DATUM PLANES BASED ON A CONSTRAINED L_1 NORM

Craig M Shakarji
Member, ASME
Physical Measurement Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
craig.shakarji@nist.gov

Vijay Srinivasan
Fellow, ASME
Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
vijay.srinivasan@nist.gov

ABSTRACT

This paper has two major goals. First, we present an algorithm for establishing planar datums suitable for a default in tolerancing standards. The algorithm is based on a constrained minimization search based on the L_1 (L1) norm after forming a convex surface from the original surface or sampled points. We prove that the problem reduces to a simple minimization search between the convex surface and its centroid. The data points in the discrete case do not need to have any corresponding weights provided with them, as appropriate weighting is part of the algorithm itself, thereby making the algorithm largely insensitive to nonuniformly sampled data points. Terse Mathematica code is included for the reader. The code is sufficient for primary and secondary planar datum fitting as well as a 3-2-1 datum reference frame generation. The second goal of this paper is to compare this new method with several other possible means for establishing datum planes, ultimately showing several appealing characteristics of the proposed algorithm. Since both the ISO and ASME standardization efforts are actively working to establish datum plane definitions, the timing of such a study is opportune.

1. BACKGROUND AND INTRODUCTION

In the world of Geometric Dimensioning and Tolerancing (GD&T), datums are used extensively to locate and orient tolerance zones [1-7]. Datum planes in particular are common and are established by mating planes to imperfect datum features on parts during inspection [3] (see Fig. 1). Distances and orientations on drawings and three-dimensional models are established from these datum planes, relative to which tolerance zones are located and oriented. In many cases there is a need for more than one datum plane. In fact a full Cartesian coordinate system in three dimensions is often established using datums. Datum planes, in particular, are widely used for this. The importance and prevalence of datum planes in specifications are given in greater detail in [8] and will not be revisited in this paper.

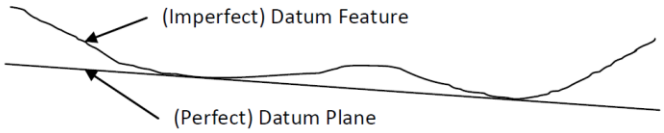

Fig. 1. Deriving a datum plane from a datum feature.

Given that datum planes are ubiquitous, it might be surprising that—short of standardization—there are several different yet reasonable approaches by which a datum plane can be established from a datum feature [9]. Furthermore, the International Organization for Standardization (ISO) and the American Society of Mechanical Engineers (ASME) are actively working to establish default datum plane definitions. Consequently, the timing of this paper is opportune, since we seek in its two major sections to provide (1) an improved algorithm for establishing planar datums (Section 2 of this paper), and (2) a comparison of the proposed algorithm with several other possible definitions for establishing datum planes (Section 3 of this paper).

2. THE IMPROVED ALGORITHM

2.1 Existing L_1 Datum Plane Definition and Algorithm

First, we describe what is meant by a constrained L_1 fit in our context. To fit a one-sided L_1 plane to a surface patch in space, we pose the following optimization problem (with reference to Fig. 2): Given a bounded surface S, and a direction $\boldsymbol{a}^*$ (that points into the material), find the plane P that minimizes $\int_S |d(\boldsymbol{p}, P)| ds$, subject to the constraint that P lies entirely to one side (as determined by $\boldsymbol{a}^*$) of the surface S.

Here $d(\boldsymbol{p}, P)$ denotes the signed perpendicular (to P) distance of a point $\boldsymbol{p}$ on surface patch S from the plane P that will be fitted. We note that $\int_S ds$ is the area of the surface patch. If the surface consists of several patches, then the integrals can be evaluated over each patch and then summed.

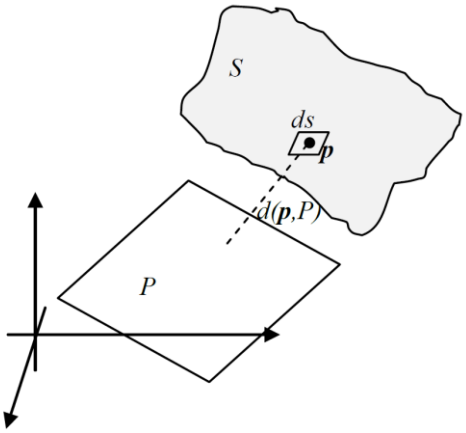

Fig. 2. Fitting a plane to a surface patch.

The objective function cannot, in general, be evaluated in closed form. So we resort to numerical integration over the surface *S*. We can sample points on a surface patch after dividing up the patch into discrete areas ΔA_i and approximate the objective function as

$$\int_S |d(\boldsymbol{p}, P)| ds \approx \sum_{i=1}^{N} |d(\boldsymbol{p}_i, P)| \cdot \Delta A_i ,$$

where $\boldsymbol{p}_i$ are the N sampled points, one in each subdivision. Thus we are led to minimizing $\sum_{i=1}^{N}[|d(\boldsymbol{p}_i, P)| \cdot \Delta A_i]$ over the parameters of the plane P, where ΔA_i's are treated as the weights.

In an earlier paper [8], we presented the theory and algorithms for datum plane establishment using a constrained minimization search based on the L_1 norm (as just defined). In short, the algorithm worked as follows: Given a surface (or set of sampled points), the datum plane was defined as the plane that (1) is constrained to lie on the nonmaterial side of the surface (or points), and (2) minimizes the integral (or sum) of absolute distances between the plane and the surface (or points). We showed that finding such a plane actually turns out to be quite simple, since we proved that it is equivalent to finding the plane that minimizes the distance between the centroid of the surface (or of the weighted points) and the plane. This simplification led to efficient algorithms (and code provided) for the primary and secondary planar datums (the tertiary case being trivial). The reader is encouraged to fill in details as desired from the earlier paper itself.

