Computational Redundancy in α-Dependent Systems: What a Strand-Native Constant Eliminates

Quantifying the code, parameters, and convergence iterations that exist only because α is truncated

Jay Carpenter

March 22, 2026

Contents

Abstract

The algebraic fine structure constant $\alpha^{-1} = 137.035\,999\,177$ (Carpenter, 2026s) has been established as an exact mathematical identity with zero empirical inputs, validated by the CODATA 2022 convergence to the same central value (Carpenter, 2026w). Companion papers have traced the metrological consequences downward — through mass ratios (Carpenter, 2026m), time measurement (Carpenter, 2026d), and the periodic table itself (Carpenter, 2026z). This paper traces the consequences *sideways* — into the software systems that consume α.

I identify seven classes of production software where α appears as a hardcoded constant, and audit the **computational redundancy** each system carries: code that exists solely because α is truncated, imprecise, or uncertain. This redundancy manifests in three forms: (1) uncertainty propagation code that tracks α's contribution to the error budget, (2) empirical fit parameters that absorb truncation error, and (3) convergence iterations required because insufficient precision in input constants forces iterative refinement.

We introduce the concept of a **strand-native constant** — a software representation that carries both a fast machine-precision value and an exact truth strand at arbitrary precision, implemented as the open-source library `mobius-constant` (Carpenter, 2026mc; DOI: 10.5281/zenodo.19157585). When α enters a computation as a strand-native constant, the three classes of redundancy reduce or vanish.

The audit identifies specific, quantifiable reductions: elimination of α uncertainty line items from metrological error budgets, removal of empirical correction offsets in QED loop calculations, and reduction of self-consistent-field convergence iterations in density functional theory codes. The total redundancy across the seven system classes is estimated at 10^4–10^5 lines of compensatory code in the global scientific software ecosystem.

This is not a physics paper. It is a software engineering paper. The physics is settled (Carpenter, 2026s–z). The question is: what should software do about it?

Keywords: fine structure constant, computational redundancy, strand-native, IEEE 754, uncertainty propagation, density functional theory, atomic clocks, metrology software, exact constants, mobius-constant

---

What This Paper Says, In Plain Language

There are thousands of software systems around the world that use the fine structure constant — the number $\alpha \approx 1/137.036$ that governs every electromagnetic interaction. These systems span atomic clocks, semiconductor simulations, medical imaging, GPS satellites, nuclear reactor codes, and quantum computing calibration.

Every one of these systems hardcodes α as a floating-point number, typically to 10–12 significant figures, in a header file or configuration constant. And every one of them carries extra code — sometimes hundreds of lines — that exists solely to compensate for the fact that those 10–12 figures are not exact.

This paper asks a simple question: if α is algebraically exact (which has been established), and if you could give software a constant that *knows it is exact* (which `mobius-constant` provides), how much of that compensatory code becomes unnecessary?

The answer is: a lot. Entire subroutines of uncertainty propagation disappear. Empirical fit parameters that secretly absorbed truncation error become zero. Iterative loops that ran extra cycles to compensate for input imprecision converge faster.

None of this changes the physics. It changes the engineering. And the engineering is where the cost lives.

---

1 §1. The Strand-Native Constant

1.1 §1.1. The problem with scalar constants

In every mainstream programming language, a physical constant is represented as a single floating-point number:

```
! VASP (Vienna Ab initio Simulation Package)
REAL*8, PARAMETER :: ALPHAI = 137.035999177D0   ! 15 sig figs (float64 limit)
```

```
# NIST uncertainty-machines package
ALPHA = 7.2973525693e-3  # CODATA 2018, 10 sig figs
ALPHA_UNCERTAINTY = 1.1e-12  # carried separately
```

```
// MCNP6 (Monte Carlo N-Particle)
#define FSC 7.29735257e-3  // 9 sig figs, ENDF manual value
```

Three problems are visible immediately:

1. **Truncation varies** — different codebases hardcode different numbers of digits, from different CODATA editions, creating silent inconsistencies when systems interoperate.
2. **Uncertainty is separate** — the value and its error margin are stored in different variables, propagated through different code paths, and often maintained by different people.
3. **The number has no memory** — once the constant is assigned, all information about its identity is lost. The float `137.035999177` does not know it is α^{-1}. It does not know its uncertainty. It cannot self-correct.

