National Physical Laboratory

Validation of Software for Dimensional Metrology

A B Forbes

February 1994

The National Physical Laboratory is an Executive Agency of the Department of Trade and Industry

NPL Report DITC 225/94

February 1994

Validation of Software for Dimensional Metrology[(1)]

A B Forbes
Division of Information Technology and Computing
National Physical Laboratory
Teddington
Middlesex
United Kingdom
TW11 0LW

ABSTRACT

In this paper we discuss three questions relating to the use of reference data sets and reference results in black box tests for validating assessment software: i) how to generate data and results, ii) how to represent solutions in a stable way and iii) how to compare test results and reference results. We describe a general method for generating data and results which goes some way to addressing all three problems and illustrate the concepts introduced for least squares form assessment and theodolite triangulation.

(1)Paper presented at the conference on Laser Dimensional Metrology in Brighton, October 1993. The conference proceedings are to be published by the Society of Photo-Optical Instrumentation Engineers (SPIE).

ISSN 0262-5369

National Physical Laboratory
Teddington, Middlesex, United Kingdom, TW11 0LW

Approved on behalf of Chief Executive, NPL,
by Mr A J Marks, Head, Division of Information Technology and Computing

CONTENTS

1 INTRODUCTION

The need for the validation of assessment software in instruments – the software which takes the raw data and computes derived parameters such as flatness, roundness, etc., – is becoming more apparent. Users want to know that the software is giving correct results. Suppliers want to be able to convince their customers (and themselves) that their software is reliable. Quality regimes such as ISO 9000 require that software is tested. The establishment of traceability of measurements to National Standards for length also implicitly requires that assessment software is validated [3, 5]. In response to this need there is now increasing activity towards the establishment of software conformance testing services, software certification, etc. However it is important that as this activity develops we do not lose sight of the overall goals of improving quality and reducing total costs. Equally, it is important not to underestimate how difficult it can be to design tests of numerical software which yield results that have some value.

The National Physical Laboratory (NPL) is interested in assessment software validation for two main reasons. Firstly NPL is responsible for the maintenance of National Standards to which traceability must be established. Secondly, NPL has for many decades been involved in providing and promoting reliable computational methods. In recent years, we have taken a particular interest in validation problems associated with assessment software in coordinate metrology within a programme in support of the UK's National Measurement System.

2 ASSESSMENT SOFTWARE AND VALIDATION

We suppose that the overall aim of an experiment is to determine one or more parameters accurately. In general, the determination of these parameters is a two-stage process. In the first stage, the measuring instrument physically reacts with the measurand and at some point sensor signals are transformed into digital data points $\mathbf{x}_i$, $i \in I$. At the second stage, *assessment software* analyses the data points $\mathbf{x}_i$ to produce best estimates of the parameters. A typical example is the determination of the circularity of an artefact by a roundness measuring machine. The coordinates of points on the artefact are first recorded and then assessment software is applied to the data points to determine a best estimate of the form deviation.

The first stage may involve the application of *calibration software* to the raw data to produce a the 'corrected' version of the data. This type of software is generally instrument-specific, hidden from the user and not immediately amenable to validation. This aspect can be worrisome, especially at standards laboratories such as NPL where the need to establish a standard or to show direct traceability to a standard strongly inhibits the use of 'black' boxes. However there are clearly practical limits to the degree of transparency that can be sensibly required of an instrument and some opaqueness can be accepted if there are adequate checks on the complete system through the use of reference artefacts calibrated using transparent methods. Assessment software, or at least what it aims to compute, is in general less instrument-specific, more public and hence more readily validated and it is assessment software which is our main concern here.

Computational aim. The starting point for any validation process in an explicit and formal statement of exactly what the software is required to compute, i.e., its *computational aim.* For most assessment software the computational aim can usually be defined as determining a model M^* in a space $\mathcal{M}$ of models which is optimal with respect to some objective criterion, e.g. is a minimiser of an error function $E(X, M)$ for a representative data set X.

Once the computational aim has been established the validation process has to measure the extent to which the software achieves the stated aim. In particular the validation test should ideally expose the use of i) approximate methods, ii) unstable numerical methods and iii) poor optimisation algorithms. The most practical way of testing assessment software is to run the software on test data sets and then analyse the results output. However this method poses three fundamental problems i) how to produce data sets for which the corresponding solutions are known accurately, ii) how to represent the 'true' solutions in a numerically stable way and iii) how to compare the results output by the software under test with the 'true' results.

