Notes: Skinning Mesh Animations

Papoj Thamjaroenporn

April 13, 2015

1 Conditions that Enforce Rotation Matrix

Given a matrix R you can enforce it to be rotation matrix by requiring:

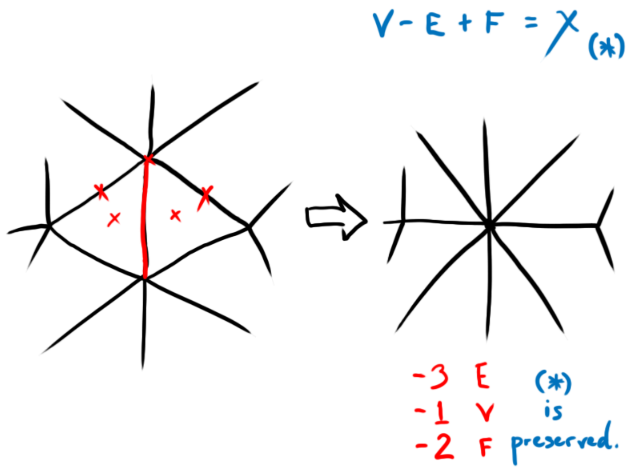
$$R^T R = I, \tag{1}$$
$$\det R = 1. \tag{2}$$

Equation (1) enforces orthonormality but it can still be a flip matrix. Equation (2) ensures that there is no flipping (because determinant equals the product of eigenvalues, and positive eigenvalues mean you have even number of minus signs. Flipping twice, even though in different axes, brings you back to the original orientation.)

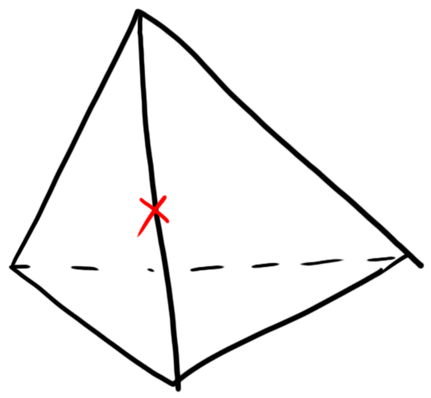
2 A Really Simple Condition for Good Edge Collapsing

By good we mean after we collapse the edge we don't get weird things like topology breaking or hanging triangles (changing manifold mesh to non-manifold). A simple way to do this is to look at Euler equation that describes relationship between the number of faces, vertices, and edges.

Normally when we collapse an edge, we subtract certain number of edges, vertices, and faces, related to the edge we are collapsing. Everyone is happy when the Euler characteristic is preserved.

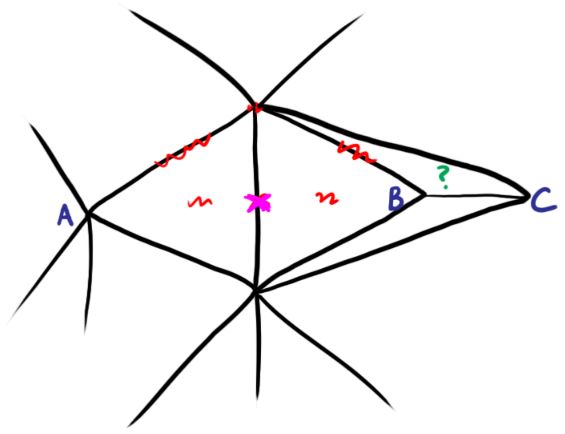


Can you think about the case that simple edge collapsing does not end happily? Consider the following special case:



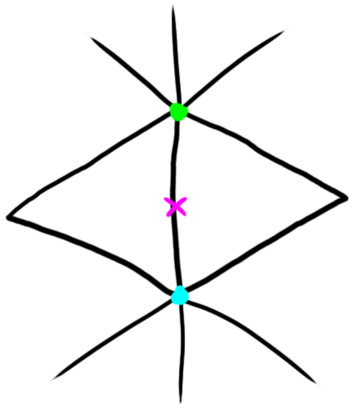
The back triangle and the bottom triangle will have a problem: they collapse into each other too, but we didn't clean them up from simple edge collapsing? Can you think of other general scenarios?

