10

A methodology for testing classes of approximation and optimisation software

Bernard Butler, Maurice Cox, Alistair Forbes, Simon Hannaby and Peter Harris
National Physical Laboratory
Mathematical Software Group, Teddington, Middlesex TW11 0LW, UK.
Telephone: +44(0)181 943 6096. Fax: +44(0)181 977 7091.
Email: mgc@newton.npl.co.uk

Abstract

A methodology for mathematical software testing is presented which is applicable to classes of approximation and optimisation software. For testing purposes, a software implementation of an algorithm is regarded as a "black box". *Reference data sets* are input to the software, and the results provided are compared with *reference results*. It is described how reference data sets and results can be generated using a *data generator*, software which, when given reference results *a priori*, produces reference data sets corresponding to them. The emphasis is on *least-squares* software, but the concepts have broader application. Factors affecting comparison are considered and quality metrics for quantifying performance of the software under test described.

Keywords

Black-box testing, data generators, least-squares, mathematical software, null-space approach, performance measures, quality metrics, reference data sets, software quality, software testing

1 INTRODUCTION

Many proprietary software products rely on mathematical computation. Their availability and ease of use have created an impression in many users that the numerical quality of the software can be taken for granted. However, quality-management systems demand that

R. F. Boisvert (ed.), *Quality of Numerical Software*

software is fit for purpose, and organisations need to demonstrate, among other things, that the numerical software they use provides solutions to an accuracy sufficient for their problem domains. This paper is concerned with a methodology for mathematical software testing which is applicable to classes of approximation and optimisation software, and which relates to this need.

1.1 Testing objectives

Three objectives relevant to mathematical-software testing can be stated: (i) quantify numerical accuracy, (ii) identify instabilities, (iii) demonstrate fitness for purpose. Quantifying numerical accuracy is arguably the most natural requirement: for given input, how close is the output to the "true" values? Identification of instabilities is essential: how often does software work near-perfectly in initial tests for undemanding input, but give poor results or fail when applied subsequently to a significant problem? The demonstration of fitness for purpose, i.e., showing that mathematical software is appropriate for its intended purpose, is a difficult quality-management requirement. The methodology presented here can be used to help address these aspects.

Section 2 of this paper outlines the basic approach considered , viz., the use of reference data sets; also it indicates the need for quantitative performance measures. Section 3 reviews two approaches to the generation of reference data sets, the use of reference software and the use of data generators. It concentrates on the latter and introduces the concept of *graded* data sets. Section 4 reviews the method NPL has found to be of great utility in generating reference data sets, the *null-space* approach. Section 5 discusses methods by which results from software under test can be compared with reference results, taking into account the major factors affecting comparison. Section 6 is concerned with quality metrics for quantifying performance of the test software. Section 7 gives simple examples of applying the methodology. Section 8 outlines the testing services concerned with the evaluation of software used in industrial inspection and analytical chemistry that are being established at NPL. Section 9 contains concluding remarks.

2 "BLACK-BOX" TESTING AND PERFORMANCE MEASURES

For testing purposes, a software implementation of a mathematical algorithm is regarded here as a "black box" capable of receiving input data and returning the corresponding results. Correctly operating software will produce (a sufficiently close approximation to) the true result for all permissible inputs, and the testing process will help to reveal the extent to which this is achieved. The process considered is based on *reference data sets*, which are input to the *test software*, and the results provided, the *test results*, are compared with *reference results*.

The main objective of the reference data sets is to furnish a simple means for assessing the performance of mathematical software, specifically to provide quantitative measures of the quality of the underlying mathematical algorithms and their implementation, and information which can be used as part of the process of deciding fitness for purpose of the test software.

The concern is with the correctness of the underlying algorithms rather than the sub-

jective decisions made by packages. Thus the approach is applicable to *module testing*; system-integration testing requires a further stage and is not considered here.

3 GENERATION OF REFERENCE DATA

Two methods of producing reference data and corresponding reference results are the use of *reference software* and *data generators*. In general, neither approach produces reference data sets with exactly known reference results. Therefore, most reference results have only limited accuracy. This accuracy needs to be stated (see Section 6) and taken into account in the comparison. *Suites* of reference data sets which are *graded* appropriately (see Section 3.3) are very valuable.

3.1 Reference software

Reference software is software known to have a reputable pedigree, with good theoretical properties and to have been widely used in the field, e.g., software from an existing high-quality software library such as the NAG Library (Ford et al, 1979; NAG, 1996), or software produced specifically for this purpose (e.g., Anthony et al, 1993). Such software is used to produce reference results corresponding to prescribed reference data.

