What is a tensor?
Tensors from physics explained in unnecessary rigour using concepts from functional programming

Simon Jacobsson

2021-07-22

If you ask a physics professor what a tensor is, they will likely tell you that "it is something that transforms like a tensor." I was personally a bit disappointed hearing this as a Master's student in theoretical physics, not only because of the apparent circularity of that statement, but because there was no interest in exploring the deeper mathematical reasons for why tensors kept appearing everywhere. But those reasons *are* interesting, and there *is* a more insightful definition of tensors!

1 Who is this explanation for?

This text is my attempt to write down what I wish I had been taught at the beginning of my Master's programme. The explanation I am about to give is aimed at those with some university level maths and maybe physics also. Ideally you have

- heard of tensors before. Possibly in the context of physics, where you'll likely have seen Einstein's summation convention. Maybe you've even felt the same frustration over the vagueness of the concept as me.
- had some linear algebra courses. You are comfortable with cross and dot products. It is good if you know what a linear space is.
- at some point programmed using some programming language.

If you google "what is a tensor", there are already many people who have tried in many ways to answer this question. Many of them also have strong opinions about the subject. So please don't think of this as *the* definition of tensors, but rather one of many ways to define them, which if nothing else shows off some cool ideas from programming.

This question on Quora gives a fairly good overview over why tensors are difficult to pin down.

2 Linearity

You can skip this section if you know what a linear space is.

Even though most of these definitions are completely general, let's for the sake of simplicity assume every linear space that we deal with is finite-dimensional.

Many physical equations are linear. I don't know why exactly—if it is fundamental to nature somehow or if it's something we as humans impose on it because it's much easier to deal with. If an equation isn't linear, we often go to great lengths to construct expansions of linear approximations. I will take it as given that linearity is fundamental in some way.

Axiom 1. Linearity is interesting.

Axiom 1 entails both linear spaces and linear maps. If you've only had one course in linear algebra, the former is probably a bit esoteric. But a linear space is basically a set that you do linear transformations on. So let's define things in that order.

Definition 2. Let R be a field ($\mathbb{R}$ and $\mathbb{C}$ are examples of fields). If U and V are linear spaces over R and $f\colon U \to V$, then f is *linear* if

1. $f(u+v) = f(u) + f(v)$ for all a, $b \in U$.
2. $f(cu) = cf(u)$ for all $a \in U$ and all $c \in R$.

This is pretty familiar stuff! But if we think about it for a bit, what must be true about V for items 1 and 2 to make sense? Well, if f maps from V and $f(a+b)$ should be well-defined, then $a+b$ should be in V. This should be true for all a and b. Similarly, if $f(ca)$ should make sense, ca should be in V for all $a \in V$ and all $c \in R$. We have some other things we generally want to be true about linear spaces, but this is basically it.

Definition 3. Let R be a field. A set V is a *linear space over* R if

1. For each u, $v \in V$, there is a unique element $u + v \in V$ (addition).
2. For each $u \in V$ and each $c \in R$, there is a unique element cu (multiplication with scalar).

and

3. $u + v = v + u$ for all u, $v \in V$.
4. $(u + v) + w = u + (v + w)$ for all u, v, $w \in V$.
5. There is an element $0 \in V$ such that $0 + u = u + 0 = u$ for all $u \in V$.
6. For each $u \in V$ there exists an element $-u \in V$ such that $u+(-u) = (-u)+u = 0$.

7. $a(bu) = (ab)u$ for all a, $b \in R$ and all $u \in V$.
8. $a(u + v) + (au) + (av)$ for all $a \in R$ and all u, $v \in V$.
9. $(a + b)u = (au) + (bu)$ for all a, $b \in R$ and all $u \in V$.
10. $1u = u$ for all $u \in V$.

As mentioned, this is just a list of formal requirements that sets must satisfy for it to make sense to apply linear functions to their elements. These particular kinds of requirements lists are known in functional programming as *typeclasses*. A simpler example of a typeclass is *totally ordered sets*, which are sets S with a some binary operation $\leq\colon S \times S \to \{\,\mathrm{True}, \mathrm{False}\,\}$ satisfying

1. $x \leq x$.
2. $x \leq y$ and $y \leq x \implies x = y$.
3. $x \leq y$ and $y \leq z \implies x \leq z$.
4. Either $x \leq y$ or $y \leq x$.

For all x, y, $z \in S$.

Note that, in items 1 and 2 of definition 2, the addition and multiplication with scalar are different operators on each side of the equals sign. In the left-hand side of item 1, $+\colon U \times U \to U$, while in the right-hand side $+\colon V \times V \to V$. Both of these operators should of course satisfy definition 3 for U and V respectively.

