

**NPL REPORT
DEM-ES 004
(CMSC 34/04 revised)**

**Testing the numerical
correctness of software**

**P M Harris, K J Lines and
I M Smith**

NOT RESTRICTED

March 2006

Testing the numerical correctness of software

P M Harris, K J Lines and I M Smith
Division of Enabling Metrology

ABSTRACT

This report describes 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.

This document is a revised edition of a previous report. It contains the results of testing more recent versions of packages previously tested (Matlab, LabVIEW and Microsoft Excel) and also the results of testing additional packages (Origin and Mathcad).

This report constitutes one of the deliverables of Project 3.2 *Numerical Software Testing* within the UK Department of Industry's National Measurement System *Software Support for Metrology* Programme 2004–2007.

© Crown copyright 2006
Reproduced with the permission of the Controller of HMSO
and Queen's Printer for Scotland

ISSN 1744-0475

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

Extracts from this report may be reproduced provided the source is acknowledged and the extract is not taken out of context.

Approved on behalf of the Managing Director, NPL
by Jonathan Williams, Knowledge Leader of the Electrical and Software Team

Contents

1	Introduction.....	1
2	Methodology.....	2
3	Specification of Test Software.....	3
3.1	Sample mean and standard deviation.....	3
3.2	Straight-line regression.....	4
3.3	Polynomial regression.....	8
4	Interfacing to the Test Software.....	11
5	Specification of Reference Data Sets.....	12
6	Specification of Performance Measures	13
7	Generation of Reference Pairs	15
7.1	Sample mean and standard deviation.....	16
7.1.1	Inputs	16
7.1.2	Outputs.....	16
7.1.3	Procedure	16
7.2	Straight-line regression.....	17
7.2.1	Inputs	17
7.2.2	Outputs.....	17
7.2.3	Procedure	17
7.3	Polynomial regression.....	18
7.3.1	Inputs	18
7.3.2	Outputs.....	18
7.3.3	Procedure	18
7.4	Residual deviations.....	19
7.4.1	Inputs	19
7.4.2	Outputs.....	19
7.4.3	Procedure	19
8	Presentation and Interpretation of Results.....	20
8.1	Sample mean and standard deviation.....	20
8.2	Straight-line regression.....	21
8.3	Polynomial regression.....	22
9	Conclusions.....	23
	Acknowledgements.....	24
	References.....	24
	Appendix A: Results for Sample Mean	27
	Appendix B: Results for sample standard deviation.....	30
	Appendix C: Results for Straight-Line Regression	33
	Appendix D: Results for Polynomial Regression.....	38
	Appendix E: Results for LabVIEW Polynomial Regression.....	42

1 Introduction

The work described here constitutes a continuation of work undertaken as part of the first and second Software Support for Metrology (SSfM) programmes concerned with testing the numerical correctness of software for computations identified as important to metrology and provided as components within proprietary software packages and libraries. It builds upon previous work [1, 2, 3, 4], and extends that work by updating the results for more recent releases of some of the software packages previously tested and by considering additional packages.

In the first SSfM programme, a general methodology for testing the correctness of scientific software was promoted. The methodology is based on the design and use of reference data sets and corresponding reference results to undertake “black box” testing of the software, together with the use of quality metrics and performance measures to make objective comparisons between reference results and the results returned by the software under test for the reference data sets.

In the second SSfM programme, a web-based facility was provided to enable users to generate, for a number of key computations, reference data sets and corresponding reference results appropriate to their own applications. The facility takes the form of data generators implemented in Java in such a way as to provide portability of the generators (across computer platforms), as well as flexibility and reproducibility of the reference data sets generated. The facility was further developed in the third SSfM programme to provide the functionality of a testing service. The facility has two modes of operation. In the first mode, the user is provided with reference data sets with corresponding reference results. In the second mode, the user is provided only with reference data sets, but may upload test results to be compared with the corresponding reference results (which are hidden from the user).

The testing described in this report relies on reference data sets generated using the above web-based facility. 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.

This report describes the application of a general methodology, supported by the above web-based facility, for testing the numerical correctness of scientific software taken from a number of proprietary software packages and libraries for the following computations:

- calculation of sample (arithmetic) mean and standard deviation,
- straight-line (ordinary) regression, and
- polynomial (ordinary) regression.

This report is a revised edition of a previous report [4]. Results are presented for more recent releases of the Matlab, Microsoft Excel and LabVIEW software packages than those considered in the earlier report, while additional packages are included, namely Origin, a spreadsheet-based program for data analysis and presentation, and Mathcad, a calculation package widely used in metrology and engineering. Results in the previous report for the NAG and IMSL Fortran libraries and the JNL and JLAPACK Java libraries are reproduced in this report. Results for the S-PLUS software package are no longer presented since S-PLUS is not widely used at the National Physical Laboratory.