3.2 Data generators

A data generator is a specially prepared item of software which, when given reference results *as input*, produces reference data sets corresponding to them. These reference results, can themselves be specified automatically given a small number of controlling parameters. They are particularly applicable to testing linear algebra, approximation and optimisation software. The emphasis here is on *least-squares* software, which straddles these three disciplines. These generators typically require as input the problem data and Jacobian J, and from this information are capable of producing data sets having required properties.

A data generator is *dual* to a problem solver in that its acts as an *evaluator*, e.g., in determining a set of data points for which the approximant is known, rather than determining an approximant given the data.

3.3 Graded reference data sets

A stable algorithm can be expected to provide results to an accuracy consistent with the conditioning of the problem. An unstable algorithm can induce ill-conditioning or exacerbate the effects of inherent ill-conditioning. *Suites* of reference data sets can often be designed to help discriminate between such algorithms; such a suite contains *graded data sets* corresponding to a range of problems of various numerical condition, and can be regarded as providing a *parametrised battery test*. The performance on graded data sets will deteriorate (as the condition worsens) at a greater rate than for a stable algorithm. By this means the instability can be recognised. The performance measure in Section 6 is designed to help detect such algorithms. The work of Lyness (1979) on the use of

performance profiles for numerical-software testing is close in spirit to the method of graded data sets.

4 THE NULL-SPACE METHOD OF DATA GENERATION

The generation of reference data sets is illustrated for the *orthogonal-distance regression (ODR) problem* (Boggs, Byrd and Schnabel, 1987) , which can be posed as follows. Given a data set $X = \{\mathbf{x}_i : i \in I\}$ and the functional form $f(\mathbf{x}; \mathbf{u}) = 0$ of a (curve or) surface, depending on parameters $\mathbf{u}$, and defining $\mathbf{d} = [d_1, \ldots, d_n]^{\mathrm{T}}$, where $d_i = d(\mathbf{x}_i; \mathbf{u})$ is the (orthogonal) distance from the point $\mathbf{x}_i$ to the surface, determine values $\mathbf{u}^*$ which minimise the ℓ_2 error of fit

$$D(X; \mathbf{u}) = \|\mathbf{d}\|_2 = \left\{ \sum_{i \in I} d_i^2 \right\}^{1/2} . \tag{1}$$

4.1 Reference bases

Reference data sets and corresponding reference results can be generated by addressing a dual problem: given $\mathbf{u}^*$, find a data set X for which $D(X; \mathbf{u})$ takes its minimum value at $\mathbf{u}^*$. To characterise a minimum, define the *Jacobian* $J = \{\partial d_i / \partial u_j\}$, and for each point i a corresponding second-order matrix $G_i = \{\partial^2 d_i / \partial u_j \partial u_k\}$. Sufficient conditions for $\mathbf{u}$ to minimise (1) are that (i) $J^{\mathrm{T}}\mathbf{d} = \mathbf{0}$ (so that $\mathbf{d}$ lies in the *null space* of J^{T}) and (ii) the *Hessian* matrix $H = J^{\mathrm{T}}J + \sum_{i \in I} d_i G_i$ is strictly positive definite (Gill, Murray and Wright, 1981).

Perturbations normal to the surface

Let $X^* = \{\mathbf{x}_i^* : i \in I\}$ be a set of *footpoints*, viz., points lying in the surface $f(\mathbf{x}; \mathbf{u}) = 0$ and J^* the associated Jacobian. For this set $\mathbf{d} = \mathbf{0}$, so that condition (i) holds. Condition (ii) holds if J^* has full rank, for then $H = J^{*\mathrm{T}}J^*$ will be strictly positive definite. The rank of J^* depends on the parametrisation $\mathbf{u}$ of f and the number and distribution of the points X^*. J^* will have full rank if (a) the parametrisation is non-singular near $\mathbf{u}^*$, which means that a change in the parameter values results in a change in the position of the surface, and (b) the points in X^* are sufficient in number and distributed so that the surface is uniquely determined (locally) by the condition that the surface passes through the points in X^*.

Given such an X^*, a perturbed set $X = \{\mathbf{x}_i : \mathbf{x}_i = \mathbf{x}_i^* + \mathbf{e}_i,\ i \in I\}$, having associated Jacobian J, is sought such that the conditions remain satisfied. In general, $J \neq J^*$. However, $J = J^*$ for perturbations *normal* to the surface (Cox and Forbes, 1992). Equivalently, the Jacobian associated with the data set $X_{\boldsymbol{\delta}} = \{\mathbf{x}_i : \mathbf{x}_i = \mathbf{x}_i^* + \delta_i \mathbf{n}_i^*,\ i \in I\}$, where the elements of the vector $\boldsymbol{\delta} = [\delta_1, \ldots, \delta_n]^{\mathrm{T}}$ are arbitrary, and $\mathbf{n}_i^*$, $i \in I$, denotes the outward unit normal to the surface at $\mathbf{x}_i^*$, is independent of $\boldsymbol{\delta}$.

