NPL Report CMSC 34/04

Report to the National Measurement System Directorate, Department of Trade & Industry

TESTING THE NUMERICAL CORRECTNESS OF SOFTWARE

BY

M G Cox, P M Harris, E G Johnson,

P D Kenward and G I Parkin

January 2004

Testing the Numerical Correctness of Software

M G Cox, P M Harris, E G Johnson, P D Kenward and G I Parkin
Centre for Mathematics and Scientific Computing

January 2004

ABSTRACT

We describe the application of a general methodology for testing the numerical correctness of scientific software to functions for the calculations of sample (arithmetic) mean and sample standard deviation, straight-line (ordinary) regression and polynomial (ordinary) regression. The functions tested are taken from a number of proprietary software packages and libraries, including the NAG and IMSL (FORTRAN) libraries, Microsoft Excel, LabVIEW, S-PLUS, Matlab and various Java numerical libraries.

Each stage of the methodology, from documenting the specifications of the functions tested through the definition of performance parameters and performance measures to the presentation and interpretation of the results of the testing, is described. In this way, and by stating any assumptions made in the application of the methodology, the testing undertaken is made as objective as possible given the ("black-box") nature of the testing. A web-based facility is used to generate the reference data sets and corresponding reference results used in the testing, and sufficient information is provided for readers to reproduce these if they wish. In this way, the testing reported here is made transparent, repeatable and traceable, and may be extended to other functions for these calculations.

This report constitutes one of the deliverables of Project 2.1 *Numerical Software Testing* within the UK Department of Industry's National Measurement System *Software Support for Metrology* Programme 2001–2004.

ISSN 1471-0005

Authorised by Dr Dave Rayner,
Head of the Centre for Mathematics and Scientific Computing

National Physical Laboratory, Queens Road, Teddington, Middlesex, TW11 0LW

Contents

1. Introduction

The work described here constitutes a continuation of work undertaken as part of the first of the Software Support for Metrology (SS*f*M) programmes[1] concerned with testing the numerical correctness of software for (numerical) computations identified as important to metrology and provided as components within proprietary packages. In particular, it builds upon previous work [1, 2, 3], and extends that work in the following ways.

Firstly, the current SS*f*M programme has provided a web-based facility to enable users to generate, for a number of key computations, reference data and corresponding reference results appropriate to their own applications. The facility takes the form of data generators implemented in Java, with the aims of providing *portability* of the generators (across computer platforms), as well as *flexibility* and *reproducibility* of the reference data sets available. The testing described in this report relies on reference data sets generated using this facility. Furthermore, sufficient information is provided for interested readers to reproduce the reference data sets used. In this way, the testing reported here is made transparent, *repeatable* and *traceable*.

Secondly, the computations considered have been extended to include polynomial (ordinary) regression, in addition to the computations of sample (arithmetic) mean and standard deviation and straight-line (ordinary) regression. Polynomial regression is an important computation in metrology, arising in many metrology areas where it is necessary to model measurement data. Polynomials for the representation of the relationship between a stimulus and response variable arise from physical considerations, as well as being employed as empirical functions in the absence of a physical model.

Finally, the packages considered have been extended to include LabVIEW and methods from various Java numerical libraries. LabVIEW is a software package used widely for instrument control and the analysis of measurement data recorded by instruments. Java is finding increasing application, particular with the expanding use of the World Wide Web within metrology.

The report is organised as follows. Section 2 gives an overview of the (general) methodology employed for evaluating the numerical correctness of the results produced by scientific software. Section 3 provides a specification of the computations that are the subject of this report, as well as listing the particular software (functions or modules) for undertaking these computations that are tested. Section 4 considers issues of implementation arising from the application of the methodology for the variety of software languages and packages covered. For each computation, sections 5, 6 and 7 provide, respectively, specifications of the performance parameters used to define the reference data sets, the performance measures used to compare reference and test results, and the procedures used to generate reference data sets and corresponding reference results. Section 8 presents, and provides an interpretation of, the results of the testing. Section 9 contains conclusions.

2. Methodology

The procedure employed in evaluating the test software[2] is described in this section. A general methodology for evaluating the numerical correctness of the results produced by scientific software has been used. The basis of the approach is the design and use of *reference data sets*

[1] See http://www.npl.co.uk/ssfm

[2] The term *test software* used throughout this report refers to the software under test, and not the software employed to do the testing.

and corresponding *reference results* to undertake "black box" testing of the software. The reference data sets and results are generated in a manner that is consistent with the functional specification of the test software, and the results returned by the test software for the reference data sets are then compared objectively with the reference results using quality metrics or performance measures. Finally, the performance measures are interpreted in order to decide whether the test software meets the requirements and is fit for its intended purpose. The methodology is described in several reports [1, 4, 5, 6], and is illustrated by a case study [7].

The testing procedure is divided into six stages:

1. documenting the specification of the test software
2. interfacing to the test software
3. specification of reference data sets
4. specification of performance measures and testing requirements
5. generation of reference pairs, i.e., reference data and corresponding reference results
6. presentation and interpretation of performance measures.

These stages adopt previous recommendations [1], with the exception that the first two stages have been modified to suit testing from the perspective of a *user* of proprietary software packages and libraries. During the software development process, the first two stages would be *specification of the test software*, and *implementation of the test software*, though in practice these are carried out with varying degrees of formality. The application of this procedure by a software developer is presented in a case study [7]. Recording the results of the first two stages is important because it serves to define the basis of the testing undertaken, and to make clear any assumptions made about the test software.

Note that the testing process involves the use of software, in addition to the test software itself, for generating reference data sets and corresponding reference results, and evaluating and presenting quality metrics and performance measures.

3. Specification of Test Software

This section provides specifications for the computations that are the subject of this report, viz.,

1. Sample (arithmetic) mean and standard deviation
2. Straight-line (ordinary) regression
3. Polynomial (ordinary) regression.

It also lists the software (functions or modules) for undertaking these computations that have been tested as part of this work.

3.1 Sample (arithmetic) mean and standard deviation

Given the sample $\{x_i: i = 1, \ldots, m\}$, the sample (arithmetic) mean $\bar{x}$ and sample standard deviation s are defined, respectively, by

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

and

$$s = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})^2}\,.$$

The calculation of the sample standard deviation is important to metrology because of its role as a statistic for estimating the standard deviation of the population from which the sample was drawn, and within uncertainty evaluation as the basis for understanding the accuracy and repeatability of a measurement process.

The following test software for the computation of sample (arithmetic) mean and standard deviation is considered:[3]

- NAG [8].

  Function **`G01AAF`** returns a variety of basic statistics for a sample of observations, including the mean, standard deviation, coefficients of skewness and kurtosis, and the minimum and maximum values.

- IMSL [9].

  Function **`DUVSTA`** returns a variety of basic statistics for a sample of observations, including the mean, standard deviation, coefficients of skewness and kurtosis, and the minimum and maximum values.

- LabVIEW [10].

  Function **`Standard Deviation and Variance`** returns the sample mean, standard deviation and variance for a sample of observations.

- Matlab [11].

  Functions **`mean`** and **`std`** return the mean and standard deviation for a sample of observations.

- Microsoft Excel [12].

  Functions **`AVERAGE`** and **`STDEV`** return the mean and standard deviation for a sample of observations.

- S-PLUS [13].

  Functions **`mean`** and **`var`** return the mean and variance for a sample of observations. The sample standard deviation is obtained by taking the square root of the variance.[4]

- Java JNL [14].

  Functions **`average`** and **`standardDeviation`** return the mean and standard deviation for a sample of observations.

3.2 Straight-line (ordinary) regression

Given discrete data (x_i, y_i), $i = 1, \ldots, m$, and the straight-line model

$$f(x, b_1, b_2) = b_1 + b_2 x,$$

the straight-line (ordinary) regression problem is to solve

$$\min_{\mathbf{b}} \quad \sum_{i=1}^{m} \left(y_i - \left(b_1 + b_2 x_i\right)\right)^2.$$

[3] The (summary) descriptions of the test software provided in this and subsequent sections are the authors' interpretations of the documentation provided with the software. The documentation itself should be regarded as the most complete source of information about the software.

[4] The square root function provided by S-PLUS is used for this purpose.

This is an example of the *linear regression* or *linear least-squares* problem

$$\min_{\mathbf{b}} \quad \|\mathbf{y} - A\mathbf{b}\|^2 \equiv \min_{\mathbf{b}} \quad \sum_{i=1}^{m} \left(y_i - \mathbf{a}_i^{\mathrm{T}}\mathbf{b}\right)^2,$$

in the case that the observation (design) matrix A consists of the row vectors

$$\mathbf{a}_i^{\mathrm{T}} = (1, x_i), \qquad i = 1, ..., m,$$

$\mathbf{b} = (b_1, b_2)^{\mathrm{T}}$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$.

Because there are a number of different ways of writing the straight-line model defined above, for example in terms of a normalised independent variable, the residual deviations

$$d_i = y_i - f(x_i, b_1, b_2), \qquad i = 1, ..., m,$$

evaluated at the solution are used here to define the solution. The residual deviations are *independent* of the particular form of straight-line model used by the test software to solve the regression problem.

The straight-line (ordinary) regression problem is important to metrology because of its application in the discrete modelling of measurement data in situations where the relationship between the stimulus and response variables may be represented by a straight line, and the measurements of the response variable only are subject to error.[5]

The following test software for the straight-line (ordinary) regression computation is considered:

- NAG [8].

  Function **G02CAF** performs a simple straight-line regression to data, returning values for the slope and intercept of the fitted straight line. The residual deviations defining the solution are calculated from the slope and intercept values.[6]

  The function is used to undertake the computation in two ways.

  Firstly, the regression is performed in terms of the "natural" (unnormalised) variables x and y, i.e, function **G02CAF** is used to solve

  $$\min_{\mathbf{b}} \quad \sum_{i=1}^{m} \left(y_i - (b_1 + b_2 x_i)\right)^2$$

  for data (x_i, y_i), $i = 1, \ldots, m$, and the residual deviations defining the solution are evaluated from

  $$d_i = y_i - (b_1 + b_2 x_i), \qquad i = 1, ..., m.$$

  The computation undertaken in this way is referred to as *NAG A*. It mimics the way many users can be expected to solve the straight-line regression problem.

  Secondly, the regression is performed in terms of normalised variables, i.e., function **G02CAF** is used to solve

  $$\min_{\tilde{\mathbf{b}}} \quad \sum_{i=1}^{m} \left(\tilde{y}_i - (\tilde{b}_1 + \tilde{b}_2 \tilde{x}_i)\right)^2,$$

[5] Hence, the problem is described as one of "ordinary" regression.

[6] The BLAS routines **F06EFF** (copy a real vector) and **F06EDF** (multiply a real vector by a scalar) provided with the NAG library are used for this purpose.

where

$$\tilde{x}_i = x_i - \bar{x}, \qquad \tilde{y}_i = y_i - \bar{y}, \qquad i = 1, ..., m,$$

with $\bar{x}$ the sample mean of the values x_i, i =1, …, m, and $\bar{y}$ the sample mean of y_i, i =1, …, m. The residual deviations defining the solution are then evaluated from

$$d_i = \tilde{y}_i - \left(\tilde{b}_1 + \tilde{b}_2 \tilde{x}_i\right), \qquad i = 1, ..., m.$$

The computation undertaken in this way is referred to as *NAG B*. It mimics the way a "sophisticated" user might use function **G02CAF** to solve the straight-line regression problem, using the knowledge that a parametrisation of the straight-line model in terms of normalised variables can be expected to have improved numerical properties.

- IMSL [9].

