//JJ

Spectral Fingerprint Function Finding: Method and Preliminary Results

Author: Richard Andrew Holland

Acknowledgement The author wishes to acknowledge the significant contribution of Anthropic Claude AI (claude-sonnet-4-6), particularly in selection and implementation of the cosine weighted hash function.

Abstract: We present an efficient and consistent method for discerning preferable and generating functions of data using filtered spectral fingerprints. Spectral analysis of functions and data is hardly new. For example, with Fourier Transforms one can classify and identify functions using coefficients. The problem is that statistical, domain and systematic error noise is baked into the spectrum of coefficients. For over two centuries Legendre polynomials have been used for spectra, however, we are unaware of prior use of this approach. We create a hash library of function spectra to quickly find preferable and/or generating functions. Software tools (Patents Pending) have been developed utilizing this methodology, including one for finding AI/ML Neuron Activation functions. Even for noisy and instrument biased data, researchers and machine optimizers can better discern preferential and underlying functions that generate noisy data. Preliminary AI/ML results are also presented.

Introduction

1.1 Background Curve Fitting

Gram-Schmidt Orthogonal Polynomials are useful for generating data-specific functions that progressively fit linear terms, as the inner product of lower and higher order polynomials over the data-specific domain is zero. [Bevington 1969]. Given a fitting function of some f(x) of the form:

$$f(x) \approx \hat{f}(x) = \sum_{n=0}^{N} a_n \ \phi_n(x)$$

where $\{\phi_n(x)\}$ is a sequence of polynomials orthogonal under the discrete inner product defined by the data points $\{x_i\}$:

$$\langle \phi_m, \phi_n \rangle = \sum_i \phi_m(x_i) \ \phi_n(x_i) = |\phi_n|^2 \, \delta_{mn}$$

and the basis is built incrementally by the three-term recurrence [Szegő 1939, Golub & Welsch 1969]:

$$\phi_0(x) = 1$$

$$\phi_1(x) = x - \alpha_1$$
$$\phi_n(x) = (x - \alpha_n)\phi_{n-1}(x) - \beta_n\phi_{n-2}(x), \quad n \geq 2$$

with moment-matching coefficients

$$\alpha_n = \frac{\langle x\,\phi_{n-1},\,\phi_{n-1}\rangle}{\langle \phi_{n-1},\,\phi_{n-1}\rangle}$$

$$\beta_n = \frac{\langle \varphi_{n-1}, \varphi_{n-1}\rangle}{\langle \varphi_{n-2}, \varphi_{n-2}\rangle}$$

Each root for the orthogonal monomials can be determined based on the data and the property of orthogonal polynomials that the convolution of different monomial orders over the domain is zero. The N roots of $\phi_n(x) = 0$ are the eigenvalues of the symmetric tridiagonal Jacobi matrix [Golub & Welsch 1969]:

$$J = \begin{pmatrix} \alpha_1 & \sqrt{\beta_2} & 0 & \cdots & 0 \\ \sqrt{\beta_2} & \alpha_2 & \sqrt{\beta_3} & \ddots & \vdots \\ 0 & \sqrt{\beta_3} & \alpha_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \sqrt{\beta_n} \\ 0 & \cdots & 0 & \sqrt{\beta_n} & \alpha_n \end{pmatrix}$$

Because J is real symmetric, all N eigenvalues are real and distinct, and they are precisely the quadrature nodes of the Gauss rule adapted to the empirical data distribution. Roots are located via Sturm-sequence bisection [Press et al. 2007].

Using conventional least squares, each coefficient represents a best statistical curve fit for the number of terms provided to underlying data. Minimizing the squared residual

$$E(a) = \sum_i \left[y_i - \sum_{n=0}^{N} a_n\ \phi_n(x_i) \right]^2 = \|y - \Phi a\|^2$$

where $\Phi_{in} = \phi_n(x_i)$, yields the normal equations

$$\Phi^T \Phi\, a = \Phi^T y$$

When the basis is orthogonal with respect to the data, $\Phi^T\Phi$ is diagonal and each coefficient simplifies to [Bevington & Robinson 2003]:

$$a_n = \frac{\sum_i y_i\ \phi_n(x_i)}{\sum_i [\phi_n(x_i)]^2}$$

For standard Legendre polynomials P_n evaluated on the canonical domain [−1, 1] the projection integral is approximated by a 64-point Gauss-Legendre rule, which is exact for polynomials up to degree 127 [Abramowitz & Stegun 1964]:

$$c_n = \frac{2n+1}{2}\int_{-1}^{1} f(\mu)\ P_n(\mu)\, d\mu \approx \frac{2n+1}{2}\sum_k w_k\ f(\mu_k)\, P_n(\mu_k)$$

1.2 Hypothesis

It is conjectured, though not proven, that as the Legendre Polynomials have a standard domain [−1, 1] the coefficients provide a spectral fingerprint where underlying functions can be effectively discerned against an applicable database of precomputed function spectra. As G-S monomials are specific to underlying data, Legendre Polynomials provide a standard domain for comparison, the coefficients defining shape in suitably aligned domains. Higher order polynomial coefficients are subject to random noise, while the two lowest order coefficients (n = 0, 1) are subject to systematic bias (offset, linear drift). Therefore, filters can be applied with a hashing technique to quickly find the best functions for fitting data using coefficients as a spectral fingerprint. Shifted, normalized or aligned domains can be used on case-by-case basis to quickly identify preferential or underlying functions. This conjecture is supported by results presented later.

Methodology

2.1 Standard Domain Spectral Fingerprinting

Every smooth function *f* defined on the interval [−1, 1] can be expanded in the Legendre basis $\{P_n\}$:

$$f(\mu) = \sum_{n=0}^{\infty} c_n\ P_n(\mu)$$

$$c_n = \frac{2n+1}{2}\int_{-1}^{1} f(\mu)\ P_n(\mu)\, d\mu$$

The coefficient vector $c = [c_0, c_1, c_2, \ldots, c_n]$ constitute a spectral fingerprint of *f*. Functions that differ only by a positive multiplicative constant A (i.e., g = A·f) have identical unit normalized fingerprints:

$$\hat{c} = \frac{c}{\|c\|_2}$$

because $\|Ac\|_2 = A\|c\|_2$ cancels in the ratio. The stored fingerprint ĉ is therefore amplitude-independent and encodes shape information. The scalar $\|c\|_2$ (stored as the Scale field) preserves amplitude for full reconstruction when needed.

For data whose natural independent variable x does not lie in [−1, 1], a linear map $\mu = 2(x - x_{lo})/(x_{hi} - x_{lo}) - 1$ is applied before characterization. For angular variables $\theta \in [0, \pi]$ the physically correct mapping is $\mu = \cos\theta$, which is the natural argument of the Legendre polynomials appearing in classical expansions of spherical harmonics [Blanco et al. 1997, NIST DLMF §14].

2.2 Bias Stripping and the Shape Fingerprint

The P_0 term (n = 0) encodes only the DC offset of *f*, and P_1 (n = 1) encodes only a linear slope. Neither is diagnostic of functional form for higher order polynomials. Moreover, experimental data frequently carries additive offsets and linear drifts that are artifacts of calibration rather than features of the underlying process. The full normalized fingerprint ĉ can be a poor basis for shape matching when these terms are large.

A bias-stripped shape fingerprint is formed by zeroing or low weighting the first two coefficients before renormalization as introduced here:

$$c' = [0,\ 0,\ c_2,\ c_3,\ \ldots,\ c_n]$$

$$\hat{c}^s = \frac{c'}{\|c'\|_2}$$

The stored field NormCoeffsShape holds $\hat{c}^s$. Both ĉ and $\hat{c}^s$ are retained for backward compatibility; queries that use $\hat{c}^s$ are insensitive to arbitrary DC offsets and linear tilts in the data, making them substantially more robust to the systematic errors described above. The algorithm can do poorly for constants and straight lines for this reason.

2.3 Weighted Cosine Similarity

Standard cosine similarity

$$cos(a, b) = \langle \hat{a}, \hat{b} \rangle = \sum_n \hat{a}_n \hat{b}_n$$

weights every order n equally. Because high-order coefficients ($n \gtrsim 15$) are susceptible to noise amplification in finite-sample least-squares fits, and n = 0 and n = 1 encode bias rather than shape, a weighted cosine is employed for matching [see also Salton & Buckley 1988 for the analogous weighting concept in information retrieval]:

$$sim_w(a, b) = \frac{\Sigma_n\, w_n\, a_n\, b_n}{\sqrt{(\Sigma_n\, w_n\, {a_n}^2)}\sqrt{(\Sigma_n\, w_n\, {b_n}^2)}}$$

The weight profile used in the present first implementation is:

w_0	= 0.05	(DC offset — nearly ignored)
w_1	= 0.10	(linear tilt — low weight)
$w_2 \ldots w_{14}$	= 1.00	(core shape fingerprint — full weight)
$w_{15} \ldots w_{20}$	= 0.60	(fine structure — moderate weight)
$w_n > 20$	$= 0.60 \cdot \exp(-0.12 \cdot (n - 20))$	(noise floor — decaying)

The core band n = 2–14 carries the most discriminating shape information for smooth functions. The exponential decay beyond n = 20 prevents high-frequency noise from dominating the match score. Weights are tunable parameters; the mid-focus profile (used in a second-pass re-query when the first result is ambiguous) restricts full weight to n = 2–10, zeroes n ≤ 1 and n > 14 entirely, and applies a linear taper in between.

2.4 The Spectrum Library

A library of precomputed spectral fingerprints is built offline or in a background thread when first loading the program and spectra are stored in a binary file format (.slib). The library encompasses:

Mode 0 — 1D base catalog: approximately 155 functions including polynomials, trigonometric, exponential, sigmoid, rational, Bessel, Voigt, Fresnel, statistical distributions, and special functions (erf, sinc, gamma, etc.).

Mode 1 — 1D extended catalog: approximately 350 functions augmenting the base with additional parametric variants and associated Legendre polynomials.

Mode 2 — 1D linear combinations: up to ~630 K entries formed as pairwise linear combinations of the base catalog ($a_1 f_1 + a_2 f_2$ for sampled coefficient pairs), dramatically expanding coverage of composite functional shapes.

Mode 3 — 2D bivariate functions: approximately 25 K entries built as tensor products, sums, radial compositions, and directional combinations of 1D base functions, plus real spherical harmonics $Y_l^m(\theta, \phi)$ and surface harmonics.

Mode 4 — 3D trivariate functions: approximately 22 K entries built as separable, radial, and cross-term combinations of 1D base functions, plus real solid harmonics.

All functions are characterized using Gauss-Legendre quadrature at the precision noted above. The combined library (all modes) contains on the order of 750 000 entries. Each entry stores the function name, category, generating formula, dimensionality, polynomial order, amplitude scale, DC component, the full normalized fingerprint $\hat{c}$, and the shape fingerprint $\hat{c}^s$. Application of library functions should be somewhat judicious as noise and systematic weighting can miss simple functions (constant, straight line) when used improperly.

2.5 The Locality-Sensitive Hash Index

Brute-force cosine search over 750 000 entries would require evaluating approximately 1.6 × 10^7 multiply-accumulate operations per query — measurable latency for an interactive tool. A layered locality-sensitive hash (LSH) index reduces query time to under one millisecond by exploiting the geometric structure of the high-dimensional coefficient space [Indyk & Motwani 1998, Charikar 2002].

Layer 1 — Symmetry classifier. The ratio

$$r = \frac{\sum_{n\ odd} c_n^2}{\sum_n c_n^2}$$

classifies each entry as EVEN (r < 0.05), ODD (r > 0.95), or MIXED otherwise. A query vector is similarly classified; entries with incompatible symmetry class are excluded from the candidate set, reducing it by approximately 3×.

Layer 2 — Dominant-mode bucket. The index m* = argmax $|c_n|$ identifies the Legendre order carrying the most energy. Entries are placed in buckets indexed by (symmetry, m*). The query bucket and its two adjacent buckets (m* ± 1) are retrieved, reducing candidates by a further 10–15× within each symmetry class.

