The Genetic Code as Executable Mathematics

From a 64-Entry Lookup Table to Exact Algebra, Graphs, and Exhaustive Search in Python

Douglas C. Youvan

Public manuscript edition · August 2026

Executable companion repository: https://github.com/DougYouvan/mathematics-of-the-genetic-code-in-a-python-program

Copyright and Software Licensing

Repository: https://github.com/DougYouvan/mathematics-of-the-genetic-code-in-a-python-program

The public repository release associated with this manuscript is v1.1.1. The source-tree repository and release ZIP are the canonical reproducibility units for the computation described here.

Contents

List of Figures

List of Tables

Preface

The genetic code is usually introduced as a biological table: sixty-four codons are translated into twenty amino acids and stop. This book asks what happens when that table is treated first as a complete finite computational object. The starting point is deliberately modest. Load all sixty-four RNA triplets. Verify that none is missing. Store translation as a deterministic lookup function. Then construct mathematical representations directly from that complete table and make every important claim executable.

The program moves through several representations of the same finite object. Codons become rows of a 64×12 positional one-hot matrix. The matrix yields an exact rank, singular spectrum, Moore–Penrose pseudoinverse, and codon-space projector. Amino-acid hydropathy becomes a numerical field on codon space and can be projected back onto nucleotide-position coordinates. Codons then become graph vertices, local nucleotide changes become edges, synonymous equivalence becomes a quotient multigraph, and every possible nucleotide path can be enumerated rather than sampled.

The aim is not to replace biology with algebra. It is to separate claims that can be proved exactly from claims that depend on empirical property values, sequence examples, graph conventions, or future null models. Exact finite results should be exact. Numerical calculations should be cross-checked against closed forms when possible. Figures should be generated from machine-readable objects. A release should fail rather than quietly promote artifacts when required invariants do not hold.

Python is therefore not incidental to the presentation. It is a second language of the book. The repository is intended to allow a reader to move from prose to code to generated artifact and back again. The mathematical object is small enough that exhaustive verification is possible, and that makes the genetic code an unusually clear case study in executable mathematics.

Notation and Conventions

- RNA notation is used throughout: G, A, C, U. T is not used as a codon symbol in the canonical lookup table.
- Translation stop is represented by X rather than an asterisk.
- D denotes the complete 64 × 12 positional one-hot incidence matrix. D^T is its transpose and D^+ its Moore–Penrose pseudoinverse.
- J denotes an all-ones matrix whose dimensions are clear from context.
- $P = DD^+$ denotes the orthogonal projector from arbitrary 64-component codon fields onto the additive nucleotide-position subspace.
- Hamming distance d(a,b) is the number of codon positions at which codons a and b differ.
- For the generalized complete word system, a denotes alphabet size and w denotes word length. The genetic code is $a = 4, w = 3$.
- $H(X)=0$ is an explicit computational convention used only when a complete 64-component amino-acid hydropathy field is required; it is not a physical assignment of hydropathy to stop.
- An isotropic codon lattice uses the same nucleotide path on all three codon axes. An anisotropic lattice permits a different path on each axis.
- Unless explicitly stated otherwise, graph edges are undirected and nucleotide order defines a linear path, not a four-state cycle.

Part I
The Lookup Table

Chapter 1

The Standard Genetic Code

1.1 Four Symbols and Three Positions

The input alphabet contains four RNA symbols: G, A, C, and U. A codon contains three ordered positions. The complete word space therefore contains $4^3 = 64$ possible codons. Because all sixty-four words exist in the input domain, there is no statistical sampling problem at this level: the program can enumerate the complete domain directly.

```
alphabet = ('G', 'A', 'C', 'U')

codons = [
   a + b + c
   for a in alphabet
   for b in alphabet
   for c in alphabet
]

assert len(codons) == 64
```

1.2 Complete Set of Sixty-Four Codons

The first responsibility of the program is not analysis but completeness. Every length-three word over the RNA alphabet must appear exactly once in the canonical dataset. Duplicate codons, missing codons, invalid letters, or unexpected word lengths are release-blocking errors because every later algebraic identity assumes the complete Cartesian product.

1.3 Translation Function

The standard genetic code is represented as a deterministic lookup function τ from the 64-codon set C to an output set Ω containing the twenty amino-acid single-letter symbols and X. In programming terms, τ is a dictionary whose keys are codons and whose values are translated output symbols. In mathematical terms, it is a function $\tau: C \rightarrow \Omega$.

```
translation = {
   'UUU': 'F',
   'UUC': 'F',
   # ... all remaining codons ...
}

output = translation['GAC']
```

1.4 Twenty Amino Acids and X

The image of the lookup function contains twenty-one output classes. Twenty correspond to standard amino acids. The twenty-first, X, represents translation stop in this repository. The notation is deliberately uniform: every output is a one-character symbol, which simplifies tables, matrices, graph labels, and machine-readable artifacts.

1.5 Loading the Genetic Code in Python

The canonical table is stored as data rather than hard-coded across analytical functions. This makes the source inspectable, hashable, independently citable, and replaceable when alternative complete genetic codes are studied. The loader validates the schema and normalizes the table into the canonical codon order used throughout the package.

1.6 Validation of Complete Input Space

Validation checks include the number of rows, uniqueness of codon keys, RNA alphabet membership, codon length, output symbol format, and equality between the observed key set and the generated set of all 64 RNA triplets. A program should stop at this boundary rather than allow malformed input to propagate into apparently sophisticated calculations.

1.7 Translation as Deterministic Lookup

Once the table is validated, translation is computationally trivial: a codon key returns exactly one output symbol. The interest comes from treating that finite function as an object whose fibers, matrices, graphs, projections, and exhaustive comparison spaces can be constructed without inserting those structures by hand.

1.8 Synonymous Codons

Two codons are synonymous when they share the same translated output. This defines an equivalence relation on the input space. The equivalence classes are the inverse images, or fibers, of τ. Synonymy is therefore not an additional annotation imposed on the table; it is induced directly by equality of lookup values.

1.9 Output Degeneracy

The cardinality of a fiber is its degeneracy. Some outputs have six codons, some four, some three or two, and two amino acids have only one codon. Degeneracy describes how many inputs share an output, but it does not tell us how those inputs are arranged in codon space. That distinction becomes central once the same lookup table is viewed as a graph.

Codon	Output	Codon	Output	Codon	Output	Codon	Output
GGG	G	AGG	R	CGG	R	UGG	W
GGA	G	AGA	R	CGA	R	UGA	X
GGC	G	AGC	S	CGC	R	UGC	C
GGU	G	AGU	S	CGU	R	UGU	C
GAG	E	AAG	K	CAG	Q	UAG	X
GAA	E	AAA	K	CAA	Q	UAA	X
GAC	D	AAC	N	CAC	H	UAC	Y
GAU	D	AAU	N	CAU	H	UAU	Y
GCG	A	ACG	T	CCG	P	UCG	S
GCA	A	ACA	T	CCA	P	UCA	S
GCC	A	ACC	T	CCC	P	UCC	S
GCU	A	ACU	T	CCU	P	UCU	S
GUG	V	AUG	M	CUG	L	UUG	L
GUA	V	AUA	I	CUA	L	UUA	L
GUC	V	AUC	I	CUC	L	UUC	F
GUU	V	AUU	I	CUU	L	UUU	F

Table 1. Complete standard RNA lookup table

1.10 Degeneracy Distribution

Counting the sizes of all twenty-one fibers produces the exact output-class degeneracy distribution. Because the input table is complete, the counts must sum to sixty-four. The distribution is a useful first invariant and a simple regression target for the verifier: any change in the canonical lookup table immediately changes at least one fiber or its membership.

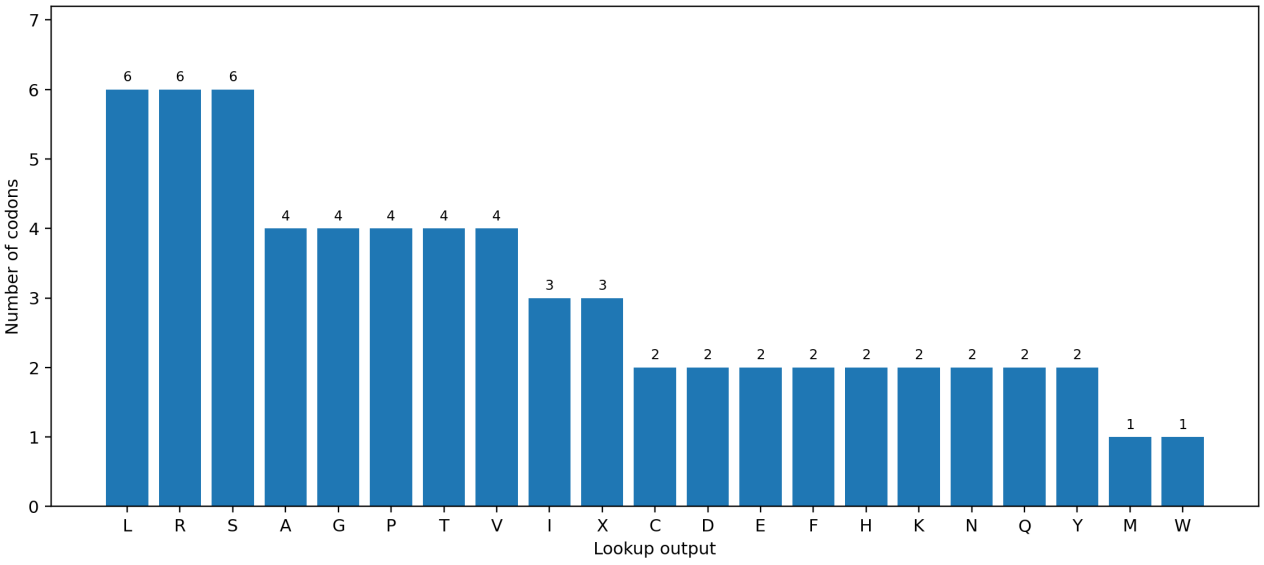


Figure 1. Output-class degeneracy histogram

Chapter 2

Codon Fibers

2.1 Inverse Images

For an output symbol ω, its fiber is $F\omega = \{c \in C : \tau(c) = \omega\}$. The program builds these inverse images by traversing the lookup table once and collecting codons by output class. The resulting mapping from output symbol to codon list is a second representation of the same genetic code.

2.2 Membership

Fiber membership is exact categorical information. A codon belongs to one and only one translated fiber. Distinct fibers are disjoint, and their union is the complete set of sixty-four codons. These properties provide simple executable partition checks.

2.3 Fiber Cardinality

Cardinality alone already separates output classes into degeneracy groups, but the cardinality is only a first statistic. A fiber of size six contains fifteen possible unordered internal codon pairs; a fiber of size four contains six. Whether those pairs are close or far in codon space remains unspecified until a geometry or adjacency relation is declared.

2.4 Six-Codon Classes

Arginine, leucine, and serine are six-codon classes in the standard code. Their equal cardinality will later provide a clean demonstration that degeneracy does not determine graph connectivity: R and L acquire six self-loop instances in the G–A–C–U quotient, whereas S acquires four.

2.5 Four-Codon Classes

Several amino-acid fibers contain four codons. Under the G–A–C–U path, five four-codon classes—A, G, P, T, and V—each contribute three synonymous local edges. A four-element fiber has six possible internal pairs, so the graph selects only half of them as local in these cases.

2.6 Three-, Two-, and One-Codon Classes

Isoleucine and X form three-codon classes in the complete lookup representation used here; each contributes two local internal adjacencies in the later quotient. Nine two-codon classes contribute one adjacency each. Methionine and tryptophan are singletons and therefore cannot contain an internal codon pair at all.

2.7 Degeneracy Versus Arrangement

The lookup table therefore contains two layers of information that should not be conflated. Degeneracy describes the size of an equivalence class. Arrangement describes how members of that class occupy a chosen codon geometry. The latter requires an explicit adjacency model and cannot be inferred from frequency alone.

2.8 Why Frequency Does Not Determine Geometry

Two fibers of equal size can be distributed differently across the input space. Equal degeneracy therefore places only an upper bound on the possible number of local internal edges. This observation motivates the later quotient multigraph, where synonymous edges become self-loops and fiber compactness becomes directly visible.

Output	Degeneracy	Codons
L	6	CUG CUA CUC CUU UUG UUA
R	6	AGG AGA CGG CGA CGC CGU
S	6	AGC AGU UCG UCA UCC UCU
A	4	GCG GCA GCC GCU
G	4	GGG GGA GGC GGU
P	4	CCG CCA CCC CCU
T	4	ACG ACA ACC ACU
V	4	GUG GUA GUC GUU
I	3	AUA AUC AUU
X	3	UGA UAG UAA
C	2	UGC UGU
D	2	GAC GAU
E	2	GAG GAA
F	2	UUC UUU
H	2	CAC CAU
K	2	AAG AAA
N	2	AAC AAU
Q	2	CAG CAA
Y	2	UAC UAU
M	1	AUG
W	1	UGG

Table 2. Output classes, degeneracies, and codon membership

2.9 Alternative Assignments

The same input space can support other complete lookup functions. A synthetic or alternative code can preserve degeneracy while changing fiber membership, or preserve some codon-box structure while changing output labels. Such alternative assignments become useful only after the comparison space is defined explicitly.

2.10 Need for Explicit Comparison Spaces

There is no unique object called a random genetic code. Any comparative claim must say what is preserved, what is randomized, whether the ensemble is exhaustively enumerable or sampled, and which statistic is evaluated. This principle will return in Chapter 19.

Part II
The Complete Codon Matrix

Chapter 3

One-Hot Encoding

3.1 Twelve Coordinates

Each of the three codon positions receives four indicator coordinates. The twelve columns are ordered as 1G, 1A, 1C, 1U; 2G, 2A, 2C, 2U; 3G, 3A, 3C, 3U. A coordinate asks a single yes-or-no question: does this codon contain this nucleotide at this position?

3.2 Encoding a Single Codon

A codon activates exactly one coordinate in each positional block. GAC, for example, activates 1G, 2A, and 3C. Its binary row therefore contains three ones and nine zeros. This is a positional encoding of the input string; no amino-acid information is needed to construct it.

3.3 Construction of the Complete Matrix

```
def encode_codon(codon, alphabet=('G','A','C','U')):
  row = []
  for base in codon:
    row.extend(int(base == symbol) for symbol in alphabet)
  return row

D = np.array([encode_codon(c) for c in codons], dtype=int)
```

Stacking the sixty-four codon rows yields the complete incidence matrix D. Because the input space is the full Cartesian product, the matrix is perfectly balanced and admits exact combinatorial formulas.

3.4 Sixty-Four Rows and Twelve Columns

The matrix has shape 64 × 12. It contains 768 binary entries, but only 192 are ones because each of the sixty-four rows contributes exactly three active coordinates.

3.5 Three Ones per Row

Every codon has exactly one nucleotide in each of three positions, hence every row sum is three. This is both a trivial biological fact and a useful program invariant. A row sum other than three indicates a broken encoder.

3.6 Sixteen Ones per Column

Fix one nucleotide at one position. The other two positions remain free, giving $4^2=16$ codons. Therefore every column sum is sixteen. This exact balance drives the simple Gram matrix and pseudoinverse that follow.

3.7 Complete Factorial Balance

Within one positional block, distinct nucleotide columns never overlap. Across different positional blocks, any fixed pair of nucleotide indicators overlaps in exactly four codons because the third position remains free. The matrix is therefore a complete balanced factorial design written in codon notation.

3.8 Translation as an Adjacent Output Vector

The matrix and the translation table can be displayed side by side without mixing their meanings. D represents codon composition. An adjacent 64-component output strip records translation. The visual juxtaposition makes it easy to see that a simple input geometry and a many-to-one output assignment coexist on the same ordered codon list.

