Linear, Geometric, and Structural Transformations

Jose Portillo

Release date: January 6, 2026
Version: 0.3

To the memory of
Mercelena Portillo Holskin,
Eldon C. Hall,
Hugh Blair-Smith

To my family

Contents

Preface

This monograph began as an attempt to understand what remains invariant when formulas, coordinates, and representations are removed.

Vector spaces are introduced as abstract sets equipped with addition and scalar multiplication. Linear transformations are defined as the maps that preserve this structure.

By introducing bilinear forms and, ultimately, inner products, vectors are allowed to interact in a way that produces scalars. From this single interaction arise norms, orthogonality, distances, and classical geometric objects such as balls, spheres, and ellipsoids.

Many constructions encountered earlier—bilinear forms, inner products, matrix multiplication—share a common feature: they are linear in each argument separately. This observation leads naturally to multilinear maps and to tensors.

Tensors are treated as abstract multilinear objects defined without coordinates. The tensor product is introduced through its universal property, explaining how multilinear problems are canonically reduced to linear ones. Index notation, arrays, and computational implementations are presented explicitly as representations.

A recurring distinction will be made between an axiomatic theory and a concrete model of that theory. In a refined axiomatic treatment of Euclidean geometry, objects such as *point*, *line*, and *plane* are taken as primitive (undefined) terms, and the axioms specify only the relations they must satisfy. A Cartesian coordinate system does not redefine these primitives; rather, it provides a *model* in which points are represented by tuples in $\mathbb{R}^n$, lines by solution sets of linear equations, and incidence becomes set membership [3]. As stated in [5], this is *the assignment of meanings*. This perspective helps place coordinate formulas as representations of structure, and not as the structure itself.

Concretely, the text is organized around four recurring structural themes (linear structure, geometric and topological structure, multilinear structure, and logarithmic structure).

These themes are successive refinements of the same organizing idea: structure is revealed through invariance under transformations.

For the reader, in order to understand the subjects, and use these notes as a starting point for more research, it is assumed prior familiarity with basic set theory [5][4][8], relations, functions [8], and n-tuples [1][4], and basic algebra [1]. It also assumes basic knowledge of calculus [8]. The reports [13] and [14] include all the basic material on tensors. Other introductory material needed is scattered throughout the bibliography.

An AI language model was used as an editor and support tool for LaTeX document organization and stylistic refinement.

This monograph presents the abstract part that underlies every algorithm and autonomous computing system, and therefore remains invariant under any change in the state of technology. There is a more physical/hardware-oriented counterpart (centered on a 1960s computer: the Apollo Guidance Computer), which is more specific to a moment in that state of the art. For the full context, see:

`https://jportillo34.github.io/ApolloGuidanceComputer/`

Chapter 1

Linear Transformations, Matrices, and Composition

$$\Big\langle \langle K, +_g, \cdot \rangle,\ \langle E, +_E \rangle,\ \circ \Big\rangle$$

The triple above defines a *vector space* E over the *field* K [4]. Elements of K are called *scalars*, and elements of E are called *vectors*. The operation $\circ : K \times E \to E$ is scalar multiplication, and $+_E$ is vector addition. In what follows, symbols $\alpha_1, \ldots, \alpha_k$ always denote scalars in K, and symbols $x_1, \ldots, x_k$ always denote vectors in E.

Informally, a vector space is a set of vectors that can be added together and scaled by elements of a field.

Example (Coordinate n-space, [6]).

Let Γ be a field. Consider the set

$$\Gamma^n = \Gamma \times \cdots \times \Gamma$$

of n-tuples $(\xi^1, \ldots, \xi^n)$, with $\xi^i \in \Gamma$. Define addition by

$$(\xi^1, \ldots, \xi^n) + (\eta^1, \ldots, \eta^n) = (\xi^1 + \eta^1, \ldots, \xi^n + \eta^n)$$

and scalar multiplication by

$$\lambda(\xi^1, \ldots, \xi^n) = (\lambda\xi^1, \ldots, \lambda\xi^n), \quad \lambda \in \Gamma.$$

With these operations, Γ^n is a vector space over Γ, called the *n-space over* Γ. In particular, Γ itself is a vector space over Γ, where scalar multiplication coincides with field multiplication.

Given vectors $x_1, \ldots, x_k \in E$ and scalars $\alpha_1, \ldots, \alpha_k \in K$, an expression of the form

$$\alpha_1 \circ x_1 +_E \alpha_2 \circ x_2 +_E \cdots +_E \alpha_k \circ x_k$$

is called a *linear combination*.

A subset $S \subset E$ is called a *system of generators* (or a *spanning set*) of E if every vector $x \in E$ can be written as a linear combination of elements of S [6]. That is, for every $x \in E$ there exist vectors $s_1, \ldots, s_k \in S$ and scalars $\alpha_1, \ldots, \alpha_k \in K$ such that

$$x = \alpha_1 \circ s_1 +_E \cdots +_E \alpha_k \circ s_k.$$

A finite family of vectors $\{x_1, \ldots, x_k\} \subset E$ is called *linearly independent* if the relation

$$\alpha_1 \circ x_1 +_E \cdots +_E \alpha_k \circ x_k = 0$$

implies

$$\alpha_1 = \cdots = \alpha_k = 0.$$

If such a nontrivial relation exists (i.e., with at least one coefficient $\alpha_i \neq 0$), the family is called *linearly dependent.*

A finite set of vectors $\{b_1, \ldots, b_n\} \subset E$ is called a *basis* of the vector space E if it is both a system of generators of E and linearly independent.

Equivalently, $\{b_1, \ldots, b_n\}$ is a basis of E if every vector $x \in E$ can be written *uniquely* as a linear combination

$$x = \alpha_1 \circ b_1 +_E \alpha_2 \circ b_2 +_E \cdots +_E \alpha_n \circ b_n, \quad \alpha_1, \ldots, \alpha_n \in K.$$

A basis therefore provides a minimal and non-redundant set of building blocks for the entire space. Once a basis is fixed, every vector is completely determined by its coefficients relative to that basis, so we can say the basis generates the space.

If the vector space E admits a finite basis, the number of vectors in any basis of E is called the *dimension* of E and is denoted by $\dim E$.

Remark (On coordinates and isomorphism [2]).

Let E be a finite-dimensional vector space over K with $\dim E = n$. Once a basis $B = \{b_1, \ldots, b_n\}$ is chosen, every vector $x \in E$ is uniquely represented by its coordinate column

$$[x]_B \in K^n.$$

This correspondence defines a linear isomorphism

$$E \longrightarrow K^n.$$

Thus, from the purely algebraic point of view, every n-dimensional vector space over K is structurally indistinguishable from K^n.

However, this identification depends on the chosen basis. Different bases produce different coordinate representations. The intrinsic properties of the space are precisely those that remain invariant under such changes of reference system.

For this reason, we treat vector spaces as abstract entities, independent of any particular coordinate realization. Coordinates are tools for computation; structure is the object of study.

Example (Polynomials as a free linear construction, [10]).

Let K be a field. Consider the set

$$K[x] = \left\{ \sum_{k=0}^{n} a_k x^k \;\middle|\; n \in \mathbb{N},\ a_k \in K \right\},$$

whose elements are called *polynomials* in the formal symbol x with coefficients in K.

The symbol x is not assigned a numerical value; it serves only as a generator of formal expressions. Addition and scalar multiplication are defined coefficientwise:

$$\left(\sum_{k=0}^{n} a_k x^k\right) +_{K[x]} \left(\sum_{k=0}^{m} b_k x^k\right) = \sum_{k=0}^{\max(n,m)} (a_k +_K b_k) x^k,$$

$$\alpha \circ \left(\sum_{k=0}^{n} a_k x^k\right) = \sum_{k=0}^{n} (\alpha \cdot a_k) x^k.$$

With these operations, $K[x]$ becomes a vector space over K.

The family

$$\{1, x, x^2, x^3, \dots\}$$

is linearly independent and generates $K[x]$. Hence it forms a basis of $K[x]$.

Every polynomial can therefore be written uniquely as a linear combination

$$u(x) = a_0 \circ 1 +_{K[x]} a_1 \circ x +_{K[x]} \cdots +_{K[x]} a_n \circ x^n, \qquad a_k \in K.$$

The vector space $K[x]$ is infinite dimensional, since its basis contains infinitely many elements.

Conceptually, $K[x]$ is the vector space freely generated by the formal powers

$$\{x^k\}_{k \geq 0}.$$

No relations exist among these generators other than those required by the axioms of a vector space.

Polynomial multiplication introduces an additional operation on $K[x]$,

$$x^i \cdot x^j = x^{i+j},$$

which is compatible with the linear structure but is not required for the definition of the vector space itself.

Thus, polynomials provide a canonical example of a linear structure generated by formal symbols. The linear framework appears independently of geometry, coordinates, or numerical interpretation. Multiplication enriches this structure, but linearity is primary.

Let E and G be vector spaces over the same field K. A function $\psi : E \to G$ is called a *linear transformation* (also known as *linear map*) if it preserves linear combinations [1]:

$$\psi(\alpha_1 \circ_E x_1 + \cdots + \alpha_k \circ_E x_k) = \alpha_1 \circ_G \psi(x_1) + \cdots + \alpha_k \circ_G \psi(x_k).$$

Linear transformations preserve the algebraic structure of vector spaces independently of any choice of basis.

