# Abstract Structures I: Linear Transformations, Matrices, and Composition

Research by: Jose Portillo

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

The system above is called a *vector space* over the field $K$ [2]. 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 $\alpha_1, \dots, \alpha_k$ always denote scalars in $K$, and symbols $x_1, \dots, x_k$ 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, [3]).*
Let $\Gamma$ be a field. Consider the set
$$\Gamma^n = \Gamma \times \cdots \times \Gamma$$
of $n$-tuples $(\xi^1, \dots, \xi^n)$, with $\xi^i \in \Gamma$. Define addition by

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

and scalar multiplication by

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

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

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

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

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 $S$ [3]. That is, for every $x \in E$ there exist vectors $s_1, \dots, s_k \in S$ and scalars $\alpha_1, \dots, \alpha_k \in K$ such that
$$x = \alpha_1 \circ s_1 +_E \cdots +_E \alpha_k \circ s_k.$$

A finite family of vectors $\{x_1, \dots, x_k\} \subset E$ 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, 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 $\dim E$.

Let $E$ and $G$ be vector spaces over the same field $K$. A function $\psi : E \to G$ is called a *linear transformation* 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 coordinates.

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 $\psi(e_i) = a_{1i} \circ g_1 +_G \cdots +_G a_{mi} \circ g_m$. The scalars $a_{ji}$ form an $m \times n$ matrix

$$A = (a_{ji}),$$

called the *matrix of* $\psi$ *relative to the chosen bases.*

Matrices thus do not define linear transformations; rather, they are concrete representations of linear transformations once bases (coordinate systems) have been chosen.

In many applications, a vector does not merely represent an abstract element of a vector space, but the *state* of a system relative to a chosen reference system. Algebraically, a reference system is nothing more than a choice of basis.

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 $B$. 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.

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, not independent algebraic tricks.

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, 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* $x_1, \dots, x_n$.

*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*

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

*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*

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

*Thus, 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 $B_E = \{e_1, \dots, e_n\}$, then every $x^* \in E^*$ is uniquely determined by

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

These scalars form a row vector

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

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. If $E$ is equipped with an inner product $\langle \cdot, \cdot \rangle$, then each vector $x \in E$ determines a linear functional

$$x^*(y) = \langle x, y \rangle.$$

Thus the inner product provides a mapping

$$E \longrightarrow E^*, \quad x \longmapsto x^*.$$

So, the vector space $E$ introduced earlier is now equipped with additional geometric structure. An *inner product space* is a vector space

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

where

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

is an inner product.

If a non-orthonormal basis is used, the coordinate expression of the inner product involves additional coefficients, typically represented by a symmetric positive-definite matrix.

To understand why inner products take the form of a sum of products in coordinates, it is necessary to make explicit the geometric structure that has been imposed.

Let $E$ be a finite-dimensional vector space equipped with an inner product $\langle \cdot, \cdot \rangle$, and let

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

be a basis of $E$. The basis $B$ is called *orthonormal* if

$$\langle e_i, e_j \rangle = \delta_{ij},$$

where $\delta_{ij}$ is the *Kronecker delta*, defined by

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

This condition encodes two geometric facts: each basis vector has unit length, and distinct basis vectors are orthogonal.

If vectors $x, y \in E$ are written relative to this basis as

$$x = \sum_{i=1}^{n} x_i e_i, \qquad y = \sum_{j=1}^{n} y_j e_j,$$

then by bilinearity of the inner product,

$$\langle x, y \rangle = \sum_{i,j} x_i y_j \langle e_i, e_j \rangle.$$

The orthonormality condition $\langle e_i, e_j \rangle = \delta_{ij}$ eliminates all cross terms with $i \neq j$, leaving

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i.$$

Thus the familiar coordinate formula for the dot product is not a definition, but the coordinate expression of the inner product relative to an orthonormal basis. Only components along the same geometric direction contribute to the value of the inner product; components in orthogonal directions contribute nothing.

When vectors are written in coordinates as column vectors, the corresponding linear functional must be written as a row vector. The operation that performs this conversion in coordinates is the *transpose.*

*Example.*
Let

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in K^2.$$

With the standard inner product on $K^2$ (e.g. $K = \mathbb{R}$),

$$\langle x, y \rangle = x_1 y_1 + x_2 y_2,$$

the associated linear functional is represented by the row vector

$$x^T = [x_1 \ \ x_2].$$

Its action on a vector $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ is given by

$$x^T y = \langle x, y \rangle.$$

The transpose operation therefore exists to convert vectors into linear functionals when an inner product is present. It is not intrinsic to vector spaces themselves, but arises from additional geometric structure imposed on them.

Matrix multiplication is not an arbitrary algebraic rule; it is the coordinate expression of *composition of linear maps.*

*Note: Linear transformations are functions that are linear in a single argument. More generally, one can consider functions that are linear in several arguments, called* multilinear maps. *A multilinear map*

$$T : E_1 \times \cdots \times E_k \longrightarrow G$$

*is linear in each argument separately. In particular, functions of the form*

$$T : E \times \cdots \times E \longrightarrow K$$

*generalize linear functionals and bilinear forms. A* tensor *of type* $(k, \ell)$ *on* $E$ *is a multilinear map*

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

*Special cases include scalars* $(0, 0)$*, vectors* $(1, 0)$*, linear functionals* $(0, 1)$*, linear transformations* $(1, 1)$*, and bilinear forms* $(0, 2)$*. Once a basis of* $E$ *is chosen, a tensor can be represented by a multidimensional array of scalars; these arrays depend on the chosen basis, but the tensor itself does not. In numerical computation and machine learning, such arrays are commonly called* tensors*, although they represent the coordinate expressions of the abstract objects. NumPy and TensorFlow tensors are multidimensional arrays that represent tensors after a choice of basis; the tensor itself is a coordinate-free object. This remark is included only to situate vectors, dual vectors, and linear maps within a broader conceptual framework. No tensor calculus or multilinear theory is required or used in this document.*

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 not by convention, but 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 memo—vector spaces, linear transformations, bases, coordinate systems, dual spaces, and inner products—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. Inner products allow vectors to define linear functionals, connecting geometry with algebra. 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.

*Note.* This memo is Part I of an ongoing three-part work on abstract linear and geometric structures. An AI language model was used as a support tool for discussion, clarification of concepts, and stylistic refinement. All mathematical choices and final wording are the responsibility of the author.

## References

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

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

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