Layer 3 — Random-projection LSH. Eight independent hash tables are constructed. In each table, 12 random unit vectors $r_1, \ldots, r_{12}$ are drawn from an isotropic Gaussian distribution [Dasgupta & Gupta 2003], and the hash key of entry c is:

$$\mathrm{key}(c) = \sum_{b=1}^{12} \mathbb{I}\left[c \cdot r_b \geq 0\right] \cdot 2^{\,b-1} \in \{0, \ldots, 4095\}$$

This is the sign-random-projection scheme of Charikar [2002], which approximates angular distance: P[key(a) = key(b)] = 1 − arccos(cos(a,b))/π. For cos ≥ 0.95 the per-table collision probability is approximately 0.68; the union over 8 independent tables gives a miss probability below 0.1%. The candidate pool from all 8 tables is the set of entries that hash-collide with the query in at least one table.

Layer 4 — Exact cosine ranking. The candidate set (typically 20–200 entries from a 750 000-entry library) is scored by exact weighted cosine similarity using single-precision arithmetic, then sorted to yield the top-N matches.

Fallback behavior. If the candidate pool after Layer 3 contains fewer than 3×N entries, the dominant-mode bucket (Layer 2) is added in full. If the pool remains below N, a full linear scan is performed. This ensures that the top-1 result is never missed due to hash collision failure.

Hash verification. Because the LSH tables are stored in 32-bit floats for speed, very sparse spectra (low-order polynomials, heavily symmetric functions) can occasionally fall out of their expected bucket. Whenever the hash-layer top-1 cosine is below 0.99, or the hash-selected winner disagrees with a full double-precision linear scan over the same candidate pool, the double-precision winner is used. This verification costs tens of milliseconds on the full 750,000-entry library and guarantees that the reported top-1 is never lost to hash collision failure.

The hash index is built once (approximately 2 s for 750 000 entries on a modern 8-core processor) and stored embedded within the .slib file alongside the spectral fingerprints.

2.6 The Auto-Detect Fitting Pipeline

The pipeline has three responsibilities: (1) turn the noisy data into a Legendre spectrum on the same footing as the library; (2) ask the library — with appropriate weighting and, when warranted, a second pass — which catalog function has the closest spectrum; and (3) fit the chosen function's free parameters (amplitude A, horizontal stretch α, horizontal shift β, and offset C) back to the data. The refinements described here are motivated by the specific failure modes observed when any of the three is done in the naïve way.

2.6.1 Common Preprocessing (All Dimensions)

Before any fitting, raw axis values are mapped to $\mu \in [-1, 1]$ according to the declared domain-variable type. The mapping must be the same one used when the library was built; otherwise the query and the library live on different grids and cosine similarity is degraded.

- Linear: $\mu = 2\cdot(x - x\_lo) / (x\_hi - x\_lo) - 1$, for a finite window on a Cartesian axis.
- CosTheta: $\mu = \cos\theta$, $\theta \in [0, \pi]$. This is the physically exact mapping for functions on the sphere — Legendre polynomials $P_n(\cos\theta)$ are the natural colatitude eigenfunctions.
- Phi: $\mu = \phi/\pi - 1$, $\phi \in [0, 2\pi]$. The azimuth is folded into the Legendre domain but Legendre polynomials are not periodic, so purely azimuthal dependence is better represented by Fourier series. The mapping is kept for mixed-dependence functions where only cross-terms with θ matter.
- SemiInfinite: $\mu = 2\cdot r/R - 1$, $r \in [0, R]$. Suitable for radial coordinates or time starting from zero, with the user supplying the characteristic scale R.

2.6.2 1-D Pipeline

Step 1 — Legendre fit (degree 20). A full normal-equations least-squares fit is solved on the Legendre basis; this is necessary because sequential projection onto the standard Legendre basis gives systematically incorrect coefficients for arbitrary (non-Gauss–Legendre) data grids. The step yields the raw coefficient vector $c = [c_0, c_1, \ldots, c_{20}]$, a

Legendre reconstruction $\hat{y}$_Leg over the same x grid, and R^2_Leg, which serves as the "best degree-20 polynomial fit to this data" baseline used later to decide whether to re-query.

Step 2 — Bias decomposition. The DC term c_0 and the linear tilt c_1 are separated from the shape band c_2 … c_{20}. Two normalized query vectors are retained: the full L2-normalized $\hat{c}$ (all 21 components) and the bias-stripped shape vector $\hat{c}^s$ (first two components zeroed, then renormalized). Both are needed in the next step.

Step 3 — Weighted shape query, two-pass with pure-first merging. The library query is deliberately split into two passes whose results are merged so that named "pure" functions are evaluated before any linear-combination entry. This is the single most important correctness refinement relative to a flat top-N query.

Pass 1 — pure functions, full cosine on $\hat{c}$. Full-spectrum cosine similarity (including the bias band) is computed against every pure-function entry. This is the only place the unweighted NormCoeffs comparison is used; it bypasses weight suppression intentionally. Rationale: nearly-linear pure functions such as the Digamma function $\psi(x)$ and Lambert W have almost no mid-order shape content. If they are scored on the bias-stripped shape vector alone, virtually any library entry with meaningful mid-order energy will outrank them — even though one of them is the correct answer. Keeping the bias band in play for pure functions preserves their rank when they are the truth.

Pass 2 — all entries (pure + linear combinations), weighted cosine on $\hat{c}^s$. Weighted cosine (§2.3) is computed against every library entry. Bias-stripping + the weighted profile is amplitude-independent, which is the correct thing to do for combination entries whose DC/slope contribution varies with the mixing ratio.

The two candidate lists are merged with pure entries placed first, combination entries second (duplicates of the pure list are dropped from the combination section). The merged list is what the cascade in Step 4 consumes. Consequently, a pure function that fits the data with $R^2 \approx 1$ preempts every combination entry, regardless of which list ranked it higher on its own. [The tool provides a list of candidates that one can select from on their own.]

Step 4 — Cascade amplitude least-squares over top candidates. For each of the top-N merged candidates (N ≈ 20 for Auto-Detect, or 1 when the user has forced a specific entry), the two-parameter amplitude-and-offset model $y \approx A \cdot \hat{f}(\mu) + C$ is solved in closed form. Minimising $\|y - A\hat{f} - C\|^2$ reduces to a 2×2 linear system, costing O(M) per candidate where M is the number of data points. The candidate with the highest amplitude-LS R^2 is declared the cascade winner and is the only candidate taken into the expensive nonlinear stage. Running this cheap gate first is what prevents the pipeline from spending L-BFGS and Powell iterations on a mis-ranked spectral match — historically the most frequent failure mode, in which the rank-1 library entry looked plausible but the rank-3 entry was actually the generating function.

Step 5 — Multi-start L-BFGS refinement (Powell fallback). On the cascade winner only, the four-parameter model $A \cdot \hat{f}(\alpha \mu + \beta) + C$ is optimised over (A, α, β, C). Three starting points are attempted in sequence — {A=1, α=1, β=0} plus two alternates — and the best minimiser is kept. Multi-start guards against local minima that would otherwise lock α or β at a sign-flipped value; these arise readily because $R^2(\alpha, \beta)$ is near-discontinuous when the rescaled μ crosses ±1 and Legendre polynomials flip sign.

Gradients are exact, computed term-by-term from the Legendre expansion using the standard P'_n recurrence; no finite differences are used. Line search is Armijo-backtracking with Wolfe conditions. Memory depth is m = 6.

If L-BFGS fails to converge ($R^2 < 0.80$), Powell's direction-set method is applied to the same four-parameter model on the same candidate. Powell uses only function evaluations and is robust to the near-discontinuous landscape that defeats gradient-based methods near the rescaling boundary.

Step 6 — Mid-focus re-query when the Legendre baseline beats the parametric fit. The Legendre reconstruction from Step 1 always fits the data at degree 20, so if R^2_Leg markedly exceeds the cascade-winner fit R^2, the initial library match is almost certainly wrong — the spectral fingerprint was dominated by noise at one of the two extremes of the spectrum. The trigger is R^2_Leg > R^2_best + 0.05 AND R^2_Leg > 0.60; the 0.60 floor prevents re-querying when both fits are meaningless (pure-noise data).

On trigger, the library is re-queried with the tighter mid-focus weight profile: $w_0 = w_1 = 0$ (bias band hard-zeroed, no longer just down-weighted), $w_2 \ldots w_{10} = 1$ (core shape at full weight), $w_{11} \ldots w_{14}$ = linear taper, w > 14 = 0 (high-order noise tail hard-zeroed). This isolates matching entirely to the most reliable mid-range shape information. The top 5 re-query candidates are passed through Steps 4 and 5, and the best overall result becomes the reported parametric fit.

Step 7 — Supplementary Gram–Schmidt and spline fits. A Gram–Schmidt orthogonal polynomial fit (§Introduction) is always computed at the user-requested degree, together with a thin-plate spline interpolation on the raw data. These are no longer fallback overrides of the parametric fit; they are retained as supplementary models shown on the Parameters tab and selectable on the Graph tab. The user — not the algorithm — decides which representation is preferable for their downstream application, especially in cases where no library entry produces a satisfactory fit.

2.6.3 2-D Pipeline

Why not trapezoidal projection. A natural first implementation of the 2-D spectral projection is the trapezoidal rule on the user's uniform (x, y) grid. It does not work. Trapezoidal quadrature on a uniform grid systematically over-estimates high-order diagonal Legendre integrals — empirically by about 1.776× at tensor order 8 — because the Euler–

Maclaurin endpoint correction is not zero for oscillatory integrands. The over-estimated diagonals rotate the query coefficient vector away from every library entry, including the correct one, so self-match fails for any function with mid-to-high-order content. The refined 2-D pipeline uses a tensor-product least-squares solve on the same Legendre basis as the library builder, which produces coefficients on the same footing as the Gauss–Legendre-quadrature library.

Step 1 — Tensor-product LS Legendre projection. LegendreFit2D solves the normal equations for the flattened coefficient vector c_{i,j}, with $K = (N+1)^2$ basis functions clamped so $K \leq M$. The fit order is capped at the library build order. The resulting coefficients are directly comparable to the library's GL-quadrature coefficients.

Step 2 — 2-D bias stripping. Three low-order terms are zeroed before renormalising: the DC term c_{0,0}, the linear-in-y term c_{0,1}, and the linear-in-x term c_{1,0}. These are the DC and linear-tilt contaminations in 2-D; stripping them makes the shape-fingerprint insensitive to calibration offsets and linear drift on either axis.

Step 3 — Weighted shape query. The 2-D query uses a single-pass weighted cosine against every library entry; there is no pure/combo split in 2-D because the library has no pure nearly-linear bivariates analogous to Digamma. Weights are keyed on total tensor order $tot = n_1 + n_2$: $tot = 0 \rightarrow 0.05$ (DC), $tot = 1 \rightarrow 0.10$ (linear bias), $tot \in [2, 12] \rightarrow 1.0$ (core), $tot \in [13, 20] \rightarrow 0.6$ (soft roll-off), $tot > 20 \rightarrow 0.6 \cdot \exp(-0.12 \cdot (tot - 20))$.

Step 4 — Cascade amplitude LS. Same closed-form two-parameter amplitude-and-offset fit as the 1-D pipeline, evaluated over the top-N candidates (N up to 10 when $R^2\_Leg > 0.60$). The best candidate is promoted to position zero of the match list so the Goodness-of-Fit tab and Graph tab display the correct winner. Note that α and β are not used in the 2-D surface fit; only the amplitude A and offset C are free.

Step 5 — Mid-focus re-query. Same trigger and intent as the 1-D pipeline: if the Legendre reconstruction R^2_Leg notably beats the cascade winner R^2 and $R^2\_Leg > 0.60$, re-query with a tightened weight profile and cascade the top re-query candidates. Supplementary GS (polynomial) and 2-D thin-plate spline fits are always stored as well.

2.6.4 3-D Pipeline

Step 1 — Kronecker-structured tensor-product LS. The 3-D trapezoidal projection has an even more severe diagonal overestimate than 2-D — up to ~5.6× at high order. The refined 3-D pipeline therefore uses a Kronecker-structured LS solve: $(G_x \otimes G_y \otimes G_z) \cdot c = R$, executed as three sequential 1-D solves along each axis. Each G is a 1-D Gram matrix of size $(N+1) \times (N+1)$. The maximum library order per axis is capped at 8 ($9^3 = 729$ coefficients per entry) to match the offline library build; fitting at higher order is wasted effort because no library entry stores those modes.

Step 2 — 3-D bias stripping. The DC term (total order 0) and the three linear-tilt terms (total order 1) are zeroed before renormalising — the direct 3-D analogue of the 1-D bias strip and the 2-D three-term strip.

