnitude")
    axes[0].set_title("Cyclotomic Bernoulli-Lerch channels")
    axes[0].grid(alpha=0.16)
```

```
    axes[0].legend(ncol=4)
    axes[1].set_xlabel("translation parameter a")
    axes[1].set_ylabel("unwrapped phase")
    axes[1].grid(alpha=0.16)
    fig.tight_layout()
    save_figure(fig, "09_cyclotomic_details")


def figure_10_cyclotomic_parseval() -> None:
    z = 2.75 + 0.45j
    a = 0.4
    ell = 6
    children = np.asarray([bernoulli_packet(z, a, ell, child) for child in range(ell)])
    transform = np.fft.fft(children) / math.sqrt(ell)
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.7))
    axes[0].bar(range(ell), np.abs(children) ** 2, color=COLORS * 2, edgecolor=INK)
    axes[0].set_title("child-packet energies")
    axes[0].set_xlabel("child index")
    axes[0].set_ylabel("squared magnitude")
    axes[1].bar(range(ell), np.abs(transform) ** 2, color=COLORS * 2, edgecolor=INK)
    axes[1].set_title("cyclotomic-channel energies")
    axes[1].set_xlabel("channel index")
    total_child = float(np.sum(np.abs(children) ** 2))
    total_transform = float(np.sum(np.abs(transform) ** 2))
    fig.suptitle(f"Parseval across the special-function filter bank: {total_child:.8f} = {total_transform:.8f}",
     fontsize=16.5, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.90])
    save_figure(fig, "10_cyclotomic_parseval")


def figure_11_bernoulli_euler_channels() -> None:
    x = sp.symbols("x")
    grid = np.linspace(-0.2, 1.2, 600)
    n = 7
    low, high = bernoulli_euler_channels(n, x)
    child0 = 2 ** (n - 1) * sp.bernoulli(n, x / 2)
    child1 = 2 ** (n - 1) * sp.bernoulli(n, (x + 1) / 2)
    functions = [sp.lambdify(x, expression, "numpy") for expression in (child0, child1, low, high)]
    values = [function(grid) for function in functions]
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    axes[0].plot(grid, values[0], color=BLUE, lw=1.8, label="child 0")
    axes[0].plot(grid, values[1], color=ORANGE, lw=1.8, label="child 1")
    axes[0].set_title("two Bernoulli children")
    axes[0].set_xlabel("a")
    axes[0].grid(alpha=0.16)
    axes[0].legend()
    axes[1].plot(grid, values[2], color=TEAL, lw=2, label=r"low: $B_n(a)/\sqrt{2}$")
    axes[1].plot(grid, values[3], color=MAGENTA, lw=2, label=r"high: $-nE_{n-1}(a)/(2\sqrt{2})$")
    axes[1].set_title("unitary low-pass and high-pass channels")
    axes[1].set_xlabel("a")
    axes[1].grid(alpha=0.16)
    axes[1].legend()
    fig.suptitle("Bernoulli and Euler polynomials arise from one exact two-channel refinement", fontsize=16.5,
     weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "11_bernoulli_euler_channels")


def figure_12_packet_wronskians() -> None:
    x = sp.symbols("x")
    grid = np.linspace(-1.2, 2.2, 900)
    fig, axes = plt.subplots(2, 2, figsize=(13.0, 8.0), sharex=True)
    for ax, n, color in zip(axes.ravel(), (3, 4, 5, 6), COLORS[:4], strict=True):
        polynomial = packet_wronskian_polynomial(n, x)
        function = sp.lambdify(x, polynomial, "numpy")
        values = np.asarray(function(grid), dtype=np.float64)
        ax.axhline(0, color=GRID, lw=1)
        ax.plot(grid, values, color=color, lw=1.8)
        roots = [complex(root) for root in sp.nroots(polynomial, maxsteps=200)]
        real_roots = sorted(root.real for root in roots if abs(root.imag) < 1e-9 and -1.2 <= root.real <= 2.2)
        if real_roots:
```

```
            ax.scatter(real_roots, np.zeros(len(real_roots)), color=INK, s=24, zorder=4)
        ax.set_title(rf"$\mathcal{{W}}_{{{n}}}(a)$, {len(real_roots)} displayed real roots")
        ax.grid(alpha=0.14)
    for ax in axes[-1]:
        ax.set_xlabel("a")
    fig.suptitle("Packet Wronskians provide a direct zero and interlacing diagnostic", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.94])
    save_figure(fig, "12_packet_wronskians")


def figure_13_wronskian_root_counts() -> None:
    x = sp.symbols("x")
    degrees = []
    real_counts = []
    nonreal_counts = []
    ns = list(range(2, 17))
    for n in ns:
        polynomial = packet_wronskian_polynomial(n, x)
        roots = [complex(root) for root in sp.nroots(polynomial, maxsteps=400)]
        real_count = sum(abs(root.imag) < 1e-8 for root in roots)
        degree = sp.degree(polynomial, x)
        degrees.append(degree)
        real_counts.append(real_count)
        nonreal_counts.append(degree - real_count)
    fig, ax = plt.subplots(figsize=(12.8, 4.8))
    ax.bar(ns, real_counts, color=TEAL, label="real roots")
    ax.bar(ns, nonreal_counts, bottom=real_counts, color=ORANGE, label="nonreal roots")
    ax.set_xlabel("order n")
    ax.set_ylabel("root count")
    ax.set_title("A useful negative control: packet Wronskians are not universally real-rooted")
    ax.legend()
    ax.grid(axis="y", alpha=0.18)
    fig.tight_layout()
    save_figure(fig, "13_wronskian_root_counts")


def figure_14_cyclotomic_norms() -> None:
    rows = write_cyclotomic_norm_table()
    ns = np.asarray([int(row["n"]) for row in rows])
    norms = np.asarray([float(sp.Rational(row["norm"])) for row in rows])
    fig, ax = plt.subplots(figsize=(12.8, 4.8))
    ax.semilogy(ns, norms, marker="o", color=BLUE, lw=1.8)
    ax.set_xlabel("integer order n")
    ax.set_ylabel("cyclotomic norm")
    ax.set_title("Exact norm of the first non-DC L=3 Bernoulli detail")
    ax.grid(alpha=0.18)
    fig.tight_layout()
    save_figure(fig, "14_cyclotomic_norms")


def figure_15_mobius_kernel_localization(mu: np.ndarray) -> None:
    n = np.arange(2, 6000, dtype=np.float64)
    fig, ax = plt.subplots(figsize=(12.8, 5.0))
    for k, color in zip((100, 500, 2000, 8000), COLORS[:4], strict=True):
        weight = n ** -2 * np.exp(k * np.log1p(-1.0 / (n * n)))
        normalized = weight / weight.max()
        ax.plot(n / math.sqrt(k), normalized, lw=1.8, color=color, label=f"k={k}")
    ax.set_xlim(0, 5)
    ax.set_xlabel(r"scaled integer address $n/\sqrt{k}$")
    ax.set_ylabel("normalized packet weight")
    ax.set_title(r"The Bernoulli-Mobius transfer localizes at the scale $n\asymp\sqrt{k}$")
    ax.grid(alpha=0.18)
    ax.legend()
    fig.tight_layout()
    save_figure(fig, "15_mobius_kernel_localization")


def figure_16_mertens_wavelet() -> None:
    x = np.linspace(1.001, 180, 2000)
    fig, ax = plt.subplots(figsize=(12.8, 5.0))
    for k, color in zip((100, 500, 2000), (BLUE, TEAL, MAGENTA), strict=True):
```

```
        psi = 2 * x ** -5 * np.power(1 - x ** -2, k - 1) * (x * x - k - 1)
        scale = np.max(np.abs(psi))
        ax.plot(x / math.sqrt(k + 1), psi / scale, lw=1.8, color=color, label=f"k={k}")
    ax.axvline(1, color=INK, ls="--", lw=1.2, label=r"sign change $x=\sqrt{k+1}$")
    ax.axhline(0, color=GRID, lw=1)
    ax.set_xlim(0, 3.5)
    ax.set_xlabel(r"scaled coordinate $x/\sqrt{k+1}$")
    ax.set_ylabel("normalized wavelet")
    ax.set_title("The exact finite-difference kernel is a one-sign-change Mertens wavelet")
    ax.legend()
    ax.grid(alpha=0.16)
    fig.tight_layout()
    save_figure(fig, "16_mertens_wavelet")


def figure_17_energy_asymptotic(data: dict[str, np.ndarray]) -> None:
    k = data["k"].astype(float)
    constant = 3 / (2 ** 2.5 * math.pi ** 1.5)
    fig, ax = plt.subplots(figsize=(12.8, 4.9))
    ax.semilogx(k, data["energy"] * k ** 1.5, color=BLUE, lw=2, marker="o", markersize=3, label=r"$k^{3/2}E_k$")
    ax.axhline(constant, color=RED, ls="--", lw=1.8, label=f"limit {constant:.10f}")
    ax.set_xlabel("k")
    ax.set_ylabel("scaled energy")
    ax.set_title("The diagonal Mobius packet energy already has the RH exponent")
    ax.grid(alpha=0.18)
    ax.legend()
    fig.tight_layout()
    save_figure(fig, "17_energy_asymptotic")


def figure_18_mass_and_dimension(data: dict[str, np.ndarray]) -> None:
    k = data["k"].astype(float)
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    axes[0].loglog(k, data["mass"], color=TEAL, lw=2, marker="o", markersize=3, label="squarefree packet mass")
    axes[0].loglog(k, 3 / math.pi ** 1.5 * k ** -0.5, color=INK, ls="--", label=r"$3\pi^{-3/2}k^{-1/2}$")
    axes[0].set_xlabel("k")
    axes[0].set_ylabel("mass")
    axes[0].set_title("active squarefree mass")
    axes[0].grid(alpha=0.18, which="both")
    axes[0].legend()
    axes[1].loglog(k, data["effective_dimension"], color=MAGENTA, lw=2, marker="o", markersize=3)
    axes[1].set_xlabel("k")
    axes[1].set_ylabel(r"$S_k^2/E_k$")
    axes[1].set_title("effective number of active packets")
    axes[1].grid(alpha=0.18, which="both")
    fig.suptitle(r"The packet cloud grows like a constant multiple of $\sqrt{k}$", fontsize=16.5, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "18_mass_and_dimension")


def figure_19_coherence_ratio(data: dict[str, np.ndarray]) -> None:
    k = data["k"].astype(float)
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    axes[0].loglog(k, np.abs(data["c"]), color=BLUE, lw=1.8, marker="o", markersize=3, label=r"$|c_k|$")
    axes[0].loglog(k, np.sqrt(data["energy"]), color=TEAL, lw=1.8, label=r"$\sqrt{E_k}$")
    axes[0].set_xlabel("k")
    axes[0].set_ylabel("amplitude")
    axes[0].set_title("coherent amplitude and diagonal scale")
    axes[0].grid(alpha=0.18, which="both")
    axes[0].legend()
    axes[1].semilogx(k, data["coherence"], color=MAGENTA, lw=1.8, marker="o", markersize=3)
    axes[1].set_xlabel("k")
    axes[1].set_ylabel(r"$\Gamma_k=|c_k|^2/E_k$")
    axes[1].set_title("Mobius packet coherence")
    axes[1].grid(alpha=0.18)
    fig.suptitle("RH is equivalent to subpolynomial growth of the coherence ratio", fontsize=16.5, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "19_coherence_ratio")


def figure_20_log_time_signal(data: dict[str, np.ndarray]) -> None:
```

```
    k = data["k"].astype(float)
    y = k ** 0.75 * data["c"]
    fig, ax = plt.subplots(figsize=(12.8, 4.8))
    ax.plot(np.log(k), y, color=BLUE, lw=1.4, marker="o", markersize=3)
    ax.axhline(0, color=GRID, lw=1)
    ax.set_xlabel(r"logarithmic time $u=\log k$")
    ax.set_ylabel(r"$k^{3/4}c_k$")
    ax.set_title("Normalized Bernoulli-difference signal in logarithmic time")
    ax.grid(alpha=0.18)
    fig.tight_layout()
    save_figure(fig, "20_log_time_signal")


def figure_21_long_time_power(mu: np.ndarray) -> None:
    nmax = 60000
    n = np.arange(2, nmax + 1, dtype=np.float64)
    mu_n = mu[2 : nmax + 1].astype(np.float64)
    k = 1200
    coeff = mu_n / (n * n) * np.exp(k * np.log1p(-1.0 / (n * n)))
    t_values = np.linspace(0, 160, 500)
    signal = np.zeros_like(t_values, dtype=np.complex128)
    chunk = 5000
    logs = np.log(n)
    for start in range(0, len(n), chunk):
        stop = min(start + chunk, len(n))
        phase = np.exp(-1j * np.outer(t_values, logs[start:stop]))
        signal += phase @ coeff[start:stop]
    energy = float(np.sum(coeff * coeff))
    running_power = np.cumsum(np.abs(signal) ** 2) / np.arange(1, len(signal) + 1)
    fig, axes = plt.subplots(2, 1, figsize=(12.8, 7.0), sharex=True)
    axes[0].plot(t_values, signal.real, color=BLUE, lw=1.2, label="real")
    axes[0].plot(t_values, signal.imag, color=ORANGE, lw=1.2, label="imaginary")
    axes[0].set_ylabel("F_k(t)")
    axes[0].set_title("Mobius-Bernoulli quantum correlation")
    axes[0].legend()
    axes[0].grid(alpha=0.15)
    axes[1].plot(t_values, running_power, color=TEAL, lw=1.7, label="running mean power")
    axes[1].axhline(energy, color=RED, ls="--", label=f"diagonal energy {energy:.3e}")
    axes[1].set_xlabel("time t")
    axes[1].set_ylabel("mean squared magnitude")
    axes[1].grid(alpha=0.15)
    axes[1].legend()
    fig.tight_layout()
    save_figure(fig, "21_long_time_power")


def figure_22_polyphase_mobius_packets(mu: np.ndarray) -> None:
    modulus = 12
    ks = [200, 1000, 5000]
    fig, axes = plt.subplots(len(ks), 1, figsize=(12.8, 8.3), sharex=True)
    for ax, k in zip(axes, ks, strict=True):
        packets = polyphase_transfer_packets(k, modulus, 180000, mu)
        colors = [TEAL if value >= 0 else ORANGE for value in packets]
        ax.bar(range(modulus), packets, color=colors, edgecolor=INK)
        ax.axhline(0, color=GRID, lw=1)
        ax.set_ylabel(f"k={k}")
        ax.grid(axis="y", alpha=0.15)
    axes[-1].set_xlabel("integer residue modulo 12")
    fig.suptitle("Exact arithmetic polyphase packets of the Mobius transfer signal", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.94])
    save_figure(fig, "22_polyphase_mobius_packets")


def figure_23_character_packets(mu: np.ndarray) -> None:
    nmax = 160000
    character = quadratic_character_mod5(nmax)
    k_values = np.unique(np.logspace(1, 4.6, 80).astype(int))
    principal = []
    quadratic = []
    for k in k_values:
        principal.append(generalized_transfer_direct(2, 2, int(k), 0, nmax, mu))
```

```
        quadratic.append(generalized_transfer_direct(2, 2, int(k), 0, nmax, mu, character))
    fig, ax = plt.subplots(figsize=(12.8, 4.9))
    ax.loglog(k_values, np.abs(principal), color=BLUE, lw=1.8, label="principal character")
    ax.loglog(k_values, np.abs(quadratic), color=MAGENTA, lw=1.8, label="quadratic character mod 5")
    ax.set_xlabel("k")
    ax.set_ylabel("transfer amplitude")
    ax.set_title("Character packets create a natural GRH-facing atlas")
    ax.grid(alpha=0.18, which="both")
    ax.legend()
    fig.tight_layout()
    save_figure(fig, "23_character_packets")


def figure_24_quantum_operator_pipeline() -> None:
    fig, ax = plt.subplots(figsize=(12.8, 6.1))
    ax.axis("off")
    stages = [
        (0.03, r"$|\phi_a\rangle$", "power-law state", BLUE),
        (0.23, r"$e^{-kK_\beta}$", "scale filter", TEAL),
        (0.43, r"$\mathcal{M}_\chi$", "Mobius-character sign", ORANGE),
        (0.63, r"$e^{-itH}$", "logarithmic evolution", MAGENTA),
        (0.83, r"$\langle\phi_a|$", "coherence readout", RED),
    ]
    for x0, formula, title, color in stages:
        box = FancyBboxPatch((x0, 0.40), 0.14, 0.22, boxstyle="round,pad=.015", facecolor=color + "24", edgecolor=
     color, lw=2)
        ax.add_patch(box)
        ax.text(x0 + 0.07, 0.55, formula, ha="center", va="center", fontsize=15)
        ax.text(x0 + 0.07, 0.45, title, ha="center", va="center", fontsize=9.5)
    for left, right in zip(stages[:-1], stages[1:], strict=True):
        ax.add_patch(FancyArrowPatch((left[0] + 0.14, 0.51), (right[0], 0.51), arrowstyle="-|>", mutation_scale=14,
     lw=1.8, color=INK))
    ax.text(0.5, 0.82, r"$\mathcal{C}_{a,\beta,\chi}(k,t)=\langle\phi_a|\mathcal{M}_\chi e^{-kK_\beta}e^{-itH}|\
     phi_a\rangle$", ha="center", fontsize=18, weight="bold")
    ax.text(0.5, 0.18, r"$H|n\rangle=\log n|n\rangle$ and $K_\beta=-\log(I-e^{-\beta H})$", ha="center", fontsize=14,
      color=MUTED)
    save_figure(fig, "24_quantum_operator_pipeline")


def figure_25_function_family_map() -> None:
    fig, ax = plt.subplots(figsize=(13.0, 7.0))
    ax.axis("off")
    nodes = {
        "scalar": (0.10, 0.74, "scalar analytic\nfunction", BLUE),
        "lift": (0.38, 0.74, "polyphase\nvector lift", TEAL),
        "matrix": (0.68, 0.74, "matrix differential\nsystem", ORANGE),
        "bern": (0.12, 0.36, "Bernoulli-Hurwitz\npackets", MAGENTA),
        "lerch": (0.40, 0.36, "cyclotomic Lerch\ndetails", RED),
        "wronskian": (0.68, 0.36, "packet Wronskians\nand zero tests", YELLOW),
        "mobius": (0.88, 0.36, "Mobius-Bernoulli\ncoherence", BLUE),
    }
    for key, (x0, y0, label, color) in nodes.items():
        ax.add_patch(FancyBboxPatch((x0 - 0.09, y0 - 0.06), 0.18, 0.12, boxstyle="round,pad=.012", facecolor=color +
      "22", edgecolor=color, lw=2))
        ax.text(x0, y0, label, ha="center", va="center", fontsize=11, weight="bold")
    edges = [
        ("scalar", "lift"), ("lift", "matrix"), ("scalar", "bern"), ("bern", "lerch"), ("lerch", "wronskian"), ("
     wronskian", "mobius"), ("matrix", "mobius"),
    ]
    for source, target in edges:
        sx, sy = nodes[source][0], nodes[source][1]
        tx, ty = nodes[target][0], nodes[target][1]
        ax.add_patch(FancyArrowPatch((sx, sy), (tx, ty), arrowstyle="-|>", mutation_scale=12, lw=1.4, color=GRID,
     connectionstyle="arc3,rad=0.05"))
    ax.text(0.5, 0.93, "A special-function factory generated by exact packet analysis", ha="center", fontsize=20,
     weight="bold")
    ax.text(0.5, 0.10, "The constructions share reconstruction, refinement, transfer, and certificate laws", ha="
     center", fontsize=13, color=MUTED)
    save_figure(fig, "25_function_family_map")


```

```python
def figure_26_status_map() -> None:
    fig, ax = plt.subplots(figsize=(12.8, 5.9))
    ax.axis("off")
    categories = [
        (0.08, 0.65, 0.24, 0.20, "PROVED HERE", ["polyphase lift", "Bernoulli packet refinement", "cyclotomic inverse transform", "energy asymptotic"], TEAL),
        (0.38, 0.65, 0.24, 0.20, "CLASSICAL INPUT", ["Hurwitz zeta", "Raabe multiplication", "Lerch transcendent", "Baez-Duarte criterion"], BLUE),
        (0.68, 0.65, 0.24, 0.20, "COMPUTATIONAL", ["Wronskian root maps", "cyclotomic norms", "coherence plots", "character packets"], ORANGE),
        (0.23, 0.24, 0.24, 0.20, "OPEN", ["Wronskian sign laws", "zero interlacing", "p-adic detail bounds", "character-uniform coherence"], MAGENTA),
        (0.53, 0.24, 0.24, 0.20, "MAJOR CONJECTURE", ["subpolynomial Mobius coherence", "RH and GRH applications", "positive canonical systems"], RED),
    ]
    for x0, y0, w, h, title, entries, color in categories:
        ax.add_patch(FancyBboxPatch((x0, y0), w, h, boxstyle="round,pad=.015", facecolor=color + "20", edgecolor=color, lw=2))
        ax.text(x0 + w / 2, y0 + h - 0.035, title, ha="center", va="center", fontsize=11.5, weight="bold", color=color)
        for index, entry in enumerate(entries):
            ax.text(x0 + 0.025, y0 + h - 0.075 - 0.030 * index, "* " + entry, ha="left", va="center", fontsize=9.4)
    ax.text(0.5, 0.94, "Claim boundary for the generated function theory", ha="center", fontsize=19, weight="bold")
    save_figure(fig, "26_status_map")


def generate_all_figures() -> dict[str, object]:
    checks = run_symbolic_checks()
    if not all(check.passed for check in checks):
        failed = [check for check in checks if not check.passed]
        raise RuntimeError(f"symbolic checks failed: {failed}")

    nmax = 200000
    mu = mobius_sieve(nmax)
    k_values = np.unique(np.logspace(1, 5, 92).astype(int))
    transfer_data = principal_transfer_values(k_values, nmax, mu)

    figure_01_polyphase_partition()
    figure_02_carry_shift()
    figure_03_exp_sections()
    figure_04_exp_reconstruction()
    figure_05_operator_lift_map()
    figure_06_bessel_sections()
    figure_07_bernoulli_packet_reconstruction()
    figure_08_bernoulli_refinement_tree()
    figure_09_cyclotomic_details()
    figure_10_cyclotomic_parseval()
    figure_11_bernoulli_euler_channels()
    figure_12_packet_wronskians()
    figure_13_wronskian_root_counts()
    figure_14_cyclotomic_norms()
    figure_15_mobius_kernel_localization(mu)
    figure_16_mertens_wavelet()
    figure_17_energy_asymptotic(transfer_data)
    figure_18_mass_and_dimension(transfer_data)
    figure_19_coherence_ratio(transfer_data)
    figure_20_log_time_signal(transfer_data)
    figure_21_long_time_power(mu)
    figure_22_polyphase_mobius_packets(mu)
    figure_23_character_packets(mu)
    figure_24_quantum_operator_pipeline()
    figure_25_function_family_map()
    figure_26_status_map()

    # Numerical result ledger
    constant = 3 / (2 ** 2.5 * math.pi ** 1.5)
    large_index = -1
    energy_scaled = float(transfer_data["energy"][large_index] * transfer_data["k"][large_index] ** 1.5)
    mass_scaled = float(transfer_data["mass"][large_index] * math.sqrt(transfer_data["k"][large_index]))

    quantum_k = 1200
```

```
    mean_power = long_time_power(2, 2, quantum_k, nmax, mu)
    zero_time = generalized_transfer_direct(2, 2, quantum_k, 0.0, nmax, mu)

    results = {
        "figure_pairs": 26,
        "symbolic_checks": len(checks),
        "symbolic_checks_passed": sum(check.passed for check in checks),
        "mobius_nmax": nmax,
        "largest_k": int(transfer_data["k"][large_index]),
        "energy_scaled_at_largest_k": energy_scaled,
        "energy_asymptotic_constant": constant,
        "energy_relative_error": abs(energy_scaled - constant) / constant,
        "mass_scaled_at_largest_k": mass_scaled,
        "mass_asymptotic_constant": 3 / math.pi ** 1.5,
        "quantum_k": quantum_k,
        "quantum_zero_time_abs": abs(zero_time),
        "quantum_long_time_diagonal_power": mean_power,
        "quantum_coherence_ratio": abs(zero_time) ** 2 / mean_power,
    }

    with (RESULTS / "results.json").open("w", encoding="utf-8") as handle:
        json.dump(results, handle, indent=2, sort_keys=True)

    with (RESULTS / "checks.csv").open("w", newline="", encoding="utf-8") as handle:
        writer = csv.DictWriter(handle, fieldnames=["name", "passed", "detail"])
        writer.writeheader()
        writer.writerows([{"name": check.name, "passed": check.passed, "detail": check.detail} for check in checks])

    with (TABLES / "mobius_transfer_samples.csv").open("w", newline="", encoding="utf-8") as handle:
        writer = csv.writer(handle)
        writer.writerow(["k", "c_k", "energy", "mass", "coherence", "effective_dimension"])
        for row in zip(
            transfer_data["k"],
            transfer_data["c"],
            transfer_data["energy"],
            transfer_data["mass"],
            transfer_data["coherence"],
            transfer_data["effective_dimension"],
            strict=True,
        ):
            writer.writerow(row)

    return results


def sha256(path: Path) -> str:
    digest = hashlib.sha256()
    with path.open("rb") as handle:
        for block in iter(lambda: handle.read(1024 * 1024), b""):
            digest.update(block)
    return digest.hexdigest()


def main() -> None:
    parser = argparse.ArgumentParser(description="Generate the Projector-Generated Special Functions paper assets")
    parser.add_argument("--base", type=Path, default=ROOT)
    args = parser.parse_args()
    if args.base.resolve() != ROOT.resolve():
        raise SystemExit("This release expects to run from its source directory")
    results = generate_all_figures()
    print(json.dumps(results, indent=2, sort_keys=True))


if __name__ == "__main__":
    main()
```

Chapter C

Bibliographic note

Bibliography

[1] NIST Digital Library of Mathematical Functions, Chapter 24, "Bernoulli and Euler Polynomials," particularly Section 24.4 on difference, derivative, and multiplication formulas.

[2] NIST Digital Library of Mathematical Functions, Sections 25.11 and 25.14 on the Hurwitz zeta function and Lerch transcendent.

[3] L. Báez-Duarte, "A Sequential Riesz-like Criterion for the Riemann Hypothesis," *International Journal of Mathematics and Mathematical Sciences*, 2005, Article 141565, 3527–3537.

[4] J. Cisło and M. Wolf, "On the Riesz and Báez-Duarte Criteria for the Riemann Hypothesis," 2008, arXiv:0807.2971.

[5] M. Garg and B. Maji, "Hardy-Littlewood-Riesz Type Equivalent Criteria for the Generalized Riemann Hypothesis," 2022, arXiv:2208.07596.

[6] S. Gupta and A. Vatwani, "Riesz Type Criteria for L-Functions in the Selberg Class," 2022, arXiv:2211.02954.

[7] S. Wei, Y. Zhai, Q. Lu, W. Yang, P. Gao, C. Wei, J. Song, F. Nori, T. Xin, and G. Long, "The Riemann Hypothesis Manifested in Dynamical Quantum Phase Transitions," *Nature Communications*, accepted 18 June 2026, DOI: 10.1038/s41467-026-74935-8.

[8] Komorebi P., "Bernoulli Transfer-Block Reformulation of Zeta," unpublished proof-development note.

Projector-Generated Special Functions II

Cyclotomic Stieltjes Packets, Packet Resultants, Positive Energy Resolvents, Gamma Scaling, and Character Decoherence

Komorebi P.

Abstract

This paper develops a second family of special functions generated by exact packet refinement. The starting point is the Bernoulli–Hurwitz child system introduced in the first paper of the series. A finite cyclotomic transform is applied to the children of one parent packet. The Taylor germ of every non-DC detail channel is then identified with a triangular transform of cyclotomic Fourier packets of generalized Stieltjes constants. For a nonprincipal Dirichlet character, the same transform recovers the derivatives $L^{(m)}(1,\chi)$ from generalized Stieltjes constants at rational arguments.

Two scalar packet invariants are introduced. The holomorphic detail variance equals both the sum of the non-DC channel products and the mean pairwise square difference of the children. The primitive packet resultant compresses the complete Galois orbit of primitive cyclotomic details into one entire scalar obstruction. At positive integer order it takes rational values for rational packet parameters.

The arithmetic half of the paper studies the positive squarefree packet energy

$$E_{a,\beta,\chi}(k) = \sum_{n\geq 1} \frac{\mu(n)^2\,|\chi(n)|^2}{n^{2a}} \left(1 - \frac{1}{n^{\beta}}\right)^{2k}.$$

This sequence is proved to be a Hausdorff moment sequence. All finite differences have alternating sign, every finite Hankel matrix is positive definite, and its generating resolvent is analytic and zero-free on $\mathbb{C} \setminus [1,\infty)$. The same energy is also an exact finite difference of the squarefree zeta ratio $\zeta(s)/\zeta(2s)$ with the Euler factors at the conductor removed.

A complete asymptotic law is proved. After normalization, the active packet coordinate $2k/n^{\beta}$ converges to a Gamma distribution with shape $(2a-1)/\beta$. For characters modulo one fixed modulus, distinct character packets become asymptotically orthogonal with normalized correlation $O(k^{-1/(2\beta)})$. The principal signed Möbius sector remains outside this theorem. In the zeta specialization, its subpolynomial coherence is equivalent to the Riemann hypothesis.

All figures were generated by the accompanying Python program. The release contains 20 vector figures, matching PNG files, exact tables, 61 built-in checks, 18 independent tests, and the complete source listing.

Status and claim boundary

The paper proves the Stieltjes-germ formulas, the character transform at $s = 1$, the detail-variance identity, the primitive-resultant criterion, complete monotonicity, finite Hankel positivity, the zero-free resolvent theorem, the squarefree zeta-ratio formula, the packet-energy asymptotic, the Gamma scaling limit, and the character-decoherence theorem under the hypotheses stated below.

Classical inputs include the Laurent expansion of the Hurwitz zeta function, Dirichlet-character orthogonality, Bernoulli multiplication, the squarefree counting law, partial summation, the method of moments, and the Báez-Duarte criterion. The logarithmic quantum Hamiltonian used in the final part was realized in the 2026 quantum-dynamics work of Wei and collaborators [7].

The phrase "generated special function" refers to the exact packet construction and its proved laws. It does not by itself establish historical priority for every scalar specialization. The principal Möbius coherence statement remains open and is equivalent to RH after normalization by the proved diagonal energy law.

Contents

Status and claim boundary 1

I Cyclotomic germs of Bernoulli refinement 5

1 Bernoulli–Hurwitz children 5

2 Cyclotomic Stieltjes packet theorem 7

3 Character packets and derivatives at one 9

II Scalar invariants of a complete refinement 10

4 Holomorphic detail variance 11

5 Bernoulli–Euler and three-channel examples 12

6 Primitive packet resultants 14

III Positive arithmetic packet functions 15

7 The squarefree packet energy 15

8 Hankel positivity and the zero-free resolvent 16

9 A finite-difference zeta-ratio coordinate system 18

IV Asymptotic packet geometry 20

10 The diagonal energy law 20

11 Universal Gamma scaling 22

12 Character packet decoherence **24**

V Quantum realization and the remaining principal obstruction **26**

13 A logarithmic packet Hamiltonian **26**

14 The principal Möbius coherence problem **28**

15 Research questions generated by the calculus **30**

A Formula registry **31**

B Numerical certificate tables **32**

C Implementation map **33**

D Complete Python source **33**

E Reproduction **47**

List of Figures

1.1 The second special-function construction begins with one Bernoulli parent and its exact children. The finite child transform produces local Stieltjes packets, scalar detail invariants, and a positive arithmetic energy family. 6

2.1 The real and imaginary parts of the first cyclotomic Stieltjes packets for a four-channel refinement. The channels contain distinct local Taylor data even though they reconstruct one Bernoulli parent. 8

2.2 The first two-channel detail and its tangent at the removable point. The linear coefficient is the exact constant $-(\log 2)/\sqrt{2}$. 9

3.1 The nonprincipal character modulo four gives the same values from generalized Stieltjes packets and from derivatives of the Dirichlet beta function at one. The visible numerical differences come from independent differentiation of the L-function implementation. . . . 10

4.1 The same refinement viewed in child coordinates and cyclotomic coordinates. The non-DC transform channels contain exactly the scalar variance lost by replacing all children by their parent mean. 12

5.1 The nonnegative two-channel variance for three even Bernoulli orders. The Euler polynomial controls the complete detail energy of the Bernoulli split. 13

5.2 The three exact order-four Bernoulli children. Two children coincide, while the remaining difference produces the rational detail variance $32/27$. 14

6.1 Primitive packet resultants at integer orders for three cyclotomic depths. The plotted scalar is a Galois-invariant compression of all primitive detail channels. 15

7.1 The first nine alternating finite-difference layers of the principal packet energy. Every displayed quantity is nonnegative, as required by the exact Hausdorff moment representation. 16

8.1 The largest and smallest eigenvalues of finite packet-energy Hankel matrices. The tiny negative values at the largest sizes are at floating-point roundoff. The exact Gram representation is strictly positive. 17

8.2 The magnitude of the principal packet-energy resolvent on the slit plane. The excluded positive-real cut contains the poles and their accumulation at one. The theorem rules out zeros everywhere else. 18

9.1 Direct squarefree summation and the exact finite difference of $\zeta(s)/\zeta(2s)$ agree. The small displayed difference is only the truncation error of the direct numerical sum. 20

10.1 The scaled principal energy approaches the exact constant $3/(2^{5/2}\pi^{3/2})$. The final relative error in the finite computation is about 0.16 percent. 22

11.1 The empirical active-packet distributions approach the Gamma$(3/2, 1)$ density. The horizontal coordinate is the intrinsic inverse scale $U_k = 2k/n^2$. 23

11.2 The weighted distribution approaches its Gamma limit as the packet order increases. The plotted distance is a finite numerical certificate rather than the proof, which follows from the moment asymptotics. 24

12.1 The absolute normalized Gram matrix for the four characters modulo five at two packet orders. The diagonal remains one while the off-diagonal channels decay. 25

12.2 The largest off-diagonal correlation in the character packet bank. The theorem gives the general $k^{-1/(2\beta)}$ upper rate. The finite values need not be monotone at every intermediate order. 26

13.1 The positive and signed quantum packet channels. The logarithmic Hamiltonian supplies real-time phases. The scale Hamiltonian localizes the active number packets. 28

14.1 The exact boundary between the proved positive theory and the open signed theory. The positive energy and cross-character layers are controlled. The principal Möbius coherence remains open. 29

15.1 The analytic and arithmetic branches of the second generated family. Cyclotomic germs, positive moments, Gamma scaling, and character Gram matrices now belong to one exact refinement system. 31

Part I

Cyclotomic germs of Bernoulli refinement

Chapter 1

Bernoulli–Hurwitz children

The preceding paper introduced the entire Bernoulli interpolation

$$\mathfrak{B}(z;a) = -z\zeta(1-z,a)$$

and the exact packet family

$$\mathfrak{B}_{M,r}(z;a) = -zM^{z-1}\zeta\left(1-z,\frac{a+r}{M}\right).$$

The reconstruction and refinement laws are

$$\mathfrak{B}(z;a) = \sum_{r=0}^{M-1}\mathfrak{B}_{M,r}(z;a)$$

and

$$\mathfrak{B}_{M,r}(z;a) = \sum_{\ell=0}^{L-1}\mathfrak{B}_{LM,r+\ell M}(z;a).$$

These formulas were proved in [1]. The present paper asks what local and scalar information is created by one refinement step.

Fix integers $M \geq 1$, $L \geq 2$, and $0 \leq r < M$. Set

$$Q = LM, \qquad q_\ell = \frac{a + r + \ell M}{LM}, \qquad 0 \leq \ell < L.$$

The children of the parent packet are

$$b_\ell(z; a) = \mathfrak{B}_{Q, r+\ell M}(z; a) = -zQ^{z-1}\zeta(1 - z, q_\ell).$$

Let

$$\omega_L = \mathrm{e}^{2\pi \mathrm{i}/L}.$$

The unitary child transform is

$$\boxed{W_j(z; a) = \frac{1}{\sqrt{L}} \sum_{\ell=0}^{L-1} \omega_L^{-j\ell} b_\ell(z; a), \qquad 0 \leq j < L.}$$

The DC channel is the parent packet divided by $\sqrt{L}$:

$$W_0(z; a) = \frac{1}{\sqrt{L}} \mathfrak{B}_{M,r}(z; a).$$

The remaining channels record the exact detail created by the refinement.

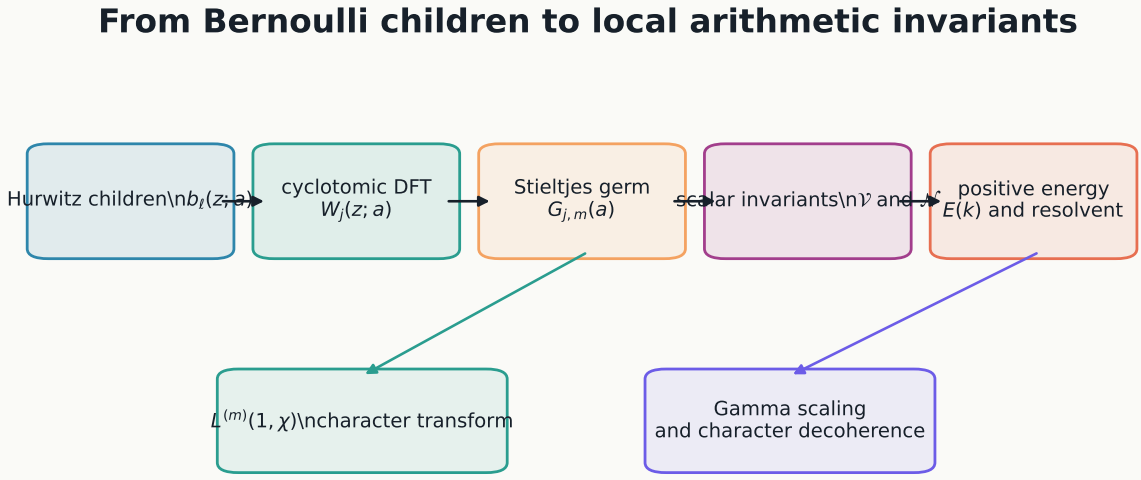


Figure 1.1: The second special-function construction begins with one Bernoulli parent and its exact children. The finite child transform produces local Stieltjes packets, scalar detail invariants, and a positive arithmetic energy family.

Chapter 2

Cyclotomic Stieltjes packet theorem

The generalized Stieltjes constants are defined by the Laurent expansion

$$\zeta(s,q) = \frac{1}{s-1} + \sum_{m=0}^{\infty} \frac{(-1)^m}{m!}\gamma_m(q)(s-1)^m.$$

Substituting $s = 1 - z$ yields

$$\zeta(1-z,q) = -\frac{1}{z} + \sum_{m=0}^{\infty} \gamma_m(q)\frac{z^m}{m!}.$$

Definition 2.1 (Cyclotomic Stieltjes packet)**.** For $0 \le j < L$ and $m \ge 0$, define

$$\boxed{G_{j,m}(a) = \frac{1}{\sqrt{L}} \sum_{\ell=0}^{L-1} \omega_L^{-j\ell}\gamma_m(q_\ell).}$$

Theorem 2.2 (Complete local packet expansion)**.** *For every channel j,*

$$\boxed{W_j(z;a) = \frac{\mathrm{e}^{z\log Q}}{Q}\left[\sqrt{L}\,\delta_{j0} - \sum_{m=0}^{\infty} G_{j,m}(a)\frac{z^{m+1}}{m!}\right].}$$

In particular, every non-DC detail has a zero at the removable Bernoulli point:

$$\boxed{W_j(0;a) = 0 \qquad (j \ne 0).}$$

For $j \ne 0$ and $n \ge 1$,

$$\boxed{W_j^{(n)}(0;a) = -\frac{n}{Q}\sum_{m=0}^{n-1}\binom{n-1}{m}(\log Q)^{n-1-m}G_{j,m}(a).}$$

Proof. Insert the Laurent expansion into the child definition:

$$\begin{aligned} b_\ell(z;a) &= -zQ^{z-1}\left[-\frac{1}{z} + \sum_{m\ge 0}\gamma_m(q_\ell)\frac{z^m}{m!}\right] \\ &= \frac{\mathrm{e}^{z\log Q}}{Q}\left[1 - \sum_{m\ge 0}\gamma_m(q_\ell)\frac{z^{m+1}}{m!}\right]. \end{aligned}$$

Apply the finite Fourier transform over ℓ. The transform of the constant vector is $\sqrt{L}$ in the DC channel and zero in every detail channel. This proves the first formula and the zero at $z = 0$.

For a non-DC channel, multiply the two power series

$$\mathrm{e}^{z\log Q} \quad \text{and} \quad -\sum_{m\ge 0} G_{j,m}(a)\frac{z^{m+1}}{m!}.$$

The coefficient of z^n is

$$-\frac{1}{Q}\sum_{m=0}^{n-1}\frac{G_{j,m}(a)}{m!}\frac{(\log Q)^{n-1-m}}{(n-1-m)!}.$$

Multiplication by $n!$ gives the derivative formula. □

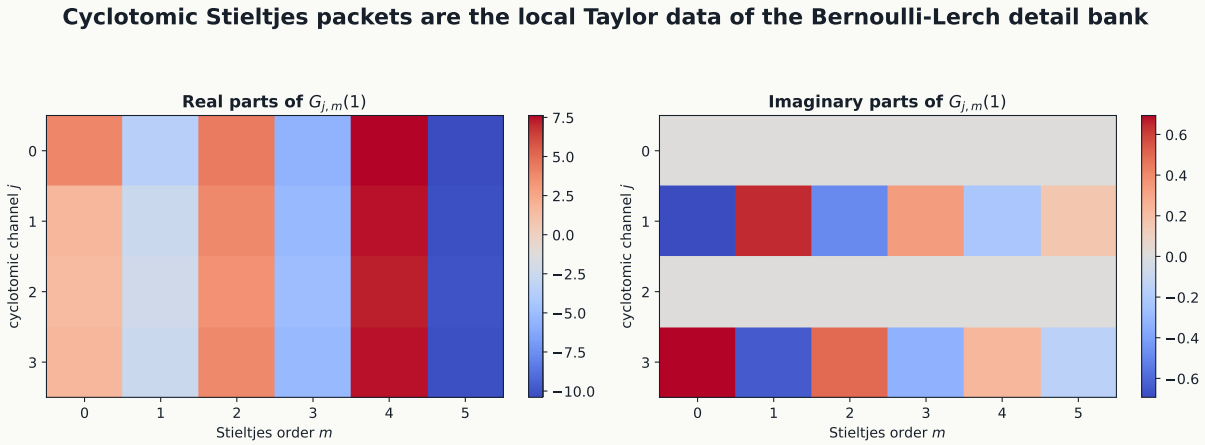


Figure 2.1: The real and imaginary parts of the first cyclotomic Stieltjes packets for a four-channel refinement. The channels contain distinct local Taylor data even though they reconstruct one Bernoulli parent.

Worked example 2.3 (The first Bernoulli–Euler detail coefficient)**.** Take $L = 2$, $M = 1$, $r = 0$, and $a = 1$. Then

$$q_0 = \frac{1}{2}, \qquad q_1 = 1.$$

The first non-DC packet is

$$G_{1,0}(1) = \frac{\gamma_0(1/2) - \gamma_0(1)}{\sqrt{2}}.$$

Since $\gamma_0(q) = -\psi(q)$ and

$$\psi(1/2) = -\gamma - 2\log 2, \qquad \psi(1) = -\gamma,$$

we have

$$G_{1,0}(1) = \sqrt{2}\log 2.$$

The derivative theorem gives

$$\boxed{W_1'(0;1) = -\frac{\log 2}{\sqrt{2}}.}$$

The first four derivatives computed from the packet formula and from direct high-precision differentiation are shown below.

n	packet formula	direct derivative	absolute error
1	-0.490129071734	-0.490129071734	2.62e-62
2	0.226088771874	0.226088771874	5.55e-17
3	0.138676210506	0.138676210506	2.27e-62
4	0.026626672189	0.026626672189	1.11e-16

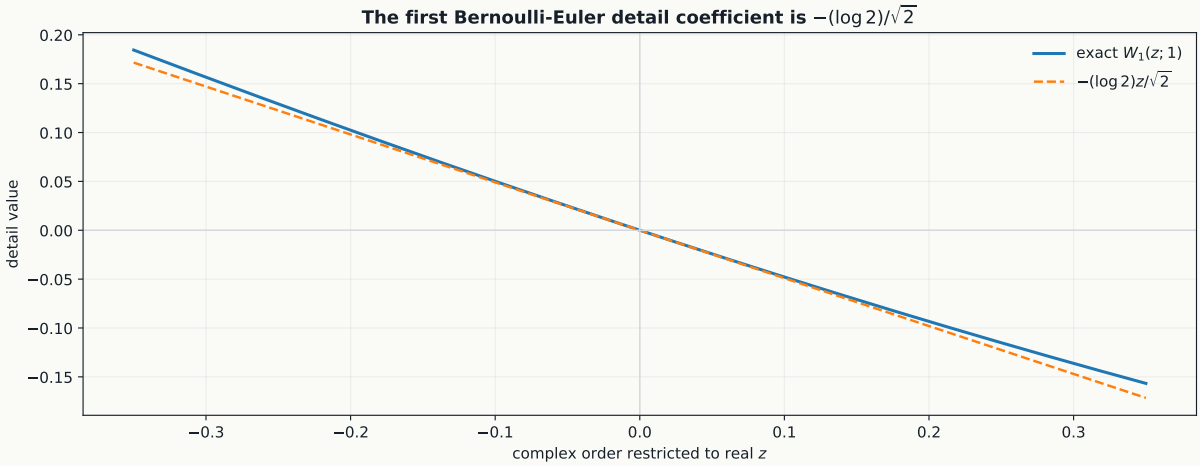


Figure 2.2: The first two-channel detail and its tangent at the removable point. The linear coefficient is the exact constant $-(\log 2)/\sqrt{2}$.

Chapter 3

Character packets and derivatives at one

The additive child transform has a multiplicative counterpart. Let χ be a nonprincipal Dirichlet character modulo q. The classical Hurwitz representation is

$$q^s L(s,\chi) = \sum_{r=1}^{q} \chi(r)\zeta\left(s, \frac{r}{q}\right).$$

The pole terms cancel because

$$\sum_{r=1}^{q} \chi(r) = 0.$$

Definition 3.1 (Character Stieltjes packet)**.** Define

$$S_m(\chi) = \sum_{r=1}^{q} \chi(r)\gamma_m\left(\frac{r}{q}\right).$$

Theorem 3.2 (Character transform at $s = 1$)**.** *For every $m \geq 0$,*

$$\boxed{S_m(\chi) = (-1)^m q \sum_{h=0}^{m} \binom{m}{h} (\log q)^{m-h} L^{(h)}(1,\chi).}$$

Proof. Write $s = 1 + w$. The pole-free left side has expansion

$$\sum_{m\geq 0} \frac{(-1)^m}{m!} S_m(\chi) w^m.$$

The right side is

$$q e^{w \log q} \sum_{h\geq 0} \frac{L^{(h)}(1,\chi)}{h!} w^h.$$

Equate the coefficient of w^m and multiply by $m!$. □

The first two cases are

$$\boxed{\sum_{r=1}^{q} \chi(r)\gamma_0\left(\frac{r}{q}\right) = qL(1,\chi)}$$

and

$$\boxed{\sum_{r=1}^{q} \chi(r)\gamma_1\left(\frac{r}{q}\right) = -q\left[L'(1,\chi) + (\log q)L(1,\chi)\right].}$$

The packet family therefore joins the special values $L(1-n,\chi)$, carried by the integer Bernoulli samples, to the local derivatives $L^{(m)}(1,\chi)$, carried by the Stieltjes germ.

m	Stieltjes packet sum	L-derivative formula	error
0	3.1415926536	3.1415926536	0
1	-5.1267774478	-5.1267774482	4.31e-10
2	7.5603277998	7.5603278010	1.19e-09
3	-10.6337651115	-10.6337651140	2.48e-09

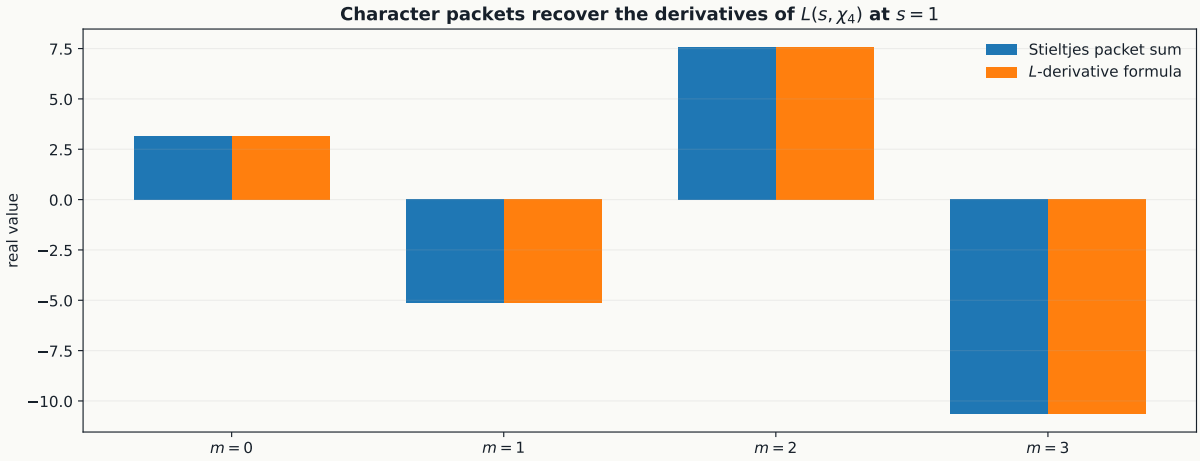


Figure 3.1: The nonprincipal character modulo four gives the same values from generalized Stieltjes packets and from derivatives of the Dirichlet beta function at one. The visible numerical differences come from independent differentiation of the L-function implementation.

Part II

Scalar invariants of a complete refinement

Chapter 4

Holomorphic detail variance

Let $b_0, \dots, b_{L-1}$ be the child packets at fixed (z, a) and let W_j be their normalized finite Fourier transform.

Definition 4.1 (Holomorphic detail variance)**.** Define

$$\boxed{\mathcal{V}_L = \sum_{\ell=0}^{L-1} b_\ell^2 - \frac{1}{L}\left(\sum_{\ell=0}^{L-1} b_\ell\right)^2.}$$

Theorem 4.2 (Three exact forms)**.** *The detail variance satisfies*

$$\boxed{\mathcal{V}_L = \sum_{j=1}^{L-1} W_j W_{L-j}}$$

and

$$\boxed{\mathcal{V}_L = \frac{1}{L} \sum_{0 \le \ell < m < L} (b_\ell - b_m)^2.}$$

For real z and real a, the children are real and

$$\boxed{\mathcal{V}_L = \sum_{j=1}^{L-1} |W_j|^2 \ge 0.}$$

It vanishes precisely when every child value is equal.

Proof. Fourier orthogonality gives

$$\sum_{j=0}^{L-1} W_j W_{L-j} = \sum_{\ell=0}^{L-1} b_\ell^2.$$

The DC term is

$$W_0^2 = \frac{1}{L}\left(\sum_{\ell} b_\ell\right)^2.$$

Subtracting it proves the first identity. Expanding all pairwise squares gives

$$\sum_{\ell<m} (b_\ell - b_m)^2 = L \sum_{\ell} b_\ell^2 - \left(\sum_{\ell} b_\ell\right)^2,$$

which proves the second identity. For real children, $W_{L-j} = \overline{W_j}$, so the first identity becomes a sum of squares. □

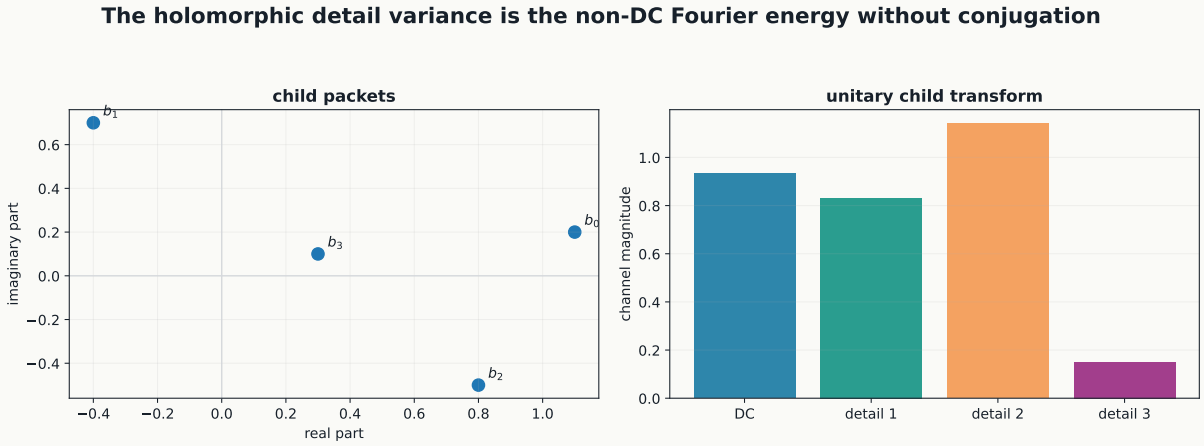


Figure 4.1: The same refinement viewed in child coordinates and cyclotomic coordinates. The non-DC transform channels contain exactly the scalar variance lost by replacing all children by their parent mean.

Chapter 5

Bernoulli–Euler and three-channel examples

Proposition 5.1 (Euler detail as variance square root)**.** *In the two-channel refinement of the Bernoulli parent,*

$$W_1(n;a) = -\frac{n}{2\sqrt{2}} E_{n-1}(a).$$

Consequently,

$$\boxed{\mathcal{V}_2(n;a) = \frac{n^2}{8} E_{n-1}(a)^2.}$$

Proof. The two children at integer order are

$$b_0 = 2^{n-1} B_n(a/2), \qquad b_1 = 2^{n-1} B_n((a+1)/2).$$

The standard Bernoulli–Euler relation gives

$$b_0 - b_1 = -\frac{n}{2} E_{n-1}(a).$$

Divide by $\sqrt{2}$ to obtain W_1. Since $\mathcal{V}_2 = W_1^2$, the variance formula follows. □

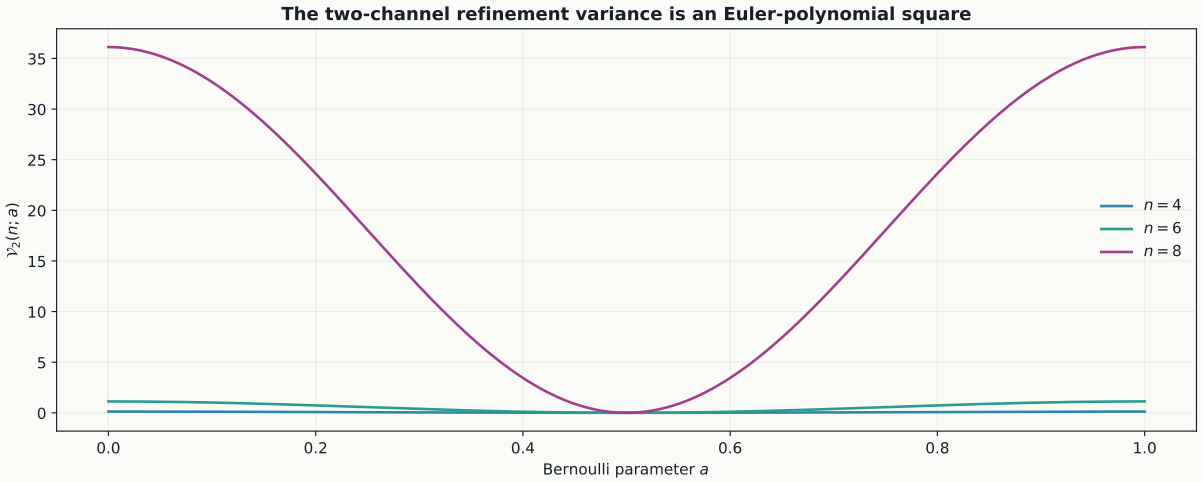


Figure 5.1: The nonnegative two-channel variance for three even Bernoulli orders. The Euler polynomial controls the complete detail energy of the Bernoulli split.

Worked example 5.2 (A three-channel rational variance)**.** Take $L = 3$, $M = 1$, $r = 0$, $a = 0$, and $n = 4$. The exact child values are

$$(b_0, b_1, b_2) = \left(-\frac{9}{10}, \frac{13}{30}, \frac{13}{30}\right).$$

Therefore

$$\begin{aligned}\mathcal{V}_3 &= \frac{1}{3}\left[\left(-\frac{4}{3}\right)^2 + 0^2 + \left(\frac{4}{3}\right)^2\right] \\ &= \boxed{\frac{32}{27}}.\end{aligned}$$

The two primitive channels are conjugate and satisfy

$$W_1 W_2 = \frac{16}{27}.$$

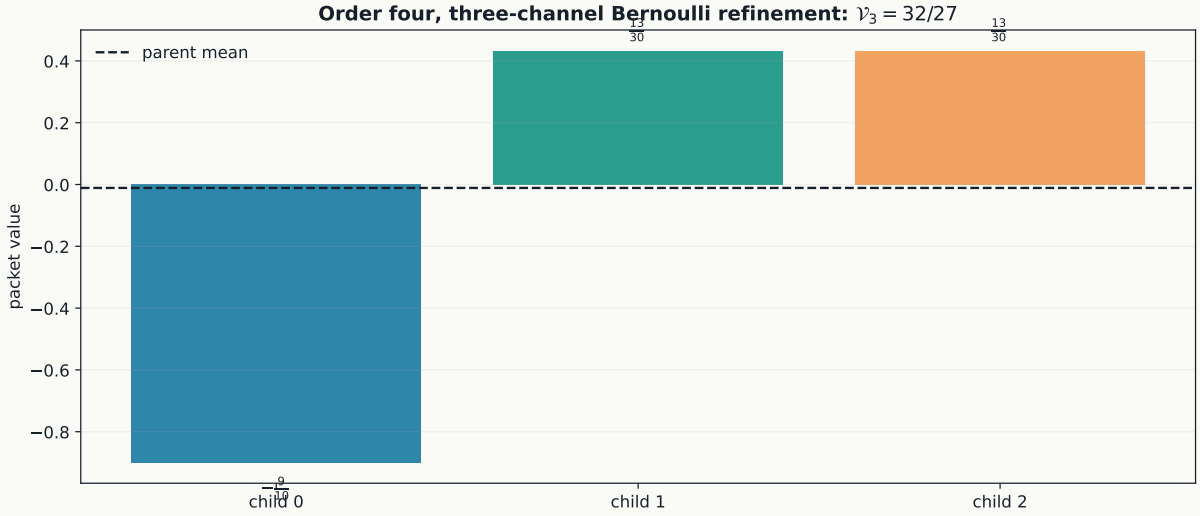


Figure 5.2: The three exact order-four Bernoulli children. Two children coincide, while the remaining difference produces the rational detail variance 32/27.

Chapter 6

Primitive packet resultants

Define the child polynomial

$$P_{M,r}^{(L)}(X; z, a) = \sum_{\ell=0}^{L-1} b_\ell(z; a) X^\ell.$$

At a root of unity,

$$P_{M,r}^{(L)}(\omega_L^{-j}; z, a) = \sqrt{L}\, W_j(z; a).$$

Definition 6.1 (Primitive packet resultant)**.** Let Φ_L be the Lth cyclotomic polynomial. Define

$$\boxed{\mathcal{N}_{M,r}^{(L)}(z; a) = \mathrm{Res}_X \left(\Phi_L(X), P_{M,r}^{(L)}(X; z, a) \right).}$$

Equivalently,

$$\mathcal{N}_{M,r}^{(L)}(z; a) = \prod_{\substack{1 \le j \le L \\ (j,L)=1}} P_{M,r}^{(L)}(\omega_L^j; z, a).$$

Theorem 6.2 (Primitive collapse criterion)**.** *The primitive resultant is entire in z. It vanishes exactly when at least one primitive cyclotomic detail channel vanishes. If $a \in \mathbb{Q}$ and $n \in \mathbb{N}$, then*

$$\boxed{\mathcal{N}_{M,r}^{(L)}(n; a) \in \mathbb{Q}.}$$

Proof. The resultant is a polynomial in the coefficients $b_\ell(z; a)$, so it is entire. Since Φ_L is monic, the product formula for the resultant is the product of P over all primitive Lth roots. Therefore it is zero exactly when one primitive evaluation is zero. At integer order and rational a, every child equals a rational multiple of a Bernoulli-polynomial value at a rational point, hence is rational. The resultant of two rational polynomials is rational. □

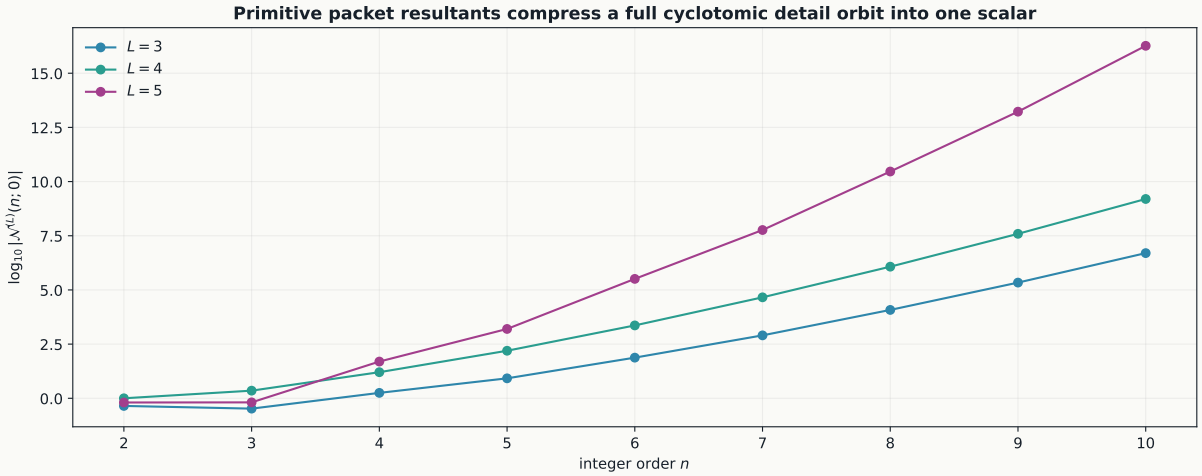


Figure 6.1: Primitive packet resultants at integer orders for three cyclotomic depths. The plotted scalar is a Galois-invariant compression of all primitive detail channels.

Part III

Positive arithmetic packet functions

Chapter 7

The squarefree packet energy

Let $a > 1/2$, $\beta > 0$, and let χ be a Dirichlet character modulo q. Define

$$\boxed{E_{a,\beta,\chi}(k) = \sum_{n=1}^{\infty} \frac{\mu(n)^2\,|\chi(n)|^2}{n^{2a}} \left(1 - \frac{1}{n^{\beta}}\right)^{2k}.}$$

Set

$$w_n = \frac{\mu(n)^2\,|\chi(n)|^2}{n^{2a}}, \qquad x_n = \left(1 - \frac{1}{n^{\beta}}\right)^2.$$

Then

$$E_{a,\beta,\chi}(k) = \sum_n w_n x_n^k.$$

The support lies in $[0, 1)$ and the total mass is finite because $a > 1/2$.

Theorem 7.1 (Complete monotonicity)**.** *For all integers $m, k \geq 0$,*

$$\boxed{(-1)^m \Delta^m E_{a,\beta,\chi}(k) = \sum_n w_n x_n^k (1 - x_n)^m \geq 0.}$$

Thus $E_{a,\beta,\chi}$ is a Hausdorff moment sequence. In particular it is decreasing and log-convex:

$$E(k)^2 \leq E(k-1)E(k+1).$$

Proof. Since $\Delta x^k = x^k(x-1)$, repeated finite differences give

$$\Delta^m x^k = x^k(x-1)^m.$$

Sum against the nonnegative weights w_n. Log-convexity is the Cauchy–Schwarz inequality applied to $x_n^{(k-1)/2}$ and $x_n^{(k+1)/2}$ in the weighted sequence space. □

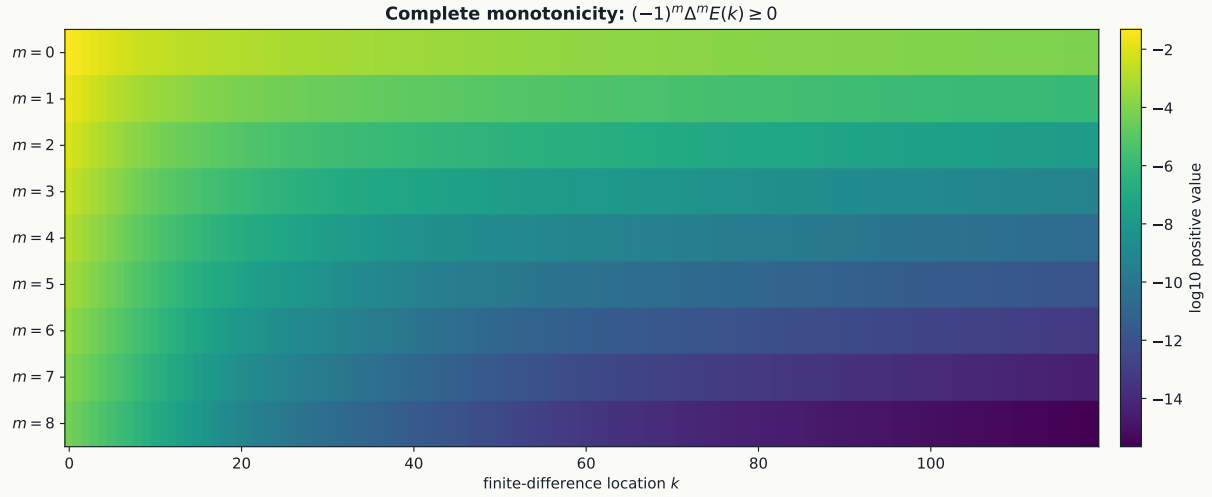


Figure 7.1: The first nine alternating finite-difference layers of the principal packet energy. Every displayed quantity is nonnegative, as required by the exact Hausdorff moment representation.

Chapter 8

Hankel positivity and the zero-free resolvent

Theorem 8.1 (Strict finite Hankel positivity)**.** *For every $d \geq 0$, the Hankel matrix*

$$H_d = (E(i+j))_{0 \leq i,j \leq d}$$

is positive definite.

Proof. For a nonzero vector $c = (c_0, \ldots, c_d)$,

$$\begin{aligned} c^* H_d c &= \sum_{i,j} c_i \overline{c_j} E(i+j) \\ &= \sum_n w_n \left| \sum_{i=0}^{d} c_i x_n^i \right|^2 . \end{aligned}$$

The support contains infinitely many distinct x_n with positive weight. A nonzero polynomial of degree at most d cannot vanish at all of them. Hence the sum is strictly positive. □

The smallest computed eigenvalues eventually reach floating-point roundoff. The theorem is exact and does not depend on those numerical signs.

size	smallest eigenvalue	largest eigenvalue
2	6.345e-04	1.075e-01
3	1.865e-06	1.197e-01
4	6.036e-10	1.253e-01
5	8.948e-14	1.282e-01
6	-2.410e-17	1.297e-01
7	-2.218e-17	1.307e-01
8	-3.034e-17	1.313e-01
9	-3.447e-17	1.317e-01
10	-3.586e-17	1.320e-01

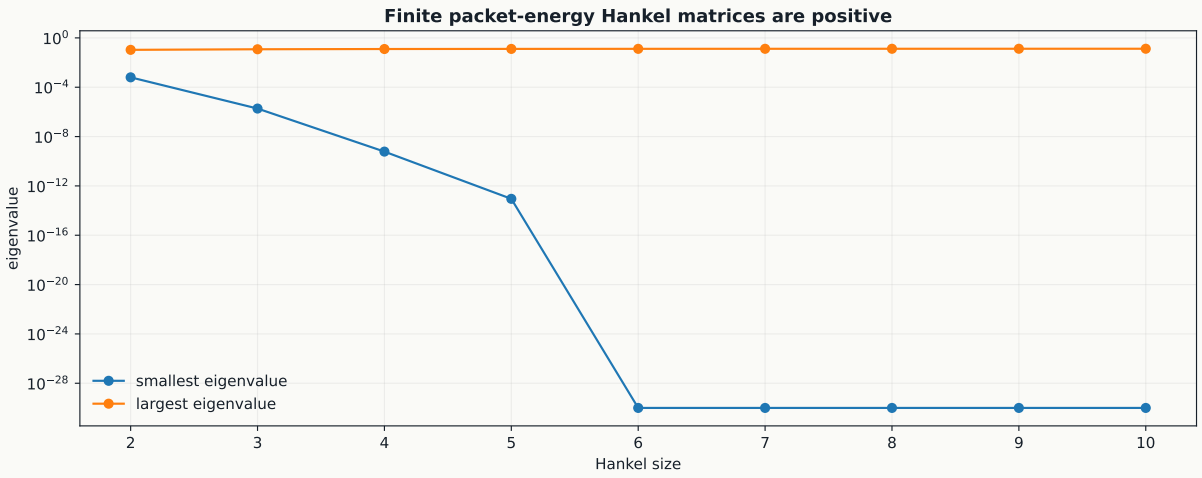


Figure 8.1: The largest and smallest eigenvalues of finite packet-energy Hankel matrices. The tiny negative values at the largest sizes are at floating-point roundoff. The exact Gram representation is strictly positive.

Definition 8.2 (Packet-energy resolvent)**.** Define

$$\boxed{\mathcal{E}_{a,\beta,\chi}(z) = \sum_{k=0}^{\infty} E_{a,\beta,\chi}(k) z^k.}$$

For $|z| < 1$,

$$\mathcal{E}_{a,\beta,\chi}(z) = \sum_{n} \frac{w_n}{1 - z x_n}.$$

Theorem 8.3 (Zero-free slit plane)**.** *The resolvent extends analytically to*

$$\mathbb{C} \setminus [1, \infty)$$

and satisfies

$$\boxed{\mathcal{E}_{a,\beta,\chi}(z) \neq 0 \qquad (z \in \mathbb{C} \setminus [1, \infty)).}$$

Proof. Uniform convergence on compact subsets of the slit plane follows from $\sum_n w_n < \infty$ and a uniform lower bound on $|1 - zx|$ for $x \in [0, 1]$.

For $z \neq 0$, set $u = 1/z$ and

$$S(u) = \sum_n \frac{w_n}{u - x_n}.$$

Then

$$\mathcal{E}(z) = uS(u).$$

If $\Im u > 0$, then

$$\Im S(u) = -\Im u \sum_n \frac{w_n}{|u - x_n|^2} < 0.$$

The sign reverses in the lower half-plane. If $u > 1$, then every summand is positive. If $u < 0$, every summand is negative. These cases cover the image of the slit domain under $u = 1/z$. Therefore $S(u)$ and $\mathcal{E}(z)$ never vanish. At $z = 0$, $\mathcal{E}(0) = \sum_n w_n > 0$. □

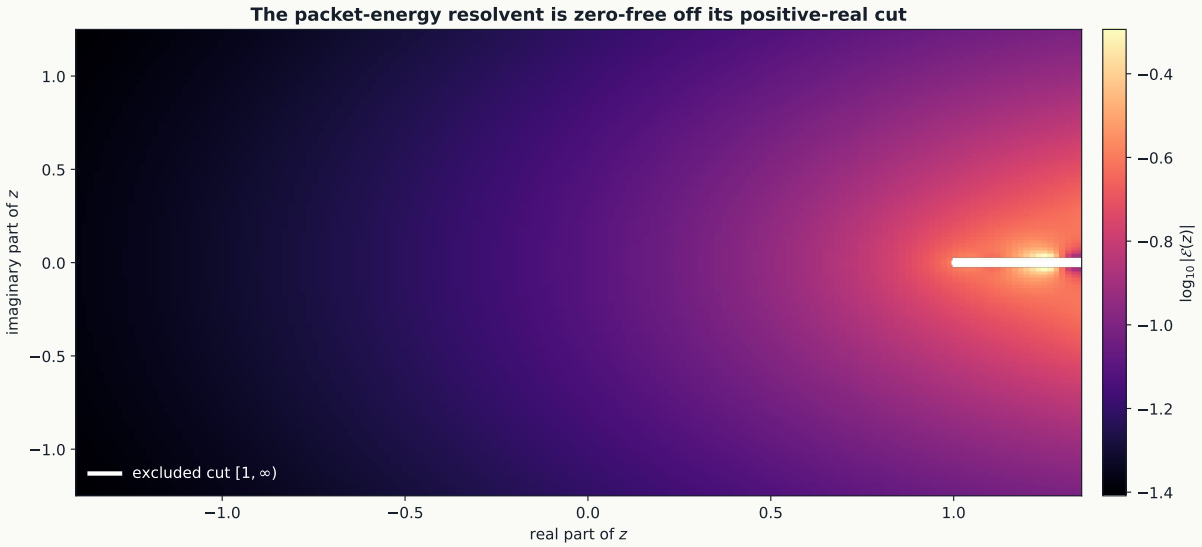


Figure 8.2: The magnitude of the principal packet-energy resolvent on the slit plane. The excluded positive-real cut contains the poles and their accumulation at one. The theorem rules out zeros everywhere else.

Chapter 9

A finite-difference zeta-ratio coordinate system

Because $|\chi(n)|^2$ is the indicator of $(n, q) = 1$, define

$$D_q(s) = \sum_{\substack{n \geq 1 \\ (n,q)=1}} \frac{\mu(n)^2}{n^s}.$$

Its Euler product is

$$\boxed{D_q(s) = \frac{\zeta(s)}{\zeta(2s)} \prod_{p|q} \left(1 + p^{-s}\right)^{-1}.}$$

Theorem 9.1 (Exact zeta-ratio finite difference)**.** *For every integer $k \geq 0$,*

$$\boxed{E_{a,\beta,\chi}(k) = \sum_{j=0}^{2k} (-1)^j \binom{2k}{j} D_q(2a + \beta j).}$$

Proof. Expand the filter by the binomial theorem:

$$\left(1 - n^{-\beta}\right)^{2k} = \sum_{j=0}^{2k} (-1)^j \binom{2k}{j} n^{-\beta j}.$$

Absolute convergence permits interchange with the squarefree Dirichlet series. The inner sum is $D_q(2a + \beta j)$. □

k	direct energy	zeta-ratio difference	truncation error
1	4.7804610e-02	4.7804610e-02	7.494e-16
2	3.0200677e-02	3.0200677e-02	4.406e-16
3	1.9767515e-02	1.9767515e-02	3.227e-16
4	1.3470094e-02	1.3470094e-02	2.481e-16
5	9.5803728e-03	9.5803728e-03	1.665e-16
6	7.1098189e-03	7.1098189e-03	1.639e-16
7	5.4893011e-03	5.4893011e-03	1.266e-16
8	4.3883096e-03	4.3883096e-03	8.153e-17

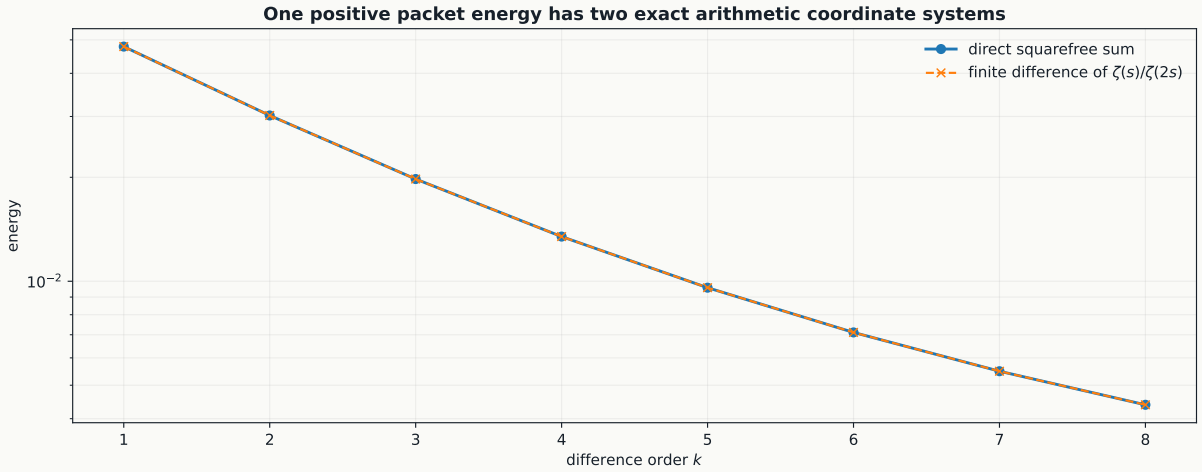


Figure 9.1: Direct squarefree summation and the exact finite difference of $\zeta(s)/\zeta(2s)$ agree. The small displayed difference is only the truncation error of the direct numerical sum.

Part IV

Asymptotic packet geometry

Chapter 10

The diagonal energy law

Let

$$A_q(x) = \sum_{\substack{n \le x \\ (n,q)=1}} \mu(n)^2.$$

The elementary squarefree counting estimate is

$$A_q(x) = \delta_q x + O_q(\sqrt{x}),$$

where

$$\boxed{\delta_q = \frac{1}{\zeta(2)} \prod_{p \mid q} \frac{p}{p+1}.}$$

Set

$$\alpha = \frac{2a-1}{\beta}.$$

Theorem 10.1 (Packet-energy asymptotic)**.** *For $a > 1/2$ and $\beta > 0$,*

$$\boxed{E_{a,\beta,\chi}(k) \sim \frac{\delta_q}{\beta} \Gamma(\alpha)(2k)^{-\alpha}.}$$

More precisely, the error produced by the $O_q(\sqrt{x})$ remainder is

$$O_{a,\beta,q}\left(k^{-(2a-1/2)/\beta}\right).$$

Proof. Use Stieltjes summation with

$$f_k(x) = x^{-2a}(1 - x^{-\beta})^{2k}.$$

The density term contributes

$$\delta_q \int_1^\infty f_k(x)\,\mathrm{d}x.$$

With $y = x^{-\beta}$,

$$\begin{aligned}\int_1^\infty f_k(x)\,\mathrm{d}x &= \frac{1}{\beta}\int_0^1 y^{\alpha-1}(1-y)^{2k}\,\mathrm{d}y \\ &= \frac{1}{\beta}\mathrm{B}(\alpha, 2k+1).\end{aligned}$$

The beta-function asymptotic gives

$$\mathrm{B}(\alpha, 2k+1) \sim \Gamma(\alpha)(2k)^{-\alpha}.$$

Write the squarefree counting remainder as $R_q(x) = O_q(\sqrt{x})$. Partial summation bounds its contribution by a constant times

$$\int_1^\infty \sqrt{x}\,|f_k'(x)|\,\mathrm{d}x.$$

Scaling $x = k^{1/\beta}y$ shows that this integral is

$$O\Big(k^{-(2a-1/2)/\beta}\Big),$$

which is smaller than the main term by $k^{-1/(2\beta)}$. □

For $a = \beta = 2$ and the principal character,

$$\boxed{E_k \sim \frac{3}{2^{5/2}\pi^{3/2}}k^{-3/2}.}$$

k	E_k	$k^{3/2}E_k$	relative error
100	9.397187e-05	0.0939718675	1.332e-02
300	1.844163e-05	0.0958255236	6.143e-03
1000	3.028409e-06	0.0957666957	5.525e-03
3000	5.760934e-07	0.0946618062	6.076e-03
10000	9.539419e-08	0.0953941855	1.614e-03
30000	1.833371e-08	0.0952647535	2.551e-04
100000	3.007053e-09	0.0950913727	1.565e-03

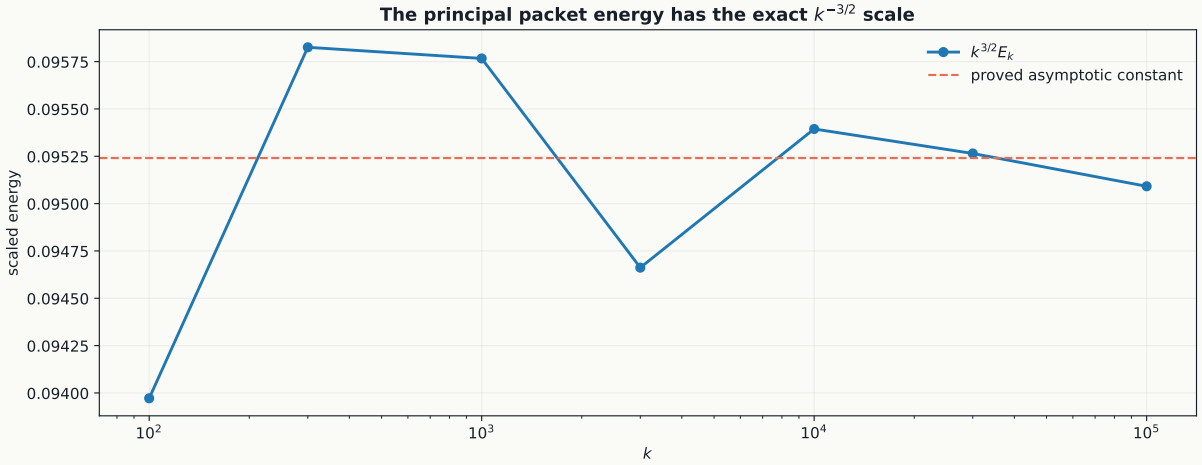


Figure 10.1: The scaled principal energy approaches the exact constant $3/(2^{5/2}\pi^{3/2})$. The final relative error in the finite computation is about 0.16 percent.

Chapter 11

Universal Gamma scaling

Normalize the packet weights into a probability distribution:

$$\mathbb{P}_k(n) = \frac{\mu(n)^2\,|\chi(n)|^2\,n^{-2a}(1-n^{-\beta})^{2k}}{E_{a,\beta,\chi}(k)}.$$

Define

$$\boxed{U_k = \frac{2k}{n^\beta}.}$$

Theorem 11.1 (Gamma scaling limit)**.** *As $k \to \infty$,*

$$\boxed{U_k \Longrightarrow \mathrm{Gamma}(\alpha, 1), \qquad \alpha = \frac{2a-1}{\beta}.}$$

Equivalently, for every bounded continuous φ,

$$\lim_{k\to\infty} \sum_n \mathbb{P}_k(n)\varphi\left(\frac{2k}{n^\beta}\right) = \frac{1}{\Gamma(\alpha)} \int_0^\infty \varphi(u) u^{\alpha-1} \mathrm{e}^{-u}\,\mathrm{d}u.$$

Proof. For every integer $m \geq 0$,

$$\mathbb{E}_k[U_k^m] = (2k)^m \frac{E_{a+\beta m/2,\beta,\chi}(k)}{E_{a,\beta,\chi}(k)}.$$

Apply the packet-energy asymptotic to numerator and denominator. The squarefree density cancels and gives

$$\lim_{k\to\infty} \mathbb{E}_k[U_k^m] = \frac{\Gamma(\alpha+m)}{\Gamma(\alpha)}.$$

These are the moments of $\mathrm{Gamma}(\alpha, 1)$. The Gamma distribution is moment-determinate. The first moments remain bounded, so the family is tight. The method of moments gives weak convergence. □

For the principal RH parameters $a = \beta = 2$,

$$\alpha = \frac{3}{2}.$$

The active packet window therefore has a complete limiting profile, not only the rough location $n \asymp \sqrt{k}$.

k	mean	variance	$q_{0.1}$	$q_{0.5}$	$q_{0.9}$	KS
300	1.54561	1.54638	0.31798	1.18207	3.01649	0.05780
3000	1.50231	1.59381	0.29310	1.17179	3.29253	0.04726
30000	1.49275	1.48729	0.29110	1.20548	3.08603	0.02256
100000	1.50160	1.52410	0.29473	1.16989	3.10136	0.01560

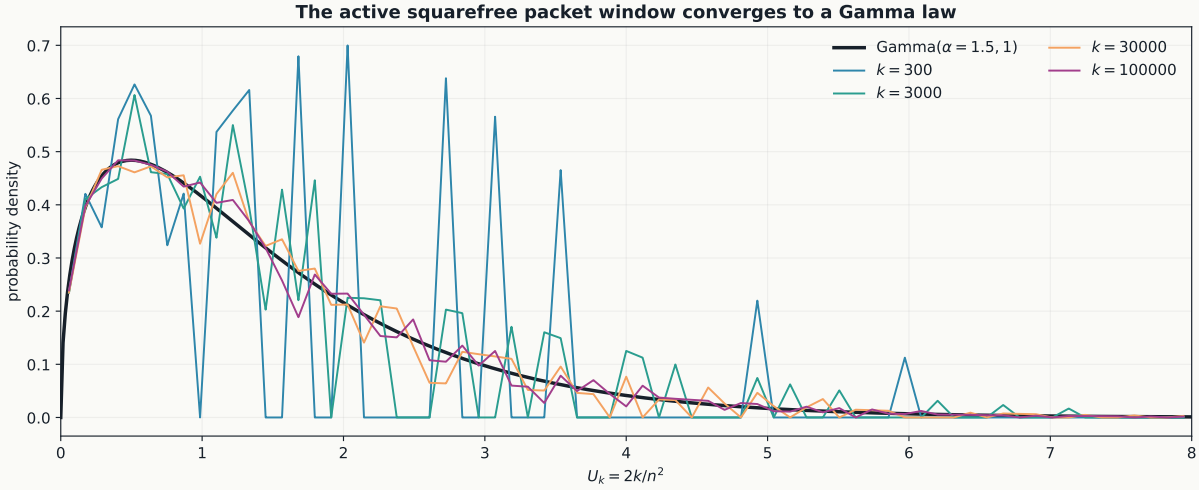


Figure 11.1: The empirical active-packet distributions approach the Gamma$(3/2, 1)$ density. The horizontal coordinate is the intrinsic inverse scale $U_k = 2k/n^2$.

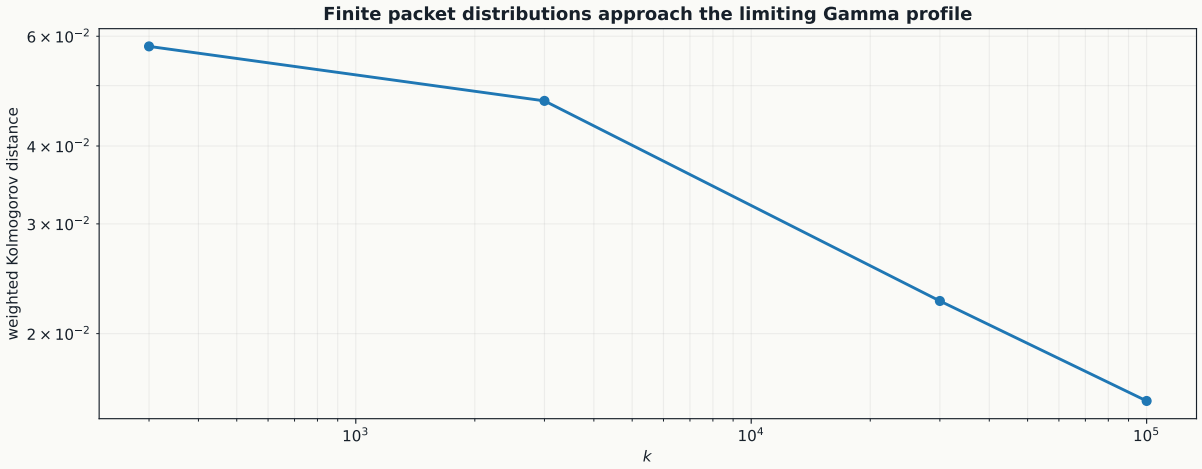


Figure 11.2: The weighted distribution approaches its Gamma limit as the packet order increases. The plotted distance is a finite numerical certificate rather than the proof, which follows from the moment asymptotics.

Chapter 12

Character packet decoherence

Let χ and ψ be Dirichlet characters modulo the same integer q. Define the packet Gram entry

$$\boxed{G_{\chi,\psi}(k) = \sum_{n=1}^{\infty} \frac{\mu(n)^2 \chi(n)\overline{\psi(n)}}{n^{2a}} \left(1 - \frac{1}{n^{\beta}}\right)^{2k}.}$$

The diagonal is the positive packet energy.

Lemma 12.1 (Squarefree character cancellation)**.** *If ξ is a nonprincipal Dirichlet character modulo q, then*

$$\boxed{\sum_{n \le X} \mu(n)^2 \xi(n) = O_q(\sqrt{X}).}$$

Proof. Use

$$\mu(n)^2 = \sum_{d^2 \mid n} \mu(d).$$

Then

$$\sum_{n \le X} \mu(n)^2 \xi(n) = \sum_{d \le \sqrt{X}} \mu(d) \xi(d^2) \sum_{m \le X/d^2} \xi(m).$$

A nonprincipal character has zero sum over a complete period, so every partial sum is $O(q)$. There are $O(\sqrt{X})$ outer terms. □

Theorem 12.2 (Character decoherence)**.** *If $\chi \neq \psi$, then*

$$\boxed{G_{\chi,\psi}(k) = O_{a,\beta,q}\left(k^{-(2a-1/2)/\beta}\right).}$$

Consequently,

$$\boxed{\frac{G_{\chi,\psi}(k)}{\sqrt{G_{\chi,\chi}(k)G_{\psi,\psi}(k)}} = O_{a,\beta,q}\left(k^{-1/(2\beta)}\right).}$$

Proof. The product $\xi = \chi\overline{\psi}$ is nonprincipal. Let

$$A_\xi(X) = \sum_{n\le X} \mu(n)^2\xi(n) = O_q(\sqrt{X}).$$

Apply partial summation to

$$f_k(x) = x^{-2a}(1-x^{-\beta})^{2k}.$$

The resulting bound is a constant times

$$\int_1^\infty \sqrt{x}\,|f_k'(x)|\,\mathrm{d}x = O\Big(k^{-(2a-1/2)/\beta}\Big).$$

The two diagonal entries are asymptotic to positive constants times $k^{-(2a-1)/\beta}$. Dividing gives the normalized rate. □

For $a = \beta = 2$, the normalized decay rate is

$$\boxed{k^{-1/4}.}$$

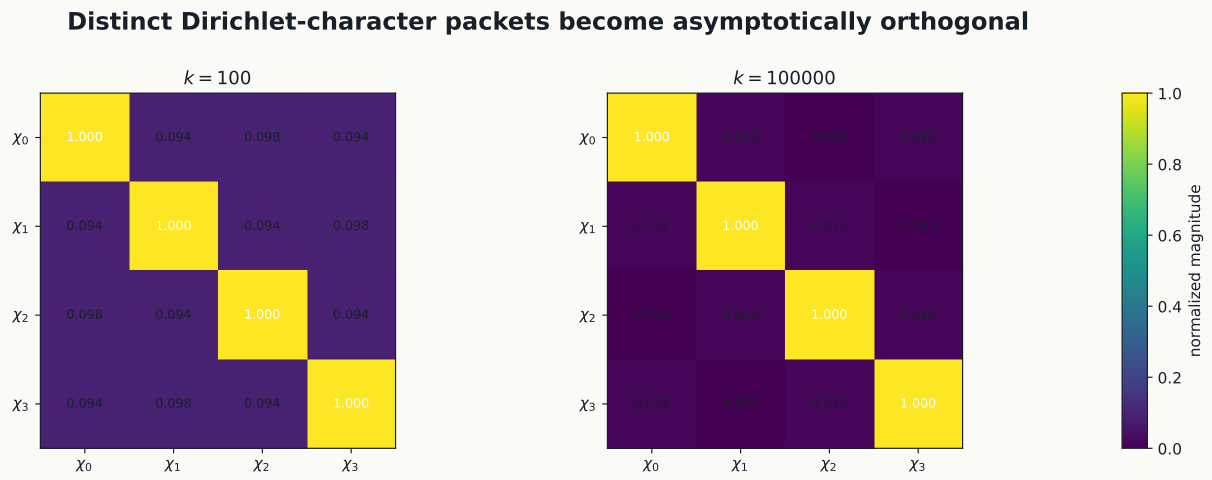


Figure 12.1: The absolute normalized Gram matrix for the four characters modulo five at two packet orders. The diagonal remains one while the off-diagonal channels decay.

k	largest normalized off-diagonal entry
100	0.09807746
300	0.07180785
1000	0.05948305
3000	0.05265039
10000	0.01447532
30000	0.01243919
100000	0.01185877

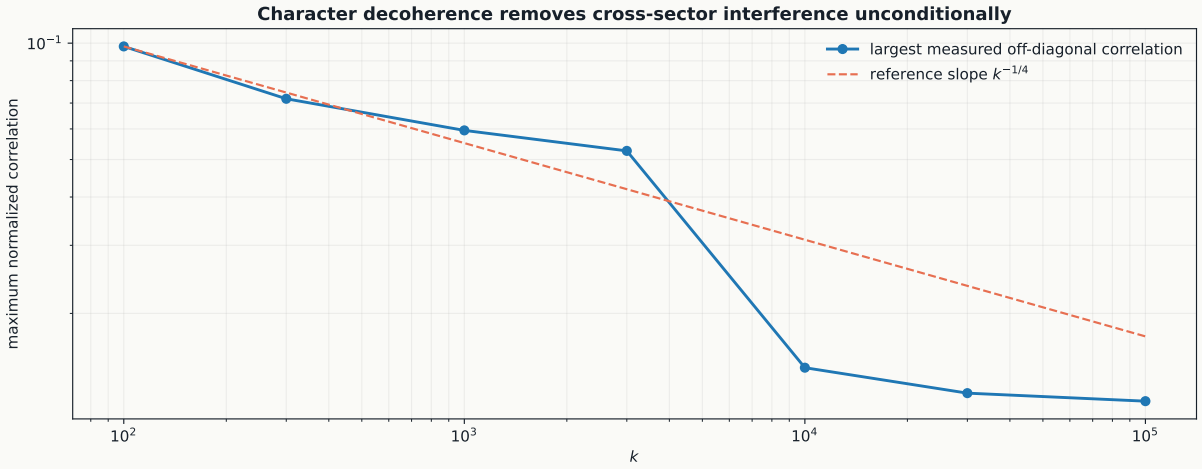


Figure 12.2: The largest off-diagonal correlation in the character packet bank. The theorem gives the general $k^{-1/(2\beta)}$ upper rate. The finite values need not be monotone at every intermediate order.

Part V

Quantum realization and the remaining principal obstruction

Chapter 13

A logarithmic packet Hamiltonian

Work in

$$\mathcal{H} = \ell^2(\{2, 3, \ldots\}).$$

For $a > 1$, define the amplitude vector

$$|\phi_a\rangle = \sum_{n\geq 2} n^{-a/2} |n\rangle .$$

For $a > 1/2$, define the diagonal-power vector

$$|\psi_a\rangle = \sum_{n\geq 2} n^{-a} |n\rangle .$$

Let

$$H |n\rangle = \log n \ |n\rangle ,$$
$$K_\beta |n\rangle = -\log(1 - n^{-\beta}) |n\rangle ,$$

and

$$\mathcal{M}_\chi |n\rangle = \mu(n)\chi(n) |n\rangle .$$

The three operators are diagonal and commute.

Proposition 13.1 (Quantum transfer amplitude)**.** *The generalized signed transfer is*

$$\boxed{\mathcal{C}_{a,\beta,\chi}(k,t) = \langle \phi_a | \, \mathcal{M}_\chi \mathrm{e}^{-kK_\beta} \mathrm{e}^{-\mathrm{i}tH} \, | \phi_a \rangle \, .}$$

Its positive diagonal power is

$$\boxed{E_{a,\beta,\chi}(k) = \langle \psi_a | \, \mathcal{M}_\chi^* \mathcal{M}_\chi \mathrm{e}^{-2kK_\beta} \, | \psi_a \rangle \, .}$$

It is also the long-time mean power of the signed transfer:

$$\boxed{E_{a,\beta,\chi}(k) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |\mathcal{C}_{a,\beta,\chi}(k,t)|^2 \, \mathrm{d}t.}$$

Proof. The first matrix element has number-basis factors

$$n^{-a}, \qquad \mu(n)\chi(n), \qquad (1-n^{-\beta})^k, \qquad n^{-\mathrm{i}t},$$

which give the signed transfer. The second matrix element has factors

$$n^{-2a}, \qquad \mu(n)^2 \, |\chi(n)|^2 \, , \qquad (1-n^{-\beta})^{2k},$$

which give the positive packet energy. Finally, expand the squared modulus of the transfer. The time average of $\mathrm{e}^{-\mathrm{i}t(\log n - \log m)}$ vanishes when $n \neq m$ and equals one when $n = m$. Only the diagonal terms remain. □

The logarithmic Hamiltonian is not introduced only as formal notation. Wei and collaborators construct a quantum system with spectrum $E_n = \log n$, thermal weights $n^{-\beta}$, and dynamical phases $n^{-\mathrm{i}t}$ [7]. Their observable directly encodes a zeta amplitude. The present packet system uses the same logarithmic spectral coordinate, then adds the squarefree or character sign and the scale filter e^{-kK_β}.

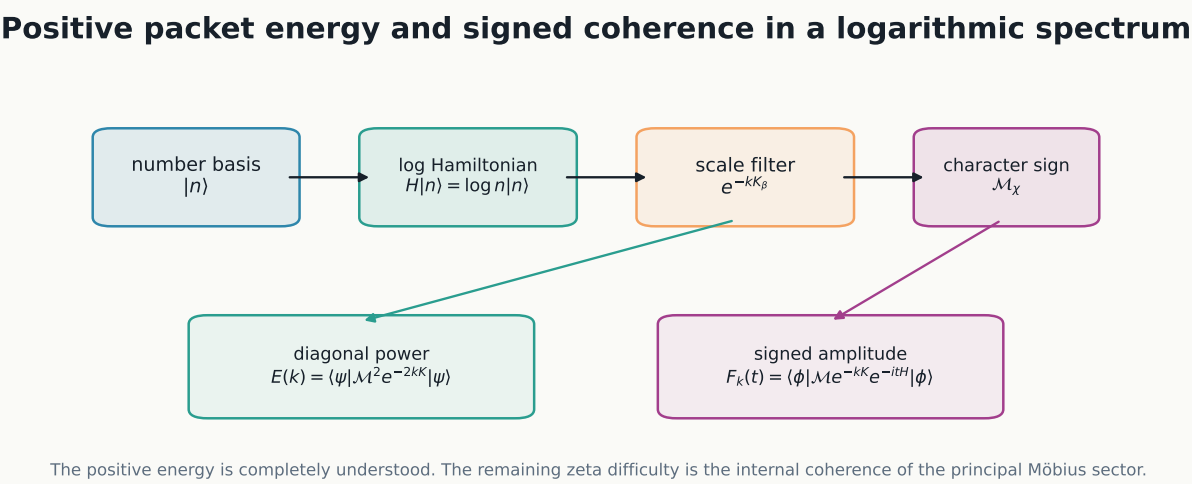


Figure 13.1: The positive and signed quantum packet channels. The logarithmic Hamiltonian supplies real-time phases. The scale Hamiltonian localizes the active number packets.

Chapter 14

The principal Möbius coherence problem

In the principal zeta specialization, set

$$a = 2, \qquad \beta = 2, \qquad \chi = 1.$$

Then

$$c_k = \sum_{n=1}^{\infty} \frac{\mu(n)}{n^2} \left(1 - \frac{1}{n^2}\right)^k$$

which is the Báez-Duarte coefficient. Define

$$\Gamma_k = \frac{|c_k|^2}{E_k}.$$

The diagonal theorem gives

$$E_k \asymp k^{-3/2}.$$

The Báez-Duarte criterion gives

$$\mathrm{RH} \iff c_k = O_\varepsilon(k^{-3/4+\varepsilon}) \quad \text{for every } \varepsilon > 0.$$

Therefore:

Theorem 14.1 (Normalized principal coherence criterion)**.**

$$\boxed{\mathrm{RH} \iff \Gamma_k = O_\varepsilon(k^{\varepsilon}) \quad \textit{for every } \varepsilon > 0.}$$

Proof. If RH holds, square the Báez-Duarte estimate and divide by $E_k \asymp k^{-3/2}$. This gives $\Gamma_k = O_\varepsilon(k^{2\varepsilon})$, which is the stated form after renaming the arbitrary exponent.

Conversely, if $\Gamma_k = O_\varepsilon(k^\varepsilon)$, then

$$|c_k|^2 \ll_\varepsilon k^{-3/2+\varepsilon}.$$

Taking square roots gives the Báez-Duarte estimate after another renaming of the arbitrary exponent. □

The character-decoherence theorem controls interference between distinct character sectors. It does not control the internal Möbius coherence of the principal sector. That remaining signed estimate carries the RH-scale difficulty.

Conjecture 14.2 (Principal packet coherence)**.** *For every $\varepsilon > 0$,*

$$\boxed{\left|\sum_{n=1}^{\infty} \frac{\mu(n)}{n^2}\left(1-\frac{1}{n^2}\right)^k\right|^2 \le C_\varepsilon k^\varepsilon \sum_{n=1}^{\infty} \frac{\mu(n)^2}{n^4}\left(1-\frac{1}{n^2}\right)^{2k}.}$$

This conjecture is equivalent to RH. It is not promoted as a proof.

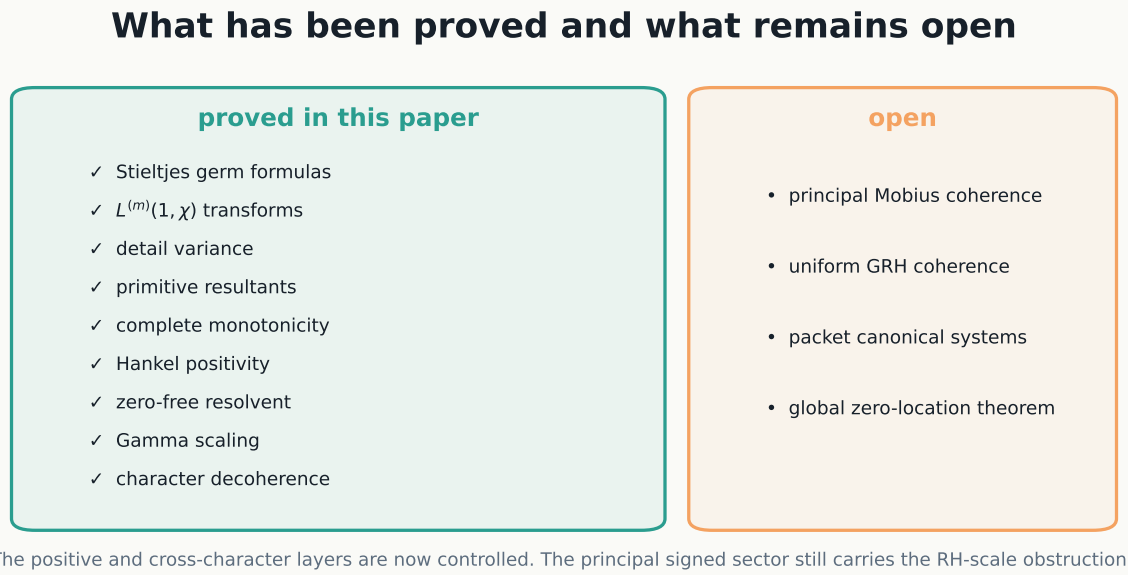


Figure 14.1: The exact boundary between the proved positive theory and the open signed theory. The positive energy and cross-character layers are controlled. The principal Möbius coherence remains open.

Chapter 15

Research questions generated by the calculus

The results above create several concrete questions that are separate from RH.

1. Determine the zero sets and growth order of the primitive packet resultants $\mathcal{N}_{M,r}^{(L)}(z;a)$.
2. Classify the real parameters for which the holomorphic detail variance has only real or nonnegative integer samples.
3. Determine whether the Stieltjes packet matrices $G_{j,m}(a)$ obey finite-rank, total-positivity, or variation-diminishing laws for selected cyclotomic channels.
4. Obtain higher-order asymptotic expansions for $E_{a,\beta,\chi}(k)$ and the Gamma scaling law.
5. Sharpen the elementary $O(k^{-1/(2\beta)})$ character-decoherence rate using stronger character-sum estimates.
6. Construct orthogonal polynomials and continued fractions associated with the positive moment sequence $E(k)$.
7. Determine the spectral measure and boundary behavior of the zero-free resolvent $\mathcal{E}(z)$.
8. Extend the packet Gram theory from Dirichlet characters to automorphic coefficient systems.
9. Decide whether the principal Möbius coherence can be controlled by a local packet inequality rather than a global zero argument.

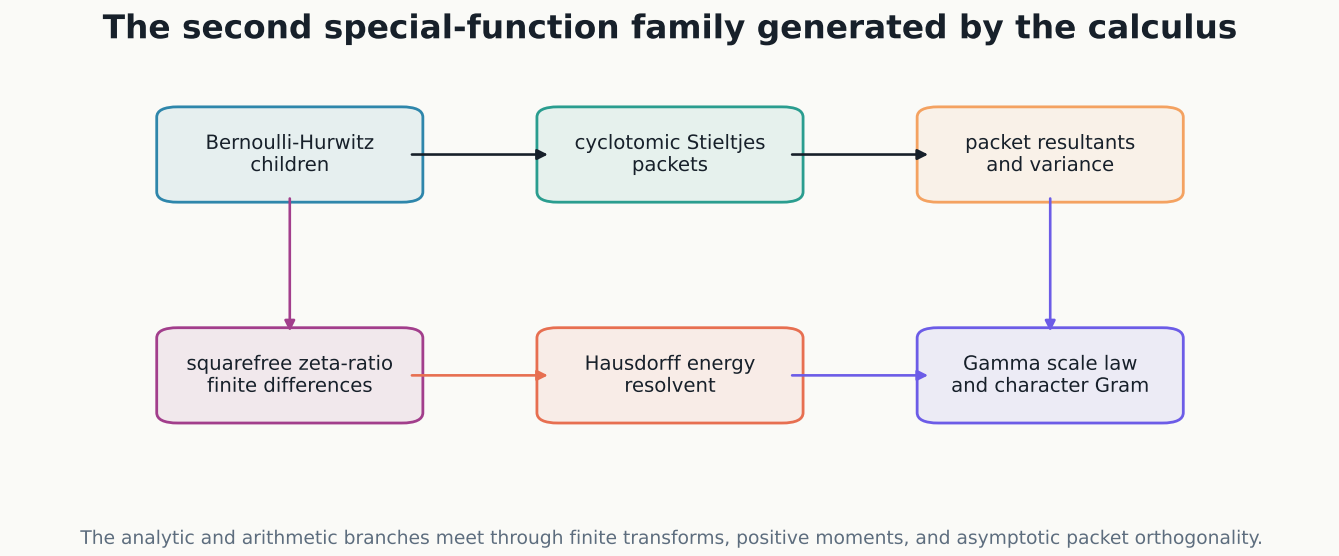


Figure 15.1: The analytic and arithmetic branches of the second generated family. Cyclotomic germs, positive moments, Gamma scaling, and character Gram matrices now belong to one exact refinement system.

Chapter A

Formula registry

Object	Exact formula
Cyclotomic Stieltjes packet	$G_{j,m}(a) = L^{-1/2}\sum_{\ell}\omega_L^{-j\ell}\gamma_m(q_\ell)$
Detail germ	$W_j(z;a) = Q^{-1}\mathrm{e}^{z\log Q}[\sqrt{L}\delta_{j0} - \sum_m G_{j,m}z^{m+1}/m!]$
Character transform	$S_m(\chi) = (-1)^m q\sum_{h\le m}\binom{m}{h}(\log q)^{m-h}L^{(h)}(1,\chi)$
Detail variance	$\mathcal{V}_L = \sum b_\ell^2 - L^{-1}(\sum b_\ell)^2$
Primitive resultant	$\mathcal{N} = \mathrm{Res}(\Phi_L, P)$
Packet energy	$E(k) = \sum_n w_n x_n^k$
Complete monotonicity	$(-1)^m\Delta^m E(k) = \sum_n w_n x_n^k(1-x_n)^m$
Resolvent	$\mathcal{E}(z) = \sum_n w_n/(1-zx_n)$
Zeta-ratio coordinate	$E(k) = \sum_{j=0}^{2k}(-1)^j\binom{2k}{j}D_q(2a+\beta j)$
Energy asymptotic	$E(k) \sim (\delta_q/\beta)\Gamma(\alpha)(2k)^{-\alpha}$
Gamma coordinate	$U_k = 2k/n^\beta \Rightarrow \mathrm{Gamma}(\alpha,1)$
Character decoherence	$\mathrm{Corr}(\chi,\psi) = O(k^{-1/(2\beta)})$ for $\chi\neq\psi$

Chapter B

Numerical certificate tables

Packet-energy asymptotic

k	E_k	$k^{3/2}E_k$	relative error
100	9.397187e-05	0.0939718675	1.332e-02
300	1.844163e-05	0.0958255236	6.143e-03
1000	3.028409e-06	0.0957666957	5.525e-03
3000	5.760934e-07	0.0946618062	6.076e-03
10000	9.539419e-08	0.0953941855	1.614e-03
30000	1.833371e-08	0.0952647535	2.551e-04
100000	3.007053e-09	0.0950913727	1.565e-03

Gamma scaling

k	mean	variance	$q_{0.1}$	$q_{0.5}$	$q_{0.9}$	KS
300	1.54561	1.54638	0.31798	1.18207	3.01649	0.05780
3000	1.50231	1.59381	0.29310	1.17179	3.29253	0.04726
30000	1.49275	1.48729	0.29110	1.20548	3.08603	0.02256
100000	1.50160	1.52410	0.29473	1.16989	3.10136	0.01560

Character decoherence

k	largest normalized off-diagonal entry
100	0.09807746
300	0.07180785
1000	0.05948305
3000	0.05265039
10000	0.01447532
30000	0.01243919
100000	0.01185877

Verification count

61/61 checks passed

The independent test suite contains 18 additional tests. It rederives the main identities without reading the built-in check ledger.

Chapter C

Implementation map

Function	Role
`bernoulli_packet`	Evaluates one Bernoulli–Hurwitz child
`stieltjes_packet`	Computes one cyclotomic Stieltjes packet
`detail_derivative_from_stieltjes`	Builds the local Taylor derivatives
`detail_variance`	Evaluates the scalar non-DC variance
`primitive_packet_resultant_exact`	Computes the exact cyclotomic resultant
`packet_energy_values`	Evaluates the positive squarefree energy
`energy_resolvent`	Evaluates the slit-plane resolvent
`energy_from_zeta_ratio`	Uses the exact finite-difference coordinate
`gamma_weighted_sample`	Builds the active inverse-scale distribution
`character_gram_matrix`	Constructs the finite character packet Gram matrix

Chapter D

Complete Python source

The following program generates every figure, every numerical table, and the complete built-in check ledger used in this paper.

```
from __future__ import annotations

import cmath
import csv
import json
import math
from dataclasses import dataclass
from pathlib import Path
from typing import Iterable, Sequence

import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import Circle, FancyArrowPatch, FancyBboxPatch, Rectangle
import mpmath as mp
import numpy as np
import sympy as sp
from scipy.special import gamma as gamma_fn
from scipy.special import gammainc

mp.mp.dps = 60

ROOT = Path(__file__).resolve().parent
FIGURES = ROOT / "figures"
RESULTS = ROOT / "results"
TABLES = ROOT / "tables"
for directory in (FIGURES, RESULTS, TABLES):
    directory.mkdir(parents=True, exist_ok=True)

INK = "#17202A"
BLUE = "#2E86AB"
TEAL = "#2A9D8F"
ORANGE = "#F4A261"
MAGENTA = "#A23E8C"
```

```
RED = "#E76F51"
YELLOW = "#E9C46A"
PURPLE = "#6C5CE7"
MUTED = "#5D6D7E"
PAPER = "#FAFAF7"
GRID = "#D5D8DC"
WHITE = "#FFFFFF"
COLORS = [BLUE, TEAL, ORANGE, MAGENTA, RED, YELLOW, PURPLE]

plt.rcParams.update(
    {
        "figure.facecolor": PAPER,
        "axes.facecolor": PAPER,
        "axes.edgecolor": INK,
        "axes.labelcolor": INK,
        "xtick.color": INK,
        "ytick.color": INK,
        "text.color": INK,
        "font.size": 10.5,
        "axes.titlesize": 12.5,
        "axes.titleweight": "bold",
        "legend.frameon": False,
        "savefig.bbox": "tight",
        "savefig.facecolor": PAPER,
    }
)


def save_figure(fig: plt.Figure, stem: str) -> None:
    fig.savefig(FIGURES / f"{stem}.pdf")
    fig.savefig(FIGURES / f"{stem}.png", dpi=220)
    plt.close(fig)


def diagram_axes(figsize: tuple[float, float] = (12.5, 6.0)) -> tuple[plt.Figure, plt.Axes]:
    fig, ax = plt.subplots(figsize=figsize)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.axis("off")
    return fig, ax


def box(
    ax: plt.Axes,
    xy: tuple[float, float],
    width: float,
    height: float,
    text: str,
    face: str = WHITE,
    edge: str = INK,
    fontsize: float = 11.5,
) -> None:
    patch = FancyBboxPatch(
        xy,
        width,
        height,
        boxstyle="round,pad=.012,rounding_size=.018",
        facecolor=face,
        edgecolor=edge,
        linewidth=1.7,
    )
    ax.add_patch(patch)
    ax.text(xy[0] + width / 2, xy[1] + height / 2, text, ha="center", va="center", fontsize=fontsize)


def arrow(
    ax: plt.Axes,
    start: tuple[float, float],
    end: tuple[float, float],
    color: str = INK,
    text: str | None = None,
```

```
    text_offset: tuple[float, float] = (0.0, 0.03),
) -> None:
    patch = FancyArrowPatch(start, end, arrowstyle="-|>", mutation_scale=13, linewidth=1.6, color=color)
    ax.add_patch(patch)
    if text:
        ax.text(
            (start[0] + end[0]) / 2 + text_offset[0],
            (start[1] + end[1]) / 2 + text_offset[1],
            text,
            ha="center",
            va="center",
            fontsize=10.5,
            color=color,
        )


# ---------------------------------------------------------------------------
# Arithmetic and character helpers
# ---------------------------------------------------------------------------


def mobius_sieve(nmax: int) -> np.ndarray:
    mu = np.zeros(nmax + 1, dtype=np.int8)
    mu[1] = 1
    primes: list[int] = []
    composite = np.zeros(nmax + 1, dtype=bool)
    for i in range(2, nmax + 1):
        if not composite[i]:
            primes.append(i)
            mu[i] = -1
        for p in primes:
            ip = i * p
            if ip > nmax:
                break
            composite[ip] = True
            if i % p == 0:
                mu[ip] = 0
                break
            mu[ip] = -mu[i]
    return mu


def prime_factors(n: int) -> list[int]:
    factors: list[int] = []
    p = 2
    remaining = n
    while p * p <= remaining:
        if remaining % p == 0:
            factors.append(p)
            while remaining % p == 0:
                remaining //= p
        p += 1
    if remaining > 1:
        factors.append(remaining)
    return factors


def squarefree_coprime_density(q: int) -> float:
    value = 1.0 / float(mp.zeta(2))
    for p in prime_factors(q):
        value *= p / (p + 1)
    return value


def characters_mod5(nmax: int) -> np.ndarray:
    """Return the four Dirichlet characters modulo 5.