3 REFERENCE DATA AND REFERENCE RESULTS

Forward and inverse error. We describe below a general method for generating data sets and corresponding results simultaneously. However it has to be recognised that there are limits on what accuracies can be achieved in these data/results pairs. Our initial goal will be to produce a data set X and a vector of reference results $\mathbf{r}^*(X)$ such that difference between $\mathbf{r}^*$ and the mathematically exact results $\mathbf{r}^{\ddagger}$ is small. How small? In numerical analysis there are two basic methods of measuring the accuracy of computed results, the forward error and inverse (or backward) error [7]. Given data X and a statistic $\mu(X)$ determined uniquely by the data from computation, the forward error between a computed estimate $\hat{\mu}$ and the mathematically exact value $\mu^{\ddagger}$ is given in terms of a bound on $|\hat{\mu} - \mu^{\ddagger}|$. The inverse error is given in terms of a bound on the size of the perturbation Δ of the data X such that the computed solution is the mathematically *exact* solution for the perturbed data:

$$\hat{\mu} = \mu^{\ddagger}(X + \Delta);$$

thus if $|\Delta|$ is small we can say that $\hat{\mu}$ is a solution to a nearby problem. We shall show that using null space methods it is possible to generate reference data and corresponding reference results which have a small inverse error. This implies that for well conditioned problems the reference results will also have a small forward error.

Choice of coordinate system. One fact that has to be taken into account is that the choice of coordinate system with which to represent measured or synthesised data will itself have an effect on the accuracy of computed statistics derived from the data. Consider the determination of the flatness of a table 1 metre square which deviates from a perfect plane by no more than 10 micrometres. If we choose a coordinate system such that the table surface coincides with the xy plane then it is possible to calculate distances etc. to full working precision. If the coordinate system is such that the normal to the plane does not align closely with an axis, then up to five decimal places can be lost in the calculations.

For example the distance d from a point (x, y, z) to a plane passing though (x_0, y_0, z_0) with normal vector (a, b, c) is given by

$$d = a(x - x_0) + b(y - y_0) + c(z - z_0).$$

For the aligned coordinate system we expect a, b and $(z - z_0)$ to be small so that the three summands each will be of the order of 10^{-5}. Thus there will a small relative error in the distance calculation. If the normal vector is not close to an axis then at least two of the three summands could be of the order of 1 and a large cancellation error is possible.

Null space methods. Given a computational problem related to a space of models $\mathcal{M}$ and data X it is usually possible to derive a set of sufficiency conditions $S(X, M)$, involving the X and M which when satisfied by M^* will guarantee that M^* is at least a local solution. Assessment software addresses the problem: given data X find a model M^* which satisfies $S(X, M^*)$. The reference data generation problem is: given a 'solution' M^* find data X such that the sufficiency conditions $S(X, M^*)$ are satisfied. The null space method addresses this problem by firstly generating points X^* which trivially satisfy these conditions and then finding perturbations P in such at way that the conditions $S(X^* + P, M^*)$ remain satisfied. We will illustrate this method firstly for providing reference data and results for least squares form assessment.

Given a set of data points $X = \{\mathbf{x}_i : i \in I\}$ the least squares best fit element $C^*(X)$ is the one which minimises the sum of squares error function