Step 3 — Gram–Schmidt baseline computed before the library query. Unlike the 1-D and 2-D pipelines, the 3-D pipeline computes the GS polynomial baseline R^2 up front. In 3-D, cascade depth and the re-query threshold both depend on knowing how well a generic polynomial fits the data; computing the baseline first lets these decisions be made with the correct information rather than retroactively.

Step 4 — Weighted query with a 3-D-specific profile. The 3-D library query uses an inline weight profile rather than the general MakeWeightProfile, because (a) the maximum library order is 8, which makes the 1-D/2-D soft-end threshold of 20 irrelevant — a hard zero above total order 7 is more faithful to the actual information content; (b) the 3-D library contains no linear-combination entries, so the pure-first two-pass merge adds no value and the single-pass weighted query is called directly; (c) total-tensor-order weighting (tot = $n_1+n_2+n_3$) is more natural in 3-D because any single axis beyond order 5 already dominates the cross terms. The profile is: tot ≤ 1 → 0 (bias, zeroed), tot ∈ [2, 5] → 1.0 (core shape), tot = 6 → 0.5 (taper), tot ≥ 7 → 0 (above library order).

Step 5 — Cascade amplitude LS. Same two-parameter (A, C) closed-form amplitude fit on each top candidate. The cascade winner becomes the reported match; the best 3-D candidate is moved to position zero of the match list for display.

Step 6 — Mid-focus re-query. Same trigger as 1-D and 2-D (R^2_GS > R^2_best + 0.05 AND R^2_GS > 0.60). The 3-D mid-focus profile is tighter still: tot ≤ 1 → 0, tot ∈ [2, 4] → 1.0, tot = 5 → 0.5, tot ≥ 6 → 0. Cutting to total order ≤ 4 isolates only the lowest-noise shape bands — in noisy 3-D data this is typically all the signal that is reliably present.

Step 7 — Supplementary fits. The Gram–Schmidt polynomial baseline (always computed in Step 3) and a stack of per-slice 2-D thin-plate splines are stored as supplementary displays. As in 1-D and 2-D, these do not override the parametric match; they give the user an alternative representation when the library's best match is a poor fit.

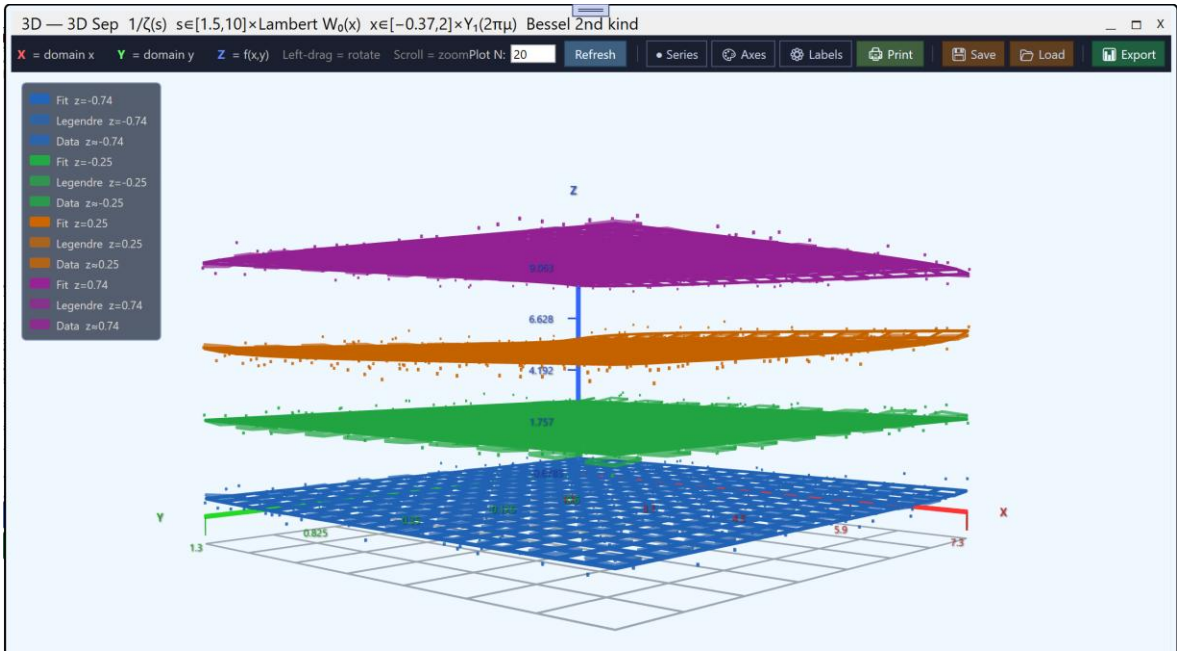
In summary: the refined pipeline separates spectral ranking (a cheap O(M) cascade amplitude LS over a small candidate list) from parametric refinement (multi-start L-BFGS with a Powell fallback), runs a second library query with a tightened weight profile when the initial match is evidently wrong, and keeps the Gram–Schmidt and spline fits as supplementary views rather than last-ditch overrides. The three dimensionalities share this architecture; they differ mainly in the projection method (all use LS on the Legendre basis) and in the weight profile appropriate to the library's build order in that dimension.

2.7 Windows based Software Tool Results

A software tool testing the hypothesis has been developed and is useful for curve fitting, spline generation and automatic detection of underlying or preferable functions using the methodology described above. Using this tool we have found correct generating functions with partial domains and noisy data. The tool provides some background on the functions and further refinements and improvements are in the works.

2.7.1 3D Sample Results

Below is an example using very noisy data with trivariate functions with the spectral fingerprint and mid-weight coefficient preference applied. Remarkably the tool often finds the generating function with statistical noise and even partial or extended transformed domains.



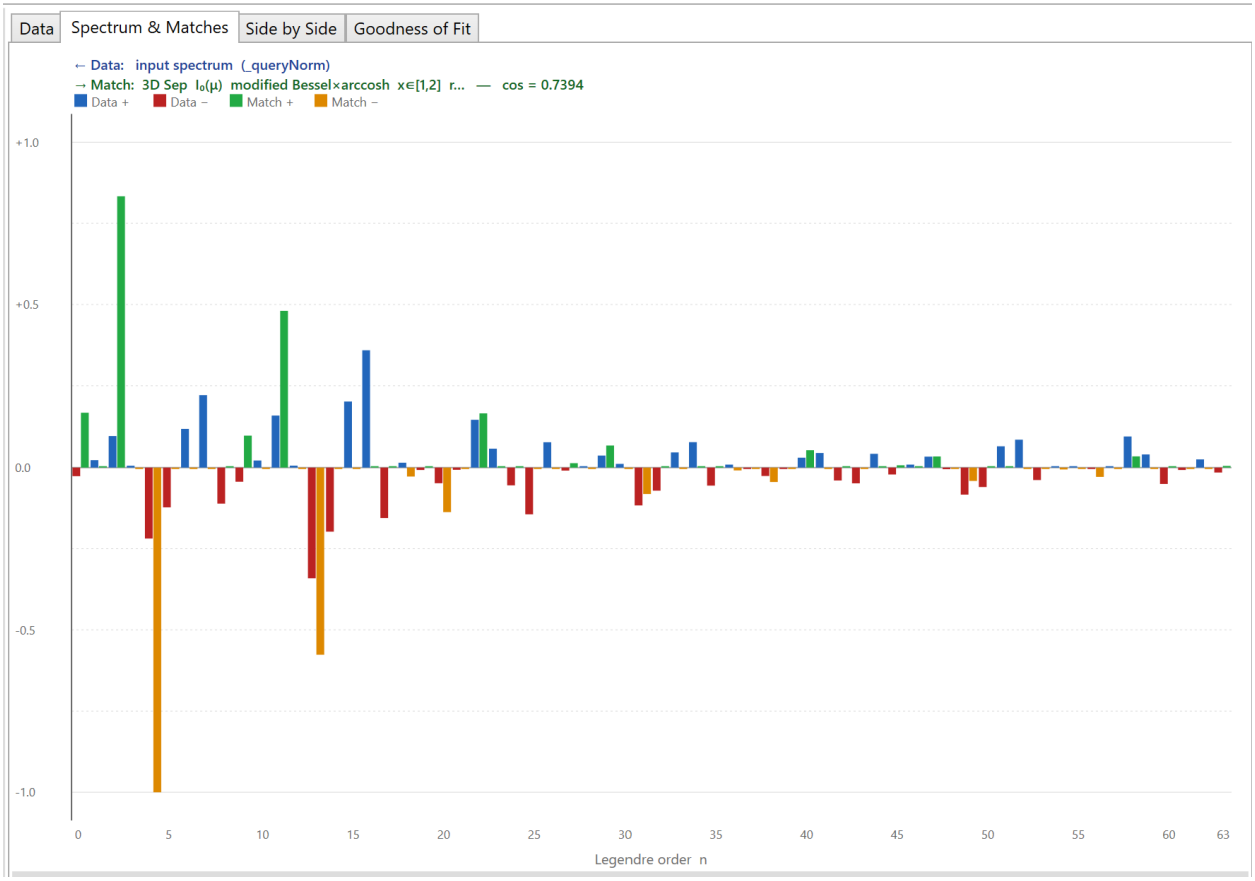
Data | Spectrum & Matches | Side by Side | Goodness of Fit

Best match:	3D Sep 1/ζ(s) s∈[1.5,10]×Lambert $W_0(x)$ x∈[−0.37,2]×$Y_1(2\pi\mu)$ Bessel 2nd kind
Cosine similarity:	0.988235
Fit method:	Shape LS (3D)
R^2 (parametric):	0.449903
R^2 (Legendre):	0.331244
RMSE:	0.2023
Bias (c_0):	-0.02916
Amplitude A:	0.9959

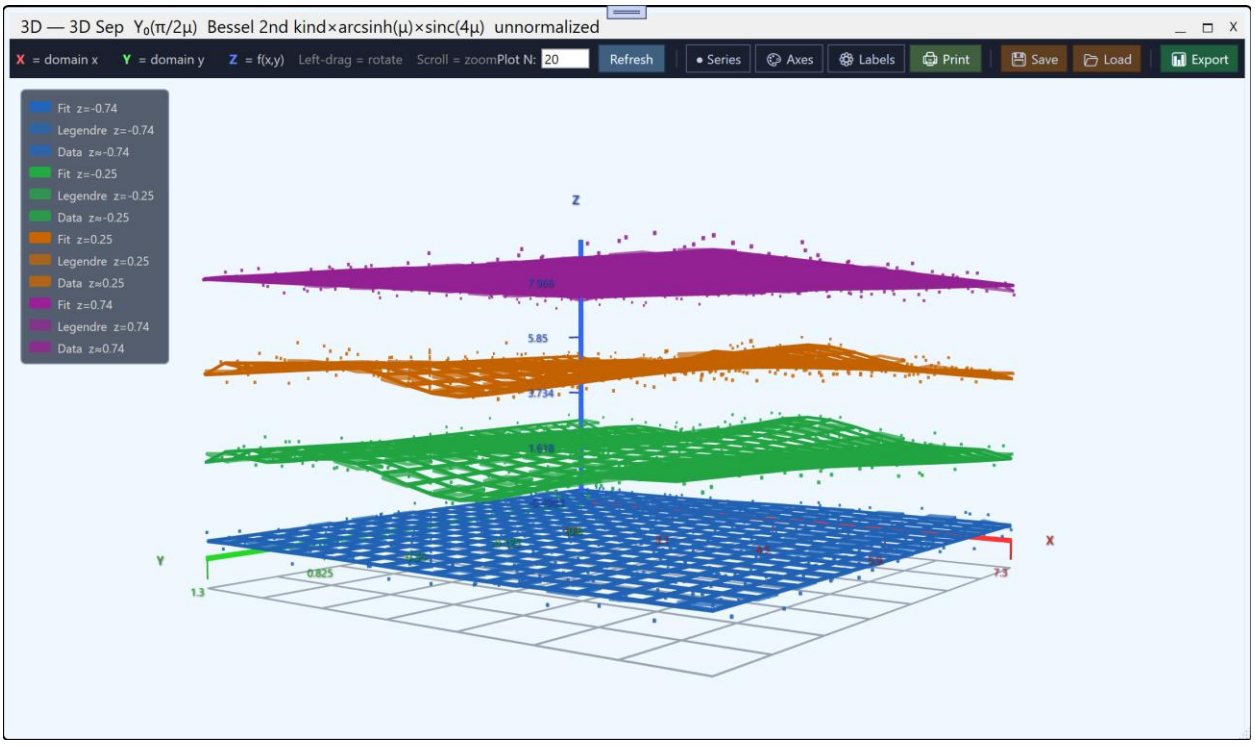
x	y	z	|residual|	% diff
1.7	-0.6	-0.99	0.1077	48.71
1.7	-0.6	-0.9039	0.1022	67.94
1.7	-0.6	-0.8178	0.3418	102.90
1.7	-0.6	-0.7317	0.1027	72.14
1.7	-0.6	-0.6457	0.1918	127.51
1.7	-0.6	-0.5596	0.1085	103.81
1.7	-0.6	-0.4735	0.003634	-9.96
1.7	-0.6	-0.3874	0.1192	202.45
1.7	-0.6	-0.3013	0.1992	83.13
1.7	-0.6	-0.2152	0.1184	86.72
1.7	-0.6	-0.1291	0.3765	135.84
1.7	-0.6	-0.04304	0.4703	161.46
1.7	-0.6	0.04304	0.1478	38.87
1.7	-0.6	0.1291	0.03719	-18.16
1.7	-0.6	0.2152	0.09775	32.72
1.7	-0.6	0.3013	0.1223	2543.10