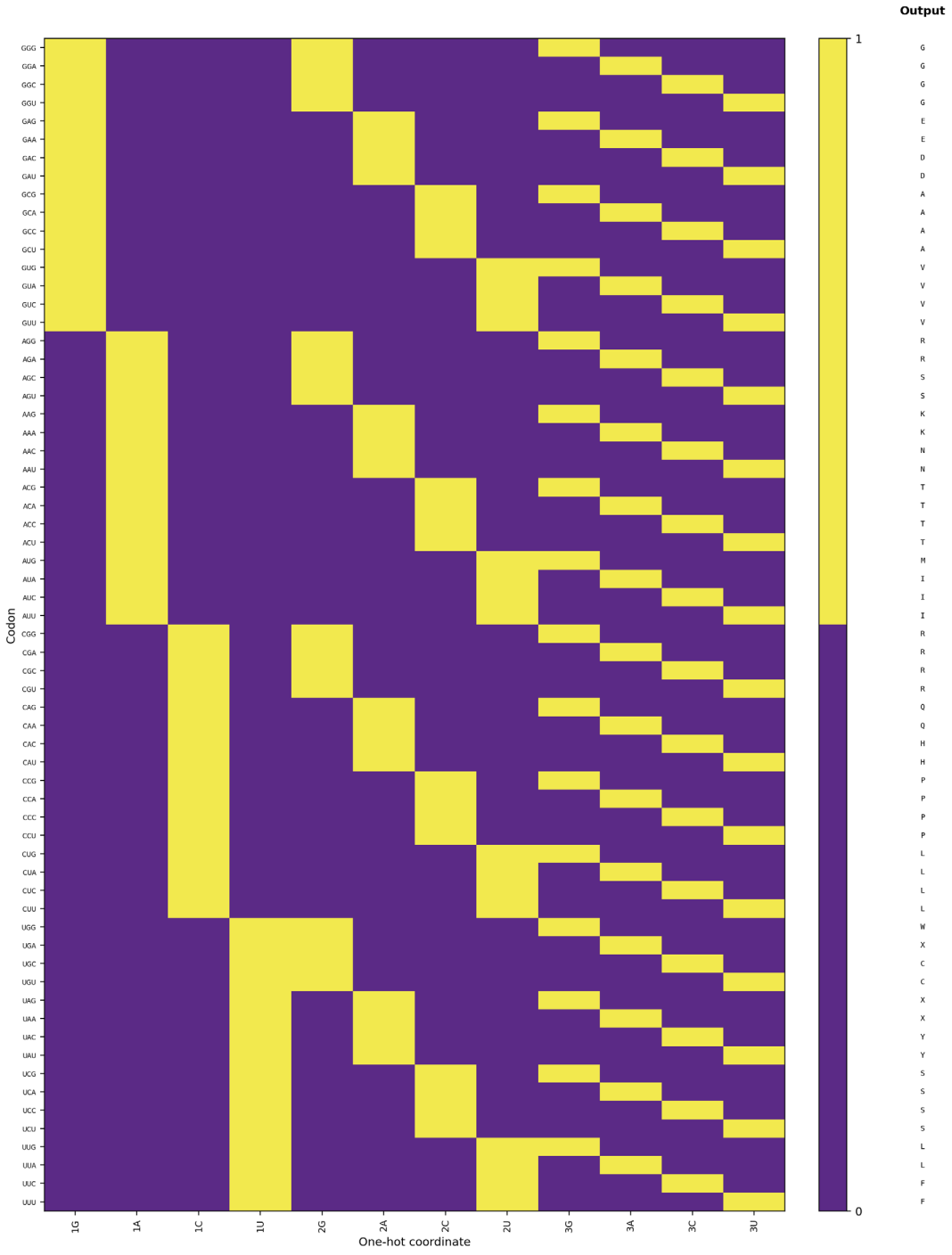


Figure 2. 64 × 12 binary incidence matrix with translated output strip

Chapter 4

Rank, Nullity, and Singular Spectrum

4.1 The Gram Matrix

The 12×12 Gram matrix $G=D^TD$ compares incidence columns. Its entries are determined entirely by the balanced word-space construction: sixteen on the diagonal, zero between distinct nucleotides in the same positional block, and four between coordinates belonging to different positions.

4.2 Within-Block Structure

Each four-column positional block has diagonal 16 and off-diagonal 0. Within a position, the four nucleotide indicator columns are mutually orthogonal because a codon cannot contain two different nucleotides at the same position.

4.3 Cross-Block Structure

Every column from one position overlaps every column from another position in exactly four codons. The cross-block 4×4 submatrices are therefore filled with fours. This regular block structure makes the spectrum accessible without relying on numerical eigensolvers.

4.4 Rank Ten

The four indicator columns in each positional block sum to the same 64-component all-ones vector. The three block sums are equal, creating two independent linear relations among the twelve columns. Hence $\text{rank}(D)=12-2=10$.

4.5 Two Right-Null Directions

Convenient right-null vectors shift a constant from one positional block to another. For example, adding one to every first-position coefficient and subtracting one from every second-position coefficient changes no codon prediction because every codon activates one coordinate in each block.

$$(1,1,1,1,-1,-1,-1,-1,0,0,0,0) \in \text{null}(D)$$

$$(1,1,1,1,0,0,0,0,-1,-1,-1,-1) \in \text{null}(D)$$

4.6 Exact Gram Eigenvalues

The coefficient space separates into one common constant direction, nine within-position contrast directions, and two null block-offset directions. On those subspaces D^TD has eigenvalues 48, 16, and 0 respectively.

$$\text{spec}(D^TD) = \{48 \times 1, 16 \times 9, 0 \times 2\}$$

4.7 Numerical SVD

A conventional numerical singular-value decomposition independently recovers the same structure. This is useful because it approaches the matrix as a generic numerical object rather than assuming the exact answer.

```
U, s, VT = np.linalg.svd(D.astype(float), full_matrices=False)
```

4.8 Singular Values

Taking square roots of the nonzero Gram eigenvalues gives one singular value $4\sqrt{3}$ and nine singular values equal to 4, followed by two zeros. Numerical values near zero are interpreted under a declared tolerance; the exact Gram calculation removes any ambiguity about rank.

$$\text{singular values: } 4\sqrt{3} \times 1,\ 4 \times 9,\ 0 \times 2$$

4.9 Multiplicities

The multiplicities are structural. Nine contrast dimensions arise because each of three positions contributes three independent contrasts among four nucleotide states. One common constant direction survives. Two block-offset directions are redundant.

4.10 Exact Structure Behind the Numerical Spectrum

The SVD is therefore not merely a numerical summary. It exposes a decomposition already implied by the complete factorial design. This same decomposition determines how the Moore–Penrose pseudoinverse must scale the identifiable subspaces.

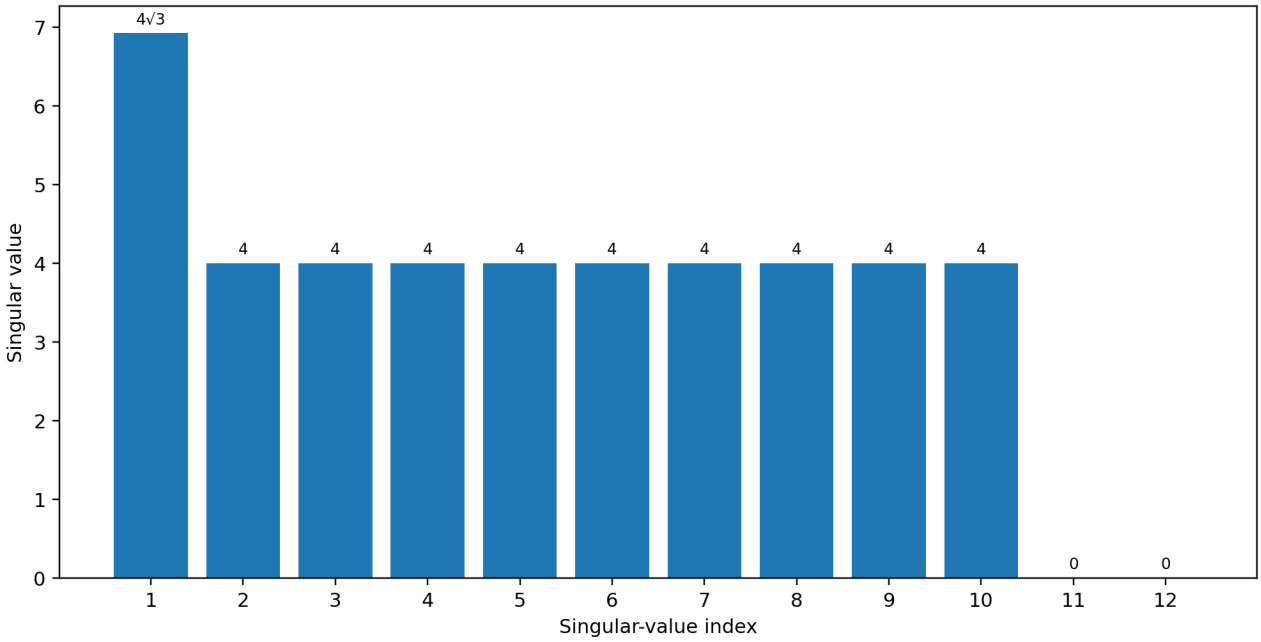


Figure 3. Singular-value spectrum of the complete incidence matrix

4.11 Structural Summary

The matrix has 12 coordinate directions = 9 nucleotide-contrast directions + 1 common constant direction + 2 redundant directions. This compact decomposition is the reason the pseudoinverse collapses to a two-value exact formula.

Invariant	Value
Matrix shape	64 × 12
Row weight	3
Column weight	16
Rank	10
Right nullity	2
Gram eigenvalue 48	multiplicity 1
Gram eigenvalue 16	multiplicity 9
Gram eigenvalue 0	multiplicity 2
Singular value $4\sqrt{3}$	multiplicity 1
Singular value 4	multiplicity 9
Singular value 0	multiplicity 2

Table 3. Structural invariants of the complete codon-incidence matrix

Part III
The Exact Moore–Penrose Pseudoinverse

Chapter 5

Numerical SVD and the Pseudoinverse

5.1 Why an Ordinary Inverse Does Not Exist

D is rectangular and rank deficient, so no ordinary inverse exists. The Moore–Penrose pseudoinverse D^+ is the unique generalized inverse characterized by four identities and is the appropriate object for finding minimum-norm additive nucleotide-position coefficients from arbitrary codon-level fields.

$$DD^+D = D \quad D^+DD^+ = D^+ \quad (DD^+)^T = DD^+ \quad (D^+D)^T = D^+D$$

5.2 Pseudoinverse from SVD

If $D=U\Sigma V^T$, replace each nonzero singular value by its reciprocal and leave the two zero singular values at zero. Then $D^+=V\Sigma^+U^T$. This yields a numerical 12×64 matrix without using any special combinatorial formula.

```
U, s, VT = np.linalg.svd(D_float, full_matrices=False)
s_plus = np.array([1/x if x > tolerance else 0.0 for x in s])
Dplus_svd = VT.T @ np.diag(s_plus) @ U.T
```

5.3 Inspecting the Numerical Answer

Although the matrix contains 768 entries, inspection reveals only two values apart from floating-point noise: approximately 0.052083333333 and -0.010416666667. Rational reconstruction identifies these immediately as $5/96$ and $-1/96$.

5.4 Where the Two Values Occur

The placement is equally simple. If the corresponding entry of D^T is one, D^+ contains $5/96$. If D^T is zero, D^+ contains $-1/96$. The entire pseudoinverse therefore has the same binary pattern as D^T after an affine rescaling.

5.5 Rational Reconstruction

```
from fractions import Fraction
Fraction(0.052083333333).limit_denominator()   # 5/96
Fraction(-0.010416666667).limit_denominator() # -1/96
```

Rational reconstruction is a discovery device, not a proof. Proof comes from constructing the candidate matrix exactly and checking the defining Moore–Penrose identities.

5.6 Counting the Two Entry Types

Each pseudoinverse row corresponds to one position-nucleotide coordinate, which occurs in sixteen codons. Hence every row contains sixteen positive entries and forty-eight negative entries. Across twelve rows there are 192 entries equal to $5/96$ and 576 equal to $-1/96$.

5.7 Row Sums

The row sum is $16(5/96)+48(-1/96)=1/3$. A constant codon-level field is therefore divided equally among the three positional blocks by the minimum-norm solution.

5.8 Column Sums

Each codon activates three of the twelve coordinates, so every pseudoinverse column contains three positive entries and nine negative entries. Its sum is $3(5/96)+9(-1/96)=1/16$.

5.9 From Numerical Discovery to Exact Structure

The numerical calculation suggests the exact closed form $D^{+}=(1/16)D^{T}-(1/96)J$. The next chapter shows that this expression is not an approximation but the exact Moore–Penrose pseudoinverse.

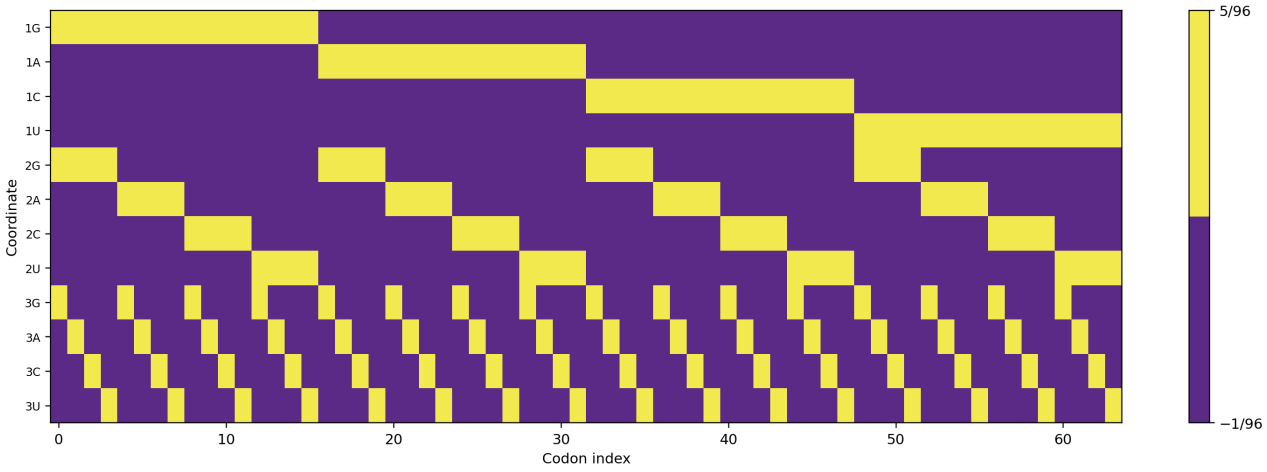


Figure 4. Exact two-value Moore–Penrose pseudoinverse

Chapter 6

The Closed-Form Solution

6.1 Exact Formula

$$D^{+} = (1/16)D^{T} - (1/96)J$$

Here J is the 12×64 all-ones matrix. Because D^{T} is binary, the formula produces $5/96$ at present coordinates and $-1/96$ elsewhere.

```
Dplus_exact = sp.Rational(1,16) * D.T - sp.Rational(1,96) * sp.ones(12,64)
```

6.2 Why the Formula Works

On the nine zero-sum within-position contrast directions, the all-ones correction vanishes and the pseudoinverse applies the reciprocal scale $1/16$. On the common constant direction, the J term corrects the scaling to the value required by the Gram eigenvalue 48. On the two null directions the Moore–Penrose solution selects zero.

6.3 Generic Exact Pseudoinverse

A symbolic linear-algebra library can compute the pseudoinverse without being told the closed form. Comparing that generic exact result with the candidate formula gives exact equality. The closed form is therefore independently reproduced by a general algorithm.

```
Dplus_generic = D.pinv()
assert Dplus_generic == Dplus_exact
```

6.4 Three Routes to the Same Matrix

The release deliberately retains three routes: numerical SVD, generic exact symbolic pseudoinverse, and closed-form rational expression. Their agreement is more informative than any one route alone because the numerical method discovers the pattern, the symbolic method establishes the exact generalized inverse, and the closed form explains the combinatorial structure.

6.5 Numerical SVD Versus Exact Matrix

The maximum elementwise difference between the floating-point SVD result and the exact matrix converted to double precision is approximately 3.6×10^{-16}, at the scale of ordinary floating-point roundoff.

6.6 Numerical Moore–Penrose Residuals

The largest numerical residual among the four defining Moore–Penrose identities in the release calculation is approximately 6.7×10^{-16}. Numerical verification and exact verification are recorded separately because they provide different kinds of evidence.

6.7 Exact Verification

```
assert D * Dplus_exact * D == D
assert Dplus_exact * D * Dplus_exact == Dplus_exact
assert (D * Dplus_exact).T == D * Dplus_exact
assert (Dplus_exact * D).T == Dplus_exact * D
```

6.8 Coefficient-Space Projector

$D^{+}D$ is a 12×12 orthogonal projector onto the ten-dimensional identifiable coefficient subspace. The two-dimensional complement is precisely the block-offset gauge freedom. The Moore–Penrose solution selects the coefficient vector orthogonal to that null space and therefore of minimum Euclidean norm.

6.9 Constant Codon Fields

If every codon is assigned the same value k, each of the twelve pseudoinverse rows returns $k/3$. Every codon then sums one coefficient from each positional block to reconstruct k exactly. This is a direct check of the common constant direction.

6.10 Gauge Freedom

Adding the same constant to all four first-position coefficients while subtracting it from all four second-position coefficients changes no codon prediction. Within-position contrasts, however, are invariant under such shifts. This is why contrasts such as $2U - 2A$ have especially clean interpretation.

6.11 The Sixty-One-Sense-Codon Matrix

Removing the three stop codons destroys the complete factorial balance. Column sums and cross-position overlaps are no longer uniform, so the two-value closed form does not apply to the 61-row matrix. The pseudoinverse remains computable, but the exceptional simplicity belongs to the complete 64-word space.

6.12 Numerical SVD Still Solves the Incomplete Problem

Generic SVD or symbolic pseudoinverse methods remain available for the 61-codon matrix. This contrast clarifies the logical status of the closed form: numerical SVD is general; the simple formula is structural.

