

On the Enduring Appeal of Least-squares Fitting in Computational Coordinate Metrology

Vijay Srinivasan
Fellow ASME
Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899, U.S.A.
email: vijay.srinivasan@nist.gov

Craig M. Shakarji
Member ASME
Physical Measurement Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899, U.S.A.
email: craig.shakarji@nist.gov

Edward P. Morse
Member ASME
Department of Mechanical Engineering and Engineering Science
University of North Carolina at Charlotte
Charlotte, NC 28223, U.S.A.
email: emorse@uncc.edu

Abstract: The vast majority of points collected with coordinate measuring machines are not used in isolation; rather, collections of these points are associated with geometric features through fitting routines. In manufacturing applications, there are two fundamental questions that persist about the efficacy of this fitting – first, do the points collected adequately represent the surface under inspection; and second, does the association of substitute (fitted) geometry with the points meet criteria consistent with the standardized geometric specification of the product. This paper addresses the second question for least-squares fitting both as a historical survey of past and current practices, and as a harbinger of the influence of new specification criteria under consideration for international standardization. It also touches upon a set of new issues posed by the international standardization on the first question as related to sampling and least-squares fitting.

Keywords: coordinate metrology, least-squares fitting, standardization, optimization, manufacturing, sampling

1 Introduction

Computational coordinate metrology deals with the problem of fitting and filtering of discrete geometric data measured on the surface (or in the interior) of a manufactured product [1, 2]. Such measurements are typically made by coordinate measuring machines (CMM). In the manufacturing industrial setting, we are interested in using these measurements (1) to verify if the manufactured part is within designer-specified tolerances, and (2) to evaluate the capability of a manufacturing process and ensure that the process stays in control during the production run.

As with virtually every technology used in manufacturing, the availability of inexpensive computing power has made advanced data analysis available to many users in a more timely manner than ever before. Initial advances in CMM technology focused on the real-time compensation of geometric errors in the CMM structure and the fitting of measured basic geometric elements.

While multiple fitting choices were available (e.g., minimum-zone fitting) the stability, speed, and robustness of least-squares algorithms made them a natural choice for early programmers. The very public release of a GIDEP alert [3] was the first indication to many users that the use of least-squares did not fully *capture the spirit* of the ASME Y14.5 tolerancing standard used to specify most products. The recognition of a ‘methods divergence’ between coordinate metrology systems and open setup (i.e., hard-gaging) systems led to the implementation of the non-least-squares algorithms described later in this paper in Section 4.

In this paper, we introduce least-squares fitting and give ten reasons in Section 2 for its enduring appeal. We note that it is not our intent to recommend one fitting choice over another. Rather we simply seek to document some of the reasons for the prevalence of the least-squares fits as seen today, including an emerging urgency posed by major developments in ISO (International Organization for Standardization) standards. In Section 3 we briefly outline ongoing computational metrology research for free-form surfaces using least-squares fitting. Section 4 describes current industrial practice driven by zone-based tolerance specifications in some detail, and offers a technical and historic perspective on the *methods divergence* alluded to earlier. It sets the stage for Section 5, which defines the continuous least-squares fitting problem and exposes a convergence problem associated with sampling that has hitherto remained hidden from general technical discussion in this field; it also provides a remedy in the form of weighted least-squares fitting.

2 Ten Reasons for the Enduring Appeal of Least-squares Fitting

A CMM user has a wide range of software fitting options ranging from using a least-squares criterion to employing (among others) minimum-zone, maximum-inscribed, minimum-circumscribed, and L^1 fits along with constrained or shifted variations of these. Over the past two decades, the least-squares fit has remained the most popular fit used in practice, despite the awareness and availability of various other fit objectives. Listed below are ten reasons for the enduring appeal of the least-squares fitting criterion.

2.1 Ordinary least-squares regression has a long history and large existing application base in data analysis.

Linear least-squares fitting has been a popular tool for analyzing experimental data since the days of Gauss and Legendre [4]. In the context of their classical work, we will cast the simple problem of fitting a straight line to a set of m points in a plane as shown in Figure 1. The fitted straight line may be represented in the familiar linear form

$$x_2 = c_1 x_1 + c_2, \quad (1)$$

where c_1 and c_2 are the coefficients to be determined. The problem becomes interesting when $m > 2$ and the points are not collinear – a typical case in coordinate metrology. Assuming that the ‘errors’ are only the vertical direction, which is a major assumption that we will remedy shortly, we have an over-determined set of equations given by

$$\begin{bmatrix} p_{1,1} & 1 \\ p_{2,1} & 1 \\ \dots & \dots \\ p_{m,1} & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} = \begin{Bmatrix} p_{1,2} \\ p_{2,2} \\ \dots \\ p_{m,2} \end{Bmatrix}, \quad (2)$$

where the Cartesian coordinates of the point p_i are represented by $(p_{i,1}, p_{i,2})$ and $m > 2$. We can write the set of equations of (2) in a convenient matrix notation as $Ac = b$ and – here lies the contribution of Gauss and Legendre – an optimizing vector c can be obtained by solving the *normal equation* $[A^T A]c = A^T b$, even though Eq. 2 generally has no solution.

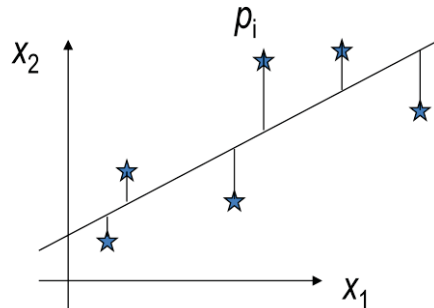


Figure 1: Linear least-squares fitting of a straight line in a plane.

It can be easily shown that this solution for c solves the optimization problem

$$\min_{c_1, c_2} \sum_{i=1}^m (p_{i,2} - p_{i,1}c_1 - c_2)^2, \quad (3)$$

where the objective function is the sum of the squares of the errors. Hence we have the *least-squares* fitting. It is also known as linear regression, especially in the statistical literature. In the parlance of popular scientific and technical computing, popularized by the likes of MATLAB, we can find this solution using the backslash operator as in $c = A \backslash b$ without resorting to the explicit solution of the normal equation. (All codes given in this paper are in the MATLAB language). Matrix generalizations of this kind are very useful for least-squares fitting (or, linear

regression) in higher dimensional cases. For example, fitting a plane to a set of points in space is a simple exercise of expanding A to an $m \times 3$ matrix.

2.2 Linear total least-squares problems are similar in nature to ordinary least-squares regression problems and are easy to solve.

A slightly different problem that is of major interest to coordinate metrology arises when we want to fit a straight line to a set of m points in a plane, without prejudice to whether the errors are in the horizontal or vertical direction. Figure 2 shows a case where the errors are considered perpendicular to the line to be fitted. In such a case, it is not a good idea to represent the straight line in the linear form (1) because it does not admit the possibility of the line being vertical. So, we pick a more general representation in the form of a point q on the line, and a vector v (usually unitized) perpendicular to the line. If we denote the perpendicular distance of a point p_i to the fitted straight line l as d_i , then we pose a new optimization problem

$$\min_{q,v} \sum_{i=1}^m d_i^2, \quad (4)$$

which is known as the *total least-squares* fitting problem [5-7]. Since the ‘error’ distances d_i ’s are measured orthogonal to the fitted straight line, it is also referred to as the *orthogonal regression* problem in some literature.

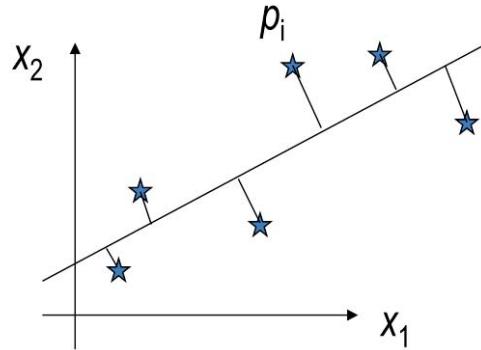


Figure 2: Total least-squares fitting of a straight line in a plane.

It is indeed fortunate that part of the solution to the problem in (4) can be found by a clever observation that the centroid of the m points lies on the fitted line. So we can always set $q = (\frac{1}{m} \sum_{i=1}^m p_{i,1}, \frac{1}{m} \sum_{i=1}^m p_{i,2})$. Having solved half the problem, we can turn to finding the vector v perpendicular to the fitted line. For this, we first translate all the points so that the centroid, in this case q , lies at the origin of the coordinate frame. Denoting such translation by the mapping of p_i to p_i^* we can then form a matrix of *central* Cartesian coordinates as

$$A = \begin{bmatrix} p_{1,1}^* & p_{1,2}^* \\ p_{2,1}^* & p_{2,2}^* \\ \dots & \dots \\ p_{m,1}^* & p_{m,2}^* \end{bmatrix}. \quad (5)$$

It can then be shown that the vector v can be found as the singular vector corresponding to the smallest singular value of A in (5). Alternatively, with some sacrifice of numerical accuracy, the same vector v can be easily found as the eigenvector corresponding to the smallest eigenvalue of $A^T A$, which is just a real symmetric 2×2 matrix.

To illustrate the simplicity of the total least-squares computation, which has contributed considerably to its appeal in computational coordinate metrology, a short code that does the entire computation is given below:

```
function [q, v] = tlsqLine(X, Y)
    q = [mean(X); mean(Y)];
    A = [(X - q(1)), (Y - q(2))];
    [U, S, V] = svd(A); % Can be replaced by [V, S] = eig(A'*A);
    [s, i] = min(diag(S));
    v = V(:, i);
return;
```

This six-line code snippet has the reputation of being the pithiest non-trivial code in all of computational coordinate metrology. What is more, the theoretical arguments and the code can be directly extended to higher dimensional cases [7].