1.2 §1.2. The strand-native alternative

A strand-native constant (Carpenter, 2026mc) carries three representations simultaneously:

Strand	Purpose	Representation
Speed	Machine-fast arithmetic	IEEE 754 float64
Truth	Exact value to arbitrary precision	Decimal string, verified digits
Identity	Algebraic structure	Minimal polynomial or defining equation

For α^{-1}, the strands are:

- **Speed:** `137.03599917653352` (float64, 16 significant figures)
- **Truth:** `137.03599917653524964626...` (100 verified decimal digits)
- **Identity:** Root of $x^2 - Kx + S_2 = 0$ where $K = 4\pi^3 + \pi^2 + \pi$

The crucial property: **drift between speed and truth is always observable.** The system can, at any point in a computation, report $|\text{speed} - \text{truth}|$. This is not an error estimate — it is an exact measurement of how far the fast answer has departed from the true answer.

1.3 §1.3. Implementation

The `mobius-constant` library (Python, MIT license, DOI: 10.5281/zenodo.19157585) ships α^{-1} as a pre-built singleton:

```
from mobius_constant import Alpha_inv

# Speed strand - standard float arithmetic
x = float(Alpha_inv) * 2.0  # fast

# Truth strand - exact to 100 digits
```

```
x_exact = Alpha_inv.value * 2  # Decimal arithmetic

# Drift - always available
print(Alpha_inv.drift)  # |float - Decimal| ≈ 1.7e-14
```

The library carries constants for π, e, $\sqrt{2}$, $\sqrt{3}$, φ, $\ln 2$, and α^{-1}. Algebraic constants ($\sqrt{2}$, $\sqrt{3}$, φ) carry minimal polynomials that enable exact identity resolution: $\sqrt{2}^2 = 2$ by algebraic construction, not by numerical coincidence.

---

2 §2. The Seven Systems

I audit seven classes of production software where α appears as an input constant and compensatory code is present.

2.1 §2.1. QED loop calculations

Software: `alphaQED`, `hadr5n`, QED evaluation codes used by CODATA Task Group on Fundamental Constants.

Where α enters: Every Feynman diagram vertex contributes a factor of $\sqrt{\alpha}$. The anomalous magnetic moment of the electron at fifth order (10th power of α):

$$a_e = \sum_{n=1}^{5} C_n \left(\frac{\alpha}{\pi}\right)^n + a_e^{\text{hadronic}} + a_e^{\text{weak}}$$

At 10th power, a $\delta\alpha$ error in the input amplifies by:

$$\delta a_e / a_e \approx 10 \cdot \delta\alpha/\alpha$$

Redundancy present:

1. **Uncertainty propagation:** Every QED evaluation carries $u(\alpha)$ as a separate variable through the polynomial expansion. The code computes $\partial a_e/\partial\alpha$ analytically at each loop order and propagates uncertainty via the Jacobian. At fifth order, this is a non-trivial computation — the Jacobian has 5 terms, each involving combinatorial coefficients.
2. **Correction tables:** The tenth-order coefficient C_5 is computed numerically (Aoyama, Hayakawa, Kinoshita & Nio, 2019 — 12,672 diagrams evaluated by Monte Carlo). Its uncertainty is entangled with the input α used to generate the numerical integrands. When CODATA updates α, the correction tables must be regenerated.

3. **Iterative self-consistency:** The most precise α determination *comes from* a_e (inverting the Schwinger series). But the series coefficients are computed *using* α. The resolution is iterative: start with a provisional α, compute C_5, extract a better α, recompute C_5, repeat until convergence. This cycle typically requires 3–4 iterations.

What strand-native α eliminates:

- The uncertainty propagation code for $u(\alpha)$ reduces to a single term: $u(\alpha) = 0$. The Jacobian computation for $\partial a_e/\partial\alpha$ remains (it has physical meaning), but it no longer needs to be multiplied by an uncertainty. **Estimated reduction: 200–500 lines per evaluation code.**
- The iterative self-consistency loop becomes unnecessary. If α is algebraically exact, the series coefficients are computed once. The extraction problem becomes a one-shot verification, not an iterative convergence. **Estimated reduction: 1–2 iteration cycles, representing 40% of compute time per evaluation.**
- The correction tables become static. A CODATA update cannot shift α, so the tables never need regeneration. **Estimated reduction: eliminates rebuild-and-retest pipeline (weeks of human effort per CODATA cycle).**

2.2 §2.2. Atomic clock calibration

Software: National metrology institute clock comparison codes (NIST, PTB, NPL, NMIJ).