Rule / comparison	Value
D^{+} entry: coordinate present in codon	5/96
D^{+} entry: coordinate absent from codon	−1/96
Numerical SVD rank	10
max $\lvert D^{+}(\text{SVD}) - D^{+}(\text{exact})\rvert$	3.556e-16
max SVD Moore–Penrose residual	6.661e-16
Projector: Hamming distance 0	5/32
Projector: Hamming distance 1	3/32
Projector: Hamming distance 2	1/32
Projector: Hamming distance 3	−1/32

Table 4. Exact pseudoinverse rules, numerical-SVD agreement, and projector values

Part IV
The Codon-Space Projector

Chapter 7

Four Cases Hidden in a 64 × 64 Matrix

7.1 Constructing DD^{+}

$$P = DD^{+}$$

P maps an arbitrary 64-component codon field to its closest additive nucleotide-position representation. It is a 64 × 64 matrix, but its 4,096 entries collapse to four exact values.

7.2 Projection Back Into Codon Space

For any codon-level vector h, the additive reconstruction is $\hat{h}=Ph$. If h already belongs to the column space of D, projection changes nothing. Components orthogonal to that subspace are removed.

7.3 Expanding the Formula

Substituting the exact pseudoinverse yields $P=(1/16)DD^{T}-(1/32)J_{64}$ because every row of D contains three ones. The remaining object DD^{T} compares codon rows.

7.4 Shared Coordinates and Hamming Distance

Two codon rows have inner product equal to the number of positions at which the codons agree. If their Hamming distance is d, they agree in $3-d$ positions. Therefore the projector entry depends only on d.

$$P(a,b) = [5 - 2d(a,b)] / 32$$

7.5 Distance Zero

At d=0, every diagonal entry is 5/32. Sixty-four such entries sum to ten, immediately reproducing trace(P)=rank(P)=10.

7.6 Distance One

At d=1, the projector value is 3/32. These are the strongest off-diagonal couplings in the additive projection.

7.7 Distance Two

At d=2, the value is 1/32. Codon pairs share only one active positional coordinate.

7.8 Distance Three

At d=3, the value is $-1/32$. Negative weights are required because an orthogonal projector must subtract components outside the additive subspace; it is not merely a positive local averaging operator.

7.9 Hamming-Sphere Populations

Relative to any codon there is 1 codon at distance 0, 9 at distance 1, 27 at distance 2, and 27 at distance 3. These counts sum to sixty-four and make every projector row sum exactly to one.

7.10 Rank and Trace

P has rank ten and trace ten. As an idempotent projector, its eigenvalues are one with multiplicity ten and zero with multiplicity fifty-four.

7.11 Idempotence

$$P^2 = P \quad \text{and} \quad P^T = P$$

Once a field is projected into the additive subspace, projecting it again makes no further change. Symmetry follows directly from the symmetry of Hamming distance.

7.12 Fifty-Four Residual Dimensions

An arbitrary codon field has sixty-four degrees of freedom. The additive model captures ten. The remaining fifty-four dimensions correspond to pairwise and three-way positional interactions that cannot be represented as independent first-, second-, and third-position contributions.

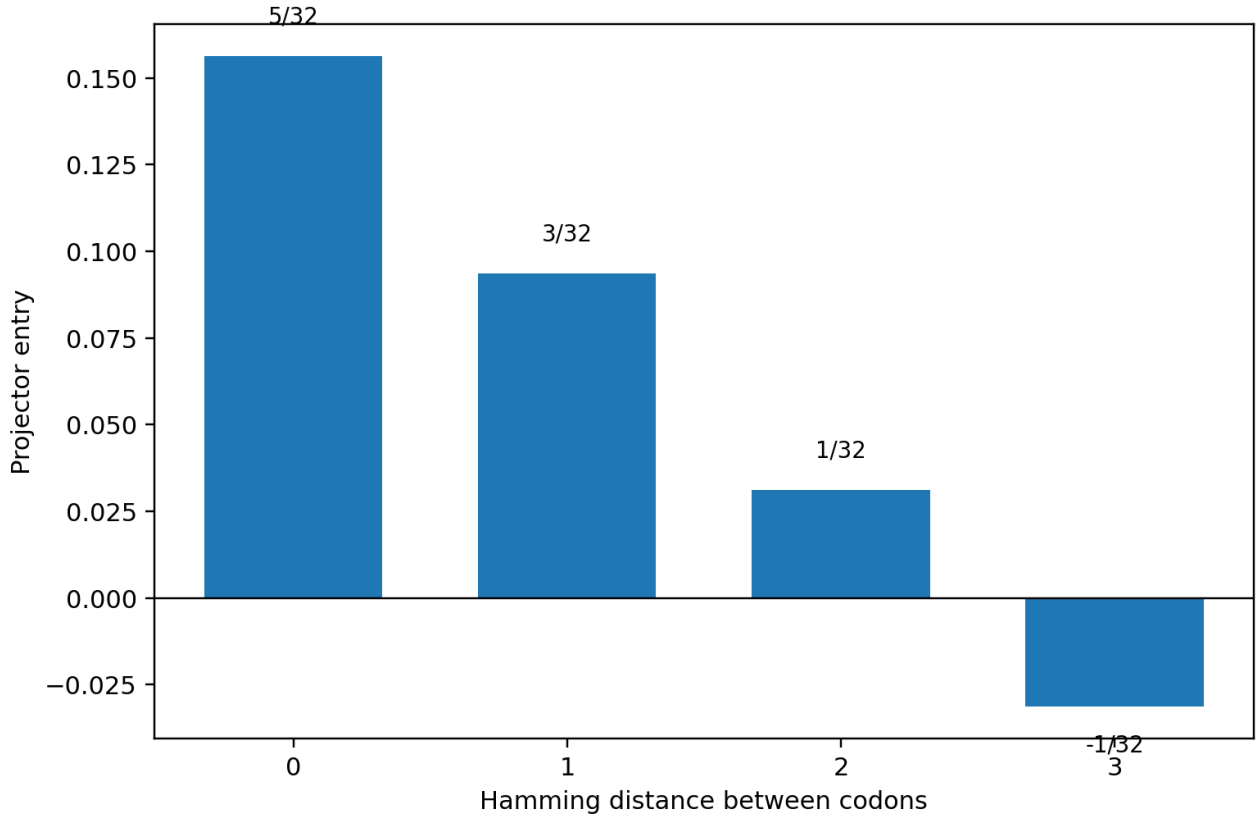


Figure 5. Exact codon-space projector value versus Hamming distance

Part V
Hydropathy as an Output Field

Chapter 8

Attaching Numerical Values

8.1 From Categorical Outputs to a Numerical Field

The translation outputs are categorical. To use linear projection, attach a numerical property to each output and lift that property through the lookup table to all sixty-four codons. The resulting 64-component vector can then be decomposed with D^{+}.

8.2 Kyte–Doolittle Hydropathy

The example property is the Kyte–Doolittle hydropathy scale. The twenty amino-acid values are loaded as source data with provenance recorded separately from the analytical code. Positive and negative values supply a well-known physicochemical field with which to test the codon-position projection.

8.3 Loading the Property

```
hydropathy = load_property_table('data/kyte_doolittle.csv')
```

The loader verifies required amino-acid symbols, numeric values, duplicates, and unexpected schema structure before the property is used.

8.4 X Convention

The complete 64-codon vector requires a numerical value at the three stop codons. The declared convention is $H(X)=0$. This is a computational padding rule and should not be interpreted as a physical hydropathy of stop.

8.5 The 64-Component Property Vector

```
h = [hydropathy_complete[code[codon]] for codon in codons]
```

Because the property is attached after translation, all synonymous codons begin with identical hydropathy values. The projection is then free to represent that field additively in nucleotide-position coordinates.

8.6 Exact Pseudoinverse Projection

$$\beta = D^{+}h$$

The twelve coefficients form the unique minimum-norm additive representation. For a codon $c_1c_2c_3$, the reconstructed value is $\beta(1c_1)+\beta(2c_2)+\beta(3c_3)$.

8.7 Twelve Coefficients

The output is organized in the same 3×4 positional structure as the design matrix. The three positional blocks can then be compared directly without changing the input geometry.

Chapter 9

Nucleotide-Position Hydropathy

9.1 First-Position Coefficients

The exact first-position values are $1G = 659/960 \approx 0.686458$, $1A = -613/960 \approx -0.638542$, $1C = -1201/960 \approx -1.251042$, and $1U = 169/192 \approx 0.880208$. The first position contributes a moderate hydropathy structure.

9.2 Second-Position Coefficients

The second-position coefficients are $2G = -1411/960 \approx -1.469792$, $2A = -2533/960 \approx -2.638542$, $2C = -157/960 \approx -0.163542$, and $2U = 3791/960 \approx 3.948958$. This block contains the largest positive and negative coefficients in the complete vector.

9.3 Third-Position Coefficients

The third-position coefficients are $3G = -373/960 \approx -0.388542$, $3A = -163/960 \approx -0.169792$, $3C = 113/960 \approx 0.117708$, and $3U = 113/960 \approx 0.117708$. Their range is much narrower than the first- and second-position ranges.

9.4 The U–A Contrast

$$\beta(2U) - \beta(2A) = 527/80 = 6.5875$$

This contrast is invariant under the block-offset gauge freedom. It therefore represents a genuine property of the projected field rather than an artifact of the minimum-norm choice of absolute coefficient levels.

9.5 Gauge-Invariant Contrasts

Any difference between two coefficients within the same positional block survives a common offset applied to that block. Such contrasts are therefore natural quantities for comparing nucleotide effects across properties.

9.6 SVD and Exact Agreement

Applying the numerical SVD pseudoinverse and the exact rational pseudoinverse to the same hydropathy field produces coefficient vectors that agree to floating-point precision. The property analysis therefore inherits the independent pseudoinverse cross-check.

9.7 Position-Specific Ranges

The approximate within-block ranges are 2.13125 for position 1, 6.58750 for position 2, and 0.50625 for position 3. The matrix treats all three positions symmetrically before property data are introduced; the observed asymmetry therefore enters through the standard genetic-code assignments and hydropathy field.

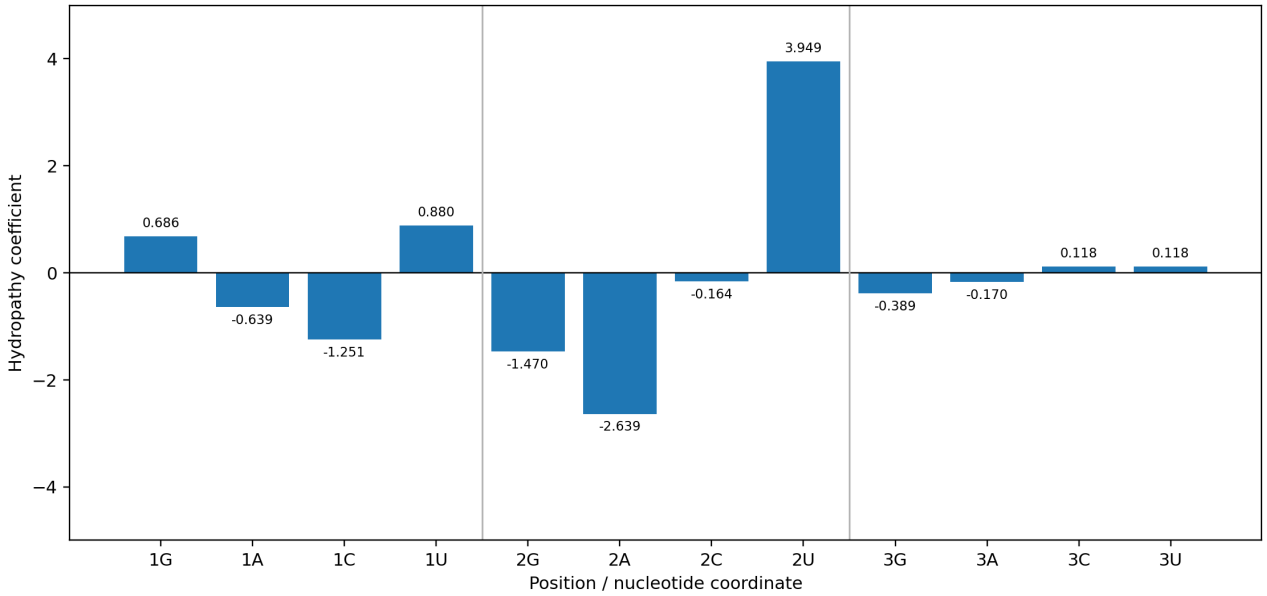


Figure 6. Twelve nucleotide-position hydropathy coefficients

9.8 Input Symmetry and Output Asymmetry

The singular spectrum does not distinguish codon positions, yet the hydropathy coefficients do. This separates universal structure of the complete word matrix from property-specific structure induced by translation.

9.9 The Coefficient Matrix

Arranging the twelve values as three rows and four nucleotide columns makes the positional contrast immediately visible. The model is not a four-parameter nucleotide model; the same nucleotide can carry very different coefficients at different codon positions.

9.10 Exact Values and Decimal Presentation

The computational pipeline retains rational values where possible and uses decimals for human-readable tables and plots. Exact arithmetic is the source of identity checks; decimal formatting is presentation.

9.11 From Coefficients to Reconstruction

The coefficients answer which additive nucleotide-position contributions best represent the codon hydropathy field. They do not yet say how accurately that model reconstructs the original twenty amino-acid values. That is the subject of the next part.

Codon position	G	A	C	U
1	0.686458	-0.638542	-1.251042	0.880208
2	-1.469792	-2.638542	-0.163542	3.948958
3	-0.388542	-0.169792	0.117708	0.117708

Table 5. Complete 3 × 4 nucleotide-position hydropathy coefficient matrix

Part VI
Reconstruction

Chapter 10

Projection and Reconstruction

10.1 From Coefficients Back to Codon Space

$$\hat{h} = D\beta = DD^{+}h = Ph$$

The reconstructed field is the orthogonal projection of the original 64-component hydropathy vector onto the ten-dimensional additive nucleotide-position subspace.

10.2 Codon-Level Reconstruction

For a codon GAC, for example, the reconstructed value is β(1G)+β(2A)+β(3C). Once β is known, the model can be evaluated directly from the codon letters without looking up an amino-acid identity.

10.3 Synonymous Codons Need Not Reconstruct Identically

The original field is constant within synonymous fibers, but the additive projection need not preserve that equality. Synonymous codons with different nucleotide coordinates can receive slightly different reconstructed values. The projection preserves the additive component of the field, not every categorical equality imposed by translation.

10.4 Returning to Amino-Acid Values

To compare with the twenty empirical amino-acid hydropathy values, reconstructed codon values are averaged within each translated amino-acid fiber. X is excluded from this twenty-point regression comparison.

10.5 Regression

The reconstructed and empirical amino-acid values show a strong linear relationship. The fitted slope is approximately 0.859718 and the intercept approximately −0.003660.

10.6 Pearson Correlation

$$r \approx 0.915646$$

Relatively hydrophobic amino acids tend to remain hydrophobic in the reconstructed scale, and relatively hydrophilic amino acids tend to remain hydrophilic. The regression summarizes an output that was generated before the regression was calculated; it is not used to fit the nucleotide-position coefficients.

10.7 Coefficient of Determination

$$R^2 \approx 0.838407$$

This amino-acid-level value should not be confused with the fraction of the original 64-dimensional codon field captured by orthogonal projection. The two statistics answer different questions.

10.8 Codon-Level Projection Fraction

$$||Ph||^2 \,/\, ||h||^2 \approx 0.816882$$

A large majority of the squared magnitude of the complete codon hydropathy field lies in the additive nucleotide-position subspace. The remainder lies in the fifty-four-dimensional nonadditive space.

10.9 Residuals

$$r = h - \hat{h}$$

The residual contains the codon-level structure not representable by independent position contributions. Because ¯^ is an orthogonal projection, the residual is orthogonal to every column of D.

10.10 Residual Orthogonality

$$D^{T}r = 0$$

This condition proves that no first-order nucleotide-position signal remains available to the model. Any remaining structure requires pairwise interactions, three-way interactions, or codon-specific effects.

10.11 What the Additive Model Captures

The model can represent any property that separates into a constant plus independent effects of the first, second, and third codon positions. It cannot represent irreducible combinations such as a contribution that occurs only for a particular pair of nucleotides at two positions.

10.12 Reconstruction Is Not Translation

Strong hydropathy reconstruction does not mean the twenty-one-class categorical lookup function has been recovered. Hydropathy is one numerical property and many outputs can share similar values. The lookup table and the property projection remain distinct objects.