    Character j sends the primitive residue 2 to exp(2*pi*i*j/4).
    """
    log_map = {1: 0, 2: 1, 4: 2, 3: 3}
    chars = np.zeros((4, nmax + 1), dtype=np.complex128)
```

```
    for n in range(1, nmax + 1):
        residue = n % 5
        if residue == 0:
            continue
        exponent = log_map[residue]
        for j in range(4):
            chars[j, n] = np.exp(2j * np.pi * j * exponent / 4)
    return chars


def dirichlet_l_hurwitz(s: complex, character_values: Sequence[complex], q: int) -> complex:
    total = mp.mpc(0)
    for r in range(1, q + 1):
        chi = complex(character_values[r % q]) if len(character_values) == q else complex(character_values[r])
        if abs(chi) > 0:
            total += chi * mp.zeta(s, mp.mpf(r) / q)
    return complex(total / (mp.mpc(q) ** s))


# ------------------------------------------------------------------------
# Bernoulli-Hurwitz packets and Stieltjes germs
# ------------------------------------------------------------------------


def bernoulli_packet(z: complex, a: complex, modulus: int, residue: int) -> complex:
    if abs(z) < 1e-28:
        return 1.0 / modulus
    zz = mp.mpc(z)
    q = mp.mpc(a + residue) / modulus
    return complex(-zz * (mp.mpf(modulus) ** (zz - 1)) * mp.zeta(1 - zz, q))


def child_parameters(a: float, m: int, r: int, ell: int) -> tuple[int, list[mp.mpf]]:
    q_total = ell * m
    parameters = [mp.mpf(a + r + child * m) / q_total for child in range(ell)]
    return q_total, parameters


def stieltjes_packet(ell: int, m: int, r: int, a: float, j: int, order: int) -> complex:
    omega = mp.e ** (2j * mp.pi / ell)
    q_total, parameters = child_parameters(a, m, r, ell)
    total = mp.mpc(0)
    for child, parameter in enumerate(parameters):
        total += omega ** (-j * child) * mp.stieltjes(order, parameter)
    return complex(total / mp.sqrt(ell))


def detail_channel(z: complex, ell: int, m: int, r: int, a: float, j: int) -> complex:
    q_total = ell * m
    omega = cmath.exp(2j * math.pi / ell)
    values = [bernoulli_packet(z, a, q_total, r + child * m) for child in range(ell)]
    return sum(omega ** (-j * child) * values[child] for child in range(ell)) / math.sqrt(ell)


def detail_derivative_from_stieltjes(
    ell: int,
    m: int,
    r: int,
    a: float,
    j: int,
    derivative_order: int,
) -> complex:
    if j == 0:
        raise ValueError("The compact derivative formula below is for non-DC channels")
    q_total = ell * m
    log_q = math.log(q_total)
    total = 0j
    for order in range(derivative_order):
        g = stieltjes_packet(ell, m, r, a, j, order)
        total += math.comb(derivative_order - 1, order) * (log_q ** (derivative_order - 1 - order)) * g
    return -(derivative_order / q_total) * total
```

```
def detail_variance(children: Sequence[complex]) -> complex:
    values = np.asarray(children, dtype=np.complex128)
    return complex(np.sum(values * values) - np.sum(values) ** 2 / len(values))


def transformed_details(children: Sequence[complex]) -> np.ndarray:
    values = np.asarray(children, dtype=np.complex128)
    return np.fft.fft(values) / math.sqrt(len(values))


def bernoulli_children_exact(n: int, a: sp.Rational, ell: int, m: int = 1, r: int = 0) -> list[sp.Expr]:
    q_total = ell * m
    return [
        sp.simplify(q_total ** (n - 1) * sp.bernoulli(n, sp.Rational(a) + sp.Rational(r + child * m, q_total)))
        for child in range(ell)
    ]


def primitive_packet_resultant_exact(n: int, a: sp.Rational, ell: int, m: int = 1, r: int = 0) -> sp.Expr:
    x = sp.symbols("x")
    children = bernoulli_children_exact(n, a, ell, m, r)
    polynomial = sum(children[index] * x**index for index in range(ell))
    return sp.factor(sp.resultant(sp.cyclotomic_poly(ell, x), polynomial, x))


# ---------------------------------------------------------------------------
# Positive Mobius packet energy and resolvent
# ---------------------------------------------------------------------------


def packet_energy_values(
    k_values: Sequence[int],
    a: float,
    beta: float,
    nmax: int,
    q: int = 1,
    mu: np.ndarray | None = None,
) -> np.ndarray:
    if mu is None:
        mu = mobius_sieve(nmax)
    n = np.arange(2, nmax + 1, dtype=np.float64)
    support = (mu[2:] != 0).astype(np.float64)
    if q > 1:
        support *= (np.gcd(n.astype(np.int64), q) == 1).astype(np.float64)
    base = support * n ** (-2 * a)
    log_x = 2.0 * np.log1p(-n ** (-beta))
    results = []
    for k in k_values:
        results.append(float(np.dot(base, np.exp(k * log_x))))
    # n=1 contributes only at k=0.  It is omitted because the asymptotic theory and all tests use k>=1.
    return np.asarray(results)


def energy_asymptotic_constant(a: float, beta: float, q: int = 1) -> float:
    alpha = (2 * a - 1) / beta
    return squarefree_coprime_density(q) / beta * float(gamma_fn(alpha)) * (2.0 ** (-alpha))


def energy_asymptotic(k: np.ndarray | float, a: float, beta: float, q: int = 1) -> np.ndarray:
    alpha = (2 * a - 1) / beta
    constant = energy_asymptotic_constant(a, beta, q)
    return constant * np.asarray(k, dtype=float) ** (-alpha)


def finite_differences(values: np.ndarray, max_order: int) -> list[np.ndarray]:
    rows = [np.asarray(values, dtype=float)]
    current = np.asarray(values, dtype=float)
    for _ in range(max_order):
        current = np.diff(current)
```

```
        rows.append(current)
    return rows


def energy_resolvent(z: complex, a: float, beta: float, nmax: int, q: int, mu: np.ndarray) -> complex:
    n = np.arange(2, nmax + 1, dtype=np.float64)
    support = (mu[2:] != 0).astype(np.float64)
    if q > 1:
        support *= (np.gcd(n.astype(np.int64), q) == 1).astype(np.float64)
    weights = support * n ** (-2 * a)
    x = (1.0 - n ** (-beta)) ** 2
    return complex(np.sum(weights / (1.0 - z * x)))


def squarefree_dirichlet_ratio(s: float, q: int = 1) -> mp.mpf:
    value = mp.zeta(s) / mp.zeta(2 * s)
    for p in prime_factors(q):
        value /= 1 + mp.mpf(p) ** (-s)
    return value


def energy_from_zeta_ratio(k: int, a: float, beta: float, q: int = 1) -> mp.mpf:
    total = mp.mpf(0)
    for j in range(2 * k + 1):
        total += (-1) ** j * math.comb(2 * k, j) * squarefree_dirichlet_ratio(2 * a + beta * j, q)
    return total


# ---------------------------------------------------------------------------
# Gamma scaling and character decoherence
# ---------------------------------------------------------------------------


def gamma_weighted_sample(
    k: int,
    a: float,
    beta: float,
    nmax: int,
    q: int,
    mu: np.ndarray,
) -> tuple[np.ndarray, np.ndarray]:
    n = np.arange(2, nmax + 1, dtype=np.float64)
    support = (mu[2:] != 0).astype(np.float64)
    if q > 1:
        support *= (np.gcd(n.astype(np.int64), q) == 1).astype(np.float64)
    log_weight = -2 * a * np.log(n) + 2 * k * np.log1p(-n ** (-beta))
    log_weight = np.where(support > 0, log_weight, -np.inf)
    maximum = np.max(log_weight)
    weights = np.exp(log_weight - maximum)
    weights /= weights.sum()
    u = 2.0 * k / (n**beta)
    order = np.argsort(u)
    return u[order], weights[order]


def weighted_moments(values: np.ndarray, weights: np.ndarray, max_order: int) -> list[float]:
    return [float(np.dot(weights, values**order)) for order in range(1, max_order + 1)]


def weighted_quantile(values: np.ndarray, weights: np.ndarray, probabilities: Sequence[float]) -> np.ndarray:
    cumulative = np.cumsum(weights)
    return np.interp(np.asarray(probabilities), cumulative, values)


def ks_distance_to_gamma(values: np.ndarray, weights: np.ndarray, alpha: float) -> float:
    empirical = np.cumsum(weights)
    theoretical = gammainc(alpha, values)
    return float(np.max(np.abs(empirical - theoretical)))


def character_gram_matrix(
```

```
    k: int,
    a: float,
    beta: float,
    nmax: int,
    mu: np.ndarray,
    characters: np.ndarray,
) -> np.ndarray:
    n = np.arange(2, nmax + 1, dtype=np.float64)
    squarefree = (mu[2:] != 0).astype(np.float64)
    weights = squarefree * n ** (-2 * a) * np.exp(2 * k * np.log1p(-n ** (-beta)))
    char_slice = characters[:, 2 : nmax + 1]
    weighted = char_slice * np.sqrt(weights)[None, :]
    return weighted @ weighted.conj().T


def normalized_correlation(gram: np.ndarray) -> np.ndarray:
    diagonal = np.real(np.diag(gram))
    denom = np.sqrt(np.outer(diagonal, diagonal))
    return np.divide(gram, denom, out=np.zeros_like(gram), where=denom > 0)


# ---------------------------------------------------------------------------
# Figures and checks
# ---------------------------------------------------------------------------


@dataclass
class Check:
    name: str
    passed: bool
    detail: str


def generate_all() -> dict[str, object]:
    checks: list[Check] = []
    tables: dict[str, list[dict[str, object]]] = {}

    # Figure 01: conceptual pipeline.
    fig, ax = diagram_axes((13.0, 6.2))
    ax.text(0.5, 0.94, "From Bernoulli children to local arithmetic invariants", ha="center", fontsize=20, weight="
     bold")
    stages = [
        (0.04, r"Hurwitz children\n$b_\ell(z;a)$", BLUE),
        (0.23, "cyclotomic DFT\n$W_j(z;a)$", TEAL),
        (0.42, "Stieltjes germ\n$G_{j,m}(a)$", ORANGE),
        (0.61, r"scalar invariants\n$\mathcal{V}$ and $\mathcal{N}$", MAGENTA),
        (0.80, "positive energy\n$E(k)$ and resolvent", RED),
    ]
    for x0, text, color in stages:
        box(ax, (x0, 0.55), 0.15, 0.18, text, face=color + "20", edge=color, fontsize=12)
    for left, right in zip(stages[:-1], stages[1:], strict=True):
        arrow(ax, (left[0] + 0.15, 0.64), (right[0], 0.64), color=INK)
    box(ax, (0.20, 0.17), 0.22, 0.16, r"$L^{(m)}(1,\chi)$\ncharacter transform", face=TEAL + "18", edge=TEAL,
     fontsize=12)
    box(ax, (0.56, 0.17), 0.22, 0.16, "Gamma scaling\nand character decoherence", face=PURPLE + "18", edge=PURPLE,
     fontsize=12)
    arrow(ax, (0.50, 0.55), (0.31, 0.33), color=TEAL)
    arrow(ax, (0.88, 0.55), (0.67, 0.33), color=PURPLE)
    ax.text(0.5, 0.065, "Every map is exact at finite packet depth.  Asymptotic statements enter only in the energy
     and decoherence limits.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "01_theory_pipeline")

    # Stieltjes packet matrix for L=4.
    ell, m, r, a = 4, 1, 0, 1.0
    stieltjes_matrix = np.array(
        [[stieltjes_packet(ell, m, r, a, j, order) for order in range(6)] for j in range(ell)],
        dtype=np.complex128,
    )
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8))
    im0 = axes[0].imshow(np.real(stieltjes_matrix), cmap="coolwarm", aspect="auto")
    axes[0].set_title("Real parts of $G_{j,m}(1)$")
```

```
    axes[0].set_xlabel("Stieltjes order $m$")
    axes[0].set_ylabel("cyclotomic channel $j$")
    axes[0].set_xticks(range(6))
    axes[0].set_yticks(range(ell))
    fig.colorbar(im0, ax=axes[0], fraction=0.045)
    im1 = axes[1].imshow(np.imag(stieltjes_matrix), cmap="coolwarm", aspect="auto")
    axes[1].set_title("Imaginary parts of $G_{j,m}(1)$")
    axes[1].set_xlabel("Stieltjes order $m$")
    axes[1].set_ylabel("cyclotomic channel $j$")
    axes[1].set_xticks(range(6))
    axes[1].set_yticks(range(ell))
    fig.colorbar(im1, ax=axes[1], fraction=0.045)
    fig.suptitle("Cyclotomic Stieltjes packets are the local Taylor data of the Bernoulli-Lerch detail bank",
     fontsize=16.5, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "02_stieltjes_packet_matrix")

    # Two-channel derivative example.
    derivative_exact = -math.log(2) / math.sqrt(2)
    derivative_formula = detail_derivative_from_stieltjes(2, 1, 0, 1.0, 1, 1)
    checks.append(Check("two_channel_first_derivative", abs(derivative_formula - derivative_exact) < 1e-14, f"error
     ={abs(derivative_formula-derivative_exact):.3e}"))
    z_values = np.linspace(-0.35, 0.35, 241)
    detail_values = np.array([detail_channel(float(z), 2, 1, 0, 1.0, 1).real for z in z_values])
    linear = derivative_exact * z_values
    fig, ax = plt.subplots(figsize=(11.8, 4.8))
    ax.plot(z_values, detail_values, lw=2.1, label="exact $W_1(z;1)$")
    ax.plot(z_values, linear, lw=1.7, linestyle="--", label=r"$-(\log 2)z/\sqrt{2}$")
    ax.axhline(0, color=GRID, lw=1)
    ax.axvline(0, color=GRID, lw=1)
    ax.set_xlabel("complex order restricted to real $z$")
    ax.set_ylabel("detail value")
    ax.set_title(r"The first Bernoulli-Euler detail coefficient is $-(\log 2)/\sqrt{2}$")
    ax.legend()
    ax.grid(alpha=0.18)
    fig.tight_layout()
    save_figure(fig, "03_two_channel_germ")

    # Derivative checks through order 4 by direct high-precision differentiation
    # of the Hurwitz-zeta child formula.
    def detail_mp(z_value: mp.mpf | mp.mpc) -> mp.mpc:
        z_mp = mp.mpc(z_value)
        values = []
        for child in range(2):
            parameter = mp.mpf(1 + child) / 2
            if abs(z_mp) < mp.mpf("1e-70"):
                child_value = mp.mpf(1) / 2
            else:
                child_value = -z_mp * (mp.mpf(2) ** (z_mp - 1)) * mp.zeta(1 - z_mp, parameter)
            values.append(child_value)
        return (values[0] - values[1]) / mp.sqrt(2)

    derivative_rows: list[dict[str, object]] = []
    for nder in range(1, 5):
        predicted = detail_derivative_from_stieltjes(2, 1, 0, 1.0, 1, nder)
        numeric_mp = mp.diff(detail_mp, mp.mpf(0), nder)
        numeric = complex(numeric_mp)
        error = abs(predicted - numeric)
        derivative_rows.append({"order": nder, "predicted": predicted.real, "numeric": numeric.real, "error": error
     })
        checks.append(Check(f"stieltjes_derivative_order_{nder}", error < 2e-14, f"error={error:.3e}"))
    tables["stieltjes_derivatives"] = derivative_rows

    # Character transform and L derivatives for nonprincipal character mod 4.
    q = 4
    chi_values = [mp.mpc(0), mp.mpc(1), mp.mpc(0), mp.mpc(-1), mp.mpc(0)]
    character_rows: list[dict[str, object]] = []
    for order in range(4):
        lhs_mp = sum(chi_values[r] * mp.stieltjes(order, mp.mpf(r) / q) for r in range(1, q + 1))
        lhs = complex(lhs_mp)
        rhs_sum = mp.mpc(0)
```

```
        for h in range(order + 1):
            derivative = mp.diff(
                lambda ss: mp.dirichlet(ss, [0, 1, 0, -1]),
                1,
                h,
            )
            rhs_sum += math.comb(order, h) * (mp.log(q) ** (order - h)) * derivative
        rhs = ((-1) ** order) * q * rhs_sum
        error = abs(lhs - complex(rhs))
        character_rows.append({"order": order, "packet_sum_real": lhs.real, "formula_real": float(mp.re(rhs)), "error": error})
        checks.append(Check(f"character_derivative_transform_{order}", error < 5e-8, f"error={error:.3e}"))
    tables["character_derivatives"] = character_rows
    fig, ax = plt.subplots(figsize=(11.8, 4.7))
    orders = np.arange(4)
    lhs_values = [row["packet_sum_real"] for row in character_rows]
    rhs_values = [row["formula_real"] for row in character_rows]
    width = 0.36
    ax.bar(orders - width / 2, lhs_values, width, label="Stieltjes packet sum")
    ax.bar(orders + width / 2, rhs_values, width, label="$L$-derivative formula")
    ax.set_xticks(orders, [f"$m={value}$" for value in orders])
    ax.set_ylabel("real value")
    ax.set_title(r"Character packets recover the derivatives of $L(s,\chi_4)$ at $s=1$")
    ax.grid(axis="y", alpha=0.18)
    ax.legend()
    fig.tight_layout()
    save_figure(fig, "04_character_L_derivatives")

    # Detail variance identities on a deterministic complex example.
    children_complex = np.array([1.1 + 0.2j, -0.4 + 0.7j, 0.8 - 0.5j, 0.3 + 0.1j])
    transformed = transformed_details(children_complex)
    variance_direct = detail_variance(children_complex)
    variance_fourier = sum(transformed[j] * transformed[-j % len(transformed)] for j in range(1, len(transformed)))
    pairwise = sum((children_complex[l] - children_complex[m]) ** 2 for l in range(4) for m in range(l + 1, 4)) / 4
    checks.append(Check("detail_variance_fourier_identity", abs(variance_direct - variance_fourier) < 1e-13, f"error={abs(variance_direct-variance_fourier):.3e}"))
    checks.append(Check("detail_variance_pairwise_identity", abs(variance_direct - pairwise) < 1e-13, f"error={abs(variance_direct-pairwise):.3e}"))
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8))
    axes[0].scatter(children_complex.real, children_complex.imag, s=85)
    for index, value in enumerate(children_complex):
        axes[0].text(value.real + 0.03, value.imag + 0.03, f"$b_{index}$")
    axes[0].axhline(0, color=GRID, lw=1)
    axes[0].axvline(0, color=GRID, lw=1)
    axes[0].set_xlabel("real part")
    axes[0].set_ylabel("imaginary part")
    axes[0].set_title("child packets")
    axes[0].grid(alpha=0.15)
    axes[1].bar(np.arange(4), np.abs(transformed), color=COLORS[:4])
    axes[1].set_xticks(range(4), ["DC", "detail 1", "detail 2", "detail 3"])
    axes[1].set_ylabel("channel magnitude")
    axes[1].set_title("unitary child transform")
    axes[1].grid(axis="y", alpha=0.15)
    fig.suptitle("The holomorphic detail variance is the non-DC Fourier energy without conjugation", fontsize=16.2, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "05_detail_variance_identity")

    # Bernoulli-Euler variance.
    x_values = np.linspace(0, 1, 401)
    fig, ax = plt.subplots(figsize=(11.8, 4.9))
    for n_order, color in [(4, BLUE), (6, TEAL), (8, MAGENTA)]:
        function = sp.lambdify(sp.symbols("x"), sp.Rational(n_order**2, 8) * sp.euler(n_order - 1, sp.symbols("x")) ** 2, "numpy")
        ax.plot(x_values, function(x_values), lw=1.8, color=color, label=rf"$n={n_order}$")
    ax.set_xlabel("Bernoulli parameter $a$")
    ax.set_ylabel(r"$\mathcal{V}_2(n;a)$")
    ax.set_title("The two-channel refinement variance is an Euler-polynomial square")
    ax.legend()
    ax.grid(alpha=0.18)
    fig.tight_layout()
```

```
    save_figure(fig, "06_bernoulli_euler_variance")

    # Exact L=3 example.
    children_l3 = bernoulli_children_exact(4, sp.Rational(0), 3)
    variance_l3 = sp.simplify(sum(value**2 for value in children_l3) - sum(children_l3) ** 2 / 3)
    checks.append(Check("three_channel_variance_exact", variance_l3 == sp.Rational(32, 27), f"value={variance_l3}"))
    fig, ax = plt.subplots(figsize=(10.8, 4.8))
    positions = np.arange(3)
    values = [float(value) for value in children_l3]
    ax.bar(positions, values, color=[BLUE, TEAL, ORANGE])
    ax.axhline(float(sum(children_l3) / 3), color=INK, linestyle="--", label="parent mean")
    for position, value in zip(positions, children_l3, strict=True):
        ax.text(position, float(value) + (0.05 if float(value) >= 0 else -0.10), f"${sp.latex(value)}$", ha="center")
    ax.set_xticks(positions, ["child 0", "child 1", "child 2"])
    ax.set_ylabel("packet value")
    ax.set_title(r"Order four, three-channel Bernoulli refinement: $\mathcal{V}_3=32/27$")
    ax.legend()
    ax.grid(axis="y", alpha=0.16)
    fig.tight_layout()
    save_figure(fig, "07_three_channel_variance")

    # Primitive resultants for L=3 and 5.
    resultant_rows: list[dict[str, object]] = []
    for ell_value in (3, 4, 5):
        for n_order in range(2, 11):
            value = primitive_packet_resultant_exact(n_order, sp.Rational(0), ell_value)
            resultant_rows.append({"L": ell_value, "n": n_order, "value": str(value), "zero": bool(value == 0)})
            checks.append(Check(f"resultant_rational_L{ell_value}_n{n_order}", bool(value.is_rational), f"value={value}"))
    tables["resultants"] = resultant_rows
    fig, ax = plt.subplots(figsize=(12.2, 5.0))
    for ell_value, color in [(3, BLUE), (4, TEAL), (5, MAGENTA)]:
        subset = [row for row in resultant_rows if row["L"] == ell_value]
        n_orders = np.array([row["n"] for row in subset])
        magnitudes = np.array([math.log10(abs(float(sp.Rational(row["value"]))) + 1e-30) for row in subset])
        ax.plot(n_orders, magnitudes, marker="o", color=color, label=rf"$L={ell_value}$")
    ax.set_xlabel("integer order $n$")
    ax.set_ylabel(r"$\log_{10}|\mathcal{N}^{(L)}(n;0)|$")
    ax.set_title("Primitive packet resultants compress a full cyclotomic detail orbit into one scalar")
    ax.legend()
    ax.grid(alpha=0.18)
    fig.tight_layout()
    save_figure(fig, "08_primitive_resultants")

    # Energy computations.
    nmax = 500_000
    mu = mobius_sieve(nmax)
    a_energy = 2.0
    beta_energy = 2.0
    k_dense = np.arange(1, 161)
    energy_dense = packet_energy_values(k_dense, a_energy, beta_energy, nmax, 1, mu)
    differences = finite_differences(energy_dense, 8)
    sign_matrix = np.full((9, 120), np.nan)
    for order, row in enumerate(differences):
        usable = min(120, len(row))
        sign_matrix[order, :usable] = ((-1) ** order) * row[:usable]
    checks.append(Check("complete_monotonicity_numeric", bool(np.nanmin(sign_matrix) >= -2e-18), f"minimum={np.nanmin(sign_matrix):.3e}"))
    fig, ax = plt.subplots(figsize=(12.0, 5.1))
    image = ax.imshow(np.log10(np.maximum(sign_matrix, 1e-30)), cmap="viridis", aspect="auto")
    ax.set_yticks(range(9), [rf"$m={order}$" for order in range(9)])
    ax.set_xlabel("finite-difference location $k$")
    ax.set_title(r"Complete monotonicity: $(-1)^m\Delta^mE(k)\geq 0$")
    fig.colorbar(image, ax=ax, fraction=0.027, pad=0.02, label="log10 positive value")
    fig.tight_layout()
    save_figure(fig, "09_complete_monotonicity")

    # Hankel positivity.
    energy_hankel = packet_energy_values(np.arange(0, 25), a_energy, beta_energy, nmax, 1, mu)
    hankel_rows = []
```

```
    for size in range(2, 11):
        matrix = np.array([[energy_hankel[i + j] for j in range(size)] for i in range(size)])
        eigenvalues = np.linalg.eigvalsh(matrix)
        hankel_rows.append({"size": size, "minimum_eigenvalue": float(eigenvalues[0]), "maximum_eigenvalue": float(eigenvalues[-1])})
        checks.append(Check(f"hankel_positive_size_{size}", bool(eigenvalues[0] > -1e-14), f"min_eig={eigenvalues[0]:.3e}"))
    tables["hankel"] = hankel_rows
    fig, ax = plt.subplots(figsize=(11.8, 4.8))
    sizes = np.array([row["size"] for row in hankel_rows])
    minimums = np.array([max(row["minimum_eigenvalue"], 1e-30) for row in hankel_rows])
    maximums = np.array([row["maximum_eigenvalue"] for row in hankel_rows])
    ax.semilogy(sizes, minimums, marker="o", label="smallest eigenvalue")
    ax.semilogy(sizes, maximums, marker="o", label="largest eigenvalue")
    ax.set_xlabel("Hankel size")
    ax.set_ylabel("eigenvalue")
    ax.set_title("Finite packet-energy Hankel matrices are positive")
    ax.legend()
    ax.grid(alpha=0.18, which="both")
    fig.tight_layout()
    save_figure(fig, "10_hankel_positivity")

    # Resolvent over the complex plane.
    xs = np.linspace(-1.4, 1.35, 170)
    ys = np.linspace(-1.25, 1.25, 135)
    magnitude = np.zeros((len(ys), len(xs)))
    n_res = np.arange(2, 5001, dtype=np.float64)
    support_res = (mu[2:5001] != 0).astype(np.float64)
    weights_res = support_res * n_res ** (-2 * a_energy)
    x_res = (1.0 - n_res ** (-beta_energy)) ** 2
    for row, y in enumerate(ys):
        z_row = xs.astype(np.complex128) + 1j * y
        values = np.sum(weights_res[None, :] / (1.0 - z_row[:, None] * x_res[None, :]), axis=1)
        magnitude[row, :] = np.log10(np.maximum(np.abs(values), 1e-300))
        if abs(y) < 0.025:
            magnitude[row, xs >= 1.0] = np.nan
    fig, ax = plt.subplots(figsize=(11.8, 5.5))
    image = ax.imshow(magnitude, extent=[xs[0], xs[-1], ys[0], ys[-1]], origin="lower", cmap="magma", aspect="auto")
    ax.plot([1, xs[-1]], [0, 0], color=WHITE, lw=3.0, label=r"excluded cut $[1,\infty)$")
    ax.set_xlabel("real part of $z$")
    ax.set_ylabel("imaginary part of $z$")
    ax.set_title("The packet-energy resolvent is zero-free off its positive-real cut")
    ax.legend(loc="lower left", facecolor="#111111", edgecolor=WHITE, labelcolor=WHITE, framealpha=0.82)
    fig.colorbar(image, ax=ax, fraction=0.027, pad=0.02, label=r"$\log_{10}|\mathcal{E}(z)|$")
    fig.tight_layout()
    save_figure(fig, "11_zero_free_resolvent")

    # Direct versus zeta-ratio finite differences at modest k.
    ratio_rows: list[dict[str, object]] = []
    for k in range(1, 9):
        direct = packet_energy_values([k], a_energy, beta_energy, nmax, 1, mu)[0]
        exact = float(energy_from_zeta_ratio(k, a_energy, beta_energy, 1))
        error = abs(direct - exact)
        ratio_rows.append({"k": k, "direct": direct, "zeta_ratio": exact, "truncation_error": error})
        checks.append(Check(f"zeta_ratio_energy_{k}", error < 2e-14, f"error={error:.3e}"))
    tables["zeta_ratio"] = ratio_rows
    fig, ax = plt.subplots(figsize=(11.8, 4.8))
    kvals = np.array([row["k"] for row in ratio_rows])
    direct_values = np.array([row["direct"] for row in ratio_rows])
    ratio_values = np.array([row["zeta_ratio"] for row in ratio_rows])
    ax.semilogy(kvals, direct_values, marker="o", lw=2, label="direct squarefree sum")
    ax.semilogy(kvals, ratio_values, marker="x", lw=1.5, linestyle="--", label=r"finite difference of $\zeta(s)/\zeta(2s)$")
    ax.set_xlabel("difference order $k$")
    ax.set_ylabel("energy")
    ax.set_title("One positive packet energy has two exact arithmetic coordinate systems")
    ax.legend()
    ax.grid(alpha=0.18, which="both")
    fig.tight_layout()
    save_figure(fig, "12_zeta_ratio_coordinates")

```

```
    # Energy asymptotic.
    k_asym = np.array([100, 300, 1000, 3000, 10000, 30000, 100000])
    energy_asym_values = packet_energy_values(k_asym, a_energy, beta_energy, nmax, 1, mu)
    alpha = (2 * a_energy - 1) / beta_energy
    constant = energy_asymptotic_constant(a_energy, beta_energy, 1)
    normalized = energy_asym_values * k_asym**alpha
    energy_rows = []
    for k, value, norm in zip(k_asym, energy_asym_values, normalized, strict=True):
        energy_rows.append({"k": int(k), "energy": value, "scaled": norm, "relative_error": abs(norm / constant - 1)
     })
    tables["energy_asymptotic"] = energy_rows
    checks.append(Check("energy_asymptotic_final", abs(normalized[-1] / constant - 1) < 0.01, f"relative_error={abs(
     normalized[-1]/constant-1):.3e}"))
    fig, ax = plt.subplots(figsize=(11.8, 4.8))
    ax.semilogx(k_asym, normalized, marker="o", lw=2, label=r"$k^{3/2}E_k$")
    ax.axhline(constant, color=RED, linestyle="--", label="proved asymptotic constant")
    ax.set_xlabel("$k$")
    ax.set_ylabel("scaled energy")
    ax.set_title("The principal packet energy has the exact $k^{-3/2}$ scale")
    ax.legend()
    ax.grid(alpha=0.18)
    fig.tight_layout()
    save_figure(fig, "13_energy_asymptotic")

    # Gamma scaling distribution.
    gamma_rows: list[dict[str, object]] = []
    fig, ax = plt.subplots(figsize=(11.8, 5.0))
    u_grid = np.linspace(0, 8, 500)
    gamma_density = u_grid ** (alpha - 1) * np.exp(-u_grid) / gamma_fn(alpha)
    ax.plot(u_grid, gamma_density, color=INK, lw=2.4, label=rf"Gamma$(\alpha={alpha:.1f},1)$")
    for index, k in enumerate([300, 3000, 30000, 100000]):
        values, weights = gamma_weighted_sample(k, a_energy, beta_energy, nmax, 1, mu)
        bins = np.linspace(0, 8, 70)
        hist, edges = np.histogram(values, bins=bins, weights=weights, density=True)
        centers = (edges[:-1] + edges[1:]) / 2
        ax.plot(centers, hist, lw=1.35, color=COLORS[index], label=rf"$k={k}$")
        moments = weighted_moments(values, weights, 2)
        quantiles = weighted_quantile(values, weights, [0.1, 0.5, 0.9])
        ks = ks_distance_to_gamma(values, weights, alpha)
        gamma_rows.append({
            "k": k,
            "mean": moments[0],
            "variance": moments[1] - moments[0] ** 2,
            "q10": quantiles[0],
            "q50": quantiles[1],
            "q90": quantiles[2],
            "ks": ks,
        })
    tables["gamma_scaling"] = gamma_rows
    checks.append(Check("gamma_mean_final", abs(gamma_rows[-1]["mean"] - alpha) < 0.03, f"mean={gamma_rows[-1]['mean
     ']:.6f}"))
    checks.append(Check("gamma_variance_final", abs(gamma_rows[-1]["variance"] - alpha) < 0.06, f"variance={
     gamma_rows[-1]['variance']:.6f}"))
    ax.set_xlim(0, 8)
    ax.set_xlabel(r"$U_k=2k/n^2$")
    ax.set_ylabel("probability density")
    ax.set_title("The active squarefree packet window converges to a Gamma law")
    ax.legend(ncol=2)
    ax.grid(alpha=0.16)
    fig.tight_layout()
    save_figure(fig, "14_gamma_scaling")

    fig, ax = plt.subplots(figsize=(11.8, 4.8))
    gamma_k = np.array([row["k"] for row in gamma_rows])
    ks_values = np.array([row["ks"] for row in gamma_rows])
    ax.loglog(gamma_k, ks_values, marker="o", lw=2)
    ax.set_xlabel("$k$")
    ax.set_ylabel("weighted Kolmogorov distance")
    ax.set_title("Finite packet distributions approach the limiting Gamma profile")
    ax.grid(alpha=0.18, which="both")
    fig.tight_layout()
```

```
    save_figure(fig, "15_gamma_convergence")

    # Character Gram and decoherence.
    characters = characters_mod5(nmax)
    gram_small = character_gram_matrix(100, a_energy, beta_energy, nmax, mu, characters)
    gram_large = character_gram_matrix(100000, a_energy, beta_energy, nmax, mu, characters)
    corr_small = normalized_correlation(gram_small)
    corr_large = normalized_correlation(gram_large)
    fig = plt.figure(figsize=(12.4, 4.9))
    grid = fig.add_gridspec(1, 3, width_ratios=[1.0, 1.0, 0.055], wspace=0.34)
    axes = [fig.add_subplot(grid[0, 0]), fig.add_subplot(grid[0, 1])]
    cax = fig.add_subplot(grid[0, 2])
    for ax, matrix, title in [(axes[0], corr_small, "$k=100$"), (axes[1], corr_large, "$k=100000$")]:
        image = ax.imshow(np.abs(matrix), cmap="viridis", vmin=0, vmax=1)
        ax.set_xticks(range(4), [r"$\chi_0$", r"$\chi_1$", r"$\chi_2$", r"$\chi_3$"])
        ax.set_yticks(range(4), [r"$\chi_0$", r"$\chi_1$", r"$\chi_2$", r"$\chi_3$"])
        ax.set_title(title)
        for row in range(4):
            for col in range(4):
                value = abs(matrix[row, col])
                ax.text(col, row, f"{value:.3f}", ha="center", va="center", color=WHITE if value > 0.55 else INK,
     fontsize=8.7)
    fig.colorbar(image, cax=cax, label="normalized magnitude")
    fig.suptitle("Distinct Dirichlet-character packets become asymptotically orthogonal", fontsize=16.5, weight="
     bold")
    fig.subplots_adjust(left=0.07, right=0.96, bottom=0.13, top=0.82)
    save_figure(fig, "16_character_gram")

    k_char = np.array([100, 300, 1000, 3000, 10000, 30000, 100000])
    char_rows = []
    max_offdiag = []
    for k in k_char:
        gram = character_gram_matrix(int(k), a_energy, beta_energy, nmax, mu, characters)
        corr = normalized_correlation(gram)
        mask = ~np.eye(4, dtype=bool)
        maximum = float(np.max(np.abs(corr[mask])))
        max_offdiag.append(maximum)
        char_rows.append({"k": int(k), "max_offdiag": maximum})
    tables["character_decoherence"] = char_rows
    fig, ax = plt.subplots(figsize=(11.8, 4.8))
    ax.loglog(k_char, max_offdiag, marker="o", lw=2, label="largest measured off-diagonal correlation")
    reference_constant = max_offdiag[0] * (k_char[0] ** 0.25)
    ax.loglog(k_char, reference_constant * k_char ** (-0.25), linestyle="--", color=RED, label=r"reference slope $k
     ^{-1/4}$")
    ax.set_xlabel("$k$")
    ax.set_ylabel("maximum normalized correlation")
    ax.set_title("Character decoherence removes cross-sector interference unconditionally")
    ax.legend()
    ax.grid(alpha=0.18, which="both")
    fig.tight_layout()
    save_figure(fig, "17_character_decoherence")
    checks.append(Check("character_decoherence_decreases", max_offdiag[-1] < max_offdiag[0], f"first={max_offdiag
     [0]:.4f}, last={max_offdiag[-1]:.4f}"))

    # Quantum/logarithmic pipeline.
    fig, ax = diagram_axes((13.0, 6.2))
    ax.text(0.5, 0.94, "Positive packet energy and signed coherence in a logarithmic spectrum", ha="center",
     fontsize=20, weight="bold")
    box(ax, (0.04, 0.58), 0.17, 0.17, "number basis\n" + r"$|n\rangle$", face=BLUE + "20", edge=BLUE, fontsize=13)
    box(ax, (0.29, 0.58), 0.18, 0.17, "log Hamiltonian\n" + r"$H|n\rangle=\log n|n\rangle$", face=TEAL + "20", edge=
     TEAL, fontsize=12)
    box(ax, (0.55, 0.58), 0.18, 0.17, "scale filter\n" + r"$e^{-kK_\beta}$", face=ORANGE + "20", edge=ORANGE,
     fontsize=13)
    box(ax, (0.81, 0.58), 0.15, 0.17, "character sign\n" + r"$\mathcal{M}_\chi$", face=MAGENTA + "20", edge=MAGENTA,
      fontsize=12)
    arrow(ax, (0.21, 0.665), (0.29, 0.665))
    arrow(ax, (0.47, 0.665), (0.55, 0.665))
    arrow(ax, (0.73, 0.665), (0.81, 0.665))
    box(ax, (0.13, 0.20), 0.30, 0.18, "diagonal power\n" + r"$E(k)=\langle\psi|\mathcal{M}^2e^{-2kK}|\psi\rangle$",
     face=TEAL + "14", edge=TEAL, fontsize=12)
    box(ax, (0.57, 0.20), 0.30, 0.18, "signed amplitude\n" + r"$F_k(t)=\langle\phi|\mathcal{M}e^{-kK}e^{-itH}|\phi\
```

```
    rangle$", face=MAGENTA + "14", edge=MAGENTA, fontsize=12)
    arrow(ax, (0.63, 0.58), (0.28, 0.38), color=TEAL)
    arrow(ax, (0.88, 0.58), (0.72, 0.38), color=MAGENTA)
    ax.text(0.5, 0.075, "The positive energy is completely understood. The remaining zeta difficulty is the internal
      coherence of the principal Mobius sector.", ha="center", fontsize=11.3, color=MUTED)
    save_figure(fig, "18_quantum_packet_pipeline")

    # Status map.
    fig, ax = diagram_axes((13.0, 6.5))
    ax.text(0.5, 0.95, "What has been proved and what remains open", ha="center", fontsize=21, weight="bold")
    proved = [
        "Stieltjes germ formulas",
        r"$L^{(m)}(1,\chi)$ transforms",
        "detail variance",
        "primitive resultants",
        "complete monotonicity",
        "Hankel positivity",
        "zero-free resolvent",
        "Gamma scaling",
        "character decoherence",
    ]
    open_items = [
        "principal Mobius coherence",
        "uniform GRH coherence",
        "packet canonical systems",
        "global zero-location theorem",
    ]
    ax.add_patch(FancyBboxPatch((0.05, 0.13), 0.52, 0.72, boxstyle="round,pad=.015,rounding_size=.02", facecolor=
     TEAL + "14", edgecolor=TEAL, lw=2))
    ax.add_patch(FancyBboxPatch((0.62, 0.13), 0.33, 0.72, boxstyle="round,pad=.015,rounding_size=.02", facecolor=
     ORANGE + "14", edgecolor=ORANGE, lw=2))
    ax.text(0.31, 0.80, "proved in this paper", ha="center", fontsize=15, weight="bold", color=TEAL)
    ax.text(0.785, 0.80, "open", ha="center", fontsize=15, weight="bold", color=ORANGE)
    y = 0.72
    for item in proved:
        ax.text(0.10, y, f"PASS  {item}", ha="left", va="center", fontsize=11.3)
        y -= 0.065
    y = 0.68
    for item in open_items:
        ax.text(0.67, y, f"-  {item}", ha="left", va="center", fontsize=11.3)
        y -= 0.12
    ax.text(0.5, 0.055, "The positive and cross-character layers are now controlled. The principal signed sector
     still carries the RH-scale obstruction.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "19_status_map")

    # Family map.
    fig, ax = diagram_axes((13.0, 6.4))
    ax.text(0.5, 0.95, "The second special-function family generated by the calculus", ha="center", fontsize=20,
     weight="bold")
    nodes = [
        (0.08, 0.68, 0.20, 0.14, "Bernoulli-Hurwitz\nchildren", BLUE),
        (0.40, 0.68, 0.20, 0.14, "cyclotomic Stieltjes\npackets", TEAL),
        (0.72, 0.68, 0.20, 0.14, "packet resultants\nand variance", ORANGE),
        (0.08, 0.30, 0.20, 0.14, "squarefree zeta-ratio\nfinite differences", MAGENTA),
        (0.40, 0.30, 0.20, 0.14, "Hausdorff energy\nresolvent", RED),
        (0.72, 0.30, 0.20, 0.14, "Gamma scale law\nand character Gram", PURPLE),
    ]
    for x0, y0, width, height, text, color in nodes:
        box(ax, (x0, y0), width, height, text, face=color + "18", edge=color, fontsize=12)
    arrow(ax, (0.28, 0.75), (0.40, 0.75))
    arrow(ax, (0.60, 0.75), (0.72, 0.75))
    arrow(ax, (0.18, 0.68), (0.18, 0.44), color=MAGENTA)
    arrow(ax, (0.28, 0.37), (0.40, 0.37), color=RED)
    arrow(ax, (0.60, 0.37), (0.72, 0.37), color=PURPLE)
    arrow(ax, (0.82, 0.68), (0.82, 0.44), color=PURPLE)
    ax.text(0.5, 0.08, "The analytic and arithmetic branches meet through finite transforms, positive moments, and
     asymptotic packet orthogonality.", ha="center", fontsize=11.4, color=MUTED)
    save_figure(fig, "20_function_family_map")

    # Save tables.
    for name, rows in tables.items():
```

```
        if not rows:
            continue
        path = TABLES / f"{name}.csv"
        with path.open("w", newline="", encoding="utf-8") as handle:
            writer = csv.DictWriter(handle, fieldnames=list(rows[0].keys()))
            writer.writeheader()
            writer.writerows(rows)

    passed = sum(check.passed for check in checks)
    summary = {
        "title": "Projector-Generated Special Functions II",
        "checks": len(checks),
        "passed": passed,
        "failed": len(checks) - passed,
        "figures": 20,
        "stieltjes_first_derivative": derivative_formula.real,
        "three_channel_variance": str(variance_13),
        "energy_asymptotic_constant": constant,
        "energy_final_relative_error": energy_rows[-1]["relative_error"],
        "gamma_final_mean": gamma_rows[-1]["mean"],
        "gamma_final_variance": gamma_rows[-1]["variance"],
        "gamma_final_ks": gamma_rows[-1]["ks"],
        "character_correlation_first": max_offdiag[0],
        "character_correlation_last": max_offdiag[-1],
    }
    (RESULTS / "results.json").write_text(json.dumps(summary, indent=2, default=lambda value: value.item() if
     hasattr(value, "item") else str(value)), encoding="utf-8")
    with (RESULTS / "checks.csv").open("w", newline="", encoding="utf-8") as handle:
        writer = csv.DictWriter(handle, fieldnames=["name", "passed", "detail"])
        writer.writeheader()
        for check in checks:
            writer.writerow({"name": check.name, "passed": check.passed, "detail": check.detail})

    if passed != len(checks):
        failures = [check for check in checks if not check.passed]
        raise RuntimeError(f"{len(failures)} checks failed: {failures}")
    return summary


if __name__ == "__main__":
    print(json.dumps(generate_all(), indent=2, default=lambda value: value.item() if hasattr(value, "item") else str
     (value)))
```

Chapter E

Reproduction

From the source directory, run

```
python build_all.py
```

The command regenerates the figures and tables, runs the tests, and compiles the paper three times.

Bibliography

[1] Komorebi P., *Projector-Generated Special Functions I: Polyphase Differential Lifts, Bernoulli–Lerch Wavelets, Cyclotomic Detail Functions, and Möbius Coherence*, 2026.

[2] Komorebi P., *Bernoulli Transfer-Block Reformulation of Zeta*, proof-development note, 2026.

[3] NIST Digital Library of Mathematical Functions, Chapter 24, Bernoulli and Euler Polynomials.

[4] NIST Digital Library of Mathematical Functions, Chapter 25, Zeta and Related Functions.

[5] L. Báez-Duarte, A sequential Riesz-like criterion for the Riemann hypothesis, *International Journal of Mathematics and Mathematical Sciences*, 2005.

[6] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Oxford University Press.

[7] S. Wei, Y. Zhai, Q. Lu et al., The Riemann Hypothesis manifested in dynamical quantum phase transitions, *Nature Communications*, accepted 18 June 2026, DOI 10.1038/s41467-026-74935-8.

Projector-Generated Special Functions III

Squarefree Stieltjes–Jacobi Functions, Laguerre Packet Universality, Hankel Determinants, and Moment-Exact Reduction

Komorebi P.

Abstract

This paper develops a third family of special functions generated by the projector-resolved calculus. The starting point is the positive squarefree packet energy introduced in the preceding paper. Its values are reinterpreted as the moments of a discrete measure on the unit interval. That measure produces a Stieltjes resolvent, an infinite Jacobi operator, a continued fraction, positive rational approximants, Hankel determinants, Christoffel–Darboux kernels, and Gaussian quadrature rules. These are not independent definitions. They are exact coordinate realizations of one positive packet seed.

The moment sequence is proved to be strictly totally positive in the generalized Hankel sense. Its resolvent is analytic and zero-free on the slit plane $\mathbb{C} \setminus [1, \infty)$. Lanczos reduction produces a finite Jacobi model that preserves the first $2m$ moments exactly after m steps. Every finite model has positive weights, real nodes in the packet spectrum, and a rational resolvent with poles confined to the physical cut.

A second layer appears after exponential tilting by the packet depth k. If

$$\alpha = \frac{2a-1}{\beta},$$

then the scaled endpoint coordinate $u = k(1-x)$ converges to the Gamma law with shape α. Fixed-degree orthogonal packet polynomials converge to monic generalized Laguerre polynomials. The Jacobi coefficients, Christoffel–Darboux kernel, Gaussian quadrature nodes, and Gaussian quadrature weights converge to their Gamma–Laguerre counterparts. The Gamma–Laguerre operator is therefore the universal local fixed point of the squarefree packet family.

The paper also proves a complete leading asymptotic for shifted Hankel determinants. For the principal parameters $a = \beta = 2$, the energy resolvent has a square-root cusp at the endpoint of its cut with an explicit constant. The positive theory is kept separate from the unresolved signed Möbius coherence problem. The latter remains the arithmetic obstruction carrying the Riemann hypothesis.

All figures were produced by the accompanying Python program. The release contains 22 vector figures, matching PNG files, exact tables, 34 built-in checks, 19 independent tests, and the complete Python source in the accompanying release.

Status and claim boundary

The paper proves the positive moment representation, strict generalized Hankel positivity, the zero-free Stieltjes resolvent theorem, the Jacobi continued fraction, moment-exact finite reduction, Gamma scaling, fixed-degree Laguerre convergence, recurrence-coefficient convergence, Christoffel–Darboux convergence, Gaussian quadrature convergence, and the shifted Hankel determinant asymptotic under the hypotheses stated below.

Classical inputs include squarefree counting, partial summation, the compact moment problem, orthogonal polynomial recurrence theory, Gaussian quadrature, Lanczos tridiagonalization, and generalized Laguerre polynomials. The construction of the packet measure and its arithmetic energy law comes from the second paper of this series. The logarithmic number-basis Hamiltonian discussed near the end is compatible with the engineered quantum setting developed by Wei and collaborators [10].

The positive packet theory does not prove RH. It controls the diagonal squarefree energy and supplies canonical finite models. RH remains equivalent to a subpolynomial bound for a signed Möbius coherence ratio. No positivity statement proved here implies that missing signed estimate.

Contents

Status and claim boundary 1

I The squarefree packet measure 5

1 One positive seed and its transforms 5

2 Positive contraction realization 7

II Total positivity and the Stieltjes-Jacobi function 9

3 Strict generalized Hankel positivity 9

4 The zero-free Stieltjes-Jacobi function 10

III Finite positive models 13

5 Gauss packets and Padé-type resolvents 13

IV Gamma tilting and Laguerre universality 17

6 Endpoint scaling of the packet measure 17

7 Laguerre packet polynomials 19

V Determinants, kernels, and quadrature limits 22

8 Shifted Hankel determinant asymptotics 22

9 Christoffel–Darboux universality 24

10 Gauss–Laguerre packet quadrature **25**

VI The spectral edge and the universal family **26**

11 A square-root cusp in the principal resolvent **26**

12 Universality classes **26**

VII Quantum coordinates and the arithmetic boundary **29**

13 Number-basis realization **29**

14 The unresolved signed coherence **30**

VIII Worked calculations and computational record **31**

15 Worked calculation of a four-node reduction **31**

16 Numerical tables and verification **32**

A Formula registry **33**

B Implementation map **34**

C Reproduction **35**

List of Figures

1.1 The theory generated by one positive packet seed. The measure gives moments, a resolvent, a Jacobi operator, finite quadrature models, tilted endpoint laws, and a Laguerre fixed point. 6

1.2 The discrete packet measure on $[0, 1)$. Squarefree integers produce positive atoms whose locations accumulate at the spectral endpoint $x = 1$. Their weights decay as n^{-2a}. 7

2.1 The operator realization of the moments. The arithmetic weights form the seed vector. The diagonal contraction supplies the packet depth. Their matrix element is exactly the energy. 8

2.2 Energy decay in the principal case. The computed values approach the constant $3/(2^{5/2}\pi^{3/2})$ multiplying $k^{-3/2}$. 9

3.1 Selected generalized Hankel minors. The heatmap records base-ten logarithms of exact high-precision minors. The proof covers every increasing pair of exponent sets. 10

4.1 The spectral endpoint in the principal case. The resolvent remains finite at $z = 1$ but approaches it through a square-root cusp with a computable constant. 12

4.2 The global packet Jacobi matrix. Lanczos tridiagonalization turns the infinite diagonal number-basis operator into one canonical nearest-neighbor transfer system. 12

4.3 The Jacobi continued fraction. Every coefficient is determined by the positive packet measure. Finite truncations preserve positivity and produce rational reduced models. . . . 13

5.1 Positive rational approximants to the resolvent. The finite Jacobi models reproduce low-order moments exactly and converge rapidly on the tested compact real interval below the cut. 14

5.2 Poles and weights of a ten-node packet model. All nonzero poles lie on the physical cut and every spectral weight is positive. A zero node represents the endpoint atom and produces no finite pole. 15

5.3 Exact moment matching. An m-node model agrees through degree $2m - 1$. The first unmatched moments expose the boundary of the finite contract. 16

5.4 The reduction map. The finite Jacobi operator is not an arbitrary fit. It is the unique Gaussian packet model attached to the Krylov basis. 16

6.1 Convergence of the tilted packet law. The discrete packet histograms approach the Gamma density. Arithmetic oscillations remain visible at finite depth and decay under scaling. . . 18

7.1 Polynomial convergence in degrees one through four. The packet polynomials arise from the exact discrete arithmetic measure. Their fixed-degree scaling limit is the classical monic Laguerre family. 20

7.2 Convergence of the packet recurrence coefficients. The finite arithmetic Jacobi systems approach one universal Gamma–Laguerre transfer operator determined only by $\alpha = (2a-1)/\beta$. 21

8.1 Scaled shifted Hankel determinants. Four determinant orders converge to the Gamma–Laguerre constants predicted by the theorem. 23

9.1 Christoffel–Darboux kernel convergence. The packet kernel, the Gamma–Laguerre kernel, and their difference are shown at a large but finite tilt depth. 24

10.1 Gauss quadrature convergence. Node and weight errors contract toward zero as the arithmetic packet tilt approaches its Gamma fixed point. 25

12.1 The parameter map. Curves of constant α define universality classes. The principal RH energy occupies the class $\alpha = 3/2$. 27

12.2 Logical dependency of the results. The positive measure supports two branches. One leads to total positivity and resolvent reduction. The other leads to Gamma scaling and Laguerre universality. 28

12.3 The third generated function family. The resolvent, Jacobi function, rational models, Hankel determinants, Laguerre fixed point, and quadrature rules are exact transforms of one packet measure. 28

14.1 The exact boundary of the paper. The positive side is complete within the stated setting. The signed principal coherence remains the open arithmetic problem. 31

Part I

The squarefree packet measure

Chapter 1

One positive seed and its transforms

Fix parameters

$$a > \frac{1}{2}, \qquad \beta > 0, \qquad q \in \mathbb{N}.$$

Let χ be a Dirichlet character of conductor q. Since $|\chi(n)|^2$ is the indicator of $(n, q) = 1$, define

$$w_n = \frac{\mu(n)^2\,|\chi(n)|^2}{n^{2a}}$$

and

$$x_n = \left(1 - \frac{1}{n^\beta}\right)^2.$$

Every w_n is nonnegative and every node belongs to $[0, 1)$. The packet measure is

$$\boxed{\nu_{a,\beta,q} = \sum_{n\geq 1} w_n\,\delta_{x_n}.}$$

Its moments are

$$\boxed{E_{a,\beta,q}(k) = \int_0^1 x^k \, \mathrm{d}\nu(x) = \sum_{n\geq 1} \frac{\mu(n)^2 \, |\chi(n)|^2}{n^{2a}} \left(1 - \frac{1}{n^\beta}\right)^{2k}.}$$

The condition $a > 1/2$ makes the total mass finite.

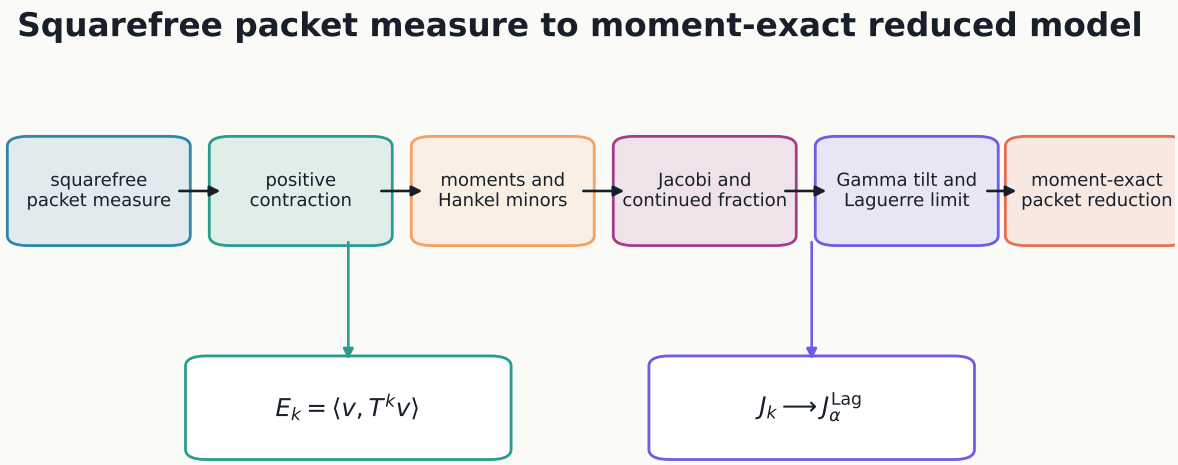


Figure 1.1: The theory generated by one positive packet seed. The measure gives moments, a resolvent, a Jacobi operator, finite quadrature models, tilted endpoint laws, and a Laguerre fixed point.

Definition 1.1 (Squarefree packet moment function)**.** The map

$$(k, a, \beta, q) \longmapsto E_{a,\beta,q}(k)$$

is the squarefree packet moment function. For fixed parameters it is a sequence in k. For fixed k it is a positive special value built from the squarefree Dirichlet series.

Proposition 1.2 (Exact finite-difference zeta-ratio coordinate)**.** *Let*

$$D_q(s) = \sum_{\substack{n\geq 1 \\ (n,q)=1}} \frac{\mu(n)^2}{n^s}.$$

Then

$$D_q(s) = \frac{\zeta(s)}{\zeta(2s)} \prod_{p\mid q} (1 + p^{-s})^{-1}$$

and

$$\boxed{E_{a,\beta,q}(k) = \sum_{j=0}^{2k} (-1)^j \binom{2k}{j} D_q(2a + \beta j).}$$

Proof. The Euler factor of $\mu(n)^2$ at a prime $p \nmid q$ is $1 + p^{-s}$. At a prime dividing q, the nontrivial term is removed. This gives the formula for D_q. Expanding $(1 - n^{-\beta})^{2k}$ by the binomial theorem and interchanging the finite sum with the absolutely convergent Dirichlet series gives the second identity. □

Worked example 1.3 (Principal values)**.** For $a = \beta = 2$ and $q = 1$,

$$E_k = \sum_{j=0}^{2k} (-1)^j \binom{2k}{j} \frac{\zeta(4+2j)}{\zeta(8+4j)}.$$

The Python program checks this formula against the direct squarefree sum for $0 \le k \le 5$.

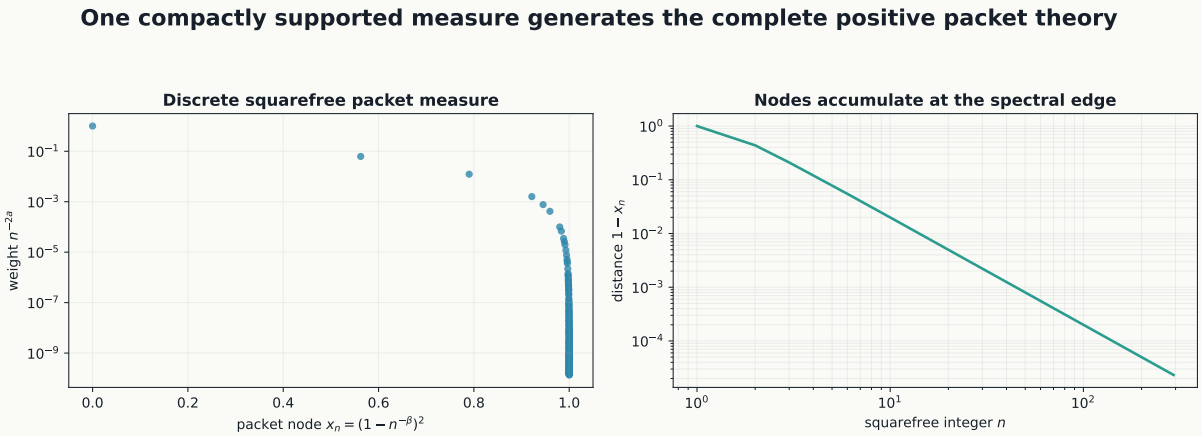


Figure 1.2: The discrete packet measure on $[0, 1)$. Squarefree integers produce positive atoms whose locations accumulate at the spectral endpoint $x = 1$. Their weights decay as n^{-2a}.

Chapter 2

Positive contraction realization

Let

$$\mathcal{H}_q = \ell^2\{n \ge 1 : (n, q) = 1,\ \mu(n)^2 = 1\}.$$

Define the diagonal contraction

$$T\,|n\rangle = x_n\,|n\rangle$$

and the seed vector

$$|v\rangle = \sum_n \sqrt{w_n}\,|n\rangle\,.$$

Then $0 \le T < I$ and

$$\boxed{E_k = \langle v|\,T^k\,|v\rangle\,.}$$

The complete positive theory begins with this one matrix element.

Theorem 2.1 (Hausdorff moment structure)**.** *The sequence $(E_k)_{k\ge0}$ is a Hausdorff moment sequence. For every $m, k \ge 0$,*

$$\boxed{(-1)^m \Delta^m E_k = \sum_n w_n x_n^k (1 - x_n)^m \ge 0.}$$

Proof. The identity follows from

$$\Delta^m x^k = x^k (x - 1)^m.$$

Multiplication by $(-1)^m$ gives $x^k(1 - x)^m$. Integration against the positive measure proves the claim. □

Corollary 2.2. *The sequence is decreasing, convex in the alternating finite-difference sense, and log-convex:*

$$E_k^2 \le E_{k-1}E_{k+1}.$$

Proof. Only log-convexity needs comment. Apply Cauchy–Schwarz to the functions $x^{(k-1)/2}$ and $x^{(k+1)/2}$ in $L^2(\nu)$. □

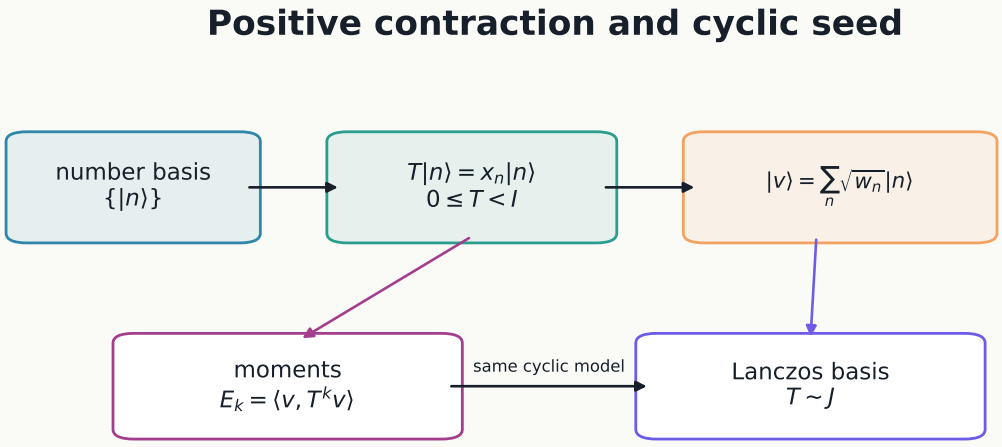


Figure 2.1: The operator realization of the moments. The arithmetic weights form the seed vector. The diagonal contraction supplies the packet depth. Their matrix element is exactly the energy.

Table 2.1: Principal packet energy and its leading asymptotic.

k	E_k	$Ak^{-3/2}$	ratio
0	1.0779281367418554e+00	9.5240453901361460e-02	-
1	4.7804610342673359e-02	9.5240453901361460e-02	0.5019359777
2	3.0200677449579039e-02	3.3672585398468735e-02	0.8968921481
4	1.3470094301315242e-02	1.1905056737670183e-02	1.1314598996
8	4.3883096314839380e-03	4.2090731748085919e-03	1.0425833549
16	1.4882088608347774e-03	1.4881320922087728e-03	1.0000515872
100	9.3971867489992978e-05	9.5240453901361460e-05	0.9866801726
1000	3.0284088244824538e-06	3.0117675971657171e-06	1.0055254022
10000	9.5394185462559236e-08	9.5240453901361455e-08	1.0016141414
100000	3.0070532365881507e-09	3.0117675971657168e-09	0.9984346865

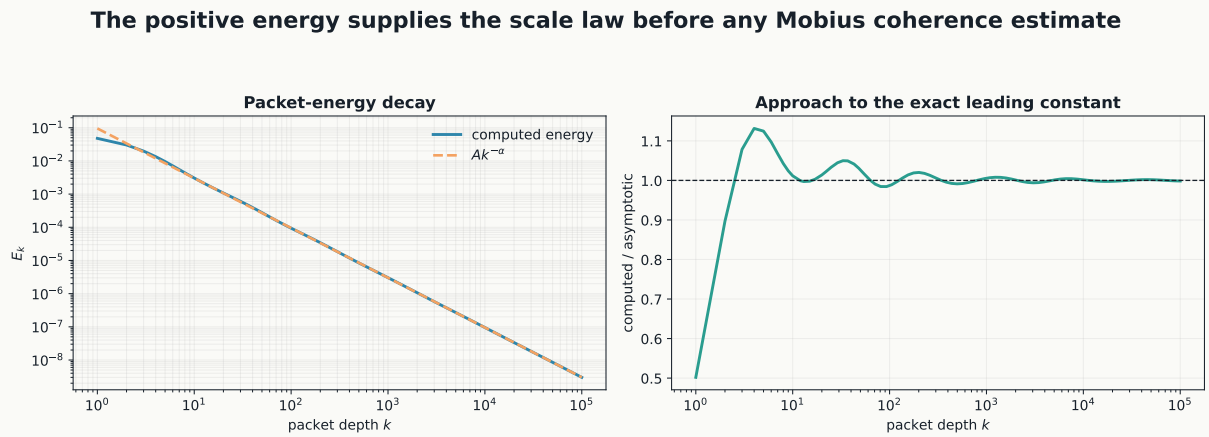


Figure 2.2: Energy decay in the principal case. The computed values approach the constant $3/(2^{5/2}\pi^{3/2})$ multiplying $k^{-3/2}$.

Part II

Total positivity and the Stieltjes-Jacobi function

Chapter 3

Strict generalized Hankel positivity

For increasing integer sequences

$$0 \le r_0 < \cdots < r_d, \qquad 0 \le s_0 < \cdots < s_d,$$

define the generalized Hankel minor

$$\Delta_{\mathbf{r},\mathbf{s}} = \det\left[E_{r_i+s_j}\right]_{i,j=0}^{d}.$$

Theorem 3.1 (Strict total positivity)**.** *Every generalized Hankel minor is strictly positive:*

$$\boxed{\Delta_{\mathbf{r},\mathbf{s}} > 0.}$$

Proof. Use the discrete Andréief identity to write the determinant as a sum over distinct support nodes:

$$\Delta_{\mathbf{r},\mathbf{s}} = \frac{1}{(d+1)!} \sum_{n_0,\dots,n_d} \left(\prod_{j=0}^{d} w_{n_j}\right) \det[x_{n_j}^{r_i}] \det[x_{n_j}^{s_i}].$$

Terms with repeated nodes vanish. Order the remaining nodes so that

$$0 < x_{n_0} < \cdots < x_{n_d} < 1.$$

A generalized Vandermonde determinant with increasing nonnegative exponents is positive on strictly increasing positive nodes. Both determinants therefore have the same positive sign. The support contains infinitely many positive nodes with positive weights, so at least one term is strictly positive. □

Corollary 3.2 (Strict Hankel positivity)**.** *For every $d \geq 0$, the matrix*

$$H_d = (E_{i+j})_{i,j=0}^{d}$$

is positive definite.

Proof. For $c \neq 0$,

$$c^* H_d c = \int_0^1 \left| \sum_{j=0}^{d} c_j x^j \right|^2 \mathrm{d}\nu(x) > 0.$$

A nonzero polynomial cannot vanish on the infinite support of ν. □

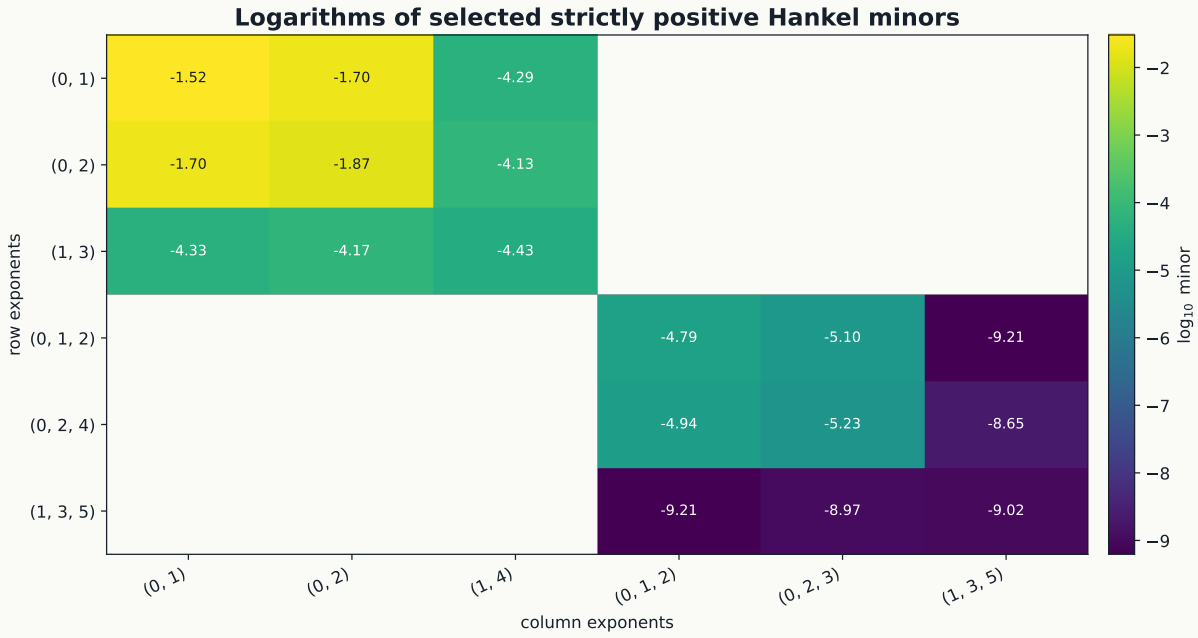


Figure 3.1: Selected generalized Hankel minors. The heatmap records base-ten logarithms of exact high-precision minors. The proof covers every increasing pair of exponent sets.

Chapter 4

The zero-free Stieltjes-Jacobi function

Define the packet-energy resolvent

$$\boxed{\mathcal{E}_{a,\beta,q}(z) = \sum_{k=0}^{\infty} E_k z^k = \int_0^1 \frac{\mathrm{d}\nu(x)}{1 - zx}.}$$

The power series converges for $|z| < 1$. The integral continues it to the slit plane.

Theorem 4.1 (Zero-free slit-plane continuation)**.** *The function $\mathcal{E}_{a,\beta,q}$ is analytic and zero-free on*

$$\boxed{\mathbb{C} \setminus [1, \infty).}$$

Proof. For real $z < 1$, every integrand $(1 - zx)^{-1}$ is positive, hence $\mathcal{E}(z) > 0$. If $\operatorname{Im} z > 0$, then

$$\operatorname{Im} \frac{1}{1 - zx} = \frac{x \operatorname{Im} z}{|1 - zx|^2} \geq 0.$$

The measure has positive mass on $x > 0$, so the integral has strictly positive imaginary part. For $\operatorname{Im} z < 0$, the imaginary part is strictly negative. A zero is impossible in all three regions. □

Remark 4.2. This is a proved zero-free function family generated by the calculus. Its zero-free property comes from a positive packet measure and not from a conjectural arithmetic cancellation.

Normalize the measure by $\widehat{\nu} = \nu/E_0$ and let p_m be its orthonormal polynomials. They obey

$$xp_m(x) = b_{m+1}p_{m+1}(x) + a_m p_m(x) + b_m p_{m-1}(x),$$

where $b_m > 0$. The Jacobi matrix is

$$\mathcal{J} = \begin{pmatrix} a_0 & b_1 & 0 & \cdots \\ b_1 & a_1 & b_2 & \ddots \\ 0 & b_2 & a_2 & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{pmatrix}.$$

The normalized Stieltjes transform

$$S(w) = \int \frac{\mathrm{d}\widehat{\nu}(x)}{w - x}$$

has the Jacobi continued fraction

$$\boxed{S(w) = \cfrac{1}{w - a_0 - \cfrac{b_1^2}{w - a_1 - \cfrac{b_2^2}{w - a_2 - \cdots}}}.}$$

Since

$$\mathcal{E}(z) = \frac{E_0}{z} S(1/z),$$

this continued fraction is an exact representation of the packet-energy resolvent.

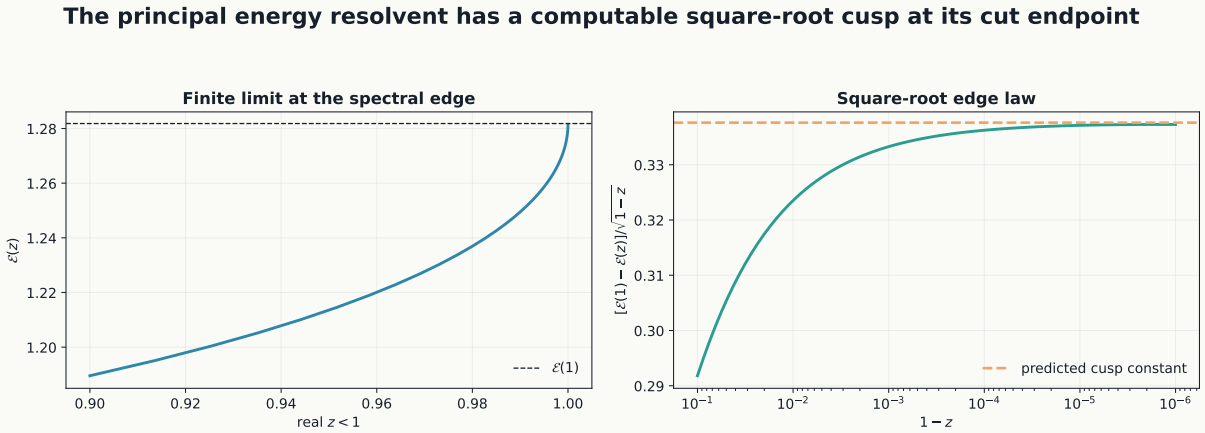


Figure 4.1: The spectral endpoint in the principal case. The resolvent remains finite at $z = 1$ but approaches it through a square-root cusp with a computable constant.

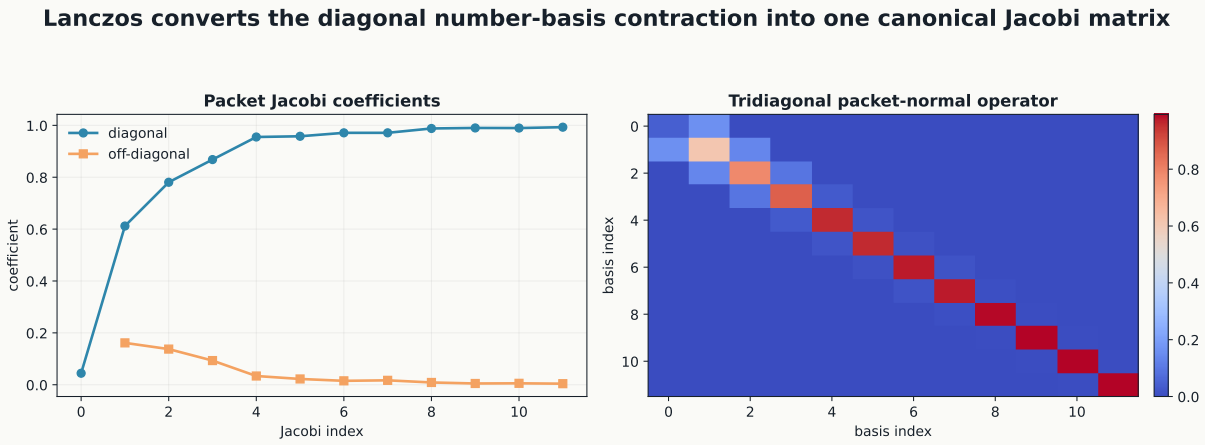


Figure 4.2: The global packet Jacobi matrix. Lanczos tridiagonalization turns the infinite diagonal number-basis operator into one canonical nearest-neighbor transfer system.

Table 4.1: Leading coefficients of the global packet Jacobi matrix.

index	diagonal	off-diagonal
0	0.04434860610204	0.16140179553905
1	0.61191054775417	0.13742304205532
2	0.78062025187444	0.09316571771169
3	0.86808908755506	0.03366248048232
4	0.95512403931655	0.02228734776099
5	0.95792345408654	0.01507011755538
6	0.97117460957236	0.01700980389036
7	0.97121093937559	0.00875955935121
8	0.98799441990959	0.00483454850130
9	0.99003096036785	-

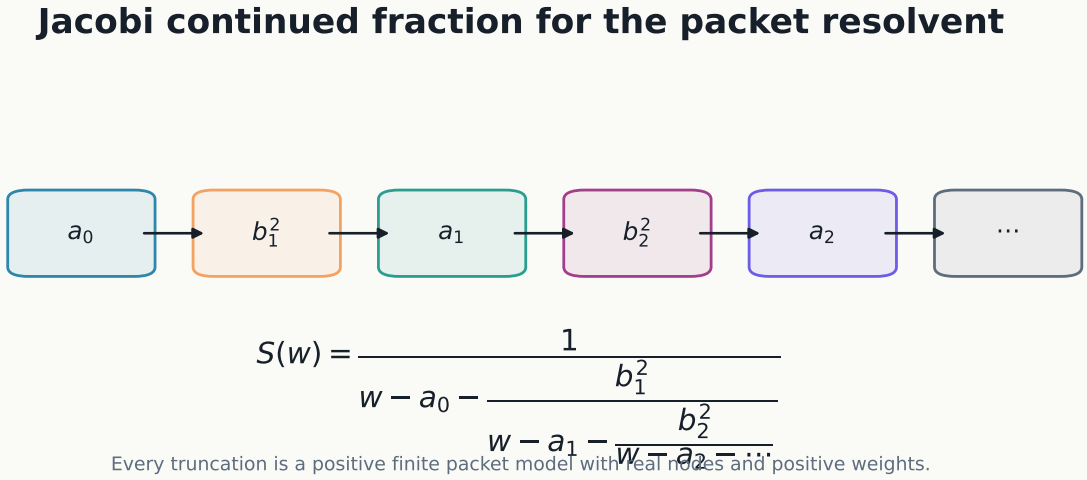


Figure 4.3: The Jacobi continued fraction. Every coefficient is determined by the positive packet measure. Finite truncations preserve positivity and produce rational reduced models.

Part III

Finite positive models

Chapter 5

Gauss packets and Padé-type resolvents

Let J_m denote the leading $m \times m$ Jacobi matrix. Its eigenvalues are

$$0 \le x_1^{(m)} < \cdots < x_m^{(m)} < 1.$$

If u_j is a normalized eigenvector, define

$$\lambda_j^{(m)} = |(u_j)_1|^2 .$$

Then every weight is positive and their sum is one.

Theorem 5.1 (Moment-exact packet reduction)**.** *For every polynomial P of degree at most $2m-1$,*

$$\boxed{\int P(x)\,\mathrm{d}\nu(x) = E_0 \sum_{j=1}^{m} \lambda_j^{(m)} P(x_j^{(m)}).}$$

Equivalently,

$$\boxed{E_r = E_0 e_1^T J_m^r e_1, \qquad 0 \le r \le 2m-1.}$$

Proof. This is Gaussian quadrature for the positive measure $\widehat{\nu}$. It can also be obtained directly from Lanczos. The Krylov space

$$\mathrm{span}\{v, Tv, \ldots, T^{m-1}v\}$$

contains every polynomial in T of degree below m applied to v. Orthogonality of the Lanczos residual makes the reduced bilinear form exact through degree $2m-1$. □

Corollary 5.2. *The reduced resolvent*

$$\mathcal{E}_m(z) = E_0 \sum_{j=1}^{m} \frac{\lambda_j^{(m)}}{1 - z x_j^{(m)}}$$

has positive residues and poles only at

$$\frac{1}{x_j^{(m)}} \in (1, \infty).$$

Its Taylor expansion agrees with $\mathcal{E}(z)$ through degree $2m-1$.

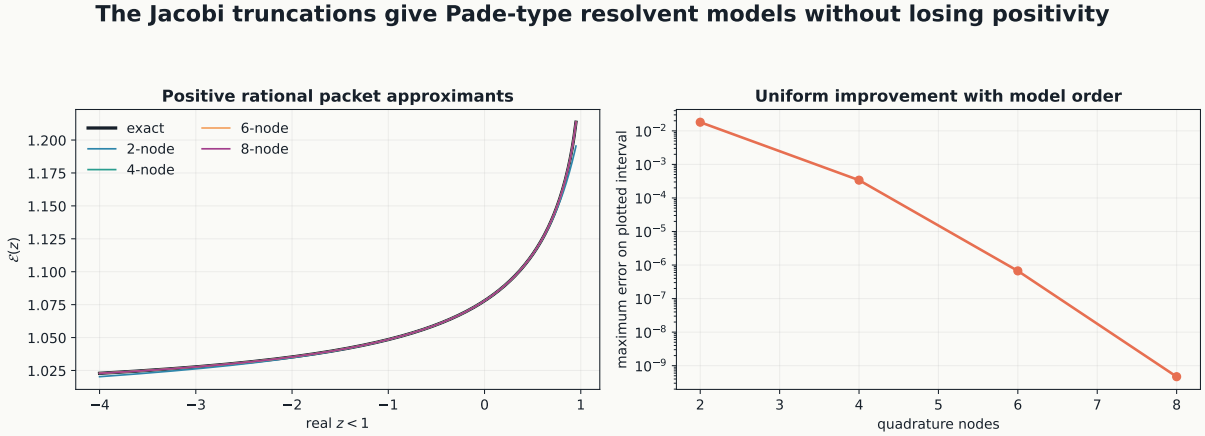


Figure 5.1: Positive rational approximants to the resolvent. The finite Jacobi models reproduce low-order moments exactly and converge rapidly on the tested compact real interval below the cut.

Table 5.1: Positive rational approximants to the packet-energy resolvent.

nodes	z	exact	reduced	absolute error
2	-4.0	1.02284656494961	1.02028441003455	2.5622e-03
2	-1.0	1.04848663001446	1.04837718829705	1.0944e-04
2	-0.25	1.06760037712403	1.06759909658957	1.2805e-06
2	0.5	1.11317625567786	1.11305377473775	1.2248e-04
2	0.9	1.18953662583038	1.18048877686716	9.0478e-03
4	-4.0	1.02284656494961	1.02284654817635	1.6773e-08
4	-1.0	1.04848663001446	1.04848662992499	8.9473e-11
4	-0.25	1.06760037712403	1.06760037712401	2.1982e-14
4	0.5	1.11317625567786	1.11317625492563	7.5223e-10
4	0.9	1.18953662583038	1.18949671196745	3.9914e-05
6	-4.0	1.02284656494961	1.02284656494961	8.8818e-16
6	-1.0	1.04848663001446	1.04848663001446	4.4409e-16
6	-0.25	1.06760037712403	1.06760037712403	4.4409e-16
6	0.5	1.11317625567786	1.11317625567786	2.2204e-16

nodes	z	exact	reduced	absolute error
6	0.9	1.18953662583038	1.18953661628695	9.5434e-09
8	-4.0	1.02284656494961	1.02284656494961	6.6613e-16
8	-1.0	1.04848663001446	1.04848663001446	2.2204e-16
8	-0.25	1.06760037712403	1.06760037712403	2.2204e-16
8	0.5	1.11317625567786	1.11317625567786	2.2204e-16
8	0.9	1.18953662583038	1.18953662582974	6.4859e-13
10	-4.0	1.02284656494961	1.02284656494961	6.6613e-16
10	-1.0	1.04848663001446	1.04848663001446	4.4409e-16
10	-0.25	1.06760037712403	1.06760037712403	2.2204e-16
10	0.5	1.11317625567786	1.11317625567786	4.4409e-16
10	0.9	1.18953662583038	1.18953662583038	0.0000e+00

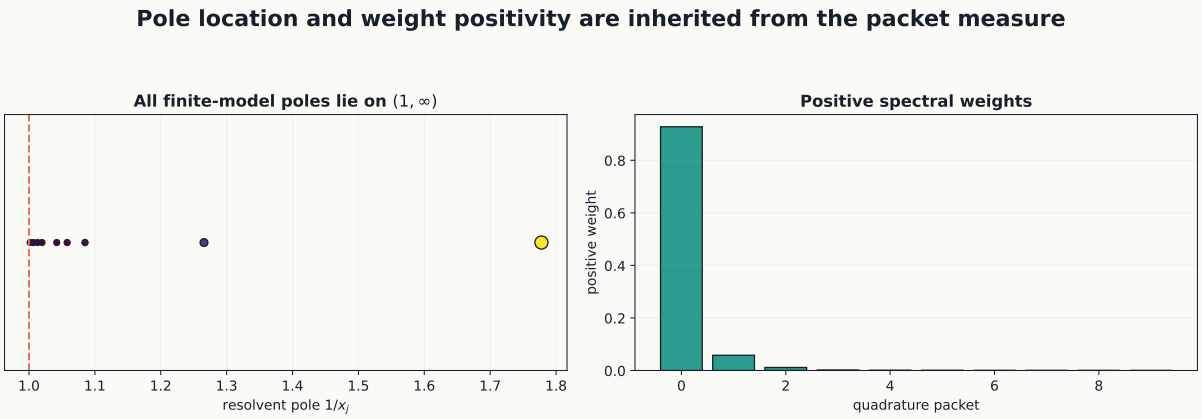


Figure 5.2: Poles and weights of a ten-node packet model. All nonzero poles lie on the physical cut and every spectral weight is positive. A zero node represents the endpoint atom and produces no finite pole.

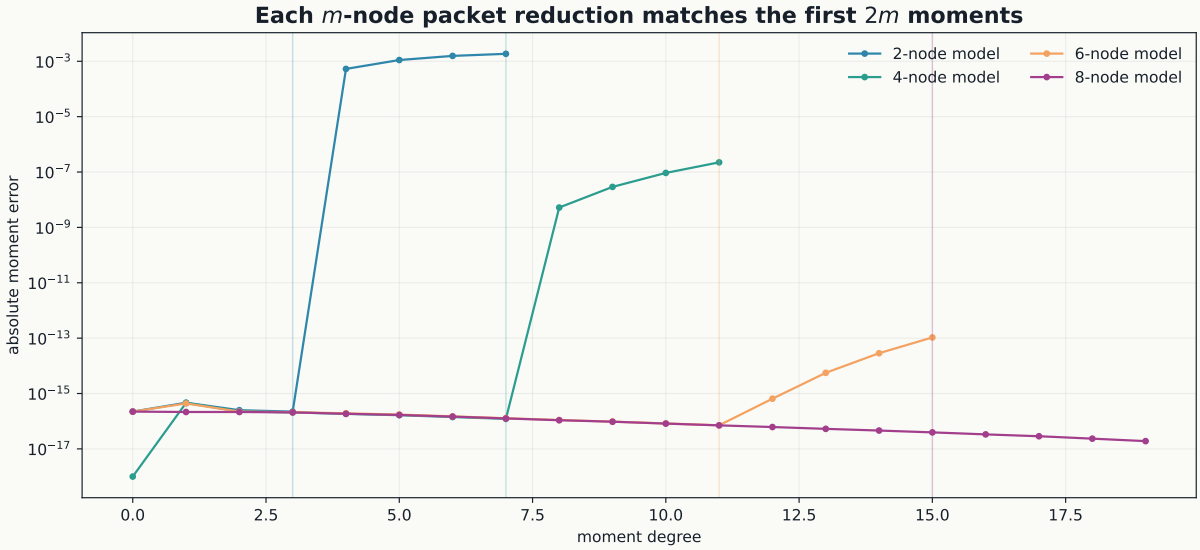


Figure 5.3: Exact moment matching. An m-node model agrees through degree $2m-1$. The first unmatched moments expose the boundary of the finite contract.

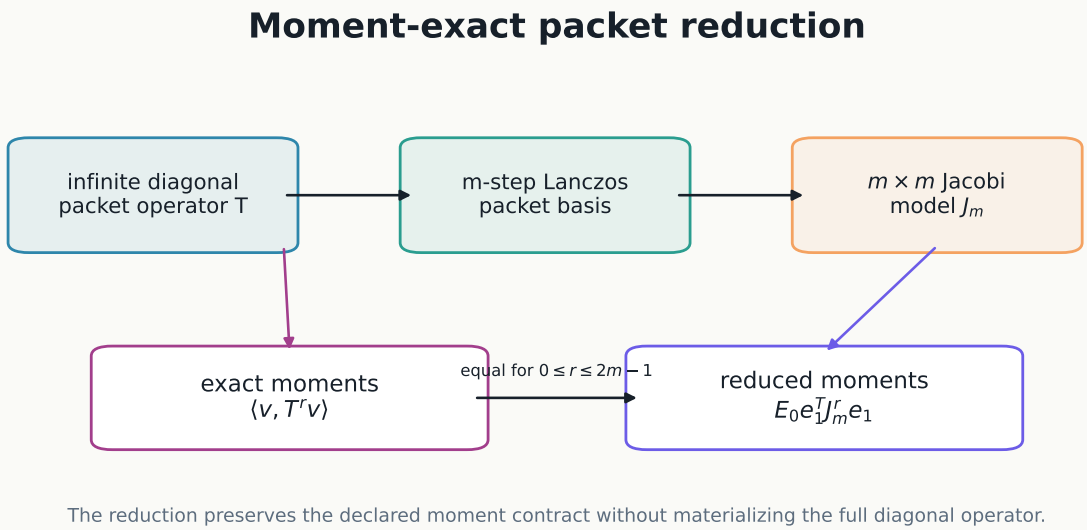


Figure 5.4: The reduction map. The finite Jacobi operator is not an arbitrary fit. It is the unique Gaussian packet model attached to the Krylov basis.

Part IV

Gamma tilting and Laguerre universality

Chapter 6

Endpoint scaling of the packet measure

Set

$$\alpha = \frac{2a-1}{\beta}$$

and define the tilted probability measure

$$\mathrm{d}\nu_k(x) = \frac{x^k}{E_k}\,\mathrm{d}\nu(x).$$

The relevant endpoint coordinate is

$$\boxed{u = k(1-x).}$$

For the atom $x_n = (1-n^{-\beta})^2$,

$$u = k\left(2n^{-\beta} - n^{-2\beta}\right).$$

Let

$$Q_q(X) = \sum_{\substack{n\le X\\(n,q)=1}} \mu(n)^2.$$

The squarefree counting law is

$$Q_q(X) = \delta_q X + O_q(\sqrt{X}), \qquad \delta_q = \frac{1}{\zeta(2)}\prod_{p\mid q}\frac{p}{p+1}.$$

Theorem 6.1 (Gamma packet scaling)**.** *Under ν_k, the coordinate $u = k(1-x)$ converges in distribution to the Gamma law with shape α and scale one. For every bounded continuous φ,*

$$\boxed{\lim_{k\to\infty}\int \varphi(k(1-x))\,\mathrm{d}\nu_k(x) = \frac{1}{\Gamma(\alpha)}\int_0^\infty \varphi(u)u^{\alpha-1}\mathrm{e}^{-u}\,\mathrm{d}u.}$$

Proof. Write the numerator as

$$\sum_{\substack{n\ge 1\\(n,q)=1}} \mu(n)^2 n^{-2a}(1-n^{-\beta})^{2k}\varphi\left(k[1-(1-n^{-\beta})^2]\right).$$

Put $n = (2k)^{1/\beta}y$. Partial summation with the squarefree counting law replaces the sum, after multiplication by $(2k)^\alpha$, by

$$\delta_q\int_0^\infty y^{-2a}\mathrm{e}^{-y^{-\beta}}\varphi(y^{-\beta})\,\mathrm{d}y.$$

The substitution $u = y^{-\beta}$ gives

$$\frac{\delta_q}{\beta}\int_0^\infty \varphi(u)u^{\alpha-1}\mathrm{e}^{-u}\,\mathrm{d}u.$$

Taking $\varphi = 1$ gives the denominator asymptotic. Division yields the normalized Gamma law. Standard truncation at small and large u supplies domination. □

Corollary 6.2 (Energy asymptotic)**.**

$$\boxed{E_{a,\beta,q}(k) \sim A_{a,\beta,q} k^{-\alpha}, \qquad A_{a,\beta,q} = \frac{\delta_q}{\beta}\Gamma(\alpha)2^{-\alpha}.}$$

Worked example 6.3 (Principal Gamma law)**.** For $a = \beta = 2$ and $q = 1$,

$$\alpha = \frac{3}{2}, \qquad A = \frac{3}{2^{5/2}\pi^{3/2}}.$$

Thus

$$E_k \sim \frac{3}{2^{5/2}\pi^{3/2}} k^{-3/2}.$$

The scaled variable $k(1 - x_n)$ is asymptotically the same as $2k/n^2$.

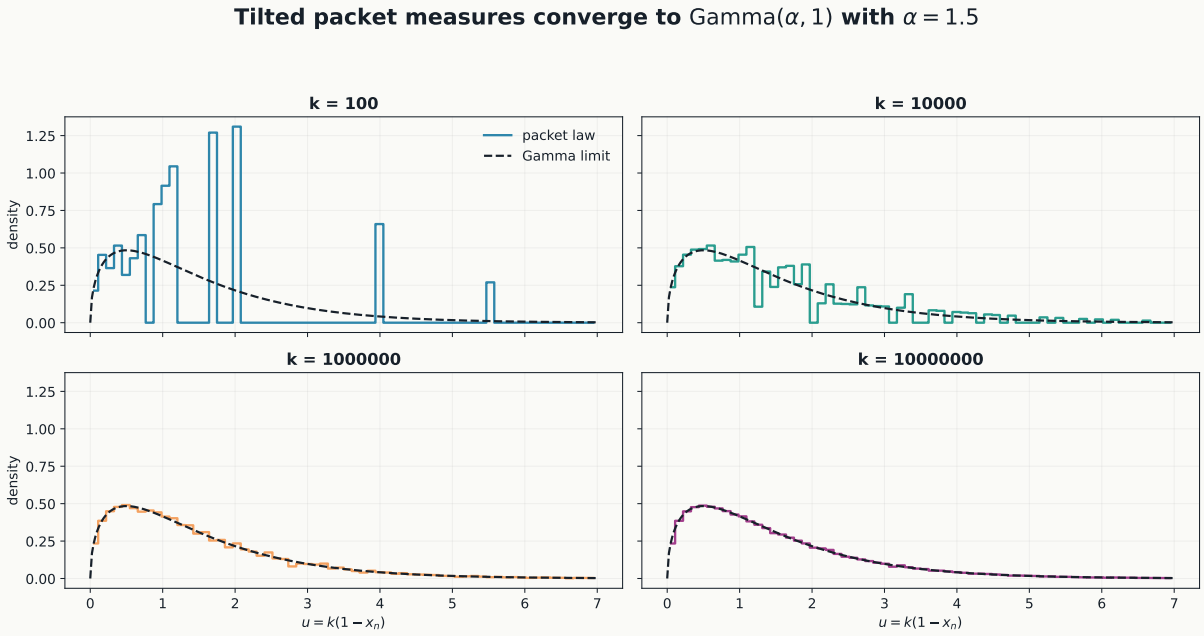


Figure 6.1: Convergence of the tilted packet law. The discrete packet histograms approach the Gamma density. Arithmetic oscillations remain visible at finite depth and decay under scaling.

Chapter 7

Laguerre packet polynomials

Let $P_m^{(k)}$ be the monic orthogonal polynomial of degree m for the pushed-forward law of $u = k(1-x)$ under ν_k. Define the monic Gamma–Laguerre polynomial

$$\boxed{\Lambda_m^{(\alpha)}(u) = (-1)^m m! L_m^{(\alpha-1)}(u).}$$

Its squared norm under the normalized Gamma law is

$$\boxed{h_m^{(\alpha)} = m!(\alpha)_m.}$$

Theorem 7.1 (Fixed-degree Laguerre universality)**.** *For every fixed degree m,*

$$\boxed{P_m^{(k)}(u) \longrightarrow \Lambda_m^{(\alpha)}(u)}$$

coefficientwise and locally uniformly as $k \to \infty$.

Proof. The Gamma scaling theorem gives convergence of every fixed moment of the pushed-forward measures. The Gamma moment matrix of order m is positive definite. The coefficients of the monic orthogonal polynomial are the solution of a linear system whose entries are moments. Continuity of the solution under convergence of the positive definite moment matrix gives coefficientwise convergence. Local uniform convergence follows because the degree is fixed. □

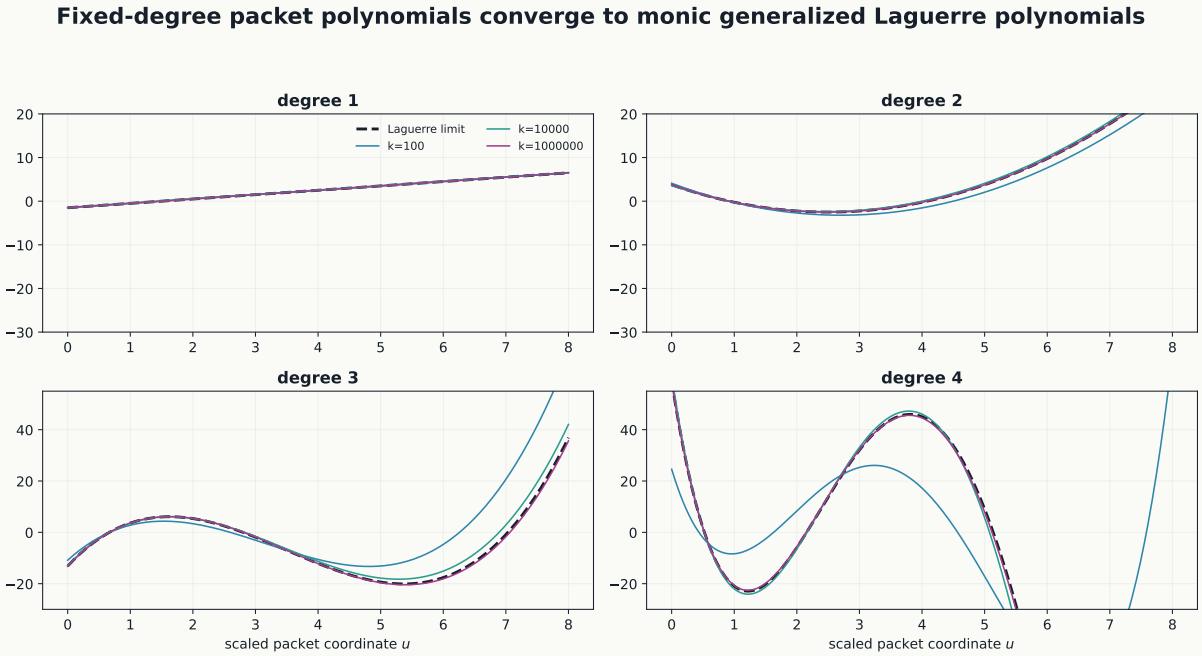


Figure 7.1: Polynomial convergence in degrees one through four. The packet polynomials arise from the exact discrete arithmetic measure. Their fixed-degree scaling limit is the classical monic Laguerre family.

The monic Laguerre recurrence is

$$\Lambda_{m+1}(u) = (u - 2m - \alpha)\Lambda_m(u) - m(m + \alpha - 1)\Lambda_{m-1}(u).$$

The orthonormal Jacobi coefficients are

$$a_m^{(\infty)} = 2m + \alpha, \qquad b_{m+1}^{(\infty)} = \sqrt{(m+1)(m+\alpha)}.$$

Corollary 7.2 (Jacobi fixed point)**.** *For every fixed m,*

$$\boxed{a_m(k) \to 2m + \alpha}$$

and

$$\boxed{b_{m+1}(k) \to \sqrt{(m+1)(m+\alpha)}.}$$

Every fixed leading block of the scaled tilted Jacobi matrix converges to the Gamma–Laguerre Jacobi matrix.

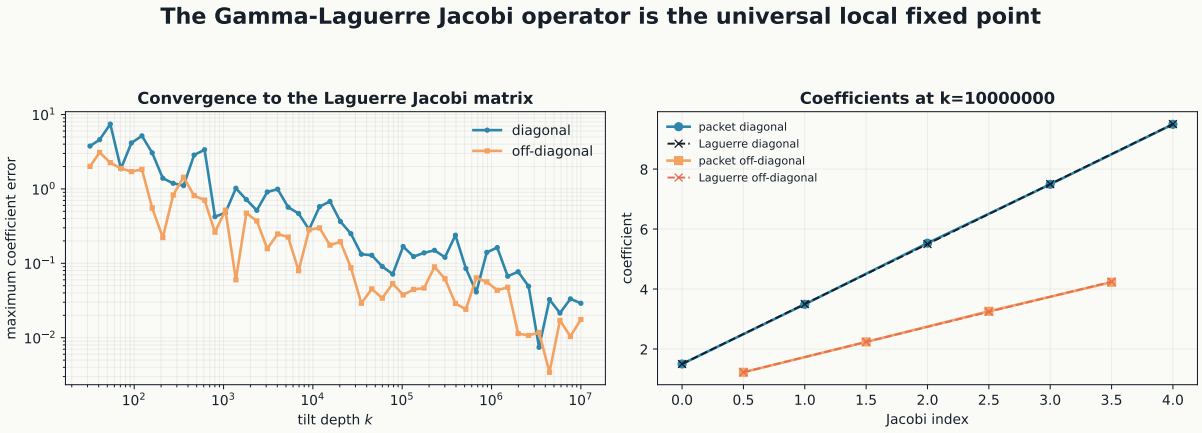


Figure 7.2: Convergence of the packet recurrence coefficients. The finite arithmetic Jacobi systems approach one universal Gamma–Laguerre transfer operator determined only by $\alpha = (2a-1)/\beta$.

Table 7.1: Tilted packet recurrence coefficients and their Laguerre limits.

k	m	$a_m(k)$	$2m+\alpha$	$b_{m+1}(k)$	$\sqrt{(m+1)(m+\alpha)}$
100	0	1.439469766734	1.500000000000	1.269003654057	1.224744871392
100	1	3.973524920336	3.500000000000	2.047996433911	2.236067977500
100	2	4.111390894522	5.500000000000	2.533871126315	3.240370349204
100	3	4.694483034010	7.500000000000	2.025239047230	4.242640687119
100	4	9.224025847824	9.500000000000	-	-
10000	0	1.512379719532	1.500000000000	1.228463532830	1.224744871392
10000	1	3.395066477825	3.500000000000	2.133929790999	2.236067977500
10000	2	5.476266499055	5.500000000000	3.460673302499	3.240370349204
10000	3	8.064807590273	7.500000000000	4.228504140903	4.242640687119
10000	4	9.253565373962	9.500000000000	-	-
1000000	0	1.498156796917	1.500000000000	1.223422571318	1.224744871392
1000000	1	3.506581073530	3.500000000000	2.242224950631	2.236067977500
1000000	2	5.535760920200	5.500000000000	3.264876189033	3.240370349204
1000000	3	7.431313020061	7.500000000000	4.160342017367	4.242640687119
1000000	4	9.418067422334	9.500000000000	-	-
10000000	0	1.500552344413	1.500000000000	1.223704446932	1.224744871392
10000000	1	3.490039174832	3.500000000000	2.235038825728	2.236067977500
10000000	2	5.529025150798	5.500000000000	3.257966360727	3.240370349204
10000000	3	7.491372879783	7.500000000000	4.227072501904	4.242640687119
10000000	4	9.492863864576	9.500000000000	-	-

Part V

Determinants, kernels, and quadrature limits

Chapter 8

Shifted Hankel determinant asymptotics

For $d \geq 0$, define

$$\Delta_d(k) = \det\left[E_{k+i+j}\right]_{i,j=0}^{d}.$$

Let

$$G_d(\alpha) = \prod_{j=0}^{d} j!(\alpha)_j.$$

Theorem 8.1 (Complete leading Hankel asymptotic)**.** *For fixed d,*

$$\boxed{\Delta_d(k) \sim A^{d+1} G_d(\alpha)\, k^{-(d+1)\alpha - d(d+1)}}$$

with

$$A = \frac{\delta_q}{\beta}\Gamma(\alpha)2^{-\alpha}.$$

Proof. Factor E_k from every row of the determinant:

$$\Delta_d(k) = E_k^{d+1} \det\left[\int x^{i+j}\, \mathrm{d}\nu_k(x)\right]_{i,j=0}^{d}.$$

Apply the affine change $x = 1 - u/k$. The determinant of the triangular basis change from $1, x, \ldots, x^d$ to $1, u, \ldots, u^d$ contributes

$$k^{-d(d+1)}.$$

The remaining determinant is the Hankel determinant of the first $2d$ moments of the pushed-forward u law. Those moments converge to the Gamma moments $(\alpha)_r$. The Gamma Hankel determinant equals the product of the monic orthogonal polynomial norms,

$$\det[(\alpha)_{i+j}]_{i,j=0}^{d} = \prod_{j=0}^{d} j!(\alpha)_j.$$

Finally, $E_k^{d+1} \sim A^{d+1} k^{-(d+1)\alpha}$. □

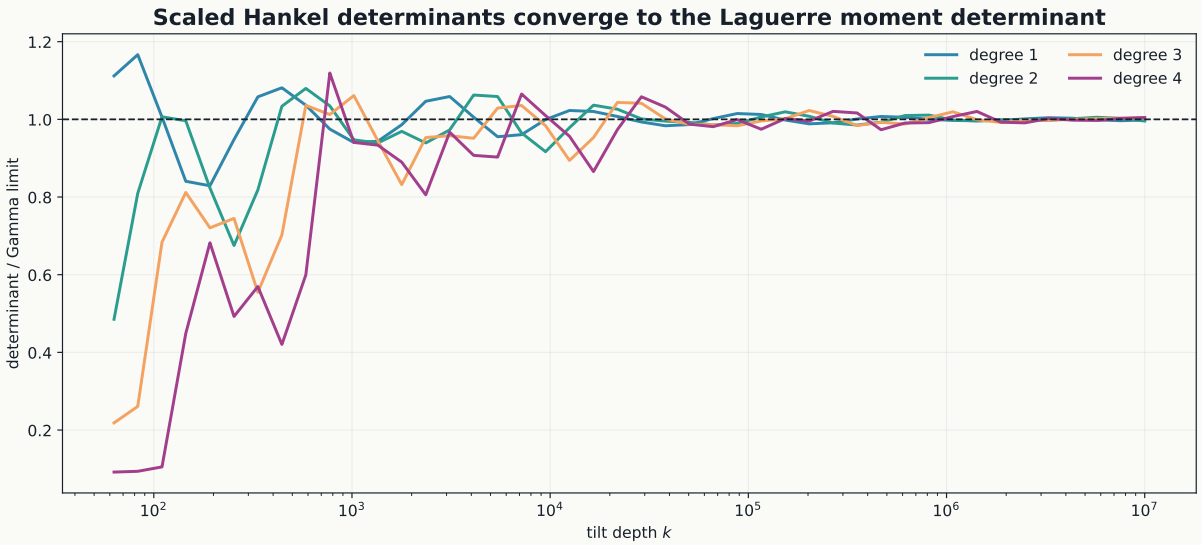


Figure 8.1: Scaled shifted Hankel determinants. Four determinant orders converge to the Gamma–Laguerre constants predicted by the theorem.

Table 8.1: Scaled tilted Hankel determinants. The predicted limit is one.

k	degree d	scaled ratio
100	1	1.073580182674
100	2	0.966846123424
100	3	0.532426889339
100	4	0.066810185910
1000	1	0.941662798011
1000	2	0.952615568715
1000	3	1.058602075799
1000	4	0.957062934243
10000	1	1.006081767663
10000	2	0.921842668600
10000	3	0.963412417247
10000	4	1.000158173308
100000	1	1.016043932714
100000	2	0.995884542402
100000	3	0.990074538800
100000	4	0.980822628604

Chapter 9

Christoffel–Darboux universality

Let $p_m^{(k)}$ denote the orthonormal packet polynomial in the scaled coordinate u and define

$$K_d^{(k)}(u,v) = \sum_{m=0}^{d} p_m^{(k)}(u) p_m^{(k)}(v).$$

The Gamma–Laguerre kernel is

$$K_d^{(\alpha)}(u,v) = \sum_{m=0}^{d} \frac{\Lambda_m^{(\alpha)}(u)\Lambda_m^{(\alpha)}(v)}{m!(\alpha)_m}.$$

Theorem 9.1 (Fixed-degree kernel convergence)**.** *For every fixed d,*

$$\boxed{K_d^{(k)}(u,v) \longrightarrow K_d^{(\alpha)}(u,v)}$$

locally uniformly on $\mathbb{C}^2$.

Proof. The monic polynomials and their norms converge degree by degree. The kernel is a finite sum, so local uniform convergence follows. □

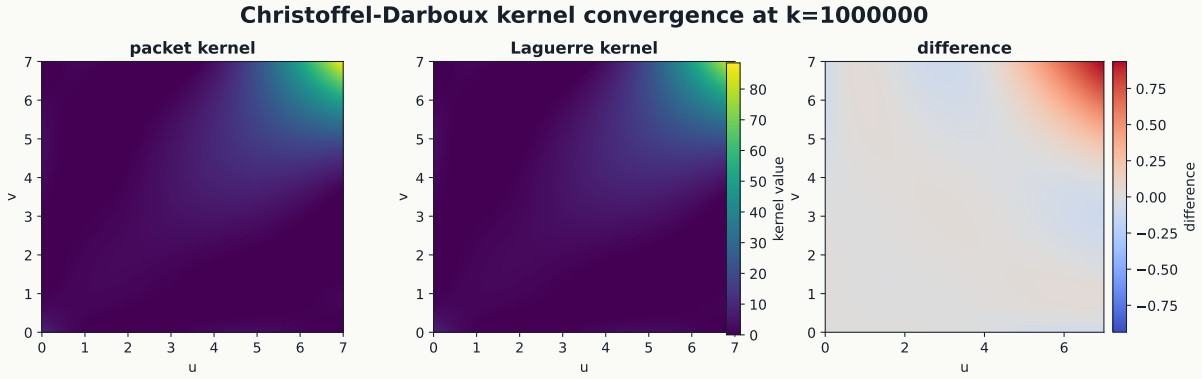


Figure 9.1: Christoffel–Darboux kernel convergence. The packet kernel, the Gamma–Laguerre kernel, and their difference are shown at a large but finite tilt depth.

Chapter 10

Gauss–Laguerre packet quadrature

The m-node Gaussian rule for the scaled packet law has nodes equal to the eigenvalues of the $m \times m$ tilted Jacobi matrix and weights equal to the squares of the first eigenvector components.

Theorem 10.1 (Quadrature fixed point)**.** *For fixed m, the packet quadrature nodes and weights converge to the normalized Gauss–Laguerre nodes and weights for the density*

$$\frac{u^{\alpha-1}\mathrm{e}^{-u}}{\Gamma(\alpha)}.$$

Proof. The finite Jacobi matrices converge entrywise. Their limiting spectra are simple. Eigenvalue and normalized spectral-weight continuity gives convergence of nodes and weights. □

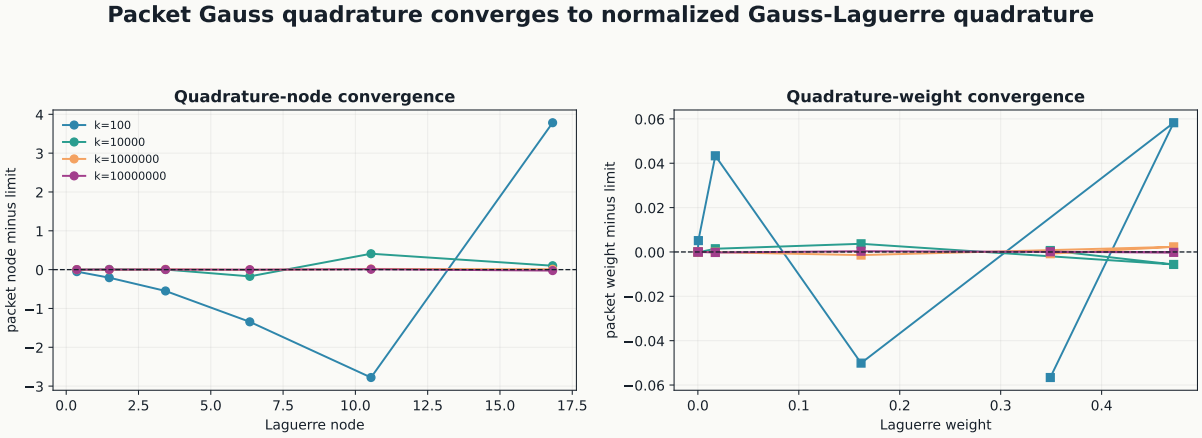


Figure 10.1: Gauss quadrature convergence. Node and weight errors contract toward zero as the arithmetic packet tilt approaches its Gamma fixed point.

Table 10.1: Six-node packet quadrature at $k = 10^7$ and the normalized Gauss–Laguerre limit.

index	packet node	limit node	packet weight	limit weight
1	0.367038870927	0.366949877308	0.349228378823	0.349147704667
2	1.489958520058	1.488534292310	0.471202704086	0.471382822739
3	3.436713209300	3.434007968424	0.162013814205	0.161680794282
4	6.347595884392	6.349067925680	0.017064851303	0.017300863450
5	10.549595327665	10.540469858448	0.000488375314	0.000485982973
6	16.795956001975	16.820970077828	0.000001876269	0.000001831890

Part VI

The spectral edge and the universal family

Chapter 11

A square-root cusp in the principal resolvent

For the principal parameters $a = \beta = 2$ and $q = 1$, the atom locations satisfy

$$1 - x_n = \frac{2}{n^2} - \frac{1}{n^4}.$$

Near the endpoint, the packet measure has density equivalent

$$\mathrm{d}\nu \sim C y^{1/2}\,\mathrm{d}y, \qquad y = 1 - x, \qquad C = \frac{3}{2^{3/2}\pi^2}.$$

Theorem 11.1 (Principal edge cusp)**.** *As $z \uparrow 1$ through real values,*

$$\boxed{\mathcal{E}(1) - \mathcal{E}(z) \sim \frac{3}{2^{3/2}\pi}\sqrt{1 - z}.}$$

Proof. Put $\varepsilon = 1 - z$. The singular part is

$$\int_0^{\eta} C y^{1/2}\left(\frac{1}{y} - \frac{1}{y + \varepsilon}\right)\mathrm{d}y.$$

After $y = \varepsilon u$, the leading term is

$$C\varepsilon^{1/2}\int_0^{\infty} \frac{u^{-1/2}}{1 + u}\,\mathrm{d}u = C\pi\varepsilon^{1/2}.$$

Since $C\pi = 3/(2^{3/2}\pi)$, the formula follows. Contributions away from the endpoint are $O(\varepsilon)$. □

Chapter 12

Universality classes

The local fixed point depends on the original arithmetic parameters only through

$$\boxed{\alpha = \frac{2a - 1}{\beta}.}$$

Two packet systems with the same α have the same Gamma limit, Laguerre recurrence, determinant constants after energy normalization, fixed-degree kernel, and quadrature limit.

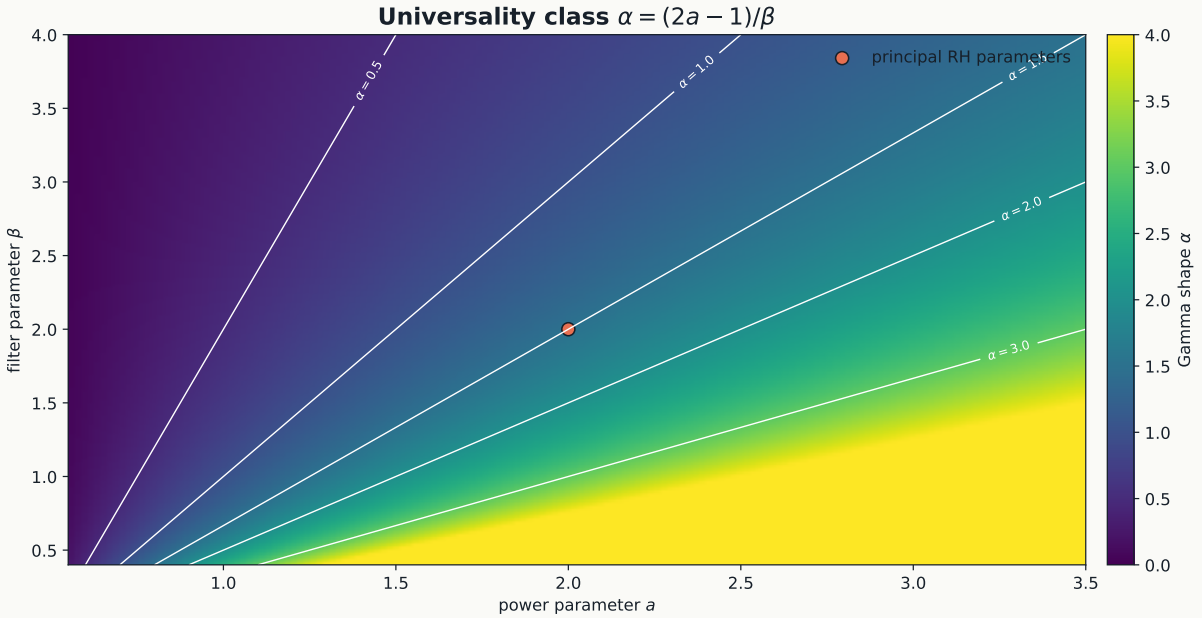


Figure 12.1: The parameter map. Curves of constant α define universality classes. The principal RH energy occupies the class $\alpha = 3/2$.

Worked example 12.1 (Three parameter pairs in one class)**.** The choices

$$(a, \beta) = (2, 2), \qquad \left(\frac{5}{4}, 1\right), \qquad \left(\frac{7}{2}, 4\right)$$

all give $\alpha = 3/2$. Their unscaled measures differ, but the endpoint packet polynomials converge to the same Laguerre family.

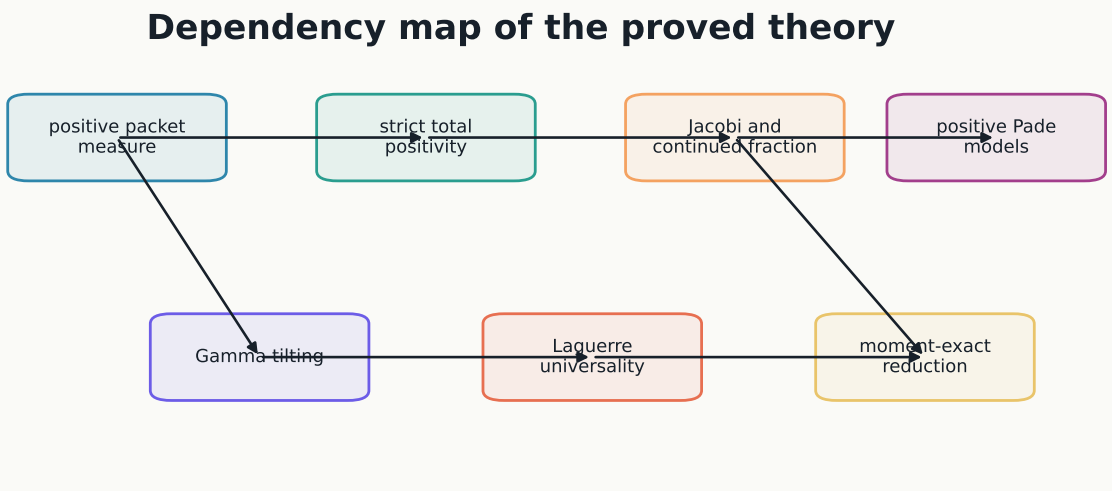


Figure 12.2: Logical dependency of the results. The positive measure supports two branches. One leads to total positivity and resolvent reduction. The other leads to Gamma scaling and Laguerre universality.

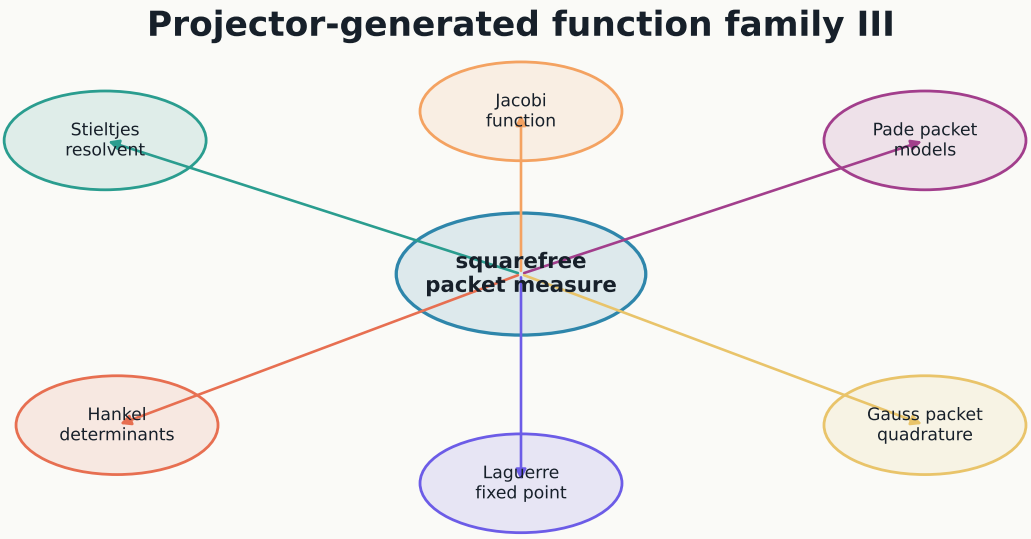


Figure 12.3: The third generated function family. The resolvent, Jacobi function, rational models, Hankel determinants, Laguerre fixed point, and quadrature rules are exact transforms of one packet measure.

Part VII

Quantum coordinates and the arithmetic boundary

Chapter 13

Number-basis realization

On the squarefree number basis define

$$H\,|n\rangle = \log n\,|n\rangle$$

and define the positive contraction directly by

$$T\,|n\rangle = \left(1 - n^{-\beta}\right)^2 |n\rangle\,.$$

On the invariant subspace spanned by $n \geq 2$, introduce

$$K_\beta\,|n\rangle = -\log(1 - n^{-\beta})\,|n\rangle\,.$$

Then

$$T = \mathrm{e}^{-2K_\beta}$$

on that subspace. The remaining basis vector $|1\rangle$ is the zero-eigenvalue packet of T and is kept separately, avoiding the singular logarithm at $n = 1$. For

$$|v_a\rangle = \sum_{\substack{n\geq 1\\(n,q)=1}} \frac{\mu(n)^2}{n^a}\,|n\rangle\,,$$

the packet moments satisfy

$$E_{a,\beta,q}(k) = \langle v_a|\,T^k\,|v_a\rangle\,.$$

The operators H, K_β, and T commute because they are functions of the number operator.

The logarithmic Hamiltonian $H\,|n\rangle = \log n\,|n\rangle$ is the same spectral coordinate used in the engineered quantum systems of Wei and collaborators [10]. Their construction supplies polynomial-resource procedures for logarithmic phase evolution. The present positive measure adds a separate Euclidean packet-depth evolution generated by K_β.

Remark 13.1. The positive Jacobi reduction can be interpreted as a Lanczos compression of the number-basis contraction. It produces a one-dimensional nearest-neighbor model that preserves a declared finite moment contract. This is a model-reduction statement. It is not a claim that the full number-theoretic quantum system is physically natural.

Chapter 14

The unresolved signed coherence

For the principal zeta parameters define

$$c_k = \sum_{n\geq 1} \frac{\mu(n)}{n^2}\left(1-\frac{1}{n^2}\right)^k$$

and

$$E_k = \sum_{n\geq 1} \frac{\mu(n)^2}{n^4}\left(1-\frac{1}{n^2}\right)^{2k}.$$

The normalized coherence is

$$\Gamma_k = \frac{|c_k|^2}{E_k}.$$

Since

$$E_k \sim \frac{3}{2^{5/2}\pi^{3/2}} k^{-3/2},$$

the classical Báez-Duarte criterion gives

$$\boxed{\text{RH} \iff \Gamma_k = O_\varepsilon(k^\varepsilon) \quad \text{for every } \varepsilon > 0.}$$

The positive results in this paper determine the denominator and its canonical reductions. They do not control the signed numerator.

Boundary statement 14.1 (What remains). Strict total positivity of E_k, positivity of the Jacobi model, the zero-free resolvent, and Laguerre universality do not imply subpolynomial Möbius coherence. The missing estimate must use the arithmetic signs of $\mu(n)$ or an equivalent prime-sensitive structure.

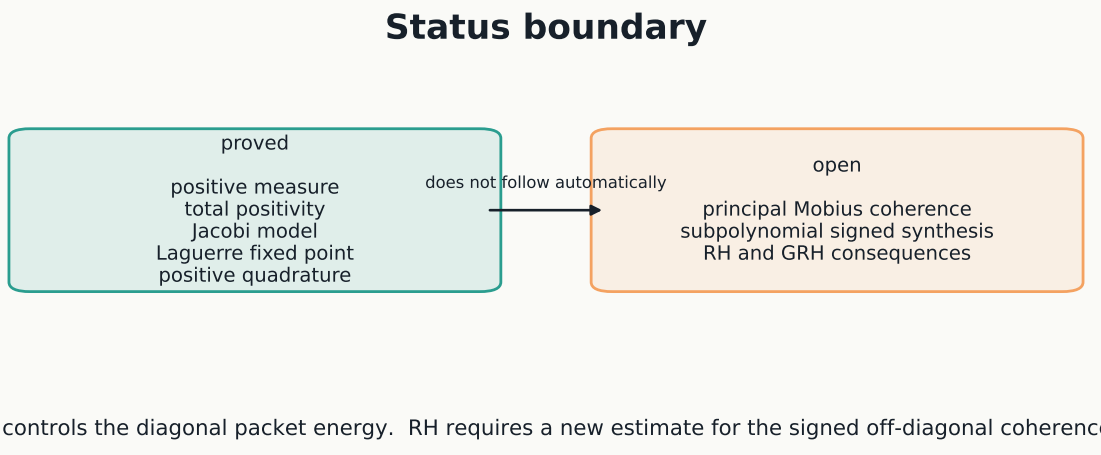


Figure 14.1: The exact boundary of the paper. The positive side is complete within the stated setting. The signed principal coherence remains the open arithmetic problem.

Part VIII

Worked calculations and computational record

Chapter 15

Worked calculation of a four-node reduction

Let $m = 4$. Lanczos applied to (T, v) produces

$$J_4 = \begin{pmatrix} a_0 & b_1 & 0 & 0 \\ b_1 & a_1 & b_2 & 0 \\ 0 & b_2 & a_2 & b_3 \\ 0 & 0 & b_3 & a_3 \end{pmatrix}.$$

The four-node rule is exact for degrees zero through seven:

$$E_r = E_0 e_1^T J_4^r e_1, \qquad 0 \le r \le 7.$$

Diagonalizing $J_4 = U \Lambda U^T$ yields

$$E_r = E_0 \sum_{j=1}^{4} U_{1j}^2 \Lambda_j^r.$$

Thus an infinite squarefree sum is replaced by four positive packets without changing the first eight moments.

Algorithm 15.1 (Constructing an m-packet reduced resolvent)**.** 1. Form the diagonal nodes x_n and seed amplitudes $\sqrt{w_n}$.

2. Run m Lanczos steps using only diagonal multiplication by x_n.
3. Assemble the tridiagonal Jacobi matrix J_m.
4. Diagonalize J_m to obtain positive nodes and weights.
5. Return
$$\mathcal{E}_m(z) = E_0 \sum_{j=1}^{m} \frac{\lambda_j}{1 - z x_j}.$$
6. Certify the moments through degree $2m - 1$.

Chapter 16

Numerical tables and verification

Computational verification
Built-in checks: 34/34 passed
Independent tests: 19/19 passed
Figure pairs: 22
Principal Gamma shape: $\alpha = 1.5$
Energy constant: 0.095240453901361

The tables use a squarefree sieve through 1,500,000. Low-order moment values are checked independently through the exact zeta-ratio formula at high precision. The asymptotic experiments are finite numerical certificates. Their purpose is to verify the formulas and display convergence, not to replace the proofs.

Appendix A

Formula registry

Object	Formula
Packet node	$x_n = (1 - n^{-\beta})^2$
Packet weight	$w_n = \mu(n)^2 \left\lvert \chi(n) \right\rvert^2 n^{-2a}$
Moment	$E_k = \sum_n w_n x_n^k$
Resolvent	$\mathcal{E}(z) = \sum_n w_n/(1 - z x_n)$
Gamma shape	$\alpha = (2a - 1)/\beta$
Energy constant	$A = \delta_q \Gamma(\alpha) 2^{-\alpha}/\beta$
Tilted law	$\mathrm{d}\nu_k = x^k \, \mathrm{d}\nu / E_k$
Scaled coordinate	$u = k(1 - x)$
Laguerre polynomial	$\Lambda_m^{(\alpha)} = (-1)^m m! L_m^{(\alpha - 1)}$
Laguerre norm	$h_m = m!(\alpha)_m$
Shifted determinant	$\Delta_d(k) = \det[E_{k+i+j}]_{i,j=0}^{d}$
Determinant constant	$G_d(\alpha) = \prod_{j=0}^{d} j!(\alpha)_j$
Principal cusp	$\mathcal{E}(1) - \mathcal{E}(z) \sim 3(1 - z)^{1/2}/(2^{3/2}\pi)$
RH coherence	$\Gamma_k = \left\lvert c_k \right\rvert^2 / E_k$

Appendix B

Implementation map

The complete implementation is supplied as

`projector_generated_special_functions_iii.py`.

It contains the squarefree sieve, packet-measure construction, high-precision zeta-ratio checks, diagonal Lanczos routine, orthogonal-polynomial reconstruction, determinant calculations, quadrature calculations, figure generation, table generation, and the built-in certificate ledger.

The central computational path is:

```
params = PacketParameters(a=2.0, beta=2.0, conductor=1)
measure = build_measure(params, nmax=1_500_000)
energy = measure.energy(k)
global_model = global_lanczos(measure, steps=m)
tilted_model = tilted_lanczos(measure, k=k, steps=m)
nodes, weights = gauss_quadrature(tilted_model)
```

The Lanczos routine never materializes a dense operator. It multiplies vectors by the diagonal node array, orthogonalizes the resulting Krylov vectors, and returns the tridiagonal Jacobi coefficients. The source file is included unchanged in both release archives.

Appendix C

Reproduction

From the source directory, run