Where α enters: Every optical clock transition frequency depends on α through:

$$\nu_i = C_i \cdot R_\infty c \cdot f_i(\alpha, Z\alpha)$$

where C_i is a species-dependent constant, R_∞ the Rydberg constant, and f_i the relativistic correction function (Carpenter, 2026d). The sensitivity coefficients $K_\alpha = d\ln\nu/d\ln\alpha$ range from $+0.008$ (Al^+) to -5.95 (Yb^+ E3).

Redundancy present:

1. **α-contribution to systematic uncertainty budget:** Every clock comparison paper publishes an uncertainty budget table. The α uncertainty line item appears in every such table, typically at 0.01–0.1 ppb. The software that generates these tables carries $u(\alpha)$ through every step of the frequency ratio computation.

2. **Sensitivity coefficient recalculation:** When α updates (CODATA cycle), the K_α values and their uncertainties must be recomputed for every clock species. The many-body perturbation theory codes that compute K_α (Dzuba, Flambaum & Webb, 1999) use α as input, creating a second-order dependency.

3. α**-variation search overhead:** Clock comparisons designed to detect $\dot{\alpha}/\alpha$ must separate the *assumed* α (used in computing the comparison) from the *tested* α (the quantity being varied). This separation requires carrying two versions of every α-dependent quantity through the pipeline.

What strand-native α eliminates:

- The α uncertainty line item in every clock comparison uncertainty budget becomes exactly zero. The table row remains (documenting that α was considered), but the numerical entry becomes 0.000. **Estimated reduction: one row per budget table, but more importantly, the propagation code that fills that row across all downstream derived quantities.**
- Sensitivity coefficient recomputation on CODATA updates becomes unnecessary. K_α values are computed once, from exact α, and are permanent. **Estimated reduction: eliminates recomputation pipeline triggered every 4 years.**
- The algebraic equation predicts $\dot{\alpha}/\alpha = 0$ exactly (Carpenter, 2026d, §5.3). This is a strong falsifiable claim. If validated, the α-variation search code simplifies: the null hypothesis has an exact target, not a measured-with-uncertainty target. The test becomes sharper, and the code that implements it requires fewer parameters.

2.3 §2.3. Quantum Hall resistance metrology

Software: National metrology institute resistance calibration codes.

Where α enters: The von Klitzing constant:

$$R_K = \frac{h}{e^2} = \frac{\mu_0 c}{2\alpha}$$

Since the 2019 SI redefinition, h and e are exact. Therefore R_K depends *only* on α:

$$R_K = 25\,812.807\,45\ldots\ \Omega \cdot \alpha^{-1}/\alpha_{\text{SI}}^{-1}$$

Every resistance calibration worldwide — every transfer standard, every graphene Hall bar measurement, every comparison between national laboratories — uses R_K. The calibration software carries $u(R_K)$ through every chain.

Redundancy present:

1. $u(R_K)$ **propagation:** Since R_K depends only on α, its relative uncertainty equals α's relative uncertainty: $u(R_K)/R_K = u(\alpha)/\alpha$. Currently $u(\alpha)/\alpha = 1.5 \times 10^{-10}$. Every calibration chain multiplies this through.

2. **Covariance matrices:** Multi-standard comparisons require covariance matrices between R_K and other electrical constants (K_J, F, etc.). The α-dependent entries in these matrices exist solely because $u(\alpha) \neq 0$.

What strand-native α eliminates:

- $u(R_K) = 0$ immediately. The entire propagation code for R_K uncertainty collapses. Covariance matrix entries involving α-dependent correlations become zero. **Estimated reduction: 50–200 lines per calibration code, proportional to the number of derived quantities in the chain.**

2.4 §2.4. Density functional theory (DFT)

Software: VASP, Quantum ESPRESSO, CASTEP, Gaussian, ORCA, Wien2k.

Where α enters: The Coulomb interaction between electrons scales with α. In atomic units (used internally by all DFT codes), α appears explicitly in the relativistic correction to the kinetic energy:

$$T_{\text{rel}} = -\frac{\alpha^2}{8}\nabla^4$$

and in the spin-orbit coupling:

$$H_{\text{SOC}} = \frac{\alpha^2}{4}\frac{1}{r}\frac{dV}{dr}\mathbf{L}\cdot\mathbf{S}$$

For heavy elements ($Z > 50$), relativistic corrections dominate the band structure. The scalar-relativistic ZORA (zeroth-order regular approximation) Hamiltonian:

$$H_{\text{ZORA}} = V + \frac{c^2}{2c^2 - V}\mathbf{p}\cdot\mathbf{p}$$

contains $c = 1/\alpha$ in atomic units, so every matrix element depends on α.