2.2 Motivation for an Improved Definition and Algorithm

Although the original planar datum definition based on the L_1 norm has many attractive properties (which will be discussed in Section 3) it has the following three drawbacks that are remedied in the improvement to be given in the next subsection of this paper:

1) The need for weights: In the discrete case, having a set of sampled points on the datum feature was not enough to compute the datum plane. The appropriate weights corresponding to the points were needed as well. (Without weighting, the computed centroid could shift with nonuniform sampling.) If one imagined that the weights (being the relative areas around each sampled point) could be calculated as part of the algorithm itself, the nominal model of the datum feature would still be required. To see why this is true, imagine a datum feature that is a rectangular surface with a bore and/or slot (see Fig. 3). To compute weights, one would have to know where the hole or slot appears in the rectangle to keep from overweighting certain points. If one tried to remedy this by always using equally spaced points, problems then arise as to how to actually obtain equal spacing when the datum feature is irregularly shaped or includes bores.

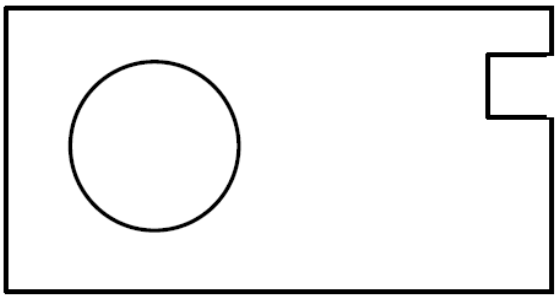

Fig. 3. An example case to show the need for part information when assigning weights.

2) Irrelevant part features: It could be argued that the bore and the slot in Fig. 3 should not affect the datum plane. The improved definition presented later in this paper solves this.

3) The applicability to broken surfaces: In a case like the one illustrated in Fig. 4, the intent seems to be that all three sections of the broken surface should contact the datum plane. If (as illustrated) one surface section were relatively large, then the existing L_1 based datum definition may very well not contact the other two, small surface sections. Again, the improved definition presented later in this paper solves this.

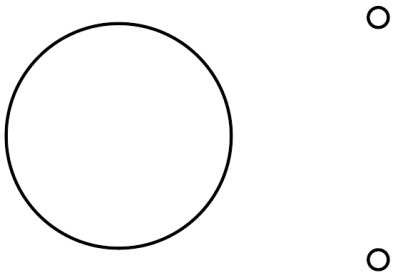

Fig. 4. An example datum feature comprised of three disjoint surfaces.

2.3 The Improved L_1 Definition and Algorithm

The new datum plane definition is again based on a constrained L_1 minimization search, but now consists of two major steps: (1) Find the appropriate convex surface from the given surface or data points, and (2) Compute the constrained (one-sided) L_1 planar fit to that convex surface. The previous algorithm did not compute the convex surface first, but applied the L_1 minimization directly to the original surface. Computing the convex surface first resolves all three of the issues presented in Section 2.2.

The first step of the definition, more specifically, is to compute the convex surface that is the part of the convex hull exterior to the material. If a part is positioned so that the material is "up" then the convex surface sought would be the "underbelly" or the "lower convex envelope" of the convex hull. Figure 5 illustrates the concept in the case of discrete points.

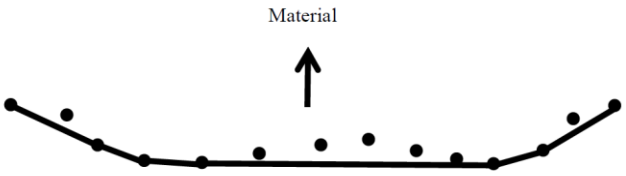

Fig. 5. An example of the lower convex envelope derived from the data points (shown in 2D for simplicity).

This envelope can also be thought of as the result of applying a morphological filter to the data, using a sphere of infinite radius (see [10]). The new surface reflects all the possible contacts between the original surface (or set of points) and a perfect plane. The development of such a surface is also documented in ISO 5459 [6], though that document then goes on to employ a constrained L_∞ fit. Furthermore, convex hull algorithms that can facilitate the computation of such a surface are mature, well documented, and widely available in commercial software.

Having thus created the convex surface, the second major step in the algorithm is to compute the constrained (one-sided) L_1 planar fit to that new, convex surface. When the convex surface is generated from a set of discrete points, it can be represented by a union of triangles (as is generally done in widely available convex hull algorithms). Assuming such a convex hull algorithm is available, we can apply the theorems in [8] to now document the algorithm, which is robust, easy to conceptualize, and simple to code. The 3D datum plane algorithm is:

Given:

1) Data points $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_N$, where each $\boldsymbol{x}_i = (x_i, y_i, z_i)$, and
2) A direction, $\boldsymbol{a}^*$ that indicates the direction into the material,

then the datum plane is established using the following steps:

1) Compute the convex hull of the data points and represent it by the union of triangles. (Each triangle will have three of the data points for its vertices.)
2) Select those triangles that are exterior to the material (i.e., the triangles that comprise the lower convex envelope). This can be accomplished by computing the normal to each triangle (pointing into the hull) and comparing its direction to $\boldsymbol{a}^*$. (The sign of the dot product can easily be used here).
3) Compute the centroid, $\overline{\boldsymbol{x}}$, of the convex surface of Step 2. The centroid of each triangular region can be trivially computed as the average of the vertices. The area of each triangle can also be easily computed. The vector sum of the triangle centroids when weighted by their relative areas is the centroid of the lower convex envelope. Given N triangles each having area A_i, then each relative weight is $w_i = A_i / \sum_{i=1}^{N} A_i$.
4) Evaluate the distance from the centroid of Step 3 with each plane containing a triangle of the surface of Step 2.
5) Choose the plane that produces the minimal distance. The solution plane is coincident with a face of the convex hull, thus a simple search through the faces will produce the correct datum plane.

