THE UNIVERSITY of TENNESSEE at CHATTANOOGA
COLLEGE of ENGINEERING and COMPUTER SCIENCE

SIMCENTER
NATIONAL CENTER for COMPUTATIONAL ENGINEERING

An Introduction to Cartesian Tensors from a Computational Perspective

A Technical Report by

W. Roger Briley

UTC-CECS-SimCenter-2012-01
September 2012

GRADUATE SCHOOL OF COMPUTATIONAL ENGINEERING
701 East M.L. King Boulevard • Chattanooga, TN 37403

AN INTRODUCTION TO CARTESIAN TENSORS FROM A COMPUTATIONAL PERSPECTIVE

W. Roger Briley

SimCenter: National Center for Computational Engineering
University of Tennessee at Chattanooga

September 2012

Preface

This report is intended to provide a self-contained introduction to Cartesian tensors for students just entering graduate school in engineering and science majors, especially those interested in computational engineering and applied computational science. This introduction assumes students have a background in multivariable calculus but no familiarity with tensors.

TABLE OF CONTENTS

© Copyright: SimCenter: National Center for Computational Engineering, 2012

AN INTRODUCTION TO CARTESIAN TENSORS FROM A COMPUTATIONAL PERSPECTIVE

1. Introduction

1.1 Tensors in Physics

Magnitude and direction are common geometric properties of physical entities. Scalars are physical quantities such as density and temperature that have magnitude (measured in a specified system of units) but no directional orientation. Vectors are physical quantities such as velocity and force with magnitude (length) and a *single* direction. The direction of vectors can be defined *only* in relation to a specified set of *N reference directions* that comprise a frame of reference for the *N*-dimensional physical space considered: typically $N = 1$, 2 or 3. The reference frame could be a set of unit vectors or a coordinate system. Vector magnitude and direction are quantified by *N* scalar components that are defined by scalar projection onto these directions. Although vector components depend on the choice of reference directions, the magnitude and direction of the vector are *invariant* physical properties that are independent of the frame of reference. Tensors are physical quantities such as stress and strain that have magnitude and *two or more* directions. For example, stress is a relationship between force and area (magnitude and two directions) and is thus a second-order tensor with N^2 components. Tensors also have invariant physical properties that are coordinate independent. True physical tensors of order higher than two are uncommon, but higher order tensors are common in mathematical descriptions of physics.

1.2 Tensors in Mathematics

Mathematically, vectors and tensors describe physical entities and their mathematical abstractions as *directional objects* represented by scalar components that are defined by projection onto a specified set of *base vectors* (typically unit vectors) comprising a basis, and that satisfy *transformation laws for a change of basis*. It is important to recognize that the term tensor is a general mathematical description for geometric objects that have magnitude and any number of directions. A tensor of order *p* has content from *p* directions and has N^p components. Thus a scalar is a zeroth-order tensor, a vector is a first-order tensor, and so on.

1.3 A Computational Perspective

The present introduction will consider vectors and tensors as encountered in computational simulations of physical *fields* in which scalar, vector and tensor quantities vary with position in space and with time. Fields require a coordinate system to locate points in space. Vector and tensor fields also require a *local basis* at each point to define vector/tensor components. Most mathematical treatments of tensors assume that this local basis is aligned with the coordinate directions: cf. [1,2] . However, the alignment of basis and coordinates introduces complexity in curvilinear orthogonal and especially in nonorthogonal coordinates that is not present in Cartesian coordinates: The local basis is constant for Cartesian coordinates but varies with spatial position using curvilinear orthogonal coordinates. Nonorthogonal coordinates introduce a dual basis: one basis is parallel to the coordinate lines (the contravariant basis) and a reciprocal basis (the covariant basis) is perpendicular to coordinate tangent planes. The resulting complications include differentiation of spatially varying base vectors, the metric tensor, dual base vectors, contravariant and covariant components, tensor differentiation, and Christoffel symbols. The calculus of tensors in general nonorthogonal coordinates is therefore significantly more complicated than that of Cartesian tensors. The complexity of variable-direction and nonorthogonal base vectors in general coordinates is commonly avoided in computational solution of both differential and integral conservation laws.

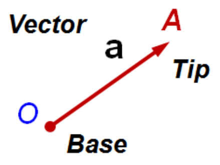
Structured Grids - In differential approaches using coordinate systems, the technique used is to transform spatial derivative terms from Cartesian to general curvilinear coordinates, while retaining a uniform Cartesian local basis for vector/tensor components. The governing equations are thereby written in general curvilinear coordinates, but the Cartesian vector/tensor components remain as dependent variables. This approach is widely discussed and has been standard practice in computational fluid dynamics for many years: cf. [3,4,5]. The use of a uniform Cartesian local basis may also reduce spatial discretization error in computations. The reason is that spatial variation in base vectors causes extraneous, nonphysical spatial variation in components that when differentiated may require higher local resolution than Cartesian components: for example in regions of large coordinate curvature.

Unstructured Grids - Another common approach for spatial discretization uses unstructured grids, for which no identifiable family of coordinate lines exists: cf. [6,7]. The governing equations are typically written in integral rather than differential form, with discrete integral approximations that require multidimensional interpolation of vectors and tensors rather than differentiation. A pointwise Cartesian local basis is used for vector/tensor components, with a local rotation of Cartesian unit vectors for alignment with local surfaces as needed in the interpolation process.