Definition (Isomorphism). Let E and F be vector spaces over the same field K. A linear transformation $\varphi : E \to F$ is called an *isomorphism* if for every $y \in F$ there exists a unique $x \in E$ such that $\varphi(x) = y$, and $\varphi(x_1) = \varphi(x_2)$ implies $x_1 = x_2$. In this case, we say that E and F are *isomorphic*.

An isomorphism preserves all linear structure: addition, scalar multiplication, linear independence, dimension, and the validity of linear relations. Hence, isomorphic vector spaces are structurally identical.

We define the *kernel* of ψ by

$$\ker(\psi) = \{x \in E : \psi(x) = 0\},$$

that is, the set of vectors in E that the transformation sends to the zero vector of G. We define the *image* of ψ by

$$\mathrm{Im}(\psi) = \{\psi(x) : x \in E\} \subseteq G$$

that is, the image is the set of values in G that ψ actually attains.

If E is finite-dimensional, then

$$\dim E = \dim \ker(\psi) + \dim \mathrm{Im}(\psi),$$

a relation known as the rank–nullity theorem.

We say that a linear transformation ψ is *injective* if $\psi(x) = \psi(y) \Rightarrow x = y$ (equivalently, $\ker(\psi) = \{0\}$), and *surjective* if for every $g \in G$ there exists $x \in E$ such that $\psi(x) = g$ (equivalently, $\mathrm{Im}(\psi) = G$).

Let $B_E = \{e_1, \ldots, e_n\}$ and $B_G = \{g_1, \ldots, g_m\}$ be bases of E and G. Each image $\psi(e_i)$ can be written uniquely as $\psi(e_i) = \alpha_{1i} \circ g_1 +_G \cdots +_G \alpha_{mi} \circ g_m$. The scalars α_{ji} form an $m \times n$ rectangular array

$$A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2n} \\ \vdots & \vdots & & \vdots \\ \alpha_{m1} & \alpha_{m2} & \ldots & \alpha_{mn} \end{bmatrix}.$$

and its abbreviated form

$$A = (\alpha_{ij}),$$

called the *matrix of ψ relative to the chosen bases.*

Note (Matrix as a function, [1]).

Let $S \neq \varnothing$ be a set and let $m, n \geq 1$. An $m \times n$ *matrix with entries in* S is a function

$$M : \{1, \ldots, m\} \times \{1, \ldots, n\} \longrightarrow S.$$

We write $a_{ij} = M(i, j)$ and abbreviate M by $M = (a_{ij})$ (or $M = (a_{ij})_{m \times n}$) when no confusion can arise.

Specialization. When $S = K$ is a field, the set of all $m \times n$ matrices over K is denoted by $M_{m \times n}(K)$.

Let $A = (\alpha_{ij})$ be an $m \times n$ matrix over a field K.

For each $i = 1, \ldots, m$, the ith row

$$(\alpha_{i1}, \ldots, \alpha_{in})$$

is called a *row vector* of A. It is naturally regarded as an element of K^n.

For each $j = 1, \ldots, n$, the jth column

$$(\alpha_{1j}, \ldots, \alpha_{mj})^T$$

is called a *column vector* of A. It is naturally regarded as an element of K^m.

When A represents a linear transformation $\psi : E \to G$ with respect to chosen bases, the jth column of A is precisely the coordinate vector of $\psi(e_j)$ in the basis of G.

Matrices are concrete representations of linear transformations once bases (reference systems) have been chosen.

Given a matrix

$$A = (\alpha_{ij}) \in M_{m \times n}(K),$$

its *transpose* is the matrix

$$A^T = (\alpha_{ji}) \in M_{n \times m}(K),$$

obtained by interchanging rows and columns.

Thus the ith row of A becomes the ith column of A^T, and the jth column of A becomes the jth row of A^T.

The transpose operation exchanges the roles of rows and columns. In particular, if a matrix represents a linear map $\psi : K^n \to K^m$, then its transpose has the reversed size and may be viewed, purely at the level of arrays, as interchanging the domain and codomain indices.

In many practical cases, a vector does not merely represent an abstract element of a vector space, but the *state* of a system relative to a chosen basis. We will refer to such a choice of basis as a *reference system.*

Let E be a finite-dimensional vector space and let

$$B = \{e_1, \ldots, e_n\}$$

be a basis of E. Any vector $x \in E$ can be written uniquely as

$$x = \alpha_1 \circ e_1 +_E \cdots +_E \alpha_n \circ e_n$$

The scalars $\alpha_1, \ldots, \alpha_n$ are the *coordinates* of the state x relative to the reference system B. Equivalently, we may collect them into the coordinate tuple $(\alpha_1, \ldots, \alpha_n)$. Writing these coordinates as a column vector,

$$[x]_B = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix},$$

is a notational convenience: it encodes the action of linear combinations relative to the chosen basis.

Suppose now that a second reference system

$$B' = \{e'_1, \ldots, e'_n\}$$

is chosen for the same space E. The vector x is the same abstract object, but its coordinates relative to B' form a different column vector $[x]_{B'}$.

The passage from $[x]_B$ to $[x]_{B'}$ is not arbitrary. It is determined by a linear transformation

$$T : K^n \longrightarrow K^n,$$

whose matrix depends only on the relation between the two bases. Thus,

$$[x]_{B'} = A\,[x]_B,$$

where A is the matrix of the change of reference system.

This is the operational origin of matrix–vector multiplication: a matrix acts on a column vector because it represents a linear transformation acting on the coordinate representation of a state.

In coordinates, each row of the matrix represents the coordinate expression of a linear functional on K^n (we will formalize this identification when discussing dual spaces), and matrix–vector multiplication consists precisely in applying each of these functionals to the coordinate vector. Concretely, if A is an $m \times n$ matrix and x is a column vector in K^n, then the ith component of Ax is the value of the functional given by the ith row of A applied to x; this is defined only when x has n components, i.e., only when the number of columns of A equals the number of rows of x.

The requirement that the number of columns of A equal the number of components of $[x]_B$ is not a convention; it expresses the fact that the codomain of one linear map must match the domain of the next.

In this sense, column vectors represent states, matrices represent transformations between reference systems, and matrix multiplication represents successive changes of coordinates. The familiar computational rules are therefore consequences of the abstract structure introduced earlier.

This abstract formulation is independent of geometry, yet it underlies most quantitative models in science and engineering. In modern machine learning, for example, embeddings and weight matrices are linear transformations on high-dimensional vector spaces (as we will see in Chapter III), composed with non-linear activations. Even where non-linearity dominates, linear transformations provide the structural backbone—the universality of a man-made linear structure, even though the universe itself is not linear.

Note

Operationally, a system of linear equations of the form

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n = b_m. \end{cases}$$

is treated as a collection of equations to be solved for the unknowns $x_1, \ldots, x_n$.

From the abstract point of view, this system is nothing more than the coordinate expression of a linear transformation.

Indeed, let

$$\psi : K^n \longrightarrow K^m$$

be the linear transformation whose matrix relative to the canonical bases is

$$A = \begin{bmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \ldots & a_{mn} \end{bmatrix}.$$

Writing

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix},$$

the system above is equivalently expressed as the single equation

$$\psi(x) = b, \quad \text{or, in coordinates,} \quad Ax = b.$$

So, a system of linear equations specifies a vector $b \in K^m$ and asks whether it lies in the image of a linear transformation $\psi : K^n \to K^m$. Questions of existence, uniqueness, or multiplicity of solutions correspond to geometric properties of this map, such as injectivity and surjectivity, and are independent of any particular algorithm used to compute them.

Algorithms such as Gaussian elimination are methods for finding a vector $x \in K^n$ *that satisfies* $Ax = b$ relative to a chosen reference system (basis). From the abstract perspective, they are simply procedures for exploring the image and kernel of a linear transformation in coordinates. That is, Gaussian elimination exploits the structure of the linear map encoded by the matrix A to determine whether a solution exists, whether it is unique, and how to express all possible solutions, all while working entirely in the chosen coordinate system.

The set of all linear transformations from E into the base field K is denoted by

$$E^* = L(E, K)$$

and is called the *dual space* of E. An element $x^* \in E^*$ is called a *linear functional*. Thus,

$$x^* : E \to K$$

satisfies

$$x^*(\alpha_1 \circ_E x_1 +_E \cdots +_E \alpha_k \circ_E x_k) = \alpha_1 x^*(x_1) + \cdots + \alpha_k x^*(x_k).$$

Linear functionals assign scalar values to vectors while preserving linear structure.

If E is finite-dimensional with basis $B_E = \{e_1, \ldots, e_n\}$, then every $x^* \in E^*$ is uniquely determined by

$$x^*(e_1), \ldots, x^*(e_n).$$

These scalars form a row vector

$$[\, x^*(e_1) \ \ldots \ x^*(e_n) \,].$$

In contrast, vectors in E are represented as column vectors. These two objects live in different spaces: E and E^*.

Column vectors represent elements of a vector space E, whereas row vectors represent linear functionals, elements of the dual space E^*. These are algebraically different objects, even though they may have the same number of components.