Before we leave this section, it is worth making some physical analogy to total least-squares fitting of lines and planes. We will have an occasion to exploit this analogy later in Section 5. What we are actually computing is the inertial ellipsoid of a set of unit point masses. To see this, consider the set of m points as physical point masses with unit weight assigned to each point. As in classical physics, treat this set of points as a rigid entity and associate a second-order tensor by taking all second moments with respect to the original coordinate system. It is then an elementary exercise in classical mechanics to compute the principal moments of inertia and the principal axes – together, they define the size and orientation of the inertial ellipsoid centered at the centroid of the point masses. The normal vector v that we compute for least-squares fitting is aligned with a principal axis. The same idea resurfaces in statistical analysis in the form of *principal component analysis*.

It is interesting to contemplate cases where one may assign non-uniform weights to the measured points, leading to a weighted total least-squares fitting problem. Later, in Section 5, we will show that this arises naturally when we consider the problem of least-squares fitting to a continuum of points (as opposed to a discrete set of points) and proceed to solve it using numerical integration. But, before that, we need to address some nonlinear total least-squares problems still involving discrete sets of points.

2.3 Least-squares fitting makes nonlinear problems more manageable.

After disposing of the linear elements such as straight lines and planes quite easily and elegantly in Section 2.2, we turn to the more sobering cases of fitting nonlinear elements such as circles, spheres, cylinders, cones, and tori to a set of measured points. It is possible to pose their fitting as nonlinear optimizations problems. In general, such nonlinear problems are quite nasty to handle.

However, the least-squares formulation provides some desirable structure to the problem that renders its solution manageable [5, 6].

All nonlinear total least-squares problems are solved iteratively, starting from an initial guess for the solution. Of the many iterative techniques that have been proposed and tried, three of the most popular ones are Gauss-Newton, Levenberg-Marquart, and ‘trust region’ algorithms. The optimization toolbox implements all of them in its *lsqnonlin* function. Since these iterative methods converge only to the local minimum close to the initial guess, it is important to start with as good an initial solution as we can find.

For circles in a plane and spheres in space, a clever observation based on parabolic transformation provides good, automatic starting solutions. To illustrate this, consider the nonlinear total least-squares fitting of a circle to a set of m points in a plane. Figure 3 illustrates a circle parameterized by its radius r and its center coordinates (x_0, y_0) . The perpendicular distance of any point p_i with coordinates (x_i, y_i) to the fitted circle is given by $d_i = r_i - r = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} - r$. To fit a circle to m such points, we pose the optimization problem,

$$\min_{x_0, y_0, r} \sum_{i=1}^m d_i^2, \quad (6)$$

which is clearly a nonlinear optimization problem due to the radical in d_i . To employ any iterative technique to solve this problem, we need to come up with a good starting solution, which is provided by the following observation.

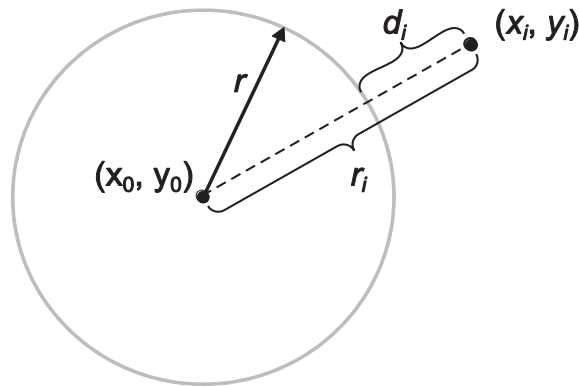


Figure 3: Perpendicular distance between a point and a circle in a plane.

Consider a unit paraboloid of the form $z = x^2 + y^2$ as shown in Figure 4. Any point in the xy plane with coordinates (x_i, y_i) can be projected vertically onto the paraboloid to a point with coordinates $(x_i, y_i, x_i^2 + y_i^2)$. It can be easily shown that co-circular points in the xy plane, when projected onto the unit paraboloid, become coplanar points in space [8]. This fact is generalizable to higher dimensions. For example, co-spherical points in three-dimensional space, when projected on the unit paraboloid in four-dimensional space become coplanar (in a hyperplane) in 4D.

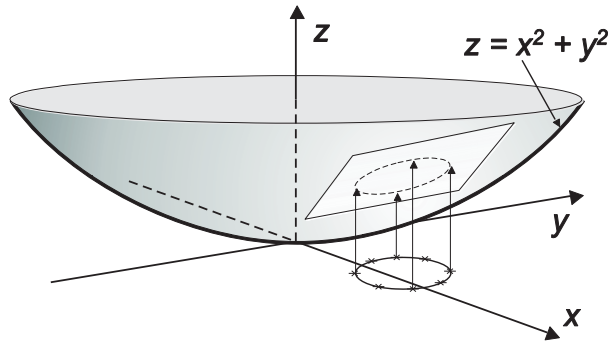


Figure 4: Parabolic projection

Coming back to our circle fitting problem, our premise is that the m points in the xy plane are not co-circular and so, when vertically projected onto the unit paraboloid, the projected points will not be coplanar in space. But this is not a problem because we can fit a linear least-squares plane P (it is not necessary to fit a total least-squares plane in this case) to these projected points in space. This plane P will intersect the paraboloid in an ellipse. When this ellipse is vertically projected back to the xy plane, it will yield a circle that is a good starting solution for our iterative method to solve the nonlinear least-squares problem. A short but complete code to find the total least-squares circle utilizing the *lsqnonlin* function in the optimization toolbox is given below to illustrate the power of automatically generating a good starting solution.

```
function [xc, yc, rc] = tlsqCircle(X, Y)
    m = size(X);
    va = [X Y ones(m)] \ (X.^2 + Y.^2);
    x0 = [va(1)/2; va(2)/2; sqrt(va(3) + (va(1)/2)^2 + (va(2)/2)^2)];
    %
    % After parabolic projection
    %
    function [F, J] = myfun(x)
        rvec = sqrt((X - x(1)).^2 + (Y - x(2)).^2);
        F = rvec - x(3);
        J = [-(X - x(1))./rvec, -(Y - x(2))./rvec, -ones(m)];
    end
    options = optimset('Jacobian', 'on');
    x = lsqnonlin(@myfun, x0, [], [], options);
    xc = x(1); yc = x(2); rc = x(3);
```

Not only has this been found to be effective for fitting circles, its generalization to total least-squares fitting of spheres has also been very successful.

This success in automatically finding good starting solutions for circles and spheres has, alas, not extended to other important geometric elements such as cylinders, cones and tori. Computational metrologists have struggled with this problem and tried various *ad hoc* techniques to get good starting solutions with only limited success. This might change with some recent theoretical and computational developments in what might be called *Plücker coordinate metrology*, which will be described briefly below.

Traditional coordinate metrology is based on measuring Cartesian, polar, cylindrical-polar or spherical coordinates of points on the surface (or in the interior) of an object. In three-dimensional space, each point has three coordinates. The type of coordinates measured (e.g., Cartesian vs cylindrical-polar) depends to a large extent on the physical structure of the coordinate measuring machine. But suppose that, in addition to measuring the point coordinates on a given physical surface, we can also measure – or estimate – the surface normal vector at that point. This gives some added information that can be exploited in further computations such as fitting. Formally, we now enter the domain of Plücker coordinate metrology, where each point is associated with six coordinates – three involving moments of the normal vector and the other three involving just the normal vector.

Theoretically, Plücker coordinates provide a $\mathbb{R}^3 \rightarrow \mathbb{R}^6$ mapping by taking a G^1 continuous surface in \mathbb{R}^3 and mapping it to a set in \mathbb{R}^6 . To accomplish this, consider a position vector p and a normal vector n of a point on a surface (which is assumed to have a normal at that point – hence the G^1 continuity assumption), as shown in Figure 5, and create the six Plücker coordinates $(p \times n, n)$. Here $p \times n$ is the vector cross product of the position vector p and the normal vector n .

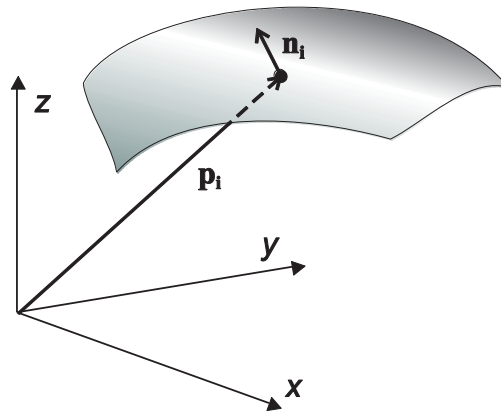


Figure 5: Position vector and normal vector at a point on a surface.

So far, we seem to have achieved nothing new in embracing Plücker coordinates. In fact, we have added more cost by including the normals. But what recent developments have shown is that if the surface in question falls under one of the six symmetry classes (out of a total of seven) for surfaces, then its mapping to \mathbb{R}^6 under Plücker coordinate transformation spans a linear subspace in \mathbb{R}^6 [9, 10]. This is eerily similar to the parabolic transformation we saw earlier, where a nonlinear object in a lower-dimensional space spans a linear subspace when mapped to a higher-dimensional space. In practice, of course, we need to fit a linear subspace to the mapped points in \mathbb{R}^6 . All the machinery of linear algebra and matrix numerical analysis can then be

brought to bear in teasing some useful information automatically out of this fitted linear subspace.

Initial experiments with the Plücker coordinate transformation show some promise in suggesting good starting solutions for the nonlinear least-squares fitting of cylinders, cones and tori. But this needs further exploration before it can be declared suitable for deployment in an industrial setting. For the moment, we will leave this as a promising research topic.

2.4 The least-squares software used in coordinate metrology has had a greater amount of formal testing through national labs than the other fits discussed.

