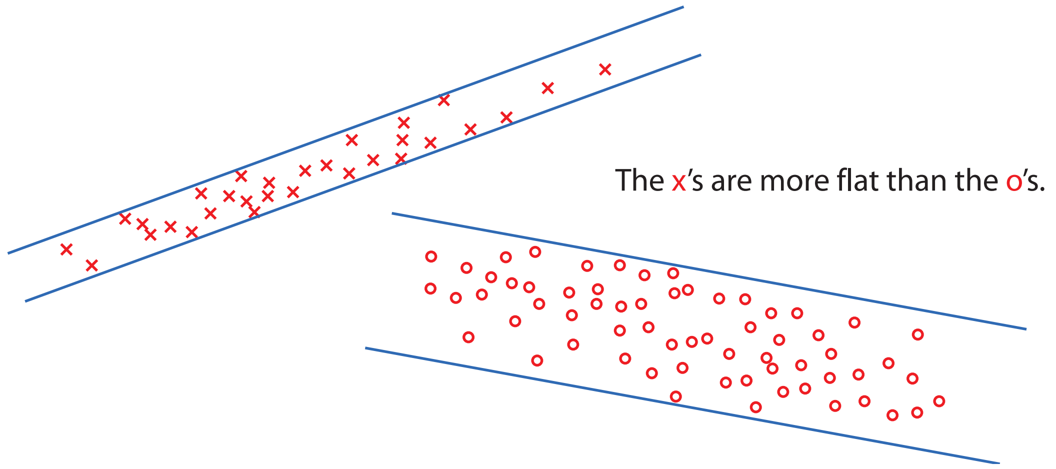
Notes on Flatness

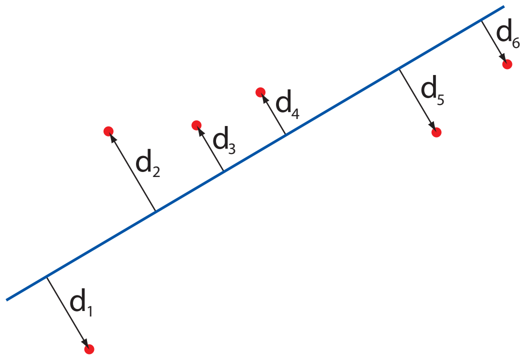
Patch Kessler, November 4, 2018

Definition: The *flatness* of a collection of points is the minimum of the distances between parallel planes that sandwich the points.

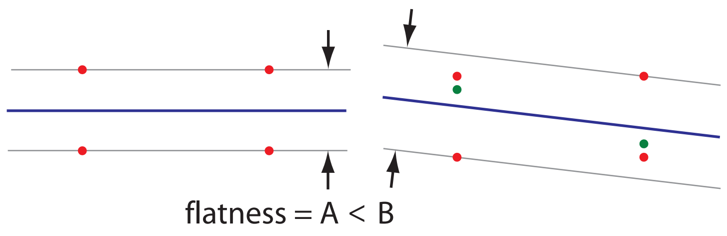


1 Approximate Flatness

Let d_i be the distance from the i^{th} point to a plane.



As the fit between the plane and the points gets better, the quantity $D = d_1^2 + d_2^2 + \cdots + d_n^2$ gets smaller. Usually one *best fit* plane exists for which D is as small as possible. Unfortunately, flatness is only approximately equal to the minimum distance between sandwiching planes parallel to this best fit plane. To see this, consider the four points below to the left, and the six points below to the right.



The best fit plane for the four points to the left is the horizontal blue line. The flatness of these points is equal to A. Adding two points (see the point set to the right) causes the best fit plane to rotate. The flatness of the point set hasn't changed, however the distance between sandwiching planes parallel to the best fit plane is now $B > A$.

Although the best fit plane fails to provide true flatness, it is useful as the start point in algorithms that do.