The conditions for $\mathbf{u}^*$ to be a minimum for $X_{\boldsymbol{\delta}}$ translate as (i) $J^{*\mathrm{T}}\boldsymbol{\delta} = \mathbf{0}$ and (ii) $H_{\boldsymbol{\delta}} = J^{*\mathrm{T}}J^* + \sum_{i \in I} \delta_i G_{\delta_i}$ is strictly positive definite, where $G_{\delta_i} = \{\partial^2 d_i(\mathbf{x}^* + \delta_i \mathbf{n}_i^*, \mathbf{u}^*) / \partial u_j \partial u_k\}$. The first condition will be satisfied if $\boldsymbol{\delta}$ lies in the *null space* Z^* of $J^{*\mathrm{T}}$. An orthonormal

basis for Z^* can be calculated from the QR decomposition of J^* (Golub and Van Loan, 1983). If the conditions hold, J^* has full rank of dimension $m \times n$, $m > n$, and any vector $\boldsymbol{\delta}$ satisfying $J^{*\mathrm{T}}\boldsymbol{\delta} = \mathbf{0}$ can be written as $Z^*\boldsymbol{\nu}$, for some vector $\boldsymbol{\nu}$ of $m - n$ elements, and *vice versa.*

Given Z^*, it remains to choose $\boldsymbol{\nu}$ so that the resulting Hessian matrix $H_{\boldsymbol{\delta}}$ is strictly positive definite. Since J^* has full rank, $H_{\boldsymbol{\delta}}$ will be strictly positive definite if the norm of $\sum_{i \in I} \delta_i G_{\delta_i}$ is sufficiently small compared with the minimum eigenvalue of $J^{*\mathrm{T}}J^*$ (Golub and Van Loan, 1983). In practice, only large perturbations give rise to indefinite Hessians.

Procedure for generating reference data sets

The following procedure provides a straightforward framework for generating reference data sets.

I Given parameter values $\mathbf{u}^*$, generate footpoints $X^* = \{\mathbf{x}_i^* : i \in I\}$, points lying in the surface $f(\mathbf{x}; \mathbf{u}) = 0$. Form the outward unit normals $N^* = \{\mathbf{n}_i^* : i \in I\}$ to the surface at X^*.

II Calculate the Jacobian matrix J^* and (a basis for) the null space $Z^* of J^{*\mathrm{T}}$.

III Select or compute multipliers $\boldsymbol{\nu}$. Set $\boldsymbol{\delta} = Z^*\boldsymbol{\nu}$ and $X_{\boldsymbol{\delta}} = \{\mathbf{x}_i^* + \delta_i \mathbf{n}_i^*,\ i \in I\}$.

IV Calculate $H_{\boldsymbol{\delta}}$ and check whether it is strictly positive definite; if it is not reduce the norm of $\boldsymbol{\delta}$ by reducing the multipliers, e.g., $\boldsymbol{\nu} := \boldsymbol{\nu}/2$, and go to Step III. (This step can be made more systematic, but as indicated above it is not difficult to generate data sets which give rise to strictly positive definite Hessians.)

Given X^*, N^* and Z^*, by varying $\boldsymbol{\nu}$ many data sets $X_{\boldsymbol{\delta}}$ can be generated in Step III; the triple $\langle X^*, N^*, Z^* \rangle$ is termed a *reference base* for generating reference pairs $\langle X_{\boldsymbol{\delta}}, \boldsymbol{\delta} \rangle$.

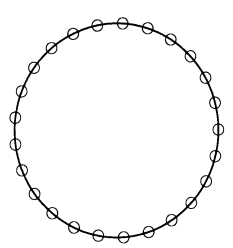
The *accuracy* of the null-space approach is considered by Cox and Forbes (1992).

The use of the null-space approach can be viewed as an extension to the inconsistent case of an old technique mentioned by Higham (1997) for testing solvers for consistent linear equations $Ax = b$ by generating b, given A and x, from $b := Ax$.