In the 1980s data was submitted to CMM software that performed least-squares fits. The errant results (due to fitting problems and/or methods divergence) led to a Government Industry Data Exchange Program (GIDEP) alert being issued in 1988 indicating that fitting software could be a significant source of overall measurement uncertainty [3].

A number of institutions had a hand in responding to this, including formal testing of least-squares fitting software being done both at NIST (National Institute of Standards and Technology) in the U.S.A. and PTB (Physikalisch-Technische Bundesanstalt) in Germany. Though there are differences in the testing procedures, both operate in the same basic way: test data sets are designed and generated to be representative of some range of measuring tasks. These data sets are submitted to both the software under test and to reference fitting software. The fit results from the software under test are compared with the reference fits, and the outcome of that comparison is summarized in a test document.

The most commonly performed test at NIST (in the Algorithm Testing and Evaluation Program – Coordinate Measuring Systems, ATEP-CMS) follows procedures and test data sets that conform to the ASME B89.4.10 Standard, which specifies a number of things about the data sets for various geometries such as size, form, location, orientation, number of points, sampling strategies, and partial feature sampling. Following these procedures helps ensure a broad spectrum of test cases that emulate a range of real-world measurements. Figure 6 is an example of what a portion of such a test report might look like.

ASME B89.4.10-2000 Standard Default Test

Geometry Type	Mean (RMS) Deviation			
	Separation (μm)	Tilt (arc seconds)	Radius/dist (μm)	Apex (arc seconds)
Lines	$< 10^{-5}$	2.1×10^{-7}	—	—
Lines 2D	$< 10^{-5}$	$< 10^{-7}$	—	—
Planes	$< 10^{-5}$	$< 10^{-7}$	—	—
Circles	8.4×10^{-5}	3.3×10^{-7}	1.5×10^{-5}	—
Circles 2D	9.6×10^{-4}	$< 10^{-7}$	9.0×10^{-4}	—
Spheres	7.0×10^{-4}	—	$< 10^{-5}$	—
Cylinders	1.3×10^{-3}	7.0×10^{-4}	1.0×10^{-3}	—
Cones	1.7×10^{-2}	2.7×10^{-3}	5.1×10^{-3}	5.5×10^{-3}

Figure 6: A part of a sample test report following the ASME B89.4.10 Standard. The Mean (RMS) deviations shown serve as reasonable valuations for the uncertainty contributed by incorrect least-squares fitting results (not considering methods divergence issues).

The testing procedure works as follows:

- 1) The testing body generates test data sets along with corresponding reference fits. The reference fits may be found by means of a reliable reference algorithm or by a technique that generates the data in a way that defines its fit ahead of time.
- 2) The same data sets are submitted to the software under test, which computes their fits.
- 3) The fits from the software under test are then compared with the reference fits by the testing body.

Some data files with reference fits are available for download from the NIST website (<http://www.nist.gov/pml/div681/grp11/cst-algorithmtesting.cfm> Algorithm Testing; Physical Measurement Laboratory).

It is important to understand exactly what any software testing does and does not cover. Currently, the national laboratories involved in formal CMM software testing services cover only the least-squares fits and only for the simple geometries and only for unconstrained fitting. While such testing is extremely valuable, as it has led to significant software errors having been corrected, it is nonetheless incorrect to extend those results to other functionality of the software, of which there are many. It is incorrect to assume that software that has undergone such a small degree of testing (albeit valuable) has been “verified,” “validated,” or “certified,” since these terms connote a sense of security and comprehensiveness.

Software that performs least-squares fits correctly might perform Chebyshev fits poorly [11]. In fact, least-squares testing of simple geometric shapes does not reveal important information about: fitting to complex surfaces, or CAD shapes, datum reference frame construction, conformance to specifications (e.g., position, profile tolerances, etc.), outlier

rejection, filtering techniques, or any of a number of other vital features that software implements. These are also topics of current research in computational coordinate metrology, and we will mention them briefly in Section 3.

2.5 Least-squares fitting problems generally have easier objective functions than those for the other fits discussed.

Fitting is achieved by finding the location, orientation, and even size of a geometry such that an objective function is minimized. The least-squares problem uses the objective function of the sum-of-squares of the residuals.

A reader not familiar with optimization algorithms can think of an analogous problem of finding the deepest point in a swimming pool filled with opaque liquid and with the shape of the bottom being unknown. It is allowable to use a long ruler to make simple vertical depth measurements, one at a time, without dragging the ruler along the bottom surface. A more sophisticated method would, at each step, allow for a vertical depth measurement plus an indication of the direction and magnitude of the slant of the bottom surface at each measurement. (Think of a small ‘foot’ attached to the end of the ruler using a ball-and-socket joint. The foot rotates to rest on the bottom at the same slant as the surface at that location – provided the location was not a jagged peak or valley.) The additional piece of slant information acquired at each step can speed the process in finding the deepest point.

In this analogy, the objective function is the shape of the bottom of the pool. The easier the shape (*e.g.*, a simple bowl shape) the less time the process takes to identify the deepest point. A pool bottom having lots of mountains and valleys would make the process more difficult, since it could appear that the low point has been found when a deeper point could exist elsewhere. See Figure 7.

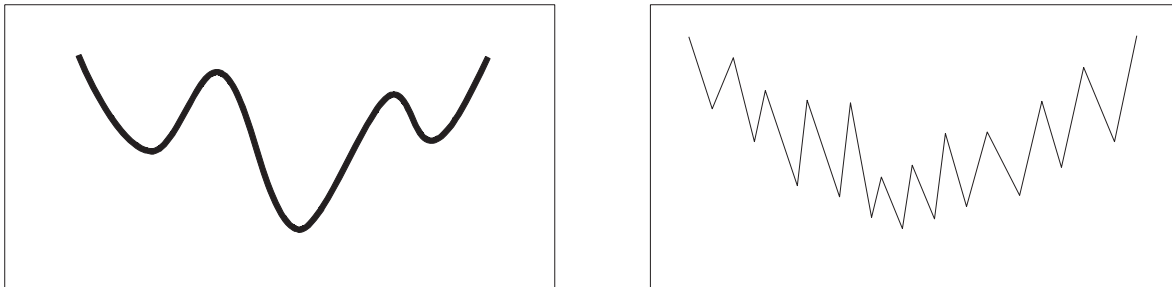


Figure 7: For the two objective functions shown, it is easier to find the minimum for the one on the left, since it is smoothly varying and since the global minimum is not so much hidden among nearby, local minima.

Objective functions for minimum-zone, maximum-inscribed, minimum-circumscribed, and L^1 fits can be difficult since the objective functions can have locations where they are not differentiable (as in the right hand side of Figure 7). Least-squares objective functions in

coordinate metrology are easy to work with, since they are generally smooth (*i.e.*, differentiable everywhere) and since they are generally convex over a large region surrounding the global minimum.

2.6 Least-squares objective functions generally have unique solutions.

Another desirable characteristic of least-squares fitting is that the solution is generally unique. (Pathological cases exist where this is not true, but these do not arise in practice—especially given sufficient sampling).

There are times when a perfect plane needs to be associated with an imperfect but nominally planar surface (*e.g.*, in datum establishment). If the imperfect surface has a peak near its center, a one-sided mating plane might not be unique, since it could ‘rock,’ pivoting on the high point from one mating orientation to another. In contrast, the least squares fit would be unique given a fixed, realistic data set.

In the case of maximum-inscribed fits, it is possible to have a non-unique situation as shown in Figure 8.

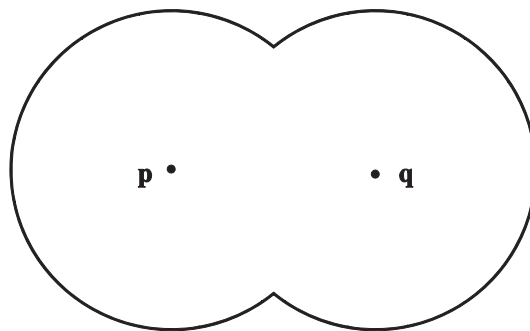


Figure 8: A set of points taken around the shape shown could have two maximum-inscribed circles, one centered at *p* and one at *q*. A least-squares fit to the same data would be unique.

2.7 Least-squares fitting is generally much faster – especially when large numbers of points are involved – than the other fits discussed.

Due to the more complex objective functions associated with other fitting choices (as mentioned in 2.5) least-squares fitting can be computationally performed much faster. For line and plane fitting, the computational time required reduces down to a singular vector problem, which takes $O(m)$ operations, where m is the number of points. For the minimum-zone line fit (as will be seen in section 4), an exhaustive search method involves a convex hull algorithm plus a search over all antipodal pairs, the latter step alone being an $O(h^2)$ task, where h is the number of points on the convex hull.

For nonlinear fitting, the least-squares fit is much faster as well. If a good initial guess is found then a fast ‘straight downhill’ minimization algorithm can be used. Examples of fast algorithms that can be used (as mentioned in Section 2.3) are Gauss-Newton and Levenberg-Marquardt. These cannot be used for many other fit objectives mentioned in this paper. For these other fits, the minimum of the objective function is hidden among several nearby local minima, making a more complex, global optimization algorithm necessary. More complex (and slower algorithms) that could be employed include brute-force exhaustive-search, simulated annealing, basin hopping, genetic algorithms, etc.

The result of needing one of these more complex algorithms is an enormous increase in computational time. Precise numbers for comparison are highly dependent on the specific set of points and algorithms used, but it is very easy for the least-squares fit to be 100 times faster than the other fit objectives discussed in this paper.

2.8 Least-squares fits are less affected by outliers when compared with the other fits discussed.

The existence of outliers in coordinate metrology data collection is a reality, for a wide range of reasons ranging from a speck of dirt to an isolated machine anomaly. An outlier might not always be removed before fitting. A minimum-zone fit is entirely determined by the locations of a few extreme points. (See, for example, minimum-zone line fitting discussed in Section 4). This means that the solution can be drastically affected by just one errant point. This effect might manifest itself in poor repeatability, if several measurements were made employing the minimum-zone fit. The instability in that fit is also seen in maximum-inscribed, minimum-circumscribed, and supporting plane fits as well.