$$E(X, C) = \sum_{i \in I} d^2(\mathbf{x}_i, C), \tag{1}$$

where $d(\mathbf{x}, C)$ is the distance from a point x to the element. We can generate reference data and results using the null space method as follows. We first select a parametrization $\mathbf{a} \longmapsto C(\mathbf{a})$ of the element and a set of 'solution' parameters $\mathbf{a}^*$. For example we can parametrize a circle using the centre coordinates (a, b) and radius r_0. Setting $a = b = 0$ and $r_0 = 1$ specifies the unit circle. Secondly, we generate a set of points $X^* = \{\mathbf{x}_i^* : i \in I\}$ lying exactly on $C(\mathbf{a}^*)$ and also compute the associated Jacobian matrix J^*, the matrix whose ijth element is the partial derivative of the distance function $d(\mathbf{x}_i^*, \mathbf{a}) = d(\mathbf{x}_i^*, C(\mathbf{a}))$ with respect to the parameter a_j. Finally we compute a perturbed set of the form $X = \{\mathbf{x}_i : \mathbf{x}_i = \mathbf{x}_i^* + d_i \mathbf{n}_i^*\}$ where $\mathbf{n}_i^*$ in the unit normal to the element C^* at $\mathbf{x}_i^*$ and the residuals d_i are chosen so that $(J^*)^{\mathrm{T}}\mathbf{d} = 0$; in other words the vector of residuals lies in the *null space* of the transpose of the Jacobian matrix. For this perturbed data set we can show that C^* remains a minimum of the error function $E(X, C)$ (if the calculations are done in exact arithmetic) [4].

In a numerical implementation of the null space method, rounding errors will occur in i) determining the exact data set X^*, ii) determining the perturbations P and iii) forming $X = X^* + P$. Thus the computed data set X will differ from the required data set by a perturbation Δ, so that the solution module M^* defined at the start of the process will be the exact solution to the data set $X + \Delta$. In order to produce accurate reference results it is necessary to control the size of these rounding errors at each step, for example in generating the data X^*. At first sight, this step in the null space method appears to be the most straightforward. For example, given a circle C^* specified by centre coordinates (a, b) and radius r_0 then a choice of angles $\theta = \{\theta : i \in I\}$ will generate data points

$$X = \{(x_i, y_i) = (a + r_0 \cos\theta_i, b + r_0 \sin\theta_i)\}$$

nominally lying on C^*. However this method can give rise to unnecessarily large rounding error if the aim is to generate points lying on a small arc of a circle. For example, suppose the centre of the circle is to be at $(0, b)$, $b > 0$, and we wish to generate points on an arc passing through the points $(-1, 0)$ and $(1, 0)$. If b and therefore r_0 are large, there will be cancellation errors in calculating the y coordinates.

An alternative approach, for this example, is to solve for y as a function of x in the interval $[-1, 1]$ from the equation

$$x^2 + (y - b)^2 = r_0^2 = 1 + b^2,$$

which gives

$$y = b - \sqrt{(1 + b^2 - x^2)}.$$

For b large evaluating this expression as it stands will also lead to large cancellation errors. However, we can rearrange the right hand side to give

$$y = \frac{(x-1)(x+1)}{b + \sqrt{(b^2 - (x-1)(x+1))}},$$

which represents a numerically stable scheme for calculating y given x. If similar care is taken at subsequent stages of the data generation, it is possible to provide data sets which are accurate close to full working precision.

An important aspect of the null space method is that while the sufficiency conditions are usually over-determined with respect to the models M they are usually under-determined with respect to the data points X. For this reason, it is much easier to solve the data generation problem.

Example: triangulation. Suppose there are n theodolites T_k at known positions $\mathbf{p}_k$ which record azimuth angles θ_k and angles of elevation ϕ_k corresponding to a target $\mathbf{x}$. The measurement aim is provide an accurate estimate of $\mathbf{x}$ from the angle measurements. If we assume that the angle measurements have random error then an estimate of $\mathbf{x}$ is found by determining the minimal perturbation of the angles which allows the associated lines of sight to intersect at a common target. This leads to the following optimisation problem:

$$\min_{\theta_k^*, \phi_k^*} \sum_k \left\{ (\theta_k - \theta_k^*)^2 + (\phi_k - \phi_k^*)^2 \right\}, \tag{2}$$

subject to the constraint equations

$$D_k(\theta_k^*, \phi_k^*, \theta_{k+1}^*, \phi_{k+1}^*) = \det \begin{bmatrix} \mathbf{p}_k - \mathbf{p}_{k+1} \\ \mathbf{a}_k^* \\ \mathbf{a}_{k+1}^* \end{bmatrix} = 0, \quad k = 1, \ldots, n-1,$$

where

$$\mathbf{a}_k^* = (\cos\theta_k^* \cos\phi_k^*, \sin\theta_k^* \cos\phi_k^*, \sin\phi_k^*), \quad k = 1, \ldots, n.$$

