Evaluating Coordinate Measuring Machine Software Geometry Uncertainties Using National and International Standards

Craig M. Shakarji
National Institute of Standards and Technology
100 Bureau Drive, Stop 8260, Gaithersburg, MD 20899-8260
Telephone: 301-975-3545 Email: shakarji@nist.gov

John Raffaldi
MicroEncoder Inc.
11533 NE 118th Street, Building M, Kirkland , WA 98034
Telephone: 425-821-3906, ext. 360 Email: JohnR@MicroEn.com

Presented at the Measurement Science Conference
Anaheim, California, USA January 12-16, 2004

Abstract

In coordinate metrology, evaluating task specific measurement uncertainty can be difficult, a problem which prompted the current development of several related standards at the national and international levels. Software that processes coordinate data can be an important, but often overlooked, component of measurement uncertainty.

This paper describes the historically significant role software has played as an error source for measurements in coordinate metrology. It describes various kinds of fitting software and some of their potential problems. It then gives a series of practical and often easy steps users can take that will either increase confidence in such software or reveal some of its problems.

1. Introduction

Often, as with printed materials, we believe what we read. Measurements we obtain from measurement equipment and associated summary statistics are generally accepted as correct. But the path the software took to finally provide those results was, in general, a long one. While measurement hardware can be the source of many measurement errors, software too can introduce error; this is the topic of this paper. In creating metrological software, functions had to be modeled, code was typed, programmers used judgment to apply functions for certain conditions to create the measurement and analyze the results. Third-party software libraries with functions may have been used to shorten the time to market or to substitute for expertise and understanding. Usually the software would have been tested under narrow, range-specific conditions that might or might not simulate use in the real world. These conditions are fertile grounds for introducing errors in the software that might go undetected as a source of measurement uncertainty. Furthermore,

the user often has decisions to make about algorithm choice that can cause errant results even when the software tool is performing its functions well.

The primary purpose of this paper is to describe the methodology and tests developed by national and international laboratories and standards bodies for establishing software uncertainties. We begin by describing the need to characterize software's contribution to uncertainty, including past and present examples to emphasize the need and highlight historical problem areas as well as some causes. The next section provides an overview of some types of errors and why they occur. We then describe methods for solving some of these problems, suggesting means to characterize and possibly reduce software's contribution to uncertainty, giving particular attention to standards and approaches taken by national and international laboratories.

2. Software Uncertainties

2.1 General Issues

Measurement uncertainties cover a broad range of conditions which contribute to measurement errors. ISO/TS 14253-2:1999(E) [1] identifies software and calculations as potential contributors to measurement uncertainty as shown below in the graphic taken from the standard.

