Jad Nohra, Private Note. Revised: August 25, 2017

$E^n, \mathbb{R}^n$, and Numerical Computation

1. Once again we struggle with the basics of the imperfect (or misunderstood) notation when relating numerical computations (e.g in a physical simulation) to Euclidean space and to the real vector space.

2. We still felt there are missing unsaid lines and hidden choices when we say that one can choose $\overrightarrow{e_1} = (1,0,0)^*$ as a basis vector in the otherwise coordinate-free E^n. But there are none, and we keep forgetting this. One can prove that $\mathbb{R}^n$ satisfies the definition of a vector space, so $(1)^*, (1,0)^*, (1,0,0)^*$ are all ***vectors*** and not coordinates. They are column vectors due to the possibility to use the list-like nature of $\mathbb{R}\times\mathbb{R}\times\mathbb{R}$ as compared to a generic three dimensional vector space V. So we can correctly write the arrow: $\overrightarrow{\begin{pmatrix}1\\0\end{pmatrix}}$.

3. The above vectors can be confused with vector coordinates because they look like them, and this confusion lead to the costly implicit refusal of saying

$$\overrightarrow{e_1} = (1,0,0)^*$$

which can be notational bent to the more acceptable

$$\overrightarrow{e_1} = \overrightarrow{(1,0,0)^*}$$

while the coordinates of a vector (assuming $\mathbb{R}$ as a field) would be written to avoid confusion as

$$(1,0,0)^*$$

4. The reason for all of this (and the utility of work in $\mathbb{R}^n$) is that it has additional structure on top of being an (abstract) vector space, so it ***brings them together***, allowing doing field-like calculations (on lists of field elements) on vectors. $\mathbb{R}^n$ is a vector space ***and*** $\mathbb{R}$ itself is a field. So if our vector space is n-dimensional over $\mathbb{R}$, its vector coordinates are in the cartesian product of the field (n times) which is again $\mathbb{R}^n$. This applies to the vector space $\mathbb{R}^n$.

5. In general, $c\,\overrightarrow{v}$ is just that, but when $\overrightarrow{v}$ is in $\mathbb{R}^n$, the extra structure allows us yet an additional thing: $2\,\overrightarrow{1} = \overrightarrow{2}$, and this explicitness is computationally convenient.

6. With the choice of canonical basis E^n, we see that vectors and their representations match exactly, this is convenient too, but can lead to confusions and thinking something is being swept under the rug, while all is correct. If anything, what is could be clarified is that we are formally identifying

$$\overrightarrow{(1,0,0)^*} \equiv (1,0,0)^*$$

noting that this is only true under a certain choice of basis.

7. Knowing the above allows us to work comfortably within a numerical simulation. When the computer code initializes a matrix in memory with

$$\big(1,0,0,0,1,0,0,0,1\big)$$

what do we mean? Is it a matrix? that is, the representation of some linear map?

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

If yes then under which choice of two bases? We did not choose any bases in the code, we just wrote down a list of numbers. Indeed, there is no need to chose a basis and consider this list to be a matrix, a representation with a mysterious choice pending (although of course we can, and nothing breaks). But we can simply consider the list to formally mean, and identify it with

$$\overrightarrow{\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}}, \overrightarrow{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}}, \overrightarrow{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}$$

and this is probably the right meaning to choose. Matrices of coordinates will still figure, since vectors need to be expressed in terms of the canonical basis or other bases, producing change of basis 'matrices', 'matrices' that transform vectors into vectors and not vector coordinates into vector coordinates, 'matrices' in a tensor-like treatment, 'matrices' for which we do not care about what linear maps they represent (meaningless in this context) and this was always an implicit confusion and problem that we finally solved here.

8. Note that the purpose of working in E^n is usually the coordinate-free treatment of physical laws, which is almost never the case when one is deriving the exact numerical manipulations to be coded. The choice of an origin and of a cartesian coordinate system need not be even mentioned, lest their properties are to be talked about. As we know, the coordinate system in E^n need not be cartesian, it could be skewed and even curvilinear (e.g cylindrical) in which case the vectors are not even drawn as straight arrows, a spherical system has angular coordinates, to which a velocity is in radians per second, we can happily draw the related coordinate system's vector as a piece of arc.

9. But why did we choose this basis of vectors? We did because it was found by Newton that cartesian coordinates is the setting in which his physics works. This was later generalized into Euclidean spaces, etc. So it need not be justified but we will still do it. In the end we mean something by the quantities we use in the simulation. Usually, this will go to rendering and the rendering has to display what we meant, in a way that it matches what we wanted to model in the real world. What we wanted to model in the real world, Newtonian dynamics, assumes we have three orthogonal (this can be done by symmetry) one meter (this can be done by comparing to certain chosen rods) sticks, held fixed, handedness chosen.

What we mean by the quantities we calculate in the simulation is usually what we mean by coordinates in the real world with respect to these rods (remember that Newton's system is coordinate-based and not coordinate-free). If we fill in our simulation a list that is supposed to be a velocity with the numbers $(0, 2, 0)$ we expect a particle having this constant velocity to move in a certain way, and we expect our simulation to reproduce that. Hence, we are identifying our system of rods with a certain basis, the basis we mentioned earlier. Can we choose other identifications? Yes. Can we choose any identifications? No. The properties of our system of rods must match the properties of the basis chosen so that we see what is expected at the display. We need:

1. The basis vectors to be pairwise orthogonal.
2. The basis vectors to be of the same length.

If these properties do not hold for our chosen basis, we will see 'wrong' results. In extreme cases, we will also calculate results that can not be adapted by a final re-adjustment step, per example, if we choose a linearly dependent set. We did not say anything about scale. Indeed, we could have chosen any scaled version of the basis (or a rotated one). This amounts to identifying the algebraic length of the basis vectors with a length in the real world.