A word should be said about the reference direction, $\boldsymbol{a}^*$. The direction usually does not need to be known very accurately, since it is only used to distinguish the top side of the convex surface from the bottom. So determining $\boldsymbol{a}^*$ from a probing direction or from a nominal vector is often sufficient. But for completeness, we define $\boldsymbol{a}^*$ as the direction normal to the least-squares plane of the original surface and pointing into (as opposed to outside) the material. See [11] for an appropriate least-squares algorithm, if needed.

A secondary datum plane can be generated by the following similar steps:

1) Project the data points from the secondary datum feature into the primary datum plane.
2) Compute the 2D convex hull of the projected data points and represent it by the union of line segments. (Each line segment will have two of the projected data points for its endpoints.)
3) Select those line segments that are exterior to the material (i.e., the line segments that comprise the lower convex envelope). This can be accomplished by computing the normal to each line segment (pointing into the hull) and comparing its direction to $\mathbf{a}^*$. (The sign of the dot product can easily be used here). In this

step, $\mathbf{a}^*$, is a vector in the primary datum plane that indicates the direction into the material. As explained above, $\mathbf{a}^*$ usually does not need to be known very accurately, but for completeness, we define $\mathbf{a}^*$ as the direction normal to the least-squares line of the contour obtained by projecting the secondary datum feature after that surface has been projected into the primary datum plane (and the normal pointing into as opposed to outside the material).

4) Compute the centroid, $\bar{\boldsymbol{x}}$, of the convex (piecewise linear) contour of Step 3. The centroid (midpoint) of each line segment can be trivially computed as the average of the endpoints. The length of each line segment can also be easily computed. The sum of the line segment centroids when weighted by their relative lengths is the centroid of the lower convex envelope. Given N line segments each having length Δ_i, then each relative weight is $w_i = \Delta_i / \sum_{i=1}^{N} \Delta_i$.
5) Evaluate the distance from the centroid of Step 4 with each line containing a line segment of the surface of Step 3.
6) Choose the line that produces the minimal distance. The solution plane is the plane containing this line and perpendicular to the primary datum plane.

Mathematica codes for the 3D and 2D cases (without the projection step) are given later and are pleasingly compact and efficient. The case of the tertiary datum plane is trivial.

Two theorems are relevant to the 3D algorithm (the 2D case is similar to the 3D case). The first was easily proved in [8] namely,

Theorem 1. *Assume that we are given a direction $\boldsymbol{a}^*$ and a bounded surface S of finite area comprised of points with $(\hat{x}, \hat{y}, \hat{z})$ coordinates and having centroid $\bar{\boldsymbol{x}} = \left(\frac{\int_S \hat{x}\, ds}{\int_S ds}, \frac{\int_S \hat{y}\, ds}{\int_S ds}, \frac{\int_S \hat{z}\, ds}{\int_S ds}\right)$. Then a plane that lies to one side of S (as determined by $\boldsymbol{a}^*$) and that minimizes $|d(\bar{\boldsymbol{x}}, P)|$ is also a constrained plane that minimizes $\int_S |d(\boldsymbol{p}, P)| ds$.*

Proof: A constrained plane that minimizes $|d(\bar{\boldsymbol{x}}, P)|$ is a plane, P^*, determined by a point on the plane $\boldsymbol{x} = (x, y, z)$ and the direction of the plane $\boldsymbol{a} = (a, b, c)$ such that $|\boldsymbol{a} \cdot (\bar{\boldsymbol{x}} - \boldsymbol{x})|$ is minimized. But by definition $\bar{\boldsymbol{x}} = \left(\frac{\int_S \hat{x}\, ds}{\int_S ds}, \frac{\int_S \hat{y}\, ds}{\int_S ds}, \frac{\int_S \hat{z}\, ds}{\int_S ds}\right)$, so that P^* minimizes $\left| a\left(\frac{\int_S \hat{x}\, ds}{\int_S ds} - x\right) + b\left(\frac{\int_S \hat{y}\, ds}{\int_S ds} - y\right) + c\left(\frac{\int_S \hat{z}\, ds}{\int_S ds} - z\right)\right|$. Multiplying by the total area, $\int_S ds$, which is positive, we get that P^* must also minimize $\left| a \int_S (\hat{x} - x)\, ds + b \int_S (\hat{y} - y)\, ds + c \int_S (\hat{z} - z)\, ds \right|$, which is $\int_S |d(\boldsymbol{p}, P)| ds$, completing the proof. ■

The second relevant theorem, which is similar to Theorem 3 of [8] is:

Theorem 2. *Given a convex envelope consisting of a union of a finite number of triangles, and direction $\boldsymbol{a}^*$, then any supporting plane that lies to one side of the points (as determined by $\boldsymbol{a}^*$) and minimizing the weighted L_1 objective, is a plane that is coincident with one of the planar (triangular) faces of the convex envelope.*

Proof: The proof of this theorem involves several steps and relies on the well-known fact that the weighted centroid $\bar{\boldsymbol{x}}$ must lie within the convex hull, and thus the material side of the convex envelope (since any weighted combination of the points is contained in the convex hull provided every weight lies in (0, 1] and provided the weights sum to one (see, for instance, [12, 13]). The theorem and proof share some similar approaches with the task of computing the width of a set of data points [14].

