1

Coordinate Transformation and Tensors

To describe the state of the deformation for a deformable body, the coordinate transformation plays an important rule, and the most appropriate way to represent the coordinate transformation is to use tensors. In this chapter, the concept of coordinate transformations and the introduction to tensor algebra in the Cartesian coordinate system are presented along with their implementation in *Mathematica*. As this book is not meant to be a textbook on continuum mechanics, the readers are referred to some good reference books including Romano et al. (2006) and Fung (1965), among others. Manipulation involving indices requires a considerable amount of algebra work when the expressions become lengthy and complicated. It is not practical to properly handle and evaluate quantities that involve tensor manipulations by conventional scientific/engineering software such as FORTRAN, C, and MATLAB. Software packages capable of handling symbolic manipulations include *Mathematica* (Wolfram 1999), Maple (Garvan 2001), and others. In this book, *Mathematica* is exclusively used for implementation and evaluation of derived formulas. A brief introduction to the basic commands in *Mathematica* is found in the appendix, which should be appropriate to understand and execute the *Mathematica* code used in this book.

1.1 Index Notation

If one wants to properly express the deformation state of deformable bodies regardless of whether they are solids or fluids, the use of tensor equations is essential. There are several different ways to denote notations of tensors, one of which uses indices and others without using indices at all. In this book, the index notation is exclusively used throughout to avert unnecessary abstraction at the expense of mathematical sophistication.

The following are the main compelling reasons to mandate the use of tensor notations in order to describe the deformation state of bodies correctly.

Micromechanics with Mathematica, First Edition. Seiichi Nomura.

Companion Website: www.wiley.com/go/nomura0615

1. The principle of physics stipulates that a physically meaningful object must be described independent of the frame of references.[1] If the equation for a physically meaningful object changes depending on the coordinate system used, that equation is no longer a correct equation.
2. Tensor equations can be shown to be invariant under the coordinate transformation. Tensor equations are thus defined as those equations that are unchanged from one coordinate system to another.

Hence, by combining the two aforementioned statements, it can be concluded that only tensor equations can describe the physical objects properly. In other words, if an equation is not in tensorial format, the equation does not represent the object physically.

The index notation, also known as the Einstein notation (Einstein et al. 1916)[2] or the summation convention, is the most widely used notation to represent tensor quantities, which will be used in this book. The index notation in the Cartesian coordinate system is summarized as follows:

1. For mathematical symbols that are referred to quantities in the x, y, and z directions, use subscripts, 1, 2, 3, as in x_1, x_2, x_3 or a_1, a_2, a_3, instead of x, y, z or a, b, c. The subscripted numbers 1, 2, and 3, refer to the x, y, and z directions, respectively. Obviously, the upper limit of the number is 2 for 2-D and 3 for 3-D.
2. If there are twice repeated indices in a term of products such as $a_i b_i$, the summation with respect to that index (i) is always assumed. For example,

$$a_i b_i \equiv \sum_{i=1}^{3} a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3 \quad \text{(3-D)}.$$

   There is no exception to this rule. An expression such as $a_i b_i c_i$ is not allowed as the number of repetitions is 3 instead of 2.

A repeated index is called the *dummy index* as it does not matter what letter is used, and an unrepeated index is called the *free index*.[3] For example,

$$x_i x_i = x_j x_j = x_\alpha x_\alpha,$$

all of which represent a summation ($= \sum_i x_i x_i$). An unrepeated index such as x_i (or x_j or x_α) stands for one of x_1, x_2, or x_3.

It should be noted that the notations and conventions introduced are valid for the Cartesian coordinate system only. In a curvilinear coordinate system such as the spherical coordinate system, the length of base vectors is not necessarily unity, and this mandates the aforementioned index notation to be modified to reflect the difference between the contravariant components and the covariant components, which will be discussed in Chapter 2.

[1] This "frame of references" refers to the Galilean transformation in classical mechanics, the Lorentz transformation in special relativity, and the general curvilinear transformation in general relativity.

[2] Albert Einstein introduced this notation in 1916.

[3] This is similar to definite integrals. The variable used in a definite integral does not matter as

$$\int_a^b f(x)dx = \int_a^b f(y)dy = \int_a^b f(z)dz.$$

The variables x, y, and z are called *dummy variables*.

1.1.1 Some Examples of Index Notation in 3-D

1. $x_i x_i$
   As the index i is repeated, the summation symbol, $\sum$, must be added in front, i.e.,

$$\begin{aligned} x_i x_i &= \sum_{i=1}^{3} x_i x_i \\ &= x_1 x_1 + x_2 x_2 + x_3 x_3 \\ &= x^2 + y^2 + z^2. \end{aligned}$$

   Note that $x_i x_i$ is different from $(x_i)^2$. While $x_i x_i$ represents a single expression with three terms, $(x_i)^2$ represents one of the three expressions $((x_1)^2, (x_2)^2$, or $(x_3)^2)$.
2. $x_i x_i x_j$
   Note that the index i is a dummy index (repeated twice) while the index j is a free index (no repeat). Therefore,

$$\begin{aligned} x_i x_i x_j &= x_j \sum_{i=1}^{3} x_i x_i \\ &= x_j((x_1)^2 + (x_2)^2 + (x_3)^2) \\ &= \begin{cases} x(x^2 + y^2 + z^2) \\ \quad \text{or} \\ y(x^2 + y^2 + z^2) \\ \quad \text{or} \\ z(x^2 + y^2 + z^2). \end{cases} \end{aligned}$$

3. $x_i x_i x_i$
   This is not a valid tensor expression as the number of repeated indices must be 2.

1.1.2 Mathematica Implementation

As *Mathematica* itself does not support tensor manipulation natively, it is necessary to devise a way to handle index notation and tensor manipulation. In this book, a list or a list of lists (a nested list) is used to represent tensor quantities. Using a nested list to define a tensor of any rank is straightforward but at the same time limited to the Cartesian tensors. For tensors defined in a curvilinear coordinate system, a slightly different approach is needed.

When running *Mathematica* first time, a default directory should be selected so that all the notebook files can be saved and accessed in this directory. By default, *Mathematica* looks for all the files stored in `c:\users\<user>\` where `<user>` is the user's home directory.[4] The `SetDirectory` command can change this location. For example, if you want to change the default directory to `c:\tmp`, the `SetDirectory` command can specify the default directory as

[4] The Windows operating system uses "\" (backslash) as the directory delimiter while the Unix system uses "/" (forward slash) as the directory delimiter. However, the "/" in the SetDirectory command in *Mathematica* works for both.

```
In[1]:= SetDirectory["c:/tmp"]
Out[1]= c:\tmp
```

It is noted that the directory delimiter needs to be entered as "/" (forward slash) even though the Windows delimiter character is "\" (backslash).