In many situations, however, a vector is used to define a linear functional. This requires additional structure (Chapter II).

Matrix multiplication is the coordinate expression of *composition of linear maps*.

Note (Generalized matrix product, [11]).

There is a useful abstraction of matrix multiplication that makes explicit which algebraic properties are really being used. If A is an $m \times n$ matrix and B is an $n \times s$ matrix, and if $\circ$ and $\bullet$ are binary operations such that $\circ$ is associative, one may define the *generalized matrix product*

$$A \underset{\bullet}{\overset{\circ}{}} B$$

to be the $m \times s$ matrix C whose entries are

$$C_{ij} = (A_{i1} \bullet B_{1j}) \circ (A_{i2} \bullet B_{2j}) \circ \cdots \circ (A_{in} \bullet B_{nj}), \qquad 1 \le i \le m,\ 1 \le j \le s.$$

The ordinary matrix product is obtained when $\circ$ is $+$ and $\bullet$ is $\cdot$.

A column vector represents an element of K^n, a row vector represents an element of $(K^n)^*$, and an $m \times n$ matrix represents a linear map

$$K^n \xrightarrow{\psi} K^m.$$

A product such as

$$[\, x^* \,]\, A$$

is defined because it corresponds to the composition

$$K^n \xrightarrow{\psi} K^m \xrightarrow{x^*} K$$

The dimensions match precisely because the codomain of the first map equals the domain of the second.

By contrast, a product of two row vectors would attempt to compose

$$K^n \to K \quad \text{with} \quad K^n \to K,$$

which is meaningless: the output of the first map does not lie in the domain of the second. The product is therefore undefined by necessity.

Matrix dimensions encode compatibility of domains and codomains. One may multiply matrices if and only if the corresponding linear maps can be composed.

Concrete numerical example.
Let

$$x^*(x, y) = 3x + 2y$$

be a linear functional on K^2. Relative to the canonical basis, it is represented by the row vector

$$[\, 3 \;\; 2 \,].$$

Let

$$A = \begin{bmatrix} 2 & 3 \\ 5 & 4 \end{bmatrix}$$

be the matrix of a linear transformation $\psi : K^2 \to K^2$.
The product

$$[\, 3 \;\; 2 \,]A = [\, 3 \;\; 2 \,] \begin{bmatrix} 2 & 3 \\ 5 & 4 \end{bmatrix} = [\, 19 \;\; 17 \,]$$

is defined because it represents the composition

$$K^2 \xrightarrow{\psi} K^2 \xrightarrow{x^*} K.$$

The result is again a row vector, corresponding to the linear functional

$$(x, y) \longmapsto 19x + 17y.$$

By contrast, the product

$$[3 \;\; 2][a \;\; b]$$

is undefined because it would attempt to compose two maps of the form

$$K^2 \to K,$$

which is impossible: the output of the first map is a scalar, not a vector in K^2.

Conceptual summary. The structures introduced in this chapter—vector spaces, linear transformations, bases, coordinate systems, and dual spaces—are all facets of the same underlying linear framework. Matrices encode linear transformations relative to a chosen basis, column vectors encode states relative to a reference system, and systems of linear equations are simply coordinate expressions of these transformations asking whether a vector lies in the image of a map. Algorithms such as Gaussian elimination, while essential for computation, do not create new structures: they explore the coordinate-level consequences of the abstract linear relationships already present. Together, these concepts illustrate how abstract algebraic definitions give rise to the familiar computational and geometric tools used in science and engineering.

Chapter 2

Geometry from Linear Structure

In Chapter I, vector spaces were introduced as purely algebraic objects. A vector was an abstract element of a set equipped with addition and scalar multiplication, and linear transformations were functions preserving this structure. No notion of length, angle, distance, or orthogonality was assumed or required.

Algebra alone allows us to decide whether vectors are independent, whether a set spans a space, or whether a linear transformation is injective or surjective. However, it does not allow us to ask geometric questions. In many situations, it is not enough to know *which* vectors are possible; we also need to know how *large* they are, how *close* they are to one another, or how *aligned* they are. These notions are expressed through length, distance, and angle. Without additional structure, the language of vector spaces does not distinguish between "larger" and "smaller," nor does it allow one to speak of perpendicularity or of the "best approximation" of one vector by others (as in projections and least squares). There is no intrinsic way, within a bare vector space, to compare the size of two vectors or to determine whether they point in perpendicular directions.

Geometry begins only when additional structure is imposed.

It is useful now to recall how geometric space is commonly represented in coordinates. Given sets $A_1, \ldots, A_n$, their *Cartesian product* (also known as *Product Set*) is

$$A_1 \times \cdots \times A_n = \{(x_1, \ldots, x_n) \mid x_i \in A_i\}.$$

In particular, the coordinate space $\mathbb{R}^n$ is the Cartesian product of n copies of $\mathbb{R}$. This construction by itself is not yet geometry: it provides a convenient *model* in which "points" are represented by tuples of real numbers. A geometry emerges only after we specify an additional rule that allows us to measure and compare such tuples.

A central theme of this document is that geometry is not a replacement for linear structure, but an enrichment of it. The vector spaces introduced earlier remain unchanged; what changes is that they are now equipped with a rule that allows vectors to interact in a way that produces scalar quantities.

Informally, this rule takes two vectors and returns a scalar. From this single operation, notions such as length, angle, and distance emerge naturally.

To make this interaction precise, we introduce the notion of a bilinear map.

Let E and G be vector spaces over the same field K. A mapping

$$B : E \times G \longrightarrow K$$

is called *bilinear* if it is linear in each argument separately. That is, for fixed $y \in G$, the map

$$x \longmapsto B(x, y)$$

is a linear functional on E, and for fixed $x \in E$, the map

$$y \longmapsto B(x, y)$$

is a linear functional on G.

Explicitly, for all $x_1, x_2 \in E$, $y_1, y_2 \in G$, and all scalars $\alpha \in K$,

$$B(x_1 + x_2, y) = B(x_1, y) + B(x_2, y), \qquad B(\alpha x, y) = \alpha B(x, y),$$

$$B(x, y_1 + y_2) = B(x, y_1) + B(x, y_2), \qquad B(x, \alpha y) = \alpha B(x, y).$$

When $E = G$, such a mapping is called a *bilinear form* on E.

At this level, no geometric interpretation is assumed. A bilinear form is merely a rule that assigns a scalar to an ordered pair of vectors, subject to linearity in each argument. It need not be symmetric, positive, or related to any notion of length or angle.

Nevertheless, bilinear forms play a fundamental structural role. They provide a systematic way to associate vectors with linear functionals and to construct scalar quantities from pairs of vectors.

Only after additional conditions are imposed on a bilinear form do familiar geometric notions emerge. These special cases will be introduced later.

Once such a rule is fixed, vectors acquire measurable properties. Two vectors may be declared orthogonal. A vector may be assigned a length. The distance between two points may be defined in terms of the vectors connecting them.

None of these concepts exist prior to the introduction of this additional structure.

As in Chapter I, coordinates play no fundamental role. Geometric quantities must be independent of any particular reference system. A formula that changes when the basis is changed describes a representation, not a geometric object.

Only after a basis is chosen do familiar coordinate expressions appear. Sums of squares, dot products, and quadratic equations arise as coordinate-level manifestations of the underlying structure, not as definitions of it.

This point of view explains why objects such as circles, spheres, and ellipsoids can be described by algebraic equations, yet retain a geometric meaning that is invariant under changes of coordinates. The equations depend on the chosen basis; the geometry does not.

The goal is therefore to show how geometry emerges from linear structure once a suitable interaction between vectors is specified.

In later parts of this document, this interaction will be generalized further. When the geometric rule itself varies from point to point, the resulting structures lead naturally to curved spaces and manifolds. For now, the focus remains on understanding how familiar geometric notions arise from linear spaces endowed with additional structure.

The central object of this work is the linear transformation. Before introducing geometric notions such as length, angle, and distance, we make explicit a fundamental construction associated with every linear transformation: its transpose.

Let E and G be vector spaces over the same field K, and let

$$\psi : E \longrightarrow G$$

be a linear transformation. Recall that the dual spaces E^* and G^* consist of all linear functionals on E and G, respectively.

For each $g^* \in G^*$, the composition

$$g^* \circ \psi : E \longrightarrow K$$

is a linear functional on E. This defines a mapping

$$\psi^* : G^* \longrightarrow E^*, \qquad \psi^*(g^*) = g^* \circ \psi.$$

The mapping ψ^* is linear and is called the *transpose* (or dual transformation) of ψ. The direction of the arrow is reversed:

$$E \xrightarrow{\psi} G \quad \Longrightarrow \quad G^* \xrightarrow{\psi^*} E^*.$$

This construction is intrinsic: it depends only on the linear transformation ψ and does not require any choice of basis, coordinates, or inner product.

When bases are chosen for E and G, and the corresponding dual bases are chosen for E^* and G^*, the linear transformation ψ is represented by a matrix A, and the dual transformation ψ^* is represented by the transposed matrix A^T. Thus, the familiar matrix transpose is the coordinate expression of the intrinsic dual transformation ψ^*.

The interaction between vectors that gives rise to geometry is not arbitrary. Its algebraic prototype is the notion of a bilinear form.