  Function **DRLINE** fits a straight line to data by the method of least squares, returning values for the slope and intercept of the fitted straight line. The residual deviations defining the solution are calculated from the slope and intercept values.[7]

- LabVIEW [10].

  Function **Linear Fit** returns the slope and intercept of the best-fit straight line to data by the method of least squares, together with the values of the best-fit line corresponding to the data x-values. The residual deviations defining the solution are calculated from the values of the best-fit line.

- Matlab [11].

  The "backslash" operator "\" is used to undertake matrix left division. In particular, if A is an $m \times n$ matrix and **y** an $m \times 1$ vector with $m > n$, the result of "**b** = A**y**" is the least-squares solution **b** to the overdetermined linear system of equations **y** = A**b**.

  The computation is undertaken in two ways.

  Firstly, the (Matlab) statements

```
A = [ones(m,1), x];
d = y – A*(A\y);
```

  are used, where **ones(m, 1)** is an $m \times 1$ vector of ones, **x** an $m \times 1$ vector containing the x_i-values, and **y** an $m \times 1$ vector containing the y_i-values. The first statement assigns the observation (design) matrix for the regression problem. The second statement evaluates the vector **d** of residual deviations defining the solution to the regression problem. The computation undertaken in this way is referred to as *Matlab A*.

  Secondly, the (Matlab) statements.

```
xt = ((x – max(x)) – (min(x) – x))/(max(x) – min(x));
A  = [ones(m,1), xt];
d  = y – A*(A\y);
```

  are used, where **min(x)** and **max(x)** are respectively, the minimum and maximum values of **x**. The computation undertaken in this way is referred to as *Matlab B*.

  As with the use of the NAG function, the above approaches correspond to undertaking the computation in terms of, respectively, an unnormalised and normalised variable. However, this time only the independent (x) variable is normalised.

[7] The BLAS routines **DCOPY** (copy a real vector), **DSCAL** (scale a real vector by a constant) and **DADD** (add a constant to a real vector) provided with the IMSL library are used for this purpose.

- Microsoft Excel [12].

  Function **`TREND`** returns the values of a least-squares linear fit (trend line) to data corresponding to specified *x*-values. The residual deviations defining the solution are calculated from the values of the trend line.

- S-PLUS [13].

  Function **`lm`** fits a straight line to data by the method of least squares, returning the coefficients (slope and intercept) of the line, the values of the fitted line, and the values of the residual deviations associated with the line.

- Java JNL [14].

  Function **`linearFit`** returns the slope and intercept of the least-squares fit of a straight line to given data. The residual deviations defining the solution are calculated from the slope and intercept values.

- Java JLAPACK [15].

  Function **`dgelsx`** computes the (minimum-norm) solution to the (real) linear least-squares system of equations **y** = *A***b**. As with the use of the Matlab "backslash" operator, it is necessary to assign the observation (design) matrix *A* prior to calling the function **`dgelsx`**. Following the call to the function, the residual deviations **d** are then calculated from the solution values (for slope and intercept) returned by the function. Both operations (assigning *A* and calculating **d**) are undertaken in Java. Unlike the use of the NAG and Matlab functions, the computation is undertaken in one way only, viz., in terms of *unnormalised* variables.[8]

3.3 Polynomial (ordinary) regression

Given discrete data (x_i, y_i), $i = 1, \ldots, m$, and the polynomial model

$$p_n(x, b_1, b_2, ..., b_{n+1}) = b_1 + b_2 x + ... + b_{n+1} x^n,$$

of (specified) degree *n*, the polynomial (ordinary) regression problem is to solve

$$\min_{\mathbf{b}} \quad \sum_{i=1}^{m} \left(y_i - \left(b_1 + b_2 x_i + ... + b_{n+1} x_i^n \right) \right)^2.$$

This is another example of a *linear regression* or *linear least-squares* problem (Section 3.2). In this case the observation (design) matrix *A* consists of the row vectors

$$\mathbf{a}_i^{\mathrm{T}} = \left(1, x_i, ..., x_i^n\right), \qquad i = 1, ..., m,$$

$\mathbf{b} = (b_1, b_2, \ldots, b_{n+1})^{\mathrm{T}}$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$.

Because there are a number of different ways of expressing the polynomial model defined above, for example in terms of Chebyshev polynomials, the residual deviations

$$d_i = y_i - p_n(x_i, b_1, b_2, ..., b_{n+1}), \qquad i = 1, ..., m,$$

evaluated at the solution are used here to define the solution. The residual deviations are (mathematically) *independent* of the particular form of polynomial model used by the test software to solve the polynomial (ordinary) regression problem.

[8] However, any conclusions drawn from the results obtained from the use of the NAG and Matlab functions in terms of unnormalised and normalised variables can be expected to apply.

The polynomial (ordinary) regression problem is important to metrology because of its application in the discrete modelling of measurement data in situations where the relationship between stimulus and response variables may be represented by a polynomial model, and the measurements of the response variable are subject to error. Polynomials provide an important class of empirical models, often used when the relationship between the stimulus and response variables cannot be established on the basis of a physical understanding of the measurement problem.

The following test software for the polynomial (ordinary) regression computation is considered:

- NAG [8].

  Function **E02ADF** computes the weighted least-squares polynomial approximations of all orders up to a specified maximum for data (x_i, y_i), $i = 1, \ldots, m$, returning the coefficients in the Chebyshev-series representation of the polynomials. Function **E02AEF** evaluates a polynomial from its Chebyshev-series representation. The residual deviations defining the solution are calculated from the values of the least-squares polynomial approximation of specified order at the data x_i-values.

- IMSL [9].

  Function **DRCURV** fits a polynomial curve to data using the method of least squares, returning the coefficients in the power-series representation of the polynomial. The residual deviations defining the solution are calculated from the coefficient values.[9]

- LabVIEW [10].

  Function **General Polynomial Fit** returns the coefficients of the best-fit polynomial curve to data by the method of least squares, together with the values of the best-fit curve corresponding to the data x-values.[10] The residual deviations defining the solution are calculated from the values of the best-fit curve.

- Matlab [11].

  Function **polyfit** fits a polynomial curve to data using the method of least squares, returning the coefficients in the power-series representation of the polynomial. Function **polyval** returns the value of a polynomial from its power-series representation. The residual deviations defining the solution are calculated from the values of the least-squares polynomial approximation at the data x_i-values.

  In cases where the polynomial curve-fitting problem is detected as being poorly conditioned, the user is recommended to use the function **polyfit** in a way that includes centring and scaling of the independent (x) variable in order to improve the numerical properties of the polynomial and the least-squares fitting algorithm. Consequently, the computation is undertaken in two ways.

  Firstly, the (Matlab) statements

  ```
  p = polyfit(x, y, n);
  d = y – polyval(p, x);
  ```

  are used, where **x** is an $m \times 1$ vector containing the x_i-values, **y** an $m \times 1$ vector containing the y_i-values, and **n** is the degree of the polynomial. The first statement returns the coefficients p defining the least-squares polynomial approximation to the data. The second

[9] The BLAS routine **DDOT** (compute the "dot product" of real vectors) provided with the IMSL library is used for this purpose.

[10] There appears to be some confusion in the documentation for this function between the terms "polynomial degree" and "polynomial order".

statement evaluates the residual deviations defining the solution. The computation undertaken in this way is referred to as *Matlab A*.

Secondly, the (Matlab) statements

```
[p,s,mu] = polyfit(x, y, n);
d = y – polyval(p, (x – mu(1))/mu(2));
```

are used, which incorporates the centering and scaling operations by expressing the polynomial in terms of

$$\hat{x} = \frac{x - \mu_1}{\mu_2},$$

where μ_1 and μ_2 are, respectively, the sample mean and sample standard deviation of the data x_i-values. The computation undertaken in this way is referred to as *Matlab B*.

- Microsoft Excel [12].

  Function **TREND** returns the values of a least-squares linear fit to data corresponding to specified *x*-values. The function can be used for polynomial curve fitting by regressing against the same (*x*) variable but raised to different powers.[11] The residual deviations defining the solution are calculated from the values of the fitted curve.

- S-PLUS [13].

  Function **poly** fits a polynomial curve to data by the method of least squares, returning the values of the fitted curve, and the values of the residual deviations associated with the curve.

4. Interfacing to the Test Software

For each function tested, a "program" (or equivalent) is implemented in the language or package[12] of which the function is a part to undertake the following generic operations:

1. Load a reference data set
2. Load the corresponding reference results
3. Apply the (test) function to the reference data set to obtain test results
4. Compare the test and reference results by the evaluation of a performance measure
5. Write the value of the performance measure to (an ASCII text) file

These operations are repeated for a number of reference data sets (see Sections 5 and 7).

The performance measures used to compare the test and reference results are defined in Section 6. The computation of the measures requires the use of intrinsic functions (such as "square root" and "logarithm") as well as the value of the *computational precision*[13] of the

[11] The example given in Excel's on-line documentation is as follows. Suppose column A contains the data *y*-values and column B the data *x*-values. To obtain a least-squares cubic polynomial approximation, enter values for x^2 in column C, x^3 in column D, and then regress columns B through D against column A.

[12] For example, for the case of Microsoft Excel, the "program" is implemented using Visual Basic for Applications (VBA).

[13] For the commonly used floating-point arithmetic, η is the smallest positive representable number u such that the value $1 + u$, computed using the arithmetic, exceeds unity. For the many floating-point processors which today employ IEEE arithmetic, $\eta = 2^{-52} \approx 2 \times 10^{-16}$, corresponding to approximately 16-digit working.

(floating-point) arithmetic used to calculate the test and reference results. This is provided by a stand-alone (FORTRAN) program that calls NAG routine **X02AJF** and writes the result to an ASCII text file. The same value is used for the calculation of the performance measure for all functions tested and all reference data sets, because all calculations are undertaken on machines with the same specification.

The presentation of the results of the testing (Section 8 and Appendices B–D) involves the following operations:

- Copy the values of the performance measures onto worksheets of the Microsoft Excel spreadsheet package
- Display the values using the "chart" function of Microsoft Excel
- Copy (as pictures) the resulting charts into this document

5. Specification of Reference Data Sets

Performance parameters are used to capture the properties of data sets that would be encountered in practice and to describe the range of admissible inputs to the test software. By varying an individual performance parameter, sequences of data sets may be generated, with the sequence forming a *graded* sequence in cases where the performance parameter relates directly to the condition or "degree of difficulty" of the problem represented by the data. By investigating the performance of the test software for such graded sequences, it is possible to identify cases where the test software is based upon a poor choice of mathematical algorithm.

Table 1 lists the performance parameters for the computations of the sample (arithmetic) mean and standard deviation. The table also gives the values assigned to each performance parameter. Three sequences of reference data sets are used to test software for these computations, where each sequence is generated by setting the values for two of the performance parameters equal to default values and varying the value of the remaining parameter according to the information provided in the table.

Tables 2 and 3 list the performance parameters for the computations of straight-line (ordinary) regression and polynomial (ordinary) regression. The reference data sets defined in Tables 1 and 2 are also used as the basis for testing methods for the corresponding computations taken from a number of Java libraries, and described in a companion report [16].

6. Specification of Performance Measures

Performance measures or quality metrics are used to quantify the performance of the test software for the reference data sets to which the test software is applied. Furthermore, by relating the values of these metrics to the requirements of the user, it is possible to assess objectively whether the test software meets these requirements and is therefore "fit for purpose".