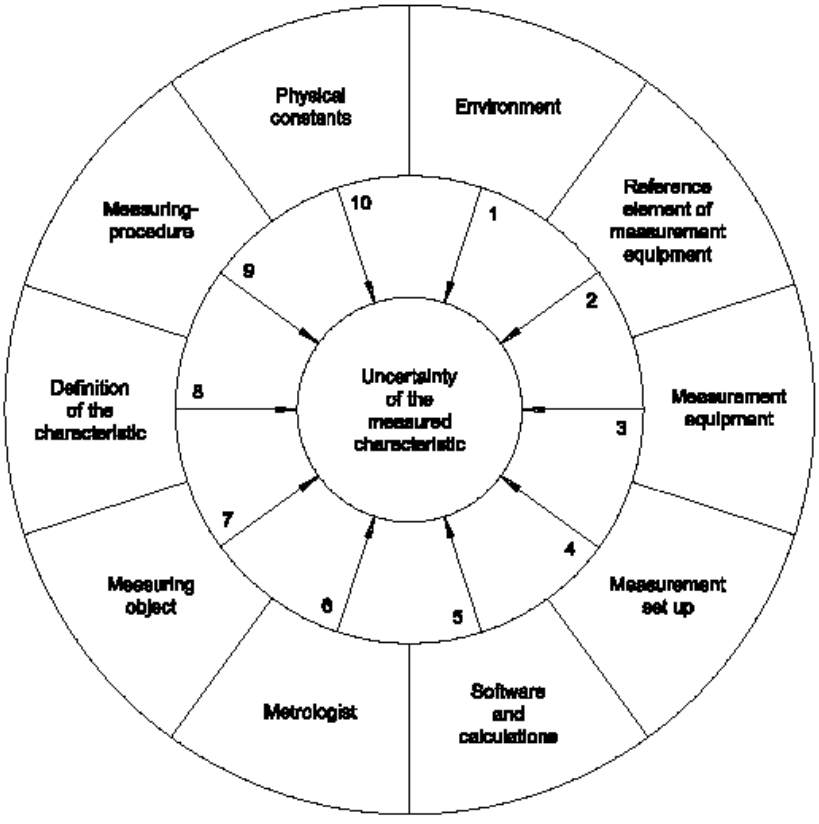

Fig. 1. Typical Uncertainty Contributors (Source ISO/TS 14253-2:1999(E)). The list includes "software and calculations" as a recognized contributor.

Typically the software end user has little control over the software uncertainty, and in some cases it may be difficult to assess the uncertainty contribution caused by the software. From the literature and calibration reports from accredited laboratories, it appears that software is typically ignored from the uncertainty budget and associated calibration calculations.

From a requirements standpoint ISO/IEC/EN 17025:1999(E) [2], General Requirements for the Competence of Calibration and Testing Laboratories (formerly ISO Guide 25 & EN45001) paragraph 5.4.7.1 states:

"Calculations and data transfers shall be subject to appropriate checks in a systematic manner."

Paragraph 5.4.7.2 states:

"When computers or automated equipment are used for the acquisition, processing, recording, reporting, storage, or retrieval of calibration or test data, the laboratory shall ensure that:

a) computer software developed by the user is documented in detail and is suitably validated as being adequate for use:

b) procedures are established and implemented for protecting the data; such procedures will include, but not be limited to, integrity and confidentiality of data or entry collection…

NOTE Commercial, off-the-shelf software (e.g., wordprocessing, database, and statistical programmes) in general use within designed application range may be considered to be sufficiently validated. However, laboratory software configuration/modifications should be validated as in 5.4.7.2a."

2.2 A Few Examples From The Past & Present Illustrating The Problem

The following examples illustrate errors that were eventually identified and might not be generally known.

CMM Least-Squares Fitting Software

Coordinate measurement machines (CMMs) have been in existence for many years and are becoming more popular as capabilities increase. On August 22, 1988, the Government-Industry Data Exchange Program (GIDEP) issued alert X1-A-88-01 [12]. CMMs were found to indicate different results for the same dimension and measurement technique using different least-squares fit algorithms. In the case of the CMMs, this problem, known as methods divergence, led to errors up to 50 % relative error in accepting bad parts or rejecting good parts [3]. These errors were once perceived as negligible [4].

CMM One- and Two-Sided Fitting Software

CMMs often include software that performs fits according to special criteria, namely maximum-inscribed, minimum-circumscribed, and minimum-zone (Chebyshev) fit objectives. In October 2002, research by the National Institute of Standards and Technology (NIST) was presented [5] indicating that serious problems can exist with these fits in present commercial software packages.

	Lines	Planes	Circles	Spheres	Cylinders	Cones
Minimum-zone	x	x	x	x	x	x
Minimum-circumscribed			x	x	x	
Maximum-inscribed			x	x	x	

Fig. 2. New reference algorithms. An "x" indicates NIST has developed a reference algorithm for the indicated fit objective. Least-squares reference algorithms already exist for all these geometries.

The paper compares the results against the data and reference results. The conclusion, even with the initial, limited number of comparisons was that there are alarming problems for several of these fit objectives. For some geometries and fit-objectives the reported results were not even reasonably close to the reference results for all ten out of ten tested data sets [5].

2.3 Types of Errors & Reasons For Software Uncertainty

Errors causing software uncertainty can form two general categories: 1) Implementation: Using the wrong solution, or not having the necessary knowledge to successfully solve the software problem, and 2) Software Development Process: Those activities that include writing and testing the software.