The report is organised as follows. Section 2 gives an overview of the 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 software¹ 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* 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 and papers [1, 5, 6, 7], and is illustrated by a case study [8].

The testing procedure is divided into the following stages:

1. documenting the specification of the test software,
2. specifying reference data sets,
3. specifying performance measures and testing requirements,
4. generating reference pairs, i.e., reference data and corresponding reference results,
5. interfacing to the test software and providing test results, and
6. determining, presenting and interpreting the performance measures.

These stages are based upon previous recommendations [1] but modified to suit testing from the perspective of a *user* of proprietary software packages and libraries. During the software development process, the first stage would be *specifying the test software*, and the fifth *implementing the test software and providing test results*, though in practice these are carried out with varying degrees of formality. The application of the procedure by a software developer is presented in a case study [8]. Recording the results of the first stage 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 example, for generating reference data sets and corresponding

¹ The term *test software* used throughout this report refers to the software under test, and not the software employed to do the testing.

reference results, and evaluating and presenting quality metrics and performance measures. Therefore, the correctness of the results of the testing (and conclusions inferred from the results) depends on the quality (correctness) of this additional software.

3 Specification of Test Software

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

1. sample mean and standard deviation,
2. straight-line regression, and
3. polynomial regression.

It also lists the software (functions or modules) for undertaking these computations that have been tested.

3.1 Sample mean and standard deviation

Given the sample $\{x_i; i = 1, \dots, m\}$, the sample 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 calculations of the sample mean and standard deviation are important for metrology in the context of uncertainty evaluation [9]. The mean provides the best estimate of a quantity for which independent realisations are available, and the standard deviation is used to characterise the variability of the (individual) indications.

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 mean and standard deviation is considered:²

- NAG [10].

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 [11].

² 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.

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.

- Microsoft Excel [12].

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

There are two ways of accessing functions for mean and standard deviation:³

- As calls in Visual Basic code – computations undertaken in this way are referred to as *VB/Excel*.
- As worksheet functions – computations undertaken in this way are referred to as *Excel*.

- Origin [13].

Storing a sample of observations in a column (or row) and selecting

Statistics

Descriptive Statistics > Statistics on Columns (or Statistics on Rows)

returns a variety of basic statistics for the sample including the mean, standard deviation, standard error of the mean, coefficient of kurtosis, and the minimum and maximum values.

- Java JNL [14].

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

- Matlab [15].

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

- Mathcad [16].

Functions **mean** and **Stdev** return, respectively, the mean and standard deviation for a sample of observations.

- LabVIEW [17].

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

3.2 Straight-line regression

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

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

the straight-line regression problem is to solve

$$\min_{b_1, b_2} \sum_{i=1}^m (y_i - (b_1 + b_2 x_i))^2.$$

³ The previous report considered only VB/Excel computations.

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

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

in the case that the observation (design) matrix $\mathbf{A}$ consists of the row vectors

$$\mathbf{a}_i^T = (1, x_i), \quad i = 1, \dots, m,$$

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

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

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

evaluated at the solution are used here as the results. The residual deviations are (mathematically) *independent* of the particular form (parametrisation) of straight-line model used by the test software to solve the regression problem.

The straight-line regression problem is important to metrology because of its application in the modelling of measurement data in situations where the relationship between the stimulus (independent) and response (dependent) variables may be represented by a straight line, and the measured values of the stimulus variable have associated uncertainties that are negligible compared with the uncertainties associated with the measured values of the response variable.⁴

The following test software for straight-line regression is considered:

- NAG [10].

Function **G02CAF** performs straight-line regression to data, returning values for the slope and intercept of the fitted straight line. The residual deviations defining the results are calculated from the slope and intercept values.⁵

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 find $\hat{b}_1$ and $\hat{b}_2$ that solve

$$\min_{b_1, b_2} \sum_{i=1}^m (y_i - (b_1 + b_2 x_i))^2$$

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

$$d_i = y_i - (\hat{b}_1 + \hat{b}_2 x_i), \quad i = 1, \dots, m.$$

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

⁴ Hence, the problem is described as one of “ordinary” regression.

⁵ 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.

G02CAF is used to find $\hat{c}_1$ and $\hat{c}_2$ that solve

$$\min_{c_1, c_2} \sum_{i=1}^m (\tilde{y}_i - (c_1 + c_2 \tilde{x}_i))^2,$$

where

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

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

$$d_i = \tilde{y}_i - (\hat{c}_1 + \hat{c}_2 \tilde{x}_i), \quad i = 1, \dots, m.$$

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 [11].

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 results are calculated from the slope and intercept values.⁶

- 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 results are calculated from the values of the trend line.

There are two ways of accessing functions for straight-line regression:⁷

- As calls in Visual Basic code – computations undertaken in this way are referred to as *VB/Excel*.
- As worksheet functions – computations undertaken in this way are referred to as *Excel*.