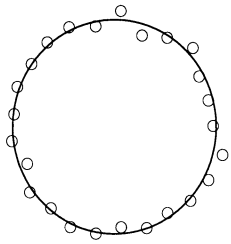
Figure 1 gives illustrative results for this procedure for generating reference data sets. (a) shows a circle defined by parameters $\mathbf{u}^*$ and a set of footpoints X^* lying on the circle. (b) shows the same circle with points $X_{\boldsymbol{\delta}}$ calculated as above for a multiplier vector $\boldsymbol{\nu}$ having a "small" magnitude. (c) is identical to (b) except that $\boldsymbol{\nu}$ has a larger magnitude. It is emphasised that all three data sets have the same solution circle with identical defining parameters $\mathbf{u}^*$. ((d) is discussed in Section 4.2.) It is to be noted that problems with large residuals and known solutions are particularly valuable for testing the performance of ODR and nonlinear-least-squares software, for which some algorithms sometimes experience difficulty in converging from a poor initial estimate of the solution parameters.

4.2 Mimicking prescribed error structure

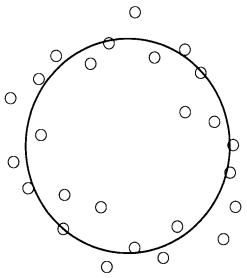
The vector $\boldsymbol{\nu}$ can be chosen "randomly", e.g., with its elements as a sample with mean zero and prescribed standard deviation from a Gaussian (pseudo-)random number generator. Alternatively, $\boldsymbol{\nu}$ can be chosen such that the data set so generated mimics prescribed error structure as follows.



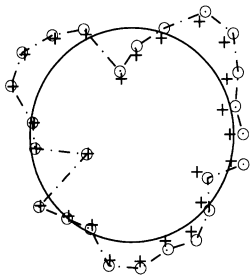
(a) Solution circle and footpoints.



(b) As (a) but points with small errors.



(c) As (a) but points with large errors.



(d) Required and mimicked error structure.

Figure 1 Illustration of the procedure for generating reference data sets for ODR software in the case of a circle. (a) shows a "solution" circle defined *a priori* and a set of 25 footpoints, points lying uniformly on the circle. (b) shows the same circle and a set of "small" null-space perturbations to the footpoints normal to the circle such that the circle is the least-squares solution to the perturbed points. (c) is identical to (b) except that the perturbations are different null-space perturbations and are larger. (d) shows the original circle, together with a defined error structure (indicated by small circles joined by dot-dashed lines) and replacement (nearest null-space) points (indicated by plus signs) such that the circle is the least-squares solution to the replacement points.

Suppose given are a data set $X = \{\mathbf{x}_i : i \in I\}$ and surface parameters $\mathbf{u}^*$. A perturbed data set $\hat{X}$ nearest X is sought for which $\mathbf{u}^*$ defines the best fit to $\hat{X}$. This problem can be solved as follows. Form footpoints $X^* = \{\mathbf{x}_i^* : i \in I\}$ satisfying $\mathbf{x}_i = \mathbf{x}_i^* + d_i\mathbf{n}_i^*$ by projecting the points X normally onto the surface. Let the required mimicking points lie on these surface normals and be written as $\hat{X} = \{\hat{\mathbf{x}}_i : \hat{\mathbf{x}}_i = \mathbf{x}_i^* + s_i\mathbf{n}_i^*,\ i \in I\}$. Then $\hat{X}$ can be determined by solving $\min_{\mathbf{s}} \|\mathbf{s} - \mathbf{d}\|_2$ subject to $J^{\mathrm{T}}\mathbf{s} = 0$. But $J^{\mathrm{T}}\mathbf{s} = 0 \equiv \mathbf{s} \in Z^*$. Hence $\mathbf{s} = Z^*\mathbf{v}$ for some $\mathbf{v}$. So it is merely necessary to solve $\min_{\mathbf{v}} \|Z^*\mathbf{v} - \mathbf{d}\|_2$, the solution to which is $\mathbf{v} = (Z^*)^{\mathrm{T}}\mathbf{d}$ since Z^* is orthonormal. So $\mathbf{s} = Z^*(Z^*)^{\mathrm{T}}\mathbf{d}$ and thus $\hat{X}$ is determined.

Figure 1(d) shows the circle used previously together with 25 data points having a desired error structure and the corresponding mimicked error structure produced by the above process.

5 COMPARISON OF TEST AND REFERENCE RESULTS

The major factors affecting objective comparison of test results and reference results are now considered.

5.1 Testable software claims

If the software supplier makes a (testable) documented claim, it is necessary to check whether this claim can be substantiated. The first two quality metrics in Section 6 are applicable here. If, e.g., the claim is that the provided routine for the inverse error function returns a result correct to three significant decimal figures for all values of the argument, the value of $N(\mathbf{b})$ in Section 6 should be three or greater.