Let's also define what we mean when we say that two linear spaces are equal.

Definition 4. Two linear spaces U and V over a field R are *isomorphic* if there exists some linear bijection $f\colon U \to V$. f is then an *isomorphism*.

Two examples of linear spaces are $\mathbb{R}^2$ over $\mathbb{R}$ and $\mathbb{C}$ over $\mathbb{R}$. These two are in fact isomorphic, easily seen by extending the map $(1, 0) \mapsto 1$, $(0, 1) \mapsto i$ linearly. Another example which is not isomorphic to those two is $\mathbb{C}$ over $\mathbb{C}$, which shows the importance of having the field in mind.

A useful notion in linear algebra is the dual space.

Definition 5. Let V be a linear space over R. The *dual space* V^* to V is the set of linear maps $: V \to R$.

If $\langle \cdot, \cdot \rangle$ is the natural inner product on $\mathbb{R}^n$, then an example of a dual space is

$$(\mathbb{R}^n)^* = \{\, \langle v, \cdot \rangle \quad \text{s.th.} \quad v \in \mathbb{R}^n \,\}.$$

Proposition 6. $V^{**} = V$.

You can prove this if you want to, but, for me at least, it feels like it really should be true.

3 Einstein's summation convention

Index notation, primarily used in relativistic physics, builds on Einstein's summation convention. Allegedly, this was one of the discoveries that Einstein was most thrilled about: that when writing out the components in linear algebraic expressions, repeated indices are always summed over. So if $v \in V$ is a vector with components v_α in an orthonormal basis B and $L\colon V \to V$ is a linear map with components $L^\alpha{}_\beta$ in B, then $L(v)$ has components

$$\sum_\alpha L^\alpha{}_\beta v_\alpha,$$

or, with Einstein's summation convention,

$$L^\alpha{}_\beta v_\alpha.$$

In this text, I will use something similar but different: *abstract indices*. In abstract index notation, indices no longer denote components but the actual tensors themselves. If $v_a \in V$ is a vector and $L^a{}_b\colon V \to V$ is a linear map, then the action of $L^a{}_b$ on v_a is still written

$$L^a{}_b v_a,$$

but the repeated index does not mean summation anymore. It just means *action*.

In this context, a repeated index is called a *contraction*.

4 Partial application

In the more well-known programming languages like Java, Python, C, etc., the usual procedure is to define some variables and then modify those until you get them to the state that you want them to be in. Consider for example the short Python program below and its output,

```
L = [1, 4, 3, 2]
L.sort()
print(L)

out: [1, 2, 3, 4]
```

Note that using the method `sort()` changes the state of `L`.

In contrast, with functional programming like Haskell, you can't modify variables once they're defined. All you can do is to write down all of the variables you want to use and then apply functions to them. I use the word "function" here in its strict mathematical sense: A *function* (or *map*) f from a set A to a set B is a rule that assigns, to each $a \in A$, a unique $b \in B$. We often write $f\colon a \mapsto b$ or $f(a) = b$. Importantly, this means that

1. a function does alter any state—there's no function for switching your clock from winter time to summer time.

2. for a given argument a, a function *always* returns the same element b.

It may seem that this is very limiting, but it actually turns out that writing code this way feels much more like doing maths than the other type of coding! Since everything that you can use are functions, you get very intimate with the properties of functions when programming functionally.

One important concept is *partial application*, or *currying*. Consider a function f that takes two arguments. Maybe it looks like this: $f\colon A \times B \to C$. What if you know that the first argument you'll be supplying is $a \in A$? Then for all intents and purposes, you have a new function $g\colon B \to C$ defined by $g(b) = f(a, b)$. But you will also have a different such function for each a. So you can think of f really as a function $\hat{f}\colon A \to B^C$ where B^C denotes the set of functions from B to C.

Are f and $\hat{f}$ really so different then? In many cases, I would like to be agnostic about the amount of arguments that I'm about to supply for f. In functional programming, f and $\hat{f}$ are the same object! It is written

$$f\colon A \to B \to C.$$

The function $\hat{f}\colon a \mapsto g$ is called a partial application of f to a.

As an example, consider the following Haskell program,

```
bool2Int True = 1
bool2Int False = 0
heaviside = bool2Int . ( > 0)
```

which defines the Heaviside step function

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}.$$

Writing `( > 0)` means we partially apply the comparison `<`: $\mathbb{R} \times \mathbb{R} \to \{\,\text{True}, \text{False}\,\}$ to 0. The dot on the third line just means *composition*, we compose the function `( > 0)` : $\mathbb{R} \to \{\,\text{True}, \text{False}\,\}$ with `bool2Int`: $\{\,\text{True}, \text{False}\,\} \to \mathbb{R}$.