Finally, it should be emphasized that tensor mathematics is a broad area of study that can be far more complicated than what is needed and discussed here as background for computational field simulation. The present introduction covers basic material that is fundamental to the understanding and computation of physical vector and tensor fields. It is hoped that the present effort to disconflate the local vector basis and coordinate system will provide useful insight into the computation of tensor fields. Detailed discussions of vectors and tensors are given in [1,2,8] and in many other references.

2. Vector Basics

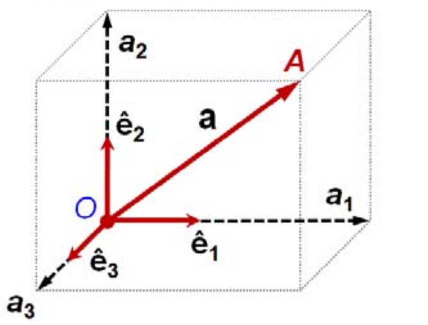
First consider a vector $\mathbf{a}$ with base *O* and tip *A*, as shown in the sketch. The vector is a directed line segment (arrow) that has inherent magnitude and direction. The vector is called a *free vector* if its location is not specified and a *fixed or bound vector* if the base *O* has a specific location in space.



The direction of $\mathbf{a}$ is quantified by *direction cosines* of the angles between $\mathbf{a}$ and a set of N arbitrary but linearly independent base vectors comprising a basis. The standard Euclidean basis is a set of right-hand *mutually orthogonal unit vectors* (called an *orthonormal basis*) located at the base *O* and denoted $\left(\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3\right)$. All examples in this introduction will assume $N = 3$.

Although the magnitude $a = \|\mathbf{a}\|$ and direction of $\mathbf{a}$ are invariants that do not depend on the choice of basis, the direction cosines are obviously basis dependent. For a given basis, a vector is represented by *N* scalar components, which are the scalar projections of the vector $\mathbf{a}$ onto the set of *N* base vectors, as shown in the nearby figure. Letting $\cos\left(\mathbf{a}, \hat{\mathbf{e}}_1\right)$ denote the cosine of the included angle between $\mathbf{a}$ and $\hat{\mathbf{e}}_1$, and with similar notation for $\hat{\mathbf{e}}_2$ and $\hat{\mathbf{e}}_3$, the components of $\mathbf{a}$ are given by

Components of a Defined by Projection onto the Orthonormal Vector Basis $\hat{\mathbf{e}}_2$



$$a_1 = \|\mathbf{a}\| \cos\left(\mathbf{a}, \hat{\mathbf{e}}_1\right); \quad a_2 = \|\mathbf{a}\| \cos\left(\mathbf{a}, \hat{\mathbf{e}}_2\right); \quad a_3 = \|\mathbf{a}\| \cos\left(\mathbf{a}, \hat{\mathbf{e}}_3\right)$$

The vector $\mathbf{a}$ is then expressed as a linear combination of the base vectors:

$$\mathbf{a} = a_1 \hat{\mathbf{e}}_1 + a_2 \hat{\mathbf{e}}_2 + a_3 \hat{\mathbf{e}}_3.$$

The vector components in a given basis are equivalent to the vector itself, since it is a simple matter to calculate the invariant magnitude and direction from known values of $\left(a_1, a_2, a_3\right)$:

$$a = \sqrt{a_1^2 + a_2^2 + a_3^2}; \quad \cos\left(\mathbf{a}, \hat{\mathbf{e}}_1\right) = a_1 / a; \quad \cos\left(\mathbf{a}, \hat{\mathbf{e}}_2\right) = a_2 / a; \quad \cos\left(\mathbf{a}, \hat{\mathbf{e}}_3\right) = a_3 / a$$

3. Index Notation for Vectors, Tensors and Matrices

Index notation is a concise way to represent vectors, matrices, and tensors. Instead of writing the components of $\mathbf{a}$ separately as $\left(a_1, a_2, a_3\right)$, the indexed variable a_i represents all components of $\mathbf{a}$ collectively as follows:

$$a_i \Leftrightarrow \left(a_1, a_2, a_3\right)$$

By convention, the index is understood to take on values in the range $i = 1, 2, 3$ or more generally $i = 1, 2, \cdots, N$. Using index notation, the complete vector $\mathbf{a}$ can be written as

$$\mathbf{a} = \sum_{i=1}^{3} a_i \hat{\mathbf{e}}_i = a_1 \hat{\mathbf{e}}_1 + a_2 \hat{\mathbf{e}}_2 + a_3 \hat{\mathbf{e}}_3$$

3.1 Einstein Summation Convention

The important summation convention states that if an index appears twice in a single term, then it is understood that the repeated index is summed over its range from 1 to *N*. The summation symbol is then redundant, and the vector can be written concisely as

$$\mathbf{a} = a_i \, \hat{\mathbf{e}}_i \;\; \Rightarrow \;\; a_1 \hat{\mathbf{e}}_1 + a_2 \hat{\mathbf{e}}_2 + a_3 \hat{\mathbf{e}}_3 \quad \textit{(with implied summation on i)}$$

Finally, once a basis $\hat{\mathbf{e}}_i$ has been identified, all vectors and tensors can be unambiguously represented by their components alone; actual display of the base vectors is unnecessary and purely a matter of notational preference. Thus the vector $\mathbf{a}$ is represented concisely as

$$\boxed{\mathbf{a} = a_i}$$

3.2 Index Rules and Terminology

The index notation can be used with any number of subscripts. For example, A_{ij} denotes the square matrix

$$A_{ij} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

In general, the *i* and *j* indices can be assigned separate ranges, for example to represent a 3×5 matrix. However, all indices are assumed to have the same $N = 3$ range in this report.

