TENSOR NOTATION

BRENDAN HUANG

The purpose of this appendix is to review some of the mathematical constructs used to describe fluid flows. It is meant to serve as a sort of tutorial for those who have not been introduced to the concepts of vector calculus but who wish to do some basic calucations in fluid mechanics. I have employed some Matlab examples in an effort to illustrate many of the concepts of matrix maniuplation. As such, most of the mathematical concepts will not be rigorously defined, but rather be presented in the most inuitive possible way for non-experts.

1. Scalars, Vectors, and Tensors

Many physical quantities can be described by a single number value. For example, the diameter D of a bead, and thus its volume V in space, can both be described by single values. For most physical quantities we will deal with, these values, called *scalars* are values drawn from the real number line $\mathbb{R}$. Scalars are typically denoted by an italic letter such as s.

In contrast to scalars, certain physical quantities require an array of values to be fully described. As introduced in Chapter 2, a given velocity measurement at some location $\mathbf{r} = \langle x, y, z \rangle$ requires a measurement of three different velocity components $\mathbf{u} = \langle u, v, w \rangle$. Thus, we see that two important physical quantities, the position of a location and the velocity at that location, must be encoded not by a single scalar quantity, but rather a set of values. Specifically, these values are usually represented in a column, and the column of quantities is known as a *vector*.

In general, each component of a vector that is n-components long can be drawn independently from the real number line $\mathbb{R}$, meaning the entire vector is exists in $\mathbb{R}^n$. We will most often deal with vectors that are either 2 or 3 components because we are working with quantities in 2- and 3-dimensional space. The position vector, for example, requires either and x and y component in 2D space, or an x, y, and z compenent in 3D space. Similarly, a three-component velocity requires the specification of u, v, and w. Vectors are typically denoted either by bold, i.e. $\mathbf{v}$, or with a superscripted arrow $\vec{v}$.

Vectors are often written as an array $\mathbf{a} = \langle a, b, c \rangle$. One alterntive notation is using subscripts, i.e. a_i. The presence of a single subscript denotes the fact that i can vary,

here from 1 to 3. Thus, instead of specifying a, b, and c, we equivalently specify a_1, a_2 and a_3.

In the context of subscript notation, for a vector, only a single index i is free to vary. Thus a_i is considered one-dimensional. The natural generalization of a one-dimensional vector quantity a_i, is to add a second subscript and make it a two-dimensional quantity A_{ij}. Now, if i and j can both range from 1-3, there are 9 separate scalar quantities that need to be specified. A_{ij} is now called a tensor, a mathematical object that requires the specification of a multi-dimensional array of values. A_{ij} is a second-order tensor because two indices, i and j, can be varied. Tensors can by of any order, and a tensor of order N, for example, has N different indices. We, however, will primarily be dealing with second-order tensors. As a point of clarity, the order of the tensor, which refers to the number of indices required to specify it, is separate from the dimensionality of the space it represents, i.e. 2D or 3D space, which is encoded by the maximum range of $i = 1, 2...n$.

Tensors are often also written as $\underline{\underline{A}}$, and we will adopt this notation here. It is worth noting that a vector is actually a first-order tensor, while a scalar is a zeroth-order tensor.

In Matlab, we can illustrate the differences between a scalar, vector, and higher order tensor by writing:

```
s = 3
v = [1;2;3]
D = [1,2,3; 4,6,8; 7,3,2]
```

As described earlier, the scalar s can be encoded by a single value, while vector $\mathbf{v}$ requires 3 separate values, and tensor $\underline{\underline{D}}$ requires 3x3 values.

We note that if we try to perform the operation

```
v+D
```

on our variables as defined above in Matlab, we get an error. This is because a vector and a second-order tensor are two different objects, and cannot necessarily be added in the same way that we add scalars[1]. In general, tensors of different order must be manipulated in specific ways. When writing down equations in fluid mechanics, for example, it is good practice to keep track of the specific types of vector and tensor quantities to ensure they are being manipulated in the proper way. These types of maniuplations are the subject of the Section 3.