- Origin [13].

The **Linear Fit tool** returns the values of a least-squares linear fit to data at the points x_i , $i = 1, \dots, m$. The residual deviations defining the results are also returned.

- Java LAPACK [18].

Function **dgelsx** computes the (minimum-norm) solution to the (real) linear least-squares system of equations $\mathbf{y} = \mathbf{A}\mathbf{b}$. As with the use of the Matlab “backslash” operator (below), it is necessary to assign the observation (design) matrix $\mathbf{A}$ prior to calling the function **dgelsx**. Following the call to the function, the vector of residual deviations $\mathbf{d} = (d_1, \dots, d_m)^T$ is then calculated from the solution values (for slope and intercept) returned by the function. Both operations (assigning $\mathbf{A}$ and calculating $\mathbf{d}$)

⁶ The BLAS routines **DCOPY** (copy a real vector), **DSCAL** (multiply a real vector by a scalar) and **DADD** (add a constant to a real vector) provided with the IMSL library are used for this purpose.

⁷ The previous report considered only VB/Excel computations.

are undertaken in Java.⁸

- 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 results are calculated from the slope and intercept values.

- Matlab [15].

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

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 results of 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 variable x is normalised.

- Mathcad [16].

The functions **slope** and **intercept** return, respectively, the slope and intercept of a least-squares straight line fit to given data.

The computation is undertaken in two ways.

Firstly, the (Mathcad) statements

⁸ Unlike the use of the NAG and Matlab functions, the computation is undertaken in one way only, viz., in terms of *unnormalised* variables.

a := intercept(x,y)

b := slope(x,y)

fit(x) := a + b.x

d := y – fit(x)

are used, where $\mathbf{x}$ is an $m \times 1$ vector containing the x_i -values, and $\mathbf{y}$ an $m \times 1$ vector containing the y_i -values. The first and second statements assign, respectively, the intercept and slope of the best-fit line. The third statement evaluates the best-fit line at each of the x_i -values. The fourth statement evaluates the vector $\mathbf{d}$ of residual deviations defining the results of the regression problem. The computation undertaken in this way is referred to as *Mathcad A*.

Secondly, the (Mathcad) statements

xt := $\frac{(\mathbf{x} - \max(\mathbf{x})) - (\min(\mathbf{x}) - \mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$

a := intercept(xt,y)

b := slope(xt,y)

fit(xt) := a + b.xt

d := y – fit(xt)

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

As with the use of the Matlab function, the above approaches correspond to undertaking the computation in terms of, respectively, an unnormalised and normalised variable, with only the independent variable x normalised.

- LabVIEW [17].

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_i -values. The residual deviations defining the results are calculated from the values of the best-fit line.

3.3 Polynomial regression

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

$$p_n(x, b_1, b_2, \dots, b_{n+1}) = b_1 + b_2x + \dots + b_{n+1}x^n,$$

of (specified) degree n (order $n + 1$), the polynomial regression problem is to solve

$$\min_{b_1, \dots, b_{n+1}} \sum_{i=1}^m (y_i - (b_1 + b_2x_i + \dots + b_{n+1}x_i^n))^2.$$

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

$$\mathbf{a}_i^T = (1, x_i, \dots, x_i^n), \quad i = 1, \dots, m,$$

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

Because there are a number of different ways of expressing the polynomial model

defined above, for example, in terms of monomials (powers of x) or Chebyshev polynomials [19], the residual deviations

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

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

The polynomial regression problem is important to metrology because of its application in the modelling of measurement data in situations where the relationship between stimulus and response variables may be represented by a polynomial model, and the measured values of the stimulus variable have associated uncertainties that are negligible compared to the uncertainties associated with the measured values of the response variable. 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 regression computation is considered:

- NAG [10].

Function **E02ADF** computes the weighted least-squares polynomial approximations of all orders up to a specified maximum for data (x_i, y_i) , $i = 1, \dots, m$, 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 results are calculated from the values of the least-squares polynomial approximation of specified order at the data x_i -values.

- IMSL [11].

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 results are calculated from the coefficient values.⁹

- 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 variable x but raised to different powers.¹⁰ The residual deviations defining the results are calculated from the values of the fitted curve.

There are two ways of accessing functions for polynomial regression:¹¹

- As calls in Visual Basic code – computations undertaken in this way are referred to as *VB/Excel*.
- As worksheet functions – computations undertaken in this way are

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

¹⁰ 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.

¹¹ The previous report considered only VB/Excel computations.

referred to as *Excel*.

- Origin [13].

The **Polynomial Fit tool** returns the values of a least-squares polynomial fit to data at the points x_i , $i = 1, \dots, m$. The residual deviations defining the results are also returned.¹²

- Matlab [15].