To enter a three-dimensional vector, $\mathbf{v} = (x^2, y^2, z^2)$, the following *Mathematica* command can be entered to create a list with braces (curly brackets) as

```
In[2]:= v = {x^2, y^2, z^2}
Out[2]= {x^2, y^2, z^2}
```

An individual component of **v** can be referenced using double square brackets (`[[...]]`) as

```
In[3]:= v[[1]]
Out[3]= x^2
```

The partial derivative of **v** with respect to x can be entered as

```
In[4]:= D[v, x]
Out[4]= {2 x, 0, 0}
```

You can also differentiate an individual component as

```
In[5]:= D[v[[2]], y]
Out[5]= 2 y
```

Implementation of the coordinate component, x_i, into *Mathematica* can be done by using the `Table` function. To define a position vector, **r**, whose components are (x_1, x_2, x_3), enter

```
In[1]:= r = Table[x[i], {i, 1, 3}]
Out[1]= {x[1], x[2], x[3]}
```

The given `Table` function generates a list of elements. For example, the following command generates a sequence of i^2 for $i = 1, \ldots, 10$.

```
In[2]:= Table[ i^2, {i, 1, 10}]
Out[2]= {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

In the definition of the position vector, **r**, it is noted that the coordinate components, (x_1, x_2, x_3), are entered as `x[1], x[2], x[3]` instead of `x[[1]], x[[2]], x[[3]]`. It is important to distinguish a single square bracket (`[...]`) and a double square bracket (`[[...]]`). The single square bracket (`[...]`) is for a parameter used in a function. The quantity, `x[1]`, stands for a function, `x`, with the argument of `1`. By using a single square bracket, the quantities such as `x[1]`, `x[2]` can stand for themselves, meaning that initial values do not have to be preassigned. On the other hand, if `x[[1]]` were used instead of `x[1]`, `0` would be returned unless a list `x` is previously defined.

To define a function in *Mathematica*, use the following syntax:

```
In[1]:= f[x_] := x^2 + 1

In[2]:= f[6]

Out[2]= 37

In[3]:= f[a^2]

Out[3]= 1 + a^4
```

In the aforementioned example code, a user-defined function, `f[x]`, that returns $x^2 + 1$ is defined. The syntax is such that variables of the function must be presented to the left of the equal sign with the underscore and the definition of the function is given to the right of a colon and an equal sign (`:=`). It is important to note that a function in *Mathematica* returns itself if no prior definition is given.

```
In[1]:= f[x_, y_] := x^2 + y^2

In[2]:= f[a^3, b^2]

Out[2]= a^6 + b^4

In[3]:= f[1]

Out[3]= f[1]

In[4]:= g[a]

Out[4]= g[a]
```

In the aforementioned example, `f[x, y]` is defined as a function that takes two variables returning $x^2 + y^2$. However, when `f` is called with only one variable, `1`, it returns itself, i.e., `f[1]`, as `f` with only one variable has not been defined. When `g[a]` is entered, it returns itself without evaluation as there is no prior definition of `g[x]` given. It is this property of a function in *Mathematica* that enables manipulating index notations.

As an example of using `x[i]` as the coordinate components, here is how to implement the summation convention. As *Mathematica* does not have support for the summation convention built in, if `x[i]x[i]` meant as $x_i x_i = x_1^2 + x_2^2 + x_3^2$ is entered as

```
In[1]:= x[i] x[i]

Out[1]= x[i]^2
```

Mathematica does not automatically expand $x_i x_i$ and reduce the result to r^2. Hence, it is necessary to explicitly use the `Sum[]` command as

```
In[6]:= Sum[x[i] x[i], {i, 1, 3}]

Out[6]= x[1]^2 + x[2]^2 + x[3]^2
```

for $x_i x_i$.

However, it is possible to implement the summation convention in *Mathematica* with the following procedure.

```
In[1]:= Unprotect[Times];
        x[i_Symbol] x[i_Symbol] := r^2;
        Protect[Times];
```

The aforementioned three-line code adds a new rule to automatically replace `x[i]x[i]` by r^2 through pattern matching. When *Mathematica* evaluates the product of two quantities, it calls its internal function, `Times`, which tries to simplify the result with various pattern-matching algorithms. It is not possible to modify the pattern-matching algorithms as they are part of the definition of the `Times` function and they are protected by default. Therefore, it is necessary to unprotect the multiplication operator, `Times`, in *Mathematica* with the `Unprotect` command so that a new rule for the summation convention can be added. The second line is to tell *Mathematica* a new rule that whenever a pattern of `x[i]x[i]` where `i` is any symbol but not a number is entered, it is automatically replaced by r^2. Finally, in the third line, the `Times` function is given back the `Protect` attribute so that any further modification is prevented. After these three lines are entered, the summation convention for `x[i] x[i]` is automatic as

```
In[4]:= x[j] x[j]
Out[4]= r^2
In[5]:= x[3] x[3]
Out[5]= x[3]^2
In[6]:= x[j] x[j] x[i]
Out[6]= r^2 x[i]
In[7]:= x[i] x[i] x[j] x[j]
Out[7]= r^4
```

In the aforementioned examples, $x_j x_j$ is reduced to r^2 but $x_3 x_3$ remains unchanged. The expression, $x_j x_j x_i$ is simplified to $r^2 x_i$ and $x_i x_i x_j x_j$ is reduced to r^4. With this pattern-matching capability of *Mathematica*, manipulation of tensor quantities can be greatly simplified. More detailed examples will be shown later.

1.1.3 Kronecker Delta

One of the most important symbols in tensor algebra is the Kronecker delta, which is defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } (i,j) = (1,1), (2,2), \text{or } (3,3) \\ 0, & \text{otherwise.} \end{cases}$$

The Kronecker delta is what is equivalent to 1 in numbers or the identity matrix in linear algebra. It is also used to define the inverse of a tensor.

Examples

1. δ_{ii} (3-D)

$$\begin{aligned}\delta_{ii} &= \sum_{i=1}^{3} \delta_{ii} \\ &= \delta_{11} + \delta_{22} + \delta_{33} \\ &= 3.\end{aligned}$$

2. $\delta_{ij}\delta_{ij}$ (3-D)

$$\begin{aligned}\delta_{ij}\delta_{ij} &= \sum_{i=1}^{3}\sum_{j=1}^{3} \delta_{ij}\delta_{ij} \\ &= \delta_{11}\delta_{11} + \delta_{12}\delta_{12} + \delta_{13}\delta_{13} \\ &+ \delta_{21}\delta_{21} + \delta_{22}\delta_{22} + \delta_{23}\delta_{23} \\ &+ \delta_{31}\delta_{31} + \delta_{32}\delta_{32} + \delta_{33}\delta_{33} \\ &= 3.\end{aligned}$$

3. $\delta_{ii}\delta_{jj}$ (3-D)

$$\begin{aligned}\delta_{ii}\delta_{jj} &= \sum_{i=1}^{3} \delta_{ii} \sum_{j=1}^{3} \delta_{jj} \\ &= (\delta_{11} + \delta_{22} + \delta_{33})(\delta_{11} + \delta_{22} + \delta_{33}) \\ &= 9.\end{aligned}$$

4. $\delta_{ij}x_i x_j$ (3-D)

$$\begin{aligned}\delta_{ij}x_i x_j &= \sum_{i=1}^{3}\sum_{j=1}^{3} \delta_{ij}x_i x_j \\ &= \delta_{11}x_1x_1 + \delta_{12}x_1x_2 + \delta_{13}x_1x_3 \\ &+ \delta_{21}x_2x_1 + \delta_{22}x_2x_2 + \delta_{23}x_2x_3 \\ &+ \delta_{31}x_3x_1 + \delta_{32}x_3x_2 + \delta_{33}x_3x_3 \\ &= (x_1)^2 + (x_2)^2 + (x_3)^2.\end{aligned}$$

The Kronecker delta can be implemented in *Mathematica* by several different ways. The following implementation is to use a function to define the Kronecker delta, which is valid for any number of dimensions.