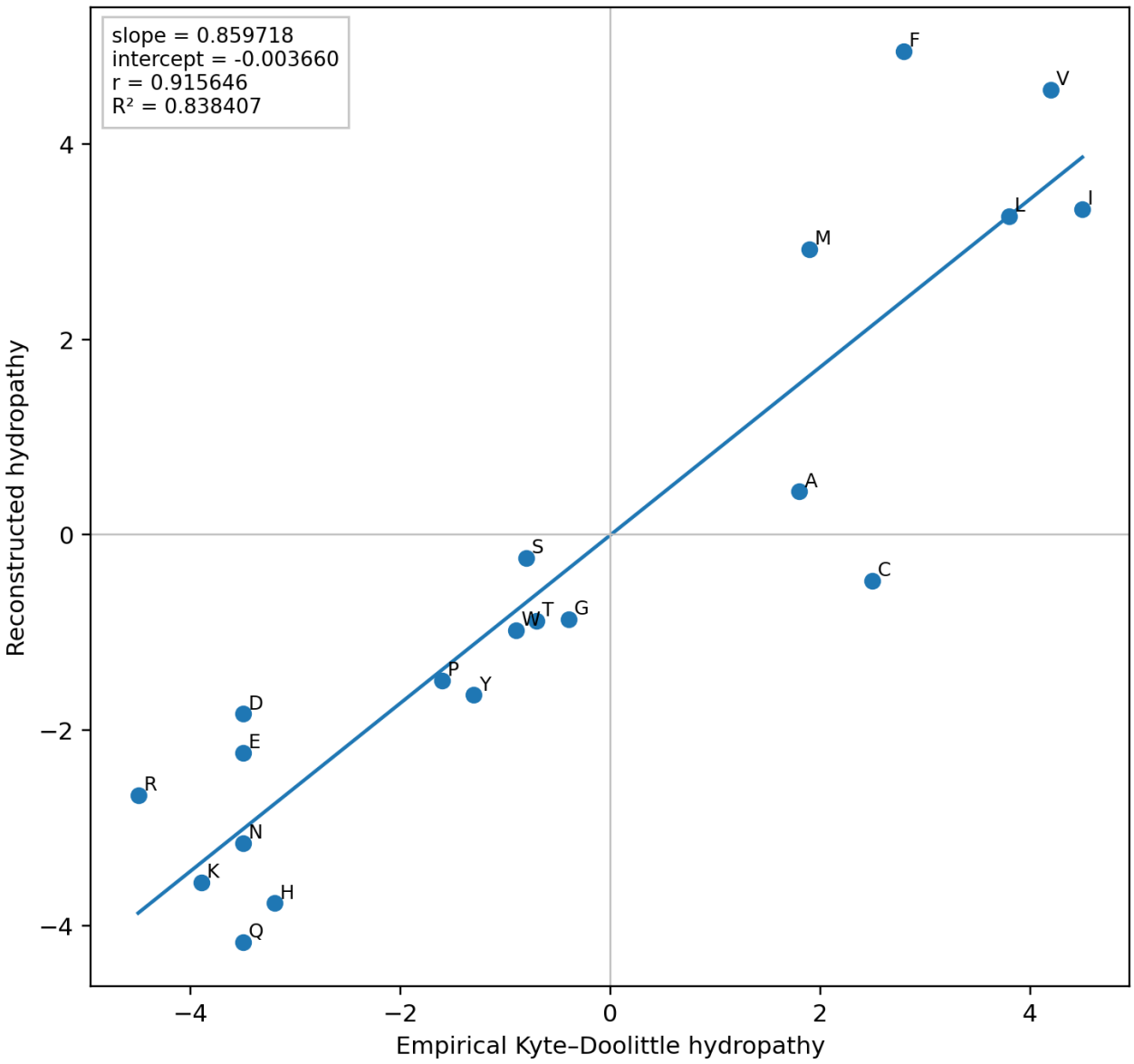


Figure 7. Empirical versus reconstructed amino-acid hydropathy

Part VII
Hydropathy Directly from Codon Position Two

Chapter 11

The U–A Profile

11.1 From Amino-Acid Hydropathy to a Nucleotide-Only Signal

The dominant second-position U−A contrast suggests a deliberately compressed sequence score: second-position U maps to +1, A to −1, and C or G to 0. The resulting profile uses only the middle base of each codon and never performs amino-acid translation during scoring.

11.2 Conventional Kyte–Doolittle Profiling

The conventional route translates the coding sequence, assigns each amino acid its Kyte–Doolittle value, and applies a sliding-window mean. The example uses a nineteen-residue window.

```
kd_values = [hydropathy[aa] for aa in protein_sequence]
kd_profile = sliding_window(kd_values, window=19)
```

11.3 Reading the Same Coding Sequence Without Translation

```
def middle_base_score(codon):
  if codon[1] == 'U': return 1.0
  if codon[1] == 'A': return -1.0
  return 0.0
```

The coding sequence is partitioned in the declared reading frame. One score is produced per codon, then smoothed with the same window alignment used for the conventional profile.

11.4 Why Position Two

The rule is motivated by the exact projected coefficients, not selected independently. The U−A contrast is the largest within-position contrast in the twelve-parameter model.

11.5 Why C and G Are Set to Zero

Zero does not assert that second-position C and G are irrelevant to hydropathy. It deliberately discards smaller distinctions to test how much of the broad profile shape is retained by the dominant signed polarity alone.

11.6 Sliding-Window Calculation

At single-codon resolution the U−A signal takes only three values. Sliding-window averaging turns that discrete sequence into a smooth profile comparable to a conventional hydropathy trace.

11.7 Plotting Normalization

The raw moving average lies between −1 and +1. For visual comparison it is multiplied by 4.5 so its theoretical range is −4.5 to +4.5. This is a plotting normalization, not a fitted physical conversion factor, and it does not change Pearson correlation.

11.8 Deposited Coding Sequence

The example uses the deposited reaction-center L-subunit coding sequence traced to GenBank accession Z11165.1, gene pufL, Rhodobacter capsulatus. The same underlying coding sequence feeds both the translated and nucleotide-only routes.

11.9 Two Routes Through the Same Sequence

The conventional route is coding sequence → codons → amino acids → twenty-value hydropathy scale → sliding-window mean. The compressed route is coding sequence → codons → second nucleotide → U/A/C/G score → sliding-window mean.

11.10 Pearson Correlation

$$\text{example profile: } r \approx 0.9431 \quad R^2 \approx 0.8895$$

The strong point-by-point correlation quantifies the visual alignment of the two smoothed traces for this deposited sequence.

11.11 Limits of the Example

The result is sequence-specific. It does not establish identical performance across all membrane proteins or all coding sequences. A broader claim would require a declared benchmark set, preprocessing rules, controls, and predefined statistics.

11.12 Reading-Frame and Synonymous-Coding Dependence

The nucleotide-only profile depends on the coding frame and can change under synonymous recoding. Many synonymous substitutions occur at the third position and therefore leave this second-position score unchanged, but invariance is not guaranteed.

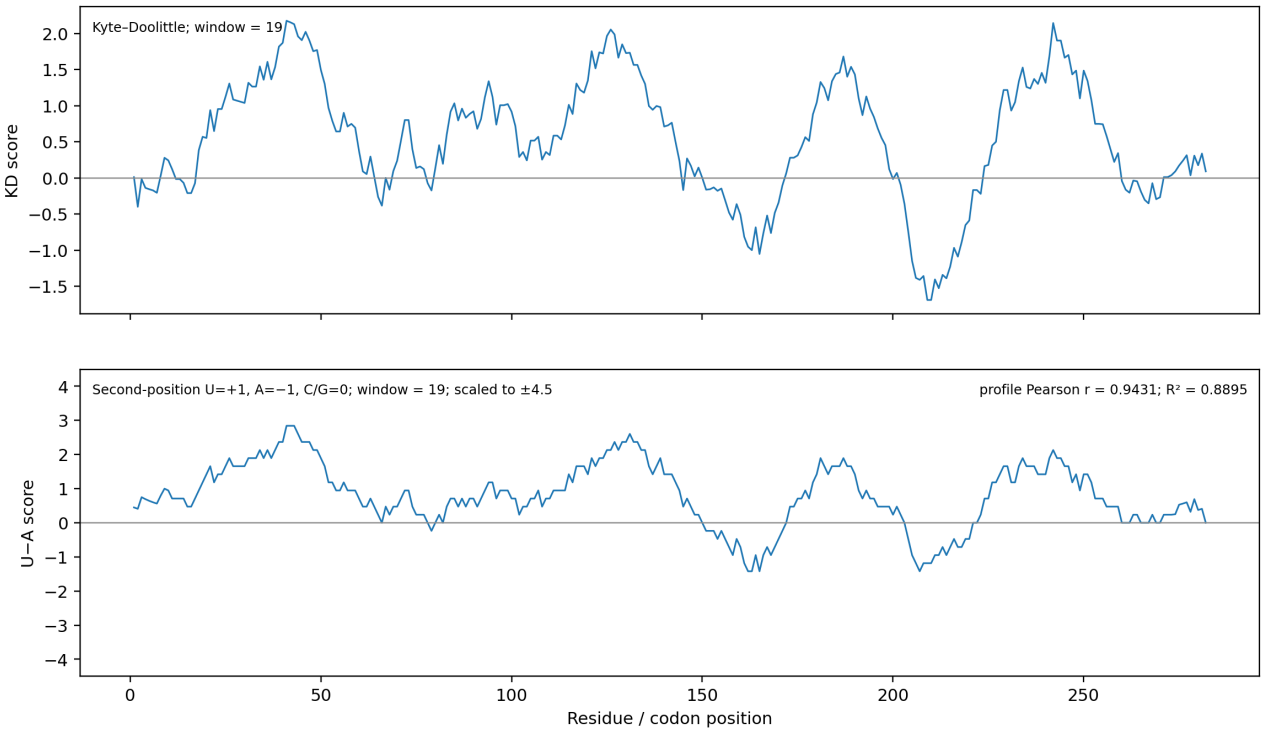


Figure 8. Kyte–Doolittle and second-position U–A profiles

Part VIII
Codon Space as a Graph

Chapter 12

The G–A–C–U Lattice

12.1 From Strings to Vertices

Every codon is now treated as a graph vertex. To define local edges we require more than Hamming distance: the four nucleotide states are placed along a declared linear path, and an edge changes one codon position by one adjacent step along that path.

12.2 The G–A–C–U Path

G — A — C — U

This path contains the unordered nucleotide adjacencies G–A, A–C, and C–U. It is linear rather than cyclic: there is no extra U–G edge.

12.3 Three Independent Axes

The path is applied independently to codon positions 1, 2, and 3. Each codon is assigned a coordinate triple whose entries are the four path levels. The complete input space therefore becomes a $4 \times 4 \times 4$ Cartesian lattice with sixty-four vertices.

12.4 Local Edges

Two codons are lattice neighbors when they differ in exactly one position and the differing nucleotides are adjacent on the selected path. This picks a structured half of all possible Hamming-distance-one substitutions.

12.5 Forty-Eight Edges per Axis

Each four-state path contains three adjacency pairs. For one codon position, the other two positions have $4 \times 4 = 16$ settings, yielding $3 \times 16 = 48$ edges. Across three positions the lattice therefore contains 144 undirected edges.

12.6 Translation Labels

The vertices retain their original lookup values. Every edge can therefore be classified by comparing the translated symbols at its endpoints.

12.7 Synonymous and Nonsynonymous Edges

For G–A–C–U, exactly forty-four of the 144 lattice edges preserve translation identity. The remaining one hundred connect different translated outputs. These counts are generated from the lookup table and lattice construction rather than drawn into the figure manually.

12.8 Position-Specific Synonymous Counts

position 1: 4 position 2: 1 position 3: 39

The extreme concentration of synonymous adjacency along the third position is a graph-theoretic asymmetry distinct from the hydropathy projection, where the second position carries the strongest property contrast.

12.9 Why 44 Is Not Yet an Optimality Result

At this point, 44 is only the score of one selected nucleotide path. There are twenty-three other oriented isotropic paths and 13,823 other oriented anisotropic axis triples. Global optimality requires examining those complete finite spaces.

12.10 Lattice Versus Full Hamming Graph

The full Hamming-distance-one graph has 288 undirected edges because each codon has nine one-substitution neighbors. A four-state path has three edges rather than all six possible nucleotide pairs, so the ordered lattice retains exactly half of the Hamming-one edges: 144.

12.11 Isotropic and Anisotropic Cubes

An isotropic cube uses one nucleotide path on all three axes. An anisotropic cube allows independent paths for positions 1, 2, and 3. The former produces 24 oriented cases; the latter $24^3=13{,}824$.

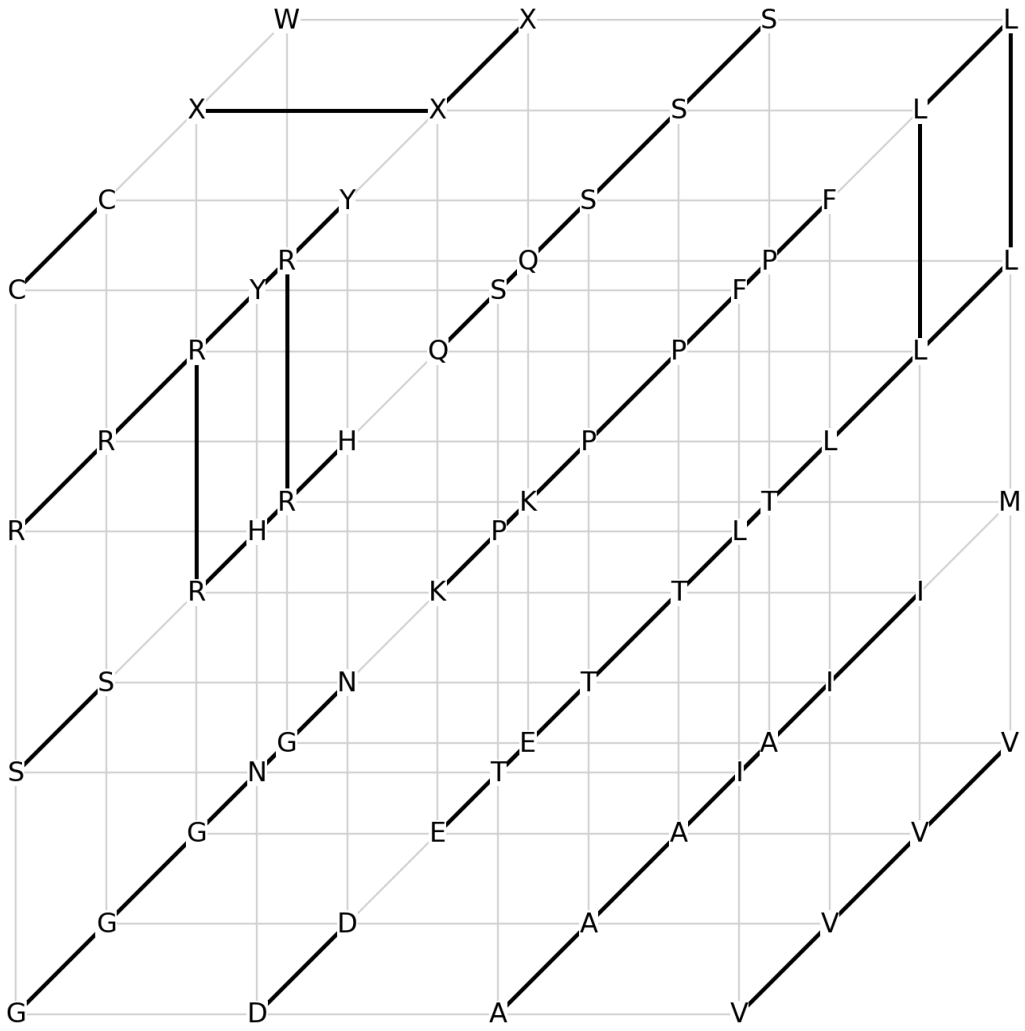


Figure 9. Translated G–A–C–U 4 × 4 × 4 codon lattice

Part IX
The Translated Quotient Multigraph

Chapter 13

Collapsing Codons by Translation Identity

13.1 From Sixty-Four Codons to Twenty-One Outputs

The equivalence relation $c_1 \sim c_2$ exactly when $\tau(c_1)=\tau(c_2)$ collapses each synonymous fiber to a single translated vertex. The 64 codon vertices therefore become twenty-one output vertices while edge instances are carried forward.

13.2 Constructing the Quotient

Each codon-level edge is replaced by the pair of translated endpoint symbols. Distinct codon edges can map to the same translated pair, so multiplicity must be retained rather than discarded in a set.

```
translated_edges = Counter()
for a, b in codon_edges:
   u, v = code[a], code[b]
   translated_edges[tuple(sorted((u, v)))] += 1
```

13.3 Synonymous Edges Become Self-Loops

If both endpoint codons translate to the same output, the quotient edge begins and ends at the same translated vertex. Every one of the forty-four synonymous lattice edges therefore becomes a self-loop instance.

13.4 Nonsynonymous Edges Become Interclass Edges

The one hundred nonsynonymous lattice edges become edge instances connecting different translated vertices. The quotient therefore retains all 144 original edge instances: 44 loops plus 100 interclass instances.