Implementation

ISO/TS 14253-2:1999(E) mentions nine potential areas causing software uncertainty:

- Rounding and Quantification
- Algorithms
- Implementation of algorithms
- Number of significant digits in the computation
- Sampling
- Filtering
- Correction of algorithm and certification of algorithm
- Interpolation / extrapolation
- Outlier handling

Software Development Process

Software development is a multi-step process, which, in most instances, leads to software that is functional and correct. However, if steps are excluded or are not executed, there is a high potential for the software to have errors. Although there are many references available for establishing a software development methodology, the NPL has written a best practice guide [7] specifically for developing software for metrology.

Some potential areas for introducing errors are:

- Not establishing user requirements
- Not defining the required software functionality
- Using a poor software architectural design
- Inadequate change control and configuration
- Lack of software code reviews
- Using and trusting untested third-party software
- An inadequate knowledge and inadequate ability to implement correct solution
- Mismatching variables, or using the wrong type
- Testing / releasing the wrong level (version) of the software
- Inadequate or incomplete testing
- Not understanding what to test or not knowing the expected results
- Displaying the wrong values
- Typing errors
- Trusting and using the published industry standards which have undocumented errors
- … an endless list?

3. Suggested Solutions and National Laboratories

As software packages can be extremely complex, it is difficult or impossible to list a few steps or precautions to take that will completely ensure the reliability of software results. Having stated that unavoidable reality up front, we now describe several suggested steps to help a user in ascertaining greater information on the reliability of metrological software. We begin by listing the easier steps—those that can return the most benefit for the least effort. We then describe some more advanced measures that can be taken if needed, particularly by a software creator.

Several of these steps involve research and testing done by national laboratories. In the United States, significant research and a testing service is available through NIST. Internationally, we describe some of the testing and work available through Physikalisch-Technische Bundesanstalt (PTB) in Germany and the National Physical Laboratory (NPL) in the United Kingdom. We note that significant work is being done in this field in other national laboratories, and their lack of mention in this paper should not indicate otherwise.

3.1 Take Advantage of the Testing of Others

Perhaps the easiest and potentially the most rewarding step is simply to contact the software supplier and ask what testing has been done on the software. Specifically, one

might ask if there is any documentation of testing and if copies can be obtained. The user should be careful to note the comparison of version numbers between the documents and the software in question. Also, the range of testing performed needs to be understood. An example of documented testing is the NIST Algorithm Testing and Evaluation Program for Coordinate Measuring Systems. We use this as an example to illustrate the user precautions.

NIST ATEP-CMS Test Service

As a result of the 1988 GIDEP Alert for CMM least-squares fitting software, NIST provides the Algorithm Testing and Evaluation Program for Coordinate Measuring Systems (ATEP-CMS). This test service involves the generation of a collection of data sets, the fitting of the data sets with both NIST reference software and the software under test, and a comparison of the corresponding fits.

The NIST ATEP-CMS supports seven geometric types that are found in most coordinate measuring systems: lines, planes, circles, spheres, cylinders, cones, and tori.

Three key tools provided by NIST are 1) a data generator, 2) reference algorithms, and 3) a comparator to analyze the test fit results. These appear in the architecture of the test process shown below:

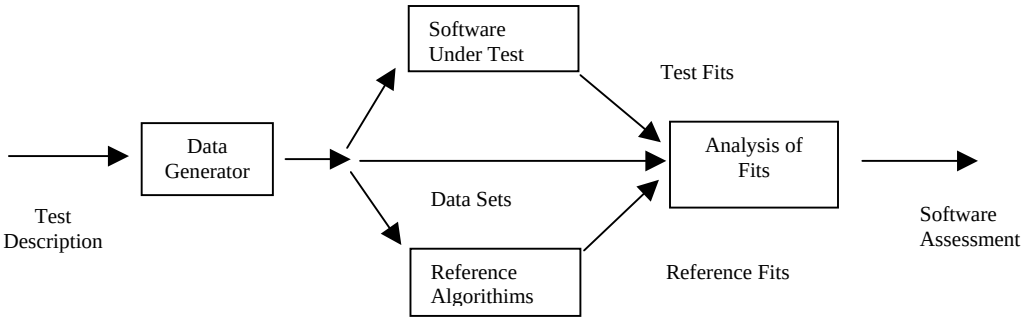

Fig. 3. ATEP-CMS Architecture

The data generator creates data sets based on four inputs: type of geometry, form errors, distribution of data points on the surface, and the random error distribution for the data points. If a customers submits points for fitting, the reference algorithms provide the ability to fit the points to geometries. The fit analysis component compares the software under test results to the reference results.