```
In[1]:= delta[i_Integer, j_Integer] := If[i == j, 1, 0]

In[2]:= delta[1, 2]

Out[2]= 0

In[3]:= delta[i, j]

Out[3]= delta[i, j]

In[4]:= Sum[delta[i, i], {i, 3}]

Out[4]= 3
```

The underline "_" after the variable name is used by *Mathematica* to restrict the type of variable to be the type that follows _. In this case, `delta[i, j]` is evaluated only when both `i` and `j` are integers. The function `If[condition, t, f]` gives `t` if `condition` is true and `f` otherwise. Note that the two equal signs (==) in "`i==j`" mean equality, not an assignment as in most of the programming languages.

1.1.3.1 Summation Convention for δ_{ij}

Some of the properties of the Kronecker delta that involve the summation convention include:

$$\delta_{ii} = 3, \;\; a_j\delta_{ij} = a_i, \;\; a_{ij}\delta_{ik} = a_{kj}, \;\; \delta_{ij}\delta_{ik} = \delta_{jk}.$$

The following *Mathematica* code implements the aforementioned rules so that summation convention that involves the Kronecker delta is always automatically executed.

```
In[10]:= SetAttribute[delta, Orderless];
         delta[i_Integer, j_Integer] := If[i == j, 1, 0];
         delta[i_Symbol, i_Symbol] := 3;

In[14]:= Unprotect[Times];
         Times[a_Symbol[j_Symbol], delta[i_, j_Symbol]] := a[i];
         Times[a_Symbol[i_Symbol, j_], delta[i_Symbol, k_]] := a[k, j];
         Times[delta[i_Symbol, j_], delta[i_Symbol, k_]] := delta[j, k];
         Protect[Times];
```

The `SetAttribute[delta, Orderless]` command sorts the variables in `delta` into standard order so that `delta[j,i]` is automatically changed to `delta[i,j]`. The part `i_Integer` specifies that the variable `i` must be an integer value, and the part `i_Symbol` specifies that the variable `i` must be a symbol, not a number. The subsequent code instructs *Mathematica* to add new rules that use the summation convention with `delta[i,j]`. After this *Mathematica* code is entered, the summation convention involving the Kronecker delta is automatically enforced.

```
In[9]:= delta[i, i]

Out[9]= 3

In[10]:= a[j] delta[i, j]

Out[10]= a[i]

In[11]:= a[i, j] delta[i, k]

Out[11]= a[k, j]

In[12]:= delta[i, j] delta[i, k]

Out[12]= delta[j, k]
```

The aforementioned automatic conversion makes manipulation of tensor products easier as will be seen in later chapters.

1.1.4 Permutation Symbols

The permutation symbol,[5] ϵ_{ijk}, is defined as

$$\epsilon_{ijk} = \begin{cases} 1, & (ijk) = (123), (231), (312), \\ -1, & (ijk) = (213), (321), (132), \\ 0, & \text{otherwise.} \end{cases}$$

The permutation symbol, ϵ_{ijk}, takes a value of 1 if (ijk) is an even permutation[6] of (123) and a value of 0 if (ijk) is an odd permutation of (123). Note that ϵ_{ijk} is 0 if any two indices are the same.

The permutation symbol is used for the vector product. The ith component of vector product, $\mathbf{a} \times \mathbf{b}$, is expressed as

$$(a \times b)_i = \epsilon_{ijk} a_j b_k.$$

In *Mathematica*, the permutation symbol is built-in and can be invoked by the `Signature` function as

```
In[1]:= Signature[{1, 2, 3}]

Out[1]= 1

In[2]:= Signature[{1, 1, 3}]

Out[2]= 0
```

[5] The permutation symbol is defined in 3-D only.

[6] If a sequence, (ijk), is obtained from (123) with an even number of permutations of two numbers, it is called an even permutation. Otherwise, it is called an odd permutation.

The vector cross product, $\mathbf{a} \times \mathbf{b}$, can be implemented as

```
In[3]:= Table[ Sum[ Signature[{i, j, k}] a[i] b[j], {i, 3}, {j, 3}], {k, 3}]
Out[3]= {-a[3] b[2] + a[2] b[3], a[3] b[1] - a[1] b[3], -a[2] b[1] + a[1] b[2]}
```

The rotation operation, rot $\mathbf{v}$, can be implemented as

```
In[4]:= Table[ Sum[ Signature[{i, j, k}] Dt[v, x[k]], {j, 3}, {k, 3}], {i, 3}]
Out[4]= {-Dt[v, x[2]] + Dt[v, x[3]], Dt[v, x[1]] - Dt[v, x[3]], -Dt[v, x[1]] + Dt[v, x[2]]}
```

where `Dt[f,x]` is the total derivative, df/dx.

1.1.5 Product of Matrices

As another example of summation convention, the product of $n \times n$ matrices, A and B, is expressed as

$$AB = C,$$

or

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}.$$

The ij component of C is expressed as

$$\begin{aligned} c_{ij} &= a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j} + \cdots + a_{in}b_{nj} \\ &= \sum_{k=1}^{n} a_{ik}b_{kj} \\ &= a_{ik}b_{kj}. \end{aligned} \tag{1.1}$$

Equation (1.1) indicates that the (ij) component of C can be computed by the (ik) component of A and the (kj) component of B.

In *Mathematica*, a matrix with the components in symbolic form can be entered as a nested array as

```
In[2]:= mat = Table[ a[i, j], {i, 1, 3}, {j, 1, 3}]
Out[2]= {{a[1, 1], a[1, 2], a[1, 3]},
         {a[2, 1], a[2, 2], a[2, 3]}, {a[3, 1], a[3, 2], a[3, 3]}}

In[5]:= MatrixForm[mat]
Out[5]//MatrixForm=
        ( a[1, 1]  a[1, 2]  a[1, 3] )
        ( a[2, 1]  a[2, 2]  a[2, 3] )
        ( a[3, 1]  a[3, 2]  a[3, 3] )
```

The multiplication between two matrices can be carried out by a dot (.).

```
In[6]:= amat = Table[ a[i, j], {i, 3}, {j, 3}]

Out[6]= {{a[1, 1], a[1, 2], a[1, 3]},
         {a[2, 1], a[2, 2], a[2, 3]}, {a[3, 1], a[3, 2], a[3, 3]}}

In[7]:= bmat = Table[b[i, j], {i, 3}, {j, 3}]

Out[7]= {{b[1, 1], b[1, 2], b[1, 3]},
         {b[2, 1], b[2, 2], b[2, 3]}, {b[3, 1], b[3, 2], b[3, 3]}}