13.5 Parallel Edges

Several distinct codon-level edges can collapse to the same translated pair. The quotient is therefore a multigraph. A simple graph would preserve adjacency but lose multiplicity, which is part of the geometry induced by the lookup function.

13.6 Twenty-One Vertices

The quotient vertex set is exactly the image of the translation function: the twenty amino-acid symbols plus X. Degeneracy is now supplemented by graph information such as internal loop count and multiplicity of connections to other classes.

13.7 Independent Reconstruction of Forty-Four Loops

The release does not rely on one route. A second implementation begins inside each translated fiber, enumerates unordered codon pairs, and counts those satisfying the G–A–C–U lattice adjacency rule. It independently reproduces the total of forty-four.

13.8 What the Quotient Preserves and Suppresses

The quotient preserves edge instances, synonymy status, translated endpoint identity, and multiplicity, but suppresses which specific codon within a fiber participated in an edge. The 64-vertex lattice and the 21-vertex quotient answer different questions and should be retained together.

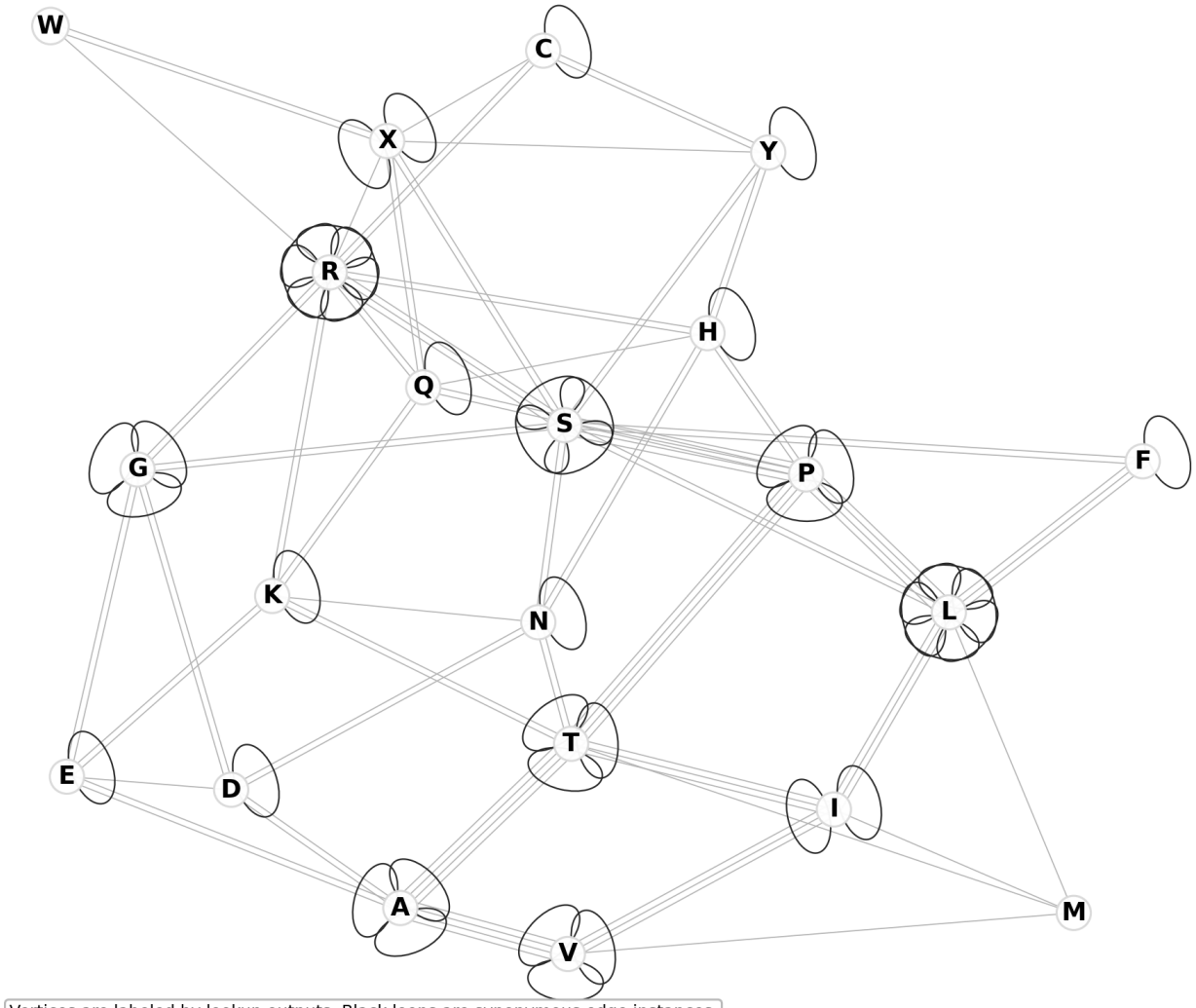


Figure 10. Twenty-one-vertex translated quotient multigraph

Chapter 14

Self-Loop Multiplicity and Fiber Geometry

14.1 Class-Specific Loop Counts

Decomposing the forty-four loop instances by output gives R6, L6, S4; A3, G3, P3, T3, V3; I2, X2; C1, D1, E1, F1, H1, K1, N1, Q1, Y1; M0, W0. The weighted total is exactly forty-four.

14.2 Six-Codon Fibers

R and L each have six codons and six internal lattice adjacencies. S also has six codons but only four. Equal degeneracy therefore does not imply equal local connectivity.

14.3 Four-Codon Classes

A, G, P, T, and V each have loop multiplicity three. A four-element fiber has six possible internal pairs, so exactly half are selected as local lattice edges in these cases.

14.4 Three- and Two-Codon Fibers

I and X each contribute two loops. Each adjacent two-codon fiber contributes the only possible internal pair and therefore has loop count one.

14.5 Singletons

M and W have one codon each and therefore no possible internal pair. Their loop count is forced to zero before geometry is considered.

14.6 Internal Connectivity Ratio

A normalized ratio can divide loop count by the number of possible internal pairs. R and L give $6/15=0.4$, S gives $4/15\approx0.267$, the four-codon examples give $3/6=0.5$, and adjacent two-codon fibers give 1. This ratio depends on the chosen nucleotide path and is not a code invariant by itself.

14.7 Three Decompositions of the Same Forty-Four Edges

The same edge set can be summed by translated output, by codon position, or by adjacent nucleotide pair. These independent decompositions all return forty-four and provide useful cross-checks.

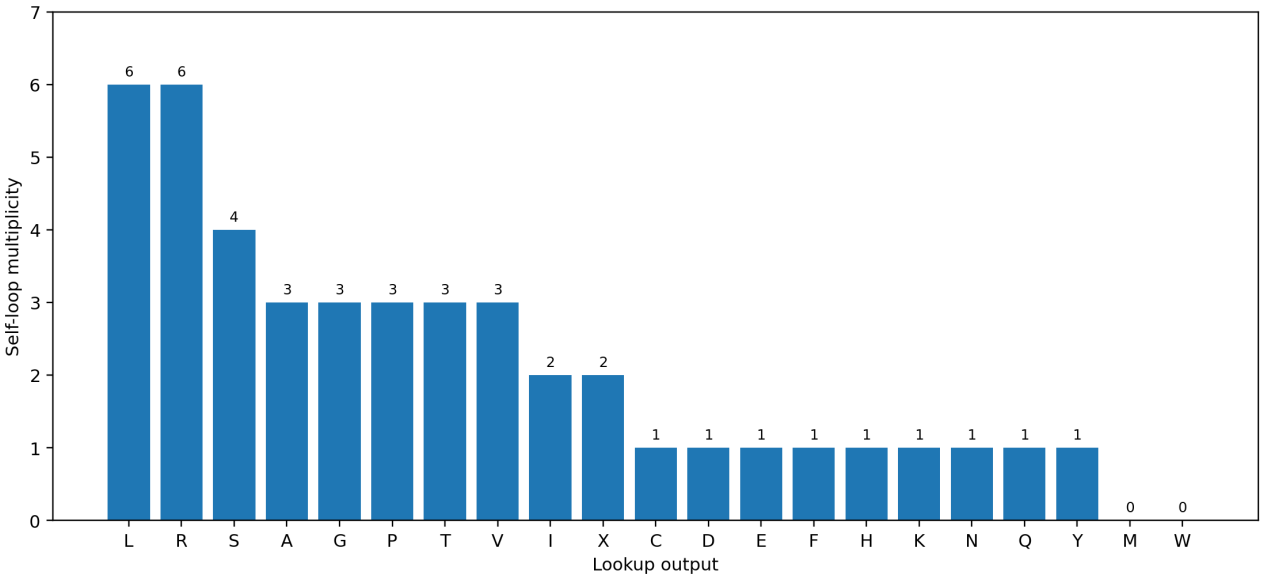


Figure 11. Self-loop multiplicity by translated output

Part X
Exhaustive Isotropic Search

Chapter 15

All Twenty-Four Nucleotide Orders

15.1 The Complete Search Space

Four distinct nucleotide symbols admit 4!=24 oriented linear orders. The search is small enough that every case can be evaluated directly; sampling or heuristic optimization would only weaken the result.

```
orders = list(permutations('GACU'))
assert len(orders) == 24
```

15.2 Every Isotropic Cube Has 144 Edges

Every four-state path contains three adjacencies, and each adjacency expands to sixteen codon edges per position. Therefore every isotropic candidate has the same 144-edge graph topology; only the assignment of nucleotide identities to path levels changes.

15.3 Isotropic Score

Define S(p) as the number of synonymous lattice edges when path p is used on all three axes. This converts each of the twenty-four orders into one exact integer score.

15.4 Exhaustive Range

$$25 \le S(p) \le 44$$

The complete isotropic search establishes both endpoints globally within the declared path model. There are no unexamined cases.

15.5 G–A–C–U

$$S(G{-}A{-}C{-}U) = 44$$

The path already used for the lattice reaches the global maximum.

15.6 Reversal Symmetry

A path and its reversal have the same undirected adjacency pairs, so every score occurs in reversal pairs. The twenty-four oriented permutations therefore represent twelve undirected paths.

15.7 Uniqueness up to Reversal

Only G–A–C–U and U–C–A–G attain 44. They are the two orientations of the same undirected path. Hence the isotropic maximizer is unique up to reversal.

15.8 Conditional Nature of the Result

The optimization is exact under the declared model: standard genetic-code labels, four-state linear path adjacency, undirected edges, and one common path on all three codon axes. A different graph model would define a different optimization problem.

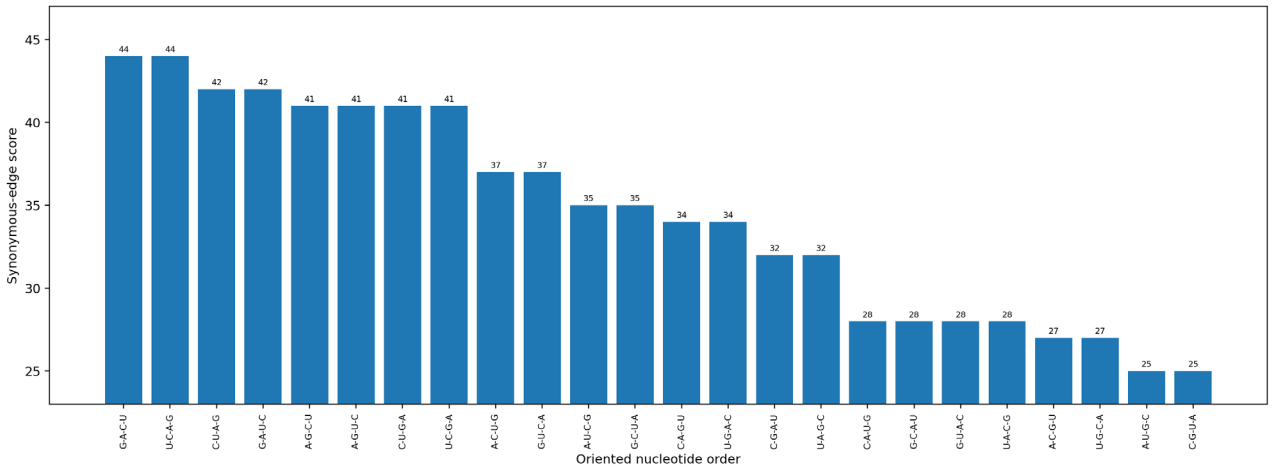


Figure 12. Synonymous-edge scores for all 24 oriented isotropic nucleotide orders

15.9 Complete Enumeration Table

The full table retains every oriented candidate rather than only the extrema. It also makes reversal-paired scores and the completeness of the search auditable without relying on the plotted order of bars.

Path	Reverse	Undirected	L1	L2	L3	Total
G-A-C-U	U-C-A-G	G-A-C-U	4	1	39	44
U-C-A-G	G-A-C-U	G-A-C-U	4	1	39	44
C-U-A-G	G-A-U-C	C-U-A-G	2	1	39	42
G-A-U-C	C-U-A-G	C-U-A-G	2	1	39	42
A-G-C-U	U-C-G-A	A-G-C-U	2	1	38	41
A-G-U-C	C-U-G-A	A-G-U-C	2	1	38	41
C-U-G-A	A-G-U-C	A-G-U-C	2	1	38	41
U-C-G-A	A-G-C-U	A-G-C-U	2	1	38	41
A-C-U-G	G-U-C-A	A-C-U-G	4	0	33	37
G-U-C-A	A-C-U-G	A-C-U-G	4	0	33	37
A-U-C-G	G-C-U-A	A-U-C-G	2	0	33	35
G-C-U-A	A-U-C-G	A-U-C-G	2	0	33	35
C-A-G-U	U-G-A-C	C-A-G-U	2	1	31	34
U-G-A-C	C-A-G-U	C-A-G-U	2	1	31	34
C-G-A-U	U-A-G-C	C-G-A-U	0	1	31	32
U-A-G-C	C-G-A-U	C-G-A-U	0	1	31	32
C-A-U-G	G-U-A-C	C-A-U-G	2	0	26	28
G-C-A-U	U-A-C-G	G-C-A-U	2	0	26	28
G-U-A-C	C-A-U-G	C-A-U-G	2	0	26	28
U-A-C-G	G-C-A-U	G-C-A-U	2	0	26	28
A-C-G-U	U-G-C-A	A-C-G-U	2	0	25	27
U-G-C-A	A-C-G-U	A-C-G-U	2	0	25	27
A-U-G-C	C-G-U-A	A-U-G-C	0	0	25	25
C-G-U-A	A-U-G-C	A-U-G-C	0	0	25	25

Table 6. Exhaustive enumeration of the 24 oriented isotropic nucleotide orders

Part XI
Position-Specific Edge Weights

Chapter 16

Scoring Paths Without Rebuilding the Cube

16.1 Six Unordered Nucleotide Pairs

Four nucleotide symbols determine six unordered pairs: A–G, A–C, A–U, C–G, C–U, and G–U. Every four-state path selects exactly three of those six pairs as adjacent.

16.2 Position-Specific Pair Weights

For each unordered pair, the program counts how many synonymous codon edges would arise if that pair were adjacent at each of the three codon positions. The resulting triples are A–G:(0,1,14), A–C:(2,0,9), A–U:(0,0,9), C–G:(0,0,8), C–U:(2,0,16), G–U:(0,0,8).

16.3 Position One

Only A–C and C–U contribute at position 1, with weight two each. The maximum position-one path score is therefore four, achieved by paths containing both adjacencies.

16.4 Position Two

Only A–G contributes at position 2, with weight one. Any path containing A–G reaches the position-two maximum of one.

16.5 Position Three

The third-position weights are 14, 9, 9, 8, 16, and 8 for A–G, A–C, A–U, C–G, C–U, and G–U respectively. The G–A–C–U path selects weights 14+9+16=39.

16.6 Reconstructing the G–A–C–U Score

Summed by nucleotide pair, G–A contributes 15, A–C contributes 11, and C–U contributes 18, giving 44. Summed by codon position, the same edge set gives 4+1+39=44.

16.7 Scoring Any Isotropic Path

An isotropic path score is simply the sum of the total weights of its three adjacent unordered nucleotide pairs. The explicit 144-edge graph therefore has a compact equivalent representation.

16.8 Independent Agreement

The release verifies that explicit graph construction and pair-weight summation agree for all twenty-four isotropic paths. The second method is both a cross-check and the key to understanding the anisotropic search analytically.

Nucleotide pair	Position 1	Position 2	Position 3	Total
A-G	0	1	14	15
A-C	2	0	9	11
A-U	0	0	9	9
C-G	0	0	8	8
C-U	2	0	16	18
G-U	0	0	8	8

Table 7. Position-specific synonymous-edge weights for all six unordered nucleotide pairs

Part XII
Exhaustive Anisotropic Search

Chapter 17

All 13,824 Three-Axis Orderings

17.1 Removing the Isotropic Constraint

Each codon axis may now choose any of the twenty-four oriented nucleotide paths independently. The complete oriented search space therefore contains $24^3 = 13{,}824$ axis triples.