A performance evaluation certificate documents the results. The data sets can be custom generated or generated in accordance with the standard ASME B89.4.10 [13] (current) or ISO 10360-6 [14] (future). An example of the documented test results is pictured below:

REPORT OF SPECIAL TEST

For

Submitted by: Good-Fit Inc.

This software package was tested on 180 data sets, representing the following geometry types: lines, circles, planes, spheres, cylinders, and cones and following the test procedures documented in NISTIR 5686. The results of the tests are as follows:

Deviation And Uncertainty	**Geometry Type**					
	lines	circles	planes	spheres	cylinders	cones
Separation (μm) Uncertainty (μm)	3.0×10^{-3} 3.5×10^{-3}	1.2×10^{-4} 1.0×10^{-4}	2.3×10^{-4} 5.4×10^{-4}	8.1×10^{-5} 5.3×10^{-5}	9.3×10^{-2} 1.8×10^{-3}	4.8×10^{-4} 5.7×10^{-4}
Tilt (arcseconds) Uncertainty	6.2×10^{-4} 3.5×10^{-4}	7.1×10^{-3} 5.5×10^{-3}	3.3×10^{-3} 3.0×10^{-3}	_____	1.4×10^{-2} 2.3×10^{-2}	3.2×10^{-3} 4.3×10^{-3}
Radius (μm) Uncertainty (μm)	_____	4.9×10^{-4} 6.2×10^{-4}	_____	7.7×10^{-6} 8.4×10^{-6}	4.3×10^{-1} 3.4×10^{-1}	_____
Distance (μm) Uncertainty (μm)	_____	_____	_____	_____	_____	3.5×10^{-5} 5.3×10^{-5}
Angle(arcseconds) Uncertainty	_____	_____	_____	_____	_____	4.9×10^{-4} 4.3×10^{-4}

Fig. 4. A scanned sample piece of a first page of test results from the NIST Algorithm Testing and Evaluation Program for Coordinate Measuring Systems

But a note of caution is needed. The test certificate includes ranges of data set characteristics used in the test. For instance, if cylinder fitting were being tested, some data ranges might be 1) the minimum and maximum number of points in a data set, 2) the smallest and largest "aspect ratio" (height/diameter) of the cylindrical data, 3) the minimum and maximum amount of angular sweep of the data, and so on. Care must be taken not to assume that since software has an associated performance certificate it means the software is reliable for all measuring situations. If the test had data sets covering a minimal sweep of 90°, one should not automatically assume it is reliable for fitting when points are measured over an angular sweep of five degrees.

The test conditions used in the ATS, as provided by the customer, were as follows:

Unit of Measurement	Measurements were in millimeters.
Sampling strategy	Random, stratified, and equispaced strategies were used with up to 1000 points.
Measurement error	Uniformly random measurement error simulations were included.
Form errors	Typical errors simulated: bends, sinusoidal, steps, etc.
Range of part size	1 mm – 1500 mm.
Part origin	Within 2000 mm of coordinate system origin.
Partial features	Planes: maximum length:width ratio was 10 Circles: arcs as small as 60 degrees Spheres: patches as small as 1/8 of the sphere's surface area Cylinders: height:diameter ratio between 0.2 and 10; sweeps between 180 and 360 degrees.

Fig. 5. A scanned sample list of test conditions for the ATEP-CMS

Another limitation to keep in mind is the type of fitting covered by the test. The NIST ATEP-CMS test service covers least-squares fitting. One cannot conclude that the test certificate also indicates reliability in other kinds of fitting, like maximum-inscribed, minimum-circumscribed, or minimum-zone fit objectives.

When receiving documented testing of software, it is prudent to run a few tests to ensure the software gives the same results as documented. (In the case of the ATEP-CMS test, the data sets and fit results from the software under test should be available from the software supplier). This precautionary step is similar to the use of a check standard.