The kth constraint equation ensures that the lines $\mathbf{p}_k + t_k \mathbf{a}_k^*$ and $\mathbf{p}_{k+1} + t_{k+1}\mathbf{a}_{k+1}^*$ intersect. This is a non-linearly constrained optimisation problem (with a sparsity structure in the constraints) which requires sophisticated optimisation software to solve (efficiently). However, the data generation problem is much simpler. We start with the target point $\mathbf{x}$ and points $\mathbf{p}_k$ and generate the corresponding exact angles θ^* and ϕ^*. We then determine perturbations δ_k and ε_k such that $\mathbf{x}$ remains a solution of (2) with $\theta_k = \theta_k^* + \delta_k$, $\phi_k = \phi_k^* + \varepsilon_k$, $k = 1, \ldots, n$.

Analysing the optimality conditions associated with (2) we can show given a set of multipliers μ_k, $k = 1, \ldots, n-1$, the perturbations satisfying

$$\begin{aligned}
\delta_1 &= \mu_1 \frac{\partial D_1}{\partial \theta_1^*}, \qquad \delta_n = \mu_{n-1} \frac{\partial D_{n-1}}{\partial \theta_n^*}, \\
\delta_k &= \mu_{k-1} \frac{\partial D_{k-1}}{\partial \theta_k^*} + \mu_k \frac{\partial D_k}{\partial \theta_k^*}, \quad k = 2, \ldots, n-1, \\
\varepsilon_1 &= \mu_1 \frac{\partial D_1}{\partial \phi_1^*}, \qquad \varepsilon_n = \mu_{n-1} \frac{\partial D_{n-1}}{\partial \phi_n^*}, \\
\varepsilon_k &= \mu_{k-1} \frac{\partial D_{k-1}}{\partial \phi_k^*} + \mu_k \frac{\partial D_k}{\partial \phi_k^*}, \quad k = 2, \ldots, n-1,
\end{aligned}$$

will define a test data set with x as the solution target, provided the perturbations are not too large. Thus the data generation reduces to calculating the gradient of the constraint equations D_k with respect to the angle parameters

3.1 REPRESENTATION OF REFERENCE SOLUTIONS.

The validation of software using data sets relies on the availability of *reference pairs* $\langle X, \mathbf{r}^*(X)\rangle$ of data sets and corresponding reference results. Furthermore, this information has to be stored on a computer with finite word length. This means that only in special circumstances will the reference results and data be exactly self-consistent. For example, for a general data set which is stored in finite precision and which represents points close to a circle it is almost certain that the least squares circle centre cannot be exactly represented in finite precision. Our aim is to produce reference results such that the the size of inconsistency is small and numerically stable in the sense that if the data X and the reference results $\mathbf{r}$ are perturbed by small amounts then the size of the inconsistency changes by a comparably small amount.

The most obvious way to represent the reference solution corresponding to a data set X is to specify the parameters $\mathbf{a}^*$ in some parametrization of the model. If the parametrization is well-conditioned *for the data set* X [1] then this way is perfectly acceptable. For example for data distributed approximately uniformly around a circle then the parameters specifying the centre coordinates (a, b) and radius r_0 are well-conditioned and the information $\langle X, a, b, r_0\rangle$ is stably consistent. On the other hand if the data points X are distributed in a small arc of a circle, the circle centre is poorly defined and a parametrization involving the centre coordinates will be poorly conditioned for this data set [6]. In this situation the information $\langle X, a, b, r_0\rangle$ is not stably consistent. Faced with these and other examples, either we try to find for each data set a parametrization which is well conditioned or we find other ways of specifying the solution.

For example, in element fitting we can choose to define an element by specifying a number of well chosen data points which lie exactly on the element. In particular if M^* is the best-fit element to a set of data X we can represent M^* by the set of points X^* on M^* closest to X. Alternatively we can specify M^* by the residual distances $d_i^* = \pm\|\mathbf{x}_i - \mathbf{x}_i^*\|$. In either case we can show that information pairs $\langle X, X^*\rangle$ and $\langle X, \mathbf{d}^*\rangle$ are stably consistent. Also X^* and $\mathbf{d}^*$ are intrinsic to the computational aim and do not depend on arbitrary choices of parametrization, etc. Furthermore, both arise naturally in the null space method for generating reference data.