The least-squares fit has more of an averaging effect over all the data, and thus the effect of a single errant point is mitigated by the locations of several other points. The averaging effect is quantifiable, in that the average residual is zero for unconstrained least-squares fits of lines and circles (considered in 2D), spheres, cylinders, and cones. Furthermore, the averaging effect is intuitive in this sense: if someone unfamiliar with fitting were shown points on a plane that approximated a line, and were then told to draw the straight line represented by the points, the line drawn would likely approximate the least-squares fit.

It turns out that the seldom-used L^1 fit is even less sensitive to outliers, but that fit is not widely used, probably due to the other reasons for the least-squares appeal discussed in this paper. Even still, all the fits discussed in this paper (including the least-squares and L^1 fits) can give meaningless results due to even a single outlier that is extremely deviant. In such cases, outlier removal or similarly effective data handling should be incorporated.

2.9 Least-squares fits have effective tests for correctness.

A necessary-condition test can be applied to a fit as a check as to its correctness. For instance, for an unconstrained least-squares circle fit (considered in 2D) the average residual must be zero. If it is not, the fit is not the least-squares fit. Similarly, a correct L^1 fit in this case would have residuals whose median is zero when the number of data points is odd. When the number of data

points is even, the test would be that middle two residuals (when sorted) cannot have the same sign. And a correct minimum-zone fit would have at least four residuals attaining the maximum absolute residual value.

These conditions are necessary but not sufficient. That is, if they are not met, the solution is certainly wrong; but the tests can be satisfied even with incorrect fits. As it turns out, these tests – though helpful – are quite weak. They are commonly satisfied even with sub-optimal fits.

But in the case of least-squares fitting, there is a much more powerful test for correctness. That is, the gradient of the objective function must be zero at the solution. This is easy to calculate – formulas for the gradient are given for all the basic shapes in [5, 6]. While this is still a necessary but not sufficient condition, it is a very strong condition that makes it quite easy to detect incorrect fits.

In the cases of non-least-squares fits – for instance maximum-inscribed circles and cylinders – an errant solution can appear correct even though the iterative algorithm has reported it's fit result without having converged on the optimal (*i.e.*, correct) solution.


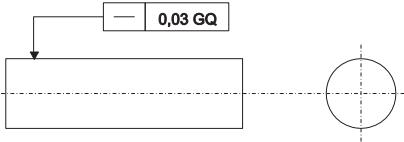
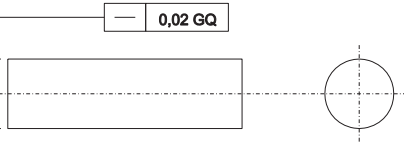
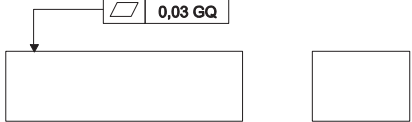
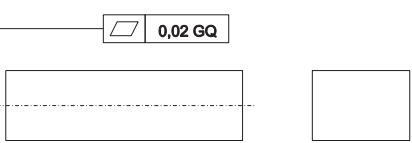
2.10 Emerging ISO Geometric Product Specifications call out the least-squares fit explicitly.

ISO 1101 will soon be amended to enable specifications such as those shown in Table 1. In addition, ISO 14405-1 will soon allow dimensional tolerance specification for linear size such as the one shown in Table 2, Example 1; a similar method is being developed for angular sizes, as shown in Example 2 in the same Table.

The semantics presented in the last columns of Tables 1 and 2 using the English language text can be augmented by figures to make them a bit clearer, but they will still remain ambiguous. We will address this issue in some detail in Section 5. For now, suffice it to point out that total least-squares fitting will soon become normative in tolerance specification, and this presents an urgent need to examine the theory and practice of total least-squares in detail.

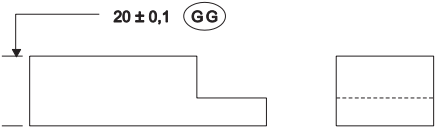
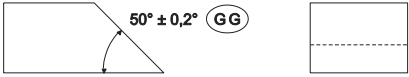
In fact, it is this urgency that has motivated us to devote this entire paper to least-squares fitting. In that sense, we have reserved the enumeration of the best reason to study least-squares fitting to the last and we will continue its exposition in the rest of the paper.

Table 1 : Examples of straightness and flatness specifications syntax, along with their semantics, allowed by the upcoming, newly amended ISO 1101.

Example	Syntax (Drawing indication)	Semantics* (Explanation)
1		<p>The root-mean-square parameter of any extracted (actual) line on the upper surface, parallel to the plane of projection in which the indication is shown and measured from the total least-squares associated straight line, shall be less than or equal to 0,02 mm.</p>
2		<p>The root-mean-square parameter of any extracted (actual) generating line on the cylindrical surface, measured from the total least-squares associated straight line, shall be less than or equal to 0,03 mm.</p>
3		<p>The root-mean-square parameter of extracted (actual) median line (i.e., axis) of the cylinder to which the tolerance applies, measured from the total least-squares associated straight line, shall be less than or equal to 0,02 mm.</p>
4		<p>The root-mean-square parameter of any extracted (actual) surface, measured from the total least-squares associated plane, shall be less than or equal to 0,03 mm.</p>
5		<p>The root-mean-square parameter of extracted (actual) median plane of the indicated feature of size, measured from the total least-squares associated plane, shall be less than or equal to 0,02 mm.</p>

*These statements of semantics are composed from different statements in the upcoming amended version of ISO 1101. These are not the formal statements of explanation associated with the drawings in the official ISO 1101 amendment.

Table 2: Examples of linear and angular size specifications syntax, along with their semantics.

Example	Syntax (Drawing indication)	Semantics* (Explanation)
1		The linear size of the indicated feature of size, with least-squares association criterion, shall be within the indicated limits.
2		The angular size of the indicated feature of size, with least-squares association criterion, shall be within the indicated limits.

*These statements of semantics are composed from different statements in ISO 14405-1. These are not the formal statements of explanation associated with the drawings in the official ISO standard.

3 Computational Metrology Research for Free-form Geometries

The examples cited in the previous section show the fitting of basic geometric elements, such as those listed in the first column of the table in Figure 6. Most of these primitive geometries have well defined fitting algorithms, and the trends in recent topical studies in computational metrology have focused on the fitting of free-form geometry. Modern point collection hardware permits the gathering of many data points in a short amount of time, and the focus of inspection software is now how to filter the data and fit it to a complex nominal surface. (The reader will recall that we alluded to this while discussing the limitations and caveats of current testing of least-squares fitting in Section 2.4.) This is an area where least-squares fitting, especially if performed with weighting, is often the method that best captures the performance characteristics desired by the designer. This is because free-form surfaces will likely be responsible for much different functionality in the final design than that of the prismatic mating surfaces used in assembly. The free-form surfaces are often used for flow control (be it fluid or gas) and the geometric properties desired are more statistical in nature. For this reason, the use of a more balanced metric such as that captured in least-squares fitting may be preferred. In this section we will briefly touch upon some of the research currently underway.

3.1 Data filtering

The filtering of dense data on free-form surfaces is often necessary because of the nature of the instruments used to collect the data. Laser-based systems often have a noisy return signal, although the mean of the points measured can have low uncertainty. The issue of filtering of the data to best capture the actual surface is of interest both to users and instrument manufacturers. The user wishes to remove the high-frequency variability associated with the instrument without biasing the measurement of the underlying geometry. While the application of filters to regular

nominal geometry can be tailored to remove biases, the application of (e.g., Gaussian) filters to free-form surfaces can distort the measurement results. Jiang et al [12] have reported on methods to address this problem but, as with all filtering problems, assumptions must still be made about the underlying character of the surface.

3.2 Fitting to free-form surfaces

Given that any filtering to remove measurement noise and/or surface texture variations has already been performed, the task of fitting the remaining data to a nominal free-form surface can still be challenging. The major components of the measurement task are: partitioning of the data – that is, identifying the feature to which each of the measured points belongs; registration of the data – that is, aligning the key points or areas of the surface(s) with the model; actually fitting the data to the nominal model; and finally, collecting additional points (re-sampling) if any metrics concerning the quality of the fit indicate that this is necessary.

3.2.1 PARTITIONING AND REGISTRATION

The partitioning of the data to match individual features is a current area of research [9, 10] and cannot, at this time, be done automatically except for certain classes of feature geometry. In almost all cases, there is operator intervention required for this process. The partitioning problem is related almost exclusively to point cloud measuring systems – the nature of Cartesian measuring machines is that each point's feature association is known *a priori* as only a relatively small number of points are collected for each feature.

The registration of point data to the nominal model is necessary when one or more datums are established on the free-form surface. In this case, initial points are measured on the surface that correspond to where the part would rest in a locating fixture. The relative location of these measured points will often not correspond to the nominal locations of the datum targets. The local coordinate system is adjusted based on the measured points, and a new set of points is measured. This process is repeated until the measured points coincide with the theoretical locations of the fixture. Where the surfaces being measured have very shallow slopes, this registration process can take many iterations. Research that breaks this problem into rough pattern matching of the nominal and measured surfaces, followed by a final fitting of the surfaces is attempting to speed up the process by reducing the time spent in the early iterations of this process [13].

3.2.2 SAMPLING STRATEGIES

When measuring surfaces with non-uniform curvature, it may be desirable to vary the density of the point sampling depending on the local curvature of the surface. Whether the sampling strategy is established based on the nominal model or is developed dynamically based on the measured surface attributes, the influence on fitting algorithms can be substantial. Edgeworth and Wilhelm [14, 15] performed some of the early work on this problem, and work on adaptive sampling continues today [16]. The impact of adaptive sampling on least-squares fitting has not – to our knowledge – been studied extensively, but it is clear that the weighting of points

acquired with non-uniform sample densities will be critical to the overall fitting process (more on this later in Section 5).