In[8]:= cmat = amat.bmat

Out[8]= {{a[1, 1] b[1, 1] + a[1, 2] b[2, 1] + a[1, 3] b[3, 1],
          a[1, 1] b[1, 2] + a[1, 2] b[2, 2] + a[1, 3] b[3, 2],
          a[1, 1] b[1, 3] + a[1, 2] b[2, 3] + a[1, 3] b[3, 3]},
         {a[2, 1] b[1, 1] + a[2, 2] b[2, 1] + a[2, 3] b[3, 1],
          a[2, 1] b[1, 2] + a[2, 2] b[2, 2] + a[2, 3] b[3, 2],
          a[2, 1] b[1, 3] + a[2, 2] b[2, 3] + a[2, 3] b[3, 3]},
         {a[3, 1] b[1, 1] + a[3, 2] b[2, 1] + a[3, 3] b[3, 1],
          a[3, 1] b[1, 2] + a[3, 2] b[2, 2] + a[3, 3] b[3, 2],
          a[3, 1] b[1, 3] + a[3, 2] b[2, 3] + a[3, 3] b[3, 3]}}
```

Note that ordinary multiplication in *Mathematica* is either an asterisk ($*$) or a space.

1.2 Coordinate Transformations (Cartesian Tensors)

In this section, Cartesian tensors that are subject to transformational rules between two Cartesian coordinate systems are introduced. This automatically excludes such curvilinear coordinate systems as the polar coordinate system. Tensors that are used in general curvilinear coordinate systems will be discussed in Chapter 2.

Consider a rotation of a coordinate system from a Cartesian coordinate system, x_i, or (x, y), to another Cartesian coordinate system, $x_{\bar{i}}$, or $(\bar{x}, \bar{y})$ as shown in Figure 1.1.

Note that, although the coordinate systems are changed, the position vector, $\mathbf{r}$, itself remains unchanged. Therefore, it follows

$$\begin{aligned}\mathbf{r} &= x_1\mathbf{e}_1 + x_2\mathbf{e}_2 \\ &= x_{\bar{1}}\mathbf{e}_{\bar{1}} + x_{\bar{2}}\mathbf{e}_{\bar{2}},\end{aligned}$$

or

$$x_{\bar{i}}\mathbf{e}_{\bar{i}} = x_i\mathbf{e}_i, \tag{1.2}$$

where $(\mathbf{e}_1, \mathbf{e}_2)$ and $(\mathbf{e}_{\bar{1}}, \mathbf{e}_{\bar{2}})$ are the base vectors in the original coordinate system and the rotated coordinate system, respectively. It is noted that the base vectors are normalized and mutually orthogonal as

$$(\mathbf{e}_i, \mathbf{e}_j) = \delta_{ij},$$

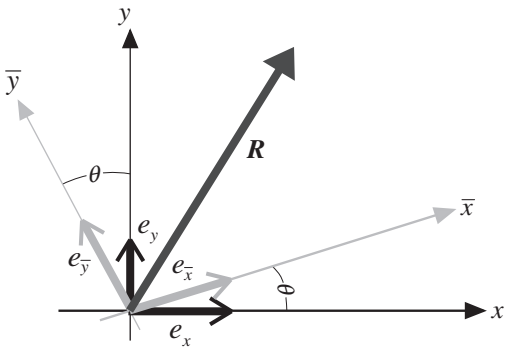

Figure 1.1 Two-dimensional coordinate transformation

where (**a**, **b**) is the inner product between two vectors, **a** and **b**. By multiplying $\mathbf{e}_{\bar{j}}$ on both sides of Equation (1.2) in the inner product, one obtains

$$x_{\bar{i}}(\mathbf{e}_{\bar{i}}, \mathbf{e}_{\bar{j}}) = x_i(\mathbf{e}_i, \mathbf{e}_{\bar{j}}),^{7}$$

or

$$x_{\bar{i}}\delta_{\bar{i}\bar{j}} = x_i(\mathbf{e}_i, \mathbf{e}_{\bar{j}}),$$

or

$$x_{\bar{j}} = \beta_{\bar{j}i}x_i, \tag{1.3}$$

where

$$\beta_{\bar{j}i} \equiv (\mathbf{e}_{\bar{j}}, \mathbf{e}_i).$$

For 2-D, Equation (1.3) is written in conventional form as

$$\begin{pmatrix} x_{\bar{1}} \\ x_{\bar{2}} \end{pmatrix} = \begin{pmatrix} \cos\ \theta, & \sin\ \theta \\ -\sin\ \theta, & \cos\ \theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.^{8} \tag{1.4}$$

The matrix, $\beta_{\bar{j}i}$, is called the transformation matrix and is a unitary matrix based on which the definition of the Cartesian tensors is made.

The inverse of the transformation matrix, $\beta_{\bar{j}i}$, can be obtained by replacing θ in Equation (1.4) by $-\theta$ as

$$\begin{aligned} B^{-1} &= \begin{pmatrix} \cos\theta, & \sin\theta \\ -\sin\theta, & \cos\theta \end{pmatrix}^{-1} \\ &= \begin{pmatrix} \cos(-\theta), & \sin(-\theta) \\ -\sin(-\theta), & \cos(-\theta) \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta, & -\sin\theta \\ \sin\theta, & \cos\theta \end{pmatrix}, \end{aligned}$$

[7] When the Kronecker delta, δ_{ij}, is multiplied by another tensor one of whose indices is contracted (summed) with one of the indices of δ_{ij}, it follows

$$\delta_{ij}a_{jklmn\ldots} = a_{iklmn\ldots}$$

that is, the Kronecker delta is absorbed.

[8] The matrix can be easily memorized if you realize that the second row is the differentiation of the first row.

or

$$B^{-1} = B^T,$$

where B^T is the transpose of B.

Here is how to enter the 2-D transformation matrix in *Mathematica*.

```
In[1]:= β = {{Cos[θ], Sin[θ]}, {-Sin[θ], Cos[θ]}}

Out[1]= {{Cos[θ], Sin[θ]}, {-Sin[θ], Cos[θ]}}

In[2]:= xnew = β.Table[x[i], {i, 1, 2}]

Out[2]= {Cos[θ] x[1] + Sin[θ] x[2], -Sin[θ] x[1] + Cos[θ] x[2]}
```

We can enter the Greek letter, β, from a keyboard by hitting `Esc`, `b`, and `Esc` keys. The transformation matrix, B, is entered as a 2×2 list. Note that, for the product between lists, a dot $(\cdot)$ must be used instead of a space or an asterisk.

1.3 Definition of Tensors

Tensors are geometrical quantities whose values are subject to the transformational rules. Tensor equations are invariant under any coordinate transformation, which is the reason why any physically significant equation must be written as a tensor equation.

In this book, only Cartesian tensors are used throughout. The Cartesian tensors are defined through the transformation matrix, $\beta_{ij}(\theta)$, of Equation (1.4) between two Cartesian coordinate systems. Zeroth- and first-rank Cartesian tensors have aliases, i.e., scalars and vectors, respectively, but there are no specific names assigned for tensors of rank 2 and higher.

Here are the definitions of Cartesian tensors of different ranks.

1.3.1 Tensor of Rank 0 (Scalar)

A single quantity, Φ, is called a zeroth-rank tensor (also called a tensor of rank 0 or a *scalar*) if Φ satisfies the following:

(a) The quantity, $\Phi(x_i)$, has a single component (obviously!).
(b) If $\Phi(x_{\bar{i}})$ denotes a quantity where x_i in $\Phi(x_i)$ is replaced by $x_{\bar{i}}$, the following is held:

$$\Phi(x_{\bar{i}}) = \Phi(x_i).$$

Examples

1. $\Phi(x_i) \equiv 1$.

   This is a scalar as Φ remains unchanged no matter what the coordinate system is.

2. $\Phi(x, y) \equiv x^2 + y^2$.

$$\begin{aligned}\Phi(\bar{x}, \bar{y}) &= \bar{x}^2 + \bar{y}^2 \\ &= (x\cos\ \theta + y\sin\ \theta)^2 + (-x\sin\ \theta + y\cos\ \theta)^2 \\ &= x^2 + y^2 \\ &= \Phi(x, y).\end{aligned}$$

Hence, $\Phi(x, y)$ is a scalar.

3. $\Phi(x, y) \equiv x^2 - y^2$.

$$\begin{aligned}\Phi(\bar{x}, \bar{y}) &= \bar{x}^2 - \bar{y}^2 \\ &= (x\cos\ \theta + y\sin\ \theta)^2 - (-x\sin\ \theta + y\cos\ \theta)^2 \\ &\neq \Phi(x, y).\end{aligned}$$

Hence, $\Phi(x, y)$ is not a scalar.

Here is a *Mathematica* code to show that $x_i x_i$ is a scalar.