It turns out that, in general when the mesh is large (not like a simple tet above), there is a really simple way to check when simple edge collapsing is appropriate. Let's take a look at this:



When the two vertices we are collapsing (whose edge is pink) share more than two neighbors (in this case there are three: A, B, C), then you can see that simple edge collapsing will create a problem with the face marked with question mark.

Therefore, the general theorem is, when the two vertices we are collapsing share exactly two neighbors, everybody should all be happy after collapsing.

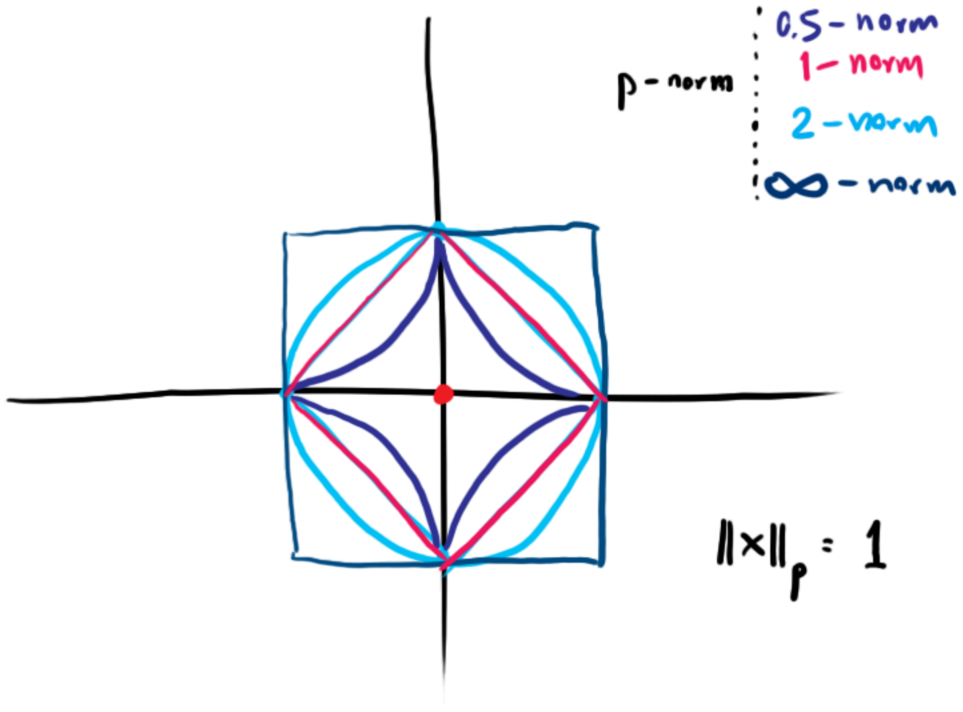


3 Sparsity of Positive Weights

To understand how we can formulate energy minimization with sparsity requirement, we first need to understand two things.

3.1 p-norm

Recall our familiar plots of different p-norms.