3.2 COMPARING TEST RESULTS WITH REFERENCE RESULTS

Suppose we have reference data X with corresponding solution model M^* (generated by null space methods or otherwise) and that test software results specify model $\hat{M}$. How do we measure how far $\hat{M}$ is from M^*? One way is to parametrize the models $M = M(\mathbf{a})$ by parameters $\mathbf{a}$ with $M^* = M(\mathbf{a}^*)$ and $\hat{M} = M(\hat{\mathbf{a}})$ for some $\mathbf{a}^*$ and $\hat{\mathbf{a}}$. A norm $\|\mathbf{a}^* - \hat{\mathbf{a}}\|$ will then give a measure of the difference between M^* and $\hat{M}$. This we may term a *forward comparison*. Although forward comparisons have the appeal of directness, there are a number of reasons why they can be unsuitable. Firstly, reference results may be expressed in terms of one parametrization and the test results in terms of another. Secondly, parametrization should be seen not as intrinsic to the computational aim but as part of the algorithm design. (Note that the computational aim defined above for a least squares best fit element makes no mention of a parametrization.) Thirdly, as discussed above forward comparisons take no direct account of the well-posedness or *condition* of the problem. There is no natural scaling associated with forward comparisons so that given a measure of the difference between computed parameters and reference parameters it can be difficult to say if it represents an acceptable difference without further information and analysis.

A method of comparison – *inverse comparison* – which in some ways is more useful can be based on the ideas relating to inverse error. If we show that a computed model $\hat{M}$ relating to $\mathcal{M}$ and X is the *exact* solution for a perturbed set of data $X + \Delta$ then the norm $||\Delta||$ of such a (minimal) perturbation can then give a measure of the difference between $\hat{M}$ and M^*. The advantages of this method are that i) the difference is expressed in terms of the data and can be compared directly with the estimated measurement error so that in this sense the measure has a natural scaling and ii) the size of the perturbation is independent of the intrinsic conditioning of the problem and the comparison can therefore give information about the behaviour of the software which is not convolved with the conditioning of the problem.

Null space methods can also be used to provide inverse comparison estimates. For example, suppose a least squares element fitting module computes element $\hat{M}$ to a set of data $X = \{\mathbf{x}_i\}$ along with residuals $\hat{\mathbf{d}}$ and that the reference solution M^* is specified by residuals $\mathbf{d}^*$. For each point $\mathbf{x}_i$ we can calculate the point $\hat{\mathbf{x}}_i$ closest to $\mathbf{x}_i$ on $\hat{M}$ and the corresponding normal vector $\hat{\mathbf{n}}_i$ at $\hat{\mathbf{x}}_i$. $\hat{M}$ is a trivial solution for the points $\hat{X} = \{\hat{\mathbf{x}}_i\}$ and we can generate a null-space perturbations of the form $\hat{\mathbf{x}}_i + d_i\hat{\mathbf{n}}_i$. Of these, we can chose the perturbation specified by residuals $\hat{\mathbf{d}}^*$ which is closest to the original data set X in the least squares sense. We note that

$$||\hat{\mathbf{x}}_i + d_i\hat{\mathbf{n}} - \mathbf{x}_i|| = ||\hat{\mathbf{x}}_i + d_i\hat{\mathbf{n}} - (\hat{\mathbf{x}}_i + \hat{d}_i\hat{\mathbf{n}})|| = |d_i - \hat{d}_i|,$$

so that determining this perturbation amounts to finding the null space vector $\hat{\mathbf{d}}^*$ closest to $\hat{\mathbf{d}}$. The quantity $||\hat{\mathbf{d}}^* - \hat{\mathbf{d}}||$ then provides an inverse measure of the accuracy of the computed solution $\hat{M}$. In practice the minimising $\hat{\mathbf{d}}^*$ is approximated well by the reference residuals $\mathbf{d}^*$ so a more straightforward inverse error measure for least squares element fitting is given by the norm of the difference between the reference residuals and the computed residuals: $||\hat{\mathbf{d}} - \mathbf{d}^*||$.

4 NUMERICAL EXAMPLES

We illustrate the concepts discussed above using two assessment problems.