Let E be a vector space over a field K. A *bilinear form* on E is a function

$$B : E \times E \longrightarrow K$$

that is linear in each argument separately. That is, for all $x, x', y, y' \in E$ and all $\lambda \in K$,

$$B(x + x', y) = B(x, y) + B(x', y), \qquad B(\lambda x, y) = \lambda B(x, y),$$

and similarly in the second argument.

Bilinear forms arise naturally from linear transformations. If

$$\psi : E \longrightarrow E^*$$

is a linear mapping, then the formula

$$B(x, y) = \psi(x)(y)$$

defines a bilinear form on E. Conversely, every bilinear form determines such a mapping. Therefore, bilinear forms encode a controlled way in which vectors may interact to produce scalars.

At this level, no geometry is present. A bilinear form may be degenerate, asymmetric, or indefinite. It allows vectors to interact, but it does not yet measure anything.

Geometry appears when the bilinear form satisfies additional constraints.

Up to this point, the scalar field K has been arbitrary. Pure linear structure does not require the ability to compare scalars. Geometry, however, depends on positivity: expressions such as $\langle x, x\rangle > 0$ and inequalities of the form $r > 0$ presuppose an order relation. For this reason, we now restrict attention to the real number system

$$\Big\langle \mathbb{R}, +, \cdot, \leq \Big\rangle,$$

viewed as an ordered field. It is precisely the order on $\mathbb{R}$ that allows lengths, distances, and orthogonality to acquire geometric meaning.

An *inner product* on a vector space E over $K = \mathbb{R}$ or $\mathbb{C}$ is a bilinear (or sesquilinear) form

$$\langle \cdot, \cdot \rangle : E \times E \longrightarrow K$$

that is symmetric (or Hermitian), positive, and non-degenerate. These properties ensure that

$$\langle x, x\rangle > 0 \quad \text{for all } x \neq 0.$$

The inner product does more than assign lengths. It establishes a canonical identification between the vector space and its dual.

For each vector $x \in E$, define the mapping

$$\Phi(x) : E \longrightarrow K, \qquad \Phi(x)(y) = \langle x, y\rangle.$$

By bilinearity, $\Phi(x)$ is a linear functional on E. Thus, we obtain a mapping

$$\Phi : E \longrightarrow E^*.$$

If the inner product is non-degenerate, this mapping is injective. In finite dimensions, since $\dim E = \dim E^*$, it is therefore an isomorphism.

Hence, once an inner product is fixed, every vector determines a unique linear functional, and every linear functional arises from a unique vector. The distinction between E and E^* disappears at the structural level.

It is precisely this identification that allows the transpose of a linear transformation to be regarded as an operator on E itself. When geometry is introduced, duality becomes internal.

Once an inner product is fixed, vectors acquire length. The norm of a vector $x \in E$ is defined by

$$\|x\| = \sqrt{\langle x, x \rangle}.$$

So, the vector space E introduced in Chapter I is now equipped with additional geometric structure

$$\Big\langle \langle K, +, \cdot \rangle,\ \langle E, + \rangle,\ \circ,\ \langle \cdot, \cdot \rangle \Big\rangle$$

The inner product is not required for linearity itself, but it allows the introduction of geometric notions such as length, angle, orthogonality, and adjoint transformations.

Algebraic expansion of the norm.
For any vectors $x, y \in E$, the square of the norm of their sum satisfies

$$\|x + y\|^2 = \langle x + y, x + y \rangle.$$

By bilinearity and symmetry of the inner product,

$$\langle x + y, x + y \rangle = \langle x, x \rangle + \langle x, y \rangle + \langle y, x \rangle + \langle y, y \rangle = \|x\|^2 + 2\langle x, y \rangle + \|y\|^2.$$

Thus,

$$\|x + y\|^2 = \|x\|^2 + \|y\|^2 + 2\langle x, y \rangle.$$

This identity holds in every inner product space. It expresses how the interaction term $\langle x, y \rangle$ measures the deviation from pure additivity of squared lengths.

The inner product also satisfies the inequality

$$|\langle x, y \rangle| \leq \|x\|\, \|y\|,$$

known as the *Cauchy–Schwarz inequality*. As a consequence,

$$\|x + y\|^2 \leq \|x\|^2 + \|y\|^2 + 2\|x\|\, \|y\| = (\|x\| + \|y\|)^2.$$

Taking square roots yields the *triangle inequality*

$$\|x + y\| \leq \|x\| + \|y\|.$$

The *Pythagorean identity* arises as the special case $\langle x, y \rangle = 0$, in which the interaction term vanishes and

$$\|x + y\|^2 = \|x\|^2 + \|y\|^2.$$

Orthogonality also becomes meaningful. Two vectors $x, y \in E$ are said to be orthogonal if

$$\langle x, y \rangle = 0.$$

The order of the vectors is irrelevant. Symmetry of the inner product implies

$$\langle x, y \rangle = \langle y, x \rangle,$$

so orthogonality is a mutual relation.

The zero vector is orthogonal to every vector in E. Indeed, for any $v \in E$,

$$\langle 0, v \rangle = \langle 0 \cdot v, v \rangle = 0 \langle v, v \rangle = 0.$$

Moreover, the zero vector is the only vector with this property.

Thus, orthogonality is precisely the condition under which the interaction term vanishes. In that case,

$$\|x + y\|^2 = \|x\|^2 + \|y\|^2.$$

Squared lengths become additive exactly when the vectors do not interact.

Interlude: Invariant Directions

Up to this point, linear transformations have been treated algebraically: a map $\psi : E \to E$ sends vectors to vectors while preserving linear structure. No geometric meaning has yet been attached to this action.

Once an inner product is introduced, however, the action of a linear transformation can be examined geometrically. Given a vector $x \in E$, the image $\psi(x)$ may differ from x in two independent ways: its length may change, and its direction may change.

For a generic vector, both effects occur simultaneously. Most vectors are stretched and rotated.

A remarkable phenomenon occurs when there exists a nonzero vector $x \in E$ such that

$$\psi(x) = \lambda x$$

for some scalar λ. In this case, the transformation acts on x by pure scaling. The direction of x is preserved.

Such a vector is called an *eigenvector* of ψ, and the scalar λ is its corresponding *eigenvalue*. Eigenvectors identify directions in the space that are geometrically invariant under the transformation.

This invariance is an intrinsic property of the linear transformation itself. Changing the basis may alter the representation of ψ, but it cannot destroy or create eigenvectors.

Under the identification $E \cong E^*$ induced by the inner product, the transpose ψ^* corresponds to a linear operator on E, called the adjoint of ψ.

When a linear transformation arises from a symmetric bilinear form (or equivalently, from a self-adjoint operator), its eigenvectors corresponding to distinct eigenvalues are orthogonal. These directions define a preferred coordinate system dictated by the geometry, not by arbitrary choice.

It is in this sense that eigenvectors serve as the principal axes of the ellipsoids associated with quadratic forms. Diagonalization is the act of aligning coordinates with the invariant directions already present in the structure.

With this geometric viewpoint in place, we return to the central construction: the inner product induces a norm, and the norm induces a distance.

Distance and the emergence of geometry.

Once an inner product is available on a vector space E, it determines a norm, and from the norm we obtain a notion of distance between vectors,

$$d(x, y) = \|x - y\|.$$

At this point, no new structure is introduced: distance is merely a reformulation of algebraic data already present.

This distance allows us to describe proximity. For a vector $x \in E$ and a real number $r > 0$, we consider the set

$$B(x, r) = \{\, y \in E \mid d(x, y) < r \,\}.$$

Such a set is called an *open ball* centered at x with radius r. Geometrically, it consists of all vectors that lie closer to x than the prescribed threshold.

The strict inequality is essential. It guarantees that every point inside the ball admits further nearby points, so that no boundary is included. This "breathing room" is what ultimately allows us to speak of continuity and deformation.

The collection of all open balls determines which subsets of E should be called open: a set is open if, around each of its points, it contains some open ball. In this way, distance generates a notion of openness without any additional axioms. The resulting structure is what is known as a *topology* on E.

A *topological space* is an ordered pair (X, τ), where X is a set and τ is a collection of subsets of X. For τ to be a *topology*, it must satisfy the following axioms:

1. $\emptyset \in \tau$ and $X \in \tau$.
2. The union of any collection of sets in τ is also in τ.
3. The intersection of any *finite* number of sets in τ is also in τ.

The elements of τ are called *open sets*.

Thus, topology does not appear here as an independent theory, but as a structural consequence of the metric induced by the inner product. Geometry gives rise to distance, distance gives rise to neighborhoods, and neighborhoods give rise to topological structure.

Geometric interpretation.

In $\mathbb{R}^n$ equipped with its standard inner product, open balls correspond to the familiar spheres of Euclidean geometry. More general norms deform these balls into ellipsoidal shapes, reflecting anisotropic scaling along different directions of the space. In all cases, each point

$$x = (x_1, \ldots, x_n)$$

is a vector of the ambient vector space, and the geometric objects considered are subsets of that space defined by distance relations among its vectors.

Outlook.