The following performance measure is used [4]:

$$P(\mathbf{x}) = \log_{10}\left(1 + \frac{1}{\kappa(\mathbf{x})\eta}\frac{\left\|\mathbf{y}^{\text{test}} - \mathbf{y}^{\text{ref}}\right\|}{\left\|\mathbf{y}^{\text{ref}}\right\|}\right), \qquad \mathbf{y}^{\text{ref}} \neq \mathbf{0}, \qquad \|\mathbf{y}\| = \sqrt{\sum_{i=1}^{m} y_i^2},$$

in which $\mathbf{x}$ denotes the input reference data set, $\mathbf{y}^{\text{test}}$ and $\mathbf{y}^{\text{ref}}$ are, respectively, the test and reference results, $\kappa(\mathbf{x})$ measures the problem "degree of difficulty" defined by the data set $\mathbf{x}$, and η is the computational precision. The performance measure $P(\mathbf{x})$ indicates, for the reference data set $\mathbf{x}$, the number of figures of accuracy lost by the test software over and above

the number expected to be lost by an optimally stable algorithm. A value of $P(\mathbf{x})$ close to zero indicates that, for the reference data set **x**, the test software returns a test result with a numerical accuracy that is comparable to that achieved by an optimally stable algorithm. A value of $P(\mathbf{x})$ close to eight, for example, indicates that, for the reference data set **x**, the test software returns a test result with eight fewer figures of accuracy than that returned by an optimally stable algorithm.

Performance parameter	*Values defining sequences of reference data sets*							
Sample (arithmetic) mean $\bar{x}$	**1**	10	10^2	10^3	10^4	10^5	10^6	10^7
Sample standard deviation s	**1**	10	10^2	10^3	10^4	10^5	10^6	10^7
Sample size m	10	**50**	100	150	200	300	400	500

Table 1: Values for performance parameters for the specification of reference data sets for the computation of sample (arithmetic) mean and standard deviation. Default values for the performance parameters are shown in bold.

Performance parameter	*Values defining sequences of reference data sets*							
Coordinate x_c defining the centroid of the x-data	**1**	10	10^2	10^3	10^4	10^5	10^6	10^7
Factor λ defining the slope $\tan \lambda\pi$ of the line	–0.33	–0.25	–0.20	–0.10	0.10	0.20	**0.25**	0.33
Number of points m	10	**50**	100	150	200	300	400	500
Length L of the interval containing the data	**1**	50	100	200	400	600	800	1000
Measurement standard deviation σ	**1**	2	3	4	5	6	7	8

Table 2: Values for performance parameters for the specification of reference data sets for the computation straight-line (ordinary) regression. Default values for the performance parameters are shown in bold.

Performance parameter	*Values defining sequences of reference data sets*							
Polynomial degree n	1	**2**	3	4	5	9	14	19
Number of points m	25	**50**	100	150	200	300	400	500
Endpoint x_{min} defining the interval $[x_{min}, x_{min}+10]$ containing the data	**1**	10	10^2	10^3	10^4	10^5	10^6	10^7
Measurement standard deviation σ	**1**	2	3	4	5	6	7	8

Table 3: Values for performance parameters for the specification of reference data sets for the computation polynomial (ordinary) regression. Default values for the performance parameters are shown in bold.

For the computations of the sample (arithmetic) mean and standard deviation, the performance measure defined above takes, respectively, the following forms:

$$P_{\bar{x}}(\mathbf{x}) = \log_{10}\left(1 + \frac{1}{\kappa(\bar{x})\eta}\frac{\left|\bar{x}^{\text{test}} - \bar{x}^{\text{ref}}\right|}{\left|\bar{x}^{\text{ref}}\right|}\right), \qquad \kappa(\bar{x}) = \max\left\{\frac{s^{\text{ref}}}{\left|\bar{x}^{\text{ref}}\right|}, 1\right\},$$

and

$$P_{s}(\mathbf{x}) = \log_{10}\left(1 + \frac{1}{\kappa(s)\eta}\frac{\left|s^{\text{test}} - s^{\text{ref}}\right|}{\left|s^{\text{ref}}\right|}\right), \qquad \kappa(s) = \max\left\{\frac{\left|\bar{x}^{\text{ref}}\right|}{s^{\text{ref}}}, 1\right\},$$

where **x** denotes the reference data set $\{x_i: i = 1, \ldots, m\}$.

For the straight-line and polynomial (ordinary) regression problems, the performance measure takes the form:

$$P_{\mathbf{d}}(\mathbf{x}, \mathbf{y}) = \log_{10}\left(1 + \frac{1}{\kappa(\mathbf{d})\eta}\frac{\left\|\mathbf{d}^{\text{test}} - \mathbf{d}^{\text{ref}}\right\|}{\left\|\mathbf{d}^{\text{ref}}\right\|}\right), \qquad \kappa(\mathbf{d}) = \max\left\{\frac{\|\mathbf{y}\|}{\left\|\mathbf{d}^{\text{ref}}\right\|}, 1\right\},$$

where **x**, **y** denotes the reference data set (x_i, y_i), $i = 1, \ldots, m$, and $\mathbf{d}^{\text{ref}}$ the reference residual deviations d_i^{ref}, $i = 1, \ldots, m$, defining the solution to the (ordinary) regression problem.

7. Generation of Reference Pairs

This section provides specifications for the calculation of reference data sets and corresponding reference results for the computations that are the subject of this report:

1. Sample (arithmetic) mean and standard deviation
2. Straight-line (ordinary) regression
3. Polynomial (ordinary) regression.

In each case, the inputs, outputs and the procedure for the calculation of reference data sets and corresponding reference results are given.

The specifications are the basis of Java implementations of data generators for the above computations that may be accessed on the web at

http://www.npl.co.uk/ssfm/theme2/rds/

Data generators for other computations are also available at the above web address, covering

- Gaussian peak (ordinary) regression
- Gaussian best-fit line (orthogonal distance)
- Gaussian best-fit circle (orthogonal distance)
- Chebyshev best-fit line (orthogonal distance)
- Chebyshev best-fit circle (orthogonal distance)
- Minimum circumscribed circle
- Maximum inscribed circle.

For each computation a Java Applet is provided for generating reference data sets and corresponding reference results. To help tailor the data sets to particular requirements, the user supplies values for a number of input parameters to the Applet that define the reference solution

and other properties of the reference data set, e.g., the sample size for the computation of sample mean and standard deviation, or the position of the centroid of the data for the computation of straight-line linear regression. An important additional input parameter is a random number seed *S*. This parameter permits

- a previously generated reference data set to be *reproduced* (by setting the same values for the input parameters, including *S*, as for the previously generated data set), or
- a *different* reference data set with the *same* reference solution to be generated (by setting the same values for the input parameters, excepting *S*).

The Applet generates output in two windows. One window contains the reference data set and the other the reference results and auxiliary information used in the comparison of reference and test results.

Appendix A illustrates the Applet for generating a reference data set and corresponding reference results for the computation of the sample (arithmetic) mean and standard deviation. The two windows generated by the Applet containing "test data" and "test parameters" are also shown.

The specifications of reference data given in Section 5 together with the software described here are sufficient to allow the reader to reproduce the particular reference data sets, with their corresponding reference results, used as the basis of the testing described in this report. All reference data sets considered in this work are generated with the value of the random number seed *S* set equal to zero.

7.1 Sample (arithmetic) mean and standard deviation

7.1.1 Inputs

The inputs for the data generator consist of values for the following parameters:

- Sample (arithmetic) mean $\bar{x}$ satisfying $|\bar{x}| \geq 1$
- Sample standard deviation s satisfying $s \geq 1$
- Sample size m satisfying $m \geq 2$
- Random number seed S

7.1.2 Outputs

The outputs from the data generator consist of the following information:

- Reference data x_i, $i = 1, \ldots, m$
- Reference constants $\bar{x}$, s, m, S, $\kappa(\bar{x})$, $\kappa(s)$

where $\kappa(\bar{x})$ and $\kappa(s)$ are, respectively, the "degrees of difficulty" for the problems of determining the sample (arithmetic) mean and sample standard deviation for the reference data x_i, $i = 1, \ldots, m$.

7.1.3 Procedure

The procedure implemented by the data generator for the computation of sample (arithmetic) mean and standard deviation is as follows:

1. Calculate the $m \times 1$ design matrix A with rows

$$\mathbf{a}_i^{\mathrm{T}} = 1, \qquad i = 1, ..., m.$$

2. Calculate residual deviations d_i, $i = 1, \ldots, m$, with mean zero (Section 7.4), setting $\sigma = s$.

3. Calculate the sample standard deviation of the residual deviations d_i, $i = 1, \ldots, m$:

$$s_d = \sqrt{\frac{\sum_i d_i^2}{m-1}}.$$

4. Scale the residual deviations to have sample standard deviation s:

$$d_i := \frac{s}{s_d} d_i, \qquad i = 1, ..., m.$$

5. Calculate reference data (with sample mean $\bar{x}$ and sample standard deviation s):

$$x_i = \bar{x} + d_i, \qquad i = 1, ..., m.$$

6. Calculate "degrees of difficulty":

$$\kappa(\bar{x}) = \max\left\{\frac{s}{|\bar{x}|}, 1\right\}, \qquad \kappa(s) = \max\left\{\frac{|\bar{x}|}{s}, 1\right\}.$$

7.2 Straight-line (ordinary) regression

7.2.1 Inputs

The inputs for the data generator consist of values for the following parameters:

- Coordinates (x_c, y_c) of the centroid of the data
- Angle α made by the line with the x-axis satisfying[14] $-4\pi/5 \le \alpha \le 4\pi/5$
- Number of points m satisfying $m > 2$
- Length L of interval of x containing the x-values
- Magnitude of measurement standard deviation σ satisfying $\sigma \ge 1$
- Random number seed S

7.2.2 Outputs

The outputs from the data generator consist of the following information:

- Reference data (x_i, y_i), $i = 1, \ldots, m$
- Reference constants x_c, y_c, α, m, L, σ, S, $\kappa(\mathbf{d})$, d_i, $i = 1, \ldots, m$

where d_i, $i = 1, \ldots, m$, are the (reference) residual deviations corresponding to the solution straight line, and $\kappa(\mathbf{d})$ is the "degree of difficulty" for the problem of determining these deviations for the reference data (x_i, y_i), $i = 1, \ldots, m$.

7.2.3 Procedure

The procedure implemented by the data generator for the computation of straight-line (ordinary) regression is as follows:

1. Calculate x-coordinates x_i, $i = 1, \ldots, m$, uniformly spaced between $x_{\min} = x_c - L/2$ and $x_{\max} = x_c + L/2$.
2. Calculate the $m \times 2$ design matrix A with elements a_{ij}, $i = 1, \ldots, m$, $j = 1, 2$:

   2.1. Normalise x-coordinates:

[14] Chosen to ensure the line is not too steep, and it is reasonable to measure errors vertically.

$$X_i = \frac{(x_i - x_{\min}) - (x_{\max} - x_i)}{(x_{\max} - x_{\min})}, \qquad i = 1, ..., m.$$

2.2. For each $i = 1, \ldots, m$:

$$a_{i,1} = 1,$$
$$a_{i,2} = X_i.$$

3. Calculate residual deviations d_i, $i = 1, \ldots, m$ (see Section 7.4).
4. Calculate y-coordinates:

$$y_i = d_i + (x_i - x_c)\tan\alpha + y_c, \qquad i = 1, ..., m.$$

5. Calculate "degree of difficulty":

$$\kappa(\mathbf{d}) = \max\left\{\frac{\|\mathbf{y}\|}{\|\mathbf{d}\|}, 1\right\}.$$

7.3 Polynomial (ordinary) regression

7.3.1 Inputs

The inputs for the data generator consist of values for the following parameters:

- Polynomial degree n
- Number of points m satisfying $m > n + 1$
- Endpoints of interval $[x_{\min}, x_{\max}]$ containing the x-values
- Magnitude of measurement standard deviation σ satisfying $\sigma \geq 1$
- Random number seed S

7.3.2 Outputs

The outputs from the data generator consist of the following information:

- Reference data (x_i, y_i), $i = 1, \ldots, m$
- Reference constants n, m, $x_{\min}$, $x_{\max}$, σ, S, $\kappa(\mathbf{d})$, d_i, $i = 1, \ldots, m$

where d_i, $i = 1, \ldots, m$, are the (reference) residual deviations corresponding to the solution polynomial, and $\kappa(\mathbf{d})$ is the "degree of difficulty" for the problem of determining these deviations for the reference data (x_i, y_i), $i = 1, \ldots, m$.