Step 1: A minimizing plane contains at least one of the given points that is on the convex surface. Suppose not. That is, suppose a plane, P, lying to the one side of the surface, minimizes the objective, but does not contact any of the surface points. It is immediately clear then that it does not contact any of the vertices of the triangles. Since there are a finite number of triangles, there must be a positive distance d that is the minimum distance from the plane to any of the vertices of the triangles. If $\boldsymbol{x}^*$ is the (or a) closest vertex, then a new plane, P^*, could be constructed that is parallel to P but at a distance $d/2$ from $\boldsymbol{x}^*$. This plane P^* also lies to the one side of the points (as determined by $\mathbf{a}^*$) but has a smaller weighted L_1 objective, because it is closer to the weighted centroid, $\bar{\boldsymbol{x}}$, since $\bar{\boldsymbol{x}}$ must also lie to the same side of P and P^* (since it is contained in the convex hull). This contradicts the supposition that P minimized the objective function, proving Step 1.

Step 2: A minimizing plane contains one of the edges of the convex envelope boundary (implying that it contains at least two of the given points). Suppose not. That is, suppose a plane, P, lying to one side of the points, minimizes the objective function, but does not contain any of the edges of the triangles that make up the convex surface. From Step 1, P contacts one of the vertices. There are a finite number of edges of triangles emanating from that point. Let $\theta > 0$ denote the minimum angle between P and the edges emanating from the contact point. Construct a line from the contact point to $\bar{\boldsymbol{x}}$. A new plane, P^*, can be constructed by rotating P by an angle $\theta/2$ in a direction making it closer to $\bar{\boldsymbol{x}}$, meaning it has a smaller weighted objective. (The rotation should be about the direction that is the cross product between the normal of P and the direction formed by the contact point and $\bar{\boldsymbol{x}}$. In the case that the two directions are coincident, a rotation of $\theta/2$ about any direction would suffice.) This rotation decreases the objective function by a factor of $1 - \cos\left(\frac{\theta}{2}\right)$. But this contradicts the supposition that P minimized the objective function, proving Step 2.

Step 3: A minimizing plane coincides with one of the triangular faces of the convex surface. Suppose not. That is, suppose a plane, P, lying to one side of the points, minimizes the objective function, but does not contain any triangular face of the convex surface. From Step 2, P contains one of the edges of a triangle. There are generally two triangular faces emanating from that edge. (The case where there is one face—meaning P contains an edge of a boundary triangle of the edge of the convex envelope—can be handled with similar reasoning.) Let $\theta > 0$ denote the minimum angle between P and those two faces. A new plane, P^*, can be constructed by rotating about the direction of the shared edge by an angle $\theta/2$ that is closer to $\bar{\boldsymbol{x}}$, meaning it has a smaller weighted objective. (Depending on the location of $\bar{\boldsymbol{x}}$, rotating one way or the other—possibly both—will decrease the distance from the plane to $\bar{\boldsymbol{x}}$). The minimal decrease by the rotation changes the objective function by a factor of $1 - \cos\left(\frac{\theta}{2}\right)$. This contradicts the supposition that P minimized the objective function, proving Step 3 and, in fact, the theorem. ■

Theorem 2 allows the algorithm to simply compute the centroid of the convex surface and search through all the triangular faces of the convex surface for the one closest to the centroid.

2.4 Mathematica Codes

Mathematica[1] (version 8) code for the cases of fitting of a plane to data in three dimensions is presented first. The function name should be understood as `l1PlaneDatum3d` = "ℓ_1 Planar Datum in three dimensions." In this code, `pts` contains the list of all the three-coordinate points and the reference direction is indicated by `refdir` (a three-dimensional vector). The function returns a list of six numbers. The first three define a point on the plane. The next three define the direction normal to the plane. If one thinks of `refdir` as defining "up," then the plane returned will be "under" the data points. The code makes use of a function called `computetrianglearea`, which computes the area of a triangle in a special way to reduce the impact of finite precision. Code for that function is given in Appendix A of this paper. A check can be added for the case that the points form a perfect plane. This check (which is not needed for the 2D case) can be performed by simply looking at the residuals from a least-squares plane fit as described in [11]. If the residuals are essentially zero, then the least-squares plane is the datum plane.

```
Needs["TetGenLink`"];
l1PlaneDatum3d[pts_, refdir_] := Module[{hullpts,
    hullsurface, surface, trianglecentroids,
    trianglareas, centroid, normals, distances,
    mindistanceindex},{hullpts, hullsurface} =
    TetGenConvexHull[pts];
surface = Select[hullsurface, Cross[hullpts[[#[[2]]]]
    - hullpts[[#[[1]]]], hullpts[[#[[3]]]]
    - hullpts[[#[[1]]]]].refdir > 0 &];
trianglecentroids =
    Table[Sum[hullpts[[surface[[i]]]][[j]]/3,{j, 3}],
    {i,Length[surface]}];
triangleareas =
    Table[computetrianglearea[hullpts[[surface[[i]]]]
    ], {i,Length[surface]}];
centroid =
    triangleareas.trianglecentroids/Total[triangleare
    as];
normals = Table[Cross[hullpts[[surface[[i]][[2]]]] -
    hullpts[[surface[[i]][[1]]]],hullpts[[surface[[i]
    ][[3]]]] - hullpts[[surface[[i]][[1]]]],{i,
    Length[surface]}];
normals = Table[normals[[i]]/Norm[normals[[i]]],{i,
    Length[normals]}];
distances = Table[normals[[i]].(centroid -
    trianglecentroids[[i]]), {i, Length[normals]}];
mindistanceindex = Ordering[distances, 1];
Flatten[{trianglecentroids[[mindistanceindex]],
    normals[[mindistanceindex]]}] ];
```

