The Transform Is Another Trajectory

Mathematical Inscription, Computational Unfolding, and Geometry in Finite Symbolic Mechanics

Kevin R. Haylett

Manchester, UK
July 2026

Initial status. This monograph is a first full exposition of a developing argument within Finite Symbolic Mechanics. It remains open to refinement, experimental testing, and extension. Classical mathematical results are retained; the proposed change concerns their finite symbolic and computational interpretation.

Abstract

A matrix is usually presented as a completed mathematical object: a rectangular arrangement of numbers that may transform vectors, encode simultaneous equations, or be decomposed into factors. The compactness of the notation is so effective that the object appears to act in a single timeless gesture. Yet no actual matrix calculation occurs in this way. Every entry must be finitely represented, every operation must be ordered, and every result must arise through a sequence of computational states. The matrix on the page is therefore not the calculation. It is a finite symbolic inscription capable of initiating one or more computational unfoldings.

This monograph traces the historical process by which procedures for solving coupled relations became compressed into tabular inscriptions, then stabilised as matrices, and finally established as primary instruments within classical mathematics. It then follows the reciprocal history of computing, in which matrix objects had to be unfolded again into finite arithmetic, storage, instruction order, rounding, convergence tests, and halting conditions. The history reveals a double movement: operations generated the matrix, and the matrix later generated operations.

The central argument is developed through the discrete cosine transform and singular value decomposition. Both are conventionally described as transforms applied to data or matrices. From the perspective of Finite Symbolic Mechanics, neither transform occupies a privileged position outside the calculation. A DCT coefficient is the terminal registration of an ordered sequence of finite products and accumulations. An SVD is likewise produced through an algorithmic trajectory of finite rotations, reflections, reductions, comparisons, sweeps, or iterative updates. Thus the central proposition is:

The transform is not applied to the trajectory. The transform is another trajectory.

The execution histories of matrix operations and transforms may themselves be observed as finite time series. Takens-style delay coordinates can then bring temporally separated registrations into spatial coexistence. In classical reconstruction these coordinates are represented as exact points. In Finite Symbolic Mechanics, each coordinate is a finite, uncertainty-bearing registration. The reconstructed trajectory is therefore better represented as an ordered geometry of finite spheres or regions whose overlap and separation express distinguishability at the chosen resolution.

This approach does not discard matrix algebra, DCT, SVD, equality, or invariance. It repositions them. Matrix objects become stabilised symbolic residues of finite trajectories; equality becomes a compression of provenance; and invariance becomes measured stability across admissible finite unfoldings. The result is a coherent account of how mathematical

objects are inscribed, unfolded, transformed, reconstructed, and restabilised within finite computation.

Preface: The Opening

The immediate opening for this work came from a simple explanation of singular value decomposition. The explanation presented a matrix not only as a table but as an action: a matrix can be treated as something that rotates, stretches, and rotates again. It also presented singular values as an ordered account of which parts of a matrix matter most for a chosen reconstruction. This familiar description is valuable because it makes an apparently formidable technique accessible. It also opens a more foundational question.

When we write

$$A = U\Sigma V^T, \tag{1}$$

what has actually happened?

On the page, the relation is spatial and simultaneous. The four symbols appear together. The equality sign removes any visible passage of time. The reader can move from the matrix A to the factors U, Σ, and V^T without seeing a single multiplication, memory access, comparison, rounding event, convergence test, or stopping decision. The notation has done precisely what strong notation is designed to do: it has removed the labour.

But the absence of visible labour is not the absence of a process. To calculate an SVD, the inscription A must be represented in some finite medium. An algorithm must act upon that representation. Intermediate states must be produced. Some criterion must determine when the process has stabilised sufficiently to halt. Only then do the terminal inscriptions U, Σ, and V^T appear.

The first insight, then, is straightforward: the matrix must be unfolded to calculate with it. The deeper insight follows one step later. It is easy to acknowledge the unfolding of the original matrix while allowing the transform itself to remain a classical primary object. In that picture, A is admitted to require a finite execution, but SVD is still imagined as an external mathematical operation that acts upon A from outside time. That position is not stable within Finite Symbolic Mechanics. The SVD matrix relations must also be unfolded. The transform does not stand outside the computational landscape. It is another finite process within that landscape.

The same applies to the DCT, to a sum, to an inverse, to a derivative as numerically evaluated, to a factorisation, and to the construction of a delay embedding. Every operation that appears complete in notation must become a sequence when instantiated. The argument is therefore not merely about one decomposition. SVD and DCT are unusually clear windows into the general relation between mathematical inscription and computational becoming.

This monograph begins historically because the history helps prevent a false impression. Matrices did not arrive fully formed as eternal objects waiting to be discovered. Long before matrix theory, people arranged coefficients and carried out ordered procedures for

solving coupled problems. The modern matrix emerged gradually from these practices, from determinant theory, from linear substitutions, and from the need to compress many related equations into a manageable notation. Once stabilised, the matrix became so powerful that its procedural ancestry could disappear from view. Electronic computing later forced the matrix back into sequences of finite operations, but numerical practice often treated this unfolding as implementation beneath the real mathematics rather than as the means by which the relation becomes measurable.

Finite Symbolic Mechanicsreverses that hierarchy. It begins with finite symbols, finite distinctions, and finite transitions. It does not deny the extraordinary value of mathematical compression. It asks us to retain the provenance that compression removes.

A Guide for Readers New to FSM

No prior knowledge of Finite Symbolic Mechanicsis assumed. The framework begins from a small number of commitments.

First, a symbol that is written, stored, transmitted, or computed is finite. It occupies some distinguishable region in a medium. It has a resolution below which further differences are not registered within the chosen apparatus. A printed digit, a voltage state, a binary word, a decimal coefficient, and a coordinate in memory all exist as finite registrations.

Second, a mathematical expression is not identical to the complete process required to produce, interpret, or use it. Expressions compress trajectories. The expression

$$\sum_{i=1}^{n} x_i$$

occupies a small region of the page, but an actual calculation requires an ordered sequence of additions or an alternative structured reduction. The notation presents the work as though it were simultaneously complete.

Third, computation proceeds through distinguishable states. A computational state can be represented as

$$C_0 \to C_1 \to \cdots \to C_T.$$

The arrow need not mean smooth physical time. It may denote instruction order, arithmetic operation number, memory update, algorithmic iteration, or another chosen computational clock. What matters is that the result does not appear without an ordered finite unfolding.

Fourth, a registered numerical value is not treated as an infinitely precise point. It is a finite distinction with a representational extent. In a geometrical model, that extent may be shown as a small region or sphere around the registered coordinate. This is not a claim that every bit is literally a physical sphere. The sphere is a geometrical representation of finite distinguishability and uncertainty.

Fifth, a stable mathematical object can be understood as the compressed residue of repeatable trajectories. This does not make the object arbitrary. It places its stability in repeated finite production rather than in an assumed existence outside all measurement and representation.

The argument will repeatedly distinguish a classical reading from an FSM reading. The distinction is not intended as a contest in which one account must erase the other. Classical mathematics supplies extraordinarily effective compressions. FSM asks what those compressions require when they are written, computed, measured, and reconstructed through finite means.

Contents

Part I

The Historical Formation of the Matrix

Chapter 1

Before the Matrix: Coupled Relations and Ordered Procedures

1.1 Problems before objects

A history that begins with the word *matrix* begins too late. People encountered coupled relations long before they possessed a general algebraic object for representing them. A field might yield several quantities of grain. A set of trades might connect several prices. Astronomical observations might impose simultaneous conditions on unknown parameters. Surveying, taxation, mechanics, and commerce repeatedly produced problems in which several quantities had to be determined together.

The primary difficulty was operational. One relation could not be settled independently because changing one unknown affected the others. The solver therefore needed a procedure: arrange the known quantities, compare rows or columns of relations, eliminate one unknown, carry the altered values forward, and continue until a solution could be recovered.

This distinction matters. The problem was not initially perceived as an instance of multiplication by a matrix. It was experienced as a need to coordinate several finite operations without losing the relations among them. The procedure preceded the bounded object.

The eighth chapter of the Chinese mathematical classic commonly known as *The Nine Chapters on the Mathematical Art* contains the *fangcheng* procedure, in which coefficients associated with simultaneous problems were arranged on a counting surface and transformed through elimination. Historians caution against simply calling these ancient arrangements matrices in the modern sense. The caution is useful. A modern label can conceal the historical difference. What we find is not nineteenth-century matrix algebra appearing two millennia early, but a sophisticated visual and operational practice for maintaining coupled relations through ordered transformations [7, 6].

The counting surface is already a striking object for the present argument. Quantities are placed spatially so that several relations can be inspected together. Yet the solution does not emerge from the arrangement by itself. Counters are moved, quantities are multiplied or subtracted, and new arrangements replace old ones. Spatial inscription and temporal procedure are already interdependent.

The earliest matrix-like practices did not begin from a timeless array. They began from a problem, a finite arrangement, and a sequence of operations.

1.2 Elimination as a trajectory

Consider three coupled equations:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1, \tag{1.1}$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2, \tag{1.2}$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3. \tag{1.3}$$

A modern reader is trained to see a matrix equation

$$Ax = b. \tag{1.4}$$

But the compact form is a later stabilisation. Operationally, elimination creates a trajectory. One row is scaled, another row is altered, a coefficient becomes zero, the changed system is inspected, and the process repeats. The state after the first elimination is not the same as the state before it. The solution depends upon preserving the provenance of each transformation.

We can write the procedure schematically as

$$E_0 \xrightarrow{\phi_0} E_1 \xrightarrow{\phi_1} E_2 \xrightarrow{\phi_2} \cdots \xrightarrow{\phi_{T-1}} E_T, \tag{1.5}$$

where E_t is the complete finite arrangement after the tth operational step. The terminal arrangement may be triangular, diagonal, or otherwise convenient for extracting the unknowns. What later becomes "row reduction" is therefore a family of admissible finite trajectories across tabular states.

Elimination developed in multiple historical settings. Grcar's history shows that what is now called Gaussian elimination accumulated from several traditions, not from a single moment of invention. The name itself became standard only after a long evolution of methods, notation, hand computation, least-squares work, and later machine-oriented presentation [6]. The lesson is not merely that credit is historically distributed. It is that a procedure can persist, migrate, and be renamed while its operational character is progressively compressed.

1.3 Determinants before matrices

Determinants developed as devices associated with systems of equations, elimination, and transformations before matrices were treated as autonomous algebraic quantities. This order is important. Many relations that modern textbooks introduce as properties of matrices were first investigated through determinants, quadratic forms, and linear substitutions. The array did not initially possess the full objecthood it now carries.

A determinant can itself be read as a compression of a large combinatorial operation. For

an $n \times n$ array $A = (a_{ij})$, the Leibniz formula is

$$\det(A) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} a_{i,\pi(i)}. \tag{1.6}$$

On the page, this is one expression. In calculation, it refers to many products, sign assignments, and additions. Even before the matrix was granted independent algebraic status, the tendency of notation to spatially compress temporal symbolic labour was already present.

1.4 The visual surface and the moving procedure

The historical surface has two functions. First, it holds several relations in a form that can be inspected together. Second, it acts as a working field across which the procedure unfolds. These functions are easily conflated. A completed table suggests that the relations are simultaneously present. A worked table shows that the relations are continually changed.

This duality will remain throughout the monograph:

$$\text{spatial inscription} \quad \longleftrightarrow \quad \text{temporal operation.} \tag{1.7}$$

The matrix later intensifies the spatial side of this relation. Computation later intensifies the temporal side. FSM holds both together.