From the tool data above (R^2 = 0.4499), and fairly flat functions over the domain of interest the correct function generating the data was discerned owing to cosine matching. We have found (not proved) that the Legendre coefficient spectra are sufficiently unique at least for the data sets we have to usually find the correct underlying function, or at least one which provides a fairly good fit with respect to the data provided. This becomes apparent when using the tool. We can compare the spectra of found function and spectra of identified closely matching functions, and one can visually see the uniqueness of the fingerprint. Here is compared the data with the next best spectral match function identified by spectral fingerprint, and one can see how different these spectra are in practice.

For complex functions such as that presented above, near candidate functions of the same form can be compared with their various parameters. The parametric fitting can be refined to interpolate in a predictor-corrector fashion between found spectral like functions to zero in on more exact generating function parameters.



An example with R^2=0.2926



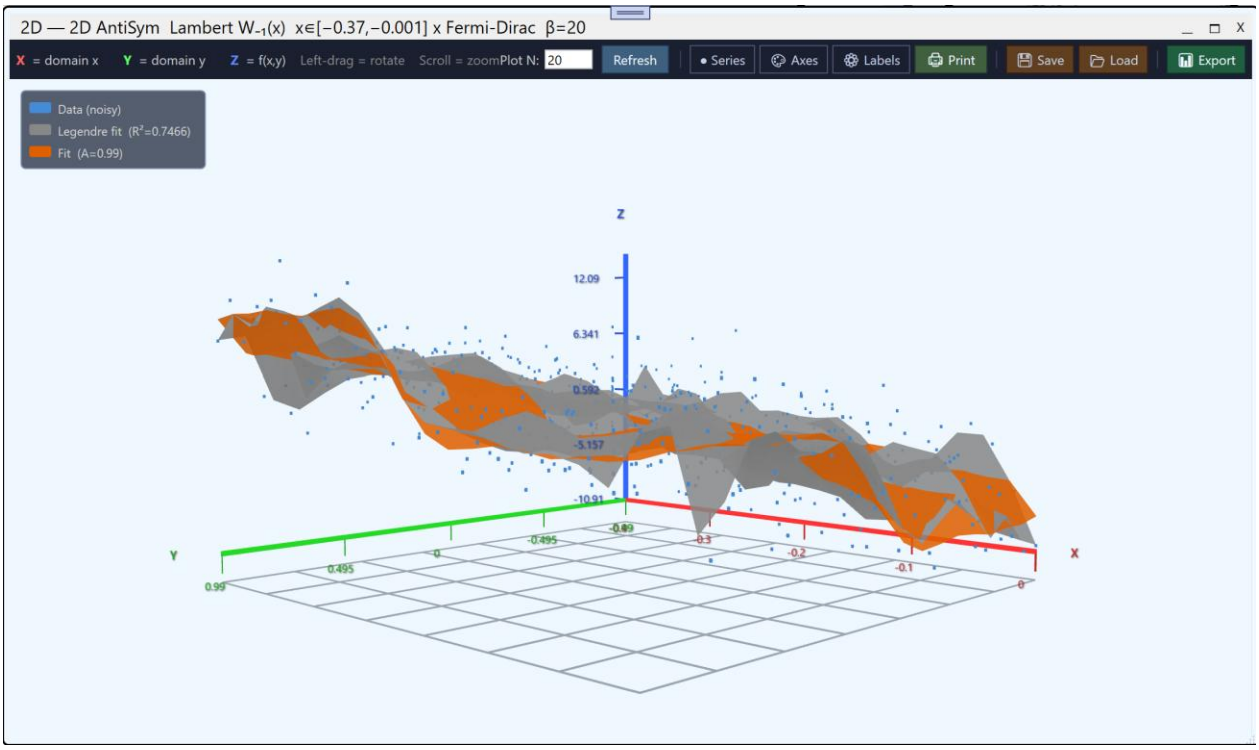
Data | Spectrum & Matches | Side by Side | Goodness of Fit

Best match:	3D Sep $Y_0(\pi/2\mu)$ Bessel 2nd kind×arcsinh(μ)×sinc(4μ) unnormalized
Cosine similarity:	0.989966
Fit method:	Shape LS (3D)
R^2 (parametric):	0.292628
R^2 (Legendre):	0.267128
RMSE:	0.1998
Bias (c_0):	-0.0005595
Amplitude A:	0.9902

x	y	z	|residual|	% diff
1.7	-0.6	-0.99	0.05129	50.36
1.7	-0.6	-0.9039	0.3671	91.16
1.7	-0.6	-0.8178	0.3028	95.98
1.7	-0.6	-0.7317	0.1984	91.93
1.7	-0.6	-0.6457	0.2235	131.27
1.7	-0.6	-0.5596	0.009846	-11.87
1.7	-0.6	-0.4735	0.3	69.17
1.7	-0.6	-0.3874	0.2185	55.77
1.7	-0.6	-0.3013	0.1281	-158.40
1.7	-0.6	-0.2152	0.2723	53.34
1.7	-0.6	-0.1291	0.2822	52.16
1.7	-0.6	-0.04304	0.263	49.38
1.7	-0.6	0.04304	0.000501	-0.19
1.7	-0.6	0.1291	0.02345	8.31
1.7	-0.6	0.2152	0.1024	-75.35
1.7	-0.6	0.3013	0.1094	34.37
1.7	-0.6	0.3874	0.1198	-223.82
1.7	-0.6	0.4735	0.1692	55.86
1.7	-0.6	0.5596	0.117	483.70
1.7	-0.6	0.6457	0.08114	290.79

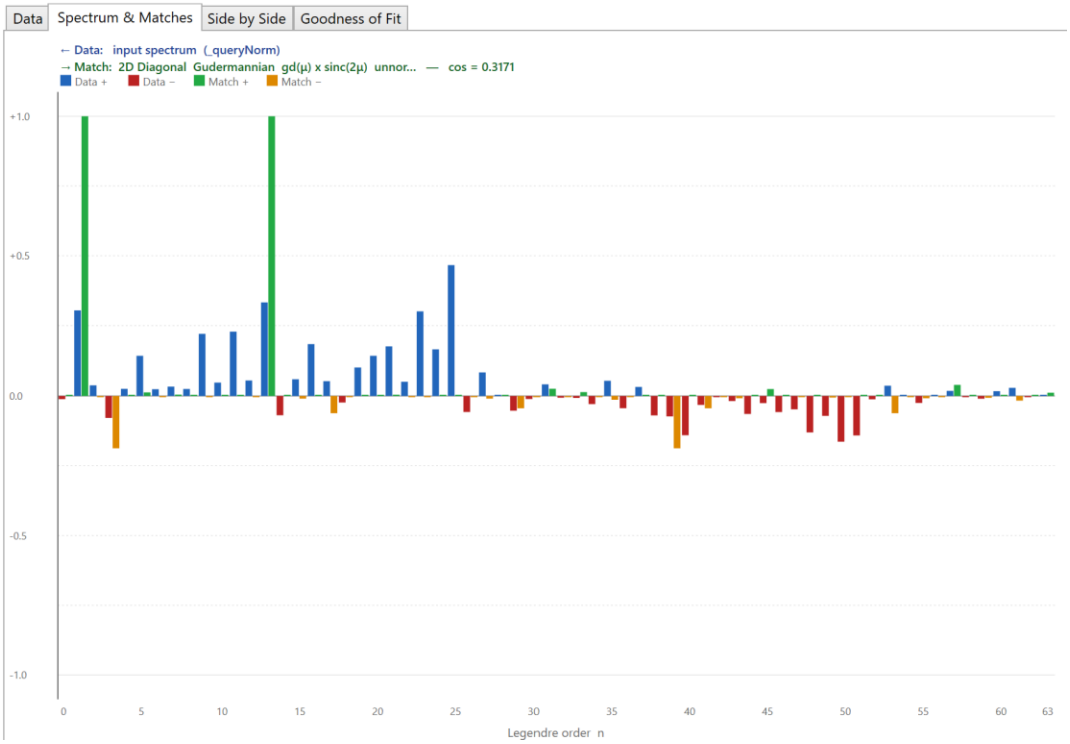
2.7.2 2D Example

At times the Legendre fit will be superior to the original generating function curve fit.



With sufficient noise in the data, an incorrect generating function will be found, and a best fit will be produced using a spectrally similar function. [Lambert $W_{-1}(x)$ x Fermi-Dirac β=20]

Even though spectra are unique, sufficient noise will throw off the high order polynomial terms, and even with a filter of these high frequency errors, the incorrect generating function can be identified. Presented below is a comparison of two function spectra with similar hash cosines, different spectra, but within a search band for matching data of an underlying generating function (A Lambert $W_{-1}(x)$ × Fermi-Dirac β=20).

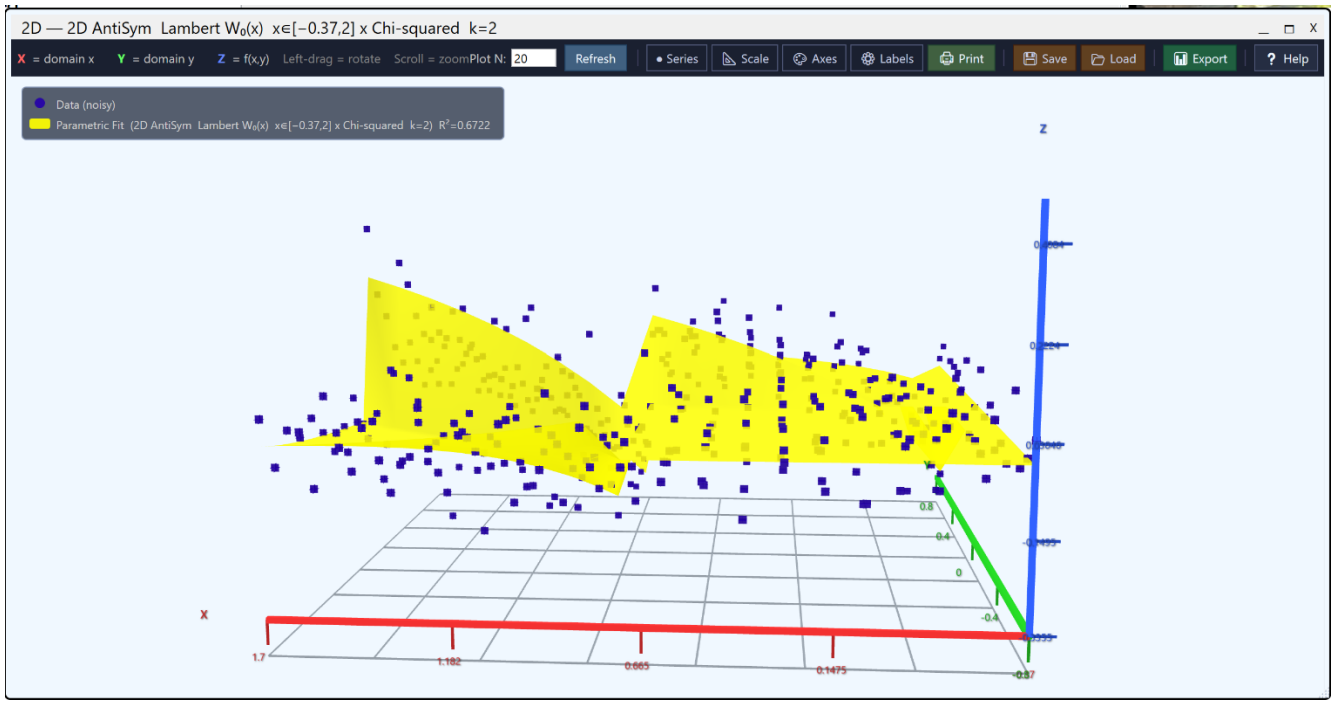


For practical purposes one can use Legendre or Gram-Shmidt polynomials for applying such functions in a traditional sense when this condition occurs. The computer code provides outputs that show when the practical Legendre fit is better than found underlying function, and at that juncture the user can choose to use G-S polynomial or Legendre fits.