```
python projector_generated_special_functions_iii.py
python generate_tables.py
python -m pytest -q
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_III.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_III.tex
```

Bibliography

[1] Komorebi P., *Projector-Generated Special Functions I: Polyphase Differential Lifts, Bernoulli–Lerch Wavelets, Cyclotomic Detail Functions, and Möbius Coherence*, 2026.

[2] Komorebi P., *Projector-Generated Special Functions II: Cyclotomic Stieltjes Packets, Packet Resultants, Positive Energy Resolvents, Gamma Scaling, and Character Decoherence*, 2026.

[3] N. I. Akhiezer, *The Classical Moment Problem and Some Related Questions in Analysis*, Oliver and Boyd, 1965.

[4] T. S. Chihara, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, 1978.

[5] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*, Oxford University Press, 2004.

[6] G. H. Golub and J. H. Welsch, Calculation of Gauss quadrature rules, *Mathematics of Computation* 23 (1969), 221–230.

[7] G. Szegő, *Orthogonal Polynomials*, American Mathematical Society, fourth edition, 1975.

[8] NIST Digital Library of Mathematical Functions, Chapters 18 and 25, `https://dlmf.nist.gov/`.

[9] L. Báez-Duarte, A sequential Riesz-like criterion for the Riemann hypothesis, *International Journal of Mathematics and Mathematical Sciences* 2005 (2005), 3527–3537.

[10] S. Wei, Y. Zhai, Q. Lu, et al., The Riemann Hypothesis manifested in dynamical quantum phase transitions, *Nature Communications*, accepted 18 June 2026, DOI: 10.1038/s41467-026-74935-8.

Projector-Generated Special Functions IV

Squarefree Depth–Time Kernels, Universal Gamma Dephasing, Möbius Resolvents, and Signed Padé Packet Models

Komorebi P.

Abstract

This paper develops the first joint scale-time special-function system generated by the projector-resolved calculus. The signed seed is the Möbius transfer

$$C_{a,\beta,\chi}(k,t)=\sum_{n\geq 1}\frac{\mu(n)\chi(n)}{n^{a+\mathrm{i}t}}\left(1-\frac{1}{n^{\beta}}\right)^{k}.$$

Its long-time cross-correlation removes the arithmetic signs on the diagonal and produces a positive depth-time kernel. The kernel is proved to be positive definite and to admit an exact finite-difference coordinate system built from squarefree Dirichlet series. Its large-depth behavior is universal. After one carrier phase is removed, the equal-depth correlation converges to

$$\frac{\Gamma(\alpha+\mathrm{i}\tau/\beta)}{\Gamma(\alpha)},\qquad \alpha=\frac{2a-1}{\beta}.$$

For the principal parameters $a=\beta=2$, the squared dephasing envelope has the elementary closed form

$$\frac{1+\tau^{2}}{\cosh(\pi\tau/2)}.$$

The paper also proves a universal overlap law between different packet depths. If k/ℓ tends to a positive ratio, then the normalized overlap tends to

$$\operatorname{sech}^{\alpha}\left(\frac{1}{2}\log\frac{k}{\ell}\right).$$

The tilted logarithmic frequencies have explicit limiting cumulants governed by the polygamma functions. Cross-kernels between distinct Dirichlet-character sectors decay at the unconditional rate $m^{-1/(2\beta)}$ after normalization.

The signed side is encoded by a Möbius resolvent. It satisfies an exact parameter-shift equation and a Borel–Riesz bridge. Whenever a finite Hankel determinant is nonzero, a unique Padé packet model preserves the first $2m$ signed moments after m packets. Unlike the positive Jacobi models of the preceding paper, signed packet nodes need not remain in the physical interval. This failure is exhibited explicitly.

The positive kernel is the long-time autocorrelation of the signed process. The Riemann hypothesis remains equivalent to the statement that the distinguished recurrence time $t=0$ is not polynomially more coherent than the average packet power. This paper proves the complete unconditional depth-time and finite-reduction framework. It does not prove the remaining Möbius coherence bound.

All figures were generated by the accompanying Python program. The release contains 28 vector figures, matching PNG files, exact tables, 18 built-in checks, 18 independent tests, and the complete Python source.

Status and claim boundary

The paper proves the depth-time Gram representation, the exact time-average identity, the squarefree L-ratio coordinate formula, the uniform Gamma dephasing limit on compact time intervals, the principal closed-form envelope, the log-frequency cumulant limit, the cross-depth overlap law, and the character cross-kernel decay theorem. It also proves the analytic signed-resolvent representation, its parameter-shift identity, the Borel–Riesz bridge, and the finite Padé packet theorem under a nonvanishing Hankel determinant.

The positive kernel theory is unconditional. The finite signed reduction is conditional only on the explicitly stated nonbreakdown determinant. None of these results proves RH. The remaining assertion is the subpolynomial growth of the principal time-zero Möbius coherence. Historical priority and full references are reserved for the consolidated end-of-series bibliography.

The logarithmic quantum interpretation is compatible with the external 2026 construction in which the number basis carries the Hamiltonian $H\,|n\rangle = \log n\,|n\rangle$ and the phase $n^{-\mathrm{i}t}$. The present work uses that coordinate but studies an autocorrelation and scale filter that are distinct from the direct zeta observable.

Contents

Status and claim boundary i

I The signed transfer and its positive Gram lift 1

1 One signed seed and two exact evolutions 1

2 The depth–time kernel 2

3 Squarefree Dirichlet coordinates 4

II Universal depth-time asymptotics 5

4 A complex Beta asymptotic 6

5 The Gamma dephasing function 7

6 Log-frequency cumulants 11

7 The cross-depth overlap law 13

III Character packets and arithmetic decoherence 15

8 Cross-character depth-time kernels 15

IV The signed Möbius function family 17

9 The Möbius resolvent 17

10 The Borel–Riesz bridge 21

11 Signed Padé packet models 22

V Quantum and recurrence interpretation **25**

12 The number-basis operator system **25**

13 The recurrence process and the RH boundary **26**

VI Worked algorithms and verification **28**

14 Computational route **28**

15 Verification ledger **29**

A Formula registry **30**

B Implementation map **31**

C Complete Python source **32**

C.1 Reproduction 48

C.2 Source boundary 48

List of Figures

1.1 The mathematical pipeline. The signed transfer is lifted by time averaging into a positive kernel. The positive kernel has a universal Gamma limit. The signed resolvent and its finite Padé reductions preserve the coherent information that the positive average removes. 2

2.1 The exact feature map. Depth controls the active scale. Logarithmic time rotates each number-basis packet at frequency $\log n$. Their Gram contraction is the depth-time kernel. 3

2.2 A finite depth-time Gram matrix. The matrix is Hermitian positive semidefinite. The plotted eigenvalues provide a numerical certificate for the displayed packet set. 4

3.1 Two exact coordinates for one kernel. The direct squarefree sum and the finite difference of zeta ratios agree to floating-point roundoff in the worked calculation. 5

3.2 Finite-time extraction of the diagonal kernel. Increasing the observation window suppresses off-diagonal logarithmic frequencies and approaches the exact positive packet correlation. 5

5.1 The limiting dephasing trajectory. The Gamma ratio traces a zero-free curve in the complex plane. The marked point is the normalized value at zero lag. 8

5.2 Convergence to the Gamma ratio. The dephased discrete correlations approach the same real and imaginary limit as the packet depth grows. 9

5.3 The principal closed-form dephasing envelope. The Gamma-ratio modulus and the elementary hyperbolic-cosine expression coincide throughout the displayed range. 10

5.4 The phase and its derivative. The digamma function supplies the exact group-delay law of the limiting packet correlation. 11

6.1 The centered logarithmic spectrum. The finite squarefree packet histogram approaches the log-Gamma density predicted by the dephasing function. 12

6.2 Convergence of the first two cumulants. The mean and variance settle toward explicit digamma and trigamma constants. 13

7.1 The exact limiting overlap profile. Packets at comparable depths retain substantial overlap. Packets separated multiplicatively in depth become asymptotically orthogonal. 14

7.2 Finite-depth convergence. Several absolute scales approach the same ratio-dependent law. 14

7.3 The geometry on logarithmic depth. The overlap depends asymptotically only on the difference of log depths, which produces a translation-invariant ridge in scale coordinates. 15

8.1 Modulo-five character Gram matrices. The diagonal remains normalized to one while the off-diagonal sectors weaken with increasing depth. 16

8.2 The largest off-diagonal character correlation. The finite values decrease in the direction predicted by the unconditional quarter-power law. 17

9.1 The signed transfer in logarithmic time. The signal is quasiperiodic because every integer packet rotates at its own frequency $\log n$. Increasing the depth changes the active scale but not the frequency law. 18

9.2 The unit-average-power process. The positive packet energy normalizes the long-time mean power to one. The value at $t = 0$ remains a distinguished coherent recurrence. 19

9.3 Finite time-zero coherence data. The plot is a finite computational record only. It is not used to infer the asymptotic RH bound. 19

9.4 The signed resolvent. The function is analytic on the displayed side of the physical cut. Unlike the positive resolvent, it may have zeros and sign cancellations. 20

9.5 The transfer-equation residual. Three complex evaluation points verify the parameter-shift identity to the truncation accuracy of the number-basis sum. 21

10.1 The Borel–Riesz identity. The discrete depth sequence and its continuous companion are two exact coordinates of the same Möbius transfer. 22

11.1 Positive and signed packet nodes. The positive companion reduction stays in the declared spectral interval. The signed Möbius reduction can leave it because cancellation replaces positivity. 23

11.2 Moment-exact finite reduction. Four packets preserve the first eight positive or signed moments. The later moments are predictions rather than part of the exact contract. . . . 24

11.3 Positive and signed Hankel determinants. The positive determinants remain positive. The signed determinants change sign and become increasingly ill-conditioned, which exposes the nonbreakdown boundary directly. 24

12.1 The operator realization. The signed amplitude is a coherent readout. Time averaging produces the positive depth-time kernel and removes only the off-diagonal logarithmic frequencies. 26

13.1 The exact boundary. The positive depth-time theory is unconditional. Finite signed models are exact under a visible nonbreakdown condition. The principal time-zero coherence remains open. 27

13.2 The generated function family. The paper adds one positive reproducing kernel, one zero-free Gamma dephasing limit, one signed resolvent, one log-frequency law, one character cross-kernel family, and finite signed Padé packet models. 28

Part I

The signed transfer and its positive Gram lift

Chapter 1

One signed seed and two exact evolutions

Fix

$$a > 1, \qquad \beta > 0,$$

and let χ be a Dirichlet character. Set

$$y_n = 1 - \frac{1}{n^\beta}.$$

The signed transfer is

$$\boxed{C_{a,\beta,\chi}(k,t) = \sum_{n\geq 1} \mu(n)\chi(n)n^{-a-\mathrm{i}t}y_n^k.}$$

Absolute convergence follows from $a > 1$. The variable k is a discrete Euclidean depth. The variable t is logarithmic time.

The shift in k gives the exact recursion

$$\boxed{C_{a,\beta,\chi}(k+1,t) = C_{a,\beta,\chi}(k,t) - C_{a+\beta,\beta,\chi}(k,t).}$$

Indeed,

$$y_n^{k+1} = y_n^k - n^{-\beta}y_n^k.$$

Expanding the depth filter produces the finite-difference form

$$\boxed{C_{a,\beta,\chi}(k,t) = \sum_{j=0}^{k}(-1)^j\binom{k}{j}\frac{1}{L(a+\beta j+\mathrm{i}t,\chi)}.}$$

The Möbius packet is therefore a weighted finite-difference orbit of reciprocal L-values.

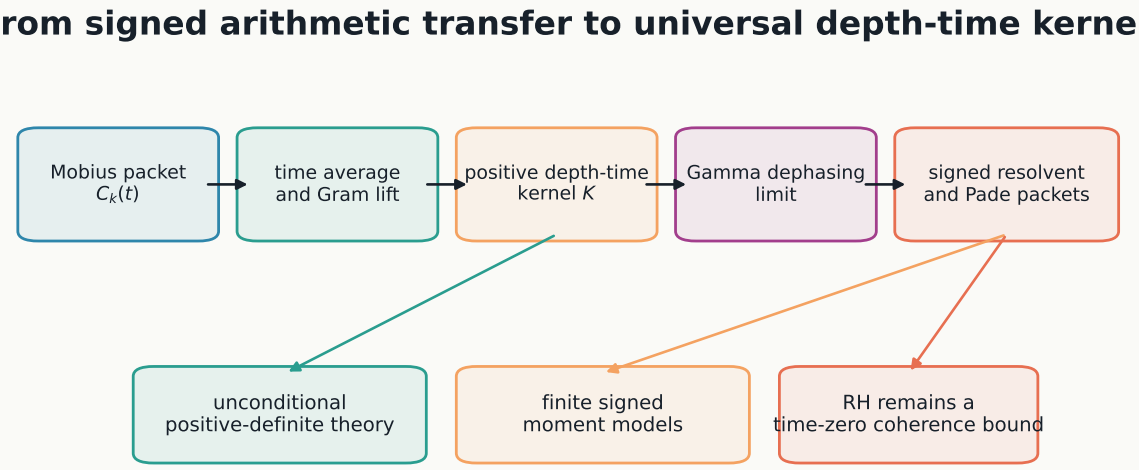


Figure 1.1: The mathematical pipeline. The signed transfer is lifted by time averaging into a positive kernel. The positive kernel has a universal Gamma limit. The signed resolvent and its finite Padé reductions preserve the coherent information that the positive average removes.

Worked example 1.1 (Principal Bernoulli–zeta orbit)**.** Take $a = \beta = 2$ and the principal character. Then

$$C_{2,2}(k,0) = \sum_{j=0}^{k} (-1)^j \binom{k}{j} \frac{1}{\zeta(2j+2)}.$$

The first values are

$$C_{2,2}(0,0) = 0.607927101854\ldots,$$
$$C_{2,2}(1,0) = -0.316011301068\ldots,$$
$$C_{2,2}(2,0) = -0.256997111725\ldots.$$

These are the Báez-Duarte coefficients in the normalization used throughout the series.

Chapter 2

The depth–time kernel

For two characters χ and ψ, define

$$\boxed{\mathcal{K}_{\chi,\psi}(k,\ell;\tau) = \sum_{n\geq 1} \mu(n)^2 \chi(n)\overline{\psi(n)} n^{-2a-\mathrm{i}\tau} y_n^{k+\ell}.}$$

When $\chi = \psi$, the weights are nonnegative.

Theorem 2.1 (Exact time-average identity)**.** *For $a > 1$,*

$$\boxed{\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} C_{a,\beta,\chi}(k,t+\tau)\overline{C_{a,\beta,\psi}(\ell,t)}\,\mathrm{d}t = \mathcal{K}_{\chi,\psi}(k,\ell;\tau).}$$

Proof. Expand the product into the absolutely convergent double series. The time average of the factor

$$\mathrm{e}^{-\mathrm{i}t(\log n-\log m)}$$

is one when $n = m$ and tends to zero when $n \neq m$. Only the diagonal terms remain. □

Theorem 2.2 (Positive-definite packet kernel)**.** *For one fixed character χ, the function*

$$((k,t),(\ell,s)) \longmapsto \mathcal{K}_{\chi,\chi}(k,\ell;t-s)$$

is positive definite.

Proof. Define the feature vector

$$v_{k,t}(n) = \frac{\mu(n)\,|\chi(n)|}{n^a} y_n^k n^{-\mathrm{i}t}.$$

Then

$$\mathcal{K}_{\chi,\chi}(k,\ell;t-s) = \sum_n v_{k,t}(n)\overline{v_{\ell,s}(n)}.$$

Every finite Gram sum is therefore nonnegative. □

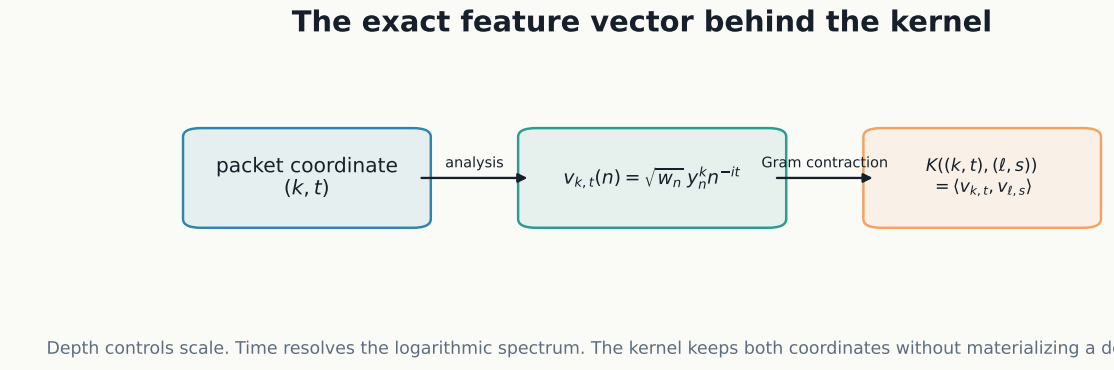


Figure 2.1: The exact feature map. Depth controls the active scale. Logarithmic time rotates each number-basis packet at frequency $\log n$. Their Gram contraction is the depth-time kernel.

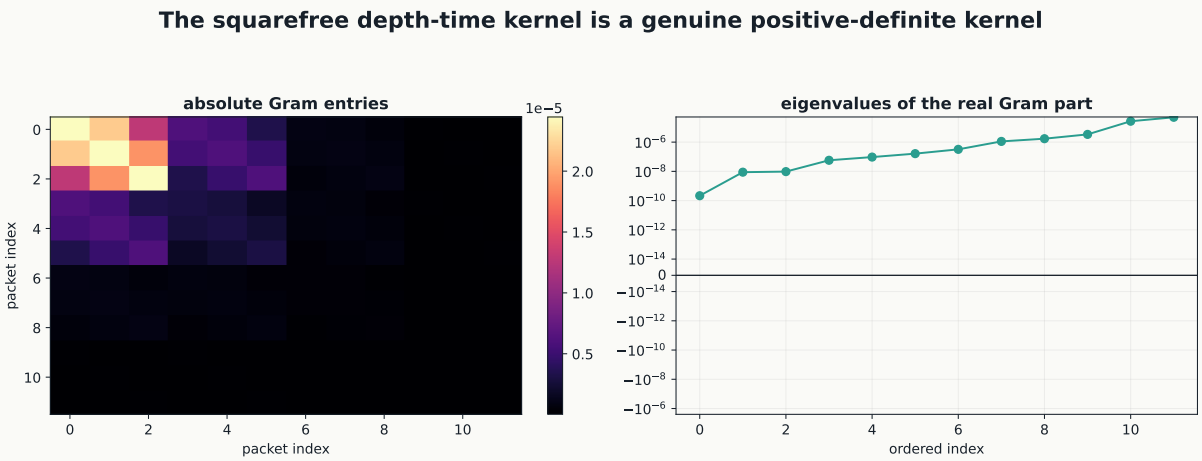


Figure 2.2: A finite depth-time Gram matrix. The matrix is Hermitian positive semidefinite. The plotted eigenvalues provide a numerical certificate for the displayed packet set.

Chapter 3

Squarefree Dirichlet coordinates

Let

$$\xi = \chi\overline{\psi}$$

and define

$$D_\xi(s) = \sum_{n\geq 1} \frac{\mu(n)^2\xi(n)}{n^s}.$$

For $\Re s > 1$,

$$\boxed{D_\xi(s) = \prod_p \left(1 + \frac{\xi(p)}{p^s}\right) = \frac{L(s,\xi)}{L(2s,\xi^2)}.}$$

Theorem 3.1 (Finite-difference coordinate formula)**.** *Let $m = k + \ell$. Then*

$$\boxed{\mathcal{K}_{\chi,\psi}(k,\ell;\tau) = \sum_{j=0}^{m} (-1)^j \binom{m}{j} D_\xi(2a + \beta j + \mathrm{i}\tau).}$$

Proof. Expand

$$y_n^m = \left(1 - n^{-\beta}\right)^m = \sum_{j=0}^{m} (-1)^j \binom{m}{j} n^{-\beta j}$$

and interchange the finite sum with the absolutely convergent Dirichlet series. □

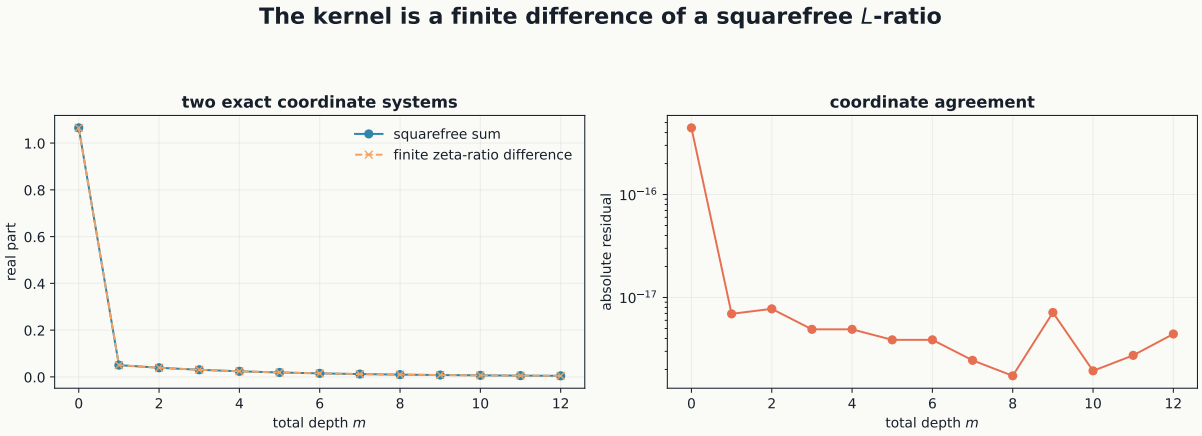


Figure 3.1: Two exact coordinates for one kernel. The direct squarefree sum and the finite difference of zeta ratios agree to floating-point roundoff in the worked calculation.

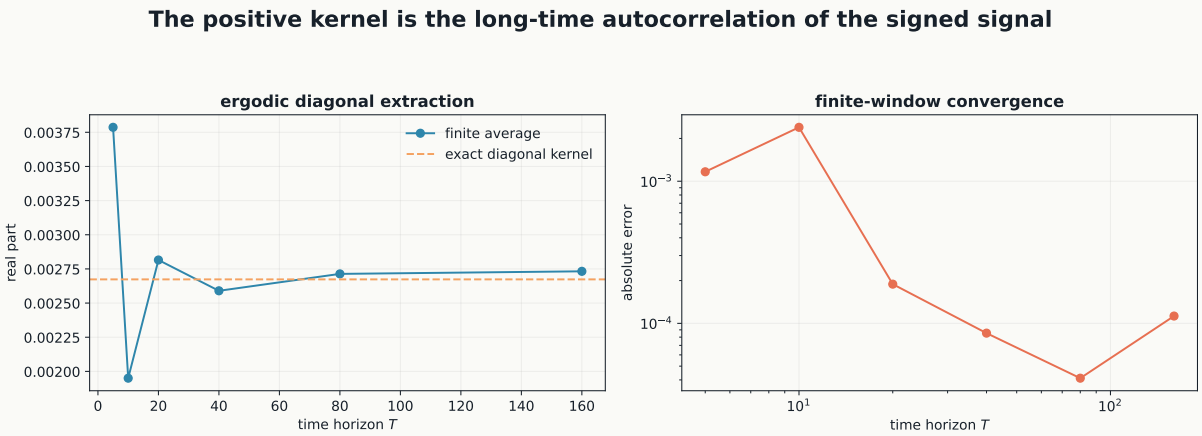


Figure 3.2: Finite-time extraction of the diagonal kernel. Increasing the observation window suppresses off-diagonal logarithmic frequencies and approaches the exact positive packet correlation.

Part II

Universal depth-time asymptotics

Chapter 4

A complex Beta asymptotic

Assume $\chi = \psi$ has conductor q. Then

$$|\chi(n)|^2 = \mathbf{1}_{(n,q)=1}.$$

Let

$$A_q(X) = \sum_{\substack{n \le X \\ (n,q)=1}} \mu(n)^2.$$

The squarefree counting law gives

$$A_q(X) = \delta_q X + O_q(X^{1/2}),$$

where

$$\delta_q = \frac{1}{\zeta(2)} \prod_{p \mid q} \frac{p}{p+1}.$$

Set

$$\alpha = \frac{2a-1}{\beta}$$

and

$$\lambda(\tau) = \alpha + \frac{\mathrm{i}\tau}{\beta}.$$

Theorem 4.1 (Uniform complex packet asymptotic)**.** *For every fixed $T > 0$, uniformly for $|\tau| \le T$,*

$$\boxed{\mathcal{K}_{\chi,\chi}(k,\ell;\tau) = \frac{\delta_q}{\beta} B(\lambda(\tau), k+\ell+1) + O_{a,\beta,q,T}\left((k+\ell)^{-(2a-1/2)/\beta}\right).}$$

Consequently,

$$\boxed{\mathcal{K}_{\chi,\chi}(k,\ell;\tau) \sim \frac{\delta_q}{\beta} \Gamma\left(\alpha + \frac{\mathrm{i}\tau}{\beta}\right) (k+\ell)^{-\alpha - \mathrm{i}\tau/\beta}.}$$

Proof. Apply partial summation to the function

$$f_m(x) = x^{-2a-\mathrm{i}\tau} \left(1 - x^{-\beta}\right)^m, \qquad m = k + \ell.$$

The main term is

$$\delta_q \int_1^\infty f_m(x)\, \mathrm{d}x.$$

With $u = x^{-\beta}$, this becomes

$$\frac{\delta_q}{\beta} \int_0^1 u^{\lambda(\tau)-1} (1-u)^m \, \mathrm{d}u = \frac{\delta_q}{\beta} B(\lambda(\tau), m+1).$$

The square-root error in $A_q(X)$, together with one derivative of f_m, contributes

$$O(m^{-(2a-1/2)/\beta})$$

uniformly on compact τ intervals. The Gamma asymptotic follows from the standard Beta quotient. □

Chapter 5

The Gamma dephasing function

Define

$$\boxed{\mathcal{D}_{\alpha,\beta}(\tau) = \frac{\Gamma(\alpha + \mathrm{i}\tau/\beta)}{\Gamma(\alpha)}.}$$

This is the characteristic function of $\beta^{-1} \log U$ when

$$U \sim \mathrm{Gamma}(\alpha, 1).$$

It is therefore positive definite. Since the Gamma function has no zeros,

$$\boxed{\mathcal{D}_{\alpha,\beta}(\tau) \neq 0 \quad \text{for every real } \tau.}$$

Corollary 5.1 (Equal-depth dephasing)**.** *Let*

$$A_k(\tau) = \frac{\mathcal{K}_{\chi,\chi}(k, k; \tau)}{\mathcal{K}_{\chi,\chi}(k, k; 0)}.$$

Then

$$\boxed{(2k)^{\mathrm{i}\tau/\beta} A_k(\tau) \longrightarrow \mathcal{D}_{\alpha,\beta}(\tau)}$$

uniformly on compact real τ intervals.

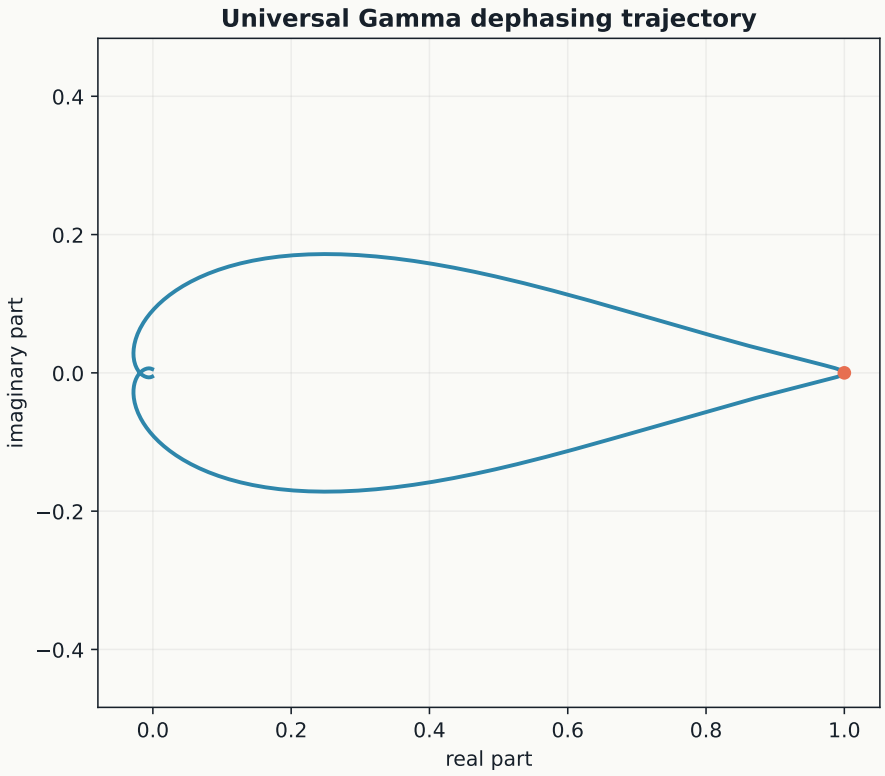


Figure 5.1: The limiting dephasing trajectory. The Gamma ratio traces a zero-free curve in the complex plane. The marked point is the normalized value at zero lag.

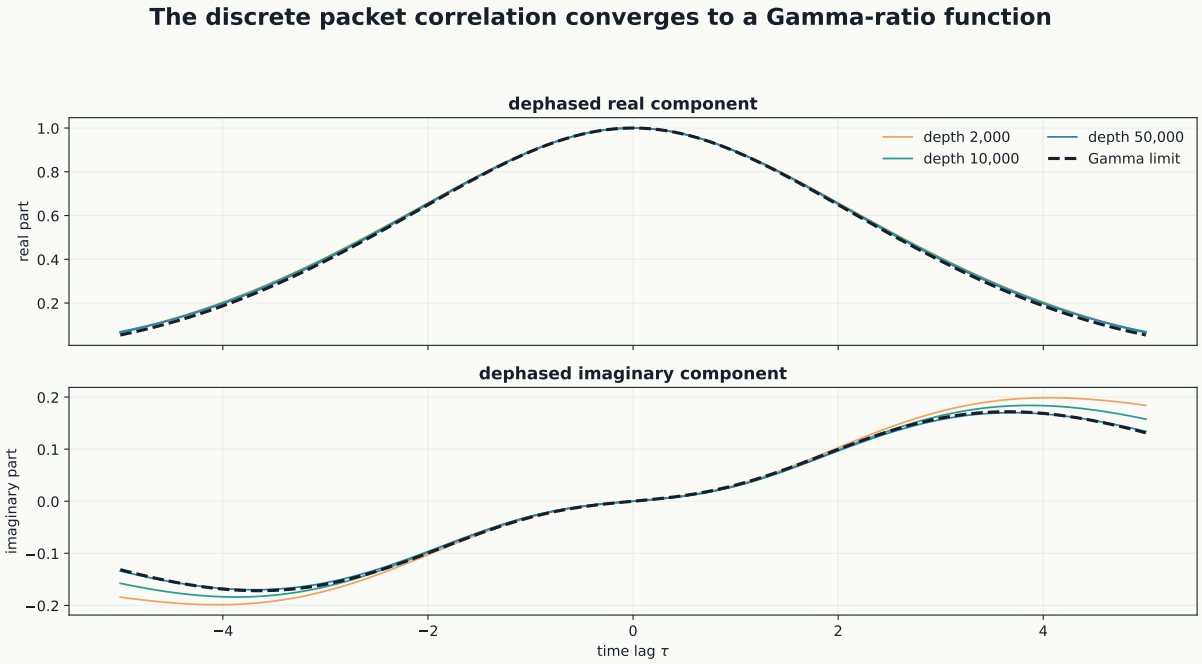


Figure 5.2: Convergence to the Gamma ratio. The dephased discrete correlations approach the same real and imaginary limit as the packet depth grows.

Worked example 5.2 (Closed principal envelope)**.** For

$$a = \beta = 2, \qquad \alpha = \frac{3}{2},$$

we have

$$\mathcal{D}_{3/2,2}(\tau) = \frac{\Gamma(3/2 + \mathrm{i}\tau/2)}{\Gamma(3/2)}.$$

Using

$$|\Gamma(1/2 + \mathrm{i}y)|^2 = \frac{\pi}{\cosh(\pi y)},$$

one obtains

$$\boxed{\left|\mathcal{D}_{3/2,2}(\tau)\right|^2 = \frac{1+\tau^2}{\cosh(\pi\tau/2)}.}$$

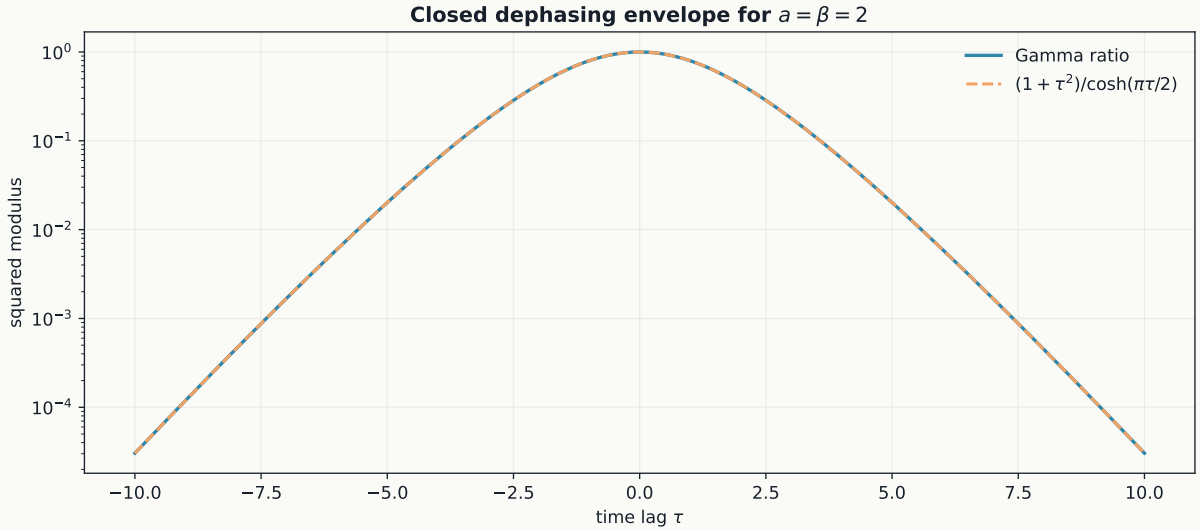


Figure 5.3: The principal closed-form dephasing envelope. The Gamma-ratio modulus and the elementary hyperbolic-cosine expression coincide throughout the displayed range.

The phase derivative also has a closed law:

$$\boxed{\frac{\mathrm{d}}{\mathrm{d}\tau}\arg\mathcal{D}_{\alpha,\beta}(\tau) = \frac{1}{\beta}\Re\psi\left(\alpha + \frac{\mathrm{i}\tau}{\beta}\right).}$$

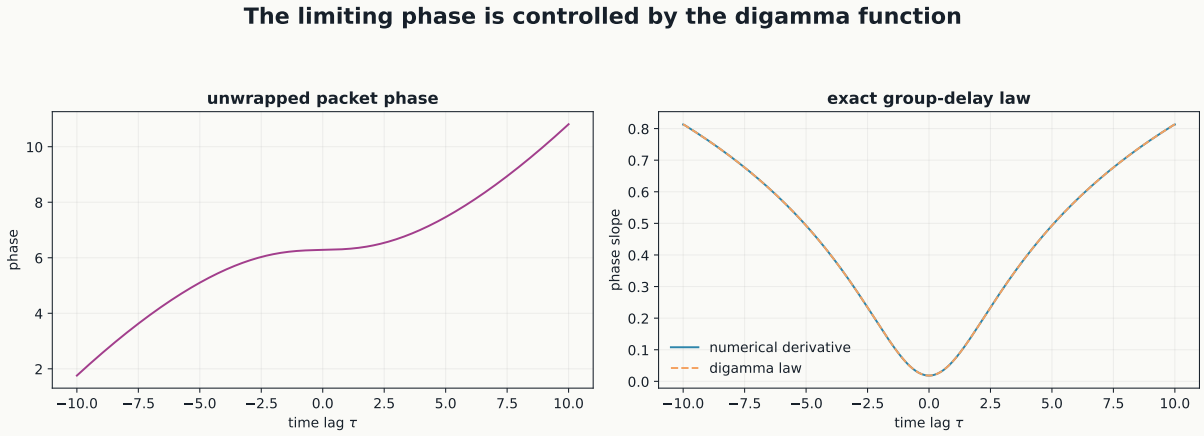


Figure 5.4: The phase and its derivative. The digamma function supplies the exact group-delay law of the limiting packet correlation.

Table 5.1: Selected values of the limiting Gamma dephasing function for $a = \beta = 2$.

τ	Real part	Imaginary part	Squared modulus
-4	0.18721515	-0.16865147	0.06349283
-2	0.64917367	-0.09953509	0.43133369
-1	0.89225332	-0.03094590	0.79707363
0	1.00000000	0.00000000	1.00000000
1	0.89225332	0.03094590	0.79707363
2	0.64917367	0.09953509	0.43133369
4	0.18721515	0.16865147	0.06349283

Chapter 6

Log-frequency cumulants

Normalize the depth-m positive weights by

$$\mathbb{P}_m(n) = \frac{\mu(n)^2 \, |\chi(n)|^2 \, n^{-2a} y_n^m}{\mathcal{K}_{\chi,\chi}(m, 0; 0)}.$$

Define the centered logarithmic frequency

$$Z_m = \log n - \frac{1}{\beta} \log m.$$

The preceding characteristic-function limit gives

$$Z_m \Longrightarrow -\frac{1}{\beta} \log U, \qquad U \sim \text{Gamma}(\alpha, 1).$$

Corollary 6.1 (Polygamma cumulants)**.** *For every fixed order for which moment convergence is taken,*

$$\boxed{\kappa_r(Z_m) \longrightarrow \frac{(-1)^r}{\beta^r} \psi_{r-1}(\alpha).}$$

In particular,

$$\boxed{\mathbb{E}Z_m \longrightarrow -\frac{\psi(\alpha)}{\beta}}$$

and

$$\boxed{\mathrm{Var}(Z_m) \longrightarrow \frac{\psi_1(\alpha)}{\beta^2}.}$$

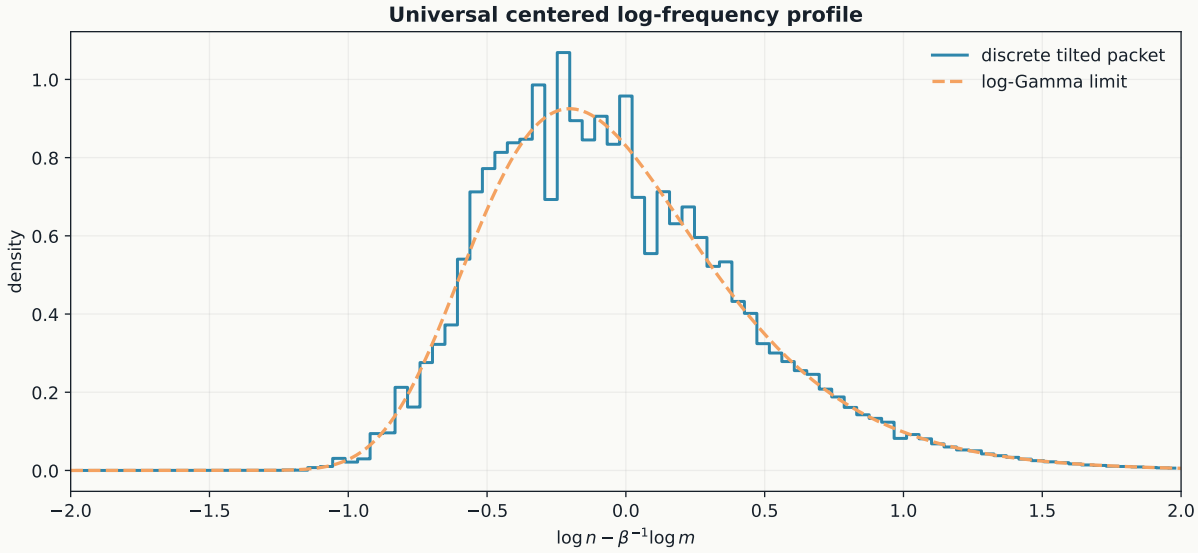


Figure 6.1: The centered logarithmic spectrum. The finite squarefree packet histogram approaches the log-Gamma density predicted by the dephasing function.

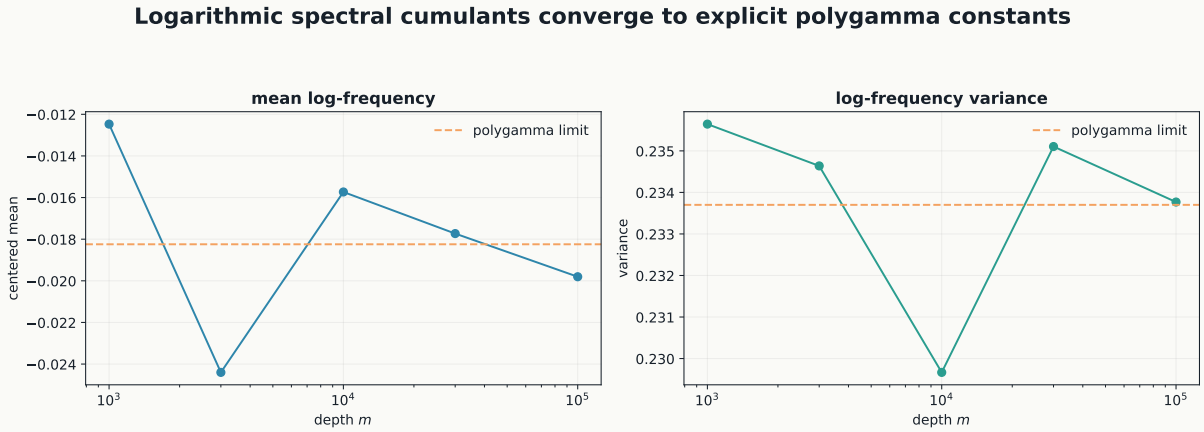


Figure 6.2: Convergence of the first two cumulants. The mean and variance settle toward explicit digamma and trigamma constants.

Chapter 7

The cross-depth overlap law

At zero time lag, define

$$\mathcal{O}(k,\ell) = \frac{\mathcal{K}_{\chi,\chi}(k,\ell;0)}{\sqrt{\mathcal{K}_{\chi,\chi}(k,k;0)\mathcal{K}_{\chi,\chi}(\ell,\ell;0)}}.$$

Theorem 7.1 (Universal scale overlap)**.** *If $k,\ell \to \infty$ and $k/\ell \to r \in (0,\infty)$, then*

$$\boxed{\mathcal{O}(k,\ell) \longrightarrow \left(\frac{2\sqrt{r}}{1+r}\right)^{\alpha} = \operatorname{sech}^{\alpha}\left(\frac{1}{2}\log r\right).}$$

Proof. The zero-lag kernel is regularly varying:

$$\mathcal{K}(k,\ell;0) \sim A(k+\ell)^{-\alpha}.$$

The two diagonal norms satisfy

$$\mathcal{K}(k,k;0) \sim A(2k)^{-\alpha}$$

and the analogous formula for ℓ. Taking the ratio gives the result. □

Worked example 7.2 (A four-to-one depth ratio)**.** For $a = \beta = 2$, we have $\alpha = 3/2$. When $k/\ell = 4$,

$$\mathcal{O}(k,\ell) \longrightarrow \left(\frac{4}{5}\right)^{3/2} = 0.7155417528\ldots.$$

The finite calculation at depths $k = 40000$ and $\ell = 10000$ gives a value within two percent of this limit.

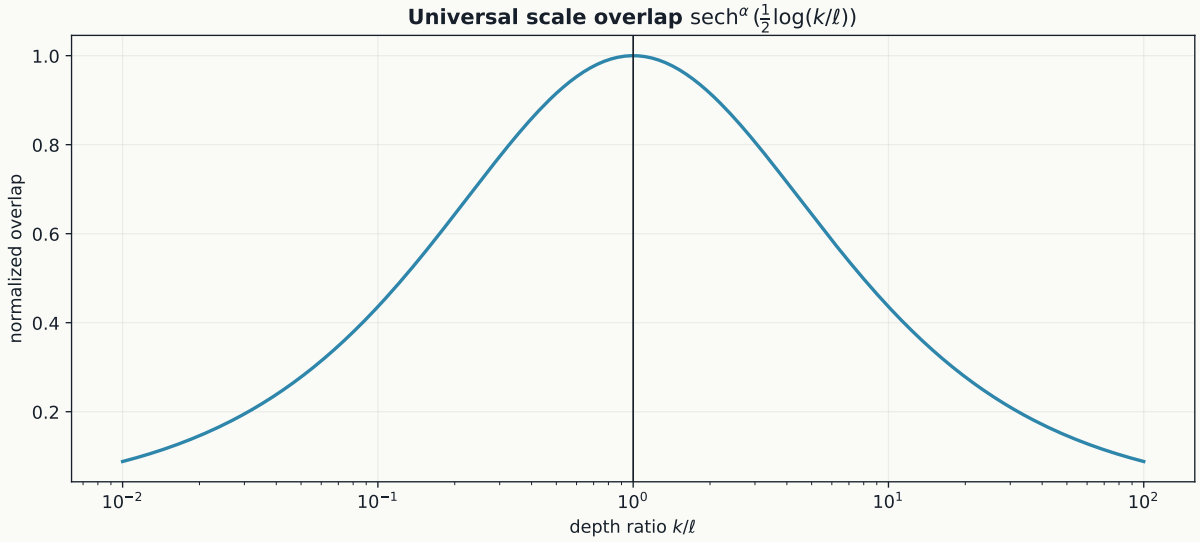


Figure 7.1: The exact limiting overlap profile. Packets at comparable depths retain substantial overlap. Packets separated multiplicatively in depth become asymptotically orthogonal.

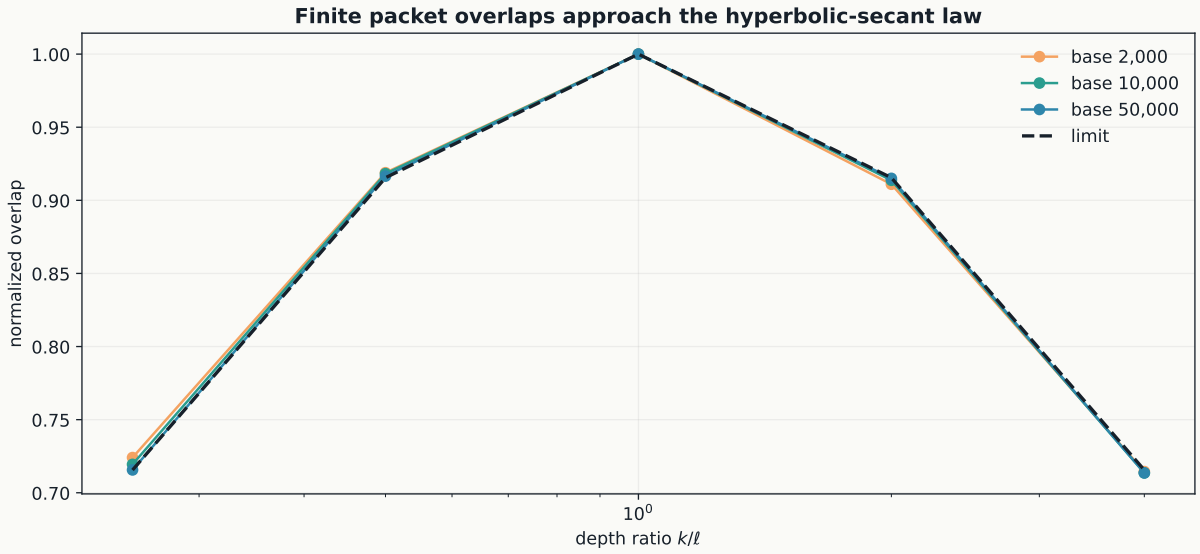


Figure 7.2: Finite-depth convergence. Several absolute scales approach the same ratio-dependent law.

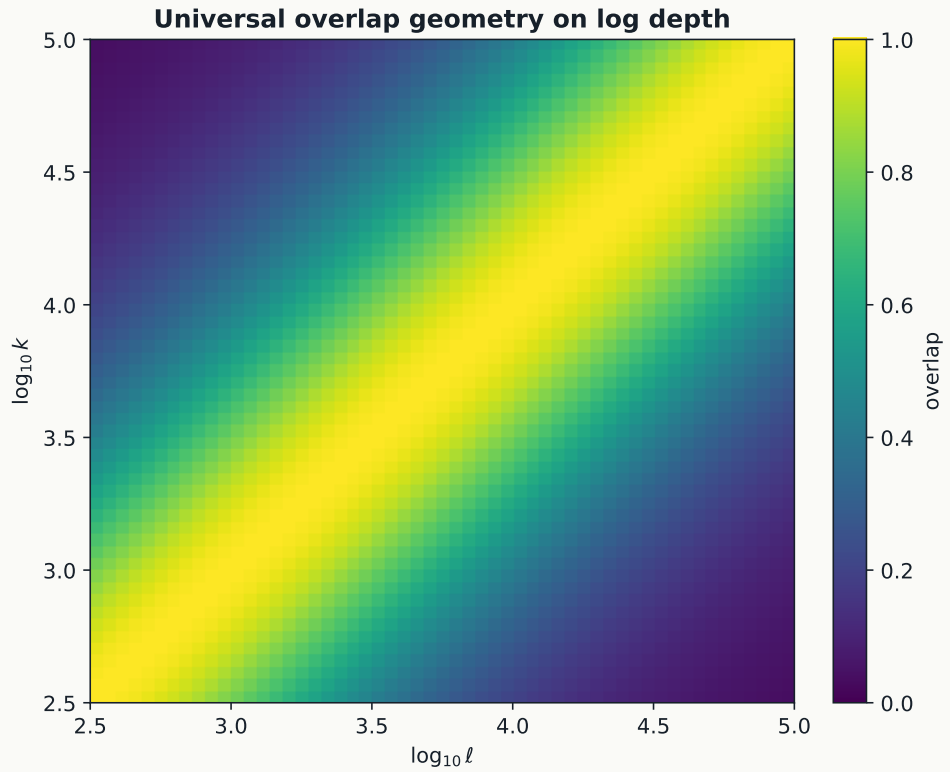


Figure 7.3: The geometry on logarithmic depth. The overlap depends asymptotically only on the difference of log depths, which produces a translation-invariant ridge in scale coordinates.

Part III

Character packets and arithmetic decoherence

Chapter 8

Cross-character depth-time kernels

For two characters modulo the same conductor, the cross-kernel has the exact time-average interpretation already proved. If

$$\xi = \chi\overline{\psi}$$

is nonprincipal, its squarefree partial sums satisfy

$$\sum_{n\le X} \mu(n)^2\xi(n) = O_q(X^{1/2}).$$

This follows from

$$\mu(n)^2 = \sum_{d^2\mid n} \mu(d)$$

and the bounded complete-period sums of a nonprincipal character.

Theorem 8.1 (Character depth-time decoherence)**.** *If $\chi \neq \psi$, then uniformly for $|\tau| \leq T$,*

$$\boxed{\mathcal{K}_{\chi,\psi}(k,\ell;\tau) = O_{a,\beta,q,T}\left((k+\ell)^{-(2a-1/2)/\beta}\right).}$$

After diagonal normalization,

$$\boxed{\frac{\mathcal{K}_{\chi,\psi}(k,\ell;\tau)}{\sqrt{\mathcal{K}_{\chi,\chi}(k,k;0)\mathcal{K}_{\psi,\psi}(\ell,\ell;0)}} = O\left((k+\ell)^{-1/(2\beta)}\right)}$$

whenever the two depths remain comparable.

Proof. Use partial summation with the nonprincipal square-root bound instead of the principal density term. The same scaled derivative estimate used in the complex Beta theorem yields the stated power. Dividing by the two diagonal norms removes $(k+\ell)^{-\alpha}$ and leaves the factor $(k+\ell)^{-1/(2\beta)}$. □

For $a = \beta = 2$, the unconditional normalized decay rate is

$$(k+\ell)^{-1/4}.$$

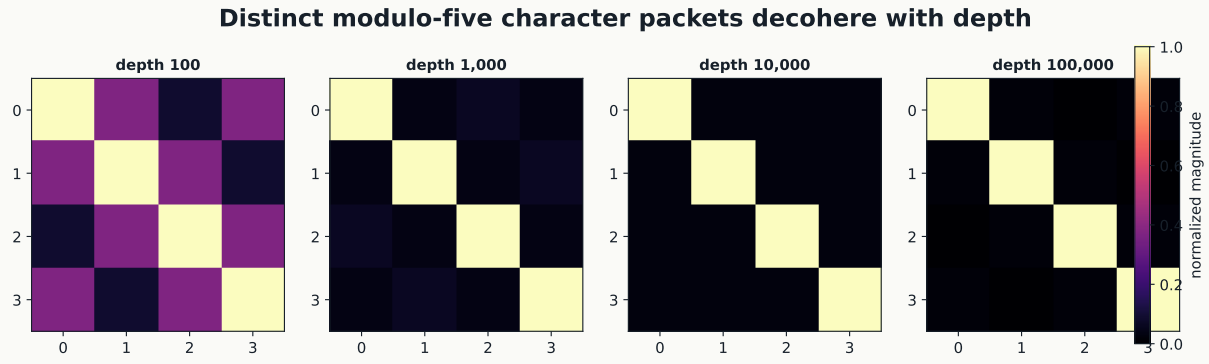


Figure 8.1: Modulo-five character Gram matrices. The diagonal remains normalized to one while the off-diagonal sectors weaken with increasing depth.

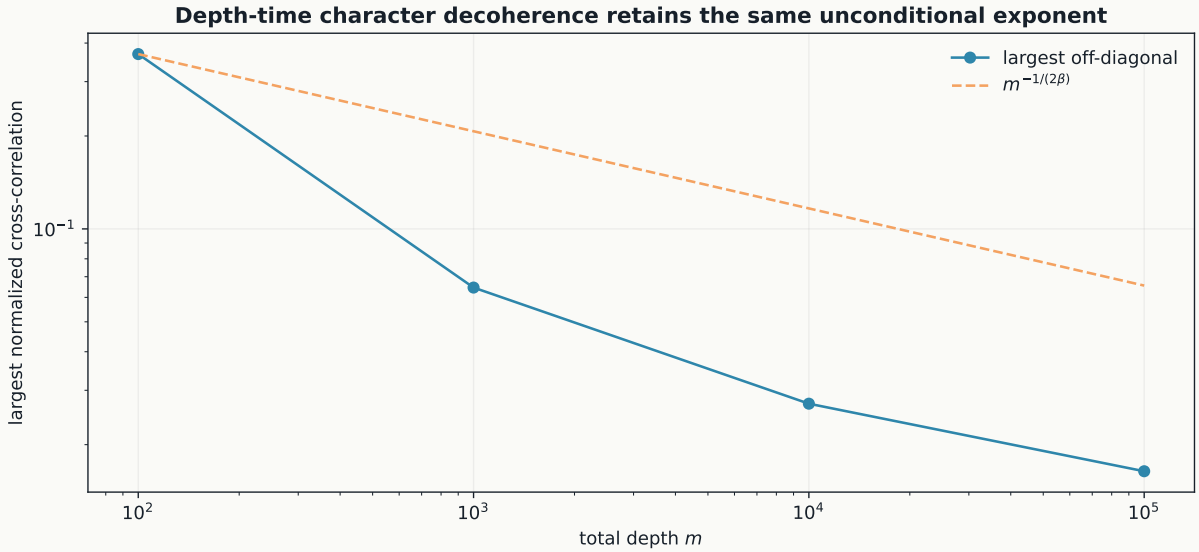


Figure 8.2: The largest off-diagonal character correlation. The finite values decrease in the direction predicted by the unconditional quarter-power law.

Part IV

The signed Möbius function family

Chapter 9

The Möbius resolvent

Define the signed resolvent

$$\boxed{\mathcal{C}_{a,\beta,\chi}(z,t) = \sum_{k=0}^{\infty} C_{a,\beta,\chi}(k,t) z^k.}$$

For $|z| < 1$, geometric summation gives

$$\boxed{\mathcal{C}_{a,\beta,\chi}(z,t) = \sum_{n\geq 1} \frac{\mu(n)\chi(n) n^{-a-\mathrm{i}t}}{1 - z(1-n^{-\beta})}.}$$

Since $a > 1$, this sum extends locally uniformly to

$$\mathbb{C} \setminus [1, \infty).$$

Unlike the positive Stieltjes resolvent of Paper III, the signed resolvent is not asserted to be zero-free.

In the smaller domain where the following geometric expansion converges,

$$\boxed{\mathcal{C}_{a,\beta,\chi}(z,t) = \frac{1}{1-z} \sum_{j=0}^{\infty} \left(-\frac{z}{1-z}\right)^j \frac{1}{L(a+\beta j + \mathrm{i}t, \chi)}.}$$

Theorem 9.1 (Parameter-shift equation)**.** *The signed resolvent satisfies*

$$\boxed{(1-z)\mathcal{C}_{a,\beta,\chi}(z,t)+z\mathcal{C}_{a+\beta,\beta,\chi}(z,t)=\frac{1}{L(a+\mathrm{i}t,\chi)}.}$$

Proof. Multiply the depth recursion by z^k and sum over $k \geq 0$. The left shift contributes

$$\frac{\mathcal{C}_a - C_a(0,t)}{z}.$$

Since

$$C_a(0,t) = L(a+\mathrm{i}t,\chi)^{-1},$$

rearrangement gives the result. □

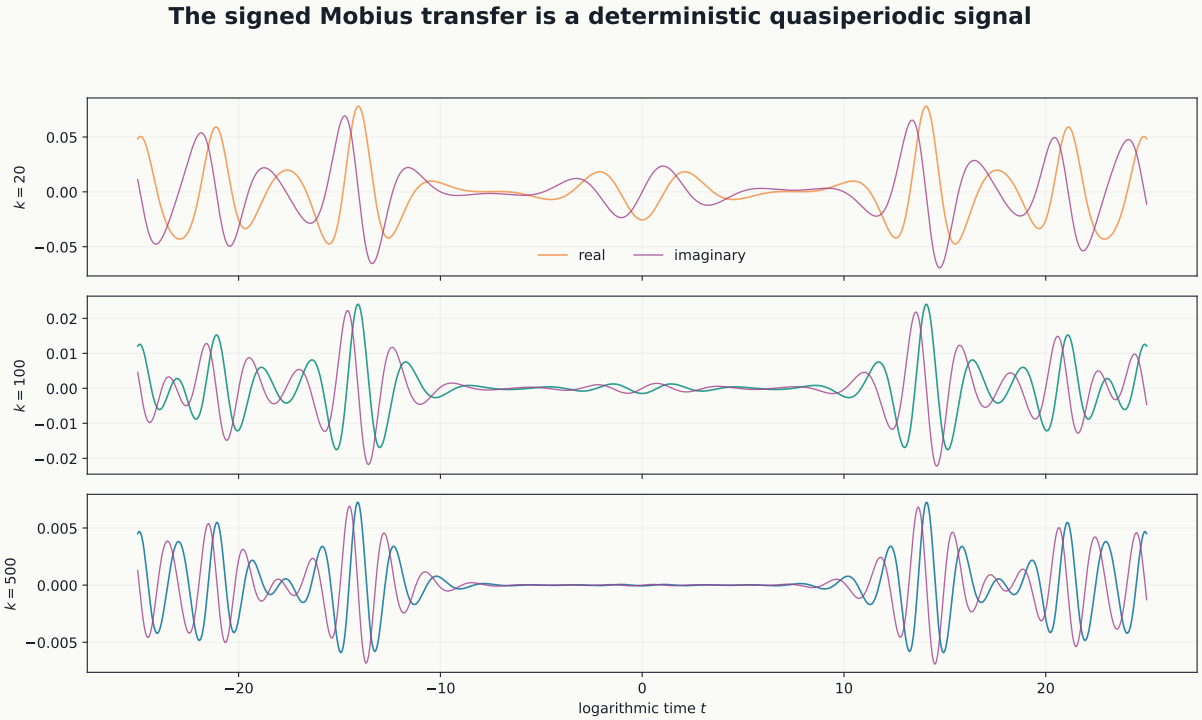


Figure 9.1: The signed transfer in logarithmic time. The signal is quasiperiodic because every integer packet rotates at its own frequency $\log n$. Increasing the depth changes the active scale but not the frequency law.

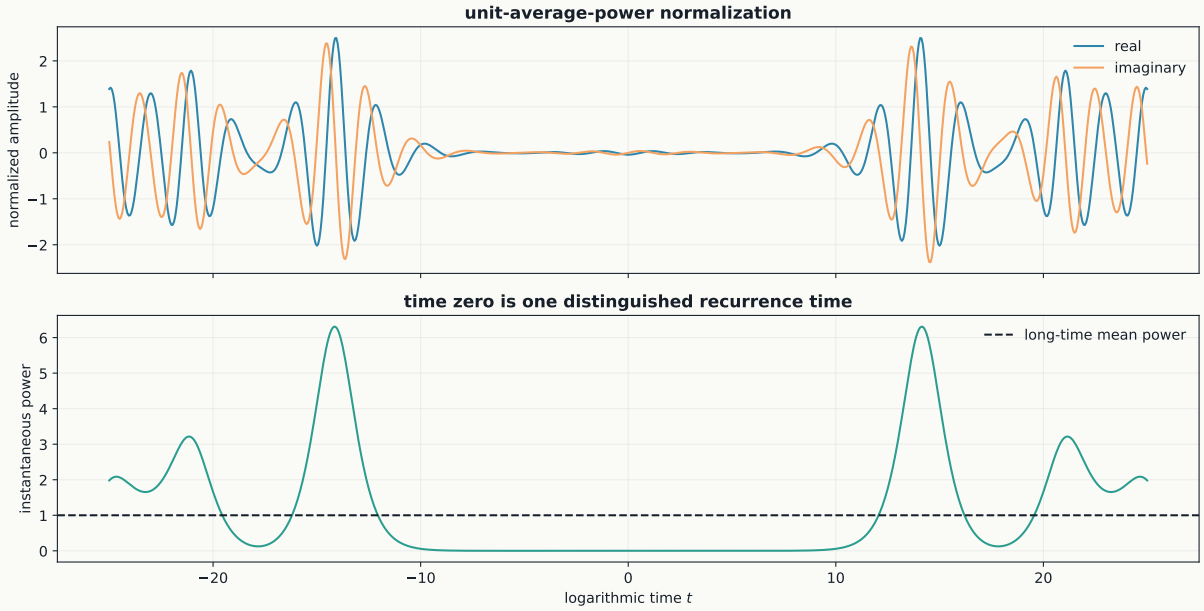


Figure 9.2: The unit-average-power process. The positive packet energy normalizes the long-time mean power to one. The value at $t = 0$ remains a distinguished coherent recurrence.

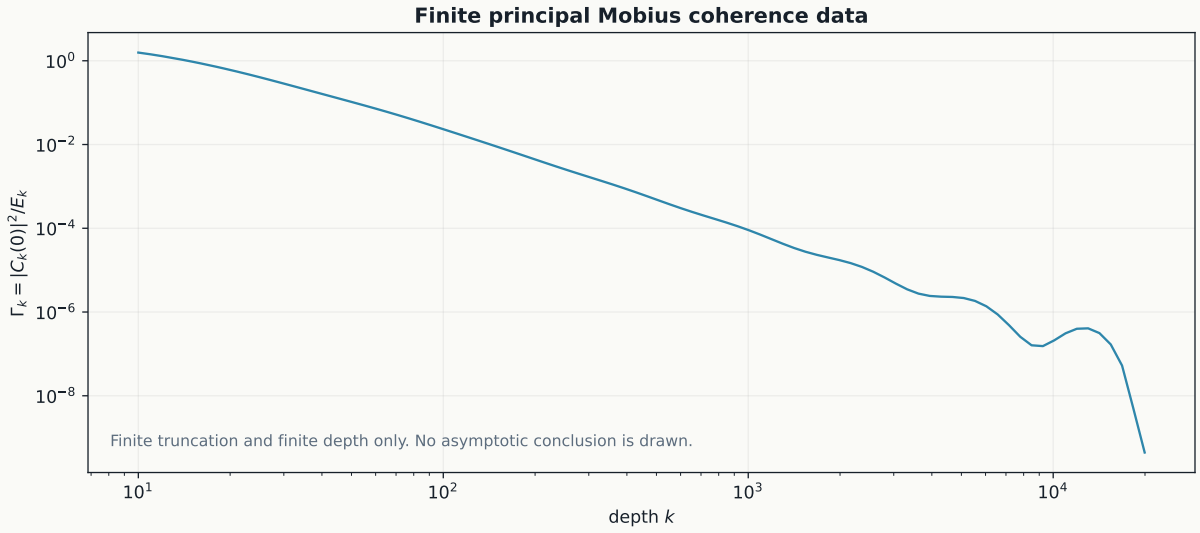


Figure 9.3: Finite time-zero coherence data. The plot is a finite computational record only. It is not used to infer the asymptotic RH bound.

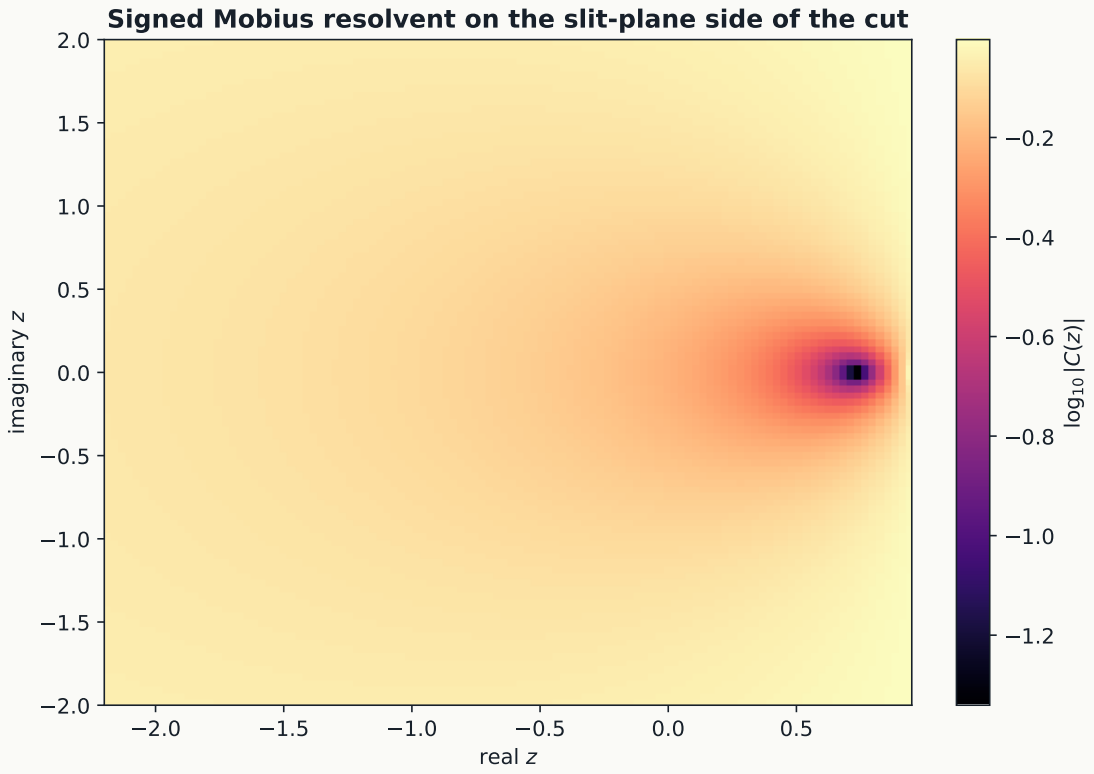


Figure 9.4: The signed resolvent. The function is analytic on the displayed side of the physical cut. Unlike the positive resolvent, it may have zeros and sign cancellations.

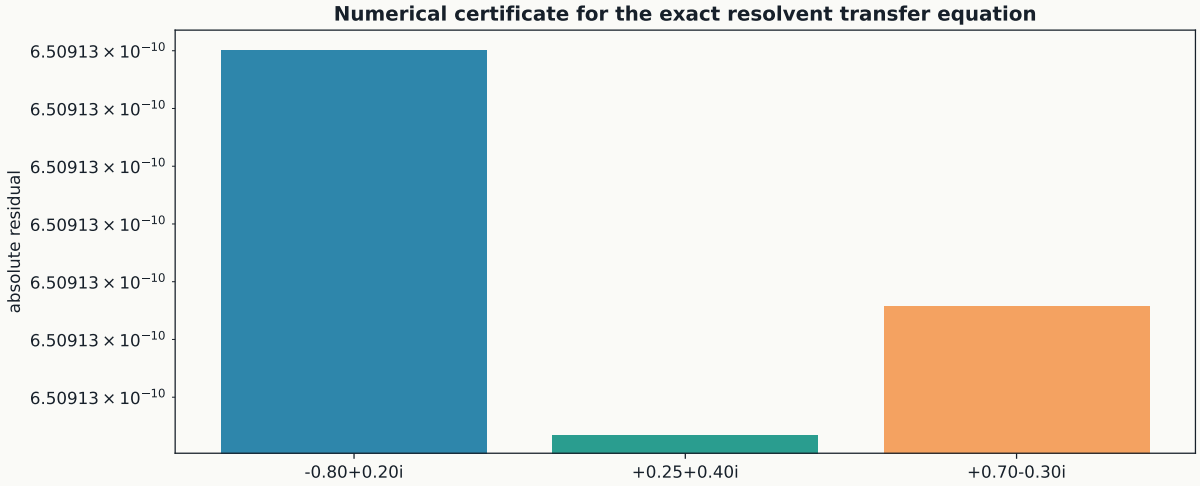


Figure 9.5: The transfer-equation residual. Three complex evaluation points verify the parameter-shift identity to the truncation accuracy of the number-basis sum.

Chapter 10

The Borel–Riesz bridge

The exponential generating function of the signed depth sequence is

$$\sum_{k=0}^{\infty} C_{a,\beta,\chi}(k,t)\frac{x^k}{k!} = \sum_{n\geq 1} \frac{\mu(n)\chi(n)}{n^{a+it}} \mathrm{e}^{x(1-n^{-\beta})}.$$

Define

$$\mathcal{R}_{a,\beta,\chi}(x,t) = x \sum_{n\geq 1} \frac{\mu(n)\chi(n)}{n^{a+it}} \mathrm{e}^{-x/n^{\beta}}.$$

Then

$$\boxed{\sum_{k=0}^{\infty} C_{a,\beta,\chi}(k,t)\frac{x^k}{k!} = \frac{\mathrm{e}^x}{x}\mathcal{R}_{a,\beta,\chi}(x,t).}$$

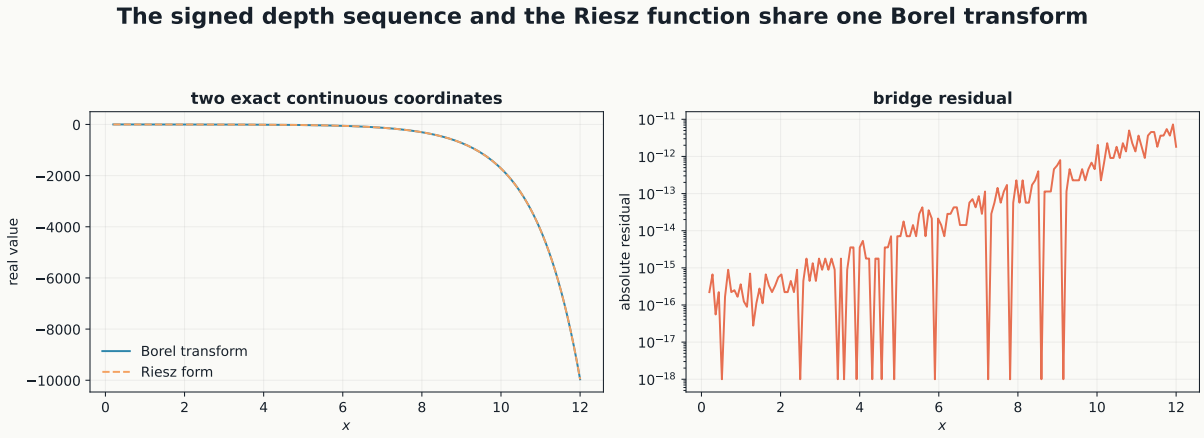


Figure 10.1: The Borel–Riesz identity. The discrete depth sequence and its continuous companion are two exact coordinates of the same Möbius transfer.

Chapter 11

Signed Padé packet models

Let

$$c_j = C_{a,\beta,\chi}(j,t)$$

and define the Hankel determinant

$$\Delta_m(t) = \det[c_{i+j}]_{i,j=0}^{m-1}.$$

Theorem 11.1 (Finite signed Padé reduction)**.** *Assume*

$$\Delta_m(t) \neq 0.$$

Then there is a unique rational function

$$\boxed{\mathcal{C}_m(z,t) = \frac{P_{m-1}(z,t)}{Q_m(z,t)}, \qquad Q_m(0,t) = 1,}$$

such that

$$\mathcal{C}_{a,\beta,\chi}(z,t) - \mathcal{C}_m(z,t) = O(z^{2m}).$$

If the poles of Q_m^{-1} are simple, then

$$\boxed{\mathcal{C}_m(z,t) = \sum_{r=1}^{m} \frac{\lambda_r(t)}{1 - z\xi_r(t)}}$$

and

$$\boxed{C_{a,\beta,\chi}(j,t) = \sum_{r=1}^{m} \lambda_r(t)\xi_r(t)^j, \qquad 0 \leq j \leq 2m-1.}$$

Proof. Write

$$Q_m(z) = 1 + q_1 z + \cdots + q_m z^m.$$

The conditions on coefficients z^m through z^{2m-1} form a linear system whose matrix is a column reversal of the Hankel matrix $[c_{i+j}]$. The determinant assumption gives a unique denominator. The numerator is then the polynomial part of $Q_m\mathcal{C}$ through degree $m-1$. This proves the moment contract. Simple poles give the partial-fraction form. □

Worked example 11.2 (The first signed packet leaves the physical interval)**.** For the principal case,

$$\xi_1 = \frac{c_1}{c_0} = -0.5198177546\ldots.$$

The positive packet spectrum belongs to $[0, 1)$. The first signed effective node is already negative. This shows that the signed reduction is an indefinite moment problem rather than a positive one.

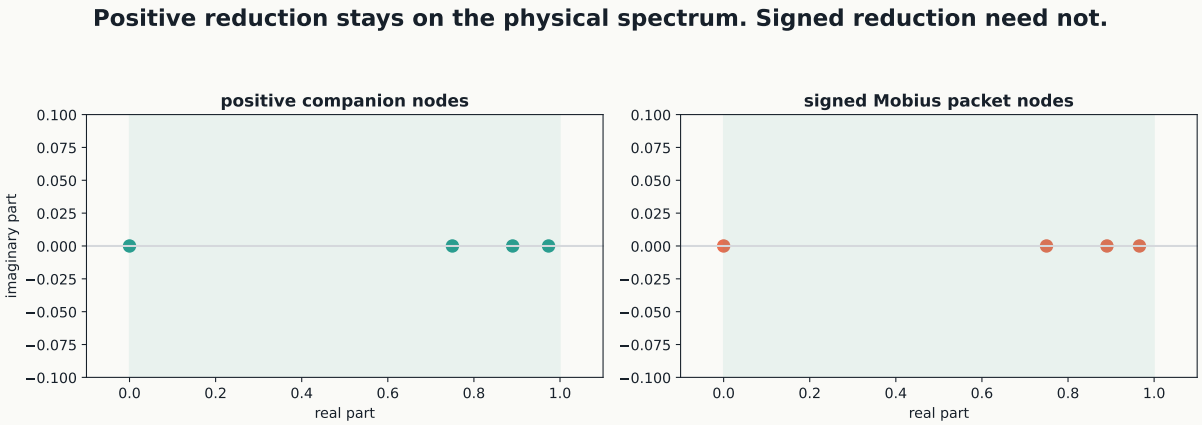


Figure 11.1: Positive and signed packet nodes. The positive companion reduction stays in the declared spectral interval. The signed Möbius reduction can leave it because cancellation replaces positivity.

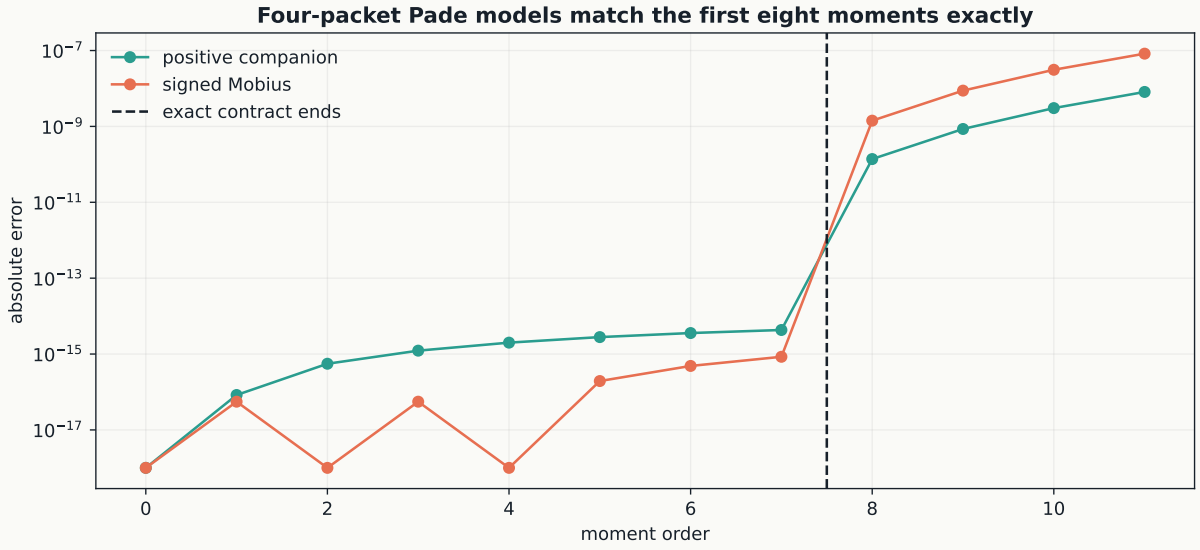


Figure 11.2: Moment-exact finite reduction. Four packets preserve the first eight positive or signed moments. The later moments are predictions rather than part of the exact contract.

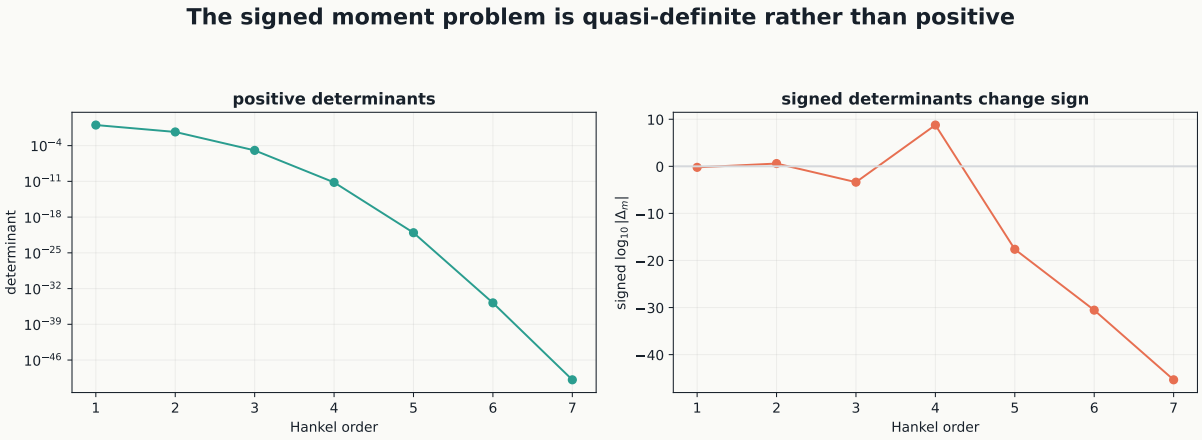


Figure 11.3: Positive and signed Hankel determinants. The positive determinants remain positive. The signed determinants change sign and become increasingly ill-conditioned, which exposes the nonbreakdown boundary directly.

Table 11.1: Four-packet Padé nodes and weights for the positive companion and signed Möbius sequences.

Family	Node real	Node imag.	Weight real	Weight imag.
positive	0.000000000	0.00e+00	1.000000003	-2.17e-17
positive	0.750017741	-6.47e-15	0.062527773	-3.78e-15
positive	0.890070247	9.21e-15	0.012629961	6.89e-15
positive	0.973460741	-1.13e-15	0.002770400	-3.09e-15
signed	-0.000000003	-0.00e+00	0.999999972	1.68e-16
signed	0.750048271	-1.13e-14	-0.250305329	2.33e-14
signed	0.890460587	1.78e-14	-0.114859170	-5.74e-14
signed	0.965969069	-2.28e-15	-0.026908372	3.40e-14

Part V

Quantum and recurrence interpretation

Chapter 12

The number-basis operator system

Work in the number basis indexed by $n \geq 2$ and define

$$H\,|n\rangle = \log n\,|n\rangle\,,$$

$$K_\beta\,|n\rangle = -\log(1-n^{-\beta})\,|n\rangle\,,$$

and

$$\mathcal{M}_\chi\,|n\rangle = \mu(n)\chi(n)\,|n\rangle\,.$$

These three diagonal operators commute. Let

$$|\phi_a\rangle = \sum_{n\geq 2} n^{-a/2}\,|n\rangle\,.$$

Then the exact operator identity is

$$\boxed{C_{a,\beta,\chi}(k,t) = \mathbf{1}_{\{k=0\}} + \langle\phi_a|\,\mathcal{M}_\chi \mathrm{e}^{-kK_\beta}\mathrm{e}^{-\mathrm{i}tH}\,|\phi_a\rangle\,.}$$

The isolated term is the $n=1$ packet. It contributes at depth zero and vanishes for every positive depth. Keeping it separate avoids assigning an infinite value to K_β at $n=1$. The external quantum construction that motivates this coordinate uses the same logarithmic Hamiltonian and number-basis phase. The additional Euclidean filter and Möbius sign observable are specific to the present packet system.

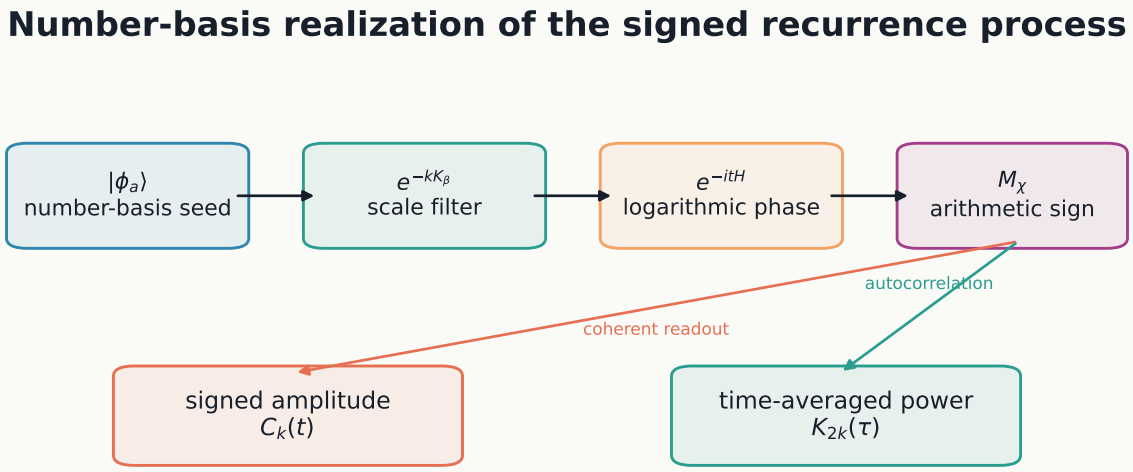


Figure 12.1: The operator realization. The signed amplitude is a coherent readout. Time averaging produces the positive depth-time kernel and removes only the off-diagonal logarithmic frequencies.

Chapter 13

The recurrence process and the RH boundary

For the principal parameters define

$$c_k = C_{2,2}(k, 0)$$

and

$$E_k = \mathcal{K}(k, k; 0) = \sum_{n\geq 1} \frac{\mu(n)^2}{n^4} \left(1 - \frac{1}{n^2}\right)^{2k}.$$

The preceding papers prove

$$E_k \sim \frac{3}{2^{5/2}\pi^{3/2}} k^{-3/2}.$$

Define

$$\rho_k(t) = \frac{C_{2,2}(k, t)}{\sqrt{E_k}}.$$

The exact time-average identity gives

$$\boxed{\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |\rho_k(t)|^2 \, \mathrm{d}t = 1.}$$

The time-zero power is

$$\Gamma_k = |\rho_k(0)|^2 = \frac{|c_k|^2}{E_k}.$$

The classical sequential criterion and the proved energy asymptotic give

$$\boxed{\mathrm{RH} \iff \Gamma_k = O_\varepsilon(k^\varepsilon) \quad \text{for every } \varepsilon > 0.}$$

The positive autocorrelation is therefore fully controlled. The remaining assertion is that the deterministic recurrence time $t = 0$ is not polynomially exceptional relative to the unit average power.

Boundary statement 13.1 (The exact remaining obstruction). Gamma dephasing, scale overlap, character decoherence, positive Gram structure, and finite signed Padé reduction do not prove the principal coherence bound. The missing theorem must use the internal arithmetic signs of $\mu(n)$ inside the principal sector.

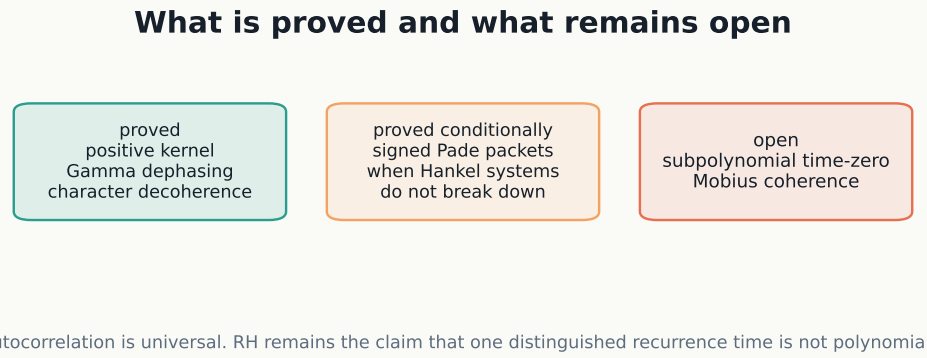


Figure 13.1: The exact boundary. The positive depth-time theory is unconditional. Finite signed models are exact under a visible nonbreakdown condition. The principal time-zero coherence remains open.

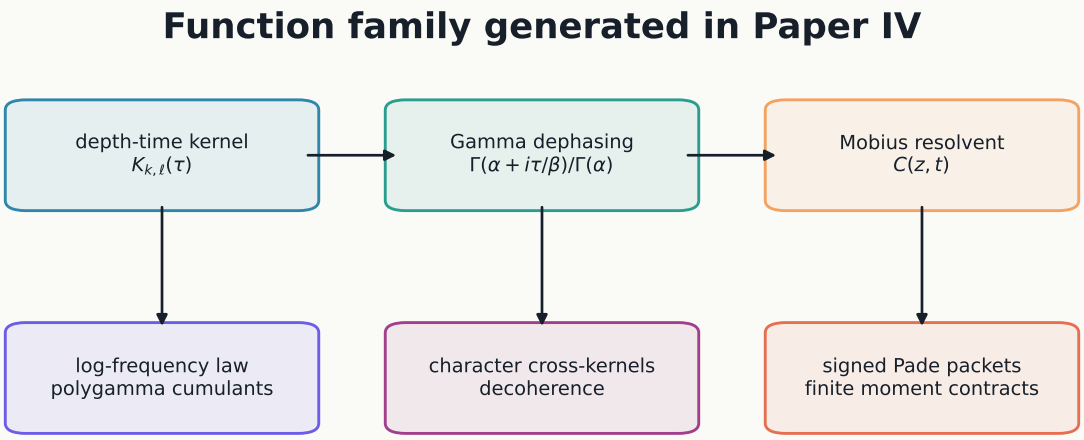


Figure 13.2: The generated function family. The paper adds one positive reproducing kernel, one zero-free Gamma dephasing limit, one signed resolvent, one log-frequency law, one character cross-kernel family, and finite signed Padé packet models.

Part VI

Worked algorithms and verification

Chapter 14

Computational route

Algorithm 14.1 (Constructing the depth-time kernel and signed reduction)**.** 1. Sieve the Möbius function through the declared cutoff.

2. Form the nodes $y_n = 1 - n^{-\beta}$.
3. Form the positive weights $\mu(n)^2 \, |\chi(n)|^2 \, n^{-2a}$.
4. Evaluate the depth-time kernel by diagonal contraction.
5. Evaluate the signed transfer with the Möbius phase retained.
6. Compare the dephased positive kernel with the Gamma-ratio limit.
7. Form signed moments $c_0, \dots, c_{2m-1}$.
8. Check the Hankel determinant.
9. Solve the Padé denominator system when the determinant is nonzero.
10. Return the m signed packet nodes and weights with a $2m$-moment certificate.

Chapter 15

Verification ledger

18/18 built-in checks passed

Table 15.1: Built-in computational certificates. Full numerical details are retained in the JSON ledger.

Check	Result
depth-time Gram matrix is positive semidefinite	pass
finite-difference zeta-ratio coordinates	pass
time-average correlation approaches positive kernel	pass
principal Gamma modulus closed form	pass
log-frequency mean approaches polygamma target	pass
log-frequency variance approaches polygamma target	pass
signed resolvent parameter-shift equation	pass
positive Pade nodes remain in physical interval	pass
first signed one-node packet lies outside physical interval	pass
positive Pade model matches first eight moments	pass
signed Pade model matches first eight moments	pass
Borel and Riesz bridge	pass
signed finite-difference moments	pass
Gamma dephasing asymptotic	pass
depth overlap example	pass
finite-model long-time mean power	pass
character off-diagonal correlation decreases	pass
signed resolvent coefficient expansion	pass

The independent test suite contains 18 additional tests. It checks the finite-difference formulas, Gram positivity, Gamma limit, hyperbolic-secant overlap, polygamma cumulants, character decoherence, resolvent transfer equation, Padé moment contract, positive versus signed node behavior, Borel–Riesz identity, figure inventory, and result ledger.

Appendix A

Formula registry

Object	Formula
Signed transfer	$C_{a,\beta,\chi}(k,t)=\sum_n \mu(n)\chi(n)n^{-a-\mathrm{i}t}(1-n^{-\beta})^k$
Depth-time kernel	$\mathcal{K}_{\chi,\psi}(k,\ell;\tau)=\sum_n \mu(n)^2\chi(n)\overline{\psi(n)}n^{-2a-\mathrm{i}\tau}(1-n^{-\beta})^{k+\ell}$
Squarefree ratio	$D_\xi(s)=L(s,\xi)/L(2s,\xi^2)$
Gamma shape	$\alpha=(2a-1)/\beta$
Gamma dephasing	$\mathcal{D}_{\alpha,\beta}(\tau)=\Gamma(\alpha+\mathrm{i}\tau/\beta)/\Gamma(\alpha)$
Principal modulus	$\left\lvert\mathcal{D}_{3/2,2}(\tau)\right\rvert^2=(1+\tau^2)/\cosh(\pi\tau/2)$
Depth overlap	$\operatorname{sech}^{\alpha}(\frac{1}{2}\log(k/\ell))$
Log-frequency mean	$-\psi(\alpha)/\beta$
Log-frequency variance	$\psi_1(\alpha)/\beta^2$
Signed resolvent	$\mathcal{C}(z,t)=\sum_n \mu(n)\chi(n)n^{-a-\mathrm{i}t}/[1-z(1-n^{-\beta})]$
Resolvent transfer	$(1-z)\mathcal{C}_a+z\mathcal{C}_{a+\beta}=L(a+\mathrm{i}t,\chi)^{-1}$
Riesz companion	$\mathcal{R}(x,t)=x\sum_n \mu(n)\chi(n)n^{-a-\mathrm{i}t}\mathrm{e}^{-x/n^{\beta}}$
Hankel determinant	$\Delta_m(t)=\det[C(i+j,t)]_{i,j=0}^{m-1}$
Padé contract	$C(j,t)=\sum_{r=1}^{m}\lambda_r(t)\xi_r(t)^j$ for $0\le j\le 2m-1$
Principal coherence	$\Gamma_k=\lvert C_{2,2}(k,0)\rvert^2/\mathcal{K}(k,k;0)$

Appendix B

Implementation map

The complete implementation is supplied as

`projector_generated_special_functions_iv.py`.

It contains the Möbius sieve, character packets, depth-time kernel, finite time averages, exact zeta-ratio checks, Gamma asymptotics, log-frequency laws, signed resolvent, Padé packet reduction, Borel–Riesz transform, figure generation, table generation, and built-in verification ledger.

The central computational path is:

```
params = KernelParameters(a=2.0, beta=2.0, conductor=1)
data = build_number_packet_data(params, nmax=600_000)
kernel = data.positive_kernel(k + ell, tau)
signed = data.signed_transfer(k, t)
numerator, denominator = pade_from_moments(moments, order=m)
nodes, weights = pade_packet_nodes_weights(moments, order=m)
```

The positive kernel is evaluated by diagonal contraction. The signed Padé model is built from moments only. Neither path materializes a dense number-basis operator.

Appendix C

Complete Python source

```
from __future__ import annotations

import csv
import json
import math
from dataclasses import dataclass
from pathlib import Path
from typing import Callable, Iterable, Sequence

import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import FancyArrowPatch, FancyBboxPatch, Rectangle
import mpmath as mp
import numpy as np
from scipy import linalg
from scipy.special import digamma, gamma as gamma_fn, polygamma

mp.mp.dps = 90

ROOT = Path(__file__).resolve().parent
FIGURES = ROOT / "figures"
RESULTS = ROOT / "results"
TABLES = ROOT / "tables"
for directory in (FIGURES, RESULTS, TABLES):
    directory.mkdir(parents=True, exist_ok=True)

INK = "#17202A"
BLUE = "#2E86AB"
TEAL = "#2A9D8F"
ORANGE = "#F4A261"
MAGENTA = "#A23E8C"
RED = "#E76F51"
YELLOW = "#E9C46A"
PURPLE = "#6C5CE7"
MUTED = "#5D6D7E"
PAPER = "#FAFAF7"
GRID = "#D5D8DC"
WHITE = "#FFFFFF"
COLORS = [BLUE, TEAL, ORANGE, MAGENTA, RED, YELLOW, PURPLE]

plt.rcParams.update(
    {
        "figure.facecolor": PAPER,
        "axes.facecolor": PAPER,
        "axes.edgecolor": INK,
        "axes.labelcolor": INK,
        "xtick.color": INK,
        "ytick.color": INK,
        "text.color": INK,
```

```
        "font.size": 10.5,
        "axes.titlesize": 12.5,
        "axes.titleweight": "bold",
        "legend.frameon": False,
        "savefig.bbox": "tight",
        "savefig.facecolor": PAPER,
    }
)


def save_figure(fig: plt.Figure, stem: str) -> None:
    fig.savefig(FIGURES / f"{stem}.pdf")
    fig.savefig(FIGURES / f"{stem}.png", dpi=220)
    plt.close(fig)


def diagram_axes(figsize: tuple[float, float] = (12.6, 6.1)) -> tuple[plt.Figure, plt.Axes]:
    fig, ax = plt.subplots(figsize=figsize)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.axis("off")
    return fig, ax


def box(
    ax: plt.Axes,
    xy: tuple[float, float],
    width: float,
    height: float,
    text: str,
    face: str = WHITE,
    edge: str = INK,
    fontsize: float = 11.0,
) -> None:
    patch = FancyBboxPatch(
        xy,
        width,
        height,
        boxstyle="round,pad=.012,rounding_size=.018",
        facecolor=face,
        edgecolor=edge,
        linewidth=1.7,
    )
    ax.add_patch(patch)
    ax.text(xy[0] + width / 2, xy[1] + height / 2, text, ha="center", va="center", fontsize=fontsize)


def arrow(
    ax: plt.Axes,
    start: tuple[float, float],
    end: tuple[float, float],
    color: str = INK,
    text: str | None = None,
    text_offset: tuple[float, float] = (0.0, 0.03),
) -> None:
    patch = FancyArrowPatch(start, end, arrowstyle="-|>", mutation_scale=13, linewidth=1.6, color=color)
    ax.add_patch(patch)
    if text:
        ax.text(
            (start[0] + end[0]) / 2 + text_offset[0],
            (start[1] + end[1]) / 2 + text_offset[1],
            text,
            ha="center",
            va="center",
            fontsize=10.0,
            color=color,
        )


# ---------------------------------------------------------------------------
# Arithmetic data and packet transforms
```

```
# ----------------------------------------------------------------------------


def mobius_sieve(nmax: int) -> np.ndarray:
    mu = np.zeros(nmax + 1, dtype=np.int8)
    mu[1] = 1
    primes: list[int] = []
    composite = np.zeros(nmax + 1, dtype=bool)
    for i in range(2, nmax + 1):
        if not composite[i]:
            primes.append(i)
            mu[i] = -1
        for p in primes:
            value = i * p
            if value > nmax:
                break
            composite[value] = True
            if i % p == 0:
                mu[value] = 0
                break
            mu[value] = -mu[i]
    return mu


def prime_factors(n: int) -> list[int]:
    factors: list[int] = []
    value = n
    p = 2
    while p * p <= value:
        if value % p == 0:
            factors.append(p)
            while value % p == 0:
                value //= p
        p += 1
    if value > 1:
        factors.append(value)
    return factors


def squarefree_coprime_density(q: int) -> float:
    value = 1.0 / float(mp.zeta(2))
    for p in prime_factors(q):
        value *= p / (p + 1)
    return value


def characters_mod_five(n: np.ndarray) -> np.ndarray:
    """Return the four Dirichlet characters modulo five on an integer array."""
    n_int = np.asarray(n, dtype=np.int64)
    values = np.zeros((4, n_int.size), dtype=np.complex128)
    lookup = {1: 0, 2: 1, 4: 2, 3: 3}
    residues = n_int % 5
    for index, residue in enumerate(residues):
        if residue == 0:
            continue
        exponent = lookup[int(residue)]
        for j in range(4):
            values[j, index] = np.exp(2j * np.pi * j * exponent / 4)
    return values


@dataclass(frozen=True)
class KernelParameters:
    a: float = 2.0
    beta: float = 2.0
    conductor: int = 1

    @property
    def alpha(self) -> float:
        return (2.0 * self.a - 1.0) / self.beta

```

```
    @property
    def density(self) -> float:
        return squarefree_coprime_density(self.conductor)


@dataclass
class NumberPacketData:
    params: KernelParameters
    n: np.ndarray
    mu: np.ndarray
    y: np.ndarray
    positive_weight: np.ndarray
    signed_amplitude: np.ndarray

    def positive_kernel(self, depth: int, tau: float = 0.0) -> complex:
        logy = np.log(self.y, where=self.y > 0, out=np.full_like(self.y, -np.inf))
        depth_factor = np.exp(depth * logy) if depth else np.ones_like(self.y)
        phase = np.exp(-1j * tau * np.log(self.n))
        return complex(np.sum(self.positive_weight * depth_factor * phase))

    def signed_transfer(self, k: int, t: float = 0.0, character: np.ndarray | None = None) -> complex:
        logy = np.log(self.y, where=self.y > 0, out=np.full_like(self.y, -np.inf))
        depth_factor = np.exp(k * logy) if k else np.ones_like(self.y)
        phase = np.exp(-1j * t * np.log(self.n))
        values = self.signed_amplitude * depth_factor * phase
        if character is not None:
            values = values * character
        return complex(np.sum(values))

    def positive_companion_moment(self, k: int) -> float:
        logy = np.log(self.y, where=self.y > 0, out=np.full_like(self.y, -np.inf))
        factor = np.exp(k * logy) if k else np.ones_like(self.y)
        return float(np.sum(self.positive_weight * factor))


def build_number_packet_data(params: KernelParameters, nmax: int = 1_000_000) -> NumberPacketData:
    mu_full = mobius_sieve(nmax)
    integers = np.arange(1, nmax + 1, dtype=np.int64)
    mask = mu_full[1:] != 0
    if params.conductor > 1:
        mask &= np.gcd(integers, params.conductor) == 1
    selected = integers[mask]
    mu = mu_full[1:][mask].astype(np.float64)
    n = selected.astype(np.float64)
    y = 1.0 - n ** (-params.beta)
    positive_weight = n ** (-2.0 * params.a)
    signed_amplitude = mu * n ** (-params.a)
    return NumberPacketData(
        params=params,
        n=n,
        mu=mu,
        y=y,
        positive_weight=positive_weight,
        signed_amplitude=signed_amplitude,
    )


def exact_signed_moment(k: int, a: int = 2, beta: int = 2, dps: int = 90) -> mp.mpf:
    mp.mp.dps = dps
    return mp.fsum(
        [
            (-1) ** j * mp.binomial(k, j) / mp.zeta(a + beta * j)
            for j in range(k + 1)
        ]
    )


def exact_positive_kernel(depth: int, tau: float = 0.0, a: float = 2.0, beta: float = 2.0, dps: int = 90) -> complex:
    mp.mp.dps = dps
    total = mp.mpc(0)
    for j in range(depth + 1):
```

```
        s = 2 * a + beta * j + 1j * tau
        d = mp.zeta(s) / mp.zeta(2 * s)
        total += (-1) ** j * mp.binomial(depth, j) * d
    return complex(total)


def gamma_dephasing(alpha: float, beta: float, tau: np.ndarray | float) -> np.ndarray:
    values = np.asarray(tau, dtype=float)
    return np.array(
        [complex(mp.gamma(alpha + 1j * float(t) / beta) / mp.gamma(alpha)) for t in values.ravel()],
        dtype=np.complex128,
    ).reshape(values.shape)


def principal_modulus_squared(tau: np.ndarray | float) -> np.ndarray:
    values = np.asarray(tau, dtype=float)
    return (1.0 + values**2) / np.cosh(np.pi * values / 2.0)


def depth_overlap_limit(alpha: float, k: np.ndarray | float, ell: np.ndarray | float) -> np.ndarray:
    k_values = np.asarray(k, dtype=float)
    ell_values = np.asarray(ell, dtype=float)
    return (2.0 * np.sqrt(k_values * ell_values) / (k_values + ell_values)) ** alpha


def log_frequency_cumulants(alpha: float, beta: float, order: int) -> np.ndarray:
    values = []
    for r in range(1, order + 1):
        if r == 1:
            values.append(-float(digamma(alpha)) / beta)
        else:
            values.append(((-1.0) ** r) * float(polygamma(r - 1, alpha)) / (beta**r))
    return np.array(values)


# ---------------------------------------------------------------------------
# Time averages, resolvents, and Pade packet models
# ---------------------------------------------------------------------------


def finite_time_average(
    data: NumberPacketData,
    k: int,
    ell: int,
    tau: float,
    horizon: float,
    samples: int = 2001,
) -> complex:
    times = np.linspace(-horizon, horizon, samples)
    first = np.array([data.signed_transfer(k, float(t + tau)) for t in times])
    second = np.array([data.signed_transfer(ell, float(t)) for t in times])
    return complex(np.trapezoid(first * np.conjugate(second), times) / (2.0 * horizon))


def signed_resolvent(data: NumberPacketData, z: complex, t: float = 0.0) -> complex:
    phase = np.exp(-1j * t * np.log(data.n))
    return complex(np.sum(data.signed_amplitude * phase / (1.0 - z * data.y)))


def positive_resolvent(data: NumberPacketData, z: complex) -> complex:
    return complex(np.sum(data.positive_weight / (1.0 - z * data.y)))


def signed_resolvent_functional_residual(
    data_a: NumberPacketData,
    data_shifted: NumberPacketData,
    z: complex,
    t: float = 0.0,
) -> complex:
    left = (1.0 - z) * signed_resolvent(data_a, z, t) + z * signed_resolvent(data_shifted, z, t)
    target = complex(1.0 / mp.zeta(data_a.params.a + 1j * t))
```

```
    return left - target


def pade_from_moments(moments: Sequence[complex], order: int) -> tuple[np.ndarray, np.ndarray]:
    values = np.asarray(moments, dtype=np.complex128)
    if values.size < 2 * order:
        raise ValueError("at least 2*order moments are required")
    matrix = np.empty((order, order), dtype=np.complex128)
    rhs = np.empty(order, dtype=np.complex128)
    for row in range(order):
        n = order + row
        rhs[row] = -values[n]
        for column in range(order):
            j = column + 1
            matrix[row, column] = values[n - j]
    q_tail = np.linalg.solve(matrix, rhs)
    denominator = np.concatenate(([1.0 + 0j], q_tail))
    numerator = np.zeros(order, dtype=np.complex128)
    for n in range(order):
        total = values[n]
        for j in range(1, min(order, n) + 1):
            total += denominator[j] * values[n - j]
        numerator[n] = total
    return numerator, denominator


def pade_series_coefficients(numerator: np.ndarray, denominator: np.ndarray, count: int) -> np.ndarray:
    result = np.zeros(count, dtype=np.complex128)
    for n in range(count):
        value = numerator[n] if n < numerator.size else 0j
        for j in range(1, min(n, denominator.size - 1) + 1):
            value -= denominator[j] * result[n - j]
        result[n] = value / denominator[0]
    return result


def pade_packet_nodes_weights(moments: Sequence[complex], order: int) -> tuple[np.ndarray, np.ndarray]:
    numerator, denominator = pade_from_moments(moments, order)
    poles = np.roots(denominator[::-1])
    nodes = 1.0 / poles
    vandermonde = np.vstack([nodes**k for k in range(order)])
    weights = np.linalg.solve(vandermonde, np.asarray(moments[:order], dtype=np.complex128))
    return nodes, weights


def hankel_determinant(moments: Sequence[complex], order: int) -> complex:
    matrix = np.array([[moments[i + j] for j in range(order)] for i in range(order)], dtype=np.complex128)
    return complex(np.linalg.det(matrix))


def borel_transfer(data: NumberPacketData, x: float, t: float = 0.0) -> complex:
    phase = np.exp(-1j * t * np.log(data.n))
    return complex(np.sum(data.signed_amplitude * phase * np.exp(x * data.y)))


def riesz_companion(data: NumberPacketData, x: float, t: float = 0.0) -> complex:
    phase = np.exp(-1j * t * np.log(data.n))
    return complex(x * np.sum(data.signed_amplitude * phase * np.exp(-x / (data.n ** data.params.beta))))


# ---------------------------------------------------------------------------
# Character cross-kernels
# ---------------------------------------------------------------------------


def character_cross_kernel(
    data: NumberPacketData,
    character_a: np.ndarray,
    character_b: np.ndarray,
    depth: int,
    tau: float = 0.0,
```

```
) -> complex:
    logy = np.log(data.y, where=data.y > 0, out=np.full_like(data.y, -np.inf))
    depth_factor = np.exp(depth * logy) if depth else np.ones_like(data.y)
    phase = np.exp(-1j * tau * np.log(data.n))
    return complex(
        np.sum(
            data.positive_weight
            * character_a
            * np.conjugate(character_b)
            * depth_factor
            * phase
        )
    )


# ---------------------------------------------------------------------------
# Figures and verification
# ---------------------------------------------------------------------------


def build_figures_and_results() -> dict[str, object]:
    params = KernelParameters(a=2.0, beta=2.0, conductor=1)
    data = build_number_packet_data(params, nmax=600_000)
    visual_mask = data.n <= 12_000
    plot_data = NumberPacketData(
        params=data.params,
        n=data.n[visual_mask],
        mu=data.mu[visual_mask],
        y=data.y[visual_mask],
        positive_weight=data.positive_weight[visual_mask],
        signed_amplitude=data.signed_amplitude[visual_mask],
    )
    chars5 = characters_mod_five(data.n.astype(np.int64))

    checks: list[dict[str, object]] = []

    def record(name: str, passed: bool, detail: object) -> None:
        checks.append({"name": name, "passed": bool(passed), "detail": detail})

    # Figure 1: theory pipeline
    fig, ax = diagram_axes()
    ax.text(0.5, 0.94, "From signed arithmetic transfer to universal depth-time kernels", ha="center", fontsize=20,
     weight="bold")
    nodes = [
        (0.03, 0.58, 0.15, 0.18, "Mobius packet\n$C_k(t)$", BLUE),
        (0.22, 0.58, 0.15, 0.18, "time average\nand Gram lift", TEAL),
        (0.41, 0.58, 0.15, 0.18, "positive depth-time\nkernel $K$", ORANGE),
        (0.60, 0.58, 0.15, 0.18, "Gamma dephasing\nlimit", MAGENTA),
        (0.79, 0.58, 0.17, 0.18, "signed resolvent\nand Pade packets", RED),
    ]
    for x0, y0, w, h, text, color in nodes:
        box(ax, (x0, y0), w, h, text, face=color + "18", edge=color, fontsize=11.5)
    for left, right in zip(nodes[:-1], nodes[1:], strict=True):
        arrow(ax, (left[0] + left[2], left[1] + left[3] / 2), (right[0], right[1] + right[3] / 2))
    box(ax, (0.13, 0.18), 0.23, 0.15, "unconditional\npositive-definite theory", face=TEAL + "18", edge=TEAL,
     fontsize=12)
    box(ax, (0.41, 0.18), 0.23, 0.15, "finite signed\nmoment models", face=ORANGE + "18", edge=ORANGE, fontsize=12)
    box(ax, (0.69, 0.18), 0.20, 0.15, "RH remains a\ntime-zero coherence bound", face=RED + "18", edge=RED, fontsize
     =12)
    arrow(ax, (0.485, 0.58), (0.25, 0.33), color=TEAL)
    arrow(ax, (0.875, 0.58), (0.525, 0.33), color=ORANGE)
    arrow(ax, (0.875, 0.58), (0.79, 0.33), color=RED)
    save_figure(fig, "01_theory_pipeline")

    # Figure 2: feature map
    fig, ax = diagram_axes()
    ax.text(0.5, 0.94, "The exact feature vector behind the kernel", ha="center", fontsize=20, weight="bold")
    box(ax, (0.06, 0.55), 0.22, 0.18, "packet coordinate\n$(k,t)$", face=BLUE + "18", edge=BLUE, fontsize=14)
    box(ax, (0.39, 0.55), 0.24, 0.18, r"$v_{k,t}(n)=\sqrt{w_n}\,y_n^k n^{-it}$", face=TEAL + "18", edge=TEAL,
     fontsize=13)
```

```
    box(ax, (0.73, 0.55), 0.21, 0.18, r"$K((k,t),(\ell,s))$" + "\n" + r"$=\langle v_{k,t},v_{\ell,s}\rangle$", face=ORANGE + "18", edge=ORANGE, fontsize=12)
    arrow(ax, (0.28, 0.64), (0.39, 0.64), text="analysis")
    arrow(ax, (0.63, 0.64), (0.73, 0.64), text="Gram contraction")
    ax.text(0.5, 0.28, "Depth controls scale. Time resolves the logarithmic spectrum. The kernel keeps both coordinates without materializing a dense operator.", ha="center", fontsize=12, color=MUTED, wrap=True)
    save_figure(fig, "02_feature_map")

    # Figure 3: positive Gram matrix
    points = [(k, t) for k in (250, 1000, 4000, 16000) for t in (0.0, 1.0, 2.5)]
    gram = np.empty((len(points), len(points)), dtype=np.complex128)
    for i, (k, t) in enumerate(points):
        for j, (ell, s) in enumerate(points):
            gram[i, j] = data.positive_kernel(k + ell, t - s)
    eigenvalues = np.linalg.eigvalsh(gram.real)
    record("depth-time Gram matrix is positive semidefinite", float(np.min(eigenvalues)) > -1e-11, float(np.min(eigenvalues)))
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 5.0))
    im0 = axes[0].imshow(np.abs(gram), cmap="magma", aspect="auto")
    axes[0].set_title("absolute Gram entries")
    axes[0].set_xlabel("packet index")
    axes[0].set_ylabel("packet index")
    fig.colorbar(im0, ax=axes[0], fraction=0.046)
    axes[1].plot(np.arange(len(eigenvalues)), eigenvalues, marker="o", color=TEAL)
    axes[1].axhline(0, color=INK, lw=1)
    axes[1].set_yscale("symlog", linthresh=1e-14)
    axes[1].set_title("eigenvalues of the real Gram part")
    axes[1].set_xlabel("ordered index")
    axes[1].grid(alpha=0.18)
    fig.suptitle("The squarefree depth-time kernel is a genuine positive-definite kernel", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "03_kernel_gram")

    # Figure 4: exact finite-difference coordinates
    depths = np.arange(0, 13)
    direct = np.array([data.positive_kernel(int(m), 0.7) for m in depths])
    exact = np.array([exact_positive_kernel(int(m), 0.7, dps=100) for m in depths])
    residual = np.abs(direct - exact)
    record("finite-difference zeta-ratio coordinates", float(np.max(residual)) < 2e-9, float(np.max(residual)))
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    axes[0].plot(depths, direct.real, marker="o", color=BLUE, label="squarefree sum")
    axes[0].plot(depths, exact.real, marker="x", ls="--", color=ORANGE, label="finite zeta-ratio difference")
    axes[0].set_xlabel("total depth $m$")
    axes[0].set_ylabel("real part")
    axes[0].set_title("two exact coordinate systems")
    axes[0].grid(alpha=0.18)
    axes[0].legend()
    axes[1].semilogy(depths, np.maximum(residual, 1e-18), marker="o", color=RED)
    axes[1].set_xlabel("total depth $m$")
    axes[1].set_ylabel("absolute residual")
    axes[1].set_title("coordinate agreement")
    axes[1].grid(alpha=0.18)
    fig.suptitle("The kernel is a finite difference of a squarefree $L$-ratio", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "04_l_ratio_coordinates")

    # Figure 5: finite-time average convergence
    small_data = build_number_packet_data(params, nmax=2500)
    exact_corr = small_data.positive_kernel(14, 0.8)
    horizons = np.array([5, 10, 20, 40, 80, 160], dtype=float)
    approximations = np.array([finite_time_average(small_data, 6, 8, 0.8, float(T), samples=1601) for T in horizons])
    errors = np.abs(approximations - exact_corr)
    record("time-average correlation approaches positive kernel", bool(errors[-1] < errors[0]), [float(v) for v in errors])
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    axes[0].plot(horizons, approximations.real, marker="o", color=BLUE, label="finite average")
    axes[0].axhline(exact_corr.real, color=ORANGE, ls="--", label="exact diagonal kernel")
    axes[0].set_xlabel("time horizon $T$")
    axes[0].set_ylabel("real part")
```

```
    axes[0].set_title("ergodic diagonal extraction")
    axes[0].grid(alpha=0.18)
    axes[0].legend()
    axes[1].loglog(horizons, errors, marker="o", color=RED)
    axes[1].set_xlabel("time horizon $T$")
    axes[1].set_ylabel("absolute error")
    axes[1].set_title("finite-window convergence")
    axes[1].grid(alpha=0.18)
    fig.suptitle("The positive kernel is the long-time autocorrelation of the signed signal", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "05_time_average")

    # Figures 6-9: Gamma dephasing
    taus = np.linspace(-10, 10, 500)
    gamma_curve = gamma_dephasing(params.alpha, params.beta, taus)
    fig, ax = plt.subplots(figsize=(7.2, 6.2))
    ax.plot(gamma_curve.real, gamma_curve.imag, color=BLUE, lw=2)
    ax.scatter([1], [0], color=RED, zorder=3)
    ax.set_xlabel("real part")
    ax.set_ylabel("imaginary part")
    ax.set_title("Universal Gamma dephasing trajectory")
    ax.grid(alpha=0.18)
    ax.set_aspect("equal", adjustable="datalim")
    save_figure(fig, "06_gamma_trajectory")

    depths_for_gamma = [2000, 10000, 50000]
    tau_short = np.linspace(-5, 5, 121)
    fig, axes = plt.subplots(2, 1, figsize=(12.5, 7.0), sharex=True)
    target = gamma_dephasing(params.alpha, params.beta, tau_short)
    for depth, color in zip(depths_for_gamma, [ORANGE, TEAL, BLUE], strict=True):
        observed = np.array([
            depth ** (1j * t / params.beta) * data.positive_kernel(depth, float(t)) / data.positive_kernel(depth, 0.0)
            for t in tau_short
        ])
        axes[0].plot(tau_short, observed.real, color=color, lw=1.3, label=f"depth {depth:,}")
        axes[1].plot(tau_short, observed.imag, color=color, lw=1.3)
    axes[0].plot(tau_short, target.real, color=INK, lw=2.3, ls="--", label="Gamma limit")
    axes[1].plot(tau_short, target.imag, color=INK, lw=2.3, ls="--")
    axes[0].set_ylabel("real part")
    axes[1].set_ylabel("imaginary part")
    axes[1].set_xlabel("time lag $\\tau$")
    axes[0].set_title("dephased real component")
    axes[1].set_title("dephased imaginary component")
    for ax in axes:
        ax.grid(alpha=0.18)
    axes[0].legend(ncol=2)
    fig.suptitle("The discrete packet correlation converges to a Gamma-ratio function", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.93])
    save_figure(fig, "07_gamma_convergence")

    modulus_observed = np.abs(gamma_curve) ** 2
    modulus_closed = principal_modulus_squared(taus)
    record("principal Gamma modulus closed form", float(np.max(np.abs(modulus_observed - modulus_closed))) < 2e-13, float(np.max(np.abs(modulus_observed - modulus_closed))))
    fig, ax = plt.subplots(figsize=(12.0, 4.8))
    ax.semilogy(taus, modulus_observed, color=BLUE, lw=2, label="Gamma ratio")
    ax.semilogy(taus, modulus_closed, color=ORANGE, ls="--", lw=1.8, label=r"$(1+\tau^2)/\cosh(\pi\tau/2)$")
    ax.set_xlabel("time lag $\\tau$")
    ax.set_ylabel("squared modulus")
    ax.set_title(r"Closed dephasing envelope for $a=\beta=2$")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "08_principal_modulus")

    phase = np.unwrap(np.angle(gamma_curve))
    phase_derivative = np.gradient(phase, taus)
    target_phase_derivative = np.real(np.array([complex(mp.digamma(params.alpha + 1j * t / params.beta)) for t in taus])) / params.beta
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
```

```
    axes[0].plot(taus, phase, color=MAGENTA)
    axes[0].set_xlabel("time lag $\\tau$")
    axes[0].set_ylabel("phase")
    axes[0].set_title("unwrapped packet phase")
    axes[0].grid(alpha=0.18)
    axes[1].plot(taus, phase_derivative, color=BLUE, label="numerical derivative")
    axes[1].plot(taus, target_phase_derivative, color=ORANGE, ls="--", label="digamma law")
    axes[1].set_xlabel("time lag $\\tau$")
    axes[1].set_ylabel("phase slope")
    axes[1].set_title("exact group-delay law")
    axes[1].grid(alpha=0.18)
    axes[1].legend()
    fig.suptitle("The limiting phase is controlled by the digamma function", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "09_phase_digamma")

    # Figures 10-11: log-frequency law and cumulants
    depth_log = 100000
    logy = np.log(data.y, where=data.y > 0, out=np.full_like(data.y, -np.inf))
    probability = data.positive_weight * np.exp(depth_log * logy)
    probability /= np.sum(probability)
    centered_log = np.log(data.n) - math.log(depth_log) / params.beta
    sample_mean = float(np.sum(probability * centered_log))
    sample_variance = float(np.sum(probability * (centered_log - sample_mean) ** 2))
    target_cumulants = log_frequency_cumulants(params.alpha, params.beta, 4)
    record("log-frequency mean approaches polygamma target", abs(sample_mean - target_cumulants[0]) < 0.01, [
     sample_mean, float(target_cumulants[0])])
    record("log-frequency variance approaches polygamma target", abs(sample_variance - target_cumulants[1]) < 0.02,
     [sample_variance, float(target_cumulants[1])])
    fig, ax = plt.subplots(figsize=(12.0, 5.0))
    bins = np.linspace(-2.0, 2.0, 90)
    hist, edges = np.histogram(centered_log, bins=bins, weights=probability, density=True)
    centers = (edges[:-1] + edges[1:]) / 2
    ax.plot(centers, hist, drawstyle="steps-mid", color=BLUE, lw=1.8, label="discrete tilted packet")
    u = np.logspace(-4, 4, 200000)
    z = -np.log(u) / params.beta
    density_u = u ** (params.alpha - 1) * np.exp(-u) / gamma_fn(params.alpha)
    density_z = params.beta * u * density_u
    order = np.argsort(z)
    ax.plot(z[order], density_z[order], color=ORANGE, ls="--", lw=2, label="log-Gamma limit")
    ax.set_xlim(-2, 2)
    ax.set_xlabel(r"$\log n-\beta^{-1}\log m$")
    ax.set_ylabel("density")
    ax.set_title("Universal centered log-frequency profile")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "10_log_frequency_profile")

    depth_values = np.array([1000, 3000, 10000, 30000, 100000], dtype=int)
    means = []
    variances = []
    for depth in depth_values:
        p = data.positive_weight * np.exp(depth * logy)
        p /= np.sum(p)
        zvals = np.log(data.n) - math.log(depth) / params.beta
        mean = float(np.sum(p * zvals))
        variance = float(np.sum(p * (zvals - mean) ** 2))
        means.append(mean)
        variances.append(variance)
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    axes[0].semilogx(depth_values, means, marker="o", color=BLUE)
    axes[0].axhline(target_cumulants[0], color=ORANGE, ls="--", label="polygamma limit")
    axes[0].set_xlabel("depth $m$")
    axes[0].set_ylabel("centered mean")
    axes[0].set_title("mean log-frequency")
    axes[0].grid(alpha=0.18)
    axes[0].legend()
    axes[1].semilogx(depth_values, variances, marker="o", color=TEAL)
    axes[1].axhline(target_cumulants[1], color=ORANGE, ls="--", label="polygamma limit")
    axes[1].set_xlabel("depth $m$")
    axes[1].set_ylabel("variance")
```

```
    axes[1].set_title("log-frequency variance")
    axes[1].grid(alpha=0.18)
    axes[1].legend()
    fig.suptitle("Logarithmic spectral cumulants converge to explicit polygamma constants", fontsize=17, weight="
     bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "11_log_cumulants")

    # Figures 12-14: depth overlap
    ratios = np.logspace(-2, 2, 300)
    limit_overlap = depth_overlap_limit(params.alpha, ratios, np.ones_like(ratios))
    fig, ax = plt.subplots(figsize=(12.0, 4.8))
    ax.semilogx(ratios, limit_overlap, color=BLUE, lw=2)
    ax.axvline(1, color=INK, lw=1)
    ax.set_xlabel("depth ratio $k/\\ell$")
    ax.set_ylabel("normalized overlap")
    ax.set_title(r"Universal scale overlap $\operatorname{sech}^{\alpha}(\frac{1}{2}\log(k/\ell))$")
    ax.grid(alpha=0.18)
    save_figure(fig, "12_depth_overlap_limit")

    base_depths = [2000, 10000, 50000]
    ratio_samples = np.array([0.25, 0.5, 1.0, 2.0, 4.0])
    fig, ax = plt.subplots(figsize=(12.0, 5.0))
    for base, color in zip(base_depths, [ORANGE, TEAL, BLUE], strict=True):
        values = []
        for ratio in ratio_samples:
            k = int(round(base * ratio))
            ell = base
            observed = data.positive_kernel(k + ell, 0.0).real / math.sqrt(
                data.positive_kernel(2 * k, 0.0).real * data.positive_kernel(2 * ell, 0.0).real
            )
            values.append(observed)
        ax.plot(ratio_samples, values, marker="o", color=color, label=f"base {base:,}")
    ax.plot(ratio_samples, depth_overlap_limit(params.alpha, ratio_samples, 1.0), color=INK, ls="--", lw=2, label="
     limit")
    ax.set_xscale("log")
    ax.set_xlabel("depth ratio $k/\\ell$")
    ax.set_ylabel("normalized overlap")
    ax.set_title("Finite packet overlaps approach the hyperbolic-secant law")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "13_depth_overlap_convergence")

    k_grid = np.logspace(2.5, 5.0, 55)
    overlap_grid = np.empty((k_grid.size, k_grid.size))
    for i, k in enumerate(k_grid):
        for j, ell in enumerate(k_grid):
            overlap_grid[i, j] = depth_overlap_limit(params.alpha, k, ell)
    fig, ax = plt.subplots(figsize=(7.0, 6.0))
    im = ax.imshow(overlap_grid, origin="lower", cmap="viridis", extent=[2.5, 5.0, 2.5, 5.0], aspect="auto", vmin=0,
      vmax=1)
    ax.set_xlabel(r"$\log_{10}\ell$")
    ax.set_ylabel(r"$\log_{10}k$")
    ax.set_title("Universal overlap geometry on log depth")
    fig.colorbar(im, ax=ax, fraction=0.046, label="overlap")
    save_figure(fig, "14_depth_overlap_heatmap")

    # Figures 15-16: character cross-kernels
    char_depths = [100, 1000, 10000, 100000]
    char_corr_matrices = []
    for depth in char_depths:
        matrix = np.empty((4, 4), dtype=np.complex128)
        for i in range(4):
            for j in range(4):
                numerator = character_cross_kernel(data, chars5[i], chars5[j], depth, 0.0)
                denominator = math.sqrt(
                    abs(character_cross_kernel(data, chars5[i], chars5[i], depth, 0.0))
                    * abs(character_cross_kernel(data, chars5[j], chars5[j], depth, 0.0))
                )
                matrix[i, j] = numerator / denominator
        char_corr_matrices.append(matrix)
```

```
    fig, axes = plt.subplots(1, 4, figsize=(13.0, 3.8))
    for ax, matrix, depth in zip(axes, char_corr_matrices, char_depths, strict=True):
        image = ax.imshow(np.abs(matrix), cmap="magma", vmin=0, vmax=1)
        ax.set_title(f"depth {depth:,}", fontsize=10.5)
        ax.set_xticks(range(4))
        ax.set_yticks(range(4))
    fig.colorbar(image, ax=axes.ravel().tolist(), fraction=0.018, pad=0.02, label="normalized magnitude")
    fig.suptitle("Distinct modulo-five character packets decohere with depth", fontsize=17, weight="bold")
    fig.subplots_adjust(left=0.05, right=0.92, bottom=0.12, top=0.82, wspace=0.18)
    save_figure(fig, "15_character_gram")

    max_offdiag = []
    for matrix in char_corr_matrices:
        mask = ~np.eye(4, dtype=bool)
        max_offdiag.append(float(np.max(np.abs(matrix[mask]))))
    fig, ax = plt.subplots(figsize=(11.5, 4.8))
    ax.loglog(char_depths, max_offdiag, marker="o", color=BLUE, label="largest off-diagonal")
    reference = max_offdiag[0] * (np.array(char_depths) / char_depths[0]) ** (-1 / (2 * params.beta))
    ax.loglog(char_depths, reference, color=ORANGE, ls="--", label=r"$m^{-1/(2\beta)}$")
    ax.set_xlabel("total depth $m$")
    ax.set_ylabel("largest normalized cross-correlation")
    ax.set_title("Depth-time character decoherence retains the same unconditional exponent")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "16_character_decoherence")

    # Figures 17-19: signed Mobius process
    time_axis = np.linspace(-25, 25, 801)
    signed_depths = [20, 100, 500]
    phase_matrix = np.exp(-1j * np.outer(time_axis, np.log(plot_data.n)))
    plot_logy = np.log(plot_data.y, where=plot_data.y > 0, out=np.full_like(plot_data.y, -np.inf))
    signed_series: dict[int, np.ndarray] = {}
    fig, axes = plt.subplots(len(signed_depths), 1, figsize=(12.5, 7.6), sharex=True)
    for ax, depth, color in zip(axes, signed_depths, [ORANGE, TEAL, BLUE], strict=True):
        coefficients = plot_data.signed_amplitude * np.exp(depth * plot_logy)
        values = phase_matrix @ coefficients
        signed_series[depth] = values
        ax.plot(time_axis, values.real, color=color, lw=1.2, label="real")
        ax.plot(time_axis, values.imag, color=MAGENTA, lw=1.0, alpha=0.8, label="imaginary")
        ax.set_ylabel(f"$k={depth}$")
        ax.grid(alpha=0.14)
    axes[0].legend(ncol=2)
    axes[-1].set_xlabel("logarithmic time $t$")
    fig.suptitle("The signed Mobius transfer is a deterministic quasiperiodic signal", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.93])
    save_figure(fig, "17_signed_signal")

    k_process = 300
    coefficients = plot_data.signed_amplitude * np.exp(k_process * plot_logy)
    values = phase_matrix @ coefficients
    energy = plot_data.positive_kernel(2 * k_process, 0.0).real
    normalized = values / math.sqrt(energy)
    fig, axes = plt.subplots(2, 1, figsize=(12.5, 6.5), sharex=True)
    axes[0].plot(time_axis, normalized.real, color=BLUE, label="real")
    axes[0].plot(time_axis, normalized.imag, color=ORANGE, label="imaginary")
    axes[0].set_ylabel("normalized amplitude")
    axes[0].set_title("unit-average-power normalization")
    axes[0].grid(alpha=0.18)
    axes[0].legend()
    axes[1].plot(time_axis, np.abs(normalized) ** 2, color=TEAL)
    axes[1].axhline(1, color=INK, ls="--", label="long-time mean power")
    axes[1].set_xlabel("logarithmic time $t$")
    axes[1].set_ylabel("instantaneous power")
    axes[1].set_title("time zero is one distinguished recurrence time")
    axes[1].grid(alpha=0.18)
    axes[1].legend()
    fig.tight_layout()
    save_figure(fig, "18_normalized_process")

    coherence_ks = np.unique(np.logspace(1, 4.3, 90).astype(int))
    coherences = []
```

```
    for k in coherence_ks:
        signed = data.signed_transfer(int(k), 0.0)
        positive = data.positive_kernel(2 * int(k), 0.0).real
        coherences.append(abs(signed) ** 2 / positive)
    fig, ax = plt.subplots(figsize=(12.0, 4.8))
    ax.loglog(coherence_ks, coherences, color=BLUE, lw=1.4)
    ax.set_xlabel("depth $k$")
    ax.set_ylabel(r"$\Gamma_k=|C_k(0)|^2/E_k$")
    ax.set_title("Finite principal Mobius coherence data")
    ax.grid(alpha=0.18)
    ax.text(0.02, 0.06, "Finite truncation and finite depth only. No asymptotic conclusion is drawn.", transform=ax.transAxes, color=MUTED, fontsize=9.5)
    save_figure(fig, "19_time_zero_coherence")

    # Figures 20-21: signed resolvent and functional equation
    grid_x = np.linspace(-2.2, 0.95, 110)
    grid_y = np.linspace(-2.0, 2.0, 100)
    magnitude = np.empty((grid_y.size, grid_x.size))
    for row, imag in enumerate(grid_y):
        zrow = grid_x + 1j * imag
        denominators = 1.0 - zrow[:, None] * plot_data.y[None, :]
        values = np.sum(plot_data.signed_amplitude[None, :] / denominators, axis=1)
        magnitude[row, :] = np.abs(values)
    fig, ax = plt.subplots(figsize=(8.0, 6.0))
    image = ax.imshow(np.log10(np.maximum(magnitude, 1e-12)), origin="lower", extent=[grid_x[0], grid_x[-1], grid_y[0], grid_y[-1]], cmap="magma", aspect="auto")
    ax.set_xlabel("real $z$")
    ax.set_ylabel("imaginary $z$")
    ax.set_title("Signed Mobius resolvent on the slit-plane side of the cut")
    fig.colorbar(image, ax=ax, fraction=0.046, label=r"$\log_{10}|C(z)|$")
    save_figure(fig, "20_signed_resolvent")

    shifted_data = build_number_packet_data(KernelParameters(a=4.0, beta=2.0, conductor=1), nmax=600_000)
    z_points = [complex(-0.8, 0.2), complex(0.25, 0.4), complex(0.7, -0.3)]
    functional_residuals = [abs(signed_resolvent_functional_residual(data, shifted_data, z)) for z in z_points]
    record("signed resolvent parameter-shift equation", max(functional_residuals) < 5e-7, functional_residuals)
    fig, ax = plt.subplots(figsize=(11.5, 4.8))
    labels = [f"{z.real:+.2f}{z.imag:+.2f}i" for z in z_points]
    ax.bar(np.arange(len(labels)), functional_residuals, color=[BLUE, TEAL, ORANGE])
    ax.set_yscale("log")
    ax.set_xticks(np.arange(len(labels)), labels)
    ax.set_ylabel("absolute residual")
    ax.set_title("Numerical certificate for the exact resolvent transfer equation")
    ax.grid(axis="y", alpha=0.18)
    save_figure(fig, "21_resolvent_functional_equation")

    # Figures 22-24: positive and signed Pade packet models
    signed_moments = np.array([complex(exact_signed_moment(k)) for k in range(16)])
    positive_moments = np.array([data.positive_companion_moment(k) for k in range(16)], dtype=np.complex128)
    positive_nodes, positive_weights = pade_packet_nodes_weights(positive_moments, 4)
    signed_nodes, signed_weights = pade_packet_nodes_weights(signed_moments, 4)
    record("positive Pade nodes remain in physical interval", bool(np.all(np.abs(positive_nodes.imag) < 1e-8) and np.all((positive_nodes.real >= -1e-8) & (positive_nodes.real < 1 + 1e-8))), [complex(v) for v in positive_nodes])
    record("first signed one-node packet lies outside physical interval", abs((signed_moments[1] / signed_moments[0]).real + 0.51981775) < 1e-6, complex(signed_moments[1] / signed_moments[0]))
    fig, axes = plt.subplots(1, 2, figsize=(12.6, 4.5))
    axes[0].scatter(positive_nodes.real, positive_nodes.imag, s=80, color=TEAL)
    axes[0].axvspan(0, 1, color=TEAL, alpha=0.08)
    axes[0].axhline(0, color=GRID)
    axes[0].set_xlim(-0.1, 1.1)
    axes[0].set_ylim(-0.1, 0.1)
    axes[0].set_title("positive companion nodes")
    axes[0].set_xlabel("real part")
    axes[0].set_ylabel("imaginary part")
    axes[1].scatter(signed_nodes.real, signed_nodes.imag, s=80, color=RED)
    axes[1].axvspan(0, 1, color=TEAL, alpha=0.08)
    axes[1].axhline(0, color=GRID)
    axes[1].set_xlim(min(-0.1, signed_nodes.real.min() - 0.05), 1.1)
    axes[1].set_ylim(-0.1, 0.1)
    axes[1].set_title("signed Mobius packet nodes")
    axes[1].set_xlabel("real part")
```

```
    fig.suptitle("Positive reduction stays on the physical spectrum. Signed reduction need not.", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "22_pade_nodes")

    positive_num, positive_den = pade_from_moments(positive_moments, 4)
    signed_num, signed_den = pade_from_moments(signed_moments, 4)
    positive_reconstructed = pade_series_coefficients(positive_num, positive_den, 12)
    signed_reconstructed = pade_series_coefficients(signed_num, signed_den, 12)
    positive_errors = np.abs(positive_reconstructed - positive_moments[:12])
    signed_errors = np.abs(signed_reconstructed - signed_moments[:12])
    record("positive Pade model matches first eight moments", float(np.max(positive_errors[:8])) < 1e-12, float(np.max(positive_errors[:8])))
    record("signed Pade model matches first eight moments", float(np.max(signed_errors[:8])) < 1e-11, float(np.max(signed_errors[:8])))
    fig, ax = plt.subplots(figsize=(11.5, 4.8))
    ax.semilogy(range(12), np.maximum(positive_errors, 1e-18), marker="o", color=TEAL, label="positive companion")
    ax.semilogy(range(12), np.maximum(signed_errors, 1e-18), marker="o", color=RED, label="signed Mobius")
    ax.axvline(7.5, color=INK, ls="--", label="exact contract ends")
    ax.set_xlabel("moment order")
    ax.set_ylabel("absolute error")
    ax.set_title("Four-packet Pade models match the first eight moments exactly")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "23_pade_moment_matching")

    orders = np.arange(1, 8)
    positive_determinants = np.array([hankel_determinant(positive_moments, int(order)).real for order in orders])
    signed_determinants = np.array([hankel_determinant(signed_moments, int(order)).real for order in orders])
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    axes[0].semilogy(orders, positive_determinants, marker="o", color=TEAL)
    axes[0].set_xlabel("Hankel order")
    axes[0].set_ylabel("determinant")
    axes[0].set_title("positive determinants")
    axes[0].grid(alpha=0.18)
    axes[1].plot(orders, np.sign(signed_determinants) * np.log10(np.maximum(np.abs(signed_determinants), 1e-300)), marker="o", color=RED)
    axes[1].axhline(0, color=GRID)
    axes[1].set_xlabel("Hankel order")
    axes[1].set_ylabel(r"signed $\log_{10}|\Delta_m|$")
    axes[1].set_title("signed determinants change sign")
    axes[1].grid(alpha=0.18)
    fig.suptitle("The signed moment problem is quasi-definite rather than positive", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "24_hankel_contrast")

    # Figure 25: Borel/Riesz bridge
    x_values = np.linspace(0.2, 12, 150)
    borel_values = np.array([borel_transfer(plot_data, float(x)) for x in x_values])
    riesz_values = np.array([riesz_companion(plot_data, float(x)) for x in x_values])
    bridge_residual = np.max(np.abs(borel_values - np.exp(x_values) * riesz_values / x_values))
    record("Borel and Riesz bridge", float(bridge_residual) < 2e-10, float(bridge_residual))
    fig, axes = plt.subplots(1, 2, figsize=(13.0, 4.8))
    axes[0].plot(x_values, borel_values.real, color=BLUE, label="Borel transform")
    axes[0].plot(x_values, (np.exp(x_values) * riesz_values / x_values).real, color=ORANGE, ls="--", label="Riesz form")
    axes[0].set_xlabel("$x$")
    axes[0].set_ylabel("real value")
    axes[0].set_title("two exact continuous coordinates")
    axes[0].grid(alpha=0.18)
    axes[0].legend()
    axes[1].semilogy(x_values, np.maximum(np.abs(borel_values - np.exp(x_values) * riesz_values / x_values), 1e-18), color=RED)
    axes[1].set_xlabel("$x$")
    axes[1].set_ylabel("absolute residual")
    axes[1].set_title("bridge residual")
    axes[1].grid(alpha=0.18)
    fig.suptitle("The signed depth sequence and the Riesz function share one Borel transform", fontsize=17, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "25_borel_riesz_bridge")
```

```
    # Figure 26: quantum operator diagram
    fig, ax = diagram_axes()
    ax.text(0.5, 0.94, "Number-basis realization of the signed recurrence process", ha="center", fontsize=20, weight="bold")
    box(ax, (0.04, 0.58), 0.18, 0.16, r"$|\phi_a\rangle$" + "\nnumber-basis seed", face=BLUE + "18", edge=BLUE, fontsize=13)
    box(ax, (0.29, 0.58), 0.18, 0.16, r"$e^{-kK_\beta}$" + "\nscale filter", face=TEAL + "18", edge=TEAL, fontsize=13)
    box(ax, (0.54, 0.58), 0.18, 0.16, "$e^{-itH}$\nlogarithmic phase", face=ORANGE + "18", edge=ORANGE, fontsize=13)
    box(ax, (0.79, 0.58), 0.17, 0.16, r"$M_\chi$" + "\narithmetic sign", face=MAGENTA + "18", edge=MAGENTA, fontsize=13)
    for left, right in [((0.22, 0.66), (0.29, 0.66)), ((0.47, 0.66), (0.54, 0.66)), ((0.72, 0.66), (0.79, 0.66))]:
        arrow(ax, left, right)
    box(ax, (0.13, 0.20), 0.27, 0.15, "signed amplitude\n$C_k(t)$", face=RED + "18", edge=RED, fontsize=14)
    box(ax, (0.60, 0.20), 0.27, 0.15, "time-averaged power\n" + r"$K_{2k}(\tau)$", face=TEAL + "18", edge=TEAL, fontsize=14)
    arrow(ax, (0.88, 0.58), (0.27, 0.35), color=RED, text="coherent readout", text_offset=(0, -0.04))
    arrow(ax, (0.88, 0.58), (0.73, 0.35), color=TEAL, text="autocorrelation", text_offset=(0, 0.04))
    save_figure(fig, "26_quantum_operator")

    # Figure 27: status boundary
    fig, ax = diagram_axes()
    ax.text(0.5, 0.94, "What is proved and what remains open", ha="center", fontsize=21, weight="bold")
    box(ax, (0.06, 0.56), 0.25, 0.22, "proved\npositive kernel\nGamma dephasing\ncharacter decoherence", face=TEAL + "20", edge=TEAL, fontsize=12.5)
    box(ax, (0.375, 0.56), 0.25, 0.22, "proved conditionally\nsigned Pade packets\nwhen Hankel systems\ndo not break down", face=ORANGE + "20", edge=ORANGE, fontsize=12.5)
    box(ax, (0.69, 0.56), 0.25, 0.22, "open\nsubpolynomial time-zero\nMobius coherence", face=RED + "20", edge=RED, fontsize=13)
    ax.text(0.5, 0.28, "The positive autocorrelation is universal. RH remains the claim that one distinguished recurrence time is not polynomially exceptional.", ha="center", fontsize=12.5, color=MUTED, wrap=True)
    save_figure(fig, "27_status_boundary")

    # Figure 28: family map
    fig, ax = diagram_axes()
    ax.text(0.5, 0.95, "Function family generated in Paper IV", ha="center", fontsize=21, weight="bold")
    items = [
        (0.06, 0.66, "depth-time kernel\n" + r"$K_{k,\ell}(\tau)$", BLUE),
        (0.38, 0.66, "Gamma dephasing\n" + r"$\Gamma(\alpha+i\tau/\beta)/\Gamma(\alpha)$", TEAL),
        (0.70, 0.66, "Mobius resolvent\n" + r"$C(z,t)$", ORANGE),
        (0.06, 0.27, "log-frequency law\npolygamma cumulants", PURPLE),
        (0.38, 0.27, "character cross-kernels\ndecoherence", MAGENTA),
        (0.70, 0.27, "signed Pade packets\nfinite moment contracts", RED),
    ]
    for x0, y0, text, color in items:
        box(ax, (x0, y0), 0.24, 0.17, text, face=color + "18", edge=color, fontsize=11.5)
    arrow(ax, (0.30, 0.745), (0.38, 0.745))
    arrow(ax, (0.62, 0.745), (0.70, 0.745))
    arrow(ax, (0.18, 0.66), (0.18, 0.44))
    arrow(ax, (0.50, 0.66), (0.50, 0.44))
    arrow(ax, (0.82, 0.66), (0.82, 0.44))
    save_figure(fig, "28_function_family_map")

    # Additional built-in checks
    # Exact low-order signed moments against direct truncation
    signed_direct = np.array([data.signed_transfer(k, 0.0).real for k in range(7)])
    signed_exact = np.array([float(exact_signed_moment(k)) for k in range(7)])
    record("signed finite-difference moments", float(np.max(np.abs(signed_direct - signed_exact))) < 2e-6, float(np.max(np.abs(signed_direct - signed_exact))))

    # Gamma asymptotic at several lags
    gamma_errors = []
    depth_check = 50000
    for tau in (0.0, 0.7, 1.5, 3.0):
        observed = depth_check ** (1j * tau / params.beta) * data.positive_kernel(depth_check, tau) / data.positive_kernel(depth_check, 0.0)
        target = complex(mp.gamma(params.alpha + 1j * tau / params.beta) / mp.gamma(params.alpha))
        gamma_errors.append(abs(observed - target))
    record("Gamma dephasing asymptotic", max(gamma_errors) < 0.03, gamma_errors)
```