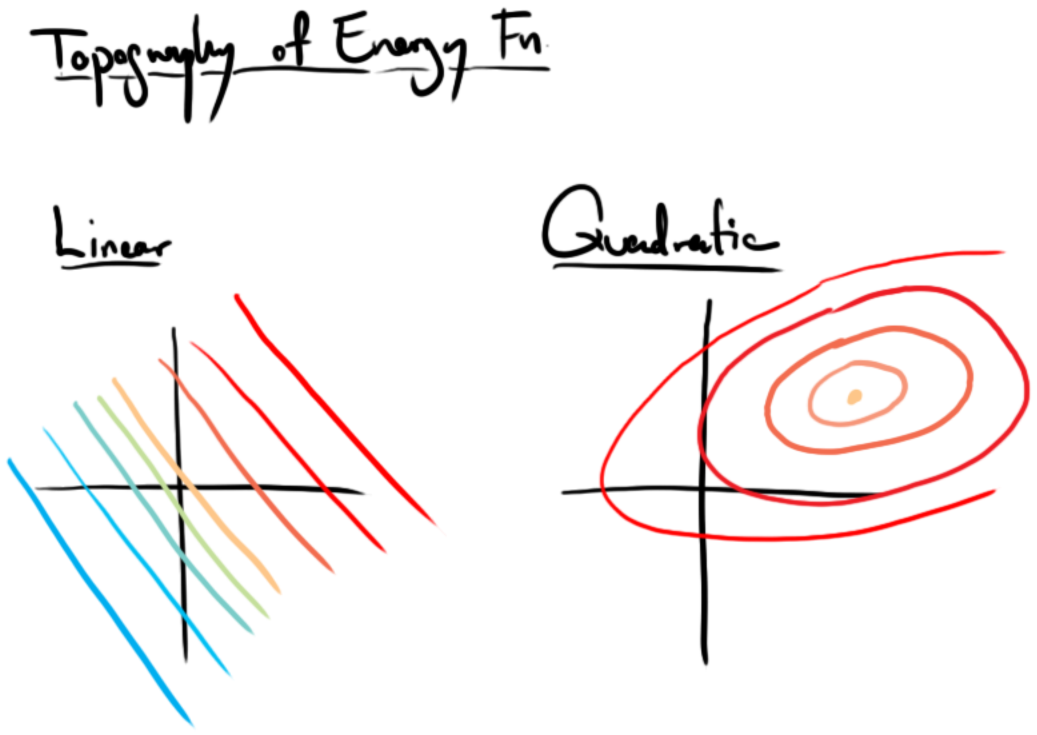
Notice that $p = 1, 2$ is well defined. When p reaches infinity, the big term becomes dominant and eventually ($v \in \mathbb{R}^n$):

$$||\mathbf{v}||_\infty = \max(|v_1|, ..., |v_n|) \tag{3}$$

Now, we can also take a limit toward zero instead of toward infinity. For $p = 0.5$, the curve starts bending inward like in the figure.

3.2 Topography of Energy Functions

We usually deal with two kinds of energy in graphics (let's say we generically write them as a function of 2D space: $E(x, y)$): linear or quadratic. When we plot their contours they look something like this:



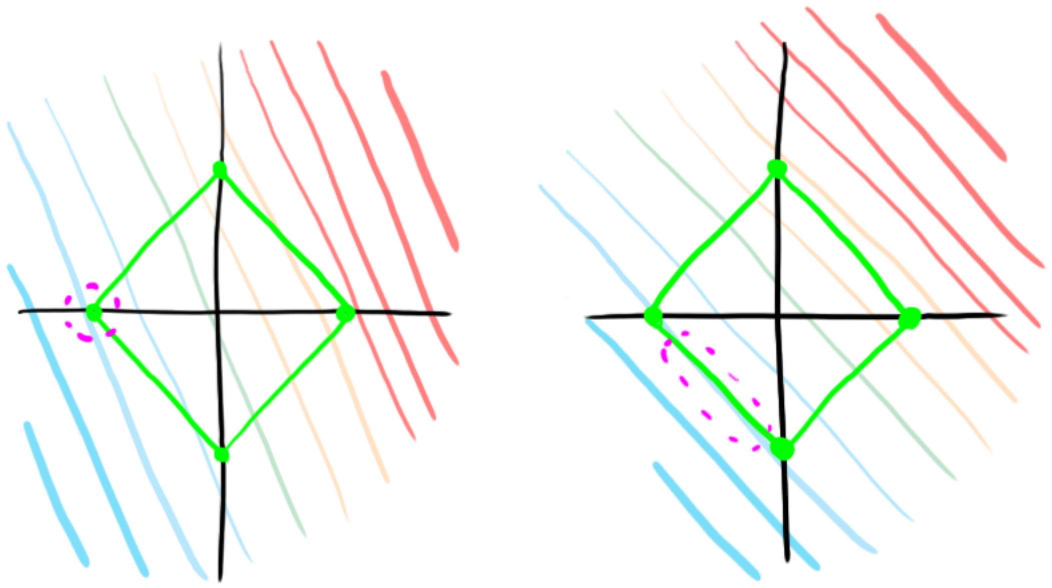
3.3 How to Promote Sparsity in your Solution