3.2.3 FITTING AND CONFORMANCE ASSESSMENT

Three main issues arise when fitting data to free-form surfaces: (1) determining the shortest perpendicular distance between the nominal surface and a measured point can be time consuming, and may have non-unique results; (2) the different densities of points in different regions may bias the fitting algorithm; and (3) the specification may have non-uniform profile requirements (as in Figure 9), where the deviations from nominal have different criteria depending on the part of the profile with which they are associated.

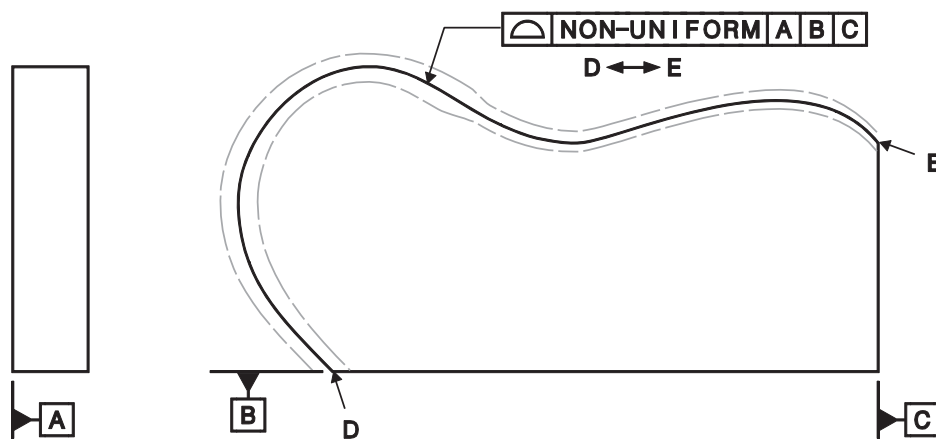


Figure 9: Specification of non-uniform Profile tolerance. The nominal profile and tolerance zone boundaries will typically be specified in a CAD system, but may be elaborated through basic dimensions on the drawing.

The problem of determining the distance between points and a surface is a computational issue, and the main difficulties arise when there are multiple surface points with the same perpendicular distance to the point, or there is no 'closest point' on the surface that has a well-defined normal direction. As is the case with partitioning, it is the automation of this task that is difficult – most algorithms will fail under certain conditions, such as sharp corners or small radii at edges.

The issues of changing sampling densities on the workpiece surface, and of non-uniform conformance zone boundaries both require some sort of weighted fitting, and the work in these areas is ongoing. An initial assessment of the impact of weighting and sample densities will be found in Section 5 of this paper.

4 Current Industrial Practice Driven by Zone-based Specifications

One of the major objectives of computational coordinate metrology is to support the verification of whether a given manufactured part is within designer specified tolerances. Given the prominence of least-squares fitting in commercially available software, it is ironic that it is not the basis for the current practice of tolerance specifications. To understand this problem and to appreciate its implication for least-squares fitting, let's describe a simple instance involving the flatness tolerance.

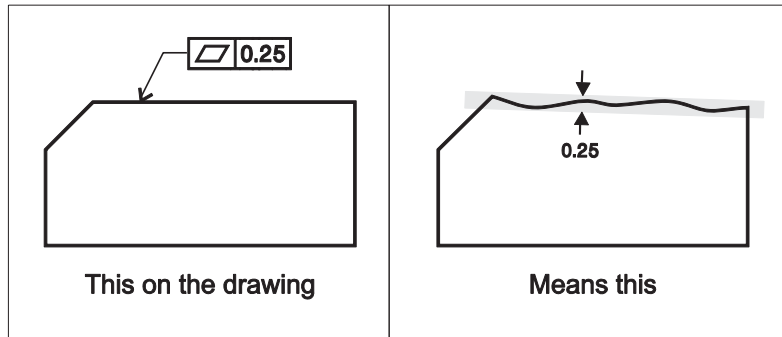


Figure 10: ASME defines flatness tolerance specification.

Figure 10 shows how the flatness tolerance specification is defined in the ASME Y14.5 standard today. An equivalent current definition by ISO can be found in Figure 11. To conform to the specification, the toleranced feature (which is supposed to be nominally flat) on a manufactured part should be contained within a tolerance zone bounded by two parallel planes that are separated by the specified tolerance value (0.25 units in Figure 10, and 0.08 mm in Figure 11) found in the tolerance frame.

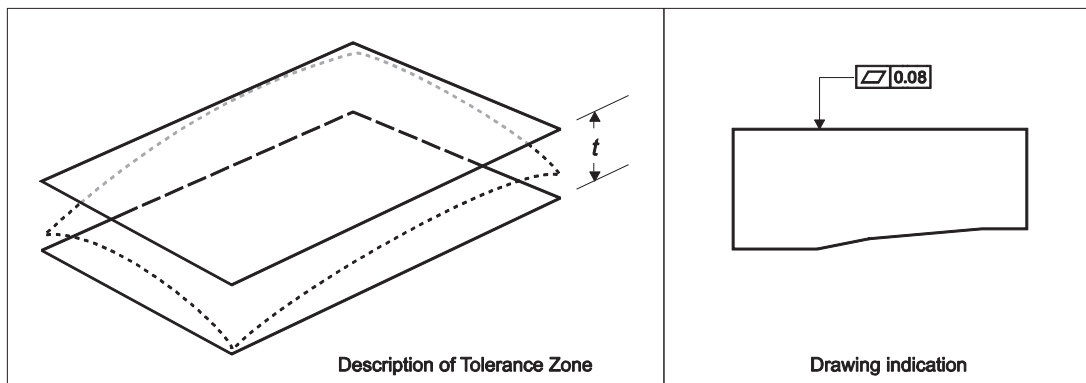


Figure 11: ISO defines flatness tolerance specification.

While such a definition of flatness expresses a particular aspect of a designer's requirement, it does not directly provide a prescriptive method to inspect and verify a given manufactured part. For this, we make the additional observation (which can be rigorously

proved) that this specification is completely equivalent to stating that the *width* of the tolerated feature should not exceed the specified tolerance value. Mathematically, the width of a set of points (whether they are continuous or discrete) is the shortest distance between two parallel planes between which the whole set is contained. In the two-dimensional case, Figure 12 illustrates how the width is realized for a discrete set of points. Also shown is the convex hull of the set of points. The computation of width is aided considerably by the observation that the width of a set of points in a plane is always realized between an edge-vertex antipodal pair on the convex hull. The center line shown in Figure 12, halfway between the two parallel lines that realize the width, is called the minimum zone line (MZline). So, it can be said that the width is the peak-to-valley distance of the points measured perpendicular to the MZline.

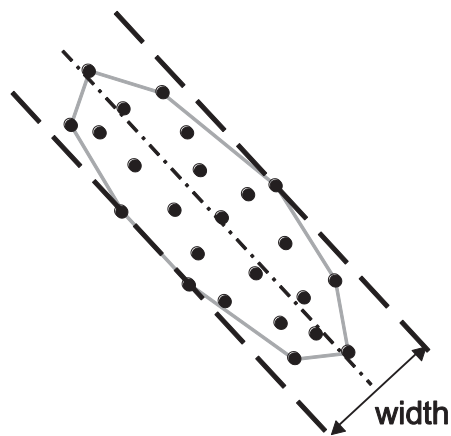


Figure 12: Width of a set of points in a plane.

It is reasonable to ask how complex can such a computation of width get. To be fair, we should compare this with the computation of total least-squares fitting described earlier in Section 2.2. Complete code that does the computation is given below:

```

function [w, p, v, i1, i2, i3] = Width2d(P)
%-----
% Width2d Computes the width of a set of points in 2D
%       It also computes the MZline in 2D
%-----
% Input
% P   Array of x and y-coordinates of input points
%     Size m x 2
%     Caution: If the input points are collinear, or almost
%               collinear, the code will fail due to problem
%               with convex hull computation.
%
% Output
% w   Width of the set
%     Size 1 X 1, scalar
% p   A point on the MZline
%     Size 2 X 1
% v   A unit vector along the MZline
%     Size 2 x 1
% i1, i2 Indices of endpoints of antipodal edge that realizes the width
%        Integer
% i3   Index of the antipodal vertex that realizes the width
%        Integer
%-----
% Error checking for number of input points
m = size(P, 1);
if m < 3
    error('Width requires at least 3 input points in 2D')
end
%
H = convhull(P(:,1), P(:,2));           % Compute the convex hull
nh = size(H,1) - 1;                    % Number of hull edges or vertices
av = zeros(nh, 1);                     % initialize antipodal vertex array
h = zeros(nh, 1);                       % initialize height array
%
% Compute the antipodal edge-vertex pairs
% av is the vector that stores the antipodal vertices
% h is the vector that stores the height of the triangle formed
%   by each antipodal edge-vertex pair.
%
for i = 1:nh                            % loop over each edge of the convex hull
    a = [P(H(i),1) P(H(i),2)]';        % first vertex of the edge
    b = [P(H(i+1),1) P(H(i+1),2)]';    % second vertex of the edge
    area2 = zeros(nh,1);                % initialize the 'twice-the-area' array
    for j = 1:nh                          % loop over each vertex of the convex hull
        c = [P(H(j),1) P(H(j),2)]';
        area2(j) = abs(det([a b c; 1 1 1]));
    end % for j
    [a2max, k] = max(area2);
    av(i) = k;
    h(i) = a2max/norm(a-b);
end % for i
%
% Now compute the width
%
[w, i] = min(h);
%
% Identify the antipodal edge-vertex pair that realizes the width
%
i1 = H(i); i2 = H(i+1); i3 = H(av(i));
a = [P(i1,1) P(i1,2)]';                % Starting point of antipodal edge
b = [P(i2,1) P(i2,2)]';                % Ending point of antipodal edge
c = [P(i3,1) P(i3,2)]';                % Antipodal vertex