Best match:	2D Diagonal Gudermannian gd(μ) x sinc(2μ) unnormalized
Cosine similarity:	0.317106
Fit method:	Shape LS (2D)
R^2 (parametric):	0.878188
R^2 (Legendre):	0.995334
RMSE:	1.185
Bias (c_0):	-0.1339
Amplitude A:	6.778

Domain interpolation and extrapolation within reason allow the correct function to be found, even with multiple dimensions and at domain mismatch. Part of this is because the

user can input a large number of tier-2 search number where close candidate R^2 are compared. Presented below is surface fit with where the domain does not match the spectra in the catalogue. With noise the tool it still found the correct function.



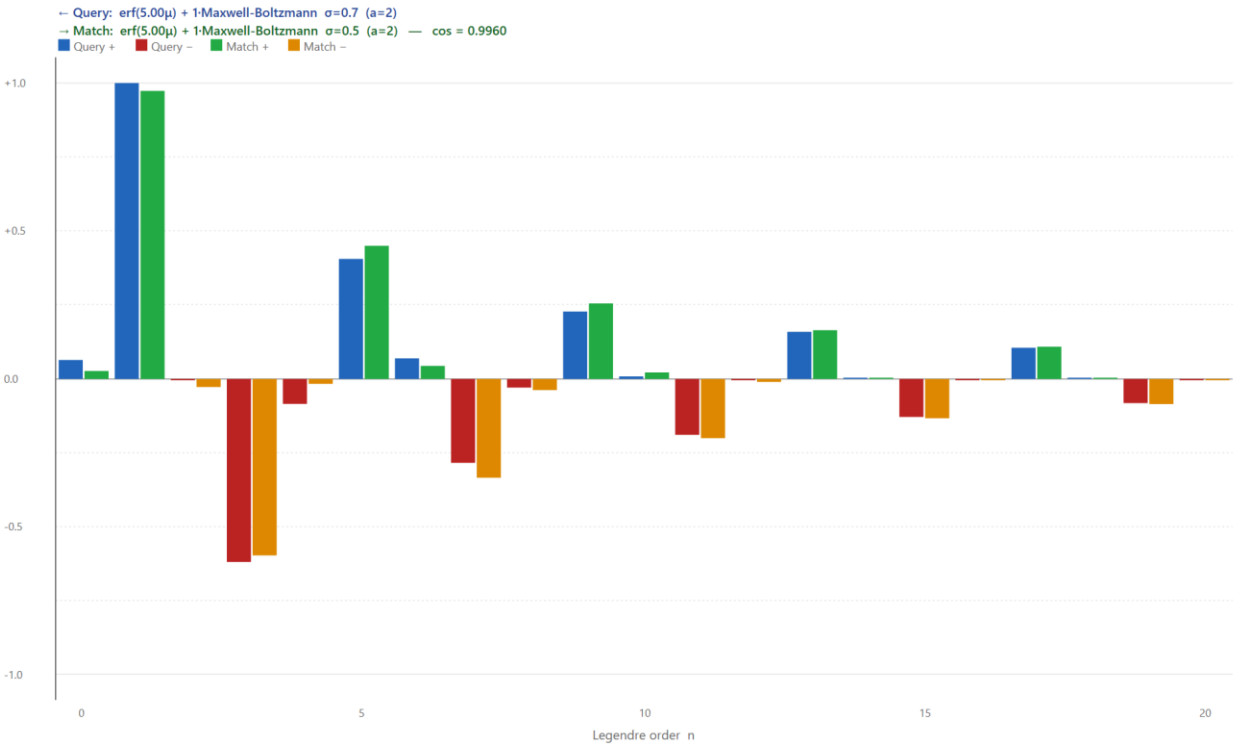
With a large number of combinations and particularly with greater dimensions and domain mismatch, the direct search using cosines and spectra do not produce the exact generating function.

In order to find the correct function, an additional tier 2 search is used to compare the R^2 of the top candidate functions based on the spectrum cosine fitting. This is a practical necessity given closely matching linear combinations of functions and multiple dimensions.

In future refinements, QR decoupling or G-S processes will be used to reconstruct the coefficient matrices of the individual functions that compose the vector products. Given the bias-stripped, weighted observed fingerprint c_{obs} and a chosen subset of K base-function fingerprints $c_1, c_2, \ldots, c_k$ drawn from Mode 0/1 of the catalog, the contributions $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_K]$ of each base function are recovered by solving $M\alpha \approx c_{obs}$ where $M = [c\_1 \mid c\_2 \mid \ldots \mid c\_K]$ via QR decomposition $M = Q\,R$ and $\alpha = R^{-1}Q^{T}c_{obs}$.

2.7.3 Near Identical Data Discrimination, 1D Example

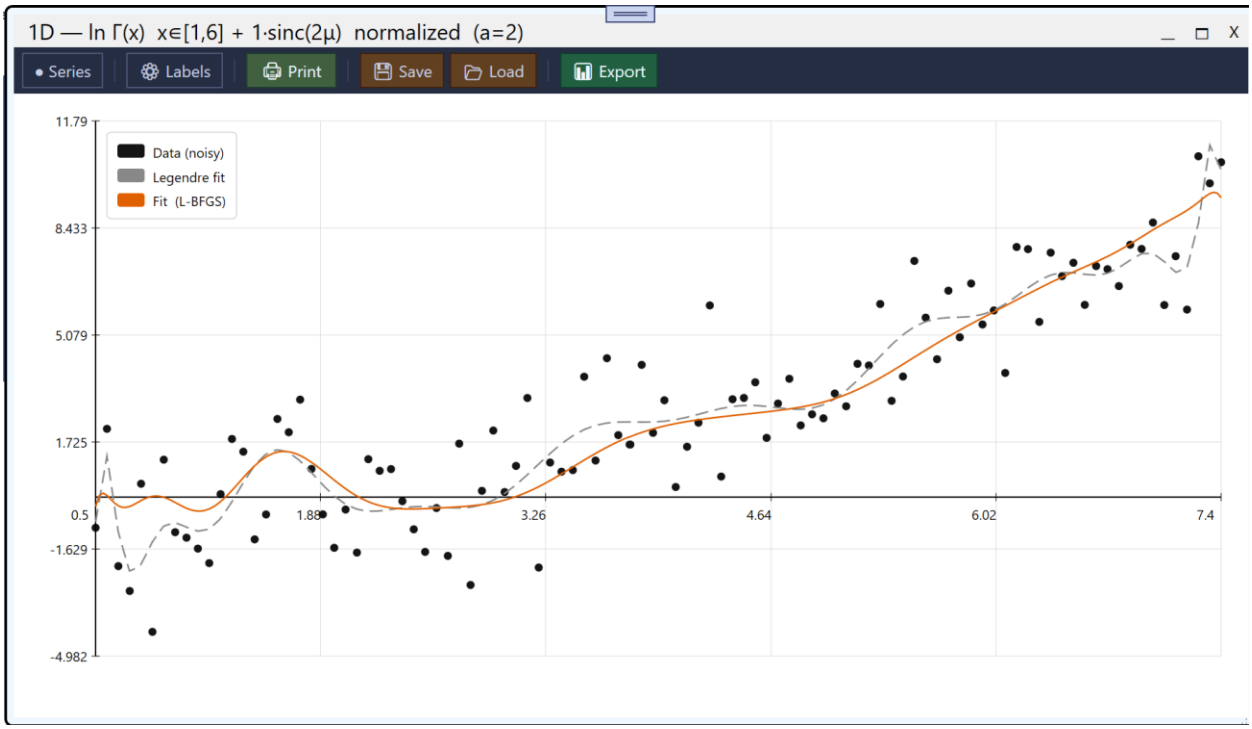
One of the remarkable things found using this tool is how well functions could be found with very similar sets of noisy data and spectra. There are limitations of course, but the following test case shows very close functions and noisy data discriminated objectively using spectra that are relatively close. Presented below is the spectral difference between of a linear combination of functions, $erf(5.00\mu) + 1 - Maxwell\ Boltzmann\ (a = 2)\ \sigma = 0.7\ and\ 0.5$. Below is a comparison of spectra followed by fitting with noise, the cosine match for the blue and red bars is 1.00, and 0.9960 for a near comparison function.


← Query: erf(5.00μ) + 1·Maxwell-Boltzmann σ=0.7 (a=2)
→ Match: erf(5.00μ) + 1·Maxwell-Boltzmann σ=0.5 (a=2) — cos = 0.9960
Query +
Query −
Match +
Match −
+1.0
+0.5
0.0
-0.5
-1.0
0
5
10
15
20
Legendre order n


Series
Scale
Labels
Print
Save
Load
Export
Help
Nosiy Data erf(5.00μ)_+_1·Maxwell-Boltzmann__σ=0.7__(a=2)_fit
Noisy Data erf(5.00μ)_+_1·Maxwell-Boltzmann__σ=0.5__(a=2)_fit
erf(5.00μ)_+_1·Maxwell-Boltzmann__σ=0.7__(a=2)_fitFUNCTION
erf(5.00μ)_+_1·Maxwell-Boltzmann__σ=0.5__(a=2)_fitFUNCTION
1.792
1.101
0.4092
-0.2822
-0.9736
-1.665
-1
-0.6
-0.2
0.2
0.6
1

2.7.4 Other 1D Examples

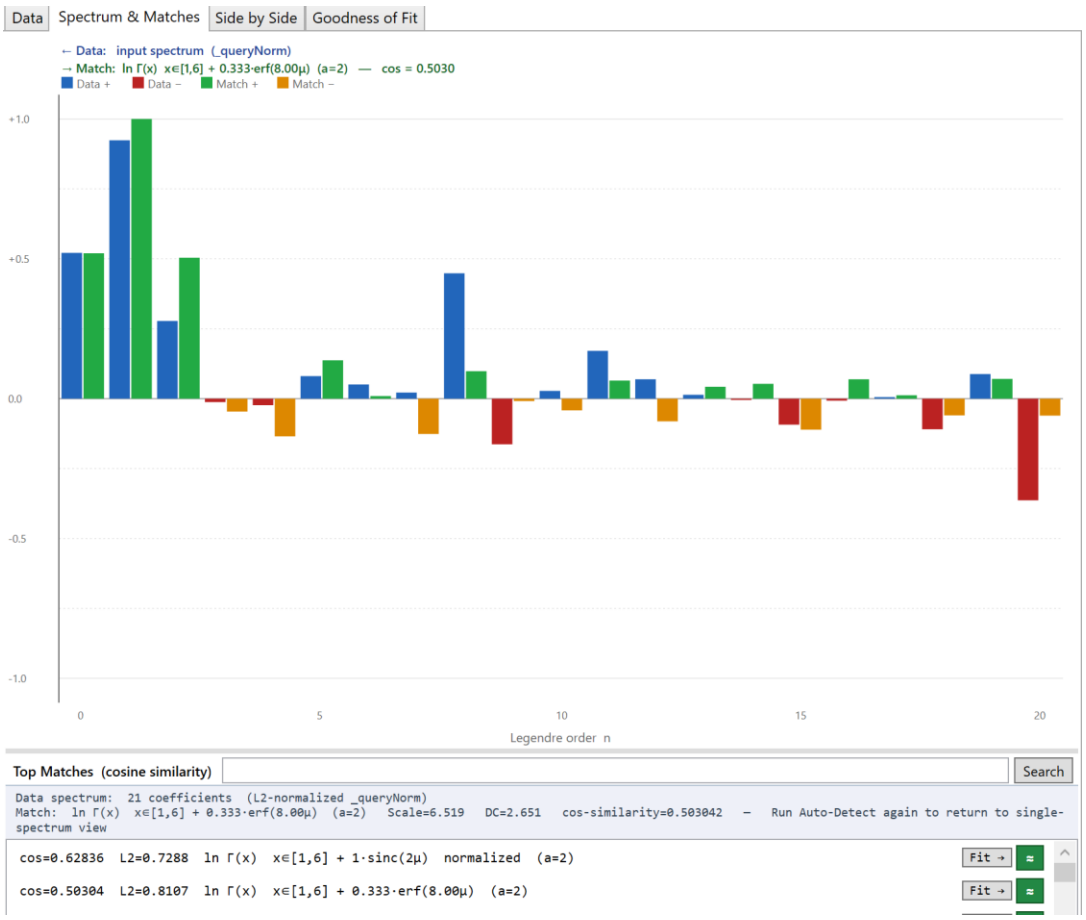
Most curve fitting is one dimensional and again, even with noisy data and poor R^2 the code automatically found the correct noisy data generating function:



Data | Spectrum & Matches | Side by Side | Goodness of Fit

Best match:	ln Γ(x) x∈[1,6] + 1·sinc(2μ) normalized (a=2)
Cosine similarity:	0.628355
Fit method:	L-BFGS
R^2 (parametric):	0.808356
R^2 (Legendre):	0.846976
χ^2 (sum sq. res.):	209.43
RMSE:	1.447
Max |residual|:	4.247
Bias (c_0, c_1):	2.773 4.914
Fit params:	A=1.016 α=1.002 β=0.005451 C=-0.08015

x	y data	y fit	|residual|	% diff
0.5	-0.960154	-0.292785	0.6674	69.51
0.569697	2.13922	0.0541914	2.085	97.47
0.639394	-2.16174	-0.272733	1.889	87.38
0.709091	-2.9397	-0.292369	2.647	90.05
0.778788	0.419202	-0.107533	0.5267	125.65
0.848485	-4.22026	0.0268736	4.247	100.64
0.918182	1.17364	-0.00500155	1.179	100.43
0.987879	-1.09919	-0.167398	0.9318	84.77
1.05758	-1.27057	-0.34871	0.9219	72.55
1.12727	-1.61344	-0.437707	1.176	72.87
1.19697	-2.06643	-0.369365	1.697	82.13
1.26667	0.0928651	-0.138036	0.2309	248.64
1.33636	1.82176	0.212618	1.609	88.33
1.40606	1.42473	0.611123	0.8136	57.11
1.47576	-1.32302	0.981261	2.304	174.17
1.54545	-0.543472	1.26041	1.804	331.92
1.61515	2.44995	1.41032	1.04	42.43
1.68485	2.03268	1.42014	0.6125	30.13



In the case of sinc() only 10 coefficients are relevant and only five have meaningful values. Domain definition and function set management for computing project specific libraries should increase the ability to discriminate functions with poor R^2.

2.7.5 $sinc(4\mu)$ Study

Presented below is a $sinc(4\mu)$ generating function of noisy data with σ=0.25 applied directly to the range as figure 2.7.5.a. The generating function in the first plot is shown along with the data. The Legendre fit of that generating function is exact.

Below it is figure 2.7.5.b showing the very poor fit Legendre polynomial, the generating function, the correctly found function fit parametrically, and two alternative functions that were close matches and also fit parametrically.

Figure 2.7.5.c shows the original spectrum of a pure $sinc(4\mu)$ function while Figure 2.7.5.d. shows the spectrum of noisy data comparable to the function generating Figure 2.7.5.a with σ=0.25 applied to the range. Even with simple even and odd functions halving the effective coefficients for comparison, the weighting scheme provides best fit find results.

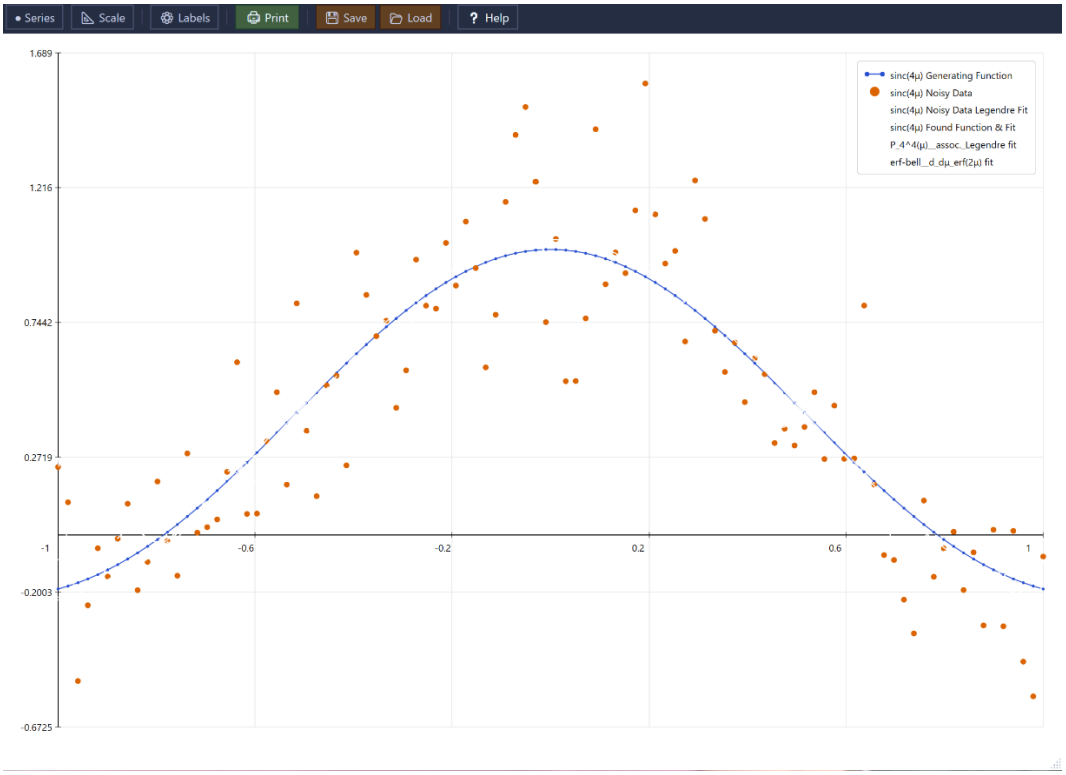


Figure 2.7.5.a. $sinc(4\mu)$ Generating Function with Significant Noise

Figure 2.7.5.b. $sinc(4\mu)$ Fitting Functions with Noise

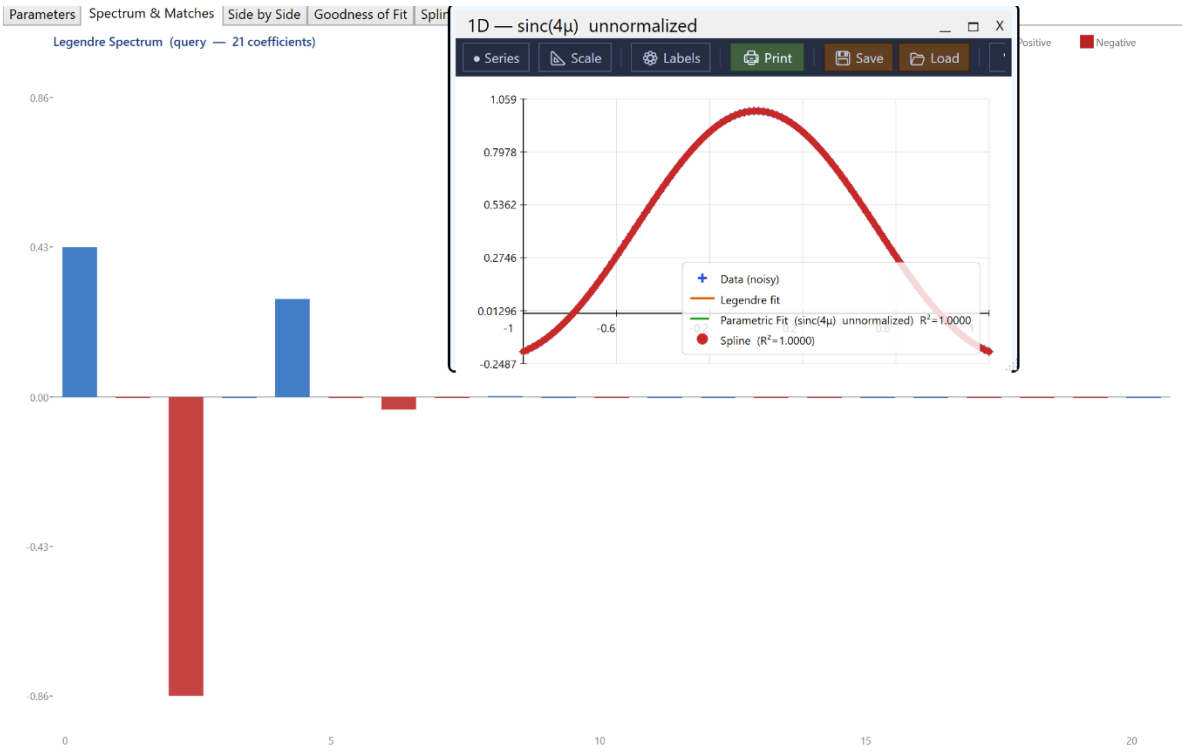


Figure 2.7.5.c. $sinc(4\mu)$ Generating Function Spectrum No Noise

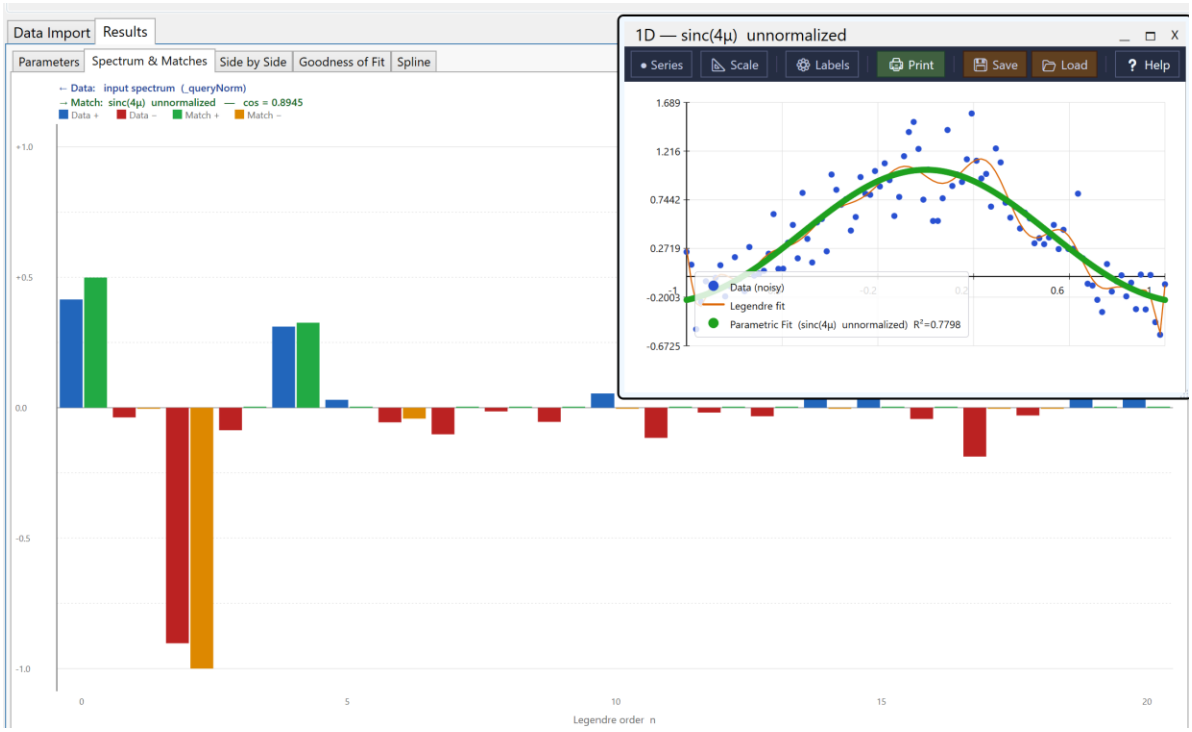


Figure 2.7.5.d. $sinc(4\mu)$ Generating Function Spectrum Noisy Data

3.0 AI/ML Preliminary Results

One immediate application of this methodology is in use to discern or correct activation functions for selected or generic neurons during the Artificial Intelligence (AI) Machine Learning (ML) process. A Multilayered Perceptron (MLP) was created and tested in various cases with MNIST data on an *i5* computer without a GPU to see if it was possible to improve upon a variety of initial warmup activation functions using statistical techniques and function lookup as described in this paper. Initial results are promising using an automated process. Details of the activation function selection criterion and network integration are reserved pending patent proceedings.