1.1 Finding the Best Fit Plane

Every plane can be parameterized by a unit normal vector $\mathbf{e}$ and a scalar γ (such that the plane contains the point $\gamma\mathbf{e}$). The sum of the squares of the distances from the points $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ to the plane given by $\mathbf{e}$ and γ is

$$\begin{aligned} D &= (\mathbf{e}\cdot(\mathbf{x}_1-\gamma\mathbf{e}))^2 + (\mathbf{e}\cdot(\mathbf{x}_2-\gamma\mathbf{e}))^2 + \cdots + (\mathbf{e}\cdot(\mathbf{x}_n-\gamma\mathbf{e}))^2, \\ &= \mathbf{e}^T\mathbf{A}\mathbf{e} - 2\gamma\mathbf{e}^T\mathbf{b} + \gamma^2, \end{aligned} \tag{1}$$

where $\mathbf{A} = \mathbf{x}_1\mathbf{x}_1^T + \mathbf{x}_1\mathbf{x}_1^T + \cdots + \mathbf{x}_n\mathbf{x}_n^T$, and $\mathbf{b} = \mathbf{x}_1 + \mathbf{x}_2 + \cdots \mathbf{x}_n$. Translating coordinates so that the centroid ($\frac{1}{n}\sum \mathbf{x}_i$) of the $\mathbf{x}_i$s coincides with the origin causes $\mathbf{b} = \mathbf{0}$, eliminating the middle term from (1). A minimum in $D = \mathbf{e}^T\mathbf{A}\mathbf{e} + \gamma^2$ is attained only if $\gamma = 0$, (i.e., only if the best fit plane passes through the centroid). Finding an optimal $\mathbf{e}$ is slightly harder. Because $\mathbf{A} = \mathbf{A}^T$, the spectral theorem guarantees the existence of an orthonormal basis $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_d\}$ such that $\mathbf{A} = \lambda_1\mathbf{e}_1\mathbf{e}_1^T + \lambda_2\mathbf{e}_2\mathbf{e}_2^T + \cdots + \lambda_d\mathbf{e}_d\mathbf{e}_d^T$. For non pathological data, one of the λ_i's will be smaller than the others. Relabel if necessary so that the smallest λ_i is λ_1. Note that $\mathbf{e} = \alpha_1\mathbf{e}_1 + \alpha_2\mathbf{e}_2 + \cdots + \alpha_d\mathbf{e}_d$ where $\alpha_1^2 + \alpha_2^2 + \cdots \alpha_d^2 = 1$. If $\mathbf{e} \neq \mathbf{e}_1$, then some $\alpha_{i\neq 1} \neq 0$ so that

$$\mathbf{e}^T\mathbf{A}\mathbf{e} = \lambda_1\alpha_1^2 + \lambda_2\alpha_2^2 + \cdots + \lambda_d\alpha_d^2 > \lambda_1 = \mathbf{e}_1^T\mathbf{A}\mathbf{e}_1. \tag{2}$$

It follows that $\mathbf{e}^T\mathbf{A}\mathbf{e}$ is minimized when $\mathbf{e} = \mathbf{e}_1$.

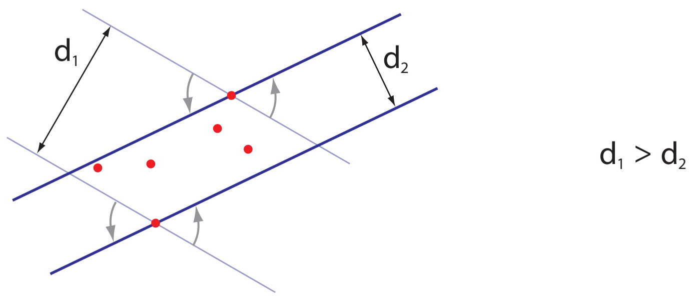
Notice that the dimension of the space of the points is d. This discussion and the following Matlab code work for *any* d. When $d = 2$ we're dealing with points in 2D; the best fit *plane* through these points is a line (as in the figures in this document). When $d = 3$ we're dealing with points in 3D; the best fit *plane* through these points is a plane in the usual sense, (i.e., a flat 2D surface). When $d > 3$, we're dealing with hyper-points and hyper-planes in higher dimensional spaces.