2. Physical Interpretation of a Tensor

To get an intuitive sense of the physical interpretation of tensors, let us consider one specific concept in fluid mechanics, that of the *Stokeslet*. A Stokeslet can be thought

[1]Although Matlab will add the scalar a to either v or D, this is an exception to the rule.

of a point force that acts on a fluid and causes the fluid at some distant location to move. Individual cilia, for example, have been modeled either as individual or collections of Stokeslets. The Stokeslet itself, $\mathbf{f}$ is a vectorial force at a given location, with components in the x,y,or$z$ direction, that causes some fluid movement in u, v, w at some distance away. More specifically, force directed along any component f_x, f_y, or f_z can cause fluid flow along any or all of the u, v, or w components. That is, even a Stokeslet f_x directed purely along the x axis will cause fluid motion in the x, y, and z direction. To denote the degree to which a force f_x causes movement in the fluid flow, we can write explicitly:

$$\begin{aligned} u &= G_{xx} f_x \\ v &= G_{yx} f_x \\ w &= G_{zx} f_x \end{aligned}$$

That is, the force f_x generates a different u, v, and w, and that is determined by the coupling coefficients G_{xx}, G_{yx} and G_{zx}. Analogously, a force in the y direction f_y will also give different contributions to $\mathbf{u}$.

$$\begin{aligned} u &= G_{xy} f_y \\ v &= G_{yy} f_y \\ w &= G_{zy} f_y \end{aligned}$$

Thus, if we have a Stokeslet with components in the x, y, and z direction, then we will have to add the contributions of $\mathbf{u}$ from each dimension. We can compactly write these contributions as a *susceptibility tensor*, G_{ij},

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} G_{xx} & G_{xy} & G_{xz} \\ G_{yx} & G_{yy} & G_{yz} \\ G_{zx} & G_{zy} & G_{zz} \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{1}$$

Thus, we see for example, that our final u component depends on all three components, as $u = G_{xx} f_x + G_{xy} f_y + G_{xz} f_z$. In fact, Eq 1 actually compactly summarizes three simultaneous equations. The physical interpreation of the tensor G_{ij} is to give all the specifications of how a force $\mathbf{f}$ translates into velocity $\mathbf{u}$. It is the components of the susceptibility, and specifically the off-diagonal components, that determine if a force directed along one axis has an influence along another axis. For example, if, $G_{yx} = G_{zx} = 0$, then that would be to say that forces along the$x$-axis translate into motion only along u.

Thus, to summarize, one function of a second order tensor is to map one vector (force) to another vector (velocity), here through a multiplicative operation.

2.1. **Einstein summation convention.** Using the subscript notation introduced earlier, Eq 1 can also be written as

$$u_i = G_{ij} f_j \tag{2}$$

The reason we have written the equation in subscript notation is to highlight what is know as the Einstein summation convention[2]. The convention states that if two objects are multiplied and contain a repeated index, the implicit rule is to sum over that index. In fact, the choice of how we assigned the indices i and j to u, G and f was not random.

To illustrate how the convention works, we note that in Eq 2 actually summarizes three separate equations for u_1, u_2, and u_3. We start by considering $i = 1$, the equation that determines our u_1 component. Eq 2 becomes $u_1 = G_{1j} f_j$. Because j appears in both G and f on the right side, we implicitly sum over all values of $j = 1, 2, 3$. Thus, the full equation can be written $u_1 = G_{11} f_1 + G_{12} f_2 + G_{13} f_3$.

The repeated index is often known as a *contraction* because it reduces the dimensionality of a tensor. For example, Eq 2, the contraction takes a tensor and a vector and produces a single vector. The contraction is an example of what is known as an inner product[3], the topic of the next section.

3. Tensor Products

In this section, we consider the various ways that tensors can be multiplied to yield higher or lower dimensional tensors. We will consider inner, outer, and cross products. As a rule of thumb, an inner product will cause a reduction in dimensionality, an outer product will cause an increase in dimensionality, and a cross-product will maintain dimensionality.