Range Convention	Variables, terms and expressions may be assigned one or more Latin index letters such as i, j, k. Each of these indices can independently take on integer values in their range $1, 2, \cdots, N$. For example, $a_i b_k$ and $a_i A_{jk}$ are terms combining vectors a_i and b_k and the matrix A_{jk}.
Index Rule	Each index letter can occur either once or twice in a single term, but no index can occur more than twice. For example, $a_j A_{jk}$ is a valid term, but $a_j A_{jj}$ is invalid.
Free Indices	An index letter that occurs only once in a single term is called a free index (or range index). A valid equation must have the same *free* indices in each term. For example, $A_{jk} = B_{jk} + C_{jk}$ is valid, but $A_{ij} = B_{ik} + C_{jk}$ is invalid. A tensor with *p* free indices has order *p* and N^p components.
Summation Indices	An index letter that occurs twice in a single term is called a summation index. The repeated index invokes a *summation* over its range. For example, $a_j b_j \Leftrightarrow \sum_{j=1}^{3} a_j b_j = a_1 b_1 + a_2 b_2 + a_3 b_3$.
Dummy Indices	A summation index is also called a dummy index because it can be replaced by a different index letter without changing its meaning. For example, in the equation $a_i A_{ij} = a_k A_{kj}$, *i* and *k* are arbitrary dummy indices, and *j* is a free index that must appear once in each term.

3.3 Special Symbols

There are two specially defined symbols that simplify index notations and operations:
The ***Kronecker delta*** δ_{ij} is defined by

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad \text{, or if expressed as an array:} \quad \delta_{ij} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The ***Levi-Civita symbol*** or ***permutation symbol*** ε_{ijk} is a three-dimensional array defined by

$$\varepsilon_{ijk} = \begin{cases} +1 & \text{if } \; i, j, k \; \text{ are an even (or cyclic) permutation of } 1, 2, 3 \\ -1 & \text{if } \; i, j, k \; \text{ are an odd (or non-cyclic) permutation of } 1, 2, 3 \\ \;\;0 & \text{if any index is repeated} \end{cases}$$

An even permutation is any three consecutive integers in the sequence $[1, 2, 3, 1, 2, 3]$. An odd permutation is any three consecutive integers in the sequence $[3, 2, 1, 3, 2, 1]$. Thus, $\varepsilon_{123} = \varepsilon_{231} = \varepsilon_{312} = +1$, and $\varepsilon_{321} = \varepsilon_{213} = \varepsilon_{132} = -1$. All other values are zero.

It is perhaps of passing interest that the following $\varepsilon - \delta$ identity can be used to generate all of the identities of vector analysis:

$$\varepsilon_{ijk} \, \varepsilon_{irs} = \delta_{jr}\delta_{ks} - \delta_{js}\delta_{rk}$$

3.4 Vector Operations in Index Notation

Scalar or Dot Product:

$$\boxed{\hat{\mathbf{e}}_i \bullet \hat{\mathbf{e}}_j = \delta_{ij}}$$

$$\boxed{\mathbf{a} \bullet \mathbf{b}} = (a_i \hat{\mathbf{e}}_i) \bullet (b_j \hat{\mathbf{e}}_j) = a_i b_j \delta_{ij} = \boxed{a_k b_k}$$

Vector or Cross Product:

$$\boxed{\hat{\mathbf{e}}_i \times \hat{\mathbf{e}}_j = \varepsilon_{ijk} \; \hat{\mathbf{e}}_k}$$

$$\boxed{\mathbf{a} \times \mathbf{b}} = (a_i \hat{\mathbf{e}}_i) \times (b_j \hat{\mathbf{e}}_j) = \boxed{a_i b_j \; \varepsilon_{ijk} \; \hat{\mathbf{e}}_k}$$

Magnitude of a Vector:

$$\| \mathbf{a} \| = (\mathbf{a} \bullet \mathbf{a})^{1/2} = \boxed{(a_k a_k)^{1/2}}$$

Determinant of a Square Matrix:

$$\det \mathbf{A} = |\mathbf{A}| = \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} = \boxed{\varepsilon_{ijk} \; A_{1i} \; A_{2j} \; A_{3k}}$$

4. Tensor Basics

We have seen that vectors are familiar geometric objects with invariant magnitude and direction. Tensors extend the description of vectors to geometric objects that have magnitude and *any* number of directions.

Order of a Tensor – The order of a tensor is equal to the number of its free indices. A tensor of order *p* has *p* free indices, involves *p* directions in an *N*-dimensional space, and has N^p components, as summarized in the following Table:

Type	**Notation**	**Order**	**Components**
Scalar	ϕ	0	N^0
Vector	a_i	1	N^1
Tensor	$A_{i\,j}$	2	N^2
Tensor	$A_{i\,j\cdots p}$	p	N^p

The indices *i* and *j* each take on the values $i = 1, 2, 3$ and $j = 1, 2, 3$ for $N = 3$ dimensions. Although any second-order tensor $A_{i\,j}$ can be interpreted as a square matrix, all square matrices are not tensors. Matrices are simple arrays of arbitrary elements; whereas, tensors incorporate geometric directional information, satisfy transformation laws for a change of basis, and have invariant properties independent of basis. Note that a common naming convention (followed in the table above) uses lower-case Greek letters to denote scalars, lower-case Latin letters for vectors, and upper-case Latin letters for matrices and tensors.