17.2 Explicit Enumeration

```
for p1 in orders:
  for p2 in orders:
    for p3 in orders:
      score = anisotropic_score(p1, p2, p3, code)
      results.append((p1, p2, p3, score))

assert len(results) == 13824
```

17.3 Position-Separable Score

$$S(p_1,p_2,p_3) = S_1(p_1) + S_2(p_2) + S_3(p_3)$$

The axes contribute independently. The full search is therefore the Cartesian sum of three twenty-four-element score lists rather than an opaque optimization problem.

17.4 Direct Enumeration Versus Convolution

The release obtains the complete score histogram by direct graph enumeration and independently by convolving the three positional score distributions. Exact agreement supplies a second verification route.

17.5 Global Range

minimum = 25 maximum = 44

Allowing independent paths does not produce a score above the isotropic G–A–C–U maximum.

17.6 Exact Distribution

The counts for scores 25 through 44 are respectively 192, 384, 960, 1536, 960, 384, 384, 192, 960, 960, 960, 960, 192, 384, 384, 960, 1536, 960, 384, and 192. They sum exactly to 13,824.

17.7 Mean, Variance, and Standard Deviation

mean = 34.5 variance = 30.25 standard deviation = 5.5

The distribution is exact, not Monte Carlo. Its simple moments arise from the balanced enumeration of nucleotide paths and their pair weights.

17.8 Symmetry

The count at score s equals the count at 69−s, making the distribution symmetric around 34.5. This symmetry is a useful independent regression condition for the search implementation.

17.9 Oriented and Undirected Maxima

Exactly 192 oriented anisotropic triples attain the maximum of 44. Independent reversal of the three undirected axis paths produces groups of $2^3=8$ orientations, leaving 24 undirected maximizing cubes.

17.10 Why Forty-Four Remains the Maximum

The position-specific maxima are 4, 1, and 39. Their sum gives a sharp global upper bound of 44, and admissible axis triples attain all three maxima simultaneously.

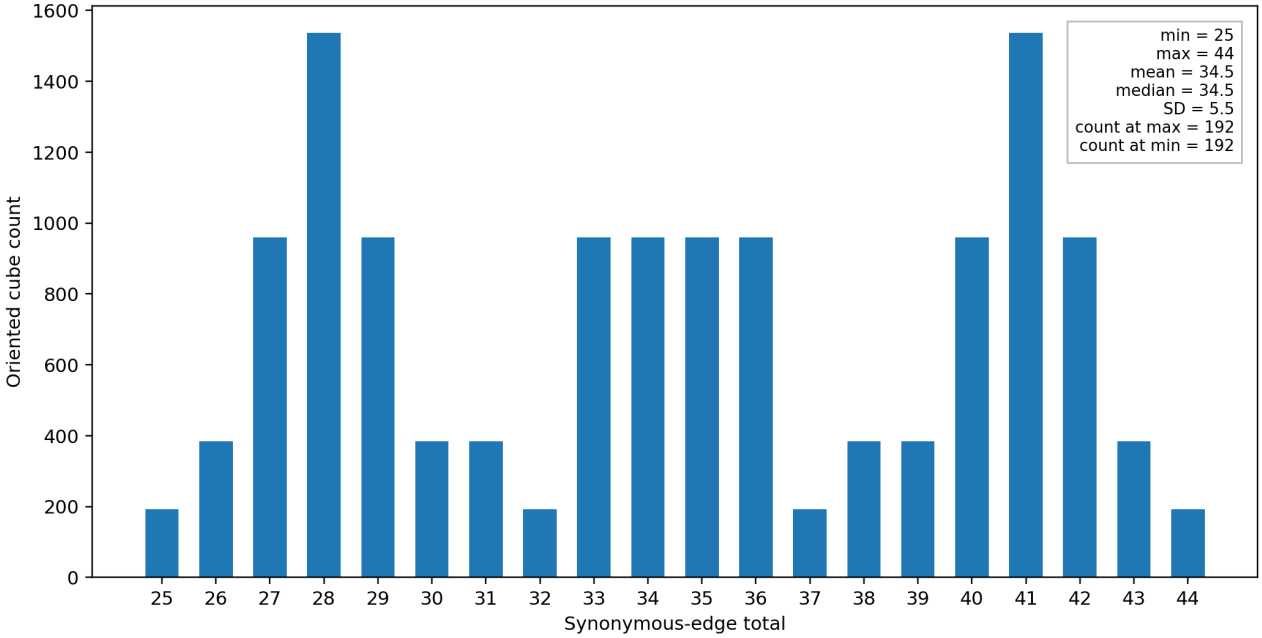


Figure 13. Exact score distribution over all 13,824 oriented anisotropic cubes

Part XIII
The Forty-Four-Edge Maximum

Chapter 18

Why the Global Maximum Is 44

18.1 From Enumeration to Derivation

The exhaustive search establishes the answer computationally. Position-separable scoring supplies a compact derivation: maximize each codon axis independently and add the results.

$$\max S = \max S_1 + \max S_2 + \max S_3 = 4 + 1 + 39 = 44$$

18.2 Position-One Maximum Family

Position one has nonzero pair weights only for A–C and C–U. A maximizing path must contain both, forcing the three-symbol segment A–C–U or its reversal, with G attached at one exposed end.

18.3 Position-Two Maximum Family

Position two is maximized exactly by paths in which A and G are adjacent. This is a broader family because only one adjacency constraint must be satisfied.

18.4 Position-Three Maximum Family

At position three, the combination C–U=16, A–G=14, and A–C=9 forms the path U–C–A–G or its reversal G–A–C–U and sums to 39. No valid four-state path has a larger position-three weight.

18.5 Sharp Bound

Because the anisotropic model permits independent choices on the three axes, every combination of a position-one maximizer, a position-two maximizer, and a position-three maximizer attains 44. The global maximizing family is the Cartesian product of the three axis-specific families.

18.6 Counting the Maximizers

The product of the oriented family sizes is 192, matching exhaustive enumeration exactly. Dividing by eight independent axis reversals yields twenty-four undirected maximizing anisotropic cubes.

18.7 The Isotropic Constraint

Isotropy replaces the Cartesian product by an intersection: one path must belong simultaneously to all three maximizing families.

18.8 Unique Common Path

G–A–C–U contains A–C and C–U for the position-one maximum, A–G for the position-two maximum, and the maximum third-position triple of adjacencies. No other undirected path satisfies all three conditions.

18.9 A Small Certificate of Global Optimality

The eighteen position-specific pair weights, the three axis maxima, and one explicit maximizing path per axis form a compact human-checkable certificate of the 44-edge bound. Exhaustive enumeration remains valuable as an independent verification rather than the sole basis of the result.

Position	Max score	Oriented count	Undirected count	Undirected maximizing paths
Position 1	4	4	2	A-C-U-G; G-A-C-U
Position 2	1	12	6	A-G-C-U; A-G-U-C; C-A-G-U; C-G-A-U; C-U-A-G; G-A-C-U
Position 3	39	4	2	C-U-A-G; G-A-C-U
Cartesian product	44	192	24	Common: G-A-C-U

Table 8. Position-specific maximum families and the 44-edge global bound

18.10 Intersection of Maximum Families

Anisotropic maximizers arise from $M_1 \times M_2 \times M_3$; isotropic maximizers arise from $M_1 \cap M_2 \cap M_3$. The latter contains one undirected path: G–A–C–U.

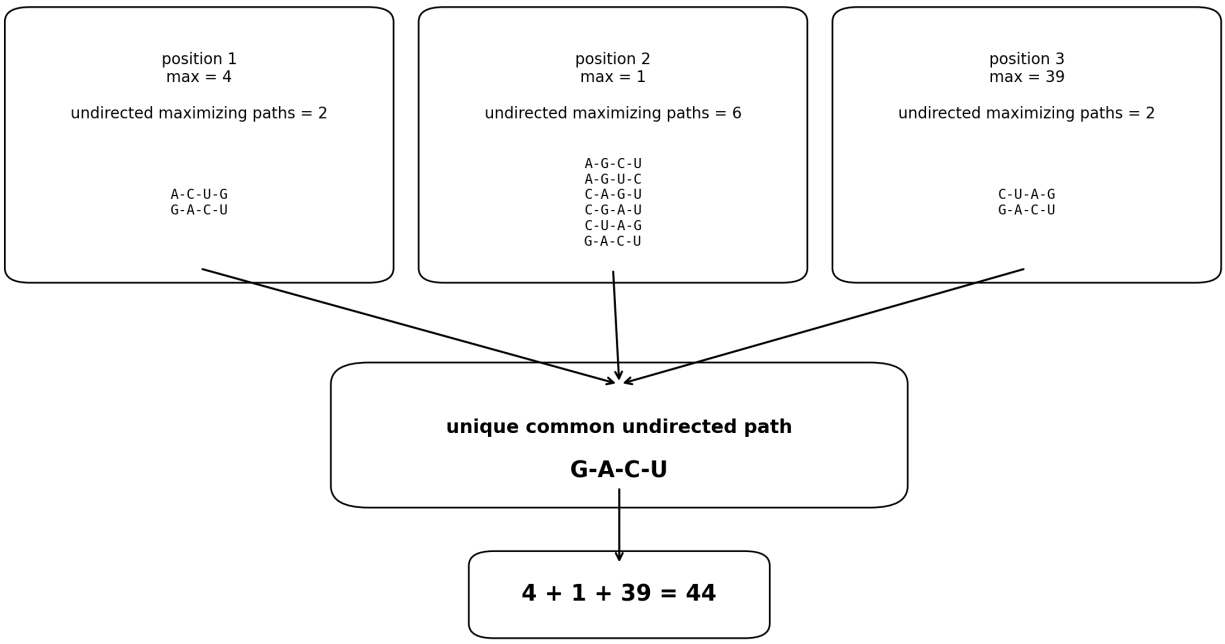


Figure 14. Intersection of position-specific maximizing families at G–A–C–U

Part XIV
Alternative Genetic Codes and Comparison Spaces

Chapter 19

Defining the Space Before Asking Whether the Code Is Unusual

19.1 Why a Null Model Must Be Explicit

Exact structural claims about the standard code require no randomization. Comparative claims do. There is no unique random genetic code, because different randomization rules preserve different aspects of the observed lookup table.

19.2 Unrestricted Codon Reassignment

The broadest ensemble allows codon outputs to be reassigned with few constraints. Such a space may contain lookup tables whose degeneracy profile and stop count differ radically from the standard code. It therefore answers a very broad question and must not be confused with more constrained alternatives.

19.3 Degeneracy-Preserving Reassignment

A narrower comparison keeps the multiset of output degeneracies fixed while changing which codons belong to each translated class. This isolates geometry from class size and is especially natural for testing synonymous-edge organization.

19.4 Codon-Box-Preserving Models

Still narrower ensembles can preserve first-two-position box structure or other declared local constraints. A code can look unusual under an unrestricted ensemble but ordinary under a strongly constrained one; neither result is meaningful without naming the ensemble.

19.5 Property-Preserving Permutations

Another comparison leaves the genetic-code fibers fixed but permutes numerical property values among output classes. For hydropathy this would test whether the observed property alignment with nucleotide-position coordinates is unusual relative to alternative assignments of the same twenty values.

19.6 Exact Enumeration Versus Sampling

Whenever the comparison space is small, exhaustive enumeration is preferable. Large code-reassignment spaces may require sampling, in which case generator, seed, sample count, statistic, tie treatment, source table, and software version become part of the scientific specification.

19.7 Choose the Statistic Before Inspecting the Distribution

A null model must be paired with a declared statistic: G–A–C–U score, global maximum, second-position hydropathy contrast, projection fraction, reconstruction correlation, or another explicitly defined quantity. Selecting the statistic after examining many outcomes introduces an avoidable selection problem.

19.8 Multiple Statistics, Multiple Questions

A code may be extreme under one statistic and ordinary under another. The correct form of a comparative conclusion is therefore: under null model N, observed statistic S occupies position P in the resulting distribution.

19.9 Figure 15 Is Reserved

Figure 15 is intentionally absent from this release. A null-distribution figure would be visually easy to produce but scientifically ambiguous without a rigorously specified comparison module. The missing figure is therefore a boundary marker, not an unfinished decorative slot.

Part XV
Verification

Chapter 20

Turning Mathematical Claims into Executable Checks

20.1 A Program Is Not Verified Merely Because It Runs

Successful execution does not guarantee a correct scientific result. A plausible figure can be generated from stale data, a regression value can be hard-coded, or a malformed input can silently propagate. Important claims therefore need explicit conditions that can fail.

20.2 Structural Checks

The verifier checks complete input cardinality, codon syntax, output classes, matrix shape, row sums, column sums, and other schema-level invariants before trusting downstream results.

20.3 Algebraic Invariants

Exact checks include rank 10, right nullity 2, the Gram spectrum, the two-value pseudoinverse, all four Moore–Penrose identities, projector rank and trace, idempotence, and the Hamming-distance entry rule.

20.4 Golden Regression Checks

Stable release quantities such as the G–A–C–U score 44, positional decomposition 4+1+39, complete loop profile, anisotropic distribution moments, and number of maximizers are retained as regression expectations. Refactoring must not change them silently.

20.5 Independent Cross-Checks

The strongest checks reconstruct the same result by different routes: SVD versus exact pseudoinverse, quotient loops versus fiber-pair adjacency, explicit cube scoring versus pair weights, and direct anisotropic enumeration versus convolution.

20.6 Numerical Consistency

Floating-point comparisons are kept distinct from exact identities. The release reports maximum SVD-versus-exact pseudoinverse difference around 3.6×10^{-16} and largest numerical Moore–Penrose residual around 6.7×10^{-16}.

20.7 Exact Arithmetic

Rational objects such as $5/96$, $-1/96$, and the four projector values are represented exactly. Exact arithmetic removes tolerance choices from claims that do not require floating-point approximation.

20.8 Tolerances

When numerical tolerances are necessary, they belong to the verification specification and should be chosen to distinguish ordinary numerical roundoff from meaningful disagreement—not widened simply to convert a failure into a pass.

20.9 Independent Pseudoinverse Verification

The pseudoinverse is checked by dimensions, exact entry values, binary placement pattern, four defining identities, equality with a generic symbolic pseudoinverse, and agreement with numerical SVD.

20.10 Independent Loop Reconstruction

The total of forty-four loops is reconstructed from global lattice edges and separately from pairs inside translated fibers. The class-specific loop profile must sum to the same total.

20.11 Independent Anisotropic Verification

The anisotropic verifier checks the search cardinality 13,824, histogram sum, extrema, moments, symmetry, number of maximizers, and agreement with position-score convolution.

20.12 Classification of Checks

Verification checks are classified as structural, exact mathematical, numerical consistency, golden regression, independent cross-check, or artifact integrity checks. The number of checks is less important than their coverage of distinct failure modes.

20.13 Fail-Closed Verification

The release build computes and verifies in staging before promoting artifacts. A failing invariant prevents promotion rather than leaving a mixture of old and new files in the canonical result directory.

20.14 Visual QA as a Separate Layer

Automated mathematics cannot certify all presentation details. Cropped labels, overlapping annotations, confusing loop geometry, or misleading perspective can occur even when the underlying data are correct. Visual review is therefore a separate release layer.

20.15 Test Suite and Release Verifier

Release v1.1.1 reports 30/30 software tests passing and 120/120 release verification checks passing. Those totals summarize rather than replace the detailed check descriptions.

20.16 Negative Tests

Malformed inputs must fail: missing codons, duplicate codons, illegal symbols, malformed property tables, unexpected schema structures, and inconsistent artifacts are all appropriate negative test cases.

20.17 Deterministic Rebuild Checks

Two independent clean builds are compared to expose hidden dependence on set iteration, unstable serialization, timestamps, or environment-sensitive ordering.

20.18 Provenance Checks

Source-tree hashes and data provenance connect generated results to the exact code and scientific inputs used by the release. Provenance establishes what was used; it does not by itself establish that an external source is scientifically correct.

20.19 Verification Is Not Peer Review

A verified implementation can still embody an uninteresting model or support an overextended interpretation. Verification strengthens review by making implementation claims auditable; it does not replace scientific judgment.