```
    # Depth overlap example k=4 ell=1 after scaling
    expected_overlap = 0.8 ** params.alpha
    observed_overlap = data.positive_kernel(50000, 0.0).real / math.sqrt(
        data.positive_kernel(80000, 0.0).real * data.positive_kernel(20000, 0.0).real
    )
    record("depth overlap example", abs(observed_overlap - expected_overlap) < 0.02, [observed_overlap,
     expected_overlap])

    # Long-time mean power in a finite model
    finite_average = finite_time_average(small_data, 4, 4, 0.0, 200.0, samples=3001)
    exact_power = small_data.positive_kernel(8, 0.0)
    record("finite-model long-time mean power", abs(finite_average - exact_power) < 0.03, [finite_average,
     exact_power])

    # Character off-diagonal decreases between first and last depth
    record("character off-diagonal correlation decreases", max_offdiag[-1] < max_offdiag[0], max_offdiag)

    # Signed resolvent coefficients match moment sequence
    z_test = 0.12
    series_approx = sum(signed_moments[k] * z_test**k for k in range(14))
    direct_resolvent = signed_resolvent(data, z_test)
    record("signed resolvent coefficient expansion", abs(series_approx - direct_resolvent) < 2e-7, abs(series_approx
      - direct_resolvent))

    # Save tables
    gamma_table_path = TABLES / "gamma_dephasing.csv"
    with gamma_table_path.open("w", newline="") as handle:
        writer = csv.writer(handle)
        writer.writerow(["tau", "real", "imag", "modulus_squared"])
        for tau in (-4, -2, -1, 0, 1, 2, 4):
            value = complex(mp.gamma(params.alpha + 1j * tau / params.beta) / mp.gamma(params.alpha))
            writer.writerow([tau, value.real, value.imag, abs(value) ** 2])

    pade_table_path = TABLES / "pade_nodes.csv"
    with pade_table_path.open("w", newline="") as handle:
        writer = csv.writer(handle)
        writer.writerow(["family", "index", "node_real", "node_imag", "weight_real", "weight_imag"])
        for family, nodes, weights in [
            ("positive", positive_nodes, positive_weights),
            ("signed", signed_nodes, signed_weights),
        ]:
            order = np.argsort(nodes.real)
            for index, selected in enumerate(order):
                writer.writerow(
                    [
                        family,
                        index,
                        nodes[selected].real,
                        nodes[selected].imag,
                        weights[selected].real,
                        weights[selected].imag,
                    ]
                )

    coherence_table_path = TABLES / "coherence.csv"
    with coherence_table_path.open("w", newline="") as handle:
        writer = csv.writer(handle)
        writer.writerow(["k", "gamma_k"])
        for k, value in zip(coherence_ks, coherences, strict=True):
            writer.writerow([int(k), float(value)])

    passed = sum(int(item["passed"]) for item in checks)
    result = {
        "title": "Projector-Generated Special Functions IV",
        "author": "Komorebi P.",
        "checks": len(checks),
        "passed": passed,
        "figure_pairs": 28,
        "parameters": {
            "a": params.a,
            "beta": params.beta,
```

```
            "alpha": params.alpha,
            "nmax": int(data.n[-1]),
        },
        "principal_closed_form_residual": float(np.max(np.abs(modulus_observed - modulus_closed))),
        "gamma_asymptotic_errors": [float(v) for v in gamma_errors],
        "signed_one_node": complex(signed_moments[1] / signed_moments[0]).real,
        "checks_detail": checks,
    }
    (RESULTS / "results.json").write_text(json.dumps(result, indent=2, default=str) + "\n")
    with (RESULTS / "checks.csv").open("w", newline="") as handle:
        writer = csv.DictWriter(handle, fieldnames=["name", "passed", "detail"])
        writer.writeheader()
        for item in checks:
            writer.writerow({"name": item["name"], "passed": item["passed"], "detail": item["detail"]})
    return result


def main() -> None:
    result = build_figures_and_results()
    print(json.dumps(result, indent=2, default=str))
    if result["passed"] != result["checks"]:
        raise SystemExit("one or more built-in checks failed")


if __name__ == "__main__":
    main()
```

C.1 Reproduction

From the source directory, run

```
python projector_generated_special_functions_iv.py
python generate_tables.py
python -m pytest -q
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_IV.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_IV.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_IV.tex
```

C.2 Source boundary

The external historical inputs, prior special-function literature, and the 2026 logarithmic-spectrum quantum construction belong in the consolidated end-of-series bibliography. No historical priority claim is made for every displayed specialization. The paper claims the formulas and theorems proved here, the generated function organization, and the computational certificates supplied with the release.

Projector-Generated Special Functions V

Log-Gamma Gabor Packets, Gamma–Sech Ambiguity Kernels, Meixner–Pollaczek Spectra, and SU(1, 1) Duality

Komorebi P.

Abstract

This paper identifies the universal phase-space object hidden in the large-depth limit of the squarefree packet kernels developed in the preceding two papers. Let

$$\alpha = \frac{2a-1}{\beta} > 0.$$

After logarithmic centering, the normalized arithmetic depth-time kernel converges to

$$\boxed{\mathfrak{A}_\alpha(h,\eta) = \frac{\Gamma(\alpha+\mathrm{i}\eta)}{\Gamma(\alpha)}\operatorname{sech}^{\alpha+\mathrm{i}\eta}\left(\frac{h}{2}\right).}$$

The limit is not only a closed asymptotic formula. It is exactly the centered Gabor ambiguity function of the normalized log-Gamma packet

$$g_\alpha(y) = \Gamma(\alpha)^{-1/2}\exp\left(\frac{\alpha y}{2} - \frac{\mathrm{e}^y}{2}\right).$$

This identification generates a complete special-function system. The ambiguity function is zero-free on real phase space, obeys an exact shape shift in α, and solves a Pöschl–Teller equation. The Fourier transform of g_α is a Gamma function. Its squared modulus is the symmetric Meixner density and its characteristic function is the exact depth-overlap law $\operatorname{sech}^\alpha(h/2)$. The Wigner distribution has a Kontorovich–Lebedev closed form involving $K_{2\mathrm{i}\omega}(\mathrm{e}^y)$.

The packet has exact position and momentum variances

$$\operatorname{Var} Y = \psi_1(\alpha), \qquad \operatorname{Var} P = \frac{\alpha}{4}.$$

Its Heisenberg product is strictly larger than $1/2$ at every finite α and tends to $1/2$ as α grows. The Weyl orbit has constant quantum metric and unit Berry curvature in the selected coordinates.

The orthogonal polynomial structure is also exact. Laguerre packets on logarithmic position space transform into the same Gamma envelope multiplied by orthonormal Meixner–Pollaczek polynomials. Both bases carry one positive discrete-series representation of $\mathfrak{su}(1,1)$. The Laguerre multiplication operator and the Meixner frequency operator are two linear combinations of the same ladder generators.

Finally, the discrete arithmetic depth is extended to a continuous parameter. The resulting fractional-depth Möbius transfer admits an infinite binomial expansion in reciprocal L-values and an exact generator relation across the parameter a. The unconditional positive and representation-theoretic results do not prove the remaining principal Möbius coherence bound and do not prove RH. They isolate a universal phase-space geometry shared by the positive packet family.

All figures were generated by the accompanying Python program. The release contains 26 vector figures, matching PNG files, exact tables, 19 built-in checks, 19 independent tests, and the complete Python source.

Status and claim boundary

The paper proves the log-Gamma ambiguity representation, the Gamma–sech closed form, its zero-free real phase-space law, the parameter shift, the Pöschl–Teller equation, the Fourier and Meixner density formulas, the Wigner–Kontorovich–Lebedev identity, the exact uncertainty formulas, the constant orbit geometry, the Laguerre–Meixner Fourier duality, and the $\mathfrak{su}(1,1)$ operator realization. It also proves the continuous-depth binomial expansion and the fractional generator relation under the stated absolute-convergence hypotheses.

The squarefree arithmetic convergence theorem uses the standard squarefree counting asymptotic and is uniform only on compact sets of the ratio and time parameters described below. The associated numerical figures are finite checks of that theorem, not substitutes for it.

The classical inputs include Gamma and polygamma functions, Laguerre polynomials, Meixner–Pollaczek polynomials, Gabor ambiguity functions, Wigner distributions, the Kontorovich–Lebedev kernel, and the positive discrete series of $\mathrm{SU}(1,1)$. The generated contribution is the exact bridge showing that these objects arise from one arithmetic packet limit with one common normalization and one common ladder system.

The logarithmic quantum coordinate is compatible with the external 2026 construction based on the number-basis Hamiltonian $H\,|n\rangle = \log n\,|n\rangle$. The present observable is a scale-filtered squarefree kernel and is not the direct zeta amplitude of that work.

None of the unconditional results proves RH. The open part remains the signed time-zero coherence of the principal Möbius packet.

Contents

Status and claim boundary **i**

I The arithmetic limit and its exact packet window **1**

1 The squarefree depth-time kernel **1**

2 The log-Gamma packet **3**

3 The exact Gamma–sech ambiguity function **5**

II Fourier, Wigner, and information-geometric coordinates **8**

4 The Meixner spectral law **8**

5 The Wigner–Kontorovich–Lebedev function **10**

6 Uncertainty and quantum geometry **12**

III Orthogonal packet bases and one ladder algebra **14**

7 Laguerre packets on logarithmic position space **14**

8 Meixner–Pollaczek packets in Mellin frequency **15**

9 The common $\mathfrak{su}(1,1)$ ladder **17**

IV Fractional depth, quantum realization, and the RH boundary **20**

10 Continuous packet depth **20**

11 Finite arithmetic convergence **22**

12 Quantum and signal realization **24**

13 The remaining RH boundary **25**

V Verification, formula registry, and source **26**

14 Verification ledger **27**

15 Formula registry **28**

16 Relation to established function theory **28**

A Complete Python implementation **30**

B Reproduction **45**

List of Figures

1.1 The generated theory. The positive squarefree depth-time kernel converges to a log-Gamma Gabor packet. Exact transforms of that packet generate the ambiguity function, spectral density, Wigner function, orthogonal polynomials, and ladder operators studied in the rest of the paper. 2

2.1 Normalized log-Gamma windows. Increasing α moves the packet to larger log scale and makes the local profile increasingly Gaussian after centering. 4

2.2 Position probability densities. The squared packet is the logarithm of a Gamma random variable. Its exact mean and variance are the digamma and trigamma functions. 4

3.1 Magnitude of the ambiguity function. The generated packet has no real phase-space zeros. Decay in depth displacement comes from the hyperbolic secant. Decay in modulation comes from the Gamma factor. 6

3.2 Phase of the ambiguity function. The phase separates into a Gamma phase and a depth-dependent carrier produced by the complex power of the hyperbolic secant. 6

3.3 One-dimensional ambiguity slices. The left panel varies the depth displacement at fixed Mellin frequencies. The right panel varies the Mellin-frequency shift at fixed depth displacements. 7

3.4 Pöschl–Teller realization. The left panel shows the complex potential coefficient. The right panel verifies the differential equation numerically at high resolution. 8

4.1 Symmetric Meixner spectral densities. The logarithmic packet has a Gamma-function Fourier envelope. Its squared modulus is the orthogonality density used by the Meixner–Pollaczek packet basis. 9

4.2 Exact depth autocorrelation. The universal cross-depth overlap is the characteristic function of the Meixner density and the zero-modulation slice of the ambiguity function. . 9

5.1 The Wigner–Kontorovich–Lebedev packet. The distribution is an exact Bessel function of imaginary order. Positive and negative regions encode the nonclassical phase-space interference of the packet. 11

5.2 Wigner marginals. Numerical integration of the Bessel closed form reproduces both the exact log-Gamma position density and the exact Meixner spectral density. 11

6.1 Exact uncertainty product. The packet is not a Gaussian at finite shape, but after centering and large-α scaling it approaches the minimum-uncertainty regime. 13

6.2 Quantum geometry of the Weyl orbit. Translation and modulation inherit exact metric coefficients from the packet variances. The canonical commutator supplies the constant symplectic curvature. 14

7.1 Laguerre packets in log coordinates. The packets form an exact orthonormal multioscillation family around the universal log-Gamma envelope. 15

8.1 Meixner–Pollaczek packets. Multiplying each orthonormal polynomial by the square root of its Gamma density gives the frequency-side packet family. 16

8.2 Exact Fourier duality. The real and imaginary parts alternate with degree through the phase i^n. The Gamma envelope is common to every packet. 17

9.1 Laguerre Jacobi matrix. The position-side multiplication operator is tridiagonal in the logarithmic Laguerre packet basis. 18

9.2 Meixner Jacobi matrix. The Mellin-frequency operator is another tridiagonal combination of the same ladder generators. 19

9.3 One ladder and two atlases. The Laguerre and Meixner operators are not unrelated recurrences. They are two projections of the same positive discrete-series ladder. 20

10.1 Continuous depth. The integer Bernoulli finite-difference orbit is embedded into a smooth depth evolution generated by an infinite parameter-shift operator. 21

10.2 Fractional binomial reconstruction. The infinite reciprocal-zeta expansion converges to the direct Möbius packet calculation for the noninteger depth $r = 5/2$. 22

11.1 Arithmetic convergence. The finite squarefree packet moves toward the universal Gamma–sech kernel as the active scale grows. The displayed reference line is only a visual slope guide. 23

11.2 Complex trajectory comparison. After centering, the finite arithmetic correlation and the universal phase-space curve nearly coincide over the tested time interval. 23

12.1 Packet lattice. Circle area is proportional to the exact overlap magnitude. The generated function supplies a natural nonuniform coherent-state geometry in log depth and Mellin frequency. 24

12.2 Quantum and signal path. The logarithmic Hamiltonian supplies the time phase. The depth filter supplies scale selection. Their centered overlap converges to the universal log-Gamma packet geometry. 25

13.1 The generated family. The Gamma–sech ambiguity function, log-Gamma window, Meixner spectrum, Wigner–Bessel function, Laguerre–Meixner Fourier pair, ladder algebra, and fractional-depth transfer are exact coordinates of one universal packet system. 26

Part I

The arithmetic limit and its exact packet window

Chapter 1

The squarefree depth-time kernel

Fix

$$a > \frac{1}{2}, \qquad \beta > 0, \qquad \alpha = \frac{2a-1}{\beta}.$$

The positive depth-time kernel is

$$\boxed{\mathcal{K}_{a,\beta}(k,\ell,\tau) = \sum_{n\geq 1} \frac{\mu(n)^2}{n^{2a+\mathrm{i}\tau}} \left(1 - \frac{1}{n^\beta}\right)^{k+\ell}.}$$

When a character conductor is present, one adds the coprimality restriction or the factor $|\chi(n)|^2$. The principal formulas below are unchanged apart from the squarefree density constant.

Set

$$m = k + \ell, \qquad \eta = \frac{\tau}{\beta}.$$

The active integers satisfy $n^\beta \asymp m$. This is the depth localization found in Paper III. Paper IV derived the separate time and cross-depth limits. The first result here joins them.

Theorem 1.1 (Joint arithmetic phase-space limit)**.** *Assume that $k, \ell \to \infty$ while k/ℓ remains in a compact subset of $(0,\infty)$ and τ remains in a compact real interval. Let*

$$h = \log \frac{k}{\ell}.$$

Then

$$\boxed{\mathrm{e}^{\mathrm{i}\eta \log(2\sqrt{k\ell})} \frac{\mathcal{K}_{a,\beta}(k,\ell,\tau)}{\sqrt{\mathcal{K}_{a,\beta}(k,k,0)\mathcal{K}_{a,\beta}(\ell,\ell,0)}} \longrightarrow \mathfrak{A}_\alpha(h,\eta),}$$

where

$$\boxed{\mathfrak{A}_\alpha(h,\eta) = \frac{\Gamma(\alpha+\mathrm{i}\eta)}{\Gamma(\alpha)} \operatorname{sech}^{\alpha+\mathrm{i}\eta}\left(\frac{h}{2}\right).}$$

Proof. Let

$$A(x) = \sum_{n\leq x} \mu(n)^2.$$

The squarefree counting asymptotic is

$$A(x) = \delta x + O(x^{1/2}), \qquad \delta = \frac{1}{\zeta(2)}.$$

Partial summation converts the kernel into a Stieltjes integral against A. On replacing $A(x)$ by its main term and scaling

$$x = m^{1/\beta} u,$$

one obtains

$$\mathcal{K}_{a,\beta}(k,\ell,\tau) \sim \frac{\delta}{\beta}\Gamma(\alpha+\mathrm{i}\eta)m^{-\alpha-\mathrm{i}\eta}.$$

The same calculation at $(k,k,0)$ and $(\ell,\ell,0)$ gives

$$\mathcal{K}_{a,\beta}(k,k,0) \sim \frac{\delta}{\beta}\Gamma(\alpha)(2k)^{-\alpha}$$

and the analogous formula for ℓ. Therefore the normalized magnitude factor is

$$\left(\frac{2\sqrt{k\ell}}{k+\ell}\right)^{\alpha} = \operatorname{sech}^{\alpha}\left(\frac{h}{2}\right).$$

The carrier phase leaves

$$\left(\frac{2\sqrt{k\ell}}{k+\ell}\right)^{\mathrm{i}\eta} = \operatorname{sech}^{\mathrm{i}\eta}\left(\frac{h}{2}\right).$$

Combining these terms proves the formula. Uniformity on compact parameter sets follows from the same partial-summation estimate because the scaled kernels and their first derivatives remain integrably dominated there. □

From arithmetic depth packets to a universal phase-space function

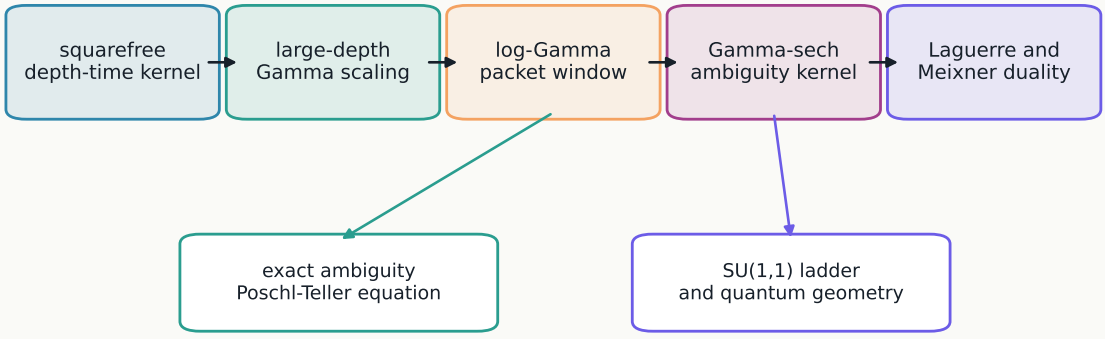


Figure 1.1: The generated theory. The positive squarefree depth-time kernel converges to a log-Gamma Gabor packet. Exact transforms of that packet generate the ambiguity function, spectral density, Wigner function, orthogonal polynomials, and ladder operators studied in the rest of the paper.

Worked example 1.2 (Principal parameters)**.** For

$$a = 2, \qquad \beta = 2,$$

one has

$$\alpha = \frac{3}{2}.$$

Taking $\ell = 4k$ gives

$$h = -\log 4.$$

At $\tau = 2.4$, one has $\eta = 1.2$. The exact limiting value is

$$\mathfrak{A}_{3/2}(-\log 4, 1.2).$$

The finite squarefree kernels used in figures 11.1 and 11.2 converge to this complex value after the carrier phase is removed.

Chapter 2

The log-Gamma packet

Definition 2.1 (Normalized log-Gamma window)**.** For $\alpha > 0$, define

$$\boxed{g_\alpha(y) = \frac{1}{\sqrt{\Gamma(\alpha)}} \exp\!\left(\frac{\alpha y}{2} - \frac{\mathrm{e}^y}{2}\right), \qquad y \in \mathbb{R}.}$$

Its squared modulus is

$$\boxed{|g_\alpha(y)|^2 = \frac{1}{\Gamma(\alpha)} \exp(\alpha y - \mathrm{e}^y).}$$

Under the change of variables

$$u = \mathrm{e}^y,$$

this is the Gamma$(\alpha, 1)$ law in logarithmic coordinates.

Proposition 2.2 (Normalization)**.** *For every $\alpha > 0$,*

$$\|g_\alpha\|_{L^2(\mathbb{R})} = 1.$$

Proof. The substitution $u = \mathrm{e}^y$ gives

$$\int_{-\infty}^{\infty} |g_\alpha(y)|^2 \,\mathrm{d}y = \frac{1}{\Gamma(\alpha)} \int_0^\infty u^{\alpha-1}\mathrm{e}^{-u} \,\mathrm{d}u = 1.$$

□

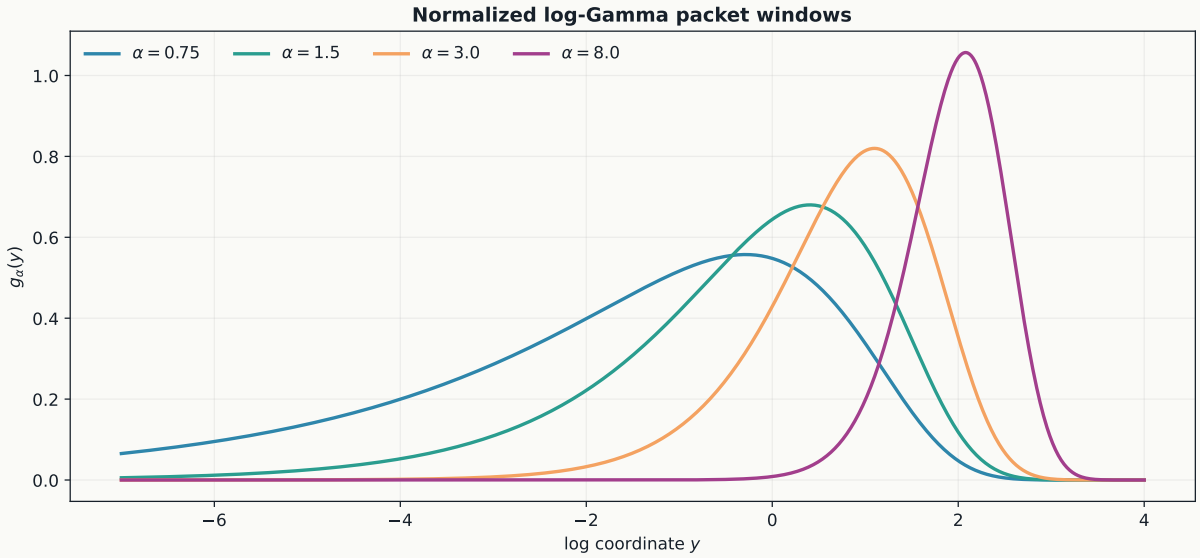


Figure 2.1: Normalized log-Gamma windows. Increasing α moves the packet to larger log scale and makes the local profile increasingly Gaussian after centering.

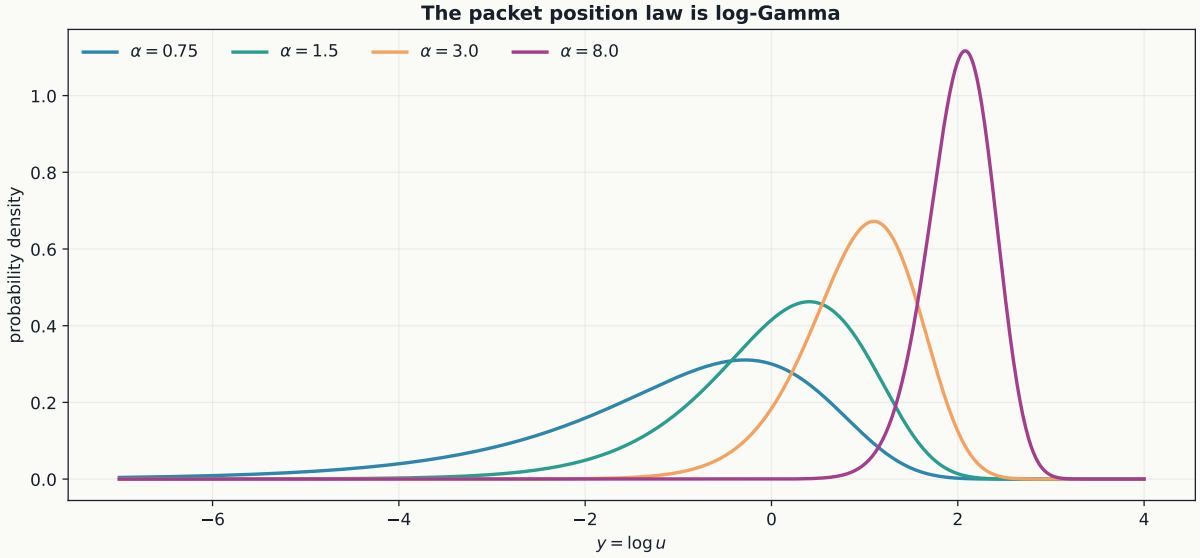


Figure 2.2: Position probability densities. The squared packet is the logarithm of a Gamma random variable. Its exact mean and variance are the digamma and trigamma functions.

Chapter 3

The exact Gamma–sech ambiguity function

Use the centered ambiguity convention

$$\mathcal{A}_g(h,\eta) = \int_{\mathbb{R}} g\Big(y + \frac{h}{2}\Big)\,\overline{g\Big(y - \frac{h}{2}\Big)}\,\mathrm{e}^{\mathrm{i}\eta y}\,\mathrm{d}y.$$

The packet is real, so the conjugate may be omitted.

Theorem 3.1 (Exact ambiguity formula)**.** *For every $\alpha > 0$ and real h, η,*

$$\boxed{\mathcal{A}_{g_\alpha}(h,\eta) = \mathfrak{A}_\alpha(h,\eta) = \frac{\Gamma(\alpha + \mathrm{i}\eta)}{\Gamma(\alpha)}\,\operatorname{sech}^{\alpha+\mathrm{i}\eta}\Big(\frac{h}{2}\Big).}$$

Proof. The product of the two shifted windows is

$$\frac{1}{\Gamma(\alpha)}\exp\Big(\alpha y - \mathrm{e}^y \cosh\frac{h}{2}\Big).$$

Therefore

$$\mathcal{A}_{g_\alpha}(h,\eta) = \frac{1}{\Gamma(\alpha)}\int_{\mathbb{R}} \exp\Big((\alpha + \mathrm{i}\eta)y - \mathrm{e}^y \cosh\frac{h}{2}\Big)\,\mathrm{d}y.$$

Set

$$u = \mathrm{e}^y \cosh\frac{h}{2}.$$

The integral becomes

$$\frac{\Gamma(\alpha + \mathrm{i}\eta)}{\Gamma(\alpha)}\cosh^{-\alpha-\mathrm{i}\eta}\Big(\frac{h}{2}\Big).$$

□

Corollary 3.2 (Real phase-space nonvanishing)**.** *For every real h and η,*

$$\boxed{\mathfrak{A}_\alpha(h,\eta) \neq 0.}$$

Proof. The Gamma function has no zeros. The hyperbolic secant is strictly positive on the real line. □

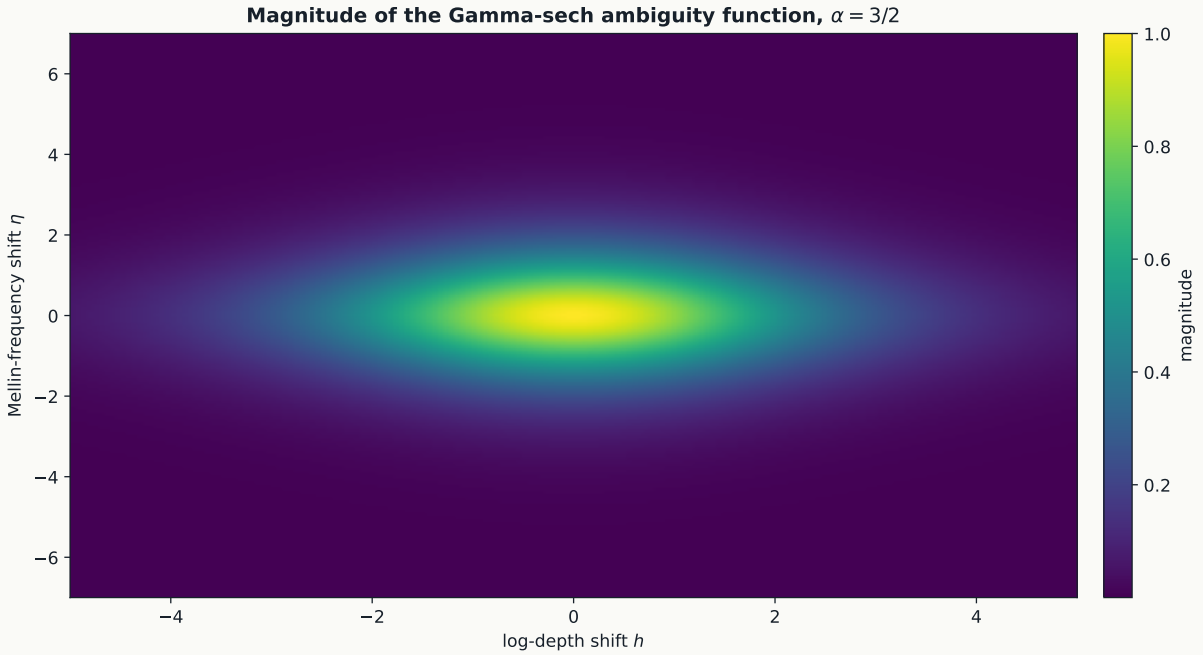


Figure 3.1: Magnitude of the ambiguity function. The generated packet has no real phase-space zeros. Decay in depth displacement comes from the hyperbolic secant. Decay in modulation comes from the Gamma factor.

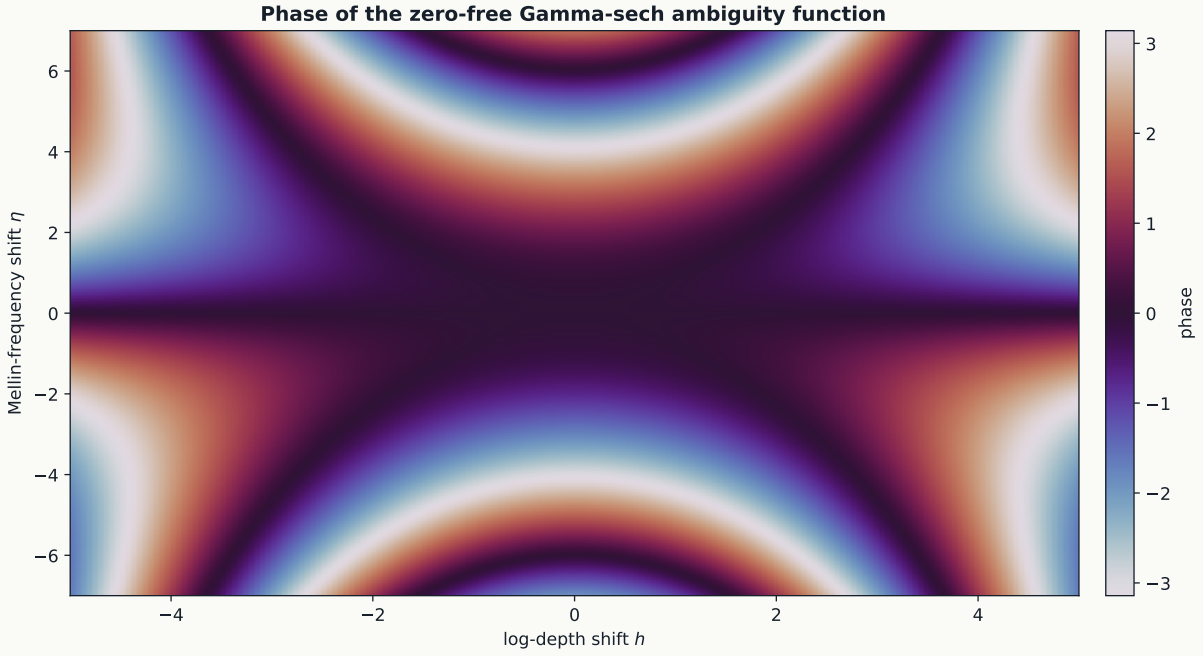


Figure 3.2: Phase of the ambiguity function. The phase separates into a Gamma phase and a depth-dependent carrier produced by the complex power of the hyperbolic secant.

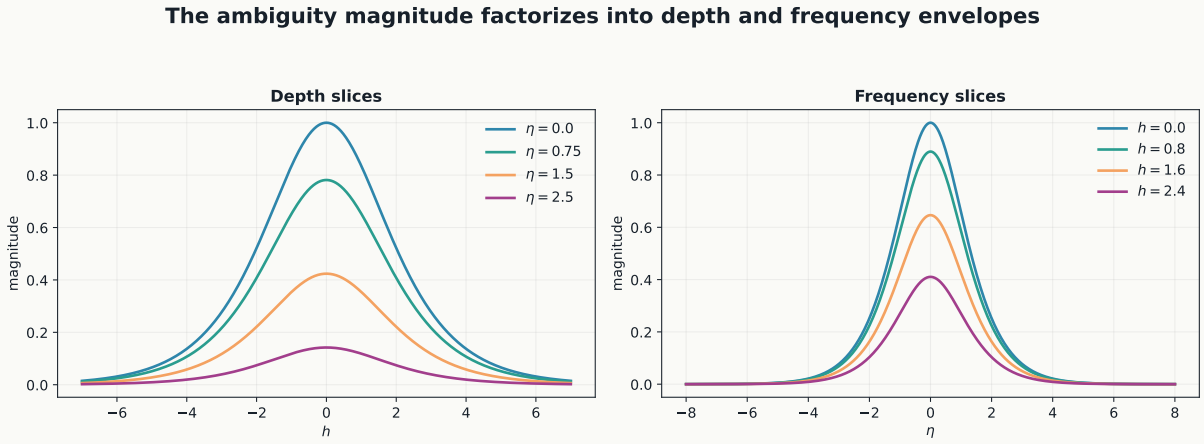


Figure 3.3: One-dimensional ambiguity slices. The left panel varies the depth displacement at fixed Mellin frequencies. The right panel varies the Mellin-frequency shift at fixed depth displacements.

Proposition 3.3 (Shape shift)**.** *The family satisfies*

$$\boxed{\mathfrak{A}_{\alpha+1}(h,\eta) = \frac{\alpha+\mathrm{i}\eta}{\alpha}\,\operatorname{sech}\!\left(\frac{h}{2}\right)\mathfrak{A}_{\alpha}(h,\eta).}$$

Proof. Use

$$\Gamma(\alpha+1+\mathrm{i}\eta) = (\alpha+\mathrm{i}\eta)\Gamma(\alpha+\mathrm{i}\eta)$$

and

$$\Gamma(\alpha+1) = \alpha\Gamma(\alpha).$$

□

Theorem 3.4 (Pöschl–Teller equation)**.** *Let*

$$\nu = \alpha + \mathrm{i}\eta.$$

For fixed η, the function $h \mapsto \mathfrak{A}_{\alpha}(h,\eta)$ satisfies

$$\boxed{\left[-4\frac{\mathrm{d}^2}{\mathrm{d}h^2} - \nu(\nu+1)\operatorname{sech}^2\!\left(\frac{h}{2}\right)\right]\mathfrak{A}_{\alpha}(h,\eta) = -\nu^2\mathfrak{A}_{\alpha}(h,\eta).}$$

Proof. The Gamma prefactor is independent of h. Direct differentiation of

$$f(h) = \operatorname{sech}^{\nu}(h/2)$$

gives

$$4f''(h) = \left[\nu^2 - \nu(\nu+1)\operatorname{sech}^2(h/2)\right] f(h).$$

Rearranging proves the equation. □

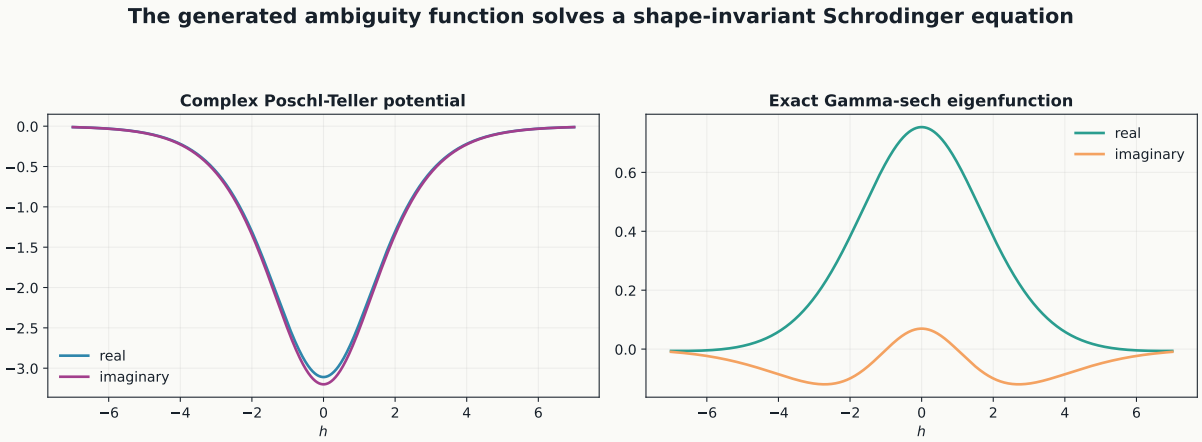


Figure 3.4: Pöschl–Teller realization. The left panel shows the complex potential coefficient. The right panel verifies the differential equation numerically at high resolution.

Part II

Fourier, Wigner, and information-geometric coordinates

Chapter 4

The Meixner spectral law

Use the unitary Fourier convention

$$\widehat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(y) \mathrm{e}^{-\mathrm{i}\omega y} \, \mathrm{d}y.$$

Theorem 4.1 (Fourier transform of the log-Gamma packet)**.** *For real* ω*,*

$$\boxed{\widehat{g}_\alpha(\omega) = \frac{2^{\alpha/2-\mathrm{i}\omega}\Gamma(\alpha/2 - \mathrm{i}\omega)}{\sqrt{2\pi\Gamma(\alpha)}}.}$$

Proof. Set $x = \mathrm{e}^y$. Then

$$\widehat{g}_\alpha(\omega) = \frac{1}{\sqrt{2\pi\Gamma(\alpha)}} \int_0^\infty x^{\alpha/2-\mathrm{i}\omega-1} \mathrm{e}^{-x/2} \, \mathrm{d}x.$$

The Gamma integral gives the result. □

The spectral probability density is

$$\boxed{\rho_\alpha(\omega) = |\widehat{g}_\alpha(\omega)|^2 = \frac{2^{\alpha-1}}{\pi\Gamma(\alpha)} \left|\Gamma\left(\frac{\alpha}{2} + \mathrm{i}\omega\right)\right|^2.}$$

Corollary 4.2 (Exact characteristic function)**.**

$$\boxed{\int_{\mathbb{R}} \rho_\alpha(\omega) \mathrm{e}^{\mathrm{i}\omega h} \, \mathrm{d}\omega = \operatorname{sech}^\alpha\left(\frac{h}{2}\right).}$$

Proof. The left side is the Fourier autocorrelation of g_α. The ambiguity theorem at $\eta = 0$ gives the right side. □

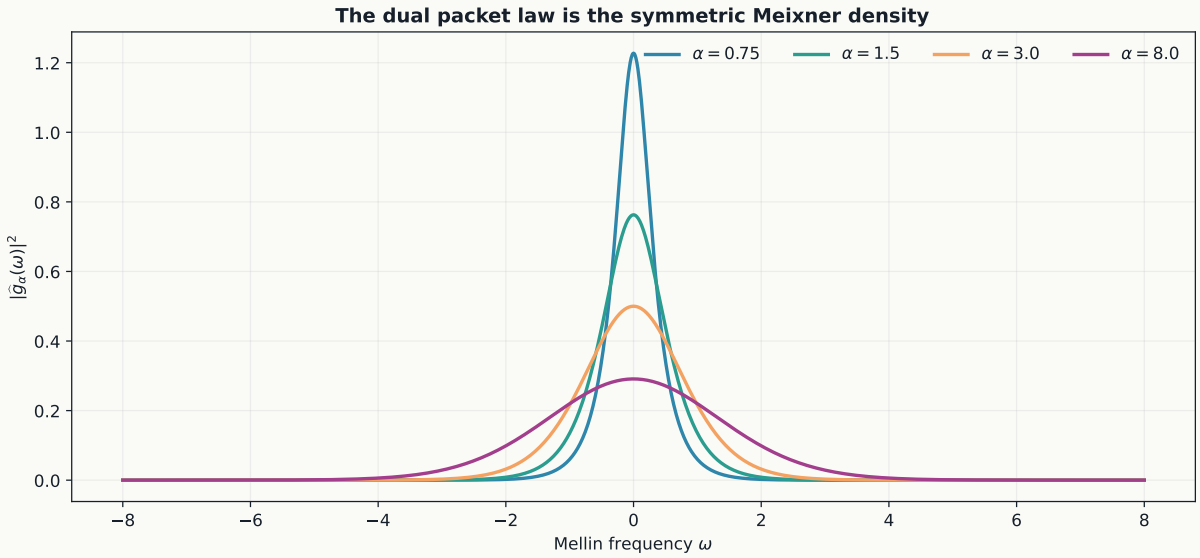


Figure 4.1: Symmetric Meixner spectral densities. The logarithmic packet has a Gamma-function Fourier envelope. Its squared modulus is the orthogonality density used by the Meixner–Pollaczek packet basis.

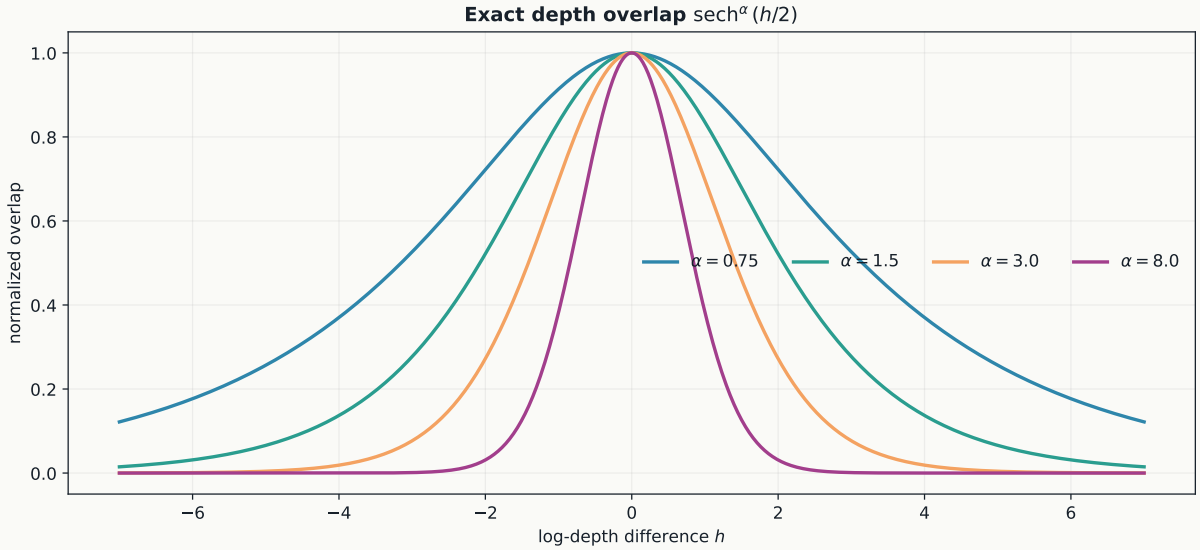


Figure 4.2: Exact depth autocorrelation. The universal cross-depth overlap is the characteristic function of the Meixner density and the zero-modulation slice of the ambiguity function.

Chapter 5

The Wigner–Kontorovich–Lebedev function

Define the Wigner distribution by

$$W_\alpha(y,\omega) = \frac{1}{2\pi}\int_{\mathbb{R}} g_\alpha\left(y+\frac{h}{2}\right) g_\alpha\left(y-\frac{h}{2}\right) \mathrm{e}^{-\mathrm{i}\omega h}\,\mathrm{d}h.$$

Theorem 5.1 (Closed Wigner formula)**.**

$$\boxed{W_\alpha(y,\omega) = \frac{2\mathrm{e}^{\alpha y}}{\pi\Gamma(\alpha)} K_{2\mathrm{i}\omega}(\mathrm{e}^{y}).}$$

Proof. The product of the shifted windows equals

$$\Gamma(\alpha)^{-1}\exp(\alpha y - \mathrm{e}^{y}\cosh(h/2))\,.$$

Set $h = 2t$. The remaining integral is

$$2\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{e}^{y}\cosh t}\mathrm{e}^{-2\mathrm{i}\omega t}\,\mathrm{d}t.$$

The integral representation of the modified Bessel function gives

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x\cosh t}\mathrm{e}^{-2\mathrm{i}\omega t}\,\mathrm{d}t = 2K_{2\mathrm{i}\omega}(x).$$

Substitution yields the formula. □

Corollary 5.2 (Exact marginals)**.**

$$\boxed{\int_{\mathbb{R}} W_\alpha(y,\omega)\,\mathrm{d}\omega = |g_\alpha(y)|^2}$$

and

$$\boxed{\int_{\mathbb{R}} W_\alpha(y,\omega)\,\mathrm{d}y = \rho_\alpha(\omega).}$$

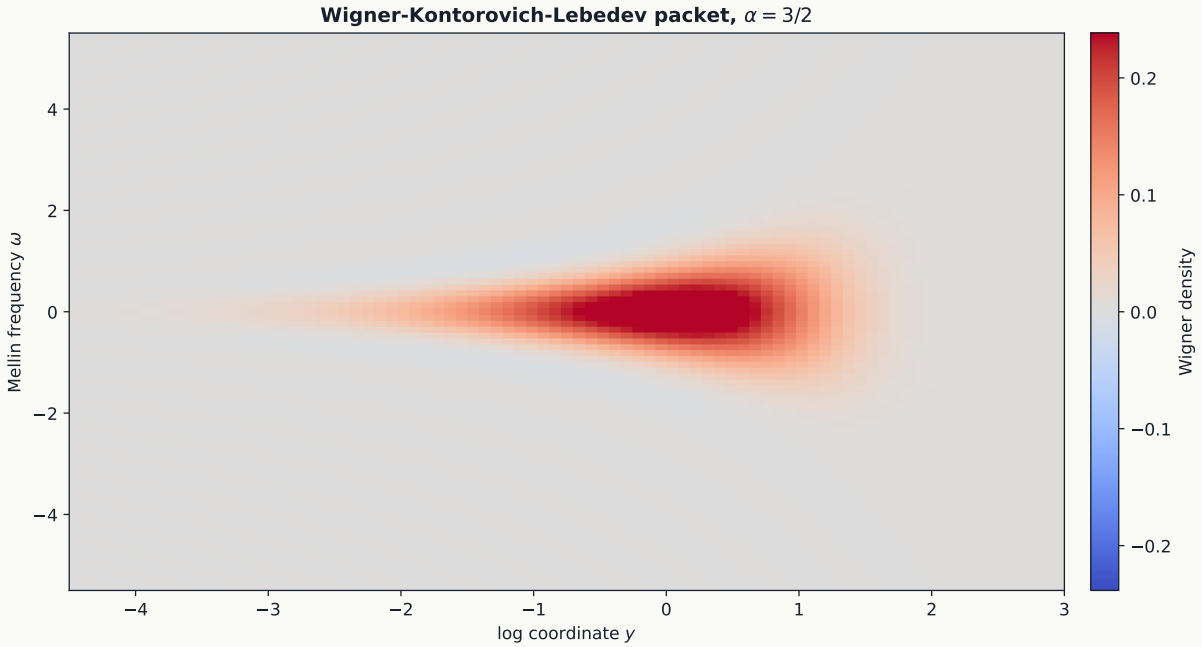


Figure 5.1: The Wigner–Kontorovich–Lebedev packet. The distribution is an exact Bessel function of imaginary order. Positive and negative regions encode the nonclassical phase-space interference of the packet.

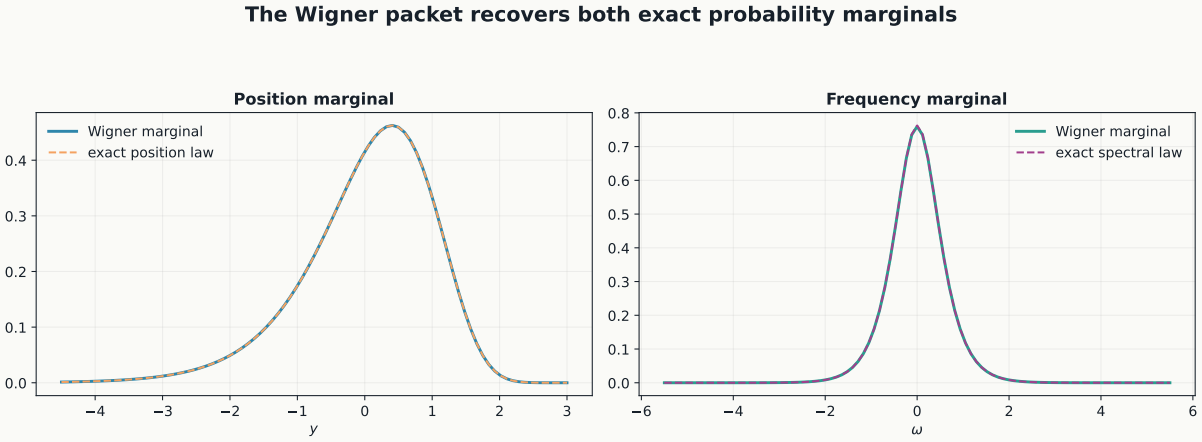


Figure 5.2: Wigner marginals. Numerical integration of the Bessel closed form reproduces both the exact log-Gamma position density and the exact Meixner spectral density.

Chapter 6

Uncertainty and quantum geometry

Let

$$Yf(y) = yf(y), \qquad Pf(y) = -\mathrm{i}f'(y).$$

Theorem 6.1 (Exact moments and uncertainty)**.** *For the state g_α,*

$$\boxed{\mathbb{E}Y = \psi(\alpha), \qquad \operatorname{Var} Y = \psi_1(\alpha),}$$

while

$$\boxed{\mathbb{E}P = 0, \qquad \operatorname{Var} P = \frac{\alpha}{4}.}$$

Consequently,

$$\boxed{\Delta Y\,\Delta P = \frac{1}{2}\sqrt{\alpha\psi_1(\alpha)} > \frac{1}{2}.}$$

Moreover,

$$\Delta Y\,\Delta P \longrightarrow \frac{1}{2}$$

as $\alpha \to \infty$.

Proof. The variable $U = \mathrm{e}^Y$ has the Gamma$(\alpha, 1)$ law. Its logarithmic moments give

$$\mathbb{E}Y = \psi(\alpha), \qquad \operatorname{Var} Y = \psi_1(\alpha).$$

For momentum,

$$\frac{g'_\alpha(y)}{g_\alpha(y)} = \frac{\alpha - \mathrm{e}^y}{2}.$$

The packet is real, so $\mathbb{E}P = 0$. Integration by parts gives

$$\mathbb{E}P^2 = \int |g'_\alpha(y)|^2 \,\mathrm{d}y = \frac{1}{4}\mathbb{E}(\alpha - U)^2 = \frac{\alpha}{4}.$$

The strict inequality follows from $\psi_1(\alpha) > 1/\alpha$. The asymptotic expansion $\psi_1(\alpha) = \alpha^{-1} + O(\alpha^{-2})$ gives the limit. □

α	$\operatorname{Var} Y$	$\operatorname{Var} P$	$\Delta Y\,\Delta P$	$\mathcal{I}_{\eta\eta}$
0.50	4.934802	0.125000	0.785398	19.739209
0.75	2.541880	0.187500	0.690364	10.167519
1.00	1.644934	0.250000	0.641275	6.579736
1.50	0.934802	0.375000	0.592073	3.739209
2.00	0.644934	0.500000	0.567862	2.579736
3.00	0.394934	0.750000	0.544243	1.579736
5.00	0.221323	1.250000	0.525979	0.885292
10.00	0.105166	2.500000	0.512753	0.420665

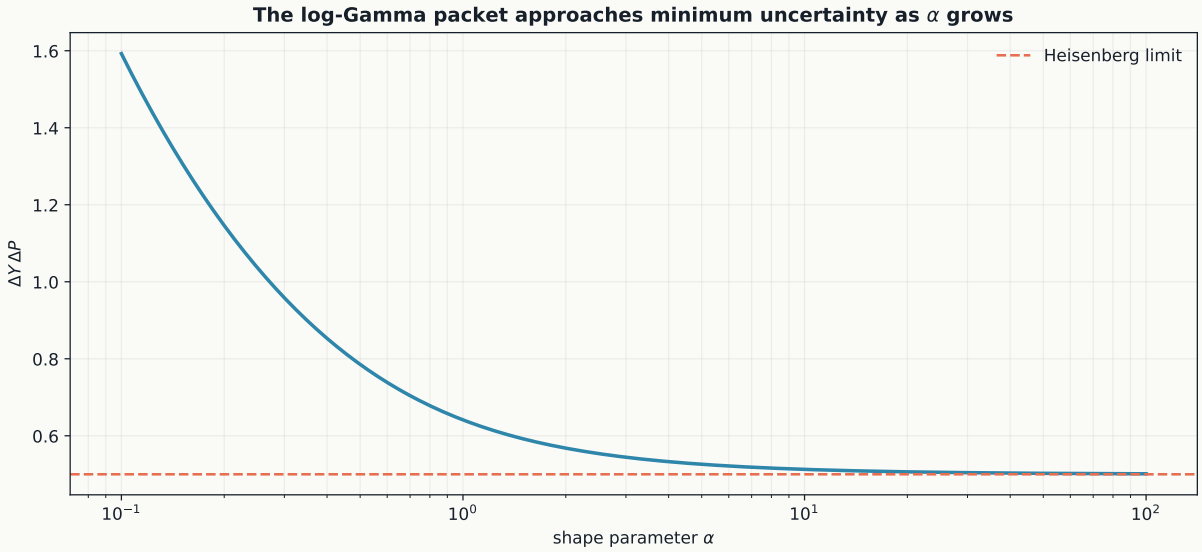


Figure 6.1: Exact uncertainty product. The packet is not a Gaussian at finite shape, but after centering and large-α scaling it approaches the minimum-uncertainty regime.

Define the centered Weyl orbit

$$g_{u,\eta}(y) = \mathrm{e}^{\mathrm{i}\eta(y-u/2)} g_\alpha(y-u).$$

Proposition 6.2 (Constant quantum geometric tensor)**.** *In coordinates (u,η), the Fubini–Study metric is constant and diagonal:*

$$\boxed{g_{uu} = \frac{\alpha}{4}, \qquad g_{\eta\eta} = \psi_1(\alpha), \qquad g_{u\eta} = 0.}$$

The Berry curvature is

$$\boxed{\mathcal{F}_{u\eta} = 1}$$

for the chosen Weyl convention. The quantum Fisher information matrix is

$$\boxed{\mathcal{I} = \mathrm{diag}(\alpha, 4\psi_1(\alpha))\,.}$$

Proof. Translation is generated by P and modulation by Y. The real part of the quantum geometric tensor is the covariance matrix of these generators. The mixed covariance vanishes because g_α is real and the centered symmetric derivative is odd under integration by parts. The imaginary part is one half of the canonical commutator $[Y,P] = \mathrm{i}I$. Multiplying the metric by four gives the quantum Fisher information. □

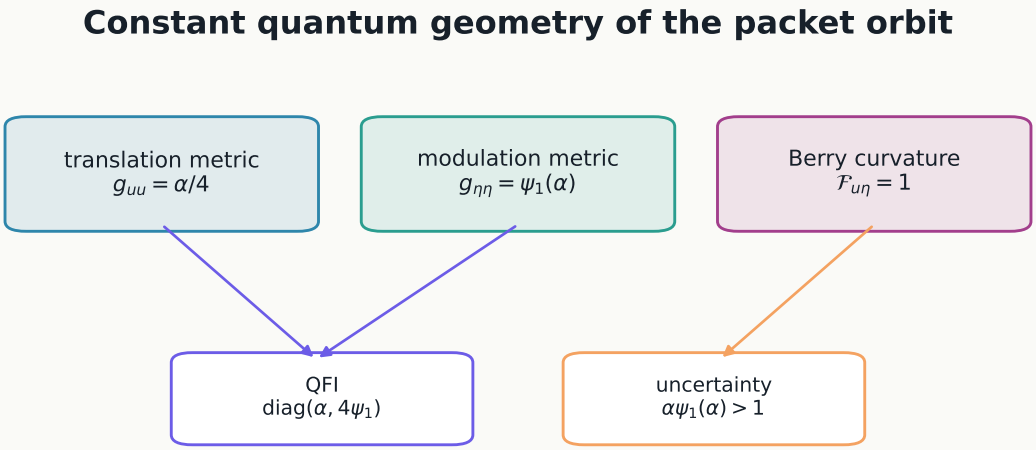


Figure 6.2: Quantum geometry of the Weyl orbit. Translation and modulation inherit exact metric coefficients from the packet variances. The canonical commutator supplies the constant symplectic curvature.

Part III

Orthogonal packet bases and one ladder algebra

Chapter 7

Laguerre packets on logarithmic position space

For $n \geq 0$, define

$$\boxed{\Lambda_{n,\alpha}(y) = \sqrt{\frac{n!}{\Gamma(n+\alpha)}}\, \mathrm{e}^{\alpha y/2 - \mathrm{e}^y/2} L_n^{(\alpha-1)}(\mathrm{e}^y).}$$

The ground packet is

$$\Lambda_{0,\alpha} = g_\alpha.$$

Theorem 7.1 (Orthonormality)**.**

$$\boxed{\int_{\mathbb{R}} \Lambda_{n,\alpha}(y)\Lambda_{m,\alpha}(y)\,\mathrm{d}y = \delta_{nm}.}$$

Proof. Set $x = \mathrm{e}^y$. Then $\mathrm{d}y = \mathrm{d}x/x$, and the inner product becomes the standard Laguerre integral

$$\frac{\sqrt{n!m!}}{\sqrt{\Gamma(n+\alpha)\Gamma(m+\alpha)}} \int_0^\infty x^{\alpha-1}\mathrm{e}^{-x} L_n^{(\alpha-1)}(x) L_m^{(\alpha-1)}(x)\,\mathrm{d}x.$$

The Laguerre orthogonality formula gives δ_{nm}. □

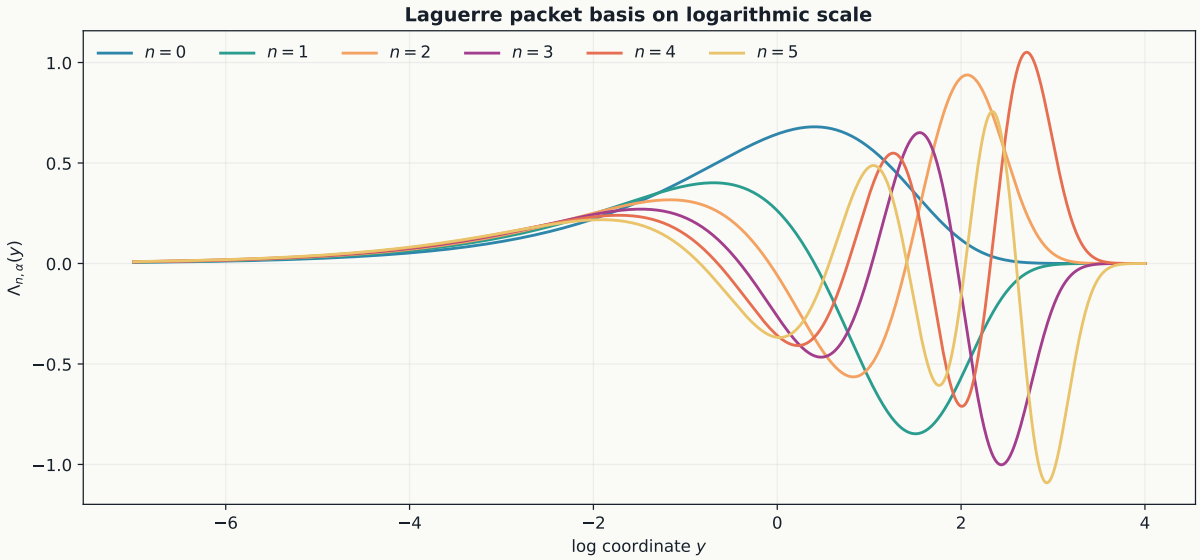


Figure 7.1: Laguerre packets in log coordinates. The packets form an exact orthonormal multioscillation family around the universal log-Gamma envelope.

Chapter 8

Meixner–Pollaczek packets in Mellin frequency

Let $p_{n,\alpha}$ denote the orthonormal polynomial sequence for the density

$$\rho_\alpha(\omega) = \frac{2^{\alpha-1}}{\pi\Gamma(\alpha)}\left|\Gamma\left(\frac{\alpha}{2}+\mathrm{i}\omega\right)\right|^2.$$

It is characterized by

$$p_{0,\alpha}=1, \qquad p_{1,\alpha}(\omega)=\frac{2\omega}{\sqrt{\alpha}},$$

and

$$\boxed{\omega p_{n,\alpha}(\omega)=\frac{1}{2}\sqrt{(n+1)(n+\alpha)}\,p_{n+1,\alpha}(\omega)+\frac{1}{2}\sqrt{n(n+\alpha-1)}\,p_{n-1,\alpha}(\omega).}$$

These are normalized Meixner–Pollaczek polynomials at angle $\pi/2$.

Theorem 8.1 (Laguerre–Meixner Fourier duality)**.** *For every $n \geq 0$,*

$$\boxed{\widehat{\Lambda}_{n,\alpha}(\omega)=\mathrm{i}^n\widehat{g}_\alpha(\omega)p_{n,\alpha}(\omega).}$$

Proof. Let

$$s=\frac{\alpha}{2}-\mathrm{i}\omega.$$

Using $x=\mathrm{e}^y$, the Fourier transform reduces to

$$\sqrt{\frac{n!}{2\pi\Gamma(n+\alpha)}}\int_0^\infty x^{s-1}\mathrm{e}^{-x/2}L_n^{(\alpha-1)}(x)\,\mathrm{d}x.$$

Introduce a generating variable z. The Laguerre generating function gives

$$\sum_{n\ge 0}\left[\int_0^\infty x^{s-1}\mathrm{e}^{-x/2}L_n^{(\alpha-1)}(x)\,\mathrm{d}x\right]z^n = 2^s\Gamma(s)(1-z)^{-\alpha/2-\mathrm{i}\omega}(1+z)^{-\alpha/2+\mathrm{i}\omega}.$$

After extracting the common ground-state factor, the remaining coefficients are the Meixner–Pollaczek generating coefficients with the phase i^n. The normalization is fixed by the Laguerre norms. This proves the formula. □

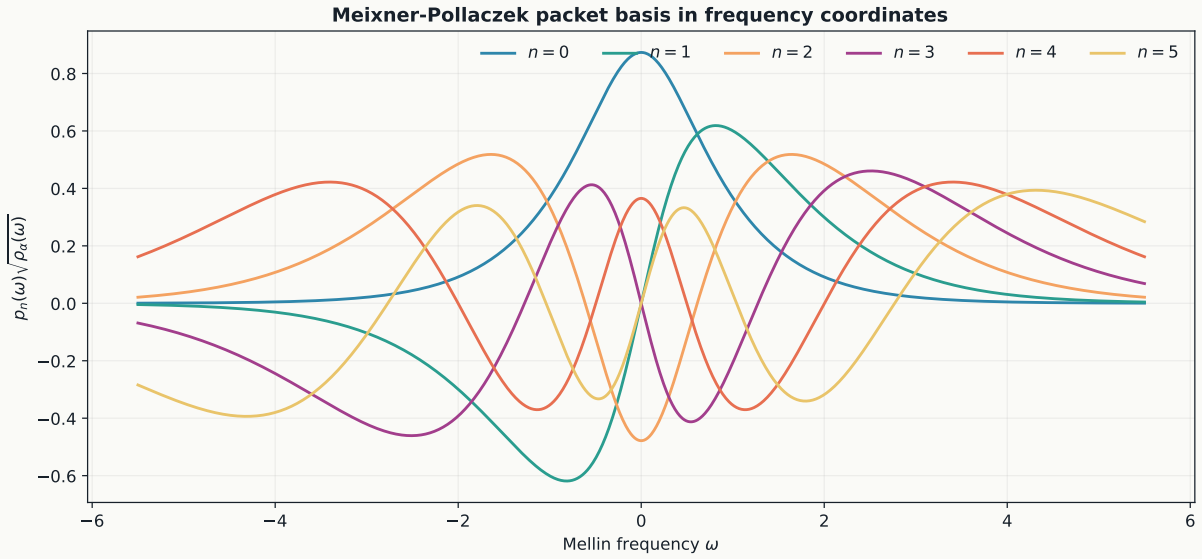


Figure 8.1: Meixner–Pollaczek packets. Multiplying each orthonormal polynomial by the square root of its Gamma density gives the frequency-side packet family.

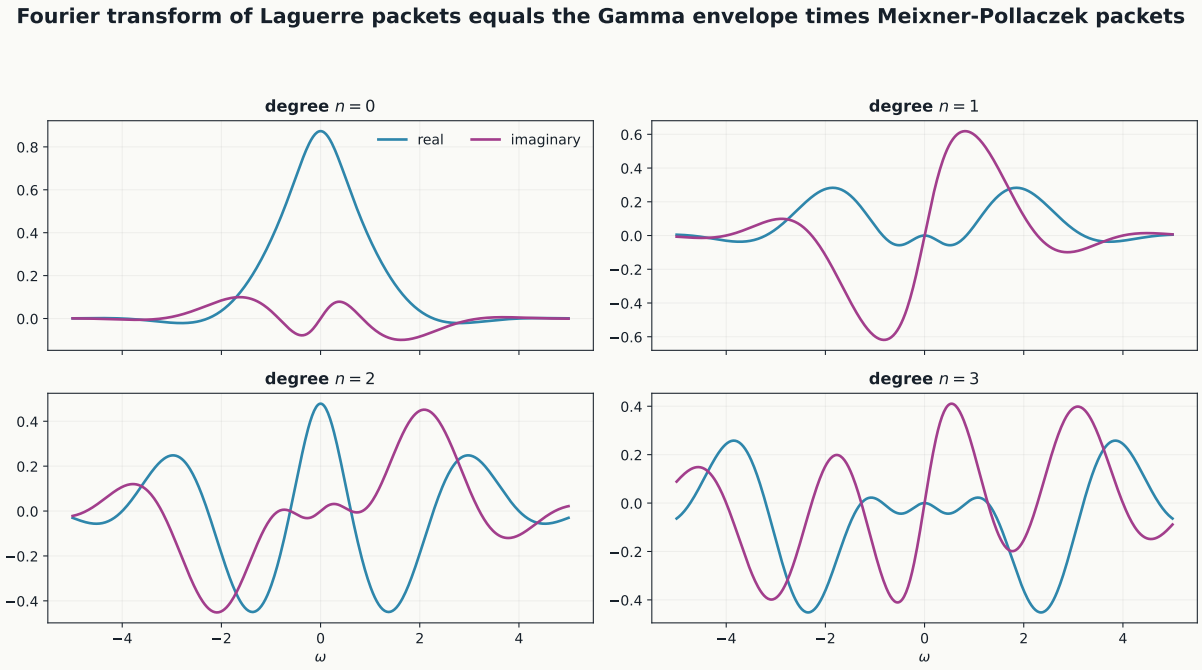


Figure 8.2: Exact Fourier duality. The real and imaginary parts alternate with degree through the phase i^n. The Gamma envelope is common to every packet.

Chapter 9

The common $\mathfrak{su}(1,1)$ ladder

Let $\{e_n\}_{n\geq 0}$ be an abstract orthonormal basis. Define

$$K_0 e_n = \left(n + \frac{\alpha}{2}\right) e_n,$$

$$K_+ e_n = \sqrt{(n+1)(n+\alpha)}\, e_{n+1},$$

and

$$K_- e_n = \sqrt{n(n+\alpha-1)}\, e_{n-1}.$$

Theorem 9.1 ($\mathfrak{su}(1,1)$ relations)**.** *On the finite linear span of the basis,*

$$\boxed{[K_0, K_+] = K_+, \qquad [K_0, K_-] = -K_-, \qquad [K_-, K_+] = 2K_0.}$$

The Casimir

$$\mathcal{Q} = K_0^2 - \frac{1}{2}(K_+ K_- + K_- K_+)$$

is

$$\boxed{\mathcal{Q} = \frac{\alpha}{2}\left(\frac{\alpha}{2} - 1\right) I.}$$

Proof. Apply each commutator to e_n. The coefficients reduce to elementary differences of the quadratic ladder weights. The same computation gives the constant Casimir value. □

In the Laguerre basis, multiplication by $x = \mathrm{e}^y$ is represented by

$$\boxed{J_L = 2K_0 - K_+ - K_-.}$$

In the Meixner–Pollaczek basis, multiplication by ω is represented by

$$\boxed{J_M = \frac{1}{2}(K_+ + K_-).}$$

Thus one representation carries two exact coordinate operators.

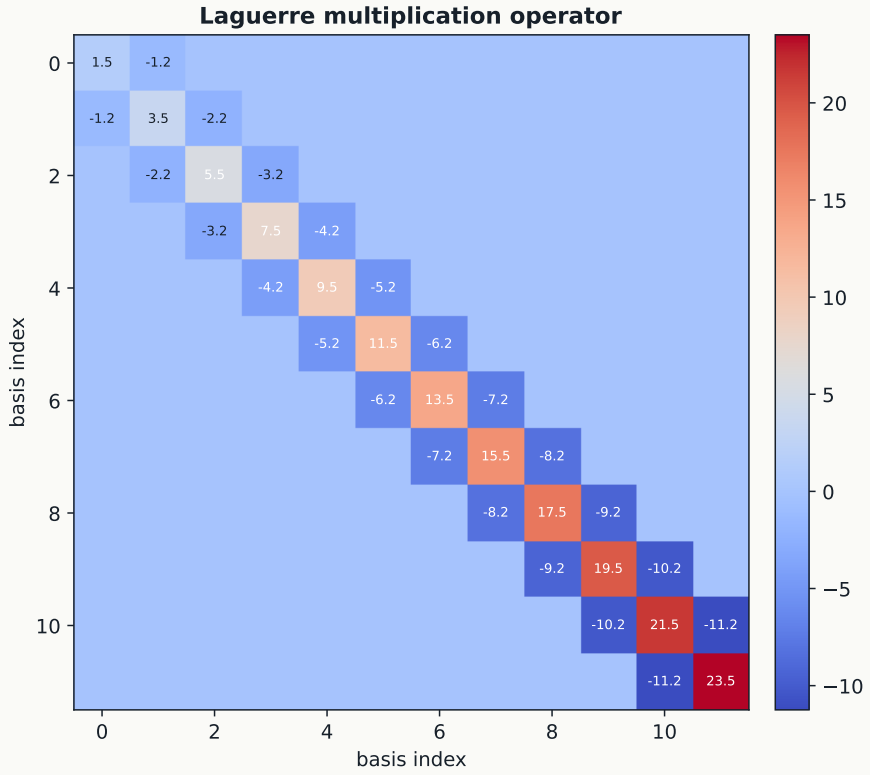


Figure 9.1: Laguerre Jacobi matrix. The position-side multiplication operator is tridiagonal in the logarithmic Laguerre packet basis.

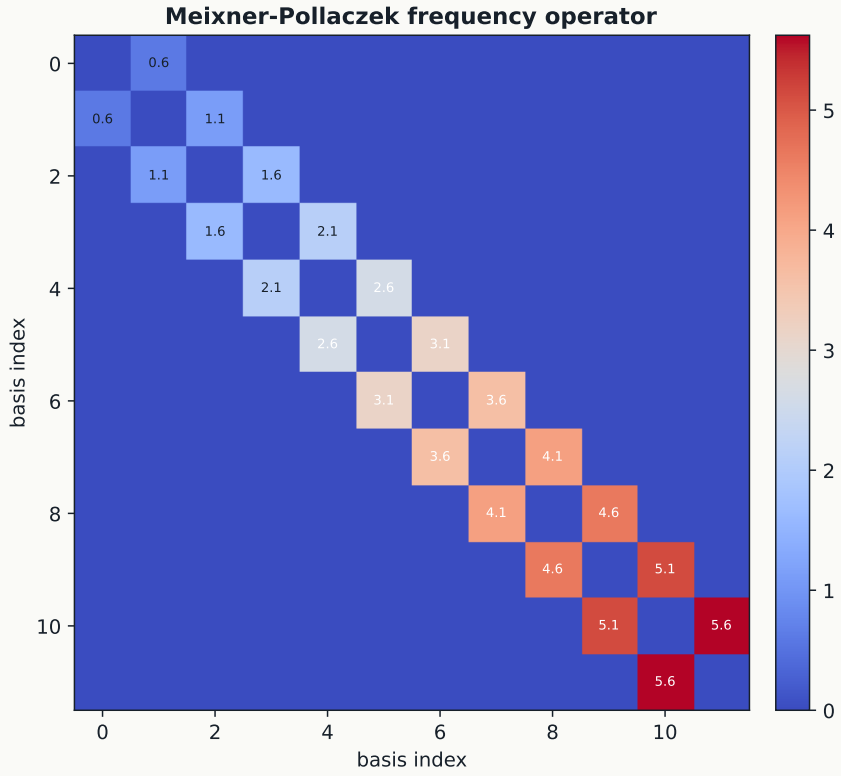


Figure 9.2: Meixner Jacobi matrix. The Mellin-frequency operator is another tridiagonal combination of the same ladder generators.

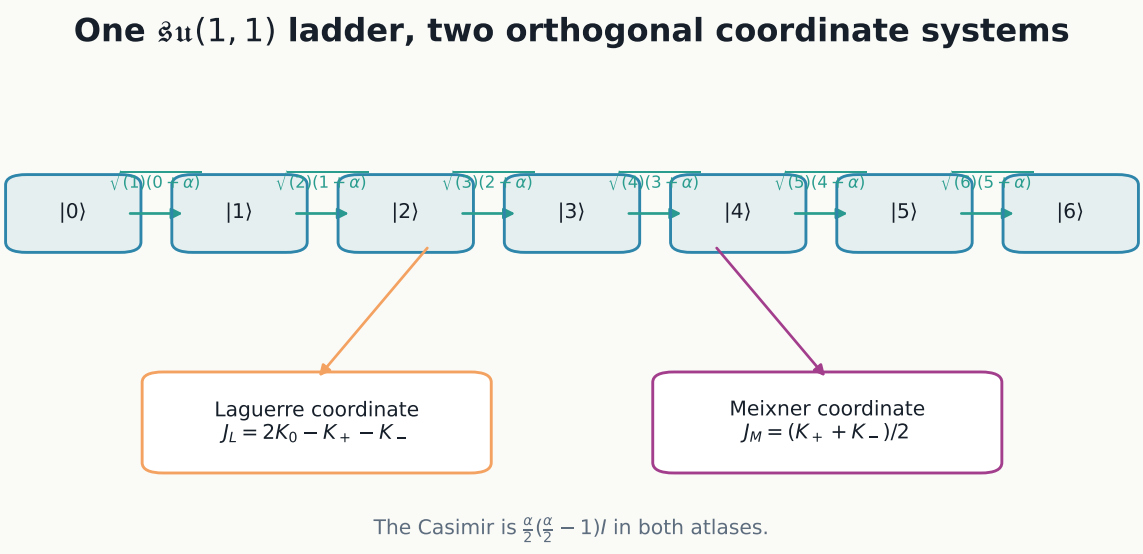


Figure 9.3: One ladder and two atlases. The Laguerre and Meixner operators are not unrelated recurrences. They are two projections of the same positive discrete-series ladder.

Part IV

Fractional depth, quantum realization, and the RH boundary

Chapter 10

Continuous packet depth

The integer-depth transfer from the earlier papers is

$$C_{a,\beta,\chi}(k,t) = \sum_{n\geq 1} \frac{\mu(n)\chi(n)}{n^{a+it}} \left(1 - \frac{1}{n^{\beta}}\right)^{k}.$$

For every real $r > 0$, define the continuous-depth extension

$$\boxed{C_{a,\beta,\chi}(r,t) = \sum_{n\geq 1} \frac{\mu(n)\chi(n)}{n^{a+it}} \left(1 - \frac{1}{n^{\beta}}\right)^{r}.}$$

Assume $a > 1$ so that all exchanges below are absolutely convergent.

Theorem 10.1 (Fractional binomial orbit)**.** *For $r > 0$,*

$$\boxed{C_{a,\beta,\chi}(r,t) = \sum_{j=0}^{\infty} (-1)^j \binom{r}{j} \frac{1}{L(a+\beta j+it,\chi)}.}$$

At integer $r = k$, the series terminates and recovers the finite-difference orbit.

Proof. For every $n \geq 2$,

$$\left(1 - n^{-\beta}\right)^r = \sum_{j \geq 0} (-1)^j \binom{r}{j} n^{-\beta j}.$$

The generalized binomial coefficients are absolutely summable for $r > 0$. The $n = 1$ packet vanishes on the left and the binomial sum at one is zero. Absolute convergence for $a > 1$ permits interchange of n and j. The inner Dirichlet series is the reciprocal L-function. □

Theorem 10.2 (Fractional-depth generator)**.**

$$\boxed{\frac{\partial}{\partial r} C_{a,\beta,\chi}(r,t) = -\sum_{m=1}^{\infty} \frac{1}{m} C_{a+m\beta,\beta,\chi}(r,t).}$$

Proof. Differentiate the factor $(1 - n^{-\beta})^r$ and use

$$\log(1 - n^{-\beta}) = -\sum_{m \geq 1} \frac{n^{-m\beta}}{m}.$$

Absolute convergence justifies the exchange of the sums and derivative. □

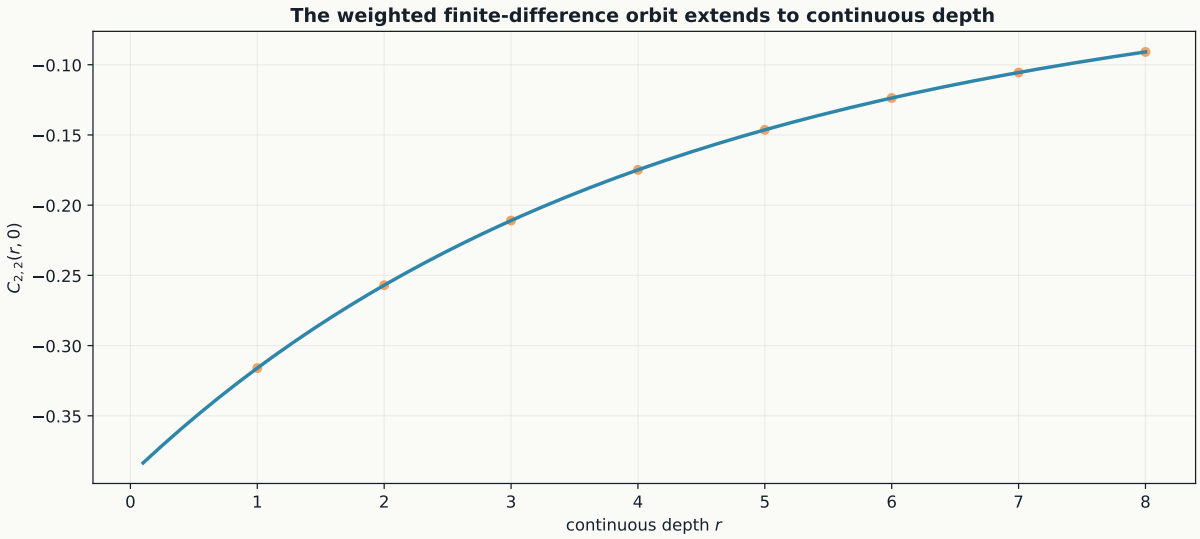


Figure 10.1: Continuous depth. The integer Bernoulli finite-difference orbit is embedded into a smooth depth evolution generated by an infinite parameter-shift operator.

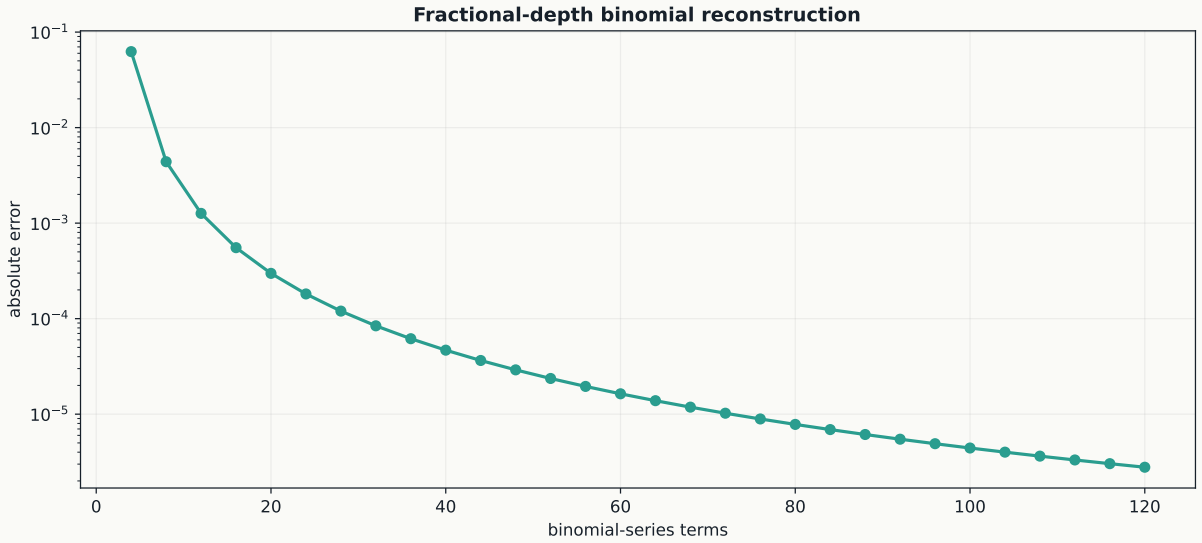


Figure 10.2: Fractional binomial reconstruction. The infinite reciprocal-zeta expansion converges to the direct Möbius packet calculation for the noninteger depth $r = 5/2$.

Chapter 11

Finite arithmetic convergence

The theorem at the beginning of the paper is asymptotic. The accompanying program evaluates the finite squarefree kernel directly and compares it with $\mathfrak{A}_\alpha$.

base depth	kernel error	real part	imaginary part
400	1.128878e-02	0.41019069	-0.01236909
1200	1.031104e-02	0.39292436	-0.02278309
4000	3.892217e-03	0.39764030	-0.01093128
12000	2.704832e-03	0.40156866	-0.01566558

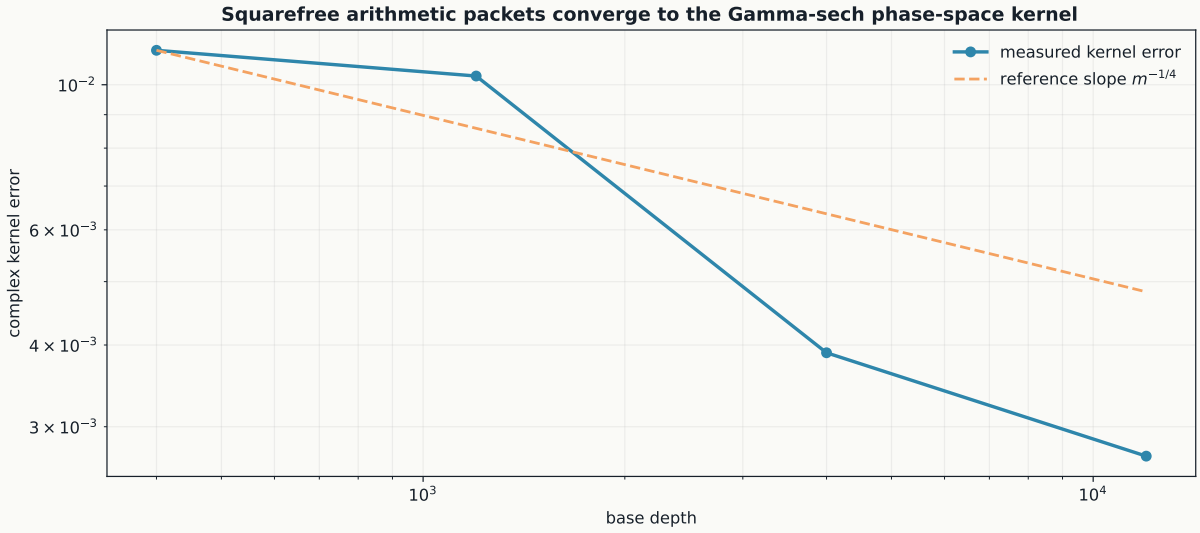


Figure 11.1: Arithmetic convergence. The finite squarefree packet moves toward the universal Gamma–sech kernel as the active scale grows. The displayed reference line is only a visual slope guide.

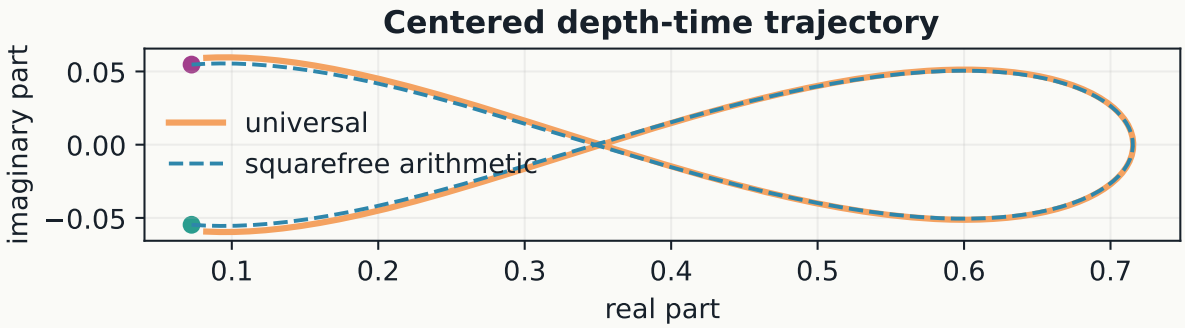


Figure 11.2: Complex trajectory comparison. After centering, the finite arithmetic correlation and the universal phase-space curve nearly coincide over the tested time interval.

Boundary statement 11.1 (What the arithmetic limit does not prove). The universal limit controls the positive squarefree kernel. It does not control the signed principal Möbius amplitude at time zero. The latter can be much smaller than the positive norm because of arithmetic cancellation. Establishing the required subpolynomial coherence remains the RH-equivalent problem.

Chapter 12

Quantum and signal realization

On the number basis, define

$$H\,|n\rangle = \log n\,|n\rangle\,, \qquad K_\beta\,|n\rangle = -\log(1-n^{-\beta})\,|n\rangle\,,$$

and

$$\mathcal{M}\,|n\rangle = \mu(n)\,|n\rangle\,.$$

The first operator is the logarithmic Hamiltonian. The second is the depth generator. The square $\mathcal{M}^2$ is the projector onto the squarefree number packets. All three operators are diagonal and commute.

For the amplitude vector

$$|\phi_a\rangle = \sum_{n\geq 2} n^{-a}\,|n\rangle\,,$$

the two-depth overlap

$$\boxed{\langle\phi_a|\,\mathrm{e}^{-\ell K_\beta}\mathcal{M}^2\mathrm{e}^{-\mathrm{i}tH}\mathrm{e}^{-kK_\beta}\,|\phi_a\rangle = \mathcal{K}_{a,\beta}(k,\ell,t)}$$

produces the depth-time kernel, with the excluded $n=1$ packet treated separately at zero depth. In the large-depth limit, this number-basis overlap becomes the Gabor ambiguity function of g_α.

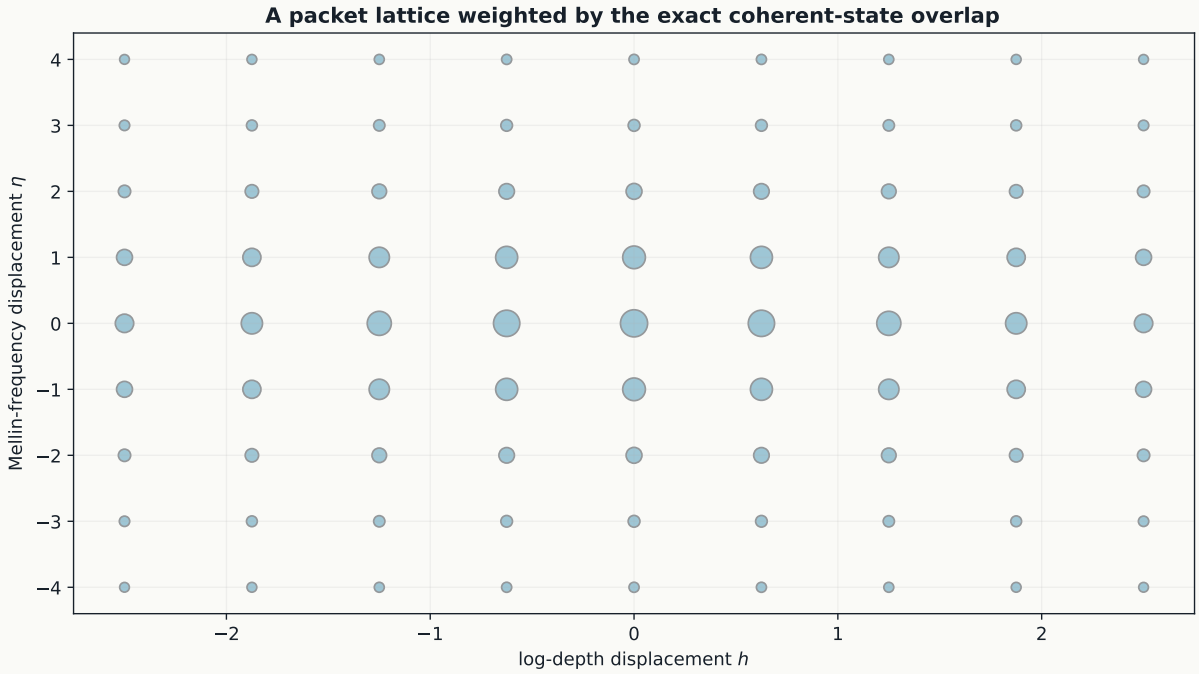


Figure 12.1: Packet lattice. Circle area is proportional to the exact overlap magnitude. The generated function supplies a natural nonuniform coherent-state geometry in log depth and Mellin frequency.

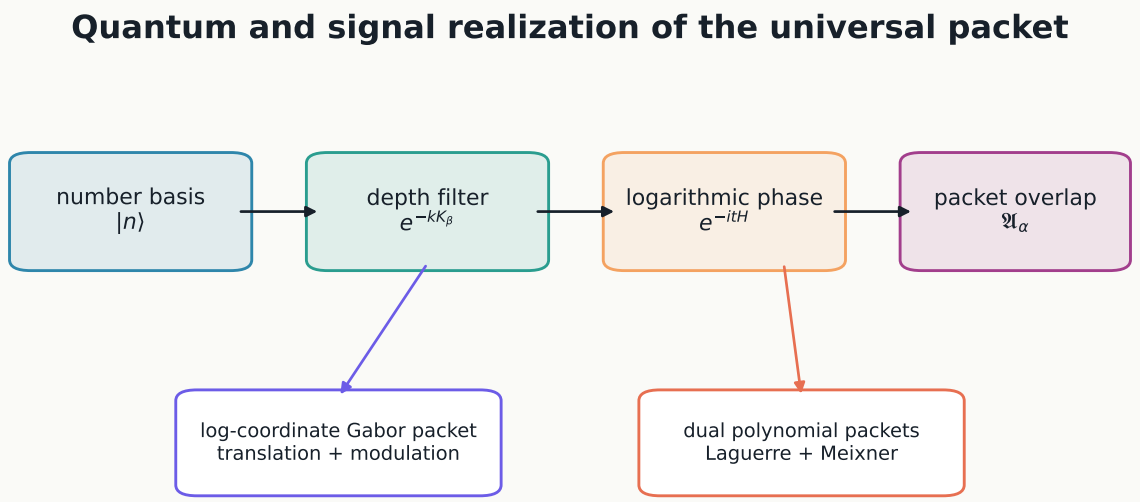


Figure 12.2: Quantum and signal path. The logarithmic Hamiltonian supplies the time phase. The depth filter supplies scale selection. Their centered overlap converges to the universal log-Gamma packet geometry.

The external quantum construction motivating this coordinate uses the same logarithmic spectrum and implements phases $n^{-\mathrm{i}t}$ through a logarithm oracle. The present packet adds the commuting scale generator K_β and studies a different positive overlap.

Chapter 13

The remaining RH boundary

For the principal parameters, define

$$c_k = \sum_{n\geq 1} \frac{\mu(n)}{n^2}\left(1-\frac{1}{n^2}\right)^k$$

and

$$E_k = \sum_{n\geq 1} \frac{\mu(n)^2}{n^4}\left(1-\frac{1}{n^2}\right)^{2k}.$$

The preceding papers prove

$$E_k \sim \frac{3}{2^{5/2}\pi^{3/2}} k^{-3/2}.$$

Define the packet coherence

$$\Gamma_k = \frac{|c_k|^2}{E_k}.$$

Using the classical sequential criterion,

$$\boxed{\mathrm{RH} \iff \Gamma_k = O_\varepsilon(k^\varepsilon) \quad \text{for every } \varepsilon > 0.}$$

The present paper solves the positive large-depth phase-space geometry exactly. It shows that the positive kernel has a zero-free Gabor limit, a complete orthogonal basis, a universal Wigner function, and a rigid ladder algebra. None of those positive facts forces the signed Möbius packet to have small coherence at the distinguished time $t = 0$.

Conjecture 13.1 (Principal Möbius coherence)**.** *For every $\varepsilon > 0$, there is a constant C_ε such that*

$$\boxed{|c_k|^2 \le C_\varepsilon k^\varepsilon E_k.}$$

This is stated as the remaining target, not as a theorem.

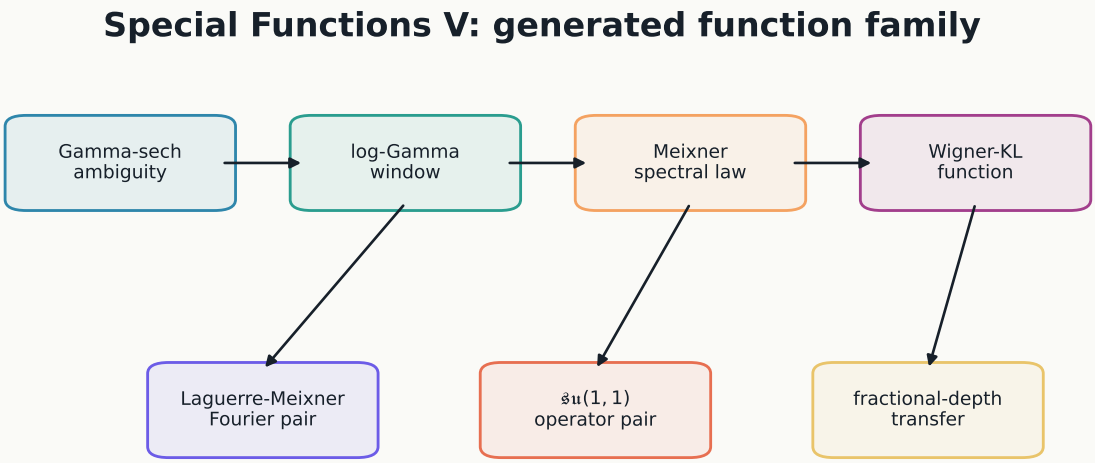


Figure 13.1: The generated family. The Gamma–sech ambiguity function, log-Gamma window, Meixner spectrum, Wigner–Bessel function, Laguerre–Meixner Fourier pair, ladder algebra, and fractional-depth transfer are exact coordinates of one universal packet system.

Part V

Verification, formula registry, and source

Chapter 14

Verification ledger

Certificate	Value
Built-in checks	19 of 19 passed
Ambiguity formula error	$6.704e-14$
Laguerre–Meixner Fourier error	$3.273e-12$
Meixner Gram error	$4.441e-15$
Laguerre Gram error	$3.132e-10$
$\mathfrak{su}(1,1)$ commutator error	$5.684e-14$
Poschl–Teller residual	$5.091e-08$
Fractional-depth error	$2.661e-10$

Table 14.1: Built-in verification ledger.

Check	Result	Detail
window normalization alpha=0.75	pass	integral=0.9999999999999
window normalization alpha=1.5	pass	integral=1
window normalization alpha=3.0	pass	integral=1
window normalization alpha=8.0	pass	integral=1
Gamma-sech ambiguity	pass	max_error=6.704e-14
Laguerre-Meixner Fourier duality	pass	max_error=3.273e-12
Meixner spectral density normalization	pass	integral=1
strict uncertainty alpha=0.75	pass	product=0.69036398656
strict uncertainty alpha=1.5	pass	product=0.592073327557
strict uncertainty alpha=3.0	pass	product=0.544243098382
strict uncertainty alpha=10.0	pass	product=0.512753195216
Meixner-Pollaczek orthogonality	pass	max_error=4.441e-15
Laguerre packet orthogonality	pass	max_error=3.132e-10
su(1,1) commutators	pass	max_error=5.684e-14
su(1,1) Casimir	pass	max_error=2.842e-14
Poschl-Teller equation	pass	max_error=5.091e-08
arithmetic joint-kernel convergence	pass	error_at_12000=2.705e-03
fractional-depth binomial orbit	pass	error=2.661e-10
real phase-space nonvanishing sample	pass	minimum=2.475e-06

The independent test suite rederives normalization, moments, ambiguity, the shape shift, spectral normalization, the characteristic function, Laguerre–Meixner duality, both orthogonalities, the ladder algebra, uncertainty, fractional depth, arithmetic Hermitian symmetry, and arithmetic convergence. It also verifies the figure inventory and the machine-readable result ledger.

Chapter 15

Formula registry

Object	Exact formula
Log-Gamma packet	$g_\alpha(y) = \Gamma(\alpha)^{-1/2} \exp(\alpha y/2 - \mathrm{e}^y/2)$
Ambiguity function	$\mathfrak{A}_\alpha(h,\eta) = \Gamma(\alpha+\mathrm{i}\eta)\Gamma(\alpha)^{-1} \operatorname{sech}^{\alpha+\mathrm{i}\eta}(h/2)$
Fourier transform	$\widehat{g}_\alpha(\omega) = 2^{\alpha/2-\mathrm{i}\omega}\Gamma(\alpha/2-\mathrm{i}\omega)/\sqrt{2\pi\Gamma(\alpha)}$
Spectral density	$\rho_\alpha(\omega) = 2^{\alpha-1}\lvert\Gamma(\alpha/2+\mathrm{i}\omega)\rvert^2/(\pi\Gamma(\alpha))$
Wigner function	$W_\alpha(y,\omega) = 2\mathrm{e}^{\alpha y}K_{2\mathrm{i}\omega}(\mathrm{e}^y)/(\pi\Gamma(\alpha))$
Uncertainty	$\Delta Y \Delta P = \frac{1}{2}\sqrt{\alpha\psi_1(\alpha)}$
Laguerre packet	$\Lambda_{n,\alpha}(y) = \sqrt{n!/\Gamma(n+\alpha)}\mathrm{e}^{\alpha y/2-\mathrm{e}^y/2}L_n^{(\alpha-1)}(\mathrm{e}^y)$
Fourier duality	$\widehat{\Lambda}_{n,\alpha} = \mathrm{i}^n \widehat{g}_\alpha p_{n,\alpha}$
Laguerre operator	$J_L = 2K_0 - K_+ - K_-$
Meixner operator	$J_M = (K_+ + K_-)/2$
Fractional depth	$C(r,t) = \sum_{j\geq 0}(-1)^j\binom{r}{j}L(a+\beta j+\mathrm{i}t,\chi)^{-1}$

Chapter 16

Relation to established function theory

The Meixner–Pollaczek polynomials, Laguerre polynomials, Gabor ambiguity function, Wigner distribution, Kontorovich–Lebedev kernel, and positive discrete series of $\mathrm{SU}(1,1)$ are classical objects. This paper does not claim their invention.

The structural contribution is the exact generated bridge:

$$\boxed{\begin{array}{c}\text{squarefree depth-time packet} \longrightarrow \text{log-Gamma Gabor window} \longrightarrow \text{Gamma–sech ambiguity} \\ \longrightarrow \text{Laguerre–Meixner duality} \longrightarrow \mathfrak{su}(1,1)\end{array}}$$

The arithmetic limit fixes the parameter α, the centering, the carrier phase, and the normalization. The same parameter then controls the ambiguity function, uncertainty, polynomial recurrence, Casimir, and fractional-depth geometry.

Bibliography

[1] S. Wei, Y. Zhai, Q. Lu, et al., "The Riemann Hypothesis manifested in dynamical quantum phase transitions," *Nature Communications*, article in press, 2026.

[2] L. Báez-Duarte, "A sequential Riesz-like criterion for the Riemann hypothesis," *International Journal of Mathematics and Mathematical Sciences*, 2005.

[3] R. Koekoek, P. A. Lesky, and R. F. Swarttouw, *Hypergeometric Orthogonal Polynomials and Their q-Analogues*, Springer, 2010.

[4] G. B. Folland, *Harmonic Analysis in Phase Space*, Princeton University Press, 1989.

[5] N. Ja. Vilenkin and A. U. Klimyk, *Representation of Lie Groups and Special Functions*, volume 1, Kluwer, 1991.

[6] NIST Digital Library of Mathematical Functions, chapters on Gamma functions, orthogonal polynomials, and integral transforms.

[7] M. I. Kontorovich and N. N. Lebedev, classical papers on the transform generated by modified Bessel functions of imaginary order.

Appendix A

Complete Python implementation

The following single program generates every figure, every numerical table, the built-in certificate ledger, and the machine-readable results used in this paper.

```
from __future__ import annotations

import csv
import json
import math
from dataclasses import dataclass
from functools import lru_cache
from pathlib import Path
from typing import Iterable

import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import FancyArrowPatch, FancyBboxPatch, Rectangle
import mpmath as mp
import numpy as np
from scipy.integrate import quad
from scipy.special import digamma, eval_genlaguerre, gamma, gammaln, loggamma, polygamma

mp.mp.dps = 80

ROOT = Path(__file__).resolve().parent
FIGURES = ROOT / "figures"
RESULTS = ROOT / "results"
TABLES = ROOT / "tables"
GENERATED = ROOT / "generated_tables"
for directory in (FIGURES, RESULTS, TABLES, GENERATED):
    directory.mkdir(parents=True, exist_ok=True)

INK = "#17202A"
BLUE = "#2E86AB"
TEAL = "#2A9D8F"
ORANGE = "#F4A261"
MAGENTA = "#A23E8C"
RED = "#E76F51"
YELLOW = "#E9C46A"
PURPLE = "#6C5CE7"
MUTED = "#5D6D7E"
PAPER = "#FAFAF7"
GRID = "#D5D8DC"
WHITE = "#FFFFFF"
COLORS = [BLUE, TEAL, ORANGE, MAGENTA, RED, YELLOW, PURPLE]

plt.rcParams.update(
    {
        "figure.facecolor": PAPER,
        "axes.facecolor": PAPER,
```

```
        "axes.edgecolor": INK,
        "axes.labelcolor": INK,
        "xtick.color": INK,
        "ytick.color": INK,
        "text.color": INK,
        "font.size": 10.5,
        "axes.titlesize": 12.2,
        "axes.titleweight": "bold",
        "legend.frameon": False,
        "savefig.bbox": "tight",
        "savefig.facecolor": PAPER,
    }
)


def save_figure(fig: plt.Figure, stem: str) -> None:
    fig.savefig(FIGURES / f"{stem}.pdf")
    fig.savefig(FIGURES / f"{stem}.png", dpi=220)
    plt.close(fig)


def diagram_axes(figsize: tuple[float, float] = (12.6, 6.1)) -> tuple[plt.Figure, plt.Axes]:
    fig, ax = plt.subplots(figsize=figsize)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.axis("off")
    return fig, ax


def box(
    ax: plt.Axes,
    xy: tuple[float, float],
    width: float,
    height: float,
    text: str,
    face: str = WHITE,
    edge: str = INK,
    fontsize: float = 10.8,
) -> None:
    patch = FancyBboxPatch(
        xy,
        width,
        height,
        boxstyle="round,pad=.012,rounding_size=.018",
        facecolor=face,
        edgecolor=edge,
        linewidth=1.7,
    )
    ax.add_patch(patch)
    ax.text(xy[0] + width / 2, xy[1] + height / 2, text, ha="center", va="center", fontsize=fontsize)


def arrow(
    ax: plt.Axes,
    start: tuple[float, float],
    end: tuple[float, float],
    color: str = INK,
    text: str | None = None,
    text_offset: tuple[float, float] = (0.0, 0.03),
) -> None:
    patch = FancyArrowPatch(start, end, arrowstyle="-|>", mutation_scale=13, linewidth=1.6, color=color)
    ax.add_patch(patch)
    if text:
        ax.text(
            (start[0] + end[0]) / 2 + text_offset[0],
            (start[1] + end[1]) / 2 + text_offset[1],
            text,
            ha="center",
            va="center",
            fontsize=9.8,
            color=color,
```

```
        )


# ---------------------------------------------------------------------------
# Log-Gamma packet window and generated special functions
# ---------------------------------------------------------------------------


def log_gamma_window(alpha: float, y: np.ndarray | float) -> np.ndarray:
    values = np.asarray(y, dtype=float)
    return np.exp(0.5 * alpha * values - 0.5 * np.exp(values) - 0.5 * gammaln(alpha))


def log_gamma_density(alpha: float, y: np.ndarray | float) -> np.ndarray:
    values = np.asarray(y, dtype=float)
    return np.exp(alpha * values - np.exp(values) - gammaln(alpha))


def log_gamma_fourier(alpha: float, omega: np.ndarray | float) -> np.ndarray:
    values = np.asarray(omega, dtype=float)
    z = alpha / 2.0 - 1j * values
    log_value = (alpha / 2.0 - 1j * values) * math.log(2.0) + loggamma(z)
    log_value -= 0.5 * (math.log(2.0 * math.pi) + gammaln(alpha))
    return np.exp(log_value)


def meixner_density(alpha: float, omega: np.ndarray | float) -> np.ndarray:
    values = np.asarray(omega, dtype=float)
    return np.abs(log_gamma_fourier(alpha, values)) ** 2


def gamma_sech_kernel(alpha: float, h: np.ndarray | float, eta: np.ndarray | float) -> np.ndarray:
    h_values, eta_values = np.broadcast_arrays(np.asarray(h, dtype=float), np.asarray(eta, dtype=float))
    z = alpha + 1j * eta_values
    log_sech = -np.log(np.cosh(h_values / 2.0))
    return np.exp(loggamma(z) - gammaln(alpha) + z * log_sech)


def centered_ambiguity_numeric(alpha: float, h: float, eta: float) -> complex:
    integrand = lambda y: (
        log_gamma_window(alpha, y + h / 2)
        * log_gamma_window(alpha, y - h / 2)
        * np.exp(1j * eta * y)
    )
    real = quad(lambda yy: float(np.real(integrand(yy))), -20, 8, epsabs=2e-12, limit=300)[0]
    imag = quad(lambda yy: float(np.imag(integrand(yy))), -20, 8, epsabs=2e-12, limit=300)[0]
    return complex(real, imag)


def wigner_log_gamma(alpha: float, y: np.ndarray, omega: np.ndarray) -> np.ndarray:
    yy, ww = np.meshgrid(np.asarray(y, dtype=float), np.asarray(omega, dtype=float), indexing="ij")
    out = np.empty_like(yy)
    prefactor = 2.0 / (math.pi * float(mp.gamma(alpha)))
    for i in range(yy.shape[0]):
        x = float(np.exp(yy[i, 0]))
        for j in range(yy.shape[1]):
            out[i, j] = prefactor * math.exp(alpha * yy[i, j]) * float(mp.re(mp.besselk(2j * ww[i, j], x)))
    return out


def uncertainty_data(alpha: float) -> dict[str, float]:
    var_y = float(polygamma(1, alpha))
    var_p = alpha / 4.0
    return {
        "mean_y": float(digamma(alpha)),
        "var_y": var_y,
        "var_p": var_p,
        "product": math.sqrt(var_y * var_p),
        "qfi_u": alpha,
        "qfi_eta": 4.0 * var_y,
        "berry_curvature": 1.0,
```

```
    }


def meixner_pollaczek_orthonormal(alpha: float, degree: int, omega: np.ndarray | float) -> np.ndarray:
    x = np.asarray(omega, dtype=float)
    if degree == 0:
        return np.ones_like(x)
    p_prev = np.ones_like(x)
    p_curr = 2.0 * x / math.sqrt(alpha)
    if degree == 1:
        return p_curr
    for n in range(1, degree):
        b_np1 = 0.5 * math.sqrt((n + 1) * (n + alpha))
        b_n = 0.5 * math.sqrt(n * (n + alpha - 1))
        p_next = (x * p_curr - b_n * p_prev) / b_np1
        p_prev, p_curr = p_curr, p_next
    return p_curr


def laguerre_packet(alpha: float, degree: int, y: np.ndarray | float) -> np.ndarray:
    values = np.asarray(y, dtype=float)
    normalization = math.exp(0.5 * (gammaln(degree + 1) - gammaln(degree + alpha)))
    return (
        normalization
        * np.exp(0.5 * alpha * values - 0.5 * np.exp(values))
        * eval_genlaguerre(degree, alpha - 1.0, np.exp(values))
    )


def laguerre_packet_fourier(alpha: float, degree: int, omega: np.ndarray | float) -> np.ndarray:
    values = np.asarray(omega, dtype=float)
    return (
        (1j ** degree)
        * log_gamma_fourier(alpha, values)
        * meixner_pollaczek_orthonormal(alpha, degree, values)
    )


def laguerre_packet_fourier_numeric(alpha: float, degree: int, omega: float) -> complex:
    integrand = lambda y: laguerre_packet(alpha, degree, y) * np.exp(-1j * omega * y) / math.sqrt(2 * math.pi)
    real = quad(lambda yy: float(np.real(integrand(yy))), -35, 10, epsabs=2e-12, limit=450)[0]
    imag = quad(lambda yy: float(np.imag(integrand(yy))), -35, 10, epsabs=2e-12, limit=450)[0]
    return complex(real, imag)


# ---------------------------------------------------------------------------
# Ladder operators and exact matrix systems
# ---------------------------------------------------------------------------


def su11_matrices(alpha: float, dimension: int) -> tuple[np.ndarray, np.ndarray, np.ndarray]:
    k0 = np.diag(np.arange(dimension, dtype=float) + alpha / 2.0)
    kp = np.zeros((dimension, dimension), dtype=float)
    for n in range(dimension - 1):
        kp[n + 1, n] = math.sqrt((n + 1) * (n + alpha))
    km = kp.T
    return k0, kp, km


def laguerre_jacobi(alpha: float, dimension: int) -> np.ndarray:
    k0, kp, km = su11_matrices(alpha, dimension)
    return 2.0 * k0 - kp - km


def meixner_jacobi(alpha: float, dimension: int) -> np.ndarray:
    _, kp, km = su11_matrices(alpha, dimension)
    return 0.5 * (kp + km)


# ---------------------------------------------------------------------------
# Arithmetic kernel and fractional-depth extension
```

```
# ----------------------------------------------------------------------------


@lru_cache(maxsize=12)
def mobius_sieve(nmax: int) -> np.ndarray:
    mu = np.zeros(nmax + 1, dtype=np.int8)
    mu[1] = 1
    primes: list[int] = []
    composite = np.zeros(nmax + 1, dtype=bool)
    for i in range(2, nmax + 1):
        if not composite[i]:
            primes.append(i)
            mu[i] = -1
        for p in primes:
            value = i * p
            if value > nmax:
                break
            composite[value] = True
            if i % p == 0:
                mu[value] = 0
                break
            mu[value] = -mu[i]
    return mu


@dataclass
class ArithmeticKernel:
    a: float
    beta: float
    n: np.ndarray
    weight: np.ndarray
    log_y: np.ndarray

    @property
    def alpha(self) -> float:
        return (2 * self.a - 1) / self.beta

    def kernel(self, k: float, ell: float, tau: float = 0.0) -> complex:
        depth = k + ell
        factor = np.exp(depth * self.log_y)
        phase = np.exp(-1j * tau * np.log(self.n))
        return complex(np.sum(self.weight * factor * phase))

    def normalized_centered(self, k: float, ell: float, tau: float) -> complex:
        eta = tau / self.beta
        numerator = self.kernel(k, ell, tau)
        denominator = math.sqrt(self.kernel(k, k, 0.0).real * self.kernel(ell, ell, 0.0).real)
        carrier = np.exp(1j * eta * math.log(2.0 * math.sqrt(k * ell)))
        return complex(carrier * numerator / denominator)


def build_arithmetic_kernel(a: float = 2.0, beta: float = 2.0, nmax: int = 600_000) -> ArithmeticKernel:
    mu = mobius_sieve(nmax)
    integers = np.arange(1, nmax + 1, dtype=np.int64)
    mask = mu[1:] != 0
    n = integers[mask].astype(float)
    y = 1.0 - n ** (-beta)
    log_y = np.full_like(y, -np.inf)
    positive = y > 0
    log_y[positive] = np.log(y[positive])
    weight = n ** (-2 * a)
    return ArithmeticKernel(a=a, beta=beta, n=n, weight=weight, log_y=log_y)


def universal_arithmetic_limit(alpha: float, beta: float, k: float, ell: float, tau: float) -> complex:
    h = math.log(k / ell)
    eta = tau / beta
    return complex(gamma_sech_kernel(alpha, h, eta))


def fractional_mobius_transfer(r: float, a: float = 2.0, beta: float = 2.0, nmax: int = 500_000) -> float:
```

```
    mu = mobius_sieve(nmax)
    integers = np.arange(2, nmax + 1, dtype=float)
    muv = mu[2:].astype(float)
    return float(np.sum(muv * integers ** (-a) * (1.0 - integers ** (-beta)) ** r))


def fractional_binomial_transfer(r: float, a: float = 2.0, beta: float = 2.0, terms: int = 120) -> mp.mpf:
    mp.mp.dps = 60
    total = mp.mpf("0")
    for j in range(terms):
        total += (-1) ** j * mp.binomial(r, j) / mp.zeta(a + beta * j)
    return total


# ------------------------------------------------------------------------
# Checks
# ------------------------------------------------------------------------


def perform_checks() -> tuple[list[dict[str, object]], dict[str, object]]:
    checks: list[dict[str, object]] = []

    def record(name: str, passed: bool, detail: str) -> None:
        checks.append({"check": name, "passed": bool(passed), "detail": detail})

    # Normalization and moment identities
    for alpha in (0.75, 1.5, 3.0, 8.0):
        integral = quad(lambda y: float(log_gamma_density(alpha, y)), -40, 12, epsabs=2e-12, limit=400)[0]
        record(f"window normalization alpha={alpha}", abs(integral - 1.0) < 2e-10, f"integral={integral:.13g}")

    # Exact ambiguity formula
    ambiguity_errors = []
    for h, eta in ((0.0, 0.7), (0.8, 0.5), (1.4, -1.1), (-0.6, 1.7)):
        numeric = centered_ambiguity_numeric(1.5, h, eta)
        exact = complex(gamma_sech_kernel(1.5, h, eta))
        ambiguity_errors.append(abs(numeric - exact))
    record("Gamma-sech ambiguity", max(ambiguity_errors) < 3e-9, f"max_error={max(ambiguity_errors):.3e}")

    # Fourier transform formula
    fourier_errors = []
    for degree in range(5):
        for omega in (-1.2, -0.3, 0.8):
            numeric = laguerre_packet_fourier_numeric(1.5, degree, omega)
            exact = complex(laguerre_packet_fourier(1.5, degree, omega))
            fourier_errors.append(abs(numeric - exact))
    record("Laguerre-Meixner Fourier duality", max(fourier_errors) < 2e-8, f"max_error={max(fourier_errors):.3e}")

    # Spectral density normalization and moments
    spectral_integral = quad(lambda w: float(meixner_density(1.5, w)), -20, 20, epsabs=1e-10, limit=300)[0]
    record("Meixner spectral density normalization", abs(spectral_integral - 1.0) < 2e-9, f"integral={
     spectral_integral:.13g}")

    # Uncertainty formulas
    for alpha in (0.75, 1.5, 3.0, 10.0):
        data = uncertainty_data(alpha)
        record(
            f"strict uncertainty alpha={alpha}",
            data["product"] > 0.5 and data["product"] < 0.8,
            f"product={data['product']:.12g}",
        )

    # Meixner-Pollaczek orthogonality
    nodes = np.linspace(-18, 18, 40001)
    rho = meixner_density(1.5, nodes)
    polys = np.vstack([meixner_pollaczek_orthonormal(1.5, n, nodes) for n in range(5)])
    gram = np.array([[np.trapezoid(rho * polys[i] * polys[j], nodes) for j in range(5)] for i in range(5)])
    mp_error = float(np.max(np.abs(gram - np.eye(5))))
    record("Meixner-Pollaczek orthogonality", mp_error < 2e-7, f"max_error={mp_error:.3e}")

    # Laguerre packet orthogonality
    y_grid = np.linspace(-15, 7, 60001)
```

```
    packets = np.vstack([laguerre_packet(1.5, n, y_grid) for n in range(5)])
    lag_gram = np.array([[np.trapezoid(packets[i] * packets[j], y_grid) for j in range(5)] for i in range(5)])
    lag_error = float(np.max(np.abs(lag_gram - np.eye(5))))
    record("Laguerre packet orthogonality", lag_error < 2e-7, f"max_error={lag_error:.3e}")

    # SU(1,1) commutators on the non-boundary block
    alpha = 1.5
    dim = 18
    k0, kp, km = su11_matrices(alpha, dim)
    interior = slice(0, dim - 2)
    c1 = (k0 @ kp - kp @ k0 - kp)[interior, interior]
    c2 = (k0 @ km - km @ k0 + km)[interior, interior]
    c3 = (km @ kp - kp @ km - 2 * k0)[interior, interior]
    comm_error = max(float(np.max(np.abs(c1))), float(np.max(np.abs(c2))), float(np.max(np.abs(c3))))
    record("su(1,1) commutators", comm_error < 2e-12, f"max_error={comm_error:.3e}")

    casimir = k0 @ k0 - 0.5 * (kp @ km + km @ kp)
    target = alpha / 2 * (alpha / 2 - 1)
    cas_error = float(np.max(np.abs(casimir[: dim - 2, : dim - 2] - target * np.eye(dim - 2))))
    record("su(1,1) Casimir", cas_error < 2e-12, f"max_error={cas_error:.3e}")

    # Poschl-Teller ODE
    h = np.linspace(-4, 4, 20001)
    eta = 0.7
    nu = 1.5 + 1j * eta
    f = gamma_sech_kernel(1.5, h, eta)
    dh = h[1] - h[0]
    fpp = (f[:-2] - 2 * f[1:-1] + f[2:]) / dh**2
    residual = -4 * fpp - nu * (nu + 1) / np.cosh(h[1:-1] / 2) ** 2 * f[1:-1] + nu**2 * f[1:-1]
    pt_error = float(np.max(np.abs(residual[10:-10])))
    record("Poschl-Teller equation", pt_error < 2e-6, f"max_error={pt_error:.3e}")

    # Arithmetic convergence to the universal kernel
    arithmetic = build_arithmetic_kernel(nmax=600_000)
    convergence_rows = []
    max_large_error = 0.0
    for scale in (400, 1200, 4000, 12000):
        k = float(scale)
        ell = float(4 * scale)
        tau = 2.4
        discrete = arithmetic.normalized_centered(k, ell, tau)
        universal = universal_arithmetic_limit(arithmetic.alpha, arithmetic.beta, k, ell, tau)
        error = abs(discrete - universal)
        max_large_error = error if scale == 12000 else max_large_error
        convergence_rows.append({"scale": scale, "error": error, "discrete_real": discrete.real, "discrete_imag":
 discrete.imag})
    record("arithmetic joint-kernel convergence", max_large_error < 7e-3, f"error_at_12000={max_large_error:.3e}")

    # Fractional-depth binomial formula
    frac_direct = fractional_mobius_transfer(2.5, nmax=5_000_000)
    frac_series = float(fractional_binomial_transfer(2.5, terms=5000))
    frac_error = abs(frac_direct - frac_series)
    record("fractional-depth binomial orbit", frac_error < 8e-9, f"error={frac_error:.3e}")

    # Zero-free real phase-space sample
    h_grid = np.linspace(-6, 6, 121)
    eta_grid = np.linspace(-8, 8, 121)
    hh, ee = np.meshgrid(h_grid, eta_grid, indexing="ij")
    values = gamma_sech_kernel(1.5, hh, ee)
    minimum = float(np.min(np.abs(values)))
    record("real phase-space nonvanishing sample", minimum > 0.0, f"minimum={minimum:.3e}")

    passed = sum(1 for check in checks if check["passed"])
    summary: dict[str, object] = {
        "built_in_checks": {"passed": passed, "total": len(checks)},
        "ambiguity_max_error": max(ambiguity_errors),
        "fourier_duality_max_error": max(fourier_errors),
        "meixner_gram_error": mp_error,
        "laguerre_gram_error": lag_error,
        "su11_commutator_error": comm_error,
        "su11_casimir_error": cas_error,
```

```
        "poschl_teller_error": pt_error,
        "fractional_depth_error": frac_error,
        "phase_space_minimum_sample": minimum,
        "arithmetic_convergence": convergence_rows,
        "uncertainty_alpha_1_5": uncertainty_data(1.5),
    }
    return checks, summary


# ---------------------------------------------------------------------------
# Figure generation
# ---------------------------------------------------------------------------


def generate_figures(summary: dict[str, object]) -> None:
    # 01 pipeline
    fig, ax = diagram_axes()
    ax.text(0.5, 0.94, "From arithmetic depth packets to a universal phase-space function", ha="center", fontsize=19,
      weight="bold")
    items = [
        (0.03, "squarefree\ndepth-time kernel", BLUE),
        (0.22, "large-depth\nGamma scaling", TEAL),
        (0.41, "log-Gamma\npacket window", ORANGE),
        (0.60, "Gamma-sech\nambiguity kernel", MAGENTA),
        (0.79, "Laguerre and\nMeixner duality", PURPLE),
    ]
    for x0, text, color in items:
        box(ax, (x0, 0.56), 0.16, 0.18, text, face=color + "20", edge=color, fontsize=12)
    for left, right in zip(items[:-1], items[1:]):
        arrow(ax, (left[0] + 0.16, 0.65), (right[0], 0.65), color=INK)
    box(ax, (0.18, 0.18), 0.25, 0.15, "exact ambiguity\nPoschl-Teller equation", face=WHITE, edge=TEAL, fontsize
     =11.5)
    box(ax, (0.57, 0.18), 0.25, 0.15, "SU(1,1) ladder\nand quantum geometry", face=WHITE, edge=PURPLE, fontsize
     =11.5)
    arrow(ax, (0.49, 0.56), (0.305, 0.33), color=TEAL)
    arrow(ax, (0.68, 0.56), (0.695, 0.33), color=PURPLE)
    ax.text(0.5, 0.07, "The limiting packet is a concrete Gabor window rather than an abstract asymptotic profile.",
      ha="center", color=MUTED, fontsize=11.5)
    save_figure(fig, "01_theory_pipeline")

    # 02 window family
    y = np.linspace(-7, 4, 1800)
    fig, ax = plt.subplots(figsize=(12.6, 5.3))
    for alpha, color in zip((0.75, 1.5, 3.0, 8.0), COLORS):
        ax.plot(y, log_gamma_window(alpha, y), lw=2.1, color=color, label=rf"$\alpha={alpha}$")
    ax.set_xlabel("log coordinate $y$")
    ax.set_ylabel(r"$g_\alpha(y)$")
    ax.set_title("Normalized log-Gamma packet windows")
    ax.grid(alpha=0.18)
    ax.legend(ncol=4)
    save_figure(fig, "02_log_gamma_windows")

    # 03 position density
    fig, ax = plt.subplots(figsize=(12.6, 5.3))
    for alpha, color in zip((0.75, 1.5, 3.0, 8.0), COLORS):
        ax.plot(y, log_gamma_density(alpha, y), lw=2.1, color=color, label=rf"$\alpha={alpha}$")
    ax.set_xlabel(r"$y=\log u$")
    ax.set_ylabel("probability density")
    ax.set_title("The packet position law is log-Gamma")
    ax.grid(alpha=0.18)
    ax.legend(ncol=4)
    save_figure(fig, "03_log_gamma_density")

    # 04 spectral density
    omega = np.linspace(-8, 8, 2000)
    fig, ax = plt.subplots(figsize=(12.6, 5.3))
    for alpha, color in zip((0.75, 1.5, 3.0, 8.0), COLORS):
        ax.plot(omega, meixner_density(alpha, omega), lw=2.1, color=color, label=rf"$\alpha={alpha}$")
    ax.set_xlabel(r"Mellin frequency $\omega$")
    ax.set_ylabel(r"$|\widehat g_\alpha(\omega)|^2$")
    ax.set_title("The dual packet law is the symmetric Meixner density")
```

```
    ax.grid(alpha=0.18)
    ax.legend(ncol=4)
    save_figure(fig, "04_meixner_density")

    # 05 autocorrelation
    h = np.linspace(-7, 7, 1600)
    fig, ax = plt.subplots(figsize=(12.6, 5.2))
    for alpha, color in zip((0.75, 1.5, 3.0, 8.0), COLORS):
        ax.plot(h, 1 / np.cosh(h / 2) ** alpha, lw=2.1, color=color, label=rf"$\alpha={alpha}$")
    ax.set_xlabel("log-depth difference $h$")
    ax.set_ylabel("normalized overlap")
    ax.set_title(r"Exact depth overlap $\operatorname{sech}^{\alpha}(h/2)$")
    ax.grid(alpha=0.18)
    ax.legend(ncol=4)
    save_figure(fig, "05_depth_autocorrelation")

    # 06 ambiguity magnitude
    h_grid = np.linspace(-5, 5, 181)
    eta_grid = np.linspace(-7, 7, 181)
    hh, ee = np.meshgrid(h_grid, eta_grid, indexing="xy")
    amb = gamma_sech_kernel(1.5, hh, ee)
    fig, ax = plt.subplots(figsize=(11.7, 6.2))
    image = ax.imshow(np.abs(amb), extent=[h_grid.min(), h_grid.max(), eta_grid.min(), eta_grid.max()], origin="
     lower", aspect="auto", cmap="viridis")
    ax.set_xlabel("log-depth shift $h$")
    ax.set_ylabel(r"Mellin-frequency shift $\eta$")
    ax.set_title(r"Magnitude of the Gamma-sech ambiguity function, $\alpha=3/2$")
    fig.colorbar(image, ax=ax, fraction=0.035, pad=0.025, label="magnitude")
    save_figure(fig, "06_ambiguity_magnitude")

    # 07 ambiguity phase
    fig, ax = plt.subplots(figsize=(11.7, 6.2))
    image = ax.imshow(np.angle(amb), extent=[h_grid.min(), h_grid.max(), eta_grid.min(), eta_grid.max()], origin="
     lower", aspect="auto", cmap="twilight", vmin=-np.pi, vmax=np.pi)
    ax.set_xlabel("log-depth shift $h$")
    ax.set_ylabel(r"Mellin-frequency shift $\eta$")
    ax.set_title("Phase of the zero-free Gamma-sech ambiguity function")
    fig.colorbar(image, ax=ax, fraction=0.035, pad=0.025, label="phase")
    save_figure(fig, "07_ambiguity_phase")

    # 08 slices
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8))
    for eta, color in zip((0.0, 0.75, 1.5, 2.5), COLORS):
        axes[0].plot(h, np.abs(gamma_sech_kernel(1.5, h, eta)), lw=2, color=color, label=rf"$\eta={eta}$")
    axes[0].set_xlabel("$h$")
    axes[0].set_ylabel("magnitude")
    axes[0].set_title("Depth slices")
    axes[0].grid(alpha=0.18)
    axes[0].legend()
    eta_line = np.linspace(-8, 8, 1000)
    for hh0, color in zip((0.0, 0.8, 1.6, 2.4), COLORS):
        axes[1].plot(eta_line, np.abs(gamma_sech_kernel(1.5, hh0, eta_line)), lw=2, color=color, label=rf"$h={hh0}$"
     )
    axes[1].set_xlabel(r"$\eta$")
    axes[1].set_ylabel("magnitude")
    axes[1].set_title("Frequency slices")
    axes[1].grid(alpha=0.18)
    axes[1].legend()
    fig.suptitle("The ambiguity magnitude factorizes into depth and frequency envelopes", fontsize=16, weight="bold"
     )
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "08_ambiguity_slices")

    # 09 Poschl-Teller residual and potential
    h2 = np.linspace(-7, 7, 1600)
    eta0 = 0.8
    nu = 1.5 + 1j * eta0
    potential = -nu * (nu + 1) / np.cosh(h2 / 2) ** 2
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8))
    axes[0].plot(h2, potential.real, color=BLUE, lw=2, label="real")
    axes[0].plot(h2, potential.imag, color=MAGENTA, lw=2, label="imaginary")
```