Chapter 2

Naming and Objectification: Sylvester and Cayley

2.1 The word *matrix*

James Joseph Sylvester introduced the term *matrix* in 1850. His metaphor was generative. An oblong arrangement of terms did not itself have to be a determinant; it could be regarded as a matrix out of which systems of determinants might be formed. The word gave a bounded identity to an arrangement that had previously been handled through several neighbouring practices. Higham's historical account notes both Sylvester's 1850 coinage and the fact that determinant theory was already well developed before Cayley's systematic matrix algebra [8, 11].

Naming is not a trivial event. A name allows an inscription to be transported as a unit. It lets writers refer to the whole arrangement without rewriting its contents or recounting the procedure that produced it. The name creates a new symbolic handle.

In FSM terms, this is an act of stabilisation. A family of arrangements and uses is compressed into a bounded term. Once the term circulates successfully, later readers can begin from the object rather than from the historical operations that made the object useful.

2.2 Cayley's algebraic step

Arthur Cayley's 1858 *Memoir on the Theory of Matrices* is widely treated as the first systematic paper on matrix algebra. Cayley states that the notion of a matrix arises naturally from abbreviated notation for systems of linear equations, then develops rules for treating matrices as quantities with addition, multiplication, powers, identity, zero, and inverse relations [2, 8].

The phrase "arises naturally from an abbreviated notation" is especially important. It records the passage from relations to compression. A system such as

$$y_1 = a_{11}x_1 + \cdots + a_{1n}x_n, \tag{2.1}$$

$$\vdots \tag{2.2}$$

$$y_m = a_{m1}x_1 + \cdots + a_{mn}x_n \tag{2.3}$$

can be abbreviated as

$$y = Ax. \tag{2.4}$$

The abbreviation does not merely save ink. It makes composition visible:

$$z = By = B(Ax) = (BA)x. \tag{2.5}$$

A long family of substitutions is stabilised as matrix multiplication. The matrix therefore becomes both a noun and an operational promise.

2.3 From abbreviation to autonomous object

Once rules are established for adding, multiplying, inverting, and powering matrices, the array no longer appears to be merely a convenient table. It becomes a mathematical quantity in its own right. One can ask about a matrix's characteristic polynomial, rank, eigenvalues, canonical forms, and functions. One can study matrices without continually returning to the systems of equations from which the notation arose.

This is a productive act of symbolic objectification:

$$\text{relations and procedures} \longrightarrow \text{abbreviated array} \longrightarrow \text{autonomous algebraic object.} \tag{2.6}$$

The change is not an error. It is one of the great successes of mathematical compression. By suppressing the complete operational ancestry of the matrix, mathematicians gained a portable instrument that could move among geometry, mechanics, differential equations, statistics, electrical theory, quantum mechanics, and many other fields.

The danger appears only when the compression is mistaken for the entire phenomenon. The matrix can then seem primary, while the finite practices that produce and use it are treated as secondary implementation.

2.4 The first inversion

The nineteenth-century development can be described as the first inversion:

$$\boxed{\text{procedures and transformations} \longrightarrow \text{matrix object}} \tag{2.7}$$

What began as a means of coordinating operations became a bounded entity from which operations could be defined. The object now appeared capable of generating the very transformations from which it had historically emerged.

This inversion is the historical precondition for the later statement that a matrix "acts" on a vector. The statement is mathematically effective, but it compresses a large change in viewpoint. The relation is no longer narrated as a sequence of substitutions. It is attributed to the object A.

Chapter 3

The Matrix as a Classical Instrument

3.1 Why matrices became indispensable

The matrix became central because it solves several symbolic problems at once. It collects coupled coefficients into a single visible surface. It permits transformation and composition to be written compactly. It separates a general operational form from any one particular vector. It allows the same symbolic object to be interpreted geometrically, algebraically, statistically, or computationally.

For a linear map from $\mathbb{R}^n$ to $\mathbb{R}^m$, the matrix

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \tag{3.1}$$

compresses m output relations involving n input components. The notation makes several structural properties available at once. Columns describe the images of basis vectors. Rows describe linear functionals applied to the input. Multiplication composes transformations. The transpose reverses the row-column relation. Eigenvectors identify directions preserved up to scale. Factorisations expose alternative coordinate relations.

This density of use made the matrix a primary instrument in the classical mathematical basin. It became not merely one notation among others but a standard region through which diverse problems could be routed.

3.2 The illusion of simultaneous action

The power of the notation also creates an illusion. When we write

$$y = Ax, \tag{3.2}$$

we speak as though A acts on x in one completed event. The output appears to follow from a timeless relation. Yet even a single component

$$y_i = \sum_{j=1}^{n} a_{ij} x_j \tag{3.3}$$

requires an unfolding when calculated.

A simple sequential implementation might produce partial states

$$s_i^{(0)} = 0, \tag{3.4}$$
$$s_i^{(1)} = Q_B(a_{i1} x_1), \tag{3.5}$$
$$s_i^{(2)} = Q_B\Big(s_i^{(1)} + Q_B(a_{i2} x_2)\Big), \tag{3.6}$$
$$\vdots \tag{3.7}$$
$$s_i^{(n)} = y_{i,B}, \tag{3.8}$$

where Q_B represents finite registration under a numerical base, word length, and rounding rule. The classical equation suppresses the sequence

$$s_i^{(0)} \to s_i^{(1)} \to \cdots \to s_i^{(n)}. \tag{3.9}$$

A parallel implementation produces a different internal order. Products may be formed simultaneously, combined through a reduction tree, fused in hardware, or accumulated in blocks. The same surface inscription can therefore initiate multiple computational trajectories.

3.3 Human calculation and intermediate surfaces

Before electronic computing, matrix calculations were already temporal. A human calculator copied values, formed products, altered rows, checked signs, and moved through intermediate tables. The work might be distributed across teams of people whose role was explicitly called "computer". A completed matrix in a textbook gave no direct view of the labour required to produce its inverse or eigenvalues.

The historical matrix thus existed in two modes:

1. as a visible, simultaneous inscription;
2. as a recipe or constraint for an ordered calculation.

The first mode encouraged objecthood. The second remained present in practice but could be omitted from the finished exposition.

3.4 The matrix as a transport instrument

The matrix's importance also arose from its ability to transport relations across domains. A covariance matrix in statistics, an inertia tensor in mechanics, a transition matrix in probability, a Hamiltonian matrix in quantum theory, and a connectivity matrix in a network

are not identical objects in use. Yet the same algebraic operations can be brought to bear upon them.

This transportability produces a powerful stabilising basin. Once a phenomenon has been represented as a matrix, a large inherited apparatus becomes available. But every representation also filters. To place a phenomenon into a matrix is to decide which quantities become entries, which order they occupy, which relationships count as rows or columns, and which distinctions are suppressed.

FSM therefore treats matrix formation as a measurement and inscription process, not as a neutral discovery of a pre-existing array.

Chapter 4

The Matrix Meets the Electronic Computer

4.1 The second inversion

The electronic computer did more than accelerate matrix arithmetic. It forced the compact object back into an explicit sequence of finite operations. A machine cannot ingest an ideal matrix. It receives finite encodings of entries, addresses, instructions, and control states. It must decide where values are stored, in what order they are accessed, how arithmetic is rounded, and when an iterative process halts.

The second inversion is therefore

$$\boxed{\text{matrix object} \longrightarrow \text{finite procedures and transformations}} \tag{4.1}$$

The complete historical cycle becomes

$$\text{procedure} \to \text{inscription} \to \text{object} \to \text{machine instruction} \to \text{procedure}. \tag{4.2}$$

4.2 Rounding enters the foreground

In pure algebra, equivalent rearrangements may be treated as exact. In finite arithmetic, order can matter. Suppose a machine represents numbers with limited precision. The two expressions

$$(a+b)+c \qquad \text{and} \qquad a+(b+c) \tag{4.3}$$

can register different terminal values after rounding. The difference is not an accidental blemish placed upon an otherwise complete calculation. It is part of the actual finite trajectory.

The early literature of numerical analysis confronted this directly. Von Neumann and Goldstine studied the numerical inversion of high-order matrices in 1947, placing error propagation and machine calculation near the centre of the problem [14]. Turing's 1948 paper, *Rounding-Off Errors in Matrix Processes*, examined finite errors in methods for solving linear systems and inverting matrices, including elimination methods [13]. These works belong to the formative period in which electronic computation made it impossible to

ignore the difference between an algebraic relation and its finite execution.

Turing's title is revealing. The subject is not merely matrix objects but matrix *processes*. The matrix must pass through operations whose numerical behaviour can be studied. The classical object has become a trajectory again.

4.3 Condition, stability, and provenance

Numerical analysis developed concepts such as conditioning and stability to distinguish several sources of difficulty. A problem may be sensitive to small changes in its input. An algorithm may amplify or control finite errors. A computed residual may be small even when the forward error is large, or vice versa. These distinctions are essential and remain fully valuable within FSM.

FSM adds a wider interpretive claim. The numerical result is not an approximation to an independently instantiated infinite object in any directly measurable sense. It is the registered outcome of a finite process under declared conditions. Its provenance includes

$$(A_B, \mathcal{A}, B, p, O, H), \tag{4.4}$$

where A_B is the finite input inscription, $\mathcal{A}$ the algorithm, B the numerical base, p the precision, O the operation order or architecture, and H the halting condition.

A result can be stable, reproducible, and extraordinarily useful. Its strength lies in the measured persistence of the relation across admissible variations, not in the removal of all finite provenance.

4.4 Memory order and the return of sequence

A matrix is displayed as a two-dimensional surface, but computer memory is addressed through an ordered structure. Row-major and column-major layouts already produce different access trajectories. Blocking and cache-aware algorithms rearrange the order of work to reduce data movement. Parallel architectures distribute submatrices across processors and later combine partial results.

This does not mean that the matrix ceases to be a useful surface. It means that the surface is repeatedly serialised, partitioned, routed, and restabilised through a machine-specific sequence. The matrix on the page is not the same thing as the matrix in motion.

Chapter 5

Factorisation and the Computational Substrate

5.1 Factorisation as compressed architecture

A matrix factorisation is usually written as a short relation:

$$A = LU, \tag{5.1}$$

$$A = QR, \tag{5.2}$$

$$A = U\Sigma V^T. \tag{5.3}$$

Each relation compresses an architecture of calculation. A difficult problem is reorganised into factors with useful structure: triangular systems, orthogonal transformations, diagonal scales, or low-rank approximations. In the computer age, such factorisations became central instruments of numerical linear algebra because they enable stable and reusable computational pathways.

The arrow

$$A \longrightarrow LU \tag{5.4}$$

is not one operation. It stands for row operations, pivot decisions, updates, and storage changes. Likewise,

$$A \longrightarrow QR \tag{5.5}$$

may be produced through Gram–Schmidt variants, Householder reflections, or Givens rotations, each with a different internal trajectory.

The decomposition notation presents the end relation. The algorithm supplies the path.

5.2 SVD becomes computable

The singular value decomposition existed as a mathematical structure before it became a routine numerical instrument. A major computational development came with Golub and Kahan's 1965 scheme, which first reduces a matrix to bidiagonal form and then diagonalises the reduced matrix. Their paper explicitly presents a numerically stable and reasonably fast procedure for producing the decomposition [5].

This history is central to the present thesis. The equation

$$A = U\Sigma V^T \tag{5.6}$$

is not sufficient to calculate U, Σ, and V^T. A further finite symbolic trajectory is required. The computational history of SVD is the history of finding workable unfoldings for a compact structural relation.

5.3 BLAS and the operational vocabulary of machines

As scientific computing matured, recurring vector and matrix operations were standardised into reusable software primitives. The Basic Linear Algebra Subprograms collected operations such as dot products, scaling, vector updates, norms, copying, swapping, and plane rotations; later levels extended this organisation to matrix-vector and matrix-matrix operations [10, 3].