We will not try to be mathematical rigorous here but rather to explain why formulating a problem in a certain way can promote sparsity in its solution.

We clarify first what we mean by the solution being "sparse". It means we want a lot of zeros in our optimal solution. In L1 norm, this means the solution should be on a vertex, where most entries are zeros except a few. Take 2D example: a point that satisfies $||v||_1 = 1$ can be sparse (like $(1, 0), (0, 1)$) or not so sparse (like $(0.5, 0.5), (-0.25, 0.75)$). As you might have noticed, the non-sparse solution for L1 constraint lies on an edge, not on a vertex.

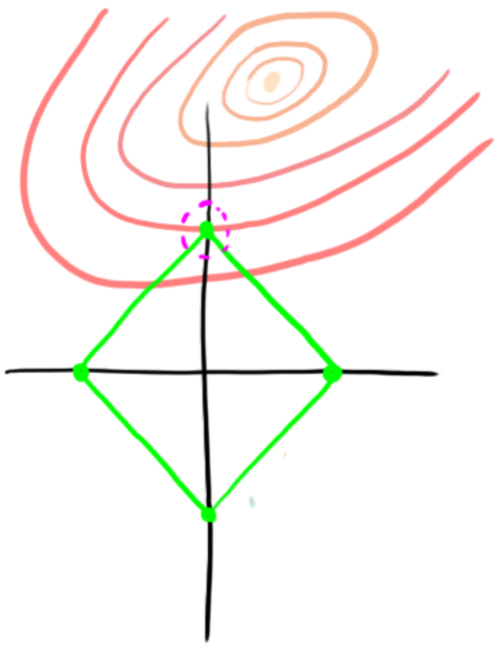
Now, we assert: *for linear and quadratic energy functions*, we can prove that L1 regularization also promotes sparsity in its solution. This is less obvious, so let's draw some figures.

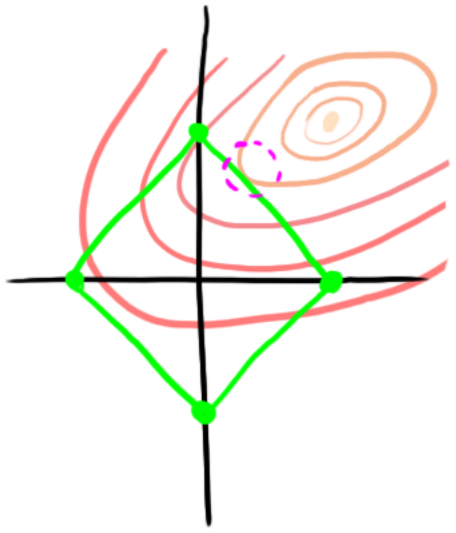
If we have a linear energy function with L1 constraint, when will our optimal solution land on an *edge* of the constraint (and not a vertex)?