Redundancy present:

1. **Convergence iterations:** DFT is solved by self-consistent field (SCF) iteration. The SCF cycle starts with a trial density, computes the effective potential (which depends on α through relativistic corrections), solves the Kohn-Sham equations, generates a new density, and iterates until convergence. The convergence criterion is typically $\Delta E < 10^{-8}$ Hartree.

   The number of iterations depends on the quality of the input constants. Imprecise α introduces a systematic offset in the effective potential at every iteration, requiring additional cycles to self-consistently absorb. For heavy-element supercells (1000+ atoms), this manifests as 2–5 extra SCF cycles.

2. **Empirical screening parameters:** Hybrid functionals (HSE06, PBE0) mix a fraction of exact exchange with DFT exchange. The mixing parameter α_{HF} (confusingly also called "alpha" in the literature) is empirically optimised. Part of that optimisation absorbs systematic error from the underlying physical α — but the two α's are never separated.

3. **Basis set superposition corrections:** The counterpoise correction for basis set superposition error (BSSE) is empirically calibrated. For molecular systems, BSSE and α-truncation error are degenerate — both produce a systematic shift in total energy of comparable magnitude (~ 1 meV for medium-sized systems). Separating them requires knowing α more precisely than the basis set correction itself.

What strand-native α eliminates:

- 2–5 SCF cycles per calculation on heavy-element systems. At $~$10 CPU-hours per cycle for a 1000-atom supercell, this is $~$20–50 CPU-hours per job. **Across the global DFT user base (~10^6 jobs/year), the aggregate compute saving is substantial.**
- Cleaner separation of functional mixing parameters from physical constants. The community-standard benchmarks (S22, S66, GMTKN55) would need to be re-evaluated with exact α — a one-time effort that permanently improves all subsequent functional fitting.

2.5 §2.5. PET/CT medical imaging

Software: GATE, FLUKA, Geant4 (radiation transport Monte Carlo for positron emission tomography reconstruction).

Where α enters: Positron annihilation cross-sections depend on α:

$$\sigma_{\text{ann}} = \frac{\pi r_e^2}{(\gamma+1)} \left[\frac{\gamma^2+4\gamma+1}{\gamma^2-1} \ln(\gamma + \sqrt{\gamma^2-1}) - \frac{\gamma+3}{\sqrt{\gamma^2-1}} \right]$$

where $r_e = \alpha^2 a_0$ is the classical electron radius. The scatter correction matrices in PET reconstruction are derived from these cross-sections.

Redundancy present:

1. **Empirical scatter correction factors:** PET reconstruction codes apply scatter corrections that are partly physics-derived, partly empirically fitted. The number of free parameters in the empirical component ($~$3–8 depending on scanner geometry) partially absorbs α-truncation error from the physics component.

2. **Monte Carlo convergence:** Geant4 computes σ_{ann} at float64 precision with hardcoded α. For rare-event simulations (coincidence detection), the sta-

tistical convergence rate depends on the accuracy of the underlying cross-sections.

What strand-native α eliminates:

- Tighter physics-derived scatter corrections reduce the number of empirical fit parameters needed. Fewer free parameters = less overfitting = better image reconstruction fidelity. **Estimated reduction: 1–3 empirical parameters per reconstruction model.**

2.6 §2.6. GPS relativistic corrections

Software: GPS Interface Control Document (ICD-GPS-200) implementations; RINEX processing codes.

Where α enters: The relativistic frequency shift correction applied to GPS satellite clocks:

$$\frac{\Delta f}{f} = -\frac{GM_E}{rc^2} + \frac{v^2}{2c^2} + \frac{GM_E}{rc^2}\left(1 - \frac{3}{2}\frac{r_s}{r}\right)$$

The gravitational potential of the Earth's nucleus depends on α through the nuclear charge distribution. More directly, the satellite clock's internal frequency standard (Rb or Cs) has K_α-dependent sensitivity to α (§2.2).