Invariant	Kind	Observed	Expected	Status
codon_count	structural invariant	64	64	PASS
translated_class_count	structural invariant	21	21	PASS
standard_degeneracy_profile	golden-value regression	{"A": 4, "C": 2, "D": 2, "E": 2...	{"A": 4, "C": 2, "D": 2, "E": 2...	PASS
matrix_shape	structural invariant	[64, 12]	[64, 12]	PASS
row_sums	structural invariant	["3"]	[3]	PASS
column_sums	structural invariant	["16"]	[16]	PASS
matrix_rank	structural invariant	10	10	PASS
right_nullity	structural invariant	2	2	PASS
gram_eigenvalues	structural invariant	{"0": 2, "16": 9, "48": 1}	{"0": 2, "16": 9, "48": 1}	PASS
pseudoinverse_entry_values	golden-value regression	["-1/96", "5/96"]	["-1/96", "5/96"]	PASS
closed_form_equals_generic_exact_pinv	independent computational cross-check	[["5/96", "5/96", "5/96", "5/96...	[["5/96", "5/96", "5/96", "5/96...	PASS
svd_vs_exact_pseudoinverse_max_abs	independent computational cross-check	3.5561831257524545e-16	0	PASS
moore_penrose_condition_1	structural invariant	True	True	PASS
moore_penrose_condition_2	structural invariant	True	True	PASS
moore_penrose_condition_3	structural invariant	True	True	PASS
moore_penrose_condition_4	structural invariant	True	True	PASS
projector_values_by_hamming_distance	golden-value regression	{"0": ["5/32"], "1": ["3/32"], ...	{"0": ["5/32"], "1": ["3/32"], ...	PASS
second_position_U_minus_A	golden-value regression	527/80	527/80	PASS
amino_acid_regression_r	numerical consistency check	0.9156459315641996	0.915645999999999 6	PASS
amino_acid_regression_r2	numerical consistency check	0.8384074719901946 1	0.8384070000000001	PASS
membrane_profile_example_pearson_r	golden-value regression	0.9431233078335483 6	0.9431233078335483 6	PASS
all_synonymous_hamming_one_pairs	golden-value regression	69	69	PASS
GACU_cube_edge_count	structural invariant	144	144	PASS
GACU_synonymous_edges	golden-value regression	44	44	PASS
GACU_nonsynonymous_edges	golden-value regression	100	100	PASS
GACU_positional_synonymous_decomposition	golden-value regression	[4, 1, 39]	[4, 1, 39]	PASS
loop_total	golden-value regression	44	44	PASS
loop_multiplicities_independent_fiber_pairs	independent computational cross-check	{"A": 3, "C": 1, "D": 1, "E": 1...	{"A": 3, "C": 1, "D": 1, "E": 1...	PASS
isotropic_oriented_path_count	structural invariant	24	24	PASS
isotropic_maximum	golden-value regression	44	44	PASS
isotropic_unique_maximum_up_to_reversal	golden-value regression	[["G", "A", "C", "U"]]	[["G", "A", "C", "U"]]	PASS
anisotropic_case_count	structural invariant	13824	13824	PASS
anisotropic_distribution_independent_cube_enumeration	independent computational cross-check	{"25": 192, "26": 384, "27": 96...	{"25": 192, "26": 384, "27": 96...	PASS
anisotropic_maximizer_count_oriented	structural invariant	192	192	PASS
anisotropic_undirected_maximizers	golden-value regression	24	24	PASS
unique_common_undirected_path	golden-value regression	[["G", "A", "C", "U"]]	[["G", "A", "C", "U"]]	PASS
ALL REQUIRED CHECKS	**summary**	**120**	**120**	**PASS**

Table 9. Verification summary and check classification

Part XVI
Reproducible Execution

Chapter 21

Fail-Closed Scientific Builds

21.1 From Script Execution to Release Construction

A scientific release chains data loading, calculation, rendering, serialization, verification, and packaging. If files are written directly into the canonical results directory as the process runs, a crash can leave a mixture of versions. The build therefore needs transactional behavior.

21.2 The Fail-Closed Principle

A new build is created in an isolated staging directory. Required mathematical and artifact checks run there. Promotion occurs only after successful verification. Failure leaves the last verified release intact.

21.3 Staging

Starting from an empty staging directory also prevents stale artifact contamination. If a later release intentionally removes a figure or table, that file cannot survive merely because it remained in an old output directory.

21.4 Verification Before Rendering

Core mathematical objects should be validated before expensive or presentation-oriented rendering. There is no reason to generate fourteen figures from a matrix whose shape or rank already violates required invariants.

21.5 Rendering Is Part of the Build

Presentation files are release artifacts too. A plotting function returning successfully does not certify that labels fit, axes are meaningful, or a graph communicates the claimed geometry. Rendering belongs inside the staged build and visual QA belongs after it.

21.6 Reverification After Artifact Generation

After generation, the staged artifact set is checked for expected files, lossless machine-readable siblings, manifests, hashes, and absence of stale outputs. Only then is it eligible for promotion.

21.7 Atomic Promotion

The intended semantics are old verified release or new verified release, not an intermediate mixture. Directory renaming on the same file system can provide near-atomic promotion when implemented carefully.

21.8 Failures Must Propagate

Critical verification errors must stop the build. Catching exceptions merely to print a warning converts a release blocker into a cosmetic message and defeats fail-closed design.

21.9 Warnings Versus Blockers

Optional dependency-version notices may be warnings. A 63-codon table, failed pseudoinverse identity, wrong search cardinality, or manifest hash mismatch should be a release blocker. The distinction must be explicit.

21.10 The Repository as a Scientific Instrument

A useful analogy is an instrument that refuses to certify output when its internal standards fail. Known inputs, declared algorithms, calibration-like invariants, provenance, failure conditions, and recorded outputs collectively turn the repository into a small computational instrument rather than a folder of scripts.

Chapter 22

Deterministic Builds

22.1 Why Determinism Matters

If equivalent scientific runs reorder rows, graph edges, JSON keys, or nodes arbitrarily, file diffs become noisy and hashes change without scientific cause. Determinism makes meaningful changes easier to isolate.

22.2 Canonical Codon Ordering

The codon set is mathematically unordered, but matrices and files are not. One canonical enumeration is reused for lookup exports, matrix rows, property vectors, projector indices, and figure labels.

22.3 Do Not Depend on Set Iteration

Sets should be sorted or converted into domain-specific canonical orders before serialization or rendering. Incidental hash-table order must not determine presentation.

22.4 Canonical Undirected Edges

An undirected edge is normalized so that (a,b) and (b,a) cannot appear as separate representations. The same principle is applied to translated multigraph pairs, while loops retain identical endpoints.

22.5 Deterministic Graph Layout

When geometry is mathematically meaningful, coordinates should come from that geometry rather than from a stochastic force-directed layout. The $4 \times 4 \times 4$ codon lattice therefore uses fixed coordinates derived from path levels.

22.6 Stable Search Enumeration

The twenty-four path permutations and 13,824 axis triples are visited in a canonical order. Histograms do not depend on this order mathematically, but stable raw files become easy to compare line by line.

22.7 Stable Serialization

JSON keys, list order, CSV columns, row order, numeric formatting, and newline conventions are fixed. Machine-readable data underlying visual tables remain complete even if a PNG is abbreviated for readability.

22.8 Timestamps and Randomness

Wall-clock timestamps are separated from scientific content where possible. Future sampled null models must record generator, seed, sample count, and sampling rule because random state becomes scientific input.

22.9 Two Clean Builds

The release performs two independent clean builds and compares deterministic artifacts. Starting from empty staging directories tests reconstruction from source rather than accidental reuse of cached or stale files.

22.10 Byte Versus Semantic Determinism

Some containers can differ in irrelevant bytes because of packaging or encoder metadata. The project distinguishes files expected to be byte-identical from artifacts for which semantic comparison is appropriate, rather than treating all differences as equivalent.

Chapter 23

Provenance, Manifests, and the Chain from Source to Figure

23.1 Reproducing a Number Is Not Enough

A reader who regenerates 44 still needs to know which genetic-code table, software version, graph convention, and source files produced that result. Provenance connects source to computation to released artifact.

23.2 Source Provenance

The repository separately records provenance for the standard RNA code, Kyte–Doolittle values, and reaction-center L-subunit example. This keeps bibliographic origin distinct from computational implementation.

23.3 Local Representation

Scientific source and local canonical file are different objects. The repository converts DNA T notation to RNA U where needed and represents stop as X. Those transformations are part of the provenance record.

23.4 Hashing Source Inputs and Code

Cryptographic hashes identify the exact bytes of data and source code used to construct a release. This is stronger than filenames or version labels alone.

23.5 Artifact Hashes

Generated CSV, JSON, verification reports, and presentation artifacts can likewise be hashed. A post-build manual edit to an image changes its hash and is therefore detectable.

23.6 Manifest as a Release Map

A useful manifest records relationships as well as checksums: which computation produced Figure 4, which source object underlies Table 7, or which verification record generated Table 9.

23.7 Figure and Table Provenance

The preferred chain is source data → computation → machine-readable result → renderer → PNG. A reader should never need to recover scientific data by reading pixels from a figure.

23.8 Provenance of Conventions

Not every input is a file. Choices such as X for stop, $H(X)=0$, linear rather than cyclic nucleotide paths, undirected edges, and a nineteen-residue window are configuration inputs and should be documented alongside source files.

23.9 Distribution Artifacts and Licensing

The source-tree repository and release ZIP are the canonical reproducibility objects. The wheel may contain only reusable library modules. Software is MIT-licensed; original documentation and generated presentation artifacts are CC BY4.0.

23.10 Reproducibility as a Graph

The project itself can be viewed as a directed acyclic graph from sources through canonical data, mathematical objects, verified results, machine-readable artifacts, and presentation files. Provenance is the structure that connects those transformations.

Part XVII
Generalization

Chapter 24

Other Numerical Properties

24.1 Hydropathy Is One Instance

Nothing in D or D^+ is specific to hydropathy. Any real-valued field y on the sixty-four codons can be projected by $\beta = D^+ y$ and reconstructed by $\hat{y} = DD^+ y$.

24.2 Properties Attached to Amino Acids

A numerical amino-acid property can be lifted through translation to codon space, provided a complete-domain convention is declared for X when necessary. Side-chain volume, mass, polarity, charge-related scales, or other declared properties can then be analyzed in the same coordinate system.

24.3 Common Twelve-Coordinate Representation

Every property produces coefficients indexed by the same twelve position-nucleotide coordinates. This allows direct comparison of which codon positions and nucleotide contrasts dominate different properties.

24.4 Gauge-Invariant Contrasts

Within-position differences remain invariant under blockwise constant offsets and are natural quantities for cross-property comparison.

24.5 Projection Fraction and Residual

The fraction $||Py||^2/||y||^2$ measures how much of a field lies in the additive subspace. The residual $(I-P)y$ can be studied rather than treated as meaningless error; it contains interaction structure unavailable to the first-order model.

24.6 Extending the Design Matrix

Pairwise and three-way interaction coordinates can be added systematically. Doing so changes the question from independent nucleotide-position effects to higher-order codon interactions, so the hierarchy should be explicit rather than mixed into one large model.

24.7 Exact Factorial Decomposition

For the $4 \times 4 \times 4$ complete word space, the sixty-four dimensions decompose as 1 constant + 9 first-order + 27 pairwise + 27 three-way interaction dimensions. The current matrix spans exactly the first ten dimensions; its fifty-four-dimensional orthogonal complement is the pairwise-plus-three-way space.

Chapter 25

Other Genetic Codes

25.1 The Incidence Matrix Does Not Change

Any complete translation table on the same sixty-four codons uses the same D, rank, singular spectrum, exact pseudoinverse, and Hamming projector. Those are input-space properties, not translation-assignment properties.

25.2 Translation-Dependent Quantities

Changing the lookup table can alter fibers, degeneracies, synonymous edges, quotient loops, optimal nucleotide paths, hydropathy fields, and search distributions. These form a second layer sitting on top of the universal input algebra.

25.3 Loading Another Complete Code

A different code can use the same two-column schema codon, output. Once its completeness is validated, existing fiber, graph, search, and property-projection routines can be reused without rewriting the core mathematics.

25.4 Comparative Atlas

A larger project could compute a common invariant set for many documented, synthetic, or sampled complete codes: fiber profiles, pair weights, isotropic and anisotropic scores, loop multiplicities, maximizer families, and property projections. The result would be an exact comparative atlas rather than a set of unrelated case studies.

Chapter 26

Complete Alphabet–Word Systems

26.1 General Definition

Let a be alphabet size and w word length. The complete input space contains a^w words. A positional one-hot encoding has aw coordinates, so the incidence matrix D(a,w) has dimensions a^w × aw. The genetic code is the case a=4, w=3.

26.2 Row and Column Counts

Every row contains w ones. Every position-symbol column contains a^(w−1) ones because the remaining w−1 positions are free. Two coordinates from different positions overlap in a^(w−2) words, while distinct coordinates within one position never overlap.

26.3 Rank and Nullity

$$\text{rank } D(a,w) = w(a-1) + 1 \qquad \text{right nullity} = w-1$$

All w positional block sums equal the same all-ones vector, leaving w−1 independent block-offset redundancies.

26.4 General Gram and Singular Spectra

The Gram eigenvalues are w a^(w−1) with multiplicity one, a^(w−1) with multiplicity w(a−1), and zero with multiplicity w−1. The corresponding singular values are the square roots of the two nonzero eigenvalues plus the null multiplicity.

26.5 General Exact Pseudoinverse

$$D(a,w)^{+} = a^{(1-w)}\, D(a,w)^{T} - [(w-1)/(w\, a^{w})]\, J$$

This is the generalized a,w equation. For a=4 and w=3, the coefficients become 1/16 and 1/96, reproducing the exact genetic-code pseudoinverse.

26.6 Two General Entry Values

Because D^{T} is binary, the general pseudoinverse is always two-valued. Present coordinates receive [wa−w+1]/(w a^w); absent coordinates receive −(w−1)/(w a^w).

26.7 General Hamming Projector

$$P(a,w)(x,y) = [a(w-d(x,y)) - (w-1)] \,/\, a^{w}$$

The complete word-space projector depends only on Hamming distance. For a=4,w=3 it specializes to the four values 5/32, 3/32, 1/32, and −1/32.

26.8 General Factorial Interaction Decomposition

Interaction order k contributes C(w,k)(a−1)^k dimensions. Summing k=0 through w gives a^w by the binomial theorem. The additive positional model consists of k=0 and k=1, with dimension 1+w(a−1), exactly the rank of D(a,w).

26.9 Lookup Functions Enter After Input Algebra

Once the universal word-space algebra is defined, a lookup function f:Σ^w→Ω can be placed on the words. Its fibers, quotient graphs, property fields, and path optimizations are lookup-specific layers built on the same input geometry.

Part XVIII
Conclusions

Chapter 27

The Genetic Code as Executable Mathematics

27.1 A Complete Lookup Table

The genetic code is finite enough to resist vagueness. All sixty-four inputs can be written down and validated. Translation can be represented as a complete deterministic map into twenty-one output classes.

27.2 Exact Input Algebra

One-hot positional encoding produces a 64 × 12 matrix of rank ten with exact Gram spectrum $48\times1, 16\times9, 0\times2$ and singular spectrum $4\sqrt{3}\times1, 4\times9, 0\times2$.

27.3 Exact Pseudoinverse and Projector

$$D^{+} = (1/16)D^{T} - (1/96)J$$

$$P(a,b) = [5 - 2d(a,b)]/32$$

The 768-entry pseudoinverse reduces to two rational values, and the 4,096-entry projector reduces to four Hamming-distance cases.

27.4 Hydropathy

Kyte–Doolittle hydropathy projected onto nucleotide-position coordinates reveals a dominant second-position U−A contrast of 527/80=6.5875. The additive codon model captures approximately 0.816882 of squared field magnitude, and fiber-averaged reconstructed amino-acid values correlate with the empirical scale at $r\approx0.915646$.

27.5 Middle-Position Compression

A deliberately compressed U=+1, A=−1, C=G=0 second-position score produces a nineteen-codon moving-window profile that correlates with the conventional Kyte–Doolittle profile at $r\approx0.9431$ for the deposited example sequence. This is a reproducible sequence-specific example rather than a universal performance claim.

27.6 Graph Geometry

The G–A–C–U 4 × 4 × 4 lattice has sixty-four vertices and 144 edges. Forty-four preserve translation identity and become forty-four self-loop instances when synonymous codons are collapsed into a twenty-one-vertex quotient multigraph.

27.7 Exhaustive Search

All twenty-four oriented isotropic paths and all 13,824 oriented anisotropic axis triples can be evaluated exactly. The global synonymous-edge maximum is 44. The anisotropic maximum family contains 192 oriented or 24 undirected cubes, while isotropy reduces the solution to one undirected path: G–A–C–U.

27.8 Exact Versus Conditional Claims

Matrix identities are universal properties of the complete 4^3 word space. Hydropathy results depend on declared property values and the X convention. The profile correlation depends on a particular sequence and window. Graph

maxima are exact within the declared linear-path adjacency model. Future null-model claims will depend on their comparison spaces. Keeping these logical categories separate is part of the scientific result.

27.9 Executability

The repository does not merely illustrate mathematical statements made elsewhere. It constructs the finite objects, exposes invariants, checks them by independent routes, records provenance, and refuses promotion when required relations fail. The same sixty-four-entry beginning becomes a lookup table, matrix, projector, property field, lattice, quotient multigraph, and complete finite search space.