In contemporary applications, such as representation learning, elements of a set (words, images, signals) are mapped into a high-dimensional vector space. Their relative positions, measured by a chosen metric, encode similarity. This metric need not be Euclidean: in practice it may be designed to reflect domain-specific invariances or even learned from data. Neighborhoods of meaning are then described by open balls, and learning can be interpreted as the discovery of a geometric and topological organization underlying the data.

Once a norm has been introduced, certain geometric sets arise as direct consequences of the structure and require no additional assumptions.

Let $(E, \langle \cdot, \cdot \rangle)$ be an inner product space and let $\|x\| = \sqrt{\langle x, x \rangle}$ be the associated norm. For a fixed vector $x_0 \in E$ and a scalar $r > 0$, the *closed ball* of radius r centered at x_0 is defined by

$$B_r(x_0) = \{\, x \in E \mid \|x - x_0\| \leq r \,\}.$$

The corresponding *sphere* is the boundary of this set,

$$S_r(x_0) = \{\, x \in E \mid \|x - x_0\| = r \,\}.$$

Remark (Coordinate description in $\mathbb{R}^n$): When $E = \mathbb{R}^n$ is equipped with its standard inner product and we use the standard coordinates $x = (x_1, \ldots, x_n)$, the same intrinsic definitions take the familiar "sum of squares" form. For a ball centered at the origin,

$$B_r(0) = \left\{ x \in \mathbb{R}^n \;\middle|\; x_1^2 + \cdots + x_n^2 \leq r^2 \right\},$$

and its boundary sphere is

$$S_r(0) = \left\{ x \in \mathbb{R}^n \;\middle|\; x_1^2 + \cdots + x_n^2 = r^2 \right\}.$$

This is simply the coordinate expression of $\|x\| = \sqrt{x_1^2 + \cdots + x_n^2}$.

These definitions are intrinsic and do not depend on any choice of coordinates or basis.

Remark (Symmetry): In the Euclidean case $E = \mathbb{R}^n$ with its standard inner product, the ball is invariant under orthogonal transformations. If $Q \in O(n)$, then $\|Qx\| = \|x\|$, and hence

$$Q\big(B_r(0)\big) = B_r(0).$$

More generally, any isometry of an inner product space carries balls to balls.

Remark (Slicing in $\mathbb{R}^n$): In $\mathbb{R}^n$, fixing the last coordinate $x_n = t$ shows that the cross-section of a radius-r ball by the hyperplane $x_n = t$ is an $(n-1)$-dimensional ball of radius $\sqrt{r^2 - t^2}$. This observation is the geometric content behind the common "integrate by slices" approach to n-dimensional volume.

Remark (Volume of Euclidean balls): In $\mathbb{R}^n$, the n-dimensional volume of a Euclidean ball scales as a constant times r^n:

$$\mathrm{Vol}_n\big(B_r(0)\big) = \omega_n\, r^n,$$

where $\omega_n = \mathrm{Vol}_n\big(B_1(0)\big)$ is the volume of the unit ball. A closed form is

$$\omega_n = \frac{\pi^{n/2}}{\Gamma\big(\frac{n}{2}+1\big)}.$$

In particular, $\omega_1 = 2$, $\omega_2 = \pi$, and $\omega_3 = \frac{4\pi}{3}$.

Remark (A high-dimensional effect): Although $B_1(0) \subset \mathbb{R}^n$ always contains many points, its n-dimensional volume satisfies $\omega_n \to 0$ as $n \to \infty$. This illustrates a basic phenomenon of high-dimensional geometry: sets that look large in low dimensions can occupy a vanishing fraction of ambient volume as the dimension increases.

More generally, let $B : E \times E \to K$ be a symmetric positive-definite bilinear form. The set

$$\mathcal{E} = \{\, x \in E \mid B(x - x_0, x - x_0) \le r^2 \,\}$$

is called an *ellipsoid*. Thus, an ellipsoid is the unit ball associated with a bilinear form, or equivalently, with a modified inner product on E.

When the bilinear form B is symmetric and positive-definite, there exists a unique self-adjoint linear operator $A : E \to E$ such that

$$B(x, y) = \langle Ax, y \rangle$$

for all $x, y \in E$.

The operator A encodes the geometry of the ellipsoid. Since A is self-adjoint on a finite-dimensional inner product space, it admits an orthonormal basis of eigenvectors. If $\{e_1, \ldots, e_n\}$ is such a basis and $Ae_i = \lambda_i e_i$ with $\lambda_i > 0$, then the defining inequality

$$B(x - x_0, x - x_0) \le r^2$$

becomes

$$\sum_{i=1}^{n} \lambda_i\, \xi_i^2 \le r^2,$$

where $\xi_i = \langle x - x_0, e_i \rangle$ are the coordinates in this eigenbasis.

Thus, the principal axes of the ellipsoid are precisely the eigenvectors of A, and the eigenvalues determine the scaling along those directions. Diagonalization does not create the axes; it merely reveals the invariant directions already present in the structure.

Historical remark (Angles and invariance).

Before eigenvectors were understood as invariant directions, their detection was often indirect. In two dimensions, determining whether a transformation preserves a direction reduces to detecting whether it preserves an angle.

This problem led naturally to the computation of arctangents. In the eighteenth century, J.H. Lambert introduced the continued fraction

$$\arctan(z) = \cfrac{z}{1 + \cfrac{z^2}{3 + \cfrac{4z^2}{5 + \cfrac{9z^2}{7 + \cdots}}}},$$

as part of his work on orbital mechanics and rotational phenomena.

From the modern point of view, this analytic effort reflects a geometric question: does a linear transformation rotate a direction, or does it merely scale it? Eigenvectors provide the structural answer to this question, rendering angle computation auxiliary rather than fundamental.

Accordingly, when a basis is chosen in which the bilinear form is diagonal, this invariant definition reduces to a familiar coordinate expression. In $\mathbb{R}^n$, one obtains

$$\mathcal{E} = \left\{ (x_1, \ldots, x_n) \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} \frac{(x_i - a_i)^2}{h_i^2} \leq r^2 \right\}.$$

Written componentwise, this inequality reads

$$\left(\frac{x_1 - a_1}{h_1}\right)^2 + \left(\frac{x_2 - a_2}{h_2}\right)^2 + \cdots + \left(\frac{x_n - a_n}{h_n}\right)^2 \leq r^2.$$

The equation depends on the chosen coordinates; the ellipsoid itself does not.

Remark (Degenerate cases). The geometric nature of the set defined above depends on the coefficients $h_1, \ldots, h_n$. If $h_1 = \cdots = h_n$, the ellipsoid reduces to an n-dimensional sphere. If one or more of the parameters h_i vanish, the defining quadratic form becomes degenerate and the ellipsoid collapses into a lower-dimensional object: a disk, a line segment, or, in the extreme case, a single point.

From the structural point of view, these degenerations correspond to a loss of rank in the associated quadratic or bilinear form. Thus, changes in the algebraic structure translate directly into changes in geometric dimension.

Remark (Quadratic forms and the chi-square statistic). Expressions of the form defining ellipsoids appear naturally in contexts that are not, at first sight, geometric. A notable example arises in the theory of random number testing, where one encounters the quantity

$$V = \frac{(Y_1 - np_1)^2}{np_1} + \frac{(Y_2 - np_2)^2}{np_2} + \cdots + \frac{(Y_n - np_n)^2}{np_n},$$

known in statistics as the *chi-square statistic* associated with the observed quantities $Y_1, \ldots, Y_n$ and reference values $np_1, \ldots, np_n$.

From the present point of view, this expression is simply a quadratic form on $\mathbb{R}^n$. It measures the squared length of the deviation vector

$$(Y_1 - np_1, \ldots, Y_n - np_n)$$

with respect to a symmetric positive-definite bilinear form whose matrix, in the standard basis, is diagonal with entries $(np_1)^{-1}, \ldots, (np_n)^{-1}$.

Consequently, the sets defined by inequalities of the form

$$V \leq c$$

are ellipsoids centered at the point $(np_1, \ldots, np_n)$. The denominators np_i determine the anisotropic scaling along the coordinate directions, exactly as in the general ellipsoid

$$\sum_{i=1}^{n} \frac{(x_i - a_i)^2}{h_i^2} \leq r^2.$$

Thus, before any probabilistic interpretation is imposed, the chi-square expression is already a geometric object: it is the squared norm induced by a particular inner product on $\mathbb{R}^n$. Statistical theory makes use of this geometry, but does not create it.

Remark. Throughout this discussion, the ambient space $E = \mathbb{R}^n$ is a vector space in the sense of Chapter I. Each element

$$x = (x_1, \ldots, x_n)$$

appearing in the definitions of balls, spheres, or ellipsoids is therefore a vector of E.

However, the sets defined by these expressions are *subsets* of E selected by quadratic constraints. They are not vector spaces themselves, since they are not closed under addition or scalar multiplication.

Conceptual summary. Vector spaces provide the stage. Linear transformations describe allowable changes of state. Geometry enters only when vectors are permitted to interact through a structured bilinear rule that produces scalars. From this interaction, lengths, angles, distances, and geometric shapes emerge. The transition from algebra to geometry is therefore not a leap, but a controlled enrichment of structure. The book [3] is a wonderful reference on the topology of metric spaces.

Chapter 3

Tensors as Multilinear Maps

Linear transformations are functions that are linear in a single argument.