7.3.3 Procedure

The procedure implemented by the data generator for the computation of polynomial (ordinary) regression is as follows:

1. Calculate x-coordinates x_i, $i = 1, \ldots, m$, uniformly spaced between $x_{\min}$ and $x_{\max}$.
2. Calculate the $m \times (n + 1)$ design matrix A with elements a_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n + 1$:

   2.1. Normalise the x-coordinates:

$$X_i = \frac{(x_i - x_{\min}) - (x_{\max} - x_i)}{(x_{\max} - x_{\min})}, \qquad i = 1, ..., m.$$

2.2. For each $i = 1, \ldots, m$:[15]

$$a_{i,1} = 1,$$
$$a_{i,2} = X_i,$$
$$a_{i,j} = 2X_i a_{i,j-1} - a_{i,j-2}, \qquad j = 3, ..., n+1.$$

3. Calculate the vector **d** of residual deviations d_i, $i = 1, \ldots, m$ (Section 7.4).
4. Generate a vector **b** of polynomial coefficients b_i, $i = 1, \ldots, n + 1$: each coefficient is chosen randomly between −1 and +1.
5. Calculate the vector **y** of y-coordinates y_i, $i = 1, \ldots, m$:

$$\mathbf{y} = A\mathbf{b} + \mathbf{d}.$$

6. Calculate "degree of difficulty":

$$\kappa(\mathbf{d}) = \max\left\{\frac{\|\mathbf{y}\|}{\|\mathbf{d}\|}, 1\right\}.$$

7.4 Residual deviations

The following calculation is common to all three procedures described above for generating reference data sets and corresponding reference results.

7.4.1 Inputs

The inputs for the data generator consist of values for the following parameters:

- Design matrix A of dimension $m \times n$ satisfying $m > n$
- Magnitude of measurement standard deviation σ satisfying $\sigma \geq 0$

7.4.2 Outputs

The outputs from the data generator consist of the following information:

- Vector **d** of residual deviations of dimension $m \times 1$

7.4.3 Procedure

The procedure implemented for the computation of residual deviations is as follows:

1. Calculate null-space N of A^{T} where N has dimension $m \times (m - n)$ [18]:
   1.1. Form the singular value decomposition (SVD) of A^{T}:

$$A^{\mathrm{T}} = USV^{\mathrm{T}}.$$

   1.2. Form N from the columns of V:

$$N_{i,j} = V_{i,n+j}, \qquad i = 1, ..., m, \qquad j = 1, ..., m - n.$$

2. Generate a vector **e** of random numbers e_i, $i = 1, \ldots, m$, sampled from a Gaussian distribution with mean zero and standard deviation σ.
3. Calculate **d** in terms of matrix-vector products as follows:

$$\mathbf{v} = N^{\mathrm{T}}\mathbf{e}, \qquad \mathbf{d} = N\mathbf{v}.$$

[15] This is the recurrence relation for evaluating Chebyshev polynomials [17].

The key step in the above procedure is to form the singular value decomposition of A^{T} (step 1.1). This is achieved using routine DGESVD provided as part of the Java library JLAPack. The library [15] provides the LAPACK [19] numerical subroutines in the form of Java class files, executable by the Java Virtual Machine (JVM), making it possible for Java applications to use established legacy numerical software that was originally written in FORTRAN.

8. Presentation and Interpretation of Results

8.1 Sample (arithmetic) mean and standard deviation

Figures 1–3 in Appendix B[16] show values of the performance measure for test software for the calculation of sample (arithmetic) mean for, respectively, the performance parameters reference sample mean, reference sample standard deviation, and sample size (see Table 1).[17] For all reference data sets and all test software, the performance measure takes values between zero and one. This indicates that the numerical accuracy of the results returned by the test software is comparable to that expected from optimally stable (reference) software for this calculation and these data sets.

Figures 4–6 in Appendix C show the results for test software for the calculation of sample standard deviation and the same performance parameters. For all reference data sets and all test software, *excepting* the Microsoft Excel function, the performance measure takes values between zero and one. For the performance parameter reference sample mean, the values of the performance measure for the Microsoft Excel function exhibit a tendency to *increase* with the performance parameter (Figure 4).

The sequence of reference data sets corresponding to the performance parameter reference sample mean constitutes a *graded* sequence because the measure $\kappa(s)$ of "degree of difficulty", given by

$$\kappa(s) = \max\left\{\frac{\left|\bar{x}^{\mathrm{ref}}\right|}{s^{\mathrm{ref}}}, 1\right\},$$

is an increasing function of the performance parameter $\bar{x}^{\mathrm{ref}}$ (Section 6). For the performance measure to increase for such a sequence provides (qualitative) evidence that the test software implements a *numerically unstable* formula for the calculation of sample standard deviation. For the (particular) reference data set for which the reference sample mean is 10^7, the measure $\kappa(s) = 10^7$ and we can expect up to seven significant figures of accuracy to be lost in the result returned by reference software. For this data set the value of the performance measure for the Microsoft Excel function is (approximately) seven and, consequently, fourteen significant figures of accuracy are lost in the result returned by the test software. This provides further (quantitative) evidence that the numerical performance of the test software is poor.[18]

---

[16] In the figures in this appendix, some of the results do not appear as they are "hidden behind" others. For example, in Figure 1, the results for the NAG library are hidden behind those for the IMSL library (which occurs below the NAG library in the list of results).

[17] In the figures in the appendices, the values of the performance measures are joined by straight lines. The lines have no meaning and are included only for visualisation purposes. Furthermore, it should be noted that for many of the graphs the scale of the "independent" variable (performance parameter) is non-uniform. Consequently, care is required in the interpretation of the straight-line segments.

[18] Generally, the Microsoft Excel function for sample standard deviation loses twice as many significant figures of accuracy compared with reference software.

The observed behaviour is known to arise when the sample standard deviation is implemented using the (one-pass) formula

$$s = \sqrt{\frac{1}{m-1}\left(\sum_{i=1}^{m} x_i^2 - m\bar{x}^2\right)},$$

which is also known to be numerically unstable. The results suggest that it is likely that the Microsoft Excel function implements the above formula, whereas the other packages implement stable algorithms for the calculation of sample standard deviation.

8.2 Straight-line (ordinary) regression

Figures 7–11 in Appendix D show values for the performance measure for test software for straight-line (ordinary) regression for the performance parameters listed in Table 2. Generally (Figures 8–11), the performance measure takes values between zero and one, with values between one and two for a small number of reference data sets (Figure 9), indicating that the test software is behaving essentially like reference software for this computation and these data sets.

However, the results illustrated in Figure 7 include values for the performance measure for certain of the test software that (a) exceed zero (appreciably), and (b) exhibit a tendency to increase with the performance parameter (the coordinate of the centroid of the data x-values). Furthermore, for the final reference data set in this sequence (for which the coordinate of the centroid of the data x-values is 10^7), two of the functions tested either fail to return a test result (S-PLUS) or provide a warning to the user about the reliability of the test result (Matlab A).

There are two issues for test software for straight-line (ordinary) regression. The first is the *parametrisation* for the straight-line model, for example whether the model is written in terms of unnormalised or normalised variables (Section 3.2). The second is the *numerical algorithm* used to solve the least-squares problem, for example whether this is done by forming and solving the normal equations or in terms of a matrix factorisation of the observation (design) matrix [20, chapter 4].

The results for NAG A and NAG B (and, similarly, for Matlab A and Matlab B) illustrate the issue of parametrisation. The results for test software NAG B and Matlab B show that the application of the NAG and Matlab functions to a problem for which the straight-line model is represented in terms of normalised variables returns test results with an accuracy that is comparable to that expected from reference software. However, when applied to a *poorly* parametrised problem, the same functions return test results that do not achieve the same level of accuracy.[19] The results suggest that it is likely that the IMSL, S-PLUS and Java (JLAPACK) test software behave in a similar way. The results for Microsoft Excel and Java (JNL), however, are (probably) indicative of the numerical algorithm used to solve the least-squares problem.[20] For these the accuracy of the test results degrades at a faster rate than for the NAG and Matlab functions (applied to a poorly parametrised problem). It is likely that the Microsoft Excel and Java (JNL) functions implement an unstable algorithm for straight-line (ordinary) regression.

8.3 Polynomial (ordinary) regression

Figures 12–15 in Appendix E show values for the performance measure for test software for polynomial (ordinary) regression for the performance parameters listed in Table 3. As for the

[19] The issue here is that the test results are the *residual deviations* associated with the solution. Although the residual deviations are *mathematically* identical for all parametrisations of a straight-line model, they will depend *numerically* on the particular parametrisation used (as is shown by these results).

[20] To be certain would require detailed information concerning the algorithm implemented.

straight-line (ordinary) regression problem (a particular example of polynomial regression), the issues of parametrisation and the numerical algorithm used will influence the accuracy of the test results.

The NAG function uses a Chebyshev-series representation of polynomials, Matlab B a power-series representation in terms of a normalised (shifted and scaled) variable (Section 3.3), and Matlab A a power-series representation in terms of the (natural) unnormalised variable [20, chapter 5]. The results for these functions, presented in Figure 12 (for which the performance parameter is the order of the polynomial), show the effect on the accuracy of the test results of using the different parametrisations. It is likely that the test results returned by the Microsoft Excel and LabVIEW functions are further affected by the numerical algorithm used to solve the least-squares problem (see also Figure 14).

The numerical accuracy of the test results is not adversely affected by the number of data points or the size of the measurement standard deviation for any of the test software (Figures 13 and 15).

The results indicate that the NAG and S-PLUS functions are behaving essentially like reference software for this computation and these data sets.

9. Conclusions

In this report we have described the application of a general methodology [1] for testing the numerical correctness of scientific software to functions for the calculations of sample (arithmetic) mean and standard deviation, straight-line (ordinary) regression and polynomial (ordinary) regression. The functions tested are taken from a number of proprietary software packages and libraries, including the NAG and IMSL (FORTRAN) libraries, Microsoft Excel, LabVIEW, S-PLUS, Matlab and various Java numerical libraries.

Each stage of the methodology, from documenting the specifications of the functions tested through the definition of performance parameters and performance measures to the presentation and interpretation of the results of the testing, has been described. In this way, and by stating any assumptions made in the application of the methodology, the testing undertaken is made repeatable and as objective as possible given the ("black-box") nature of the testing.

The results obtained and conclusions drawn from the testing undertaken must be interpreted in the context of the particular calculations considered, and do not necessarily reflect on other functions of the software libraries and packages considered. The black-box testing described here has been carried out in such a way that the functions have been used without taking account of information elsewhere (where it exists), for example, as contained in publications or posted on the World Wide Web. This mode of use is deliberate, since we believe it accords with that adopted by most users generally, and within metrology in particular. Furthermore, it allows for the consistent testing of the functions given the differing quality and quantity of information provided with each.