The following Matlab code accepts a $d \times n$ array as input (the n columns of which locate n points in $\mathbb{R}^d$), and returns the best fit plane normal $\mathbf{e}$ as output, along with the corresponding flatness approximation f.

```
function [e,f]=ApproximateFlatness(X)
n=size(X,2);
c=sum(X,2)/n;
A=(X-c*ones(1,n))*X';
[V,D]=eig(A);
[Dmin,I]=min(diag(D));
e=V(:,I);
d=e'*(X-c*ones(1,n));
f=max(d)-min(d);
```

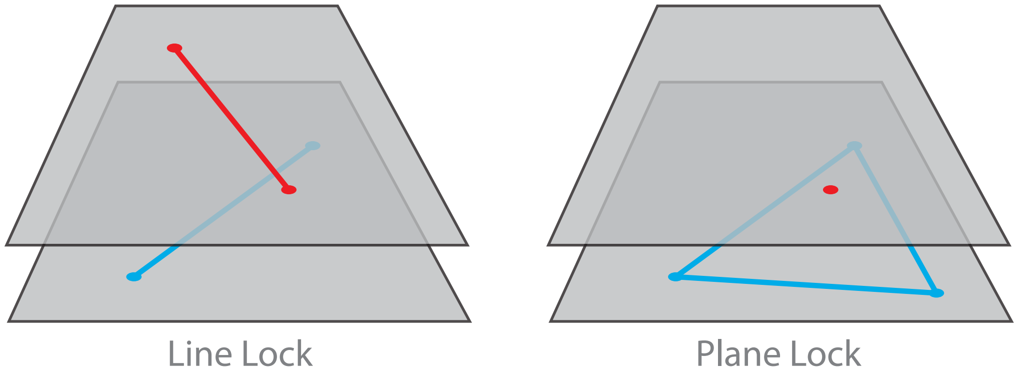
2 Path to True Flatness

If the sandwiching lines of a 2D point set each contain one point, then they can be rotated closer together.



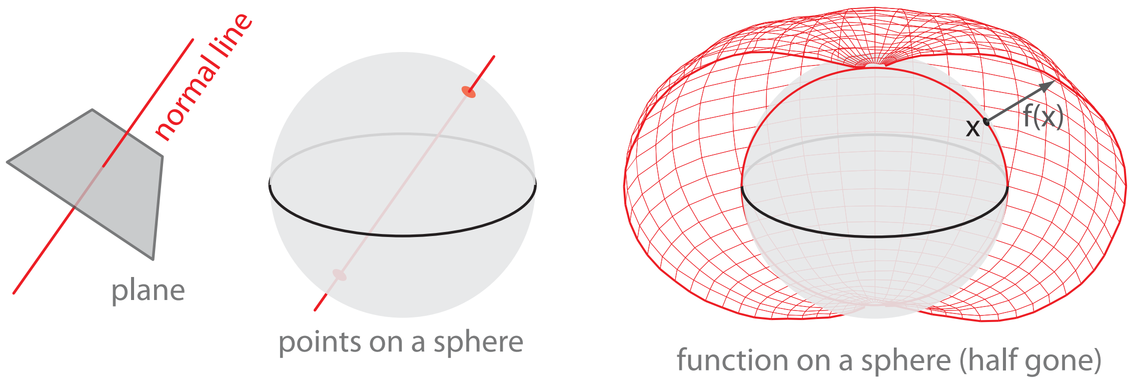
The lines can be rotated up to where one of them intersects a new point. When one line contains two points and the other contains one point, the lines are said to comprise a *lock*, and the distance between them is called a *lock value*.

The situation in 3D is similar, but with a number of new interesting structures and relationships. Instead of locks made by lines, locks are now made by planes. Assuming the point set is in general position[1], every lock is either a *line lock*, with two points on each plane, or a *plane lock*, with one point on one plane and three points on the other.



2.1 The Projective Sphere

The orientation of a plane is characterized by the line normal to the plane that passes through the origin. A line through the origin corresponds to opposite points on a sphere. There is no difference between these points for our purposes, and so we identify them as a single point on what is called the *projective sphere*. Associated with every planar orientation x (i.e., every point on the projective sphere) is the minimum distance $f(x)$ between like oriented planes that sandwich a given point set. We envision $f(x)$ as a surface on the sphere-a balloon being pressed inward by numerous needles.