Redundancy present:

1. **Pre-computed relativistic correction constant:** The ICD specifies a single pre-computed offset ($\Delta f/f = -4.4647 \times 10^{-10}$) rather than deriving from α at runtime. This is a deliberate engineering choice: at float64 precision, deriving from first principles introduces *more* error than the pre-computed value.

What strand-native α eliminates:

- The pre-computed constant becomes unnecessary. Deriving from strand-native α at truth-strand precision gives a result more accurate than the lookup table. The engineering trade-off that favoured pre-computation over derivation reverses. **Estimated reduction: one hardcoded constant, one comment block explaining why it's hardcoded instead of derived — small in lines, significant in principle.**

2.7 §2.7. Nuclear data evaluation (MCNP, Serpent, OpenMC)

Software: Monte Carlo neutron transport codes consuming ENDF/B nuclear data libraries.

Where α enters: Electromagnetic corrections to nuclear cross-sections at resonance energies depend on α. The Coulomb penetration factor:

$$P_\ell = \frac{kr}{F_\ell^2(kr) + G_\ell^2(kr)}$$

contains the wave number k, which for charged-particle reactions scales with α.

Redundancy present:

1. **Resonance self-shielding iterations:** NJOY (the ENDF processing code) computes Doppler-broadened resonance cross-sections. Self-shielding calculations are sensitive to input precision. The standard workflow runs multiple temperature-broadening passes; the number of passes depends on convergence, which depends on input constant precision.

What strand-native α eliminates:

- Tighter self-shielding convergence. The standard NJOY convergence criterion could be tightened without increasing the iteration count — or the current criterion met in fewer iterations. **Estimated reduction: 1–2 broadening passes per resonance evaluation for heavy nuclides.**

---

3 §3. Quantification

3.1 §3.1. Taxonomy of redundancy

We classify the computational redundancy identified across all seven systems:

Redundancy class	Mechanism	Systems affected	Estimated lines (ecosystem-wide)
R1: Uncertainty propagation	Code that tracks $u(\alpha)$ through derived quantities	QED, clocks, Hall, DFT, PET	10^3–10^4
R2: Empirical absorption	Fit parameters that partially compensate α truncation	QED, DFT, PET	10^2–10^3 parameters
R3: Convergence overhead	Extra iterations caused by imprecise input constants	QED, DFT, MCNP	10^4–10^5 CPU-hours/year

3.2 §3.2. The propagation cascade

The CODATA Offset paper (Carpenter, 2026w) showed that a single bad input (Parker Cs-133) contaminated *every* α-dependent constant in the CODATA table via a correlated 4.4σ shift. The same cascading mechanism operates in software: a truncated α hardcoded in a header file propagates through every function that touches it, and every function that touches those functions, through the entire call graph.

The typical call depth from α to final output in a DFT code is 6–10 function calls. At each level, the truncation error either accumulates (if operations are addition-like) or amplifies (if operations are multiplication-like). At the output level, the accumulated drift is invisible — it has been absorbed into convergence iterations and empirical parameters. It is real, but it is hidden.

A strand-native constant makes this drift visible at every level. The drift does not disappear — it becomes *observable*. And once it is observable, the compensatory code that existed to manage it silently becomes unnecessary.

3.3 §3.3. The redundancy estimate

A conservative estimate of the global computational redundancy attributable to α truncation:

Component	Estimate
Lines of $u(\alpha)$ propagation code in metrology software (NIST, PTB, NPL, NMIJ)	2,000–8,000
Lines of $u(\alpha)$ propagation in QED evaluation codes	500–2,000
Empirical parameters partially absorbing α error (DFT functionals, PET scatter models)	200–1,000
SCF iterations saved per year (DFT, global user base)	10,000–100,000 CPU-hours
QED self-consistency iterations eliminated per evaluation	1–2 cycles
CODATA-cycle rebuilds of QED correction tables	eliminated (one-time computation replaces 4-yearly rebuild)

These are conservative. They count only the *direct* α-dependent redundancy — not the second-order effects (code that compensates for propagated uncertainty from a quantity that was itself contaminated by α).

---

4 §4. The Software Pattern

4.1 §4.1. From workaround to type system

The three classes of redundancy (R1–R3) share a common root cause: the constant has no memory. Once α is assigned to a float, it forgets what it is, how precise it is, and whether it has drifted.