```
In[1]:= β = {{Cos[θ], Sin[θ]}, {-Sin[θ], Cos[θ]}}
Out[1]= {{Cos[θ], Sin[θ]}, {-Sin[θ], Cos[θ]}}
In[2]:= xnew = β.Table[xold[i], {i, 2}]
Out[2]= {Cos[θ] xold[1] + Sin[θ] xold[2], -Sin[θ] xold[1] + Cos[θ] xold[2]}
In[3]:= xnew[[1]]^2 + xnew[[2]]^2
Out[3]= (-Sin[θ] xold[1] + Cos[θ] xold[2])^2 + (Cos[θ] xold[1] + Sin[θ] xold[2])^2
In[4]:= Simplify[%]
Out[4]= xold[1]^2 + xold[2]^2
```

In the aforementioned example, β is the 2-D transformation matrix and `xold` represents (x, y) in the original coordinate system. The variables in `xnew` are the components in the rotated coordinate system, $(\bar{x}, \bar{y})$, which can be obtained by $\beta_{ij} x_j$. The quantity $x_{\bar{i}} x_{\bar{i}}$ is computed and is simplified by using the `Simplify` function to be reduced to $x_i x_i$.

1.3.2 Tensor of Rank 1 (Vector)

A tensor of rank 1 is defined as a quantity, Φ_j, a function of x_i, satisfying the following properties:

(a) Having two components in 2-D (3 in 3-D).
(b) When the coordinate system is changed from x_i to $x_{\bar{i}}$,

$$\Phi_{\bar{i}}(x_{\bar{1}}, x_{\bar{2}}) = \beta_{\bar{i}j} \Phi_j(x_1, x_2).$$

Examples

1. $\Phi_1 = x$, $\Phi_2 = y$.
   This is obviously a tensor (vector) as the components of Φ_i are identical to those of the coordinate components.
2. $\Phi_1 = y$, $\Phi_2 = x$.
   To check to see if this quantity is a tensor, one has to test the transformation rule for the first-rank tensor,

$$\begin{aligned}\Phi_{\bar{1}} &= \bar{y} \\ &= -x\sin\ \theta + y\cos\ \theta,\end{aligned}$$

$$\begin{aligned}\beta_{\bar{1}j}\Phi_j &= \beta_{\bar{1}1}\Phi_1 + \beta_{\bar{1}2}\Phi_2\\ &= y\cos\ \theta + x\sin\ \theta,\end{aligned}$$

Therefore,

$$\Phi_{\bar{1}} \neq \beta_{\bar{1}j}\Phi_j.$$

Hence, Φ_i is not a vector.

3. $\Phi_1 = -y$, $\Phi_2 = x$.

$$\begin{aligned}\Phi_{\bar{1}} &= -\bar{y}\\ &= x\sin\ \theta - y\cos\ \theta,\\ \beta_{\bar{1}j}\Phi_j &= \beta_{\bar{1}1}\Phi_1 + \beta_{\bar{1}2}\Phi_2\\ &= -y\cos\ \theta + x\sin\ \theta.\end{aligned}$$

Therefore,

$$\Phi_{\bar{1}} = \beta_{1j}\Phi_j.$$

Also,

$$\begin{aligned}\Phi_{\bar{2}} &= \bar{x}\\ &= x\cos\ \theta + y\sin\ \theta,\\ \beta_{\bar{2}j}\Phi_j &= \beta_{\bar{2}1}\Phi_1 + \beta_{\bar{2}2}\Phi_2\\ &= y\sin\ \theta + x\cos\ \theta.\end{aligned}$$

Therefore,

$$\Phi_{\bar{2}} = \beta_{\bar{2}j}\Phi_j.$$

Hence, Φ_i is a vector.

Here is an example of a code to transform ϕ_i with $\phi_1 = y$, $\phi_2 = -x$.

```
In[5]:= β = {{Cos[θ], Sin[θ]}, {-Sin[θ], Cos[θ]}};
        ϕ = {-y, x};
        Table[Sum[β[[i, j]] ϕ[[j]], {j, 2}], {i, 2}]
Out[7]= {-y Cos[θ] + x Sin[θ], x Cos[θ] + y Sin[θ]}
```

1.3.3 Tensor of Rank 2

A tensor of rank 2 (a second-rank tensor) is defined as a quantity, Φ_{ij}, satisfying the following properties:

(a) Having 4 components (9 in 3-D).
(b) When the coordinate system is changed from x_i to $x_{\bar{i}}$, the following relation holds:

$$\Phi_{\bar{i}\bar{j}}(x_{\bar{k}}) = \beta_{\bar{i}i}\beta_{\bar{j}j}\Phi_{ij}(x_k).$$

Examples

1.
$$\Phi_{ij} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$
$$\Phi_{\bar{i}\bar{j}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

To prove that this is a tensor, the tensorial transformation rule needs to be examined for every single index. For the indices (11),

$$\begin{aligned} \Phi_{\bar{1}\bar{1}} &= 1, \\ \beta_{\bar{1}i}\beta_{\bar{1}j}\Phi_{ij} &= \beta_{\bar{1}1}\beta_{\bar{1}1}\Phi_{11} + \beta_{\bar{1}1}\beta_{\bar{1}2}\Phi_{12} + \beta_{\bar{1}2}\beta_{\bar{1}1}\Phi_{21} + \beta_{\bar{1}2}\beta_{\bar{1}2}\Phi_{22} \\ &= \beta_{\bar{1}1}\beta_{\bar{1}1} + \beta_{\bar{1}2}\beta_{\bar{1}2} \\ &= \cos^2\theta + \sin^2\theta \\ &= 1. \end{aligned}$$

Hence
$$\Phi_{\bar{1}\bar{1}} = \beta_{\bar{1}i}\beta_{\bar{1}j}\Phi_{ij}.$$

This can be repeated for the rest of the indices; thus, Φ_{ij} is a second-rank tensor.

2. A constant matrix:
$$\Phi_{ij} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix},$$
$$\Phi_{\bar{i}\bar{j}} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

This is not a second-rank tensor. For example, for the indices (1, 1),

$$\begin{aligned} \Phi_{\bar{1}\bar{1}} &= 1, \\ \beta_{\bar{1}i}\beta_{\bar{1}j}\Phi_{ij} &= \beta_{\bar{1}1}\beta_{\bar{1}1}\Phi_{11} + \beta_{\bar{1}1}\beta_{\bar{1}2}\Phi_{12} + \beta_{\bar{1}2}\beta_{\bar{1}1}\Phi_{21} + \beta_{\bar{1}2}\beta_{\bar{1}2}\Phi_{22} \\ &= \cos^2\theta + 2\cos\theta\sin\theta + \sin^2\theta. \end{aligned}$$

Hence
$$\Phi_{\bar{1}\bar{1}} \neq \beta_{\bar{1}i}\beta_{\bar{1}j}\Phi_{ij}.$$

It is not necessary to check the rest of the indices.

3.
$$\Phi_{ij} = \begin{pmatrix} x^2 & xy \\ xy & y^2 \end{pmatrix}.$$

It is possible to prove that this is a second-rank tensor (exercise).[9]

[9] This is known as the moment.

1.3.4 Tensor of Rank 3

A tensor of rank 3 (third-rank tensor) is defined as a quantity, Φ_{ijk}, satisfying the following:

(a) Having eight components (27 in 3-D).
(b) When the coordinate system is changed from x_i to $x_{\bar{i}}$, the following relationship holds:

$$\Phi_{\bar{i}\bar{j}\bar{k}} = \beta_{\bar{i}i}\beta_{\bar{j}j}\beta_{\bar{k}k}\Phi_{ijk}.$$

1.3.5 Tensor of Rank 4

A tensor of rank 4 (fourth-rank tensor) is defined as a quantity, Φ_{ijkl}, satisfying the following:

(a) Having 16 components in 2-D (81 in 3-D).
(b) When the coordinate system is changed from x_i to $x_{\bar{i}}$, the following relationship holds:

$$\Phi_{\bar{i}\bar{j}\bar{k}\bar{l}} = \beta_{\bar{i}i}\beta_{\bar{j}j}\beta_{\bar{k}k}\beta_{\bar{l}l}\Phi_{ijkl}.$$

Fourth-rank tensors are important in continuum mechanics, as elastic constants are an example of fourth-rank tensors.

Here is an example of a code to implement a fourth-rank tensor, C_{ijkl}, defined as

$$C_{ijkl} = \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) + \lambda\delta_{ij}\delta_{kl},$$

and transform the tensor into the coordinate system rotated from the original by θ.