While the test service might lend itself best to software suppliers, some software users, particularly in cases where specialized testing is needed, can have the NIST test service executed themselves.

To contact NIST for further information:

Craig M. Shakarji
National Institute of Standards and Technology
100 Bureau Drive, Stop 8260, Gaithersburg, MD 20899-8260
Telephone: 301-975-3545 Email: shakarji@nist.gov

Test Service at PTB

Least-squares fitting testing is also conducted by PTB and is based on the European Commission ("Testing of three coordinate measuring machine algorithms, Phase II", BCR Report 13417 EN, 1991)

One difference between the two test services is that while NIST reports numerical information regarding deviations from the reference values, PTB categorizes results into four performance categories.

To contact PTB for further information:

Matthias Franke
Fachlaboratorium fuer Koordinatenmessgeraete / CMM section
Physikalisch-Technische Bundesanstalt
Bundesallee 100, Postfach 3345
38023 Braunschweig
Tel.: +49 531 5925324 e-mail: matthias.franke@ptb.de
Fax : +49 531 5925305

3.2 Take Advantage of the Reference Results of Others

If getting information from the software supplier is not sufficient, one can still take advantage of others' work. An easy way to do this is obtaining reference results. These are basically "problems" for which the "correct answer" is available. We list several sources for reference results that can be invaluable for time-saving means of testing software.

Reference Pairs for various fit objectives

NIST has available reference pairs for self evaluation by a user or software developer. These are data sets with corresponding reference fits. Correctly working software should be able to produce fits that are in close agreement with the reference values. Pairs are available for least-squares, minimum-zone (Chebyshev), maximum-inscribed, and minimum-circumscribed fit objectives. These collections can be a valuable tool for self-testing the performance of fitting software. Data sets are available (where applicable) for lines, planes, circles, spheres, cylinders, cones, and tori. (Contact Craig Shakarji, an author of this paper, for reference doubles).

The following figure shows points from a NIST data set simulating points measured on a sphere. The sphere shows the three dimensional perspective of the points in space, but it is not part of the testing process. Note that the distribution of points in three dimensional space simulates an actual measurement.

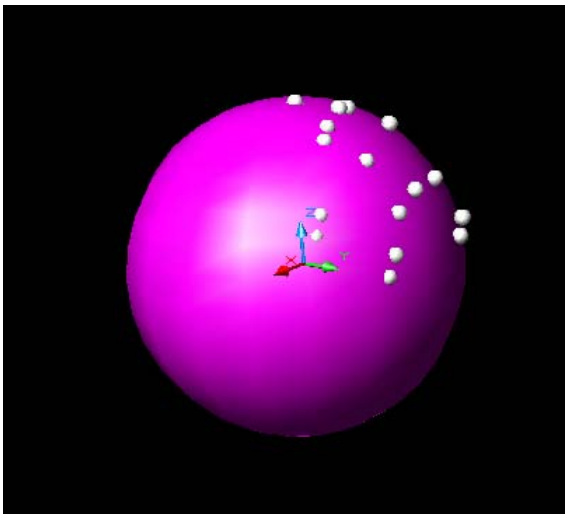

Fig. 6. Data points on a sphere shown in three dimensions

The data sets for these reference doubles are stored in files in tab delimited format that allows the data to be imported into the software being tested. A typical data file is shown below:

```
6
0.03031731    0.08511462    -0.18059486
0.01129568    0.06505772    -0.19048825
-0.03310788   0.06973422    -0.18475206
0.11348681    0.12710686    -0.10401784
0.0485886     0.1946364     -0.00526581
-0.09011113   0.14899397    -0.09483632
```

Fig. 7. A sample data file for a sphere

The first line has the number of test data points, and the following lines contain the *x*-, *y*-, and *z*-coordinates of the test data points. These data files can be read into the software in order to calculate the required results for comparisons with reference fits.

The reference fits are also given in files. A reference fit file for a sphere is shown below:

```
+3.4429994843466637630 0e+01
+8.4979999796554110960 0e+01
-1.1340073788329888770 0e+01
+1.1958044923909916590 0e-02
```

Fig. 8. A fit file for a sphere. The center and diameter are reported.