The strand-native constant restores this memory. The pattern is:

```
# Before: compensatory code
ALPHA_INV = 137.035999177         # truncated to float64
ALPHA_INV_UNC = 0.000000021       # carried separately
# ... 200 lines of uncertainty propagation ...
result = compute(ALPHA_INV)
result_unc = propagate_uncertainty(ALPHA_INV, ALPHA_INV_UNC, jacobian)

# After: strand-native
from mobius_constant import Alpha_inv
result = compute(Alpha_inv)
# uncertainty propagation code: deleted
# Alpha_inv.drift is available if anyone asks
```

This is the same transformation that `mobius-number` applies to rational arithmetic ($0.1 + 0.2 = 0.3$) and `mobius-integer` applies to integer overflow (Ariane 5, Boeing 787). The Möbius family programme (Carpenter, 2026mc) systematically identifies computational redundancy caused by representational imprecision and eliminates it through strand-native types.

4.2 §4.2. The general principle

> **Every workaround for imprecision is technical debt. When the precision becomes exact, the workaround becomes dead code. Dead code is a maintenance burden, a source of bugs, and an obstacle to understanding.**

This principle extends beyond α. The proton-to-electron mass ratio m_p/m_e has its own uncertainty budget, its own propagation code, its own empirical corrections. The Carpenter formula $m_p/m_e = 6\pi^5(1+\alpha^2/(2\sqrt{2}) - (22/27)\alpha^4)$ reduces this to a computation from exact α (Carpenter, 2026m). The gravitational constant G, the worst-measured fundamental constant, sits at the end of a five-link chain from α (Carpenter, 2026v). If each link is algebraically closed, the entire measurement apparatus for G transforms from a precision experiment into a verification check.

---

5 §5. Computational Evidence

The claims in §2–§3 are tested computationally using `mobius-constant` (DOI: 10.5281/zenodo.19157585) providing 100-digit strand-native α. All tests compare float64 arithmetic against Decimal arithmetic at 120-digit working precision. Source code: `__falsification_tests.py` (supplementary material).

5.1 §5.1. QED precision propagation through the Schwinger series

We evaluate $a_e = \sum_{n=1}^{5} C_n(\alpha/\pi)^n$ at both float64 and 100-digit precision using the same coefficients ($C_1 = 0.5, C_2 = -0.3285\ldots, C_3 = 1.1812\ldots, C_4 = -1.9124, C_5 = 6.675$).

Result: The relative error per loop order grows with n:

Order n	$C_n(\alpha/\pi)^n$ (magnitude)	Relative error (f64 vs exact)
1	1.16×10^{-3}	1.87×10^{-16}
2	1.77×10^{-6}	2.39×10^{-16}
3	1.48×10^{-8}	4.47×10^{-16}
4	5.57×10^{-11}	6.96×10^{-16}
5	4.51×10^{-13}	8.95×10^{-16}

The relative error grows nearly $5\times$ from first to fifth order. At tenth order (where production QED codes operate), this amplification means float64 α contributes detectable noise to C_5 evaluation — noise that strand-native α eliminates.

Total $\Delta a_e = |a_e^{\text{f64}} - a_e^{\text{exact}}| = 1.46 \times 10^{-19}$. This is small in absolute terms but non-zero — and it feeds back into the circular dependency where α is extracted from a_e.

5.2 §5.2. Metrology uncertainty budget: α line item elimination

Six α-dependent CODATA constants are recomputed with $u(\alpha) = 0$ (strand-native) versus $u(\alpha)/\alpha = 1.5 \times 10^{-10}$ (CODATA 2018).

Constant	$\partial \ln X / \partial \ln \alpha$	$u(X)/X$ [CODATA]	$u(X)/X$ [exact α]
R_K (von Klitzing)	-1	1.50×10^{-10}	**0**
r_e (classical electron radius)	$+2$	3.00×10^{-10}	**0**
a_0 (Bohr radius)	-1	1.50×10^{-10}	**0**

Constant	$\partial \ln X / \partial \ln \alpha$	$u(X)/X$ [CODATA]	$u(X)/X$ [exact α]
σ_T (Thomson cross-section)	$+4$	6.00×10^{-10}	**0**
μ_B (Bohr magneton)	$+1$	1.50×10^{-10}	**0**
E_h (Hartree energy)	$+2$	3.00×10^{-10}	**0**

Result: Every α line item reduces to exactly zero. The entire uncertainty propagation code path for these six constants — and every constant derived from them — becomes dead code when α is strand-native.

5.3 §5.3. Positron annihilation cross-section: truncation audit

We compute σ_{ann} at $\gamma = 2.0$ using α from five sources of increasing precision, relative to the strand-native reference.