Physical Example - The stress tensor of continuum mechanics is a familiar example of a second-order tensor. Stress is a force per unit area acting on an internal surface within a material body. The state of stress at a point is uniquely determined by knowledge of the three-component stress vector acting on each of three mutually perpendicular planes. Therefore, stress is described by a second-order tensor defined by nine component stresses. The stress vector acting on any arbitrary plane passing through a point can be found by a projection of the stress tensor onto the normal direction of that plane.

Contraction of Free Indices – The summation convention is invoked by equating any two free indices (replacing them by a common index letter), and this is called a ***contraction*** of indices. Contraction of two indices reduces the order of a tensor by two.

- Contraction of *i* and *j* in $A_{i\,j}$ gives the scalar $A_{k\,k}$. The ***trace*** of a square matrix is a contraction of its indices: $\mathbf{tr}\left(A_{i\,j}\right) = A_{i\,i}$
- Contraction of *i* and *j* in $A_{i\,j\,k}$ gives the first-order tensor $A_{m\,m\,k}$ (a vector).

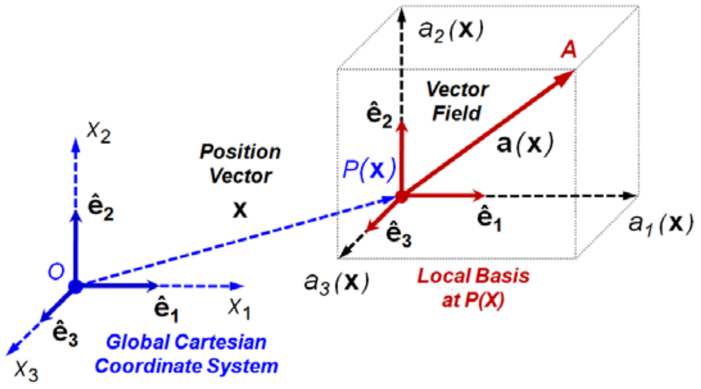
Outer and Inner Products of Tensors

- The *outer* or *tensor product* of two tensors $\mathbf{A}$ and $\mathbf{B}$ is expressed as $\mathbf{A} \otimes \mathbf{B}$. The tensor product symbol $\otimes$ is usually omitted when using index notation, so that if $\mathbf{A}$ and $\mathbf{B}$ are second-order tensors, then $\mathbf{A} \otimes \mathbf{B}$ is simply written as $A_{jk}\, B_{rs}$. Other examples of outer products are $A_i\, B_{jk}$, $A_i\, B_j$, and $A_{ijk}\, B_r$.
- The *inner* or *dot product* of two tensors is written as $\mathbf{A} \bullet \mathbf{B}$, or if $\mathbf{A}$ and $\mathbf{B}$ are second-order tensors: $\mathbf{A} \bullet \mathbf{B} = A_{jk} \bullet B_{rs}$. It is evaluated by contracting the innermost pair of free indices, made up of one index from each term in the product (in this case *k* and *r*). For example, both $A_{jk} \bullet B_{rs} = A_{jm} B_{ms} = C_{js}$ and $a_i \bullet b_{jk} = a_j b_{jk} = c_k$ are inner products. An inner product is thus an outer product plus a contraction of the two innermost indices. This dot-product contraction reduces the order of a tensor by two.

Recall that a vector $\mathbf{a}$ can be written either as $\mathbf{a} = a_i$ or $\mathbf{a} = a_i\, \hat{\mathbf{e}}_i$. A second-order tensor $\mathbf{A}$ with components A_{ij} can be written either as $\mathbf{A} = A_{ij}$ or $\mathbf{A} = A_{ij}\, \hat{\mathbf{e}}_i\, \hat{\mathbf{e}}_j$. The outer product of two vectors such as $\hat{\mathbf{e}}_i\, \hat{\mathbf{e}}_j$ or $a_i\, b_j$ is called a *dyad*, a term from vector analysis.

5. Vector and Tensor Fields

The previous discussion has only considered vectors and tensors defined at a point, without stating where the point is located. Vector and tensor *fields* are defined by assigning a vector or tensor to each point in a region of space, and their components are then functions of spatial position, as located by a global coordinate system. A vector field is illustrated in the nearby figure. Points in space are located by a spatial *position vector* $\mathbf{x}$ directed from a fixed origin *O* to each arbitrary point $P(\mathbf{x})$, and then vector components are defined using a local basis. A rectangular Cartesian coordinate system is defined by choosing an orthonormal coordinate basis $\hat{\mathbf{e}}_i = (\hat{\mathbf{e}}_1,\ \hat{\mathbf{e}}_2,\ \hat{\mathbf{e}}_3)$ located at the origin *O*. The Cartesian coordinates $x_i = (x_1,\ x_2,\ x_3)$ of point *P* are simply the scalar components of the position vector $\mathbf{x}$ defined by projection onto the Cartesian basis:



$$\mathbf{x} = x_i\, \hat{\mathbf{e}}_i = x_1 \hat{\mathbf{e}}_1 + x_2 \hat{\mathbf{e}}_2 + x_3 \hat{\mathbf{e}}_3$$