In Chapter I, we studied linear transformations between vector spaces.

Recall (see Chapter I) that a linear map $T : E \to F$ preserves linear combinations (equivalently, it is additive and homogeneous in its single argument).

But more generally, one can consider functions that are linear in several arguments, called *multilinear maps*.

Linearity encodes the idea that the map respects the algebraic structure of the vector space. Many constructions in mathematics (and, in practice, in physics and machine learning), however, depend on functions that take *several* vector arguments simultaneously, while remaining linear in each argument when the others are held fixed.

This observation leads naturally to the notion of multilinearity.

Let $E_1, E_2, \ldots, E_n$ and F be vector spaces over the same field K.

Definition 3.1 (Multilinear map)**.** A map

$$T : E_1 \times E_2 \times \cdots \times E_n \to F$$

is called *multilinear* if it is linear in each argument separately. That is, for each index $i \in \{1, \ldots, n\}$, and for all vectors

$$x_i, y_i \in E_i, \quad \alpha \in K,$$

we have

$$T(x_1, \ldots, x_i +_{E_i} y_i, \ldots, x_n) = T(x_1, \ldots, x_i, \ldots, x_n) +_F T(x_1, \ldots, y_i, \ldots, x_n),$$

and

$$T(x_1, \ldots, \alpha \circ x_i, \ldots, x_n) = \alpha \circ T(x_1, \ldots, x_i, \ldots, x_n),$$

with all other arguments held fixed.

When $n = 1$, a multilinear map is simply a linear map. Thus multilinear maps are not a new kind of object, but a direct generalization of linear transformations.

Multilinear maps arise naturally in many contexts: bilinear products, pairings between vectors and linear functionals, and higher-order interactions that cannot be reduced to a single input. Their systematic study leads to the concept of tensors.

Informally, a tensor is an object designed to encode multilinear behavior in an intrinsic and coordinate-free way.

There are two equivalent viewpoints that will be used throughout this document. The first treats tensors directly as multilinear maps.

Let E be a vector space over K, and let E^* denote its dual space, that is, the vector space of linear maps from E to K.

Definition 3.2 (Tensor as a multilinear map)**.** A *tensor of type* (k, ℓ) on E is a multilinear map

$$T : \underbrace{E^* \times \cdots \times E^*}_{k \text{ times}} \times \underbrace{E \times \cdots \times E}_{\ell \text{ times}} \longrightarrow K.$$

Such a tensor takes k linear functionals and ℓ vectors as inputs, and produces a scalar. Linearity is required in each argument separately.

This definition makes precise several familiar cases:

- A scalar in K can be viewed as a tensor of type $(0, 0)$.
- A vector in E corresponds to a tensor of type $(0, 1)$.
- A linear functional in E^* is a tensor of type $(1, 0)$.
- A bilinear form on E is a tensor of type $(0, 2)$.
- If E is finite-dimensional, a linear map $T : E \to E$ can be identified with a tensor of type $(1, 1)$.

Thus vectors and matrices do not disappear in tensor theory; they reappear as the lowest-order cases of a single unified framework.

At this level of abstraction, no coordinates, arrays, or geometric interpretations are involved. A tensor is defined entirely by how it acts on vectors and linear functionals, and by the multilinearity of this action.

In machine learning terms, this abstract view underlies common objects such as embeddings (vectors in a representation space) and parameters (tensors encoding linear or multilinear maps), independent of any chosen coordinate system.

The definition of tensors as multilinear maps is conceptually clear, but it has a practical limitation: Multilinear maps do form vector spaces under pointwise operations. However, they

do not admit a simple composition theory analogous to that of linear maps. The difficulty is that the outputs and inputs of multilinear maps do not naturally match in a way that allows canonical composition. The tensor product resolves this by reducing multilinearity to linearity via a universal construction.

This construction is the *tensor product*.

Let $E_1, \dots, E_n$ be vector spaces over the same field K. Consider the set of all multilinear maps

$$T : E_1 \times \cdots \times E_n \longrightarrow F$$

into an arbitrary vector space F. The tensor product is defined so that every such multilinear map corresponds uniquely to a linear map defined on a single vector space.

Definition 3.3 (Tensor product)**.** The *tensor product* of the vector spaces $E_1, \dots, E_n$ is a vector space, denoted

$$E_1 \otimes \cdots \otimes E_n,$$

together with a multilinear map

$$\otimes : E_1 \times \cdots \times E_n \longrightarrow E_1 \otimes \cdots \otimes E_n,$$

satisfying the following universal property:
For every vector space F and every multilinear map

$$T : E_1 \times \cdots \times E_n \longrightarrow F,$$

there exists a unique linear map

$$\widetilde{T} : E_1 \otimes \cdots \otimes E_n \longrightarrow F$$

such that

$$T(x_1, \dots, x_n) = \widetilde{T}(x_1 \otimes \cdots \otimes x_n)$$

for all $x_i \in E_i$.

This property characterizes the tensor product completely. The elements $x_1 \otimes \cdots \otimes x_n$ are called *simple tensors*. Every element of the tensor product space is a finite linear combination of simple tensors.

The tensor product is therefore not defined by a formula, but by what it *does*: it converts multilinear problems into linear ones in a canonical way.

As a consequence, the study of multilinear maps is equivalent to the study of linear maps defined on tensor products. Specifically,

$$\text{Multilinear maps } E_1 \times \cdots \times E_n \to F \quad \longleftrightarrow \quad \text{Linear maps } E_1 \otimes \cdots \otimes E_n \to F.$$

This equivalence is the conceptual foundation of tensor theory. It explains why tensors can be manipulated algebraically, composed with linear maps, and represented concretely once bases are chosen.

Returning to the case of a single vector space E, a tensor of type (k, ℓ) may equivalently be viewed as a linear map

$$T : \underbrace{E^* \otimes \cdots \otimes E^*}_{k} \otimes \underbrace{E \otimes \cdots \otimes E}_{\ell} \longrightarrow K.$$

Thus tensors are not mysterious higher-dimensional objects. They are linear maps defined on tensor products, and their apparent complexity arises only when coordinates are introduced.

Note: Once bases are chosen, tensor products become finite-dimensional vector spaces, and tensors admit coordinate representations as multidimensional arrays. These representations depend on the chosen bases and do not define the tensor itself.

Let E be a finite-dimensional vector space over K, and let

$$B = \{e_1, \ldots, e_n\}$$

be a basis of E. Its dual basis

$$B^* = \{e^1, \ldots, e^n\} \subset E^*$$

is defined by the relations

$$e^i(e_j) = \delta^i_j,$$

where δ^i_j is the Kronecker delta.

Every vector $x \in E$ admits a unique expansion

$$x = x^i \circ e_i,$$

and every linear functional $\varphi \in E^*$ admits a unique expansion

$$\varphi = \varphi_i \circ e^i.$$

The scalars x^i and φ_i are the *components* of x and φ relative to the chosen bases.

Let now T be a tensor of type (k, ℓ) on E, that is,

$$T : \underbrace{E^* \times \cdots \times E^*}_{k} \times \underbrace{E \times \cdots \times E}_{\ell} \longrightarrow K.$$

The action of T is completely determined by its values on basis elements:

$$T(e^{i_1}, \ldots, e^{i_k}, e_{j_1}, \ldots, e_{j_\ell}) = T^{i_1 \ldots i_k}{}_{j_1 \ldots j_\ell}.$$

These scalars are called the *components of T relative to the basis B*. They encode the tensor entirely once a basis has been fixed.

Given arbitrary arguments

$$(\varphi^1, \dots, \varphi^k, x_1, \dots, x_\ell),$$

multilinearity implies the expansion

$$T(\varphi^1, \dots, \varphi^k, x_1, \dots, x_\ell) = T^{i_1 \dots i_k}{}_{j_1 \dots j_\ell}\, \varphi^1_{i_1} \cdots \varphi^k_{i_k}\, x_1^{j_1} \cdots x_\ell^{j_\ell}.$$

Thus index notation is not an extra language: it is the coordinate expression of multilinearity after a basis is chosen.

In particular, a bilinear map

$$B : E \times F \to G$$

between finite-dimensional vector spaces is represented, once bases are chosen, by components

$$B^k{}_{ij},$$

so that for vectors $x \in E$ and $y \in F$,

$$(B(x,y))^k = B^k{}_{ij}\, x^i y^j.$$

Thus multidimensional arrays of coefficients appearing in numerical algorithms are coordinate representations of tensors encoding multilinear operations.

Once a basis has been fixed, the components

$$T^{i_1 \dots i_k}{}_{j_1 \dots j_\ell}$$

can be arranged in a multidimensional array of scalars.

This array is *not* the tensor itself. It is a coordinate representation of the tensor, in the same way that a column vector represents a vector only after a basis has been chosen.

Changing the basis of E changes the numerical values of the components according to precise transformation rules, while the tensor itself remains unchanged.

Note (Tensors via Transformation Laws).

The abstract definition of a tensor as a multilinear map is intrinsic and coordinate-free. However, there exists an equivalent characterization that emphasizes how components behave under a change of basis.

This viewpoint makes explicit the invariant content that survives changes of coordinates, which is precisely what allows tensorial laws to appear in physics and in machine learning models independent of the chosen representation.