The fit file contains four values, the first three numbers are the *x*-, *y*-, and *z*-coordinates of the center of the sphere. The fourth number is the diameter. No attempt is made to truncate the reference results commensurate with the precision of the data. Rather, if the data is thought of as precise (having infinite trailing zeros), the digits reported in the reference fit files should all be correct. (That is, calculations were carried out in higher precision to assure the correctness of all the reported digits).

The following figure shows the required output fit results necessary for evaluation for each type of geometry:

Geometry	Required Output Fit Information
Lines 6 numbers	3 numbers represent a point on the line 3 numbers represent the direction cosines of the line
Planes 6 numbers	3 numbers represent a point on the plane 3 numbers represent the direction cosines of the normal to the plane
Circles 7 numbers	3 numbers represent the center of the circle 3 numbers represent the direction cosines of the normal of the plane containing the circle 1 number represents the diameter of the circle
Spheres 4 numbers	3 numbers represent the center of the sphere 1 number represents the diameter of the sphere
Cylinders 7 numbers	3 numbers represent a point on the cylinder axis 3 numbers represent the direction cosines of the cylinder axis 1 number represents the diameter of the cylinder
Cones 8 numbers	3 numbers represent a point on the cone axis 3 numbers represent the direction cosines of the cone axis 1 number represents the orthogonal distance from the reported point on the axis to the surface of the cone. 1 number represents the full apex angle of the cone in degrees (less than 180)
Tori 8 numbers	3 numbers represent the center of the torus 3 numbers represent the axis of the torus 1 number represents the major radius of the torus 1 number represents the minor radius of the torus

Fig. 9. Format of the fit files

Reference Results for More General Surfaces

NIST also has available some reference results for more general surfaces. These consist of three components: 1) the definition of the surface, 2) a set of data points, and 3) the correct rigid transformation (translation and rotation) that fits the data to the surface in a least-squares sense. Triples are similar to the reference pairs described above but for rigid transformations of more general surfaces. Currently the surfaces used are simply-defined mathematical surfaces, but the concept can be expanded to complex surfaces. Currently triples exist for shapes like paraboloids, saddles, and ogives. (Contact Craig Shakarji, an author of this paper, for reference triples).

Resources from NPL

The National Physical Laboratory (United Kingdom) has evaluated software and is very active in this field. Extensive information can be found at (http://www.npl.co.uk/ssfm/).

3.3 Take Advantage of Calibrated Artifacts

If the above resources do not suffice, a calibrated artifact can be indirectly used as a reference result by which software can be tested. For example, one can use a CMM to measure the diameter of a cylinder. The reported result includes the software processing.

While the testing of the software is not isolated from hardware errors, the influence of the software will impact the reported result, and serious software errors in the calculation will affect the deviation from the calibrated value.

The disadvantage of the method is clear; one generally does not have several calibrated artifacts or the time to run extensive software tests this way. Nevertheless the technique is powerful and is a documented means of obtaining task-specific measurement uncertainty, as in ISO Technical Specification 15530-3 [15]. Without going into detail, the method basically evaluates the uncertainty of a measurement task by analyzing the errors of similar measurements made on a similar, calibrated artifact. Since the measured values (including software calculations) are compared with the calibrated value, the effects of software errors would be revealed in the evaluated uncertainty.

3.4 Take Advantage of Other Software Packages

If independently written software is available elsewhere, select data can be submitted to various software packages and the results can be compared, even without having a reference result. While lack of a reference result keeps this method from representing complete testing, deviations among software packages alerts the existence of problems needing further investigation.

3.5 Take Advantage of Simple Cases

When all the above avenues fail, one might still be able to construct reference results for simplified cases. Software that claims to handle a more general set of problems should at least be able to provide correct answers for the simplified cases for which answers are known. An example of this occurred when a user tested some software packages that claimed to fit complex surfaces. Not having reference results available, the user constructed the complex surfaces, sampled data exactly on the surface (no errors included) and rotated and translated the data. Since there were no simulated errors included in the data, the reference result was known (simply the inverse transformation applied to the data). Certainly a software package should be able to fit the data to the surface when the data has no errors included, or so one might think. Interestingly, this test discriminated among some of the packages, since some software failed even that simplified case.