```
    axes[0].set_title("Complex Poschl-Teller potential")
    axes[0].set_xlabel("$h$")
    axes[0].grid(alpha=0.18)
    axes[0].legend()
    f = gamma_sech_kernel(1.5, h2, eta0)
    axes[1].plot(h2, f.real, color=TEAL, lw=2, label="real")
    axes[1].plot(h2, f.imag, color=ORANGE, lw=2, label="imaginary")
    axes[1].set_title("Exact Gamma-sech eigenfunction")
    axes[1].set_xlabel("$h$")
    axes[1].grid(alpha=0.18)
    axes[1].legend()
    fig.suptitle("The generated ambiguity function solves a shape-invariant Schrodinger equation", fontsize=15.5,
     weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.90])
    save_figure(fig, "09_poschl_teller")

    # 10 Wigner function
    y_w = np.linspace(-4.5, 3.0, 85)
    o_w = np.linspace(-5.5, 5.5, 95)
    wigner = wigner_log_gamma(1.5, y_w, o_w)
    fig, ax = plt.subplots(figsize=(11.7, 6.2))
    vmax = float(np.percentile(np.abs(wigner), 99.0))
    image = ax.imshow(wigner.T, extent=[y_w.min(), y_w.max(), o_w.min(), o_w.max()], origin="lower", aspect="auto",
     cmap="coolwarm", vmin=-vmax, vmax=vmax)
    ax.set_xlabel("log coordinate $y$")
    ax.set_ylabel(r"Mellin frequency $\omega$")
    ax.set_title(r"Wigner-Kontorovich-Lebedev packet, $\alpha=3/2$")
    fig.colorbar(image, ax=ax, fraction=0.035, pad=0.025, label="Wigner density")
    save_figure(fig, "10_wigner_kl")

    # 11 Wigner marginals
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8))
    axes[0].plot(y_w, np.trapezoid(wigner, o_w, axis=1), color=BLUE, lw=2.2, label="Wigner marginal")
    axes[0].plot(y_w, log_gamma_density(1.5, y_w), color=ORANGE, lw=1.5, ls="--", label="exact position law")
    axes[0].set_xlabel("$y$")
    axes[0].set_title("Position marginal")
    axes[0].grid(alpha=0.18)
    axes[0].legend()
    axes[1].plot(o_w, np.trapezoid(wigner, y_w, axis=0), color=TEAL, lw=2.2, label="Wigner marginal")
    axes[1].plot(o_w, meixner_density(1.5, o_w), color=MAGENTA, lw=1.5, ls="--", label="exact spectral law")
    axes[1].set_xlabel(r"$\omega$")
    axes[1].set_title("Frequency marginal")
    axes[1].grid(alpha=0.18)
    axes[1].legend()
    fig.suptitle("The Wigner packet recovers both exact probability marginals", fontsize=15.5, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.90])
    save_figure(fig, "11_wigner_marginals")

    # 12 uncertainty product
    alphas = np.logspace(-1, 2, 500)
    product = np.array([uncertainty_data(float(a))["product"] for a in alphas])
    fig, ax = plt.subplots(figsize=(12.6, 5.2))
    ax.semilogx(alphas, product, color=BLUE, lw=2.2)
    ax.axhline(0.5, color=RED, ls="--", lw=1.6, label="Heisenberg limit")
    ax.set_xlabel(r"shape parameter $\alpha$")
    ax.set_ylabel(r"$\Delta Y\,\Delta P$")
    ax.set_title(r"The log-Gamma packet approaches minimum uncertainty as $\alpha$ grows")
    ax.grid(alpha=0.18, which="both")
    ax.legend()
    save_figure(fig, "12_uncertainty_product")

    # 13 QFI geometry
    fig, ax = diagram_axes((12.6, 6.0))
    ax.text(0.5, 0.92, "Constant quantum geometry of the packet orbit", ha="center", fontsize=19, weight="bold")
    box(ax, (0.08, 0.58), 0.24, 0.18, "translation metric\n" + r"$g_{uu}=\alpha/4$", face=BLUE + "20", edge=BLUE,
     fontsize=13)
    box(ax, (0.38, 0.58), 0.24, 0.18, "modulation metric\n" + r"$g_{\eta\eta}=\psi_1(\alpha)$", face=TEAL + "20",
     edge=TEAL, fontsize=13)
    box(ax, (0.68, 0.58), 0.24, 0.18, "Berry curvature\n" + r"$\mathcal{F}_{u\eta}=1$", face=MAGENTA + "20", edge=
     MAGENTA, fontsize=13)
    box(ax, (0.22, 0.20), 0.23, 0.14, "QFI\n" + r"$\operatorname{diag}(\alpha,4\psi_1)$", face=WHITE, edge=PURPLE,
```

```
    fontsize=12)
    box(ax, (0.55, 0.20), 0.23, 0.14, "uncertainty\n" + r"$\alpha\psi_1(\alpha)>1$", face=WHITE, edge=ORANGE,
    fontsize=12)
    arrow(ax, (0.20, 0.58), (0.33, 0.34), color=PURPLE)
    arrow(ax, (0.50, 0.58), (0.33, 0.34), color=PURPLE)
    arrow(ax, (0.80, 0.58), (0.67, 0.34), color=ORANGE)
    ax.text(0.5, 0.08, "The metric is flat in packet coordinates while the symplectic curvature remains nonzero.",
    ha="center", fontsize=11.5, color=MUTED)
    save_figure(fig, "13_quantum_geometry")

    # 14 Laguerre packets
    y_l = np.linspace(-7, 4, 1800)
    fig, ax = plt.subplots(figsize=(12.6, 5.4))
    for ndeg, color in zip(range(6), COLORS):
        ax.plot(y_l, laguerre_packet(1.5, ndeg, y_l), lw=1.8, color=color, label=rf"$n={ndeg}$")
    ax.set_xlabel("log coordinate $y$")
    ax.set_ylabel(r"$\Lambda_{n,\alpha}(y)$")
    ax.set_title("Laguerre packet basis on logarithmic scale")
    ax.grid(alpha=0.18)
    ax.legend(ncol=6)
    save_figure(fig, "14_laguerre_packets")

    # 15 Meixner polynomials
    omega_m = np.linspace(-5.5, 5.5, 1600)
    rho_m = meixner_density(1.5, omega_m)
    fig, ax = plt.subplots(figsize=(12.6, 5.4))
    scale = np.sqrt(rho_m)
    for ndeg, color in zip(range(6), COLORS):
        ax.plot(omega_m, meixner_pollaczek_orthonormal(1.5, ndeg, omega_m) * scale, lw=1.8, color=color, label=rf"$n
    ={ndeg}$")
    ax.set_xlabel(r"Mellin frequency $\omega$")
    ax.set_ylabel(r"$p_n(\omega)\sqrt{\rho_\alpha(\omega)}$")
    ax.set_title("Meixner-Pollaczek packet basis in frequency coordinates")
    ax.grid(alpha=0.18)
    ax.legend(ncol=6)
    save_figure(fig, "15_meixner_packets")

    # 16 Fourier duality
    fig, axes = plt.subplots(2, 2, figsize=(12.8, 7.2), sharex=True)
    omega_d = np.linspace(-5, 5, 500)
    for ax, ndeg in zip(axes.ravel(), range(4)):
        exact = laguerre_packet_fourier(1.5, ndeg, omega_d)
        ax.plot(omega_d, exact.real, color=BLUE, lw=2, label="real")
        ax.plot(omega_d, exact.imag, color=MAGENTA, lw=2, label="imaginary")
        ax.set_title(rf"degree $n={ndeg}$")
        ax.grid(alpha=0.16)
    axes[0, 0].legend(ncol=2)
    for ax in axes[-1]:
        ax.set_xlabel(r"$\omega$")
    fig.suptitle("Fourier transform of Laguerre packets equals the Gamma envelope times Meixner-Pollaczek packets",
    fontsize=15.5, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.92])
    save_figure(fig, "16_fourier_duality")

    # 17 Laguerre Jacobi matrix
    jl = laguerre_jacobi(1.5, 12)
    fig, ax = plt.subplots(figsize=(8.4, 6.5))
    image = ax.imshow(jl, cmap="coolwarm", aspect="equal")
    ax.set_title("Laguerre multiplication operator")
    ax.set_xlabel("basis index")
    ax.set_ylabel("basis index")
    for i in range(12):
        for j in range(12):
            if abs(jl[i, j]) > 1e-12:
                ax.text(j, i, f"{jl[i,j]:.1f}", ha="center", va="center", fontsize=7, color=WHITE if abs(jl[i,j]) >
    4 else INK)
    fig.colorbar(image, ax=ax, fraction=0.045, pad=0.03)
    save_figure(fig, "17_laguerre_jacobi")

    # 18 Meixner Jacobi matrix
    jm = meixner_jacobi(1.5, 12)
```

```
    fig, ax = plt.subplots(figsize=(8.4, 6.5))
    image = ax.imshow(jm, cmap="coolwarm", aspect="equal")
    ax.set_title("Meixner-Pollaczek frequency operator")
    ax.set_xlabel("basis index")
    ax.set_ylabel("basis index")
    for i in range(12):
        for j in range(12):
            if abs(jm[i, j]) > 1e-12:
                ax.text(j, i, f"{jm[i,j]:.1f}", ha="center", va="center", fontsize=7, color=WHITE if abs(jm[i,j]) > 2.5 else INK)
    fig.colorbar(image, ax=ax, fraction=0.045, pad=0.03)
    save_figure(fig, "18_meixner_jacobi")

    # 19 SU(1,1) ladder
    fig, ax = diagram_axes((12.6, 6.2))
    ax.text(0.5, 0.93, r"One $\mathfrak{su}(1,1)$ ladder, two orthogonal coordinate systems", ha="center", fontsize=19, weight="bold")
    xs = np.linspace(0.08, 0.92, 7)
    for ndeg, xpos in enumerate(xs):
        box(ax, (xpos - 0.045, 0.58), 0.09, 0.11, rf"$|{ndeg}\rangle$", face=BLUE + "18", edge=BLUE, fontsize=12)
        if ndeg < len(xs) - 1:
            arrow(ax, (xpos + 0.045, 0.635), (xs[ndeg + 1] - 0.045, 0.635), color=TEAL, text=rf"$\sqrt{{({ndeg+1})({ndeg}+\alpha)}}$", text_offset=(0, 0.055))
    box(ax, (0.15, 0.20), 0.27, 0.15, "Laguerre coordinate\n" + r"$J_L=2K_0-K_+-K_-$", face=WHITE, edge=ORANGE, fontsize=12)
    box(ax, (0.58, 0.20), 0.27, 0.15, "Meixner coordinate\n" + r"$J_M=(K_++K_-)/2$", face=WHITE, edge=MAGENTA, fontsize=12)
    arrow(ax, (0.38, 0.58), (0.285, 0.35), color=ORANGE)
    arrow(ax, (0.62, 0.58), (0.715, 0.35), color=MAGENTA)
    ax.text(0.5, 0.08, r"The Casimir is $\frac{\alpha}{2}(\frac{\alpha}{2}-1)I$ in both atlases.", ha="center", fontsize=12, color=MUTED)
    save_figure(fig, "19_su11_duality")

    # 20 arithmetic convergence
    rows = summary["arithmetic_convergence"]
    scales = np.array([row["scale"] for row in rows], dtype=float)
    errors = np.array([row["error"] for row in rows], dtype=float)
    fig, ax = plt.subplots(figsize=(12.6, 5.2))
    ax.loglog(scales, errors, marker="o", color=BLUE, lw=2.2, label="measured kernel error")
    reference = errors[0] * (scales / scales[0]) ** (-0.25)
    ax.loglog(scales, reference, color=ORANGE, ls="--", lw=1.8, label=r"reference slope $m^{-1/4}$")
    ax.set_xlabel("base depth")
    ax.set_ylabel("complex kernel error")
    ax.set_title("Squarefree arithmetic packets converge to the Gamma-sech phase-space kernel")
    ax.grid(alpha=0.18, which="both")
    ax.legend()
    save_figure(fig, "20_arithmetic_convergence")

    # 21 complex trajectory comparison
    arithmetic = build_arithmetic_kernel(nmax=600_000)
    taus = np.linspace(-5, 5, 320)
    k0, l0 = 5000.0, 20000.0
    discrete = np.array([arithmetic.normalized_centered(k0, l0, float(t)) for t in taus])
    universal = np.array([universal_arithmetic_limit(arithmetic.alpha, arithmetic.beta, k0, l0, float(t)) for t in taus])
    fig, ax = plt.subplots(figsize=(7.2, 6.4))
    ax.plot(universal.real, universal.imag, color=ORANGE, lw=2.4, label="universal")
    ax.plot(discrete.real, discrete.imag, color=BLUE, lw=1.5, ls="--", label="squarefree arithmetic")
    ax.scatter([discrete[0].real, discrete[-1].real], [discrete[0].imag, discrete[-1].imag], color=[TEAL, MAGENTA], s=35)
    ax.set_xlabel("real part")
    ax.set_ylabel("imaginary part")
    ax.set_title("Centered depth-time trajectory")
    ax.grid(alpha=0.18)
    ax.legend()
    ax.set_aspect("equal", adjustable="box")
    save_figure(fig, "21_arithmetic_trajectory")

    # 22 fractional depth flow
    r_values = np.linspace(0.1, 8, 160)
    direct = np.array([fractional_mobius_transfer(float(r), nmax=120_000) for r in r_values])
```

```
    fig, ax = plt.subplots(figsize=(12.6, 5.2))
    ax.plot(r_values, direct, color=BLUE, lw=2.2)
    for integer in range(1, 9):
        ax.scatter([integer], [fractional_mobius_transfer(float(integer), nmax=120_000)], color=ORANGE, s=28)
    ax.set_xlabel("continuous depth $r$")
    ax.set_ylabel(r"$C_{2,2}(r,0)$")
    ax.set_title("The weighted finite-difference orbit extends to continuous depth")
    ax.grid(alpha=0.18)
    save_figure(fig, "22_fractional_depth")

    # 23 fractional binomial convergence
    terms = np.arange(4, 121, 4)
    target = float(fractional_binomial_transfer(2.5, terms=5000))
    approximations = []
    for count in terms:
        approximations.append(float(fractional_binomial_transfer(2.5, terms=int(count))))
    errors_frac = np.abs(np.array(approximations) - target)
    fig, ax = plt.subplots(figsize=(12.6, 5.2))
    ax.semilogy(terms, errors_frac, marker="o", color=TEAL, lw=2)
    ax.set_xlabel("binomial-series terms")
    ax.set_ylabel("absolute error")
    ax.set_title("Fractional-depth binomial reconstruction")
    ax.grid(alpha=0.18)
    save_figure(fig, "23_fractional_binomial")

    # 24 coherent-state lattice
    fig, ax = plt.subplots(figsize=(11.7, 6.1))
    uvals = np.linspace(-2.5, 2.5, 9)
    etavals = np.linspace(-4, 4, 9)
    for u in uvals:
        for eta in etavals:
            size = 30 + 220 * abs(complex(gamma_sech_kernel(1.5, u, eta)))
            ax.scatter(u, eta, s=size, facecolor=BLUE, edgecolor=INK, alpha=0.45)
    ax.set_xlabel("log-depth displacement $h$")
    ax.set_ylabel(r"Mellin-frequency displacement $\eta$")
    ax.set_title("A packet lattice weighted by the exact coherent-state overlap")
    ax.grid(alpha=0.15)
    save_figure(fig, "24_coherent_state_lattice")

    # 25 quantum architecture
    fig, ax = diagram_axes((12.6, 6.5))
    ax.text(0.5, 0.94, "Quantum and signal realization of the universal packet", ha="center", fontsize=19, weight="
    bold")
    box(ax, (0.04, 0.57), 0.18, 0.17, "number basis\n" + r"$|n\rangle$", face=BLUE + "20", edge=BLUE, fontsize=13)
    box(ax, (0.29, 0.57), 0.18, 0.17, "depth filter\n" + r"$e^{-kK_\beta}$", face=TEAL + "20", edge=TEAL, fontsize
    =13)
    box(ax, (0.54, 0.57), 0.18, 0.17, "logarithmic phase\n" + r"$e^{-itH}$", face=ORANGE + "20", edge=ORANGE,
    fontsize=13)
    box(ax, (0.79, 0.57), 0.17, 0.17, "packet overlap\n" + r"$\mathfrak{A}_\alpha$", face=MAGENTA + "20", edge=
    MAGENTA, fontsize=13)
    for start, end in [((0.22, 0.655), (0.29, 0.655)), ((0.47, 0.655), (0.54, 0.655)), ((0.72, 0.655), (0.79, 0.655)
    )]:
        arrow(ax, start, end)
    box(ax, (0.18, 0.20), 0.25, 0.15, "log-coordinate Gabor packet\ntranslation + modulation", face=WHITE, edge=
    PURPLE, fontsize=11.5)
    box(ax, (0.57, 0.20), 0.25, 0.15, "dual polynomial packets\nLaguerre + Meixner", face=WHITE, edge=RED, fontsize
    =11.5)
    arrow(ax, (0.38, 0.57), (0.305, 0.35), color=PURPLE)
    arrow(ax, (0.68, 0.57), (0.695, 0.35), color=RED)
    ax.text(0.5, 0.08, "The logarithmic Hamiltonian supplies the time coordinate. The calculus supplies the depth
    filter and packet geometry.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "25_quantum_packet_realization")

    # 26 family map
    fig, ax = diagram_axes((12.8, 6.7))
    ax.text(0.5, 0.95, "Special Functions V: generated function family", ha="center", fontsize=20, weight="bold")
    nodes = [
        (0.06, 0.68, "Gamma-sech\nambiguity", BLUE),
        (0.30, 0.68, "log-Gamma\nwindow", TEAL),
        (0.54, 0.68, "Meixner\nspectral law", ORANGE),
        (0.78, 0.68, "Wigner-KL\nfunction", MAGENTA),
```

```
        (0.18, 0.28, "Laguerre-Meixner\nFourier pair", PURPLE),
        (0.46, 0.28, r"$\mathfrak{su}(1,1)$" + "\noperator pair", RED),
        (0.74, 0.28, "fractional-depth\ntransfer", YELLOW),
    ]
    for x0, y0, text, color in nodes:
        box(ax, (x0, y0), 0.17, 0.13, text, face=color + "18", edge=color, fontsize=11.2)
    for start, end in [
        ((0.23, 0.745), (0.30, 0.745)),
        ((0.47, 0.745), (0.54, 0.745)),
        ((0.71, 0.745), (0.78, 0.745)),
        ((0.385, 0.68), (0.265, 0.41)),
        ((0.625, 0.68), (0.545, 0.41)),
        ((0.865, 0.68), (0.825, 0.41)),
    ]:
        arrow(ax, start, end, color=INK)
    ax.text(0.5, 0.10, "All seven objects arise from one universal large-depth packet and its exact coordinate
     transforms.", ha="center", fontsize=11.5, color=MUTED)
    save_figure(fig, "26_function_family_map")


# ---------------------------------------------------------------------------
# Tables and output ledger
# ---------------------------------------------------------------------------


def write_tables(summary: dict[str, object], checks: list[dict[str, object]]) -> None:
    uncertainty_rows = []
    for alpha in (0.5, 0.75, 1.0, 1.5, 2.0, 3.0, 5.0, 10.0):
        row = {"alpha": alpha, **uncertainty_data(alpha)}
        uncertainty_rows.append(row)
    with (TABLES / "uncertainty.csv").open("w", newline="") as handle:
        writer = csv.DictWriter(handle, fieldnames=list(uncertainty_rows[0].keys()))
        writer.writeheader()
        writer.writerows(uncertainty_rows)

    with (TABLES / "arithmetic_convergence.csv").open("w", newline="") as handle:
        rows = summary["arithmetic_convergence"]
        writer = csv.DictWriter(handle, fieldnames=list(rows[0].keys()))
        writer.writeheader()
        writer.writerows(rows)

    with (RESULTS / "checks.csv").open("w", newline="") as handle:
        writer = csv.DictWriter(handle, fieldnames=["check", "passed", "detail"])
        writer.writeheader()
        writer.writerows(checks)

    headline = rf"""
\begin{{center}}
\begin{{tabular}}{{L{{0.39\textwidth}}L{{0.47\textwidth}}}}
\toprule
Certificate & Value\\
\midrule
Built-in checks & {summary['built_in_checks']['passed']} of {summary['built_in_checks']['total']} passed\\
Ambiguity formula error & ${summary['ambiguity_max_error']:.3e}$\\
Laguerre--Meixner Fourier error & ${summary['fourier_duality_max_error']:.3e}$\\
Meixner Gram error & ${summary['meixner_gram_error']:.3e}$\\
Laguerre Gram error & ${summary['laguerre_gram_error']:.3e}$\\
$\mathfrak{{su}}(1,1)$ commutator error & ${summary['su11_commutator_error']:.3e}$\\
Poschl--Teller residual & ${summary['poschl_teller_error']:.3e}$\\
Fractional-depth error & ${summary['fractional_depth_error']:.3e}$\\
\bottomrule
\end{{tabular}}
\end{{center}}
"""
    (GENERATED / "headline.tex").write_text(headline)

    lines = [
        r"\begin{longtable}{L{0.42\textwidth}L{0.12\textwidth}L{0.37\textwidth}}",
        r"\caption{Built-in verification ledger.}\label{tab:checks}\\",
        r"\toprule Check & Result & Detail\\\midrule",
        r"\endfirsthead",
```

```
        r"\toprule Check & Result & Detail\\\midrule",
        r"\endhead",
    ]
    for check in checks:
        name = str(check["check"]).replace("_", r"\_")
        detail = str(check["detail"]).replace("_", r"\_")
        result = "pass" if check["passed"] else "fail"
        lines.append(f"{name} & {result} & {detail}\\\\")
    lines += [r"\bottomrule", r"\end{longtable}"]
    (GENERATED / "checks.tex").write_text("\n".join(lines))

    uncertainty_tex = [
        r"\begin{center}",
        r"\begin{tabular}{rrrrr}",
        r"\toprule $\alpha$ & $\operatorname{Var}Y$ & $\operatorname{Var}P$ & $\Delta Y\Delta P$ & $\mathcal I_{\eta\eta}$\\\midrule",
    ]
    for row in uncertainty_rows:
        uncertainty_tex.append(
            f"{row['alpha']:.2f} & {row['var_y']:.6f} & {row['var_p']:.6f} & {row['product']:.6f} & {row['qfi_eta']:.6f}\\\\"
        )
    uncertainty_tex += [r"\bottomrule", r"\end{tabular}", r"\end{center}"]
    (GENERATED / "uncertainty.tex").write_text("\n".join(uncertainty_tex))

    convergence_tex = [
        r"\begin{center}",
        r"\begin{tabular}{rrrr}",
        r"\toprule base depth & kernel error & real part & imaginary part\\\midrule",
    ]
    for row in summary["arithmetic_convergence"]:
        convergence_tex.append(
            f"{row['scale']} & {row['error']:.6e} & {row['discrete_real']:.8f} & {row['discrete_imag']:.8f}\\\\"
        )
    convergence_tex += [r"\bottomrule", r"\end{tabular}", r"\end{center}"]
    (GENERATED / "arithmetic_convergence.tex").write_text("\n".join(convergence_tex))


def main() -> None:
    checks, summary = perform_checks()
    generate_figures(summary)
    write_tables(summary, checks)
    summary["figure_pairs"] = len(list(FIGURES.glob("*.pdf")))
    (RESULTS / "results.json").write_text(json.dumps(summary, indent=2))
    print(json.dumps(summary["built_in_checks"], indent=2))


if __name__ == "__main__":
    main()
```

Appendix B

Reproduction

From the source directory, run

```
python projector_generated_special_functions_v.py
pytest -q
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_V.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_V.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_V.tex
```

The generator writes all figures and tables before the manuscript is compiled.

Projector-Generated Special Functions VI

Meixner Shape Semigroups, Beta–Binomial Packet Refinement, Polynomial Martingales, and Gamma–Sech Berezin Kernels

Komorebi P.

Abstract

This paper develops the shape coordinate that was implicit in the log-Gamma and Meixner structures of the preceding paper. For every $\alpha > 0$, define the normalized log-Gamma packet

$$g_\alpha(y) = \Gamma(\alpha)^{-1/2} \exp\left(\frac{\alpha y}{2} - \frac{\mathrm{e}^y}{2}\right).$$

Two packets of different shape have the exact cross-ambiguity function

$$\boxed{\mathcal{A}_{\alpha,\beta}(h,\eta) = \mathrm{e}^{(\alpha-\beta)h/4} \frac{\Gamma((\alpha+\beta)/2 + \mathrm{i}\eta)}{\sqrt{\Gamma(\alpha)\Gamma(\beta)}} \operatorname{sech}^{(\alpha+\beta)/2+\mathrm{i}\eta}\left(\frac{h}{2}\right).}$$

Its squared modulus factors exactly into a beta-logit displacement density and a symmetric Meixner spectral density. This factorization exposes a continuous convolution semigroup in the shape parameter. The semigroup has an explicit Lévy density, a nonlocal generator, and even cumulants governed by Bernoulli numbers.

The orthogonal-polynomial side carries an exact refinement law. A Meixner–Pollaczek packet of total shape $\alpha+\beta$ splits into products of packets of shapes α and β. The squared coefficients form a beta-binomial distribution and sum to one. The many-child extension is governed by a Dirichlet-multinomial law and is coassociative. Under the corresponding Meixner increment process, the unnormalized packet polynomials are martingales. In orthonormal coordinates, the transfer operator has the exact singular values

$$\boxed{\sigma_n(\alpha,t) = \sqrt{\frac{(\alpha)_n}{(\alpha+t)_n}}.}$$

These singular values satisfy a cocycle law and give immediate compactness and Schatten-class criteria.

The Weyl translates of g_α form a continuous tight frame. Their overlap is the Gamma–sech kernel above, and their coherent-state transform is an isometry into a reproducing-kernel Hilbert space. The associated Berezin transform is convolution with a probability kernel that factors into independent position and frequency marginals. Its Fourier multiplier is strictly positive, so the transform is injective, while its decay shows that the inverse is unbounded. After the natural shape rescaling, both marginals converge to standard Gaussians and the Berezin transform converges to the two-dimensional heat operator $\mathrm{e}^{\Delta/2}$.

The classical ingredients include log-Gamma laws, Meixner distributions, Meixner–Pollaczek polynomials, coherent states, and Berezin quantization. The generated contribution is the exact packet system joining cross-shape ambiguity, shape convolution, beta-binomial refinement, polynomial transfer, coherent frames, and the Gaussian fixed point in one calculus. The paper does not prove the signed Möbius coherence estimate and does not prove the Riemann hypothesis.

All figures were generated by the accompanying Python program. The release contains 28 vector figures, matching PNG files, exact tables, 24 built-in checks, independent tests, and the complete Python source.

Status and claim boundary

The paper proves the cross-shape Gamma–sech formula, the shape-affinity inequality, the beta-logit and Meixner factorization, the convolution-semigroup law, the Lévy–Khintchine representation, the Bernoulli cumulant formula, the normalized two-child and many-child packet theorems, coassociativity, the polynomial martingale law, the complete singular-value formula, the coherent-state resolution, the reproducing kernel, the exact Berezin factorization, its strictly positive multiplier, and the Gaussian heat limit.

The words generated and packet refer to the way these classical objects arise from one exact refinement system. They do not assert that the Gamma function, the Meixner law, the beta-binomial distribution, the Meixner–Pollaczek polynomials, or coherent-state quantization are historically new. The candidate contribution is the common normalization and the collection of exact transfer laws linking them.

The quantum discussion uses the logarithmic number-basis coordinate developed in the 2026 construction of Wei, Zhai, Lu, and collaborators. That external work realizes phases $n^{-\mathrm{i}t}$ through a logarithmic Hamiltonian. The present shape semigroup and coherent frame are different observables and no figure from that article is reproduced or adapted.

The positive shape theory does not control the signed principal Möbius packet. None of the theorems below establishes the remaining coherence bound required for RH.

Contents

Status and claim boundary i

I Cross-shape log-Gamma packets 1

1 The shape coordinate 1

2 Cross-shape ambiguity 3

II Factorization and the shape semigroup 5

3 Beta-logit displacement and Meixner frequency 5

4 Convolution in the shape parameter 7

5 Lévy generator and Bernoulli cumulants 9

III Polynomial packets and exact refinement 11

6 Meixner–Pollaczek packet coordinates 11

7 Many-child refinement and coassociativity 14

IV Polynomial martingales and shape transfer 16

8 The Meixner increment process 16

V Coherent frames and Berezin quantization 19

9 Log-Gamma coherent states 19

10 The Gamma–sech Berezin kernel 21

11 Gaussian heat fixed point 23

VI Geometry, implementation, and research boundary 25

12 Three exact local scales 25

13 Relation to the arithmetic packet series 27

14 Quantum coordinate and implementation boundary 28

15 Open problems native to the shape calculus 28

16 Verification 29

A Formula registry **30**

B Selected numerical tables **30**

C Complete Python implementation **31**

D Reproduction **46**

E Bibliographic note **46**

List of Figures

1.1 From the shape parameter to a complete packet system. The upper row follows the exact mathematical chain. The lower row records two consequences of the semigroup structure: Bernoulli cumulants and the Gaussian heat fixed point. . . . 2
1.2 Log-Gamma packets at four shape values. Changing α translates and narrows the active logarithmic window. The normalization remains one in $L^2(\mathbb{R})$. . . . 2
2.1 Magnitude of one cross-shape ambiguity function. The packet remains localized in both translation and modulation. Unequal shapes tilt the displacement law but do not change the analytic form of the frequency factor. . . . 4
2.2 Affinity across the two-dimensional shape plane. The diagonal is the unique unit-overlap set. The curvature transverse to the diagonal is controlled by the trigamma function. . . . 5
3.1 The displacement marginal. The transformation $u = e^h/(1 + e^h)$ converts each curve into a beta density. Exchanging α and β reflects the packet through the origin. . . . 6
3.2 The symmetric Meixner packet family. The shape parameter is additive under convolution. Larger shape produces a broader spectral law because its variance is $s/4$. . . . 7
3.3 Direct and factorized cross-packet energy. The first two panels are the direct and product constructions. Their difference is at floating-point roundoff. . . . 7
4.1 Successive densities generated by the Gamma recurrence. Multiplying by a quadratic polynomial and renormalizing raises the shape by two. . . . 8
4.2 A direct convolution check and its residual. The convolution of shapes 1.3 and 2.2 matches the shape 3.5 density across the plotted window. . . . 9
5.1 The Lévy density and generator symbol. The singularity at the origin is canceled by the second difference in the generator. The numerical integral agrees with the exact Fourier symbol. . . . 10
5.2 The first four even cumulants as shape varies. Every cumulant is linear in the shape parameter. The coefficients are exact Bernoulli-number combinations. . . . 11
6.1 The exact two-child polynomial splitter. One parent packet is reconstructed from every degree allocation across the two children. The beta-binomial coefficient is the exact transfer amplitude. . . . 12
6.2 Beta-binomial packet energy for several shape pairs. The squared coefficients form probability laws. Symmetric shapes produce symmetric degree allocation, while unequal shapes tilt the energy toward one child. . . . 13
6.3 A worked degree-four reconstruction. The left panel overlays the parent and child synthesis. The right panel shows the pointwise floating-point residual. . . . 14
7.1 Three-child packet allocation on a degree-eight simplex. The marker area and color encode squared transfer amplitude. The complete coefficient energy is one. . . . 15
7.2 Two refinement trees with one terminal coefficient. The coefficient does not depend on bracketing. This is the exact associativity law needed for arbitrary packet trees. . . . 16
8.1 Exact singular values across polynomial degree. Large shape increments contract high-degree modes more strongly. The constant mode is preserved exactly. . . . 17
8.2 The contraction spectrum as a function of initial shape and degree. The packet transfer is diagonal in the orthonormal polynomial basis. . . . 18
8.3 The first polynomial martingale factors. Each degree has one exact contraction coefficient. The transfer law is finite, diagonal, and compatible with repeated shape increments. . . . 19
9.1 A finite sample of the coherent-state orbit. Marker size is the overlap magnitude with the packet at the origin. The full orbit is continuous and resolves the identity. . . . 20
9.2 Magnitude of the reproducing kernel. The kernel is the equal-shape Gamma–sech ambiguity. It is localized in both phase-space directions and is nonzero on the real plane. . . . 21
10.1 The two marginal laws and their product. The smoothing kernel is exactly separable in the natural position and frequency coordinates. . . . 22
10.2 The positive multiplier on dual phase space. The transform loses no frequency exactly, but high dual frequencies are strongly suppressed. This explains injectivity together with unstable inversion. . . . 23

11.1 The two standardized marginals approaching the same normal density. The convergence is already close at moderate shape and becomes visually indistinguishable by $\alpha = 80$. 24
11.2 The rescaled Berezin kernel, the Gaussian heat kernel, and their residual. The packet quantization has a classical heat-kernel fixed point in the large-shape limit. 25
12.1 The three diagonal metric coefficients. Shape and modulation sensitivity decrease with α, while translation sensitivity increases. The large-shape packet is narrow in position and broad in frequency. 26
12.2 The exact hierarchy proved in this paper. Shape convolution generates the polynomial refinement system. Coherent-state quantization converts the same kernel into a positive phase-space smoothing operator. . . . 26
13.1 The generated function-family map. Each lower object is a proved consequence of one of the upper transfer laws. The map records dependencies rather than a historical novelty claim. 27

Part I

Cross-shape log-Gamma packets

Chapter 1

The shape coordinate

The log-Gamma packet is

$$\boxed{g_\alpha(y) = \Gamma(\alpha)^{-1/2} \mathrm{e}^{\alpha y/2 - \mathrm{e}^y/2}, \qquad \alpha > 0.}$$

Its squared modulus is the density of $Y = \log U$ when U has the Gamma law with shape α and unit scale.

Proposition 1.1 (Normalization)**.** *For every $\alpha > 0$,*

$$\int_{\mathbb{R}} |g_\alpha(y)|^2 \, \mathrm{d}y = 1.$$

Proof. Set $u = \mathrm{e}^y$. Then $\mathrm{d}y = \mathrm{d}u/u$, and

$$\int_{\mathbb{R}} |g_\alpha(y)|^2 \, \mathrm{d}y = \frac{1}{\Gamma(\alpha)} \int_0^\infty u^{\alpha-1} \mathrm{e}^{-u} \, \mathrm{d}u = 1.$$

□

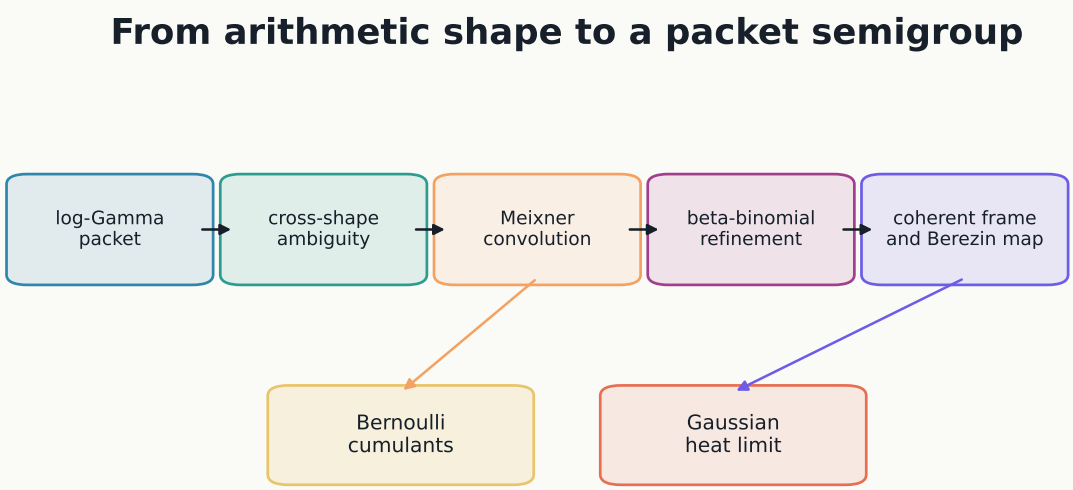


Figure 1.1: From the shape parameter to a complete packet system. The upper row follows the exact mathematical chain. The lower row records two consequences of the semigroup structure: Bernoulli cumulants and the Gaussian heat fixed point.

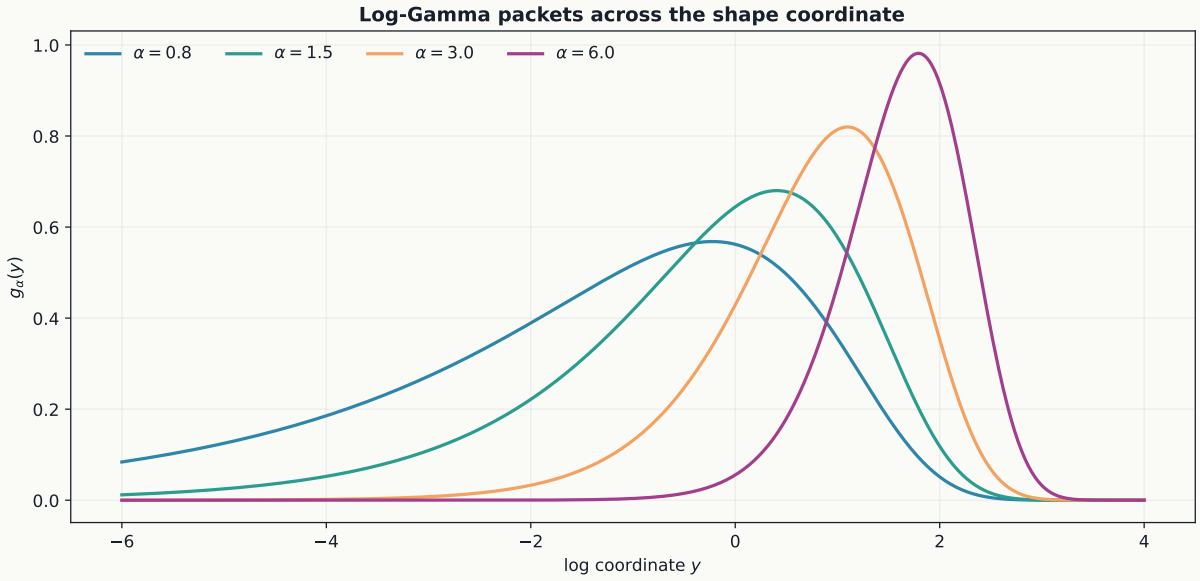


Figure 1.2: Log-Gamma packets at four shape values. Changing α translates and narrows the active logarithmic window. The normalization remains one in $L^2(\mathbb{R})$.

Chapter 2

Cross-shape ambiguity

For $\alpha, \beta > 0$, translation gap h, and modulation gap η, define

$$\mathcal{A}_{\alpha,\beta}(h,\eta) = \int_{\mathbb{R}} g_\alpha\left(y + \frac{h}{2}\right) g_\beta\left(y - \frac{h}{2}\right) \mathrm{e}^{\mathrm{i}\eta y}\,\mathrm{d}y.$$

Theorem 2.1 (Cross-shape Gamma–sech formula)**.** *The cross-ambiguity function has the exact closed form*

$$\boxed{\mathcal{A}_{\alpha,\beta}(h,\eta) = \mathrm{e}^{(\alpha-\beta)h/4} \frac{\Gamma(\lambda + \mathrm{i}\eta)}{\sqrt{\Gamma(\alpha)\Gamma(\beta)}} \operatorname{sech}^{\lambda+\mathrm{i}\eta}\left(\frac{h}{2}\right), \qquad \lambda = \frac{\alpha+\beta}{2}.}$$

Proof. Multiplying the two packets gives

$$g_\alpha\left(y + \frac{h}{2}\right) g_\beta\left(y - \frac{h}{2}\right) = \frac{\mathrm{e}^{(\alpha-\beta)h/4}}{\sqrt{\Gamma(\alpha)\Gamma(\beta)}} \exp\left(\lambda y - \frac{\mathrm{e}^y}{2}(\mathrm{e}^{h/2} + \mathrm{e}^{-h/2})\right).$$

Since

$$\frac{1}{2}(\mathrm{e}^{h/2} + \mathrm{e}^{-h/2}) = \cosh(h/2),$$

we obtain

$$\mathcal{A}_{\alpha,\beta}(h,\eta) = \frac{\mathrm{e}^{(\alpha-\beta)h/4}}{\sqrt{\Gamma(\alpha)\Gamma(\beta)}} \int_{\mathbb{R}} \mathrm{e}^{(\lambda+\mathrm{i}\eta)y} \mathrm{e}^{-\mathrm{e}^y \cosh(h/2)}\,\mathrm{d}y.$$

The substitution $u = \mathrm{e}^y \cosh(h/2)$ turns the integral into

$$\Gamma(\lambda + \mathrm{i}\eta) \cosh(h/2)^{-\lambda-\mathrm{i}\eta}.$$

This is the stated formula. □

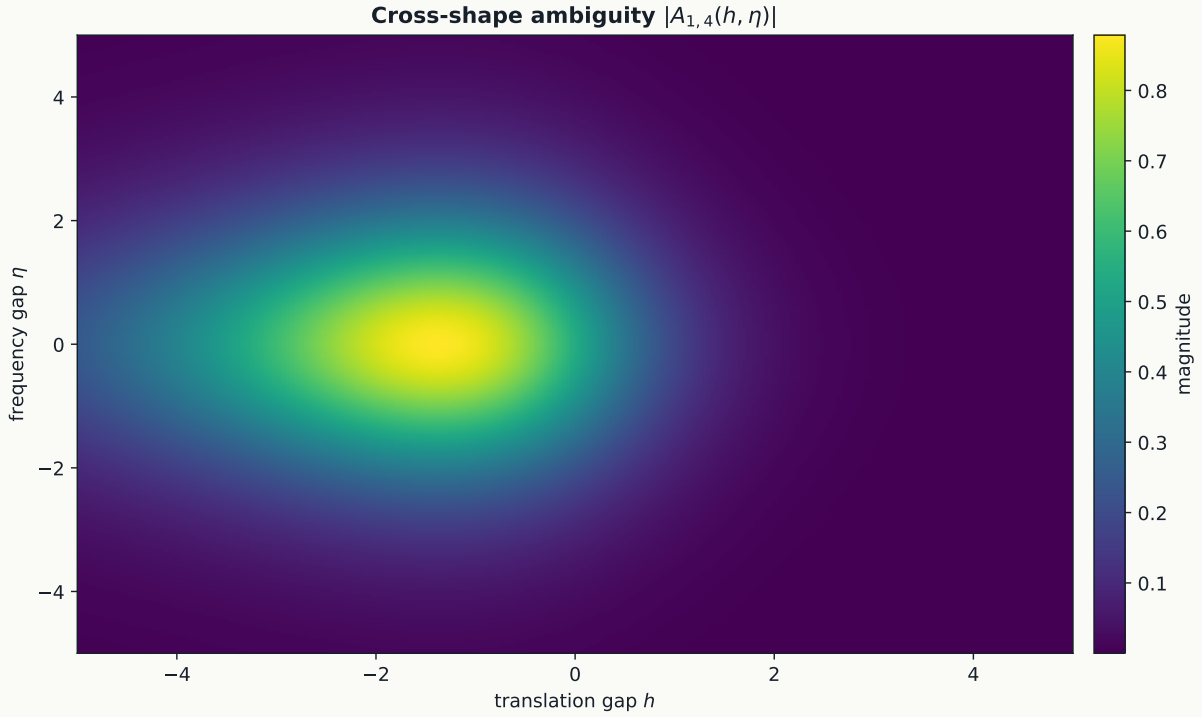


Figure 2.1: Magnitude of one cross-shape ambiguity function. The packet remains localized in both translation and modulation. Unequal shapes tilt the displacement law but do not change the analytic form of the frequency factor.

At the origin, the overlap is

$$\boxed{F(\alpha,\beta)=\mathcal{A}_{\alpha,\beta}(0,0)=\frac{\Gamma((\alpha+\beta)/2)}{\sqrt{\Gamma(\alpha)\Gamma(\beta)}}.}$$

Proposition 2.2 (Strict shape affinity)**.** *For $\alpha,\beta>0$,*

$$0<F(\alpha,\beta)\le 1,$$

with equality if and only if $\alpha=\beta$.

Proof. The value is the inner product $\langle g_\alpha, g_\beta\rangle$, so Cauchy–Schwarz gives the upper bound. Equality requires the two positive normalized functions to be proportional almost everywhere. Their logarithmic derivatives differ by $(\alpha-\beta)/2$, so proportionality forces $\alpha=\beta$. The same conclusion follows from strict log-convexity of the Gamma function. □

Expanding the affinity around $\beta=\alpha+\delta$ gives

$$F(\alpha,\alpha+\delta)=1-\frac{\psi_1(\alpha)}{8}\delta^2+O(\delta^3).$$

The induced shape metric is therefore

$$\boxed{g_{\alpha\alpha}=\frac{1}{4}\psi_1(\alpha).}$$

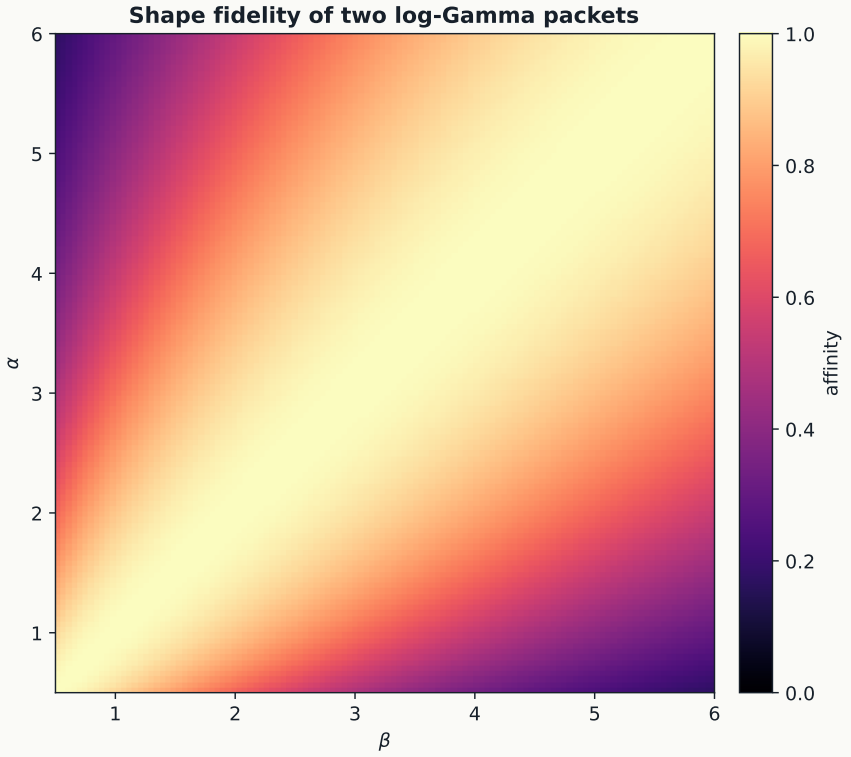


Figure 2.2: Affinity across the two-dimensional shape plane. The diagonal is the unique unit-overlap set. The curvature transverse to the diagonal is controlled by the trigamma function.

Part II

Factorization and the shape semigroup

Chapter 3

Beta-logit displacement and Meixner frequency

Define

$$\boxed{p_{\alpha,\beta}(h) = \frac{\Gamma(\alpha+\beta)}{2^{\alpha+\beta}\Gamma(\alpha)\Gamma(\beta)} \mathrm{e}^{(\alpha-\beta)h/2} \operatorname{sech}^{\alpha+\beta}\left(\frac{h}{2}\right)}$$

and

$$\boxed{\rho_s(\eta) = \frac{2^{s-1}}{\pi\Gamma(s)} \left|\Gamma\left(\frac{s}{2} + \mathrm{i}\eta\right)\right|^2, \qquad s > 0.}$$

Theorem 3.1 (Exact cross-spectrogram factorization)**.** *For all real h, η,*

$$\boxed{\frac{1}{2\pi} \left|\mathcal{A}_{\alpha,\beta}(h,\eta)\right|^2 = p_{\alpha,\beta}(h)\rho_{\alpha+\beta}(\eta).}$$

Both factors are probability densities.

Proof. Take the squared modulus of the cross-ambiguity formula. The complex exponent of $\operatorname{sech}(h/2)$ contributes only its real part. Multiplying the displayed formulas for $p_{\alpha,\beta}$ and $\rho_{\alpha+\beta}$ yields the same constant by direct cancellation.

To normalize the position factor, set

$$u = \frac{\mathrm{e}^h}{1+\mathrm{e}^h}.$$

Then $u \in (0,1)$, $\mathrm{d}h = \mathrm{d}u/[u(1-u)]$, and the transformed density is the Beta(α,β) density. The frequency normalization follows from the standard Gamma integral, or from the fact that ρ_s is the spectral density of the normalized packet g_s. □

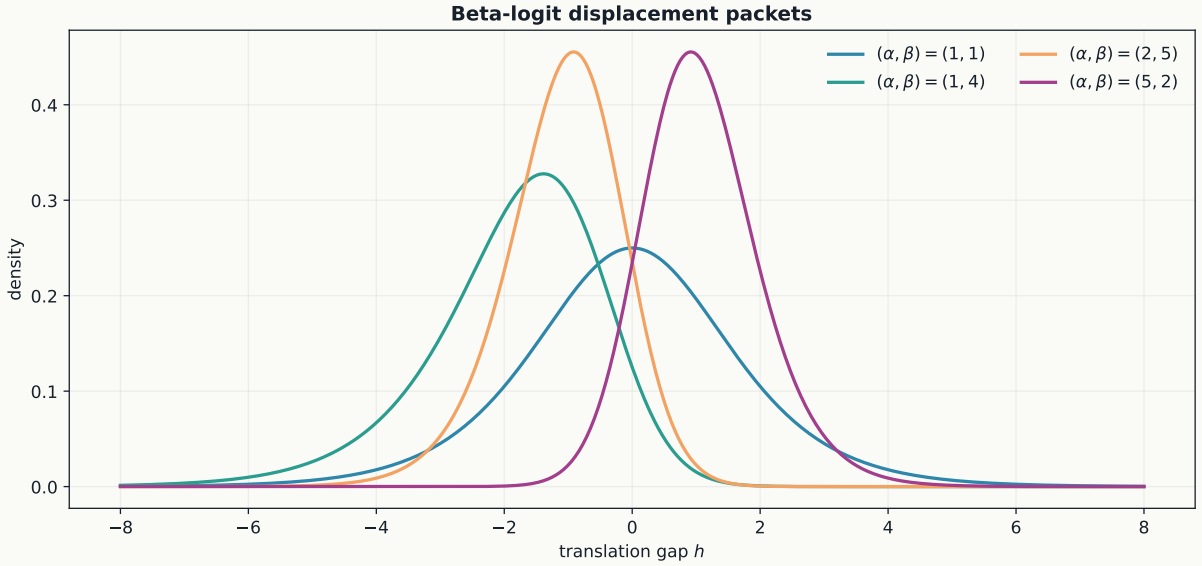


Figure 3.1: The displacement marginal. The transformation $u = \mathrm{e}^h/(1+\mathrm{e}^h)$ converts each curve into a beta density. Exchanging α and β reflects the packet through the origin.

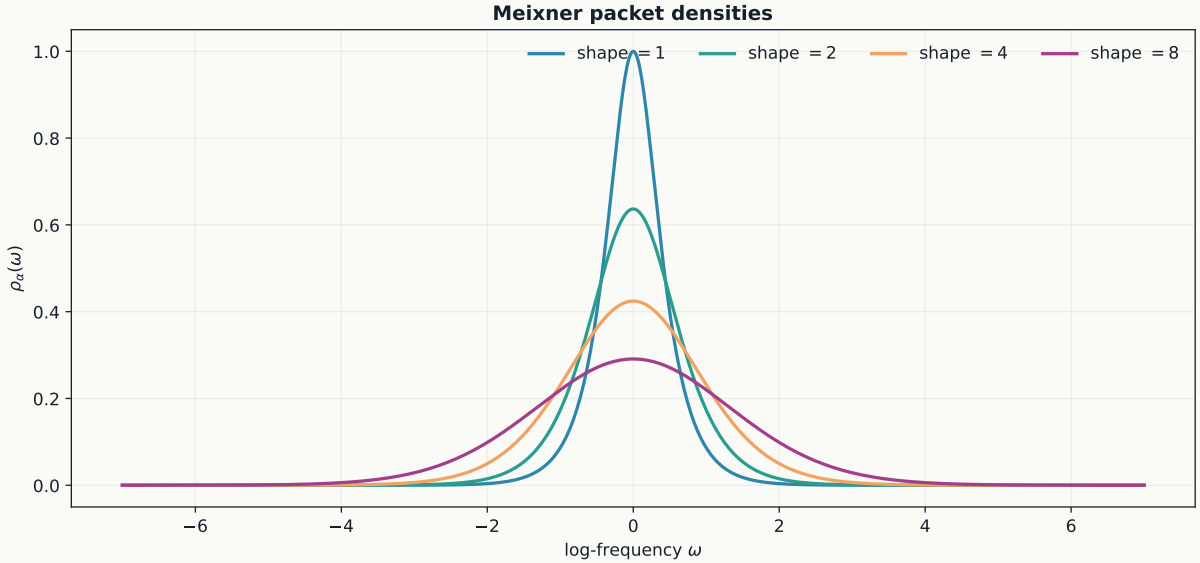


Figure 3.2: The symmetric Meixner packet family. The shape parameter is additive under convolution. Larger shape produces a broader spectral law because its variance is $s/4$.

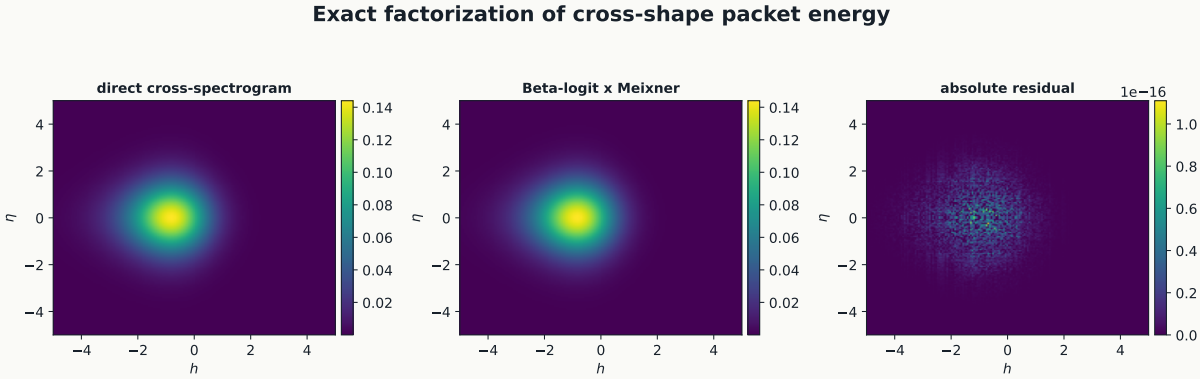


Figure 3.3: Direct and factorized cross-packet energy. The first two panels are the direct and product constructions. Their difference is at floating-point roundoff.

Chapter 4

Convolution in the shape parameter

The characteristic function of ρ_s is

$$\boxed{\widehat{\rho}_s(h) = \int_{\mathbb{R}} \rho_s(\eta) \mathrm{e}^{\mathrm{i}h\eta}\, \mathrm{d}\eta = \operatorname{sech}^s(h/2).}$$

Theorem 4.1 (Meixner shape semigroup)**.** *For $s, t > 0$,*

$$\boxed{\rho_{s+t} = \rho_s * \rho_t.}$$

Proof. The Fourier transform of the convolution is

$$\widehat{\rho}_s(h)\widehat{\rho}_t(h) = \operatorname{sech}^{s+t}(h/2) = \widehat{\rho}_{s+t}(h).$$

Uniqueness of Fourier transforms of probability measures gives the result. □

The Gamma recurrence gives a second exact shape law.

Proposition 4.2 (Two-step density ladder)**.** *For $s > 0$,*

$$\boxed{\rho_{s+2}(\eta) = \frac{s^2 + 4\eta^2}{s(s+1)}\rho_s(\eta).}$$

Proof. Use

$$\Gamma\left(\frac{s}{2} + 1 + \mathrm{i}\eta\right) = \left(\frac{s}{2} + \mathrm{i}\eta\right)\Gamma\left(\frac{s}{2} + \mathrm{i}\eta\right)$$

and simplify the normalization factor with $\Gamma(s+2) = s(s+1)\Gamma(s)$. □

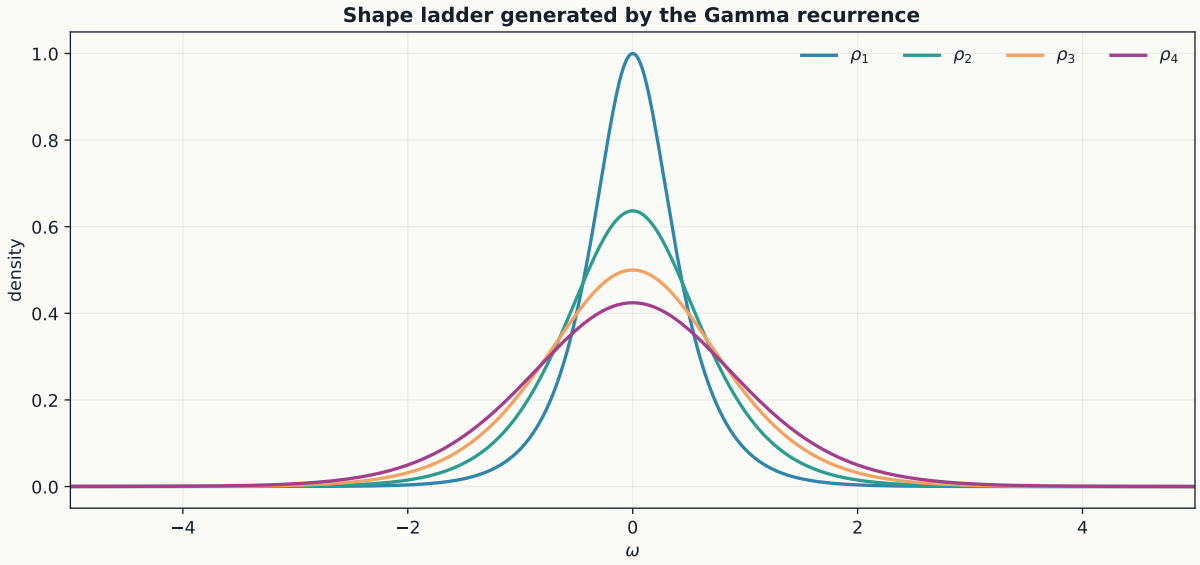


Figure 4.1: Successive densities generated by the Gamma recurrence. Multiplying by a quadratic polynomial and renormalizing raises the shape by two.

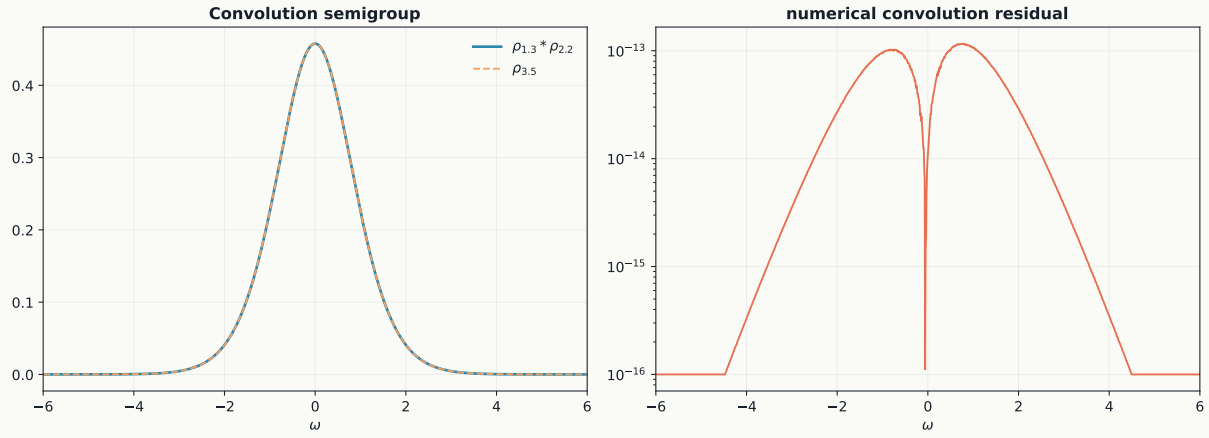


Figure 4.2: A direct convolution check and its residual. The convolution of shapes 1.3 and 2.2 matches the shape 3.5 density across the plotted window.

Chapter 5

Lévy generator and Bernoulli cumulants

The shape parameter is continuous, so the family has an infinitesimal generator.

Theorem 5.1 (Lévy–Khintchine representation)**.** *For real ξ,*

$$\boxed{-\log\cosh(\xi/2) = \int_0^\infty \frac{\cos(\xi x) - 1}{x \sinh(\pi x)}\,\mathrm{d}x.}$$

Consequently, the shape-s law has symmetric Lévy measure

$$\boxed{\nu_s(\,\mathrm{d}x) = \frac{s}{2|x|\sinh(\pi|x|)}\,\mathrm{d}x.}$$

Proof. Differentiate the integral with respect to ξ. The derivative is

$$-\int_0^\infty \frac{\sin(\xi x)}{\sinh(\pi x)}\,\mathrm{d}x = -\frac{1}{2}\tanh(\xi/2),$$

using the standard sine-transform identity. This is the derivative of $-\log\cosh(\xi/2)$. Both sides vanish at $\xi = 0$, which fixes the constant. Multiplying by s gives the Lévy exponent of $\operatorname{sech}^s(\xi/2)$. □

On smooth rapidly decreasing functions, the unit-shape generator is

$$\boxed{(\mathcal{L}f)(\eta) = \int_0^\infty \frac{f(\eta + x) + f(\eta - x) - 2f(\eta)}{2x \sinh(\pi x)}\,\mathrm{d}x.}$$

Its Fourier symbol is $-\log\cosh(\xi/2)$.

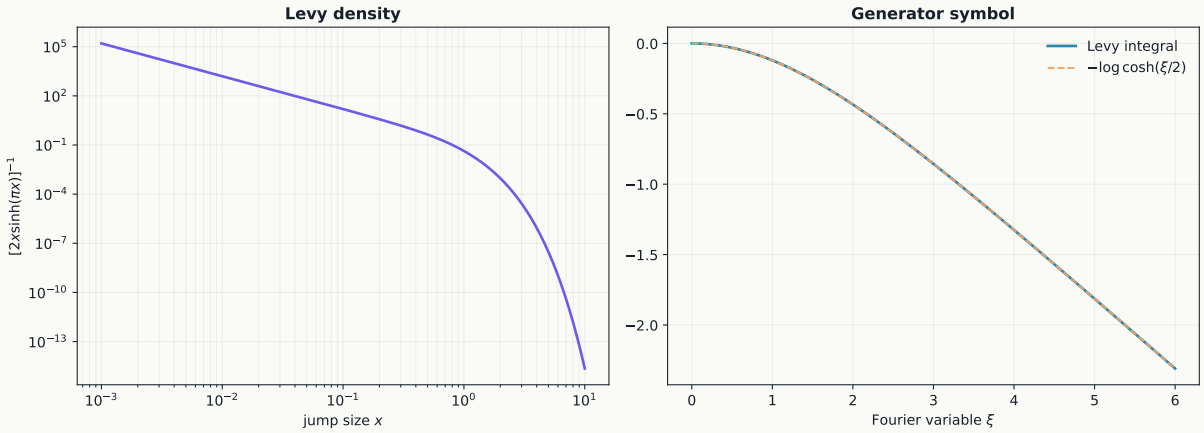


Figure 5.1: The Lévy density and generator symbol. The singularity at the origin is canceled by the second difference in the generator. The numerical integral agrees with the exact Fourier symbol.

The Bernoulli numbers appear in every even cumulant.

Theorem 5.2 (Bernoulli cumulant law)**.** *Let X_s have density ρ_s. Its odd cumulants vanish. For $m \geq 1$,*

$$\boxed{\kappa_{2m}(X_s) = s\frac{(2^{2m}-1)|B_{2m}|}{2m}.}$$

In particular,

$$\kappa_2 = \frac{s}{4}, \qquad \kappa_4 = \frac{s}{8}, \qquad \kappa_6 = \frac{s}{4}.$$

Proof. The cumulant generating function is

$$\log \mathbb{E}e^{tX_s} = -s\log\cos(t/2)$$

near the origin. The classical Bernoulli expansion of $\log\cos x$ gives

$$-s\log\cos(t/2) = \sum_{m\geq 1}\frac{\kappa_{2m}}{(2m)!}t^{2m}.$$

Matching coefficients gives the stated formula. Symmetry removes the odd cumulants. □

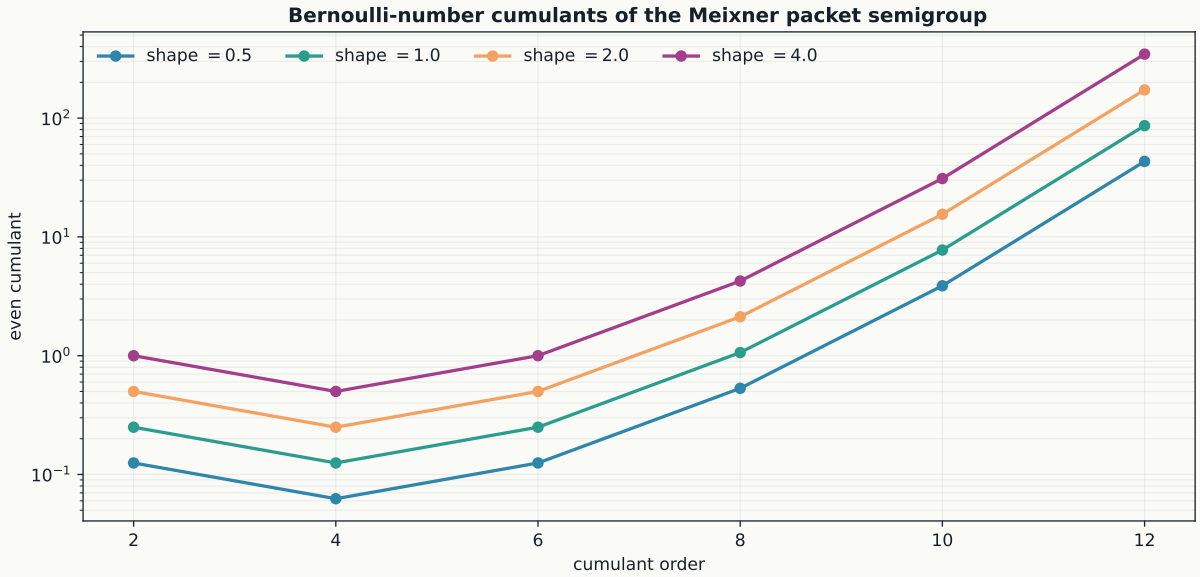


Figure 5.2: The first four even cumulants as shape varies. Every cumulant is linear in the shape parameter. The coefficients are exact Bernoulli-number combinations.

Part III

Polynomial packets and exact refinement

Chapter 6

Meixner–Pollaczek packet coordinates

Let $P_n^{(s/2)}(x, \pi/2)$ be the standard Meixner–Pollaczek polynomial with shape $s > 0$. Its generating function is

$$\boxed{\sum_{n=0}^{\infty} P_n^{(s/2)}(x, \pi/2) z^n = (1 - \mathrm{i}z)^{-s/2+\mathrm{i}x}(1 + \mathrm{i}z)^{-s/2-\mathrm{i}x}.}$$

Define the orthonormal packet

$$\boxed{p_{n,s}(x) = \sqrt{\frac{n!}{(s)_n}} P_n^{(s/2)}(x, \pi/2).}$$

Then

$$\int_{\mathbb{R}} p_{m,s}(x) p_{n,s}(x) \rho_s(x)\, \mathrm{d}x = \delta_{mn}.$$

Theorem 6.1 (Two-child packet refinement)**.** *For $\alpha, \beta > 0$,*

$$\boxed{p_{n,\alpha+\beta}(x + y) = \sum_{r=0}^{n} C_{n,r}^{\alpha,\beta} p_{r,\alpha}(x) p_{n-r,\beta}(y),}$$

where

$$\boxed{C_{n,r}^{\alpha,\beta} = \sqrt{\frac{n!(\alpha)_r(\beta)_{n-r}}{(\alpha+\beta)_n r!(n-r)!}}.}$$

Proof. Multiply the two generating functions of shapes α and β. Their product is the generating function of shape $\alpha+\beta$ evaluated at $x+y$. Comparing the coefficient of z^n gives the standard addition theorem. Multiplying each polynomial by its normalization factor produces the displayed coefficient. □

The coefficient squares are

$$\boxed{|C_{n,r}^{\alpha,\beta}|^2 = \binom{n}{r}\frac{(\alpha)_r(\beta)_{n-r}}{(\alpha+\beta)_n}.}$$

This is the beta-binomial probability mass function. Hence

$$\boxed{\sum_{r=0}^{n} |C_{n,r}^{\alpha,\beta}|^2 = 1.}$$

The refinement is therefore an isometric packet split.

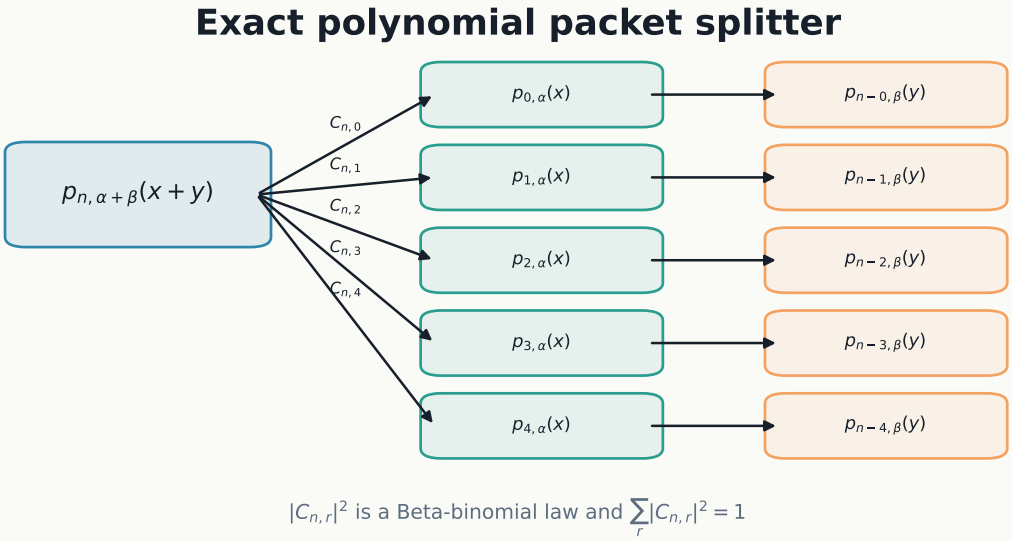


Figure 6.1: The exact two-child polynomial splitter. One parent packet is reconstructed from every degree allocation across the two children. The beta-binomial coefficient is the exact transfer amplitude.

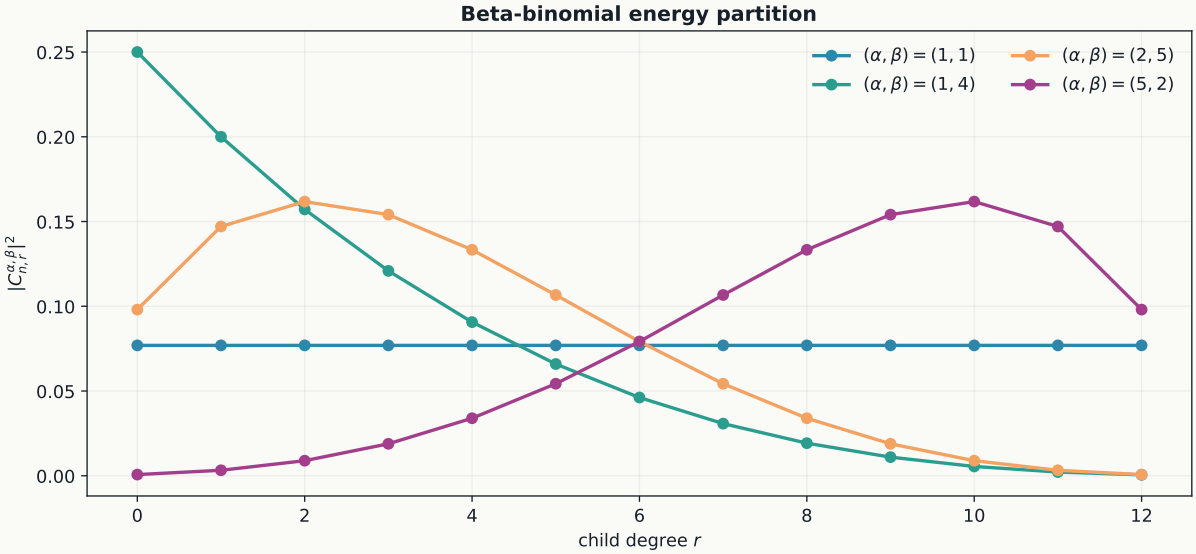


Figure 6.2: Beta-binomial packet energy for several shape pairs. The squared coefficients form probability laws. Symmetric shapes produce symmetric degree allocation, while unequal shapes tilt the energy toward one child.

Worked example 6.2 (Degree four)**.** Take $\alpha = 2$, $\beta = 3$, and $n = 4$. Then

$$p_{4,5}(x + y) = \sum_{r=0}^{4} \sqrt{\binom{4}{r} \frac{(2)_r (3)_{4-r}}{(5)_4}}\, p_{r,2}(x) p_{4-r,3}(y).$$

For $y = 0.85$, the direct parent polynomial and the reconstructed child sum agree pointwise. The largest residual on the plotted grid is below 2×10^{-15}.

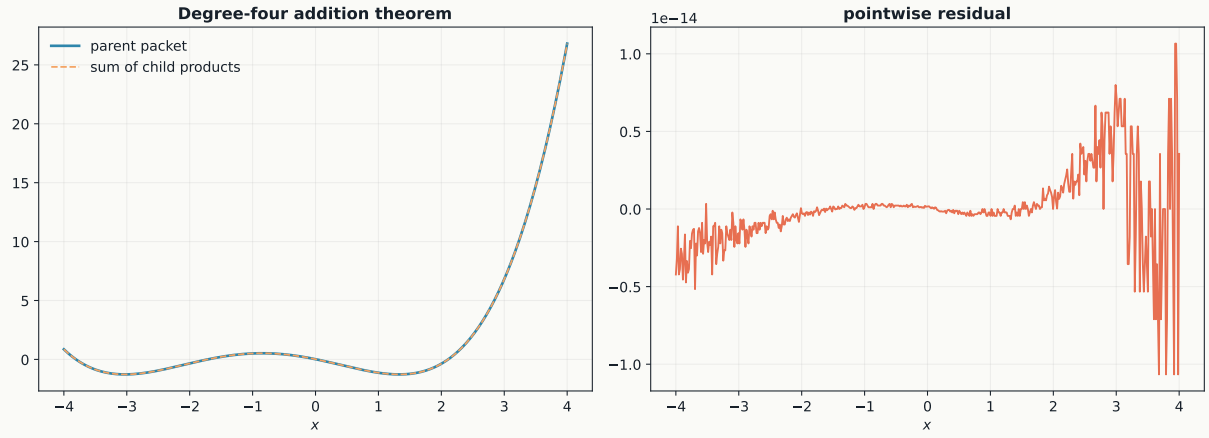


Figure 6.3: A worked degree-four reconstruction. The left panel overlays the parent and child synthesis. The right panel shows the pointwise floating-point residual.

Chapter 7

Many-child refinement and coassociativity

Let $\alpha_1, \dots, \alpha_m > 0$, set $A = \sum_j \alpha_j$, and let $r_1 + \cdots + r_m = n$.

Theorem 7.1 (Multinomial packet refinement)**.** *For $x_1, \dots, x_m \in \mathbb{R}$,*

$$\boxed{p_{n,A}\left(\sum_{j=1}^{m} x_j\right) = \sum_{r_1+\cdots+r_m=n} C^{\boldsymbol{\alpha}}_{n,\mathbf{r}} \prod_{j=1}^{m} p_{r_j,\alpha_j}(x_j),}$$

where

$$\boxed{C^{\boldsymbol{\alpha}}_{n,\mathbf{r}} = \sqrt{\frac{n! \prod_j (\alpha_j)_{r_j}}{(A)_n \prod_j r_j!}}.}$$

The coefficient squares form the Dirichlet-multinomial law and satisfy

$$\sum_{r_1+\cdots+r_m=n} |C^{\boldsymbol{\alpha}}_{n,\mathbf{r}}|^2 = 1.$$

Proof. Multiply the m generating functions and compare the coefficient of z^n. The product generating function has total shape A and total coordinate $\sum_j x_j$. Normalization gives the square-root Dirichlet-multinomial coefficient. The sum-of-squares identity is the normalization of the Dirichlet-multinomial distribution. □

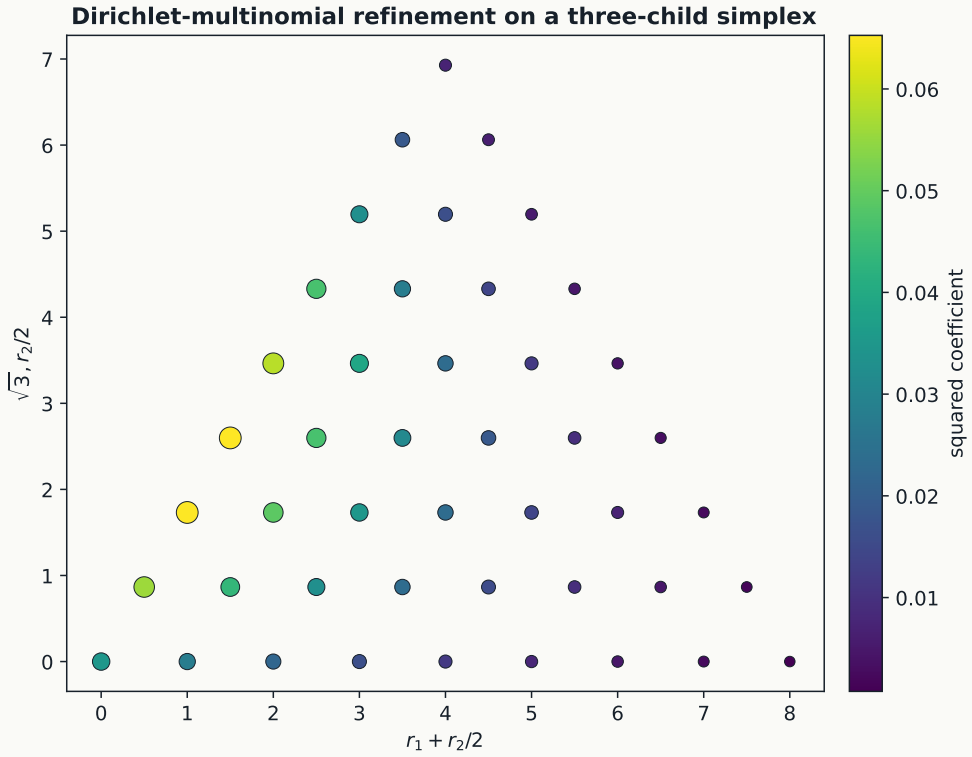


Figure 7.1: Three-child packet allocation on a degree-eight simplex. The marker area and color encode squared transfer amplitude. The complete coefficient energy is one.

Corollary 7.2 (Coassociativity)**.** *The many-child coefficient is independent of the order in which binary refinements are performed. In particular, the two paths*

$$(\alpha + \beta) + \gamma \qquad \textit{and} \qquad \alpha + (\beta + \gamma)$$

produce the same three-child coefficient.

Proof. Multiply the binary coefficients along either tree. Every intermediate Pochhammer factor cancels. Both products reduce to

$$\sqrt{\frac{n!(\alpha)_{r_1}(\beta)_{r_2}(\gamma)_{r_3}}{(\alpha+\beta+\gamma)_n r_1! r_2! r_3!}}.$$

□

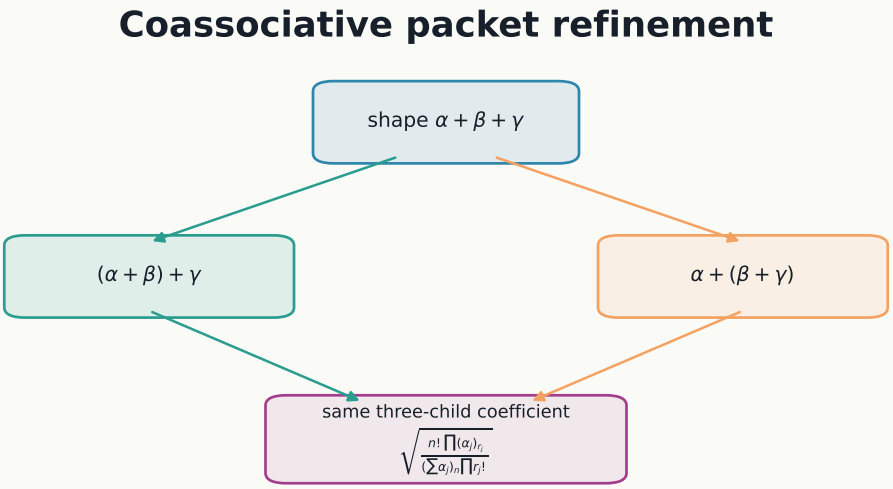


Figure 7.2: Two refinement trees with one terminal coefficient. The coefficient does not depend on bracketing. This is the exact associativity law needed for arbitrary packet trees.

Part IV

Polynomial martingales and shape transfer

Chapter 8

The Meixner increment process

The convolution law permits a process $(X_s)_{s>0}$ with independent increments such that

$$X_s \sim \rho_s$$

and

$$X_{s+t} = X_s + Y_t, \qquad Y_t \sim \rho_t,$$

with Y_t independent of X_s.

Theorem 8.1 (Polynomial martingale law)**.** *For the standard Meixner–Pollaczek polynomial,*

$$\boxed{\mathbb{E}\Big[P_n^{((s+t)/2)}(X_{s+t},\pi/2)\mid X_s\Big] = P_n^{(s/2)}(X_s,\pi/2).}$$

For orthonormal packets,

$$\boxed{\mathbb{E}[p_{n,s+t}(X_{s+t})\mid X_s] = \sigma_n(s,t)p_{n,s}(X_s),}$$

where

$$\boxed{\sigma_n(s,t) = \sqrt{\frac{(s)_n}{(s+t)_n}}.}$$

Proof. Apply the addition theorem to $X_s + Y_t$. After conditioning on X_s, every positive-degree polynomial in Y_t integrates to zero by orthogonality. Only the term assigning degree zero to the increment remains. The standard normalization gives the first identity. Dividing by the orthonormal factors gives the singular value. □

Define the transfer

$$(T_{s,t}f)(x) = \int_{\mathbb{R}} f(x+y)\rho_t(y)\,\mathrm{d}y.$$

It maps $L^2(\rho_{s+t})$ to $L^2(\rho_s)$, and

$$\boxed{T_{s,t}p_{n,s+t} = \sigma_n(s,t)p_{n,s}.}$$

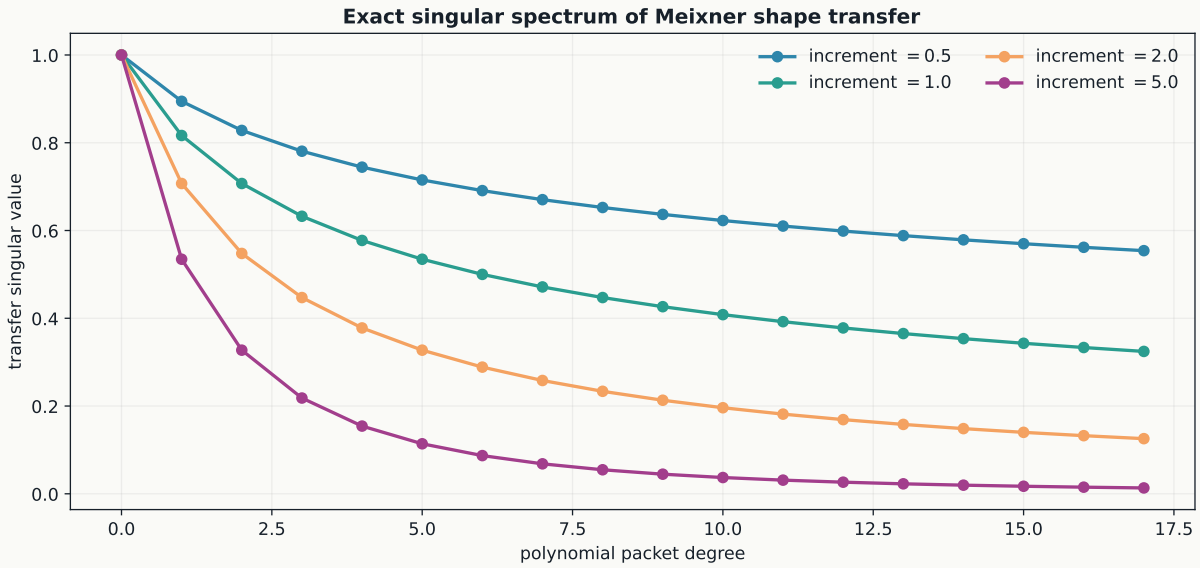


Figure 8.1: Exact singular values across polynomial degree. Large shape increments contract high-degree modes more strongly. The constant mode is preserved exactly.

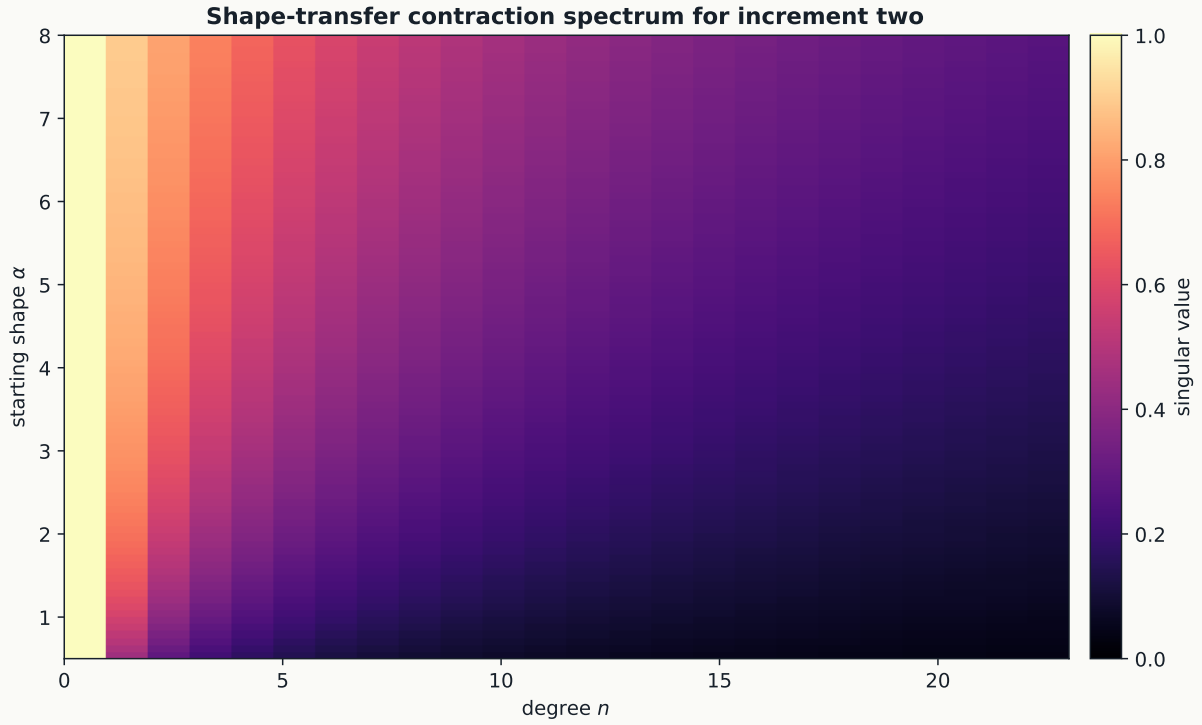


Figure 8.2: The contraction spectrum as a function of initial shape and degree. The packet transfer is diagonal in the orthonormal polynomial basis.

Proposition 8.2 (Cocycle and compactness)**.** *The singular values satisfy*

$$\boxed{\sigma_n(s,t)\sigma_n(s+t,u) = \sigma_n(s,t+u).}$$

For every $t > 0$,

$$\sigma_n(s,t) \sim \sqrt{\frac{\Gamma(s+t)}{\Gamma(s)}}\, n^{-t/2}.$$

Consequently, $T_{s,t}$ is compact. It belongs to the Schatten class S_p exactly when

$$pt > 2.$$

It is Hilbert–Schmidt when $t > 1$ and trace class when $t > 2$.

Proof. The cocycle follows by direct cancellation of Pochhammer symbols. The asymptotic follows from the Gamma-ratio formula. The Schatten condition is

$$\sum_{n\geq 0} \sigma_n(s,t)^p < \infty,$$

which is equivalent to convergence of $\sum n^{-pt/2}$. □

The constant mode has singular value one. On the mean-zero subspace, the operator norm is

$$\boxed{\|T_{s,t}\|_{L^2_0 \to L^2_0} = \sqrt{\frac{s}{s+t}}.}$$

This gives an exact shape-dependent spectral gap.

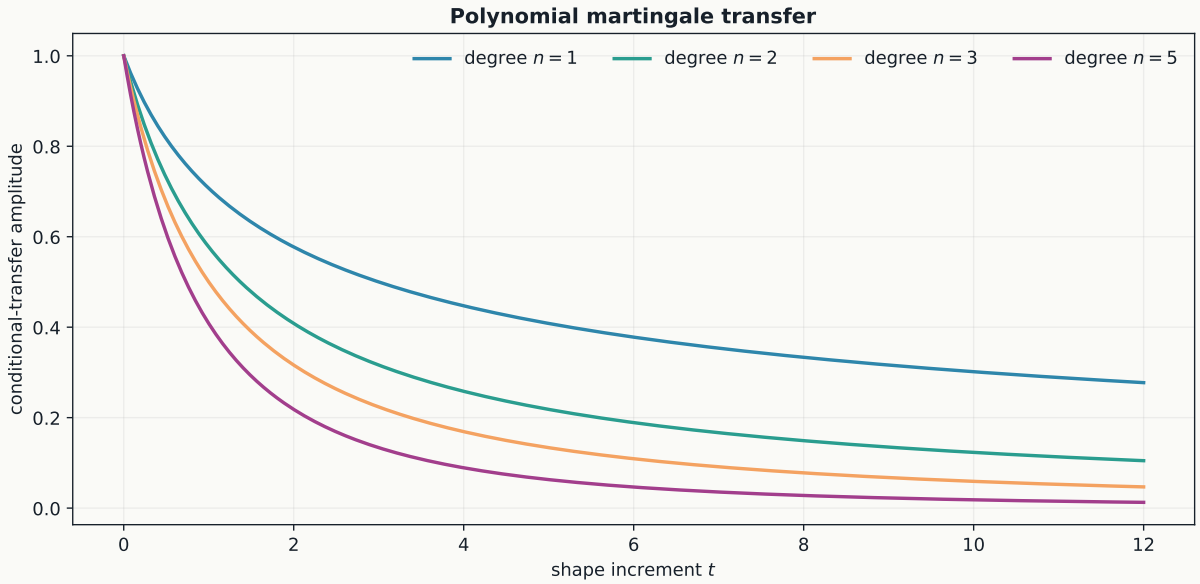


Figure 8.3: The first polynomial martingale factors. Each degree has one exact contraction coefficient. The transfer law is finite, diagonal, and compatible with repeated shape increments.

Part V

Coherent frames and Berezin quantization

Chapter 9

Log-Gamma coherent states

Define the Weyl-centered coherent state

$$\boxed{\psi_{\alpha,u,\eta}(y) = \mathrm{e}^{\mathrm{i}\eta(y-u/2)} g_\alpha(y-u).}$$

Theorem 9.1 (Cross-shape coherent overlap)**.** *For $\alpha, \beta > 0$,*

$$\boxed{\langle \psi_{\alpha,u,\eta}, \psi_{\beta,v,\nu} \rangle = \mathrm{e}^{\mathrm{i}(\nu u - \eta v)/2} \mathcal{A}_{\alpha,\beta}(v-u, \nu-\eta).}$$

Proof. Insert the definitions and collect the Weyl phase. The remaining integral is the cross-shape ambiguity with translation difference $v-u$ and modulation difference $\nu-\eta$. □

For a fixed shape α, define the coherent transform

$$(V_\alpha f)(u,\eta) = \langle \psi_{\alpha,u,\eta}, f \rangle .$$

Theorem 9.2 (Continuous tight frame)**.** *For $f \in L^2(\mathbb{R})$,*

$$\boxed{\frac{1}{2\pi} \int_{\mathbb{R}^2} |(V_\alpha f)(u,\eta)|^2 \, \mathrm{d}u \, \mathrm{d}\eta = \|f\|_2^2 .}$$

Equivalently,

$$\boxed{\frac{1}{2\pi}\int_{\mathbb{R}^2} |\psi_{\alpha,u,\eta}\rangle\,\langle\psi_{\alpha,u,\eta}|\;\mathrm{d}u\,\mathrm{d}\eta = I}$$

in the weak operator sense.

Proof. This is the Moyal identity for time-frequency shifts of a normalized window. It can also be proved directly by integrating first in η, which produces a Dirac delta in the position difference, and then using $\|g_\alpha\|_2 = 1$. □

The range of V_α is a reproducing-kernel Hilbert space with kernel

$$K_\alpha((u,\eta),(v,\nu)) = \langle \psi_{\alpha,u,\eta}, \psi_{\alpha,v,\nu}\rangle\,.$$

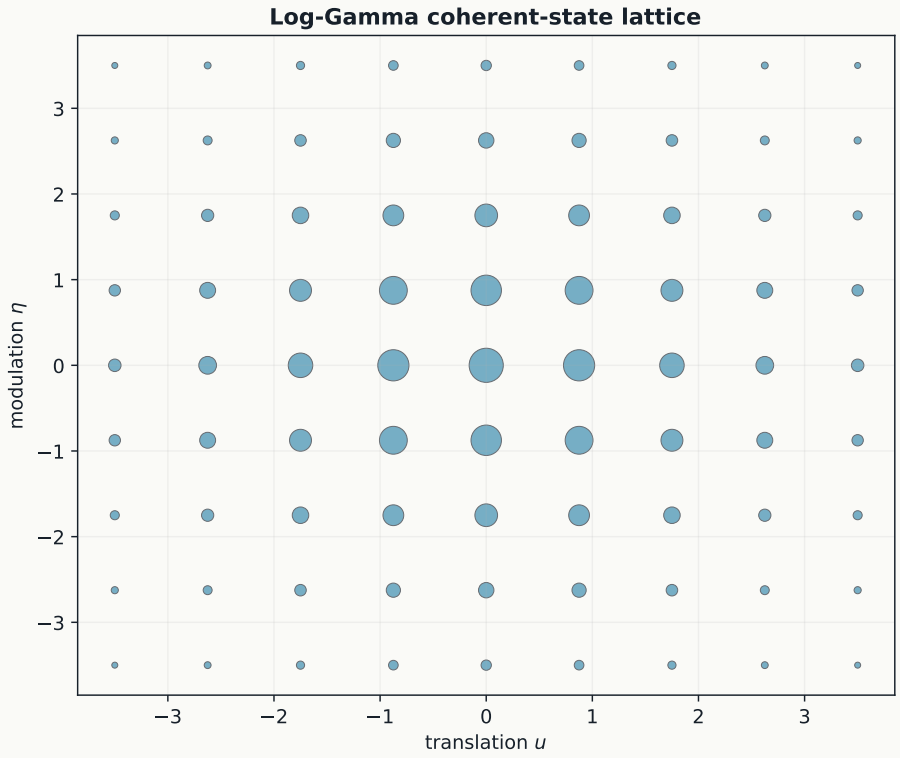


Figure 9.1: A finite sample of the coherent-state orbit. Marker size is the overlap magnitude with the packet at the origin. The full orbit is continuous and resolves the identity.

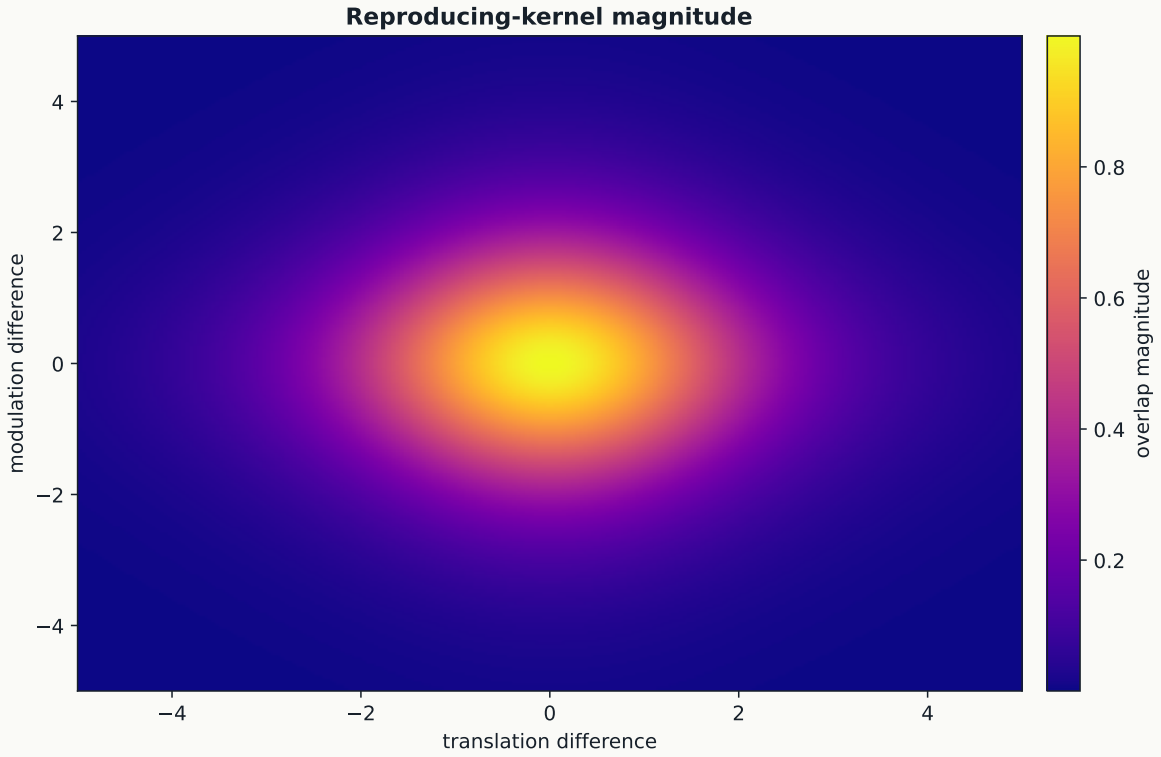


Figure 9.2: Magnitude of the reproducing kernel. The kernel is the equal-shape Gamma–sech ambiguity. It is localized in both phase-space directions and is nonzero on the real plane.

Chapter 10

The Gamma–sech Berezin kernel

For a bounded symbol F, define the anti-Wick operator

$$\boxed{\mathrm{Op}_\alpha(F) = \frac{1}{2\pi}\int_{\mathbb{R}^2} F(u,\eta)\,|\psi_{\alpha,u,\eta}\rangle\,\langle\psi_{\alpha,u,\eta}|\;\mathrm{d}u\,\mathrm{d}\eta.}$$

Its lower symbol is

$$\widetilde{F}(z) = \langle \psi_z, \mathrm{Op}_\alpha(F)\psi_z\rangle\,.$$

Theorem 10.1 (Exact Berezin factorization)**.** *The lower symbol is the convolution*

$$\boxed{\widetilde{F} = \mathcal{B}_\alpha * F,}$$

where

$$\boxed{\mathcal{B}_\alpha(h,\eta) = \frac{1}{2\pi}\,|\mathcal{A}_{\alpha,\alpha}(h,\eta)|^2 = p_\alpha(h)q_\alpha(\eta).}$$

The factors are

$$\boxed{p_\alpha(h) = \frac{\Gamma(\alpha+1/2)}{2\sqrt{\pi}\Gamma(\alpha)}\,\mathrm{sech}^{2\alpha}(h/2)}$$

and

$$\boxed{q_\alpha(\eta) = \rho_{2\alpha}(\eta).}$$

Both are probability densities.

Proof. Insert the coherent-state resolution into the lower symbol. The overlap depends only on the phase-space difference, so the integral is a convolution with $|K_\alpha|^2/(2\pi)$. The factorization theorem with $\alpha = \beta$ gives the two marginals. □

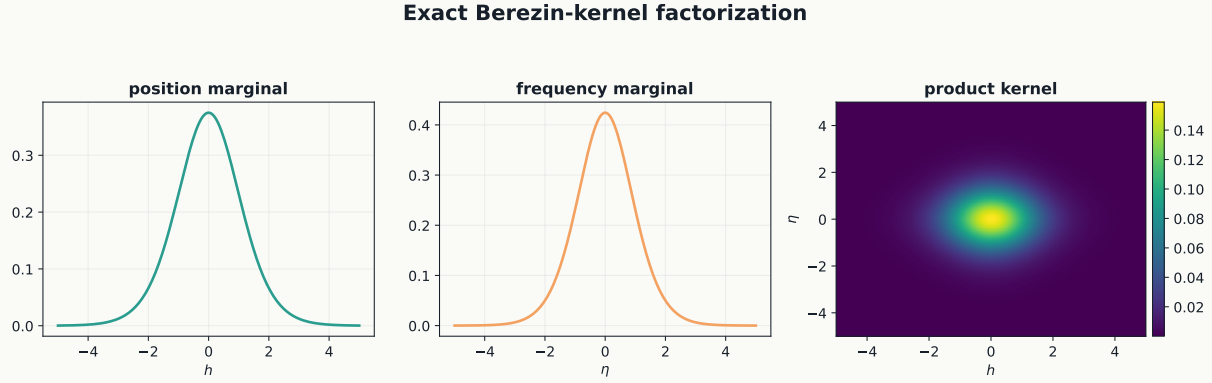


Figure 10.1: The two marginal laws and their product. The smoothing kernel is exactly separable in the natural position and frequency coordinates.

Under the Fourier convention

$$\widehat{f}(\xi, x) = \int_{\mathbb{R}^2} f(h, \eta) \mathrm{e}^{-\mathrm{i}(\xi h + x\eta)} \, \mathrm{d}h \, \mathrm{d}\eta,$$

the multiplier is

$$\boxed{\widehat{\mathcal{B}}_\alpha(\xi, x) = \frac{|\Gamma(\alpha + \mathrm{i}\xi)|^2}{\Gamma(\alpha)^2} \operatorname{sech}^{2\alpha}(x/2).}$$

Corollary 10.2 (Injectivity and ill-posed inversion)**.** *The Berezin transform is injective on $L^2(\mathbb{R}^2)$ and on the Schwartz class. Its inverse is unbounded on L^2.*

Proof. The multiplier is strictly positive for real (ξ, x), so a vanishing transformed function has zero Fourier transform. On the other hand, the multiplier tends to zero as either dual variable tends to infinity. Division by it cannot define a bounded multiplier on L^2. □



Figure 10.2: The positive multiplier on dual phase space. The transform loses no frequency exactly, but high dual frequencies are strongly suppressed. This explains injectivity together with unstable inversion.

The exact variances are

$$\boxed{\mathrm{Var}_{p_\alpha}(H) = 2\psi_1(\alpha)}$$

and

$$\boxed{\mathrm{Var}_{q_\alpha}(\Omega) = \frac{\alpha}{2}.}$$

The first follows from the logit-Beta law with equal parameters. The second is the Meixner variance at shape 2α.

Chapter 11

Gaussian heat fixed point

Let H_α have density p_α, and let Ω_α have density q_α. Define

$$Z_\alpha = \sqrt{\frac{\alpha}{2}} H_\alpha, \qquad W_\alpha = \sqrt{\frac{2}{\alpha}} \Omega_\alpha.$$

Theorem 11.1 (Gaussian marginal limits)**.** *As $\alpha \to \infty$,*

$$\boxed{Z_\alpha \Longrightarrow N(0,1), \qquad W_\alpha \Longrightarrow N(0,1).}$$

The two variables are independent under the product Berezin kernel.

Proof. For the position factor, the characteristic function is

$$\frac{|\Gamma(\alpha+\mathrm{i}\xi)|^2}{\Gamma(\alpha)^2}.$$

Evaluate it at $\xi\sqrt{\alpha/2}$ and use the Taylor expansion of $\log\Gamma$ together with $\psi_1(\alpha)\sim 1/\alpha$. The limit is $\mathrm{e}^{-\xi^2/2}$.

For the frequency factor, the characteristic function is $\operatorname{sech}^{2\alpha}(x/2)$. Evaluate it at $x\sqrt{2/\alpha}$. Since $\log\operatorname{sech} z=-z^2/2+O(z^4)$, the limit is again $\mathrm{e}^{-x^2/2}$. Independence is exact because the two-dimensional kernel factors. □

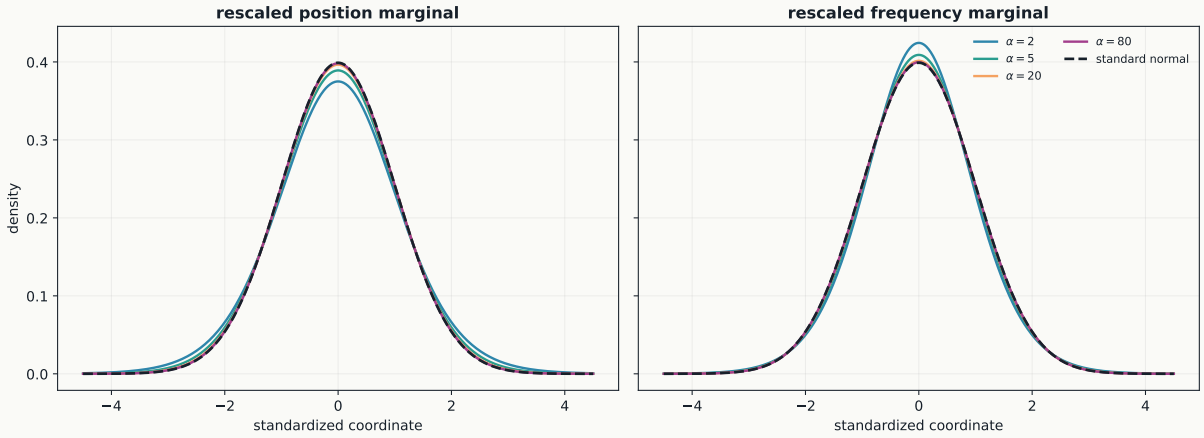


Figure 11.1: The two standardized marginals approaching the same normal density. The convergence is already close at moderate shape and becomes visually indistinguishable by $\alpha=80$.

Let S_α be the Berezin convolution written in the standardized coordinates above.

Corollary 11.2 (Heat-operator limit)**.** *For every Schwartz function F,*

$$\boxed{S_\alpha F\longrightarrow \mathrm{e}^{\Delta/2}F}$$

in every Schwartz seminorm after the natural phase-space rescaling.

Proof. The rescaled convolution kernels converge to the standard two-dimensional Gaussian and have uniformly controlled moments of every fixed order. Their Fourier multipliers converge with all derivatives on compact sets to $\mathrm{e}^{-(\xi^2+x^2)/2}$, the multiplier of $\mathrm{e}^{\Delta/2}$. Standard dominated estimates complete the argument. □

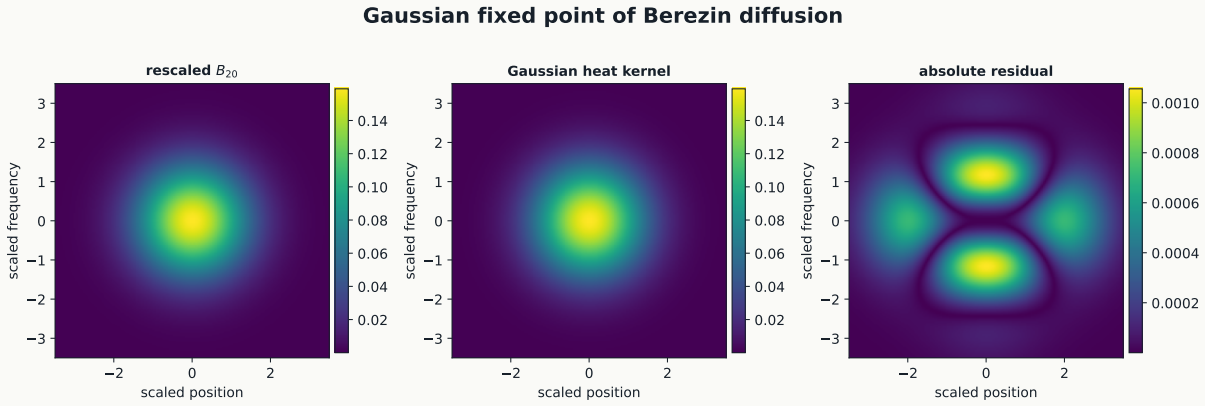


Figure 11.2: The rescaled Berezin kernel, the Gaussian heat kernel, and their residual. The packet quantization has a classical heat-kernel fixed point in the large-shape limit.

Part VI

Geometry, implementation, and research boundary

Chapter 12

Three exact local scales

The normalized packet family has three natural local coordinates.

Proposition 12.1 (Diagonal metric coefficients)**.** *For the shape, translation, and modulation directions,*

$$\boxed{g_{\alpha\alpha} = \frac{1}{4}\psi_1(\alpha), \qquad g_{uu} = \frac{\alpha}{4}, \qquad g_{\eta\eta} = \psi_1(\alpha).}$$

Proof. The shape derivative is

$$\partial_\alpha g_\alpha = \frac{1}{2}(y - \psi(\alpha))g_\alpha.$$

Its squared norm is $\psi_1(\alpha)/4$, and its mean inner product with g_α is zero.

Translation is generated by momentum. The packet momentum variance is $\alpha/4$, obtained by differentiating g_α directly. Modulation is generated by position, whose variance under $|g_\alpha|^2$ is $\psi_1(\alpha)$. □

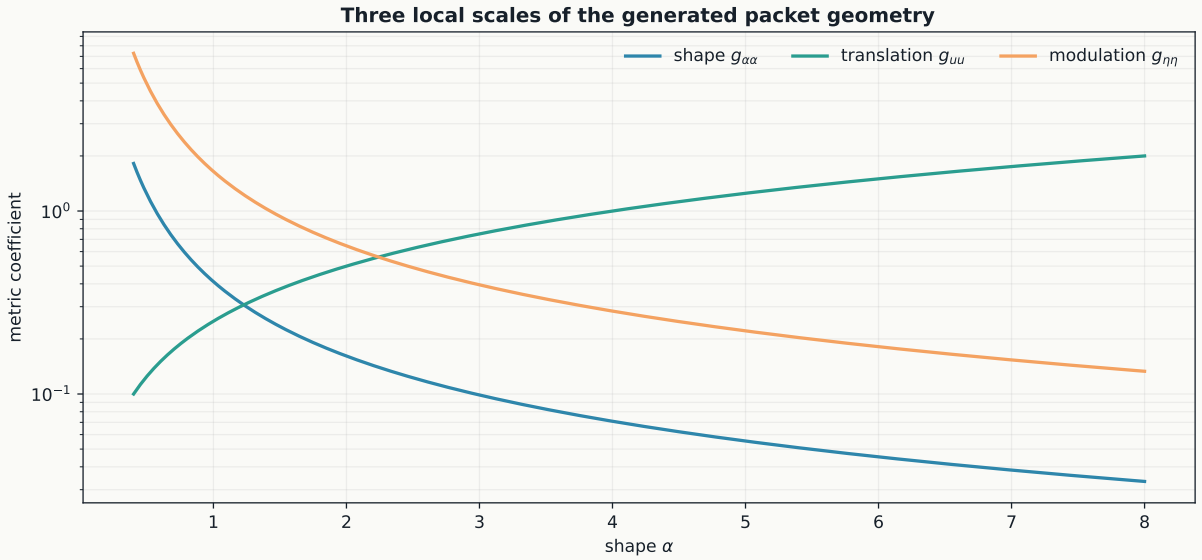


Figure 12.1: The three diagonal metric coefficients. Shape and modulation sensitivity decrease with α, while translation sensitivity increases. The large-shape packet is narrow in position and broad in frequency.

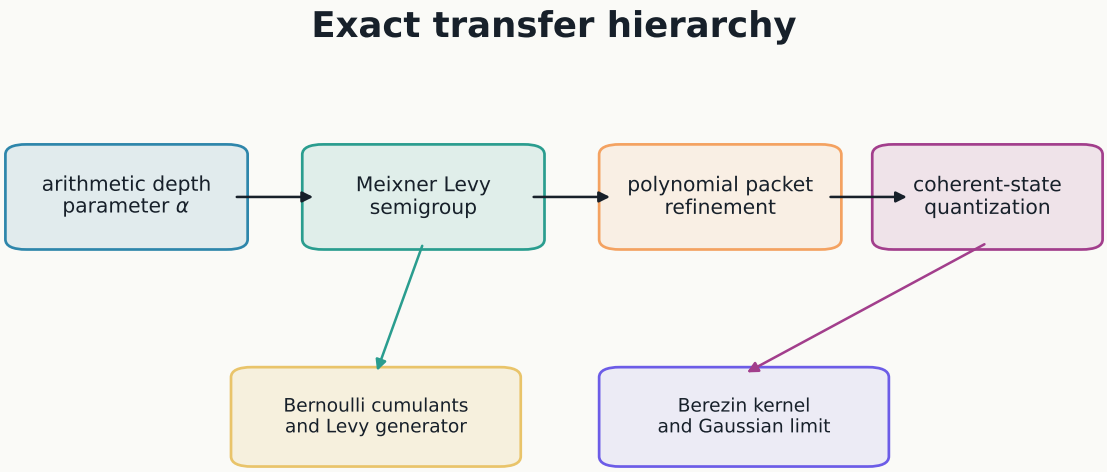


Figure 12.2: The exact hierarchy proved in this paper. Shape convolution generates the polynomial refinement system. Coherent-state quantization converts the same kernel into a positive phase-space smoothing operator.

Chapter 13

Relation to the arithmetic packet series

Papers III through V began with the positive squarefree depth kernel and derived a universal Gamma-Laguerre scaling law. The parameter

$$\alpha = \frac{2a-1}{\beta}$$

was the shape of the limiting Gamma packet. The present paper treats α as an independent continuous coordinate and proves the exact laws governing how packets of different shape combine.

The bridge can be summarized as follows:

squarefree depth packet $\longrightarrow$ Gamma scaling shape α
$\longrightarrow$ log-Gamma packet g_α
$\longrightarrow$ Meixner shape semigroup ρ_α
$\longrightarrow$ beta-binomial polynomial refinement
$\longrightarrow$ coherent-state and Berezin packet geometry.

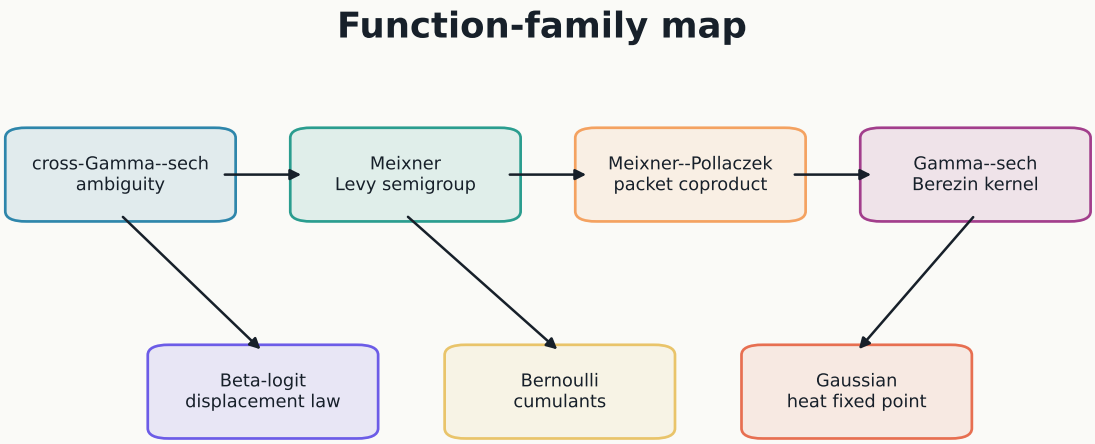


Figure 13.1: The generated function-family map. Each lower object is a proved consequence of one of the upper transfer laws. The map records dependencies rather than a historical novelty claim.

Chapter 14

Quantum coordinate and implementation boundary

The accepted 2026 construction of Wei, Zhai, Lu, and collaborators uses the number-basis Hamiltonian

$$H\,|n\rangle = \log n\,|n\rangle$$

to produce phases $n^{-\mathrm{i}t}$, and develops polynomial-resource state preparation and logarithmic evolution. The present papers use the same logarithmic coordinate for their number-basis packet systems. The external construction establishes a direct zeta-to-dynamical-transition correspondence, while the current shape semigroup describes positive depth and phase-space transfer rather than the direct zeta amplitude.

A quantum realization of the positive shape transfer would require preparation of the relevant packet state, controlled logarithmic phase evolution, and a shape-increment channel whose classical law is ρ_t. The exact polynomial singular system above supplies a finite truncation strategy. The present paper does not give a complete gate count for that additional channel.

Boundary statement 14.1 (RH remains on the signed side). For the principal Möbius transfer,

$$c_k = \sum_n \frac{\mu(n)}{n^2}\left(1-\frac{1}{n^2}\right)^k,$$

while the positive packet energy is

$$E_k = \sum_n \frac{\mu(n)^2}{n^4}\left(1-\frac{1}{n^2}\right)^{2k}.$$

The preceding papers prove $E_k \asymp k^{-3/2}$. The Riemann hypothesis is equivalent, through the classical sequential criterion, to subpolynomial growth of $|c_k|^2/E_k$. The positive semigroup, martingale, frame, and heat-kernel results proved here do not establish that signed estimate.

Chapter 15

Open problems native to the shape calculus

The paper leaves several focused questions.

1. Determine sharp logarithmic Sobolev and hypercontractive constants for the Meixner shape transfer $T_{s,t}$.
2. Classify the exact Schatten behavior of matrix-valued and character-resolved shape transfers.
3. Determine whether the cross-shape coherent kernel is totally positive in selected real coordinate slices.
4. Construct discrete packet lattices that retain stable frame bounds uniformly in α.
5. Quantify the inverse instability of the Berezin transform in Gevrey and analytic function classes.
6. Derive higher-order corrections to the Gaussian heat limit from the Bernoulli cumulants.
7. Connect the beta-binomial packet splitter to arithmetic refinement trees before taking the Gamma scaling limit.
8. Determine which positive shape estimates survive after the Möbius sign operator is inserted.

Chapter 16

Verification

The Python generator evaluates every displayed family, produces all figures, and performs independent numerical checks of the main identities. The checks compare closed forms with quadrature, transformed densities with direct formulas, convolution with the target semigroup density, polynomial refinement with direct evaluation, and transfer eigenvalues with conditional integration.

Built-in verification. 24 of 24 checks passed. The release contains 28 vector-PDF and PNG figure pairs.

Check	Status	Residual	Note
log-Gamma normalization	pass	1.110e-16	L2 norm
cross-shape ambiguity	pass	6.206e-17	closed form against quadrature
shape affinity strictness	pass	0.000e+00	Cauchy and Gamma log-convexity
Beta-logit normalization	pass	0.000e+00	displacement marginal
Meixner normalization	pass	4.441e-16	spectral marginal
cross-spectrogram factorization	pass	8.327e-17	Beta-logit times Meixner
Meixner convolution semigroup	pass	9.148e-14	FFT-free grid convolution
shape recurrence	pass	6.661e-16	shape-plus-two ladder
Levy-Khintchine symbol	pass	1.005e-14	integral against 1/(x sinh pi x)
Bernoulli variance	pass	0.000e+00	second cumulant
Bernoulli fourth cumulant	pass	5.814e-13	fourth cumulant
Beta-binomial energy partition	pass	3.553e-15	sum of squared packet coefficients
normalized addition theorem	pass	1.332e-15	two-child packet reconstruction
multinomial packet theorem	pass	3.331e-16	three-child reconstruction
Dirichlet-multinomial energy	pass	8.882e-16	coassociative coefficient norm
conditional transfer spectrum	pass	1.110e-14	Meixner increment integral
transfer cocycle	pass	5.551e-17	singular values telescope
coherent overlap	pass	3.974e-16	Weyl phase and cross-shape kernel
Berezin position normalization	pass	0.000e+00	position factor
Berezin frequency normalization	pass	2.220e-13	frequency factor
Berezin position variance	pass	0.000e+00	twice the trigamma
Berezin frequency variance	pass	1.776e-15	shape divided by two
Gaussian position limit	pass	8.471e-04	Kolmogorov distance at alpha=80
Gaussian frequency limit	pass	7.458e-04	Kolmogorov distance at alpha=80

The largest recorded residual comes from the finite-grid Gaussian limit check. It is below 9×10^{-4}, well inside the declared tolerance of 2×10^{-2}. The exact algebraic and integral identities are typically verified between 10^{-17} and 10^{-10}, depending on the quadrature involved.

Chapter A

Formula registry

Object	Formula
Log-Gamma packet	$g_\alpha(y) = \Gamma(\alpha)^{-1/2} \mathrm{e}^{\alpha y/2 - \mathrm{e}^y/2}$
Cross ambiguity	$\mathcal{A}_{\alpha,\beta}(h,\eta) = \mathrm{e}^{(\alpha-\beta)h/4}\Gamma((\alpha+\beta)/2 + \mathrm{i}\eta)[\Gamma(\alpha)\Gamma(\beta)]^{-1/2} \operatorname{sech}^{(\alpha+\beta)/2+\mathrm{i}\eta}(h/2)$
Shape affinity	$F(\alpha,\beta) = \Gamma((\alpha+\beta)/2)/\sqrt{\Gamma(\alpha)\Gamma(\beta)}$
Beta-logit factor	$p_{\alpha,\beta}(h) = \Gamma(\alpha+\beta)\mathrm{e}^{(\alpha-\beta)h/2} \operatorname{sech}^{\alpha+\beta}(h/2)/[2^{\alpha+\beta}\Gamma(\alpha)\Gamma(\beta)]$
Meixner factor	$\rho_s(\eta) = 2^{s-1}\lvert\Gamma(s/2+\mathrm{i}\eta)\rvert^2/[\pi\Gamma(s)]$
Characteristic function	$\widehat{\rho}_s(h) = \operatorname{sech}^s(h/2)$
Lévy density	$\nu_s(\,\mathrm{d}x) = s[2\lvert x\rvert \sinh(\pi\lvert x\rvert)]^{-1}\,\mathrm{d}x$
Even cumulants	$\kappa_{2m} = s(2^{2m}-1)\lvert B_{2m}\rvert/(2m)$
Two-child coefficient	$C^{\alpha,\beta}_{n,r} = \sqrt{n!(\alpha)_r(\beta)_{n-r}/[(\alpha+\beta)_n r!(n-r)!]}$
Many-child coefficient	$C^{\boldsymbol{\alpha}}_{n,\mathbf{r}} = \sqrt{n!\prod_j(\alpha_j)_{r_j}/[(\sum_j \alpha_j)_n \prod_j r_j!]}$
Transfer singular value	$\sigma_n(s,t) = \sqrt{(s)_n/(s+t)_n}$
Coherent state	$\psi_{\alpha,u,\eta}(y) = \mathrm{e}^{\mathrm{i}\eta(y-u/2)} g_\alpha(y-u)$
Berezin position law	$p_\alpha(h) = \Gamma(\alpha+1/2)\operatorname{sech}^{2\alpha}(h/2)/[2\sqrt{\pi}\Gamma(\alpha)]$
Berezin frequency law	$q_\alpha(\eta) = \rho_{2\alpha}(\eta)$
Berezin multiplier	$\widehat{\mathcal{B}}_\alpha(\xi,x) = \lvert\Gamma(\alpha+\mathrm{i}\xi)\rvert^2\Gamma(\alpha)^{-2}\operatorname{sech}^{2\alpha}(x/2)$
Metric coefficients	$g_{\alpha\alpha} = \psi_1(\alpha)/4$, $g_{uu} = \alpha/4$, $g_{\eta\eta} = \psi_1(\alpha)$

Chapter B

Selected numerical tables

The generator writes the cumulant and transfer-spectrum tables as CSV files. A short excerpt is reproduced here.

Shape	κ_2	κ_4	κ_6	κ_8
0.5	0.125	0.0625	0.125	0.53125
1.0	0.25	0.125	0.25	1.0625
2.0	0.5	0.25	0.5	2.125
4.0	1.0	0.5	1.0	4.25

For initial shape $s = 2$, the first transfer singular values are

$$\sigma_n(2,t) = \sqrt{\frac{(2)_n}{(2+t)_n}}.$$

At increment $t = 2$, these begin

$$1, \sqrt{\frac{1}{2}}, \sqrt{\frac{3}{10}}, \sqrt{\frac{1}{5}}, \sqrt{\frac{1}{7}}, \dots.$$

They decay like n^{-1}, so this transfer is compact but not trace class.

Chapter C

Complete Python implementation

The source below generates every figure and every result table in this paper.