The results suggest that, even for these common and supposedly straightforward calculations, the (numerical) quality of the software tested is variable. For each calculation considered, test software behaving essentially like reference software for the calculation was identified. Where test software was found to lose more figures of accuracy than reference software, it is likely that the cause of the inaccuracy is a poor choice of model parametrisation and/or a poor choice of numerical formula or algorithm. Such causes of inaccuracy are avoidable, and advice on appropriate avoidance techniques is available [20, 21].

Some particular conclusions are as follows:

- The software tested for the calculation of sample (arithmetic) mean return test results for the reference data sets considered having a numerical accuracy that is *comparable* to that expected from reference software.
- For the calculation of sample standard deviation, a *critical* performance parameter is the (reference) value for the sample mean of the data set. Reference data sets with different values for this performance parameter are used to expose (possible) deficiencies in the formulae used by test software for this calculation. The results of the testing indicate that the Microsoft Excel function implements a numerically unstable formula for the calculation of sample standard deviation, whereas the other software tested implement numerically stable formulae. The results for the Microsoft Excel function show a *systematic* degradation in the numerical performance of this function as this (critical) performance parameter is increased (although it is noted that for two of the reference data sets the function returns test results that "by chance" are numerically accurate).
- For the calculation of straight-line (ordinary) regression, a *critical* parameter is the coordinate of the centroid of the data *x*-values. Reference data sets with different values for this performance parameter are used to expose (possible) deficiencies in the model parametrisations and numerical algorithms used by software to undertake this calculation. It is strongly recommended that the data be centred and scaled (or the straight-line model is represented in terms of a normalised variable) before undertaking straight-line (ordinary) regression. The results of the testing indicate that the functions provided by the NAG and IMSL libraries, Matlab, Java (JLAPACK) and S-PLUS, when applied to data that is normalised as above, return test results for the reference data sets considered having a numerical accuracy that is *comparable* to that expected from reference software.
- For the calculation of polynomial (ordinary) regression, *critical* performance parameters are the order (degree + 1) of the polynomial and the lower endpoint defining the interval containing the data *x*-values. Reference data sets with different values for these performance parameters are used to expose (possible) deficiencies in the model parametrisations and numerical algorithms used by software to undertake this calculation. It is strongly recommended that a Chebyshev-series representation (in terms of a normalised variable) of polynomials be used when undertaking polynomial (ordinary) regression. The results of the testing indicate that the functions provided by the NAG library and S-PLUS return test results for the reference data sets considered having a numerical accuracy that is *comparable* to that expected from reference software.

10. Acknowledgements

This report constitutes one of the deliverables of Project 2.1 of the 2001–2004 NMS *Software Support for Metrology* Programme, and has been funded by the National Measurement System Directorate of the UK Department of Trade and Industry.

11. References

[1] H.R. Cook, M.G. Cox, M.P. Dainton and P.M. Harris. A methodology for testing spreadsheets and other packages used in metrology. *NPL Report CISE 25/99*, 1999.

[2] M.G. Cox, M.P. Dainton and P.M. Harris. Testing functions for the calculation of standard deviation. *NPL Report CMSC 07/00*, 2000.

[3] M.G. Cox, M.P. Dainton and P.M. Harris. Testing functions for linear regression. *NPL*

Report CMSC 08/00, 2000.

[4] S.L.R. Ellison, M.G. Cox, A.B. Forbes, B.P. Butler, S.A. Hannaby, P.M. Harris, and S.M. Hodson. Development of data sets for the validation of analytical instrumentation. *Journal of AOAC International*, **77**(3), pp 1-5, 1994.

[5] Statistics Software Qualification. Edited by B.P. Butler, M.G. Cox, S.L.R. Ellison and W.A. Hardcastle. The Royal Society of Chemistry, 1996.

[6] M.G. Cox and P.M. Harris. Design and use of reference data sets for testing scientific software. *Analytica Chimica Acta* **380**, pp 339 – 351, 1999.

[7] H.R. Cook, M.G. Cox, M.P. Dainton and P.M. Harris. Testing spreadsheets and other packages used in metrology: A case study. *NPL Report CISE 26/99*, September 1999.

[8] The NAG Fortran Library (Mark 20). Copyright © 2001, Numerical Algorithms Group Ltd.

[9] IMSL Fortran 90 MP Library (Version 4.01). Copyright © 1999, Visual Numerics, Inc.

[10] LabVIEW (Version 7.0). Copyright © 2003, National Instruments Corporation.

[11] Matlab (Version 6.5.0.180913a Release 13). Copyright © 1984–2002, The MathWorks, Inc.

[12] Microsoft ® Excel 2000. Copyright © 1985–1999, Microsoft Corporation.

[13] S-PLUS 4.0 (Release 3). Copyright © 1988–1997, MathSoft, Inc.

[14] JNL, a Numerical Library for Java. Copyright © 1997, Visual Numerics, Inc.

[15] D.M. Doolin, J. Dongarra and K. Seymour. JLAPACK – Compiling LAPACK FORTRAN to Java. *Scientific Programming* **7** (2), pp 111 – 138, 1999.

[16] M.G. Cox, P.M. Harris, E.G. Johnson and G.I. Parkin. Testing methods of Java libraries. *NPL Report CMSC 35/04*, January 2004.

[17] L. Fox and I.B. Parker. *Chebyshev Polynomials in Numerical Analysis*. Oxford University Press, 1968.

[18] G.H. Golub and C.F. Van Loan. *Matrix Computations (third edition)*. John Hopkins University Press, 1996.

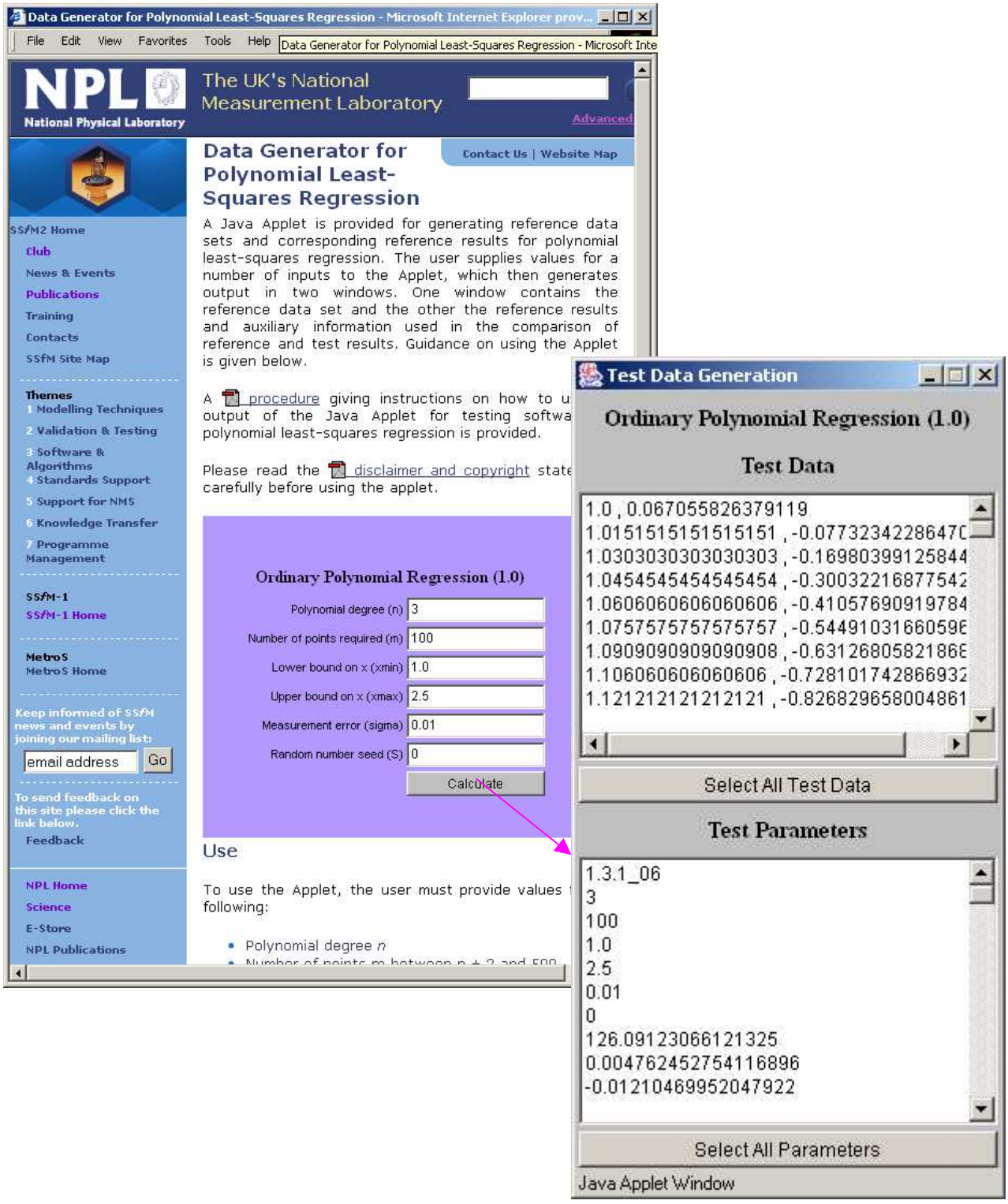
[19] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammerling, A. McKenny, S. Ostouchov and D. Dorensen. *LAPACK User's Guide, Second Edition*. SIAM, Philadelphia, PA, 1995.

[20] M.G. Cox, A.B. Forbes and P.M. Harris. *Software Support for Metrology Best Practice Guide No.4: Discrete Modelling*, National Physical Laboratory, 2002.

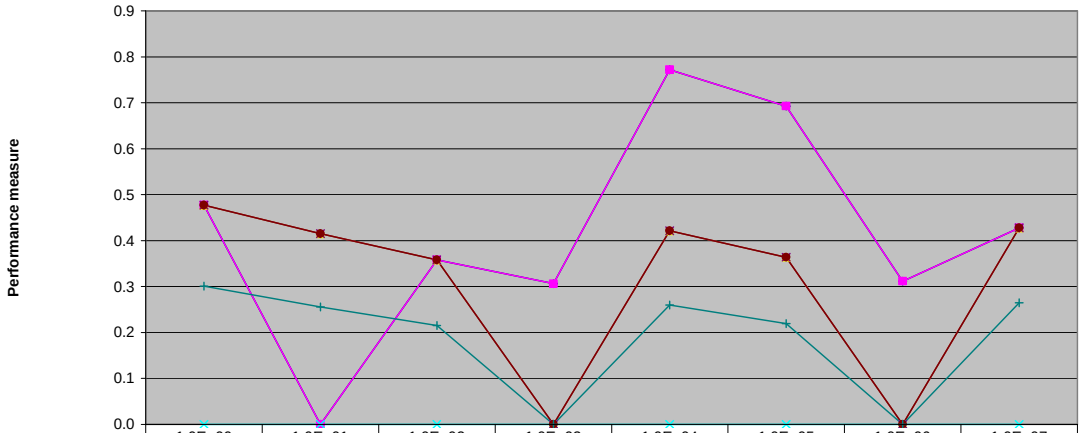
[21] M.G. Cox and P.M. Harris. Guidelines to help users select and use software for their metrology applications. *NPL Report CMSC 04/00*, 2000.

Appendix A: Interface to Web-Based Data Generators

Illustrated below is the Applet for generating a reference data set and corresponding reference results for the computation of the polynomial (ordinary) regression. The two windows generated by the Applet containing “test data” and “test parameters” are also shown.