It is difficult to automate the checking of claims or even to make such checking fall readily within the framework of the methodology of this paper since the nature of claims can be very different and it is not possible to anticipate claims made by suppliers of new software. Some of the aspects below, however, are relevant.

5.2 Computational precision

Different computer systems often operate to different accuracies, the *machine precision* or *unit roundoff* being a statement of this accuracy. Some computer systems use "software floating-point" or some other software implementation of arithmetic; sometimes computed values are held in (some parts of) memory to a lower accuracy than that of the floating-point processor (Gentleman and Marovich, 1974). In such a case, the relevant quantity is the precision to which the software is effectively operating rather than the unit roundoff. In any case, the quantity to be used to indicate the accuracy of the arithmetic is referred to here as the *computational precision* η.

5.3 Degree of difficulty

Even software of the highest quality will not generally deliver results to the accuracy indicated by η. This would only normally be possible for *perfectly conditioned* problems (condition $\kappa = 1$). Important problems regularly arise in which the conditioning is significant ($\kappa \gg 1$) and for which no algorithm, *however good*, can provide results to the accuracy obtainable for well-conditioned problems. The conditioning of the problem is reflected within the performance measure (Section 6) by a factor K, known as the "degree of difficulty".

5.4 Scale of the problem

If a problem is scaled such that the reference results are increased by a factor, the departure of the test results from these will likewise be increased. Thus, the *scale*, S say, is a necessary factor in the comparison. Because, for the performance measures here, S and the problem conditioning κ only appears as their product, the concept of the *scaled* degree of difficulty $K = S\kappa$ is used and referred to simply as the "degree of difficulty".

5.5 Different forms for the results

Different suppliers may produce differently parametrised results for the same problem. For example, polynomial-regression procedures may return monomial coefficients in terms of the user's independent variable or a normalised independent variable, or return Chebyshev-polynomial coefficients. In order to assess software on the basis of bespoke parametrised output, reference data sets and corresponding reference results have to be generated in the bespoke format and a suitable method of comparison developed which takes into account conditioning and other factors. Such an assessment places a considerable technological and organisational overhead on the testing process.

This difficulty can be overcome in one sense by testing instead canonical quantities that are *independent* of the manner in which the polynomial f is represented. An example of quantities that have this property are the *residuals* of the regression, viz., the differences between the data values of the dependent variable y and those of f. Computed *reference residuals* are compared with *test residuals*, the nature of the polynomial coefficients used being of no concern in this regard. (These coefficients would of course enter the *evaluation* of the residuals, unless the latter were computed by some other means such as the use of a projection operator.) Similar considerations apply to univariate regression and multiple linear regression, where different representations are also used, including centred and non-centred forms for straight lines and mean-corrected, normalised and un-normalised forms in multiple regression. Many software packages provide the residuals required for this approach, because it is of concern to assess how well the fitted function models the data.

5.6 Precision of the reference data sets

In general, reference data and results can be produced only to a limited precision. (There are exceptions in which integer- or rational-valued results can be provided.) It is therefore possible that test software could produce results to more figures than present in reference data, perhaps because of the high quality of the test software or the smallness of the computational precision η or a combination of these. In these circumstances, the use of direct comparison of test and reference results could wrongly disfavour the test software. To avoid this possibility, it is necessary that the precision of the reference data sets is quantified and accorded a value indicating its relative accuracy: see the performance measure P in Section 6.

6 QUALITY METRICS

Following the application of each reference data set three quality metrics which indicate how the software under test performed on that data set can be determined: (i) the difference between test and reference results, an *absolute* measure of departure, (ii) the number of figures of agreement, a *relative* measure of departure, and (iii) a performance measure. The first two metrics can be used to help decide whether a supplier's claim can be substantiated (Section 5). The performance measure is a metric which takes into account the dominant contributory factors also considered in Section 5. It is indicated below how each of these three quality metrics can be determined, and how they can be interpreted in practice. Input/output effects are also considered briefly.

6.1 Difference $d(\mathbf{b})$ between test and reference results

Consider a computation that produces (among its results) values $\mathbf{b}$ that are meaningful to be tested as a set (e.g., polynomial residuals). Let $\mathbf{d} = \mathbf{b}^{(\text{test})} - \mathbf{b}^{(\text{ref})}$ be the difference between the test results $\mathbf{b}^{(\text{test})}$ and the reference results $\mathbf{b}^{(\text{ref})}$. Use $d(\mathbf{b}) = \text{RMS}(\mathbf{d})$ to measure this difference, where $\text{RMS}(\mathbf{x}) = \|\mathbf{x}\|_2/n$ denotes the root-mean-square (RMS) value of the elements of an n-vector $\mathbf{x}$.

