Should You Be Concerned With Software Measurement Uncertainty?

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 International Dimensional Workshop
Nashville, Tennessee, USA May 12-16, 2003

Abstract

The contribution of software to the uncertainty of measurements is an important but often overlooked aspect of uncertainty evaluations. In coordinate metrology, software is often relied upon for complicated fitting of data, filtering, calibrations, and statistical calculations. Metrology equipment using software can range from hand-held digital calipers to coordinate measurement machines spanning several meters and can occur in almost any industry.

We show that historically software has often been responsible for measurement errors. While some of these errors have been reported, others are still surfacing. This paper highlights some of the errors, explores potential problem areas and their causes, and recommends methods to manufacturers and end users to identify and in some cases evaluate and reduce the uncertainty component due to software.

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.

What makes the situation worse is the tendency to accept software results without considering the quality of the result. Craig Shakarji, an author of this paper, offers anecdotal evidence of this:

"While grading tests of an undergraduate calculus class, I noticed that a third of the students gave a bizarre answer to an arc length calculation. The graph of the smooth, convex arc was pictured on the test paper, along with units, so any glance would indicate the answer would have to be less than, say, 10 units. But due to a commonly made mistake in the students' calculations, one third of the class reported the same astronomical answer, which was on the order of 10^{39} units. Not one of these students indicated on the test paper any suspicion of the erroneous result. They apparently had simply written down, without questioning their inputs, the results displayed by their calculators."

In this case, the calculators were not to blame, but rather the inputs. However, the widespread acceptance of the clearly erroneous result is telling of the tendency to uncritically accept automatically calculated results.

The purpose of this paper is to identify methods for recognizing, and in some cases evaluating and reducing the uncertainty component caused by software. 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. The next section provides an overview of some types of errors and why they occur. We then describe methods used in industry for solving these problems and suggest means to characterize and possibly reduce software's contribution to uncertainty.

2. Is There a Need to be Concerned With Uncertainty Due to Software?

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.

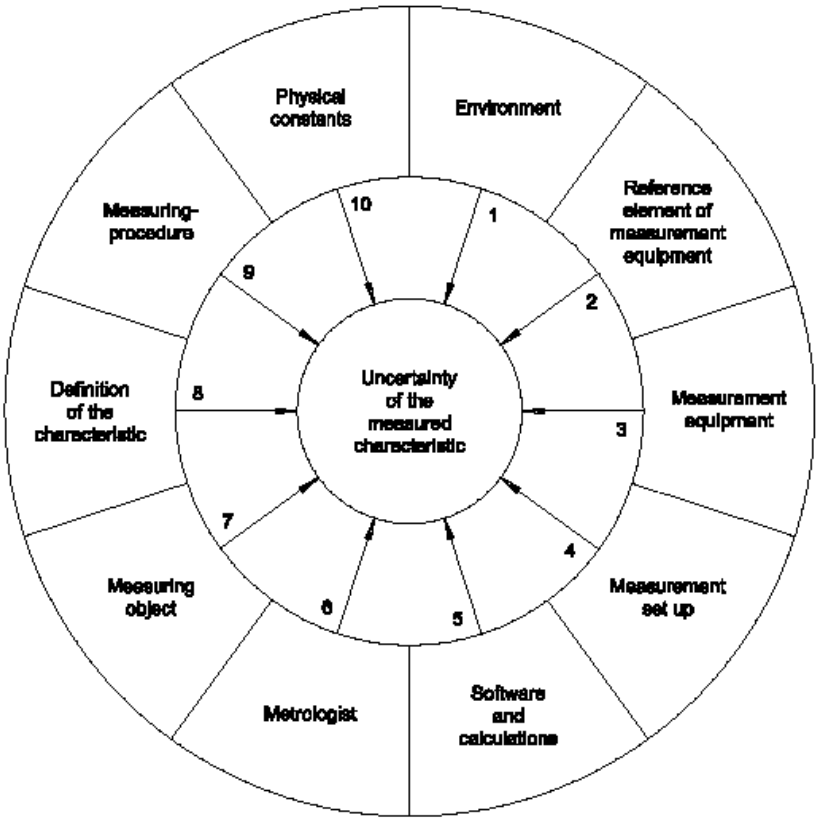

Figure 1: Typical Uncertainty Contributors (Source ISO/TS 14253-2:1999(E)). The list includes "software and calculations" as a recognized contributor.

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].

Spreadsheet Function STDEV

The National Physical Laboratory (NPL) in the United Kingdom has undertaken a study to evaluate many types of software for errors. One application tested was Microsoft Excel version 7.0a [see the disclaimer following the conclusion of this paper]. Several functions were tested: STDEV, LINEST, TREND…The report [6] states:

"We conclude that Excel's function does not implement a reliable algorithm for the calculation of sample standard deviation."

"The test results indicate that the IMSL, NAG, Matlab, S-PLUS, and MathCAD packages provide reliable software for the calculation of sample standard deviation [see the disclaimer following the conclusion of this paper]. *However, this is not the case for the Excel spreadsheet package that appears to implement an algorithm whose performance degrades as a function of problem degree of difficulty."*

The paper goes on to suggest that preprocessing the data helps obtain better results as a work around if needed.