This standardisation is historically profound. Matrix operations moved from being descriptions in mathematical texts to becoming part of the operational vocabulary of computational systems. A programmer could invoke a matrix multiplication routine without specifying every lower-level arithmetic step. The software interface became a new compression layer.

The trajectory was not removed. It was displaced into a library, then into compilers, kernels, vector units, and hardware. Each layer stabilised a higher-level symbol by assuming reconstructive competence below it.

5.4 The matrix-computing feedback loop

Matrices shaped computing, and computing reshaped matrix mathematics. Hardware encouraged blocked algorithms. Memory hierarchies changed preferred operation order. Parallel machines favoured decompositions that could be distributed. Numerical analysis developed backward-error viewpoints and condition estimates in response to finite machine behaviour. Matrix libraries, in turn, influenced the design and benchmarking of scientific computers.

The relation is therefore cross-generative:

$$\text{matrix mathematics} \rightleftarrows \text{computational architecture.} \tag{5.7}$$

The matrix became both a classical symbolic instrument and a computational substrate. This joint history prepares the central FSM question: when the matrix and its transforms are treated as trajectories, what geometry do their finite unfoldings produce?

Part II

Finite Symbolic Foundations

Chapter 6

The Finite Symbol and the Inscription

6.1 Beginning from the registered mark

Finite Symbolic Mechanicsbegins from the finite symbol. A symbol that participates in an actual calculation must be registered somewhere: on paper, in a memory cell, as an electrical state, in an optical pattern, or through another distinguishable medium. It has a finite extent and a finite resolution. Its identity depends upon a practical boundary between what is admitted as the same mark and what is registered as different.

The finite symbol is therefore not an abstract type floating outside all inscription. The type is itself a later compression across sufficiently similar finite tokens. A digit 7 written by hand, printed in a book, and stored as a character code may be treated as the same symbol for a particular purpose, but that sameness is an admissible relation established across distinct physical and historical trajectories.

For numerical work, a symbol also belongs to a representational system. The finite word

$$1.2345 \tag{6.1}$$

is not merely a transparent window onto a real number. It is a finite base-ten inscription with five digits after the leading digit. A binary floating-point word is a different finite inscription, with a different spacing of representable values and a different arithmetic trajectory.

6.2 The Alphonic limit

Within the broader Geofinite framework, the *Alphonic limit* names the smallest distinction that the chosen symbolic apparatus can register as a separate mark. The term does not assert one universal hardware size. It denotes the active lower boundary of distinguishability for a given inscription and measurement regime.

If two attempted values fall within the same registered region, the apparatus produces one symbol rather than two. The output is not an infinitely precise point with an attached error bar. It is the finite distinction the apparatus can make.

We may represent a scalar registration x_B as a region

$$\mathcal{S}(x_B, \delta), \tag{6.2}$$

where δ expresses the local finite extent or uncertainty under the chosen representation. In one dimension this region may be an interval. In an isotropic geometric illustration it may be shown as a sphere. The sphere is a model of finite symbolic extent, not a claim that all uncertainty is physically spherical.

6.3 Inscription, interpretation, execution, registration

A central discipline of FSM is to separate stages that classical notation often merges. For a matrix expression, we distinguish:

1. **Inscription**: the finite arrangement of symbols, such as A or $A = U\Sigma V^T$.
2. **Interpretation**: the rules by which a reader or machine assigns operational meaning to the inscription.
3. **Execution**: the ordered finite trajectory generated under those rules.
4. **Registration**: the terminal or intermediate symbolic result admitted by the apparatus.

These stages can be written as

$$I_B \xrightarrow{\mathcal{G}} \mathcal{T} \xrightarrow{H} R_B, \tag{6.3}$$

where I_B is the finite inscription, $\mathcal{G}$ the operative grammar or algorithm, $\mathcal{T}$ the generated trajectory, H a halting or registration condition, and R_B the resulting finite inscription.

The same inscription may initiate different trajectories under different grammars. Conversely, different inscriptions may be treated as equivalent because they repeatedly lead to sufficiently similar registered results.

6.4 A mathematical object as a stabilised residue

FSM does not abolish mathematical objects. It explains their stability through finite practice. A matrix object can be treated as a stabilised residue of many trajectories of writing, teaching, reading, storing, transforming, and comparing. Its identity persists because a community and its instruments repeatedly reconstruct sufficiently compatible uses.

Schematically,

$$O_B = \text{Stabilise}(\mathcal{T} \mid B, p, H, \mathcal{C}), \tag{6.4}$$

where $\mathcal{C}$ includes the admissibility and consensus rules under which the result is accepted as an instance of the object.

This is not social relativism. A machine can fail, a calculation can be inconsistent, and a result can be experimentally rejected. Admissibility is constrained by finite interactions, repeated measurements, and the capacity of trajectories to cohere. The claim is that objecthood is maintained through these finite relations rather than guaranteed by a view from outside them.

Chapter 7

Mathematical Notation Compresses Temporal Labour

7.1 The quiet work of notation

Mathematical notation is one of humanity's most effective compression systems. A short inscription can hold an instruction for a vast amount of symbolic work. The summation

$$S = \sum_{i=1}^{n} x_i \tag{7.1}$$

places the initial value, index range, repeated operation, and terminal symbol in one spatial arrangement. It omits the explicit states

$$S_0 = 0, \tag{7.2}$$
$$S_1 = S_0 + x_1, \tag{7.3}$$
$$S_2 = S_1 + x_2, \tag{7.4}$$
$$\vdots \tag{7.5}$$
$$S_n = S_{n-1} + x_n. \tag{7.6}$$

Likewise, an integral sign compresses partitioning, evaluation, accumulation, and limiting commitments. An inverse symbol A^{-1} compresses the conditions under which an inverse is admitted and the process by which it might be calculated. A derivative symbol compresses a local relational construction and, when evaluated numerically, a finite difference or other computational scheme.

The notation is powerful precisely because it allows the reader to skip the full trajectory. Shared training supplies the missing unfolding.

Mathematical notation spatially compresses temporal symbolic labour.

7.2 Calculation reverses the compression

To execute the expression, a person or machine must reverse enough of the compression to generate an admissible trajectory. The movement is

$$\text{spatial inscription} \longrightarrow \text{temporal unfolding}. \tag{7.7}$$

When the calculation halts, the trajectory is compressed again into a terminal inscription:

$$\text{temporal unfolding} \longrightarrow \text{registered result}. \tag{7.8}$$

The broader cycle is

$$\boxed{\text{trajectory} \to \text{inscription} \to \text{trajectory} \to \text{inscription}}. \tag{7.9}$$

This cycle is not confined to arithmetic. A proof compresses many possible acts of interpretation into a finite text. A theorem name compresses a proof and its historical basin. A software function call compresses a library implementation. A chart compresses a data-processing trajectory. Every successful symbolic surface relies upon recoverable but usually hidden temporal work.

7.3 Compression removes provenance

Compression necessarily removes detail. The expression

$$y = Ax \tag{7.10}$$

does not tell us whether the multiplication was performed row by row, column by column, in blocks, with fused multiply-add instructions, on a GPU, or by hand. It does not display the base, precision, rounding mode, memory layout, or error checks. These may be irrelevant for one purpose and decisive for another.

Thus compression can be written schematically as

$$\mathcal{T} \longrightarrow \chi, \tag{7.11}$$

where the compact symbol χ retains selected relations while discarding visible trajectory. To use χ, the reader must supply an admissible reconstruction from within a shared symbolic basin.

This is why the same notation can be both exact in a formal system and incomplete as a description of a finite calculation. Formal exactness concerns relations among inscriptions under declared rules. Computational completeness would require the provenance of an actual unfolding.

7.4 The reader is part of the reconstruction

A matrix equation is not self-executing on the page. A trained reader brings conventions concerning rows, columns, multiplication, transpose, equality, and numerical interpretation.

A compiler brings another set of conventions. A hardware device brings yet another.

The reader is therefore not outside the inscription. Reading is a trajectory that reconstructs operational relationships from finite marks. The apparent simultaneity of the page is converted into a sequence of attention, interpretation, and inference.

This observation will matter when delay embeddings are introduced. A reconstructed geometry is not simply "there" in a time series. It is produced through another finite procedure, interpreted by another trajectory, and stabilised through a chosen visual or numerical representation.

Chapter 8

Finite Computational States and Computational Time

8.1 The state sequence

Let C_t denote the registered computational state after step t. A general execution is

$$C_0 \xrightarrow{\phi_0} C_1 \xrightarrow{\phi_1} \cdots \xrightarrow{\phi_{T-1}} C_T. \tag{8.1}$$

The state may include

$$C_t = (R_t, M_t, I_t, P_t, E_t), \tag{8.2}$$

where R_t denotes active registers, M_t memory values, I_t the instruction or control position, P_t partial results, and E_t accumulated finite discrepancy or diagnostic state.

No experiment will usually record the entire machine state. Instead, an observable is selected:

$$s_t = h(C_t). \tag{8.3}$$

This produces a scalar or low-dimensional sequence

$$s_0, s_1, \ldots, s_T. \tag{8.4}$$

The choice of h is a measurement decision. One observable may reveal periodic structure while another does not. The resulting geometry is conditional upon what has been registered.

8.2 Several computational clocks

The index t is not automatically physical time. It may count:

- arithmetic operations;
- completed loop iterations;
- matrix sweeps;
- memory writes;
- processor clock cycles;

- wall-clock samples;
- algorithmic checkpoints;
- symbolic changes above a chosen resolution.

These clocks are not interchangeable. A recurrence every eight arithmetic updates is a frequency in operations, not necessarily a frequency in hertz. A parallel process may perform many operations during one wall-clock interval. A blocked algorithm may have a natural local clock within a larger sweep clock.

The computational clock must therefore be included in the provenance of any time-series analysis.

8.3 Serialisation and partial order

Not every computation has one natural total order. Parallel processes may generate a partial order: some events depend upon earlier events, while others occur independently. A recorded sequence imposes a serialisation on that partial order. Different serialisations can preserve the final dependency structure while producing different one-dimensional observations.

FSM does not require that all computation be metaphysically serial. It requires that any finite observation or inscription register an order or a set of ordered relations. A delay reconstruction from a parallel calculation must therefore state how the events were sampled or linearised.

8.4 Finite transitions

The transition

$$C_t \to C_{t+1} \tag{8.5}$$

is not a movement between exact points. Each state is a finite registered region. If two successive states differ below the active resolution, they may be recorded as the same symbol. If they cross a distinguishability boundary, a new state is registered.

We can therefore define a simple symbolic change measure

$$\Delta_t = d_B(C_{t+1}, C_t), \tag{8.6}$$

where d_B is a finite, representation-dependent distance or difference count. Possible choices include a norm of changed numerical values, the number of altered bits or digits, a residual update, or a weighted count of changed registers.

This change sequence can itself become the observable for phase-space reconstruction.

Chapter 9

From Exact Points to Finite Spheres

9.1 The classical point

Classical state-space descriptions place a system at a point

$$z_t \in \mathbb{R}^m. \tag{9.1}$$

The point has no extent. It is distinguished from every neighbouring point, however close. This idealisation is mathematically productive, but no finite instrument registers an infinitely precise coordinate.

A recorded state is instead a finite symbolic value. Under FSM, the coordinate is written as a region of admissible distinction:

$$z_{t,B} \rightsquigarrow \mathcal{Z}_t. \tag{9.2}$$

In one dimension $\mathcal{Z}_t$ may be an interval. In two dimensions it may be a cell or disc. In three dimensions an isotropic model may be represented as a sphere.