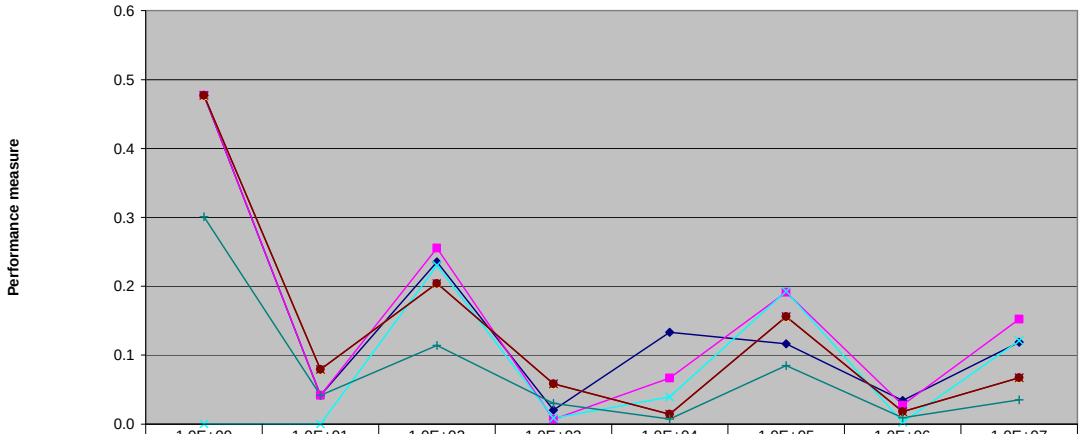
Appendix B: Results for Sample Mean



	1.0E+00	1.0E+01	1.0E+02	1.0E+03	1.0E+04	1.0E+05	1.0E+06	1.0E+07
NAG	0.5	0.0	0.4	0.3	0.8	0.7	0.3	0.4
IMSL	0.5	0.0	0.4	0.3	0.8	0.7	0.3	0.4
VB/Excel	0.5	0.4	0.4	0.0	0.4	0.4	0.0	0.4
LabVIEW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S-PLUS	0.5	0.4	0.4	0.0	0.4	0.4	0.0	0.4
Java (JNL)	0.5	0.4	0.4	0.0	0.4	0.4	0.0	0.4
MatLab	0.3	0.3	0.2	0.0	0.3	0.2	0.0	0.3

Sample mean

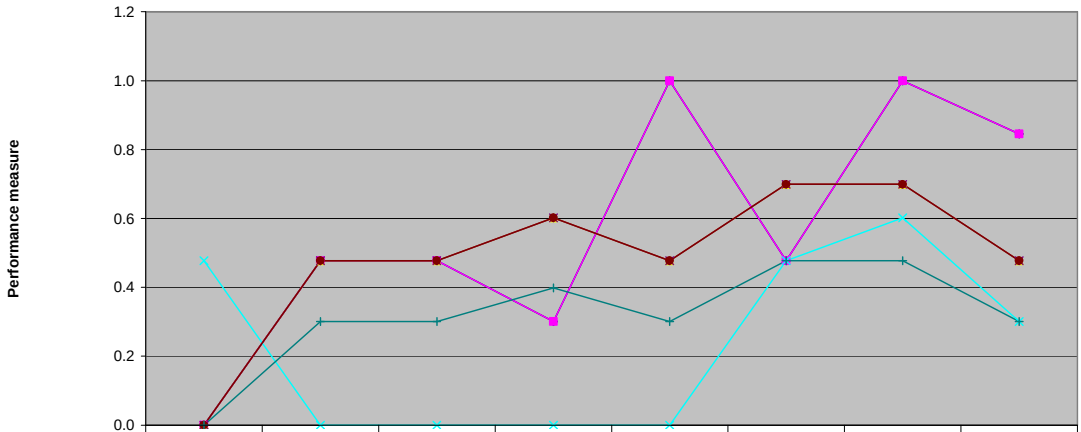
Figure 1: Values of the performance measure for test software for the computation of the sample (arithmetic) mean. The performance parameter is the reference value for the sample mean. Note the NAG results are identical to those of IMSL, and the Microsoft Excel and S-PLUS results are identical to those of Java (JNL).



	1.0E+00	1.0E+01	1.0E+02	1.0E+03	1.0E+04	1.0E+05	1.0E+06	1.0E+07
NAG	0.5	0.0	0.2	0.0	0.1	0.1	0.0	0.1
IMSL	0.5	0.0	0.3	0.0	0.1	0.2	0.0	0.2
VB/Excel	0.5	0.1	0.2	0.1	0.0	0.2	0.0	0.1
LabVIEW	0.0	0.0	0.2	0.0	0.0	0.2	0.0	0.1
S-PLUS	0.5	0.1	0.2	0.1	0.0	0.2	0.0	0.1
Java (JNL)	0.5	0.1	0.2	0.1	0.0	0.2	0.0	0.1
MatLab	0.3	0.0	0.1	0.0	0.0	0.1	0.0	0.0

Sample standard deviation

Figure 2: As Figure 1 except that the performance parameter is the reference value for the sample standard deviation. Note the Microsoft Excel and S-PLUS results are identical to those of Java (JNL).

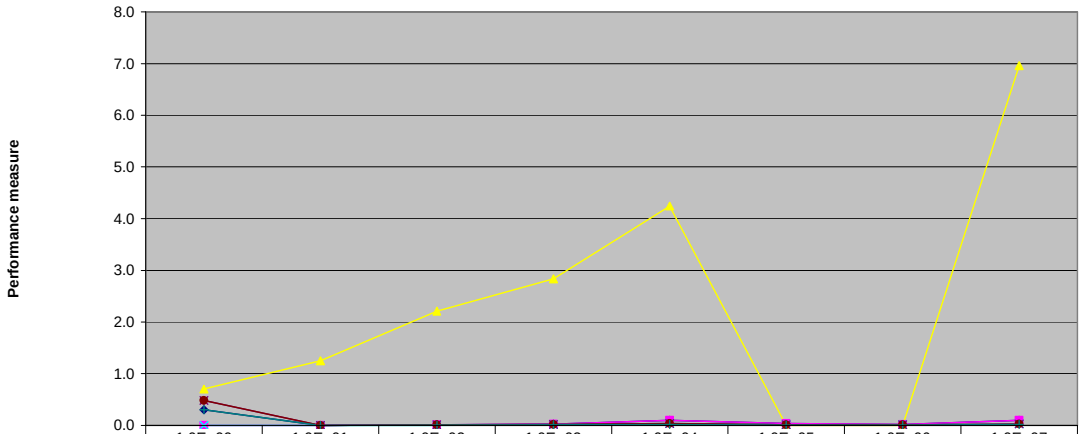


	10	50	100	150	200	300	400	500
NAG	0.0	0.5	0.5	0.3	1.0	0.5	1.0	0.8
IMSL	0.0	0.5	0.5	0.3	1.0	0.5	1.0	0.8
VB/Excel	0.0	0.5	0.5	0.6	0.5	0.7	0.7	0.5
LabVIEW	0.5	0.0	0.0	0.0	0.0	0.5	0.6	0.3
S-PLUS	0.0	0.5	0.5	0.6	0.5	0.7	0.7	0.5
Java (JNL)	0.0	0.5	0.5	0.6	0.5	0.7	0.7	0.5
MatLab	0.0	0.3	0.3	0.4	0.3	0.5	0.5	0.3

Number of points

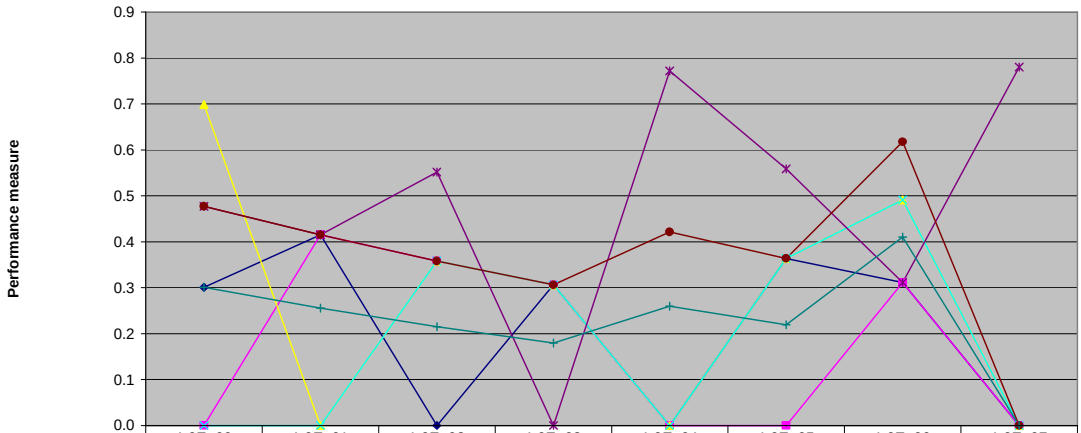
Figure 3: As Figure 1 except that the performance parameter is the sample size. Note the NAG results are identical to those of IMSL, and the Microsoft Excel and S-PLUS results are identical to those of Java (JNL).

Appendix C: Results for Sample Standard Deviation



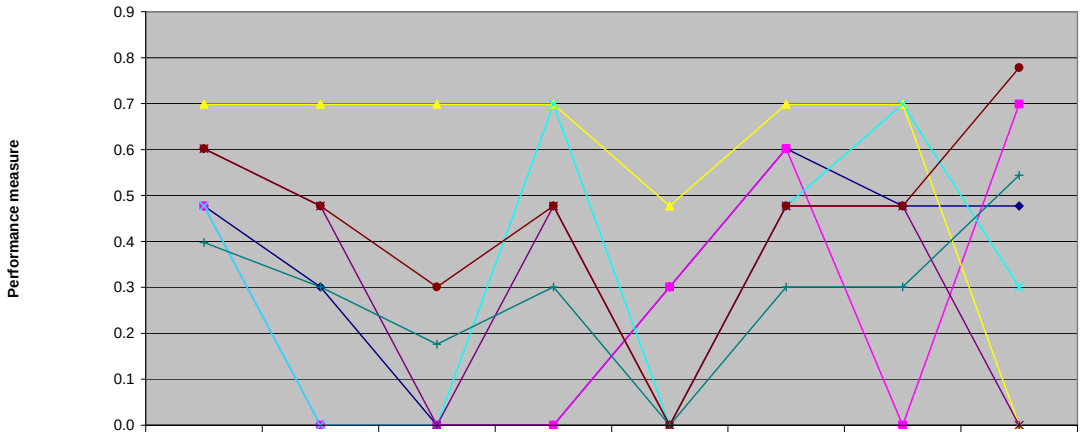
	1.0E+00	1.0E+01	1.0E+02	1.0E+03	1.0E+04	1.0E+05	1.0E+06	1.0E+07
NAG	0.3	0.0	0.0	0.0	0.1	0.0	0.0	0.1
IMSL	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1
VB/Excel	0.7	1.3	2.2	2.8	4.2	0.0	0.0	7.0
LabVIEW	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S-PLUS	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Java (JNL)	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MatLab	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 4: Values of the performance measure for test software for the computation of the sample standard deviation. The performance parameter is the reference value for the sample mean.