3.6 Take Advantage of Intuition and Pictures

The basic understanding of a specific problem can be used to one’s advantage in detecting software errors. Sometimes we can intuitively guess some measuring situations that might be problematic to software. In such cases extra care is prudent in scrutinizing the results. Examples include measuring cones with very small or very large apex angles, measuring very small arcs of circles, or measuring over small patches of surfaces.

One “good” thing about software errors is that they can sometimes produce results very far from the true value. This means our basic understanding of the solution can be used as a check (recall the story of students’ reporting the arc length of 10^{39}). Fitting software generally seeks to find a minimum from an initial starting guess it computes. If the starting guess is poorly chosen by the software, the reported result might not be the correct fit. These false fits are often completely different from the correct fit. For instance, a false cylinder fit will likely have its axis directed nearly 90° from the correct orientation.

This means that even our unrefined knowledge of the solution can sometimes be helpful in checking software. Creating a graph of the data and the corresponding fit can be helpful for this kind of checking. Better yet is the case when the software automatically includes a graph of the data and corresponding fit.

4. Conclusion

Software has been shown to be a component of measurement uncertainty in both the past and present. Often the software measurement uncertainty component is ignored, is difficult to evaluate, or may be subject to misplaced trust. There are many reasons for the errors, in both implementation of software solutions and the software development process. By using the services of National Standards Laboratories, the measurement uncertainty component due to software can, for specific conditions, be evaluated and, in some cases, reduced.

References

[1] ISO/TS 14253-2:1999(E), Geometric Product Specifications (GPS) – Inspection by measurement of workpieces and measuring equipment.

[2] ISO/IEC/EN 17025, 1999(E), General Requirements for the Competence of Calibration and Testing Laboratories.

[3] Feng Shaw C. & Hopp Theodore H., A Review of Current Geometric Tolerancing Theories and Inspection Data Analysis Algorithms, NISTIR 4509, U.S. Department of Commerce, National Institute of Standards and Technology, February 1991.

[4] ASME, 1985, ANSI/ASME Standard PTC 19.1-1985, Measurement Uncertainty, American Society of Mechanical Engineers, New York, NY.

[5] Shakarji, C. M., Evaluation of One and two Sided Geometric Fitting Algorithms In Industrial Software, Proceedings, American Society for Precision Engineering, 100-105, October 20-25, 2002.

[6] Cox M. G., Dainton M. P., Harris P. M., Testing Spreadsheets and Other packages Used in Metrology, NPL Report CMSC 07/00, National Physical Laboratory, UK, October 2000.

[7] Brinkly D., Logica, Software Support For Metrology, Best Practice Guide No. 3, Guidance on Developing Software for Metrology, National Physical Laboratory, UK, April 2001.

[8] MSA-3 Measurement System Analysis, March 2002, Automotive Industry Action Group, 26200 Lahser Rd., Suite 200, Southfield, MI 48034-7100 USA.

[9] Ermer Donald, Pythagorean Theorem To The Rescue, The Standard Volume 15,1, Spring 2001 / Fall 2000.

[10] Shakarji, C. M., Least-Squares Fitting Algorithms of the NIST Algorithm Testing System, Journal of Research of the National Institute of Standards and Technology. **103** (6), 633-641 (1998).

[11] Hopp Theodore H., Computational Metrology, U.S. Department of Commerce, National Institute of Standards and Technology, December 1993.

[12] Walker, R., CMM Form Tolerance Algorithm Testing, GIDEP Alert X1-A-88-01, Government-Industry Data Exchange Program, DOD, Washington, DC, 1988.

[13] ASME, Methods for Performance Evaluation of Coordinate Measuring System Software, B89.4.10-2000, American Society of Mechanical Engineers, New York, NY, 2000.

[14] ISO 10360-6:2000(E), Geometrical Product Specifications (GPS) —Acceptance test and reverification test for coordinate measuring machines (CMM) Part 6: Estimation of errors in computing Gaussian associated features.

[15] ISO TS 15530-3, Geometrical Product Specification (GPS)—Techniques of Determining the Uncertainty of Measurement in Coordinate Metrology—Part 3: Experimental Uncertainty Assessment Using Calibrated Workpieces.