9.2 The sphere as a distinguishability region

Let the registered centre be $\hat{z}_t \in \mathbb{R}^3$ and the finite radius be $r_t > 0$. Then

$$\mathcal{Z}_t = \{z : \|z - \hat{z}_t\| \leq r_t\}. \tag{9.3}$$

The radius can encode numerical resolution, measurement uncertainty, model discrepancy, or a combination declared by the investigator. The choice must be explicit. The sphere is not an ontological replacement for the point. It is a finite geometric representation of what the apparatus can distinguish.

The trajectory becomes

$$\mathcal{Z}_0 \to \mathcal{Z}_1 \to \cdots \to \mathcal{Z}_T. \tag{9.4}$$

Rather than an infinitely thin curve, we obtain an ordered packing of finite regions.

A finite computational trajectory may be represented as a temporally ordered necklace of uncertainty-bearing symbolic spheres.

9.3 Overlap and separation

The overlap between successive regions carries information. If

$$\mathcal{Z}_t \cap \mathcal{Z}_{t+1} \neq \varnothing, \tag{9.5}$$

then the two states are not fully separated at the chosen resolution. This does not mean that nothing changed physically. It means that the registered distinction remains within overlapping finite extent.

If

$$\mathcal{Z}_t \cap \mathcal{Z}_{t+1} = \varnothing, \tag{9.6}$$

then a new distinguishable region has been traversed. The trajectory has crossed a finite symbolic boundary.

This supplies a geometry for computational stability. A convergent algorithm may enter a dense region of overlapping states. An oscillatory algorithm may cycle among separated regions. A rounding-induced stall may produce repeated inscriptions even while lower-level physical activity continues.

9.4 Thickness, folding, and projection

A finite trajectory has thickness. When projected into two or three dimensions, distinct higher-dimensional states may overlap. A fold in the visual reconstruction can therefore mean several things: genuine recurrence in the observed variables, projection of separate states onto nearby coordinates, or insufficient resolution to distinguish them.

FSM requires that these possibilities remain attached to the interpretation. The visual geometry is a measured reconstruction, not an unqualified revelation of the system's final ontology.

Part III

The Transform as a Finite Trajectory

Chapter 10

The Discrete Cosine Transform: Basis, Base, and Execution

10.1 The classical DCT

The discrete cosine transform was introduced by Ahmed, Natarajan, and Rao in 1974 as a real orthogonal transform with efficient computational properties [1]. A common one-dimensional DCT-II is

$$c_k = \alpha_k \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], \qquad k = 0, \ldots, N-1. \tag{10.1}$$

The coefficient c_k expresses the projection of the finite sequence $x_0, \ldots, x_{N-1}$ onto a prescribed cosine pattern. The family of cosine patterns forms a basis under the chosen boundary conventions.

The DCT became especially important in digital signal and image processing because many natural signals concentrate substantial variation into relatively few low-frequency coefficients. This concentration makes selective storage and quantisation effective. Yet the compact formula conceals the finite trajectory required to produce every coefficient.

10.2 One coefficient is a sequence

For a fixed k, define

$$\theta_{kn} = \frac{\pi}{N}\left(n+\frac{1}{2}\right)k. \tag{10.2}$$

A finite implementation may form partial states

$$D_{k,0} = 0, \tag{10.3}$$

$$D_{k,j+1} = Q_B(D_{k,j} + Q_B(x_j Q_B(\cos\theta_{kj}))). \tag{10.4}$$

The terminal registration is

$$c_{k,B} = Q_B(\alpha_k D_{k,N}). \tag{10.5}$$

Thus one displayed coefficient compresses a sequence of generated or retrieved cosine values, products, accumulations, and registrations.

The complete DCT is a bundle of such trajectories:

$$\mathcal{T}_{\mathrm{DCT}} = \{\mathcal{T}_{\mathrm{DCT},0}, \ldots, \mathcal{T}_{\mathrm{DCT},N-1}\}. \tag{10.6}$$

An efficient fast algorithm changes the exact bundle, sharing intermediate results and reorganising the operation order. The terminal transform may remain sufficiently stable while the internal trajectory differs substantially.

10.3 Basis and base

The words *basis* and *base* must be separated.

The *basis* is the family of cosine trajectories used to measure the sequence. It defines a coordinate geometry:

$$x \longmapsto (c_0, c_1, \ldots, c_{N-1}). \tag{10.7}$$

The numerical *base* or radix is the finite symbolic system in which x_n, the cosine values, intermediate products, and coefficients are registered.

Conventional analysis often treats the basis as mathematically central and the numerical base as implementation. FSM retains both:

$$\text{DCT result} = \text{basis-dependent projection} + \text{base-dependent finite registration}. \tag{10.8}$$

Even the apparently fixed cosine basis is not present without a trajectory. The values may be stored in a finite table, generated recursively, approximated by library functions, or embedded in specialised hardware. The basis has provenance.

10.4 DCT as a measurement of recurrence

If the index n is physical sample time, k is interpreted in relation to temporal frequency. If n is pixel position, the coefficients describe spatial recurrence. If n counts computational steps, the DCT measures recurrence in the unfolding of an algorithm.

The units may then be cycles per operation, cycles per sweep, or cycles per block. A strong coefficient may indicate an alternating update, a nested loop rhythm, a repeated correction, or a regular memory-access pattern. This is an endogenous computational frequency, not automatically a physical frequency in hertz.

The DCT can therefore be applied at two levels:

1. to the data processed by an algorithm;

2. to an observable generated by the algorithm's own execution.

These levels must not be conflated. The transform of the input and the trajectory of the transform are distinct, though they can interact.

10.5 The DCT is not outside the line

A classical description says that the DCT maps a finite sequence into coefficient space. FSM adds that the mapping itself must unfold through another finite sequence. The correct relation is not simply

$$x \xrightarrow{\text{DCT}} c. \tag{10.9}$$

It is

$$I_{x,B} \longrightarrow D_0 \longrightarrow D_1 \longrightarrow \cdots \longrightarrow D_T \longrightarrow I_{c,B}. \tag{10.10}$$

The coefficient vector is the terminal stabilisation of the DCT execution trajectory.

Chapter 11

Singular Value Decomposition and the Removal of Privilege

11.1 The classical decomposition

For a real $m \times n$ matrix A, the singular value decomposition is conventionally written

$$A = U\Sigma V^T, \tag{11.1}$$

where U and V are orthogonal matrices and Σ is diagonal or rectangular diagonal with non-negative singular values

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0. \tag{11.2}$$

The decomposition can be interpreted geometrically as a change of coordinates, directional scaling, and a second change of coordinates. It also supports low-rank approximation:

$$A_k = U_k \Sigma_k V_k^T. \tag{11.3}$$

The Eckart–Young result establishes the optimality of the truncated SVD for low-rank approximation under standard norms [4].

This classical account is coherent and useful. The FSM argument begins not by denying it but by asking how the displayed factors come into finite existence.

11.2 The easy concession and the hidden return

It is easy to say that multiplying A by a vector requires finite operations. One can acknowledge the sequence of products and sums while continuing to treat the SVD as a primary transform that acts upon the matrix from outside the sequence. This leaves a hidden classical privilege in place.

The correction is decisive. The expression

$$A \longmapsto U\Sigma V^T \tag{11.4}$$

does not describe an indivisible event. An SVD algorithm begins from a finite inscription of

A and produces intermediate states:

$$S_0 \to S_1 \to \cdots \to S_T. \tag{11.5}$$

Only at a selected halting condition do the terminal inscriptions

$$U_B, \qquad \Sigma_B, \qquad V_B^T \tag{11.6}$$

appear.

A more explicit FSM relation is

$$A_B \xrightarrow[H]{\mathcal{A}_{\mathrm{SVD}}} (U_B, \Sigma_B, V_B^T), \tag{11.7}$$

followed by a finite reconstruction relation

$$A_B \sim_B U_B \Sigma_B V_B^T. \tag{11.8}$$

The notation $\sim_B$ is not intended to replace every classical equality sign. It marks the fact that an actual reconstruction is registered within a base and precision.

The transform is not applied to the trajectory. The transform is another trajectory.

11.3 The factors are terminal stabilisations

Classical language often says that SVD "reveals" directions latent inside A. FSM treats this phrasing carefully. The factors are not arbitrary, and their stability across methods is important. Yet the actual inscriptions are generated through a finite algorithm. They are terminal stabilisations of a reproducible relation involving input, method, precision, conventions, and stopping conditions.

This becomes especially clear when singular values are repeated or nearly repeated. The singular subspace may be stable while individual singular vectors vary under sign choices, rotations within a degenerate subspace, finite perturbations, or implementation conventions. The object that remains reproducible may be a relational region rather than one exact displayed vector.

Thus the SVD output is better written conditionally:

$$(U_B, \Sigma_B, V_B^T) \mid (A_B, \mathcal{A}, B, p, O, H). \tag{11.9}$$

The condition does not weaken the result. It supplies its provenance.

11.4 A trajectory producing a description of a trajectory

Suppose the matrix A was itself produced by another computation or by arranging measurements from a time series. Then there are at least two trajectories:

$$\mathcal{T}_A = (A_0, A_1, \ldots, A_N), \tag{11.10}$$

$$\mathcal{T}_{\mathrm{SVD}} = (S_0, S_1, \ldots, S_T). \tag{11.11}$$

The second trajectory produces a compact description that is interpreted as a property of the first inscription or of the data from which it arose.

The complete relation is therefore

$$\begin{aligned} \text{first finite trajectory} &\longrightarrow \text{matrix inscription} \\ &\longrightarrow \text{SVD trajectory} \\ &\longrightarrow \text{terminal factor inscriptions.} \end{aligned} \tag{11.12}$$

The measurement instrument is itself a dynamical finite symbolic process.

Chapter 12

SVD Algorithms as Families of Trajectories

12.1 There is no single SVD trajectory

The phrase "the SVD trajectory" is a useful shorthand but must not imply one compulsory path. Different algorithms and implementations produce different internal sequences. A Jacobi method may repeatedly apply plane rotations. A bidiagonalisation method may use Householder transformations followed by an iterative diagonalisation. A truncated or randomised method may seek only a dominant subspace. A GPU implementation may reorganise the same high-level method into blocks and parallel kernels.

We therefore have a family

$$\mathfrak{T}_{\mathrm{SVD}} = \{\mathcal{T}_{\mathrm{Jacobi}}, \mathcal{T}_{\mathrm{bidiag}}, \mathcal{T}_{\mathrm{iter}}, \mathcal{T}_{\mathrm{blocked}}, \ldots\}. \tag{12.1}$$

The trajectories may differ in length, local recurrence, memory movement, and rounding behaviour while producing terminal outputs that agree within declared tolerances.

12.2 A generic unfolding

Without committing to one implementation, an SVD execution may include stages such as:

1. load and register the matrix entries;
2. compute norms or scaling factors;
3. construct finite rotations or reflections;
4. update rows, columns, or blocks;
5. reduce the matrix toward bidiagonal or diagonal form;
6. repeat sweeps or iterations;
7. evaluate residuals or convergence criteria;
8. order the singular values;
9. apply sign or orientation conventions;

10. register and store the terminal factors.

Each stage contains subordinate trajectories. A norm calculation is a sum of squares and a square-root procedure. A Householder vector requires finite subtraction, scaling, and normalisation. A convergence test requires a threshold and comparison. The high-level transform is a nested hierarchy of unfoldings.