	1.0E+00	1.0E+01	1.0E+02	1.0E+03	1.0E+04	1.0E+05	1.0E+06	1.0E+07
NAG	0.3	0.4	0.0	0.3	0.0	0.4	0.3	0.0
IMSL	0.0	0.4	0.4	0.3	0.0	0.0	0.3	0.0
VB/Excel	0.7	0.0	0.4	0.3	0.0	0.4	0.5	0.0
LabVIEW	0.0	0.0	0.4	0.3	0.0	0.4	0.5	0.0
S-PLUS	0.5	0.4	0.6	0.0	0.8	0.6	0.3	0.8
Java (JNL)	0.5	0.4	0.4	0.3	0.4	0.4	0.6	0.0
MatLab	0.3	0.3	0.2	0.2	0.3	0.2	0.4	0.0

Figure 5: As Figure 4 except that the performance parameter is the reference value for the sample standard deviation.

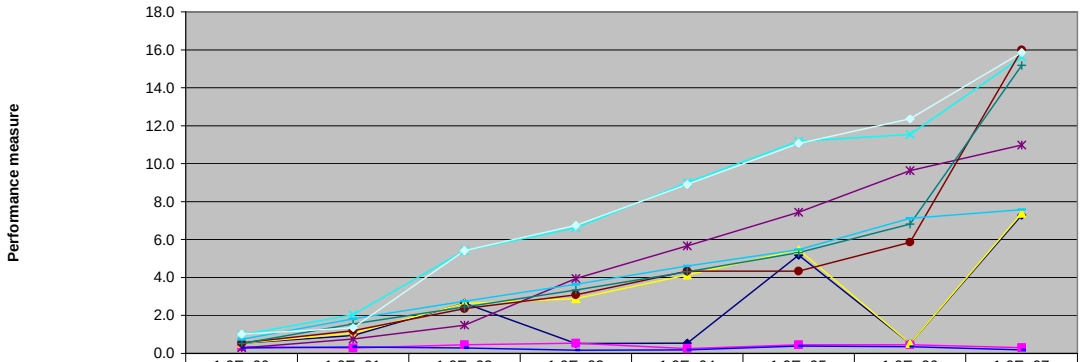


	10	50	100	150	200	300	400	500
NAG	0.5	0.3	0.0	0.0	0.3	0.6	0.5	0.5
IMSL	0.5	0.0	0.0	0.0	0.3	0.6	0.0	0.7
VB/Excel	0.7	0.7	0.7	0.7	0.5	0.7	0.7	0.0
LabVIEW	0.5	0.0	0.0	0.7	0.0	0.5	0.7	0.3
S-PLUS	0.6	0.5	0.0	0.5	0.0	0.5	0.5	0.0
Java (JNL)	0.6	0.5	0.3	0.5	0.0	0.5	0.5	0.8
MatLab	0.4	0.3	0.2	0.3	0.0	0.3	0.3	0.5

Number of points

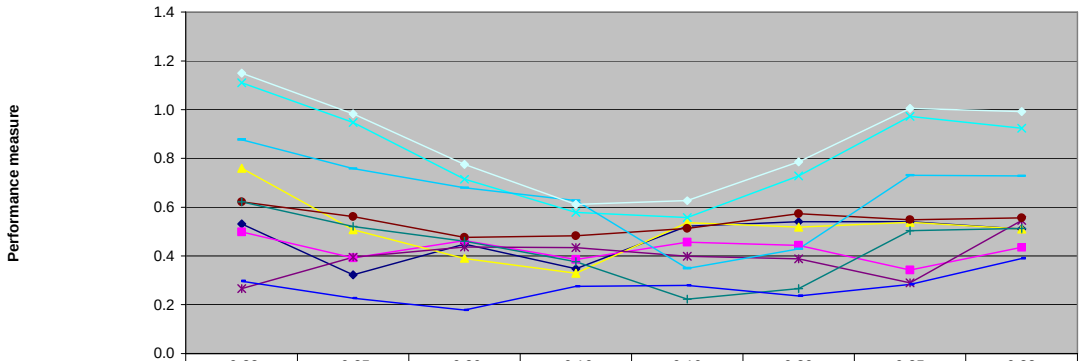
Figure 6: As Figure 4 except that the performance parameter is the sample size.

Appendix D: Results for Straight-Line (Ordinary) Regression



	1.0E+00	1.0E+01	1.0E+02	1.0E+03	1.0E+04	1.0E+05	1.0E+06	1.0E+07
NAG A	0.5	0.9	2.6	0.5	0.5	5.2	0.5	7.3
NAG B	0.3	0.3	0.4	0.5	0.2	0.5	0.4	0.3
IMSL	0.5	1.1	2.6	2.9	4.1	5.5	0.5	7.4
VB/Excel	1.0	2.0	5.4	6.6	9.0	11.2	11.5	15.6
LabVIEW	0.3	0.8	1.5	3.9	5.7	7.4	9.6	11.0
S-PLUS	0.5	1.2	2.3	3.1	4.3	4.3	5.9	16.0
Matlab A	0.5	1.5	2.4	3.3	4.3	5.3	6.8	15.2
Matlab B	0.3	0.3	0.3	0.2	0.2	0.4	0.3	0.2
Java (JLAPACK)	0.7	1.8	2.7	3.6	4.6	5.5	7.1	7.6
Java (JNL)	1.0	1.3	5.4	6.7	8.9	11.1	12.4	15.9

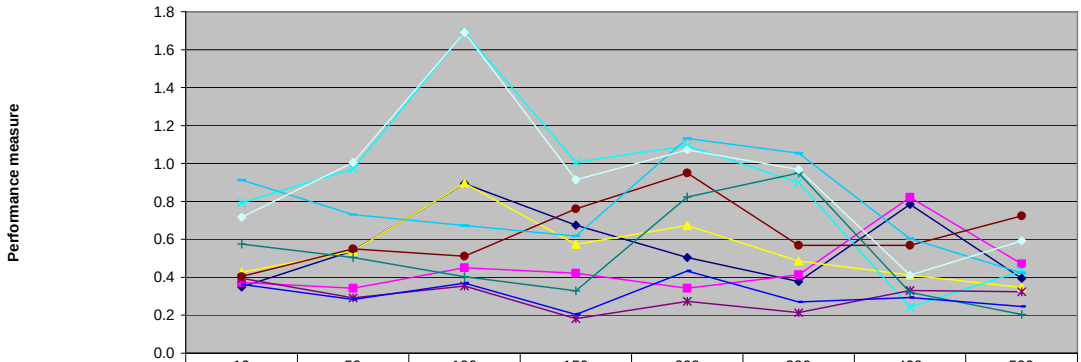
Figure 7: Values of the performance measure for test software for the computation of straight-line (ordinary) regression. The performance parameter is the coordinate of the centroid of the data *x*-values.[21]



	-0.33	-0.25	-0.20	-0.10	0.10	0.20	0.25	0.33
NAG A	0.5	0.3	0.4	0.3	0.5	0.5	0.5	0.5
NAG B	0.5	0.4	0.5	0.4	0.5	0.4	0.3	0.4
IMSL	0.8	0.5	0.4	0.3	0.5	0.5	0.5	0.5
VB/Excel	1.1	0.9	0.7	0.6	0.6	0.7	1.0	0.9
LabVIEW	0.3	0.4	0.4	0.4	0.4	0.4	0.3	0.5
S-PLUS	0.6	0.6	0.5	0.5	0.5	0.6	0.5	0.6
Matlab A	0.6	0.5	0.5	0.4	0.2	0.3	0.5	0.5
Matlab B	0.3	0.2	0.2	0.3	0.3	0.2	0.3	0.4
Java (JLAPACK)	0.9	0.8	0.7	0.6	0.3	0.4	0.7	0.7
Java (JNL)	1.1	1.0	0.8	0.6	0.6	0.8	1.0	1.0

Figure 8: As Figure 7 except that the performance parameter is the factor defining the slope of the solution line.

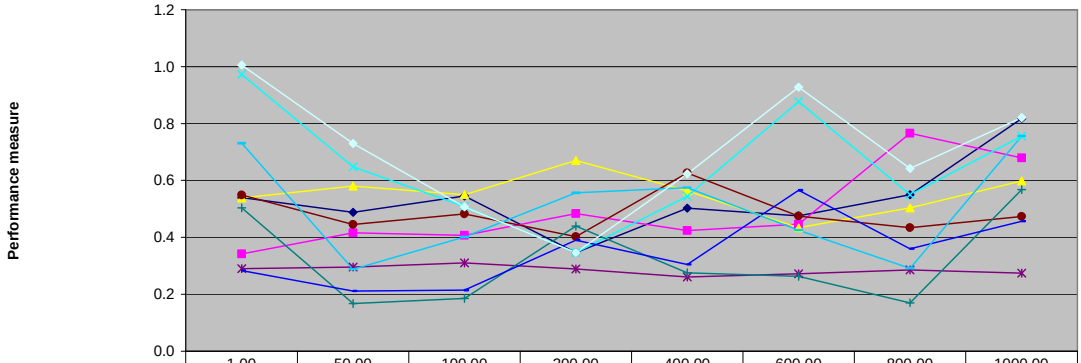
[21] For the particular reference data set for which the coordinate of the centroid of the data *x*-values is 10^7, S-PLUS does not return a test result and Matlab A provides a warning to the user about the reliability of the test result.



	10	50	100	150	200	300	400	500
NAG A	0.3	0.5	0.9	0.7	0.5	0.4	0.8	0.4
NAG B	0.4	0.3	0.4	0.4	0.3	0.4	0.8	0.5
IMSL	0.4	0.5	0.9	0.6	0.7	0.5	0.4	0.3
VB/Excel	0.8	1.0	1.7	1.0	1.1	0.9	0.2	0.4
LabVIEW	0.4	0.3	0.4	0.2	0.3	0.2	0.3	0.3
S-PLUS	0.4	0.5	0.5	0.8	1.0	0.6	0.6	0.7
Matlab A	0.6	0.5	0.4	0.3	0.8	1.0	0.3	0.2
Matlab B	0.4	0.3	0.4	0.2	0.4	0.3	0.3	0.2
Java (JLAPACK)	0.9	0.7	0.7	0.6	1.1	1.1	0.6	0.4
Java (JNL)	0.7	1.0	1.7	0.9	1.1	1.0	0.4	0.6

Number of points

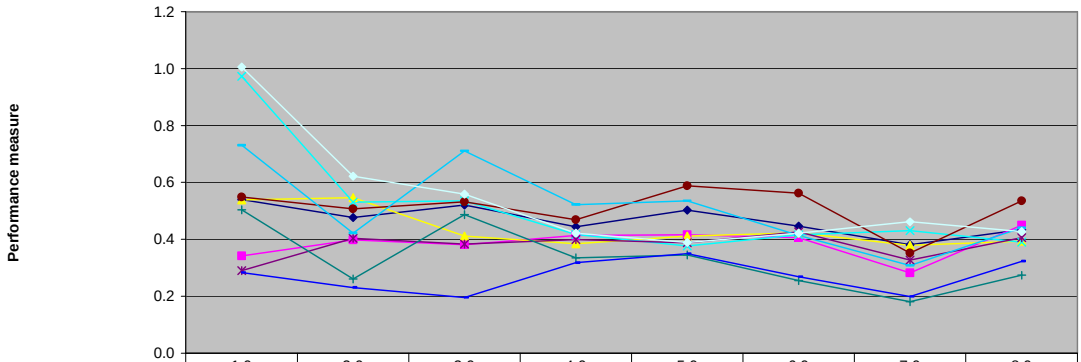
Figure 9: As Figure 7 except that the performance parameter is the number of data points.