Let $B = \{e_i\}$ and $B' = \{e'_i\}$ be two bases of E, related by

$$e'_i = A^j{}_i\, e_j,$$

where $A^j{}_i$ is an invertible change-of-basis matrix. The corresponding dual bases satisfy

$$e'^i = (A^{-1})^i{}_j \, e^j.$$

Let T be a tensor of type (k, ℓ) on E, with components relative to B given by

$$T^{i_1 \dots i_k}{}_{j_1 \dots j_\ell}.$$

The matrix $A^j{}_i$ expresses the new basis vectors in terms of the old ones, as discussed in Chapter I Accordingly, the transformation rule below describes how the coordinate representation of T changes when passing from the basis B to B'.

Then the components relative to the new basis B' are given by

$$T'^{i_1 \dots i_k}{}_{j_1 \dots j_\ell} = A^{i_1}{}_{p_1} \cdots A^{i_k}{}_{p_k} \, (A^{-1})^{q_1}{}_{j_1} \cdots (A^{-1})^{q_\ell}{}_{j_\ell} \, T^{p_1 \dots p_k}{}_{q_1 \dots q_\ell}.$$

Contravariant indices (upper indices) transform with the matrix A, while covariant indices (lower indices) transform with its inverse. This transformation rule is not an additional assumption. It is a direct consequence of multilinearity and of how vectors and dual vectors transform under a change of basis.

Conversely, suppose that for each choice of basis one assigns a multidimensional array of scalars

$$T^{i_1 \dots i_k}{}_{j_1 \dots j_\ell}$$

in such a way that these arrays are related by the transformation law above whenever the basis is changed. Then these coordinate collections define a unique multilinear map

$$T : \underbrace{E^* \times \cdots \times E^*}_{k} \times \underbrace{E \times \cdots \times E}_{\ell} \longrightarrow K,$$

independent of the chosen basis.

Thus the following two viewpoints are equivalent:

- A tensor is a multilinear map.
- A tensor is a collection of components that transform according to the tensor transformation law.

The first definition emphasizes intrinsic structure. The second emphasizes invariance under change of reference frame. Their equivalence explains why tensor theory plays a central role in geometry and physics: tensorial equations remain valid in every coordinate system, because both sides transform according to the same law.

Thus:

- tensors are abstract multilinear objects;

- arrays are basis-dependent representations;
- indices record how multilinearity decomposes into coordinates.

This distinction is essential for interpreting computational implementations.

Consider a physical system modeled on a discrete two-dimensional grid. At each spatial point (i, j) and at each time step t, a force vector

$$F(t, i, j) \in \mathbb{R}^2$$

is measured, with components (F_x, F_y) relative to a fixed spatial frame.

Let:

- $t \in \{1, \ldots, T\}$ index time,
- $(i, j) \in \{1, \ldots, N\} \times \{1, \ldots, M\}$ index spatial position,
- $\alpha \in \{1, 2\}$ index force components.

The collected data can be written as an array

$$F_{tij\alpha},$$

with shape $(T, N, M, 2)$.

Each index corresponds to an independent modeling choice: time, space, and vector components. The array stores numerical values relative to chosen coordinate systems.

Abstractly, this data can be interpreted as the coordinate representation of a tensor-valued function on a discrete domain. The array itself, however, is a *representation*, not the tensor. The indices (t, i, j) label independent parameters of the model; only the component index α corresponds to a vector slot.

Modern numerical libraries such as TensorFlow.js operate directly on multidimensional arrays of numbers.

For example, a TensorFlow.js tensor with shape $(2, 2, 2, 2)$ is created as:

```
const tensor = tf.tensor([[[[...]]]]);
```

Operations such as addition or elementwise multiplication act on the stored components:

```
tensor1.add(tensor2);
tensor1.mul(tensor2);
```

These operations are performed at the level of coordinate arrays. They do not implement tensor contraction or abstract multilinear composition unless explicitly programmed to do so.

TensorFlow.js tensors are therefore best understood as efficient containers for tensor *representations*. The mathematical tensor exists independently of the data structure used to store its components.

This separation between abstract structure and numerical representation explains both the power and the limitations of tensor-based computation in machine learning and scientific computing.

The fundamental operation that allows tensors to interact is *tensor contraction*. This operation generalizes matrix multiplication and explains the appearance of summation over repeated indices.

Let E be a finite-dimensional vector space over K. Consider a tensor

$$T \in \mathcal{T}^{(k,\ell)}(E)$$

and a tensor

$$S \in \mathcal{T}^{(m,n)}(E).$$

Assume that one covariant slot of T and one contravariant slot of S are selected. Tensor contraction is the operation that pairs these two slots using the natural evaluation map

$$E^* \times E \to K.$$

Definition 3.4 (Tensor contraction)**.** Let

$$T : E^* \times \cdots \times E \times \cdots \to K, \quad S : E^* \times \cdots \times E \times \cdots \to K.$$

The contraction of T and S over one dual–vector pair is the tensor obtained by composing the selected E^* and E slots with the natural evaluation map

$$E^* \times E \longrightarrow K.$$

In coordinates, this composition appears as summation over a basis.

The resulting tensor has type

$$(k + m - 1,\ \ell + n - 1).$$

Contraction reduces the total number of tensor slots by two: one contravariant and one covariant slot are paired and eliminated. No coordinates are required to define this operation.

Fix a basis $\{e_i\}$ of E and its dual basis $\{e^i\}$.

Let

$$T^i{}_j \quad \text{and} \quad S^j{}_k$$

be the components of two tensors of type $(1, 1)$.

The contraction over the index j produces a new tensor with components

$$R^i{}_k = T^i{}_j \, S^j{}_k.$$

Here the repeated index j indicates summation:

$$R^i{}_k = \sum_{j=1}^{n} T^i{}_j \, S^j{}_k.$$

This summation is not an additional rule. It is the coordinate expression of the abstract pairing between E^* and E.

Let $A : E \to E$ and $B : E \to E$ be linear maps. Each can be identified with a tensor of type $(1, 1)$.

Relative to a basis, their components are written as

$$A^i{}_j, \qquad B^j{}_k.$$

The composition $A \circ B : E \to E$ has components

$$(A \circ B)^i{}_k = A^i{}_j \, B^j{}_k.$$

Thus matrix multiplication is exactly tensor contraction between two $(1, 1)$ tensors. Nothing new is happening: the familiar rule

$$(AB)_{ik} = \sum_j A_{ij} B_{jk}$$

is simply the coordinate expression of contraction.

The previous discussion shows that matrix multiplication is the coordinate expression of tensor contraction between two $(1, 1)$ tensors.

At the abstract level we have:

$$\text{Tensor contraction}$$

which, after choosing bases, becomes

$$\text{Index summation} \quad \longrightarrow \quad C_{ik} = \sum_j A_{ij} B_{jk}.$$

This coordinate expression is precisely the rule implemented in matrix multiplication routines.

In numerical linear algebra libraries (such as BLAS), optimized implementations compute this summation efficiently using blocking strategies, cache-aware algorithms, and parallel execution.

Modern graphics processing units (GPUs) go further. They include specialized hardware units designed to evaluate matrix products of small blocks at extremely high throughput.

At the hardware level, a so-called *tensor core* computes operations of the form

$$C_{ik} = \sum_j A_{ij} B_{jk} + D_{ik},$$

that is, a fused multiply–accumulate applied to matrix tiles.

Conceptually, nothing new has appeared. The tensor core performs a tensor contraction: a pairing over one covariant and one contravariant index, expressed in coordinates and realized in silicon.

The chain of interpretation is therefore:

Abstract contraction $\longrightarrow$ Index summation $\longrightarrow$ Matrix multiplication $\longrightarrow$ Optimized linear algebra ker

The apparent technological sophistication of modern machine learning hardware does not replace tensor theory. Rather, it embodies one of its fundamental operations at massive scale.

In this sense, contemporary tensor-processing hardware implements, in physical form, the algebraic operation of contraction introduced above.

The bridge between abstraction and silicon is therefore direct: tensor contraction is an operation now realized by billions of transistors, repeated in parallel across thousands of execution units.

Elementwise array multiplication, as implemented in numerical libraries, is defined by

$$(C)_{i_1 \ldots i_n} = A_{i_1 \ldots i_n} \, B_{i_1 \ldots i_n}.$$

This operation does not pair dual and vector indices. No summation occurs. No slots are eliminated.

Therefore elementwise multiplication is *not* tensor contraction. It is an operation on coordinate arrays, not on tensors as multilinear maps.

Confusing these two operations leads to incorrect interpretations of computational results, especially in machine learning contexts.

The passage from linear maps to tensors is not a departure from linear structure, but its systematic extension to multiple interacting inputs.

- Tensor contraction is an abstract operation defined without coordinates.
- Index summation arises from evaluation of dual–vector pairs.
- Matrix multiplication is a special case of tensor contraction.
- Elementwise array operations are not tensor operations.

Understanding this distinction is essential for interpreting tensors in both mathematics and computation.

A classical example: the inertia tensor [12]

In classical mechanics, the distribution of mass in a rigid body is encoded by a second-order tensor called the *inertia tensor*. Relative to a chosen orthonormal basis of $\mathbb{R}^3$, its components are given by

$$I_{ij} = \int_\Omega \left(\delta_{ij}\|x\|^2 - x_i x_j\right) dm,$$

where Ω is the body and x is the position vector.