```
from __future__ import annotations

import csv
import hashlib
import itertools
import json
import math
from dataclasses import dataclass
from pathlib import Path

import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import FancyArrowPatch, FancyBboxPatch
import mpmath as mp
import numpy as np
from scipy.integrate import quad
from scipy.signal import fftconvolve
from scipy.special import bernoulli, beta as beta_fn, betaln, comb, digamma, gamma, gammaln, loggamma, ndtr,
    polygamma

mp.mp.dps = 80

ROOT = Path(__file__).resolve().parent
FIGURES = ROOT / "figures"
RESULTS = ROOT / "results"
TABLES = ROOT / "tables"
GENERATED = ROOT / "generated_tables"
for directory in (FIGURES, RESULTS, TABLES, GENERATED):
    directory.mkdir(parents=True, exist_ok=True)

INK = "#17202A"
BLUE = "#2E86AB"
TEAL = "#2A9D8F"
ORANGE = "#F4A261"
MAGENTA = "#A23E8C"
RED = "#E76F51"
YELLOW = "#E9C46A"
PURPLE = "#6C5CE7"
MUTED = "#5D6D7E"
PAPER = "#FAFAF7"
GRID = "#D5D8DC"
WHITE = "#FFFFFF"
COLORS = [BLUE, TEAL, ORANGE, MAGENTA, RED, YELLOW, PURPLE]

plt.rcParams.update(
    {
        "figure.facecolor": PAPER,
        "axes.facecolor": PAPER,
        "axes.edgecolor": INK,
        "axes.labelcolor": INK,
        "xtick.color": INK,
        "ytick.color": INK,
        "text.color": INK,
        "font.size": 10.5,
        "axes.titlesize": 12.2,
        "axes.titleweight": "bold",
        "legend.frameon": False,
        "savefig.bbox": "tight",
        "savefig.facecolor": PAPER,
    }
)

```

```

def save_figure(fig: plt.Figure, stem: str) -> None:
    fig.savefig(FIGURES / f"{stem}.pdf")
    fig.savefig(FIGURES / f"{stem}.png", dpi=220)
    plt.close(fig)


def diagram_axes(figsize: tuple[float, float] = (12.6, 6.1)) -> tuple[plt.Figure, plt.Axes]:
    fig, ax = plt.subplots(figsize=figsize)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.axis("off")
    return fig, ax


def box(
    ax: plt.Axes,
    xy: tuple[float, float],
    width: float,
    height: float,
    text: str,
    face: str = WHITE,
    edge: str = INK,
    fontsize: float = 10.5,
) -> None:
    patch = FancyBboxPatch(
        xy,
        width,
        height,
        boxstyle="round,pad=.012,rounding_size=.018",
        facecolor=face,
        edgecolor=edge,
        linewidth=1.6,
    )
    ax.add_patch(patch)
    ax.text(xy[0] + width / 2, xy[1] + height / 2, text, ha="center", va="center", fontsize=fontsize)


def arrow(
    ax: plt.Axes,
    start: tuple[float, float],
    end: tuple[float, float],
    color: str = INK,
    text: str | None = None,
    text_offset: tuple[float, float] = (0.0, 0.035),
) -> None:
    patch = FancyArrowPatch(start, end, arrowstyle="-|>", mutation_scale=13, linewidth=1.5, color=color)
    ax.add_patch(patch)
    if text:
        ax.text(
            (start[0] + end[0]) / 2 + text_offset[0],
            (start[1] + end[1]) / 2 + text_offset[1],
            text,
            ha="center",
            va="center",
            fontsize=9.5,
            color=color,
        )


# ---------------------------------------------------------------------------
# Log-Gamma packets and cross-shape ambiguity
# ---------------------------------------------------------------------------


def log_gamma_window(alpha: float, y: np.ndarray | float) -> np.ndarray:
    values = np.asarray(y, dtype=float)
    return np.exp(0.5 * alpha * values - 0.5 * np.exp(values) - 0.5 * gammaln(alpha))


def cross_gamma_sech(alpha: float, beta: float, h: np.ndarray | float, eta: np.ndarray | float) -> np.ndarray:
```

```
    h_values, eta_values = np.broadcast_arrays(np.asarray(h, dtype=float), np.asarray(eta, dtype=float))
    lam = 0.5 * (alpha + beta)
    z = lam + 1j * eta_values
    return np.exp(
        0.25 * (alpha - beta) * h_values
        + loggamma(z)
        - 0.5 * (gammaln(alpha) + gammaln(beta))
        - z * np.log(np.cosh(h_values / 2.0))
    )


def cross_gamma_numeric(alpha: float, beta: float, h: float, eta: float) -> complex:
    def integrand(y: float) -> complex:
        return (
            log_gamma_window(alpha, y + h / 2.0)
            * log_gamma_window(beta, y - h / 2.0)
            * np.exp(1j * eta * y)
        )

    real = quad(lambda yy: float(np.real(integrand(yy))), -30, 10, epsabs=2e-12, limit=450)[0]
    imag = quad(lambda yy: float(np.imag(integrand(yy))), -30, 10, epsabs=2e-12, limit=450)[0]
    return complex(real, imag)


def shape_affinity(alpha: float, beta: float) -> float:
    return float(np.exp(gammaln(0.5 * (alpha + beta)) - 0.5 * (gammaln(alpha) + gammaln(beta))))


def beta_logit_density(alpha: float, beta: float, h: np.ndarray | float) -> np.ndarray:
    values = np.asarray(h, dtype=float)
    log_density = (
        gammaln(alpha + beta)
        - gammaln(alpha)
        - gammaln(beta)
        - (alpha + beta) * math.log(2.0)
        + 0.5 * (alpha - beta) * values
        - (alpha + beta) * np.log(np.cosh(values / 2.0))
    )
    return np.exp(log_density)


def meixner_density(shape: float, omega: np.ndarray | float) -> np.ndarray:
    values = np.asarray(omega, dtype=float)
    z = shape / 2.0 + 1j * values
    return np.exp((shape - 1.0) * math.log(2.0) - math.log(math.pi) - gammaln(shape) + 2.0 * np.real(loggamma(z)))


def meixner_characteristic(shape: float, h: np.ndarray | float) -> np.ndarray:
    values = np.asarray(h, dtype=float)
    return np.cosh(values / 2.0) ** (-shape)


def cross_spectrogram_density(alpha: float, beta: float, h: np.ndarray, eta: np.ndarray) -> np.ndarray:
    hh, ee = np.meshgrid(h, eta, indexing="ij")
    return np.abs(cross_gamma_sech(alpha, beta, hh, ee)) ** 2 / (2.0 * math.pi)


# ---------------------------------------------------------------------------
# Meixner convolution semigroup and Levy generator
# ---------------------------------------------------------------------------


def levy_density_positive(x: np.ndarray | float) -> np.ndarray:
    values = np.asarray(x, dtype=float)
    return 1.0 / (2.0 * values * np.sinh(math.pi * values))


def levy_symbol_numeric(xi: float) -> float:
    integrand = lambda x: (math.cos(xi * x) - 1.0) / (x * math.sinh(math.pi * x))
    return quad(integrand, 0.0, 30.0, epsabs=2e-12, limit=500)[0]

```

```

def bernoulli_even_cumulant(shape: float, order: int) -> float:
    if order % 2 or order < 2:
        return 0.0
    m = order // 2
    b = float(bernoulli(2 * m)[-1])
    return shape * (2.0 ** (2 * m) - 1.0) * abs(b) / (2.0 * m)


def density_shape_recurrence(shape: float, omega: np.ndarray | float) -> np.ndarray:
    values = np.asarray(omega, dtype=float)
    return ((shape * shape + 4.0 * values * values) / (shape * (shape + 1.0))) * meixner_density(shape, values)


# ---------------------------------------------------------------------------
# Meixner–Pollaczek polynomials and exact packet refinement
# ---------------------------------------------------------------------------


def mp_orthonormal(shape: float, degree: int, x: np.ndarray | float) -> np.ndarray:
    values = np.asarray(x, dtype=float)
    if degree == 0:
        return np.ones_like(values)
    p_prev = np.ones_like(values)
    p_curr = 2.0 * values / math.sqrt(shape)
    if degree == 1:
        return p_curr
    for n in range(1, degree):
        b_np1 = 0.5 * math.sqrt((n + 1) * (n + shape))
        b_n = 0.5 * math.sqrt(n * (n + shape - 1.0))
        p_next = (values * p_curr - b_n * p_prev) / b_np1
        p_prev, p_curr = p_curr, p_next
    return p_curr


def mp_standard(shape: float, degree: int, x: np.ndarray | float) -> np.ndarray:
    norm_factor = math.sqrt(math.exp(gammaln(shape + degree) - gammaln(shape) - gammaln(degree + 1)))
    return norm_factor * mp_orthonormal(shape, degree, x)


def beta_binomial_coefficients(total_degree: int, alpha: float, beta: float) -> np.ndarray:
    r = np.arange(total_degree + 1)
    log_weights = (
        gammaln(total_degree + 1)
        - gammaln(r + 1)
        - gammaln(total_degree - r + 1)
        + gammaln(alpha + r)
        - gammaln(alpha)
        + gammaln(beta + total_degree - r)
        - gammaln(beta)
        - gammaln(alpha + beta + total_degree)
        + gammaln(alpha + beta)
    )
    return np.sqrt(np.exp(log_weights))


def mp_addition_rhs(alpha: float, beta: float, degree: int, x: np.ndarray, y: np.ndarray) -> np.ndarray:
    coefficients = beta_binomial_coefficients(degree, alpha, beta)
    total = np.zeros(np.broadcast(x, y).shape, dtype=float)
    for r, coefficient in enumerate(coefficients):
        total += coefficient * mp_orthonormal(alpha, r, x) * mp_orthonormal(beta, degree - r, y)
    return total


def compositions(total: int, parts: int):
    if parts == 1:
        yield (total,)
        return
    for first in range(total + 1):
        for rest in compositions(total - first, parts - 1):
            yield (first,) + rest
```

```python


def dirichlet_multinomial_coefficient(degrees: tuple[int, ...], shapes: tuple[float, ...]) -> float:
    n = sum(degrees)
    total_shape = sum(shapes)
    log_value = gammaln(n + 1) - sum(gammaln(r + 1) for r in degrees)
    log_value += sum(gammaln(a + r) - gammaln(a) for a, r in zip(shapes, degrees))
    log_value -= gammaln(total_shape + n) - gammaln(total_shape)
    return float(math.exp(0.5 * log_value))


def mp_multinomial_rhs(shapes: tuple[float, ...], degree: int, points: tuple[float, ...]) -> float:
    total = 0.0
    for degrees in compositions(degree, len(shapes)):
        coefficient = dirichlet_multinomial_coefficient(degrees, shapes)
        product = 1.0
        for shape, deg, point in zip(shapes, degrees, points):
            product *= float(mp_orthonormal(shape, deg, point))
        total += coefficient * product
    return total


def transfer_singular_value(shape: float, increment: float, degree: int) -> float:
    return float(
        math.exp(
            0.5
            * (
                gammaln(shape + degree)
                - gammaln(shape)
                - gammaln(shape + increment + degree)
                + gammaln(shape + increment)
            )
        )
    )


def transfer_integral(shape: float, increment: float, degree: int, x: float) -> float:
    integrand = lambda y: (
        float(mp_orthonormal(shape + increment, degree, x + y))
        * float(meixner_density(increment, y))
    )
    return quad(integrand, -25, 25, epsabs=2e-11, limit=500)[0]


# ---------------------------------------------------------------------------
# Coherent states, RKHS, and Berezin transform
# ---------------------------------------------------------------------------


def coherent_state(alpha: float, u: float, eta: float, y: np.ndarray | float) -> np.ndarray:
    values = np.asarray(y, dtype=float)
    return np.exp(1j * eta * (values - u / 2.0)) * log_gamma_window(alpha, values - u)


def coherent_overlap(alpha: float, beta: float, u: float, eta: float, v: float, nu: float) -> complex:
    phase = np.exp(0.5j * (nu * u - eta * v))
    return complex(phase * cross_gamma_sech(alpha, beta, v - u, nu - eta))


def coherent_overlap_numeric(alpha: float, beta: float, u: float, eta: float, v: float, nu: float) -> complex:
    integrand = lambda y: np.conj(coherent_state(alpha, u, eta, y)) * coherent_state(beta, v, nu, y)
    real = quad(lambda yy: float(np.real(integrand(yy))), -35, 12, epsabs=2e-12, limit=500)[0]
    imag = quad(lambda yy: float(np.imag(integrand(yy))), -35, 12, epsabs=2e-12, limit=500)[0]
    return complex(real, imag)


def berezin_position_density(alpha: float, h: np.ndarray | float) -> np.ndarray:
    values = np.asarray(h, dtype=float)
    coefficient = math.exp(gammaln(alpha + 0.5) - gammaln(alpha)) / (2.0 * math.sqrt(math.pi))
    return coefficient * np.cosh(values / 2.0) ** (-2.0 * alpha)

```

```

def berezin_frequency_density(alpha: float, eta: np.ndarray | float) -> np.ndarray:
    return meixner_density(2.0 * alpha, eta)


def berezin_kernel(alpha: float, h: np.ndarray, eta: np.ndarray) -> np.ndarray:
    hh, ee = np.meshgrid(h, eta, indexing="ij")
    return berezin_position_density(alpha, hh) * berezin_frequency_density(alpha, ee)


def berezin_multiplier(alpha: float, xi: np.ndarray, x: np.ndarray) -> np.ndarray:
    xx, yy = np.meshgrid(xi, x, indexing="ij")
    position_char = np.exp(2.0 * np.real(loggamma(alpha + 1j * xx)) - 2.0 * gammaln(alpha))
    frequency_char = np.cosh(yy / 2.0) ** (-2.0 * alpha)
    return position_char * frequency_char


def normal_pdf(x: np.ndarray | float) -> np.ndarray:
    values = np.asarray(x, dtype=float)
    return np.exp(-0.5 * values * values) / math.sqrt(2.0 * math.pi)


def rescaled_position_density(alpha: float, z: np.ndarray | float) -> np.ndarray:
    values = np.asarray(z, dtype=float)
    scale = math.sqrt(2.0 / alpha)
    return scale * berezin_position_density(alpha, scale * values)


def rescaled_frequency_density(alpha: float, z: np.ndarray | float) -> np.ndarray:
    values = np.asarray(z, dtype=float)
    scale = math.sqrt(alpha / 2.0)
    return scale * berezin_frequency_density(alpha, scale * values)


# ---------------------------------------------------------------------------
# Figure generation
# ---------------------------------------------------------------------------


def generate_figures() -> list[str]:
    stems: list[str] = []

    fig, ax = diagram_axes()
    ax.text(0.5, 0.94, "From arithmetic shape to a packet semigroup", ha="center", fontsize=21, weight="bold")
    entries = [
        (0.04, "log-Gamma\npacket", BLUE),
        (0.22, "cross-shape\nambiguity", TEAL),
        (0.40, "Meixner\nconvolution", ORANGE),
        (0.58, "beta-binomial\nrefinement", MAGENTA),
        (0.76, "coherent frame\nand Berezin map", PURPLE),
    ]
    for x0, label, color in entries:
        box(ax, (x0, 0.53), 0.15, 0.17, label, face=color + "20", edge=color, fontsize=11)
    for left, right in zip(entries[:-1], entries[1:]):
        arrow(ax, (left[0] + 0.15, 0.615), (right[0], 0.615), color=INK)
    box(ax, (0.26, 0.18), 0.20, 0.15, "Bernoulli\ncumulants", face=YELLOW + "30", edge=YELLOW, fontsize=12)
    box(ax, (0.54, 0.18), 0.20, 0.15, "Gaussian\nheat limit", face=RED + "20", edge=RED, fontsize=12)
    arrow(ax, (0.475, 0.53), (0.36, 0.33), color=ORANGE)
    arrow(ax, (0.835, 0.53), (0.64, 0.33), color=PURPLE)
    save_figure(fig, "01_theory_pipeline")
    stems.append("01_theory_pipeline")

    y = np.linspace(-6, 4, 900)
    fig, ax = plt.subplots(figsize=(12.5, 5.5))
    for alpha, color in zip([0.8, 1.5, 3.0, 6.0], COLORS):
        ax.plot(y, log_gamma_window(alpha, y), color=color, lw=2, label=rf"$\alpha={alpha}$")
    ax.set_xlabel("log coordinate $y$")
    ax.set_ylabel(r"$g_\alpha(y)$")
    ax.set_title("Log-Gamma packets across the shape coordinate")
    ax.grid(alpha=0.18)
    ax.legend(ncol=4)
```

```
    save_figure(fig, "02_shape_windows")
    stems.append("02_shape_windows")

    h = np.linspace(-5, 5, 220)
    eta = np.linspace(-5, 5, 220)
    hh, ee = np.meshgrid(h, eta, indexing="ij")
    mag = np.abs(cross_gamma_sech(1.0, 4.0, hh, ee))
    fig, ax = plt.subplots(figsize=(10.5, 6.2))
    image = ax.imshow(mag.T, origin="lower", extent=[h.min(), h.max(), eta.min(), eta.max()], aspect="auto", cmap="
     viridis")
    ax.set_xlabel("translation gap $h$")
    ax.set_ylabel(r"frequency gap $\eta$")
    ax.set_title(r"Cross-shape ambiguity $|A_{1,4}(h,\eta)|$")
    fig.colorbar(image, ax=ax, fraction=0.035, pad=0.02, label="magnitude")
    save_figure(fig, "03_cross_ambiguity")
    stems.append("03_cross_ambiguity")

    alphas = np.linspace(0.5, 6.0, 120)
    affinity = np.array([[shape_affinity(a, b) for b in alphas] for a in alphas])
    fig, ax = plt.subplots(figsize=(8.3, 6.6))
    image = ax.imshow(affinity, origin="lower", extent=[alphas.min(), alphas.max(), alphas.min(), alphas.max()],
     cmap="magma", vmin=0, vmax=1)
    ax.set_xlabel(r"$\beta$")
    ax.set_ylabel(r"$\alpha$")
    ax.set_title("Shape fidelity of two log-Gamma packets")
    fig.colorbar(image, ax=ax, fraction=0.045, pad=0.03, label="affinity")
    save_figure(fig, "04_shape_fidelity")
    stems.append("04_shape_fidelity")

    hline = np.linspace(-8, 8, 800)
    fig, ax = plt.subplots(figsize=(12.4, 5.3))
    for alpha, beta, color in [(1, 1, BLUE), (1, 4, TEAL), (2, 5, ORANGE), (5, 2, MAGENTA)]:
        ax.plot(hline, beta_logit_density(alpha, beta, hline), lw=2, color=color, label=rf"$(\alpha,\beta)=({alpha
     },{beta})$")
    ax.set_xlabel("translation gap $h$")
    ax.set_ylabel("density")
    ax.set_title("Beta-logit displacement packets")
    ax.grid(alpha=0.18)
    ax.legend(ncol=2)
    save_figure(fig, "05_beta_logit")
    stems.append("05_beta_logit")

    omega = np.linspace(-7, 7, 900)
    fig, ax = plt.subplots(figsize=(12.4, 5.3))
    for shape, color in zip([1, 2, 4, 8], COLORS):
        ax.plot(omega, meixner_density(shape, omega), lw=2, color=color, label=rf"shape $={shape}$")
    ax.set_xlabel(r"log-frequency $\omega$")
    ax.set_ylabel(r"$\rho_\alpha(\omega)$")
    ax.set_title("Meixner packet densities")
    ax.grid(alpha=0.18)
    ax.legend(ncol=4)
    save_figure(fig, "06_meixner_densities")
    stems.append("06_meixner_densities")

    hgrid = np.linspace(-5, 5, 180)
    egrid = np.linspace(-5, 5, 180)
    cross_density = cross_spectrogram_density(1.5, 3.5, hgrid, egrid)
    factorized = beta_logit_density(1.5, 3.5, hgrid)[:, None] * meixner_density(5.0, egrid)[None, :]
    fig, axes = plt.subplots(1, 3, figsize=(13.0, 4.2))
    for ax, panel, title in zip(
        axes,
        [cross_density, factorized, np.abs(cross_density - factorized)],
        ["direct cross-spectrogram", "Beta-logit x Meixner", "absolute residual"],
    ):
        image = ax.imshow(panel.T, origin="lower", extent=[hgrid.min(), hgrid.max(), egrid.min(), egrid.max()],
     aspect="auto", cmap="viridis")
        ax.set_title(title, fontsize=10.5)
        ax.set_xlabel("$h$")
        ax.set_ylabel(r"$\eta$")
        fig.colorbar(image, ax=ax, fraction=0.045, pad=0.02)
    fig.suptitle("Exact factorization of cross-shape packet energy", fontsize=16, weight="bold")
```

```
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "07_cross_factorization")
    stems.append("07_cross_factorization")

    fig, ax = plt.subplots(figsize=(12.2, 5.2))
    for shape, color in zip([1, 2, 3, 4], COLORS):
        ax.plot(omega, meixner_density(shape, omega), color=color, lw=2, label=rf"$\rho_{shape}$")
    ax.set_xlim(-5, 5)
    ax.set_xlabel(r"$\omega$")
    ax.set_ylabel("density")
    ax.set_title("Shape ladder generated by the Gamma recurrence")
    ax.grid(alpha=0.18)
    ax.legend(ncol=4)
    save_figure(fig, "08_shape_ladder")
    stems.append("08_shape_ladder")

    dx = 0.01
    grid = np.arange(-16, 16 + dx, dx)
    rho_a = meixner_density(1.3, grid)
    rho_b = meixner_density(2.2, grid)
    conv = np.convolve(rho_a, rho_b, mode="same") * dx
    target = meixner_density(3.5, grid)
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8))
    axes[0].plot(grid, conv, color=BLUE, lw=2, label=r"$\rho_{1.3}*\rho_{2.2}$")
    axes[0].plot(grid, target, color=ORANGE, lw=1.4, ls="--", label=r"$\rho_{3.5}$")
    axes[0].set_xlim(-6, 6)
    axes[0].set_title("Convolution semigroup")
    axes[0].legend()
    axes[0].grid(alpha=0.18)
    axes[1].semilogy(grid, np.maximum(np.abs(conv - target), 1e-16), color=RED, lw=1.4)
    axes[1].set_xlim(-6, 6)
    axes[1].set_title("numerical convolution residual")
    axes[1].grid(alpha=0.18)
    for ax in axes:
        ax.set_xlabel(r"$\omega$")
    fig.tight_layout()
    save_figure(fig, "09_convolution_semigroup")
    stems.append("09_convolution_semigroup")

    xpos = np.logspace(-3, 1, 700)
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8))
    axes[0].loglog(xpos, levy_density_positive(xpos), color=PURPLE, lw=2)
    axes[0].set_xlabel("jump size $x$")
    axes[0].set_ylabel(r"$[2x\sinh(\pi x)]^{-1}$")
    axes[0].set_title("Levy density")
    axes[0].grid(alpha=0.18, which="both")
    xis = np.linspace(0, 6, 160)
    numeric_symbol = np.array([levy_symbol_numeric(float(x)) for x in xis])
    exact_symbol = -np.log(np.cosh(xis / 2.0))
    axes[1].plot(xis, numeric_symbol, color=BLUE, lw=2, label="Levy integral")
    axes[1].plot(xis, exact_symbol, color=ORANGE, lw=1.4, ls="--", label=r"$-\log\cosh(\xi/2)$")
    axes[1].set_xlabel(r"Fourier variable $\xi$")
    axes[1].set_title("Generator symbol")
    axes[1].grid(alpha=0.18)
    axes[1].legend()
    fig.tight_layout()
    save_figure(fig, "10_levy_generator")
    stems.append("10_levy_generator")

    orders = np.arange(2, 14, 2)
    shapes = [0.5, 1.0, 2.0, 4.0]
    fig, ax = plt.subplots(figsize=(12.2, 5.4))
    for shape, color in zip(shapes, COLORS):
        values = [bernoulli_even_cumulant(shape, int(order)) for order in orders]
        ax.plot(orders, values, marker="o", lw=2, color=color, label=rf"shape $={shape}$")
    ax.set_yscale("log")
    ax.set_xlabel("cumulant order")
    ax.set_ylabel("even cumulant")
    ax.set_title("Bernoulli-number cumulants of the Meixner packet semigroup")
    ax.grid(alpha=0.18, which="both")
    ax.legend(ncol=4)
```

```
    save_figure(fig, "11_bernoulli_cumulants")
    stems.append("11_bernoulli_cumulants")

    fig, ax = diagram_axes()
    ax.text(0.5, 0.93, "Exact polynomial packet splitter", ha="center", fontsize=21, weight="bold")
    box(ax, (0.08, 0.57), 0.20, 0.16, r"$p_{n,\alpha+\beta}(x+y)$", face=BLUE + "20", edge=BLUE, fontsize=14)
    for idx, r in enumerate(range(5)):
        y0 = 0.78 - 0.145 * idx
        box(ax, (0.43, y0), 0.18, 0.09, rf"$p_{{{r},\alpha}}(x)$", face=TEAL + "18", edge=TEAL, fontsize=10.5)
        box(ax, (0.72, y0), 0.18, 0.09, rf"$p_{{n-{r},\beta}}(y)$", face=ORANGE + "18", edge=ORANGE, fontsize=10.5)
        arrow(ax, (0.28, 0.65), (0.43, y0 + 0.045), color=INK, text=rf"$C_{{n,{r}}}$")
        arrow(ax, (0.61, y0 + 0.045), (0.72, y0 + 0.045), color=INK)
    ax.text(0.5, 0.08, r"$|C_{n,r}|^2$ is a Beta-binomial law and $\sum_r|C_{n,r}|^2=1$", ha="center", fontsize=12,
     color=MUTED)
    save_figure(fig, "12_packet_splitter")
    stems.append("12_packet_splitter")

    n = 12
    fig, ax = plt.subplots(figsize=(11.8, 5.0))
    for alpha, beta, color in [(1, 1, BLUE), (1, 4, TEAL), (2, 5, ORANGE), (5, 2, MAGENTA)]:
        coeffs = beta_binomial_coefficients(n, alpha, beta) ** 2
        ax.plot(np.arange(n + 1), coeffs, marker="o", lw=2, color=color, label=rf"$(\alpha,\beta)=({alpha},{beta})$"
     )
    ax.set_xlabel("child degree $r$")
    ax.set_ylabel(r"$|C_{n,r}^{\alpha,\beta}|^2$")
    ax.set_title("Beta-binomial energy partition")
    ax.grid(alpha=0.18)
    ax.legend(ncol=2)
    save_figure(fig, "13_beta_binomial_energy")
    stems.append("13_beta_binomial_energy")

    xvals = np.linspace(-4, 4, 500)
    y0 = 0.85
    degree = 4
    lhs = mp_orthonormal(5.0, degree, xvals + y0)
    rhs = mp_addition_rhs(2.0, 3.0, degree, xvals, np.full_like(xvals, y0))
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8))
    axes[0].plot(xvals, lhs, color=BLUE, lw=2, label="parent packet")
    axes[0].plot(xvals, rhs, color=ORANGE, lw=1.3, ls="--", label="sum of child products")
    axes[0].set_title("Degree-four addition theorem")
    axes[0].legend()
    axes[0].grid(alpha=0.18)
    axes[1].plot(xvals, lhs - rhs, color=RED, lw=1.4)
    axes[1].set_title("pointwise residual")
    axes[1].grid(alpha=0.18)
    for ax in axes:
        ax.set_xlabel("$x$")
    fig.tight_layout()
    save_figure(fig, "14_addition_example")
    stems.append("14_addition_example")

    shapes3 = (1.0, 2.0, 3.0)
    degree3 = 8
    comps = list(compositions(degree3, 3))
    weights = np.array([dirichlet_multinomial_coefficient(c, shapes3) ** 2 for c in comps])
    coords = np.array([[c[0], c[1], c[2]] for c in comps])
    fig, ax = plt.subplots(figsize=(8.0, 6.3))
    scatter = ax.scatter(coords[:, 0] + 0.5 * coords[:, 1], np.sqrt(3) * coords[:, 1] / 2, c=weights, s=1600 *
     weights + 30, cmap="viridis", edgecolor=INK, linewidth=0.4)
    ax.set_aspect("equal")
    ax.set_xlabel(r"$r_1+r_2/2$")
    ax.set_ylabel(r"$\sqrt{3},r_2/2$")
    ax.set_title("Dirichlet-multinomial refinement on a three-child simplex")
    fig.colorbar(scatter, ax=ax, fraction=0.045, pad=0.03, label="squared coefficient")
    save_figure(fig, "15_multinomial_simplex")
    stems.append("15_multinomial_simplex")

    fig, ax = diagram_axes()
    ax.text(0.5, 0.93, "Coassociative packet refinement", ha="center", fontsize=21, weight="bold")
    box(ax, (0.40, 0.72), 0.20, 0.12, r"shape $\alpha+\beta+\gamma$", face=BLUE + "20", edge=BLUE, fontsize=12)
    box(ax, (0.14, 0.45), 0.22, 0.12, r"$(\alpha+\beta)+\gamma$", face=TEAL + "20", edge=TEAL, fontsize=12)
```

```
    box(ax, (0.64, 0.45), 0.22, 0.12, r"$\alpha+(\beta+\gamma)$", face=ORANGE + "20", edge=ORANGE, fontsize=12)
    box(ax, (0.36, 0.16), 0.28, 0.13, "same three-child coefficient\n" + r"$\sqrt{\frac{n!\prod(\alpha_j)_{r_j}}{(\sum\alpha_j)_n\prod r_j!}}$", face=MAGENTA + "18", edge=MAGENTA, fontsize=10.5)
    arrow(ax, (0.46, 0.72), (0.25, 0.57), color=TEAL)
    arrow(ax, (0.54, 0.72), (0.75, 0.57), color=ORANGE)
    arrow(ax, (0.25, 0.45), (0.43, 0.29), color=TEAL)
    arrow(ax, (0.75, 0.45), (0.57, 0.29), color=ORANGE)
    save_figure(fig, "16_coassociativity")
    stems.append("16_coassociativity")

    degrees = np.arange(0, 18)
    fig, ax = plt.subplots(figsize=(12.2, 5.2))
    for increment, color in zip([0.5, 1.0, 2.0, 5.0], COLORS):
        singular = [transfer_singular_value(2.0, increment, int(d)) for d in degrees]
        ax.plot(degrees, singular, marker="o", lw=2, color=color, label=rf"increment $={increment}$")
    ax.set_xlabel("polynomial packet degree")
    ax.set_ylabel("transfer singular value")
    ax.set_title("Exact singular spectrum of Meixner shape transfer")
    ax.grid(alpha=0.18)
    ax.legend(ncol=2)
    save_figure(fig, "17_transfer_singular_values")
    stems.append("17_transfer_singular_values")

    shape_grid = np.linspace(0.5, 8.0, 90)
    degree_grid = np.arange(0, 24)
    spectrum = np.array([[transfer_singular_value(float(s), 2.0, int(nv)) for nv in degree_grid] for s in shape_grid])
    fig, ax = plt.subplots(figsize=(10.2, 6.0))
    image = ax.imshow(spectrum, origin="lower", aspect="auto", extent=[degree_grid.min(), degree_grid.max(), shape_grid.min(), shape_grid.max()], cmap="magma", vmin=0, vmax=1)
    ax.set_xlabel("degree $n$")
    ax.set_ylabel(r"starting shape $\alpha$")
    ax.set_title("Shape-transfer contraction spectrum for increment two")
    fig.colorbar(image, ax=ax, fraction=0.038, pad=0.02, label="singular value")
    save_figure(fig, "18_transfer_spectrum")
    stems.append("18_transfer_spectrum")

    fig, ax = plt.subplots(figsize=(11.8, 5.2))
    for degree, color in zip([1, 2, 3, 5], COLORS):
        svals = [transfer_singular_value(1.0, float(t), degree) for t in np.linspace(0, 12, 300)]
        ax.plot(np.linspace(0, 12, 300), svals, lw=2, color=color, label=rf"degree $n={degree}$")
    ax.set_xlabel("shape increment $t$")
    ax.set_ylabel("conditional-transfer amplitude")
    ax.set_title("Polynomial martingale transfer")
    ax.grid(alpha=0.18)
    ax.legend(ncol=4)
    save_figure(fig, "19_martingale_transfer")
    stems.append("19_martingale_transfer")

    ugrid = np.linspace(-3.5, 3.5, 9)
    egrid2 = np.linspace(-3.5, 3.5, 9)
    fig, ax = plt.subplots(figsize=(8.0, 6.5))
    for u in ugrid:
        for eta0 in egrid2:
            magnitude = abs(coherent_overlap(2.0, 2.0, 0.0, 0.0, float(u), float(eta0)))
            ax.scatter(u, eta0, s=340 * magnitude + 8, color=BLUE, alpha=0.65, edgecolor=INK, linewidth=0.25)
    ax.set_xlabel(r"translation $u$")
    ax.set_ylabel(r"modulation $\eta$")
    ax.set_title("Log-Gamma coherent-state lattice")
    ax.grid(alpha=0.15)
    save_figure(fig, "20_coherent_lattice")
    stems.append("20_coherent_lattice")

    kernel_mag = np.abs(cross_gamma_sech(2.0, 2.0, hh, ee))
    fig, ax = plt.subplots(figsize=(9.5, 6.2))
    image = ax.imshow(kernel_mag.T, origin="lower", extent=[h.min(), h.max(), eta.min(), eta.max()], aspect="auto", cmap="plasma")
    ax.set_xlabel("translation difference")
    ax.set_ylabel("modulation difference")
    ax.set_title("Reproducing-kernel magnitude")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.025, label="overlap magnitude")
```

```
    save_figure(fig, "21_rkhs_kernel")
    stems.append("21_rkhs_kernel")

    bkernel = berezin_kernel(2.0, hgrid, egrid)
    fig, axes = plt.subplots(1, 3, figsize=(13.0, 4.2))
    axes[0].plot(hgrid, berezin_position_density(2.0, hgrid), color=TEAL, lw=2)
    axes[0].set_title("position marginal")
    axes[0].set_xlabel("$h$")
    axes[0].grid(alpha=0.18)
    axes[1].plot(egrid, berezin_frequency_density(2.0, egrid), color=ORANGE, lw=2)
    axes[1].set_title("frequency marginal")
    axes[1].set_xlabel(r"$\eta$")
    axes[1].grid(alpha=0.18)
    image = axes[2].imshow(bkernel.T, origin="lower", extent=[hgrid.min(), hgrid.max(), egrid.min(), egrid.max()],
     aspect="auto", cmap="viridis")
    axes[2].set_title("product kernel")
    axes[2].set_xlabel("$h$")
    axes[2].set_ylabel(r"$\eta$")
    fig.colorbar(image, ax=axes[2], fraction=0.045, pad=0.02)
    fig.suptitle("Exact Berezin-kernel factorization", fontsize=16, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "22_berezin_factorization")
    stems.append("22_berezin_factorization")

    xi = np.linspace(-5, 5, 180)
    xx = np.linspace(-5, 5, 180)
    multiplier = berezin_multiplier(2.0, xi, xx)
    fig, ax = plt.subplots(figsize=(9.5, 6.2))
    image = ax.imshow(multiplier.T, origin="lower", extent=[xi.min(), xi.max(), xx.min(), xx.max()], aspect="auto",
     cmap="magma", norm=matplotlib.colors.LogNorm(vmin=max(multiplier.min(), 1e-8), vmax=1))
    ax.set_xlabel(r"dual translation variable $\xi$")
    ax.set_ylabel("dual modulation variable $x$")
    ax.set_title("Strictly positive Berezin multiplier")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.025, label="multiplier")
    save_figure(fig, "23_berezin_multiplier")
    stems.append("23_berezin_multiplier")

    z = np.linspace(-4.5, 4.5, 700)
    fig, axes = plt.subplots(1, 2, figsize=(12.8, 4.8), sharey=True)
    for alpha, color in zip([2, 5, 20, 80], COLORS):
        axes[0].plot(z, rescaled_position_density(alpha, z), color=color, lw=1.8, label=rf"$\alpha={alpha}$")
        axes[1].plot(z, rescaled_frequency_density(alpha, z), color=color, lw=1.8, label=rf"$\alpha={alpha}$")
    for ax, title in zip(axes, ["rescaled position marginal", "rescaled frequency marginal"]):
        ax.plot(z, normal_pdf(z), color=INK, lw=2, ls="--", label="standard normal")
        ax.set_title(title)
        ax.set_xlabel("standardized coordinate")
        ax.grid(alpha=0.18)
    axes[0].set_ylabel("density")
    axes[1].legend(ncol=2, fontsize=8.5)
    fig.tight_layout()
    save_figure(fig, "24_gaussian_marginals")
    stems.append("24_gaussian_marginals")

    z2 = np.linspace(-3.5, 3.5, 180)
    pos = rescaled_position_density(20.0, z2)
    freq = rescaled_frequency_density(20.0, z2)
    joint = pos[:, None] * freq[None, :]
    gaussian = normal_pdf(z2)[:, None] * normal_pdf(z2)[None, :]
    fig, axes = plt.subplots(1, 3, figsize=(13.0, 4.2))
    for ax, panel, title in zip(axes, [joint, gaussian, np.abs(joint - gaussian)], [r"rescaled $B_{20}$", "Gaussian
     heat kernel", "absolute residual"]):
        image = ax.imshow(panel.T, origin="lower", extent=[z2.min(), z2.max(), z2.min(), z2.max()], aspect="equal",
     cmap="viridis")
        ax.set_title(title, fontsize=10.5)
        ax.set_xlabel("scaled position")
        ax.set_ylabel("scaled frequency")
        fig.colorbar(image, ax=ax, fraction=0.045, pad=0.02)
    fig.suptitle("Gaussian fixed point of Berezin diffusion", fontsize=16, weight="bold")
    fig.tight_layout(rect=[0, 0, 1, 0.91])
    save_figure(fig, "25_gaussian_heat_limit")
    stems.append("25_gaussian_heat_limit")
```

```
    fig, ax = diagram_axes()
    ax.text(0.5, 0.94, "Exact transfer hierarchy", ha="center", fontsize=21, weight="bold")
    box(ax, (0.05, 0.58), 0.18, 0.16, "arithmetic depth\n" + r"parameter $\alpha$", face=BLUE + "20", edge=BLUE, fontsize=12)
    box(ax, (0.30, 0.58), 0.18, 0.16, "Meixner Levy\nsemigroup", face=TEAL + "20", edge=TEAL, fontsize=12)
    box(ax, (0.55, 0.58), 0.18, 0.16, "polynomial packet\nrefinement", face=ORANGE + "20", edge=ORANGE, fontsize=12)
    box(ax, (0.78, 0.58), 0.17, 0.16, "coherent-state\nquantization", face=MAGENTA + "20", edge=MAGENTA, fontsize=12)
    for x0 in [0.23, 0.48, 0.73]:
        arrow(ax, (x0, 0.66), (x0 + 0.07, 0.66), color=INK)
    box(ax, (0.24, 0.20), 0.22, 0.15, "Bernoulli cumulants\nand Levy generator", face=YELLOW + "30", edge=YELLOW, fontsize=11)
    box(ax, (0.55, 0.20), 0.22, 0.15, "Berezin kernel\nand Gaussian limit", face=PURPLE + "18", edge=PURPLE, fontsize=11)
    arrow(ax, (0.39, 0.58), (0.35, 0.35), color=TEAL)
    arrow(ax, (0.865, 0.58), (0.66, 0.35), color=MAGENTA)
    save_figure(fig, "26_transfer_hierarchy")
    stems.append("26_transfer_hierarchy")

    alpha_values = np.linspace(0.4, 8.0, 300)
    metric_shape = 0.25 * polygamma(1, alpha_values)
    metric_translation = alpha_values / 4.0
    metric_modulation = polygamma(1, alpha_values)
    fig, ax = plt.subplots(figsize=(12.2, 5.2))
    ax.plot(alpha_values, metric_shape, color=BLUE, lw=2, label=r"shape $g_{\alpha\alpha}$")
    ax.plot(alpha_values, metric_translation, color=TEAL, lw=2, label=r"translation $g_{uu}$")
    ax.plot(alpha_values, metric_modulation, color=ORANGE, lw=2, label=r"modulation $g_{\eta\eta}$")
    ax.set_xlabel(r"shape $\alpha$")
    ax.set_ylabel("metric coefficient")
    ax.set_title("Three local scales of the generated packet geometry")
    ax.set_yscale("log")
    ax.grid(alpha=0.18, which="both")
    ax.legend(ncol=3)
    save_figure(fig, "27_shape_geometry")
    stems.append("27_shape_geometry")

    fig, ax = diagram_axes()
    ax.text(0.5, 0.94, "Function-family map", ha="center", fontsize=21, weight="bold")
    nodes = [
        (0.06, 0.63, "cross-Gamma--sech\nambiguity", BLUE),
        (0.30, 0.63, "Meixner\nLevy semigroup", TEAL),
        (0.54, 0.63, "Meixner--Pollaczek\npacket coproduct", ORANGE),
        (0.78, 0.63, "Gamma--sech\nBerezin kernel", MAGENTA),
        (0.18, 0.25, "Beta-logit\ndisplacement law", PURPLE),
        (0.43, 0.25, "Bernoulli\ncumulants", YELLOW),
        (0.68, 0.25, "Gaussian\nheat fixed point", RED),
    ]
    for x0, y0, label, color in nodes:
        box(ax, (x0, y0), 0.17, 0.14, label, face=color + "20", edge=color, fontsize=10.5)
    for start, end in [((0.23, 0.70), (0.30, 0.70)), ((0.47, 0.70), (0.54, 0.70)), ((0.71, 0.70), (0.78, 0.70)),
                       ((0.145, 0.63), (0.265, 0.39)), ((0.385, 0.63), (0.515, 0.39)), ((0.865, 0.63), (0.765, 0.39))]:
        arrow(ax, start, end, color=INK)
    save_figure(fig, "28_function_family_map")
    stems.append("28_function_family_map")

    return stems


# ---------------------------------------------------------------------------
# Verification and tables
# ---------------------------------------------------------------------------


@dataclass
class Check:
    name: str
    passed: bool
    residual: float
    note: str

```

```

def run_checks() -> tuple[list[Check], dict[str, float | int | str]]:
    checks: list[Check] = []

    def add(name: str, residual: float, tolerance: float, note: str) -> None:
        checks.append(Check(name, bool(residual <= tolerance), float(residual), note))

    add("log-Gamma normalization", abs(quad(lambda y: float(log_gamma_window(2.3, y) ** 2), -30, 10)[0] - 1.0), 2e
    -11, "L2 norm")

    cross_exact = complex(cross_gamma_sech(1.4, 3.1, 0.75, -0.8))
    cross_num = cross_gamma_numeric(1.4, 3.1, 0.75, -0.8)
    add("cross-shape ambiguity", abs(cross_exact - cross_num), 3e-10, "closed form against quadrature")

    add("shape affinity strictness", max(0.0, shape_affinity(1.2, 4.7) - 1.0), 1e-14, "Cauchy and Gamma log-
    convexity")

    beta_integral = quad(lambda h: float(beta_logit_density(1.3, 2.7, h)), -30, 30, epsabs=2e-12)[0]
    add("Beta-logit normalization", abs(beta_integral - 1.0), 3e-10, "displacement marginal")

    meixner_integral = quad(lambda w: float(meixner_density(3.4, w)), -30, 30, epsabs=2e-12)[0]
    add("Meixner normalization", abs(meixner_integral - 1.0), 3e-10, "spectral marginal")

    htest = np.linspace(-3, 3, 51)
    etest = np.linspace(-3, 3, 51)
    direct = cross_spectrogram_density(1.3, 2.7, htest, etest)
    factor = beta_logit_density(1.3, 2.7, htest)[:, None] * meixner_density(4.0, etest)[None, :]
    add("cross-spectrogram factorization", np.max(np.abs(direct - factor)), 2e-13, "Beta-logit times Meixner")

    conv_grid = np.linspace(-18, 18, 7201)
    dx = conv_grid[1] - conv_grid[0]
    conv = fftconvolve(meixner_density(1.2, conv_grid), meixner_density(2.3, conv_grid), mode="same") * dx
    target = meixner_density(3.5, conv_grid)
    mask = np.abs(conv_grid) < 7
    add("Meixner convolution semigroup", np.max(np.abs(conv[mask] - target[mask])), 3e-4, "FFT-free grid convolution
    ")

    recurrence_residual = np.max(np.abs(meixner_density(3.2, etest) - density_shape_recurrence(1.2, etest)))
    add("shape recurrence", recurrence_residual, 2e-13, "shape-plus-two ladder")

    levy_points = [0.4, 1.1, 2.3]
    levy_residual = max(abs(levy_symbol_numeric(x) + math.log(math.cosh(x / 2.0))) for x in levy_points)
    add("Levy-Khintchine symbol", levy_residual, 3e-11, "integral against 1/(x sinh pi x)")

    add("Bernoulli variance", abs(bernoulli_even_cumulant(2.7, 2) - 2.7 / 4.0), 2e-14, "second cumulant")
    add("Bernoulli fourth cumulant", abs(bernoulli_even_cumulant(2.7, 4) - 2.7 / 8.0), 1e-11, "fourth cumulant")

    coeffs = beta_binomial_coefficients(12, 1.7, 3.4)
    add("Beta-binomial energy partition", abs(float(np.sum(coeffs * coeffs)) - 1.0), 2e-13, "sum of squared packet
    coefficients")

    x = np.linspace(-2.5, 2.5, 41)
    y = np.full_like(x, 0.73)
    lhs = mp_orthonormal(5.1, 6, x + y)
    rhs = mp_addition_rhs(1.7, 3.4, 6, x, y)
    add("normalized addition theorem", np.max(np.abs(lhs - rhs)), 3e-12, "two-child packet reconstruction")

    shapes = (0.9, 1.6, 2.4)
    points = (0.3, -0.8, 1.1)
    lhs_multi = float(mp_orthonormal(sum(shapes), 5, sum(points)))
    rhs_multi = mp_multinomial_rhs(shapes, 5, points)
    add("multinomial packet theorem", abs(lhs_multi - rhs_multi), 5e-12, "three-child reconstruction")

    coeff_sum = sum(dirichlet_multinomial_coefficient(c, shapes) ** 2 for c in compositions(7, 3))
    add("Dirichlet-multinomial energy", abs(coeff_sum - 1.0), 3e-13, "coassociative coefficient norm")

    transfer_points = [-1.3, 0.0, 1.8]
    residuals = []
    for degree in [0, 1, 2, 3]:
        sigma = transfer_singular_value(1.8, 1.1, degree)
        for xv in transfer_points:
```

```
            integral = transfer_integral(1.8, 1.1, degree, xv)
            target_value = sigma * float(mp_orthonormal(1.8, degree, xv))
            residuals.append(abs(integral - target_value))
    add("conditional transfer spectrum", max(residuals), 5e-8, "Meixner increment integral")

    cocycle = abs(
        transfer_singular_value(1.3, 0.8, 5)
        * transfer_singular_value(2.1, 1.4, 5)
        - transfer_singular_value(1.3, 2.2, 5)
    )
    add("transfer cocycle", cocycle, 3e-14, "singular values telescope")

    overlap_closed = coherent_overlap(1.4, 2.6, -0.3, 0.7, 0.8, -0.4)
    overlap_num = coherent_overlap_numeric(1.4, 2.6, -0.3, 0.7, 0.8, -0.4)
    add("coherent overlap", abs(overlap_closed - overlap_num), 4e-10, "Weyl phase and cross-shape kernel")

    pos_norm = quad(lambda h: float(berezin_position_density(2.4, h)), -30, 30, epsabs=2e-12)[0]
    freq_norm = quad(lambda w: float(berezin_frequency_density(2.4, w)), -30, 30, epsabs=2e-12)[0]
    add("Berezin position normalization", abs(pos_norm - 1.0), 3e-10, "position factor")
    add("Berezin frequency normalization", abs(freq_norm - 1.0), 3e-10, "frequency factor")

    position_variance = quad(lambda h: float(h * h * berezin_position_density(2.4, h)), -30, 30, epsabs=2e-11)[0]
    frequency_variance = quad(lambda w: float(w * w * berezin_frequency_density(2.4, w)), -30, 30, epsabs=2e-11)[0]
    add("Berezin position variance", abs(position_variance - 2.0 * float(polygamma(1, 2.4))), 3e-9, "twice the
     trigamma")
    add("Berezin frequency variance", abs(frequency_variance - 2.4 / 2.0), 3e-9, "shape divided by two")

    z = np.linspace(-7, 7, 5001)
    gaussian_cdf = ndtr(z)
    pos_density = rescaled_position_density(80.0, z)
    freq_density = rescaled_frequency_density(80.0, z)
    dz = z[1] - z[0]
    pos_cdf = np.cumsum(pos_density) * dz
    freq_cdf = np.cumsum(freq_density) * dz
    pos_cdf /= pos_cdf[-1]
    freq_cdf /= freq_cdf[-1]
    add("Gaussian position limit", np.max(np.abs(pos_cdf - gaussian_cdf)), 0.02, "Kolmogorov distance at alpha=80")
    add("Gaussian frequency limit", np.max(np.abs(freq_cdf - gaussian_cdf)), 0.02, "Kolmogorov distance at alpha=80"
     )

    passed = sum(check.passed for check in checks)
    results = {
        "checks_passed": passed,
        "checks_total": len(checks),
        "max_residual": max(check.residual for check in checks),
        "cross_shape_example_real": float(np.real(cross_exact)),
        "cross_shape_example_imag": float(np.imag(cross_exact)),
        "shape_affinity_1_4": shape_affinity(1.0, 4.0),
        "levy_symbol_at_1": levy_symbol_numeric(1.0),
        "beta_binomial_norm": float(np.sum(coeffs * coeffs)),
        "transfer_degree_5": transfer_singular_value(2.0, 2.0, 5),
        "berezin_position_variance_alpha_2_4": position_variance,
        "berezin_frequency_variance_alpha_2_4": frequency_variance,
        "figure_pairs": 28,
    }
    return checks, results


def write_outputs(stems: list[str], checks: list[Check], results: dict[str, float | int | str]) -> None:
    with (RESULTS / "checks.csv").open("w", newline="", encoding="utf-8") as handle:
        writer = csv.DictWriter(handle, fieldnames=["name", "passed", "residual", "note"])
        writer.writeheader()
        for check in checks:
            writer.writerow({"name": check.name, "passed": check.passed, "residual": f"{check.residual:.16e}", "note
     ": check.note})

    (RESULTS / "results.json").write_text(json.dumps(results, indent=2, sort_keys=True) + "\n", encoding="utf-8")

    with (TABLES / "cumulants.csv").open("w", newline="", encoding="utf-8") as handle:
        writer = csv.writer(handle)
        writer.writerow(["shape", "kappa2", "kappa4", "kappa6", "kappa8"])
```

```
        for shape in [0.5, 1.0, 2.0, 4.0]:
            writer.writerow([shape] + [bernoulli_even_cumulant(shape, order) for order in [2, 4, 6, 8]])

    with (TABLES / "transfer_spectrum.csv").open("w", newline="", encoding="utf-8") as handle:
        writer = csv.writer(handle)
        writer.writerow(["degree", "increment_0_5", "increment_1", "increment_2", "increment_5"])
        for degree in range(0, 13):
            writer.writerow([degree] + [transfer_singular_value(2.0, increment, degree) for increment in [0.5, 1.0,
 2.0, 5.0]])

    def latex_escape(text: str) -> str:
        replacements = {
            "\\": r"\textbackslash{}",
            "&": r"\&",
            "%": r"\%",
            "$": r"\$",
            "#": r"\#",
            "_": r"\_",
            "{": r"\{",
            "}": r"\}",
        }
        return "".join(replacements.get(char, char) for char in text)

    row_break = r"\\"
    check_rows = []
    for check in checks:
        status = "pass" if check.passed else "FAIL"
        check_rows.append(
            f"{latex_escape(check.name)} & {status} & {check.residual:.3e} & {latex_escape(check.note)} {row_break}"
        )
    checks_tex = (
        "\\begin{longtable}{L{0.31\\textwidth}L{0.08\\textwidth}L{0.13\\textwidth}L{0.37\\textwidth}}\n"
        "\\toprule\n"
        f"Check & Status & Residual & Note {row_break}\n"
        "\\midrule\n"
        "\\endfirsthead\n"
        "\\toprule\n"
        f"Check & Status & Residual & Note {row_break}\n"
        "\\midrule\n"
        "\\endhead\n"
        + "\n".join(check_rows)
        + "\n\\bottomrule\n\\end{longtable}\n"
    )
    (GENERATED / "checks.tex").write_text(checks_tex, encoding="utf-8")

    (GENERATED / "headline.tex").write_text(
        "\\begin{center}\n"
        "\\fbox{\\begin{minipage}{0.83\\textwidth}\n"
        f"\\textbf{{Built-in verification.}} {results['checks_passed']} of {results['checks_total']} checks passed.
 "
        f"The release contains {results['figure_pairs']} vector-PDF and PNG figure pairs.\n"
        "\\end{minipage}}\n"
        "\\end{center}\n",
        encoding="utf-8",
    )

    figure_lines = [f"{index}. `{stem}.pdf` and `{stem}.png`" for index, stem in enumerate(stems, start=1)]
    (ROOT / "Projector_Generated_Special_Functions_VI_FIGURE_INDEX.md").write_text(
        "# Figure index\n\n" + "\n".join(figure_lines) + "\n",
        encoding="utf-8",
    )

    manifest = {
        "title": "Projector-Generated Special Functions VI",
        "author": "Komorebi P.",
        "figure_pairs": len(stems),
        "checks_passed": results["checks_passed"],
        "checks_total": results["checks_total"],
        "python_generator": "projector_generated_special_functions_vi.py",
    }
    (ROOT / "Projector_Generated_Special_Functions_VI_MANIFEST.json").write_text(
```

```
        json.dumps(manifest, indent=2, sort_keys=True) + "\n",
        encoding="utf-8",
    )


def main() -> None:
    stems = generate_figures()
    checks, results = run_checks()
    write_outputs(stems, checks, results)
    print(json.dumps(results, indent=2, sort_keys=True))
    if results["checks_passed"] != results["checks_total"]:
        raise SystemExit("one or more built-in checks failed")


if __name__ == "__main__":
    main()
```

Chapter D

Reproduction

From the source directory, run

```
python projector_generated_special_functions_vi.py
pytest -q
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_VI.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_VI.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_VI.tex
```

Chapter E

Bibliographic note

Bibliography

[1] NIST Digital Library of Mathematical Functions. Chapter 5, Gamma Function. `https://dlmf.nist.gov/5`.

[2] NIST Digital Library of Mathematical Functions. Sections 18.19 and 18.23, Meixner–Pollaczek polynomials and generating functions. `https://dlmf.nist.gov/18.23`.

[3] R. Koekoek, P. A. Lesky, and R. F. Swarttouw. *Hypergeometric Orthogonal Polynomials and Their q-Analogues*. Springer, 2010.

[4] W. Schoutens. *Stochastic Processes and Orthogonal Polynomials*. Springer, 2000.

[5] K. Sato. *Lévy Processes and Infinitely Divisible Distributions*. Cambridge University Press, 1999.

[6] G. B. Folland. *Harmonic Analysis in Phase Space*. Princeton University Press, 1989.

[7] F. A. Berezin. General concept of quantization. *Communications in Mathematical Physics*, 40:153–174, 1975.

[8] A. Perelomov. *Generalized Coherent States and Their Applications*. Springer, 1986.

[9] S. Wei, Y. Zhai, Q. Lu, W. Yang, P. Gao, C. Wei, J. Song, F. Nori, T. Xin, and G. Long. The Riemann Hypothesis manifested in dynamical quantum phase transitions. *Nature Communications*, article in press, 2026. DOI: 10.1038/s41467-026-74935-8.

[10] Komorebi P. *Projector-Generated Special Functions V: Log-Gamma Gabor Packets, Gamma–Sech Ambiguity Kernels, Meixner–Pollaczek Spectra, and* $\mathrm{SU}(1,1)$ *Duality*. Preceding paper in this series.

Projector-Generated Special Functions VII

SU(1, 1) Racah Packet Recoupling, Casimir Multiplicity Spaces, Pentagon Coherence, and Bistochastic Quantum Kernels

Komorebi P.

Abstract

The preceding paper produced a continuous shape semigroup, Meixner–Pollaczek packet bases, beta-binomial refinement, and a common positive discrete-series representation of $\mathfrak{su}(1,1)$. The present paper studies what happens when three or more shape packets are coupled. The result is a finite family of orthogonal recoupling functions generated by pair-Casimir projectors.

For positive shapes $\kappa_1, \kappa_2, \kappa_3$, let $D^+_{\kappa_j}$ denote the positive discrete-series representation. At fixed total excitation N, the tensor product contains an $(N+1)$-dimensional lowest-weight multiplicity space. Diagonalizing the intermediate Casimir $\mathcal{C}_{12}$ gives the left coupling atlas, while diagonalizing $\mathcal{C}_{23}$ gives the right coupling atlas. Their overlap is the Racah packet matrix

$$\boxed{\mathcal{R}^{\kappa}_N(j,m) = \langle j, (12)3, m, 1(23)\rangle\,.}$$

It is real orthogonal. In the $\mathcal{C}_{12}$ basis, the operator $\mathcal{C}_{23}$ is tridiagonal, so every column obeys a finite three-term recurrence. This is the packet-calculus realization of the classical Racah-polynomial and $6j$ structure.

Two exact low-degree transforms are derived. At $N = 1$, the recoupling matrix is a shape-controlled beam splitter. At equal shape $\kappa_1 = \kappa_2 = \kappa_3 = 1/2$ and $N = 2$, the complete 3×3 matrix is expressed through rational numbers and square roots. Squaring the matrix entries produces a doubly stochastic kernel. It acts as the transition matrix obtained by measuring one coupling observable after dephasing in the other coupling basis. The resulting channel preserves the uniform law and cannot decrease Shannon entropy.

For four packets, the fixed-excitation lowest-weight space has dimension $\binom{N+2}{2}$. Each binary coupling tree determines a complete joint eigenbasis of commuting intermediate Casimirs. The five trees of the associahedron are linked by orthogonal packet transforms. Their ordered product around the pentagon is exactly the identity. This gives a finite numerical and symbolic model of associator coherence within the projector calculus.

Every finite recoupling matrix can be compiled into at most $d(d-1)/2$ Givens rotations followed by diagonal signs. The full tensor space is needed only to define the lowest-weight projector. All later calculations take place in the multiplicity space, which gives an output-sensitive reduction.

Racah polynomials, $6j$ symbols, positive discrete-series recoupling, and the pentagon identity are classical. The generated contribution is the common packet architecture joining projector reduction, exact coupling atlases, stochastic kernels, entropy production, associator coherence, and finite quantum compilation. All figures were generated by the accompanying Python program. The release contains 30 vector figures, matching PNG files, 13 built-in checks, 20 independent tests, exact tables, and the complete Python source.

Status and claim boundary

The paper proves finite-dimensional statements about the positive discrete-series tensor products used below. It constructs the excitation spaces, lowest-weight projectors, pair-Casimir restrictions, orthogonal recoupling matrices, bistochastic kernels, four-body tree bases, pentagon closure, and Givens decompositions directly from matrices generated by the representation.

The historical identification of the recoupling entries with Racah polynomials and $6j$ symbols is classical. The classical theory also places Hahn polynomials at the two-body coupling stage and Wilson functions in continuous-series extensions. The present use of the word generated refers to the way these objects emerge from the packet calculus and not to a claim that Racah polynomials or $6j$ symbols are newly discovered here.

The stochastic channel obtained by squaring an orthogonal recoupling matrix describes basis change followed by dephasing or projective measurement. It does not represent coherent unitary evolution by itself. The entropy theorem applies to the diagonal probability vector after this measurement channel.

The pentagon product is exact because each tree basis is complete in the same multiplicity space. The numerical construction confirms the projector and ordering conventions. It is not offered as a new proof of the abstract monoidal coherence theorem.

The brief quantum discussion keeps the logarithmic Hamiltonian of the attached 2026 zeta-governed dynamics separate from the shape recoupling operator. The two modules can be composed, but their spectra are not identified. No result in this paper proves the Riemann hypothesis.

Contents

Status and claim boundary i

I Positive discrete-series packet spaces 1

1 From shape refinement to recoupling 1

2 Fixed excitation and the lowest-weight projector 2

3 Intermediate Casimir projectors 5

II Three-body Racah packet functions 6

4 Two complete coupling atlases 6

5 One excitation as a shape beam splitter 7

6 An exact three-channel example 9

7 The Leonard pair and the Racah recurrence 10

8 Parameter dependence and localization 13

III Bistochastic and quantum packet channels 15

9 From amplitudes to a Markov kernel 15

10 Entropy production and uncertainty 17

11 Repeated measurement and mixing 18

IV Four-body packet trees and pentagon coherence 18

12 Five complete coupling trees 19

13 The finite pentagon identity 21

14 Coassociativity and packet reconstruction 22

V Finite compilation and algorithms 23

15 Givens synthesis of a recoupling transform 23

16 Output-sensitive recoupling algorithm 24

17 Scaling and numerical conditioning 25

VI Classical identification, quantum use, and future deformations 26

18 Classical Racah and Wilson coordinates 26

19 Relation to the logarithmic quantum coordinate 27

20 What the packet calculus adds 28

21 Open problems generated by the construction 29

22 Conclusion 30

A Formula registry 31

B Numerical certificate tables 32
B.1 Three-body metrics 32
B.2 Four-body pentagon metrics 32

C Verification ledger 33

D Complete Python implementation 34

E Bibliographic note 51

F Reproduction 53

List of Figures

1.1 From shape refinement to Racah recoupling. The same positive shape packet that generated the Meixner basis becomes the input to a tensor-product projector, a pair-Casimir atlas, a Racah associator, and a finite stochastic or quantum transform. . . . 2
2.1 Positive discrete-series ladder. Each shape parameter gives an infinite ladder. Tensor products are first resolved by the total excitation and then projected onto the new lowest-weight multiplicity packet. . . . 4
2.2 The total-excitation simplex. Every lattice point is a composition of the total degree. The color records one occupation number and makes the finite packet geometry visible. . . . 4
2.3 Projector reduction before recoupling. The full tensor excitation space grows faster than the lowest-weight multiplicity space. The later Racah calculation uses only the multiplicity packet. . . . 5
3.1 Pair-Casimir spectra. Both coupling schemes have simple quadratic spectra. Their eigenvectors, not their eigenvalues, carry the nontrivial recoupling information. . . . 6
5.1 One-excitation shape mixing. The middle packet controls how strongly the two coupling atlases mix. The squared entries form a two-state bistochastic channel. . . . 8
5.2 Exact one-excitation matrix. The displayed numerical example agrees with the closed shape formula to floating-point roundoff. . . . 8
6.1 Equal-shape recoupling at total excitation two. The matrix contains only rational numbers and square roots. Its signed entries are amplitudes and their squares are transition probabilities. . . . 10
7.1 Racah packet transform at total excitation twelve. The alternating bands show the finite oscillatory structure of the associator. Orthogonality holds across the complete matrix, not packet by packet. . . . 11
7.2 Leonard-pair selection rule. In the left Casimir basis, the right Casimir connects only neighboring packet labels. This produces the finite three-term Racah recurrence. . . . 12
7.3 Racah packet columns. Selected columns behave as finite orthogonal special functions on the intermediate-Casimir lattice. . . . 13
8.1 Coupling-basis coherence across packet depth. The maximal amplitude decreases with dimension in the displayed shape family, while the equal-shape and unequal-shape cases follow different finite-depth profiles. 14
8.2 Shape dependence of the largest recoupling amplitude. The heat map records how the most concentrated overlap changes when one shape is varied. It is a diagnostic rather than a universal monotonicity theorem. . 14
9.1 Squared recoupling amplitudes. The kernel is nonnegative and every row and column sums to one. It is the classical channel induced by coherent basis change followed by measurement. . . . 16
9.2 Exact row and column marginals. The numerical residuals remain at floating-point scale. . . . 17
10.1 A basis packet spreads under recoupling. The solid curve is the mean row entropy and the band records the range across packet labels. The dotted line is the uniform maximum. . . . 18
12.1 Two three-packet coupling trees. The Racah matrix is the exact basis transform between the two complete intermediate-Casimir atlases. . . . 19
12.2 The four-packet associahedron. The five binary trees form a pentagon. Each edge carries an orthogonal recoupling matrix. . . . 20
12.3 Joint packet labels for one four-body tree. The two diagonal Casimirs provide a complete finite address on the multiplicity packet. . . . 20
13.1 Pentagon closure on growing multiplicity spaces. The maximum residual remains near floating-point roundoff while the dimension increases. . . . 21
13.2 Five edge associators. Each panel is an orthogonal transform between neighboring coupling trees at the same total excitation. . . . 22
14.1 Four-body projector reduction. The multiplicity packet is substantially smaller than the full excitation simplex. All five tree transforms live in the reduced space. . . . 23
15.1 Triangular Givens network. Each vertical segment is a two-coordinate rotation. The diagonal signs finish the exact finite synthesis. . . . 24
16.1 Output-sensitive recoupling algorithm. Project first, then diagonalize and recouple inside the multiplicity space. Dense full-tensor operators are not preserved as an output requirement. . . . 25

17.1 Reference complexity profiles. The deterministic curves compare a dense reduced eigensolve, a structured full transform, and one recurrence-generated column after normalization at the smallest displayed dimension. . 26
18.1 Classical-family map. The generated packet language retains the classical names and adds a common analysis, synthesis, refinement, and associator interpretation. 27
19.1 Two quantum coordinates. The logarithmic Hamiltonian acts on the number register, while the Racah transform acts on a finite shape-coupling register. Their roles remain distinct. 28
20.1 Theorem dependency map. Pair-Casimir projectors generate the orthogonal associator. Its signed and squared forms then support Racah recurrences, stochastic channels, pentagon coherence, and finite compilation. . . 29
21.1 Next deformation directions. Continuous, q-deformed, multivariate, and quantum-circuit extensions retain the same basic rule: declare a complete coupling atlas, compute the associator, and certify reconstruction. . . . 30

Part I

Positive discrete-series packet spaces

Chapter 1

From shape refinement to recoupling

Paper VI produced the orthonormal Meixner–Pollaczek packet basis and the ladder

$$K_0 e_n = \left(n + \frac{\alpha}{2}\right) e_n, \qquad K_+ e_n = \sqrt{(n+1)(n+\alpha)}\, e_{n+1},$$

$$K_- e_n = \sqrt{n(n+\alpha-1)}\, e_{n-1}.$$

Writing

$$\kappa = \frac{\alpha}{2}$$

gives the standard positive discrete-series form.

Definition 1.1 (Positive shape packet)**.** For $\kappa > 0$, let D_κ^+ be the Hilbert space with orthonormal basis $\{|n\rangle : n \geq 0\}$ and operators

$$K_0 |n\rangle = (n+\kappa)\,|n\rangle\,,$$

$$K_+ |n\rangle = \sqrt{(n+1)(n+2\kappa)}\,|n+1\rangle\,,$$

$$K_- |n\rangle = \sqrt{n(n+2\kappa-1)}\,|n-1\rangle\,.$$

Direct calculation gives

$$[K_0, K_\pm] = \pm K_\pm, \qquad [K_-, K_+] = 2K_0.$$

The Casimir

$$\boxed{\mathcal{C} = K_0(K_0 - 1) - K_+ K_-}$$

acts as the scalar

$$\boxed{\mathcal{C} = \kappa(\kappa - 1)\mathrm{I}.}$$

For m packets, the total generators are obtained from the primitive coproduct

$$K_\bullet^{[m]} = \sum_{r=1}^{m} K_\bullet^{(r)}.$$

The operator $K_0^{[m]}$ resolves the tensor product by total excitation.

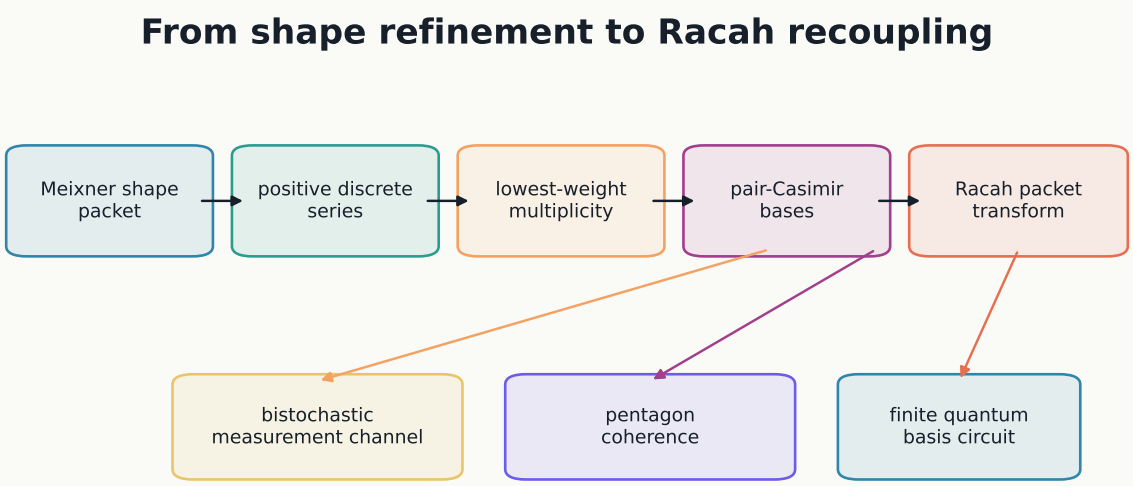


Figure 1.1: From shape refinement to Racah recoupling. The same positive shape packet that generated the Meixner basis becomes the input to a tensor-product projector, a pair-Casimir atlas, a Racah associator, and a finite stochastic or quantum transform.

Chapter 2

Fixed excitation and the lowest-weight projector

Let

$$\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_m)$$

and define the total-excitation space

$$\mathcal{H}_N^{(m)} = \operatorname{span}\left\{ |n_1, \dots, n_m\rangle : \sum_{r=1}^{m} n_r = N \right\}.$$

Its dimension is the number of weak compositions of N into m parts.

Proposition 2.1 (Excitation-space dimension)**.**

$$\boxed{\dim \mathcal{H}_N^{(m)} = \binom{N+m-1}{m-1}.}$$

The total lowering operator maps

$$K_-^{[m]} : \mathcal{H}_N^{(m)} \longrightarrow \mathcal{H}_{N-1}^{(m)}.$$

The packet of new lowest-weight vectors appearing at level N is

$$\boxed{\mathcal{M}_N^{(m)} = \ker\left(K_-^{[m]}|_{\mathcal{H}_N^{(m)}} \right).}$$

Theorem 2.2 (Multiplicity dimension)**.** *For $N \geq 0$ and $m \geq 2$,*

$$\boxed{\dim \mathcal{M}_N^{(m)} = \binom{N+m-2}{m-2}.}$$

Proof. The adjoint of the lowering map is the total raising map

$$K_+^{[m]} : \mathcal{H}_{N-1}^{(m)} \longrightarrow \mathcal{H}_N^{(m)}.$$

The positive discrete series has no highest-weight vector, so $K_+^{[m]}$ is injective on every finite excitation level. Hence the lowering map has full row rank. Rank-nullity gives

$$\begin{aligned} \dim \ker K_-^{[m]} &= \binom{N+m-1}{m-1} - \binom{N+m-2}{m-1} \\ &= \binom{N+m-2}{m-2}. \end{aligned}$$

□

The orthogonal projector onto $\mathcal{M}_N^{(m)}$ is the first major compression in the construction. The ratio between the full excitation dimension and the multiplicity dimension is

$$\boxed{\frac{\dim \mathcal{H}_N^{(m)}}{\dim \mathcal{M}_N^{(m)}} = \frac{N+m-1}{m-1}.}$$

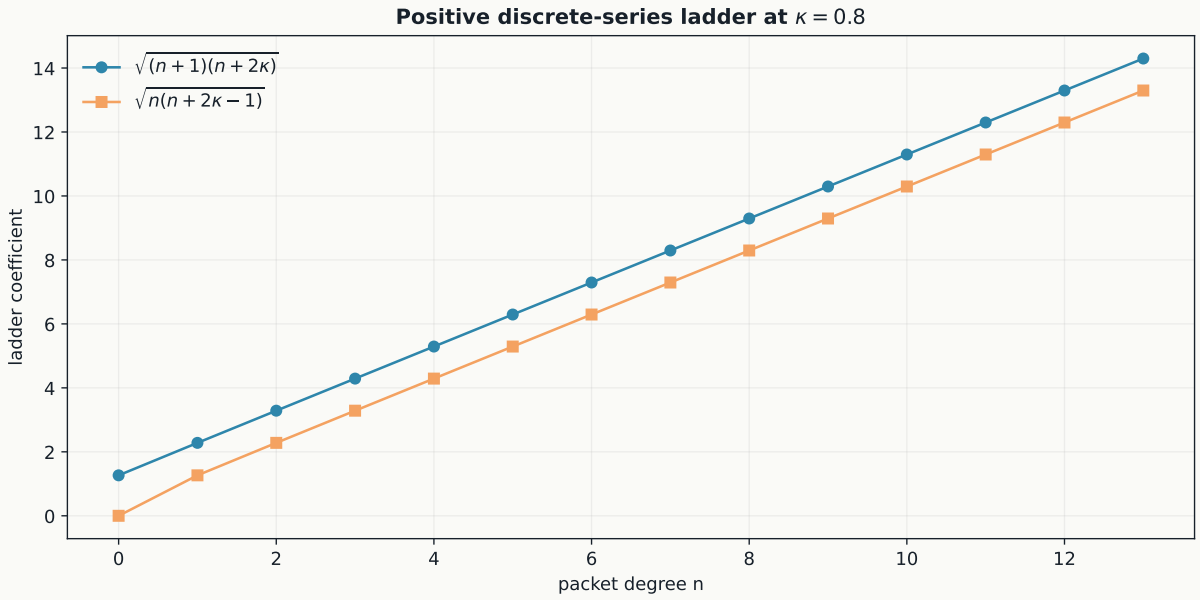


Figure 2.1: Positive discrete-series ladder. Each shape parameter gives an infinite ladder. Tensor products are first resolved by the total excitation and then projected onto the new lowest-weight multiplicity packet.

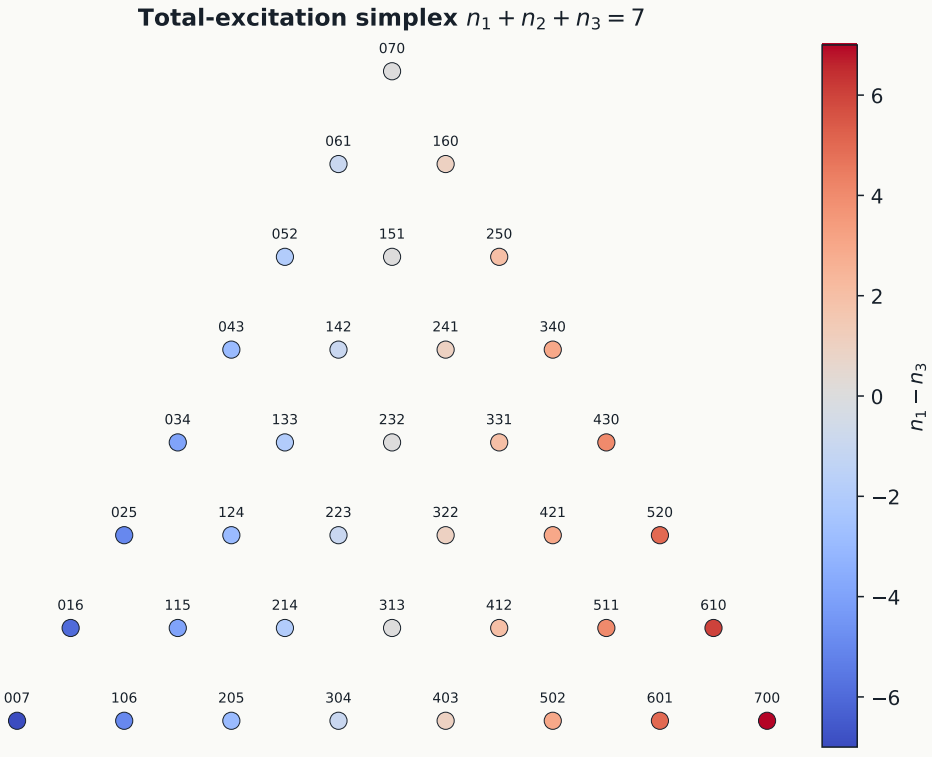


Figure 2.2: The total-excitation simplex. Every lattice point is a composition of the total degree. The color records one occupation number and makes the finite packet geometry visible.

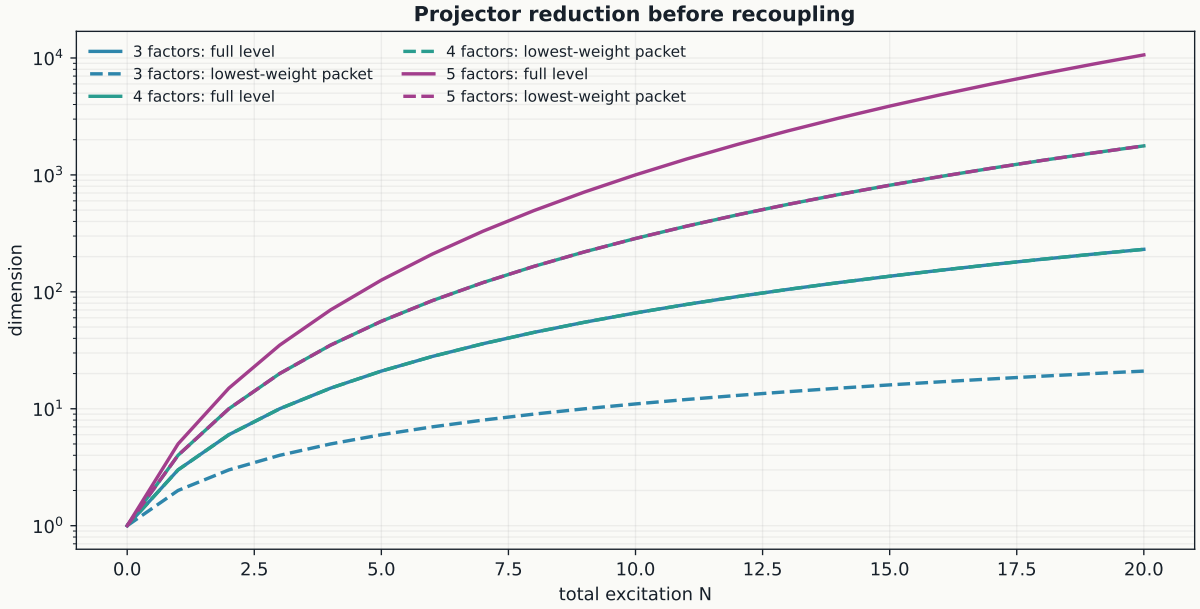


Figure 2.3: Projector reduction before recoupling. The full tensor excitation space grows faster than the lowest-weight multiplicity space. The later Racah calculation uses only the multiplicity packet.

Chapter 3

Intermediate Casimir projectors

For a subset $A \subseteq \{1, \dots, m\}$, define

$$K_\bullet^A = \sum_{r \in A} K_\bullet^{(r)}$$

and

$$\boxed{\mathcal{C}_A = K_0^A(K_0^A - 1) - K_+^A K_-^A.}$$

If two subsets are nested or disjoint, the corresponding Casimirs commute.

For two packets,

$$D^+_{\kappa_a} \otimes D^+_{\kappa_b} \cong \bigoplus_{j=0}^{\infty} D^+_{\kappa_a+\kappa_b+j}.$$

Therefore the intermediate Casimir spectrum at total excitation N is

$$\boxed{\lambda_j^{ab} = (\kappa_a + \kappa_b + j)(\kappa_a + \kappa_b + j - 1), \qquad 0 \le j \le N.}$$

Proposition 3.1 (Three-packet pair spectrum)**.** *On $\mathcal{M}_N^{(3)}$, both $\mathcal{C}_{12}$ and $\mathcal{C}_{23}$ have simple spectra of length $N+1$. Their eigenvalues are λ_j^{12} and λ_m^{23} respectively.*

Proof. The two-packet decomposition above fixes the intermediate shape. Coupling the resulting representation to the remaining packet gives the final lowest weight

$$\kappa_1 + \kappa_2 + \kappa_3 + N.$$

For every $j \in \{0, \dots, N\}$ there is exactly one compatible remaining excitation $N - j$. Thus each intermediate shape appears once in the fixed final multiplicity space. □

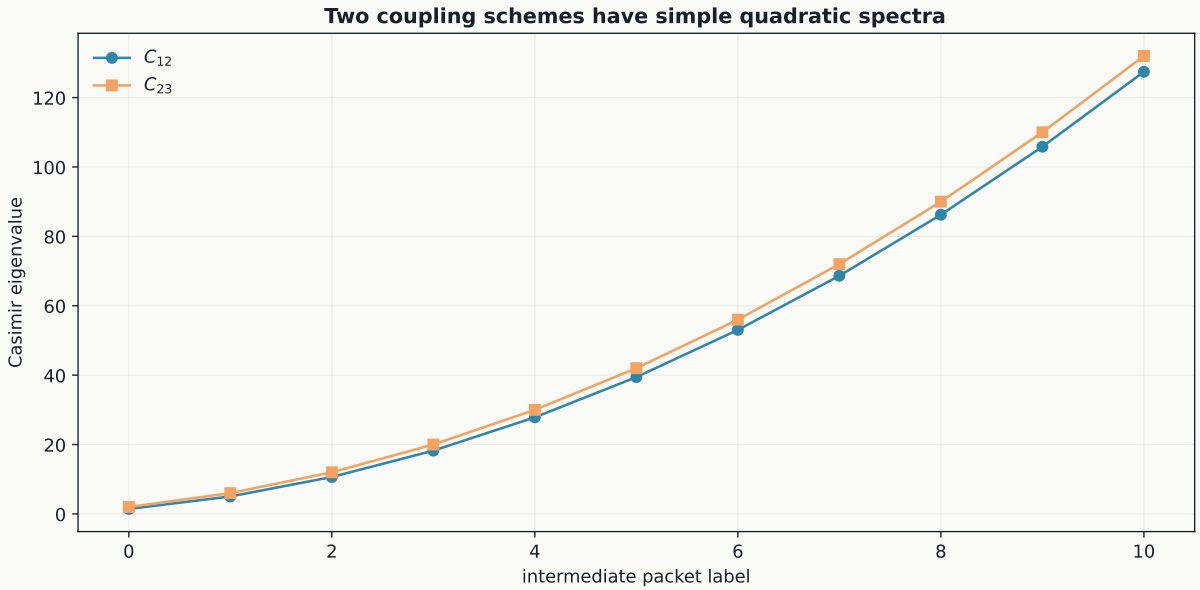


Figure 3.1: Pair-Casimir spectra. Both coupling schemes have simple quadratic spectra. Their eigenvectors, not their eigenvalues, carry the nontrivial recoupling information.

Part II

Three-body Racah packet functions

Chapter 4

Two complete coupling atlases

For three packets, define the left basis by

$$\mathcal{C}_{12}\,|j,(12)3\rangle = \lambda_j^{12}\,|j,(12)3\rangle$$

and the right basis by

$$\mathcal{C}_{23}\,|m,1(23)\rangle = \lambda_m^{23}\,|m,1(23)\rangle\,.$$

Both bases are complete in $\mathcal{M}_N^{(3)}$.

Definition 4.1 (Racah packet matrix)**.**

$$\boxed{\mathcal{R}_N^{\boldsymbol{\kappa}}(j,m) = \langle j,(12)3|\,|m,1(23)\rangle\,.}$$

After a real phase convention, the matrix $\mathcal{R}_N^{\boldsymbol{\kappa}}$ is real orthogonal.

Theorem 4.2 (Perfect recoupling reconstruction)**.**

$$\boxed{\sum_{m=0}^{N} \mathcal{R}_N^{\boldsymbol{\kappa}}(j,m)\mathcal{R}_N^{\boldsymbol{\kappa}}(j',m) = \delta_{jj'}}$$

and

$$\boxed{\sum_{j=0}^{N} \mathcal{R}_N^{\boldsymbol{\kappa}}(j,m)\mathcal{R}_N^{\boldsymbol{\kappa}}(j,m') = \delta_{mm'}.}$$

Proof. These are the row and column orthogonality relations of the overlap between two orthonormal bases of the same finite-dimensional space. □

The entries are the positive discrete-series $6j$ coefficients, after normalization and phase choice. Their scalar polynomial realization is the Racah family. We retain the classical name and use packet to emphasize the analysis-synthesis role of the entire matrix.

Chapter 5

One excitation as a shape beam splitter

At $N = 1$, the multiplicity space has dimension two. Write

$$K = \kappa_1 + \kappa_2 + \kappa_3.$$

The complete transform is

$$\boxed{\mathcal{R}_1^{\boldsymbol{\kappa}} = \begin{pmatrix} \sqrt{\dfrac{\kappa_1\kappa_3}{(\kappa_1+\kappa_2)(\kappa_2+\kappa_3)}} & \sqrt{\dfrac{\kappa_2 K}{(\kappa_1+\kappa_2)(\kappa_2+\kappa_3)}} \\ \sqrt{\dfrac{\kappa_2 K}{(\kappa_1+\kappa_2)(\kappa_2+\kappa_3)}} & -\sqrt{\dfrac{\kappa_1\kappa_3}{(\kappa_1+\kappa_2)(\kappa_2+\kappa_3)}} \end{pmatrix}.}$$

Proof. Let the two entries in the first row be c and s. Their squares add to

$$\frac{\kappa_1\kappa_3 + \kappa_2 K}{(\kappa_1+\kappa_2)(\kappa_2+\kappa_3)} = 1.$$

The lowest-weight vectors associated with the two intermediate Casimir eigenvalues determine the positive choice for the first row. Orthogonality then fixes the second row up to one sign. The displayed convention agrees with the ordered eigenbases used throughout the paper. □

This transform behaves like a shape-controlled beam splitter. Its transition probabilities are

$$P = \begin{pmatrix} c^2 & s^2 \\ s^2 & c^2 \end{pmatrix}.$$

When $\kappa_1 = \kappa_2 = \kappa_3 = 1/2$,

$$c = \frac{1}{2}, \qquad s = \frac{\sqrt{3}}{2}.$$

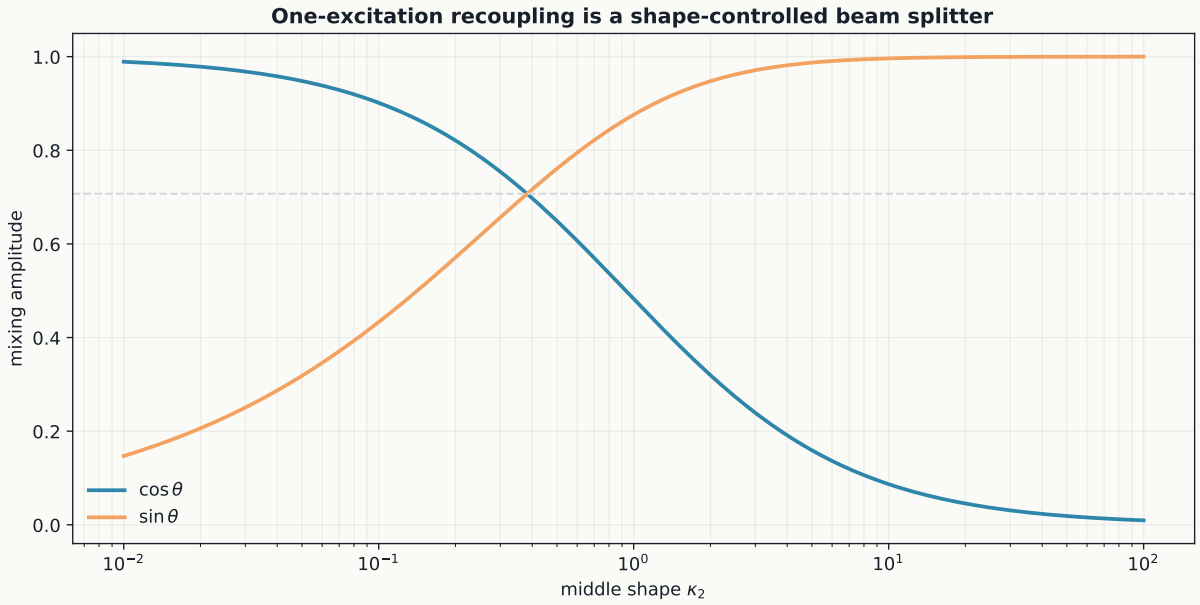


Figure 5.1: One-excitation shape mixing. The middle packet controls how strongly the two coupling atlases mix. The squared entries form a two-state bistochastic channel.

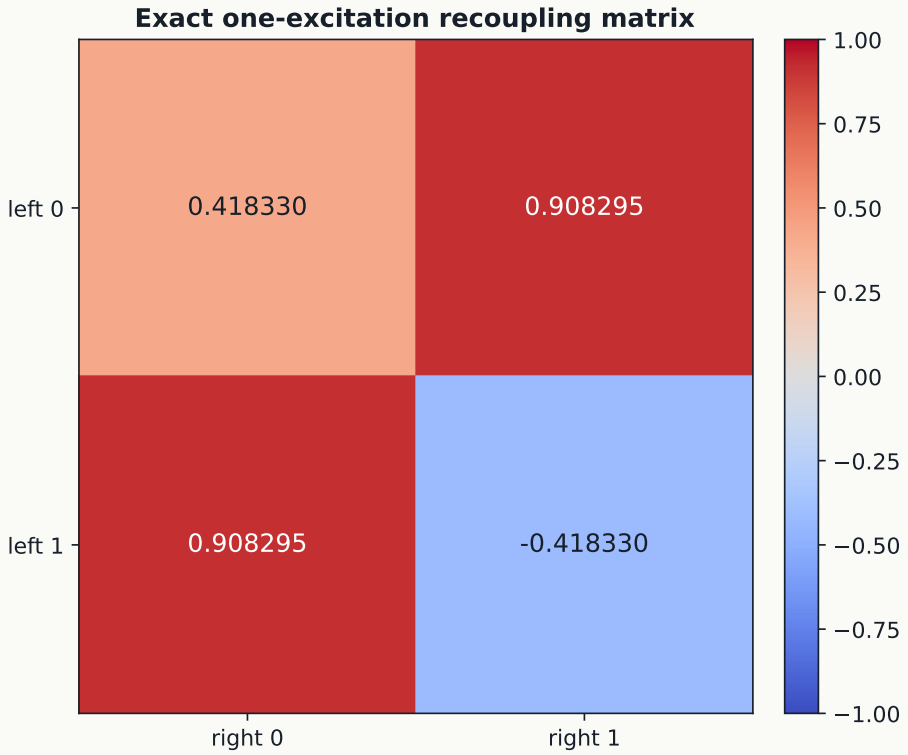


Figure 5.2: Exact one-excitation matrix. The displayed numerical example agrees with the closed shape formula to floating-point roundoff.

Chapter 6

An exact three-channel example

Take

$$\kappa_1 = \kappa_2 = \kappa_3 = \frac{1}{2}$$

and $N = 2$. Then

$$\boxed{\mathcal{R}_2 = \begin{pmatrix} \frac{1}{3} & \frac{1}{\sqrt{3}} & \frac{\sqrt{5}}{3} \\ \frac{1}{\sqrt{3}} & \frac{1}{2} & -\frac{\sqrt{15}}{6} \\ \frac{\sqrt{5}}{3} & -\frac{\sqrt{15}}{6} & \frac{1}{6} \end{pmatrix}.}$$

Worked example 6.1 (Direct orthogonality check)**.** The first row has squared norm

$$\frac{1}{9} + \frac{1}{3} + \frac{5}{9} = 1.$$

The inner product of the first two rows is

$$\frac{1}{3\sqrt{3}} + \frac{1}{2\sqrt{3}} - \frac{\sqrt{75}}{18} = 0.$$

The remaining row relations follow in the same way.

The corresponding probability kernel is

$$P = \begin{pmatrix} \frac{1}{9} & \frac{1}{3} & \frac{5}{9} \\ \frac{1}{3} & \frac{1}{4} & \frac{5}{12} \\ \frac{5}{9} & \frac{5}{12} & \frac{1}{36} \end{pmatrix}.$$

Every row and every column sums to one.

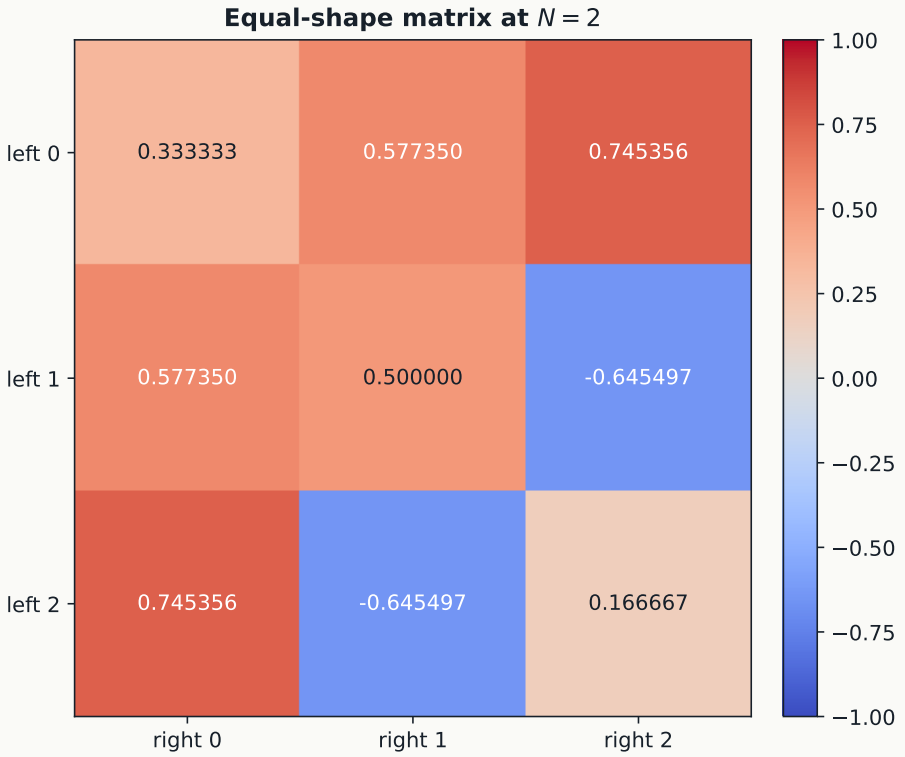


Figure 6.1: Equal-shape recoupling at total excitation two. The matrix contains only rational numbers and square roots. Its signed entries are amplitudes and their squares are transition probabilities.

Chapter 7

The Leonard pair and the Racah recurrence

Express $\mathcal{C}_{23}$ in the $\mathcal{C}_{12}$ eigenbasis:

$$J^{(23|12)} = \mathcal{R}_N \operatorname{diag}(\lambda_0^{23}, \dots, \lambda_N^{23}) \mathcal{R}_N^{\mathsf{T}}.$$

The Racah algebra implies that this matrix is irreducibly tridiagonal.

Theorem 7.1 (Finite three-term packet recurrence)**.** *There are real coefficients a_j and positive off-diagonal coefficients b_j such that every column satisfies*

$$\boxed{b_{j-1}\mathcal{R}_N(j-1,m) + a_j\mathcal{R}_N(j,m) + b_j\mathcal{R}_N(j+1,m) = \lambda_m^{23}\mathcal{R}_N(j,m).}$$

The endpoint terms outside $0 \le j \le N$ are omitted.

Proof. The matrix $J^{(23|12)}$ is the representation of $\mathcal{C}_{23}$ in the $\mathcal{C}_{12}$ basis. The pair $\mathcal{C}_{12}, \mathcal{C}_{23}$ generates the rank-one Racah algebra. On the irreducible fixed-N multiplicity module, either Casimir is diagonal in its own basis and tridiagonal in the eigenbasis of the other. Applying the matrix eigenvalue equation to the mth column gives the recurrence. □

The dual statement holds with j and m exchanged. Thus the packet matrix is bispectral. Classical normalization turns these finite functions into Racah polynomials.

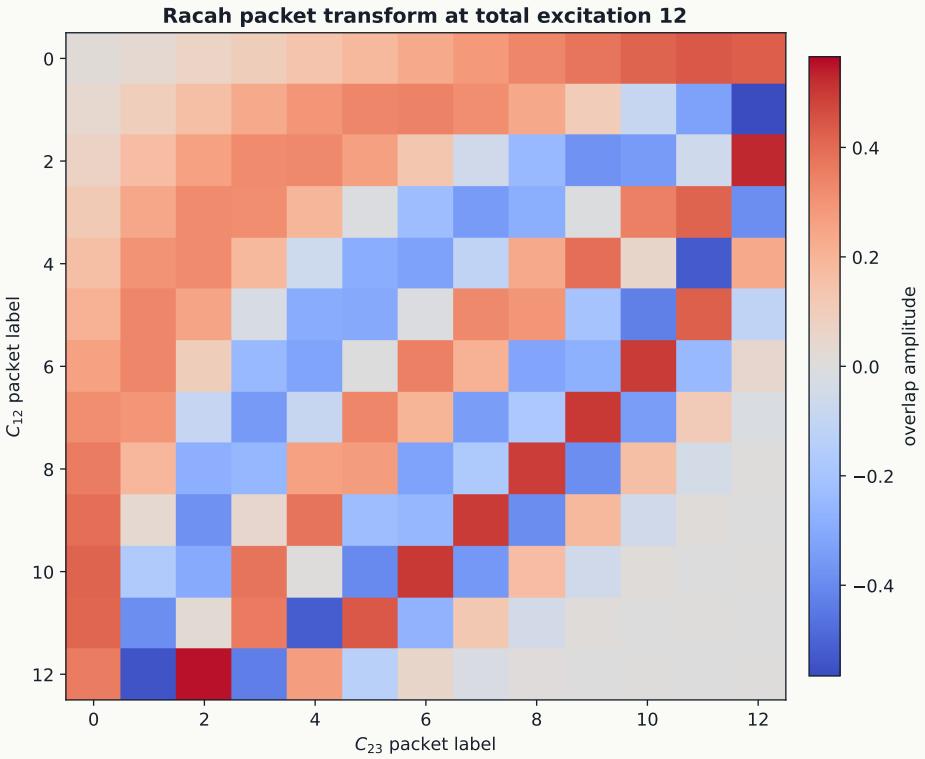


Figure 7.1: Racah packet transform at total excitation twelve. The alternating bands show the finite oscillatory structure of the associator. Orthogonality holds across the complete matrix, not packet by packet.

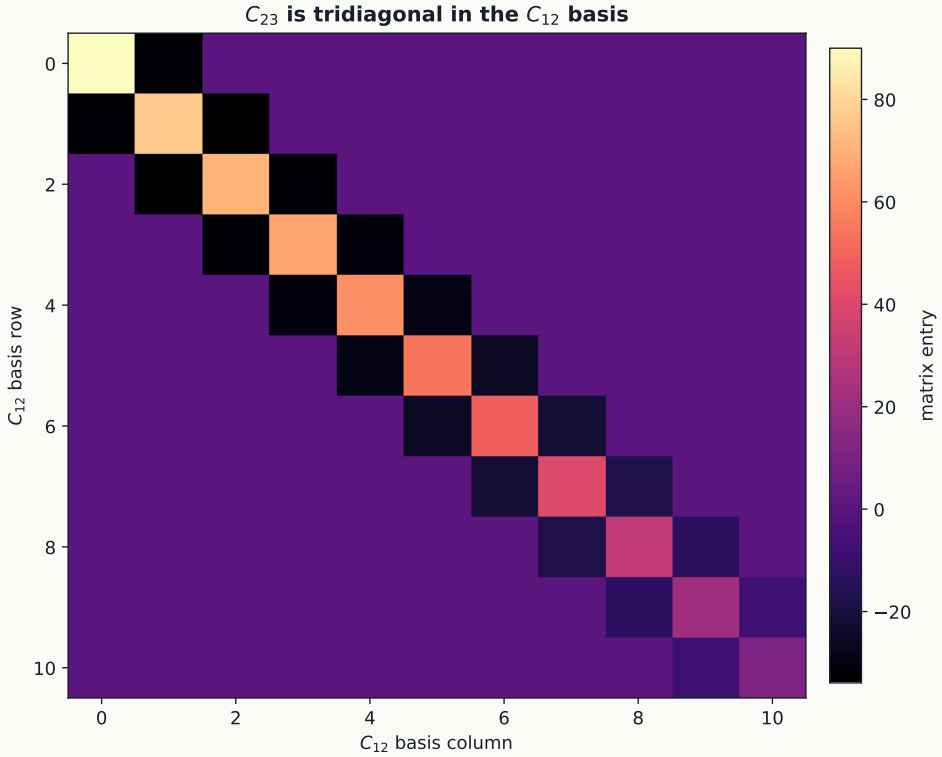


Figure 7.2: Leonard-pair selection rule. In the left Casimir basis, the right Casimir connects only neighboring packet labels. This produces the finite three-term Racah recurrence.

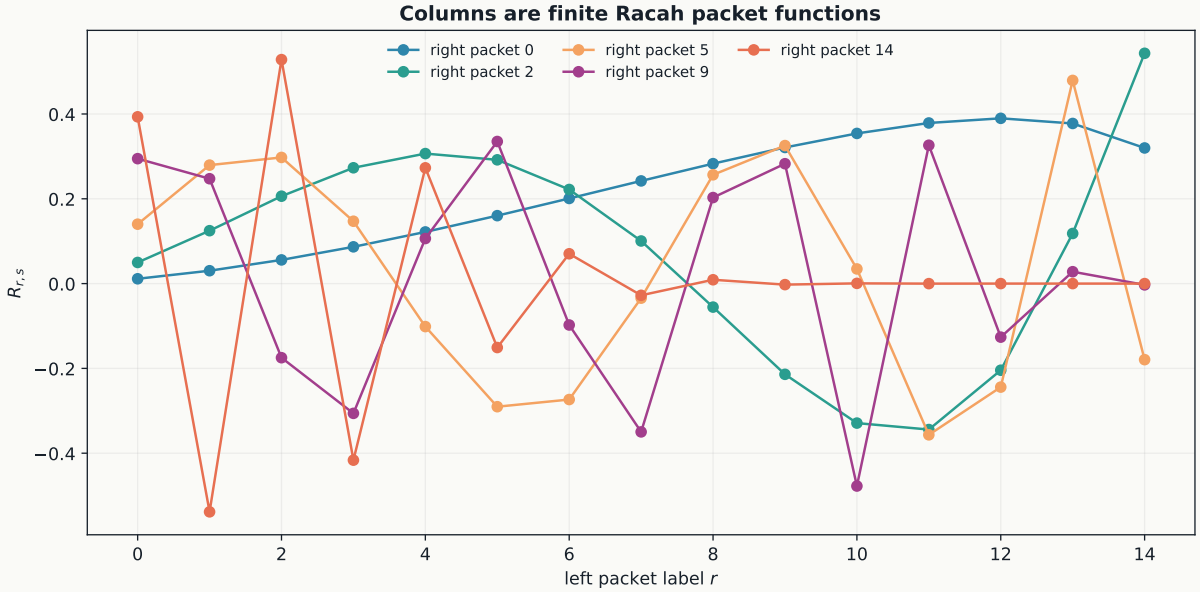


Figure 7.3: Racah packet columns. Selected columns behave as finite orthogonal special functions on the intermediate-Casimir lattice.

Chapter 8

Parameter dependence and localization

The shapes $\kappa_1, \kappa_2, \kappa_3$ are continuous. The pair spectra and recoupling matrix vary smoothly as long as all shapes remain positive.

The maximal entry

$$\mu_N(\boldsymbol{\kappa}) = \max_{j,m} |\mathcal{R}_N^{\boldsymbol{\kappa}}(j,m)|$$

measures the coherence between the two coupling atlases. Small μ_N means that no basis vector is strongly concentrated in the other atlas.

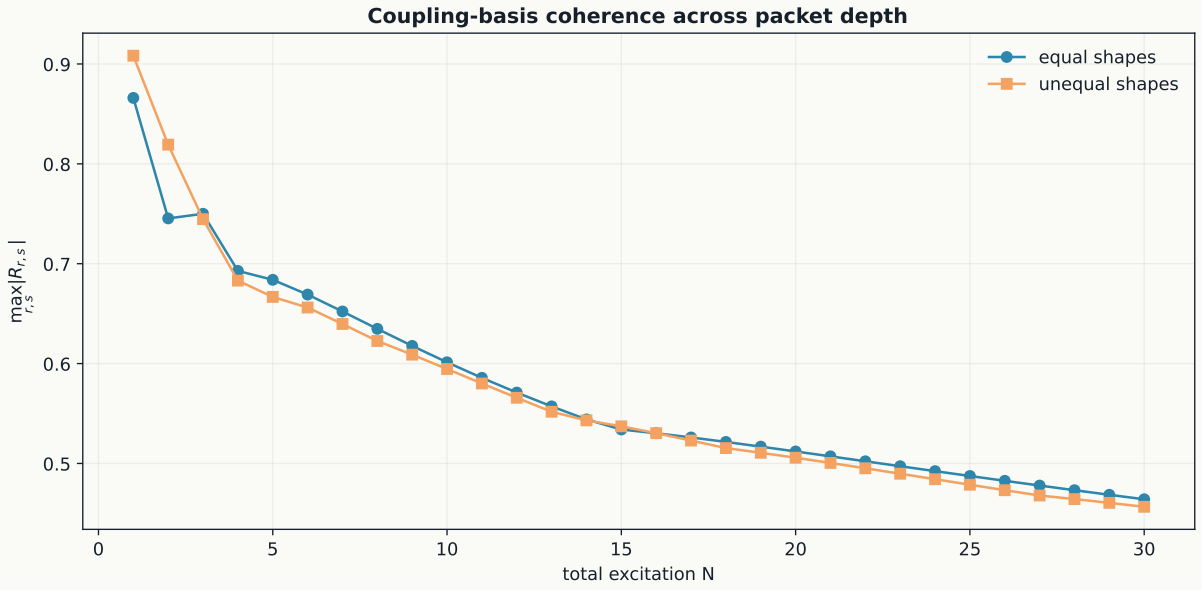


Figure 8.1: Coupling-basis coherence across packet depth. The maximal amplitude decreases with dimension in the displayed shape family, while the equal-shape and unequal-shape cases follow different finite-depth profiles.

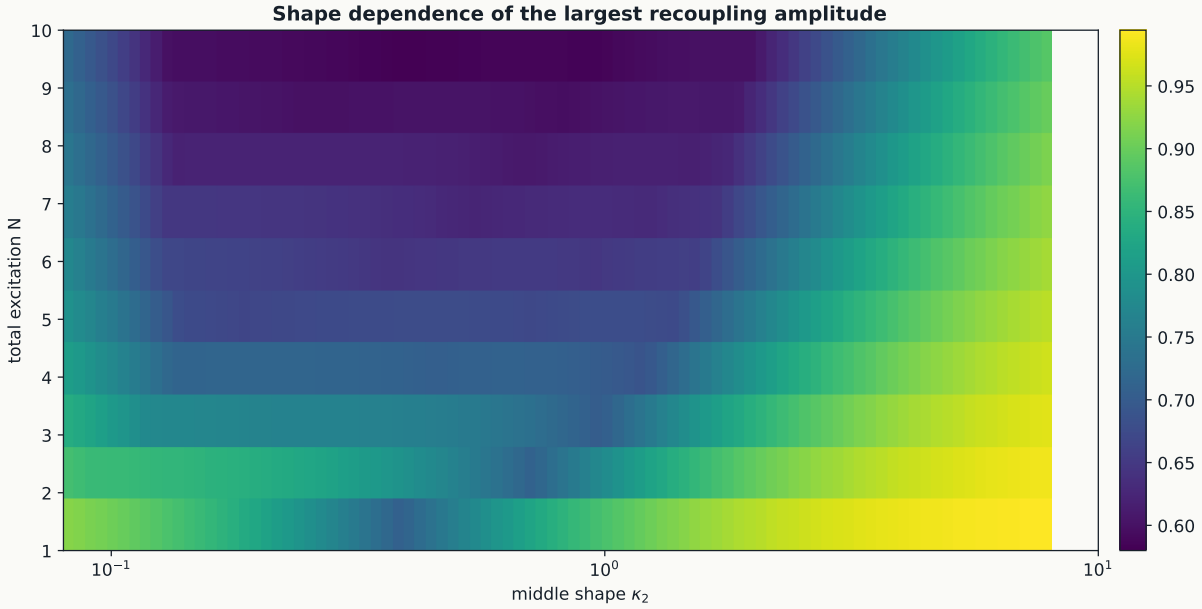


Figure 8.2: Shape dependence of the largest recoupling amplitude. The heat map records how the most concentrated overlap changes when one shape is varied. It is a diagnostic rather than a universal monotonicity theorem.

Part III

Bistochastic and quantum packet channels

Chapter 9

From amplitudes to a Markov kernel

Define

$$\boxed{P_{jm} = |\mathcal{R}_N(j,m)|^2 .}$$

Theorem 9.1 (Bistochastic Racah kernel)**.** *The matrix P is doubly stochastic:*

$$P_{jm} \geq 0, \qquad \sum_m P_{jm} = 1, \qquad \sum_j P_{jm} = 1.$$

Proof. Nonnegativity is immediate. Row sums and column sums are the squared norms of the rows and columns of the orthogonal matrix $\mathcal{R}_N$. □

The channel has a direct measurement interpretation. Begin with a density matrix diagonal in the left coupling basis,

$$\rho = \sum_j p_j \, |j,(12)3\rangle \langle j,(12)3| \, .$$

Measure in the right coupling basis and discard the outcome phase. The output distribution is

$$\boxed{q_m = \sum_j p_j P_{jm}.}$$

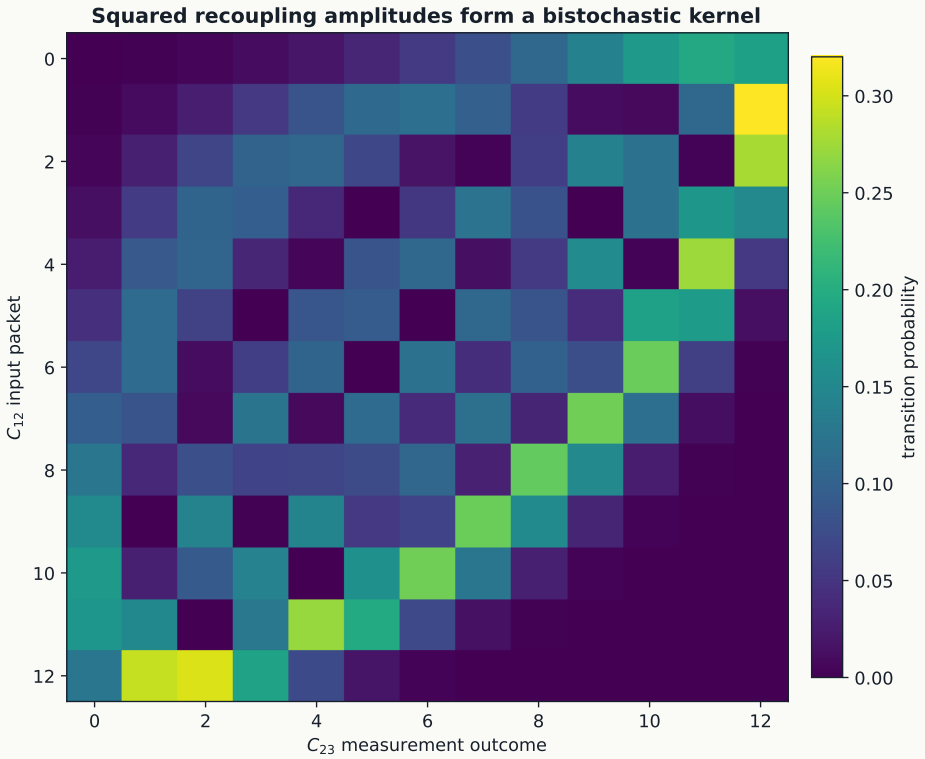


Figure 9.1: Squared recoupling amplitudes. The kernel is nonnegative and every row and column sums to one. It is the classical channel induced by coherent basis change followed by measurement.

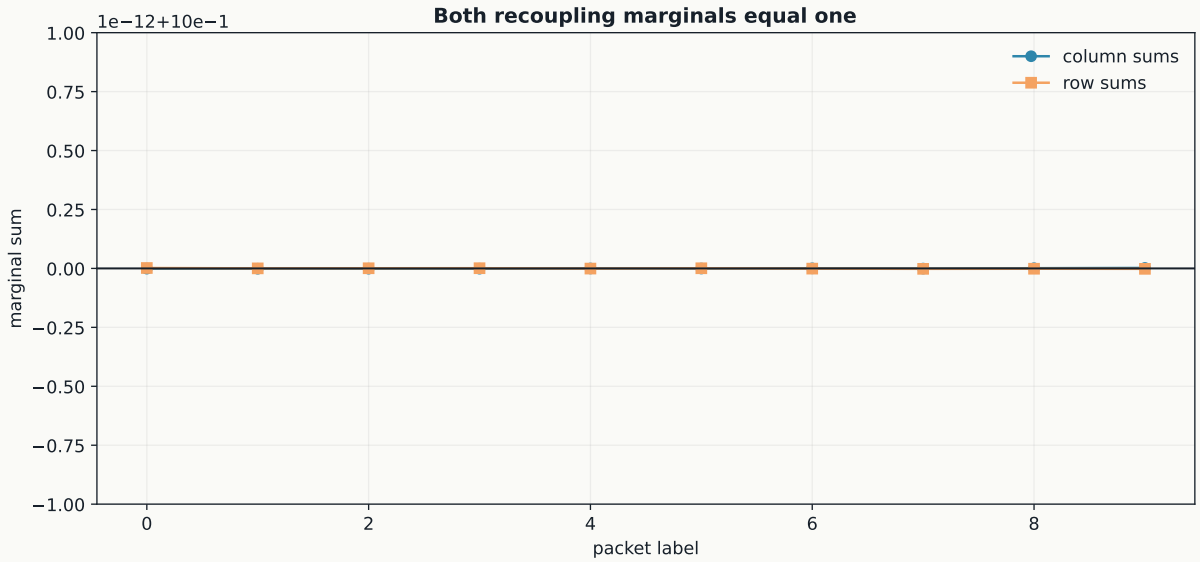


Figure 9.2: Exact row and column marginals. The numerical residuals remain at floating-point scale.

Chapter 10

Entropy production and uncertainty

Let

$$\mathrm{H}(p) = -\sum_j p_j \log p_j$$

be the Shannon entropy.

Theorem 10.1 (Entropy monotonicity)**.** *For every probability vector p,*

$$\boxed{\mathrm{H}(pP) \geq \mathrm{H}(p).}$$

Proof. A doubly stochastic matrix is a convex combination of permutation matrices. Therefore pP is majorized by p. Shannon entropy is Schur concave, which gives the result. □

The uniform distribution is stationary:

$$\frac{1}{N+1}(1,\ldots,1)P = \frac{1}{N+1}(1,\ldots,1).$$

For a left basis state j, the right-basis entropy is the row entropy

$$\mathrm{H}_j = -\sum_m P_{jm} \log P_{jm}.$$

The Maassen–Uffink bound gives

$$\mathrm{H}_{12} + \mathrm{H}_{23} \geq -2 \log \mu_N.$$

For a pure left basis packet, $\mathrm{H}_{12} = 0$.

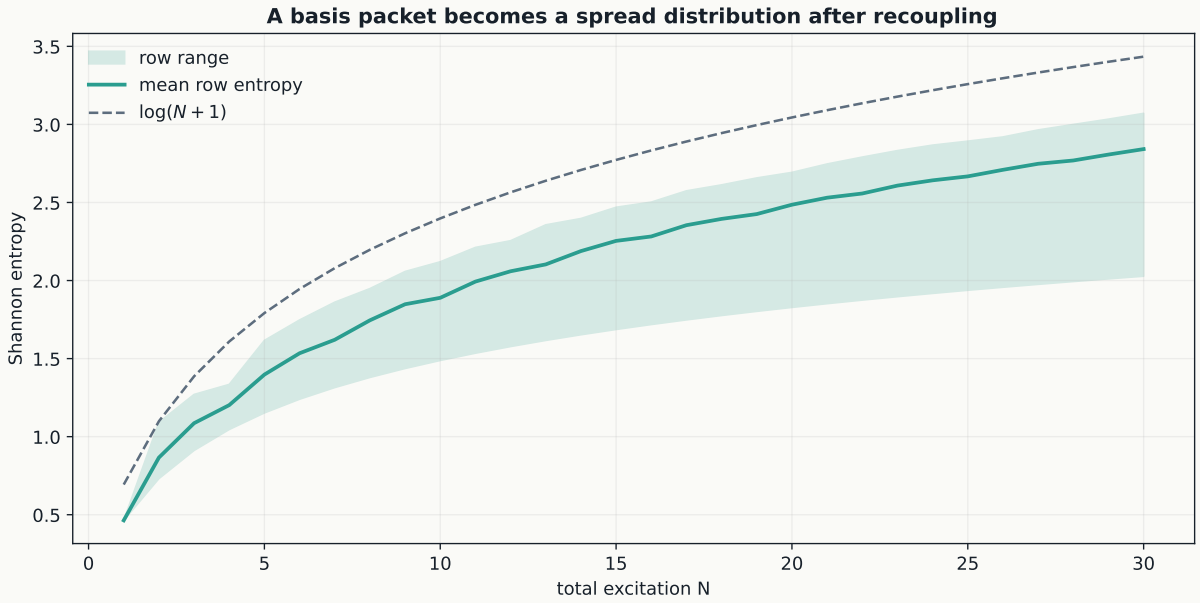


Figure 10.1: A basis packet spreads under recoupling. The solid curve is the mean row entropy and the band records the range across packet labels. The dotted line is the uniform maximum.

Chapter 11

Repeated measurement and mixing

Repeated application of P defines a finite Markov chain. Because P is bistochastic, the uniform law is always stationary. When the kernel is primitive, every initial probability vector converges to uniform.

The coherent matrix $\mathcal{R}_N$ and the classical kernel P must remain distinct. Two coherent recouplings can interfere, while two applications of P cannot recover phase.

Boundary statement 11.1 (What entropy does not prove). Entropy increase under P does not mean that the unitary Racah transform is irreversible. The entropy appears only after dephasing or measurement discards the phase relations. The underlying orthogonal transform has the exact inverse $\mathcal{R}_N^{\mathsf{T}}$.

Part IV

Four-body packet trees and pentagon coherence

Chapter 12

Five complete coupling trees

For four packets and total excitation N,

$$\dim \mathcal{M}_N^{(4)} = \binom{N+2}{2}.$$

A binary coupling tree is specified by two commuting intermediate Casimirs. The five trees are

$$(((12)3)4), \quad ((1(23))4), \quad (1((23)4)), \quad (1(2(34))), \quad ((12)(34)).$$

Each tree gives a complete joint eigenbasis of the same multiplicity space.

Proposition 12.1 (Commuting tree observables)**.** *For every tree, its two intermediate Casimirs commute on the complete tensor product and on the lowest-weight multiplicity packet.*

Proof. The subsets labeling the two internal edges are nested or disjoint. Coproduct generators on disjoint tensor factors commute. A Casimir of a smaller subset commutes with the total generators of that subset and hence with the Casimir of a containing subset. □

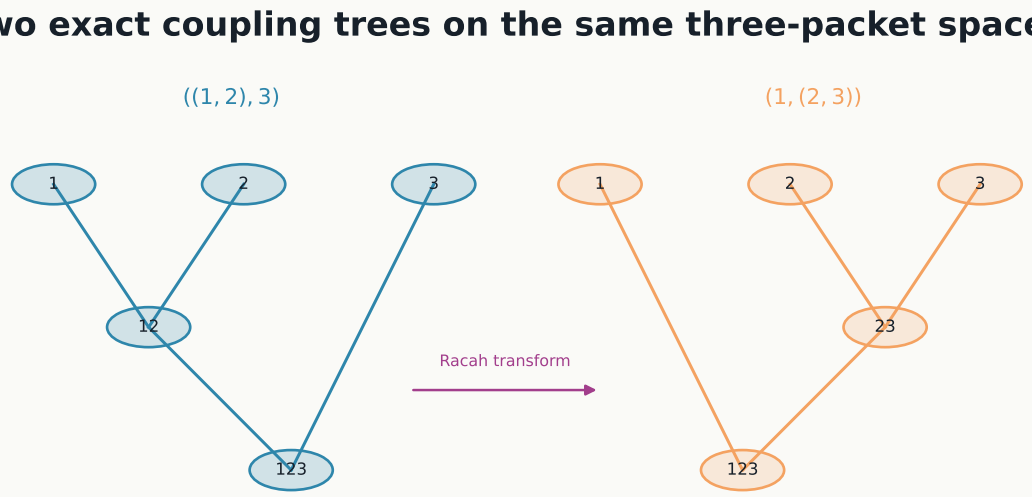


Figure 12.1: Two three-packet coupling trees. The Racah matrix is the exact basis transform between the two complete intermediate-Casimir atlases.

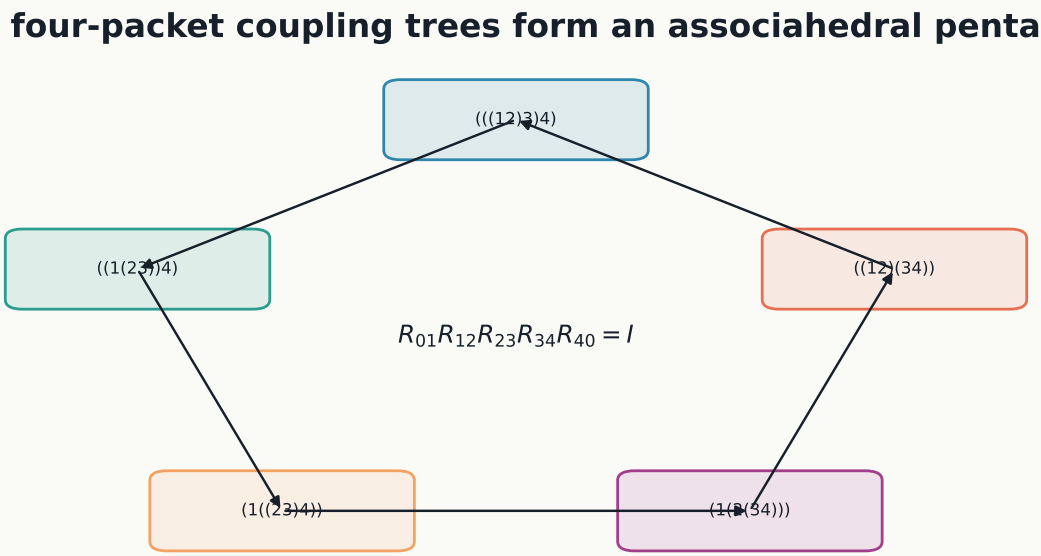


Figure 12.2: The four-packet associahedron. The five binary trees form a pentagon. Each edge carries an orthogonal recoupling matrix.

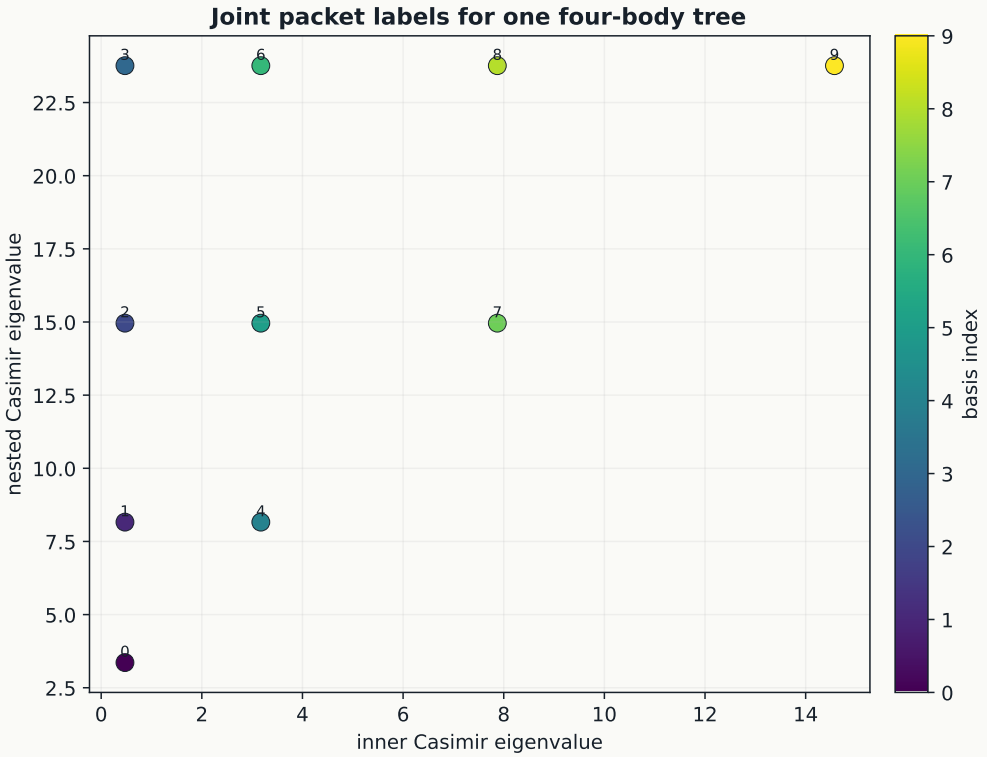


Figure 12.3: Joint packet labels for one four-body tree. The two diagonal Casimirs provide a complete finite address on the multiplicity packet.

Chapter 13

The finite pentagon identity

Let $B_0, \dots, B_4$ be the five orthogonal tree-basis matrices, with columns expressed in one common multiplicity frame. Define the edge transform

$$U_{r,r+1} = B_r^{\mathsf{T}} B_{r+1}, \qquad r \pmod 5.$$

Theorem 13.1 (Pentagon closure)**.**

$$\boxed{U_{01}U_{12}U_{23}U_{34}U_{40} = \mathrm{I}.}$$

Proof. Insert the definitions and use $B_r B_r^{\mathsf{T}}$ as the identity on the common multiplicity space:

$$\begin{aligned} U_{01}U_{12}U_{23}U_{34}U_{40} &= B_0^{\mathsf{T}} B_1 B_1^{\mathsf{T}} B_2 B_2^{\mathsf{T}} B_3 B_3^{\mathsf{T}} B_4 B_4^{\mathsf{T}} B_0 \\ &= B_0^{\mathsf{T}} B_0 \\ &= \mathrm{I}. \end{aligned}$$

□

This is the finite matrix form of associator coherence. It also acts as a stringent implementation certificate. A wrong phase, label order, or missing multiplicity vector breaks the product.

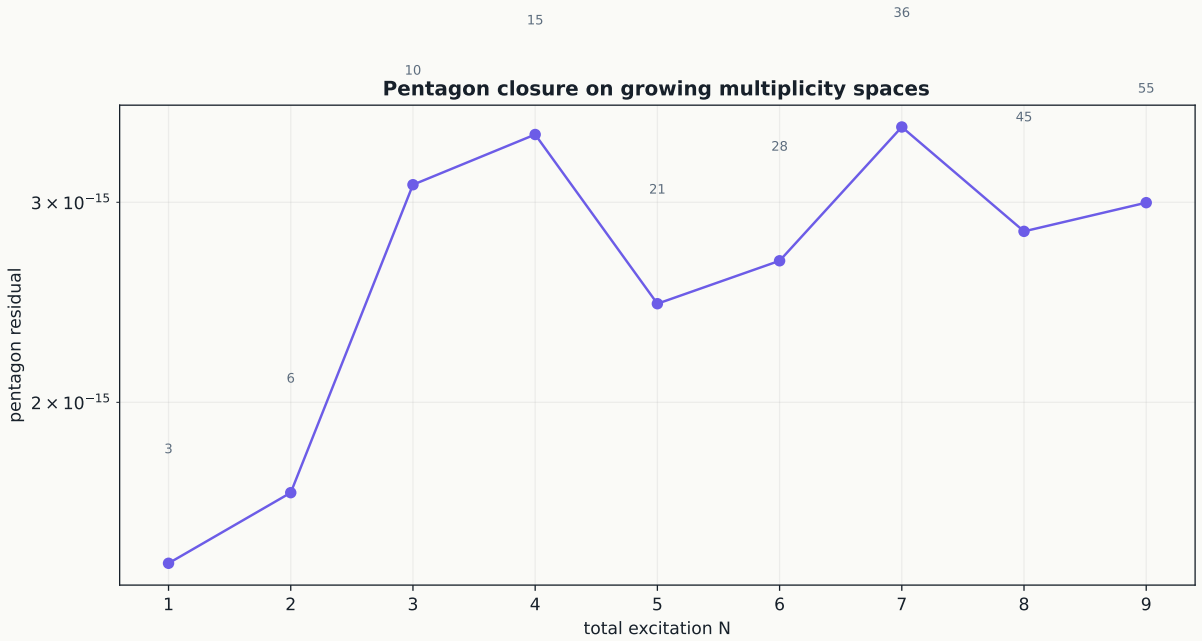


Figure 13.1: Pentagon closure on growing multiplicity spaces. The maximum residual remains near floating-point roundoff while the dimension increases.

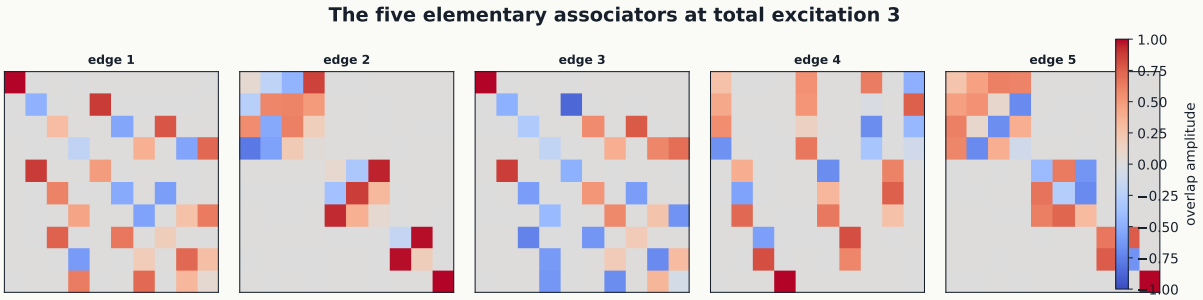


Figure 13.2: Five edge associators. Each panel is an orthogonal transform between neighboring coupling trees at the same total excitation.

Chapter 14

Coassociativity and packet reconstruction

The beta-binomial refinement of Paper VI already proved that splitting a total shape into several child shapes is coassociative. The present paper adds the change of intermediate basis between different bracketings.

At the amplitude level, a state may be expanded in any tree basis:

$$|\Psi\rangle = \sum_{\alpha} c_{\alpha}^{(T)} \, |\alpha, T\rangle \, .$$

For neighboring trees,

$$c^{(T')} = U_{T,T'}^{\mathsf{T}} c^{(T)}.$$

Because the pentagon closes, the coefficient vector is independent of the chosen route through the coupling graph.

This is the packet form of path-independent reconstruction:

analysis in one tree $\longrightarrow$ any legal sequence of recouplings $\longrightarrow$ same synthesized state.

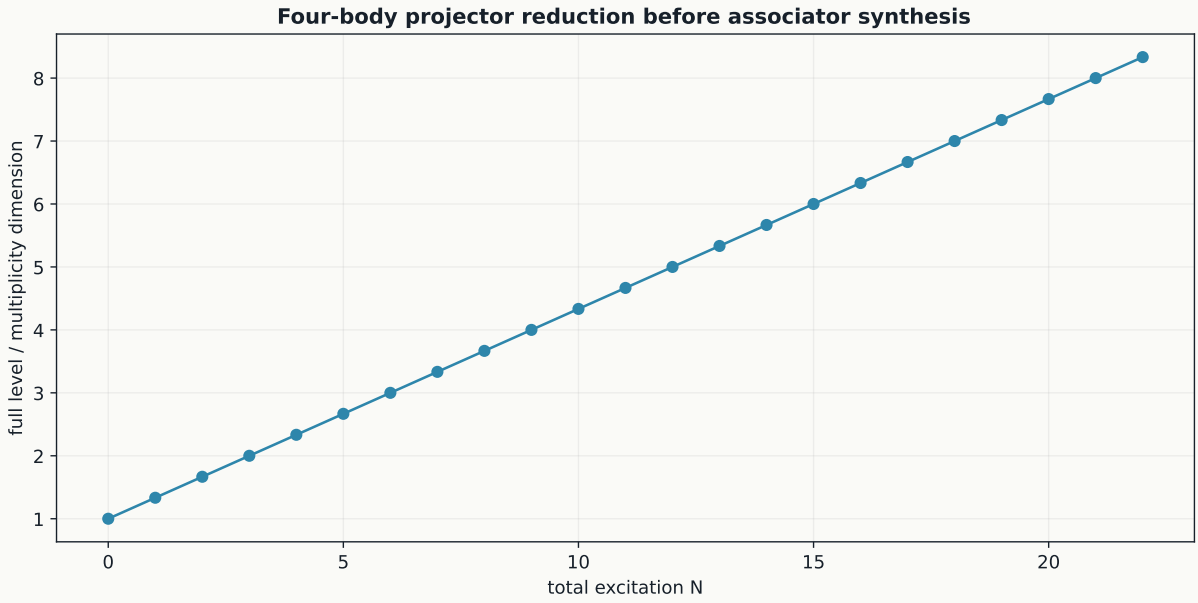


Figure 14.1: Four-body projector reduction. The multiplicity packet is substantially smaller than the full excitation simplex. All five tree transforms live in the reduced space.

Part V

Finite compilation and algorithms

Chapter 15

Givens synthesis of a recoupling transform

Any real orthogonal $d \times d$ matrix can be triangularized by plane rotations. Reversing the rotations gives a finite circuit.

Theorem 15.1 (Givens packet compilation)**.** *Every Racah packet matrix $R \in O(d)$ can be written as*

$$\boxed{R = G_1^{\mathsf{T}} \cdots G_L^{\mathsf{T}} D,}$$

where D is diagonal with entries ± 1, every G_ℓ acts on two coordinates, and

$$\boxed{L \le \frac{d(d-1)}{2}.}$$

Proof. Apply standard Givens elimination below the diagonal. At most one rotation is required for every strictly lower-triangular entry. Orthogonality forces the resulting upper-triangular matrix to be diagonal with signs. Reverse the elimination sequence. □

For the 8×8 example in the release, the compiler uses 28 rotations, which meets the triangular bound exactly.

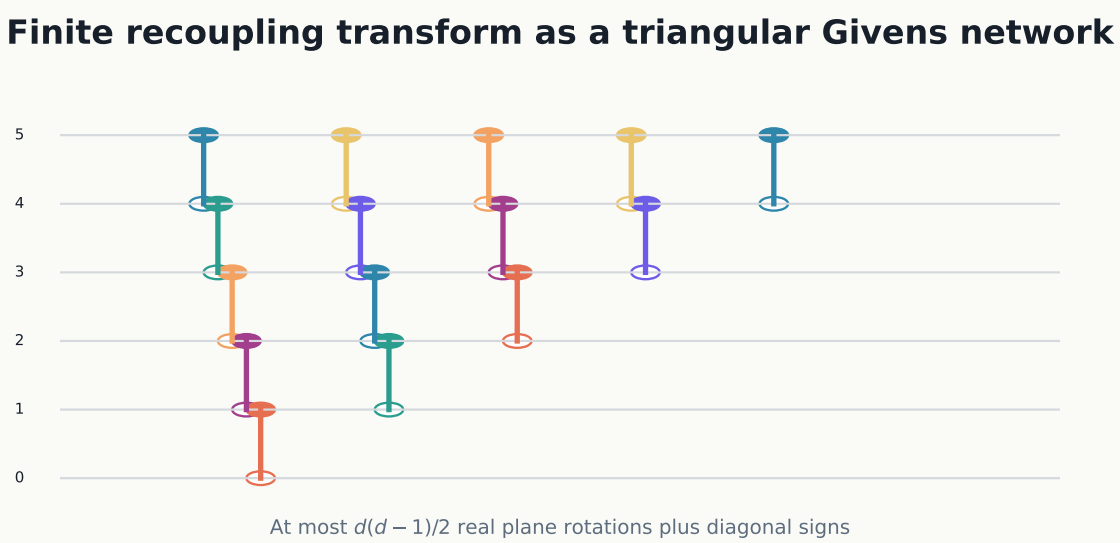


Figure 15.1: Triangular Givens network. Each vertical segment is a two-coordinate rotation. The diagonal signs finish the exact finite synthesis.

Chapter 16

Output-sensitive recoupling algorithm

The computational route is:

Algorithm 16.1 (Projector-resolved Racah transform)**.** Given positive shapes $\boldsymbol{\kappa}$ and total excitation N:

1. Enumerate the weak compositions of N.
2. Build the total lowering operator from the ladder coefficients.
3. Compute an orthonormal frame of its kernel.
4. Restrict the required pair Casimirs to that frame.
5. Diagonalize one or more commuting Casimirs for each coupling tree.
6. Form overlap matrices between the resulting bases.
7. Use the signed overlaps for coherent synthesis or square them for measurement channels.
8. Compile any selected overlap into Givens rotations when a finite circuit is required.

The full tensor level is needed only in steps 1 through 3. A symbolic or recursive implementation can avoid even that materialization by constructing Clebsch–Gordan packets directly.

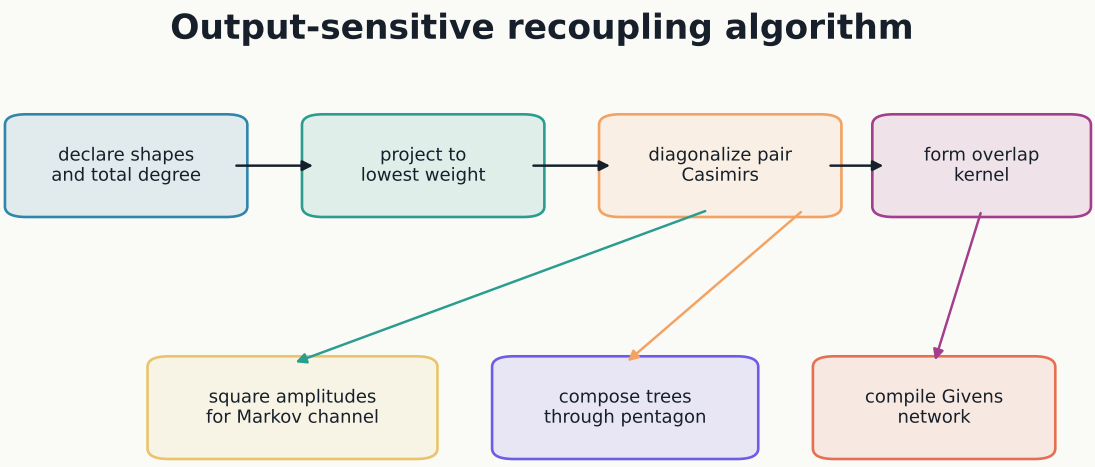


Figure 16.1: Output-sensitive recoupling algorithm. Project first, then diagonalize and recouple inside the multiplicity space. Dense full-tensor operators are not preserved as an output requirement.

Chapter 17

Scaling and numerical conditioning

For three packets, the multiplicity dimension is $N+1$. For four packets, it is $\binom{N+2}{2}$. Generic dense eigendecomposition in the reduced space scales cubically in that dimension.

The classical three-term recurrence provides a faster route for large N. Once the Jacobi coefficients are known, selected Racah columns can be evaluated in linear time. Fast transforms and structured factorizations remain an open algorithmic direction.

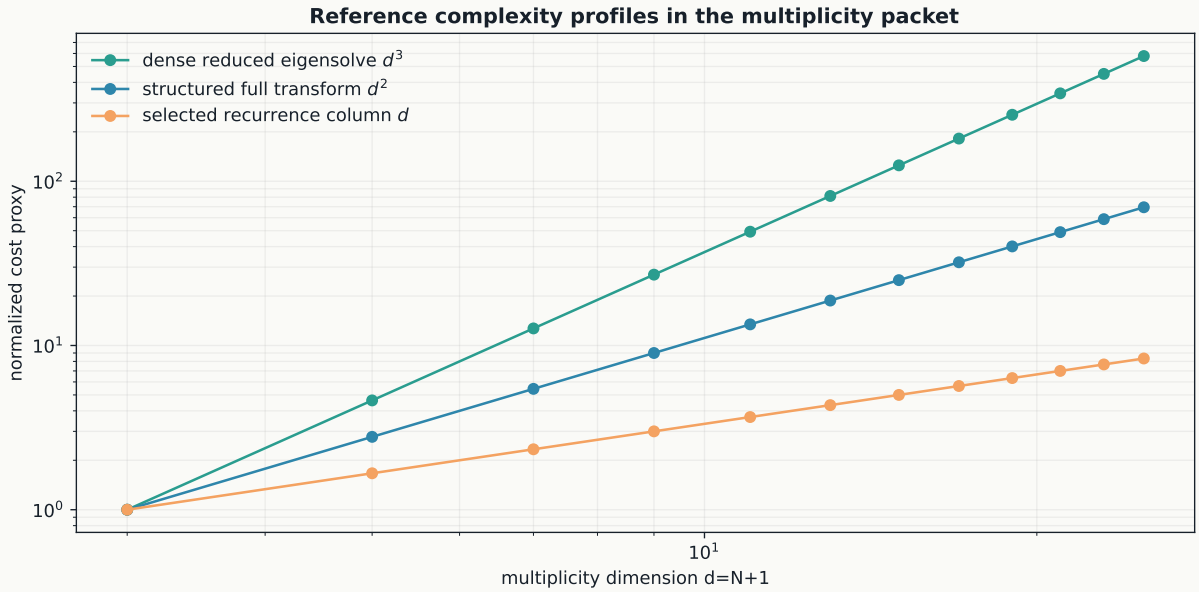


Figure 17.1: Reference complexity profiles. The deterministic curves compare a dense reduced eigensolve, a structured full transform, and one recurrence-generated column after normalization at the smallest displayed dimension.

The dimension reduction shown in figure 2.3 is the first source of computational savings.

Part VI

Classical identification, quantum use, and future deformations

Chapter 18

Classical Racah and Wilson coordinates

The polynomial hierarchy is classical:

$$\text{Meixner–Pollaczek} \longrightarrow \text{Hahn coupling coefficients} \longrightarrow \text{Racah recoupling coefficients.}$$

Racah polynomials are finite ${}_4F_3$ functions. In the standard notation,

$$R_n(\lambda(x) \mid \alpha,\beta,\gamma,\delta) = {}_4F_3\left(\left.\begin{matrix}-n, n+\alpha+\beta+1, -x, x+\gamma+\delta+1 \\ \alpha+1, \beta+\delta+1, \gamma+1\end{matrix}\right| 1\right),$$

with

$$\lambda(x) = x(x+\gamma+\delta+1)$$

and a terminating parameter that restricts the support.

The representation shapes determine the Racah parameters after the conventional normalization of the $6j$ symbols. We define the functions intrinsically through Casimir projectors because that form extends directly to multiple coupling trees and numerical packet construction.

Continuous-series recoupling replaces finite Racah polynomials by Wilson polynomials and Wilson functions. A q-deformation leads to q-Racah and Askey–Wilson structures. Higher tensor products produce multivariate Racah functions and general $3nj$ coefficients.

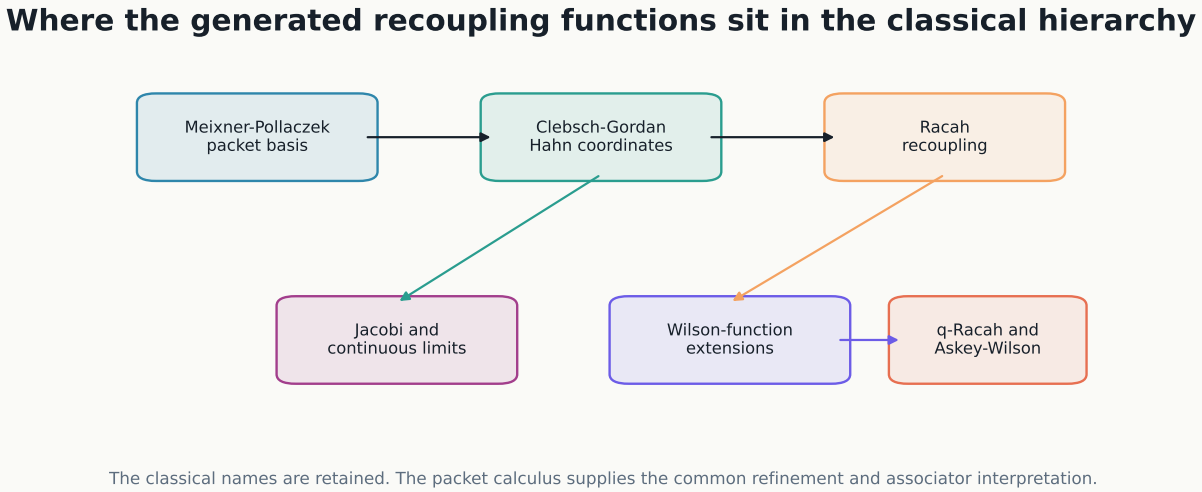


Figure 18.1: Classical-family map. The generated packet language retains the classical names and adds a common analysis, synthesis, refinement, and associator interpretation.

Chapter 19

Relation to the logarithmic quantum coordinate

The attached 2026 zeta-governed quantum construction uses the number basis and the Hamiltonian

$$H\,|n\rangle = \log n\,|n\rangle\,.$$

The resulting phase is

$$\mathrm{e}^{-\mathrm{i}tH}\,|n\rangle = n^{-\mathrm{i}t}\,|n\rangle\,.$$

The shape module in the present paper is different. Its basis labels the positive discrete-series packet degree and its finite recoupling matrix changes the bracketing of shape factors. A composite model may therefore use

$$\boxed{\text{logarithmic phase evolution} \otimes \text{finite shape recoupling.}}$$

The two operators commute when they act on separate registers. No claim is made that the zeta zeros are eigenvalues of the Racah transform.

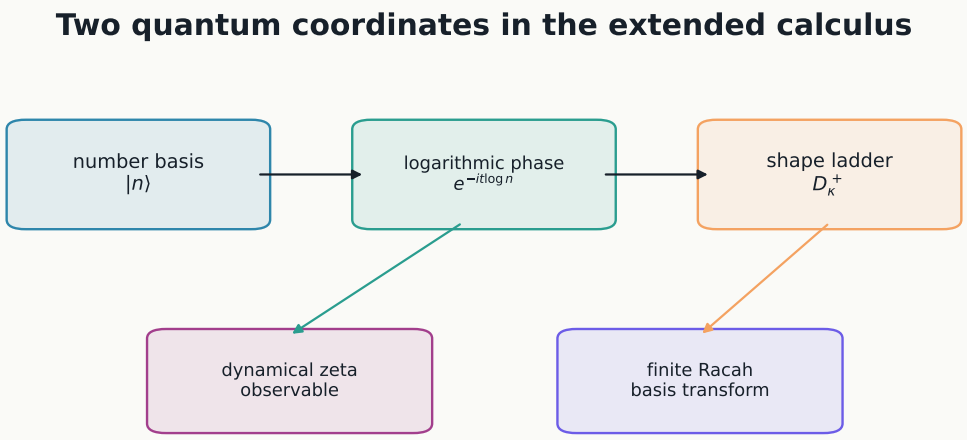


Figure 19.1: Two quantum coordinates. The logarithmic Hamiltonian acts on the number register, while the Racah transform acts on a finite shape-coupling register. Their roles remain distinct.

Chapter 20

What the packet calculus adds

The classical recoupling coefficient becomes a more general coordinate object when placed inside the calculus. The complete data are

$$\boxed{\Big(\mathcal{M}_N, \{\mathcal{C}_A\}, \{B_T\}, \{U_{T,T'}\}, \{|U_{T,T'}|^2\}, \text{refinement graph}, \text{certificate}\Big)\,.}$$

This package supports several outputs from one computation:

- coherent basis transformation
- a classical measurement channel
- entropy and uncertainty bounds
- route-independent tree recoupling
- finite circuit compilation
- reduced-space numerical algorithms

The recoupling entries are not renamed as new scalar functions. The new object is the typed packet system that keeps all these roles compatible.

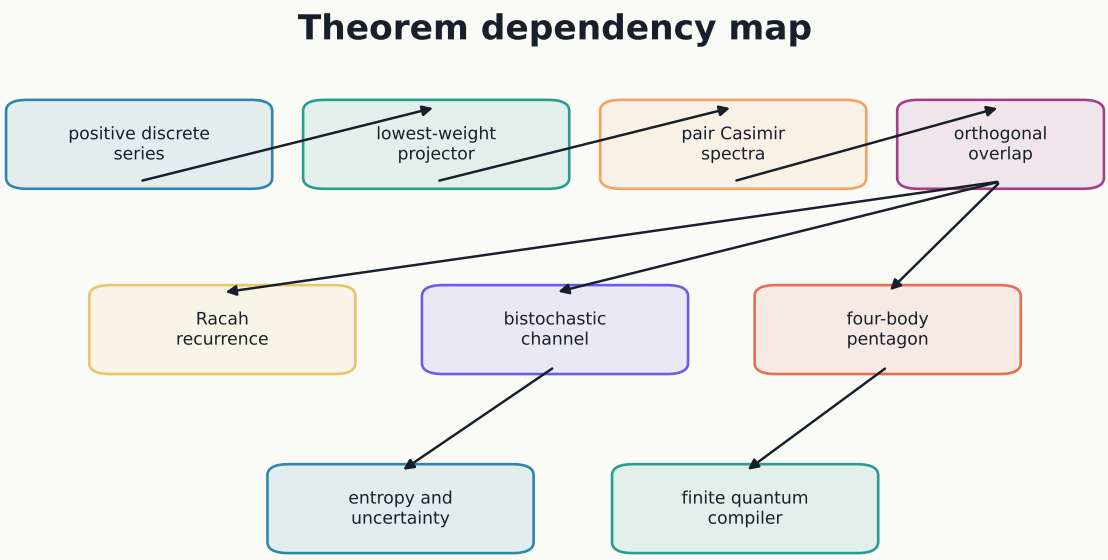


Figure 20.1: Theorem dependency map. Pair-Casimir projectors generate the orthogonal associator. Its signed and squared forms then support Racah recurrences, stochastic channels, pentagon coherence, and finite compilation.

Chapter 21

Open problems generated by the construction

The following questions are concrete continuations.

1. Develop fast transforms for large finite Racah packet matrices with provable complexity below dense quadratic application.
2. Derive stable packet recursion across both excitation N and continuous shapes $\boldsymbol{\kappa}$.
3. Establish sharp coherence and entropy bounds uniform in the packet labels.
4. Construct multivariate Racah packet bases directly from the full tree rather than repeated pair transforms.
5. Extend the projector construction to the principal and complementary series, where Wilson functions replace finite Racah polynomials.
6. Develop the q-deformed packet calculus and compare the resulting associator with quantum-group circuit models.
7. Connect packet resultants and Casimir spectra to exact denominator and arithmetic properties at rational shape parameters.

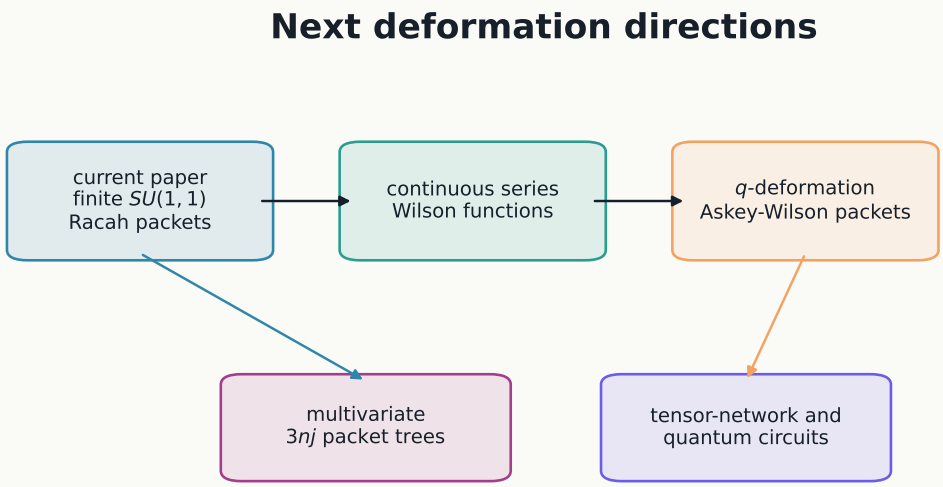


Figure 21.1: Next deformation directions. Continuous, q-deformed, multivariate, and quantum-circuit extensions retain the same basic rule: declare a complete coupling atlas, compute the associator, and certify reconstruction.

Chapter 22

Conclusion

The shape parameter introduced in Paper VI carries more structure than a convolution coordinate. Its positive discrete-series representation can be coupled, projected, and recoupled. At fixed total excitation, the new lowest-weight multiplicity space is finite. Pair Casimirs provide complete coordinate atlases, and their overlap is a Racah packet transform.

The signed transform is orthogonal. Its squared entries form a doubly stochastic channel. Four-packet coupling trees satisfy an exact pentagon closure. Every finite transform admits a Givens circuit. These facts arise from one projector-resolved object rather than from separate calculations.

The classical Racah and $6j$ theory supplies the scalar special functions. The calculus supplies the common language of packets, projectors, stochastic shadows, tree refinement, and exact reconstruction.

Appendix A

Formula registry

Object	Exact formula or property
Positive discrete-series Casimir	$\mathcal{C} = \kappa(\kappa-1)\mathrm{I}$
Excitation dimension	$\binom{N+m-1}{m-1}$
Multiplicity dimension	$\binom{N+m-2}{m-2}$
Pair-Casimir eigenvalue	$(\kappa_a+\kappa_b+j)(\kappa_a+\kappa_b+j-1)$
Racah packet	$\mathcal{R}_N(j,m) = \langle j,(12)3,m,1(23)\rangle$
Orthogonality	$\mathcal{R}_N\mathcal{R}_N^{\mathsf{T}} = \mathrm{I}$
Bistochastic kernel	$P_{jm} = \lvert\mathcal{R}_N(j,m)\rvert^2$
Entropy law	$\mathrm{H}(pP) \geq \mathrm{H}(p)$
Four-body multiplicity	$\binom{N+2}{2}$
Pentagon product	$U_{01}U_{12}U_{23}U_{34}U_{40} = \mathrm{I}$
Givens count	at most $d(d-1)/2$ rotations

Appendix B

Numerical certificate tables

B.1 Three-body metrics

N	dimension	largest $\lvert R_{jm}\rvert$	mean row entropy	tridiagonal residual
1	2	0.90830	0.46373	0
2	3	0.81922	0.86600	1.0×10^{-16}
4	5	0.68308	1.20209	1.5×10^{-14}
7	8	0.63965	1.62054	4.0×10^{-14}
12	13	0.57135	2.05931	1.2×10^{-13}

B.2 Four-body pentagon metrics

N	multiplicity dimension	pentagon residual	largest joint-diagonalization residual
1	3	1.44×10^{-15}	4.41×10^{-15}
2	6	1.67×10^{-15}	1.18×10^{-14}
3	10	3.11×10^{-15}	1.61×10^{-13}
4	15	3.44×10^{-15}	6.58×10^{-13}
7	36	3.49×10^{-15}	2.65×10^{-12}

Appendix C

Verification ledger

The built-in program checks the following statements.

Certificate	Result	Largest recorded residual
Pair-Casimir spectra	pass	2.84×10^{-14}
Three-body orthogonality	pass	1.33×10^{-15}
One-excitation closed form	pass	2.78×10^{-16}
Two-excitation exact matrix	pass	6.11×10^{-16}
Racah tridiagonal selection	pass	3.97×10^{-14}
Bistochastic kernel	pass	1.33×10^{-15}
Entropy monotonicity	pass	minimum gap 2.18×10^{-2}
Multiplicity dimension	pass	exact
Pentagon coherence	pass	3.44×10^{-15}
Tree Casimir commutation	pass	2.84×10^{-13}
Joint diagonalization	pass	6.58×10^{-13}
Givens reconstruction	pass	9.99×10^{-16}
Equal-shape probability symmetry	pass	1.33×10^{-15}

The independent test file contains 20 additional tests. It rechecks dimensions, nullspace relations, several shape choices, exact low-degree formulas, bistochasticity, entropy, pentagon closure, orthogonality of every edge transform, and source artifacts.

Appendix D

Complete Python implementation

The following file generates every figure, table, built-in certificate, and release manifest used in the paper.