6.2 Degree of agreement $N(\mathbf{b})$

The degree of agreement $N(\mathbf{b})$ is the number of figures of agreement between $\mathbf{b}^{(\text{test})}$ and $\mathbf{b}^{(\text{ref})}$. Because most reference results have finite precision, $N(\mathbf{b})$ is limited in such cases by the number of correct significant figures, $M(\mathbf{b})$, in the reference results, and is defined by

$$N(\mathbf{b}) = \begin{cases} \min\left\{M(\mathbf{b}),\ \log_{10}\left(1 + \text{RMS}(\mathbf{b}^{(\text{ref})})/\text{RMS}(d(\mathbf{b}))\right)\right\}, & d(\mathbf{b}) \neq 0, \\ M(\mathbf{b}), & d(\mathbf{b}) = 0. \end{cases} \tag{2}$$

The rationale for this definition is as follows. If the quotient in (2) is of the order of 10^k, approximately k decimal figures in $\mathbf{b}^{(\text{test})}$ can be expected to be correct. The $\log_{10}$ function converts this quotient to the number of figures (k) correct. The use of $M(\mathbf{b})$ and the "min" operator ensures that the delivered result is sensible in the situation where the test software produces more than $M(\mathbf{b})$ correct figures.

There are circumstances where $N(\mathbf{b})$ could give a misleading impression. If, e.g., $\mathbf{b}$ is the mean of a set of numbers, some positive and some negative, and is very small in magnitude compared with the individual values, $\mathbf{b}^{(\text{test})}$ cannot be expected to have great relative precision and hence $N(\mathbf{b})$ could be small. This does not in itself imply that the test software is poor. In general, $N(\mathbf{b})$ should be examined in conjunction with the other quality metrics.

6.3 Performance measure $P(\mathbf{b})$

To take into account the problem degree of difficulty K (Section 5) it is meaningful to compare $d(\mathbf{b})$ with a tolerance equal to $K\eta$. K is a value that is provided for the problem, in general, a different value for each reference test data set. If the test software performs well, $d(\mathbf{b})$ is expected to be comparable in size to $K\eta$. Otherwise, $d(\mathbf{b})$ can exceed $K\eta$, by a large factor if the underlying algorithm is unstable. This comparison is appropriate unless η is especially small, in which case a measure based on this computation could give a spurious degradation of the results obtained from good-quality test software. Therefore, $d(\mathbf{b})$ is compared instead with $L = \max\{K\eta,\ C10^{-M}\}$, where C is a constant relating specifically to the reference data set. For cases in which the computational precision is particularly small, $d(\mathbf{b})$ is compared instead with a multiple of 10^{-M}, the relative accuracy of $\mathbf{b}^{(\mathrm{ref})}$.

In general, the quantity $d(\mathbf{b})/L$ has the property that it is of order unity for software that performs well on the reference data set and possibly very large for poorly-performing software.

The performance measure $P(\mathbf{b})$ is defined by

$$P(\mathbf{b}) = \log_{10}\left(1 + \frac{d(\mathbf{b})}{L}\right), \quad \text{where} \quad L = \max\{K\eta,\ C10^{-M}\}.$$

It indicates the number of figures of accuracy *lost* by the test software compared with the reference results, being near zero if $\mathbf{b}^{(\mathrm{test})}$ and $\mathbf{b}^{(\mathrm{ref})}$ agree closely, and having a value of about k if $\mathbf{b}^{(\mathrm{test})}$ has about k figures less accuracy than $\mathbf{b}^{(\mathrm{ref})}$.

A related performance measure is used in testing software for evaluating special functions (Van Snyder, 1997).

6.4 Interpretation of quality metrics

Quality metrics $d(\mathbf{b})$ and $N(\mathbf{b})$ can be used to help in assessing *fitness for purpose* of the test software. The user should first judge whether the reference data set can be regarded as representative of the application. If this is the case, and absolute errors are important for the application, it is necessary to decide whether $d(\mathbf{b})$ is acceptably small and, if relative errors are important, whether $N(\mathbf{b})$ is adequate for the intended purpose.

The performance measure $P(\mathbf{b})$ provides a *test of software quality* in that it indicates the number of figures of accuracy *lost* by the test software compared with the reference results. A large value may indicate that the underlying algorithm is unstable.

If, for *graded reference data sets* (taken in order), the values of the performance measure $P(\mathbf{b})$ have a tendency to *increase*, it is likely that an unstable method has been employed.