For reference, we define:

$$\underline{\underline{G}} = \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \tag{3}$$

and

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{4}$$

[2]Here we follow the convention from Pozrikidis, which makes no distinction between covariant and contravariant vectors, as opposed to what is typically followed in general relatively. A nice summary of the Einstein convention with proper contra- / covariance can be found in [1]

[3]Technically the tensor contraction becomes an inner product only when a metric tensor is given, which here we assume will be the Euclidean metric $g_{ij} = \mathbb{1}$

We have already seen first type of inner product, the inner product of a tensor with a vector. The inner product, often called the dot product is the same as right multiplying a tensor by a vector $\underline{\underline{G}} \cdot \mathbf{a} \equiv \underline{\underline{G}}\mathbf{a}$. The product is defined as:

$$\underline{\underline{G}} \cdot \mathbf{a} = \begin{bmatrix} a_1 G_{11} + a_2 G_{12} + a_3 G_{13} \\ a_1 G_{21} + a_2 G_{22} + a_3 G_{23} \\ a_1 G_{31} + a_2 G_{32} + a_3 G_{33} \end{bmatrix} \tag{5}$$

We can also left multiply a tensor by a vector, but it involves taking the transpose of the vector first first. In order to avoid the transpose notation, left multiplication is also often written as an inner product $\mathbf{a}^T \underline{\underline{G}} \equiv \mathbf{a} \cdot \underline{\underline{G}}$

$$\mathbf{a} \cdot \underline{\underline{G}} = \begin{bmatrix} a_1 G_{11} + a_2 G_{21} + a_3 G_{31} \\ a_1 G_{12} + a_2 G_{22} + a_3 G_{32} \\ a_1 G_{13} + a_2 G_{23} + a_3 G_{33} \end{bmatrix}^T$$

Note that the final object in both cases is another vector. To demonstrate these two types of dot products in Matlab, we can either simply use matrix multiplication

```
D*v
v'*D
```

Two other types of inner products exist. The first is the classic vector dot product. The dot product between two vectors $\mathbf{a}$ and $\mathbf{b}$, gives us the degree of overlap of those two vectors and is defined as $\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$. In the case of two vectors, the dot product takes two vectors and yields a scalar quantity. In our Matlab example, if we want to take the dot product of our vector v, we input

```
IP=v'*v
```

which yields a scalar quantity that is a measure of the length of the vector. This command can alternatively be written as

```
dot(v,v)
```

Finally, entire tensors can be mapped to a single scalar quantity by the double-dot product, denoted by the : symbol. The double dot, analogous to the inner product of a vector, is defined as the sum of each component squared:

$$\underline{\underline{G}} : \underline{\underline{G}} = G_{11}G_{11} + G_{12}G_{12} + G_{13}G_{13} + \\ + G_{21}G_{21} + G_{22}G_{22} + G_{23}G_{23} + G_{31}G_{31} + G_{32}G_{32} + G_{33}G_{33}$$

and the double dot product takes two tensors and yields a scalar.

In Matlab, the double dot product can be generated by the following command:

```
DD = sum(dot(D,D))
```

As we said earlier, inner products in general map higher-order tensors to lower order tensors. Outer products, or tensor products, on the other hand, map lower-order tensors to higher order tensors. For example, the tensor product of two vectors, often denoted with $\bigotimes$ is given as:

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix} \tag{6}$$

When a second order tensor is formed by the tensor product of two vectors, it is often called a dyad. Beyond dyads, tensors of arbitrarily high order can be formed using outer products of lower rank tensors by the $\bigotimes$ operation.

A vector dyad can be generated by matrix multiplication in Matlab:

```
OP=v*v'
```

Lastly, the cross product of two vectors, is usually defined in three dimensions as the following determinant of a matrix:

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} \tag{7}$$

where $\hat{i}$, $\hat{j}$, and $\hat{k}$ are the unit vectors in the x, y, and z axis. The cross product of two vectors yields a third vector.

4. Vector Fields

We have described scalars, vectors, and tensors by specifying a single array. If instead of assigning a value at a single location, we rather specify these quantities at every location in space, then the resulting object is known as a field. For example, a scalar field specifies a scalar quantity at every location in space, and a vector field specifies a vector at every location in space.

