Hi Justin,

One possibility is to make all the edges linear springs (not angular springs). This is is pretty easy: if your vertex in question has vectors $\mathbf{v}_i$ of actual length v_i and rest length L_i to the neighboring vertices, then the force from each vertex is

$$F_i = -k\,(v_i - L_i)\,\hat{\mathbf{v}}_i$$

and the total force is just the sum of the individual forces.

Another possiblity is to use the edge angles you've described. The spring we could assume responds with a torque $N_i = -k\,(\phi_i - \Theta_i)$ where Θ_i is the rest angle. To calculate the angle, one can compute, for example:

$$\begin{aligned} \mathbf{n}_1 \cdot \mathbf{n}_2 &= \cos(\phi) \\ \mathbf{n}_2 \cdot \mathbf{v}_i &= \sin(\psi) \\ \phi &= \operatorname{sgn}(\mathbf{n}_2 \cdot \mathbf{v}_\mathrm{i})\cos^{-1}(\mathbf{n}_1 \cdot \mathbf{n}_2) \end{aligned}$$

where sgn is the sign function, $\mathbf{n}_1$ and $\mathbf{n}_2$ are the normal vectors for the planes containing the vertex and the one across the edge, respectively, and $\mathbf{v}_i$ is a vector from the vertex you're moving to either of the vertices in the edge (which one doesn't matter). The key is to recognize that a vector from the vertex to the edge, forming angle ϕ with the normal of the other plane, is greater or less than $\pi/2$ exactly when it is above or below the plane, which resolves the sign issue.

Or you could be crafty and define the torque as

$$\mathbf{N}_i = k\mathbf{n}_1 \times \mathbf{n}_2$$

and then compute the resulting force on the vertex as

$$\mathbf{F}_i = \mathbf{N}_i \times \frac{\mathbf{v}_i}{v_i^2}$$

You then apply a restoring force $-\mathbf{F}$ computed from the rest angle, so that at the original configuration the system goes nowhere. Finally, since the system is moving and you probably don't want the system to drift sideways based on a mismatch between the initial and updated torque vectors, you can recast the restoring force in terms of the normal vector:

$$-\mathbf{F} = F\mathbf{n}_1 = -(\mathbf{F} \cdot \mathbf{n}_1)\mathbf{n}_1$$

where the term $F = \mathbf{F} \cdot \mathbf{n}_1$ is computed at the original angle, but then is applied in the new $\mathbf{n}_1$ directions throughout the simulation (so that it opposes the force from the torque).

Hopefully this all makes sense without a picture.

—Rex