GR&R Study Calculations – AIAG Range Method

The Automotive Action Industry Group (AIAG) (www.aiag.org) has been a pioneer in related automobile supplier standards. Of interest to metrologists and quality professionals are the gage reproducibility and repeatability study books under the Measurement System Analysis titles [9]. In the article, the author discusses the calculations which lead to misleading results. He points out that the final ratios for Equipment Variation (EV), Appraiser Variation (AV), and Part Variation (PV) divided by Total Variation (TV) are calculated using standard deviations and not variances.

In the article summary, the author states:

"Present AIAG R&R methods may be very misleading and should be modified according to the method given in this paper."

The following table highlights the differences between the AIAG range, analysis of variance (ANOVA), and improved AIAG range method.

	AIAG Method Using Standard Deviations	ANOVA	Improved AIAG Using Variances
%EV/TV	15.68 %	3.45 %	2.45 %
%AV/TV	17.76 %	2.18 %	3.09 %
%R&R/TV	23.7 %	5.63 %	5.54 %
%PV/TV	97.18 %	94.37 %	94.46 %

Table 1: GR&R Results

When variances are used for the calculations, it is seen that the results closely match the accepted ANOVA method results.

2.3 Types of Errors & Reasons For Software Uncertainty

Errors causing software uncertainty can form two general categories:

- Implementation – Using the wrong solution, or not having the necessary knowledge to successfully solve the software problem.

- 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

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.

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.

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. 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. 3. 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. 4. 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.

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).

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).

Other NIST Data Sets

NIST has additional data sets outside the field of coordinate metrology that are available for convenient download. There are 58 datasets covering, analysis of variance, linear

regression, nonlinear regression, and univariate summary statistics. Details can be found at (http://www.itl.nist.gov/div898/strd/).

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.

3.7 Tools Software Suppliers Can Provide

As mentioned above, software that provides a graph of the data set along with the corresponding fit can be helpful to alert the user to clearly erroneous results. While a correct looking picture is not a guarantee of a correct result, a picture can immediately reveal some mistakes. There are a number of other features a CMM software supplier can provide that also serve as self-checks of the reported solution. Like the graph, these checks cannot guarantee the solution is correct, but the failure of one of these checks immediately indicates a problem. While these checks have generally not found their way into most current CMM software, we list them here as possible future considerations.

The Gradient Test for Least-Squares Solutions

For a reported least-squares fit, a value can be calculated (the magnitude of the gradient of the objective function) that must be zero for a correct least-squares fit in normal measuring situations. For standard shapes, this number is not too difficult to compute. (The formulas are given for several shapes in [10]). A fit whose magnitude of its gradient is not nearly zero is cause for suspicion in the reported result. Since this is only a necessary condition, we emphasize that a zero gradient does not by itself prove correctness of the result. Although we have not found it commonly done, it would seem advantageous if fitting software were to include the magnitude of the gradient when reporting least-squares fits.

Inscribed, Circumscribed and Minimum-Zone Features

An inscribed feature must be, as simple it sounds, an inscribed feature. This means, for instance, that if one were to compute the signed distances from points in a data set to the correct maximum-inscribed circle (the "residuals"), they would all be greater than or equal to zero (i.e., none would lie inside the circle). While this seems trivially obvious, we note that in observing one commercial software package, some "maximum-inscribed" feature results actually contained data points, meaning the feature was not inscribed at all. The following was representative of calculated results:

Maximum residual:	0.215
Minimum residual:	-0.061

The minimum residual for an inscribed feature must not be negative (representing a point inside the feature). For a *maximum*-inscribed feature, the minimum signed residual must be zero. Clearly this example shows a problematic result. Similar statements can be made about minimum-circumscribed features and their maximum residuals.

One can use stronger necessary conditions. For instance, a minimum-circumscribed circle must touch three data points or touch two data points that are at opposite ends of its diameter. When points are sampled about a full circle, a maximum-inscribed circle should touch three points. Again, reported fits that satisfy these criteria are not proved to be correct, but the correct fits must satisfy these conditions.

Similar to the previous cases is the minimum-zone (or Chebyshev) fit objective. A necessary condition of a minimum-zone fit is that the maximum and minimum signed residual values be of opposite sign and equal size.

The maximum and minimum signed residuals that are often reported with results should be observed for these basic necessary conditions. Software can even include more residual information for checking more conditions (e.g., reporting the three minimum residuals for a maximum-inscribed circle problem, which should all be zero for a correct 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. Using a rigorous software development process and comparing calculation results to known data sets, the software uncertainty component can be reduced or eliminated. Therefore we conclude that the answer to the title of this paper is yes—software uncertainty *is* cause for concern. However, the prudent level of concern and effort can vary depending on how critical the application is. Thus end users of software-

driven metrology systems should satisfy themselves that the measurement results are acceptable and meet requirements.

Those who cannot remember the past are condemned to repeat it. -- George Santayana

DISCLAIMER: Commercial equipment and materials are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

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.

Table 1 data extracted from The Standard Volume 15,1, Spring 2001 / Fall 2000, (Data used with permission of author / ASQ Measurement Quality Division).