To encode fields computationally, we then need a matrix that is $m \times n$ for each point, and at each of those $m \times n$ points also can encode the appropriate tensor quantity. To demonstrate the concept of scalar and vector fields in Matlab, we first generate a grid of points:

```
m=10;
n=20;
[x,y]=meshgrid(linspace(0,10,m),linspace(0,10,n));
```

We then specify a scalar (pressure) field. Because a scalar field has only a single value to represent at every location, it can be readily visualized as an image:

```
pressure=x;
imagesc(pressure);
```

Note the size of the array by using $size(pressure)$ command in Matlab. The scalar field is encoded by an $m \times n$ array. On the other hand, if we want to encode a vector field, we need an extra dimension of encoding. That is, our matrix to hold the values goes from $m \times n$ to $m \times n \times 2$. In order to proper visualize such a quantity, we then want visualize a vector with an arrow at every location in space. Such a visualization is give by the quiver command in matlab.

```
velocity=zeros(n,m,2);
velocity(:,:,1)=(10-y).*y;
velocity(:,:,2)=zeros(n,m);
quiver(x,y,velocity(:,:,1),velocity(:,:,2));
```

5. Div, Grad, Curl, and All That

Now that we have introduced all the possible tensor operations as well as the concept of a field, we complete the picture by introducing differential operators on our fields. Differential operators acting on scalar and vector fields require the introduction of just one concept, the del operator ∇:

$$\nabla = \langle \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \rangle$$

If we recognize that ∇ is a vector, then we can then take create inner products, outer products, and cross products as we introduced previously[4]. Thus, the divergence of a vector field is just the inner product of the del operator with a vector field. If that vector is the velocity vector field, $\mathbf{u} = \langle u, v, w \rangle$, then:

$$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

Thus, as any inner product, the divergence operator takes two vectors and yields a scalar. (In the context of fluid mechanics, the divergence of the vector field is known as the compressibility. $\nabla \cdot \mathbf{u} = 0$ for an incompressible fluid, which is the only types of fluids we consider.)

The del operator can also be left multiplied to a scalar, yielding the gradient, also a vector:

[4]Like other matrix operators, it is important to remember that ∇ is non-commutative. That is, it depends whether ∇ is multiplied from the right or the left.

$$\nabla p = \langle \frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z} \rangle$$

In general, if a vector field can be written as ∇p, it is called a *conservative* field. The function p is called the scalar potential. (Indeed, a class of flows satisfying $\mathbf{u} = \nabla p$ exists and are known as potential flows. We will not consider these types of flow in this thesis).

The cross product of del with a vector field is known as the curl. In analogy with any vector operation, it is given by:

$$\nabla \times \mathbf{u} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u & v & w \end{vmatrix} \tag{8}$$

Like other cross products, the curl takes two vectors and yields another vector[5]. (In the context of fluid mechanics, the curl is of a vector field yields the vorticity. The vorticity is a measure of how much circulation there is in a field. A class of flows that we will not consider have vanishing voriticity and are known as *irrotational*.)

Finally, the dyadic (outer) product of the del operator with another vector, $\nabla \otimes \mathbf{u}$ yields a tensor. For convenience, it is often written without the $\otimes$ symbol, as $\nabla \mathbf{u}$:

$$\nabla \mathbf{u} = \begin{bmatrix} \frac{\partial}{\partial x} u & \frac{\partial}{\partial x} v & \frac{\partial}{\partial x} w \\ \frac{\partial}{\partial y} u & \frac{\partial}{\partial y} v & \frac{\partial}{\partial y} w \\ \frac{\partial}{\partial z} u & \frac{\partial}{\partial z} v & \frac{\partial}{\partial z} w \end{bmatrix} \tag{9}$$

In the context of fluid mechanics, $\nabla \mathbf{u}$ is known as the velocity gradient, while the quantity $\underline{\underline{S}} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$ is known as the rate of strain tensor, where the ^{T} superscript denotes the transpose. The rate of strain tensor gives the spatial rate of change of each of the three velocity components in each of three dimensions. The dyadic product of the rate of strain tensor, under certain circumstances, yields a metric for energy expenditure per unit volume due to viscous dissipation. This quantity ϕ is known as the dissipation function:

$$\phi = \mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) : (\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$$

where μ is the viscosity of the fluid.

We will defer Matlab examples on these vector quantities until the next appendix.

[5]Technically a pseudovector, but the distinction is beyond this appendix

References

[1] J. B. Hartle. *The Description of Curved Spacetime*, pages xxii, 582 p. Addison-Wesley, San Francisco, 2003.