The inertia tensor defines a linear relation between angular velocity ω and angular momentum L:

$$L_i = I_{ij}\,\omega^j.$$

Under a change of basis determined by an invertible matrix A, its components transform according to

$$I'_{ij} = A^p{}_i\, A^q{}_j\, I_{pq},$$

which is precisely the transformation law of a tensor of type $(0, 2)$.

Thus, although the numerical entries of I_{ij} depend on the chosen reference frame, the geometric object it represents does not. The physical content — resistance to rotational acceleration — is invariant under change of coordinates.

As in the case of linear maps, tensor contraction is invariant under change of basis. Although its coordinate expression depends on the chosen reference system, the resulting tensor is independent of that choice. Thus tensor theory extends the principle introduced in Chapter I: structure is identified not by coordinates, but by invariance under transformations. Tensors generalize linear structure without abandoning it. They enlarge the domain of linearity while preserving its invariant character.

Chapter 4

Logarithms as Structure-Preserving Transformations

In previous chapters, linear structure was developed independently of coordinates, and geometry was introduced as an enrichment rather than a replacement. The present document continues this philosophy in a different setting: logarithms are treated not as formulas to memorize, but as transformations that preserve and reveal structure.

At the most basic level, a logarithm answers the question: *what exponent produces a given number?* Yet historically and mathematically, logarithms owe their power to something deeper: they convert multiplication into addition.

A straight line is described algebraically by

$$y = mx + b,$$

where the slope m measures the rate of change of y with respect to x. Given two points (x_1, y_1) and (x_2, y_2), the slope is

$$m = \frac{y_2 - y_1}{x_2 - x_1}.$$

This notion of rate of change extends beyond straight lines. For a function $y = f(x)$, the average rate of change over an interval is

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1}.$$

To understand behavior at a single point, we introduce a small increment h and examine

$$\frac{f(a+h) - f(a)}{h}.$$

As $h \to 0$, this expression converges—when it converges—to the derivative

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}.$$

The derivative represents instantaneous change. Geometrically, it is the slope of the tangent line to the graph of f at a.

Near a point x_n, a differentiable function behaves approximately like its tangent line. This gives the linear approximation

$$f(x) \approx f(x_n) + f'(x_n)(x - x_n).$$

This local linearity is the foundation of one of the most important numerical methods in analysis.

This passage from a nonlinear function to its tangent line is an instance of the same principle developed in Chapter II: geometry and analysis become tractable when expressed through linear structure. The derivative replaces a nonlinear transformation by its best linear model at a point. Local complexity is controlled by linear approximation.

Suppose we wish to solve the equation

$$f(x) = 0.$$

Starting from an initial guess x_n, we replace the curve by its tangent line and solve

$$0 = f(x_n) + f'(x_n)(x_{n+1} - x_n),$$

which yields the iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

This is the Newton–Raphson method. It transforms a nonlinear problem into a sequence of linear ones.

Logarithms arise naturally from this perspective. To compute $\ln(x)$, we solve the implicit equation

$$e^y = x.$$

Define

$$f(y) = e^y - x.$$

Then

$$f'(y) = e^y.$$

Newton–Raphson gives

$$y_{n+1} = y_n - \frac{e^{y_n} - x}{e^{y_n}} = y_n + \frac{x - e^{y_n}}{e^{y_n}}.$$

The simplicity of this derivative is not accidental. Choosing base e minimizes algebraic overhead and improves numerical stability. Base-10 logarithms are obtained afterward by scaling:

$$\log_{10}(x) = \frac{\ln(x)}{\ln(10)}.$$

The constant e itself admits several independent constructions. One classical definition arises from compound interest:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e.$$

Another appears in its continued fraction expansion:

$$e = [2; 1, 2, 1, 1, 4, 1, 1, 6, \dots].$$

It can also be written in expanded form:

$$e = 2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{4 + \cdots}}}}}.$$

These representations converge rapidly and are well suited for computation.

Applying Newton–Raphson to compute $\ln(5)$ illustrates the method concretely. Starting from $y_0 = 1.5$, successive iterations converge rapidly to

$$\ln(5) \approx 1.60928.$$

At a deeper level, logarithms admit a geometric interpretation. By definition,

$$\ln(a) = \int_1^a \frac{1}{x}\,dx,$$

the area under the curve $y = \frac{1}{x}$ from 1 to a.
This representation explains why

$$\frac{d}{da}\ln(a) = \frac{1}{a}.$$

Logarithms also arise from first principles using limits:

$$\frac{d}{da}\ln(a) = \lim_{h\to 0}\frac{\ln(a+h) - \ln(a)}{h}.$$

The derivative emerges as a limit of ratios, just as slope does for straight lines.

The most striking feature of the logarithm, however, is structural.

In Chapter I, an isomorphism was defined as a structure-preserving mapping between algebraic systems. The logarithm provides a non-linear but exact realization of that idea: it preserves algebraic structure while changing its appearance.

The mapping

$$\log : \mathbb{R}^+ \to \mathbb{R}$$

is not merely a bijection. It is an isomorphism between algebraic structures.

More precisely, the underlying structure can be stated as follows:

1. A multiplicative group, for instance $(\mathbb{R}^+, \cdot)$.
2. An additive group, for instance $(\mathbb{R}, +)$.
3. An isomorphism between them,

$$\log : (\mathbb{R}^+, \cdot) \longrightarrow (\mathbb{R}, +),$$

whose inverse is $\exp : (\mathbb{R}, +) \to (\mathbb{R}^+, \cdot)$.

On $\mathbb{R}^+$ under multiplication,

$$(\mathbb{R}^+, \cdot),$$

and on $\mathbb{R}$ under addition,

$$(\mathbb{R}, +),$$

the logarithm satisfies

$$\log(ab) = \log(a) + \log(b), \qquad \log(1) = 0.$$

Thus multiplication corresponds to addition, and multiplicative identity corresponds to additive identity.

This isomorphism is made physical in the slide rule. Distances on a logarithmic scale represent logarithmic values. Adding distances corresponds to multiplying numbers.

Lengths are not numbers, but they obey analogous structural rules. By exploiting this correspondence, the slide rule becomes an analog computer.

The construction of logarithmic scales proceeds by mapping

$$x \mapsto \log_{10}(x),$$

placing numbers at distances proportional to their logarithms. Equal ratios correspond to equal distances.

This same idea underlies historical algorithms for logarithms. Henry Briggs computed base-10 tables by accumulating small multiplicative corrections. In [9] there is a closely related method suited to binary arithmetic.

The algorithm builds a number by successive multiplications of the form

$$\frac{10^k}{10^k - 1},$$

accumulating the corresponding logarithms when the product does not overshoot the target.

What appears as computation is, in fact, structure preservation.

The logarithm transforms nonlinear growth into linear motion. It converts multiplication into addition, products into sums, and exponential scales into linear ones.

Two complementary manifestations of the invariant spirit are therefore visible. The logarithm is a global isomorphism between multiplicative and additive structures. The derivative, by contrast, provides a local linearization of arbitrary transformations. In both cases, complexity is rendered intelligible by expressing it through linear relations.

This is why logarithms appear simultaneously in analysis, geometry, numerical methods, and physical instruments. They do not belong to a single domain. They preserve structure across domains.

Bibliography

[1] Iribarren Ignacio L., *Álgebra Lineal*, Editorial Equinoccio, 2009.

[2] Halmos, Paul R., *Finite-Dimensional Vector Spaces*, Springer-Verlag, 1987.

[3] Iribarren, Ignacio L., *Topología de Espacios Métricos*, Editorial Limusa-Wiley, 1973.

[4] Cohen, Leon and Ehrlich, Gertrude, *The Structure of the Real Number System*, D. Van Nostrand Company, 1963.

[5] Wilder, Raymond L., *Introduction to The Foundations of Mathematics*, Dover Publications, Inc., 2012.

[6] Greub, Werner, *Linear Algebra*, Springer-Verlag, 1975.

[7] Rada, Juan, *Introducción al Álgebra Lineal*, Consejo de Publicaciones de la Universidad de Los Andes, 2011.

[8] Apostol, Tom M., *Calculus Volume I*, John Wiley & Sons Inc., 1967.

[9] Knuth, Donald E., *The Art of Computer Programming Volume I: Fundamental Algorithms*, Addison-Wesley, 1997.

[10] Knuth, Donald E., *The Art of Computer Programming Volume II: Seminumerical Algorithms*, Addison-Wesley, 1981.

[11] Knuth, Donald E., *The Art of Computer Programming Volume I, Fascicle 1: MMIX—A RISC Computer for the New Millennium*, Addison-Wesley, 2005.

[12] Feynman Richard P., Leighton Robert, Sands Matthew, *The Feynman Lectures on Physics, Volume II*, California Institute of Technology, 2010.

[13] Kolecki, Joseph C., *An Introduction to Tensors for Students of Physics and Engineering*, NASA/TM-2002-211716, 2002.

[14] Kolecki, Joseph C., *Foundations of Tensor Analysis for Students of Physics and Engineering With an Introduction to the Theory of Relativity*, NASA/TP—2005-213115, 2005.