3.1 MNIST 10 digit MLP Spectral Search Preliminary Results

3.1.1 MNIST 10 Digit Activation Function Loss Curves

Superior activation functions included Softplus and Sigmoid, however the new Spectral Search using Softplus as the warm-up (default) activation function provided the best results. Various statistical criterion was used to determine if selected neurons could benefit from activation function search, and in many instances, though not all, the application of the methodology to find a best generating function for noisy data yielded a comparable or better loss curve as shown below.

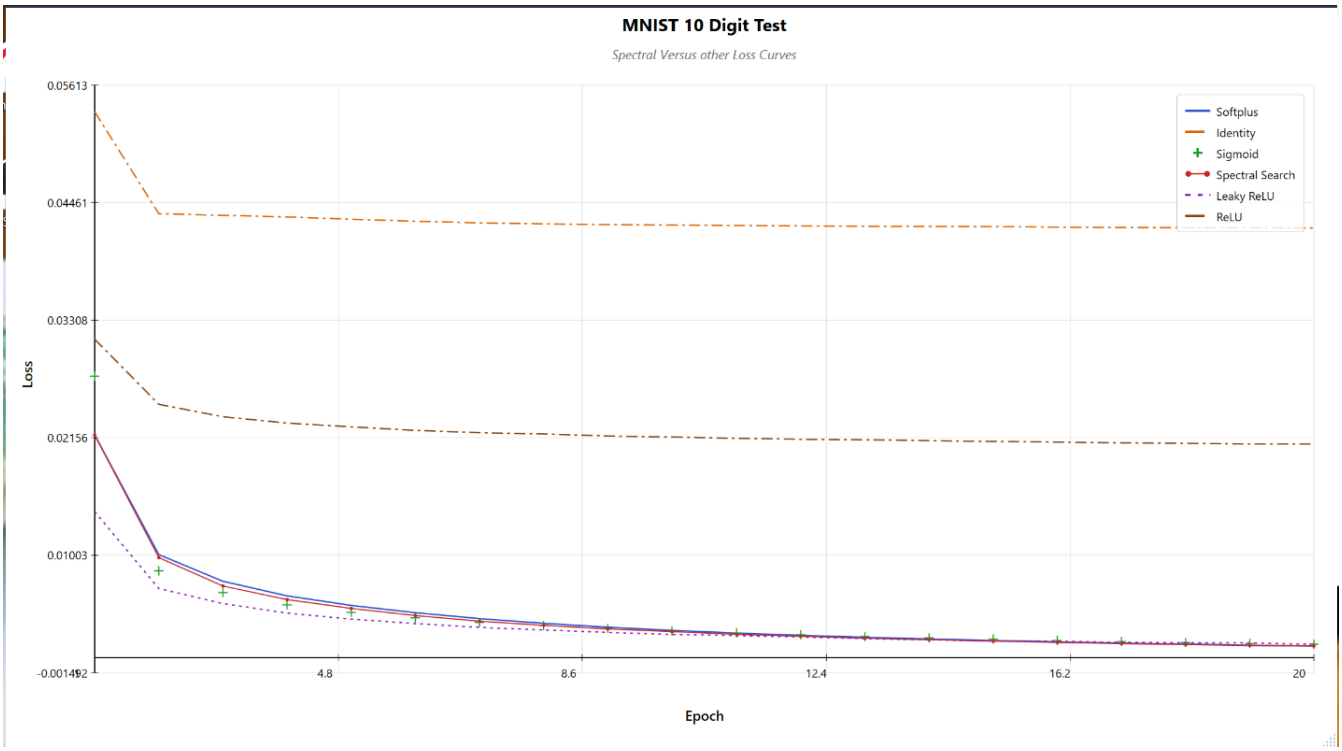


FIG. 3.1.1 MNIST 10 Digit Activation Function Loss Curves

3.1.2 Typical Spot Check

Presented below is an example of a sanity spot check of test labels against test training data for the MLP after a search process using the preliminary tool for MNIST 10 digit data.

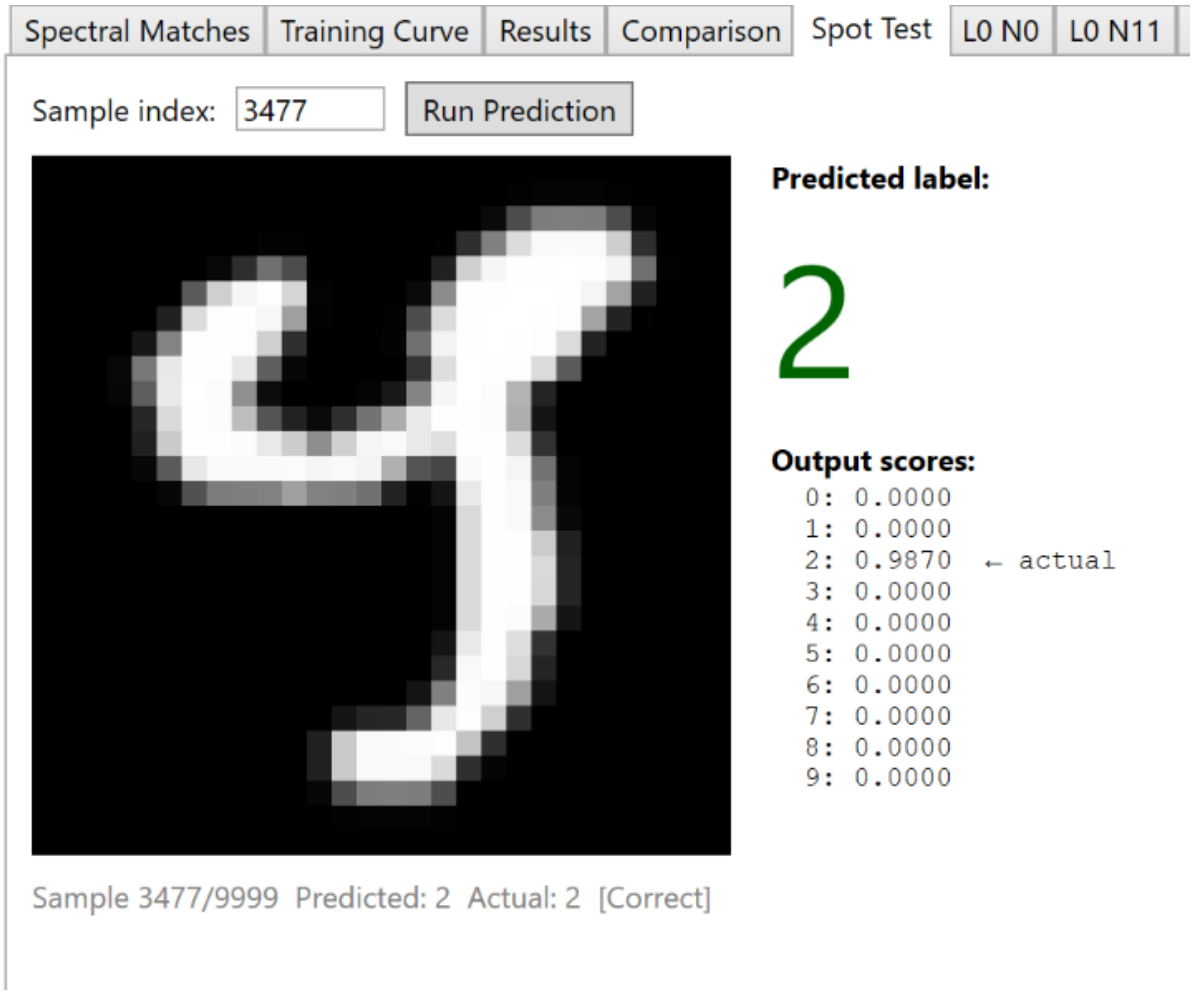


FIG. 3.1.2 Spot Check of MNIST MLP Output

3.1.3 Activation breakdown

Shown in the figure below shows the middle layer and neurons with associated activation functions. The initial warmup activation function was Softplus, however the code automatically found ELU activation functions for certain neurons by evaluating the statistics and fitting using the Cosine method presented in this paper.

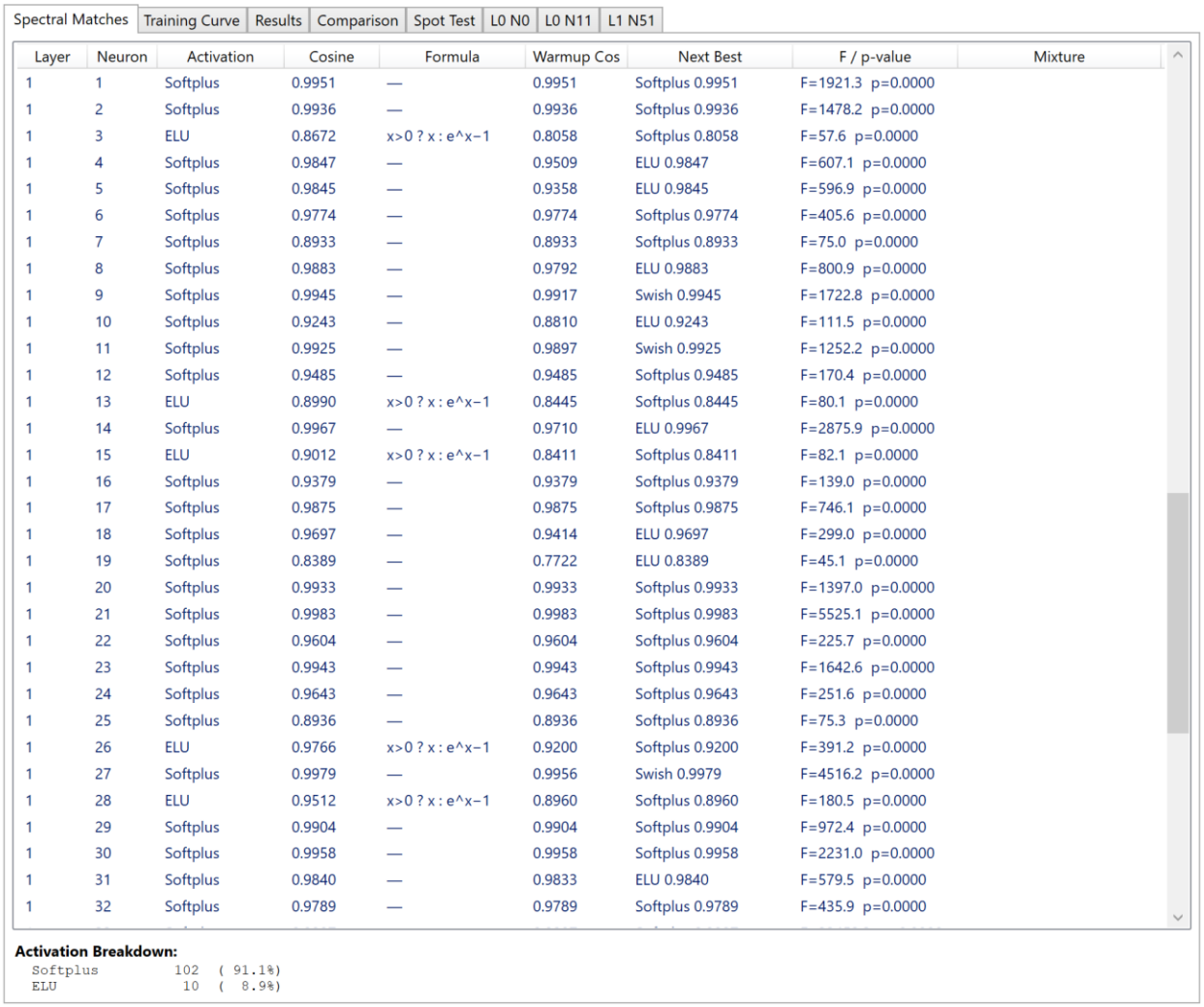


FIG. 3.1.3 Activation breakdown using sin² warmup activation function

3.2 MNIST 52 letter + 10 digit MLP Spectral Search Preliminary Results

3.2.1 MNIST 62 Character MLP Spectral Search

Presented below are the loss curves for a 62 character MLP AI/ML session. As ReLU was used, a special condition to substitute LeakyReLU for ReLU was placed in the code and in place of ReLU when the neuron was detected as being dead. ReLU was a poor function for the network selected, and in this case both the spectral search and the LeakyReLU substitution cases showed large improvement in loss function value verses Epoch.