```

```

%
% Compute a point on the MZline and a unit vector along the MZline
%
p = 0.5*(a + c);           % MZline bisects any line-segment from
                           % the antipodal edge to the antipodal vertex
v = (b - a)/norm(b-a);    % MZline is parallel to the antipodal edge
% END of Width2d

```

The code above uses the *convhull* function to compute the convex hull. It is not the most efficient code nor is it the shortest, but it serves to illustrate the point that the width can be computed in two-dimensional cases by completely solving the combinatorial optimization problem. In other words, the global minimum can be found for this nonlinear optimization problem using just a page-long code. The corresponding width computation problem for a set of points in three-dimensional space can be similarly solved using convex hull in three-dimensions. For the sake of completeness we reproduce the corresponding code below:

```

function [w, p, v] = Width3d(P)
%-----
% Width3d Computes the width of a set of points in 3D,
%           by solving several 2D width problems.
%-----
% Input
% P   Array of x, y and z-coordinates of input points
%     Size m x 3
%     Caution: If the input points are coplanar, or almost
%             coplanar, the code will fail due to problem
%             with convex hull computation.
%
% Output
% w   Width of the set
%     Size 1 X 1, scalar
% p   A point on the MZplane
%     Size 3 X 1
% v   A unit vector normal to the MZplane
%     Size 3 x 1
%-----
% Error checking for number of input points
m = size(P, 1);
if m < 4
    error('Width requires at least 4 input points in 3D')
end
%
H = convhulln(P);           % Compute the 3D convex hull
nf = size(H,1);           % Number of hull faces
Haug = [H, H(:,1)];       % Augment the face array
V = union(union(H(:,1),H(:,2)), H(:,3)); % array of vertices of convex hull.
                                           % Contains only unique indices.
W = zeros(nf, 3);         % Initialize the width array
%
% Compute the 2D widths by projecting the hull vertices perpendicular to every edge
%
for i = 1:nf               % loop over each face of the convex hull
    a = P(H(i,1),:);      % first vertex of the face
    b = P(H(i,2),:);      % second vertex of the face
    c = P(H(i,3),:);      % third vertex of the face
    vec = cross((b-a), (c-b)); % compute a vector normal to the face
    nvec = vec/norm(vec);  % it is the unit vector normal to the face
    for j = 1:3           % loop over each edge of the face

```



```

    v1 = P(Haug(i,j),:); % starting vertex of the edge
    v2 = P(Haug(i,j+1),:); % ending vertex of the edge
    evec = (v2 - v1)/norm(v2 - v1); % unit vector along the edge
    nvec = cross(evec, nvec); % nvec, yvec, and evec form a right-handed
    % unit triad
    Q = P(V,:)*[nvec, yvec, evec]; % project hull vertices perpendicular
    % to the edge
    w2 = Width2d(Q); % 2D width of the projected points
    W(i,j) = w2; % Store the 2D width in the W array
end % for j
end % for i
%
% Now compute the width
%
[wmin, K] = min(W); % Find maximum of each column and their
% row indices
[w, k] = min(wmin); % w is the width
%
% Now compute the MZplane
% Note that we will repeat several steps that were computed within the
% 'for' loops above.
%
irow = K(k); icol = k; % index into the right face and
% starting vertex
a = P(H(irow,1),:); % first vertex of the face
b = P(H(irow,2),:); % second vertex of the face
c = P(H(irow,3),:); % third vertex of the face
vec = cross((b-a), (c-b)); % compute a vector normal to the face
nvec = vec/norm(vec); % it is the unit vector normal to the face
v1 = P(Haug(irow,icol),:); % starting vertex of the edge
v2 = P(Haug(irow,icol+1),:); % ending vertex of the edge
evec = (v2 - v1)/norm(v2 - v1); % unit vector along the edge
nvec = cross(evec, nvec); % nvec, yvec, and evec form a right-handed
% unit triad
Q = P(V,:)*[nvec, yvec, evec]; % project hull vertices perpendicular to
% the edge
[w2, p2, v2] = Width2d(Q); % 2D width of the projected points
% Get a 3D point from p2
p3 = [p2(1); p2(2) ; 0.5*(max(Q(:,3)) + min(Q(:,3)))];
v3 = [-v2(2); v2(1) ; 0]; % Get a 3D vector normal to the MZplane
p = [nvec, yvec, evec]*p3; % Transform the point back to the
% original system
v = [nvec, yvec, evec]*v3; % Transform the vector back to the
% original system
% END of Width3d

```

The code shown above uses the *convhulln* function and the *Width2d* function presented earlier. Note that the width can be realized between a face-vertex or an edge-edge (where both edges are skew) antipodal pair of the convex hull, and the code checks for both cases. It is not the most efficient code, but it does find the global solution to the combinatorial optimization problem, and it can be understood and maintained by expending only a moderate amount of effort. Once the MZplane is found, we can consider the width to be the peak-to-valley distance of the points measured perpendicular to the MZplane.

A few words of analysis about the width computation in two- and three-dimensional spaces are worth making at this stage. Note that width is the *normative* definition for straightness (for a curve in a plane) and flatness, in the sense that this is completely equivalent to what the ASME and ISO standards mean when they define straightness and flatness tolerances today. It is

clear that the width computations are considerably more involved than the least-squares computations for lines and planes described in earlier sections, even though we have demonstrated just above that they can be reliably computed using some of the modern, popular computing tools.

In reality, many CMM software vendors just compute the least-squares plane and estimate the width as the peak-to-valley distance of the points measured perpendicular to the least-squares plane. It is well known that this is only an approximation, and it overestimates the width. Nevertheless, the computational convenience (coupled with robustness and uncertainty considerations, and a myriad of other reasons outlined in Section 2) overrides the normative requirements.

The reader may wonder how the width was measured before the era of CMMs and ubiquitous computers. Figure 13 shows how the flatness can be checked by moving the part under inspection over the inspection surface plate, all the while maintaining contact between the tolerated surface and the surface plate. The full indicator movement of the dial indicator is taken to be a measure of the width. This may appear to be a crude way to verify flatness – and it has several other drawbacks – but it is a practice that can be carried out even in modest shop floor environments and is widely prevalent even today. *In fact, we assert that the existence of this type of open setup to check for flatness is the main reason why its definition in ASME and ISO standards has stayed this way thus far.*

We observe an interesting dichotomy here. While the width (required for flatness verification) is harder to compute, it is easily measured – albeit crudely – by commonly available, relatively cheap metrological apparatus. Just a full indicator movement will do. On the other hand, least-squares fitting is relatively easy to compute, but it has no simple open setup arrangement, such as a full indicator movement, to measure according to it; we invariably need a computer to do the calculations.

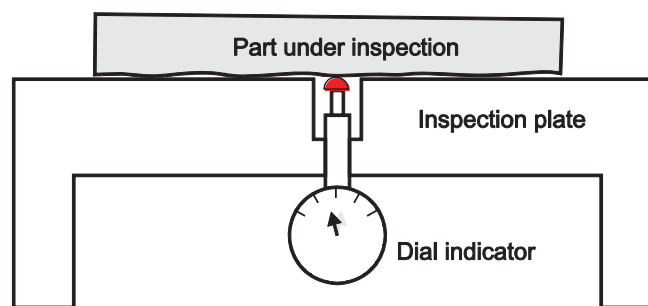


Figure 13: An open setup to check for flatness using a surface plate and a dial indicator.

This brief encounter with flatness tolerance is only a small example of a much wider scope covered by current ASME and ISO standards. They all rely almost exclusively on zone-based semantics for their geometric tolerance specifications. The least-squares fitting is not in the normative scope of the current versions of these standards; the least-squares fitting is popularly used more as a computational convenience. (Even in some of the commercial software that

comply with zone-based semantics, the initial guess for an iterative technique to solve the optimization problem happens to be a least-squares solution.)

All this is about to change because ISO is preparing to issue a whole series of standards that expand the scope of geometric tolerancing language beyond the zone-based semantics. And they will place least-squares fitting on a firm normative footing. This recent resurgence, and its implication for physical and computational coordinate metrology, are addressed in the next section.

5 The continuous least squares problem definitions, convergence, and weighted least-squares fitting

As noted in Section 2.10, the semantics presented in the last columns of Tables 1 and 2 using the English language text can be augmented by figures to make them a bit clearer, but they will still remain ambiguous. We will address this issue in some detail now. We will provide mathematical semantics to remove any ambiguity in these definitions, and to guide further verification processes such as sampling and designing fitting algorithms. We illustrate our approach using three examples: (1) fitting a total least-squares straight line to a curve in a plane, (2) fitting a total least-squares plane to a surface patch in space, and (3) fitting two parallel planes to two surface patches in space.

We start with continuous sets of points in the form of curves and surfaces, and define the optimization problems to fit straight lines and planes to these continuous sets. A bounded curve is contained within a sphere of finite radius; we also demand that it has finite length (that is, we don't allow fractal curves). It can consist of one or finitely many curve segments or arcs; each segment or arc is path connected. Similarly, a bounded surface is contained within a sphere of finite radius; we also demand that it has finite area (that is, we don't allow fractal surfaces). It can consist of one or finitely many surface patches; each patch is path connected.

In Section 2.2 we considered the total least-squares fitting of a straight line to a finite, discrete set of points. We now consider fitting a total least-squares straight line to a curve in a plane. Referring to Figure 14, we pose the optimization problem as follows:

Tlsq2dLine: Given a bounded curve C in a plane, find the straight line L that minimizes $\int_C d^2(p, L) ds$.