6.5 Input/output effects

Properly rounded conversion of numbers from internal to external format or *vice versa* introduces a *relative error* with a known bound. The results of such rounding affect the inputting of the reference data sets and the outputting of the test results. These effects

Table 1 The values of the quality metrics d (RMS difference), N (number of figures of agreement) and P (performance measure) for the mean $\bar{x}$ and standard deviation s of a sample of size 26 having a small coefficient of variation. K denotes the problem degree of difficulty and M the number of correct figures in the reference data.

Value	Ref. constants K	M	Reference value	Test value	d	N	P
$\bar{x}$	2.6×10^4	16	1.001×10^3	$1.001000000000000 \times 10^3$	0	16.0	0
s	6.3×10^4	14	1.6×10^{-2}	$1.599999921582637 \times 10^{-2}$	7.841740×10^{-10}	7.3	3.0

may be significant in that it is a *combination* of the software and the input/output routines that is being tested. In this case the user would need to decide whether this form of test is satisfactory for the intended purpose. Input/output effects are minimised if the rounding on conversion is bounded by η.

7 EXAMPLES OF APPLICATION

Two simple examples are presented to illustrate the methodology. One is for the mean and standard deviation of a set of numbers The other is for polynomial regression. In both cases graded reference data sets are used to help identify possible weaknesses in the test software. For these examples test software was executed on a computer with $\eta \approx 1.4 \times 10^{-17}$.

7.1 Mean and standard deviation

Given a data set (sample) $\{x_i\}_1^n$ the mean of the set is $\bar{x} = \sum_{i=1}^n x_i/n$ and the standard deviation is $s = \sqrt{\sum_{i=1}^n (x_i - \bar{x})^2/(n-1)}$. Proprietary test software for computing these quantities was executed for a reference data set consisting of 26 data values, lying in [1000, 1002], and thus having three figures in common. Table 1 summarises the reference constants K and M and the (exact) reference results, the test results and the quality metrics. From these results it can be concluded that for this data set the test software has returned a value for $\bar{x}$ correct to sixteen significant figures, which matches the accuracy achievable by reference software. However, the value for s is accurate to only seven significant figures. The computed performance measure indicates that for this data set and accounting for its "difficulty" the test software is losing three significant figures compared with reference results.

The same software was used to return results for eight graded reference data sets, having K-values 1, 10, ..., 10^7. The values of the performance measure $P(s)$ for s were 0.7, 0.8, 2.6, 3.0, 3.2, 5.3, 4.9, 6.6. Since the $P(s)$ are essentially increasing linearly for these data sets (good software would produce $P(s) \approx 1$ in all cases), it is concluded that the test software may be implementing an unstable method. It is possible that use is made of the "classical" formula $s = \sqrt{\{(\sum_{i=1}^n x_i^2) - (\sum_{i=1}^n x_i)^2/n\}/(n-1)}$, which is mathematically

sound but numerically can suffer from severe loss of figures through cancellation if the sample coefficient of variation $s/\bar{x}$ is small, as here.

7.2 Univariate polynomial regression

Given data points (x_i, y_i), $i = 1, \ldots, n$, the univariate polynomial regression problem (Clenshaw and Hayes, 1965) is to find the polynomial $f(x)$ of specified degree p that minimises the measure of error $\sum_{i=1}^{n} \{y_i - f(x_i)\}^2$. A reference data set for testing software for solving this problem contains such points and a value for p. The corresponding reference results contain the residuals $\mathbf{r} = \{r_i\}$, where $r_i = y_i - f(x_i)$ for the solution. A group of four graded reference data sets was used to discover whether test software for polynomial regression utilises commonly adopted unstable methods.

Test software gave the following performance measures $P(\mathbf{r})$ for this group: 1.1, 5.3, 7.7, 10.6. Since the $P(\mathbf{r})$ are increasing rapidly, it can be concluded that an unstable algorithm is implemented. Because reference data sets in the group had values of x removed successively further from the origin, it would appear that the underlying algorithm fails to normalise (or at least to centre) the data before fitting (cf. Clenshaw and Hayes, 1965). A further indication of instability was that for an additional group of graded reference data sets which were all identical, apart from the degree to be fitted, a rapid increase in $P(\mathbf{r})$ with respect to degree was also observed. It is also possible that *monomials* were used by the test software as a basis for polynomials, rather than, say, Chebyshev polynomials, thus worsening further the natural conditioning of the problem (again see Clenshaw and Hayes, 1965).