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 results 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 variable x in order to improve the numerical properties of the polynomial 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 $\mathbf{x}$ is an $m \times 1$ vector containing the x_i -values, $\mathbf{y}$ an $m \times 1$ vector containing the y_i -values, and $\mathbf{n}$ is the degree of the polynomial. The first statement returns the coefficients $\mathbf{p}$ defining the least-squares polynomial approximation to the data. The second statement evaluates the residual deviations defining the results. 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 as a power series in the normalised variable $\tilde{x}$, where

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

and 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*.

- Mathcad [16].

Function **regress** fits a polynomial curve to data using the method of least-squares, returning the coefficients in the power-series representation of the polynomial. Function **interp** returns the value of a polynomial from its power-series representation. The residual deviations defining the results are calculated from the

¹² There appears to be some confusion in the Origin documentation for this function. The value of “polynomial order” requested by the **Polynomial Fit tool** is in fact the polynomial degree.

values of the least-squares polynomial approximation at the data x_i -values.¹³

In cases where the polynomial curve-fitting problem is detected as being poorly conditioned, the user is recommended to use the function **regress** in a way that includes centring and scaling of the independent variable x 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 (Mathcad) statements

```
p := regress(x,y,n)
j := 0..m – 1
fitj := interp(p,x,y,xj)
d := y – fit
```

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 remaining statements evaluate the residual deviations defining the results. The computation undertaken in this way is referred to as *Mathcad A*.

Secondly, the (Mathcad) statements

```
xt := 
$$\frac{(\mathbf{x} - \mathbf{max}(\mathbf{x})) - (\mathbf{min}(\mathbf{x}) - \mathbf{x})}{\mathbf{max}(\mathbf{x}) - \mathbf{min}(\mathbf{x})}$$

p := regress(xt,y,n)
j := 0..m – 1
fitj := interp(p,x,y,xtj)
d := y – fit
```

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 *Mathcad B*.

- LabVIEW [17].

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_i -values.¹⁴ The residual deviations defining the results are calculated from the values of the best-fit curve.

4 Interfacing to the Test Software

For each function tested, a “program” (or equivalent) is implemented in the language or package¹⁵ of which the function is a part to undertake the following generic operations:

¹³ There appears to be some confusion in the Mathcad documentation for this function. The value of “polynomial order” requested by the **regress** function is in fact the polynomial degree.

¹⁴ There appears to be some confusion in the LabVIEW documentation for this function. The value of “polynomial order” requested by the **General Polynomial Fit** function is in fact the polynomial degree.

¹⁵ For example, for the case of Microsoft Excel, the “program” may be implemented using Visual Basic for Applications (VBA).

1. load or import a reference data set,
2. load or import 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, and
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*¹⁶ of the (floating-point) arithmetic used to calculate the test and reference results. Some of the packages provide intrinsic functions for calculating the computational precision. For these packages (NAG and Matlab) the values for the computational precision returned by the intrinsic functions (**X02AJF** and **EPS**, respectively) are used. For all other packages, an approximate value of 2×10^{-16} is used.

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

1. copy the values of the performance measures onto worksheets of the Microsoft Excel spreadsheet package,
2. display the values using the “chart” function of Microsoft Excel, and
3. 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 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

¹⁶ For the commonly used floating-point arithmetic, the computational precision 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-decimal digit working.

regression and polynomial 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 [20].

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 [5]:

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

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 of the (floating-point) arithmetic used to calculate the test and reference results. 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 $\mathbf{x}$, 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 $\mathbf{x}$, the test software returns a test result with eight fewer figures of accuracy than that returned by an optimally stable algorithm.

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

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

and

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

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

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

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

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

<i>Performance parameter</i>	<i>Values defining sequences of reference data sets</i>							
Sample 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 mean and standard deviation. Default values for the performance parameters are shown in bold.

<i>Performance parameter</i>	<i>Values defining sequences of reference data sets</i>							
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 angle $\alpha = \lambda\pi$ that the line makes with the x -axis	-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 regression. Default values for the performance parameters are shown in bold.

<i>Performance parameter</i>	<i>Values defining sequences of reference data sets</i>							
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 regression. Default values for the performance parameters are shown in bold.

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. 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/ssfm3/theme3/numerical_software_testing/project3_2/milestone3/

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, and
- maximum inscribed circle.

For each computation data generators are provided in the form of:

- Java Applets that run on the user's machine, and
- Java Servlets that run on the host's machine (at NPL).

To help tailor the data sets to particular requirements, the user supplies values for a number of input parameters to the Applet or Servlet that define the reference results and other properties of the reference data set, for example, 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 results 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. The Servlet generates output as a HTML page, which can be viewed using a web browser and saved to the user's machine.

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 using the Java Applets, with the value of the random number seed S set equal to zero.

For linear regression, the value of the coordinate defining the centroid y_c of the y -data is also set to zero.