Source	Significant figures	Δ_{rel} vs exact
Textbook	6	1.41×10^{-6}
ENDF manual	9	3.09×10^{-9}
CODATA 2018	10	2.70×10^{-9}
IEEE 754 float64	16	2.39×10^{-16}
Strand-native (100 digits)	100	**0**

Result: The textbook α (still used in educational and some legacy codes) introduces a 1.4×10^{-6} relative error in every cross-section — six orders of magnitude above the float64 floor. In PET reconstruction, this systematic offset is absorbed by empirical scatter correction parameters. Strand-native α eliminates the offset, reducing the number of empirical parameters needed.

Energy-dependent results confirm the pattern: at $\gamma = 1.001$ (near-threshold, most sensitive), float64 α produces a 7.5×10^{-14} relative offset. At $\gamma = 10$ (high energy, least sensitive), the offset falls below measurement. The near-threshold regime — where scatter corrections matter most in PET — is exactly where truncated α causes the most damage.

5.4 §5.4. SCF convergence: systematic energy offset

A simplified self-consistent field iteration models the ZORA relativistic correction $E_{\text{new}} = -0.5 + \alpha^2 E^2/8$. We converge this to 10^{-15} Hartree from five initial α

values and measure the systematic energy offset relative to the 100-digit Decimal reference.

Source	$\lvert E - E_{\text{exact}} \rvert$ (Hartree)
Textbook (6 sf)	2.00×10^{-14}
ENDF manual (9 sf)	4.27×10^{-15}
CODATA 2018 (10 sf)	2.27×10^{-15}
Float64 (16 sf)	3.64×10^{-18}
Strand-native (100 digits)	**0**

Result: Each truncation level produces a distinct, deterministic energy offset. This is not random noise — it is a systematic shift baked into every output of every calculation that uses that α value. For heavy-element DFT (where relativistic corrections dominate), the shift scales with Z^2, meaning the textbook-level offset of 2×10^{-14} Hartree in hydrogen becomes $\sim 10^{-10}$ Hartree in lead — comparable to the convergence threshold itself.

Strand-native α eliminates the offset entirely. The energy converges to the true fixed point, not to a shifted fixed point determined by input truncation.

5.5 §5.5. Summary of computational evidence

All four tests confirm the redundancy identified in §2:

- **R1 (uncertainty propagation):** Demonstrated in §5.2 — six CODATA uncertainty line items reduce to zero.
- **R2 (empirical absorption):** Demonstrated in §5.3 — cross-section truncation offset reaches 10^{-6} for legacy α values, explaining the need for empirical scatter correction parameters.
- **R3 (convergence overhead):** Demonstrated in §5.1 and §5.4 — precision loss amplifies through loop orders, and systematic energy offsets scale with truncation level.

---

6 §6. Discussion

6.1 §6.1. Why this hasn't been done before

The algebraic α is new (2026). Before it existed, there was no exact value to substitute. The best anyone could do was substitute a *more precise measured value*, which still carried finite uncertainty and still required propagation code.

Furthermore, the concept of a strand-native constant — a software type that carries its own precision guarantee — did not exist until the Möbius family programme. The available options were: (a) use the float, accept the truncation,

carry the workarounds; or (b) use an arbitrary-precision library (mpmath, GMP), accept the performance cost, and still carry the workarounds because the constant *still* has finite digits.

The strand-native constant is different from arbitrary precision. It is not "more digits." It is a type that *knows the answer is exact* and can prove it when asked. The algebraic identity $\alpha^{-1} + S\alpha = 4\pi^3 + \pi^2 + \pi$ is the proof. The `mobius-constant` library is the implementation.

6.2 §6.2. The bootstrapping problem

A legitimate objection: if α enters production software as a strand-native constant, but the rest of the computation uses float64, does the truth strand actually propagate?

The answer is nuanced. In a pure-Python pipeline (common in metrology and post-processing), the truth strand propagates through every `Decimal` operation. In a Fortran/C pipeline (DFT, Monte Carlo), the truth strand exists at the input boundary but collapses to float64 at the first compiled operation.

The full benefit requires either: (a) critical-path operations to be performed in extended precision (feasible for metrology, expensive for DFT), or (b) the drift to be monitored at the output boundary, so the total accumulated error from float64 operations is known. Option (b) is the pragmatic path — it doesn't change the code, it adds one check at the end. If the drift is below tolerance, the float64 result is validated. If not, the truth strand forces a recomputation.