4.1 LEAST SQUARES BEST FIT CIRCLE

Given a set of data points $X = \{\mathbf{x}_i = (x_i, y_i), i \in I\}$ the least squares best fit circle $C^*(X)$ is the one which minimises the sum of squares error function

$$E(X, C) = \sum_{i \in I} d^2(\mathbf{x}_i, C) \tag{3}$$

where $d(\mathbf{x}, C)$ is the distance from a point $\mathbf{x}$ to the circle C. It can can be shown that minimising E involves the solution of nonlinear equations for which some form of iterative algorithm is required. However, we can form a related error function

$$\tilde{E}(X, C) = \sum_{i \in I} A^2(\mathbf{x}_i, C), \tag{4}$$

where $A(\mathbf{x}, C)$ is the difference between the area of C and the area of the circle concentric with C passing through $\mathbf{x}$. The minimum of $\tilde{E}$ can be found by solving a *linear* set of equations and for this reason the approximate error function $\tilde{E}$ is often used instead of E [2].

We have used the null space method to generate three data sets X_k, $k = 1, 2, 3$, each with the best fit circle centred at $(0, 0)$ and radius 30 to investigate the behaviour of an algorithm designed to minimise the approximate error function (4). We first compare the the best fit parameters $(\hat{a}, \hat{b})$ and $\hat{r}_0$ output by the software with their reference values: this is a forward comparison. We also look at the difference between the reference residual errors $d_i^* = d(\mathbf{x}_i, C(\mathbf{a}^*))$ and the residual errors calculated by the software, an inverse comparison.

Table 1 summarises the results of applying software designed of minimise the approximate error function $\bar{E}$ defined by (4).

Table 1. The effect of using an approximate method for finding the least squares best fit circle is demonstrated with three data sets. The first numerical row gives the norm of the vector of reference residuals $\mathbf{d}^*$ while the next gives the norm of the difference between the computed residuals and the reference residuals. The last three rows give the difference between the computed values of centre coordinates $(\hat{a}, \hat{b})$ and radius $\hat{r}_0$ and their reference values for the three data sets.

	X_1	X_2	X_3
$\|\mathbf{d}^*\|$	0.1	0.1	0.5
$\|\hat{\mathbf{d}} - \mathbf{d}^*\|$	8.8215e-05	5.0037e-01	6.4967e-01
$\hat{a}$	-2.8441e-05	3.2426e-01	3.1397e+00
$\hat{b}$	7.4476e-07	1.9505e+00	1.8907e+01
$\hat{r}_0 - 30$	1.7206e-06	-1.9699e+00	-1.8982e+01

The results show that for data set 1 the agreement with the reference results can be considered acceptable (if the measurements are in millimetres the inverse error corresponds to perturbations of the order of nanometres in the data) while for data sets 2 and 3 the agreement is poor (and corresponds to perturbations of the order of tens of micrometres in the data). For data set X_3 the estimate of the radius (18.9) has *no* figure of agreement with the reference answer (30.0). The difference between data set X_1 and the remaining two sets is that while X_1 represents data points approximately uniformly distributed around the circle, the points in data sets X_2 and X_3 are concentrated in a small arc of the circle. These results illustrate the general point that approximate methods only approximate well for certain types of data and should be used with care.

Figure 1 graphs the reference residuals $\mathbf{d}^*$, the test residuals $\hat{\mathbf{d}}$ and the difference $\mathbf{d}^* - \hat{\mathbf{d}}^*$ between the reference residuals and the residuals $\hat{\mathbf{d}}^*$ specifying the minimal orthogonal perturbation Δ of X_3 for which the approximate solution $\hat{C}$ is the true solution for $X + \Delta$. The graph shows that $\mathbf{d}^*$ is a good approximation to $\hat{\mathbf{d}}^*$.