Mathematica (version 8) code for the cases of fitting of a plane to data in two dimensions is presented next. The function name should be understood as `l1PlaneDatum2d` = "ℓ_1 Planar Datum in two dimensions." It should be understood that this is a case of a constrained planar datum, which, when constrained to two dimensions is actually a line. In this code, `pts` contains the list of all the two-coordinate points and the reference direction is indicated by `refdir` (a vector of dimension two). The function returns a list of four numbers. The first two define a point on the line. The next two define the direction normal to the line. If one thinks of `refdir` as defining "up," then the line returned will be "under" the data points.

```
Needs["ComputationalGeometry`"];
l1PlaneDatum2d[pts_, refdir_] := Module[{indices,
    newpts, midpoints, vectors, alldata, lowerdata,
    lengths, centroid, normals, distances,
    mindistanceindex},
indices = ConvexHull[pts];
newpts = pts[[indices]];
newpts = Append[newpts, newpts[[1]]];
midpoints = Table[(newpts[[i + 1]] + newpts[[i]])/2,
    {i, Length[newpts] - 1}];
vectors = Table[newpts[[i + 1]] - newpts[[i]],{i,
    Length[newpts] - 1}];
newpts = Drop[newpts, -1];
normals = Table[{-vectors[[i]][[2]],
    vectors[[i]][[1]]}, {i, Length[vectors]}];
normals = Table[normals[[i]]/Norm[normals[[i]]],{i,
    Length[normals]}];
alldata = Transpose[{vectors, midpoints, normals}];
lowerdata = Select[alldata, #[[3]].refdir > 0 &];
{vectors, midpoints, normals} = Transpose[lowerdata];
lengths = Table[Norm[vectors[[i]]],{i,
    Length[vectors]}];
centroid = lengths.midpoints/Total[lengths];
distances = Table[normals[[i]].(centroid-
    midpoints[[i]]), {i, Length[normals]}];
mindistanceindex = Ordering[distances, 1];
```

[1] Certain commercial software packages are identified in this paper in order to specify the experimental procedures and code adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the software tools identified are necessarily the best available for the purpose.

```
Flatten[{midpoints[[mindistanceindex[[1]]]],
    normals[[mindistanceindex[[1]]]]}];
```

3. A COMPARISON OF SEVERAL POSSIBLE DATUM PLANE DEFINITIONS

For the sake of a reasonable scope, we will simply consider defining a primary datum plane to a planar feature. Indeed, secondary and tertiary datum planes are important as well, but the reader can extend the implications of this work to the additional cases without much difficulty. We consider the following eight possible means to define a primary datum plane given a planar datum feature—all of which have been proposed at some level in standards development. (Abbreviated names are given to each for convenient use throughout this paper.)

3.1 The Planar Datum Definitions

1) L_1 *constrained convex envelope*: This is the definition presented in Section 2 of this paper.

This mathematical definition actually has an easy-to-conceptualize physical understanding. Minimizing the L_1 norm between a plane and a surface is equivalent to simply setting the surface on a level plane and allowing gravity to bring the two into contact. In this definition, such contact is made—not with the original surface—but with the convex envelope about the surface.

2) L_1 *constrained*: Find the plane that lies to one side (the nonmaterial side) of the surface that minimizes the L_1 norm between the plane and the datum feature.

This is similar to (1) above, but does not create the convex envelope first. This is the method described in [8].

3) L_2 *unconstrained*: Find the plane fit to the planar feature in a least-squares sense (see [11]).

4) L_2 *shifted*: Find the plane as in (3) above, but translate the plane such that it contacts the datum feature but does not pass through any material. (In other words, shift to the "high point" of the surface—or, as they are pictured here—the "low point" of the surface.)

5) L_2 *constrained*: Find the plane that is constrained to lie on the nonmaterial side of the datum feature and that minimizes the L_2 norm (least-squares) of their separation.

6) L_2 *constrained convex envelope*: Find the plane that is constrained to lie on the nonmaterial side of the datum feature and that minimizes L_2 norm (least-squares) of the separation between the plane and the convex envelope of the datum feature.

7) L_∞ *constrained*: Find the plane that is constrained to lie on the nonmaterial side of the datum feature and that minimizes the maximal separation between the plane and the datum feature.

8) L_∞ *constrained convex envelope*: Find the plane that is constrained to lie on the nonmaterial side of the datum feature and that minimizes the maximal separation between the plane and the convex envelope of the datum feature.

3.2 Criteria for Comparison

While there are an infinite number of datum features to which a datum plane could be mated, we found that using only three example cases (actually shown and performed in 2D), we could largely distinguish the behavior of various definitions.

Using these, combined with other desirable properties, we can evaluate the various planar datum definitions. A benefit of these criteria is that it is a framework for the reader to use for evaluation. While we give our evaluations of "good," "fair," and "poor," the reader is free to alter any evaluations.