```
In[9]:= delta[i_, j_] := If[i == j, 1, 0]
        β = {{Cos[θ], Sin[θ]}, {-Sin[θ], Cos[θ]}};
        c = Table[
            μ (delta[i, k] delta[j, l] + delta[i, l] delta[j, k]) +
             λ delta[i, j] delta[k, l],
            {i, 2}, {j, 2}, {k, 2}, {l, 2}];
        cnew = Table[Sum[β[[i, i1]] β[[j, j1]] β[[k, k1]]
              β[[l, l1]] c[[i1, j1, k1, l1]], {i1, 2}, {j1, 2},
             {k1, 2}, {l1, 2}], {i, 2}, {j, 2}, {k, 2}, {l, 2}];
```

The quantity `c` is defined as a fourth-rank tensor. The quantity `cnew` is an expression of `c` in the coordinate system rotated from the original coordinate system by θ.

```
In[13]:= c
Out[13]= {{{{λ + 2 μ, 0}, {0, λ}}, {{0, μ}, {μ, 0}}},
          {{{0, μ}, {μ, 0}}, {{λ, 0}, {0, λ + 2 μ}}}}
```

It is noted that the fourth-rank tensor is stored as a nested list. For instance, the C_{1111} component can be referenced as

```
In[14]:= c[[1, 1, 1, 1]]
Out[14]= λ + 2 μ
```

The components of C_{ijkl} in the new coordinate system are expressed as

```
In[15]:= cnew
```

```
Out[15]= {{{{(λ + 2 μ) Cos[θ]^4 + 2 λ Cos[θ]^2 Sin[θ]^2 + 4 μ Cos[θ]^2 Sin[θ]^2 +
          (λ + 2 μ) Sin[θ]^4, λ Cos[θ]^3 Sin[θ] + 2 μ Cos[θ]^3 Sin[θ] -
          (λ + 2 μ) Cos[θ]^3 Sin[θ] - λ Cos[θ] Sin[θ]^3 -
          2 μ Cos[θ] Sin[θ]^3 + (λ + 2 μ) Cos[θ] Sin[θ]^3},
         {λ Cos[θ]^3 Sin[θ] + 2 μ Cos[θ]^3 Sin[θ] -
          (λ + 2 μ) Cos[θ]^3 Sin[θ] - λ Cos[θ] Sin[θ]^3 -
          2 μ Cos[θ] Sin[θ]^3 + (λ + 2 μ) Cos[θ] Sin[θ]^3,
          λ Cos[θ]^4 - 4 μ Cos[θ]^2 Sin[θ]^2 +
          2 (λ + 2 μ) Cos[θ]^2 Sin[θ]^2 + λ Sin[θ]^4}},
        {{λ Cos[θ]^3 Sin[θ] + 2 μ Cos[θ]^3 Sin[θ] -
          (λ + 2 μ) Cos[θ]^3 Sin[θ] - λ Cos[θ] Sin[θ]^3 -
          2 μ Cos[θ] Sin[θ]^3 + (λ + 2 μ) Cos[θ] Sin[θ]^3,
          μ Cos[θ]^4 - 2 λ Cos[θ]^2 Sin[θ]^2 - 2 μ Cos[θ]^2 Sin[θ]^2 +
          2 (λ + 2 μ) Cos[θ]^2 Sin[θ]^2 + μ Sin[θ]^4},
         {μ Cos[θ]^4 - 2 λ Cos[θ]^2 Sin[θ]^2 - 2 μ Cos[θ]^2 Sin[θ]^2 +
          2 (λ + 2 μ) Cos[θ]^2 Sin[θ]^2 + μ Sin[θ]^4,
          -λ Cos[θ]^3 Sin[θ] - 2 μ Cos[θ]^3 Sin[θ] +
          (λ + 2 μ) Cos[θ]^3 Sin[θ] + λ Cos[θ] Sin[θ]^3 +
          2 μ Cos[θ] Sin[θ]^3 - (λ + 2 μ) Cos[θ] Sin[θ]^3}}},
       {{{λ Cos[θ]^3 Sin[θ] + 2 μ Cos[θ]^3 Sin[θ] -
          (λ + 2 μ) Cos[θ]^3 Sin[θ] - λ Cos[θ] Sin[θ]^3 -
          2 μ Cos[θ] Sin[θ]^3 + (λ + 2 μ) Cos[θ] Sin[θ]^3,
          μ Cos[θ]^4 - 2 λ Cos[θ]^2 Sin[θ]^2 - 2 μ Cos[θ]^2 Sin[θ]^2 +
          2 (λ + 2 μ) Cos[θ]^2 Sin[θ]^2 + μ Sin[θ]^4},
         {μ Cos[θ]^4 - 2 λ Cos[θ]^2 Sin[θ]^2 - 2 μ Cos[θ]^2 Sin[θ]^2 +
          2 (λ + 2 μ) Cos[θ]^2 Sin[θ]^2 + μ Sin[θ]^4,
          -λ Cos[θ]^3 Sin[θ] - 2 μ Cos[θ]^3 Sin[θ] +
          (λ + 2 μ) Cos[θ]^3 Sin[θ] + λ Cos[θ] Sin[θ]^3 +
          2 μ Cos[θ] Sin[θ]^3 - (λ + 2 μ) Cos[θ] Sin[θ]^3}},
        {{λ Cos[θ]^4 - 4 μ Cos[θ]^2 Sin[θ]^2 + 2 (λ + 2 μ) Cos[θ]^2 Sin[θ]^2 +
          λ Sin[θ]^4, -λ Cos[θ]^3 Sin[θ] - 2 μ Cos[θ]^3 Sin[θ] +
          (λ + 2 μ) Cos[θ]^3 Sin[θ] + λ Cos[θ] Sin[θ]^3 +
          2 μ Cos[θ] Sin[θ]^3 - (λ + 2 μ) Cos[θ] Sin[θ]^3},
         {-λ Cos[θ]^3 Sin[θ] - 2 μ Cos[θ]^3 Sin[θ] +
          (λ + 2 μ) Cos[θ]^3 Sin[θ] + λ Cos[θ] Sin[θ]^3 +
          2 μ Cos[θ] Sin[θ]^3 - (λ + 2 μ) Cos[θ] Sin[θ]^3,
          (λ + 2 μ) Cos[θ]^4 + 2 λ Cos[θ]^2 Sin[θ]^2 +
          4 μ Cos[θ]^2 Sin[θ]^2 + (λ + 2 μ) Sin[θ]^4}}}}
```

which can be simplified to