12.3 Nested phases

The nested structure may be represented as

$$\text{element operation} \subset \text{local update} \subset \text{sweep} \subset \text{iteration} \subset \text{halting trajectory}. \tag{12.2}$$

This hierarchy introduces several possible computational phases. A local index may cycle rapidly while a sweep index advances more slowly. An outer convergence loop may modulate both. When an observable mixes these phases, a delay reconstruction may produce circular, toroidal, or folded geometry.

The geometry is not a property of the abstract matrix alone. It is a property of an observed execution under a chosen algorithm and clock.

12.4 Measured stability instead of assumed invariance

Classical theory identifies invariant quantities and subspaces. FSM does not discard these relations, but it changes the order of commitment. Rather than assuming that the invariant is primary and the computation merely approximates it, FSM asks which relations remain stable across admissible finite unfoldings.

Let R_j be the terminal registration from trajectory $\mathcal{T}_j$. A relation P is measured as stable when

$$P(R_1), P(R_2), \ldots, P(R_m) \tag{12.3}$$

agree within the declared resolution and purpose. This yields the distinction

$$\boxed{\text{assumed invariance} \neq \text{measured stability across finite trajectories}}. \tag{12.4}$$

The measured stability may be exceptionally strong. Singular values computed by different high-quality algorithms may agree to many digits. FSM accepts that strength as evidence. It refuses only the removal of the finite trajectories through which the evidence was produced.

12.5 Algorithmic equivalence and trajectory difference

Two algorithms can be equivalent at the level of terminal formal relation while non-equivalent as finite trajectories. They may require different numbers of operations, traverse different intermediate magnitudes, accumulate different rounding residues, or respond differently to ill-conditioning.

This distinction is familiar in numerical analysis, but FSM generalises its significance. The internal trajectory is not merely an engineering route to the mathematical answer. It is

part of the finite meaning of the operation when the operation is instantiated.

Part IV

Reconstructing Computational Geometry

Chapter 13

From the Computational Line to Delay Coordinates

13.1 The one-dimensional observation

A complex computational process may be observed through a scalar sequence

$$s_0, s_1, \ldots, s_T, \tag{13.1}$$

where $s_t = h(C_t)$. The sequence is a one-dimensional projection of a higher-dimensional process. Information about the full state is compressed into the selected observable.

Takens' delay-embedding result showed, under appropriate smoothness and genericity conditions, that a state-space structure can be reconstructed from delayed observations of a scalar measurement [12]. The classical delay vector is

$$z_t = (s_t, s_{t-\tau}, \ldots, s_{t-(m-1)\tau}). \tag{13.2}$$

For $m = 2$,

$$z_t = (s_t, s_{t-\tau}), \tag{13.3}$$

and for $m = 3$,

$$z_t = (s_t, s_{t-\tau}, s_{t-2\tau}). \tag{13.4}$$

Temporally separated registrations are brought into coordinate coexistence.

13.2 What the theorem does and does not grant

The theorem does not say that any arbitrary finite sequence automatically reveals a unique hidden object. Its conditions concern a smooth dynamical system, a suitable observable, and an embedding dimension sufficient for the underlying attractor. Finite data, noise, quantisation, nonstationarity, and algorithmic discontinuities complicate practical reconstruction.

FSM therefore uses Takens-style delays as a measurement method rather than as a licence to claim perfect recovery. The key operation is still valuable:

$$\text{sequential difference} \longrightarrow \text{simultaneous relational coordinate}. \tag{13.5}$$

The reconstructed geometry is a finite artefact produced under a chosen delay, dimension, sampling clock, observable, and resolution.

13.3 Delay coordinates as finite regions

Classically, z_t is a point. FSM writes

$$z_{t,B} \rightsquigarrow \mathcal{Z}_t = \mathcal{S}(s_t) \times \mathcal{S}(s_{t-\tau}) \times \cdots . \tag{13.6}$$

In three dimensions, an isotropic approximation may be shown as a sphere centred at

$$\hat{z}_t = (\hat{s}_t, \hat{s}_{t-\tau}, \hat{s}_{t-2\tau}). \tag{13.7}$$

The reconstructed path is an ordered packing of finite regions, not an infinitely thin line.

This modification has immediate consequences. Close returns must be interpreted relative to sphere overlap. Apparent fixed points may be finite clusters. A thin torus may disappear when the sphere radius exceeds the separation between neighbouring turns. Resolution becomes part of the geometry rather than an afterthought.

13.4 The delay and the clock

The delay τ is measured in units of the chosen computational clock. A delay of eight operations is not the same as eight processor cycles or eight milliseconds. If the algorithm has nested rhythms, delays chosen near fractions of those periods may expose different geometrical relations.

The reconstruction should therefore report at least

$$(h, \tau, m, \kappa, B, p), \tag{13.8}$$

where h is the observable, τ the delay, m the embedding dimension, κ the computational clock, and B, p the representation and precision.

Chapter 14

Reconstructing DCT and SVD Execution

14.1 Observing the DCT process

Let D_t be the DCT computational state and choose an observable

$$d_t = h_D(D_t). \tag{14.1}$$

Possible observables include the current partial coefficient, the norm of all active partial sums, the number of coefficients updated at a step, or the distance between successive coefficient vectors.

A three-coordinate delay reconstruction is

$$z_t^{(D)} = (d_t, d_{t-\tau}, d_{t-2\tau}). \tag{14.2}$$

If the implementation processes frequency indices in a repeated block structure, the reconstruction may reveal a local cycle. If it alternates even and odd components or shares butterflies in a fast transform, different recurrences may appear.

The purpose is not to declare one geometry to be the essence of DCT. It is to measure how a particular DCT implementation unfolds under a particular observation.

14.2 Observing the SVD process

Let S_t be the SVD state and select

$$s_t = h_S(S_t). \tag{14.3}$$

Candidate observables include

$$s_t = \|A_t\|_F, \tag{14.4}$$
$$s_t = \|A_t - A_{t-1}\|_F, \tag{14.5}$$
$$s_t = \text{current off-diagonal residual}, \tag{14.6}$$
$$s_t = \text{active rotation angle}, \tag{14.7}$$
$$s_t = \text{symbolic change count}, \tag{14.8}$$
$$s_t = \text{largest unresolved singular estimate}. \tag{14.9}$$

The delay reconstruction is

$$z_t^{(S)} = (s_t, s_{t-\tau}, s_{t-2\tau}). \tag{14.10}$$

Different algorithms may generate visibly different trajectories even when their terminal singular values agree. This provides an experimental route for separating terminal stability from internal path dependence.

14.3 Hankel matrices and the classical bridge

There is already a well-established classical bridge between time delays and SVD. Successive delay vectors can be arranged as columns of a Hankel matrix,

$$H = \begin{bmatrix} s_0 & s_1 & \cdots & s_K \\ s_1 & s_2 & \cdots & s_{K+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m-1} & s_m & \cdots & s_{K+m-1} \end{bmatrix}, \tag{14.11}$$

and decomposed using SVD to obtain effective coordinates. Modern time-delay modelling methods use this combination to identify low-dimensional structures in dynamical data [9].

FSM accepts this method but applies its central correction recursively. The Hankel matrix is a finite inscription produced from the sequence. The SVD of the Hankel matrix is another finite trajectory. The plotted coordinates are terminal inscriptions from that trajectory. No stage stands outside finite construction.

14.4 A hierarchy of trajectories

A complete experiment may involve

$$\mathcal{T}_{\text{problem}} \to \mathcal{T}_{\text{matrix formation}} \to \mathcal{T}_{\text{transform}} \to \mathcal{T}_{\text{delay construction}} \to \mathcal{T}_{\text{visualisation}} \to \mathcal{T}_{\text{interpretation}}. \tag{14.12}$$

Each arrow is itself a compression. Each stage has its own base, precision, operation order, and halting conditions. The hierarchy need not be expanded without limit. The investigator stops when the registered account is adequate for the stated purpose. The halting condition is part of the method.

Chapter 15

Circular, Toroidal, and Folded Computational Geometry

15.1 One recurrence and the circle

For a periodic scalar sequence

$$s_t = A\cos(\omega t + \phi), \tag{15.1}$$

a two-coordinate delay reconstruction is

$$z_t = (s_t, s_{t-\tau}). \tag{15.2}$$

For a suitable delay, the points trace an ellipse. If the two coordinates are approximately in quadrature and equally scaled, the ellipse approaches a circle.

In FSM, the circle is not an exact one-dimensional curve. It is a finite band formed by the sequence of regions $\mathcal{Z}_t$. The thickness depends upon representation, noise, and the chosen uncertainty model. If the regions overlap strongly, the cycle may be registered as a continuous annulus. At coarser resolution, separate turns may merge.

15.2 Two phases and the torus

Suppose an observable contains two independent recurrent phases:

$$s_t = A_1\cos(\omega_1 t + \phi_1) + A_2\cos(\omega_2 t + \phi_2). \tag{15.3}$$

The pair of phases lies on

$$S^1 \times S^1, \tag{15.4}$$

which is a two-torus. A suitable embedding or projection may therefore reveal toroidal geometry.

Nested algorithms naturally supply multiple phases. In an SVD computation, an element-level operation may cycle inside a row or column update; updates may cycle inside a sweep; sweeps may repeat within a convergence loop. In a block DCT, sample position, frequency index, and block number may form interacting periodicities.

The torus is therefore a plausible registered geometry of coupled computational rhythms.

It is not guaranteed by the words SVD or DCT. It arises when the observed execution contains sufficiently independent recurrent phases and when the chosen embedding preserves their distinction.

15.3 Why toroidal images recur

Toroidal diagrams appear frequently in dynamical systems, signal analysis, and computational visualisation because they offer a compact representation of multiple cyclic coordinates. FSM adds a historical and computational interpretation: the torus may emerge when a linear record is reconstructed so that several recurrent indices coexist spatially.

The mapping can be understood as a movement from a line with layered periodicities to a surface that holds those periodicities together:

$$\text{ordered line with two phases} \longrightarrow \text{toroidal relational surface.} \tag{15.5}$$

The surface is a reconstruction of properties of the line, not a replacement for the finite trajectory that generated it.

15.4 Beyond the torus

Algorithms can exhibit fixed regions, limit cycles, transient basins, intermittent corrections, branching due to thresholds, and irregular trajectories caused by nonlinear update rules or finite rounding. Delay reconstructions may show folds, sheets, clusters, or strange-attractor-like forms.

Care is essential. A visually complex attractor can be produced by projection, sampling, or numerical artefacts. FSM does not use geometry as decoration. Every shape must be tied to an observable, clock, delay, resolution, and repeatability test.

15.5 A proposed geometric hierarchy

For computational trajectories, a provisional hierarchy is

$$\text{stable terminal state} \longrightarrow \text{finite point-like cluster,} \tag{15.6}$$

$$\text{single recurrence} \longrightarrow \text{circular or elliptical band,} \tag{15.7}$$

$$\text{two recurrent phases} \longrightarrow \text{toroidal band,} \tag{15.8}$$

$$\text{multiple coupled phases} \longrightarrow \text{higher toroidal projection,} \tag{15.9}$$

$$\text{nonlinear recurrence and folding} \longrightarrow \text{folded finite attractor.} \tag{15.10}$$

This hierarchy is an experimental guide, not a universal classification.

Chapter 16

Recursive Measurement and the Absence of an External Transform

16.1 The reconstruction also unfolds

A delay coordinate appears compactly as

$$z_t = (s_t, s_{t-\tau}, s_{t-2\tau}). \tag{16.1}$$

Yet constructing the coordinate requires indexing, copying, storing, and arranging finite values. Plotting the point requires another chain of coordinate transformations and display operations. Measuring distances among reconstructed points requires further arithmetic.