10. A year after this note, we had similar doubts whilst derive the matrices of a pinhole-camera model projection here https://github.com/jadnohra/jupyter_goodies . One additional conclusion is that the right and actually correct way is to first do analytic geometric in 3D on paper. There is no fancy way around that essentially even if one is deluded of the possibility. At the end of the day, for a task such as modeling a pinhole-camera projection, the job is exactly to encode into computation what is happening in the analytic geometry. So one better put the sensor CCD in the drawing itself, as well as the image pixels, along with any direction vectors. One can then using analytic geometry and equations derive all the matrices needed, by encoding the equations into them.

Sometimes matrices are not enough, e.g for the perspective projection step. Here, the trick is to use homogenous coordinates but one must note that this is but a trick! We are not 'fully' using homogeneous coordinates nor do we need to think about them in the analytic setting. In the analytic setting, we draw axes and use coordinates. As mentioned in the previous note, we could choose the axes at will, but we will not be getting what we expect because that is not what we are modeling. The question does arise whether the geometry of the real world is captured exactly by this, but we don't care about being so exact, and common day experience (short of proving it) shows that this is a good approximation.

Another way that can circumvent have to do many messy drawings on paper (where the number of lines is more than the magic number of sever and the untrained or unwilling brain is stumped), is to find out the types of the matrices involved by stipulating about the kinds of transformations being done in the analytic drawing. Then, some detail can be lost about handedness and similar. If one does nothing about it, one is again in the situation where one can recover the detail by trial and error. If the number of combinations is too large, one can additionally try to recover detail by constraining matrices, per example by requiring that some special (e.g unit) vectors transform in determined ways to other special vectors, and adjusting the matrices to satisfy these constraints.

Having done all the analytic geometry and turned it into matrices or other nonlinear transformations, how should one interpret in less geometric ways the results and go back to the happy land of symbolic manipulation. Hand-wavingly and by implicit copy-catting of what learned to work in practice, one could be confident that the matrices and vectors, seen now as only containing coordinates with actual abstract vectors nowhere to be found, actually sill do model what is hoped. A proof of that would be some kind of meta-proof that checks that the properties of the objects in the analytic sense do match the properties of these in the abstract sense, but note that here one already did jump over a bridge of abstraction (that one need not necessarily do for simple applications). In the abstract sense, one is usually actually dealing with points in $\mathbb{R}^3$ (not vectors, for that case one is probably talking about points at the tips and bases of rods). When talking about rods, one is probably not talking about points in the space but about transformations in groups such as the special Euclidean group [1]

$$SE(3) = SO(3) \times T(3)$$

which is a subgroup of the Euclidean group $E(3)$, of which both $SO(3)$ and $T(3)$ are subgroups. A 'coordinate frame' such as the 'camera transform' belongs then to this group, although one can also encode it as four actual points (the point at which the camera is and three points for the tips of rods starting at the camera point) in the space $\mathbb{R}^3$ from which the coordinate transformation matrix can be calculated having made 'some choices'.

[1] https://en.wikipedia.org/wiki/Euclidean_group

Possibly though, what is doing when choosing points to encode a coordinate space is that one is working within an affine space and using barycentric coordinates [2]

> An affine frame of an affine space consists of a point, called the origin, and a linear basis of the associated vector space.
>
> Example: In Euclidean geometry, Cartesian coordinates are affine coordinates relative to an orthonormal frame, that is an affine frame (o, v1, ..., vn) such that (v1, ..., vn) is an orthonormal basis.

Note that the relation between the Euclidean space E^3 and the Euclidean group $E(3)$ is [3]

> The structure of Euclidean spaces – distances, lines, vectors, angles (up to sign), and so on – is invariant under the transformations of their associated Euclidean group.

What is the formal definition of the cartesian coordinate system? [4]

> The use of a coordinate system allows problems in geometry to be translated into problems about numbers and vice versa; this is the basis of analytic geometry
>
> The prototypical example of a coordinate system is the Cartesian coordinate system. In the plane, two perpendicular lines are chosen and the coordinates of a point are taken to be the signed distances to the lines.

It seems this is no easy question. [5] So once more we find a concept that is used a lot but with no rigorous standard definition. As usual, we think that the problem lies in the fact that it may be way too boring to draw the correspondence between the geometric/analytic and the abstract in full detailed proof although it can be done. But here is a good answer (on mst by none other than John Lee himself!):

> No, the geometry has no bearing on the coordinate maps you can construct, because coordinates do not have to have any direct relationship to a metric. (The definition of a "coordinate chart" makes no reference to a metric.) You can use Cartesian coordinates if and only if your manifold is an open subset of Rn, regardless of whether it's endowed with the Euclidean metric, some other metric, or no metric. What is true is that unless the metric you're interested in on that open set is the Euclidean one, then Cartesian coordinates will not have any relationship with the metric.

[2] https://en.wikipedia.org/wiki/Affine_space#Affine_coordinates
[3] https://en.wikipedia.org/wiki/Euclidean_space#Euclidean_group
[4] https://en.wikipedia.org/wiki/Cartesian_coordinate_system
[5] https://math.stackexchange.com/questions/1333946