7.1 Sample mean and standard deviation

7.1.1 Inputs

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

- sample mean $\bar{x}$ satisfying $|\bar{x}| \geq 1$,
- sample standard deviation s satisfying $s \geq 1$,
- sample size m satisfying $m \geq 2$, and
- random number seed S .

7.1.2 Outputs

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

- reference data x_i , $i = 1, \dots, m$, and
- 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 mean and sample standard deviation for the reference data x_i , $i = 1, \dots, m$.

7.1.3 Procedure

The procedure implemented by the data generator for the computation of sample mean and standard deviation is as follows.

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

$$\mathbf{a}_i^T = 1, \quad i = 1, \dots, m.$$

2. Calculate residual deviations d_i , $i = 1, \dots, m$, with mean zero (Section 7.4), setting $\sigma = s$.
3. Calculate the sample standard deviation of the residual deviations d_i , $i = 1, \dots, m$:

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

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

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

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

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

6. Calculate “degrees of difficulty”:

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

7.2 Straight-line 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 satisfying¹⁷ $-4\pi/5 \leq \alpha \leq 4\pi/5$,
- number of points m satisfying $m > 2$,
- length L of interval of x containing the x -values,
- magnitude of measurement standard deviation σ satisfying $\sigma \geq 1$, and
- 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) , $i = 1, \dots, m$, and
- reference constants $x_c, y_c, \alpha, m, L, \sigma, S, \kappa(\mathbf{d}), d_i$, $i = 1, \dots, m$,

where d_i , $i = 1, \dots, m$, 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) , $i = 1, \dots, m$.

7.2.3 Procedure

The procedure implemented by the data generator for the computation of straight-line regression is as follows.

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

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

- b. For each $i = 1, \dots, m$:

$$A_{i,1} = 1,$$

$$A_{i,2} = X_i.$$

¹⁷ Chosen to ensure the line is not too steep, and so it is reasonable to regard measurement deviations as associated with measured values for the response variable only. For the testing of software for straight-line regression described in this report, reference data sets defined by angles α between $-\pi/3$ and $\pi/3$ were used (Table 2).

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

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

5. Calculate “degree of difficulty”:

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

7.3 Polynomial regression

7.3.1 Inputs

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

- polynomial degree $n \geq 0$,
- number of points m satisfying $m > n + 1$,
- endpoints of interval $[x_{\min}, x_{\max}]$ containing the x -values,¹⁸
- magnitude of measurement standard deviation σ satisfying $\sigma \geq 1$, and
- 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) , $i = 1, \dots, m$, and
- reference constants $n, m, x_{\min}, x_{\max}, \sigma, S, \kappa(\mathbf{d}), d_i$, $i = 1, \dots, m$,

where d_i , $i = 1, \dots, m$, 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) , $i = 1, \dots, m$.

7.3.3 Procedure

The procedure implemented by the data generator for the computation of polynomial regression is as follows.

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

- a. Normalise the x -coordinates:

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

- b. For each $i = 1, \dots, m$:¹⁹

¹⁸ For the testing of software for polynomial regression described in this report, the lower endpoint $x_{\min}$ was used as a performance parameter in the specification of reference data sets, with the length $x_{\max} - x_{\min}$ of the interval $[x_{\min}, x_{\max}]$ containing the x -values constant (Section 5 and Table 3).

$$\begin{aligned}
A_{i,1} &= 1, \\
A_{i,2} &= X_i, \\
A_{i,j} &= 2X_i A_{i,j-1} - A_{i,j-2}, \quad j = 3, \dots, n+1.
\end{aligned}$$

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

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

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 consist of values for the following parameters:

- design matrix $\mathbf{A}$ of dimension $m \times n$ satisfying $m > n$, and
- magnitude of measurement standard deviation σ satisfying $\sigma \geq 0$.

7.4.2 Outputs

The outputs consist of the following information:

- vector $\mathbf{d}$ 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 $\mathbf{N}$ of $\mathbf{A}^T$ where $\mathbf{N}$ has dimension $m \times (m - n)$ [21]:
 - a. Form the singular value decomposition (SVD) of $\mathbf{A}^T$:

$$\mathbf{A}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T.$$

- b. Form $\mathbf{N}$ from the columns of $\mathbf{V}$:

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

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

¹⁹ This is the recurrence relation for evaluating Chebyshev polynomials [19].

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

The key step in the above procedure is to form the singular value decomposition of $\mathbf{A}^T$ (step 1.1). This is achieved using routine **DGESVD** provided as part of the Java library J LAPack. The library [18] provides the LAPACK [22] 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 mean and standard deviation

Figures 1–3 in Appendix A²⁰ show values of the performance measure²¹ for test software for the calculation of sample mean for, respectively, the performance parameters reference sample mean, reference sample standard deviation, and sample size (see Table 1).²² For all reference data sets and all test software *excepting* Microsoft Excel (for which the test software is accessed as worksheet functions), the values of the performance measure are between zero and one. The results indicate that the numerical accuracy of the test software is comparable to that expected from optimally stable (reference) software for this calculation and these data sets.