4.2 TRIANGULATION

In this section we examine the behaviour of two approximate methods to solve the triangulation problem (2). The first (LLS) provides an estimate of the target x by solving

$$\min_{\mathbf{x}} E(\mathbf{x}, \theta, \phi) = \sum_k d^2(\mathbf{x}, L_k)$$

where $d(\mathbf{x}, L_k)$ is the distance from $\mathbf{x}$ to the kth line of sight $L_k = \{\mathbf{p}_k + t\mathbf{a}_k : t \in \mathcal{R}\}$. This problem can be formulated as a linear least squares problem to determine $\mathbf{x}$ and as such is

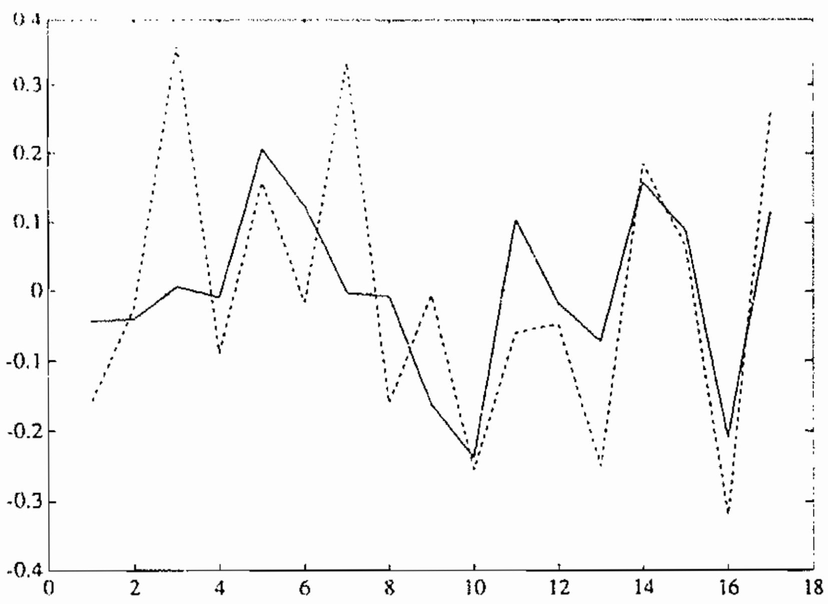

Figure 1 Reference residuals (solid), test residuals (dashed) and the difference (dotted) between the reference residuals and the residuals specifying a minimal perturbation of data set X_3.

a much simpler problem to solve than the original problem. However it does not take into account the fact that the standard error in position is proportional to the distance from the target to the theodolite. The second algorithm (WLLS) uses the target estimate $\mathbf{x}_0$ provided by the LLS algorithm to estimate the distances t_k from the target to the theodolites and solves a weighted linear least squares problem:

$$\min_{\mathbf{x}} E_W(\mathbf{x}, \theta, \phi) = \sum_k d^2(\mathbf{x}, L_k)/t_k^2$$

We have used the null space method to produce reference angle data and results for four theodolites with xy-coordinates $(\pm 2, \pm 1)$ and z-coordinates 0.0, 0.0, 0.0 and 0.1 for two targets $\mathbf{x} = (0.0, 0.0, 0.0)$ and $\mathbf{x} = (3.0, 0.5, 0.2)$. The first target is approximately the same distance from all the theodolites and we therefore expect little difference in algorithm behaviour for this target.

Table 2. Difference between the computed targets and reference solutions for the algorithms LLS and WLLS.

	$\mathbf{x} = (0.0, 0.0, 0.0)$		$\mathbf{x} = (3.0, 0.5, 0.2)$	
	LLS	WLLS	LLS	WLLS
$\hat{x} - x$	1.7306e-07	1.6030e-07	-4.6781e-05	-9.1672e-08
$\hat{y} - y$	6.4136e-09	1.5987e-08	-3.9665e-05	-2.6293e-07
$\hat{z} - z$	-1.6726e-07	2.1014e-09	-3.2207e-05	-1.9636e-10

Table 2 gives a forward comparison between the targets calculated by the approximate algorithms and the target used to generate the data while Table 3 gives an inverse comparison given by the difference between the angles ($\hat{\theta}_k$ and $\hat{\phi}_k$) defining the computed targets and the angles (θ_k^* and ϕ_k^*) defining the actual solution target. Both Tables give quantitative information on the accuracy of the two approximate methods and show the advantage of using the weighted model. Through the use of null space methods, this comparison with the solution of the exact problem (2) can be made without having to solve (2).

Table 3. Differences between the angles $\hat{\theta}_k$ and $\hat{\phi}_k$ defining the computed solution and the angles θ_k^* and ϕ_k^* defining the exact solution for theodolites T_k.