```
In[16]:= Simplify[cnew]
Out[16]= {{{{λ + 2 μ, 0}, {0, λ}}, {{0, μ}, {μ, 0}}},
          {{{0, μ}, {μ, 0}}, {{λ, 0}, {0, λ + 2 μ}}}}
```

which shows that C_{ijkl} is isotropic.

1.3.6 Differentiation

In index notation, partial derivatives are abbreviated by a comma (,) as

$$\frac{\partial}{\partial x_i} \Rightarrow , i.$$

Examples

1.
$$x_{1,2} = \frac{\partial x}{\partial y} = 0.$$

2.
$$\begin{aligned} x_{i,i} &= x_{1,1} + x_{2,2} + x_{3,3} \\ &= \frac{\partial x}{\partial x} + \frac{\partial y}{\partial y} + \frac{\partial z}{\partial z} \\ &= 3. \end{aligned}$$

3.
$$\begin{aligned} x_{i,j} &= \frac{\partial x_i}{\partial x_j} \\ &= \delta_{ij}. \end{aligned}$$

4.
$$\begin{aligned} f_{,12} &= \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ &= \frac{\partial^2 f}{\partial x \partial y}, \end{aligned}$$

that is, higher order derivatives can be specified by additional indices after the comma.

5.
$$\begin{aligned} v_{,i} &= \frac{\partial v}{\partial x_i} \\ &\equiv \nabla v, \end{aligned}$$

hence, this is the gradient operator (∇).

6.
$$\begin{aligned} v_{i,i} &= v_{1,1} + v_{2,2} + v_{3,3} \\ &= \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \\ &\equiv \nabla \cdot v, \end{aligned}$$

hence, this is the divergence operator ($\nabla\cdot$).

Example Prove

$$\Delta(r^n) = n(n+1)r^{n-2},$$

where $r = \sqrt{x^2 + y^2 + z^2}$.

Solution: Note that $r_{,i} = x_i/r$.

$$(r^n)_{,i} = nr^{n-1}\frac{x_i}{r} = nr^{n-2}x_i.$$

Therefore,

$$\begin{aligned}(r^n)_{,ii} &= n(n-2)r^{n-3}\frac{x_i}{r}x_i + nr^{n-2}x_{i,i}\\ &= n(n-2)r^{n-3}x_i\frac{x_i}{r} + 3nr^{n-2}\\ &= n(n-2)r^{n-2} + 3nr^{n-2}\\ &= n(n+1)r^{n-2}.\end{aligned}$$

Here is a *Mathematica* code to do this automatically.

```
In[1]:= rvector = Table[x[i], {i, 1, 3}]
Out[1]= {x[1], x[2], x[3]}

In[2]:= r = Sqrt[ rvector.rvector]
Out[2]= Sqrt[x[1]^2 + x[2]^2 + x[3]^2]

In[3]:= laplace[f_] := Sum[D[f, {x[i], 2}], {i, 3}]

In[4]:= laplace[r^n] // Simplify
Out[4]= n (1 + n) (x[1]^2 + x[2]^2 + x[3]^2)^(-1 + n/2)

In[5]:= % /. {x[1]^2 + x[2]^2 + x[3]^2 → R^2}
Out[5]= n (1 + n) (R^2)^(-1 + n/2)
```

In the aforementioned code, the position vector, $\mathbf{r} = (x_1, x_2, x_3)$, is entered as a function, `x[i]`. The magnitude of $\mathbf{r}$ is stored in `r`. The Laplacian operator is denoted as `laplace`. The previous output can be represented by a % (a percent sign). The right arrow symbol (→) is the substitution rule (replace) that replaces what is to the left of the arrow by what is to the right of the arrow.

1.3.7 Differentiation of Cartesian Tensors

If a Cartesian tensor, say, $v_{ij\ldots}$, is differentiated, its derivative, $v_{ij\ldots,kl\ldots}$, is also a Cartesian tensor.[10]

[10] This is true for Cartesian tensors but not true for general tensors as will be discussed later.

Proof. For illustration purposes, assume v_i is a first-rank tensor. Hence, its transformation rule is

$$v_{\bar{i}} = \beta_{\bar{i}j} v_j. \tag{1.5}$$

Differentiating both sides of Equation (1.5) with respect to $x_{\bar{j}}$, yields

$$\begin{aligned} v_{\bar{i},\bar{j}} &= \beta_{\bar{i}j} v_{j,\bar{j}}{}^{11} \\ &= \beta_{\bar{i}j} v_{j,k} x_{\bar{j}}{}^{12} \\ &= \beta_{\bar{i}j} v_{j,k} (\beta^{-1})_{k\bar{j}} \\ &= \beta_{\bar{i}j} v_{j,k} (\beta^{T})_{k\bar{j}} \\ &= \beta_{\bar{i}j} \beta_{\bar{j}k} v_{j,k}. \end{aligned} \tag{1.6}$$

Equation (1.6) implies that $v_{i,j}$ is a second-rank tensor. The rank of the resulting tensor depends on the operation performed. For example, if v is a scalar, $v_{,i}$, is a first-rank tensor, $v_{,ij}$ is a second-rank tensor, but $v_{,ii}$ is a scalar.

Example (curl, rotation) The ith component of rotation of a vector, $\mathbf{v}$, is expressed as

$$(\nabla \times v)_i = \epsilon_{ijk} v_{j,k}.$$

The rotation operator, rot $\mathbf{v}$, can be implemented as