3.2.1 Asymmetric convex datum feature

When the datum feature is a convex shape as shown in Fig. 6, all the definitions yield a similar (negative) slope except the L_∞ based definitions, which are horizontal. This is especially troubling when one considers that the same undesired effect of the L_∞ based definition would occur if the datum feature were a flat plane with just a corner bent up. (The bent corner—ever so small—would nonetheless have significant impact on the slope of the mating datum plane in the L_∞ cases.) Thus a "poor" evaluation is assigned to those definitions in this case. Also, the L_2 *unconstrained* datum passes through the material (not shown in the figure), which should be assigned a "poor" evaluation, since planar datums are often associated with assembly that cannot have interference of material. The same issue of interference will result in a "poor" evaluation for the L_2 *unconstrained* definition in the next two cases as well. See Table 1.

In the figures for the test cases, the desired datum plane is shown by a dotted line (and is in fact the L_1-based fit in each of the three cases shown). Behavior of other datum plane definitions is depicted in dashed line(s). A few (not all) other (i.e., dashed) lines are shown in each figure for clarity.

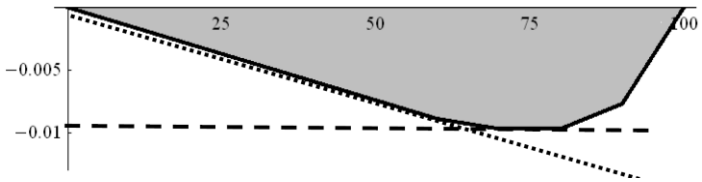

Fig. 6. Example case showing the L_∞ based definitions yielding an undesired slope of zero.

Table 1. Asymmetric convex case evaluation

Datum Definition	Evaluation
L_1 constrained convex envelope	Good
L_1 constrained	Good
L_2 unconstrained	Poor
L_2 shifted	Good
L_2 constrained	Good
L_2 constrained convex envelope	Good
L_∞ constrained	Poor
L_∞ constrained convex envelope	Poor

3.2.2 Asymmetric concave datum feature

As seen in Table 2, when the datum feature is a convex shape as shown in Fig. 7, all the definitions yield an expected horizontal plane except the L_2 *unconstrained* and L_2 *shifted* definitions, which yield an undesired negative slope.

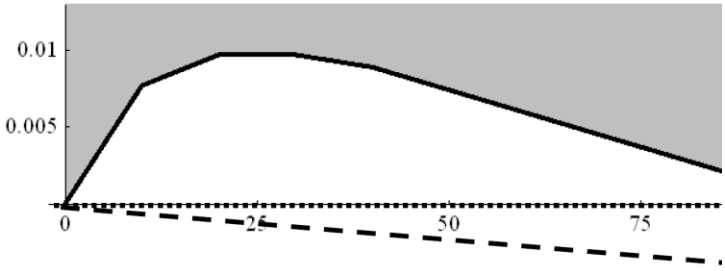

Fig. 7. Example case showing some L_2 based definitions that have an undesired negative slope.

Table 2. Asymmetric concave case evaluation

Datum Definition	Evaluation
L_1 constrained convex envelope	Good
L_1 constrained	Good
L_2 unconstrained	Poor
L_2 shifted	Poor
L_2 constrained	Good
L_2 constrained convex envelope	Good
L_∞ constrained	Good
L_∞ constrained convex envelope	Good

3.2.3 Convex and concave (wavy) datum feature

When the datum feature is a wavy shape as shown in Fig. 8, the L_1 based planes contact the surface at the left side and near the minimum as shown. The L_∞ *constrained convex envelope* is exactly horizontal. The other fits fall somewhere in between. (Depending on the size of the concave section, the L_2 *constrained* datum plane may or may not be coincident with the L_1 based planes.) One interesting note here: the implementation of the lower convex envelope actually produces an inferior plane in this case for the L_2 and L_∞ definitions. The L_1 plane is unchanged in this example case. See Table 3.

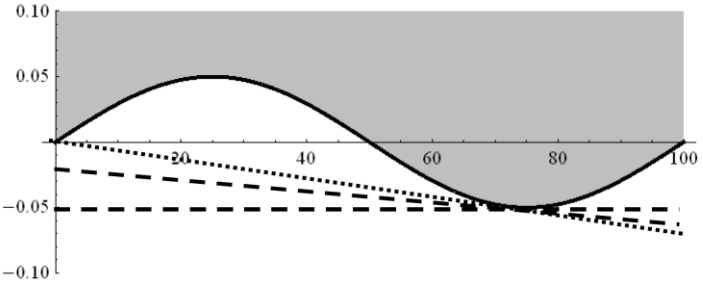

Fig. 8. Example of a wavy datum feature showing various datum definitions.

Table 3. Asymmetric concave case evaluation

Datum Definition	Evaluation
L_1 constrained convex envelope	Good
L_1 constrained	Good
L_2 unconstrained	Poor
L_2 shifted	Fair
L_2 constrained	Fair
L_2 constrained convex envelope	Poor
L_∞ constrained	Fair
L_∞ constrained convex envelope	Poor

3.2.4 Correspondence to the established practice of mating the physical part on a surface plate.

Constrained L_1 planar fitting has the effect of mating a planar datum feature to a plane in a manner that all points on the datum feature are "pulled" to the datum plane with equal force. (See [10] for a more detailed explanation of the L_1 norm in fitting.) This is the same as the physical effect of gravity pulling a datum feature surface to a "perfect" plane approximated by a surface plate (all points on the datum feature surface are pulled equally).