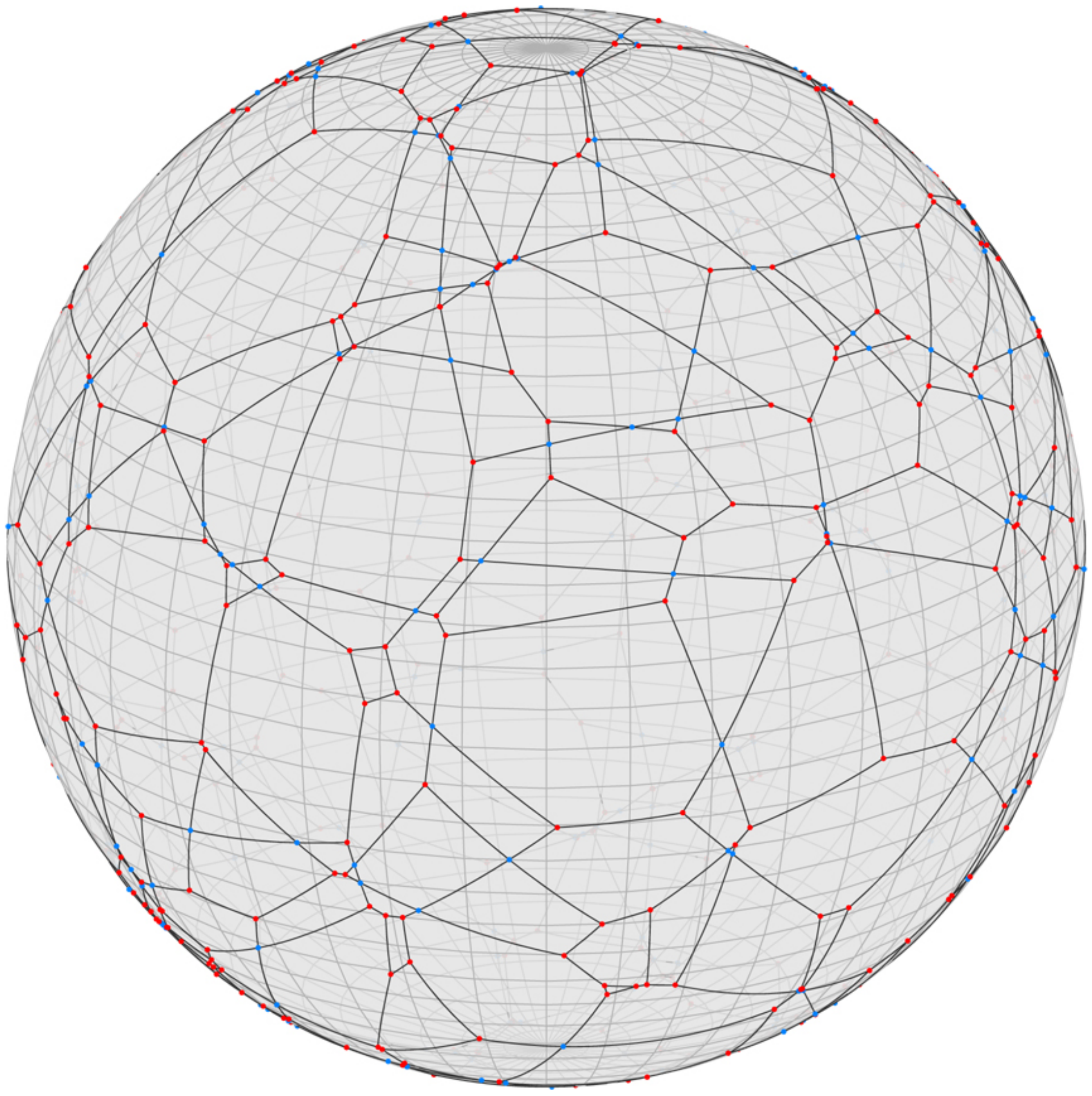


The flatness of the given point set is the minimum value of $f(x)$. Finding flatness by gradient methods is difficult because $f(x)$ is riddled with local minima (the needles pressing the balloon, corresponding to line and plane locks). We turn these minima to our advantage in the following section.

[1]A point set is in *general position* if there is nothing unlikely about how its points are arranged: no three points on the same line, no four points on the same plane, no two lines parallel (each formed by a pair of points), no two planes parallel (each formed by a triple of points), etc. Data from measuring devices often falls on a rectilinear grid, and needs to be perturbed slightly to be in general position.