$x = (0.0, 0.0, 0.0)$				
	LLS		WLLS	
T_k	$\hat{\theta} - \theta^*$	$\hat{\phi} - \phi^*$	$\hat{\theta} - \theta^*$	$\hat{\phi} - \phi^*$
1	-3.2046e-08	-7.4803e-08	-2.5664e-08	9.3979e-10
2	3.7177e-08	-7.4803e-08	3.8454e-08	9.3979e-10
3	3.2046e-08	-7.4803e-08	2.5664e-08	9.3979e-10
4	-3.7177e-08	-7.7686e-08	-3.8454e-08	-1.7811e-09
$x = (3.0, 0.5, 0.2)$				
T_k	$\hat{\theta} - \theta^*$	$\hat{\phi} - \phi^*$	$\hat{\theta} - \theta^*$	$\hat{\phi} - \phi^*$
1	-4.7029e-06	-5.7489e-06	-4.3199e-08	1.1596e-09
2	-8.7809e-06	-6.0625e-06	-5.3881e-08	4.7545e-10
3	-5.0446e-05	-2.4178e-05	-2.4702e-07	-5.6887e-09
4	9.3870e-06	-1.6003e-05	-3.8593e-08	8.1621e-09

5 SUMMARY AND CONCLUDING REMARKS

The use of data sets to test software with an explicit computational aim requires reference pairs $\langle X, \mathbf{r}^* \rangle$ where $\mathbf{r}^*$ specifies the reference solution for the data set X. The reference results $\mathbf{r}^*$ should be chosen so that the pair is stably consistent. The null space method can be used to generate reference pairs from exact data by perturbations constrained to satisfy sufficiency conditions for the computational problem. Inverse measures can be found for the difference between reference solutions and solutions computed by software under test using null space methods. For the case of least squares element fitting (and other least squares approximation problems) residual errors can be used both to specify the reference results and as a basis for an inverse measure of error.

With these concepts – computational aim, reference pairs and null space methods, forward and inverse comparisons – it is possible to develop practical conformance tests which give reliable information on the quality of the software under test. However, their function should be as a final, formal confirmation that the assessment software is operating satisfactorily. By far the most important stages in the software development cycle involve the mathematical modelling, measurement strategy and experimental design, the parametrization of the mathematical model (a fundamental step in the development of a stable algorithm) and algorithm design.

6 ACKNOWLEDGEMENTS

My colleague Prof. Maurice Cox made many helpful comments on a draft of this paper. This work was funded by the Department of Trade and Industry as part of a programme in support of the UK National Measurement System.

7 REFERENCES

[1] G. T. Anthony, H. M. Anthony, M. G. Cox, and A. B. Forbes. The parame[illegible] of fundamental geometric form. Technical Report EUR 13517 EN, Commission of the European Communities (BCR Information), Luxembourg, 1991.

[2] G. T Anthony and M. G. Cox. Reliable algorithms for roundness assessment according to BS3730. In M. G. Cox and G. N. Peggs, editors, *Software for Co-ordinate Measuring Machines*, pages 30 – 37, Teddington, 1986. National Physical Laboratory.

[3] G. T. Anthony, M. G. Cox, and A. B. Forbes. Correctness of assessment software in co-ordinate metrology. In N. Ikawa, S. Shimada, T. Moriwaki, P. A. McKeown, and R. C. Spragg, editors, *International Progress in Precision Engineering – Proceedings of the 7th International Precision Engineering Seminar*, pages 28–39, Boston, May 1993. Butterworth-Heinemann.

[4] M. G. Cox and A. B. Forbes. Strategies for testing form assessment software. Technical Report DITC 211/92, National Physical Laboratory, Teddington, December 1992.

[5] M. G. Cox *et al.* Proposed draft ISO Standard: Method for testing software for computing Gaussian substitute elements on co-cordinate metrology. Submitted to ISO/TC3/WG10 for discussion, 1993.

[6] A. B. Forbes. Robust circle and sphere fitting by least squares. Technical Report DITC 153/89, National Physical Laboratory, Teddington, 1989.

[7] J. H. Wilkinson. Modern error analysis. *SIAM Review*, 13:548–568, 1971.