References

Kyte J, Doolittle RF. A simple method for displaying the hydropathic character of a protein. Journal of Molecular Biology. 1982;157(1):105–132. PMID 7108955. DOI: 10.1016/0022-2836(82)90515-0.

NCBI Taxonomy. Genetic Codes: The Standard Code (translation table 1). The repository canonicalizes the table to RNA U notation and uses X for stop.

DDBJ/EMBL/GenBank accession Z11165.1. Rhodobacter capsulatus, CDS 41735..42583, gene pufL, protein ID CAA77554.1. Used as the deposited reaction-center L-subunit coding-sequence example after DNA T to RNA U normalization.

Executable companion and full source provenance: https://github.com/DougYouvan/mathematics-of-the-genetic-code-in-a-python-program

Part Appendices
Technical and Reproducibility Material

Appendix A

Installation and Execution

A.1 Source Distribution

The source-tree distribution is the canonical computational object associated with the book. It contains the code, authoritative data, tests, release verifier, artifact generators, machine-readable outputs, documentation, and provenance needed for full reproduction.

A.2 Basic Workflow

```
python -m venv .venv
# activate the environment
python -m pip install -r requirements.txt
python run_verify.py
python -m unittest discover -s tests -v
```

Release v1.1.1 is tested across Python 3.10 through 3.13. A successful verification run ends with all required invariants passing and regenerates the fourteen figures and nine tables plus machine-readable siblings.

A.3 Clean Build and Determinism

For strongest reproduction, build from clean staging state and perform two independent clean builds. Compare deterministic artifacts rather than relying on an output directory that may contain stale files from an earlier run.

A.4 PNG Files Are Not Proof

A directory full of images is not evidence that the mathematics passed verification. The intended order is validate inputs → compute → verify → render → verify artifacts → promote.

Appendix B

Canonical Standard Genetic-Code Dataset

B.1 Domain and Output Alphabet

The canonical input contains all 64 three-letter RNA words over G,A,C,U. The output set contains the standard twenty amino-acid one-letter symbols plus X for stop.

B.2 Validation

The loader rejects missing or duplicate codons, invalid symbols, incorrect codon lengths, malformed output symbols, and unexpected schema structure. The normalized mapping is then ordered canonically for matrices and vectors.

B.3 Fibers

$$F\omega = \{ c : \mathrm{code}(c) = \omega \}$$

The fibers form a disjoint partition of codon space and support both degeneracy analysis and the later graph quotient.

B.4 Data Provenance

The repository records the Standard Code as NCBI translation table 1, converted from DNA T notation to RNA U notation and using X for stop.

Appendix C

Construction of the Complete Incidence Matrix

C.1 Coordinate Order

1G 1A 1C 1U | 2G 2A 2C 2U | 3G 3A 3C 3U

Each codon activates exactly one coordinate in each block. The 64 rows therefore contain three ones apiece; every column contains sixteen ones.

C.2 Gram Matrix

D^TD has diagonal 16, within-block off-diagonal 0, and cross-block entries 4. The three positional block sums are identical, producing two independent right-null relations and rank ten.

C.3 Exact Null Directions

$$(1,1,1,1,-1,-1,-1,-1,0,0,0,0)$$

$$(1,1,1,1,0,0,0,0,-1,-1,-1,-1)$$

Appendix D

Gram Matrix, Rank, and Singular-Value Checks

D.1 Exact Spectrum

$$D^{T}D: 48 \times 1, 16 \times 9, 0 \times 2$$

$$D: 4\sqrt{3} \times 1, 4 \times 9, 0 \times 2$$

Exact eigenvalues remove numerical rank ambiguity, while numerical SVD supplies an independent generic computation. The two routes should remain distinct in the software.

Appendix E

Numerical SVD and the Exact Pseudoinverse

E.1 Numerical Discovery

Numerical SVD yields a 12×64 pseudoinverse whose entries cluster at $0.052083333333\ldots$ and $-0.010416666667\ldots$, suggesting $5/96$ and $-1/96$.

E.2 Exact Genetic-Code Formula

$$D^{+} = (1/16)D^{T} - (1/96)J$$

E.3 Exact Verification

$$DD^{+}D=D \quad D^{+}DD^{+}=D^{+} \quad (DD^{+})^{T}=DD^{+} \quad (D^{+}D)^{T}=D^{+}D$$

The maximum SVD-versus-exact elementwise difference is approximately 3.6×10^{-16}. The largest numerical Moore–Penrose residual is approximately 6.7×10^{-16}.

Appendix F

The Codon-Space Projector

F.1 Closed Form

$$P = DD^{+} = (1/16)DD^{T} - (1/32)J_{64}$$

F.2 Hamming Kernel

$$P(a,b) = [5 - 2d(a,b)]/32$$

Distances 0,1,2,3 map to $5/32$, $3/32$, $1/32$, $-1/32$. Hamming-sphere populations are 1,9,27,27. P has rank ten, trace ten, $P^2=P$, and $P^T=P$.

Appendix G

Hydropathy Projection

G.1 Exact Coefficients

Position 1: G 659/960, A -613/960, C -1201/960, U 169/192
Position 2: G -1411/960, A -2533/960, C -157/960, U 3791/960
Position 3: G -373/960, A -163/960, C 113/960, U 113/960

G.2 Central Contrast

$$2U - 2A = 527/80 = 6.5875$$

G.3 Reconstruction Statistics

projection fraction ≈ 0.816882; slope ≈ 0.859718; intercept ≈ −0.003660; r ≈ 0.915646; R^2 ≈ 0.838407

Appendix H

The Middle-Nucleotide Membrane Profile

H.1 Scoring Rule

$$\text{second-position U} \rightarrow +1;\ \text{A} \rightarrow -1;\ \text{C,G} \rightarrow 0$$

H.2 Window and Scaling

The example uses a nineteen-codon window. The raw nucleotide score is multiplied by 4.5 only for plotting, giving a theoretical display range of −4.5 to +4.5.

H.3 Example Result

$$r \approx 0.9431;\ R^2 \approx 0.8895$$

The result is tied to the deposited pufL coding sequence, declared reading frame, score rule, and window width.

Appendix I

Exhaustive Isotropic Path Search

I.1 Search Space and Score

There are 24 oriented or 12 undirected linear nucleotide paths. Every candidate produces 144 codon edges. The score counts synonymous edges.

I.2 Exact Result

minimum = 25; maximum = 44; unique undirected maximizer = G–A–C–U

I.3 Pair-Weight Cross-Check

Total unordered pair weights are A–G=15, A–C=11, A–U=9, C–G=8, C–U=18, G–U=8. G–A–C–U selects 15+11+18=44.

Appendix J

Exhaustive Anisotropic Search

J.1 Search Space

$$24^3 = 13{,}824 \text{ oriented axis triples}$$

J.2 Separable Score

$$S(p_1,p_2,p_3) = S_1(p_1)+S_2(p_2)+S_3(p_3)$$

J.3 Exact Distribution

```
25:192  26:384  27:960  28:1536  29:960
30:384  31:384  32:192  33:960   34:960
35:960  36:960  37:192  38:384   39:384
40:960  41:1536 42:960  43:384   44:192
```

J.4 Moments and Maxima

mean 34.5; variance 30.25; standard deviation 5.5; 192 oriented maxima; 24 undirected maxima

Appendix K

Quotient Multigraph and Self-Loop Reconstruction

K.1 Quotient Totals

21 vertices; 44 loop instances; 100 interclass edge instances; 144 total

K.2 Loop Profile

```
R6 L6 S4
A3 G3 P3 T3 V3
I2 X2
C1 D1 E1 F1 H1 K1 N1 Q1 Y1
M0 W0
```

K.3 Independent Reconstruction

The loop count is obtained both by quotienting the global lattice edge set and by enumerating adjacent pairs independently within translated fibers.

K.4 Rendering Principle

The machine-readable multiplicity object is authoritative. Visual loop geometry may be improved without changing the verified multiplicities.

Appendix L

Verification Matrix

L.1 Structural and Exact Checks

The verifier covers complete codon input, matrix invariants, exact pseudoinverse identities, projector rules, hydropathy coefficients, graph counts, exhaustive search cardinalities, and maximum-family results.

L.2 General a,w Formula Checks

$$D(a,w)^{+} = a^{(1-w)}\, D(a,w)^{T} - [(w-1)/(w\, a^{w})]J$$

The general formula is tested in several small exact finite cases and explicitly specialized at a=4,w=3 to the genetic-code formula.

L.3 Negative, Artifact, and Determinism Checks

Malformed scientific inputs must fail. Expected release files must be complete. Two clean builds must agree for deterministic artifacts. Visual QA remains a separate layer because not every layout defect is machine-detectable.

Appendix M

Release Provenance and Reproducibility Record

M.1 Release Identity

The computational release associated with this manuscript is v1.1.1, released 2026-08-08. Version labels identify lineage; source and artifact hashes identify exact bytes.

M.2 Scientific Inputs

Principal source inputs are the Standard RNA genetic-code table, Kyte–Doolittle hydropathy values, and the reaction-center L-subunit coding-sequence example. Human-readable provenance is kept separate from analytical code.

M.3 Figure and Table Provenance

Every presentation artifact should be traceable backward through a machine-readable result and scientific computation to source inputs. Presentation changes and scientific-data changes should therefore be distinguishable in release history.

M.4 Licensing

Appendix N

General Exact Mathematics for Alphabet Size a and Word Size w

N.1 Complete Word Incidence Matrix

$$D(a,w) \text{ has dimensions } a^w \times aw$$

Every row has w ones. Every column has a^(w−1) ones. Cross-position overlaps equal a^(w−2).

N.2 Rank and Spectrum

$$\text{rank } D(a,w)=w(a-1)+1; \text{ right nullity}=w-1$$

$$\text{Gram eigenvalues: } w\,a^{(w-1)} \times 1;\ a^{(w-1)} \times w(a-1);\ 0 \times (w-1)$$

N.3 General Exact Moore–Penrose Pseudoinverse

$$D(a,w)^{+} = a^{(1-w)}\, D(a,w)^{T} - [(w-1)/(w\,a^w)]\, J$$

This is the generalized a,w equation. It should be regarded as the central closed form for the complete positional word-incidence family.

N.4 Genetic-Code Specialization

$$a=4,\ w=3 \ \Rightarrow\ D^{+}=(1/16)D^{T}-(1/96)J$$

N.5 General Two Entry Values

$$\text{present coordinate: } [wa-w+1]/(w\,a^w) \qquad \text{absent coordinate: } -(w-1)/(w\,a^w)$$

N.6 General Word-Space Projector

$$P(a,w)(x,y) = [a(w-d(x,y)) - (w-1)] \,/\, a^w$$

N.7 General Interaction Decomposition

$$\text{dimension at interaction order } k = C(w,k)(a-1)^k$$

Summing all interaction orders gives a^w. The additive model consists of the constant and first-order terms and has dimension 1+w(a−1), exactly the rank of D(a,w).

Appendix O

Figure and Table Reproduction Map

O.1 Reproduction Principle

No scientific value should exist only as pixels. Every figure or table should be traceable to numerical, categorical, or graph data that can be inspected without image interpretation. Conversely, a visual correction should not require changing verified mathematics unless the correction reveals a scientific defect.

O.2 Figures

Figure 1: Output-class degeneracy histogram — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 2: 64 × 12 binary incidence matrix with translated output strip — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 3: Singular-value spectrum of the complete incidence matrix — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 4: Exact two-value Moore–Penrose pseudoinverse — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 5: Exact codon-space projector value versus Hamming distance — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 6: Twelve nucleotide-position hydropathy coefficients — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 7: Empirical versus reconstructed amino-acid hydropathy — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 8: Kyte–Doolittle and second-position U−A profiles — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 9: Translated G–A–C–U 4 × 4 × 4 codon lattice — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 10: Twenty-one-vertex translated quotient multigraph — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 11: Self-loop multiplicity by translated output — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 12: Synonymous-edge scores for all 24 oriented isotropic nucleotide orders — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 13: Exact score distribution over all 13,824 oriented anisotropic cubes — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 14: Intersection of position-specific maximizing families at G–A–C–U — generated from the corresponding verified result object in results/ figures/ with a CSV or JSON sibling where appropriate.

Figure 15: intentionally reserved for a future rigorously specified null-model comparison.

O.3 Tables

Table 1: Complete standard RNA lookup table — generated from the corresponding complete machine-readable table object in results/tables/.

Table 2: Output classes, degeneracies, and codon membership — generated from the corresponding complete machine-readable table object in results/tables/.

Table 3: Structural invariants of the complete codon-incidence matrix — generated from the corresponding complete machine-readable table object in results/tables/.

Table 4: Exact pseudoinverse rules, numerical-SVD agreement, and projector values — generated from the corresponding complete machine-readable table object in results/tables/.

Table 5: Complete 3 × 4 nucleotide-position hydropathy coefficient matrix — generated from the corresponding complete machine-readable table object in results/tables/.

Table 6: Exhaustive enumeration of the 24 oriented isotropic nucleotide orders — generated from the corresponding complete machine-readable table object in results/tables/.

Table 7: Position-specific synonymous-edge weights for all six unordered nucleotide pairs — generated from the corresponding complete machine-readable table object in results/tables/.

Table 8: Position-specific maximum families and the 44-edge global bound — generated from the corresponding complete machine-readable table object in results/tables/.

Table 9: Verification summary and check classification — generated from the corresponding complete machine-readable table object in results/tables/.

O.4 Accepted Presentation Imperfections in Frozen v1.1.1

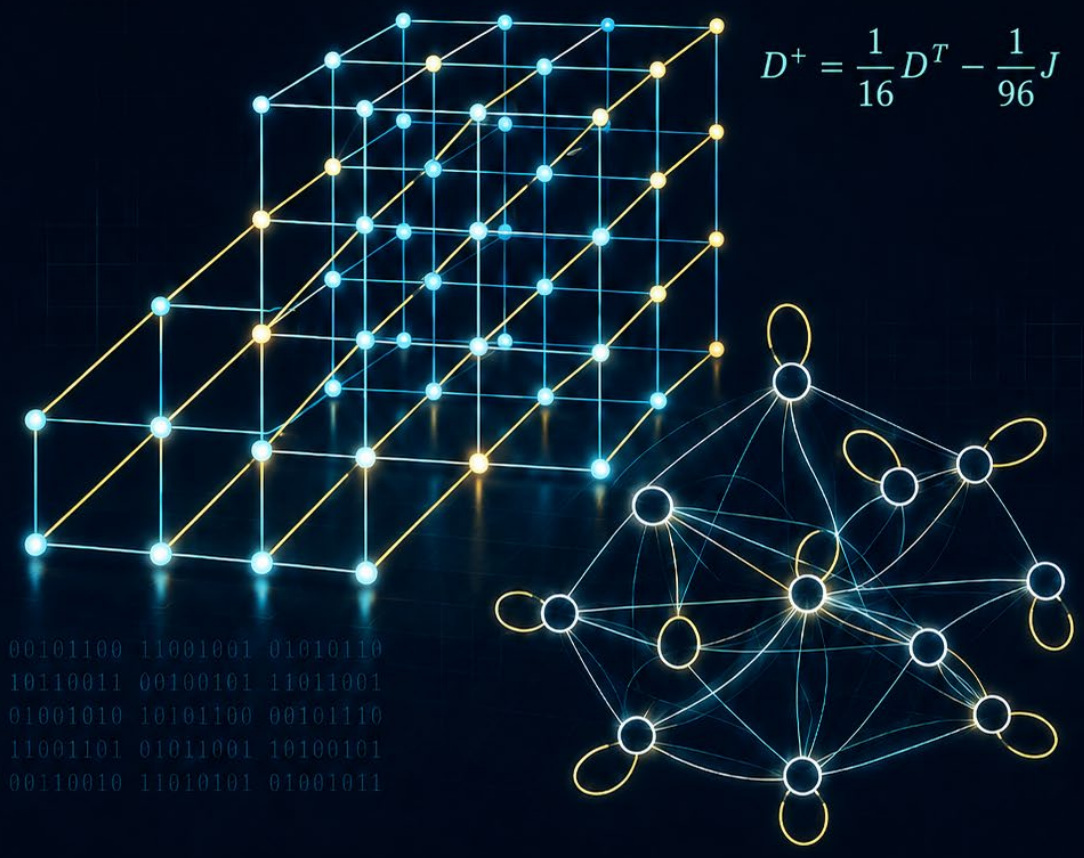
The v1.1.1 release is frozen because the remaining observed issues are presentation-level and do not alter the verified mathematics. Figure 5 places extreme value labels close to plot boundaries. In Figure 10, multiple self-loop lobes around high-loop vertices can visually merge into an outer ring. Figure 9 is retained as drawn: it is a structurally correct 4 × 4 × 4 oblique lattice projection.

O.5 Repository

Public GitHub repository: https://github.com/DougYouvan/mathematics-of-the-genetic-code-in-a-python-program

The Genetic Code as Executable Mathematics

From a 64-Entry Lookup Table to Exact Algebra, Graphs, and Exhaustive Search in Python



Douglas C. Youvan