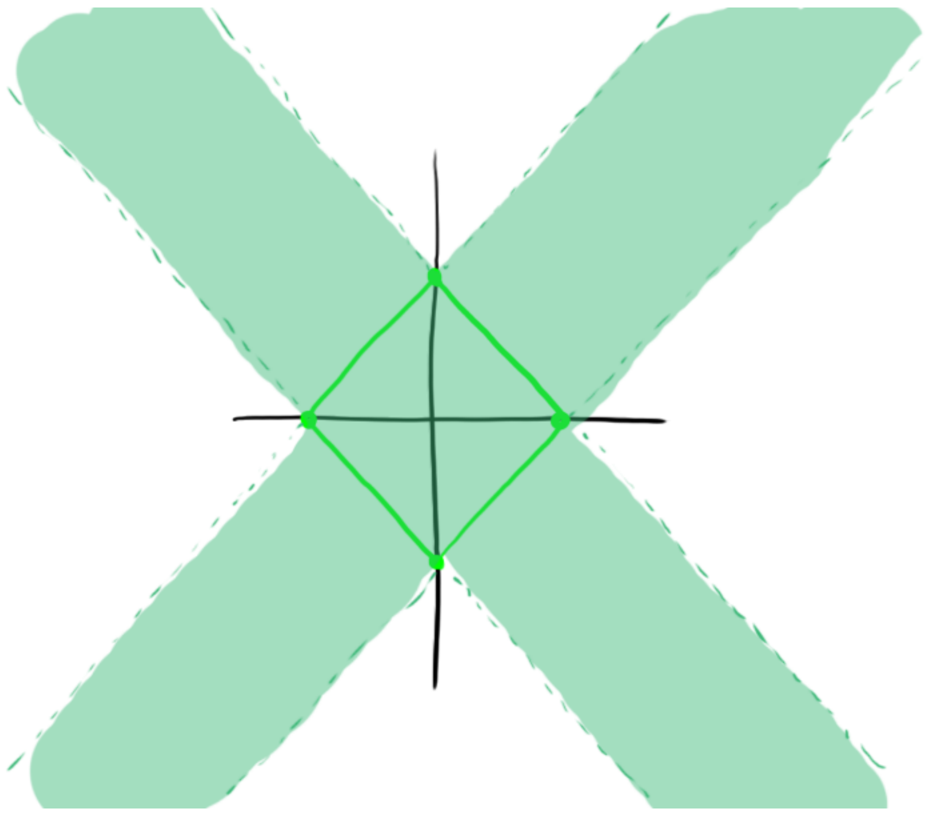
Normally, your optimal solution will land on a vertex like on the left part of the following figure – the solution set is sparse. The special scenario is when the slope of your energy matches one of the slopes on L1 contour exactly, like in the right part of the figure. However, the second case is *very* rare, since we have infinite possibilities for slopes and only 4 of them won't promote sparsity in our solution. That's why linear energy with L1-norm constraint promotes sparsity (remember: in high-dimensional energy minimization, having slope matching that of the constraint is practically rare).

Let's consider the same question in quadratic-energy setting. First of all, how do you find an optimal solution for quadratic energy, given some L1 constraint? It's where one of your energy contours touches the constraint contour.





If we look at the non-sparse case where our optimal solution falls on the edge, it turns out that this can only happen when the minimum of the energy falls into a shaded region shown below.



This seems like a lot, but if you zoom out far enough, these regions are infinitesimal compared to all the places a quadratic minimum can be in. (Especially, in the case where the norm is constraint to 1.0, zooming out to a scale of, say 100, will give you a good sense that these non-sparse regions are practically zero). Therefore, L1 norm constraint for quadratic energy also promotes sparse solutions (ignoring all the very rare cases)!

Back to the paper, in section 5.2, they mentioned that the non-negativity constraint encourages sparse solutions. Why is this? Remember that we have a sum constraint earlier: $\sum_b w_{ib} = 1$. That looks so strangely similar to an L1 norm constraint, which encourages sparse solutions. Actually, it's an L1 norm constraint in disguise! Since we force all the weights to be non-negative, then their values are the same as their absolute values ($w_{ib} = ||w_{ib}||$)!

With their formulation of constraint without an absolute value, it's easier to solve, by the way, by using Quadratic Programming method.