2.2 Network of Neighboring Locks

Every lock is defined by four points. Two locks are *neighbors* if they have three points in common. Every line lock has four neighbors, and every plane lock has three neighbors. Links between neighbors tie all the locks together in a vast network, like the roads and waterways that tie together cities on the globe.



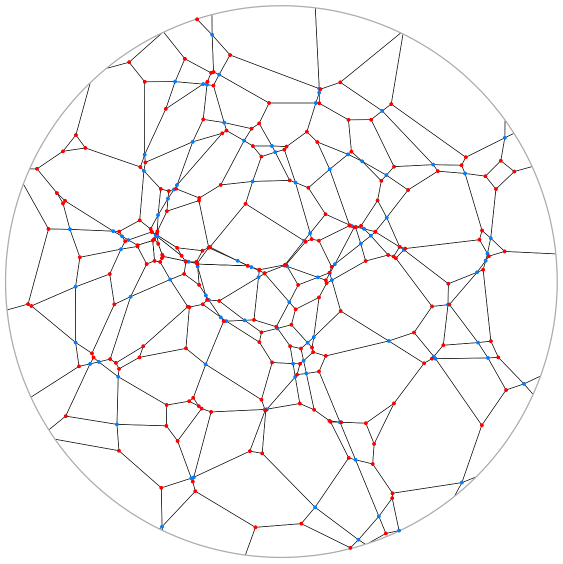
Traversal of the lock network allows for a rapid and efficient evaluation of every single lock value. The minimum of these values is the flatness of the original point set.

The lock network naturally resides on the projective sphere, and so the above graphic on a regular sphere contains redundant data. If you look closely you can see that every node has an opposing twin on the other side of the sphere, and that all the details of the network around a node also exist inverted on the other side. These opposing inversions eventually meet up if you consider enough of the network around any given node.

Notice that there seem to be two *separate* networks, the connecting lines of which cross at the line locks (blue dots). Call one of these A and the other B. If you trace A around to the opposite side of the sphere, you will find that it has continuity with the inversion of B! In fact, A and B are inversions of each other. This redundancy can be eliminated by orienting the plane locks, causing the network to be defined on the sphere, rather than the projective sphere. Doing this causes each line lock (blue dot) to have two neighbors rather than four.

2.3 Flattening the Network

For practical visualizations, we stereographically project half of the previous sphere onto the plane. This results in a planar graphic that no longer contains two versions of every node. Nodes are connected by straight lines rather than great circles. Here's a projection of the graph from the previous page.



2.4 Convex Hull

A point set can have many more elements than its convex hull, however a point set and its convex hull have the same flatness. Throwing out points not in the convex hull can therefore reduce the work needed to find flatness. We caution that discarding points can also alter the balance of a distribution, and therefore the orientation of the least squares plane through the data (which is usually a good starting point for finding true flatness).

Each plane lock corresponds to a facet of the convex hull. Thus our method generates the convex hull as a byproduct. (Not as efficiently as other methods for the convex hull.)

2.5 It's Been Done Before...

I am grateful to Jonathan Shewchuk (`http://www.cs.berkeley.edu/~jrs/`) for looking at the above work and giving me several helpful comments about it. It turns out that in computational geometry, flatness is called *the width of a point set*. Many people have worked on this problem, and unfortunately, one pair of workers published an algorithm in 2001 which essentially solves it the same way I do.

> Bernd Gärtner, Thomas Herrmann, **Computing the Width of a Point Set in 3-Space**,
> In Abstracts for the Thirteenth Canadian Conference on Computational Geometry, pages 101-103.
> University of Waterloo, August 13-15 2001.

Their paper as well as a nice set of slides describing their method can be found on Herrmann's website: `http://www.ifor.math.ethz.ch/staff/herrmann/`. Also, their algorithm is implemented in CGAL, the open source Computational Geometry Algorithms Library: `http://www.cgal.org`.

If I had seen Herrmann's work before I was done with mine, I would have had a solution, but none of the excitement of discovery. I'm happy with the time I've spent working through everything- ten years from now, hopefully someone else will feel the same way after they rediscover it all again!