Thus the delay method is not an external observer. It is another finite symbolic trajectory:

$$\mathcal{T}_{\text{source}} \to \mathcal{T}_{\text{measurement}} \to \mathcal{T}_{\text{reconstruction}} \to \mathcal{T}_{\text{display}}. \tag{16.2}$$

A finite symbolic trajectory can measure another trajectory only by producing a further finite symbolic trajectory.

16.2 No view from outside the landscape

The recursive structure does not imply that knowledge is impossible. It means that every claim has a finite provenance. A measuring trajectory can be calibrated against other trajectories. Independent implementations can be compared. Predictions can meet exogenous measurements. Stability can be strengthened by repetition.

What is removed is the imagined position of a transform or observer outside all representation. The SVD does not stand outside the matrix. The Takens reconstruction does not stand outside the SVD execution. The reader does not stand outside the text. Each relation is produced from within the finite symbolic landscape.

16.3 Halting without infinite regress

If every measurement requires another trajectory, why does the account not continue forever? In practice, every investigation halts. It stops when a result is sufficiently stable for a declared purpose, when resources are exhausted, when a threshold is crossed, or when an external decision is made.

FSM makes this stopping condition explicit:

$$H : \mathcal{T} \longrightarrow \{\text{continue}, \text{register}\}. \tag{16.3}$$

The terminal object is the inscription accepted at H. Another investigation may reopen it and generate a further trajectory. Finality is local to the purpose and epoch of measurement.

16.4 Geometry without hidden ontology

The phrase "hidden geometry" can be useful, but it can also recreate the classical privilege in a new form. It may suggest that the torus or attractor existed as a complete primary object behind the sequence, waiting to be revealed.

A more careful FSM statement is:

> Delay reconstruction produces a measurable spatial representation of relationships compressed by sequential symbolic presentation.

This wording retains the power of geometry while attaching it to the finite act of reconstruction.

Part V

Consequences for Mathematics and Computation

Chapter 17

Coefficients, Singular Values, and Terminal Residues

17.1 The coefficient is produced

A coefficient is often described as a property extracted from the input. The language is convenient but can conceal the trajectory. In a DCT, the coefficient c_k is not present as a separately registered object before the finite multiplications and accumulations occur. In an SVD, the singular value σ_k is not handed to the machine by the matrix. It is produced and stabilised through an algorithm.

We can write

$$c_{k,B} = \mathrm{halt}(\mathcal{T}_{\mathrm{DCT},k}), \tag{17.1}$$

$$\sigma_{k,B} = \mathrm{halt}(\mathcal{T}_{\mathrm{SVD},k}). \tag{17.2}$$

The notation halt does not mean that every coefficient has an independent program. It marks the terminal registration of the relevant computational subtrajectory.

The coefficient is therefore a residue: a compact symbolic value left after most of the execution history has been removed from view.

17.2 Residue does not mean arbitrary remainder

The word *residue* may sound accidental. Here it means a stable terminal compression. A coefficient can be reproducible across runs, architectures, and algorithms. It can support excellent reconstructions and predictions. Its status is strengthened when diverse finite trajectories converge within the declared resolution.

The FSM claim is not that singular values are invented freely. It is that the registered singular values are generated in a finite relation and carry that relation's provenance. Their apparent independence from the trajectory is an achievement of stability, not a starting assumption.

17.3 The ranked residue

The ordering

$$\sigma_1 \geq \sigma_2 \geq \cdots \tag{17.3}$$

provides a ranked terminal description. In low-rank approximation, retaining the first k singular components creates

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T. \tag{17.4}$$

This is a second compression. The SVD execution has already compressed its trajectory into factors; truncation then removes selected terminal components according to a norm and purpose.

The phrase "importance score" is therefore conditional. A singular value measures extent under a particular matrix inscription, inner-product geometry, and decomposition. It is important for a chosen reconstruction criterion. It does not supply a universal significance independent of representation.

17.4 Compression removes trajectory twice

Transform methods often involve two levels of trajectory removal:

1. the calculation trajectory is compressed into coefficients or factors;
2. the coefficient set is truncated or quantised into a smaller inscription.

For the DCT, high-frequency coefficients may be coarsened or discarded. For SVD, smaller singular components may be omitted. The resulting object can still reconstruct selected visible structure because the retained terminal residues stabilise dominant relations under the chosen measure.

The cost is the removal of provenance and fine distinction. FSM does not condemn this cost. It insists that the cost be understood as a change in the available trajectory, not merely as a reduction in an abstract file size.

Chapter 18

Equality, Provenance, and Algorithmic Plurality

18.1 The equality sign as a boundary inscription

The equation

$$A = U\Sigma V^T \tag{18.1}$$

places the beginning and end of a process together. It states a structural relation and suppresses the path by which the factors were obtained. In an exact formal setting, the equality is fully legitimate. In a finite implementation, the reconstructed product may register as

$$A_B \sim_B U_B \Sigma_B V_B^T . \tag{18.2}$$

The relation may be extraordinarily close under a declared norm, yet the finite inscription on the right is not produced without rounding and representation.

FSM therefore distinguishes several uses:

$$= \quad \text{formal equality under declared symbolic rules,} \tag{18.3}$$

$$=_B \quad \text{identity of finite inscriptions within a representation,} \tag{18.4}$$

$$\sim_B \quad \text{finite reconstruction relation in base } B, \tag{18.5}$$

$$\approx_{p,H} \quad \text{agreement under precision } p \text{ and halting } H. \tag{18.6}$$

These symbols are proposals for exposition, not compulsory replacements for standard notation. Their purpose is to prevent an equality sign from silently erasing computational provenance when provenance matters.

18.2 Equality and reproducibility

An equality can be read as a compressed claim of reproducibility: under the accepted rules, either side may be substituted for the other without changing the relevant downstream trajectory. This is a powerful property. It does not require that the two inscriptions have identical historical origins.

From the FSM perspective, equality is stable when transformations preserve the distinc-

tions relevant to the purpose. This suggests a practical reading:

> Two finite symbolic constructions are equal for a stated operation when admissible substitutions repeatedly preserve the registered relation at the chosen resolution.

Formal mathematics strengthens this into rule-governed exactness within its symbolic system. FSM restores the measurement conditions when the symbols are instantiated.

18.3 Algorithmic plurality

The same formal relation may be produced through many trajectories. Matrix multiplication alone admits several loop orders, blocking strategies, parallel decompositions, and arithmetic precisions. The SVD admits several algorithmic families. The DCT admits direct and fast implementations.

Let

$$\mathfrak{A}(I) = \{\mathcal{A}_1, \ldots, \mathcal{A}_m\} \tag{18.7}$$

be the set of admissible algorithms for an inscription I. Each produces

$$R_j = \mathrm{halt}(\mathcal{T}_j). \tag{18.8}$$

A stable mathematical relation is supported when the R_j agree in the distinctions relevant to the task.

Plurality is therefore not a threat to objecthood. It is one of the means by which object stability can be measured.

18.4 Provenance as part of the result

A complete finite result should be understood as

$$R^* = (R_B, \Pi), \tag{18.9}$$

where R_B is the terminal inscription and Π is its provenance. The provenance need not contain every transistor event. It contains the information necessary for the intended evaluation: source data, algorithm, version, base, precision, tolerances, operation order when relevant, and halting conditions.

Different purposes require different provenance thickness. A classroom calculation may need only the method and rounding convention. A safety-critical computation may require hardware, compiler, library, and reproducibility records. FSM does not prescribe one universal ledger. It identifies provenance as structurally inseparable from finite meaning.

Chapter 19

Measured Complexity and the Geometry of Algorithms

19.1 Beyond counting operations

Classical computational complexity studies how resource requirements grow with input size under abstract models. This remains indispensable. FSM suggests a complementary measured geometry of execution.

An algorithm may be examined through quantities such as:

- the number of distinguishable registered states;
- the total symbolic distance traversed;
- the overlap between successive finite regions;
- the recurrence rate of previously visited regions;
- the number of resolution-boundary crossings;
- the thickness of the reconstructed trajectory;
- the stability of the terminal basin across implementations;
- the amount of trajectory removed by the final compression.

These measures do not replace big-O notation. They describe features that asymptotic operation counts intentionally ignore.

19.2 Symbolic path length

Given a representation-dependent distance d_B, define the symbolic path length

$$L_B(\mathcal{T}) = \sum_{t=0}^{T-1} d_B(C_{t+1}, C_t). \tag{19.1}$$

Two algorithms with the same operation count may have different path lengths because one changes many digits or registers per step while another makes smaller local updates. A

path with repeated large excursions may be more sensitive to finite precision than a path contained within a stable region.

The measure depends upon the chosen state description and metric. That dependence is not a defect if it is declared. Every measurement requires a ruler.

19.3 Overlap density

Let $\mathcal{Z}_t$ be the finite region associated with state C_t. Define a simple local overlap indicator

$$o_t = \begin{cases} 1, & \mathcal{Z}_t \cap \mathcal{Z}_{t+1} \neq \varnothing, \\ 0, & \text{otherwise.} \end{cases} \tag{19.2}$$

An overlap density is

$$\rho_O = \frac{1}{T} \sum_{t=0}^{T-1} o_t. \tag{19.3}$$

A high value may indicate slow movement relative to resolution, convergence, or an overly coarse measurement. A low value may indicate large updates or a fine resolution. Interpretation requires comparison across controlled conditions.

19.4 Terminal basin stability

Run an algorithm under small admissible variations of input registration, precision, or operation order. Let the terminal regions be $\mathcal{B}_1, \ldots, \mathcal{B}_m$. Their intersection, clustering, or pairwise distance provides a measured account of terminal stability.

This could complement classical conditioning. Conditioning describes sensitivity under a mathematical problem definition. Terminal basin geometry describes the finite registered distribution produced by actual trajectories.

19.5 Compression ratio as trajectory removal

A usual compression ratio compares storage sizes. FSM proposes a related conceptual measure:

$$\Gamma = \frac{\text{registered distinctions in trajectory}}{\text{registered distinctions retained in residue}}. \tag{19.4}$$

This is not immediately a universal scalar because "distinction" depends upon representation. But it frames an important question: how much of the execution and provenance has been removed to produce the final object?

A singular value list can be tiny compared with the state history that generated it. The extraordinary efficiency of mathematics lies partly in this ratio.

Chapter 20

An Experimental Programme

20.1 Purpose

The monograph's claims can be explored through modest computational experiments. The goal is not to prove that every algorithm possesses one true toroidal or spherical geometry. The goal is to demonstrate that apparently timeless matrix operations have observable finite execution trajectories and that these trajectories can be compared under controlled changes of algorithm, base, precision, and measurement.

20.2 Experiment one: direct matrix multiplication

Choose a small matrix A and vector x. Implement several multiplication orders:

1. row-wise sequential accumulation;
2. column-wise accumulation;
3. pairwise reduction;
4. blocked multiplication;
5. reduced-precision arithmetic.

At each step record:

$$s_t^{(1)} = \text{current partial sum}, \tag{20.1}$$

$$s_t^{(2)} = \|C_t - C_{t-1}\|, \tag{20.2}$$

$$s_t^{(3)} = \text{number of changed digits or bits}, \tag{20.3}$$

$$s_t^{(4)} = \text{current residual against a high-precision reference}. \tag{20.4}$$

Construct two- and three-coordinate delay embeddings for each observable. Compare internal geometry and terminal registration.

20.3 Experiment two: direct and fast DCT

Apply a direct DCT and a fast DCT to the same finite block. Instrument the code so that every partial coefficient update or butterfly stage is recorded. Compare:

- execution length;
- recurrence spectra of the execution observable;
- delay geometry;
- terminal coefficient differences;
- sensitivity to fixed-point and floating-point representations.

This experiment tests the distinction between a shared transform relation and different computational trajectories.