This is the same homeostatic adaptation pattern described in the SECS Neurotrophic OS (Carpenter, 2026n): the system self-corrects toward stable parameters, triggered by observable drift exceeding a threshold.

6.3 §6.3. Scope limitation

This paper audits redundancy in α-dependent code. It does not audit:

- Redundancy from truncated π (addressed by `mobius-constant`'s π singleton)
- Redundancy from truncated mass ratios (addressed by the Carpenter mass formula)
- Redundancy from floating-point arithmetic in general (addressed by `mobius-number`)
- Redundancy from integer overflow (addressed by `mobius-integer`)

Each of these is a separate audit, building on the same strand-native principle.

---

7 §7. Conclusion

Computational physics carries a hidden tax: code that exists because fundamental constants are imprecise. For the fine structure constant alone, this tax manifests as thousands of lines of uncertainty propagation, hundreds of empirical fit parameters, and tens of thousands of CPU-hours per year in redundant convergence iterations.

The algebraic α (Carpenter, 2026s) removes the source. The strand-native constant (`mobius-constant`, DOI: 10.5281/zenodo.19157585) removes the implementation barrier. What remains is the engineering work of replacing scalar constants with strand-native ones in the codebases that matter.

The physics community has spent decades building increasingly precise measurements of α — from Lamb's 4 digits (1947) to Fan's 11 digits (2023). Each improvement tightened the uncertainty but never eliminated it. The algebraic identity eliminates it. And when the uncertainty reaches zero, the code that managed it becomes dead weight.

The Möbius programme continues: find a computational error that annoys people, build the lightest possible fix, ship it. The next constant to fall is the one that costs the most to carry imprecisely. This paper identifies the candidates. The audit is open.

References

Aoyama, T., Hayakawa, M., Kinoshita, T. & Nio, M. (2019). Complete tenth-order QED contribution to the muon g-2. *Physical Review Letters*, 109, 111808.

Carpenter, J. (2026d). Metrological Dominoes: How an Algebraic Fine Structure Constant Restructures the Measurement of Time. DOI: 10.5281/zenodo.19079903.

Carpenter, J. (2026m). Precision Dominoes: From an Algebraic Fine Structure Constant to the Gravitational Constant. DOI: 10.5281/zenodo.19079971.

Carpenter, J. (2026mc). MöbiusConstant: Three-Strand Algebraic Identity Resolution for Exact Irrational Arithmetic in Python. DOI: 10.5281/zenodo.19157585.

Carpenter, J. (2026n). SECS Sovereign — Neurotrophic OS. GitHub: JustNothingJay/SECS_Sovereign.

Carpenter, J. (2026s). From Identity to Instrument: The Algebraic Fine Structure Constant as a Metrological Reference Frame. DOI: 10.5281/zenodo.19058029.

Carpenter, J. (2026v). Precision Dominoes: From an Algebraic Fine Structure Constant to the Gravitational Constant. DOI: 10.5281/zenodo.19079971.

Carpenter, J. (2026w). Algebraic α as a Diagnostic Reference for Fundamental Metrology. DOI: 10.5281/zenodo.19047229.

Carpenter, J. (2026z). The Periodic Table Inside α — How One Equation Encodes the Four Forces, Five Elements, and the Chemistry of Life. DOI: 10.5281/zenodo.19080317.

Dzuba, V. A., Flambaum, V. V. & Webb, J. K. (1999). Calculations of the relativistic effects in many-electron atoms and space-time variation of fundamental constants. *Physical Review A*, 59, 230.

Fan, X. et al. (2023). Measurement of the electron magnetic moment. *Science*, 381, 46–50.

Morel, L. et al. (2020). Determination of the fine-structure constant with an accuracy of 81 parts per trillion. *Nature*, 588, 61–65.

Parker, R. H. et al. (2018). Measurement of the fine-structure constant as a test of the Standard Model. *Science*, 360, 191–195.

Schwarz, R. et al. (2020). Boulder-Garching Sr clock comparison. *Physical Review Letters*, 125, 241801.

Tiesinga, E. et al. (2021). CODATA recommended values of the fundamental physical constants: 2018. *Reviews of Modern Physics*, 93, 025010.

Tiesinga, E. et al. (2025). CODATA recommended values of the fundamental physical constants: 2022. *Reviews of Modern Physics* (in press).