For Excel, the values of the performance measure are slightly greater than one (and similar results are obtained for the calculation of standard deviation and for straight-line and polynomial regression). Comparison of the results for Excel and VB/Excel suggests that the “interactive” method of inputting data from reference data files to worksheets may be responsible for the loss of approximately one digit of accuracy, and the likelihood that the worksheet function in Excel implements an optimally stable algorithm.

Figures 4–6 in Appendix B 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* Excel and VB/Excel, the values of the performance measure are between zero and one. For Excel, the values of the performance measure are typically slightly greater than one (see above). For VB/Excel, the performance measure for a single reference data set has the value two while for all other reference data sets the measure takes values between zero and one.

In previous testing of Microsoft Excel 2000 [4], the values of the performance measure

²⁰ In the figures in the appendices, 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).

²¹ In the figures in the appendices, the values for the performance measures are rounded to one decimal place and plotted. Further rounding the values to the nearest integer gives an indication of the number of decimal digits of accuracy lost by the test software over and above the number expected to be lost by an optimally stable algorithm. Values of the performance measure greater than 16 indicate that the test software loses all digits of accuracy.

²² 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.

showed a tendency to increase with the performance parameter reference sample mean. This behaviour suggested the likelihood that Microsoft Excel 2000 implemented the formula

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

for the sample standard deviation, which is known to be numerically unstable. The results presented in this report (Figure 4) indicate that Microsoft Excel 2003 implements a numerically stable formula for the calculation of sample standard deviation [23].

For the Matlab and LabVIEW packages, the values of the performance measures shown in the figures are very similar to those reported in the previous edition of this report [4] for older releases of these packages.

8.2 Straight-line regression

Figures 7–11 in Appendix C²³ show values for the performance measure for test software for straight-line regression for the performance parameters listed in Table 2. For Matlab, LabVIEW and VB/Excel, the values of the performance measure are very similar to those reported in the previous edition of this report [4] for older versions of these packages.

Generally (Figures 8–11), the values of the performance measure are between zero and one, *excepting* for Excel for which the values are typically between one and two (see Section 8.1), and Java (JNL) for which there are values between one and two for a small number of reference data sets (Figures 8 and 9). The results indicate 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_i -values). Furthermore, for the final reference data set in this sequence (for which the coordinate of the centroid of the data x_i -values is 10^7), Matlab A provides a warning to the user about the reliability of the test result.

There are two issues for test software for straight-line 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 [24, 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 NAG B and Matlab B show that the application of the NAG and Matlab functions to a problem for which the straight-

²³ For ease of visualisation each figure in this appendix consists of two plots. The first plot displays values for the performance measure for the Fortran and Java libraries (unchanged from the previous edition of this report). The second plot displays values for the performance measure for the remaining packages.

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.²⁴ The results suggest that it is likely that IMSL, Java (JLAPACK), Excel, VB/Excel and LabVIEW test software will behave in a similar manner. The results for Mathcad A and Mathcad B are very similar to each other, and suggest that normalisation of the independent variable is undertaken as part of the Mathcad calculation. The results for Java (JNL), however, are (probably) indicative of the numerical algorithm used to solve the least-squares problem.²⁵ For Java (JNL) 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 Java (JNL) function implements an unstable algorithm for straight-line regression.

8.3 Polynomial regression

Figures 12–15 in Appendix D show values for the performance measure for test software for polynomial regression for the performance parameters listed in Table 3. From a comparison of the values of the performance measure presented here with those obtained for older releases of Matlab, LabVIEW and VB/Excel reported in the previous edition of this report [4], the following observations can be made:

- For Matlab, the values of the performance measure are very similar for all performance parameters.
- For LabVIEW, the values of the performance measure are very similar for all performance parameters *excepting* the lower endpoint defining the interval containing the data x_i -values. For this performance parameter, the values of the performance measure are appreciably *higher* for the more recent release.
- For VB/Excel, the values of the performance measure are very similar for all performance parameters *excepting* the polynomial order and the lower endpoint defining the interval containing the data x_i -values. For these performance parameters, the values of the performance measure are appreciably *lower* for the more recent release.

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

As for the straight-line regression problem (a particular example of polynomial regression), the issues of model parametrisation and the numerical algorithm used will influence the accuracy of the test results. The NAG function uses a Chebyshev-series representation of polynomials [21], Matlab B and Mathcad B a power-series representation in terms of a normalised (shifted and scaled) variable (Section 3.3), and Matlab A and Mathcad A a power-series representation in terms of the (natural) unnormalised variable [24, 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 LabVIEW function are further affected by the

²⁴ 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).