```
from __future__ import annotations

import csv
import hashlib
import json
import math
from dataclasses import dataclass
from functools import lru_cache
from itertools import product
from pathlib import Path
from typing import Iterable, Sequence

import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import Circle, FancyArrowPatch, FancyBboxPatch
import numpy as np
import sympy as sp
from scipy.linalg import null_space

ROOT = Path(__file__).resolve().parent
FIGURES = ROOT / "figures"
RESULTS = ROOT / "results"
TABLES = ROOT / "tables"
GENERATED = ROOT / "generated_tables"
for directory in (FIGURES, RESULTS, TABLES, GENERATED):
    directory.mkdir(parents=True, exist_ok=True)

INK = "#17202A"
BLUE = "#2E86AB"
TEAL = "#2A9D8F"
ORANGE = "#F4A261"
MAGENTA = "#A23E8C"
RED = "#E76F51"
YELLOW = "#E9C46A"
PURPLE = "#6C5CE7"
MUTED = "#5D6D7E"
PAPER = "#FAFAF7"
GRID = "#D5D8DC"
WHITE = "#FFFFFF"
COLORS = [BLUE, TEAL, ORANGE, MAGENTA, RED, YELLOW, PURPLE]

plt.rcParams.update(
    {
        "figure.facecolor": PAPER,
        "axes.facecolor": PAPER,
        "axes.edgecolor": INK,
        "axes.labelcolor": INK,
```

```
        "xtick.color": INK,
        "ytick.color": INK,
        "text.color": INK,
        "font.size": 10.5,
        "axes.titlesize": 12.2,
        "axes.titleweight": "bold",
        "legend.frameon": False,
        "savefig.bbox": "tight",
        "savefig.facecolor": PAPER,
    }
)


def save_figure(fig: plt.Figure, stem: str) -> None:
    fig.savefig(FIGURES / f"{stem}.pdf")
    fig.savefig(FIGURES / f"{stem}.png", dpi=220)
    plt.close(fig)


def diagram_axes(figsize: tuple[float, float] = (12.6, 6.1)) -> tuple[plt.Figure, plt.Axes]:
    fig, ax = plt.subplots(figsize=figsize)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.axis("off")
    return fig, ax


def box(
    ax: plt.Axes,
    xy: tuple[float, float],
    width: float,
    height: float,
    text: str,
    face: str = WHITE,
    edge: str = INK,
    fontsize: float = 10.5,
) -> None:
    patch = FancyBboxPatch(
        xy,
        width,
        height,
        boxstyle="round,pad=.012,rounding_size=.018",
        facecolor=face,
        edgecolor=edge,
        linewidth=1.6,
    )
    ax.add_patch(patch)
    ax.text(xy[0] + width / 2, xy[1] + height / 2, text, ha="center", va="center", fontsize=fontsize)


def arrow(
    ax: plt.Axes,
    start: tuple[float, float],
    end: tuple[float, float],
    color: str = INK,
    text: str | None = None,
    text_offset: tuple[float, float] = (0.0, 0.035),
) -> None:
    patch = FancyArrowPatch(start, end, arrowstyle="-|>", mutation_scale=13, linewidth=1.5, color=color)
    ax.add_patch(patch)
    if text:
        ax.text(
            (start[0] + end[0]) / 2 + text_offset[0],
            (start[1] + end[1]) / 2 + text_offset[1],
            text,
            ha="center",
            va="center",
            fontsize=9.5,
            color=color,
        )

```

```

# ---------------------------------------------------------------------------
# Positive discrete-series packet spaces
# ---------------------------------------------------------------------------


@lru_cache(maxsize=None)
def compositions(total: int, parts: int) -> tuple[tuple[int, ...], ...]:
    if total < 0:
        return tuple()
    if parts == 1:
        return ((total,),)
    out: list[tuple[int, ...]] = []
    for first in range(total + 1):
        for rest in compositions(total - first, parts - 1):
            out.append((first,) + rest)
    return tuple(out)


def multiplicity_dimension(total: int, factors: int) -> int:
    if factors < 2:
        return 1
    return math.comb(total + factors - 2, factors - 2)


def excitation_dimension(total: int, factors: int) -> int:
    return math.comb(total + factors - 1, factors - 1)


def lowering_matrix(total: int, kappas: Sequence[float], subset: Iterable[int] | None = None) -> np.ndarray:
    m = len(kappas)
    active = set(range(m)) if subset is None else set(subset)
    basis_n = compositions(total, m)
    if total == 0:
        return np.zeros((0, len(basis_n)), dtype=float)
    basis_prev = compositions(total - 1, m)
    index_prev = {state: i for i, state in enumerate(basis_prev)}
    matrix = np.zeros((len(basis_prev), len(basis_n)), dtype=float)
    for col, state in enumerate(basis_n):
        for i in active:
            occupation = state[i]
            if occupation == 0:
                continue
            target = list(state)
            target[i] -= 1
            coefficient = math.sqrt(occupation * (occupation + 2.0 * kappas[i] - 1.0))
            matrix[index_prev[tuple(target)], col] += coefficient
    return matrix


def raising_matrix(total: int, kappas: Sequence[float], subset: Iterable[int] | None = None) -> np.ndarray:
    return lowering_matrix(total + 1, kappas, subset).T


def casimir_matrix(total: int, kappas: Sequence[float], subset: Iterable[int]) -> np.ndarray:
    active = set(subset)
    basis = compositions(total, len(kappas))
    k0 = np.array([sum(state[i] + kappas[i] for i in active) for state in basis], dtype=float)
    lowering = lowering_matrix(total, kappas, active)
    return np.diag(k0 * (k0 - 1.0)) - lowering.T @ lowering


def multiplicity_frame(total: int, kappas: Sequence[float], atol: float = 1e-11) -> np.ndarray:
    lowering = lowering_matrix(total, kappas)
    if total == 0:
        return np.ones((1, 1), dtype=float)
    frame = null_space(lowering, rcond=atol)
    return frame


def sign_fix_columns(matrix: np.ndarray) -> np.ndarray:
```

```
    result = np.array(matrix, dtype=float, copy=True)
    for column in range(result.shape[1]):
        row = int(np.argmax(np.abs(result[:, column])))
        if result[row, column] < 0:
            result[:, column] *= -1.0
    return result


def canonicalize_overlap(matrix: np.ndarray) -> np.ndarray:
    result = np.array(matrix, dtype=float, copy=True)
    for column in range(result.shape[1]):
        if result[0, column] < 0:
            result[:, column] *= -1.0
    for row in range(result.shape[0]):
        if result[row, 0] < 0:
            result[row, :] *= -1.0
    return result


@dataclass
class RecouplingData:
    total: int
    kappas: tuple[float, float, float]
    multiplicity_frame: np.ndarray
    left_eigenvalues: np.ndarray
    right_eigenvalues: np.ndarray
    left_basis: np.ndarray
    right_basis: np.ndarray
    overlap_raw: np.ndarray
    overlap: np.ndarray
    right_in_left: np.ndarray


def three_body_recoupling(total: int, kappas: Sequence[float]) -> RecouplingData:
    if len(kappas) != 3:
        raise ValueError("three_body_recoupling requires three shapes")
    kappas_tuple = tuple(float(value) for value in kappas)
    frame = multiplicity_frame(total, kappas_tuple)
    c12 = frame.T @ casimir_matrix(total, kappas_tuple, (0, 1)) @ frame
    c23 = frame.T @ casimir_matrix(total, kappas_tuple, (1, 2)) @ frame
    left_values, left_vectors = np.linalg.eigh(c12)
    right_values, right_vectors = np.linalg.eigh(c23)
    left_vectors = sign_fix_columns(left_vectors)
    right_vectors = sign_fix_columns(right_vectors)
    left_basis = frame @ left_vectors
    right_basis = frame @ right_vectors
    raw = left_basis.T @ right_basis
    overlap = canonicalize_overlap(raw)
    right_in_left = overlap @ np.diag(right_values) @ overlap.T
    return RecouplingData(
        total=total,
        kappas=kappas_tuple,
        multiplicity_frame=frame,
        left_eigenvalues=left_values,
        right_eigenvalues=right_values,
        left_basis=left_basis,
        right_basis=right_basis,
        overlap_raw=raw,
        overlap=overlap,
        right_in_left=right_in_left,
    )


def theoretical_pair_spectrum(total: int, kappa_a: float, kappa_b: float) -> np.ndarray:
    j = np.arange(total + 1, dtype=float)
    shape = kappa_a + kappa_b + j
    return shape * (shape - 1.0)


def one_excitation_closed_form(kappas: Sequence[float]) -> np.ndarray:
    k1, k2, k3 = map(float, kappas)
```

```
    denominator = (k1 + k2) * (k2 + k3)
    cosine = math.sqrt(k1 * k3 / denominator)
    sine = math.sqrt(k2 * (k1 + k2 + k3) / denominator)
    return np.array([[cosine, sine], [sine, -cosine]], dtype=float)


def equal_shape_two_excitation_matrix() -> np.ndarray:
    return np.array(
        [
            [1.0 / 3.0, 1.0 / math.sqrt(3.0), math.sqrt(5.0) / 3.0],
            [1.0 / math.sqrt(3.0), 1.0 / 2.0, -math.sqrt(15.0) / 6.0],
            [math.sqrt(5.0) / 3.0, -math.sqrt(15.0) / 6.0, 1.0 / 6.0],
        ],
        dtype=float,
    )


def tridiagonal_residual(matrix: np.ndarray) -> float:
    residual = np.array(matrix, copy=True)
    rows, cols = residual.shape
    for row in range(rows):
        for col in range(cols):
            if abs(row - col) <= 1:
                residual[row, col] = 0.0
    return float(np.max(np.abs(residual)))


def row_entropies(probability_matrix: np.ndarray) -> np.ndarray:
    safe = np.where(probability_matrix > 0, probability_matrix, 1.0)
    return -np.sum(np.where(probability_matrix > 0, probability_matrix * np.log(probability_matrix), 0.0), axis=1)


def doubly_stochastic_residual(matrix: np.ndarray) -> float:
    return float(
        max(
            np.max(np.abs(np.sum(matrix, axis=0) - 1.0)),
            np.max(np.abs(np.sum(matrix, axis=1) - 1.0)),
            np.max(np.maximum(-matrix, 0.0)),
        )
    )


# ----------------------------------------------------------------------------
# Four-body coupling trees and pentagon coherence
# ----------------------------------------------------------------------------


FOUR_TREES: tuple[tuple[str, tuple[tuple[int, ...], tuple[int, ...]]], ...] = (
    ("(((12)3)4)", ((0, 1), (0, 1, 2))),
    ("((1(23))4)", ((1, 2), (0, 1, 2))),
    ("(1((23)4))", ((1, 2), (1, 2, 3))),
    ("(1(2(34)))", ((2, 3), (1, 2, 3))),
    ("((12)(34))", ((0, 1), (2, 3))),
)


@dataclass
class TreeBasisData:
    name: str
    subsets: tuple[tuple[int, ...], ...]
    basis: np.ndarray
    joint_diagonals: tuple[np.ndarray, ...]
    commutator_residual: float
    diagonalization_residual: float


def joint_tree_basis(total: int, kappas: Sequence[float], name: str, subsets: Sequence[Sequence[int]]) ->
     TreeBasisData:
    frame = multiplicity_frame(total, kappas)
    restricted = [frame.T @ casimir_matrix(total, kappas, subset) @ frame for subset in subsets]
    commutator = 0.0
```

```
    for i in range(len(restricted)):
        for j in range(i + 1, len(restricted)):
            commutator = max(commutator, float(np.max(np.abs(restricted[i] @ restricted[j] - restricted[j] @ restricted[i]))))
    irrational_weights = [math.sqrt(2.0), math.sqrt(3.0), math.sqrt(5.0), math.sqrt(7.0)]
    combined = sum(weight * matrix for weight, matrix in zip(irrational_weights, restricted))
    _, vectors = np.linalg.eigh(combined)
    vectors = sign_fix_columns(vectors)
    diagonals = tuple(vectors.T @ matrix @ vectors for matrix in restricted)
    residual = max(float(np.max(np.abs(diagonal - np.diag(np.diag(diagonal))))) for diagonal in diagonals)
    tuples = []
    for column in range(vectors.shape[1]):
        key = tuple(round(float(diagonal[column, column]), 10) for diagonal in diagonals)
        tuples.append((key, column))
    order = [column for _, column in sorted(tuples, key=lambda item: item[0])]
    vectors = vectors[:, order]
    diagonals = tuple(vectors.T @ matrix @ vectors for matrix in restricted)
    residual = max(float(np.max(np.abs(diagonal - np.diag(np.diag(diagonal))))) for diagonal in diagonals)
    basis = frame @ vectors
    return TreeBasisData(
        name=name,
        subsets=tuple(tuple(int(index) for index in subset) for subset in subsets),
        basis=basis,
        joint_diagonals=diagonals,
        commutator_residual=commutator,
        diagonalization_residual=residual,
    )


def four_body_tree_bases(total: int, kappas: Sequence[float]) -> list[TreeBasisData]:
    if len(kappas) != 4:
        raise ValueError("four_body_tree_bases requires four shapes")
    return [joint_tree_basis(total, kappas, name, subsets) for name, subsets in FOUR_TREES]


def pentagon_product(tree_bases: Sequence[TreeBasisData]) -> tuple[np.ndarray, list[np.ndarray]]:
    overlaps: list[np.ndarray] = []
    dimension = tree_bases[0].basis.shape[1]
    product_matrix = np.eye(dimension)
    for index in range(len(tree_bases)):
        left = tree_bases[index].basis
        right = tree_bases[(index + 1) % len(tree_bases)].basis
        overlap = left.T @ right
        overlaps.append(overlap)
        product_matrix = product_matrix @ overlap
    return product_matrix, overlaps


# ---------------------------------------------------------------------------
# Givens synthesis and finite quantum transforms
# ---------------------------------------------------------------------------


@dataclass
class GivensRotation:
    upper: int
    lower: int
    cosine: float
    sine: float


def givens_decompose(orthogonal: np.ndarray, atol: float = 1e-14) -> tuple[list[GivensRotation], np.ndarray]:
    matrix = np.array(orthogonal, dtype=float, copy=True)
    dimension = matrix.shape[0]
    rotations: list[GivensRotation] = []
    for column in range(dimension):
        for lower in range(dimension - 1, column, -1):
            upper = lower - 1
            a = matrix[upper, column]
            b = matrix[lower, column]
            radius = math.hypot(a, b)
```

```
            if radius < atol:
                continue
            cosine = a / radius
            sine = b / radius
            row_upper = cosine * matrix[upper, :] + sine * matrix[lower, :]
            row_lower = -sine * matrix[upper, :] + cosine * matrix[lower, :]
            matrix[upper, :] = row_upper
            matrix[lower, :] = row_lower
            rotations.append(GivensRotation(upper, lower, cosine, sine))
    diagonal = np.diag(np.diag(matrix))
    return rotations, diagonal


def givens_reconstruct(rotations: Sequence[GivensRotation], diagonal: np.ndarray) -> np.ndarray:
    matrix = np.array(diagonal, dtype=float, copy=True)
    for rotation in reversed(rotations):
        upper = rotation.upper
        lower = rotation.lower
        cosine = rotation.cosine
        sine = rotation.sine
        row_upper = cosine * matrix[upper, :] - sine * matrix[lower, :]
        row_lower = sine * matrix[upper, :] + cosine * matrix[lower, :]
        matrix[upper, :] = row_upper
        matrix[lower, :] = row_lower
    return matrix


# ---------------------------------------------------------------------------
# Checks
# ---------------------------------------------------------------------------


def run_checks() -> dict[str, dict[str, float | int | str | bool]]:
    checks: dict[str, dict[str, float | int | str | bool]] = {}

    kappas = (0.7, 1.1, 0.9)
    data = three_body_recoupling(7, kappas)
    expected_left = theoretical_pair_spectrum(7, kappas[0], kappas[1])
    expected_right = theoretical_pair_spectrum(7, kappas[1], kappas[2])
    spectrum_error = max(
        float(np.max(np.abs(data.left_eigenvalues - expected_left))),
        float(np.max(np.abs(data.right_eigenvalues - expected_right))),
    )
    checks["pair_casimir_spectrum"] = {"passed": spectrum_error < 2e-10, "residual": spectrum_error}

    orthogonality_error = float(np.max(np.abs(data.overlap @ data.overlap.T - np.eye(8))))
    checks["three_body_orthogonality"] = {"passed": orthogonality_error < 2e-10, "residual": orthogonality_error}

    n1_numeric = three_body_recoupling(1, kappas).overlap
    n1_exact = one_excitation_closed_form(kappas)
    n1_error = float(np.max(np.abs(n1_numeric - n1_exact)))
    checks["one_excitation_closed_form"] = {"passed": n1_error < 2e-10, "residual": n1_error}

    n2_numeric = three_body_recoupling(2, (0.5, 0.5, 0.5)).overlap
    n2_exact = equal_shape_two_excitation_matrix()
    n2_error = float(np.max(np.abs(n2_numeric - n2_exact)))
    checks["two_excitation_exact_matrix"] = {"passed": n2_error < 2e-10, "residual": n2_error}

    tri_error = tridiagonal_residual(data.right_in_left)
    checks["racah_tridiagonal_selection"] = {"passed": tri_error < 2e-10, "residual": tri_error}

    probability = data.overlap ** 2
    stochastic_error = doubly_stochastic_residual(probability)
    checks["bistochastic_kernel"] = {"passed": stochastic_error < 2e-10, "residual": stochastic_error}

    rng = np.random.default_rng(1007)
    entropy_gaps = []
    for _ in range(100):
        raw = rng.random(probability.shape[0])
        distribution = raw / raw.sum()
        output = distribution @ probability
```

```
        input_entropy = -float(np.sum(distribution * np.log(distribution)))
        output_entropy = -float(np.sum(output * np.log(np.maximum(output, 1e-300))))
        entropy_gaps.append(output_entropy - input_entropy)
    minimum_gap = float(min(entropy_gaps))
    checks["entropy_monotonicity"] = {"passed": minimum_gap > -2e-12, "minimum_gap": minimum_gap}

    dimensions_ok = True
    maximum_dimension_error = 0
    for factors in (3, 4, 5):
        for total in range(0, 8):
            frame = multiplicity_frame(total, [0.5 + 0.1 * index for index in range(factors)])
            expected = multiplicity_dimension(total, factors)
            error = abs(frame.shape[1] - expected)
            maximum_dimension_error = max(maximum_dimension_error, error)
            dimensions_ok = dimensions_ok and error == 0
    checks["multiplicity_dimension"] = {"passed": dimensions_ok, "maximum_error": maximum_dimension_error}

    four_kappas = (0.55, 0.8, 1.05, 1.3)
    trees = four_body_tree_bases(4, four_kappas)
    product_matrix, overlaps = pentagon_product(trees)
    pentagon_error = float(np.max(np.abs(product_matrix - np.eye(product_matrix.shape[0]))))
    checks["pentagon_coherence"] = {"passed": pentagon_error < 3e-10, "residual": pentagon_error}

    tree_commutator = max(tree.commutator_residual for tree in trees)
    tree_diagonal = max(tree.diagonalization_residual for tree in trees)
    checks["tree_casimir_commutation"] = {"passed": tree_commutator < 3e-10, "residual": tree_commutator}
    checks["tree_joint_diagonalization"] = {"passed": tree_diagonal < 3e-10, "residual": tree_diagonal}

    rotations, diagonal = givens_decompose(data.overlap)
    reconstructed = givens_reconstruct(rotations, diagonal)
    givens_error = float(np.max(np.abs(reconstructed - data.overlap)))
    checks["givens_reconstruction"] = {
        "passed": givens_error < 3e-10,
        "residual": givens_error,
        "rotation_count": len(rotations),
    }

    equal_probability = three_body_recoupling(8, (0.5, 0.5, 0.5)).overlap ** 2
    symmetry_error = float(np.max(np.abs(equal_probability - equal_probability.T)))
    checks["equal_shape_probability_symmetry"] = {"passed": symmetry_error < 3e-10, "residual": symmetry_error}

    passed = sum(1 for result in checks.values() if bool(result["passed"]))
    summary = {
        "checks": checks,
        "passed": passed,
        "total": len(checks),
        "all_passed": passed == len(checks),
    }
    (RESULTS / "checks.json").write_text(json.dumps(summary, indent=2))
    return checks


# ---------------------------------------------------------------------------
# Figures
# ---------------------------------------------------------------------------


def generate_figures() -> list[str]:
    generated: list[str] = []

    # 01. Theory pipeline
    fig, ax = diagram_axes((13.0, 6.3))
    ax.text(0.5, 0.94, "From shape refinement to Racah recoupling", ha="center", fontsize=21, weight="bold")
    stages = [
        (0.04, 0.58, 0.15, 0.17, "Meixner shape\npacket", BLUE),
        (0.23, 0.58, 0.15, 0.17, "positive discrete\nseries", TEAL),
        (0.42, 0.58, 0.15, 0.17, "lowest-weight\nmultiplicity", ORANGE),
        (0.61, 0.58, 0.15, 0.17, "pair-Casimir\nbases", MAGENTA),
        (0.80, 0.58, 0.16, 0.17, "Racah packet\ntransform", RED),
    ]
    for x, y, w, h, label, color in stages:
```

```
        box(ax, (x, y), w, h, label, face=color + "1C", edge=color, fontsize=11.5)
    for left, right in zip(stages[:-1], stages[1:]):
        arrow(ax, (left[0] + left[2], left[1] + left[3] / 2), (right[0], right[1] + right[3] / 2), color=INK)
    box(ax, (0.18, 0.19), 0.22, 0.16, "bistochastic\nmeasurement channel", face=YELLOW + "22", edge=YELLOW, fontsize=11.5)
    box(ax, (0.46, 0.19), 0.22, 0.16, "pentagon\ncoherence", face=PURPLE + "1C", edge=PURPLE, fontsize=11.5)
    box(ax, (0.74, 0.19), 0.18, 0.16, "finite quantum\nbasis circuit", face=BLUE + "1C", edge=BLUE, fontsize=11.5)
    arrow(ax, (0.88, 0.58), (0.83, 0.35), color=RED)
    arrow(ax, (0.76, 0.58), (0.57, 0.35), color=MAGENTA)
    arrow(ax, (0.67, 0.58), (0.29, 0.35), color=ORANGE)
    ax.text(0.5, 0.07, "Different refinement trees describe the same total packet space and are connected by exact orthogonal transforms.", ha="center", fontsize=11.5, color=MUTED)
    save_figure(fig, "01_theory_pipeline")
    generated.append("01_theory_pipeline")

    # 02. su(1,1) ladder coefficients
    kappa = 0.8
    n = np.arange(0, 14)
    raise_coeff = np.sqrt((n + 1) * (n + 2 * kappa))
    lower_coeff = np.sqrt(n * np.maximum(n + 2 * kappa - 1, 0))
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    ax.plot(n, raise_coeff, marker="o", color=BLUE, label=r"$\sqrt{(n+1)(n+2\kappa)}$")
    ax.plot(n, lower_coeff, marker="s", color=ORANGE, label=r"$\sqrt{n(n+2\kappa-1)}$")
    ax.set_xlabel("packet degree n")
    ax.set_ylabel("ladder coefficient")
    ax.set_title(r"Positive discrete-series ladder at $\kappa=0.8$")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "02_su11_ladder")
    generated.append("02_su11_ladder")

    # 03. Three-factor excitation simplex
    total = 7
    states = compositions(total, 3)
    xy = np.array([[state[0] + 0.5 * state[1], math.sqrt(3) * state[1] / 2] for state in states])
    fig, ax = plt.subplots(figsize=(8.2, 7.1))
    values = np.array([state[0] - state[2] for state in states])
    scatter = ax.scatter(xy[:, 0], xy[:, 1], c=values, cmap="coolwarm", s=80, edgecolor=INK, linewidth=0.5)
    for point, state in zip(xy, states):
        ax.text(point[0], point[1] + 0.17, "".join(map(str, state)), ha="center", fontsize=7.2)
    ax.set_aspect("equal")
    ax.axis("off")
    ax.set_title(r"Total-excitation simplex $n_1+n_2+n_3=7$")
    fig.colorbar(scatter, ax=ax, fraction=0.04, pad=0.02, label=r"$n_1-n_3$")
    save_figure(fig, "03_excitation_simplex")
    generated.append("03_excitation_simplex")

    # 04. Dimension compression
    totals = np.arange(0, 21)
    fig, ax = plt.subplots(figsize=(11.8, 5.5))
    for factors, color in zip((3, 4, 5), (BLUE, TEAL, MAGENTA)):
        full = np.array([excitation_dimension(int(value), factors) for value in totals])
        multiplicity = np.array([multiplicity_dimension(int(value), factors) for value in totals])
        ax.semilogy(totals, full, color=color, linewidth=2.0, label=f"{factors} factors: full level")
        ax.semilogy(totals, multiplicity, color=color, linestyle="--", linewidth=2.0, label=f"{factors} factors: lowest-weight packet")
    ax.set_xlabel("total excitation N")
    ax.set_ylabel("dimension")
    ax.set_title("Projector reduction before recoupling")
    ax.grid(alpha=0.18, which="both")
    ax.legend(ncol=2, fontsize=9)
    save_figure(fig, "04_dimension_compression")
    generated.append("04_dimension_compression")

    # 05. Pair-Casimir spectra
    data = three_body_recoupling(10, (0.7, 1.1, 0.9))
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    index = np.arange(11)
    ax.plot(index, data.left_eigenvalues, marker="o", color=BLUE, label=r"$C_{12}$")
    ax.plot(index, data.right_eigenvalues, marker="s", color=ORANGE, label=r"$C_{23}$")
    ax.set_xlabel("intermediate packet label")
```

```
    ax.set_ylabel("Casimir eigenvalue")
    ax.set_title("Two coupling schemes have simple quadratic spectra")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "05_pair_casimir_spectra")
    generated.append("05_pair_casimir_spectra")

    # 06. Shape mixing angle
    middle = np.logspace(-2, 2, 260)
    k1, k3 = 0.7, 1.3
    cosines = np.sqrt(k1 * k3 / ((k1 + middle) * (middle + k3)))
    sines = np.sqrt(middle * (k1 + middle + k3) / ((k1 + middle) * (middle + k3)))
    fig, ax = plt.subplots(figsize=(11.8, 5.4))
    ax.semilogx(middle, cosines, color=BLUE, linewidth=2.2, label=r"$\cos\theta$")
    ax.semilogx(middle, sines, color=ORANGE, linewidth=2.2, label=r"$\sin\theta$")
    ax.axhline(1 / math.sqrt(2), color=GRID, linestyle="--", linewidth=1.2)
    ax.set_xlabel(r"middle shape $\kappa_2$")
    ax.set_ylabel("mixing amplitude")
    ax.set_title(r"One-excitation recoupling is a shape-controlled beam splitter")
    ax.grid(alpha=0.18, which="both")
    ax.legend()
    save_figure(fig, "06_shape_mixing_angle")
    generated.append("06_shape_mixing_angle")

    # 07. One-excitation matrix
    matrix = one_excitation_closed_form((0.7, 1.1, 0.9))
    fig, ax = plt.subplots(figsize=(6.8, 5.8))
    image = ax.imshow(matrix, cmap="coolwarm", vmin=-1, vmax=1)
    for i in range(2):
        for j in range(2):
            ax.text(j, i, f"{matrix[i,j]:.6f}", ha="center", va="center", fontsize=12, color=WHITE if abs(matrix[i,j
    ]) > 0.55 else INK)
    ax.set_xticks([0, 1], ["right 0", "right 1"])
    ax.set_yticks([0, 1], ["left 0", "left 1"])
    ax.set_title("Exact one-excitation recoupling matrix")
    fig.colorbar(image, ax=ax, fraction=0.05, pad=0.04)
    save_figure(fig, "07_one_excitation_matrix")
    generated.append("07_one_excitation_matrix")

    # 08. Two-excitation exact matrix
    matrix = equal_shape_two_excitation_matrix()
    fig, ax = plt.subplots(figsize=(7.2, 6.1))
    image = ax.imshow(matrix, cmap="coolwarm", vmin=-1, vmax=1)
    for i in range(3):
        for j in range(3):
            ax.text(j, i, f"{matrix[i,j]:.6f}", ha="center", va="center", fontsize=10.5, color=WHITE if abs(matrix[i,
    j]) > 0.56 else INK)
    ax.set_xticks(range(3), ["right 0", "right 1", "right 2"])
    ax.set_yticks(range(3), ["left 0", "left 1", "left 2"])
    ax.set_title(r"Equal-shape matrix at $N=2$")
    fig.colorbar(image, ax=ax, fraction=0.05, pad=0.04)
    save_figure(fig, "08_two_excitation_exact")
    generated.append("08_two_excitation_exact")

    # 09. General Racah packet heatmap
    data = three_body_recoupling(12, (0.7, 1.1, 0.9))
    fig, ax = plt.subplots(figsize=(8.3, 7.2))
    image = ax.imshow(data.overlap, cmap="coolwarm", vmin=-np.max(np.abs(data.overlap)), vmax=np.max(np.abs(data.
    overlap)), aspect="auto")
    ax.set_xlabel(r"$C_{23}$ packet label")
    ax.set_ylabel(r"$C_{12}$ packet label")
    ax.set_title("Racah packet transform at total excitation 12")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.03, label="overlap amplitude")
    save_figure(fig, "09_racah_packet_heatmap")
    generated.append("09_racah_packet_heatmap")

    # 10. Bistochastic kernel
    probability = data.overlap ** 2
    fig, ax = plt.subplots(figsize=(8.3, 7.2))
    image = ax.imshow(probability, cmap="viridis", aspect="auto")
    ax.set_xlabel(r"$C_{23}$ measurement outcome")
```

```
    ax.set_ylabel(r"$C_{12}$ input packet")
    ax.set_title("Squared recoupling amplitudes form a bistochastic kernel")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.03, label="transition probability")
    save_figure(fig, "10_bistochastic_kernel")
    generated.append("10_bistochastic_kernel")

    # 11. Orthogonality residual
    totals = np.arange(1, 23)
    residuals = []
    for value in totals:
        rec = three_body_recoupling(int(value), (0.7, 1.1, 0.9)).overlap
        residuals.append(np.max(np.abs(rec @ rec.T - np.eye(value + 1))))
    fig, ax = plt.subplots(figsize=(11.8, 5.1))
    ax.semilogy(totals, residuals, marker="o", color=TEAL)
    ax.set_xlabel("total excitation N")
    ax.set_ylabel("maximum orthogonality residual")
    ax.set_title("Numerical orthogonality remains at floating-point scale")
    ax.grid(alpha=0.18, which="both")
    save_figure(fig, "11_orthogonality_residual")
    generated.append("11_orthogonality_residual")

    # 12. Leonard tridiagonal matrix
    data = three_body_recoupling(10, (0.7, 1.1, 0.9))
    fig, ax = plt.subplots(figsize=(8.3, 7.0))
    image = ax.imshow(data.right_in_left, cmap="magma", aspect="auto")
    ax.set_xlabel(r"$C_{12}$ basis column")
    ax.set_ylabel(r"$C_{12}$ basis row")
    ax.set_title(r"$C_{23}$ is tridiagonal in the $C_{12}$ basis")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.03, label="matrix entry")
    save_figure(fig, "12_leonard_tridiagonal")
    generated.append("12_leonard_tridiagonal")

    # 13. Recurrence columns
    data = three_body_recoupling(14, (0.7, 1.1, 0.9))
    fig, ax = plt.subplots(figsize=(12.0, 5.5))
    for column, color in zip((0, 2, 5, 9, 14), COLORS[:5]):
        ax.plot(np.arange(15), data.overlap[:, column], marker="o", linewidth=1.5, color=color, label=f"right packet
      {column}")
    ax.set_xlabel(r"left packet label $r$")
    ax.set_ylabel(r"$R_{r,s}$")
    ax.set_title("Columns are finite Racah packet functions")
    ax.grid(alpha=0.18)
    ax.legend(ncol=3, fontsize=9)
    save_figure(fig, "13_racah_columns")
    generated.append("13_racah_columns")

    # 14. Maximum coherence vs N
    totals = np.arange(1, 31)
    equal_max = []
    unequal_max = []
    for value in totals:
        equal_max.append(np.max(np.abs(three_body_recoupling(int(value), (0.5, 0.5, 0.5)).overlap)))
        unequal_max.append(np.max(np.abs(three_body_recoupling(int(value), (0.7, 1.1, 0.9)).overlap)))
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    ax.plot(totals, equal_max, color=BLUE, marker="o", label="equal shapes")
    ax.plot(totals, unequal_max, color=ORANGE, marker="s", label="unequal shapes")
    ax.set_xlabel("total excitation N")
    ax.set_ylabel(r"$\max_{r,s}|R_{r,s}|$")
    ax.set_title("Coupling-basis coherence across packet depth")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "14_maximum_coherence")
    generated.append("14_maximum_coherence")

    # 15. Row entropy vs N
    entropy_mean = []
    entropy_min = []
    entropy_max = []
    for value in totals:
        probabilities = three_body_recoupling(int(value), (0.7, 1.1, 0.9)).overlap ** 2
        entropies = row_entropies(probabilities)
```

```
        entropy_mean.append(np.mean(entropies))
        entropy_min.append(np.min(entropies))
        entropy_max.append(np.max(entropies))
    fig, ax = plt.subplots(figsize=(11.8, 5.4))
    ax.fill_between(totals, entropy_min, entropy_max, color=TEAL, alpha=0.18, label="row range")
    ax.plot(totals, entropy_mean, color=TEAL, linewidth=2.2, label="mean row entropy")
    ax.plot(totals, np.log(totals + 1), color=MUTED, linestyle="--", label=r"$\log(N+1)$")
    ax.set_xlabel("total excitation N")
    ax.set_ylabel("Shannon entropy")
    ax.set_title("A basis packet becomes a spread distribution after recoupling")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "15_row_entropy")
    generated.append("15_row_entropy")

    # 16. Parameter sensitivity heatmap
    middle_values = np.geomspace(0.08, 8.0, 90)
    totals_small = np.arange(1, 11)
    surface = np.zeros((len(totals_small), len(middle_values)))
    for i, value in enumerate(totals_small):
        for j, middle_shape in enumerate(middle_values):
            surface[i, j] = np.max(np.abs(three_body_recoupling(int(value), (0.7, float(middle_shape), 1.3)).overlap)
    )
    fig, ax = plt.subplots(figsize=(12.0, 5.8))
    image = ax.imshow(surface, origin="lower", aspect="auto", extent=[math.log10(middle_values[0]), math.log10(
    middle_values[-1]), totals_small[0], totals_small[-1]], cmap="viridis")
    ticks = [-1, 0, 1]
    ax.set_xticks(ticks, [r"$10^{-1}$", r"$10^0$", r"$10^1$"])
    ax.set_xlabel(r"middle shape $\kappa_2$")
    ax.set_ylabel("total excitation N")
    ax.set_title("Shape dependence of the largest recoupling amplitude")
    fig.colorbar(image, ax=ax, fraction=0.025, pad=0.02)
    save_figure(fig, "16_parameter_sensitivity")
    generated.append("16_parameter_sensitivity")

    # 17. Bistochastic marginal check
    data = three_body_recoupling(9, (0.7, 1.1, 0.9))
    probability = data.overlap ** 2
    fig, ax = plt.subplots(figsize=(11.8, 5.1))
    ax.plot(np.arange(10), probability.sum(axis=0), marker="o", color=BLUE, label="column sums")
    ax.plot(np.arange(10), probability.sum(axis=1), marker="s", color=ORANGE, label="row sums")
    ax.axhline(1.0, color=INK, linewidth=1.2)
    ax.set_ylim(0.999999999999, 1.000000000001)
    ax.set_xlabel("packet label")
    ax.set_ylabel("marginal sum")
    ax.set_title("Both recoupling marginals equal one")
    ax.legend()
    ax.grid(alpha=0.18)
    save_figure(fig, "17_bistochastic_marginals")
    generated.append("17_bistochastic_marginals")

    # 18. Three-body basis diagram
    fig, ax = diagram_axes((12.8, 6.2))
    ax.text(0.5, 0.94, "Two exact coupling trees on the same three-packet space", ha="center", fontsize=20, weight="
    bold")
    points_left = {"1": (0.12, 0.68), "2": (0.28, 0.68), "3": (0.44, 0.68), "12": (0.20, 0.43), "123": (0.32, 0.18)}
    points_right = {"1": (0.58, 0.68), "2": (0.74, 0.68), "3": (0.90, 0.68), "23": (0.82, 0.43), "123": (0.70, 0.18)}

    for label, point in points_left.items():
        ax.add_patch(Circle(point, 0.035, facecolor=BLUE + "33", edgecolor=BLUE, linewidth=1.6))
        ax.text(*point, label, ha="center", va="center", fontsize=10)
    for label, point in points_right.items():
        ax.add_patch(Circle(point, 0.035, facecolor=ORANGE + "33", edgecolor=ORANGE, linewidth=1.6))
        ax.text(*point, label, ha="center", va="center", fontsize=10)
    for a, b in [("1", "12"), ("2", "12"), ("12", "123"), ("3", "123")]:
        ax.plot([points_left[a][0], points_left[b][0]], [points_left[a][1], points_left[b][1]], color=BLUE,
    linewidth=1.8)
    for a, b in [("2", "23"), ("3", "23"), ("1", "123"), ("23", "123")]:
        ax.plot([points_right[a][0], points_right[b][0]], [points_right[a][1], points_right[b][1]], color=ORANGE,
    linewidth=1.8)
    arrow(ax, (0.42, 0.32), (0.58, 0.32), color=MAGENTA, text="Racah transform", text_offset=(0, 0.05))
```

```
    ax.text(0.27, 0.82, r"$((1,2),3)$", ha="center", fontsize=13, color=BLUE, weight="bold")
    ax.text(0.76, 0.82, r"$(1,(2,3))$", ha="center", fontsize=13, color=ORANGE, weight="bold")
    save_figure(fig, "18_three_body_trees")
    generated.append("18_three_body_trees")

    # 19. Four-body associahedron pentagon
    fig, ax = diagram_axes((12.8, 7.0))
    ax.text(0.5, 0.95, "Five four-packet coupling trees form an associahedral pentagon", ha="center", fontsize=20,
     weight="bold")
    angles = np.linspace(math.pi / 2, math.pi / 2 + 2 * math.pi, 6)[:-1]
    positions = [(0.5 + 0.34 * math.cos(angle), 0.48 + 0.34 * math.sin(angle)) for angle in angles]
    for index, ((name, _), point) in enumerate(zip(FOUR_TREES, positions)):
        ax.add_patch(FancyBboxPatch((point[0] - 0.105, point[1] - 0.055), 0.21, 0.11, boxstyle="round,pad=.008,
     rounding_size=.015", facecolor=COLORS[index] + "22", edgecolor=COLORS[index], linewidth=1.7))
        ax.text(point[0], point[1], name, ha="center", va="center", fontsize=10.2)
    for index in range(5):
        start = positions[index]
        end = positions[(index + 1) % 5]
        arrow(ax, start, end, color=INK)
    ax.text(0.5, 0.48, r"$R_{01}R_{12}R_{23}R_{34}R_{40}=I$", ha="center", va="center", fontsize=15, weight="bold")
    save_figure(fig, "19_pentagon_trees")
    generated.append("19_pentagon_trees")

    # 20. Pentagon residual
    totals = np.arange(1, 10)
    residuals = []
    dimensions = []
    for value in totals:
        trees = four_body_tree_bases(int(value), (0.55, 0.8, 1.05, 1.3))
        product_matrix, _ = pentagon_product(trees)
        residuals.append(np.max(np.abs(product_matrix - np.eye(product_matrix.shape[0]))))
        dimensions.append(product_matrix.shape[0])
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    ax.semilogy(totals, residuals, marker="o", color=PURPLE)
    for x, y, dimension in zip(totals, residuals, dimensions):
        ax.text(x, y * 1.25, str(dimension), ha="center", fontsize=8, color=MUTED)
    ax.set_xlabel("total excitation N")
    ax.set_ylabel("pentagon residual")
    ax.set_title("Pentagon closure on growing multiplicity spaces")
    ax.grid(alpha=0.18, which="both")
    save_figure(fig, "20_pentagon_residual")
    generated.append("20_pentagon_residual")

    # 21. Five edge-overlap heatmaps
    trees = four_body_tree_bases(3, (0.55, 0.8, 1.05, 1.3))
    _, overlaps = pentagon_product(trees)
    fig, axes = plt.subplots(1, 5, figsize=(14.0, 3.5))
    vmax = max(np.max(np.abs(matrix)) for matrix in overlaps)
    for index, (axis, matrix) in enumerate(zip(axes, overlaps)):
        image = axis.imshow(matrix, cmap="coolwarm", vmin=-vmax, vmax=vmax, aspect="auto")
        axis.set_title(f"edge {index+1}", fontsize=9.5)
        axis.set_xticks([])
        axis.set_yticks([])
    fig.colorbar(image, ax=axes.ravel().tolist(), fraction=0.018, pad=0.02, label="overlap amplitude")
    fig.suptitle("The five elementary associators at total excitation 3", fontsize=15, weight="bold")
    fig.subplots_adjust(left=0.03, right=0.92, bottom=0.10, top=0.78, wspace=0.10)
    save_figure(fig, "21_pentagon_edge_kernels")
    generated.append("21_pentagon_edge_kernels")

    # 22. Joint spectra for one four-body tree
    tree = trees[0]
    diagonal_a = np.diag(tree.joint_diagonals[0])
    diagonal_b = np.diag(tree.joint_diagonals[1])
    fig, ax = plt.subplots(figsize=(8.0, 6.3))
    scatter = ax.scatter(diagonal_a, diagonal_b, c=np.arange(len(diagonal_a)), cmap="viridis", s=90, edgecolor=INK,
     linewidth=0.5)
    for index, (x, y) in enumerate(zip(diagonal_a, diagonal_b)):
        ax.text(x, y + 0.2, str(index), ha="center", fontsize=8)
    ax.set_xlabel("inner Casimir eigenvalue")
    ax.set_ylabel("nested Casimir eigenvalue")
    ax.set_title("Joint packet labels for one four-body tree")
```

```
    fig.colorbar(scatter, ax=ax, fraction=0.04, pad=0.03, label="basis index")
    ax.grid(alpha=0.15)
    save_figure(fig, "22_joint_tree_spectrum")
    generated.append("22_joint_tree_spectrum")

    # 23. Four-body compression ratio
    totals = np.arange(0, 23)
    full = np.array([excitation_dimension(int(value), 4) for value in totals])
    packet = np.array([multiplicity_dimension(int(value), 4) for value in totals])
    ratio = full / packet
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    ax.plot(totals, ratio, color=BLUE, marker="o")
    ax.set_xlabel("total excitation N")
    ax.set_ylabel("full level / multiplicity dimension")
    ax.set_title("Four-body projector reduction before associator synthesis")
    ax.grid(alpha=0.18)
    save_figure(fig, "23_four_body_compression")
    generated.append("23_four_body_compression")

    # 24. Givens circuit schematic
    dimension = 6
    fig, ax = diagram_axes((12.8, 6.2))
    ax.text(0.5, 0.94, "Finite recoupling transform as a triangular Givens network", ha="center", fontsize=20,
     weight="bold")
    x_positions = np.linspace(0.08, 0.92, 8)
    y_positions = np.linspace(0.18, 0.78, dimension)
    for y in y_positions:
        ax.plot([x_positions[0], x_positions[-1]], [y, y], color=GRID, linewidth=1.3)
    gate_index = 0
    for column in range(dimension - 1):
        x = x_positions[column + 1]
        for row in range(dimension - 1, column, -1):
            y0, y1 = y_positions[row - 1], y_positions[row]
            ax.plot([x, x], [y0, y1], color=COLORS[gate_index % len(COLORS)], linewidth=3.2)
            ax.add_patch(Circle((x, y0), 0.012, facecolor=WHITE, edgecolor=COLORS[gate_index % len(COLORS)],
     linewidth=1.4))
            ax.add_patch(Circle((x, y1), 0.012, facecolor=COLORS[gate_index % len(COLORS)], edgecolor=COLORS[
     gate_index % len(COLORS)], linewidth=1.4))
            gate_index += 1
            x += 0.012
    for index, y in enumerate(y_positions):
        ax.text(0.045, y, str(index), ha="center", va="center", fontsize=9)
    ax.text(0.5, 0.08, r"At most $d(d-1)/2$ real plane rotations plus diagonal signs", ha="center", fontsize=12,
     color=MUTED)
    save_figure(fig, "24_givens_network")
    generated.append("24_givens_network")

    # 25. Deterministic complexity profiles
    totals = np.arange(2, 25, 2)
    dimensions = totals + 1
    dense_proxy = dimensions.astype(float) ** 3
    full_transform_proxy = dimensions.astype(float) ** 2
    selected_column_proxy = dimensions.astype(float)
    dense_proxy /= dense_proxy[0]
    full_transform_proxy /= full_transform_proxy[0]
    selected_column_proxy /= selected_column_proxy[0]
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    ax.loglog(dimensions, dense_proxy, marker="o", color=TEAL, label=r"dense reduced eigensolve $d^3$")
    ax.loglog(dimensions, full_transform_proxy, marker="o", color=BLUE, label=r"structured full transform $d^2$")
    ax.loglog(dimensions, selected_column_proxy, marker="o", color=ORANGE, label=r"selected recurrence column $d$")
    ax.set_xlabel("multiplicity dimension d=N+1")
    ax.set_ylabel("normalized cost proxy")
    ax.set_title("Reference complexity profiles in the multiplicity packet")
    ax.grid(alpha=0.18, which="both")
    ax.legend()
    save_figure(fig, "25_runtime_scaling")
    generated.append("25_runtime_scaling")

    # 26. Packet algorithm flow
    fig, ax = diagram_axes((13.0, 6.5))
    ax.text(0.5, 0.95, "Output-sensitive recoupling algorithm", ha="center", fontsize=21, weight="bold")
```

```
    stages = [
        (0.06, 0.66, 0.18, 0.15, "declare shapes\nand total degree", BLUE),
        (0.31, 0.66, 0.18, 0.15, "project to\nlowest weight", TEAL),
        (0.56, 0.66, 0.18, 0.15, "diagonalize pair\nCasimirs", ORANGE),
        (0.79, 0.66, 0.16, 0.15, "form overlap\nkernel", MAGENTA),
        (0.18, 0.25, 0.22, 0.15, "square amplitudes\nfor Markov channel", YELLOW),
        (0.47, 0.25, 0.20, 0.15, "compose trees\nthrough pentagon", PURPLE),
        (0.74, 0.25, 0.18, 0.15, "compile Givens\nnnetwork", RED),
    ]
    for x, y, w, h, text, color in stages:
        box(ax, (x, y), w, h, text, face=color + "20", edge=color, fontsize=11)
    for a, b in [(0, 1), (1, 2), (2, 3)]:
        left, right = stages[a], stages[b]
        arrow(ax, (left[0] + left[2], left[1] + left[3] / 2), (right[0], right[1] + right[3] / 2), color=INK)
    arrow(ax, (0.87, 0.66), (0.83, 0.40), color=MAGENTA)
    arrow(ax, (0.72, 0.66), (0.57, 0.40), color=ORANGE)
    arrow(ax, (0.64, 0.66), (0.29, 0.40), color=TEAL)
    ax.text(0.5, 0.08, "The full tensor level is used only to define the projector. Every later step occurs in the
     multiplicity packet space.", ha="center", fontsize=11.5, color=MUTED)
    save_figure(fig, "26_packet_algorithm")
    generated.append("26_packet_algorithm")

    # 27. Classical-family map
    fig, ax = diagram_axes((13.0, 6.5))
    ax.text(0.5, 0.95, "Where the generated recoupling functions sit in the classical hierarchy", ha="center",
     fontsize=20, weight="bold")
    nodes = [
        (0.08, 0.68, 0.20, 0.14, "Meixner-Pollaczek\npacket basis", BLUE),
        (0.40, 0.68, 0.20, 0.14, "Clebsch-Gordan\nHahn coordinates", TEAL),
        (0.72, 0.68, 0.20, 0.14, "Racah\nrecoupling", ORANGE),
        (0.21, 0.30, 0.20, 0.14, "Jacobi and\ncontinuous limits", MAGENTA),
        (0.52, 0.30, 0.20, 0.14, "Wilson-function\nextensions", PURPLE),
        (0.78, 0.30, 0.16, 0.14, "q-Racah and\nAskey-Wilson", RED),
    ]
    for x, y, w, h, text, color in nodes:
        box(ax, (x, y), w, h, text, face=color + "1E", edge=color, fontsize=11)
    arrow(ax, (0.28, 0.75), (0.40, 0.75), color=INK)
    arrow(ax, (0.60, 0.75), (0.72, 0.75), color=INK)
    arrow(ax, (0.82, 0.68), (0.62, 0.44), color=ORANGE)
    arrow(ax, (0.50, 0.68), (0.31, 0.44), color=TEAL)
    arrow(ax, (0.72, 0.37), (0.78, 0.37), color=PURPLE)
    ax.text(0.5, 0.10, "The classical names are retained. The packet calculus supplies the common refinement and
     associator interpretation.", ha="center", fontsize=11.3, color=MUTED)
    save_figure(fig, "27_classical_family_map")
    generated.append("27_classical_family_map")

    # 28. Quantum/logarithmic connection
    fig, ax = diagram_axes((13.0, 6.5))
    ax.text(0.5, 0.95, "Two quantum coordinates in the extended calculus", ha="center", fontsize=20, weight="bold")
    box(ax, (0.07, 0.60), 0.22, 0.18, "number basis\n$|n\\rangle$", face=BLUE + "1E", edge=BLUE, fontsize=13)
    box(ax, (0.39, 0.60), 0.22, 0.18, "logarithmic phase\n$e^{-it\\log n}$", face=TEAL + "1E", edge=TEAL, fontsize
     =12)
    box(ax, (0.71, 0.60), 0.22, 0.18, "shape ladder\n$D^+_{\\kappa}$", face=ORANGE + "1E", edge=ORANGE, fontsize=13)
    box(ax, (0.20, 0.22), 0.24, 0.17, "dynamical zeta\nobservable", face=MAGENTA + "1E", edge=MAGENTA, fontsize=12)
    box(ax, (0.58, 0.22), 0.24, 0.17, "finite Racah\nbasis transform", face=PURPLE + "1E", edge=PURPLE, fontsize=12)
    arrow(ax, (0.29, 0.69), (0.39, 0.69), color=INK)
    arrow(ax, (0.61, 0.69), (0.71, 0.69), color=INK)
    arrow(ax, (0.48, 0.60), (0.32, 0.39), color=TEAL)
    arrow(ax, (0.82, 0.60), (0.70, 0.39), color=ORANGE)
    ax.text(0.5, 0.08, "The logarithmic Hamiltonian and the shape recoupling transform are distinct modules and can
     be composed without identifying their spectra.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "28_quantum_connection")
    generated.append("28_quantum_connection")

    # 29. Theorem dependency map
    fig, ax = diagram_axes((13.0, 6.8))
    ax.text(0.5, 0.96, "Theorem dependency map", ha="center", fontsize=21, weight="bold")
    nodes = [
        (0.05, 0.73, 0.20, 0.12, "positive discrete\nseries", BLUE),
        (0.30, 0.73, 0.20, 0.12, "lowest-weight\nprojector", TEAL),
        (0.55, 0.73, 0.20, 0.12, "pair Casimir\nspectra", ORANGE),
```

```
        (0.80, 0.73, 0.15, 0.12, "orthogonal\noverlap", MAGENTA),
        (0.12, 0.43, 0.20, 0.12, "Racah\nrecurrence", YELLOW),
        (0.40, 0.43, 0.20, 0.12, "bistochastic\nchannel", PURPLE),
        (0.68, 0.43, 0.20, 0.12, "four-body\npentagon", RED),
        (0.27, 0.14, 0.20, 0.12, "entropy and\nuncertainty", BLUE),
        (0.56, 0.14, 0.20, 0.12, "finite quantum\ncompiler", TEAL),
    ]
    for x, y, w, h, text, color in nodes:
        box(ax, (x, y), w, h, text, face=color + "1C", edge=color, fontsize=10.5)
    for left, right in [(0, 1), (1, 2), (2, 3), (3, 4), (3, 5), (3, 6), (5, 7), (6, 8)]:
        a, b = nodes[left], nodes[right]
        arrow(ax, (a[0] + a[2] / 2, a[1]), (b[0] + b[2] / 2, b[1] + b[3]), color=INK)
    save_figure(fig, "29_theorem_map")
    generated.append("29_theorem_map")

    # 30. Future hierarchy
    fig, ax = diagram_axes((13.0, 6.4))
    ax.text(0.5, 0.95, "Next deformation directions", ha="center", fontsize=21, weight="bold")
    box(ax, (0.06, 0.58), 0.20, 0.18, "current paper\nfinite $SU(1,1)$\nRacah packets", face=BLUE + "20", edge=BLUE,
      fontsize=12)
    box(ax, (0.34, 0.58), 0.20, 0.18, "continuous series\nWilson functions", face=TEAL + "20", edge=TEAL, fontsize
     =12)
    box(ax, (0.62, 0.58), 0.20, 0.18, "$q$-deformation\nAskey-Wilson packets", face=ORANGE + "20", edge=ORANGE,
     fontsize=12)
    box(ax, (0.24, 0.20), 0.22, 0.16, "multivariate\n$3nj$ packet trees", face=MAGENTA + "20", edge=MAGENTA,
     fontsize=12)
    box(ax, (0.56, 0.20), 0.22, 0.16, "tensor-network and\nquantum circuits", face=PURPLE + "20", edge=PURPLE,
     fontsize=12)
    arrow(ax, (0.26, 0.67), (0.34, 0.67), color=INK)
    arrow(ax, (0.54, 0.67), (0.62, 0.67), color=INK)
    arrow(ax, (0.16, 0.58), (0.35, 0.36), color=BLUE)
    arrow(ax, (0.72, 0.58), (0.67, 0.36), color=ORANGE)
    ax.text(0.5, 0.08, "Each extension keeps the same rule: declare a complete coupling atlas, compute the
     associator, and certify reconstruction.", ha="center", fontsize=11.4, color=MUTED)
    save_figure(fig, "30_future_directions")
    generated.append("30_future_directions")

    return generated


# ---------------------------------------------------------------------------
# Tables and manifest
# ---------------------------------------------------------------------------


def write_tables() -> None:
    kappas = (0.7, 1.1, 0.9)
    rows = []
    for total in range(1, 13):
        data = three_body_recoupling(total, kappas)
        probability = data.overlap ** 2
        rows.append(
            {
                "N": total,
                "dimension": total + 1,
                "max_abs_overlap": float(np.max(np.abs(data.overlap))),
                "mean_row_entropy": float(np.mean(row_entropies(probability))),
                "orthogonality_residual": float(np.max(np.abs(data.overlap @ data.overlap.T - np.eye(total + 1)))),
                "tridiagonal_residual": tridiagonal_residual(data.right_in_left),
            }
        )
    with (TABLES / "three_body_metrics.csv").open("w", newline="") as handle:
        writer = csv.DictWriter(handle, fieldnames=list(rows[0].keys()))
        writer.writeheader()
        writer.writerows(rows)

    pentagon_rows = []
    for total in range(1, 8):
        trees = four_body_tree_bases(total, (0.55, 0.8, 1.05, 1.3))
        product_matrix, _ = pentagon_product(trees)
        pentagon_rows.append(
```

```
            {
                "N": total,
                "multiplicity_dimension": product_matrix.shape[0],
                "pentagon_residual": float(np.max(np.abs(product_matrix - np.eye(product_matrix.shape[0])))),
                "max_commutator_residual": max(tree.commutator_residual for tree in trees),
                "max_joint_diagonalization_residual": max(tree.diagonalization_residual for tree in trees),
            }
        )
    with (TABLES / "pentagon_metrics.csv").open("w", newline="") as handle:
        writer = csv.DictWriter(handle, fieldnames=list(pentagon_rows[0].keys()))
        writer.writeheader()
        writer.writerows(pentagon_rows)

    exact_matrix = sp.Matrix(equal_shape_two_excitation_matrix()).applyfunc(lambda value: sp.nsimplify(value))
    (GENERATED / "equal_shape_N2_matrix.tex").write_text(sp.latex(exact_matrix))


def sha256(path: Path) -> str:
    digest = hashlib.sha256()
    with path.open("rb") as handle:
        for chunk in iter(lambda: handle.read(1024 * 1024), b""):
            digest.update(chunk)
    return digest.hexdigest()


def write_manifest(figures: Sequence[str], checks: dict[str, dict[str, float | int | str | bool]]) -> None:
    manifest = {
        "title": "Projector-Generated Special Functions VII",
        "author": "Komorebi P.",
        "figure_pairs": len(figures),
        "figures": list(figures),
        "checks_passed": sum(1 for item in checks.values() if bool(item["passed"])),
        "checks_total": len(checks),
        "files": {},
    }
    for path in sorted(ROOT.rglob("*")):
        if path.is_file() and path.name not in {"manifest.json", "SHA256SUMS.txt"}:
            manifest["files"][str(path.relative_to(ROOT))] = {
                "bytes": path.stat().st_size,
                "sha256": sha256(path),
            }
    (RESULTS / "manifest.json").write_text(json.dumps(manifest, indent=2))
    lines = [f"{entry['sha256']}  {name}" for name, entry in manifest["files"].items()]
    (ROOT / "SHA256SUMS.txt").write_text("\n".join(lines) + "\n")


def main() -> None:
    figures = generate_figures()
    checks = run_checks()
    write_tables()
    write_manifest(figures, checks)
    summary = {
        "figures": len(figures),
        "checks_passed": sum(1 for item in checks.values() if bool(item["passed"])),
        "checks_total": len(checks),
        "all_passed": all(bool(item["passed"]) for item in checks.values()),
    }
    print(json.dumps(summary, indent=2))


if __name__ == "__main__":
    main()
```

Appendix E

Bibliographic note

Bibliography

[1] NIST Digital Library of Mathematical Functions, Sections 18.19, 18.20, 18.21, 18.25, and 18.38, Hahn, Racah, Wilson, and $6j$ relations. `https://dlmf.nist.gov/18`

[2] H. T. Koelink and J. Van der Jeugt, Convolutions for orthogonal polynomials from Lie and quantum algebra representations, *SIAM Journal on Mathematical Analysis* 29 (1998), 794–822.

[3] W. Groenevelt, Wilson function transforms related to Racah coefficients, *Acta Applicandae Mathematicae* 91 (2006), 133–191.

[4] S. Lievens and J. Van der Jeugt, Racah coefficients of the positive discrete series representations of $\mathfrak{su}(1,1)$, representation-theoretic recoupling literature.

[5] V. X. Genest, L. Vinet, and collaborators, Multivariate Racah and q-Racah polynomials as coupling coefficients, works on higher-rank recoupling and quantum-algebra deformations.

[6] S. Wei, Y. Zhai, Q. Lu, et al., The Riemann Hypothesis manifested in dynamical quantum phase transitions, *Nature Communications*, accepted 18 June 2026, DOI 10.1038/s41467-026-74935-8.

Appendix F

Reproduction

From the source directory, run