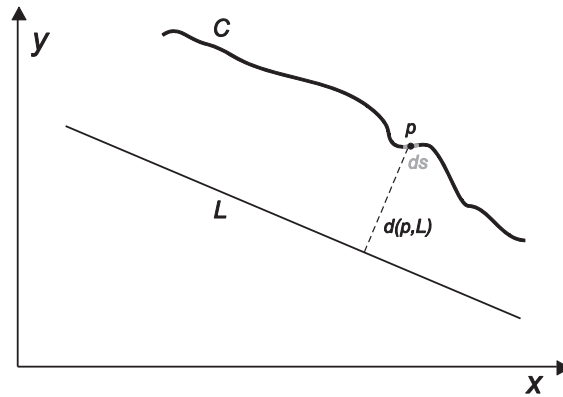


Figure 14: Fitting a straight line to a curve in a plane.

Here $d(p, L)$ denotes the perpendicular distance of a point p on curve C from the straight line L that will be fitted. Once such a straight line L has been found, the root-mean-square parameter for the bounded curve C is given by

$$\sqrt{\frac{\int_C d^2(p, L) ds}{\int_C ds}}. \quad (7)$$

Some important observations are worth noting here.

1. $\int_C ds$ is the length of the curve C .
2. If the curve consists of several segments (arcs), then the integrals can be evaluated over each segment and then summed.
3. It can be shown that the total least-squares line L passes through the centroid of C , and is aligned with one of the principal axes of C .
4. It can also be shown that the root-mean-square parameter is the ‘radius of gyration’ of the curve; it is equivalent to the ‘standard deviation’ in statistics.

Similarly, to fit a total least-squares plane to a surface patch in space, we pose the following optimization problem (with reference to Figure 15):

TlsqPlane: Given a bounded surface S , find the plane P that minimizes $\int_S d^2(p,P)ds$.

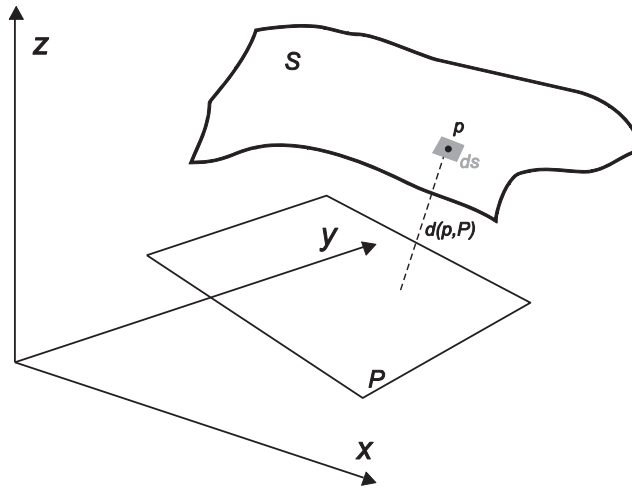


Figure 15: Fitting a plane to a surface patch.

Here $d(p,P)$ denotes the perpendicular distance of a point p on surface patch S from the plane P that will be fitted. Once such a plane P has been found, the root-mean-square parameter for the bounded surface S is given by

$$\sqrt{\frac{\int_S d^2(p,P)ds}{\int_S ds}}. \quad (8)$$

We again note that

1. $\int_S ds$ is the area of the surface patch.
2. If the surface consists of several patches, then the integrals can be evaluated over each patch and then summed.
3. It can be shown that the total least-squares plane P passes through the centroid of S , and is perpendicular to one of the principal axes of S .
4. The root-mean-square parameter is equivalent to the ‘standard deviation’ in statistics.

To address the issue of size as specified in ISO 14405-1, we need to define fitting problems such as the following (with corresponding illustration in Figure 16):

TlsqParallelPlanes: Given two bounded surfaces S_1 and S_2 , find two parallel planes P_1 and P_2 that minimize $\int_{S_1} d^2(p, P_1) ds + \int_{S_2} d^2(p, P_2) ds$.

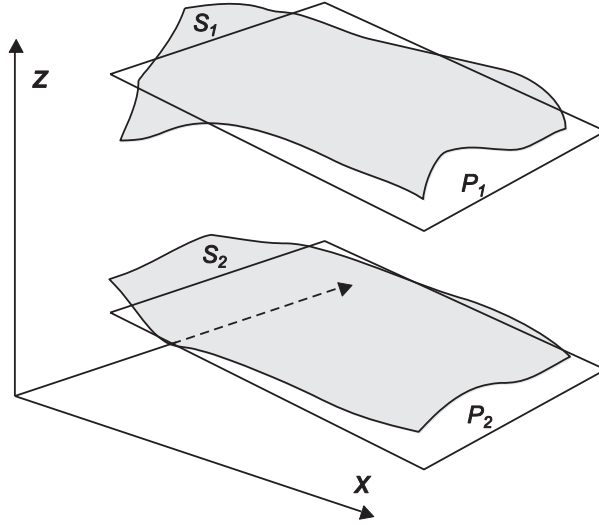


Figure 16: Fitting two parallel planes to two surface patches.

Once such parallel planes P_1 and P_2 have been found, the distance between them is the linear size (that has been defined with the least-squares criterion).

With such definitions in place, we can start designing algorithms to compute the fitted lines and planes. Here we run into an interesting issue related to sampling and discretization. The objective functions posed in `Tlsq2dLine`, `TlsqPlane` and `TlsqParallelPlanes` cannot, in general, be evaluated in closed form. So we resort to approximate evaluations of the integrals in those objective functions by sampling a discrete set of points on the continuous curves and surfaces. If we exercise some care in sampling these points *uniformly* on the curves and surfaces, we can simplify the subsequent calculations and use most of the readily available algorithms and software mentioned in earlier sections.

For example, in fitting a straight line to a curve in `Tlsq2dLine`, we divide the curve into uniform arc length interval of Δs and approximate the objective function as

$$\int_C d^2(p, L) ds \approx \sum_{i=1}^m \{d^2(p_i, L)\} \cdot \Delta s = \Delta s \cdot \sum_{i=1}^m d^2(p_i, L), \quad (9)$$

where p_i are the m sampled points, one in each interval. Since Δs can be treated as a constant within the optimization problem, the approximate objective function to minimize is $\sum_{i=1}^m d^2(p_i, L)$. It is worth noting that this approximate objective function is a continuous and smooth function of the parameters of the fitted line L , thus admitting good solutions to be found.

Similarly, in fitting planes to surface patches in `TlsqPlane`, we can sample points on a surface patch after dividing up the patch into uniform areas of ΔA and approximate the objective function as

$$\int_S d^2(p, P) ds \approx \sum_{i=1}^m \{d^2(p_i, P)\} \cdot \Delta A = \Delta A \cdot \sum_{i=1}^m d^2(p_i, P), \quad (10)$$

where p_i are the m sampled points, one in each subdivision. Since ΔA can be treated as a constant within the optimization problem, the approximate objective function to minimize is $\sum_{i=1}^m d^2(p_i, P)$. Again, it is worth noting that this approximate objective function is a continuous and smooth function of the parameters of the fitted plane P , thus enabling good solutions to be found.

We observe that almost all of the efforts expended over the past two decades in total least-squares fitting of straight lines and planes have used these approximate objective functions as the starting point to find efficient and robust solutions. They have made a subtle assumption about the uniformity of sampling, without explicitly cautioning the users of its importance.

Once the total least-squares fitting line is computed, an approximation for the root-mean-square parameter can be found for the curve C as

$$\sqrt{\frac{\int_C d^2(p, L) ds}{\int_C ds}} \approx \sqrt{\frac{\sum_{i=1}^m \{d^2(p_i, L)\} \cdot \Delta s}{\sum_{i=1}^m \Delta s}} = \sqrt{\frac{\sum_{i=1}^m d^2(p_i, L)}{m}}. \quad (11)$$

Similarly, once the total least-squares fitting plane is computed, an approximation for the root-mean-square parameter can be found for the surface S as

$$\sqrt{\frac{\int_S d^2(p, P) ds}{\int_S ds}} \approx \sqrt{\frac{\sum_{i=1}^m \{d^2(p_i, P)\} \cdot \Delta A}{\sum_{i=1}^m \Delta A}} = \sqrt{\frac{\sum_{i=1}^m d^2(p_i, P)}{m}}. \quad (12)$$

This brings us now to the subject of convergence. The topic of total least-squares fitting of lines and planes to discrete sets of points presented in Section 2.2 is well developed, and the related algorithms have been implemented and tested over the past two decades. It is only natural to ask if these solutions will converge to the lines and planes – and to the root-mean-square parameters – defined for continuous sets of points, as we sample more and more points on the curves and surfaces.

We can show that uniform sampling when refined uniformly will lead to the desired convergence. We cannot guarantee convergence to the right answer otherwise. It is important to note that uniform sampling should be observed across all segments or patches involved, and not just within one. Figure 17 illustrates what we mean by uniform and non-uniform sampling and their refinement.