5 Defining tensors as multilinear maps

A *multilinear map* is a function that takes several arguments and is linear in each one. Cross products are an example since

$$\begin{aligned} (\vec{a} + \vec{b}) \times \vec{c} &= \vec{a} \times \vec{c} + \vec{b} \times \vec{c}, \\ (r\vec{a}) \times \vec{b} &= r(\vec{a} \times \vec{b}), \\ \vec{a} \times (\vec{b} + \vec{c}) &= \vec{a} \times \vec{b} + \vec{a} \times \vec{c}, \\ \vec{a} \times (r\vec{b}) &= r(\vec{a} \times \vec{b}). \end{aligned}$$

Another example is scalar products for similar reasons. Yet another example is the (signed) volume of a parallelepiped. If you have three vectors $\vec{a}$, $\vec{b}$, and $\vec{c}$ in $\mathbb{R}^3$, then they define a parallelepiped by figure 1. Look closely at the figure and try to see why each vector contributes linearly to the volume.

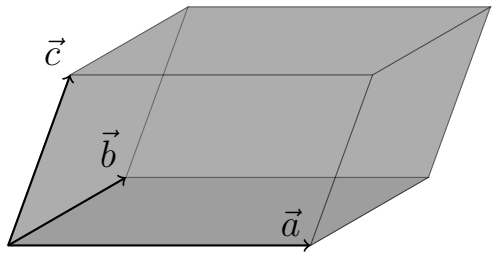

Figure 1: If $\vec{a}$, $\vec{b}$, and $\vec{c}$ are positively oriented according to the right-hand rule, then the volume is positive.

The magic is admittedly a bit lost if you know that the signed volume of the parallelepiped is $\vec{a} \cdot (\vec{b} \times \vec{c})$. But there *are* examples of multilinear maps that are not cross or dot products. The *outer product* of two vectors is

$$\left(\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \right) \mapsto \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \end{bmatrix}. \tag{1}$$

So these are a few examples of multilinear maps. Now let's make the definition that you came here to for.

Definition 7. Let V be a linear space over a field R. A valence (k, l) *tensor* is multilinear map

$$f\colon \underbrace{V \times \cdots \times V}_{\times l} \to \underbrace{V \times \cdots \times V}_{\times k}.$$

Or, equivalently, a tensor is a multilinear map

$$f\colon \underbrace{V \times \cdots \times V}_{\times l} \times V^* \to \underbrace{V \times \cdots \times V}_{\times (k-1)}$$

or a multilinear map

$$f\colon \underbrace{V \times \cdots \times V}_{\times (l-1)} \to V^* \times \underbrace{V \times \cdots \times V}_{\times k}$$

etc.

To see the isomorphism between these sets of multilinear maps, we will use partial application, along with proposition 6. For the special case of a map $f\colon V \to V$, the argument goes like this. Map $f\colon V \to V$ to the multilinear $f'\colon V \times V^* \to R$ by $f'(v, u^*) = u^*(f(v))$. Then map $f'\colon V \times V^* \to R$ to $f''\colon V \to (V^*)^*$ by $f''(v) = f'(v, \cdot)$. But since $(V^*)^* = V$, $f''\colon V \to V$. The kernels of each of the two maps $f \mapsto f'$ and $f' \mapsto f''$ are clearly 0 and they are both clearly linear, so each of them must be an isomorphism[1]. It is not hard to extend this argument to all different kinds of combinations of V and V^*.

The preceding paragraph is fairly dense and contains some plausible-sounding but non-obvious things. Read it slowly. In essence, it proves that we can write tensors as multilinear maps

$$f\colon \underbrace{V \to \cdots \to V}_{\times l} \to \underbrace{V^* \to \cdots \to V^*}_{\times k} \to R.$$

6 Tensor product

You could stop reading here and still get away with a pretty solid understanding of what a tensor is, but if you're interested, we could talk a bit about the symbol $\otimes$ that you may have seen in relation to tensors.

Definition 8. Let $U_1, \ldots, U_n$ be vector spaces over a field R. Then $U_1 \otimes \cdots \otimes U_n$ is a linear space equipped with a multilinear map $\pi\colon U_1 \times \cdots \times U_n \to U_1 \otimes \cdots \otimes U_n$ such that for any multilinear map $f\colon U_1 \times \cdots \times U_n \to R$ there exists a unique linear map $g\colon U_1 \otimes \cdots \otimes U_n \to R$ such that $f = g \circ \pi$.