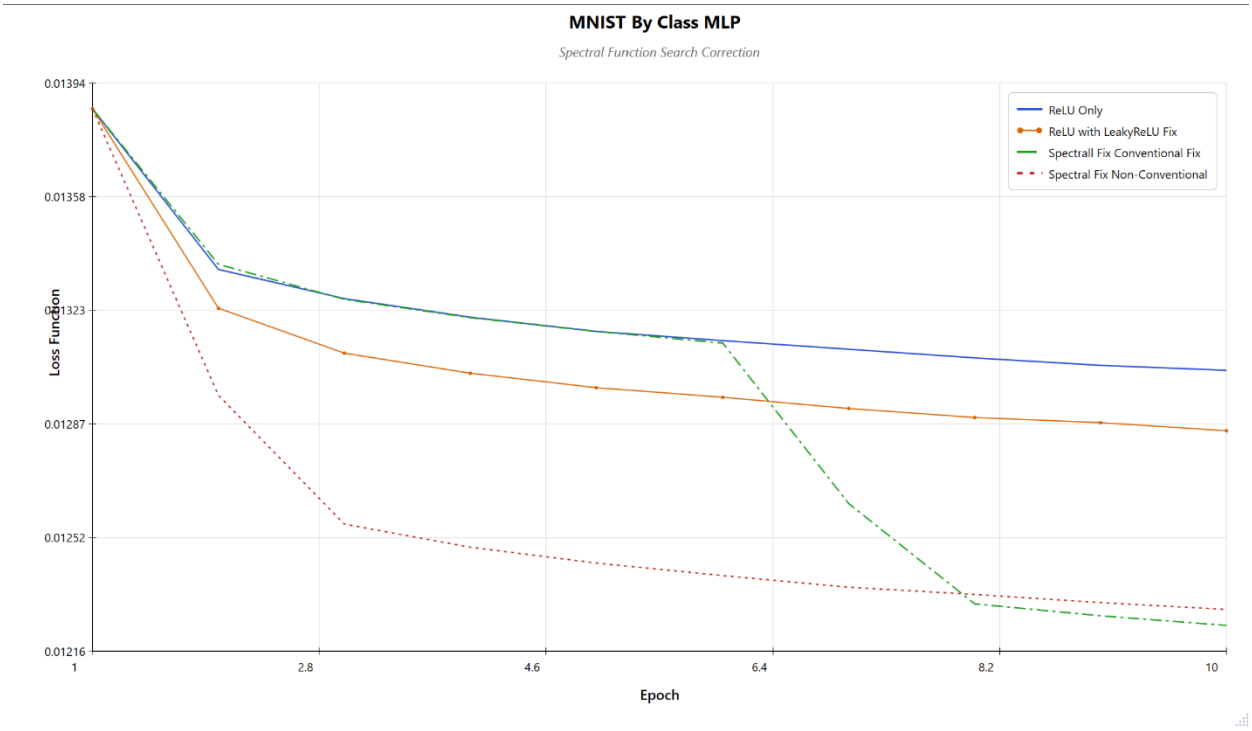


FIG. 3.2.1 MNIST 62 Character MLP Spectral Search

3.2.2 MNIST 62 Character Spot Check

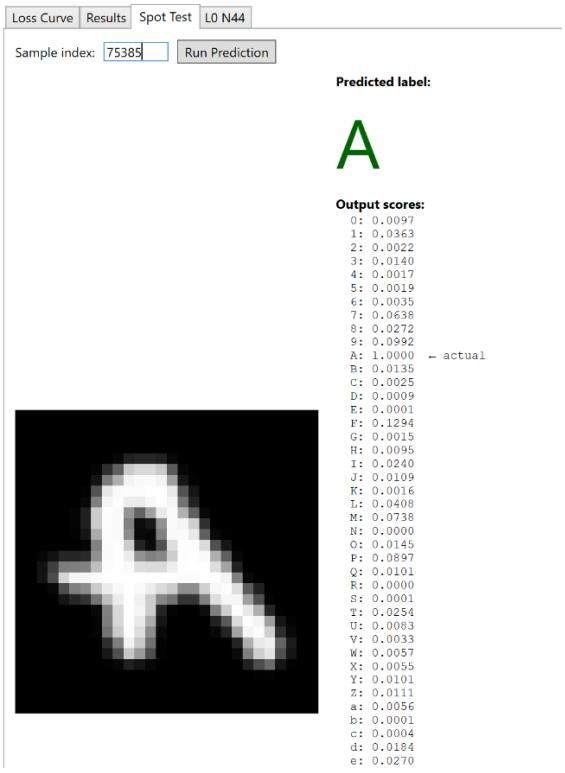


FIG. 3.2.2 MNIST 62 Character Spot Check

3.2.3 MNIST 62 Character MLP Spectral Search Linear Warmup

Presented below is a loss function comparison of a linear activation function against the same with a spectral search and improvements in selective activation functions.

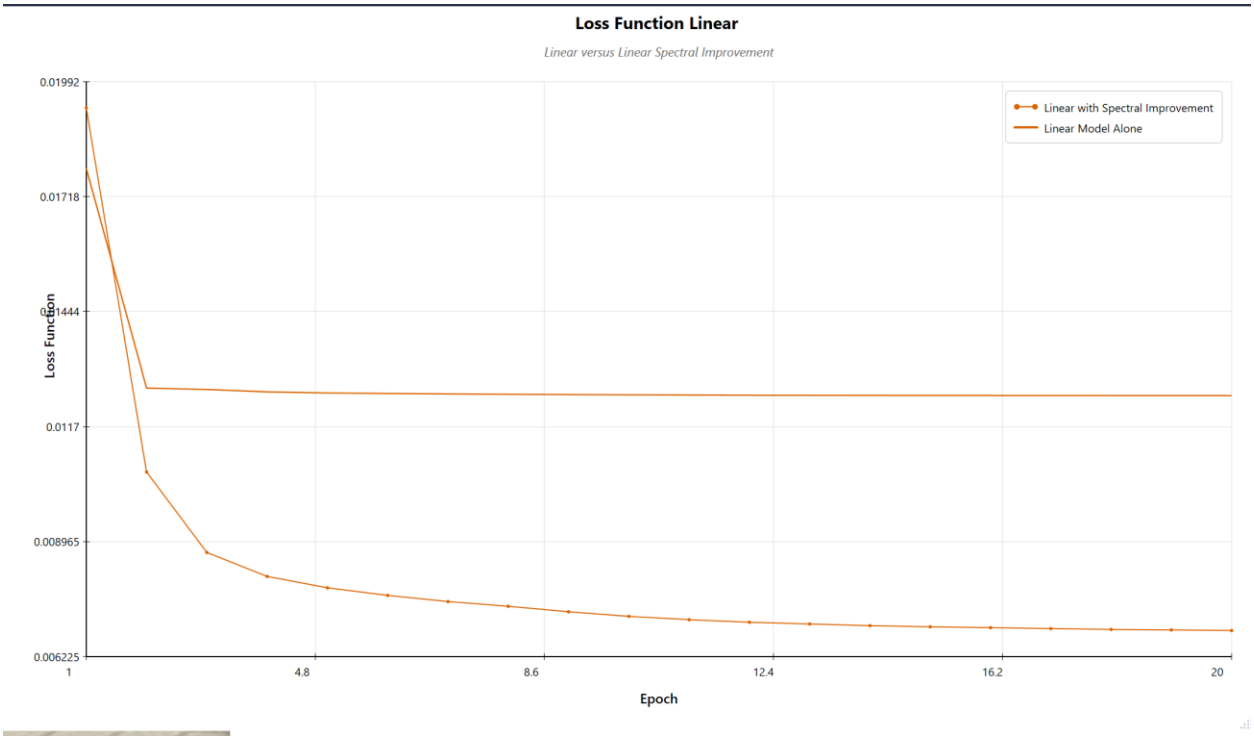


FIG. 3.2.3 MNIST 62 Character MLP Spectral Search Linear Warmup

Conclusion

Similar to electronic noise filtering and combined with the concept of spectra in physics and chemistry, fast sharp discernment of underlying generating functions is demonstrated to be viable using this spectral technique. Using the tool developed on this theory, with more discerned studies and particularly with extended and partial domains, a practical engineering methodology is available to discover generating functions.

With multi-threading, domain-specific project libraries can be rapidly generated for researcher specific domains and the applicable function sets. The current tool is for academic research and demonstration provided along with plotting tools at: **https://spectral-analytics.com/**

An AI application of this method (**Patent Pending**) is being developed to exploit this capability in the AI/ML world. This will have its own paper and test demonstration coding will be supplied in the above when ready.

Reinforcement of Acknowledgement and Statement of Researcher Bias

Intuiting use of normalized and hashed spectra was, in this instance, a human idea as was use in AI/ML for activation function discovery, selection and improvement. As patents and commercialized coding are generated from this effort the researcher has a vested financial interest in the results and conclusions.

References

[Bevington 1969]
Bevington (1969). Orthogonal Polynomials. Data Reduction and Error Analysis for the Physical Sciences. New York McGraw-Hill Book Company, Library Congress 69-16942

[Abramowitz & Stegun 1964]
Abramowitz, M., and Stegun, I. A. (1964). Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. National Bureau of Standards Applied Mathematics Series 55. Washington, D.C.: U.S. Government Printing Office. (Dover reprint 1972.)

[Bevington & Robinson 2003]
Bevington, P. R., and Robinson, D. K. (2003). Data Reduction and Error Analysis for the Physical Sciences, 3rd ed. New York: McGraw-Hill.

[Blanco et al. 1997]
Blanco, M. A., Flórez, M., and Bermejo, M. (1997). Evaluation of the rotation matrices in the basis of real spherical harmonics. Journal of Molecular Structure (Theochem), 419(1–3), 19–27.

[Charikar 2002]
Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02), pp. 380–388. New York: ACM Press.

[Dasgupta & Gupta 2003]
Dasgupta, S., and Gupta, A. (2003). An elementary proof of a theorem of Johnson and Lindenstrauss. Random Structures & Algorithms, 22(1), 60–65.

[Golub & Welsch 1969]
Golub, G. H., and Welsch, J. H. (1969). Calculation of Gauss quadrature rules. Mathematics of Computation, 23(106), 221–230.

[Indyk & Motwani 1998]
Indyk, P., and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC '98), pp. 604–613. New York: ACM Press.

[Liu & Nocedal 1989]
Liu, D. C., and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. Mathematical Programming, 45(1–3), 503–528.

[NIST DLMF]
Olver, F. W. J., Lozier, D. W., Boisvert, R. F., and Clark, C. W. (eds.) (2010). NIST Digital Library of Mathematical Functions. Release 1.2.0. National Institute of Standards and Technology. https://dlmf.nist.gov/

[Nocedal & Wright 2006]
Nocedal, J., and Wright, S. J. (2006). Numerical Optimization, 2nd ed. New York: Springer.

[Powell 1964]
Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. Computer Journal, 7(2), 155–162.

[Press et al. 2007]
Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). Numerical Recipes: The Art of Scientific Computing, 3rd ed. Cambridge: Cambridge University Press.

[Salton & Buckley 1988]
Salton, G., and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513–523.

[Szegő 1939]
Szegő, G. (1939). Orthogonal Polynomials. American Mathematical Society Colloquium Publications, Vol. 23. Providence, RI: American Mathematical Society. (4th ed. 1975.)
//SDG