Furthermore, a primary datum feature will contact a surface plate at (at least) three points (unless a rocking condition exists, a case described in [4]). Likewise a secondary datum feature will contact its datum simulator at two points in general, and the tertiary at one. This 3-2-1 behavior is exactly mimicked by the constrained L_1 planar datum definitions, but this is not shared by any of the other planar datum definitions as seen in Table 4. See [8] for a complete 3-2-1 description related to L_1 planar datum definitions.

Table 4. Physical surface plate correspondence

Datum Definition	Evaluation
L_1 constrained convex envelope	Good
L_1 constrained	Good
L_2 unconstrained	Poor
L_2 shifted	Poor
L_2 constrained	Poor
L_2 constrained convex envelope	Poor
L_∞ constrained	Poor
L_∞ constrained convex envelope	Poor

3.2.5 Ability to establish the datum plane using both coordinate metrology and physical mating processes

It would be advantageous if a default planar datum definition could be realized with both coordinate metrology and using physical mating of parts. All the definitions can be realized using coordinate metrology, reducing the question to be which definitions can be realized physically. Both of the L_1 definitions can be realized physically, since the mating can occur by simply applying a downward force on the part above the location of the centroid (The centroid in both cases can be found easily using physical processes).

Also, the L_∞ *constrained convex envelope* definition can be realized physically by using shims to ensure the maximum separation between the edges of the datum feature and the surface plate is as small as possible. (This is easy to picture in 2D, since the separation at each end has to be of the same size. The extension to 3D would have at least three such critical locations.)

The L_∞ *constrained* definition requires information that is hidden to the user (in the middle section of the part that is mated to the surface plate) making it difficult or impossible to realize in practice. The L_2 based definitions are easily solved on a computer but are simply not attainable to someone having only physical, shop-floor resources. See Table 5.

Table 5. Ability to realize datum plane using surface plate or coordinate metrology

Datum Definition	Evaluation
L_1 constrained convex envelope	Good
L_1 constrained	Good
L_2 unconstrained	Poor
L_2 shifted	Poor
L_2 constrained	Poor
L_2 constrained convex envelope	Poor
L_∞ constrained	Poor
L_∞ constrained convex envelope	Good

3.2.6 Ability to use only sampled points to obtain the datum plane, without requiring weights or part information

Several of the methods in coordinate metrology, when implemented correctly, require weighted information when point sampling is not uniform (see, for example, [8, 11]). An algorithm that generates its own weights typically would need part information (though there are workarounds with varying success depending on the sampling density). The references [8, 11] show why the L_1 *constrained* and all the L_2 based definitions need weighting information. Since the L_∞ fits are determined by a few extreme points, the effect of the other points do not alter the fit, making weighting not important. We have shown in Section 2, that the L_1 *constrained convex envelope* definition can be realized without needing weights or part information, as the correct weights arise automatically within the algorithm. See Table 6.

Table 6. The need for weights to accompany sampled points

Datum Definition	Evaluation
L_1 constrained convex envelope	Good
L_1 constrained	Poor
L_2 unconstrained	Poor
L_2 shifted	Poor
L_2 constrained	Poor
L_2 constrained convex envelope	Poor
L_∞ constrained	Good
L_∞ constrained convex envelope	Good

3.2.7 Uniqueness of the datum plane

It has been pointed out in [8] that the L_1 based can lead to a rare case of nonuniqueness. An example case is a perfect "V" shaped datum feature. If there is any asymmetry in that shape, though, the L_1 based definitions produce unique datum planes. However, not only are these cases rare due to the symmetry required, they correspond to the reality of having nonuniqueness in physical situations of mating datum features with surface plates. See Table 7.

Table 7. The Uniqueness of the datum plane

Datum Definition	Evaluation
L_1 constrained convex envelope	Fair
L_1 constrained	Fair
L_2 unconstrained	Good
L_2 shifted	Good
L_2 constrained	Good
L_2 constrained convex envelope	Good
L_∞ constrained	Good
L_∞ constrained convex envelope	Good

3.2.8 Applicability to handle broken surfaces.

While all the definitions under consideration give a result when applied to a datum feature that is a broken surface, it is possible that the intent would be missed. For instance in Section 2, a case was described where the intent was that each of the three parts of the datum feature would contact the datum plane. The methods that first create the convex envelope are better at attaining this intent. The evaluations in Table 8 are based on the likelihood that the datum plane would contact points in all three regions of a 3-region broken surface.

Table 8. The ability to follow intent with broken surfaces

Datum Definition	Evaluation
L_1 constrained convex envelope	Good
L_1 constrained	Fair
L_2 unconstrained	Poor
L_2 shifted	Poor
L_2 constrained	Fair
L_2 constrained convex envelope	Good
L_∞ constrained	Poor
L_∞ constrained convex envelope	Poor

3.3 Results of Comparison

Only the improved version of the L_1 fit presented in Section 2 of this paper (the L_1 *constrained convex envelope*) was considered "good" across all eight of the criteria for consideration except one. The one exception was uniqueness where it was "fair" and those rare cases of nonuniqueness reflect the actual physical reality of the mating.

4. TIME COMPLEXITY OF ALGORITHMS

The 3D version of the algorithm relies on computing the convex hull of a given set of points, which turns out to be the dominant step in terms of computing time. Thus the computing time will largely be determined by the choice of convex hull algorithm. Quickhull [15] is a well-known, efficient convex hull algorithm that typically requires $O(n \log n)$ operations on average, where n is the number of points. However, depending on the particular data set, Quickhull can require $O(n^2)$ operations in the worst case.