²⁵ To be certain would require detailed information concerning the algorithm implemented.

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).

For polynomial regression, LabVIEW offers a number of options for the numerical algorithm used to solve the least-squares problem (although little advice is provided regarding when to use which algorithm). These options are SVD (the default option, and used to obtain the results in Figures 12–15), Givens, Givens2, Householder, LU factorisation and Cholesky factorisation. Figures 16–20 in Appendix E show values for the performance measure for these algorithms for the performance measures listed in Table 3.

The results indicate a clear grouping of the algorithms. Givens, Givens2 and Householder return identical results and are the best algorithms (returning the lowest values for the performance measure) for the polynomial regression problem. LU factorisation and Cholesky factorisation return similar results while SVD returns the poorest results (highest values for the performance measure). For polynomial fitting where the polynomial order is high, or where the data x_i -values are particularly large, the design matrix $\mathbf{A}$ (see Section 3.3) may be badly conditioned (i.e., have a high condition number). In such cases the choice of algorithm used to solve the polynomial regression problem is of increased importance.

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 mean and standard deviation, straight-line regression and polynomial regression. The functions tested are taken from a number of proprietary software packages and libraries, including the NAG and IMSL (FORTRAN) libraries, Microsoft Excel (accessed as worksheet functions or using Visual Basic for Applications), Origin, LabVIEW, Matlab, Mathcad 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 relate to 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 digits 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 [24, 25].

Some particular conclusions are as follows:

- The software tested for the calculation of sample mean and sample standard deviation return test results for the reference data sets considered to have a numerical accuracy that is *comparable* to that expected from reference software.
- For the calculation of straight-line regression, a *critical* parameter is the coordinate of the centroid of the data x_i -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 regression. The results of the testing indicate that the functions provided by the NAG and IMSL libraries, Microsoft Excel, Origin, Matlab, Mathcad and Java (JLAPACK), 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 regression, *critical* performance parameters are the order (degree + 1) of the polynomial and the lower endpoint defining the interval containing the data x_i -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 regression. The results of the testing indicate that the functions provided by the NAG library returns test results for the reference data sets considered having a numerical accuracy that is *comparable* to that expected from reference software.

Acknowledgements

This report constitutes one of the deliverables of Project 3.2 of the 2004–2007 NMS *Software Support for Metrology* Programme, and has been funded by the National Measurement System Directorate of the UK Department of Trade and Industry.

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] M. G. Cox, P. M. Harris, E. G. Johnson, P. D. Kenward and G. I. Parkin. Testing the numerical correctness of software. *NPL Report CMSC 34/04*, 2004.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] BIPM, IEC, IFCC, ISO, IUPAC, IUPAP and OIML. Guide to the Expression of Uncertainty in Measurement. Geneva, Switzerland, 1995. ISBN 92-67-10188-9, second edition.
- [10] The NAG Fortran Library (Mark 20). Copyright © 2001, Numerical Algorithms Group Ltd.
- [11] IMSL Fortran 90 MP Library (Version 4.01). Copyright © 1999, Visual Numerics, Inc.
- [12] Microsoft ® Office Excel 2003 (11.5612.5606). Copyright © 1985–2003, Microsoft Corporation.
- [13] Origin (OriginPro 7.5 SR1 v7.5776 B776). Copyright © 1991–2003, OriginLab Corporation.
- [14] JNL, a Numerical Library for Java. Copyright © 1997, Visual Numerics, Inc.
- [15] Matlab (Version 7.1.0.246 (R14) Service Pack 3). Copyright © 1984–2005, The MathWorks, Inc.
- [16] Mathcad (Version 13.0 (509121419)). Copyright © 1986-2005, Mathsoft Engineering & Education, Inc.
- [17] LabVIEW (Version 8.0). Copyright © 2005, National Instruments.
- [18] D.M. Doolin, J. Dongarra and K. Seymour. JLAPACK – Compiling LAPACK FORTRAN to Java. *Scientific Programming* **7** (2), pp 111 – 138, 1999.
- [19] L. Fox and I.B. Parker. *Chebyshev Polynomials in Numerical Analysis*. Oxford University Press, 1968.
- [20] 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.
- [21] G.H. Golub and C.F. Van Loan. *Matrix Computations (third edition)*. John Hopkins University Press, 1996.
- [22] 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.