A scalar field is expressed as $\phi(\mathbf{x})$, and a vector field is denoted by $\mathbf{a}(\mathbf{x})$. A local basis is needed to define the scalar components $a_i(\mathbf{x})$ at each point in the field. The simplest choice for a local basis is shown in the preceding figure, which is to use the Cartesian coordinate basis $\hat{\mathbf{e}}_i$ at the origin *O* but to relocate it by translation without rotation from the origin *O* to the point *P*. The vector field is then given by

$$\mathbf{a}(\mathbf{x}) = a_i(\mathbf{x})\,\hat{\mathbf{e}}_i = a_1(\mathbf{x})\,\hat{\mathbf{e}}_1 + a_2(\mathbf{x})\,\hat{\mathbf{e}}_2 + a_3(\mathbf{x})\,\hat{\mathbf{e}}_3$$

A second-order tensor field $\mathbf{A}(\mathbf{x})$ with components $A_{ij}(\mathbf{x})$ is expressed as $\mathbf{A}(\mathbf{x}) = A_{ij}(\mathbf{x})\;\hat{\mathbf{e}}_i\,\hat{\mathbf{e}}_j$. Note that the $\mathbf{x}$ dependence in terms such as $a_i(\mathbf{x})$ and $A_{ij}(\mathbf{x})$ has been explicitly shown here for emphasis. As in other areas of calculus, however, this $\mathbf{x}$ dependence is often omitted for conciseness, and must be implied by context. In this example, the Cartesian local basis $\hat{\mathbf{e}}_i$ does not vary with $\mathbf{x}$.

6. Calculus Operations in Cartesian Tensor Notation

The following calculus operations are applicable to rectangular Cartesian coordinate systems with the same Cartesian local basis for vector and tensor components:

Gradient Operator: The gradient operator implies differentiation by a vector, and its components are written as

$$\boxed{\nabla(\cdot) = \frac{\partial}{\partial x_i} = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3}\right)}$$

It has been mentioned that the display of base vectors is optional, but they can be shown as

$$\boxed{\nabla(\cdot) = \hat{\mathbf{e}}_i \frac{\partial}{\partial x_i} = \hat{\mathbf{e}}_1 \frac{\partial}{\partial x_1} + \hat{\mathbf{e}}_2 \frac{\partial}{\partial x_2} + \hat{\mathbf{e}}_3 \frac{\partial}{\partial x_3}}$$

The "Comma" Derivative Notation: A comma subscript followed by an index is a concise notation indicating differentiation, as shown in the following examples:

$$\boxed{\frac{\partial(\cdot)}{\partial x_i} = (\cdot)_{,i}} \qquad A_{,j} = \frac{\partial A}{\partial x_j}\;; \qquad A_{i,j} = \frac{\partial A_i}{\partial x_j}\;; \qquad A_{ij,mn} = \frac{\partial^2 A_{ij}}{\partial x_m \partial x_n}$$

Gradient of Scalar Field: If $\phi = \phi(\mathbf{x})$ is a scalar field, then its gradient is

$\boxed{\nabla\phi = grad\;\phi = \frac{\partial\phi}{\partial x_i} = \phi_{,i}}$ or if base vectors are added: $\boxed{\nabla\phi = \hat{\mathbf{e}}_i \frac{\partial\phi}{\partial x_i} = \hat{\mathbf{e}}_i\,\phi_{,i}}$

Directional Derivative Operator: If $\mathbf{n} = n_i\,\hat{\mathbf{e}}_i$ is a unit vector, then the directional derivative for direction $\mathbf{n}$ is

$$\boxed{\mathbf{n} \bullet \nabla(\cdot) \;=\; n_i \frac{\partial(\cdot)}{\partial x_i}}$$

Divergence of a Vector Field: If $\mathbf{u} = \mathbf{u}(\mathbf{x})$ is a vector field, then its divergence is

$$\boxed{\nabla \bullet \mathbf{u} \;=\; div\; \mathbf{u} \;=\; \frac{\partial u_i}{\partial x_i} \;=\; u_{i\,,\,i}}$$

Curl of a Vector Field: If $\mathbf{u} = \mathbf{u}(\mathbf{x})$ is a vector field, then its curl has components

$$\boxed{\nabla \times \mathbf{u} \;=\; curl\; \mathbf{u} \;=\; \varepsilon_{ijk} \frac{\partial u_k}{\partial x_j} \;=\; \varepsilon_{ijk}\, u_{k\,,\,j}}$$

or if base vectors are added: $$\boxed{\nabla \times \mathbf{u} \;=\; \varepsilon_{ijk} \frac{\partial u_k}{\partial x_j}\, \hat{\mathbf{e}}_i \;=\; \varepsilon_{ijk}\, u_{k\,,\,j}\, \hat{\mathbf{e}}_i}$$

Note that the vector cross product $\mathbf{a} \times \mathbf{b} = \varepsilon_{ijk}\, a_i\, b_j\, \hat{\mathbf{e}}_k$ has a different subscript ordering.