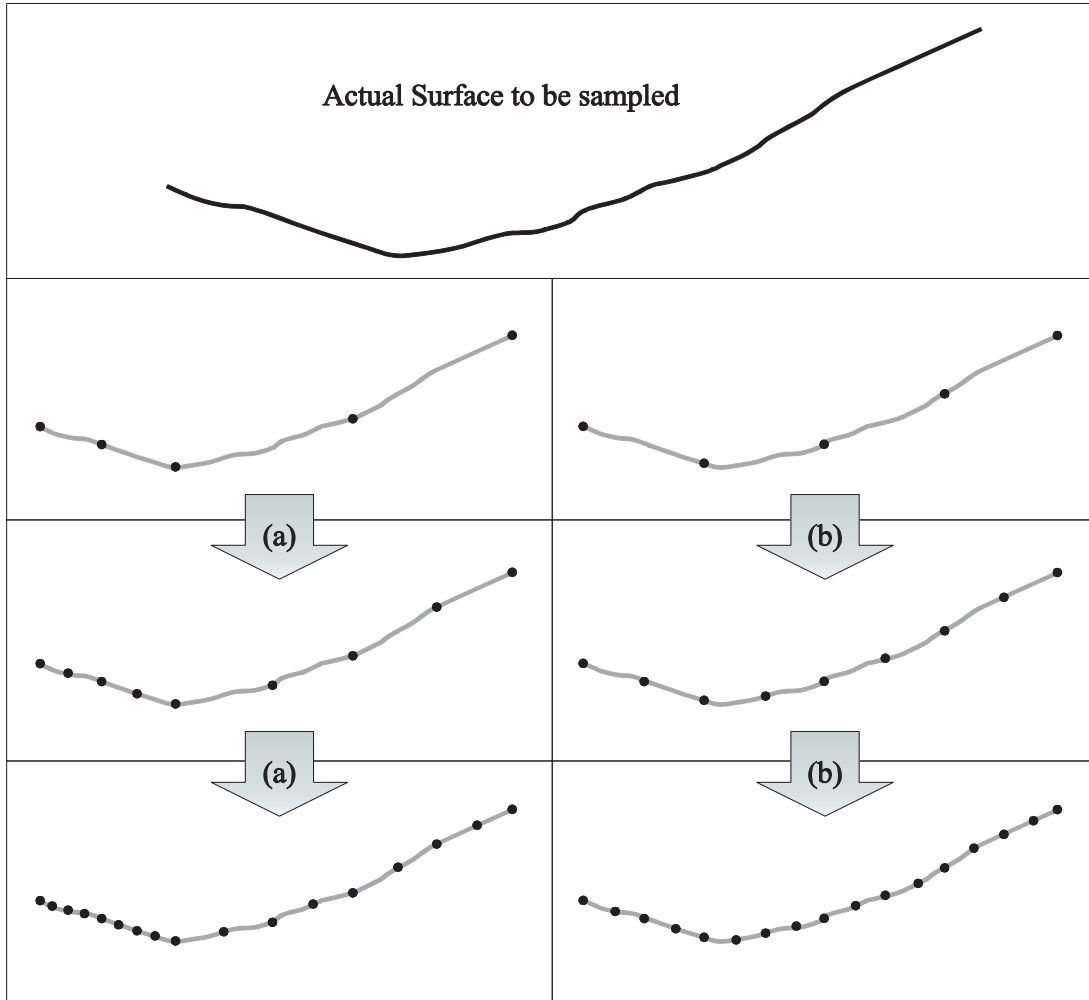


Figure 17: (a) Example of non-uniform sampling and its refinement, (b) Example of uniform sampling and its refinement.

If, for some reason, we are not able to sample points uniformly, then we may have to resort to *weighted* total least-squares techniques to solve the discrete version of the problem. (Weighting may also be used if there are different uncertainties associated with different sampled points [17].) These weights arise naturally from the fact that Δs and ΔA are not constants and therefore cannot be taken out of the summation. So we have to minimize $\sum_{i=1}^m [\{d^2(p_i, L)\} \cdot \Delta s_i]$ for `Tlsq2dLine`, where Δs_i 's are treated as the weights. Similarly, we have to minimize $\sum_{i=1}^m [\{d^2(p_i, P)\} \cdot \Delta A_i]$ for `TlsqPlane`, where ΔA_i 's are treated as the weights. These weights will also influence the calculation of the root-mean-square parameter, after the total least-squares fits have been found.

The need for uniform discretization or weighted total least-squares is best illustrated using the simple example shown in Figure 18. Here a simple, symmetric v-shaped curve requires a total least-squares fitting of a straight line. The correct solution is a horizontal line positioned

halfway between the top and bottom of the v-shaped curve, as shown in Figure 18 using long dashes and dots. However, if we discretize the curve non-uniformly by choosing twice as many points on the left arm as on the right, as shown in Figure 18, then the resulting total least-squares fit will be tilted more towards the left if we do not use any weights. This will continue to be the case even in the limit as n tends to infinity, thus converging to the wrong answer. (Note that this is *not* the case in the width computations associated with current zone-based tolerance specifications.) So, an unscrupulous inspector can make the fitted line and the root-mean-square value to be almost anything he wants within a rather wide interval and show that he has converged to them by increasing the sample density. A weighted total least-squares will not lead us to such wrong answers.

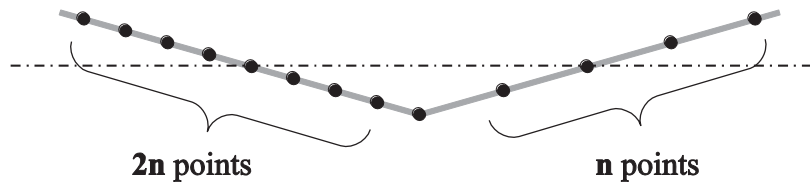


Figure 18: Convergence using non-uniform discretization

The examples illustrated in this section are just the tip of the iceberg. Several other problems of similar nature arise when least-squares fitting is made normative. Both linear and nonlinear total least-squares fitting problems have to be defined for continuous curves and surface patches, and their discrete versions have to be developed as weighted total least-squares problems. We believe the current total least-squares fitting algorithms and the corresponding software implementations can be extended to cover the weighted total least-squares. But these have to be rigorously tested to ensure that they conform to the emerging normative specifications.

6 Summary

A resurgence of least-squares fitting is under way with the new specification methods that are being made available to designers. The thrust of this paper has been to describe the wealth of experience and tools that are currently available to metrologists faced with these specifications, along with identifying some of the reasons for the popularity of the least-squares choice of fit objective. As the use of such specifications becomes commonplace, it is anticipated that further research and discussion will be needed to better understand the uncertainty present when a finite sample set is used to estimate the least-squares solution for the entire continuous surface. Thus the appeal of least-squares fitting will continue to endure for foreseeable future in research and practice of computational coordinate metrology.

Acknowledgement and Disclaimer

We extend thanks to several of our colleagues in the ASME and ISO standards committees who have generously contributed their time in the development of various technologies and standards presented in this paper.

Disclaimer: Certain commercial equipment, instruments, software, or materials are identified in this paper in order to specify the experimental procedure 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 materials or equipment identified are necessarily the best available for the purpose.

References

- [1] Srinivasan, V., 2005, "Elements of Computational Metrology," *DIMACS Book Series*, Vol. 67, R. Janardan, M. Smid, and D. Dutta, eds. Computer Aided Design and Manufacturing, American Mathematical Society, R.I., pp. 79-116.
- [2] Srinivasan, V., 2007, "Computational Metrology for the Design and Manufacture of Product Geometry: A Classification and Synthesis," *Transactions of the ASME, Journal of Computing and Information Science in Engineering*, **7**, pp. 3-9.
- [3] Walker, R., 1988, GIDEP Alert No. X1-A-88-01, Government-Industry Data Exchange Program, Aug. 22, 1988.
- [4] Stigler, S.M., 1990, *The History of Statistics: The Measurement of Uncertainty before 1900*, Belknap Press of Harvard University Press, Cambridge, MA.
- [5] Shakarji, C.M., 1998, "Least-squares Fitting Algorithms of the NIST Algorithm Testing System," *J. Res. Natl. Inst. Stand. Technol.*, **103**, pp.633-641.
- [6] Forbes, A., 1991, *Least-squares Best-fit Geometric Elements*, NPL Report, DITC 140/89, National Physical Laboratory, U.K.
- [7] Nievergelt, Y., 1994, "Total Least Squares: State-of-the-Art Regression in Numerical Analysis," *SIAM Review*, **36**, No. 2, pp. 258-264.
- [8] O'Rourke, J., 2001, *Computational Geometry in C*, 2nd Edition, Cambridge University Press, New York, NY.
- [9] Pottmann, H., Hofer, M., Odehnl, B., and Wallner, J., 2004, "Line Geometry for 3D Shape Understanding and Reconstruction," *European Conference on Computer Vision, Lecture Notes in Computer Science*, Springer, Berlin, pp. 297-309.
- [10] Gelfand, N. and Guibas, L. J., 2004, "Shape Segmentation Using Local Slippage Analysis," *Proceedings of the Eurographics Symposium on Geometry Processing*, pp. 214-223.
- [11] Shakarji, C.M., 2003, "Evaluation of One- and Two-Sided Geometric Fitting Algorithms in Industrial Software," *Proceedings of the American Society of Precision Engineering Annual Meeting*, pp 100-105.
- [12] Jiang, X., Cooper, P., and Scott, P.J., *Proc. R. Soc. A*, "Free-form Surface Filtering Using the Diffusion Equation," Published online 9 September, 2010 (doi: 10.1098/rspa.2010.0307)
- [13] Jiang, X., Zhang, X., and Scott, P., 2010, "Template Matching of Freeform Surfaces Based on Orthogonal Distance Fitting for Precision Metrology," *Meas. Sci. Technol.* **21**, (10 pp).
- [14] Edgeworth, R., and Wilhelm, R.G., 1996, "Uncertainty Management for CMM Probe Sampling of Complex Surfaces," *ASME Mfg. Sci. and Engr.*, MED-4: 511-518.

- [15] Edgeworth, R., and Wilhelm, R.G., 1999, "Adaptive Sampling for Coordinate Metrology," *Precision Engineering*, **23**, pp. 144–154
- [16] Wang, J., Jiang, X., and Blunt, L., 2010, "Simulation of Adaptive Sampling in Profile Measurement for Structured Surfaces," Computing and Engineering Researchers' Conference, University of Huddersfield, U.K.
- [17] Krystek, M., and Anton, M., 2007, "A Weighted Total Least-squares Algorithm for Fitting a Straight Line," *Meas. Sci. Technol.* **18** , pp. 3438–3442