8 DISSEMINATION

The National Physical Laboratory is preparing a data-sets-based testing service for a range of ODR computations, important in industrial inspection. Currently covered is the testing of software for ℓ_2 lines and circles (in two and three dimensions), planes, spheres, cylinders, cones and tori. The scope is being extended to other geometric features and to the ℓ_∞ norm. The work is informing international standardisation activities through ISO/TC 3/WG 10, Dimensional and Geometrical Coordinate Measurement. A standard in preparation, Method for Testing Gaussian Substitute Features in Co-ordinate Metrology, will form the basis for testing this regression software.

The methodology has further been applied to the testing of software for some of the statistical computations widely used by analytical chemists (Ellison et al, 1994), including the sample mean and standard deviation, linear (straight-line) regression, multiple linear regression, partial least squares, principal components analysis, polynomial regression, and peak fitting/integration. This work provides the basis for user self-testing of such software (Butler et al, 1996).

9 CONCLUDING REMARKS

This paper has presented brief details of some of the work at the National Physical Laboratory on mathematical-software testing. Two of the main features of the work are (i) the generation of reference data sets and corresponding reference results, to be used in "black-box" testing of software, and (ii) quality metrics based on the comparison of test results and reference results. In (i), the reference data is derived from a knowledge of the necessary and sufficient conditions that obtain at the mathematical solution of the problem corresponding to the data set. The ability to produce reference data having known properties, such as graded data sets to identify instabilities, is seen as particularly important. In regard to (ii), although much testing has traditionally been carried out in the development of mathematical software, it is felt that the *user* increasingly needs access for quality-management purposes to simple measures of software performance. The measures described here have been used by mathematicians, metrologists and analytical chemists, and their use by a wider community will help to identify any limitations and hopefully suggest improvements.

REFERENCES

Anthony, G. T., Butler, B. P., Cox, M. G., Forbes, A. B., Hannaby, S. A., Harris, P. M., Bittner, B., Drieschner, R., Elligsen, R., Gross, H. and Kok, J. (1993) Chebyshev reference software for the evaluation of coordinate measuring machine data. Technical Report EUR 15304 EN, National Physical Laboratory, Teddington, UK. Published on behalf of Commision of the European Communities.

Boggs, P. T., Byrd, R. H. and Schnabel, R. B. (1987) A stable and efficient algorithm for nonlinear orthogonal distance regression. *J. Sci. Stat. Comput.*, **8**(6), 1052–1078.

Butler, B. P., Cox, M. G., Forbes, A. B., Hannaby, S. A., Harris, P. M. and Hodson, S. M. (1996) *Statistics Software Qualification: Reference Data Sets.* Royal Society of Chemistry, London. Edited by M. G. Cox, W. A. Hardcastle and S. L. R Ellison. In press.

Clenshaw, C. W. and Hayes, J. G. (1965) Curve and surface fitting. *J. Inst. Math. Appl.*, **1**, 164–183.

Cox, M. G. and Forbes, A. B. (1992) Strategies for testing form assessment software. Technical Report DITC 211/92, National Physical Laboratory, Teddington, UK.

Ellison, S. L. R., Cox,M. G., Forbes, A. B., Butler, B. P., Hannaby, S. A., Harris, P. M. and Hodson, Susan M. (1994) Development of data sets for the validation of analytical instrumentation. *J. AOAC International*, **77**, 777–781.

Ford, B., Bentley, J., du Croz, J. J. and Hague, S. J. (1979) The NAG Library 'machine'. *Software - Practice and Experience*, **9**, 56–72.

Gentleman, W. M. and Marovich, S. B. (1974) More on algorithms that reveal properties of floating point arithmetic units. *Comm. Ass. Comput. Mach.*, **17**, 276–277.

Gill, P. E., Murray, W. and Wright, M. H. (1981) *Practical Optimization.* Academic Press, London.

Golub, G. H. andVan Loan, C. F. (1983) *Matrix Computations.* North Oxford Academic, Oxford.

Higham, N. J. (1997) Testing the correctness of linear algebra software, in *The Quality of Numerical Software: Assessment and Enhancement* (ed. R. F. Boisvert), Chapman & Hall, London. (this volume)

Lyness, J. N. (1979) Performance profiles and software evaluation, in L. D. Fosdick, editor, *Performance Evaluation of Numerical Software*, Amsterdam. North-Holland, 51–58.

The Numerical Algorithms Group Limited (1996) *The NAG Fortran Library, Mark 17.* Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, UK.

Snyder, W. Van (1997) Testing functions of one and two arguments, in *The Quality of Numerical Software: Assessment and Enhancement* (ed. R. F. Boisvert), Chapman & Hall, London. (this volume)