Laplacian Operator:

$$\boxed{\nabla^2(\cdot) \;=\; div\; grad\; (\cdot) \;=\; \nabla \bullet \nabla\; (\cdot) \;=\; \frac{\partial^2(\cdot)}{\partial x_i \partial x_i} \;=\; (\cdot)_{,\,ii}}$$

If $\phi = \phi(\mathbf{x})$ is a scalar field, then its Laplacian is

$$\boxed{\nabla^2 \phi \;=\; \left(\frac{\partial}{\partial x_i}\, \hat{\mathbf{e}}_i\right) \bullet \left(\frac{\partial \phi}{\partial x_j}\, \hat{\mathbf{e}}_j\right) \;=\; \frac{\partial^2 \phi}{\partial x_i \partial x_j} \left(\hat{\mathbf{e}}_i \bullet \hat{\mathbf{e}}_j\right) \;=\; \frac{\partial^2 \phi}{\partial x_i \partial x_j}\, \delta_{ij} \;=\; \frac{\partial^2 \phi}{\partial x_i \partial x_i} \;=\; \phi_{,\,ii}}$$

If $\mathbf{u} = \mathbf{u}(\mathbf{x})$ is a vector field, then its Laplacian has components

$$\boxed{\nabla^2 \mathbf{u} \;=\; \frac{\partial^2 u_k}{\partial x_i \partial x_i} \;=\; u_{k,\,ii}}$$ or adding base vectors: $$\boxed{\nabla^2 \mathbf{u} \;=\; \frac{\partial^2 u_k}{\partial x_i \partial x_i}\, \hat{\mathbf{e}}_k \;=\; u_{k,\,ii}\, \hat{\mathbf{e}}_k}$$

Divergence of a Second-Order Tensor:

$$\boxed{\nabla \bullet \mathbf{A} \;=\; \nabla \bullet A_{ij} \;=\; \frac{\partial}{\partial x_i} A_{ij} \;=\; A_{ij\,,\,i}}$$

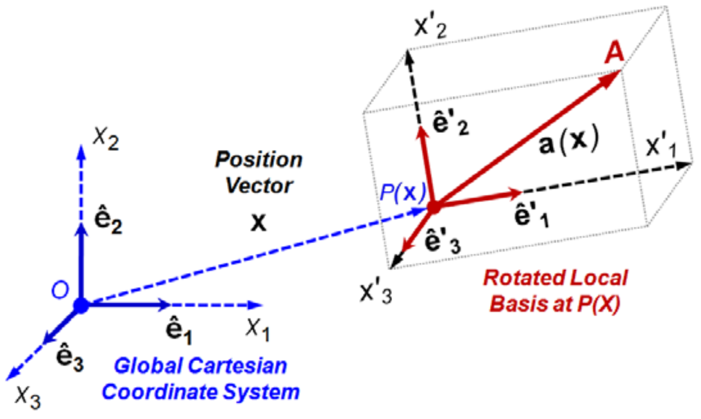
Double Dot Product: The term $\tau_{ij} : \nabla\, \mathbf{u}$ arises in fluid mechanics and is the "double dot" product of two tensors. It is evaluated as the double contraction of $\tau_{ij} \otimes \dfrac{\partial}{\partial x_m} \otimes u_n$, giving

$$\boxed{\tau_{ij} : \nabla\, \mathbf{u} \;=\; \tau_{ij} \frac{\partial}{\partial x_j} u_i \;=\; \tau_{ij}\, u_{i\,,\,j}}$$

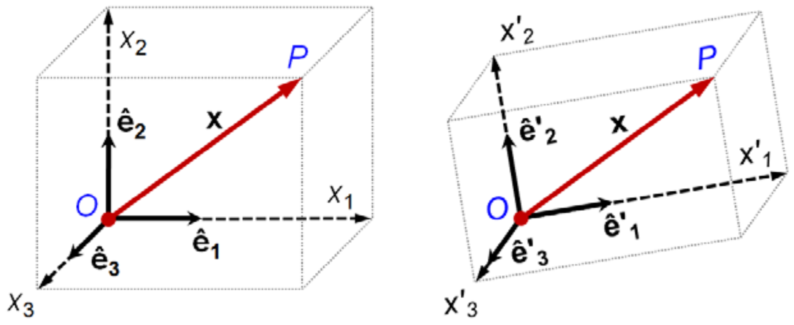
7. Transformation Laws for Cartesian Coordinates and Tensor Components

It is sometimes useful to introduce a rotation of vector basis and coordinates to a different orientation. The tensor components and coordinates then satisfy *linear transformation laws* that define coordinates and components in one basis in terms of known components in another basis. This section will give the linear transformation laws for a change from one Cartesian basis to another basis that has been rotated about the same origin.

Unstructured Grid Example - A computational example of this basis transformation arises when solving integral conservation equations using unstructured grids. Since there are no coordinate lines connecting grid points, the spatial points and solution variables are identified individually by their global Cartesian coordinates. Discrete approximations for integral field equations are then constructed for individual volume and surface elements using multidimensional interpolation among localized point groupings. Components of vector/tensor field variables are defined using the global coordinate basis, but components aligned with individual surface elements are needed in a rotated local basis, as indicated in the figure above.


x'2
A
ê'2
a(x)
x'1
x2
Position Vector
x
P(x)
ê2
ê'1
ê'3
x'3
Rotated Local Basis at P(X)
O
ê1
x1
ê3
Global Cartesian Coordinate System
x3