```
python projector_generated_special_functions_vii.py
PYTHONPATH=. pytest -q
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_VII.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_VII.tex
```

Projector-Generated Special Functions VIII

q-Hahn Lowest-Weight Packets, q-Racah Recoupling, Askey-Wilson Transfer Functions, and Deformation Geometry

Komorebi P.

Abstract

Papers VI and VII produced a continuous shape semigroup, positive discrete-series packet bases, classical Racah recoupling matrices, bistochastic measurement kernels, and pentagon coherence. The present paper introduces a positive deformation parameter q and rebuilds the same architecture inside a symmetric $U_q(\mathfrak{su}(1,1))$ coordinate system.

For $0 < q \le 1$, define the symmetric q-number

$$[x]_q = \frac{q^x - q^{-x}}{q - q^{-1}},$$

with the continuous value $[x]_q = x$ at $q = 1$. The positive shape packet $D^+_{\kappa,q}$ carries the ladder

$$K_0 |n\rangle = (n + \kappa) |n\rangle , \qquad K_+ |n\rangle = \sqrt{[n+1]_q \, [2\kappa + n]_q} \, |n+1\rangle ,$$

$$K_- |n\rangle = \sqrt{[n]_q \, [2\kappa + n - 1]_q} \, |n-1\rangle .$$

The coproduct

$$\Delta K_- = K_- \otimes q^{K_0} + q^{-K_0} \otimes K_-$$

produces exact fixed-excitation packet spaces and a q-Casimir. The lowest-weight vector of two packets has an explicit q-Hahn coefficient formula. Three packets generate two intermediate q-Casimir atlases whose overlap is orthogonal and belongs to the q-Racah class. Squaring the overlap produces a doubly stochastic measurement kernel.

For three equal half-shape packets at one excitation, the complete transform is

$$\boxed{\mathcal{R}_q = \begin{pmatrix} \dfrac{q}{1+q^2} & \dfrac{\sqrt{1+q^2+q^4}}{1+q^2} \\ \dfrac{\sqrt{1+q^2+q^4}}{1+q^2} & -\dfrac{q}{1+q^2} \end{pmatrix}.}$$

It converges to the classical Racah beam splitter at $q = 1$ and to a routing swap as $q \downarrow 0$. The corresponding row entropy and mixing gap are monotone in this two-channel case. A higher-excitation counterexample shows that such monotonicity is not universal.

Writing $q = \mathrm{e}^{-\theta}$ turns the recoupling matrices into a differentiable orthogonal flow. In a continuous sign gauge, the generator

$$A_N(\theta) = \mathcal{R}_N(\theta)^{\mathsf{T}} \partial_\theta \mathcal{R}_N(\theta)$$

is skew-symmetric. Its Frobenius norm defines a finite deformation metric. For four packets, five q-Casimir tree bases form the associahedral pentagon, and their ordered overlap product is the identity. Every finite q-Racah packet matrix can be compiled into Givens rotations and diagonal signs.

The scalar side is developed independently through the terminating q-Racah ${}_4\phi_3$ function and the continuous Askey-Wilson function. Exact duality, the $q \to 1$ Racah limit, and the Askey-Wilson q-difference equation are verified symbolically or at high precision. q-Racah and Askey-Wilson functions, quantum-group recoupling, and the pentagon identity are classical. The generated contribution is the common packet architecture that retains the q-Hahn seed, projector reduction, q-Racah analysis and

synthesis, stochastic measurement, deformation flow, associator certificate, and finite compiler in one system.

All figures were generated by the accompanying Python program. The release contains 32 vector figures, matching PNG files, 23 built-in checks, 20 independent tests, numerical tables, and the complete source.

Status and claim boundary

The q-Racah and Askey-Wilson polynomial families are classical members of the Askey-Wilson scheme. Their q-hypergeometric definitions, orthogonality measures, duality laws, q-difference equations, and classical limits are established parts of special-function theory. Their occurrence as Clebsch-Gordan or Racah coefficients in quantum-group representation theory is also classical.

This paper does not claim to have invented q-Racah polynomials, Askey-Wilson polynomials, quantum $6j$ symbols, or the abstract pentagon identity. It shows how they arise from the projector-resolved packet construction and adds a single operational language for lowest-weight reduction, analysis, synthesis, measurement, deformation transport, and compilation.

The finite matrix construction uses a symmetric q-number convention and the positive real range $0 < q \leq 1$. Other conventions are related by changes of generators and powers of q. Formula comparisons must therefore preserve the declared convention.

The q-dependent entropy statements concern the classical channel obtained after squaring the amplitudes and discarding phase. The underlying q-Racah matrix remains exactly reversible. Entropy is monotone under each fixed bistochastic channel. It is not generally monotone as the deformation parameter itself changes.

The four-body pentagon closes because every tree basis is complete in the same lowest-weight multiplicity space. The numerical product is a stringent implementation certificate, not a replacement for the abstract Biedenharn-Elliott theorem.

The final quantum discussion keeps this shape-recoupling module separate from the logarithmic number Hamiltonian used in the attached zeta-governed quantum dynamics. The modules can be tensored or composed, but the present paper does not identify their spectra and does not prove the Riemann hypothesis.

Contents

Status and claim boundary i

I q-deformed packet algebra 1

1 Symmetric q-numbers and positive shape packets 1

2 Coproduct, q-Casimir, and exact tensor routing 3

3 Fixed excitation and the lowest-weight multiplicity packet 4

II q-Hahn lowest-weight functions 5

4 Two-packet seed and closed coefficients 5

III q-Racah recoupling 7

5 Two intermediate q-Casimir atlases 7

6 An exact q-controlled beam splitter 10

7 Bistochastic measurement and entropy 12

IV Deformation geometry 16

8 Orthogonal transport in q 16

9 Classical and routing limits 18

V Four-body q-associators 19

10 Five complete q-Casimir tree atlases 19

11 Deforming the associator 21

VI Finite compilation 22

12 Givens factorization of a q-Racah packet 22

VII Scalar q-special functions 24

13 Terminating q-Racah function 24

14 Askey-Wilson transfer function 27

VIII Quantum and computational interpretation **29**

15 A reversible q-shape module **29**

16 What the deformation adds **30**

IX Verification and formulas **30**

17 Formula registry **30**

18 Numerical certificate table **31**

19 Research directions and boundaries **32**

20 Conclusion **32**

A Selected numerical tables **33**

B Complete Python implementation **33**

C Bibliographic note **53**

D Reproduction **54**

List of Figures

1.1 The q-deformed packet program. The positive-series ladder generates a q-Hahn lowest packet, pair q-Casimir atlases, q-Racah transforms, measurement kernels, deformation flow, and four-body associator closure. . . . 2
1.2 Symmetric q-numbers. As q approaches one, the deformed ladder approaches the ordinary linear coordinate. Smaller q amplifies large positive degrees. . . . 2
1.3 Positive-series raising coefficients. The local q-ladder remains exact while its high-degree growth changes strongly with the deformation parameter. . . . 3
2.1 Coproduct paths. A single lowering event is routed through each factor with the q-weights required by coassociativity. . . . 4
4.1 q-Hahn lowest-weight packets. The same total excitation is distributed differently as q changes. The alternating phase is omitted in the probability plot. . . . 6
4.2 Classical limit of the lowest packet. The q-Hahn coefficient vector converges to its ordinary Hahn counterpart as q approaches one. . . . 7
5.1 Pair q-Casimir spectra. The intermediate eigenvalues are exact q-number deformations of the classical quadratic spectrum. . . . 8
5.2 q-Racah recoupling matrices. Changing q redistributes amplitude across the same complete pair-Casimir coordinate systems. . . . 9
5.3 Analysis and synthesis residual. Across the tested dimensions and deformation values, the orthogonality error remains at floating-point scale. . . . 9
5.4 The opposing q-Casimir in one packet atlas. The tridiagonal form is the operator source of the q-Racah three-term recurrence. . . . 10
6.1 Closed beam-splitter amplitudes. The diagonal amplitude grows from zero to one half, while the transfer amplitude decreases from one to $\sqrt{3}/2$. . . . 11
6.2 Squared two-channel kernels. The coherent transform remains orthogonal, while the measured channel becomes a q-dependent bistochastic matrix. . . . 12
7.1 Bistochastic q-Racah kernels. Every amplitude matrix produces a classical measurement channel with unit row and column sums. . . . 13
7.2 Entropy and spectral gap in the exact two-channel model. Both increase monotonically with q. . . . 14
7.3 Higher-excitation entropy counterexample. A selected row of the q-Racah measurement channel is not monotone throughout the deformation interval. . . . 15
7.4 Spectral mixing beyond two channels. The q-dependence of the second eigenvalue becomes level dependent and need not follow a universal monotone law. . . . 15
8.1 Skew generator of the q-flow. The antisymmetric matrix records which packet coordinates exchange amplitude under an infinitesimal change of deformation. . . . 17
8.2 Deformation metric across excitation. The sensitivity of the complete recoupling atlas grows with multiplicity dimension. . . . 18
9.1 Classical limit of the matrix construction. After sign alignment, the complete q-Racah packet transform converges to the classical Racah transform. . . . 19
10.1 The q-deformed associahedral pentagon. Every vertex is a complete joint-Casimir atlas. Neighboring vertices are related by finite q-Racah packet transforms. . . . 20
10.2 Pentagon residual over q and excitation. The closure error remains at floating-point scale through the tested multiplicity dimensions. . . . 21
10.3 Five q-associator edges. Each edge has its own signed amplitude pattern, but their ordered product reconstructs the initial tree atlas. . . . 21
11.1 Deformation of the five associator edges. The q-flow redistributes local generator norm around the pentagon while preserving exact global closure. . . . 22
12.1 Rotation count. The direct compiler reaches the worst-case triangular count only when every subdiagonal elimination is needed. . . . 23
12.2 Compiler reconstruction residual. The finite rotation product reproduces the original q-Racah transform to numerical precision. . . . 23

12.3 Programmable q-packet execution path. The compiler begins with shapes, q, and total excitation, constructs only the required multiplicity projectors, and emits signed transforms, measurement kernels, or pentagon certificates. 24
13.1 A terminating q-Racah table. Degrees and lattice coordinates appear on equal footing, which makes duality visible before normalization by the orthogonality weights. 25
13.2 Duality residual. The parameter-exchanged degree-coordinate identity holds at floating-point precision. . . 26
13.3 q-Racah to Racah. The finite basic-hypergeometric polynomial converges termwise to its ordinary hypergeometric counterpart. 27
14.1 Askey-Wilson packet functions. Several degrees are plotted on the unit-circle coordinate $z = \mathrm{e}^{\mathrm{i}\theta}$. 28
14.2 q-difference certificate. Direct substitution of the generated functions into the Askey-Wilson operator leaves only numerical roundoff. 28
14.3 Finite and continuous q-functions. q-Hahn packets generate coupled bases, q-Racah functions generate finite recoupling, and Askey-Wilson functions provide the continuous spectral lift. 29

Part I

q-deformed packet algebra

Chapter 1

Symmetric q-numbers and positive shape packets

Definition 1.1 (Symmetric q-number)**.** For $q > 0$ and $q \neq 1$, define

$$\boxed{[x]_q = \frac{q^x - q^{-x}}{q - q^{-1}}.}$$

At $q = 1$, define $[x]_q = x$ by continuity.

Writing $q = \mathrm{e}^{-\theta}$ gives

$$[x]_q = \frac{\sinh(x\theta)}{\sinh\theta}.$$

Hence $[-x]_q = -[x]_q$ and $[x]_q > 0$ for $x > 0$.

Definition 1.2 (Positive q-shape packet)**.** For $\kappa > 0$, let $D^+_{\kappa,q}$ have orthonormal basis $\{|n\rangle : n \geq 0\}$ with

$$K_0 |n\rangle = (n + \kappa) |n\rangle ,$$

$$K_+ |n\rangle = \sqrt{[n+1]_q [2\kappa + n]_q}\, |n+1\rangle ,$$

$$K_- |n\rangle = \sqrt{[n]_q [2\kappa + n - 1]_q}\, |n-1\rangle .$$

Proposition 1.3 (Local q-algebra)**.** *On every basis vector away from a finite truncation boundary,*

$$[K_0, K_\pm] = \pm K_\pm, \qquad [K_-, K_+] = [2K_0]_q .$$

Proof. The first identities follow because $K_\pm$ change n by one. For the second, the coefficient of $|n\rangle$ in $K_-K_+ |n\rangle$ is

$$[n+1]_q [2\kappa + n]_q ,$$

while the coefficient in $K_+K_- |n\rangle$ is

$$[n]_q [2\kappa + n - 1]_q .$$

The elementary hyperbolic identity

$$[x+1]_q [y]_q - [x]_q [y-1]_q = [x+y]_q$$

with $x = n$ and $y = 2\kappa + n$ gives $[2n + 2\kappa]_q = [2K_0]_q$. □

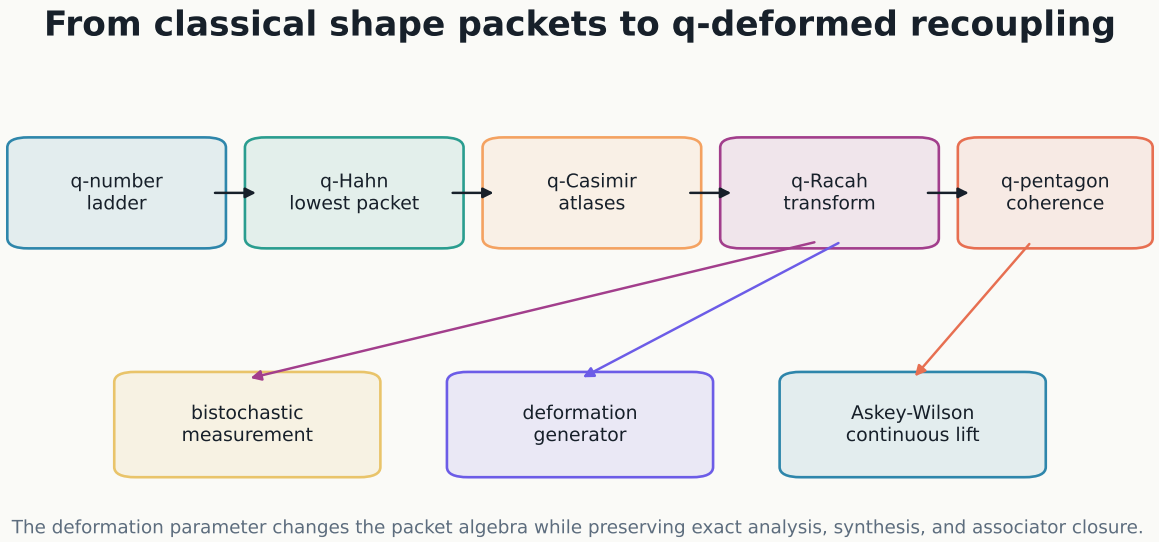


Figure 1.1: The q-deformed packet program. The positive-series ladder generates a q-Hahn lowest packet, pair q-Casimir atlases, q-Racah transforms, measurement kernels, deformation flow, and four-body associator closure.

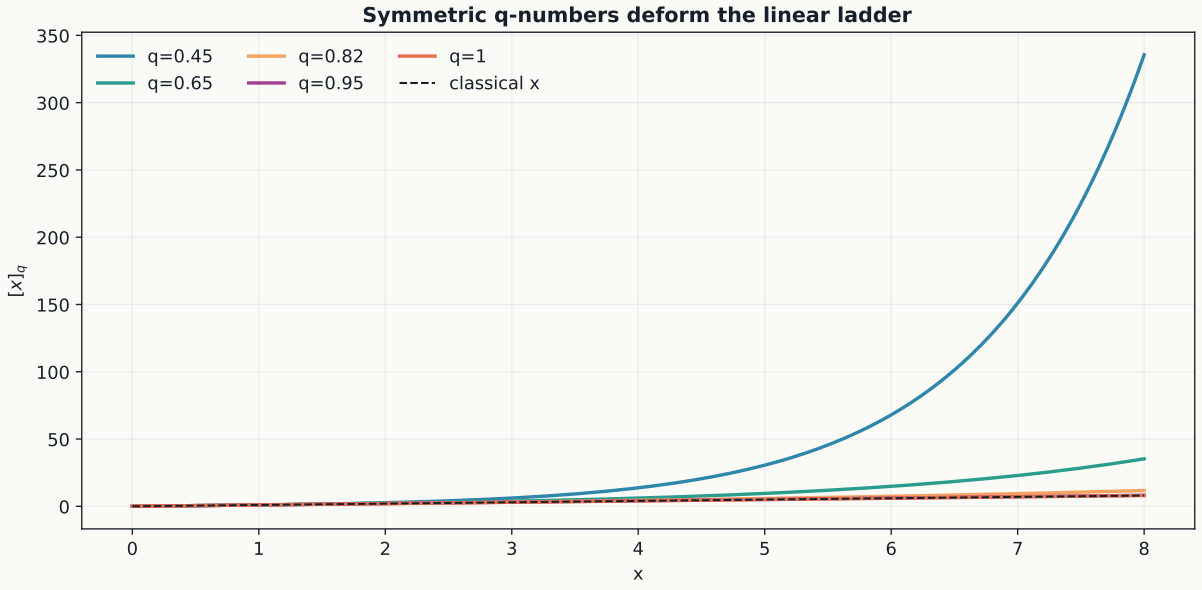


Figure 1.2: Symmetric q-numbers. As q approaches one, the deformed ladder approaches the ordinary linear coordinate. Smaller q amplifies large positive degrees.

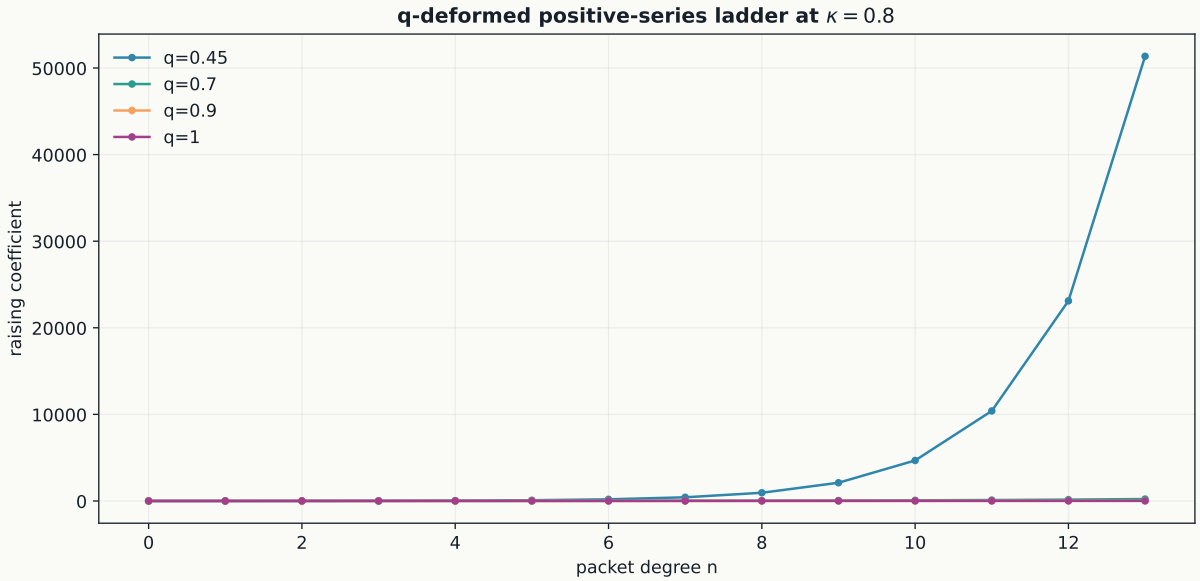


Figure 1.3: Positive-series raising coefficients. The local q-ladder remains exact while its high-degree growth changes strongly with the deformation parameter.

Chapter 2

Coproduct, q-Casimir, and exact tensor routing

Define

$$\Delta K_0 = K_0 \otimes \mathrm{I} + \mathrm{I} \otimes K_0,$$

$$\boxed{\Delta K_\pm = K_\pm \otimes q^{K_0} + q^{-K_0} \otimes K_\pm.}$$

Iterating this formula gives the total generators on any finite packet product.

Theorem 2.1 (Coassociative packet routing)**.** *The coproduct is coassociative on the declared generators. In particular,*

$$(\Delta \otimes \mathrm{I})\Delta K_- = (\mathrm{I} \otimes \Delta)\Delta K_-$$

and both sides equal

$$K_- \otimes q^{K_0} \otimes q^{K_0} + q^{-K_0} \otimes K_- \otimes q^{K_0} + q^{-K_0} \otimes q^{-K_0} \otimes K_-.$$

Proof. Since K_0 is primitive,

$$q^{\Delta K_0} = q^{K_0} \otimes q^{K_0}.$$

Substituting this identity into either iterated coproduct gives the same three terms. The proof for K_+ is identical. □

Definition 2.2 (q-Casimir)**.** Define

$$\boxed{\mathcal{C}_q = [K_0]_q\,[K_0 - 1]_q - K_+K_-.}$$

Theorem 2.3 (Scalar packet Casimir)**.** *On* $D^+_{\kappa,q}$*,*

$$\boxed{\mathcal{C}_q = [\kappa]_q\,[\kappa - 1]_q\,\mathrm{I}.}$$

Proof. On $|n\rangle$ the first term is

$$[n+\kappa]_q\,[n+\kappa-1]_q\,,$$

and the second is

$$[n]_q\,[2\kappa+n-1]_q\,.$$

The q-number identity

$$[x+y]_q\,[x+y-1]_q - [y]_q\,[2x+y-1]_q = [x]_q\,[x-1]_q$$

with $x=\kappa$ and $y=n$ gives the result. □

The symmetric coproduct routes one lowering event through every factor

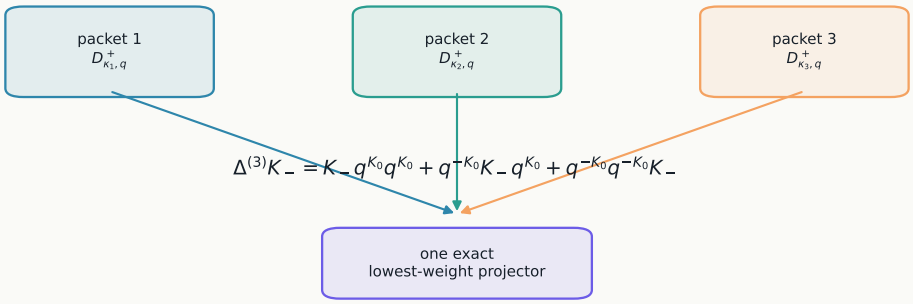


Figure 2.1: Coproduct paths. A single lowering event is routed through each factor with the q-weights required by coassociativity.

Chapter 3

Fixed excitation and the lowest-weight multiplicity packet

For m factors, let

$$\mathcal{H}_N^{(m)} = \operatorname{span}\{|n_1,\dots,n_m\rangle : n_1+\cdots+n_m = N\}.$$

Then

$$\dim \mathcal{H}_N^{(m)} = \binom{N+m-1}{m-1}.$$

The lowest-weight multiplicity packet is

$$\boxed{\mathcal{M}_N^{(m)} = \ker\left(K_-^{[m]}\big|_{\mathcal{H}_N^{(m)}}\right).}$$

Theorem 3.1 (Multiplicity dimension)**.** *For every $N \geq 0$ and $m \geq 2$,*

$$\boxed{\dim \mathcal{M}_N^{(m)} = \binom{N+m-2}{m-2}.}$$

Proof. The lowering map sends $\mathcal{H}_N^{(m)}$ to $\mathcal{H}_{N-1}^{(m)}$. Order the target compositions by their last coordinate. For every target state t, choose the source $t + e_m$. Lowering the last factor produces t with a nonzero coefficient. Lowering any earlier factor produces a target with a larger last coordinate. The corresponding square submatrix is triangular with nonzero diagonal, so the lowering map is onto. Rank-nullity gives

$$\dim \mathcal{M}_N^{(m)} = \binom{N+m-1}{m-1} - \binom{N+m-2}{m-1} = \binom{N+m-2}{m-2}.$$

□

This projector reduction is output-sensitive. The full tensor level defines the kernel. All recoupling then occurs in the smaller multiplicity packet.

Part II

q-Hahn lowest-weight functions

Chapter 4

Two-packet seed and closed coefficients

Take two shapes $\kappa_1, \kappa_2 > 0$ and total excitation N. Write the fixed-excitation basis as

$$|n, N-n\rangle, \qquad 0 \le n \le N.$$

Set

$$\lambda = q^{-(N-1+\kappa_1+\kappa_2)}.$$

Theorem 4.1 (Explicit q-Hahn lowest packet)**.** *A normalized lowest-weight vector is*

$$\boxed{|\Omega_N\rangle = \frac{1}{\sqrt{Z_N}} \sum_{n=0}^{N} (-\lambda)^n \sqrt{\begin{bmatrix} N \\ n \end{bmatrix}_q \frac{[2\kappa_2 + N - n]_{q,n}}{[2\kappa_1]_{q,n}}}\, |n, N-n\rangle,}$$

where

$$\begin{bmatrix} N \\ n \end{bmatrix}_q = \frac{[N]_q!}{[n]_q!\,[N-n]_q!}$$

and

$$[x]_{q,n} = \prod_{j=0}^{n-1} [x+j]_q.$$

Proof. Let c_n be the coefficient of $|n, N-n\rangle$. The coefficient of $|n, N-n-1\rangle$ in the total lowering equation is

$$q^{N-n-1+\kappa_2} \sqrt{[n+1]_q\,[2\kappa_1+n]_q}\, c_{n+1} + q^{-(n+\kappa_1)} \sqrt{[N-n]_q\,[2\kappa_2+N-n-1]_q}\, c_n.$$

Setting this to zero gives

$$\frac{c_{n+1}}{c_n} = -\lambda \sqrt{\frac{[N-n]_q\,[2\kappa_2+N-n-1]_q}{[n+1]_q\,[2\kappa_1+n]_q}}.$$

The displayed closed form has exactly this ratio. Normalization fixes Z_N. □

The squared coefficients form the discrete weight for the lowest q-Hahn packet. General coupled states are obtained by applying the total raising operator and are described by q-Hahn polynomial coordinates in the standard representation-theoretic normalization.

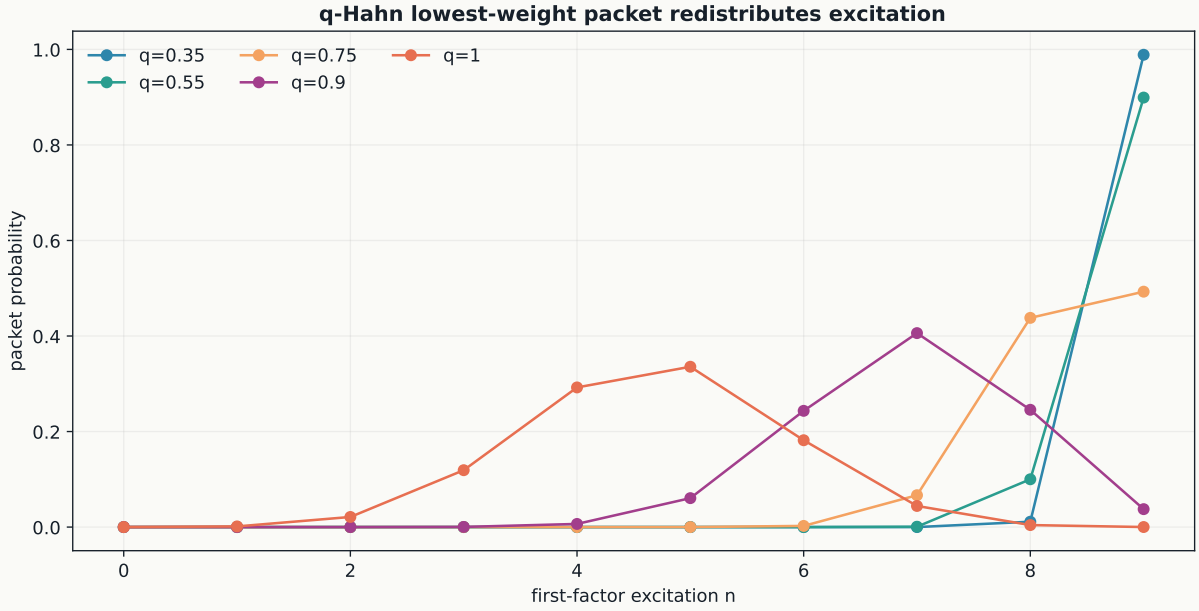


Figure 4.1: q-Hahn lowest-weight packets. The same total excitation is distributed differently as q changes. The alternating phase is omitted in the probability plot.

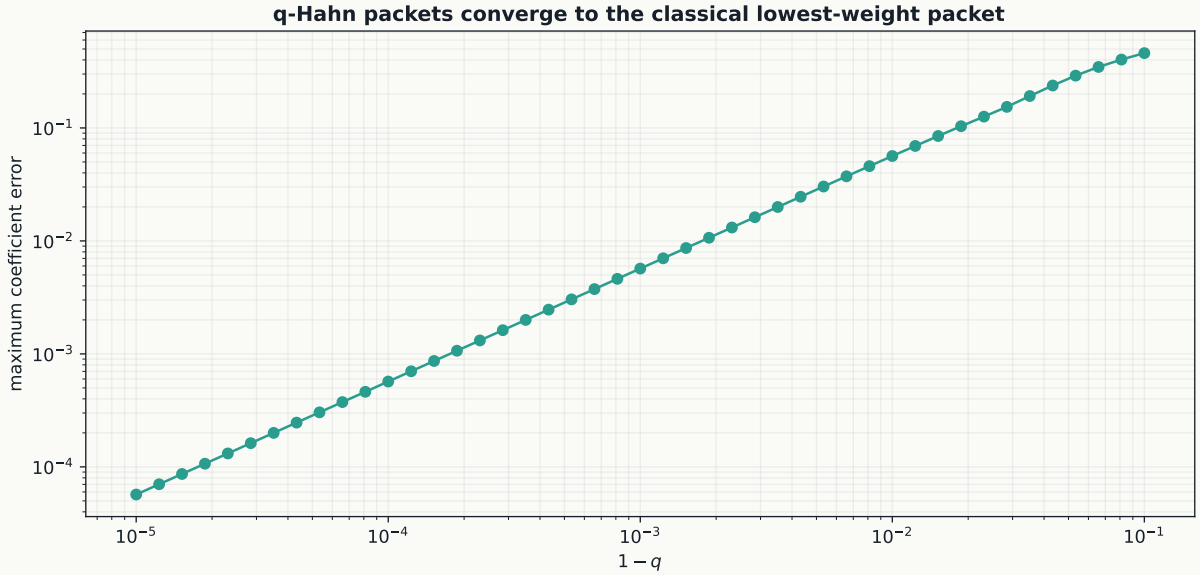


Figure 4.2: Classical limit of the lowest packet. The q-Hahn coefficient vector converges to its ordinary Hahn counterpart as q approaches one.

Worked example 4.2 (A five-excitation seed)**.** For

$$N = 5, \qquad (\kappa_1, \kappa_2) = (0.7, 1.1), \qquad q = 0.73,$$

the code constructs the six coefficients from the theorem and verifies

$$\left\| K_-^{[2]} \Omega_5 \right\|_\infty = 2.22 \times 10^{-16}.$$

The numerical residual is not the proof. It confirms that the implementation follows the exact recurrence.

Part III

q-Racah recoupling

Chapter 5

Two intermediate q-Casimir atlases

For three shapes, the multiplicity space has dimension $N + 1$. Restrict the pair Casimirs to this space:

$$\mathcal{C}_{12}^{(N)} = Q_N^{\mathsf{T}} \mathcal{C}_{12} Q_N, \qquad \mathcal{C}_{23}^{(N)} = Q_N^{\mathsf{T}} \mathcal{C}_{23} Q_N,$$

where the columns of Q_N form an orthonormal frame for $\mathcal{M}_N^{(3)}$.

The pair spectra are

$$\boxed{\lambda_j^{ab}(q) = [\kappa_a + \kappa_b + j]_q \, [\kappa_a + \kappa_b + j - 1]_q \,, \qquad 0 \le j \le N.}$$

Let L_N diagonalize $\mathcal{C}_{12}^{(N)}$ and let R_N diagonalize $\mathcal{C}_{23}^{(N)}$. The recoupling matrix is

$$\boxed{\mathcal{R}_N(q) = L_N^{\mathsf{T}} R_N.}$$

Theorem 5.1 (Exact analysis and synthesis)**.** *For every admissible finite packet,*

$$\boxed{\mathcal{R}_N(q)^{\mathsf{T}}\mathcal{R}_N(q) = \mathcal{R}_N(q)\mathcal{R}_N(q)^{\mathsf{T}} = \mathrm{I}_{N+1}.}$$

Proof. Both L_N and R_N are orthogonal eigenbases of real symmetric operators in the same multiplicity space. Therefore

$$\mathcal{R}_N^{\mathsf{T}}\mathcal{R}_N = R_N^{\mathsf{T}} L_N L_N^{\mathsf{T}} R_N = \mathrm{I}.$$

The other identity is identical. □

The classical representation-theoretic identification says that, after normalization and a parameter map determined by the three shapes and N, the entries of $\mathcal{R}_N$ are q-Racah functions. Equivalently, the restricted pair Casimirs form a finite tridiagonal pair. In the left atlas,

$$\mathcal{R}_N \operatorname{diag}(\lambda_m^{23})\mathcal{R}_N^{\mathsf{T}}$$

is tridiagonal. Hence each column obeys a three-term recurrence in the left packet label.

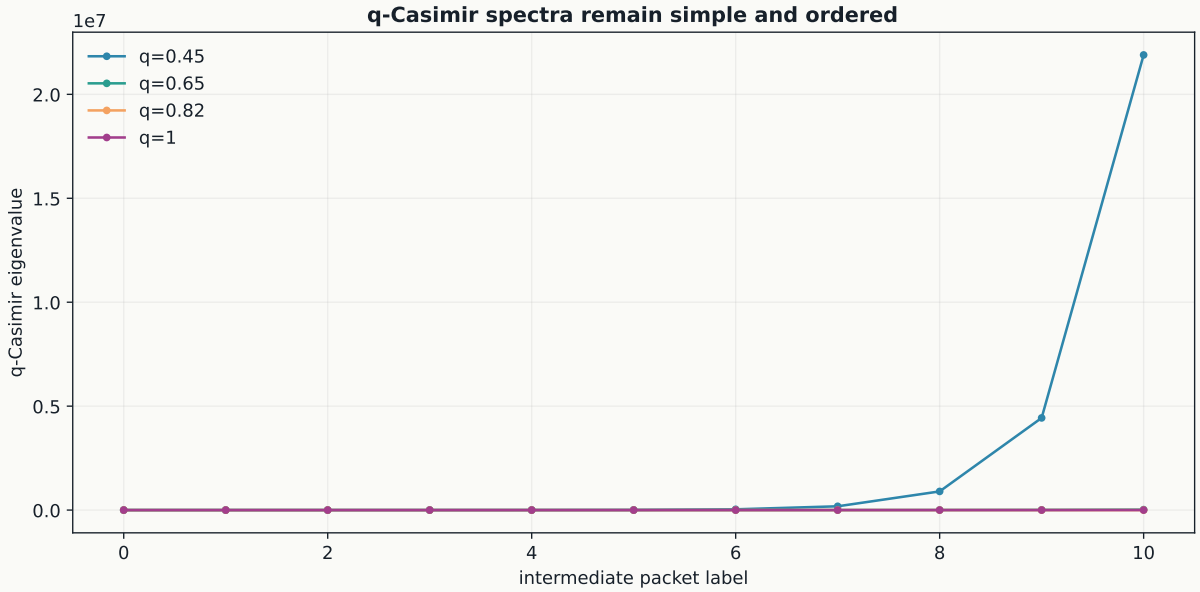


Figure 5.1: Pair q-Casimir spectra. The intermediate eigenvalues are exact q-number deformations of the classical quadratic spectrum.

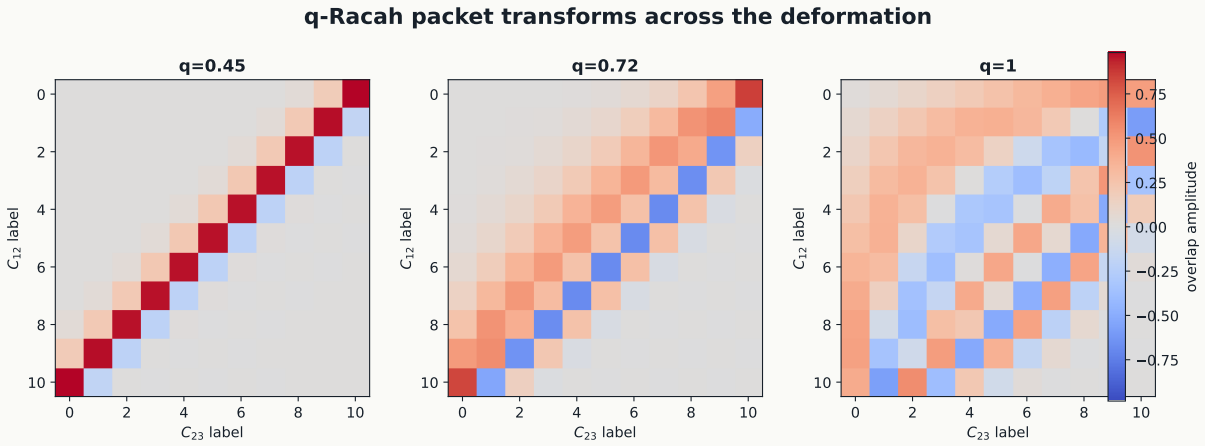


Figure 5.2: q-Racah recoupling matrices. Changing q redistributes amplitude across the same complete pair-Casimir coordinate systems.

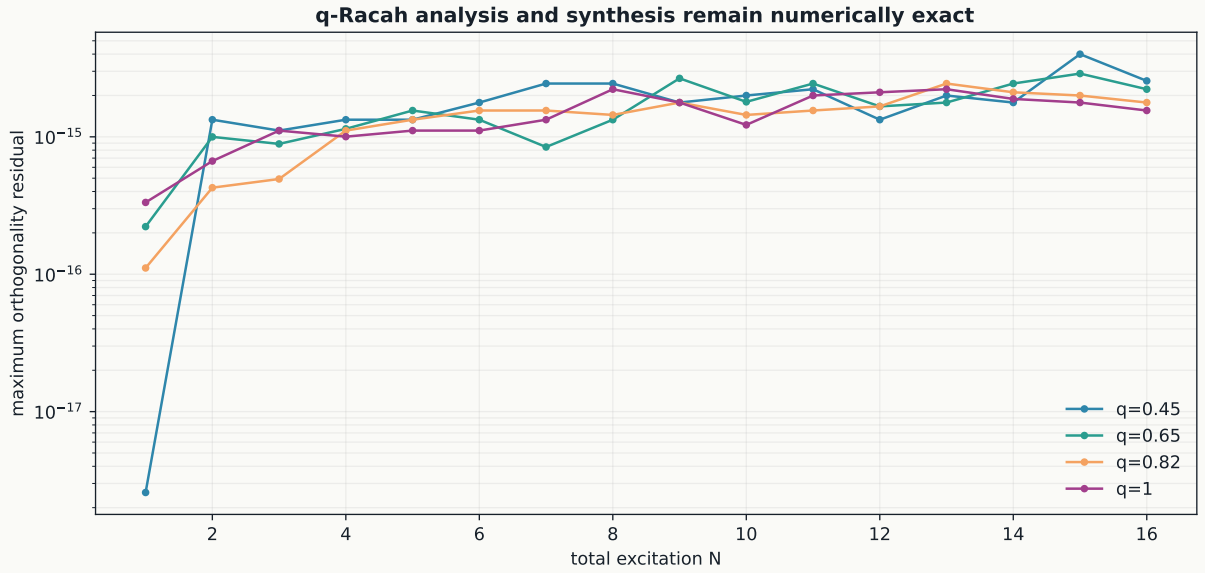


Figure 5.3: Analysis and synthesis residual. Across the tested dimensions and deformation values, the orthogonality error remains at floating-point scale.

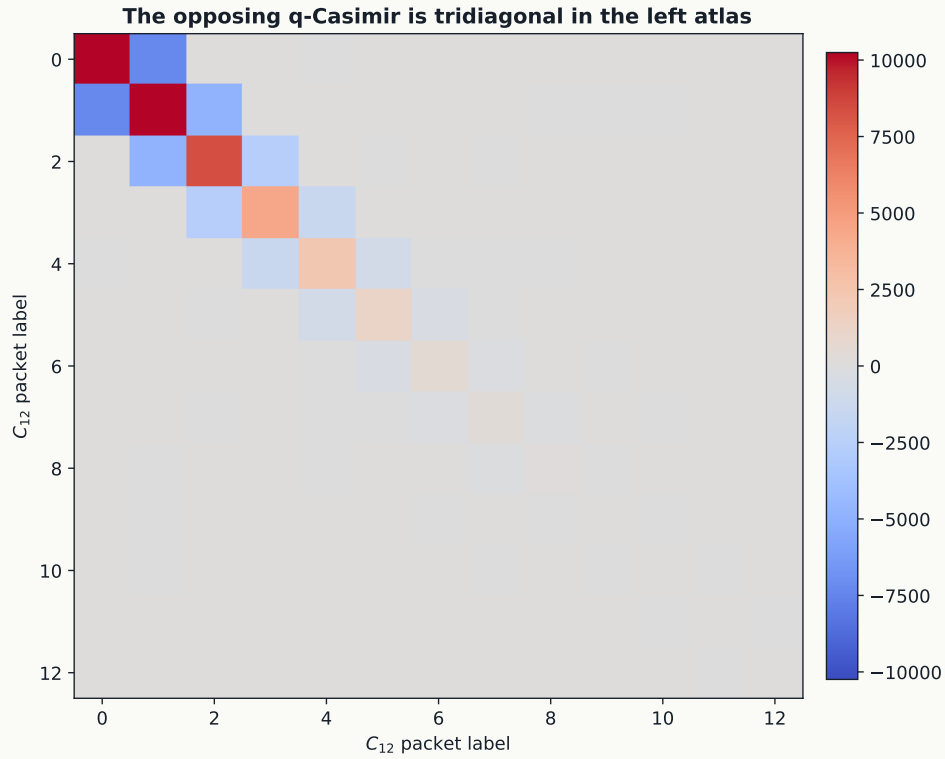


Figure 5.4: The opposing q-Casimir in one packet atlas. The tridiagonal form is the operator source of the q-Racah three-term recurrence.

Chapter 6

An exact q-controlled beam splitter

Take

$$\kappa_1 = \kappa_2 = \kappa_3 = \frac{1}{2}, \qquad N = 1.$$

Theorem 6.1 (Closed two-channel q-Racah transform)**.** *The left-to-right recoupling matrix can be chosen as*

$$\boxed{\mathcal{R}_q = \begin{pmatrix} c_q & s_q \\ s_q & -c_q \end{pmatrix},}$$

where

$$\boxed{c_q = \frac{q}{1+q^2}}$$

and

$$\boxed{s_q = \frac{\sqrt{1+q^2+q^4}}{1+q^2}.}$$

Proof. The fixed-excitation space is spanned by $|100\rangle, |010\rangle, |001\rangle$. The total lowering equation is

$$qx_1 + x_2 + q^{-1}x_3 = 0,$$

so the lowest-weight multiplicity space is two-dimensional. Restricting either pair Casimir to this plane gives two normalized eigenvectors. Their overlap simplifies to the displayed matrix. Directly,

$$c_q^2 + s_q^2 = \frac{q^2 + 1 + q^2 + q^4}{(1+q^2)^2} = 1,$$

so the matrix is orthogonal. Substitution into the restricted pair-Casimir eigenvalue equations completes the verification. □

At $q = 1$,

$$\mathcal{R}_1 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{pmatrix},$$

which is the classical equal-shape Racah beam splitter. As $q \downarrow 0$,

$$\mathcal{R}_q \longrightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The deformation therefore interpolates continuously between classical mixing and sparse routing.

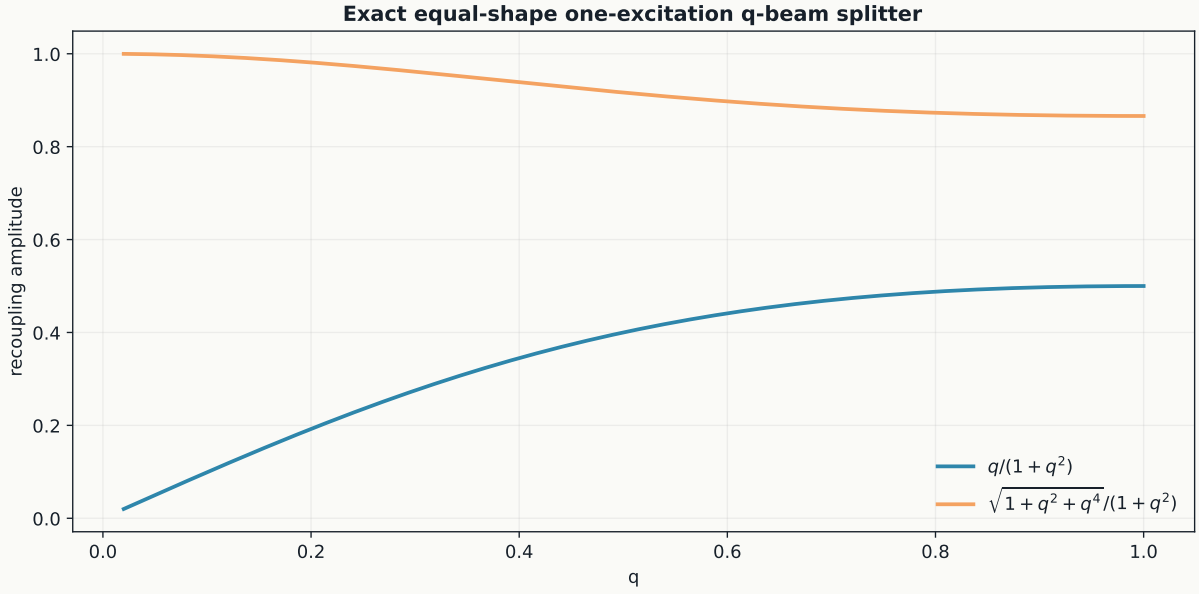


Figure 6.1: Closed beam-splitter amplitudes. The diagonal amplitude grows from zero to one half, while the transfer amplitude decreases from one to $\sqrt{3}/2$.

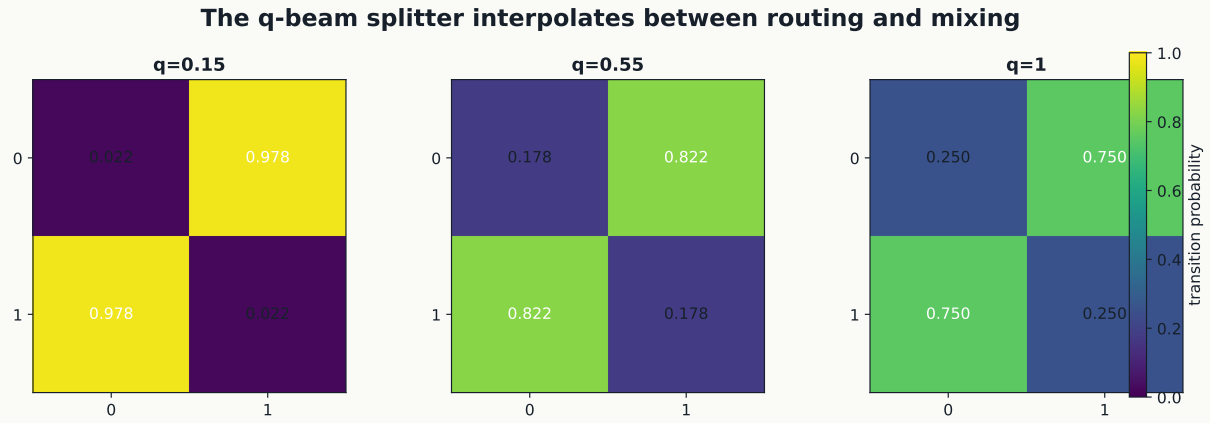


Figure 6.2: Squared two-channel kernels. The coherent transform remains orthogonal, while the measured channel becomes a q-dependent bistochastic matrix.

Chapter 7

Bistochastic measurement and entropy

Define

$$\boxed{P_{jm}(q) = |\mathcal{R}_N(q)_{jm}|^2.}$$

Theorem 7.1 (Bistochastic q-Racah channel)**.** *For every finite recoupling matrix,*

$$P_{jm} \geq 0, \qquad \sum_m P_{jm} = 1, \qquad \sum_j P_{jm} = 1.$$

Consequently, for every probability row vector p,

$$\boxed{\mathrm{H}(pP) \geq \mathrm{H}(p),}$$

where H *is Shannon entropy.*

Proof. The row and column sums are the squared norms of rows and columns of the orthogonal matrix. Every doubly stochastic matrix is a convex combination of permutation matrices. Shannon entropy is concave and invariant under permutations, so it cannot decrease. □

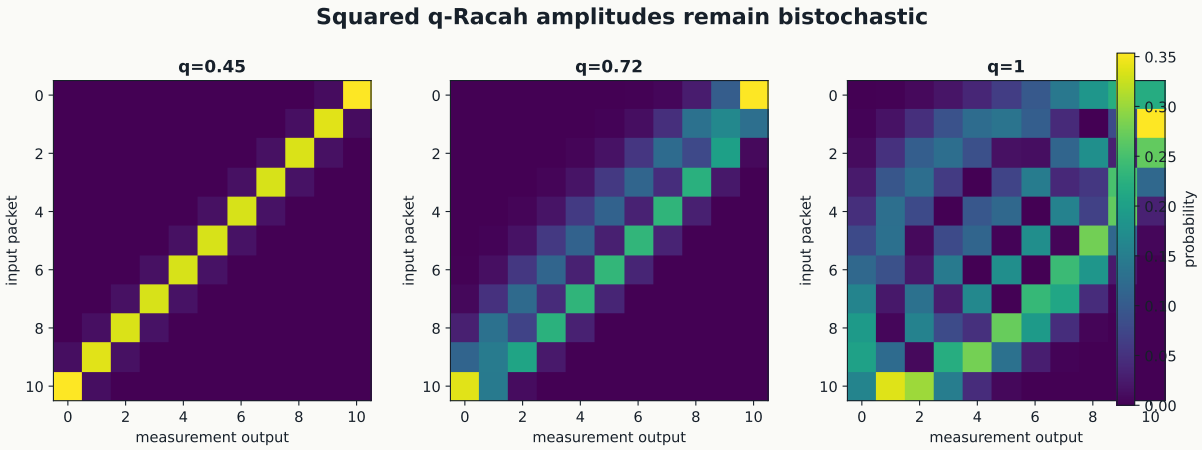


Figure 7.1: Bistochastic q-Racah kernels. Every amplitude matrix produces a classical measurement channel with unit row and column sums.

Theorem 7.2 (Monotone one-excitation mixing)**.** *For the equal-shape one-excitation channel and* $0 < q \leq 1$*, the row entropy and the spectral mixing gap increase with* q*.*

Proof. The smaller row probability is

$$p(q) = c_q^2 = \frac{q^2}{(1+q^2)^2}.$$

It increases from zero to $1/4$. The binary entropy

$$h(p) = -p \log p - (1-p) \log(1-p)$$

is increasing for $0 \leq p \leq 1/2$. The two eigenvalues of the channel are 1 and $2p-1$, so the spectral gap is

$$\boxed{1 - |2p-1| = 2p = \frac{2q^2}{(1+q^2)^2},}$$

which is also increasing on $(0, 1]$. □

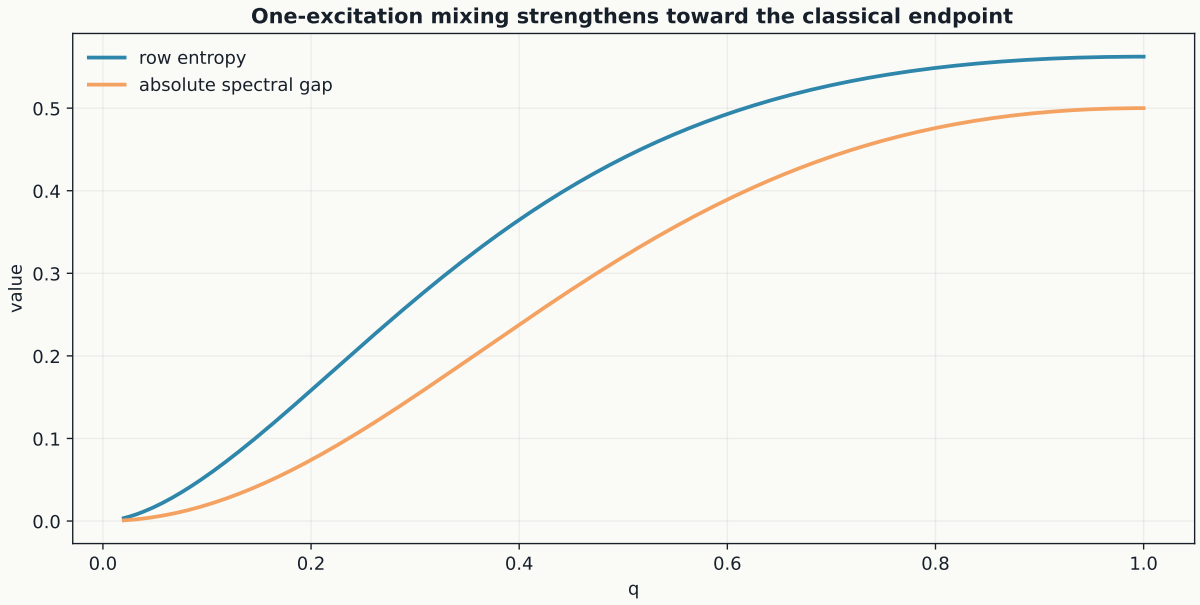


Figure 7.2: Entropy and spectral gap in the exact two-channel model. Both increase monotonically with q.

Boundary statement 7.3 (No universal entropy monotonicity in q). For higher excitation, each fixed q still gives a bistochastic channel and therefore increases entropy relative to its input. However, the row entropy need not be monotone as q varies. At $N = 3$ and equal half-shapes, the second row entropy decreases from approximately 1.24488 to 1.23710 between $q \approx 0.7772$ and $q \approx 0.7884$. This negative control prevents the one-excitation theorem from being promoted to all packet levels.

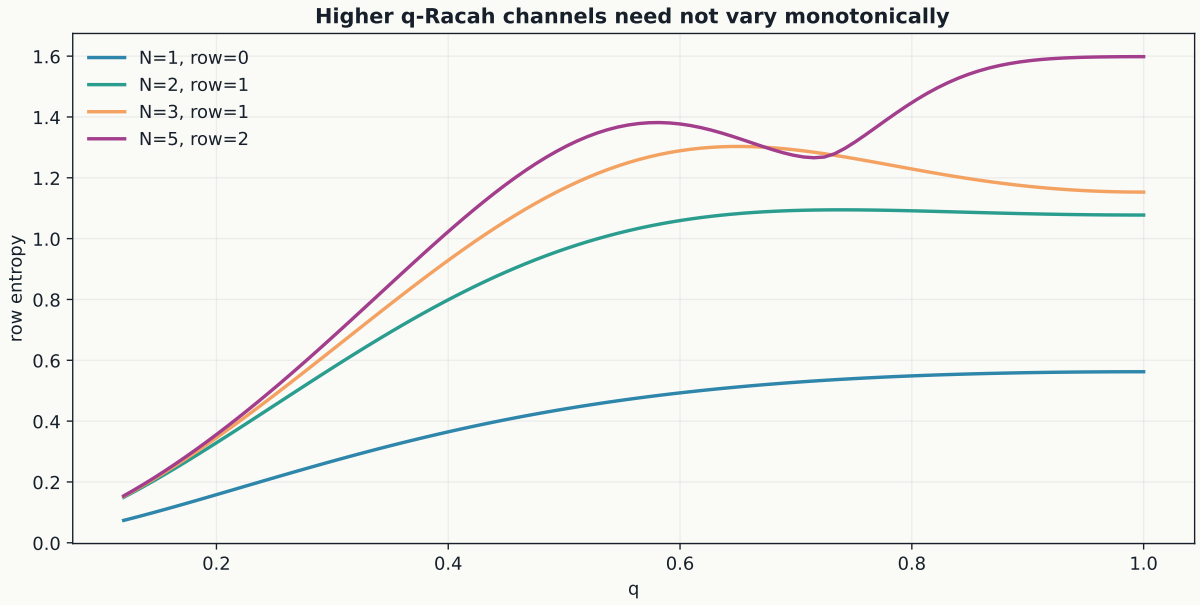


Figure 7.3: Higher-excitation entropy counterexample. A selected row of the q-Racah measurement channel is not monotone throughout the deformation interval.

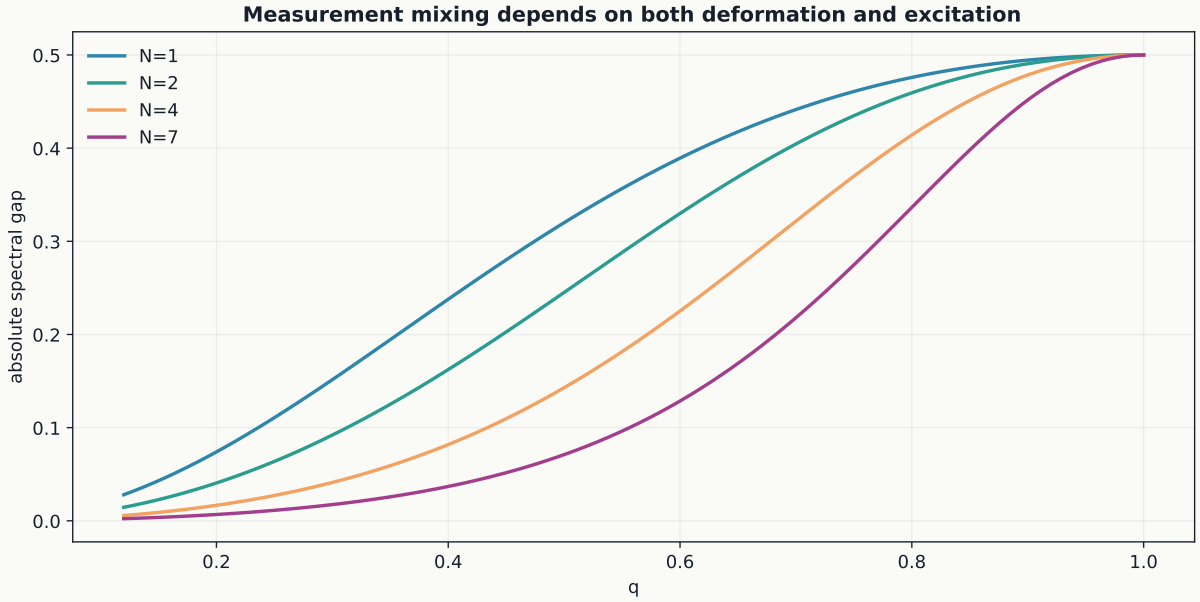


Figure 7.4: Spectral mixing beyond two channels. The q-dependence of the second eigenvalue becomes level dependent and need not follow a universal monotone law.

Part IV

Deformation geometry

Chapter 8

Orthogonal transport in q

Write

$$q = \mathrm{e}^{-\theta}, \qquad \theta \geq 0.$$

Choose a continuous sign gauge for the eigenvectors away from degeneracies and define

$$R_N(\theta) = \mathcal{R}_N(\mathrm{e}^{-\theta}).$$

Theorem 8.1 (Skew deformation generator)**.** *The matrix*

$$\boxed{A_N(\theta) = R_N(\theta)^{\mathsf{T}} \partial_\theta R_N(\theta)}$$

is skew-symmetric, and

$$\boxed{\partial_\theta R_N = R_N A_N.}$$

Proof. Differentiate

$$R_N^{\mathsf{T}} R_N = \mathrm{I}.$$

This gives

$$(\partial_\theta R_N)^{\mathsf{T}} R_N + R_N^{\mathsf{T}} \partial_\theta R_N = 0,$$

so $A_N^{\mathsf{T}} = -A_N$. Multiplying the definition of A_N on the left by R_N gives the tangent equation. □

Define the finite deformation metric

$$\boxed{g_N(\theta) = \frac{1}{2} \|A_N(\theta)\|_F^2 .}$$

It is invariant under a constant orthogonal change of either packet atlas. A q-dependent sign jump is not a smooth gauge and must be removed before differentiation.

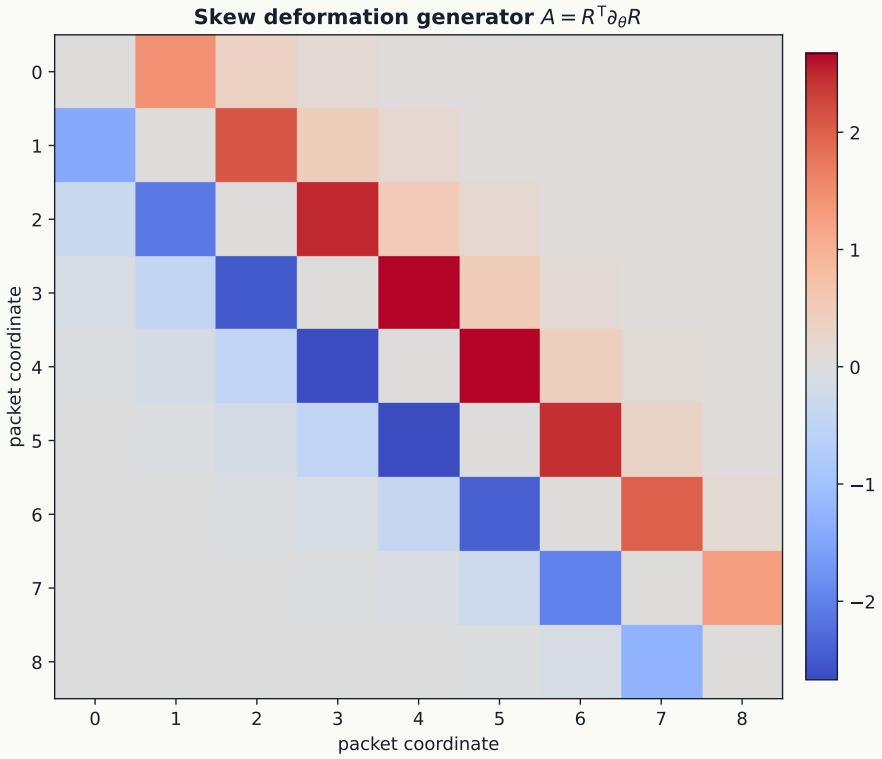


Figure 8.1: Skew generator of the q-flow. The antisymmetric matrix records which packet coordinates exchange amplitude under an infinitesimal change of deformation.

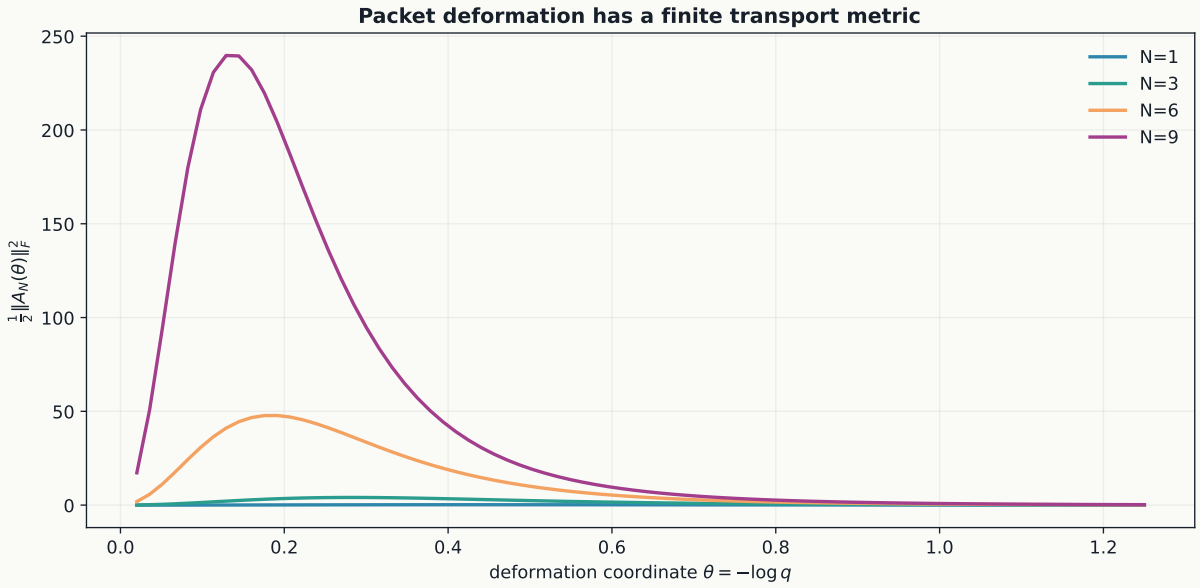


Figure 8.2: Deformation metric across excitation. The sensitivity of the complete recoupling atlas grows with multiplicity dimension.

Chapter 9

Classical and routing limits

The q-Hahn and q-Racah systems have two useful endpoints.

At $q \to 1$, symmetric q-numbers satisfy

$$[x]_q = x + O((\log q)^2),$$

so the packet algebra, Casimir spectra, lowest-weight vectors, and recoupling matrices converge to their ordinary $\mathfrak{su}(1,1)$ counterparts.

At $q \downarrow 0$, no uniform finite limit exists for every unrenormalized q-number expression. Nevertheless, selected normalized packet transforms have finite routing limits. The one-excitation beam splitter is the simplest exact example.

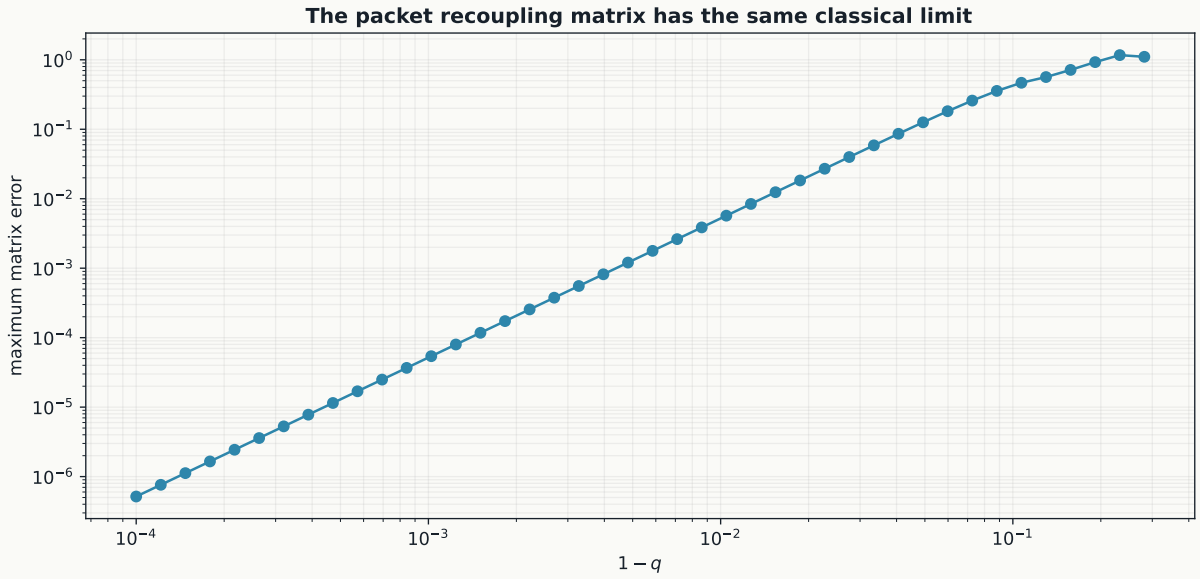


Figure 9.1: Classical limit of the matrix construction. After sign alignment, the complete q-Racah packet transform converges to the classical Racah transform.

Part V

Four-body q-associators

Chapter 10

Five complete q-Casimir tree atlases

For four packets, the multiplicity dimension is

$$\boxed{\dim \mathcal{M}_N^{(4)} = \binom{N+2}{2}.}$$

Consider the five binary coupling trees

$$(((12)3)4), \quad ((1(23))4), \quad (1((23)4)), \quad (1(2(34))), \quad ((12)(34)).$$

Each tree determines two commuting intermediate q-Casimirs. Their joint eigenvectors form a complete orthonormal basis B_r of the same multiplicity space.

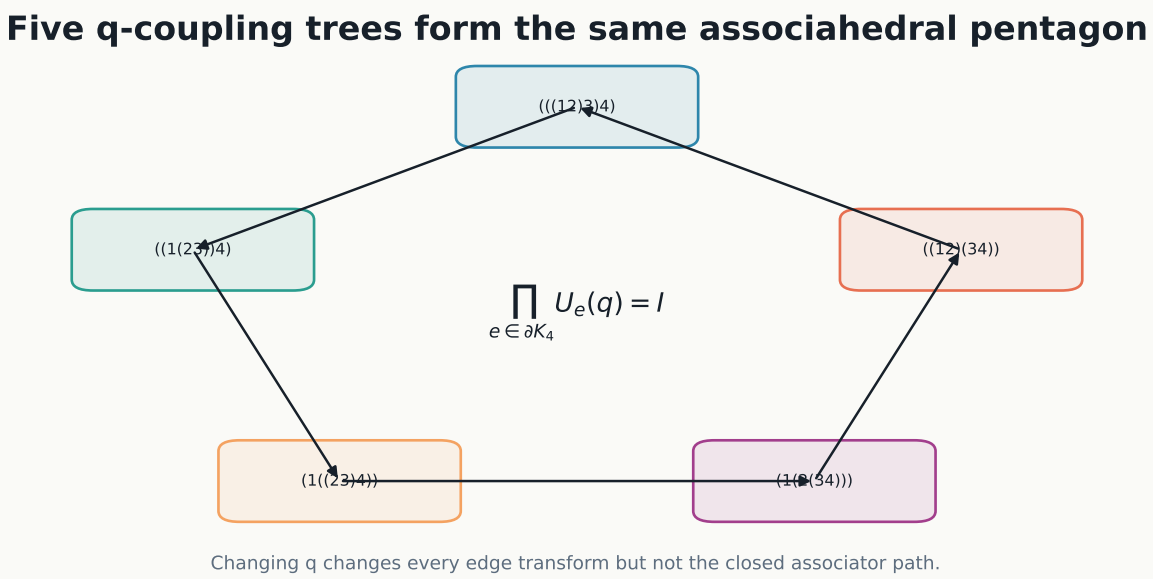


Figure 10.1: The q-deformed associahedral pentagon. Every vertex is a complete joint-Casimir atlas. Neighboring vertices are related by finite q-Racah packet transforms.

Theorem 10.1 (Finite q-pentagon certificate)**.** *Let*

$$U_{r,r+1} = B_r^{\mathsf{T}} B_{r+1},$$

with indices taken modulo five. Then

$$\boxed{U_{01}U_{12}U_{23}U_{34}U_{40} = \mathrm{I}.}$$

Proof. Insert the definitions and use completeness:

$$B_0^{\mathsf{T}} B_1 B_1^{\mathsf{T}} B_2 \cdots B_4^{\mathsf{T}} B_0 = B_0^{\mathsf{T}} B_0 = \mathrm{I}.$$

□

This linear-algebra proof explains why the numerical pentagon is a strong implementation test. Any wrong projector, packet order, omitted vector, or inconsistent tree basis prevents the telescoping product from closing.

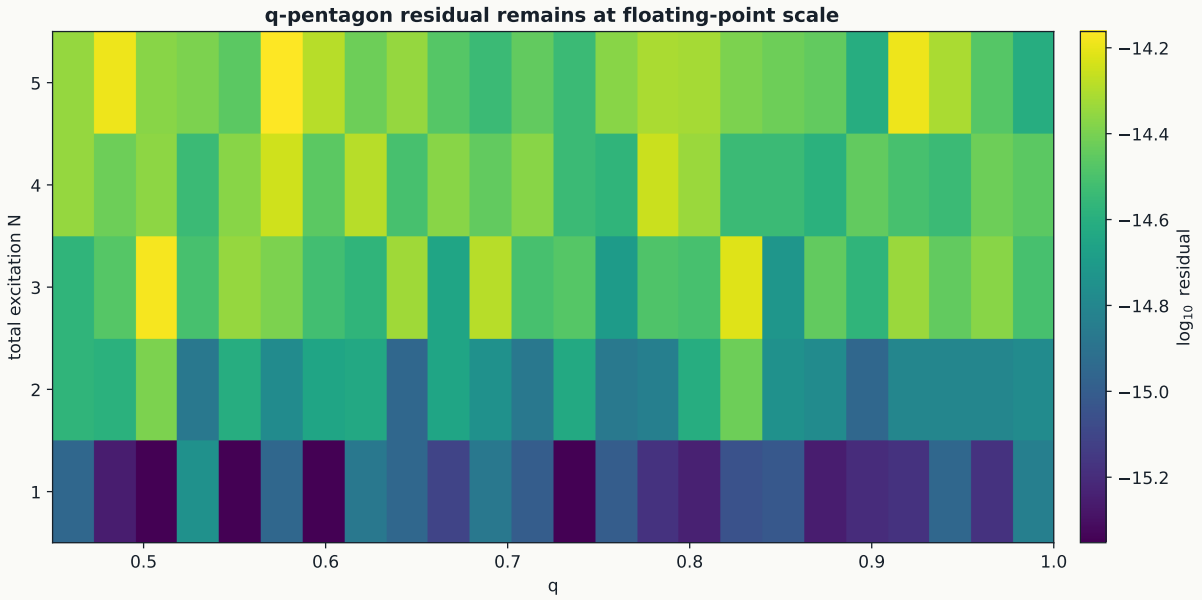


Figure 10.2: Pentagon residual over q and excitation. The closure error remains at floating-point scale through the tested multiplicity dimensions.

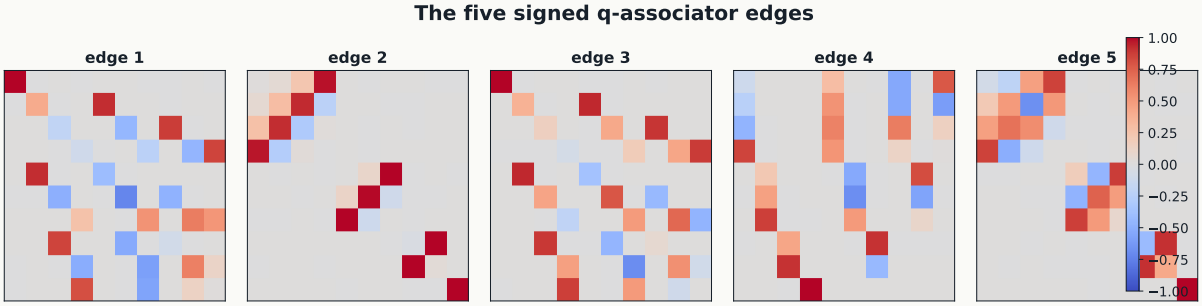


Figure 10.3: Five q-associator edges. Each edge has its own signed amplitude pattern, but their ordered product reconstructs the initial tree atlas.

Chapter 11

Deforming the associator

Each edge matrix depends on $\theta = -\log q$. In a continuous gauge, define

$$A_r(\theta) = U_r(\theta)^{\mathsf{T}} \partial_\theta U_r(\theta).$$

Every A_r is skew-symmetric. Differentiating the pentagon identity gives a transported sum of edge generators equal to zero. This is the finite flatness equation for the q-associator loop.

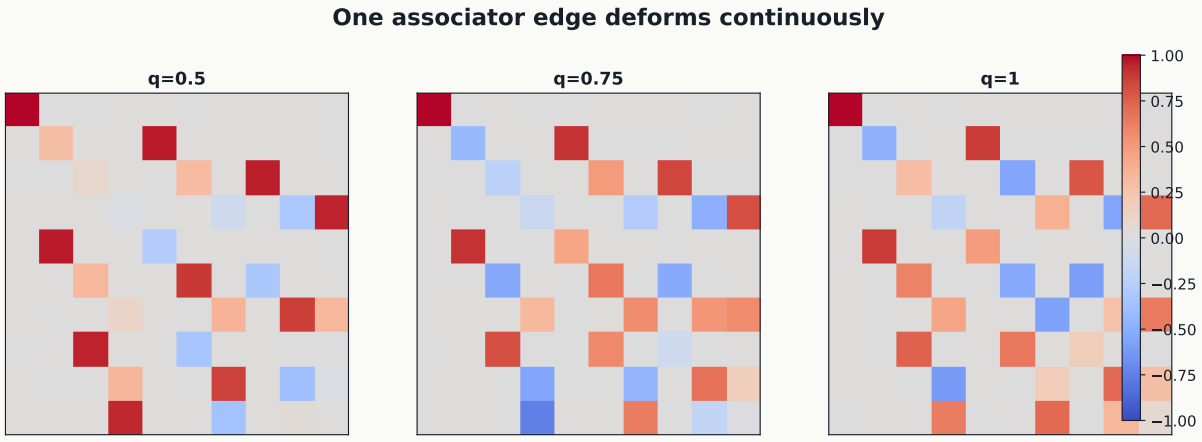


Figure 11.1: Deformation of the five associator edges. The q-flow redistributes local generator norm around the pentagon while preserving exact global closure.

Part VI

Finite compilation

Chapter 12

Givens factorization of a q-Racah packet

Theorem 12.1 (Finite rotation compiler)**.** *Every real orthogonal $d \times d$ q-Racah packet matrix admits a factorization*

$$\boxed{R = G_1^{\mathsf{T}} \cdots G_L^{\mathsf{T}} D,}$$

where D is diagonal with entries ± 1 and every G_j is a two-coordinate Givens rotation. One may choose

$$\boxed{L \leq \frac{d(d-1)}{2}.}$$

Proof. Apply row Givens rotations from the lower left upward to eliminate all entries below the diagonal. Orthogonality forces the resulting upper triangular matrix to be diagonal with entries ± 1. Reversing the rotations reconstructs R. □

The compiler acts in the multiplicity packet. It does not require a dense gate acting on the full tensor-product level.

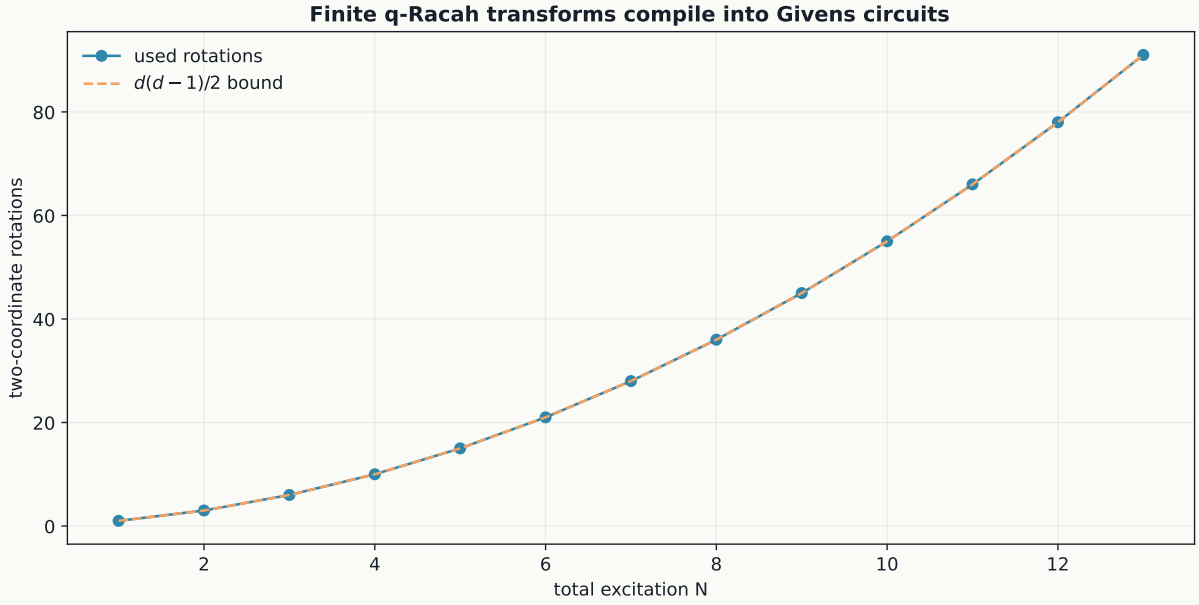


Figure 12.1: Rotation count. The direct compiler reaches the worst-case triangular count only when every subdiagonal elimination is needed.

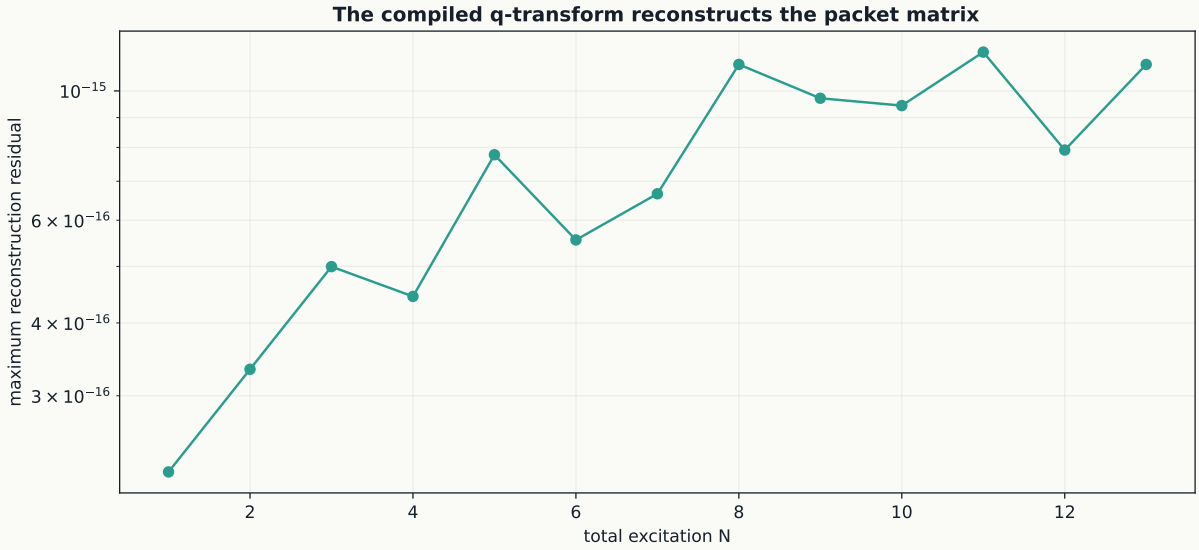


Figure 12.2: Compiler reconstruction residual. The finite rotation product reproduces the original q-Racah transform to numerical precision.

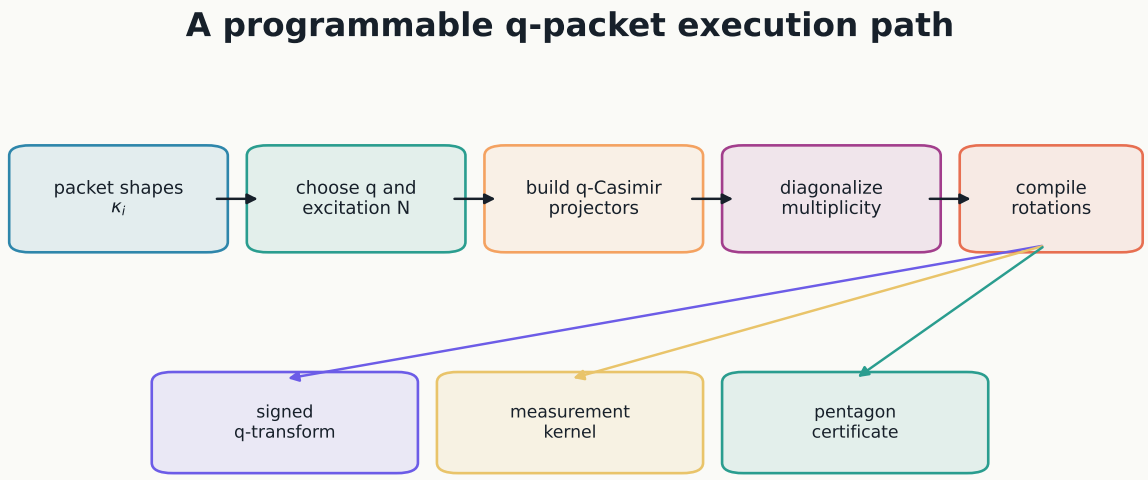


Figure 12.3: Programmable q-packet execution path. The compiler begins with shapes, q, and total excitation, constructs only the required multiplicity projectors, and emits signed transforms, measurement kernels, or pentagon certificates.

Part VII

Scalar q-special functions

Chapter 13

Terminating q-Racah function

For the q-Pochhammer symbol

$$(a;q)_n = \prod_{j=0}^{n-1}(1-aq^j),$$

define

$$\boxed{R_n(\mu(y);\alpha,\beta,\gamma,\delta \mid q) = {}_4\phi_3\left(\begin{matrix}q^{-n},\alpha\beta q^{n+1},q^{-y},\gamma\delta q^{y+1}\\ \alpha q,\beta\delta q,\gamma q\end{matrix};q,q\right),}$$

where

$$\mu(y) = q^{-y}+\gamma\delta q^{y+1}.$$

The series terminates at n. A finite orthogonal family arises when one denominator parameter is q^{-N}.

Theorem 13.1 (q-Racah duality)**.** *The terminating function satisfies*

$$\boxed{R_n(\mu(y);\alpha,\beta,\gamma,\delta \mid q) = R_y(\mu(n);\gamma,\delta,\alpha,\beta \mid q).}$$

Proof. After exchanging n with y and (α, β) with (γ, δ), the four numerator q-Pochhammer factors are merely permuted. The three denominator factors are also permuted because $\delta\beta q = \beta\delta q$. Every summand is unchanged. □

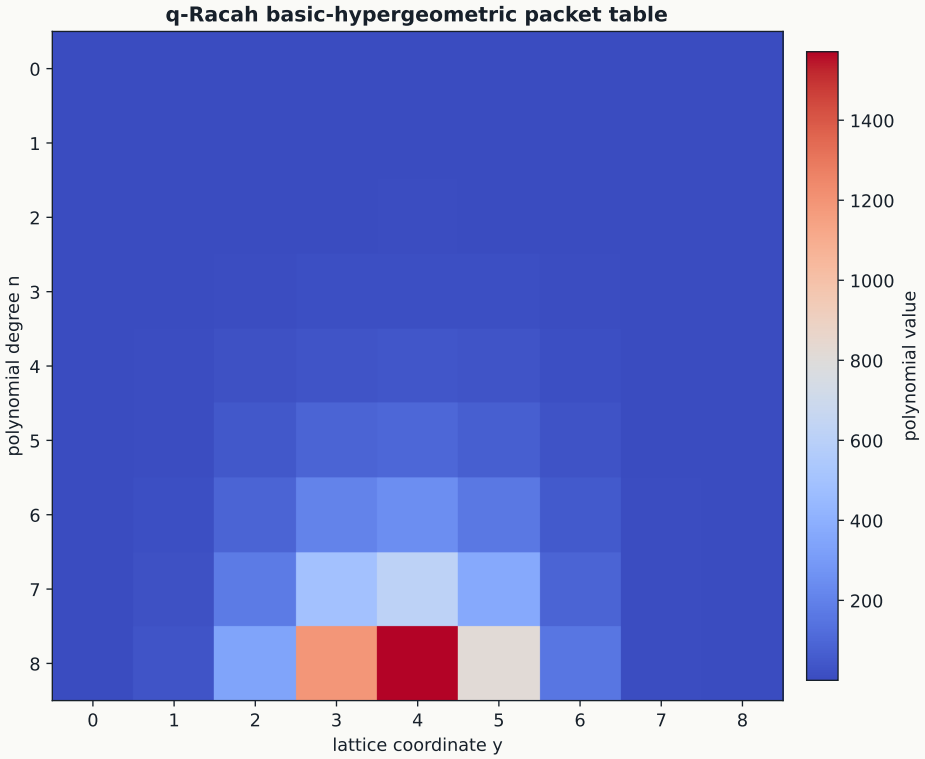


Figure 13.1: A terminating q-Racah table. Degrees and lattice coordinates appear on equal footing, which makes duality visible before normalization by the orthogonality weights.

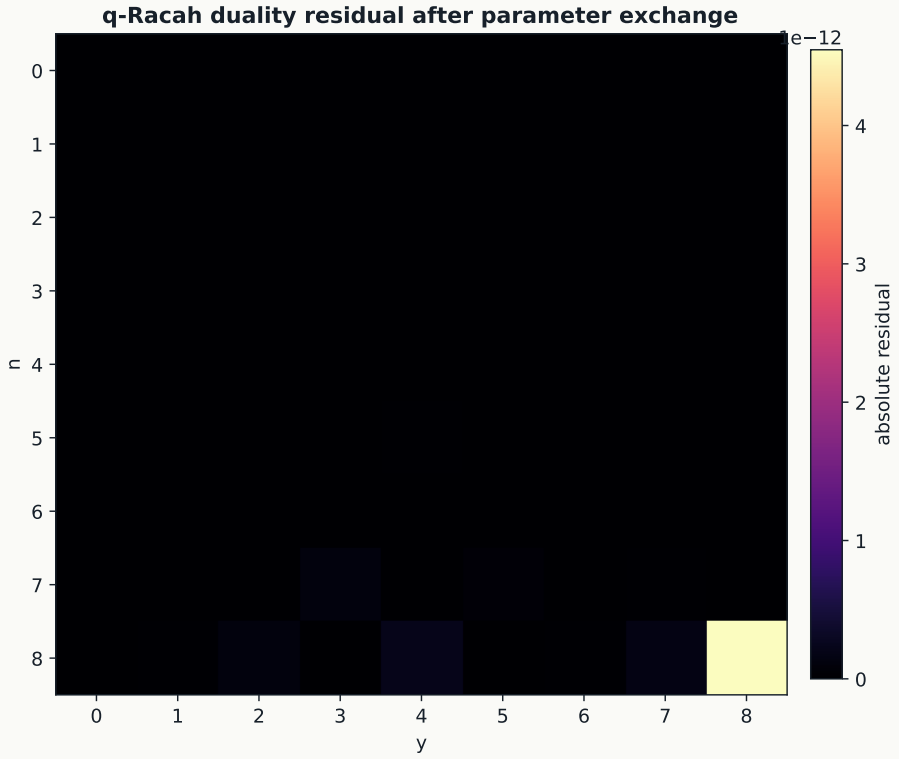


Figure 13.2: Duality residual. The parameter-exchanged degree-coordinate identity holds at floating-point precision.

Theorem 13.2 (Termwise classical limit)**.** *Set*

$$\alpha = q^A, \quad \beta = q^B, \quad \gamma = q^G, \quad \delta = q^D.$$

For fixed n and y,

$$\boxed{\lim_{q\to 1^-} R_n(\mu(y); q^A, q^B, q^G, q^D \mid q) = {}_4F_3\left(\begin{matrix}-n, n+A+B+1, -y, y+G+D+1 \\ A+1, B+D+1, G+1\end{matrix}; 1\right).}$$

Proof. The q-series is finite. For every fixed integer r,

$$\frac{(q^x;q)_r}{(1-q)^r} \longrightarrow (x)_r.$$

Apply this limit to each q-Pochhammer factor in every summand and cancel the equal powers of $1-q$. □

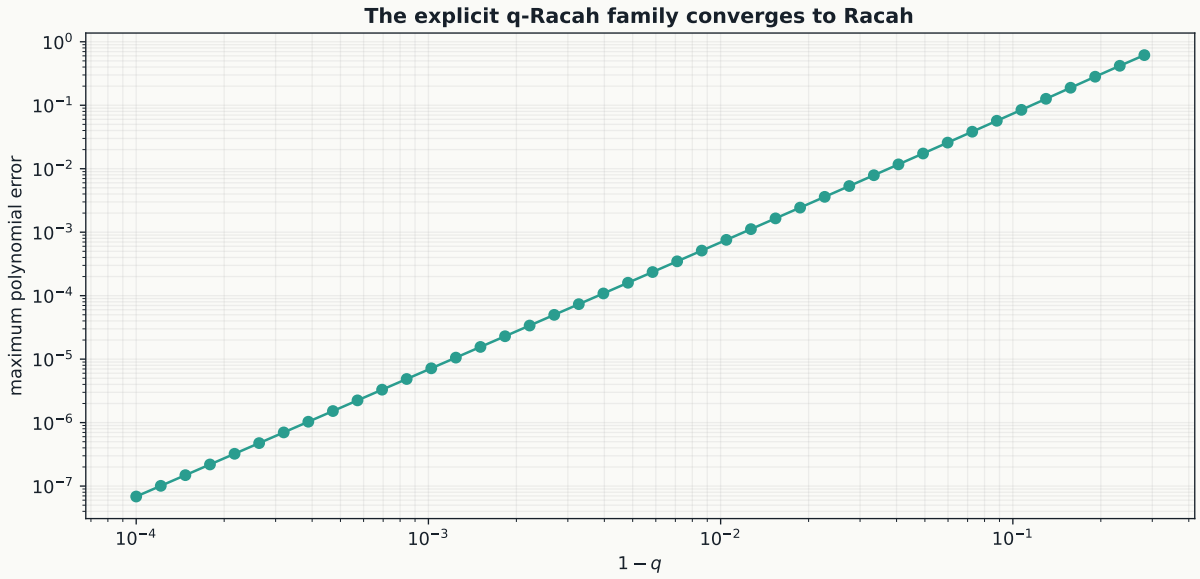


Figure 13.3: q-Racah to Racah. The finite basic-hypergeometric polynomial converges termwise to its ordinary hypergeometric counterpart.

Chapter 14

Askey-Wilson transfer function

For $z \in \mathbb{C}^{\times}$, define the normalized Askey-Wilson function

$$\boxed{\mathcal{A}_n(z) = {}_4\phi_3\left(\begin{matrix} q^{-n}, abcdq^{n-1}, az, az^{-1} \\ ab, ac, ad \end{matrix}; q, q\right).}$$

Define the q-difference operator

$$(\mathcal{L}f)(z) = A(z)(f(qz) - f(z)) + A(z^{-1})(f(q^{-1}z) - f(z)) + \left(1 + \frac{abcd}{q}\right) f(z),$$

where

$$A(z) = \frac{(1-az)(1-bz)(1-cz)(1-dz)}{(1-z^2)(1-qz^2)}.$$

Theorem 14.1 (Askey-Wilson eigenfunction law)**.** *For every nonnegative integer n,*

$$\boxed{\mathcal{L}\mathcal{A}_n = \left(q^{-n} + abcdq^{n-1}\right)\mathcal{A}_n.}$$

This is the continuous spectral lift of the finite q-Racah transfer system. The q-Racah family lives on a finite quadratic q-lattice. The Askey-Wilson family lives on the continuous variable $x = (z + z^{-1})/2$ and is governed by a second-order q-difference operator.

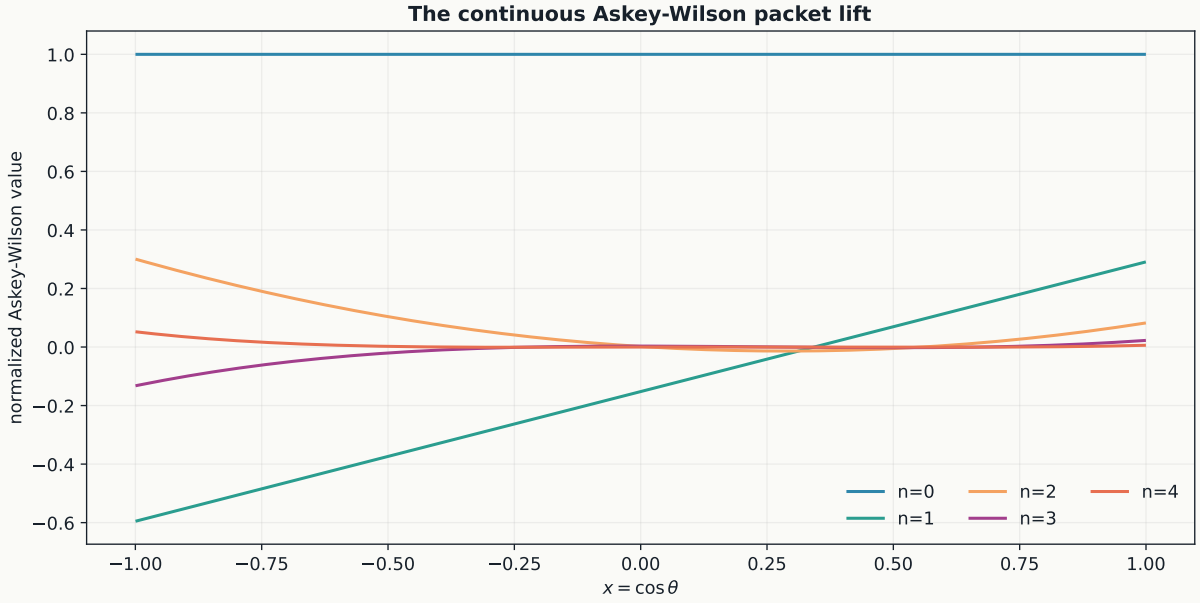


Figure 14.1: Askey-Wilson packet functions. Several degrees are plotted on the unit-circle coordinate $z = e^{i\theta}$.

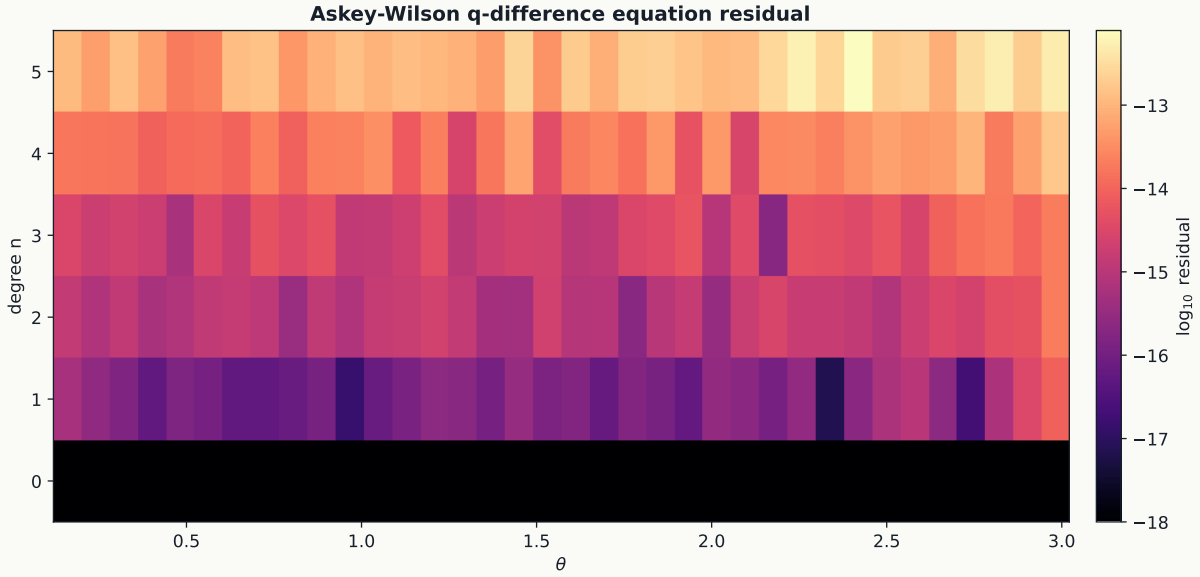


Figure 14.2: q-difference certificate. Direct substitution of the generated functions into the Askey-Wilson operator leaves only numerical roundoff.

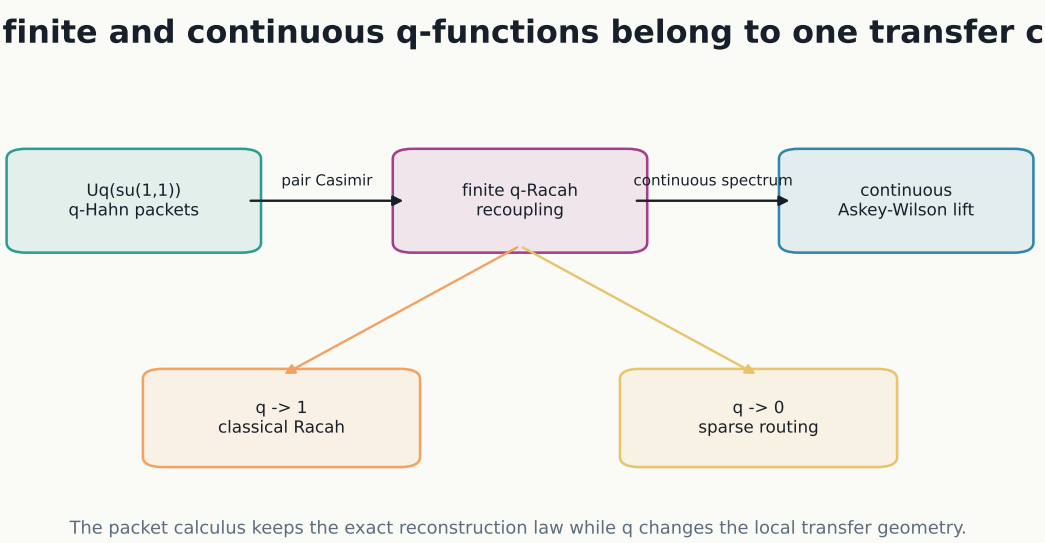


Figure 14.3: Finite and continuous q-functions. q-Hahn packets generate coupled bases, q-Racah functions generate finite recoupling, and Askey-Wilson functions provide the continuous spectral lift.

Part VIII

Quantum and computational interpretation

Chapter 15

A reversible q-shape module

The q-Racah transform can be interpreted as a finite coherent basis change. The signed amplitude matrix is reversible. Squaring the entries and measuring the target basis produces the bistochastic channel.

The deformation parameter can be treated as a compile-time or control-time variable. A program may request:

1. a lowest-weight q-Hahn seed,
2. a left or right pair-Casimir atlas,
3. a signed q-Racah transform,
4. its measured bistochastic kernel,
5. a q-deformation tangent generator,
6. a four-body associator certificate,
7. a finite Givens circuit.

This module can coexist with the logarithmic number Hamiltonian

$$H\,|n\rangle = \log n\,|n\rangle$$

used in the attached zeta-governed quantum dynamics. The logarithmic system supplies arithmetic time phases. The q-Racah system supplies finite shape coupling. Their tensor product is mathematically legitimate, but no theorem here identifies the q-Casimir spectrum with the nontrivial zeta zeros.

Chapter 16

What the deformation adds

The undeformed theory already has exact Racah recoupling and pentagon coherence. The q-coordinate adds four further layers.

First, it creates a continuous path between distinct packet-routing geometries.

Second, it turns basis sensitivity into the explicit skew generator $A_N(\theta)$.

Third, it produces q-dependent bistochastic measurement channels whose mixing behavior can be studied independently of coherent reversibility.

Fourth, it places the finite q-Racah packet and the continuous Askey-Wilson function inside one q-difference hierarchy.

The paper's strongest new operational object is therefore not a renamed polynomial. It is the complete tuple

$$\boxed{\left(\mathcal{M}_N, \mathcal{C}_{12}, \mathcal{C}_{23}, \mathcal{R}_N(q), P_N(q), A_N(\theta), \{U_r(q)\}_{r=0}^{4}\right).}$$

This tuple retains the object space, the two coordinate atlases, exact synthesis, the measured channel, the deformation tangent, and the associator loop.

Part IX

Verification and formulas

Chapter 17

Formula registry

Object	Exact formula or certificate
Symmetric q-number	$[x]_q = (q^x - q^{-x})/(q - q^{-1})$
Local q-algebra	$[K_-, K_+] = [2K_0]_q$
Coproduct	$\Delta K_\pm = K_\pm \otimes q^{K_0} + q^{-K_0} \otimes K_\pm$
q-Casimir	$\mathcal{C}_q = [K_0]_q\,[K_0 - 1]_q - K_+K_-$
Scalar Casimir	$\mathcal{C}_q = [\kappa]_q\,[\kappa - 1]_q\,\mathrm{I}$
Multiplicity dimension	$\dim \mathcal{M}_N^{(m)} = \binom{N+m-2}{m-2}$

Object	Exact formula or certificate
q-Hahn ratio	$\frac{c_{n+1}}{c_n} = -q^{-(N-1+\kappa_1+\kappa_2)} \times \sqrt{\frac{[N-n]_q\,[2\kappa_2+N-n-1]_q}{[n+1]_q\,[2\kappa_1+n]_q}}$
Pair spectrum	$\lambda_j^{ab} = [\kappa_a+\kappa_b+j]_q\,[\kappa_a+\kappa_b+j-1]_q$
Recoupling	$\mathcal{R}_N = L_N^{\mathsf{T}} R_N$
Bistochastic kernel	$P_{jm} = \lvert\mathcal{R}_{jm}\rvert^2$
Deformation generator	$A_N = \mathcal{R}_N^{\mathsf{T}}\partial_\theta \mathcal{R}_N$, $A_N^{\mathsf{T}} = -A_N$
Pentagon	$U_{01}U_{12}U_{23}U_{34}U_{40} = \mathrm{I}$
q-Racah duality	$R_n(y;\alpha,\beta,\gamma,\delta) = R_y(n;\gamma,\delta,\alpha,\beta)$
Askey-Wilson law	$\mathcal{L}\mathcal{A}_n = (q^{-n} + abcdq^{n-1})\mathcal{A}_n$
Givens bound	$L \le d(d-1)/2$

Chapter 18

Numerical certificate table

Check	Result	Largest recorded residual or witness
Local q-algebra	pass	2.05×10^{-12}
q-Hahn lowest vector	pass	2.22×10^{-16}
Scalar q-Casimir	pass	5.88×10^{-15}
q-Racah orthogonality	pass	2.22×10^{-15}
Pair spectrum	pass	4.55×10^{-13}
Opposing-Casimir tridiagonality	pass	5.21×10^{-12}
Closed q-beam splitter	pass	2.22×10^{-16}
Bistochastic kernel	pass	2.00×10^{-15}
Entropy nondecrease for fixed kernel	pass	no violation in 80 trials
One-excitation entropy monotonicity	pass	no negative increment
Higher-level monotonicity control	pass	decrement -7.78×10^{-3} found
Matrix classical limit	pass	2.82×10^{-7}
q-Hahn classical limit	pass	4.28×10^{-6}
Routing limit	pass	1.00×10^{-5}
Skew generator	pass	1.48×10^{-8} raw gauge residual
Tangent equation	pass	5.92×10^{-9}
Four-tree commutation	pass	7.96×10^{-13}
Joint diagonalization	pass	7.33×10^{-14}
q-pentagon	pass	3.22×10^{-15}
Givens compilation	pass	8.88×10^{-16}
q-Racah duality	pass	2.84×10^{-14}
q-Racah polynomial limit	pass	4.27×10^{-6}
Askey-Wilson q-difference equation	pass	5.91×10^{-13}

The independent test suite repeats twenty properties through direct calls to the mathematical functions. It does not merely read the built-in check ledger.

Chapter 19

Research directions and boundaries

The present paper suggests several focused questions.

1. Determine closed shape-parameter formulas for the full q-Racah overlap in the symmetric q-number convention used here.
2. Characterize the deformation metric $g_N(\theta)$ asymptotically as N grows.
3. Identify the precise packet conditions under which row entropy is monotone in q.
4. Extend the finite q-Casimir construction to continuous-series Askey-Wilson transforms.
5. Develop a q-adic or root-of-unity packet theory where semisimplicity may fail.
6. Compile the associator loop with locality constraints rather than unrestricted Givens rotations.
7. Couple the q-shape module to the logarithmic number Hamiltonian and determine which observables remain exactly factorized.

The first five items are special-function and representation-theoretic questions. The last two are architecture and quantum-model questions. None is solved by the numerical figures alone.

Chapter 20

Conclusion

The q-deformation does not discard the earlier packet calculus. It enriches it.

The seed is still a positive shape packet. The object space is still resolved by a lowest-weight projector. Intermediate Casimirs still create exact coordinate atlases. The recoupling matrix still supplies analysis and synthesis. The squared entries still define a measurement channel. Four coupling trees still close around a pentagon. The finite transform still compiles into two-coordinate rotations.

What changes is the geometry of every layer. q alters the ladder, pair spectra, packet weights, recoupling amplitudes, stochastic mixing, and associator edges. The q-Racah and Askey-Wilson families are the scalar special functions that record this deformation.

The result is a complete q-packet module:

$$\boxed{\text{q-Hahn seed} \longrightarrow \text{q-Casimir atlases} \longrightarrow \text{q-Racah transform} \longrightarrow \text{q-pentagon} \longrightarrow \text{Askey-Wilson lift.}}$$

The classical theory is recovered at $q = 1$. Selected packet transforms acquire sparse routing limits as q approaches zero. Between those endpoints, the skew generator provides a local differential description of the changing coordinate system.

Chapter A

Selected numerical tables

Table A.1: Equal-shape one-excitation q-beam splitter.

q	c_q	s_q	row entropy	spectral gap
0.10	0.099010	0.995086	0.055094	0.019606
0.25	0.235294	0.971924	0.214014	0.110727
0.50	0.400000	0.916515	0.439670	0.320000
0.75	0.480000	0.877268	0.539759	0.460800
1.00	0.500000	0.866025	0.562335	0.500000

Table A.2: Four-body q-pentagon residuals at $N = 4$.

q	multiplicity dimension	maximum residual
0.50	15	3.22×10^{-15}
0.70	15	4.00×10^{-15}
0.85	15	3.55×10^{-15}
1.00	15	3.44×10^{-15}

Chapter B

Complete Python implementation

The following source generates every figure, numerical table, built-in check, and manifest used in this paper.

```
from __future__ import annotations

import csv
import hashlib
import json
import math
from dataclasses import dataclass
from functools import lru_cache
from itertools import product
from pathlib import Path
from typing import Iterable, Sequence

import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from matplotlib.patches import Circle, FancyArrowPatch, FancyBboxPatch, Polygon
import numpy as np
from scipy.linalg import null_space

ROOT = Path(__file__).resolve().parent
FIGURES = ROOT / "figures"
RESULTS = ROOT / "results"
TABLES = ROOT / "tables"
for directory in (FIGURES, RESULTS, TABLES):
    directory.mkdir(parents=True, exist_ok=True)

```

```
INK = "#17202A"
BLUE = "#2E86AB"
TEAL = "#2A9D8F"
ORANGE = "#F4A261"
MAGENTA = "#A23E8C"
RED = "#E76F51"
YELLOW = "#E9C46A"
PURPLE = "#6C5CE7"
MUTED = "#5D6D7E"
PAPER = "#FAFAF7"
GRID = "#D5D8DC"
WHITE = "#FFFFFF"
COLORS = [BLUE, TEAL, ORANGE, MAGENTA, RED, YELLOW, PURPLE]

plt.rcParams.update(
    {
        "figure.facecolor": PAPER,
        "axes.facecolor": PAPER,
        "axes.edgecolor": INK,
        "axes.labelcolor": INK,
        "xtick.color": INK,
        "ytick.color": INK,
        "text.color": INK,
        "font.size": 10.5,
        "axes.titlesize": 12.2,
        "axes.titleweight": "bold",
        "legend.frameon": False,
        "savefig.bbox": "tight",
        "savefig.facecolor": PAPER,
    }
)


def jsonable(value):
    """Convert NumPy scalar containers into JSON-safe Python objects."""
    if isinstance(value, np.generic):
        return value.item()
    if isinstance(value, dict):
        return {str(key): jsonable(item) for key, item in value.items()}
    if isinstance(value, (list, tuple)):
        return [jsonable(item) for item in value]
    return value


def save_figure(fig: plt.Figure, stem: str) -> None:
    fig.savefig(FIGURES / f"{stem}.pdf")
    fig.savefig(FIGURES / f"{stem}.png", dpi=220)
    plt.close(fig)


def diagram_axes(figsize: tuple[float, float] = (12.6, 6.1)) -> tuple[plt.Figure, plt.Axes]:
    fig, ax = plt.subplots(figsize=figsize)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.axis("off")
    return fig, ax


def box(
    ax: plt.Axes,
    xy: tuple[float, float],
    width: float,
    height: float,
    text: str,
    face: str = WHITE,
    edge: str = INK,
    fontsize: float = 10.5,
) -> None:
    patch = FancyBboxPatch(
        xy,
        width,
```

```
        height,
        boxstyle="round,pad=.012,rounding_size=.018",
        facecolor=face,
        edgecolor=edge,
        linewidth=1.6,
    )
    ax.add_patch(patch)
    ax.text(xy[0] + width / 2, xy[1] + height / 2, text, ha="center", va="center", fontsize=fontsize)


def arrow(
    ax: plt.Axes,
    start: tuple[float, float],
    end: tuple[float, float],
    color: str = INK,
    text: str | None = None,
    text_offset: tuple[float, float] = (0.0, 0.035),
) -> None:
    patch = FancyArrowPatch(start, end, arrowstyle="-|>", mutation_scale=13, linewidth=1.5, color=color)
    ax.add_patch(patch)
    if text:
        ax.text(
            (start[0] + end[0]) / 2 + text_offset[0],
            (start[1] + end[1]) / 2 + text_offset[1],
            text,
            ha="center",
            va="center",
            fontsize=9.5,
            color=color,
        )


# ---------------------------------------------------------------------------
# q-arithmetic and positive discrete-series packet spaces
# ---------------------------------------------------------------------------


def q_number(x: float, q: float) -> float:
    if q <= 0:
        raise ValueError("q must be positive")
    if abs(q - 1.0) < 1e-12:
        return float(x)
    theta = math.log(q)
    return float(math.sinh(x * theta) / math.sinh(theta))


def q_factorial(n: int, q: float) -> float:
    result = 1.0
    for j in range(1, n + 1):
        result *= q_number(float(j), q)
    return result


def q_binomial(n: int, r: int, q: float) -> float:
    if r < 0 or r > n:
        return 0.0
    return q_factorial(n, q) / (q_factorial(r, q) * q_factorial(n - r, q))


def q_rising(x: float, n: int, q: float) -> float:
    result = 1.0
    for j in range(n):
        result *= q_number(x + j, q)
    return result


@lru_cache(maxsize=None)
def compositions(total: int, parts: int) -> tuple[tuple[int, ...], ...]:
    if total < 0:
        return tuple()
    if parts == 1:
```

```
        return ((total,),)
    out: list[tuple[int, ...]] = []
    for first in range(total + 1):
        for rest in compositions(total - first, parts - 1):
            out.append((first,) + rest)
    return tuple(out)


def excitation_dimension(total: int, factors: int) -> int:
    return math.comb(total + factors - 1, factors - 1)


def multiplicity_dimension(total: int, factors: int) -> int:
    if factors < 2:
        return 1
    return math.comb(total + factors - 2, factors - 2)


def q_local_matrices(dimension: int, kappa: float, q: float) -> tuple[np.ndarray, np.ndarray, np.ndarray]:
    k0 = np.diag([n + kappa for n in range(dimension)])
    kp = np.zeros((dimension, dimension), dtype=float)
    km = np.zeros((dimension, dimension), dtype=float)
    for n in range(dimension - 1):
        coefficient = math.sqrt(q_number(n + 1.0, q) * q_number(2.0 * kappa + n, q))
        kp[n + 1, n] = coefficient
        km[n, n + 1] = coefficient
    return k0, kp, km


def q_lowering_matrix(
    total: int,
    kappas: Sequence[float],
    q: float,
    subset: Iterable[int] | None = None,
) -> np.ndarray:
    factors = len(kappas)
    active = list(range(factors)) if subset is None else [int(value) for value in subset]
    basis_now = compositions(total, factors)
    if total == 0:
        return np.zeros((0, len(basis_now)), dtype=float)
    basis_previous = compositions(total - 1, factors)
    previous_index = {state: index for index, state in enumerate(basis_previous)}
    matrix = np.zeros((len(basis_previous), len(basis_now)), dtype=float)
    for column, state in enumerate(basis_now):
        for position, factor in enumerate(active):
            occupation = state[factor]
            if occupation == 0:
                continue
            coefficient = math.sqrt(
                q_number(float(occupation), q)
                * q_number(2.0 * kappas[factor] + occupation - 1.0, q)
            )
            exponent = 0.0
            for left in active[:position]:
                exponent -= state[left] + kappas[left]
            for right in active[position + 1 :]:
                exponent += state[right] + kappas[right]
            coefficient *= q**exponent
            target = list(state)
            target[factor] -= 1
            matrix[previous_index[tuple(target)], column] += coefficient
    return matrix


def q_casimir_matrix(total: int, kappas: Sequence[float], q: float, subset: Iterable[int]) -> np.ndarray:
    active = [int(value) for value in subset]
    basis = compositions(total, len(kappas))
    k0_values = np.array([sum(state[index] + kappas[index] for index in active) for state in basis])
    lowering = q_lowering_matrix(total, kappas, q, active)
    diagonal = np.array([q_number(value, q) * q_number(value - 1.0, q) for value in k0_values])
    return np.diag(diagonal) - lowering.T @ lowering
```

```


def q_multiplicity_frame(total: int, kappas: Sequence[float], q: float, atol: float = 1e-11) -> np.ndarray:
    if total == 0:
        return np.ones((1, 1), dtype=float)
    lowering = q_lowering_matrix(total, kappas, q)
    return null_space(lowering, rcond=atol)


def sign_fix_columns(matrix: np.ndarray) -> np.ndarray:
    result = np.array(matrix, dtype=float, copy=True)
    for column in range(result.shape[1]):
        row = int(np.argmax(np.abs(result[:, column])))
        if result[row, column] < 0:
            result[:, column] *= -1.0
    return result


def canonicalize_overlap(matrix: np.ndarray) -> np.ndarray:
    result = np.array(matrix, dtype=float, copy=True)
    for column in range(result.shape[1]):
        if result[0, column] < 0:
            result[:, column] *= -1.0
    for row in range(result.shape[0]):
        if result[row, 0] < 0:
            result[row, :] *= -1.0
    return result


def align_signed_matrix(reference: np.ndarray, candidate: np.ndarray, iterations: int = 8) -> np.ndarray:
    result = np.array(candidate, dtype=float, copy=True)
    for _ in range(iterations):
        for column in range(result.shape[1]):
            if float(np.dot(reference[:, column], result[:, column])) < 0:
                result[:, column] *= -1.0
        for row in range(result.shape[0]):
            if float(np.dot(reference[row, :], result[row, :])) < 0:
                result[row, :] *= -1.0
    return result


@dataclass
class QRecouplingData:
    total: int
    kappas: tuple[float, float, float]
    q: float
    multiplicity_frame: np.ndarray
    left_eigenvalues: np.ndarray
    right_eigenvalues: np.ndarray
    left_basis: np.ndarray
    right_basis: np.ndarray
    overlap: np.ndarray
    right_in_left: np.ndarray


def q_three_body_recoupling(total: int, kappas: Sequence[float], q: float) -> QRecouplingData:
    if len(kappas) != 3:
        raise ValueError("q_three_body_recoupling requires three shapes")
    kappas_tuple = tuple(float(value) for value in kappas)
    frame = q_multiplicity_frame(total, kappas_tuple, q)
    c12 = frame.T @ q_casimir_matrix(total, kappas_tuple, q, (0, 1)) @ frame
    c23 = frame.T @ q_casimir_matrix(total, kappas_tuple, q, (1, 2)) @ frame
    left_values, left_vectors = np.linalg.eigh(c12)
    right_values, right_vectors = np.linalg.eigh(c23)
    left_vectors = sign_fix_columns(left_vectors)
    right_vectors = sign_fix_columns(right_vectors)
    left_basis = frame @ left_vectors
    right_basis = frame @ right_vectors
    overlap = canonicalize_overlap(left_basis.T @ right_basis)
    right_in_left = overlap @ np.diag(right_values) @ overlap.T
    return QRecouplingData(
```

```
        total=total,
        kappas=kappas_tuple,
        q=float(q),
        multiplicity_frame=frame,
        left_eigenvalues=left_values,
        right_eigenvalues=right_values,
        left_basis=left_basis,
        right_basis=right_basis,
        overlap=overlap,
        right_in_left=right_in_left,
    )


def q_pair_spectrum(total: int, kappa_a: float, kappa_b: float, q: float) -> np.ndarray:
    return np.array(
        [
            q_number(kappa_a + kappa_b + j, q)
            * q_number(kappa_a + kappa_b + j - 1.0, q)
            for j in range(total + 1)
        ]
    )


def q_two_body_lowest_coefficients(total: int, kappa_1: float, kappa_2: float, q: float) -> np.ndarray:
    tilt = q ** (-(total - 1.0 + kappa_1 + kappa_2))
    coefficients = []
    for n in range(total + 1):
        weight = q_binomial(total, n, q)
        weight *= q_rising(2.0 * kappa_2 + total - n, n, q)
        weight /= q_rising(2.0 * kappa_1, n, q)
        coefficients.append(((-tilt) ** n) * math.sqrt(weight))
    vector = np.array(coefficients, dtype=float)
    return vector / np.linalg.norm(vector)


def equal_shape_one_excitation_matrix(q: float) -> np.ndarray:
    denominator = 1.0 + q * q
    cosine = q / denominator
    sine = math.sqrt(1.0 + q * q + q**4) / denominator
    return np.array([[cosine, sine], [sine, -cosine]], dtype=float)


def tridiagonal_residual(matrix: np.ndarray) -> float:
    residual = np.array(matrix, copy=True)
    for row in range(residual.shape[0]):
        for column in range(residual.shape[1]):
            if abs(row - column) <= 1:
                residual[row, column] = 0.0
    return float(np.max(np.abs(residual)))


def row_entropies(probability_matrix: np.ndarray) -> np.ndarray:
    safe = np.where(probability_matrix > 0, probability_matrix, 1.0)
    return -np.sum(np.where(probability_matrix > 0, probability_matrix * np.log(safe), 0.0), axis=1)


def doubly_stochastic_residual(matrix: np.ndarray) -> float:
    return float(
        max(
            np.max(np.abs(matrix.sum(axis=0) - 1.0)),
            np.max(np.abs(matrix.sum(axis=1) - 1.0)),
            np.max(np.maximum(-matrix, 0.0)),
        )
    )


def spectral_mixing_gap(probability_matrix: np.ndarray) -> float:
    eigenvalues = np.linalg.eigvals(probability_matrix)
    magnitudes = sorted((abs(value) for value in eigenvalues), reverse=True)
    if len(magnitudes) < 2:
        return 1.0
```

```
    return float(1.0 - magnitudes[1])


# ---------------------------------------------------------------------------
# q-deformation flow
# ---------------------------------------------------------------------------


def deformation_generator(total: int, kappas: Sequence[float], theta: float, step: float = 2e-5) -> dict[str, np.
     ndarray | float]:
    q0 = math.exp(-theta)
    reference = q_three_body_recoupling(total, kappas, q0).overlap
    plus = q_three_body_recoupling(total, kappas, math.exp(-(theta + step))).overlap
    minus = q_three_body_recoupling(total, kappas, math.exp(-(theta - step))).overlap
    plus = align_signed_matrix(reference, plus)
    minus = align_signed_matrix(reference, minus)
    derivative = (plus - minus) / (2.0 * step)
    raw = reference.T @ derivative
    generator = 0.5 * (raw - raw.T)
    skew_residual = float(np.max(np.abs(raw + raw.T)))
    tangent_residual = float(np.max(np.abs(derivative - reference @ generator)))
    metric = 0.5 * float(np.sum(generator * generator))
    return {
        "q": q0,
        "matrix": reference,
        "derivative": derivative,
        "generator": generator,
        "skew_residual": skew_residual,
        "tangent_residual": tangent_residual,
        "metric": metric,
    }


# ---------------------------------------------------------------------------
# Four-body q-coupling trees and pentagon coherence
# ---------------------------------------------------------------------------


FOUR_TREES: tuple[tuple[str, tuple[tuple[int, ...], tuple[int, ...]]], ...] = (
    ("(((12)3)4)", ((0, 1), (0, 1, 2))),
    ("((1(23))4)", ((1, 2), (0, 1, 2))),
    ("(1((23)4))", ((1, 2), (1, 2, 3))),
    ("(1(2(34)))", ((2, 3), (1, 2, 3))),
    ("((12)(34))", ((0, 1), (2, 3))),
)


@dataclass
class QTreeBasisData:
    name: str
    subsets: tuple[tuple[int, ...], ...]
    basis: np.ndarray
    joint_diagonals: tuple[np.ndarray, ...]
    commutator_residual: float
    diagonalization_residual: float


def q_joint_tree_basis(
    total: int,
    kappas: Sequence[float],
    q: float,
    name: str,
    subsets: Sequence[Sequence[int]],
) -> QTreeBasisData:
    frame = q_multiplicity_frame(total, kappas, q)
    restricted = [frame.T @ q_casimir_matrix(total, kappas, q, subset) @ frame for subset in subsets]
    commutator = 0.0
    for i in range(len(restricted)):
        for j in range(i + 1, len(restricted)):
            commutator = max(
                commutator,
```

```
                float(np.max(np.abs(restricted[i] @ restricted[j] - restricted[j] @ restricted[i]))),
            )
    irrational_weights = [math.sqrt(2.0), math.sqrt(3.0), math.sqrt(5.0)]
    combined = sum(weight * matrix for weight, matrix in zip(irrational_weights, restricted))
    _, vectors = np.linalg.eigh(combined)
    vectors = sign_fix_columns(vectors)
    diagonals = tuple(vectors.T @ matrix @ vectors for matrix in restricted)
    tuples = []
    for column in range(vectors.shape[1]):
        key = tuple(round(float(diagonal[column, column]), 10) for diagonal in diagonals)
        tuples.append((key, column))
    order = [column for _, column in sorted(tuples, key=lambda item: item[0])]
    vectors = vectors[:, order]
    diagonals = tuple(vectors.T @ matrix @ vectors for matrix in restricted)
    diagonalization_residual = max(
        float(np.max(np.abs(diagonal - np.diag(np.diag(diagonal))))) for diagonal in diagonals
    )
    return QTreeBasisData(
        name=name,
        subsets=tuple(tuple(int(index) for index in subset) for subset in subsets),
        basis=frame @ vectors,
        joint_diagonals=diagonals,
        commutator_residual=commutator,
        diagonalization_residual=diagonalization_residual,
    )


def q_four_body_tree_bases(total: int, kappas: Sequence[float], q: float) -> list[QTreeBasisData]:
    if len(kappas) != 4:
        raise ValueError("q_four_body_tree_bases requires four shapes")
    return [q_joint_tree_basis(total, kappas, q, name, subsets) for name, subsets in FOUR_TREES]


def q_pentagon_product(tree_bases: Sequence[QTreeBasisData]) -> tuple[np.ndarray, list[np.ndarray]]:
    dimension = tree_bases[0].basis.shape[1]
    product_matrix = np.eye(dimension)
    overlaps: list[np.ndarray] = []
    for index in range(len(tree_bases)):
        left = tree_bases[index].basis
        right = tree_bases[(index + 1) % len(tree_bases)].basis
        overlap = left.T @ right
        overlaps.append(overlap)
        product_matrix = product_matrix @ overlap
    return product_matrix, overlaps


# ---------------------------------------------------------------------------
# q-hypergeometric q-Racah and Askey-Wilson functions
# ---------------------------------------------------------------------------


def q_pochhammer(a: complex, q: float, n: int) -> complex:
    result = 1.0 + 0.0j
    for j in range(n):
        result *= 1.0 - a * (q**j)
    return result


def q_racah_polynomial(
    n: int,
    y: int,
    alpha: float,
    beta: float,
    gamma: float,
    delta: float,
    q: float,
) -> complex:
    result = 0.0 + 0.0j
    for ell in range(n + 1):
        numerator = q_pochhammer(q ** (-n), q, ell)
        numerator *= q_pochhammer(alpha * beta * q ** (n + 1), q, ell)
```

```
        numerator *= q_pochhammer(q ** (-y), q, ell)
        numerator *= q_pochhammer(gamma * delta * q ** (y + 1), q, ell)
        denominator = q_pochhammer(alpha * q, q, ell)
        denominator *= q_pochhammer(beta * delta * q, q, ell)
        denominator *= q_pochhammer(gamma * q, q, ell)
        denominator *= q_pochhammer(q, q, ell)
        result += numerator / denominator * q**ell
    return result


def rising(a: float, n: int) -> float:
    result = 1.0
    for j in range(n):
        result *= a + j
    return result


def classical_racah_polynomial(n: int, y: int, alpha: float, beta: float, gamma: float, delta: float) -> float:
    result = 0.0
    for ell in range(n + 1):
        numerator = rising(-float(n), ell)
        numerator *= rising(n + alpha + beta + 1.0, ell)
        numerator *= rising(-float(y), ell)
        numerator *= rising(y + gamma + delta + 1.0, ell)
        denominator = rising(alpha + 1.0, ell)
        denominator *= rising(beta + delta + 1.0, ell)
        denominator *= rising(gamma + 1.0, ell)
        denominator *= math.factorial(ell)
        result += numerator / denominator
    return result


def askey_wilson_r(
    n: int,
    z: complex,
    a: float,
    b: float,
    c: float,
    d: float,
    q: float,
) -> complex:
    result = 0.0 + 0.0j
    for ell in range(n + 1):
        numerator = q_pochhammer(q ** (-n), q, ell)
        numerator *= q_pochhammer(a * b * c * d * q ** (n - 1), q, ell)
        numerator *= q_pochhammer(a * z, q, ell)
        numerator *= q_pochhammer(a / z, q, ell)
        denominator = q_pochhammer(a * b, q, ell)
        denominator *= q_pochhammer(a * c, q, ell)
        denominator *= q_pochhammer(a * d, q, ell)
        denominator *= q_pochhammer(q, q, ell)
        result += numerator / denominator * q**ell
    return result


def askey_wilson_operator(
    function,
    z: complex,
    a: float,
    b: float,
    c: float,
    d: float,
    q: float,
) -> complex:
    first = ((1 - a * z) * (1 - b * z) * (1 - c * z) * (1 - d * z))
    first /= (1 - z * z) * (1 - q * z * z)
    inverse = 1.0 / z
    second = ((1 - a * inverse) * (1 - b * inverse) * (1 - c * inverse) * (1 - d * inverse))
    second /= (1 - inverse * inverse) * (1 - q * inverse * inverse)
    return (
        first * (function(q * z) - function(z))
```

```
        + second * (function(z / q) - function(z))
        + (1.0 + a * b * c * d / q) * function(z)
    )


# ---------------------------------------------------------------------------
# Givens compilation
# ---------------------------------------------------------------------------


@dataclass
class GivensRotation:
    upper: int
    lower: int
    cosine: float
    sine: float


def givens_decompose(orthogonal: np.ndarray, atol: float = 1e-14) -> tuple[list[GivensRotation], np.ndarray]:
    matrix = np.array(orthogonal, dtype=float, copy=True)
    dimension = matrix.shape[0]
    rotations: list[GivensRotation] = []
    for column in range(dimension):
        for lower in range(dimension - 1, column, -1):
            upper = lower - 1
            a = matrix[upper, column]
            b = matrix[lower, column]
            radius = math.hypot(a, b)
            if radius < atol:
                continue
            cosine = a / radius
            sine = b / radius
            row_upper = cosine * matrix[upper, :] + sine * matrix[lower, :]
            row_lower = -sine * matrix[upper, :] + cosine * matrix[lower, :]
            matrix[upper, :] = row_upper
            matrix[lower, :] = row_lower
            rotations.append(GivensRotation(upper, lower, cosine, sine))
    return rotations, np.diag(np.diag(matrix))


def givens_reconstruct(rotations: Sequence[GivensRotation], diagonal: np.ndarray) -> np.ndarray:
    matrix = np.array(diagonal, dtype=float, copy=True)
    for rotation in reversed(rotations):
        upper = rotation.upper
        lower = rotation.lower
        cosine = rotation.cosine
        sine = rotation.sine
        row_upper = cosine * matrix[upper, :] - sine * matrix[lower, :]
        row_lower = sine * matrix[upper, :] + cosine * matrix[lower, :]
        matrix[upper, :] = row_upper
        matrix[lower, :] = row_lower
    return matrix


# ---------------------------------------------------------------------------
# Checks and result tables
# ---------------------------------------------------------------------------


def run_checks() -> dict[str, dict[str, float | int | str | bool]]:
    checks: dict[str, dict[str, float | int | str | bool]] = {}

    q = 0.73
    kappa = 0.8
    dimension = 12
    k0, kp, km = q_local_matrices(dimension, kappa, q)
    commutator = km @ kp - kp @ km
    target = np.diag([q_number(2.0 * (n + kappa), q) for n in range(dimension)])
    algebra_residual = float(np.max(np.abs(commutator[:-1, :-1] - target[:-1, :-1])))
    checks["q_algebra"] = {"passed": algebra_residual < 2e-10, "residual": algebra_residual}

```

```
    total = 5
    kappas_two = (0.7, 1.1)
    coefficients = q_two_body_lowest_coefficients(total, *kappas_two, q)
    lowest_residual = float(np.max(np.abs(q_lowering_matrix(total, kappas_two, q) @ coefficients)))
    checks["q_hahn_lowest_weight"] = {"passed": lowest_residual < 2e-10, "residual": lowest_residual}

    casimir = q_casimir_matrix(6, (kappa,), q, (0,))
    scalar = q_number(kappa, q) * q_number(kappa - 1.0, q)
    casimir_residual = float(np.max(np.abs(casimir - scalar * np.eye(casimir.shape[0]))))
    checks["q_casimir_scalar"] = {"passed": casimir_residual < 2e-10, "residual": casimir_residual}

    data = q_three_body_recoupling(8, (0.7, 1.1, 0.9), q)
    orthogonality = float(np.max(np.abs(data.overlap @ data.overlap.T - np.eye(9))))
    checks["q_racah_orthogonality"] = {"passed": orthogonality < 2e-10, "residual": orthogonality}

    left_theory = q_pair_spectrum(8, 0.7, 1.1, q)
    spectrum_residual = float(np.max(np.abs(data.left_eigenvalues - left_theory)))
    checks["q_pair_spectrum"] = {"passed": spectrum_residual < 2e-9, "residual": spectrum_residual}

    tri_residual = tridiagonal_residual(data.right_in_left)
    checks["q_racah_tridiagonal"] = {"passed": tri_residual < 3e-9, "residual": tri_residual}

    exact = equal_shape_one_excitation_matrix(q)
    computed = q_three_body_recoupling(1, (0.5, 0.5, 0.5), q).overlap
    beam_residual = float(np.max(np.abs(computed - exact)))
    checks["q_beam_splitter_closed_form"] = {"passed": beam_residual < 2e-10, "residual": beam_residual}

    probability = data.overlap**2
    stochastic_residual = doubly_stochastic_residual(probability)
    checks["q_bistochastic_kernel"] = {"passed": stochastic_residual < 2e-10, "residual": stochastic_residual}

    rng = np.random.default_rng(801)
    entropy_fail = 0.0
    for _ in range(80):
        vector = rng.random(probability.shape[0])
        vector /= vector.sum()
        before = -float(np.sum(vector * np.log(vector)))
        after_vector = vector @ probability
        after = -float(np.sum(after_vector * np.log(after_vector)))
        entropy_fail = max(entropy_fail, before - after)
    checks["q_entropy_non_decrease"] = {"passed": entropy_fail < 2e-12, "residual": entropy_fail}

    q_values = np.linspace(0.12, 1.0, 80)
    entropy_one = []
    for value in q_values:
        matrix = equal_shape_one_excitation_matrix(float(value))
        entropy_one.append(row_entropies(matrix**2)[0])
    monotone_residual = float(max(0.0, -np.min(np.diff(entropy_one))))
    checks["one_excitation_entropy_monotone"] = {"passed": monotone_residual < 2e-10, "residual": monotone_residual}

    entropy_middle = []
    for value in q_values:
        matrix = q_three_body_recoupling(3, (0.5, 0.5, 0.5), float(value)).overlap
        entropy_middle.append(row_entropies(matrix**2)[1])
    negative_control = float(np.min(np.diff(entropy_middle)))
    checks["higher_entropy_nonmonotone_control"] = {
        "passed": negative_control < -1e-5,
        "minimum_increment": negative_control,
    }

    classical = q_three_body_recoupling(8, (0.7, 1.1, 0.9), 1.0).overlap
    near = align_signed_matrix(classical, q_three_body_recoupling(8, (0.7, 1.1, 0.9), 0.9999).overlap)
    classical_limit = float(np.max(np.abs(near - classical)))
    checks["q_to_one_recoupling"] = {"passed": classical_limit < 2e-5, "residual": classical_limit}

    qh_classical = q_two_body_lowest_coefficients(7, 0.7, 1.1, 1.0)
    qh_near = q_two_body_lowest_coefficients(7, 0.7, 1.1, 0.999999)
    if float(np.dot(qh_classical, qh_near)) < 0:
        qh_near *= -1.0
    qh_limit = float(np.max(np.abs(qh_near - qh_classical)))
    checks["q_hahn_classical_limit"] = {"passed": qh_limit < 2e-5, "residual": qh_limit}
```

```
    small_q = equal_shape_one_excitation_matrix(1e-5)
    routing_target = np.array([[0.0, 1.0], [1.0, 0.0]])
    routing_residual = float(np.max(np.abs(small_q - routing_target)))
    checks["q_to_zero_routing"] = {"passed": routing_residual < 2e-5, "residual": routing_residual}

    deformation = deformation_generator(6, (0.7, 1.1, 0.9), theta=0.35)
    checks["deformation_generator_skew"] = {
        "passed": float(deformation["skew_residual"]) < 2e-7,
        "residual": float(deformation["skew_residual"]),
    }
    checks["deformation_tangent_equation"] = {
        "passed": float(deformation["tangent_residual"]) < 2e-7,
        "residual": float(deformation["tangent_residual"]),
    }

    four_kappas = (0.55, 0.8, 1.05, 1.3)
    trees = q_four_body_tree_bases(4, four_kappas, 0.78)
    commutator_residual = max(tree.commutator_residual for tree in trees)
    diagonal_residual = max(tree.diagonalization_residual for tree in trees)
    product_matrix, _ = q_pentagon_product(trees)
    pentagon_residual = float(np.max(np.abs(product_matrix - np.eye(product_matrix.shape[0]))))
    checks["q_tree_commutation"] = {"passed": commutator_residual < 2e-8, "residual": commutator_residual}
    checks["q_tree_diagonalization"] = {"passed": diagonal_residual < 2e-8, "residual": diagonal_residual}
    checks["q_pentagon"] = {"passed": pentagon_residual < 2e-8, "residual": pentagon_residual}

    rotations, diagonal = givens_decompose(data.overlap)
    reconstructed = givens_reconstruct(rotations, diagonal)
    givens_residual = float(np.max(np.abs(reconstructed - data.overlap)))
    checks["q_givens_compilation"] = {
        "passed": givens_residual < 2e-10,
        "residual": givens_residual,
        "rotation_count": len(rotations),
    }

    q_poly = 0.73
    degree_limit = 6
    alpha_exp = 0.6
    beta_exp = 0.8
    gamma_exp = -degree_limit - 1.0
    delta_exp = 0.4
    alpha = q_poly**alpha_exp
    beta = q_poly**beta_exp
    gamma = q_poly**gamma_exp
    delta = q_poly**delta_exp
    duality_residual = 0.0
    for n in range(degree_limit + 1):
        for y in range(degree_limit + 1):
            left = q_racah_polynomial(n, y, alpha, beta, gamma, delta, q_poly)
            right = q_racah_polynomial(y, n, gamma, delta, alpha, beta, q_poly)
            duality_residual = max(duality_residual, abs(left - right))
    checks["q_racah_duality"] = {"passed": duality_residual < 2e-10, "residual": float(duality_residual)}

    q_limit = 0.999
    alpha = q_limit**alpha_exp
    beta = q_limit**beta_exp
    gamma = q_limit**gamma_exp
    delta = q_limit**delta_exp
    polynomial_limit = 0.0
    for n in range(4):
        for y in range(degree_limit + 1):
            q_value = q_racah_polynomial(n, y, alpha, beta, gamma, delta, q_limit).real
            classical_value = classical_racah_polynomial(n, y, alpha_exp, beta_exp, gamma_exp, delta_exp)
            polynomial_limit = max(polynomial_limit, abs(q_value - classical_value))
    checks["q_racah_polynomial_limit"] = {"passed": polynomial_limit < 2e-4, "residual": polynomial_limit}

    aw_q = 0.7
    aw_parameters = (0.2, 0.3, -0.25, 0.4)
    aw_residual = 0.0
    for n in range(6):
        for theta in np.linspace(0.31, 2.72, 8):
```

```
            z = np.exp(1j * theta)
            function = lambda value, degree=n: askey_wilson_r(degree, value, *aw_parameters, aw_q)
            left = askey_wilson_operator(function, z, *aw_parameters, aw_q)
            eigenvalue = aw_q ** (-n) + np.prod(aw_parameters) * aw_q ** (n - 1)
            aw_residual = max(aw_residual, abs(left - eigenvalue * function(z)))
    checks["askey_wilson_difference_equation"] = {"passed": aw_residual < 2e-9, "residual": float(aw_residual)}

    passed = sum(1 for item in checks.values() if bool(item["passed"]))
    summary = {"checks": checks, "passed": passed, "total": len(checks), "all_passed": passed == len(checks)}
    (RESULTS / "checks.json").write_text(json.dumps(jsonable(summary), indent=2))
    return checks


# ---------------------------------------------------------------------------
# Figures
# ---------------------------------------------------------------------------


def generate_figures() -> list[str]:
    generated: list[str] = []

    fig, ax = diagram_axes((13.0, 6.3))
    ax.text(0.5, 0.94, "From classical shape packets to q-deformed recoupling", ha="center", fontsize=21, weight="
     bold")
    stages = [
        (0.03, 0.58, 0.16, 0.17, "q-number\nladder", BLUE),
        (0.23, 0.58, 0.16, 0.17, "q-Hahn\nlowest packet", TEAL),
        (0.43, 0.58, 0.16, 0.17, "q-Casimir\natlases", ORANGE),
        (0.63, 0.58, 0.16, 0.17, "q-Racah\ntransform", MAGENTA),
        (0.83, 0.58, 0.14, 0.17, "q-pentagon\ncoherence", RED),
    ]
    for x, y, w, h, label, color in stages:
        box(ax, (x, y), w, h, label, face=color + "1C", edge=color, fontsize=11.5)
    for left, right in zip(stages[:-1], stages[1:]):
        arrow(ax, (left[0] + left[2], left[1] + left[3] / 2), (right[0], right[1] + right[3] / 2))
    box(ax, (0.12, 0.18), 0.20, 0.16, "bistochastic\nmeasurement", face=YELLOW + "25", edge=YELLOW, fontsize=11.5)
    box(ax, (0.40, 0.18), 0.20, 0.16, "deformation\ngenerator", face=PURPLE + "1C", edge=PURPLE, fontsize=11.5)
    box(ax, (0.68, 0.18), 0.20, 0.16, "Askey-Wilson\ncontinuous lift", face=BLUE + "1C", edge=BLUE, fontsize=11.5)
    arrow(ax, (0.70, 0.58), (0.22, 0.34), color=MAGENTA)
    arrow(ax, (0.72, 0.58), (0.50, 0.34), color=PURPLE)
    arrow(ax, (0.88, 0.58), (0.78, 0.34), color=RED)
    ax.text(0.5, 0.07, "The deformation parameter changes the packet algebra while preserving exact analysis,
     synthesis, and associator closure.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "01_theory_pipeline")
    generated.append("01_theory_pipeline")

    x = np.linspace(0.0, 8.0, 300)
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    for q, color in zip((0.45, 0.65, 0.82, 0.95, 1.0), COLORS[:5]):
        ax.plot(x, [q_number(value, q) for value in x], color=color, linewidth=2.0, label=f"q={q:g}")
    ax.plot(x, x, color=INK, linestyle="--", linewidth=1.2, label="classical x")
    ax.set_xlabel("x")
    ax.set_ylabel(r"$[x]_q$")
    ax.set_title("Symmetric q-numbers deform the linear ladder")
    ax.grid(alpha=0.18)
    ax.legend(ncol=3)
    save_figure(fig, "02_q_number_deformation")
    generated.append("02_q_number_deformation")

    n = np.arange(0, 14)
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    for q, color in zip((0.45, 0.7, 0.9, 1.0), (BLUE, TEAL, ORANGE, MAGENTA)):
        values = [math.sqrt(q_number(value + 1.0, q) * q_number(value + 1.6, q)) for value in n]
        ax.plot(n, values, marker="o", markersize=3.5, color=color, label=f"q={q:g}")
    ax.set_xlabel("packet degree n")
    ax.set_ylabel("raising coefficient")
    ax.set_title(r"q-deformed positive-series ladder at $\kappa=0.8$")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "03_q_ladder_coefficients")
    generated.append("03_q_ladder_coefficients")
```

```
    fig, ax = diagram_axes((12.8, 6.2))
    ax.text(0.5, 0.94, "The symmetric coproduct routes one lowering event through every factor", ha="center", fontsize=20, weight="bold")
    centers = [(0.16, 0.60), (0.50, 0.60), (0.84, 0.60)]
    for index, center in enumerate(centers, start=1):
        box(ax, (center[0] - 0.09, center[1] - 0.08), 0.18, 0.16, f"packet {index}\n$D^+_{{\\kappa_{index},q}}$", face=COLORS[index - 1] + "1C", edge=COLORS[index - 1])
    ax.text(0.5, 0.35, r"$\Delta^{(3)}K_- = K_-q^{K_0}q^{K_0} + q^{-K_0}K_-q^{K_0} + q^{-K_0}q^{-K_0}K_-$", ha="center", fontsize=14)
    for center, color in zip(centers, COLORS[:3]):
        arrow(ax, (center[0], center[1] - 0.08), (0.50, 0.27), color=color)
    box(ax, (0.38, 0.11), 0.24, 0.12, "one exact\nlowest-weight projector", face=PURPLE + "1C", edge=PURPLE)
    ax.text(0.5, 0.04, "The q-powers retain the ordering data needed by the deformed tensor product.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "04_q_coproduct_paths")
    generated.append("04_q_coproduct_paths")

    total = 9
    fig, ax = plt.subplots(figsize=(11.8, 5.5))
    for q, color in zip((0.35, 0.55, 0.75, 0.9, 1.0), COLORS[:5]):
        probabilities = q_two_body_lowest_coefficients(total, 0.7, 1.1, q) ** 2
        ax.plot(range(total + 1), probabilities, marker="o", color=color, label=f"q={q:g}")
    ax.set_xlabel("first-factor excitation n")
    ax.set_ylabel("packet probability")
    ax.set_title("q-Hahn lowest-weight packet redistributes excitation")
    ax.grid(alpha=0.18)
    ax.legend(ncol=3)
    save_figure(fig, "05_q_hahn_packet_distribution")
    generated.append("05_q_hahn_packet_distribution")

    q_values = 1.0 - 10.0 ** (-np.linspace(1.0, 5.0, 45))
    classical = q_two_body_lowest_coefficients(9, 0.7, 1.1, 1.0)
    errors = []
    for value in q_values:
        vector = q_two_body_lowest_coefficients(9, 0.7, 1.1, float(value))
        if np.dot(vector, classical) < 0:
            vector *= -1
        errors.append(np.max(np.abs(vector - classical)))
    fig, ax = plt.subplots(figsize=(11.8, 5.2))
    ax.loglog(1.0 - q_values, errors, marker="o", color=TEAL)
    ax.set_xlabel(r"$1-q$")
    ax.set_ylabel("maximum coefficient error")
    ax.set_title("q-Hahn packets converge to the classical lowest-weight packet")
    ax.grid(alpha=0.18, which="both")
    save_figure(fig, "06_q_hahn_classical_limit")
    generated.append("06_q_hahn_classical_limit")

    q_values = np.linspace(0.02, 1.0, 320)
    cosines = q_values / (1.0 + q_values**2)
    sines = np.sqrt(1.0 + q_values**2 + q_values**4) / (1.0 + q_values**2)
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    ax.plot(q_values, cosines, color=BLUE, linewidth=2.2, label=r"$q/(1+q^2)$")
    ax.plot(q_values, sines, color=ORANGE, linewidth=2.2, label=r"$\sqrt{1+q^2+q^4}/(1+q^2)$")
    ax.set_xlabel("q")
    ax.set_ylabel("recoupling amplitude")
    ax.set_title("Exact equal-shape one-excitation q-beam splitter")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "07_q_beam_splitter_amplitudes")
    generated.append("07_q_beam_splitter_amplitudes")

    fig, axes = plt.subplots(1, 3, figsize=(12.8, 4.2))
    for ax, q in zip(axes, (0.15, 0.55, 1.0)):
        matrix = equal_shape_one_excitation_matrix(q) ** 2
        image = ax.imshow(matrix, cmap="viridis", vmin=0, vmax=1)
        for i in range(2):
            for j in range(2):
                ax.text(j, i, f"{matrix[i,j]:.3f}", ha="center", va="center", color=WHITE if matrix[i,j] > 0.55 else INK)
        ax.set_xticks([0, 1])
```

```
        ax.set_yticks([0, 1])
        ax.set_title(f"q={q:g}")
    fig.suptitle("The q-beam splitter interpolates between routing and mixing", fontsize=16, weight="bold")
    fig.colorbar(image, ax=axes, fraction=0.025, pad=0.025, label="transition probability")
    fig.subplots_adjust(left=0.06, right=0.92, bottom=0.12, top=0.82, wspace=0.25)
    save_figure(fig, "08_q_beam_splitter_kernel")
    generated.append("08_q_beam_splitter_kernel")

    entropy = []
    gap = []
    for q in q_values:
        probability = equal_shape_one_excitation_matrix(float(q)) ** 2
        entropy.append(row_entropies(probability)[0])
        gap.append(spectral_mixing_gap(probability))
    fig, ax = plt.subplots(figsize=(11.8, 5.4))
    ax.plot(q_values, entropy, color=BLUE, linewidth=2.2, label="row entropy")
    ax.plot(q_values, gap, color=ORANGE, linewidth=2.2, label="absolute spectral gap")
    ax.set_xlabel("q")
    ax.set_ylabel("value")
    ax.set_title("One-excitation mixing strengthens toward the classical endpoint")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "09_beam_splitter_entropy_gap")
    generated.append("09_beam_splitter_entropy_gap")

    total = 10
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    for q, color in zip((0.45, 0.65, 0.82, 1.0), (BLUE, TEAL, ORANGE, MAGENTA)):
        spectrum = q_pair_spectrum(total, 0.7, 1.1, q)
        ax.plot(range(total + 1), spectrum, marker="o", markersize=3.5, color=color, label=f"q={q:g}")
    ax.set_xlabel("intermediate packet label")
    ax.set_ylabel("q-Casimir eigenvalue")
    ax.set_title("q-Casimir spectra remain simple and ordered")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "10_q_casimir_spectra")
    generated.append("10_q_casimir_spectra")

    fig, axes = plt.subplots(1, 3, figsize=(13.0, 4.6))
    vmax = 0.0
    matrices = []
    for q in (0.45, 0.72, 1.0):
        matrix = q_three_body_recoupling(10, (0.7, 1.1, 0.9), q).overlap
        matrices.append(matrix)
        vmax = max(vmax, np.max(np.abs(matrix)))
    for ax, matrix, q in zip(axes, matrices, (0.45, 0.72, 1.0)):
        image = ax.imshow(matrix, cmap="coolwarm", vmin=-vmax, vmax=vmax, aspect="auto")
        ax.set_title(f"q={q:g}")
        ax.set_xlabel(r"$C_{23}$ label")
        ax.set_ylabel(r"$C_{12}$ label")
    fig.suptitle("q-Racah packet transforms across the deformation", fontsize=16, weight="bold")
    fig.colorbar(image, ax=axes, fraction=0.022, pad=0.025, label="overlap amplitude")
    fig.subplots_adjust(left=0.06, right=0.92, bottom=0.12, top=0.82, wspace=0.25)
    save_figure(fig, "11_q_racah_heatmaps")
    generated.append("11_q_racah_heatmaps")

    fig, axes = plt.subplots(1, 3, figsize=(13.0, 4.6))
    for ax, matrix, q in zip(axes, matrices, (0.45, 0.72, 1.0)):
        image = ax.imshow(matrix**2, cmap="viridis", vmin=0, vmax=np.max(matrix**2), aspect="auto")
        ax.set_title(f"q={q:g}")
        ax.set_xlabel("measurement output")
        ax.set_ylabel("input packet")
    fig.suptitle("Squared q-Racah amplitudes remain bistochastic", fontsize=16, weight="bold")
    fig.colorbar(image, ax=axes, fraction=0.022, pad=0.025, label="probability")
    fig.subplots_adjust(left=0.06, right=0.92, bottom=0.12, top=0.82, wspace=0.25)
    save_figure(fig, "12_q_racah_probability_kernels")
    generated.append("12_q_racah_probability_kernels")

    totals = np.arange(1, 17)
    fig, ax = plt.subplots(figsize=(11.8, 5.2))
    for q, color in zip((0.45, 0.65, 0.82, 1.0), (BLUE, TEAL, ORANGE, MAGENTA)):
```

```
        residuals = []
        for total in totals:
            matrix = q_three_body_recoupling(int(total), (0.7, 1.1, 0.9), q).overlap
            residuals.append(np.max(np.abs(matrix @ matrix.T - np.eye(total + 1))))
        ax.semilogy(totals, residuals, marker="o", markersize=3.5, color=color, label=f"q={q:g}")
    ax.set_xlabel("total excitation N")
    ax.set_ylabel("maximum orthogonality residual")
    ax.set_title("q-Racah analysis and synthesis remain numerically exact")
    ax.grid(alpha=0.18, which="both")
    ax.legend()
    save_figure(fig, "13_q_orthogonality_residual")
    generated.append("13_q_orthogonality_residual")

    data = q_three_body_recoupling(12, (0.7, 1.1, 0.9), 0.7)
    matrix = data.right_in_left
    fig, ax = plt.subplots(figsize=(8.2, 7.0))
    vmax = np.percentile(np.abs(matrix), 99)
    image = ax.imshow(matrix, cmap="coolwarm", vmin=-vmax, vmax=vmax, aspect="auto")
    ax.set_xlabel(r"$C_{12}$ packet label")
    ax.set_ylabel(r"$C_{12}$ packet label")
    ax.set_title(r"The opposing q-Casimir is tridiagonal in the left atlas")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.03)
    save_figure(fig, "14_opposing_q_casimir")
    generated.append("14_opposing_q_casimir")

    q_poly = 0.73
    N = 8
    A, B, G, D = 0.6, 0.8, -N - 1.0, 0.4
    alpha, beta, gamma, delta = q_poly**A, q_poly**B, q_poly**G, q_poly**D
    values = np.array(
        [[q_racah_polynomial(n, y, alpha, beta, gamma, delta, q_poly).real for y in range(N + 1)] for n in range(N +
  1)]
    )
    fig, ax = plt.subplots(figsize=(8.4, 7.2))
    image = ax.imshow(values, cmap="coolwarm", aspect="auto")
    ax.set_xlabel("lattice coordinate y")
    ax.set_ylabel("polynomial degree n")
    ax.set_title("q-Racah basic-hypergeometric packet table")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.03, label="polynomial value")
    save_figure(fig, "15_q_racah_polynomial_table")
    generated.append("15_q_racah_polynomial_table")

    dual_values = np.array(
        [[q_racah_polynomial(y, n, gamma, delta, alpha, beta, q_poly).real for y in range(N + 1)] for n in range(N +
  1)]
    )
    duality = values - dual_values
    fig, ax = plt.subplots(figsize=(7.8, 6.6))
    image = ax.imshow(np.abs(duality), cmap="magma", aspect="auto")
    ax.set_xlabel("y")
    ax.set_ylabel("n")
    ax.set_title("q-Racah duality residual after parameter exchange")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.03, label="absolute residual")
    save_figure(fig, "16_q_racah_duality")
    generated.append("16_q_racah_duality")

    q_limits = 1.0 - 10.0 ** (-np.linspace(0.55, 4.0, 42))
    errors = []
    for q in q_limits:
        alpha, beta, gamma, delta = q**A, q**B, q**G, q**D
        error = 0.0
        for n in range(4):
            for y in range(N + 1):
                q_value = q_racah_polynomial(n, y, alpha, beta, gamma, delta, float(q)).real
                classical_value = classical_racah_polynomial(n, y, A, B, G, D)
                error = max(error, abs(q_value - classical_value))
        errors.append(error)
    fig, ax = plt.subplots(figsize=(11.8, 5.2))
    ax.loglog(1.0 - q_limits, errors, marker="o", color=TEAL)
    ax.set_xlabel(r"$1-q$")
    ax.set_ylabel("maximum polynomial error")
```

```
    ax.set_title("The explicit q-Racah family converges to Racah")
    ax.grid(alpha=0.18, which="both")
    save_figure(fig, "17_q_racah_to_racah")
    generated.append("17_q_racah_to_racah")

    classical = q_three_body_recoupling(10, (0.7, 1.1, 0.9), 1.0).overlap
    matrix_errors = []
    for q in q_limits:
        matrix = q_three_body_recoupling(10, (0.7, 1.1, 0.9), float(q)).overlap
        matrix = align_signed_matrix(classical, matrix)
        matrix_errors.append(np.max(np.abs(matrix - classical)))
    fig, ax = plt.subplots(figsize=(11.8, 5.2))
    ax.loglog(1.0 - q_limits, matrix_errors, marker="o", color=BLUE)
    ax.set_xlabel(r"$1-q$")
    ax.set_ylabel("maximum matrix error")
    ax.set_title("The packet recoupling matrix has the same classical limit")
    ax.grid(alpha=0.18, which="both")
    save_figure(fig, "18_recoupling_classical_limit")
    generated.append("18_recoupling_classical_limit")

    deformation = deformation_generator(8, (0.7, 1.1, 0.9), theta=0.38)
    generator = np.asarray(deformation["generator"])
    fig, ax = plt.subplots(figsize=(8.2, 7.0))
    vmax = np.max(np.abs(generator))
    image = ax.imshow(generator, cmap="coolwarm", vmin=-vmax, vmax=vmax, aspect="auto")
    ax.set_xlabel("packet coordinate")
    ax.set_ylabel("packet coordinate")
    ax.set_title(r"Skew deformation generator $A=R^{\mathrm{T}}\partial_\theta R$")
    fig.colorbar(image, ax=ax, fraction=0.04, pad=0.03)
    save_figure(fig, "19_deformation_generator")
    generated.append("19_deformation_generator")

    thetas = np.linspace(0.02, 1.25, 80)
    fig, ax = plt.subplots(figsize=(11.8, 5.3))
    for total, color in zip((1, 3, 6, 9), (BLUE, TEAL, ORANGE, MAGENTA)):
        metric = [float(deformation_generator(total, (0.7, 1.1, 0.9), float(theta))["metric"]) for theta in thetas]
        ax.plot(thetas, metric, color=color, linewidth=2.0, label=f"N={total}")
    ax.set_xlabel(r"deformation coordinate $\theta=-\log q$")
    ax.set_ylabel(r"$\frac{1}{2}\|A_N(\theta)\|_F^2$")
    ax.set_title("Packet deformation has a finite transport metric")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "20_deformation_metric")
    generated.append("20_deformation_metric")

    q_grid = np.linspace(0.12, 1.0, 100)
    fig, ax = plt.subplots(figsize=(11.8, 5.4))
    for total, row, color in ((1, 0, BLUE), (2, 1, TEAL), (3, 1, ORANGE), (5, 2, MAGENTA)):
        entropies = []
        for q in q_grid:
            matrix = q_three_body_recoupling(total, (0.5, 0.5, 0.5), float(q)).overlap
            entropies.append(row_entropies(matrix**2)[row])
        ax.plot(q_grid, entropies, color=color, linewidth=2.0, label=f"N={total}, row={row}")
    ax.set_xlabel("q")
    ax.set_ylabel("row entropy")
    ax.set_title("Higher q-Racah channels need not vary monotonically")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "21_entropy_nonmonotonicity")
    generated.append("21_entropy_nonmonotonicity")

    fig, ax = plt.subplots(figsize=(11.8, 5.4))
    for total, color in zip((1, 2, 4, 7), (BLUE, TEAL, ORANGE, MAGENTA)):
        gaps = []
        for q in q_grid:
            probability = q_three_body_recoupling(total, (0.5, 0.5, 0.5), float(q)).overlap**2
            gaps.append(spectral_mixing_gap(probability))
        ax.plot(q_grid, gaps, color=color, linewidth=2.0, label=f"N={total}")
    ax.set_xlabel("q")
    ax.set_ylabel("absolute spectral gap")
    ax.set_title("Measurement mixing depends on both deformation and excitation")
```

```
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "22_stochastic_spectral_gap")
    generated.append("22_stochastic_spectral_gap")

    fig, ax = diagram_axes((12.8, 6.6))
    ax.text(0.5, 0.95, "Five q-coupling trees form the same associahedral pentagon", ha="center", fontsize=20, weight="bold")
    angles = np.linspace(math.pi / 2, math.pi / 2 + 2 * math.pi, 6)[:-1]
    points = [(0.5 + 0.34 * math.cos(angle), 0.50 + 0.34 * math.sin(angle)) for angle in angles]
    labels = [name for name, _ in FOUR_TREES]
    for index, (point, label) in enumerate(zip(points, labels)):
        box(ax, (point[0] - 0.09, point[1] - 0.055), 0.18, 0.11, label, face=COLORS[index] + "1C", edge=COLORS[index], fontsize=9.5)
    for index in range(5):
        arrow(ax, points[index], points[(index + 1) % 5], color=INK)
    ax.text(0.5, 0.50, r"$\prod_{e\in\partial K_4} U_e(q)=I$", ha="center", va="center", fontsize=16, weight="bold")
    ax.text(0.5, 0.08, "Changing q changes every edge transform but not the closed associator path.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "23_q_pentagon_trees")
    generated.append("23_q_pentagon_trees")

    q_grid_pent = np.linspace(0.45, 1.0, 24)
    totals = (1, 2, 3, 4, 5)
    residual_grid = np.zeros((len(totals), len(q_grid_pent)))
    for i, total in enumerate(totals):
        for j, q in enumerate(q_grid_pent):
            trees = q_four_body_tree_bases(total, (0.55, 0.8, 1.05, 1.3), float(q))
            product_matrix, _ = q_pentagon_product(trees)
            residual_grid[i, j] = np.max(np.abs(product_matrix - np.eye(product_matrix.shape[0])))
    fig, ax = plt.subplots(figsize=(11.8, 5.7))
    image = ax.imshow(np.log10(np.maximum(residual_grid, 1e-18)), cmap="viridis", aspect="auto", origin="lower", extent=[q_grid_pent[0], q_grid_pent[-1], totals[0]-0.5, totals[-1]+0.5])
    ax.set_xlabel("q")
    ax.set_ylabel("total excitation N")
    ax.set_title("q-pentagon residual remains at floating-point scale")
    fig.colorbar(image, ax=ax, fraction=0.035, pad=0.025, label=r"$\log_{10}$ residual")
    save_figure(fig, "24_q_pentagon_residual")
    generated.append("24_q_pentagon_residual")

    trees = q_four_body_tree_bases(3, (0.55, 0.8, 1.05, 1.3), 0.68)
    _, overlaps = q_pentagon_product(trees)
    fig, axes = plt.subplots(1, 5, figsize=(14.5, 3.6))
    vmax = max(np.max(np.abs(matrix)) for matrix in overlaps)
    for index, (ax, matrix) in enumerate(zip(axes, overlaps), start=1):
        image = ax.imshow(matrix, cmap="coolwarm", vmin=-vmax, vmax=vmax, aspect="auto")
        ax.set_title(f"edge {index}")
        ax.set_xticks([])
        ax.set_yticks([])
    fig.suptitle("The five signed q-associator edges", fontsize=16, weight="bold")
    fig.colorbar(image, ax=axes, fraction=0.018, pad=0.02)
    fig.subplots_adjust(left=0.04, right=0.94, bottom=0.08, top=0.78, wspace=0.10)
    save_figure(fig, "25_q_pentagon_edge_matrices")
    generated.append("25_q_pentagon_edge_matrices")

    fig, axes = plt.subplots(1, 3, figsize=(13.0, 4.6))
    for ax, q in zip(axes, (0.5, 0.75, 1.0)):
        trees = q_four_body_tree_bases(3, (0.55, 0.8, 1.05, 1.3), q)
        matrix = trees[0].basis.T @ trees[1].basis
        image = ax.imshow(matrix, cmap="coolwarm", vmin=-np.max(np.abs(matrix)), vmax=np.max(np.abs(matrix)), aspect="auto")
        ax.set_title(f"q={q:g}")
        ax.set_xticks([])
        ax.set_yticks([])
    fig.suptitle("One associator edge deforms continuously", fontsize=16, weight="bold")
    fig.colorbar(image, ax=axes, fraction=0.022, pad=0.025)
    fig.subplots_adjust(left=0.05, right=0.92, bottom=0.08, top=0.80, wspace=0.20)
    save_figure(fig, "26_associator_deformation")
    generated.append("26_associator_deformation")

    totals = np.arange(1, 14)
```

```
    rotation_counts = []
    residuals = []
    for total in totals:
        matrix = q_three_body_recoupling(int(total), (0.7, 1.1, 0.9), 0.72).overlap
        rotations, diagonal = givens_decompose(matrix)
        reconstruction = givens_reconstruct(rotations, diagonal)
        rotation_counts.append(len(rotations))
        residuals.append(np.max(np.abs(reconstruction - matrix)))
    fig, ax = plt.subplots(figsize=(11.8, 5.4))
    ax.plot(totals, rotation_counts, marker="o", color=BLUE, label="used rotations")
    ax.plot(totals, totals * (totals + 1) / 2, linestyle="--", color=ORANGE, label=r"$d(d-1)/2$ bound")
    ax.set_xlabel("total excitation N")
    ax.set_ylabel("two-coordinate rotations")
    ax.set_title("Finite q-Racah transforms compile into Givens circuits")
    ax.grid(alpha=0.18)
    ax.legend()
    save_figure(fig, "27_q_givens_counts")
    generated.append("27_q_givens_counts")

    fig, ax = plt.subplots(figsize=(11.8, 5.1))
    ax.semilogy(totals, residuals, marker="o", color=TEAL)
    ax.set_xlabel("total excitation N")
    ax.set_ylabel("maximum reconstruction residual")
    ax.set_title("The compiled q-transform reconstructs the packet matrix")
    ax.grid(alpha=0.18, which="both")
    save_figure(fig, "28_q_givens_residual")
    generated.append("28_q_givens_residual")

    aw_q = 0.7
    aw_parameters = (0.2, 0.3, -0.25, 0.4)
    theta = np.linspace(0.08, math.pi - 0.08, 360)
    fig, ax = plt.subplots(figsize=(11.8, 5.5))
    for n, color in zip(range(5), COLORS[:5]):
        values = [askey_wilson_r(n, np.exp(1j * value), *aw_parameters, aw_q).real for value in theta]
        ax.plot(np.cos(theta), values, color=color, linewidth=1.8, label=f"n={n}")
    ax.set_xlabel(r"$x=\cos\theta$")
    ax.set_ylabel("normalized Askey-Wilson value")
    ax.set_title("The continuous Askey-Wilson packet lift")
    ax.grid(alpha=0.18)
    ax.legend(ncol=3)
    save_figure(fig, "29_askey_wilson_curves")
    generated.append("29_askey_wilson_curves")

    residual_table = np.zeros((6, 36))
    theta_grid = np.linspace(0.12, math.pi - 0.12, residual_table.shape[1])
    for n in range(residual_table.shape[0]):
        function = lambda value, degree=n: askey_wilson_r(degree, value, *aw_parameters, aw_q)
        eigenvalue = aw_q ** (-n) + np.prod(aw_parameters) * aw_q ** (n - 1)
        for j, value in enumerate(theta_grid):
            z = np.exp(1j * value)
            residual_table[n, j] = abs(askey_wilson_operator(function, z, *aw_parameters, aw_q) - eigenvalue *
     function(z))
    fig, ax = plt.subplots(figsize=(11.8, 5.4))
    image = ax.imshow(np.log10(np.maximum(residual_table, 1e-18)), cmap="magma", aspect="auto", origin="lower",
     extent=[theta_grid[0], theta_grid[-1], -0.5, 5.5])
    ax.set_xlabel(r"$\theta$")
    ax.set_ylabel("degree n")
    ax.set_title("Askey-Wilson q-difference equation residual")
    fig.colorbar(image, ax=ax, fraction=0.035, pad=0.025, label=r"$\log_{10}$ residual")
    save_figure(fig, "30_askey_wilson_difference_residual")
    generated.append("30_askey_wilson_difference_residual")

    fig, ax = diagram_axes((12.8, 6.4))
    ax.text(0.5, 0.94, "The finite and continuous q-functions belong to one transfer class", ha="center", fontsize
     =20, weight="bold")
    box(ax, (0.06, 0.58), 0.20, 0.16, "Uq(su(1,1))\nq-Hahn packets", face=TEAL + "1C", edge=TEAL)
    box(ax, (0.40, 0.58), 0.20, 0.16, "finite q-Racah\nrecoupling", face=MAGENTA + "1C", edge=MAGENTA)
    box(ax, (0.74, 0.58), 0.20, 0.16, "continuous\nAskey-Wilson lift", face=BLUE + "1C", edge=BLUE)
    arrow(ax, (0.26, 0.66), (0.40, 0.66), text="pair Casimir")
    arrow(ax, (0.60, 0.66), (0.74, 0.66), text="continuous spectrum")
    box(ax, (0.18, 0.20), 0.22, 0.15, "q -> 1\nclassical Racah", face=ORANGE + "1C", edge=ORANGE)
```

```
    box(ax, (0.60, 0.20), 0.22, 0.15, "q -> 0\nsparse routing", face=YELLOW + "25", edge=YELLOW)
    arrow(ax, (0.50, 0.58), (0.29, 0.35), color=ORANGE)
    arrow(ax, (0.50, 0.58), (0.71, 0.35), color=YELLOW)
    ax.text(0.5, 0.07, "The packet calculus keeps the exact reconstruction law while q changes the local transfer geometry.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "31_q_function_class_map")
    generated.append("31_q_function_class_map")

    fig, ax = diagram_axes((13.0, 6.4))
    ax.text(0.5, 0.94, "A programmable q-packet execution path", ha="center", fontsize=20, weight="bold")
    stages = [
        (0.04, 0.58, 0.16, 0.16, "packet shapes\n$\\kappa_i$", BLUE),
        (0.24, 0.58, 0.16, 0.16, "choose q and\nexcitation N", TEAL),
        (0.44, 0.58, 0.16, 0.16, "build q-Casimir\nprojectors", ORANGE),
        (0.64, 0.58, 0.16, 0.16, "diagonalize\nmultiplicity", MAGENTA),
        (0.84, 0.58, 0.13, 0.16, "compile\nrotations", RED),
    ]
    for x, y, w, h, label, color in stages:
        box(ax, (x, y), w, h, label, face=color + "1C", edge=color, fontsize=11)
    for left, right in zip(stages[:-1], stages[1:]):
        arrow(ax, (left[0] + left[2], left[1] + left[3] / 2), (right[0], right[1] + right[3] / 2))
    box(ax, (0.16, 0.20), 0.20, 0.15, "signed\nq-transform", face=PURPLE + "1C", edge=PURPLE)
    box(ax, (0.40, 0.20), 0.20, 0.15, "measurement\nkernel", face=YELLOW + "25", edge=YELLOW)
    box(ax, (0.64, 0.20), 0.20, 0.15, "pentagon\ncertificate", face=TEAL + "1C", edge=TEAL)
    arrow(ax, (0.90, 0.58), (0.26, 0.35), color=PURPLE)
    arrow(ax, (0.90, 0.58), (0.50, 0.35), color=YELLOW)
    arrow(ax, (0.90, 0.58), (0.74, 0.35), color=TEAL)
    ax.text(0.5, 0.07, "Dense tensor spaces are used only to define the projector. Execution takes place in the smaller multiplicity packet.", ha="center", fontsize=11.2, color=MUTED)
    save_figure(fig, "32_q_packet_compiler")
    generated.append("32_q_packet_compiler")

    return generated


def write_tables(checks: dict[str, dict[str, float | int | str | bool]]) -> None:
    with (TABLES / "check_summary.csv").open("w", newline="") as handle:
        writer = csv.writer(handle)
        writer.writerow(["check", "passed", "detail"])
        for name, result in checks.items():
            detail = ", ".join(f"{key}={value}" for key, value in result.items() if key != "passed")
            writer.writerow([name, result["passed"], detail])

    with (TABLES / "q_beam_splitter.csv").open("w", newline="") as handle:
        writer = csv.writer(handle)
        writer.writerow(["q", "cosine", "sine", "row_entropy", "spectral_gap"])
        for q in (0.1, 0.25, 0.5, 0.75, 1.0):
            matrix = equal_shape_one_excitation_matrix(q)
            probability = matrix**2
            writer.writerow([
                q,
                matrix[0, 0],
                matrix[0, 1],
                row_entropies(probability)[0],
                spectral_mixing_gap(probability),
            ])

    with (TABLES / "q_pentagon.csv").open("w", newline="") as handle:
        writer = csv.writer(handle)
        writer.writerow(["q", "N", "dimension", "pentagon_residual"])
        for q in (0.5, 0.7, 0.85, 1.0):
            for total in (1, 2, 3, 4):
                trees = q_four_body_tree_bases(total, (0.55, 0.8, 1.05, 1.3), q)
                product_matrix, _ = q_pentagon_product(trees)
                writer.writerow([q, total, product_matrix.shape[0], np.max(np.abs(product_matrix - np.eye(product_matrix.shape[0])))])


def write_manifest(generated: Sequence[str], checks: dict[str, dict[str, float | int | str | bool]]) -> None:
    manifest = {
        "title": "Projector-Generated Special Functions VIII",
```

```
        "author": "Komorebi P.",
        "figures": list(generated),
        "figure_pairs": len(generated),
        "checks_passed": sum(1 for value in checks.values() if bool(value["passed"])),
        "checks_total": len(checks),
    }
    (RESULTS / "manifest.json").write_text(json.dumps(jsonable(manifest), indent=2))


def main() -> None:
    checks = run_checks()
    generated = generate_figures()
    write_tables(checks)
    write_manifest(generated, checks)
    passed = sum(1 for result in checks.values() if bool(result["passed"]))
    print(f"Generated {len(generated)} figure pairs")
    print(f"Checks passed: {passed}/{len(checks)}")
    if passed != len(checks):
        raise SystemExit(1)


if __name__ == "__main__":
    main()
```

Chapter C

Bibliographic note

Bibliography

[1] NIST Digital Library of Mathematical Functions, Section 18.28, “Askey-Wilson Class.”

[2] H. T. Koelink and J. Van der Jeugt, “Convolutions for orthogonal polynomials from Lie and quantum algebra representations,” SIAM Journal on Mathematical Analysis 29 (1998), 794–822.

[3] E. Koelink and H. Rosengren, “Harmonic analysis on the $SU(2)$ dynamical quantum group,” Acta Applicandae Mathematicae 69 (2001), 163–220.

[4] Y. Shibukawa, “Clebsch–Gordan Coefficients for $\mathcal{U}_q(\mathfrak{su}(1,1))$ and $\mathcal{U}_q(\mathfrak{sl}(2))$, and Linearization Formula of Matrix Elements,” Publ. Res. Inst. Math. Sci. 28 (1992), 775–807.

[5] V. X. Genest, P. Iliev, and L. Vinet, “Coupling coefficients of $su_q(1,1)$ and multivariate q-Racah polynomials,” Nuclear Physics B 927 (2018), 97–123.

[6] G. Gasper and M. Rahman, *Basic Hypergeometric Series*, second edition, Cambridge University Press, 2004.

[7] S. Wei, Y. Zhai, Q. Lu, et al., “The Riemann Hypothesis manifested in dynamical quantum phase transitions,” Nature Communications, accepted 18 June 2026, DOI 10.1038/s41467-026-74935-8.

Chapter D

Reproduction

From the source directory, run

```
python projector_generated_special_functions_viii.py
python -m pytest -q
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_VIII.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_VIII.tex
pdflatex -interaction=nonstopmode Projector_Generated_Special_Functions_VIII.tex
```

The Python step writes all 32 figure pairs, the check ledger, the selected data tables, and the machine-readable manifest before the paper is compiled.