20.4 Experiment three: two SVD algorithms

Use a small matrix for which detailed state recording is feasible. Compare, for example, a Jacobi-style SVD with a bidiagonalisation-based implementation. Record a residual such as the off-diagonal norm, update magnitudes, active rotation parameters, and symbolic change counts.

The central comparison is

$$\text{different internal geometry} \quad \text{with} \quad \text{stable terminal singular structure.} \tag{20.5}$$

Repeated singular values and ill-conditioned examples should be included to show where the terminal factors themselves become less uniquely stabilised.

20.5 Experiment four: base and precision

Repeat the DCT and SVD experiments using:

- binary floating point at several precisions;
- decimal arithmetic;
- fixed-point arithmetic;
- alternative rounding modes where available.

Associate each recorded value with a finite region determined by the local representational spacing or a declared uncertainty model. Examine when circular or toroidal structures remain distinguishable and when they collapse into overlapping bands.

20.6 Experiment five: the reconstruction trajectory

Instrument the delay-embedding code itself. Record the operations required to construct the Hankel or delay matrix, perform its SVD, and render the plot. This experiment demonstrates

the recursive principle: the method used to reveal a computational trajectory possesses another trajectory of its own.

20.7 Minimal pseudocode

A generic instrumentation pattern is:

```
state_log = []
state = initialise(input, representation)
state_log.append(observe(state))

while not halt(state):
    state = finite_update(state)
    state_log.append(observe(state))

result = register(state)
embedding = delay_embed(state_log, tau, dimension)
```

The simplicity of the outline is useful. The conceptual difficulty lies not in logging values but in choosing observables and interpreting the finite geometry without returning to an undeclared classical object.

Part VI

Discussion and Synthesis

Chapter 21

Discussion: From Procedure to Object and Back Again

21.1 The full arc

The argument of this monograph began before matrices. People faced coupled relations and developed procedures for transforming finite arrangements. The operation came before the modern object. Coefficients were placed on working surfaces because spatial arrangement made several relations available together, but solutions still required ordered transformations.

Determinants, elimination methods, quadratic forms, and linear substitutions gradually prepared a symbolic basin in which the array could acquire autonomy. Sylvester's term *matrix* gave the arrangement a stable name. Cayley's algebra established that matrices could be added, multiplied, inverted, and powered as quantities. A compression of procedures and relations became a primary mathematical instrument.

This first inversion was extraordinarily productive:

$$\text{procedure} \longrightarrow \text{matrix object}. \tag{21.1}$$

Once the matrix existed as a stable object, it could unify diverse domains. Transformations, statistics, mechanics, geometry, probability, and later quantum theory could all be routed through a common symbolic surface.

Electronic computation produced the reciprocal inversion:

$$\text{matrix object} \longrightarrow \text{finite procedure}. \tag{21.2}$$

The machine could not act on the matrix as an ideal whole. It required finite entries, memory locations, operation order, arithmetic rules, comparisons, and a stopping condition. Numerical analysis developed because the unfolding could not be ignored. Rounding, conditioning, stability, and error propagation became visible properties of matrix processes.

The history therefore contains the central FSM structure:

$$\boxed{\text{operations generate the matrix; the matrix later generates operations.}} \tag{21.3}$$

21.2 Why the classical frame is so easy to recover

The classical frame is difficult to escape because the notation is exceptionally successful. Once A has been stabilised as an object, it is efficient to say that A acts, rotates, stretches, or decomposes. The language lets us reason at a high level without repeatedly unfolding every arithmetic step.

Even after recognising that A must be executed, one can unconsciously preserve the transform as an external operator. The SVD is then imagined as a primary mathematical object applied to a merely finite input. This was the critical point in the development of the present argument. The matrix was allowed to become a trajectory, but the SVD remained privileged.

The FSM correction had to be applied again:

$$\text{SVD} \quad \text{also requires an unfolding.} \tag{21.4}$$

The transform does not occupy a timeless position above the sequence. It is generated through a second sequence of finite states. Its factors are terminal registrations. Once this correction is accepted, the entire classical hierarchy changes.

21.3 The central proposition

The central proposition can now be read in its full strength:

The transform is not applied to the trajectory. The transform is another trajectory.

This is more than the observation that computers use steps. It removes the transform's privileged ontological and explanatory position. The DCT is not an abstract cosine basis touching finite data from outside. The basis values must be stored or generated; products must be formed; sums must be accumulated; coefficients must be registered. The SVD is not a timeless disassembly of A. Rotations, reflections, reductions, sweeps, residual tests, and halting decisions must occur.

The transform relation remains valid as compression. What changes is its status. It is no longer the unexplained primary event. It is the terminal boundary inscription around a family of finite trajectories.

21.4 The line is operationally prior

A matrix appears as a surface. All entries are visibly present together. A transform result may appear as another surface or coefficient list. Yet actual calculation produces an ordered line of states, even when the underlying machine is parallel and the recorded order is a chosen serialisation.

This leads to a powerful inversion of emphasis:

$$\text{surface inscription} \longrightarrow \text{linear execution} \longrightarrow \text{surface inscription}. \tag{21.5}$$

The surface compresses relationships into coexistence. The execution unfolds them into time. The terminal surface compresses the new trajectory again.

From an FSM perspective, the line is not necessarily metaphysically more real than every surface. It is operationally prior whenever an inscription must be enacted. The matrix can remain on the page without changing. The calculation exists only through a sequence of finite transitions.

21.5 The sphere adds finite extent

A one-dimensional record of execution can be reconstructed into two or three coordinates using delays. Classical reconstruction places exact points in a state space. FSM replaces the point with a finite registration region.

The resulting geometry is a sequence of spheres or cells. This modification prevents the visual curve from falsely suggesting infinite precision. It also creates new measurable relations. Overlap indicates that successive states are not fully distinguishable at the declared resolution. Separation indicates a registered transition. A convergent process may produce a dense cluster of overlapping spheres. A periodic process may produce a thick ring. Coupled phases may produce a toroidal band.

The sphere therefore connects the Alphonic limit to computational dynamics. It gives finite symbolic extent a spatial form.

21.6 DCT and SVD: similar at the deeper level

At the classical level, DCT and SVD differ significantly. The DCT uses a prescribed orthogonal cosine basis. The SVD produces data-conditioned singular directions. The DCT is usually applied blockwise to a sequence or image. The SVD acts upon a matrix and orders singular components by magnitude.

Those distinctions remain. At the deeper FSM level, however, both share the form

$$\text{finite input inscription} \rightarrow \text{ordered symbolic interactions} \rightarrow \text{terminal coefficient structure}. \tag{21.6}$$

The fixed-basis versus adaptive-basis distinction is secondary to this more fundamental commonality. Both are trajectories that stabilise compressed residues.

The original intuition that DCT and SVD are related through properties of a finite line can therefore be retained with precision. A DCT measures recurrence against prescribed finite modes. An SVD can be used on a matrix constructed from delayed windows to measure dominant directions of the reconstructed line. But in both cases the measuring transform must itself be unfolded.

21.7 Takens reconstruction in the corrected hierarchy

Takens-style delay coordinates provide the bridge from sequential observation to spatial relation:

$$s_t \longmapsto (s_t, s_{t-\tau}, s_{t-2\tau}). \tag{21.7}$$

The method brings separated moments into coexistence. It makes curvature, recurrence, and folding available to inspection.

The corrected hierarchy is not

$$\text{trajectory} \to \text{external Takens geometry}. \tag{21.8}$$

It is

$$\text{source trajectory} \to \text{delay-construction trajectory} \to \text{registered geometry}. \tag{21.9}$$

The reconstructed geometry is conditional upon the observable, delay, dimension, clock, base, precision, and display. It is a measured artefact with provenance.

This does not make the geometry less useful. It makes its evidential basis clearer.

21.8 The torus and nested computational phases

The suggestion of toroidal geometry becomes coherent when the execution contains nested recurrent indices. A DCT may cycle through samples inside frequencies inside blocks. An SVD may cycle through local updates inside sweeps inside convergence iterations. Two sufficiently independent phases can be represented on $S^1 \times S^1$, and a delay reconstruction may display a torus or a projection of one.

The torus is therefore not a mystical hidden form of matrices. It is a possible registered geometry of multiple computational rhythms. Its appearance should be tested by changing the observable, delay, and resolution. A robust toroidal relation will persist across reasonable variations; an artefact will not.

This measured approach is characteristic of FSM. Geometry is neither denied nor granted unconditional primacy. It is constructed, compared, and stabilised.

21.9 Invariance as measured stability

One of the monograph's strongest consequences concerns invariance. Classical mathematics often treats invariant relations as belonging to the object before computation. Numerical work then attempts to recover them.

FSM reverses the evidential order. Repeated finite unfoldings generate terminal registrations. When a relation persists across different admissible algorithms, bases, precisions, and operation orders, it becomes measured as stable. The stability may support the classical invariant with extraordinary strength.

This yields

$$\text{invariance claim} \quad \rightsquigarrow \quad \text{stable relation across a family of finite trajectories}. \tag{21.10}$$

The classical theorem and the FSM measurement need not conflict. The theorem supplies a compressed rule within a formal basin. The measurement shows how that rule stabilises when instantiated.

21.10 Equality as compressed provenance

The equality sign is among the most powerful trajectory-removal devices in mathematics. It permits substitution without retelling the path. In the SVD equation, it places A and $U\Sigma V^T$ together while removing the algorithm that produced the factors.

FSM does not demand that every equation be replaced by a decorated approximation sign. It asks the writer to recognise when finite provenance matters. In a theorem, exact equality may be the appropriate relation. In a computational report, the base, precision, residual, and stopping condition may be inseparable from the claim.

Thus equality has layers. It can be a formal rule, a finite identity of inscriptions, a measured agreement, or a practical substitution relation. The monograph's proposed notation makes these layers visible when needed.

21.11 Mathematical objects survive the critique

A common misunderstanding would be to conclude that FSM dissolves matrices into arbitrary processes. The opposite is intended. The matrix survives because it is a remarkably stable compression. Across different media, notations, algorithms, and readers, sufficiently compatible trajectories repeatedly reconstruct the same relational instrument.

The object is strengthened when its stability is demonstrated rather than assumed. A matrix can be treated as real within the finite symbolic landscape because it has consequences, can be reproduced, can be transformed, and can coordinate measurements. What is rejected is only the claim that its practical and mathematical power requires a primary existence outside all inscription and trajectory.

21.12 The reader's changed position

By the end of the monograph, the reader should no longer see

$$A = U\Sigma V^T \tag{21.11}$$

as a single complete event. The equation now marks the compressed boundary of several histories:

$$\mathcal{T}_{\text{input}}, \quad \mathcal{T}_{\text{storage}}, \quad \mathcal{T}_{\text{SVD}}, \quad \mathcal{T}_{\text{reconstruction}}, \quad \mathcal{T}_{\text{interpretation}}. \tag{21.12}$$

The equation has not become false. It has become thicker.

This thickness is one of the main goals of FSM. A finite symbolic trajectory has history, uncertainty, admissibility, and extent. Compression makes these features quiet. The monograph makes them audible again.

21.13 The broader proposition about mathematics

The discussion extends beyond matrices:

$$\boxed{\text{Mathematical notation spatially compresses temporal symbolic labour.}} \tag{21.13}$$

A theorem, integral, derivative, transform, algorithm name, or software call is a finite inscription that relies upon a reconstructive basin. Calculation temporally unfolds the inscription. The result stabilises another symbol. The cycle repeats.

This is not an attack on abstraction. Abstraction is the means by which trajectories become portable. The FSM contribution is to retain the finite cost, provenance, and geometry that abstraction removes.