Cartesian Transformation Laws - Consider two rectangular Cartesian coordinate systems whose origins coincide, as shown in the nearby figure. The position vector $\mathbf{x}$ from the origin to an arbitrary point P is identical in both coordinate systems. The base vectors $\hat{\mathbf{e}}_j$ and coordinates x_j are *unprimed* in the first system, and they are denoted by *primed* variables $\hat{\mathbf{e}}'_i$ and x'_i in the second system. Thus,



$$\mathbf{x} = x_j \,\hat{\mathbf{e}}_j = x'_i \,\hat{\mathbf{e}}'_i \tag{7.1}$$

The coordinate transformation law is a linear relation that defines the x'_i coordinates in terms of x_j, which are assumed to be known. The x'_i coordinates are determined by their fundamental definition as scalar projections of the position vector $\mathbf{x}$ onto each of the unit vectors $\hat{\mathbf{e}}'_i$. The x'_i coordinates can be calculated individually using (7.1) as

$$x'_1 = \hat{\mathbf{e}}'_1 \bullet \mathbf{x} = \hat{\mathbf{e}}'_1 \bullet \left(x_j \,\hat{\mathbf{e}}_j \right) = x_j \,(\hat{\mathbf{e}}'_1 \bullet \hat{\mathbf{e}}_j) = x_j \,(1)(1)\cos(\hat{\mathbf{e}}'_1 , \hat{\mathbf{e}}_j) \tag{7.2a}$$

Similarly,

$$x'_2 = x_j \cos(\hat{\mathbf{e}}'_2, \hat{\mathbf{e}}_j) \; ; \qquad x'_3 = x_j \cos(\hat{\mathbf{e}}'_3, \hat{\mathbf{e}}_j) \tag{7.2b}$$

Each x'_i is a sum of x_j components weighted by the direction cosines of the angles between the respective x'_i and x_j axes. The individual equations in (7.2) are then written collectively as

$$x'_i = x_j \cos(\hat{\mathbf{e}}'_i, \hat{\mathbf{e}}_j) \tag{7.3}$$

For conciseness, the matrix of direction cosines is rewritten as

$$R_{ij} \Leftarrow \cos(\hat{\mathbf{e}}'_i, \hat{\mathbf{e}}_j) \tag{7.4}$$

where the first and second indices correspond to the primed and unprimed basis vectors, respectively. Substituting in (7.3) then gives the linear coordinate transformation law

$$\boxed{x'_i = R_{ij} x_j \qquad \text{Coordinate Transformation}} \tag{7.5}$$

The inverse transformation is obtained by a similar derivation in which x_j is calculated from known values of x'_i. The result is

$$\boxed{x_i = R_{ji} x'_j \qquad \text{Inverse Transformation}} \tag{7.6}$$

Note that the second index corresponding to x_j is summed in (7.5), whereas the first index corresponding to x'_j is summed in (7.6). Since the coordinates x_i and x'_i are just components of an arbitrary position vector $\mathbf{x}$, this same transformation applies to components of an arbitrary field vector $\mathbf{a}$, so that

$$\boxed{\begin{array}{ll} a'_i = R_{ij} a_j & \text{Component Transformation} \\ a_i = R_{ji} a'_j & \text{Inverse Transformation} \end{array}} \tag{7.7}$$

Analogous transformation laws for tensors of any order are given below: The forward and inverse transformation laws for second-order tensors are given by

$$\boxed{\begin{array}{ll} A'_{ij} = R_{ik} R_{jl} A_{kl} & \text{Component Transformation} \\ A_{ij} = R_{mi} R_{nj} A'_{mn} & \text{Inverse Transformation} \end{array}} \tag{7.8}$$

The forward and inverse transformation laws for higher-order tensors are given by

$$\boxed{\begin{array}{ll} A'_{ijk\ldots} = R_{ir} R_{js} R_{kt} \cdots A_{rst\ldots} & \text{Component Transformation} \\ A_{ijk\ldots} = R_{ri} R_{sj} R_{tk} \cdots A'_{rst\ldots} & \text{Inverse Transformation} \end{array}} \tag{7.9}$$

Computation of Direction Cosine Matrix - Finally, the elements of the constant matrix R_{ij} can be calculated by differentiating (7.5):

$$\frac{\partial x'_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left(R_{im}\,x_m\right) = R_{im}\frac{\partial x_m}{\partial x_j} = R_{im}\,\delta_{mj} = R_{ij} = \mathbf{R} \tag{7.10}$$

A similar differentiation of (7.6) gives the matrix transpose:

$$\frac{\partial x_i}{\partial x'_j} = \frac{\partial}{\partial x'_j}\left(R_{mi}\,x'_m\right) = R_{mi}\frac{\partial x'_m}{\partial x'_j} = R_{mi}\,\delta_{mj} = R_{ji} = \mathbf{R}^T \tag{7.11}$$

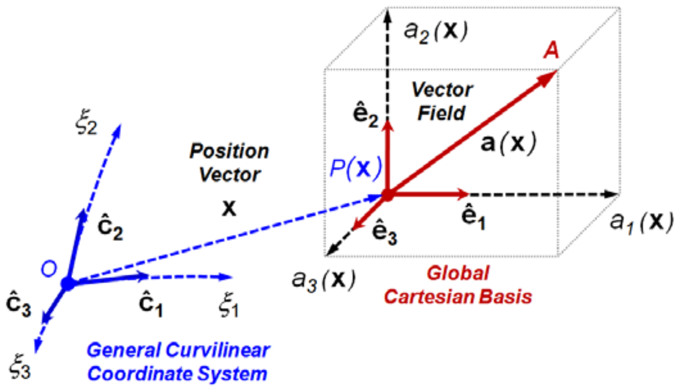
Combining (7.10) and (7.11) with the chain rule gives

$$\frac{\partial x_i}{\partial x'_j}\frac{\partial x'_j}{\partial x_k} = \delta_{ik} = \mathbf{I} = \mathbf{R}^T\mathbf{R} = \mathbf{R}^{-1}\mathbf{R} \tag{7.12}$$

Equation (7.12) shows that $\mathbf{R}^T = \mathbf{R}^{-1}$, and thus $\mathbf{R} = R_{ij}$ is an orthogonal matrix.