The space $U \otimes V$ is called the *tensor product space* of U and V. Definition 8 is fairly dense, but what it says is basically that for every function bilinear on $U \times V$, there is a function that is linear on $U \otimes V$. So we have yet another way of writing definition 7: a tensor is a linear map

$$f\colon \underbrace{V \otimes \cdots \otimes V}_{\times l} \otimes \underbrace{V^* \otimes \cdots \otimes V^*}_{\times k} \to R.$$

This looks somewhat familiar thought. Recalling definition 5, we can rewrite the

[1]There are three statements here that must be motivated: $\ker(f \mapsto f') = 0$, $\ker(f' \mapsto f'') = 0$, and that $f \mapsto f''$ is an isomorphism. Can you come up with concise arguments for these when V is infinite-dimensional? I don't think that I can.

set of valence (k,l) tensors as

$$\begin{aligned}
&\{\, f \quad \text{s.th.} \quad f \text{ is a valence } (k,l) \text{ tensor} \,\} \\
&= \{\, f\colon \underbrace{V \otimes \cdots \otimes V}_{\times l} \otimes \underbrace{V^* \otimes \cdots \otimes V^*}_{\times k} \to R \quad \text{s.th.} \quad f \text{ is linear} \,\} \\
&= \Big(\underbrace{V \otimes \cdots \otimes V}_{\times l} \otimes \underbrace{V^* \otimes \cdots \otimes V^*}_{\times k}\Big)^* \\
&= \underbrace{V^* \otimes \cdots \otimes V^*}_{\times l} \otimes \underbrace{V \otimes \cdots \otimes V}_{\times k}.
\end{aligned}$$

The last step is that taking the dual space distributes over tensor products. It is not hard to believe, and it is not difficult to prove[2]. Hence we can make one last very neat reformulation of definition 7: A tensor is an element of

$$\underbrace{V^* \otimes \cdots \otimes V^*}_{\times l} \otimes \underbrace{V \otimes \cdots \otimes V}_{\times k}.$$

7 How to transform like a tensor

What about the physics professor's definition of a tensor? Well, let's think about what happens to the components of a tensor $T_i{}^j\colon V \to V^* \to R$ when we make a coordinate change to V. If v is a vector in V whose components transform as $v^i \mapsto A_j{}^i v^j$ by a change of coordinates, and if $v^* = \langle v, \cdot \rangle \in V^*$, then the components of v^* should transform as $(v^*)_i \mapsto (A^{-1})^j{}_i (v^*)_j$. The reason for this is that $|v|^2 = v^* v$ is invariant under coordinate changes, and

$$\begin{aligned}
v^* v &= v^i (v^*)_i \\
&\mapsto v^j A_j{}^i (A^{-1})^k{}_i (v^*)_k \\
&= v^j \delta_j{}^k (v^*)_k \\
&= v^* v
\end{aligned}$$

shows that this is satisfied. So with some abuse of notation, let's write down how $T(v, v^*)$ transforms:

$$\begin{aligned}
T_i{}^j\big(v^i, (v^*)_j\big) &\mapsto T\big(A_k{}^i v^k, (A^{-1})^l{}_j (v^*)_l\big) \\
&= A_k{}^i (A^{-1})^l{}_j T_i{}^j\big(v^k, (v^*)_l\big).
\end{aligned}$$

Hence the components of $T_i{}^j$ transform as

$$T_i{}^j \mapsto A_k{}^i (A^{-1})^l{}_j T_i{}^j. \tag{2}$$

[2]The idea is that $\phi\colon V^* \otimes W^* \to (V \otimes W)^*$ defined by $\phi(f \otimes g)(v \otimes w) = f(v)g(w)$ is an isomorphism.

In physics you often deal with tensor fields. Now I don't use "field" as in $\mathbb{R}$ or $\mathbb{C}$ anymore but as in a function from the space or spacetime manifold M to the space of tensors. Then V is often the tangent space to M. If we make a coordinate change $x \mapsto \hat{x}$ on M, then that induces a coordinate change ${A_i}^j = \partial \hat{x}_i / \partial x_j$ of the tangent space. But just substituting this into (2) and using $(\partial \hat{x}_i / \partial x_j)^{-1} = \partial x_i / \partial \hat{x}_j$ gives

$$ {T_i}^j \mapsto \frac{\partial \hat{x}_k}{\partial x_i} \frac{\partial x_j}{\partial \hat{x}_l} {T_i}^j. $$

This is the transformation law that physicists mean when they say something transforms as a tensor. It generalizes in the obvious way for higher valence tensors.