	1.00	50.00	100.00	200.00	400.00	600.00	800.00	1000.00
NAG A	0.5	0.5	0.5	0.3	0.5	0.5	0.5	0.8
NAG B	0.3	0.4	0.4	0.5	0.4	0.4	0.8	0.7
IMSL	0.5	0.6	0.6	0.7	0.6	0.4	0.5	0.6
VB/Excel	1.0	0.6	0.5	0.3	0.5	0.9	0.5	0.8
LabVIEW	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
S-PLUS	0.5	0.4	0.5	0.4	0.6	0.5	0.4	0.5
Matlab A	0.5	0.2	0.2	0.4	0.3	0.3	0.2	0.6
Matlab B	0.3	0.2	0.2	0.4	0.3	0.6	0.4	0.5
Java (JLAPACK)	0.7	0.3	0.4	0.6	0.6	0.4	0.3	0.8
Java (JNL)	1.0	0.7	0.5	0.3	0.6	0.9	0.6	0.8

Length of interval containing *x* data

Figure 10: As Figure 7 except that the performance parameter is the length of the interval containing the data *x*-values.

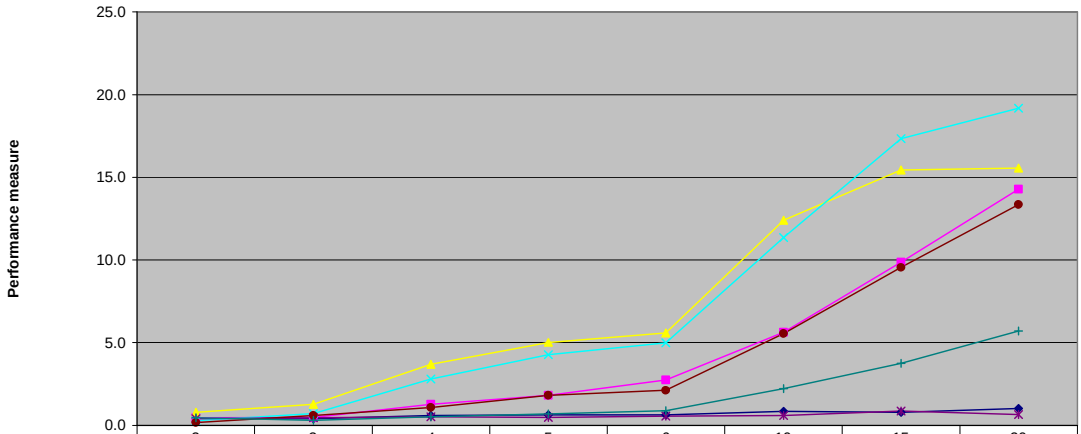


	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0
NAG A	0.5	0.5	0.5	0.4	0.5	0.4	0.4	0.4
NAG B	0.3	0.4	0.4	0.4	0.4	0.4	0.3	0.5
IMSL	0.5	0.5	0.4	0.4	0.4	0.4	0.4	0.4
VB/Excel	1.0	0.5	0.5	0.4	0.4	0.4	0.4	0.4
LabVIEW	0.3	0.4	0.4	0.4	0.4	0.4	0.3	0.4
S-PLUS	0.5	0.5	0.5	0.5	0.6	0.6	0.4	0.5
Matlab A	0.5	0.3	0.5	0.3	0.3	0.3	0.2	0.3
Matlab B	0.3	0.2	0.2	0.3	0.3	0.3	0.2	0.3
Java (JLAPACK)	0.7	0.4	0.7	0.5	0.5	0.4	0.3	0.4
Java (JNL)	1.0	0.6	0.6	0.4	0.4	0.4	0.5	0.4

Measurement standard deviation

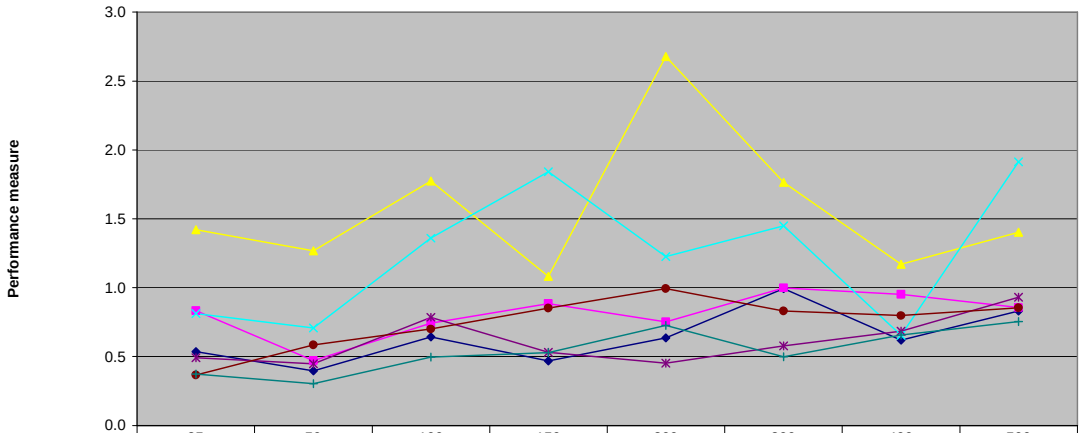
Figure 11: As Figure 7 except that the performance parameter is the measurement standard deviation.

Appendix E: Results for Polynomial (Ordinary) Regression



	2	3	4	5	6	10	15	20
NAG	0.4	0.4	0.6	0.6	0.6	0.8	0.8	1.0
IMSL	0.5	0.5	1.3	1.8	2.7	5.6	9.9	14.3
VB/Excel	0.8	1.3	3.7	5.0	5.6	12.4	15.4	15.6
LabVIEW	0.3	0.7	2.8	4.3	5.0	11.4	17.3	19.2
S-PLUS	0.4	0.4	0.5	0.5	0.5	0.6	0.9	0.7
Matlab A	0.2	0.6	1.1	1.8	2.1	5.5	9.5	13.4
Matlab B	0.5	0.3	0.5	0.7	0.9	2.2	3.7	5.7

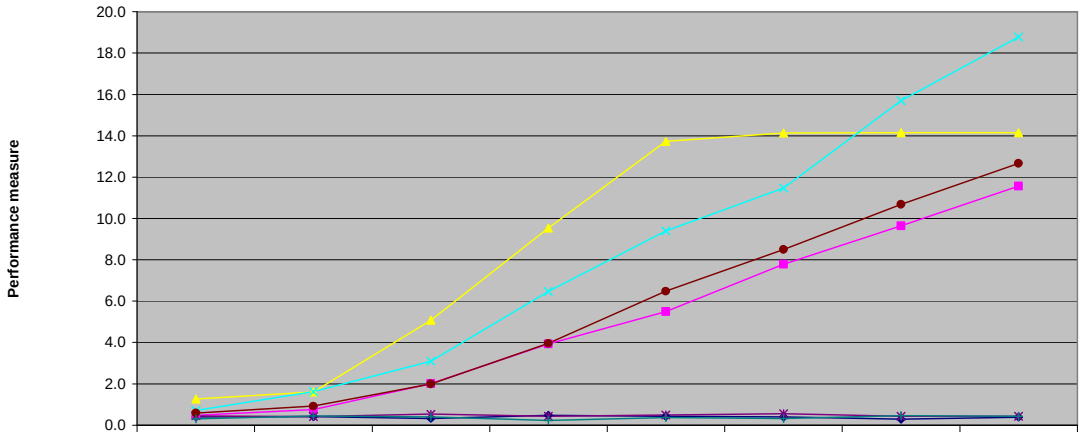
Figure 12: Values of the performance measure for test software for the computation of polynomial (ordinary) regression. The performance parameter is the order (degree + 1) of the polynomial.[22]



	25	50	100	150	200	300	400	500
NAG	0.5	0.4	0.6	0.5	0.6	1.0	0.6	0.8
IMSL	0.8	0.5	0.7	0.9	0.8	1.0	1.0	0.9
VB/Excel	1.4	1.3	1.8	1.1	2.7	1.8	1.2	1.4
LabVIEW	0.8	0.7	1.4	1.8	1.2	1.4	0.7	1.9
S-PLUS	0.5	0.4	0.8	0.5	0.5	0.6	0.7	0.9
Matlab A	0.4	0.6	0.7	0.9	1.0	0.8	0.8	0.9
Matlab B	0.4	0.3	0.5	0.5	0.7	0.5	0.7	0.8

Figure 13: As Figure 12 except that the performance parameter is the number of data points.

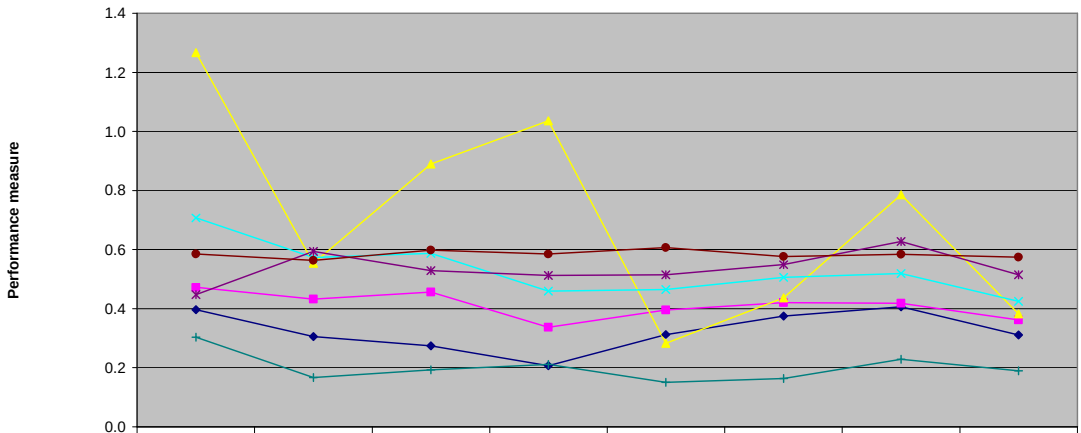
[22] For the last three reference data sets in this sequence (corresponding to orders 10, 15 and 20), Matlab A provides a warning to the user about the reliability of the test result.



	1.0E+00	1.0E+01	1.0E+02	1.0E+03	1.0E+04	1.0E+05	1.0E+06	1.0E+07
NAG	0.4	0.4	0.3	0.5	0.4	0.4	0.3	0.4
IMSL	0.5	0.8	2.0	3.9	5.5	7.8	9.6	11.6
VB/Excel	1.3	1.6	5.1	9.5	13.7	14.1	14.1	14.1
LabVIEW	0.7	1.6	3.1	6.5	9.4	11.5	15.7	18.8
S-PLUS	0.4	0.4	0.5	0.4	0.5	0.6	0.4	0.4
Matlab A	0.6	0.9	2.0	4.0	6.5	8.5	10.7	12.7
Matlab B	0.3	0.4	0.4	0.2	0.4	0.3	0.5	0.4

Lower endpoint of interval containing the data

Figure 14: As Figure 12 except that the performance parameter is the lower endpoint defining the interval containing the data *x*-values.[23]



	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0
NAG	0.4	0.3	0.3	0.2	0.3	0.4	0.4	0.3
IMSL	0.5	0.4	0.5	0.3	0.4	0.4	0.4	0.4
VB/Excel	1.3	0.6	0.9	1.0	0.3	0.4	0.8	0.4
LabVIEW	0.7	0.6	0.6	0.5	0.5	0.5	0.5	0.4
S-PLUS	0.4	0.6	0.5	0.5	0.5	0.5	0.6	0.5
Matlab A	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Matlab B	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2

Measurement standard deviation

Figure 15: As Figure 12 except that the performance parameter is the measurement standard deviation.

[23] For the last five reference data sets in this sequence (corresponding to lower-endpoint values ranging from 10^3 to 10^7), Matlab A provides a warning to the user about the reliability of the test result.