21.14 What the monograph has established

The complete argument can be condensed into twelve propositions:

1. Coupled procedures historically preceded the modern matrix object.
2. Matrix notation compressed those procedures into a stable symbolic surface.
3. Matrix algebra transformed the surface into a primary classical instrument.
4. Electronic computation forced the object back into finite ordered operations.
5. The matrix inscription is not identical to its execution trajectory.
6. The DCT is a bundle of finite coefficient-producing trajectories.
7. The SVD is a family of finite decomposition trajectories.
8. The SVD cannot be retained as an external primary operator once the matrix is treated finitely.
9. Delay coordinates reconstruct relationships in a finite computational line through another trajectory.
10. Reconstructed coordinates have finite extent and can be represented as spheres or regions.
11. Circular and toroidal forms may arise from recurrent computational phases under declared measurements.
12. Equality, invariance, and mathematical objecthood can be retained as compressed, measured, and reproducible relations with provenance.

These propositions form a coherent feed-forward trajectory. They open a practical research programme in which algorithms are not studied only by their final answers or asymptotic counts, but also by the finite geometry of their unfolding.

Chapter 22

Conclusion: The Unfolding of the Object

We began with a simple matrix decomposition and arrived at a general account of mathematical inscription. The journey was possible because the SVD equation is unusually revealing. It presents a rich operation in a compact surface. It appears to say that a matrix is already equal to three completed factors. Yet actual calculation requires an algorithmic history.

The historical introduction showed that this tension is not new. Procedures for coupled relations existed before matrices. The matrix emerged as an abbreviation, acquired a name, developed an algebra, and became a primary instrument. The computer later returned it to ordered operations. The history is not a straight progression from crude calculation to pure object. It is an oscillation between procedure and inscription.

FSM makes that oscillation foundational:

$$\text{trajectory} \to \text{inscription} \to \text{trajectory} \to \text{inscription}. \tag{22.1}$$

The inscription compresses. The calculation unfolds. The result stabilises. Another reader or machine begins again.

The DCT and SVD are not exceptions to this cycle. They are exemplary cases. Their coefficients and factors are compressed residues of finite interactions. Their apparent geometries can be reconstructed from execution histories. Their stability can be compared across algorithmic families. Their terminal relations can remain powerful without being detached from base, precision, operation order, and halting.

The final propositions are therefore simple:

The symbol compresses the trajectory.
The calculation unfolds the symbol.
The result stabilises another symbol.
The transform is another trajectory.

Once this sequence is held firmly, the matrix is no longer diminished. It becomes more intelligible. We can see both why it became one of mathematics' greatest instruments and what must occur whenever that instrument is made to act.

Part VII

Appendices

Appendix A

Core Terminology

Term	Working definition
Finite symbol	A distinguishable registered mark or state with finite extent, resolution, and provenance.
Finite symbolic trajectory	An ordered sequence of finite symbolic states generated by interpretation, calculation, measurement, or communication.
Functional Symbolic Trajectory	A finite symbolic pathway that carries representation, constraint, uncertainty, and use through a system.
Alphonic limit	The active lower boundary at which the chosen symbolic apparatus can register one distinction rather than another.
Alphonic sphere	An isotropic geometric representation of the finite extent or uncertainty associated with a registered coordinate.
Inscription	A finite arrangement of symbols capable of being interpreted or executed.
Unfolding	The ordered finite trajectory generated when a compressed inscription is enacted.
Stabilisation	The local acceptance of a finite registration as sufficiently persistent and reproducible for a purpose.
Residue	A terminal symbolic compression left after an execution trajectory has been largely removed from view.
Provenance	The relevant history of input, representation, algorithm, precision, operation order, measurement, and halting.
Admissibility	The rules and constraints under which a finite symbolic result is accepted for a declared purpose.
Measured stability	Persistence of a relation across a family of admissible finite trajectories.
Computational clock	The chosen ordering unit for an execution sequence, such as operation count, sweep count, processor cycle, or wall time.
Delay reconstruction	A finite procedure that brings temporally separated observations into coordinate coexistence.

Appendix B

Classical Formulae Used in the Monograph

B.1 Matrix multiplication

For $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$,

$$y = Ax, \qquad y_i = \sum_{j=1}^{n} a_{ij}x_j. \tag{B.1}$$

B.2 DCT-II

One common normalisation is

$$c_k = \alpha_k \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], \tag{B.2}$$

with

$$\alpha_0 = \sqrt{\frac{1}{N}}, \qquad \alpha_k = \sqrt{\frac{2}{N}} \quad (k>0). \tag{B.3}$$

B.3 Singular value decomposition

For $A \in \mathbb{R}^{m\times n}$,

$$A = U\Sigma V^T, \tag{B.4}$$

with orthogonal U, V and non-negative diagonal singular values in Σ.

B.4 Truncated SVD

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T. \tag{B.5}$$

Under standard conditions this is a best rank-k approximation in the spectral and Frobenius norms [4].

B.5 Delay coordinates

$$z_t = (s_t, s_{t-\tau}, \ldots, s_{t-(m-1)\tau}). \tag{B.6}$$

B.6 Hankel delay matrix

$$H_{ij} = s_{i+j}, \tag{B.7}$$

with indexing adjusted to the chosen finite sequence and window.

Appendix C

Classical and FSM Readings Compared

Classical compression	**FSM expansion**
The matrix is a primary object.	The matrix is a stable finite inscription produced and maintained through trajectories.
The matrix acts on a vector.	The inscription initiates a finite execution that registers an output.
The transform is applied to the data.	The transform unfolds as another finite trajectory interacting with the data inscription.
The coefficient is extracted.	The coefficient is produced and stabilised at a halting condition.
The basis is fixed.	The basis values and operations have a finite computational provenance.
SVD reveals latent directions.	An SVD trajectory produces factors whose relational stability is measured across admissible unfoldings.
A state is an exact point.	A registered state is a finite region of distinguishability.
The attractor is revealed.	A spatial reconstruction is produced under a declared observable, delay, dimension, and resolution.
Invariance belongs to the object.	Stability is measured across families of finite trajectories.
Equality gives a timeless relation.	Equality is a powerful compression that may suppress finite provenance.
Implementation is secondary.	The implementation trajectory is the means by which the relation becomes finitely measurable.

Appendix D

Historical Timeline

Period	Development relevant to the argument
Ancient China	The *fangcheng* procedures in *The Nine Chapters* arrange coefficients and use ordered elimination methods for coupled problems.
Seventeenth–eighteenth centuries	Determinants, elimination procedures, and linear substitutions develop in several traditions; procedures remain primary.
1850	Sylvester introduces the term *matrix* for an arrangement from which determinant systems can be formed.
1858	Cayley publishes a systematic matrix algebra, treating arrays as quantities with addition, multiplication, powers, identity, and inverse.
Late nineteenth–early twentieth centuries	Matrix theory expands through canonical forms, spectral theory, linear transformations, mechanics, and mathematical physics.
1936	Eckart and Young publish the best low-rank approximation result associated with truncated singular-value structure.
1947	Von Neumann and Goldstine analyse numerical inversion of high-order matrices in the context of electronic computation and finite error.
1948	Turing publishes *Rounding-Off Errors in Matrix Processes*, foregrounding finite computation and stability.
1965	Golub and Kahan publish a practical stable scheme for calculating singular values and the pseudo-inverse.
1974	Ahmed, Natarajan, and Rao introduce the discrete cosine transform.
1979 onward	BLAS and later extensions standardise basic vector and matrix operations as reusable computational primitives.
Contemporary computing	Matrix operations become central software and hardware substrates for simulation, data analysis, graphics, signal processing, and machine learning.
FSM reading	The historical movement is reinterpreted as procedure → inscription → object → computational trajectory.

Appendix E

A More Detailed Experimental Outline

E.1 Data record

A reproducible experiment should store, for each event,

$$(t, \kappa, C_t, h(C_t), B, p, H, \text{metadata}), \tag{E.1}$$

where κ identifies the computational clock. If the full state is too large, record a hash together with selected observables and periodic checkpoints.

E.2 Suggested observables

- partial output values;
- update norms;
- residual norms;
- active row, column, frequency, or sweep indices;
- Givens or Householder parameters;
- number of altered digits, bits, or array entries;
- accumulated rounding residue estimated against higher precision;
- memory movement or cache-miss proxies;
- elapsed physical time at controlled checkpoints.

E.3 Delay selection

Compare several delays rather than selecting one visually attractive result. Candidate methods include autocorrelation decay, mutual information, known algorithmic periods, and explicit fractions of nested loop lengths. Record all choices.

E.4 Embedding dimension

Begin with two and three dimensions for visualisation, but use quantitative tests to determine whether higher dimensions are required. The displayed geometry should be described as a projection when the state is known to be higher dimensional.

E.5 Finite radii

Possible sphere radii include:

- half the spacing to adjacent representable values;
- propagated numerical uncertainty;
- empirical run-to-run variation;
- a declared Alphonic resolution for the visual instrument.

Different radius models answer different questions and should not be mixed without explanation.

E.6 Comparisons

For each algorithm pair, compare:

1. terminal numerical agreement;
2. residual and backward error;
3. path length;
4. recurrence statistics;
5. overlap density;
6. reconstructed topology under controlled delays;
7. sensitivity to base and precision;
8. sensitivity to halting criteria.

Bibliography

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974.

[2] A. Cayley, "A Memoir on the Theory of Matrices," *Philosophical Transactions of the Royal Society of London*, vol. 148, pp. 17–37, 1858.

[3] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, "An Extended Set of FORTRAN Basic Linear Algebra Subprograms," *ACM Transactions on Mathematical Software*, vol. 14, no. 1, pp. 1–17, 1988.

[4] C. Eckart and G. Young, "The Approximation of One Matrix by Another of Lower Rank," *Psychometrika*, vol. 1, pp. 211–218, 1936.

[5] G. Golub and W. Kahan, "Calculating the Singular Values and Pseudo-Inverse of a Matrix," *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, vol. 2, no. 2, pp. 205–224, 1965.

[6] J. F. Grcar, "How Ordinary Elimination Became Gaussian Elimination," *Historia Mathematica*, vol. 38, no. 2, pp. 163–218, 2011.

[7] R. Hart, *The Chinese Roots of Linear Algebra*. Baltimore: Johns Hopkins University Press, 2011.

[8] N. J. Higham, "Cayley, Sylvester, and Early Matrix Theory," *Linear Algebra and its Applications*, vol. 428, pp. 39–43, 2008.

[9] S. M. Hirsh, S. M. Ichinaga, S. L. Brunton, J. N. Kutz, and B. W. Brunton, "Structured Time-Delay Models for Dynamical Systems with Connections to the Frenet–Serret Frame," *Proceedings of the Royal Society A*, vol. 477, 2021, article 20210097.

[10] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic Linear Algebra Subprograms for FORTRAN Usage," *ACM Transactions on Mathematical Software*, vol. 5, no. 3, pp. 308–323, 1979.

[11] J. J. Sylvester, "Additions to the Articles, 'On a New Class of Theorems,' and 'On Pascal's Theorem'," *Philosophical Magazine*, vol. 37, pp. 363–370, 1850.

[12] F. Takens, "Detecting Strange Attractors in Turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, D. A. Rand and L.-S. Young, eds., Lecture Notes in Mathematics, vol. 898. Berlin: Springer, pp. 366–381, 1981.

[13] A. M. Turing, "Rounding-Off Errors in Matrix Processes," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 1, no. 1, pp. 287–308, 1948.

[14] J. von Neumann and H. H. Goldstine, "Numerical Inverting of Matrices of High Order," *Bulletin of the American Mathematical Society*, vol. 53, pp. 1021–1099, 1947.