8. Transformation from Cartesian to General Curvilinear Coordinates

Structured Grid Example – When solving differential forms of field equations, general curvilinear coordinates and structured grids are often used to conform to geometric or boundary surfaces. Although standard treatments of non-Cartesian coordinate systems generally use local basis vectors aligned with the coordinates, this requires differentiation of spatially varying base vectors for curvilinear coordinates and introduces dual base vectors for nonorthogonal coordinates. As mentioned in the Introduction, these complexities are avoided by transforming the spatial derivative terms from Cartesian to general curvilinear coordinates, while retaining a uniform Cartesian local basis for vector/tensor components, as indicated in the adjacent figure. The governing equations are thereby expressed in general curvilinear coordinates, but the dependent variables are the Cartesian vector/tensor components. A simple chain-rule derivative transformation is all that is needed to implement this approach.



Coordinate Transformation – A given point in space $P(\mathbf{x})$ can be defined using Cartesian coordinates $x_i = \left(x_1, x_2, x_3\right)$ or general curvilinear coordinates $\xi_i = \left(\xi_1, \xi_2, \xi_3\right)$. The transformation from Cartesian to general coordinates is given by

$$\xi_1 = \xi_1\left(x_1, x_2, x_3\right)\ ; \quad \xi_2 = \xi_2\left(x_1, x_2, x_3\right)\ ; \quad \xi_3 = \xi_3\left(x_1, x_2, x_3\right)$$

and the inverse transformation is given by

$$x_1 = x_1(\xi_1, \xi_2, \xi_3) \; ; \quad x_2 = x_2(\xi_1, \xi_2, \xi_3) \; ; \quad x_3 = x_3(\xi_1, \xi_2, \xi_3)$$

In index notation, these become $\xi_j = \xi_j(x_i)$ and $x_i = x_i(\xi_j)$.

Assuming the governing differential equations are available in Cartesian coordinates, the transformed equations are obtained by replacing all Cartesian derivative terms with the chain-rule substitution

$$\boxed{\frac{\partial(\cdot)}{\partial x_i} = \left(\frac{\partial \xi_j}{\partial x_i}\right) \frac{\partial(\cdot)}{\partial \xi_j}} \tag{8.1}$$

Equation (8.1) can be implemented analytically if the grid transformation $\xi_j(x_i)$ is known (for example in cylindrical or spherical coordinates). However in computations, the structured grid is usually defined by the Cartesian coordinates of each grid point $x_i(\xi_j)$. In this case, it is a simple matter to calculate the derivatives of the inverse transformation $\partial x_i / \partial \xi_j$ numerically and make use of the identity

$$\left(\frac{\partial x_i}{\partial \xi_j}\right)\left(\frac{\partial \xi_j}{\partial x_i}\right) = \delta_{ij} = \mathbf{I} \tag{8.2}$$

The substitution (8.1) can then be written in terms of $\partial x_i / \partial \xi_j$ as

$$\boxed{\frac{\partial(\cdot)}{\partial x_i} = \left(\frac{\partial x_i}{\partial \xi_j}\right)^{-1} \frac{\partial(\cdot)}{\partial \xi_j}} \tag{8.3}$$

REFERENCES

1. Aris, Rutherford, Vectors, Tensors and the Basic Equations of Fluid Mechanics. Dover Press, ISBN-10:0486661105, 1962.
2. Fleisch, Daniel, A Student's Guide to Vectors and Tensors. Cambridge Press, ISBN-10: 0521171903, ISBN-13: 978-0521171908, 2011.
3. Vinokur, M, Conservation Equations of Gas Dynamics in Curvilinear Coordinate Systems. *J. Computational Physics*, **14**: 105-125, 1974.
4. Steger, J. L., Implicit Finite-Difference Solution of Flow about Arbitrary Two-Dimensional Geometries. *AIAA Journal*, **16**(7):679-686, 1978.
5. Taylor, L. K. and Whitfield, D. L., Unsteady Three-Dimensional Incompressible Euler and Navier-Stokes Solver for Stationary and Dynamic Grids. AIAA Paper 91-1650, 1991.
6. Anderson, W. K., and Bonhaus, D.L., An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids. *Computers and Fluids*, **23**(1):1-21, 1994.
7. Mavriplis, D. J., Unstructured Grid Techniques, Annual Review of Fluid Mechanics. **29**:473-514, 1997
8. V. N. Kaliakin, Introduction to Approximate Solution Techniques, Numerical Modeling, and Finite Element Methods. Marcel Decker, Inc., New York, ISBN: 0-8247-0679-X, 2002. *http://www.ce.udel.edu/faculty/kaliakin/appendix_tensors.pdf*