```
In[4]:= Table[Sum[Signature[{i, j, k}] Dt[v, x[k]], {j, 3}, {k, 3}], {i, 3}]
Out[4]= {-Dt[v, x[2]] + Dt[v, x[3]], Dt[v, x[1]] - Dt[v, x[3]], -Dt[v, x[1]] + Dt[v, x[2]]}
```

1.4 Invariance of Tensor Equations

Theorem 1.1 *If $A_{ij\ldots}$ and $B_{ij\ldots}$ are both tensors, their linear combination, $\alpha A_{ij\ldots} + \beta B_{ij\ldots}$, is also a tensor.*

Proof. Let $C_{ij\ldots} \equiv \alpha A_{ij\ldots} + \beta B_{ij\ldots}$, it follows

$$\begin{aligned} C_{\bar{i}\bar{j}\ldots} &= \alpha A_{\bar{i}\bar{j}\ldots} + \beta B_{\bar{i}\bar{j}\ldots} \\ &= \alpha \beta_{\bar{i}i} \beta_{\bar{j}j} \ldots A_{ij\ldots} + \beta \beta_{\bar{i}i} \beta_{\bar{j}j} \ldots B_{ij\ldots} \\ &= \beta_{\bar{i}i} \beta_{\bar{j}j} \ldots (\alpha A_{ij\ldots} + \beta B_{ij\ldots}) \\ &= \beta_{\bar{i}i} \beta_{\bar{j}j} \ldots C_{ij\ldots}. \end{aligned}$$

[11] As it is not possible to differentiate v_j with respect to $x_{\bar{j}}$ (a different coordinate system), the chain differentiation rule,

$$\frac{\partial}{\partial x_{\bar{j}}} = \frac{\partial x_k}{\partial x_{\bar{j}}} \frac{\partial}{\partial x_k}$$

was used.

[12] Note that

$$\frac{\partial x_{\bar{i}}}{\partial x_j} \frac{\partial x_j}{\partial x_{\bar{k}}} = \delta_{\bar{i}\bar{k}},$$

So,

$$\frac{\partial x_j}{\partial x_{\bar{k}}} = \left(\frac{\partial x_{\bar{i}}}{\partial x_j} \right)^{-1}.$$

Theorem 1.2 *If $v_{ij\ldots} = 0$ in one coordinate system, $v_{\bar{i}\bar{j}\ldots}$ vanishes in any other coordinate system.*

Proof.

$$\begin{aligned} v_{\bar{i}\bar{j}\ldots} &= \beta_{\bar{i}i}\beta_{\bar{j}j} \ldots v_{ij\ldots} \\ &= \beta_{\bar{i}i}\beta_{\bar{j}j} \ldots 0 \\ &= 0. \end{aligned}$$

Theorem 1.3 *If $a_{ij\ldots} = b_{ij\ldots}$ holds in one coordinate system, it holds in any other coordinate system.*[13]

Proof. Let $C_{ij\ldots} \equiv a_{ij\ldots} - b_{ij\ldots}$.
It follows

(a) $C_{ij\ldots}$ is a tensor (from Theorem 1).
(b) Since $C_{ij\ldots} = 0$, $C_{\bar{i}\bar{j}\ldots} = a_{\bar{i}\bar{j}\ldots} - b_{\bar{i}\bar{j}\ldots} = 0$ (from Theorem 2).
(c) Hence $a_{\bar{i}\bar{j}\ldots} = b_{\bar{i}\bar{j}\ldots}$.

By this statement, it was shown that a (Cartesian) tensor equation is invariant in any coordinate system.

1.5 Quotient Rule

If $AB = C$ holds and A and C are both tensors (indices not shown), B must be a tensor.

Examples

1. Force

   The force, f, is a tensor as it is defined by

   $$f u_i = W,$$

   where u_i is the displacement and W is the work done. As both the displacement, u_i, (a first-rank tensor) and the work done, W, (a zeroth -rank tensor) are tensors, f must be a tensor from the quotient rule. Moreover, f must have an index, i, for both sides to be consistent. Thus, the force must be a first-rank tensor, f_i.
2. Elastic constants

   The elastic constant, C, is defined as the proportionality factor between the stress, σ_{ij}, and the strain, ϵ_{ij}, as

   $$\sigma = C\epsilon.$$

   Since both the stress and the strain are second-rank tensors as will be shown in the next chapter, the above equation must be written as

   $$\sigma_{ij} = C_{?}\epsilon_{kl}.$$[14]

[13] Because of this theorem, tensor equations are independent of the frame of reference.
[14] The direction of σ is not necessarily colinear with the direction of ϵ, hence, different indices (ij and kl).

From the quotient rule, C must be a tensor as both σ and ϵ are tensors. Furthermore, C must have indices of $ijkl$ for both sides of the aforementioned equation to be consistent. Thus, we have

$$\sigma_{ij} = C_{ijkl}\epsilon_{kl}.$$

The component, C_{ijkl}, can be interpreted as the ij component of the stress for the kl component of the applied strain.

3. Thermal conductivity
   The thermal conductivity, k, is defined as the proportionality factor between the heat flux, h_j, and the temperature gradient, $T_{,i}$, as

$$h_j = k_? T_{,i}.$$

As both h_j and $T_{,i}$ are first-rank tensors, k must be a second-rank tensor as

$$h_i = -k_{ij}T_{,j},$$

where the minus sign is introduced for a historical reason.

1.6 Exercises

1. Rewrite the following using the summation convention.
   (a) $a_1x_1 + a_2x_2 + a_3x_3 = p$.
   (b) $(a_{11} + a_{22} + a_{33})(a_{11}^2 + a_{12}^2 + a_{21}^2 + a_{22}^2)$.
   (c) $a_{11}x^2 + a_{12}xy + a_{21}xy + a_{22}y^2 = c$.
   (d) $\frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial z}dz$.
   (e) $a_{11}a_{22} - a_{12}a_{21}$.
2. Given $\phi_1 = -y$, $\phi_2 = x$,
   (a) Is ϕ_i a vector ? Why ?
   (b) Is $\phi_{i,j}$ a tensor ? Why ?
3. Compute $(\frac{1}{r})_{,ii}$ where $r = \sqrt{x^2 + y^2}$.
4. Is $v_{ij} = \begin{pmatrix} xy & y^2 \\ x^2 & -xy \end{pmatrix}$ a tensor ? Why ?
5. Thermal conductivity for an orthotropic material is a second-rank tensor and is expressed as $k_{ij} = \begin{pmatrix} k_{11} & 0 \\ 0 & k_{22} \end{pmatrix}$.
   (a) Obtain the components of $k_{\bar{i}\bar{j}}$ in the coordinate system $(x_{\bar{i}})$ rotated from the original coordinate system by $\pi/4$.
   (b) Calculate $k_{\bar{i}\bar{j}}k_{\bar{i}\bar{j}}$ in the rotated coordinate system $(x_{\bar{i}})$.
6. Express

$$C_{ijkl}w_{,ijkl}$$

where

$$C_{ijkl} = \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) + \lambda\delta_{ij}\delta_{kl},$$

without using index notation (i.e., using x and y) in 2-D.

7. Express the following in full notation (in terms of x and y assuming 2-D).

$$C_{ijkl}w_{,ij}w_{,kl} = 0,$$

where $C_{ijkl} = \alpha(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) + \beta\delta_{ij}\delta_{kl}$.

8. Compute $\Delta\Delta r$ where $r^2 = x^2 + y^2 + z^2$ and $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial x^2}$.

9. Is

$$v_{ij} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

a tensor? Give your rationale.

10. Write down $(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{a}$ in index notation and simplify the result.

11. (a) Prove the following identities using *Mathematica*.

$$\nabla \cdot (r^n \mathbf{r}) = (n+3)r^n,$$
$$\nabla \times (r^n \mathbf{r}) = 0,$$
$$\Delta(r^n) = n(n+1)r^{n-2}.$$

(b) Write a code in *Mathematica* to automatically generate all the equations of the following formula without index notation.

$$G\left(u_{i,kk} + \frac{1}{1-2\nu}u_{k,ki}\right) + X_i = \rho\frac{\partial^2 u_i}{\partial t^2}.$$

Hint: Use Dt rather than D. Note: Refer to the companion webpage for solutions for the Exercises section.

References

Einstein A *et al.* 1916 The foundation of the general theory of relativity. *Annalen der Physik* **49**(769–822), 31.

Fung Y 1965 *Foundations of Solid Mechanics*, vol. 351. Prentice-Hall, Englewood Cliffs, NJ.

Garvan F 2001 *The Maple Book*. Chapman & Hall/CRC.

Romano A, Lancellotta R and Marasco A 2006 Continuum mechanics using mathematica. *Continuum Mechanics using Mathematica: Fundamentals, Applications and Scientific Computing, Modeling and Simulation in Science, Engineering and Technology*. Birkhäuser, Boston, MA. ISBN: 978-0-8176-3240-3.

Wolfram S 1999 *The Mathematica Book*. Cambridge university press.