The remaining steps in the algorithm require $O(n)$ operations, and are thus (for large n) not the dominant steps in computing time.

5. CONCLUSIONS

An improved definition for datum planes has been proposed for primary and secondary datums along with corresponding algorithms and codes. (The tertiary case is trivial.) The definition has been compared with seven other definitions under some consideration for the default planar datum definition. The candidate definitions were compared over eight realistic criteria and from our best evaluation, the method we present seems to have the strongest case. A reader is free to use the criteria laid out in this paper but with subjective changes to the "good," "fair," and "poor" evaluations to those given here.

ACKNOWLEDGEMENT

Dr. Paul Thomas had generated excellent graphs of some popular datum algorithms, which have been adapted into the more general discussion of Figures 6-8.

REFERENCES

[1] Srinivasan, V., 2013, "Reflections on the role of science in the evolution of dimensioning and tolerancing standards," Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, **227**(1), pp. 3-11, DOI: 10.1177/0954405412464012

[2] Tandler, W., 2008, "All Those Datum Things," Quality Digest Magazine, February.

[3] Tandler, W., 2008 "Establishing Datum Reference Frames," Quality Digest Magazine, March.

[4] ANSI/ASME Y14.5.1M, 2009 "Dimensioning and Tolerancing," The American Society of Mechanical Engineers, New York.

[5] ANSI/ASME Y14.5.1M, 1994 "Dimensioning and Tolerancing," The American Society of Mechanical Engineers, New York.

[6] ISO 5459, 2011. "Geometrical product specifications (GPS)—geometrical tolerancing—datums and datum systems." Geneva: International Organization for Standardization.

[7] Zhang, Xuzeng, and Roy, Utpal, 1993 "Criteria for establishing datums in manufactured parts" Journal of Manufacturing Systems, **12**(1), pp 36–50.

[8] Shakarji, C. M., and Srinivasan V., 2013, "Theory and Algorithms for L1 Fitting Used for Planar Datum Establishment in Support of Tolerancing Standards," ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2013), Paper No. DETC2013-12372.

[9] Hopp, T. H., 1990, "The Mathematics of Datums," American Society for Precision Engineering Newsletter, September. Available at: http://www.mel.nist.gov/msidlibrary/doc/hopp90.pdf (accessed in January 2015).

[10] Shakarji, C. M., 2011 "Coordinate Measuring System Algorithms and Filters," J. Hocken, and P. H. Pereira (Eds.), *Coordinate Measuring Machines and Systems*, CRC Press, Boca Raton, FL, pp. 153-182, Chap 8.

[11] Shakarji, C. M., and Srinivasan, V., 2013, "Theory and Algorithms for Weighted Total Least-Squares Fitting of Lines, Planes, and Parallel Planes to Support Tolerancing Standards," ASME J. Comput. Inf. Sci. Eng, **13**(3).

[12] Weisstein, Eric W. "Convex Hull." From MathWorld—A Wolfram Web Resource. Available at http://mathworld.wolfram.com/ConvexHull.html (accessed in January 2015).

[13] O'Rourke J., 1998, *Computational Geometry in C*, 2nd ed., Cambridge University Press, Cambridge, UK.

[14] Houle, M. E., Toussaint, G. T., 1988, "Computing the Width of a Set," IEEE Transactions on Pattern Analysis and Machine Intelligence, **10**(5), pp.761-765.

[15] Barber, C. B., Dobkin D. P., and Huhdanpaa H., 1996, "The quickhull algorithm for convex hulls," ACM Transactions on Mathematical Software, **22**(4), pp. 469-483.

APPENDIX A: STABLE CODE FOR COMPUTING THE AREA OF A TRIANGLE

The following Mathematica function computes the area of a triangle given coordinates of its vertices. It is reliable even for triangles with one or two small angles.

```
computetrianglearea[pt1_, pt2_, pt3_] :=
    Module[{sidelengths, a, b, c, t1, t2, t3, t4},
sidelengths = {Norm[pt2 - pt1], Norm[pt3 - pt2],
    Norm[pt1 - pt3]};
sidelengths = Sort[sidelengths];
a = sidelengths[[1]];
b = sidelengths[[2]];
c = sidelengths[[3]];
t1 = b + c;
t1 = a + t1;
t2 = a - b;
t2 = c - t2;
t3 = a - b;
t3 = c + t3;
t4 = b - c;
t4 = a + t4;
Sqrt[t1*t2*t3*t4]/4
];
```

Figure captions:

Fig. 1. Deriving a datum plane from a datum feature.

Fig. 2. Fitting a plane to a surface patch.

Fig. 3. An example case to show the need for part information when assigning weights.

Fig. 4. An example datum feature comprised of three disjoint surfaces.

Fig. 5. An example of the lower convex envelope derived from the data points (shown in 2D for simplicity).

Fig. 6. Example case showing the L_∞ based definitions yielding an undesired slope of zero.

Fig. 7. Example case showing some L_2 based definitions that have an undesired negative slope.

Fig. 8. Example of a wavy datum feature showing various datum definitions.

Table captions:

Table 1. Asymmetric convex case evaluation

Table 2. Asymmetric concave case evaluation

Table 3. Asymmetric concave case evaluation

Table 4. Physical surface plate correspondence

Table 5. Ability to realize datum plane using surface plate or coordinate metrology

Table 6. The need for weights to accompany sampled points

Table 7. The Uniqueness of the datum plane

Table 8. The ability to follow intent with broken surfaces