- [23] B.D. McCullough and B. Wilson. On the accuracy of statistical procedures in Microsoft Excel 2003. *Computational Statistics and Data Analysis* **49** (4), 1244-1252, 2005.
- [24] 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.
- [25] 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: Results for Sample Mean

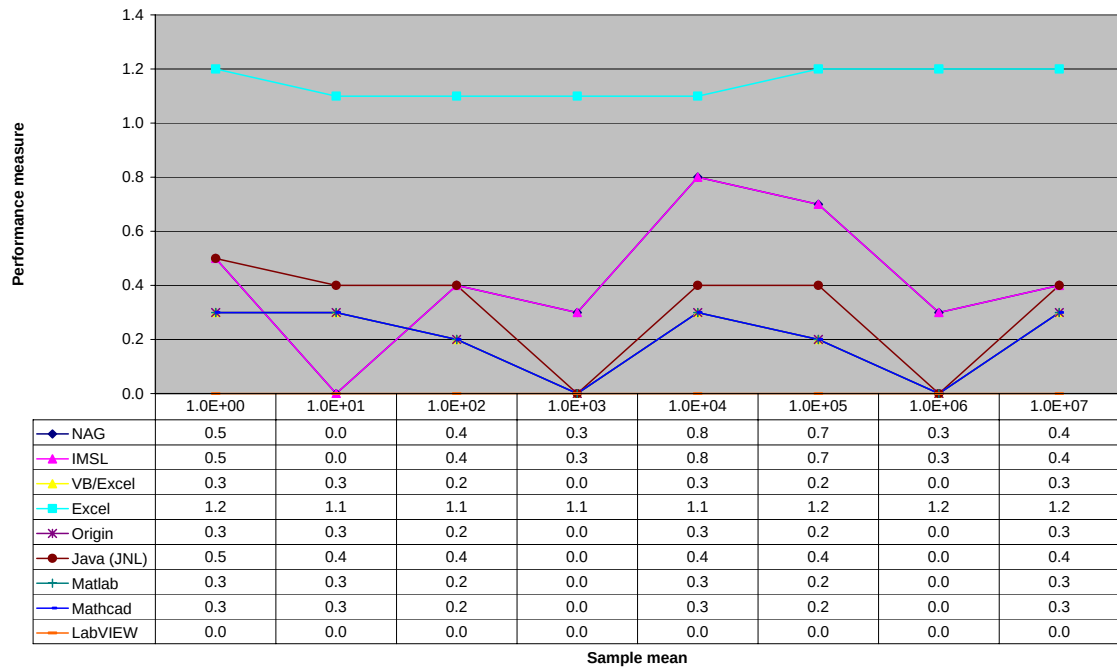


Figure 1: Values of the performance measure for test software for the computation of the sample mean. The performance parameter is the reference value for the sample mean.

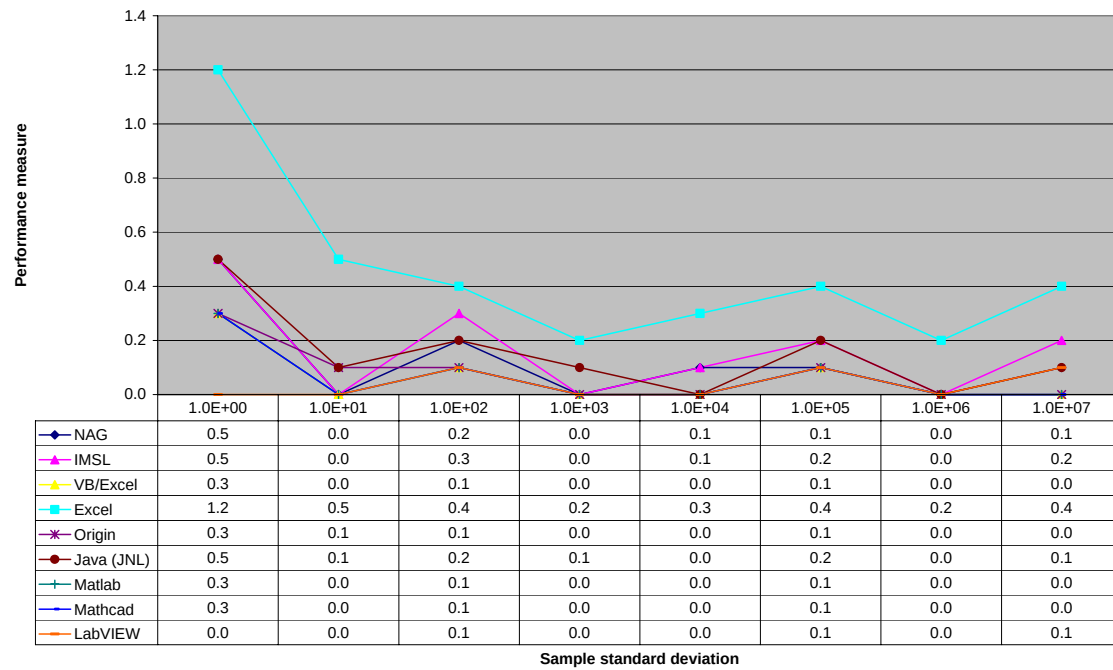


Figure 2: As Figure 1 except that the performance parameter is the reference value for the sample standard deviation.

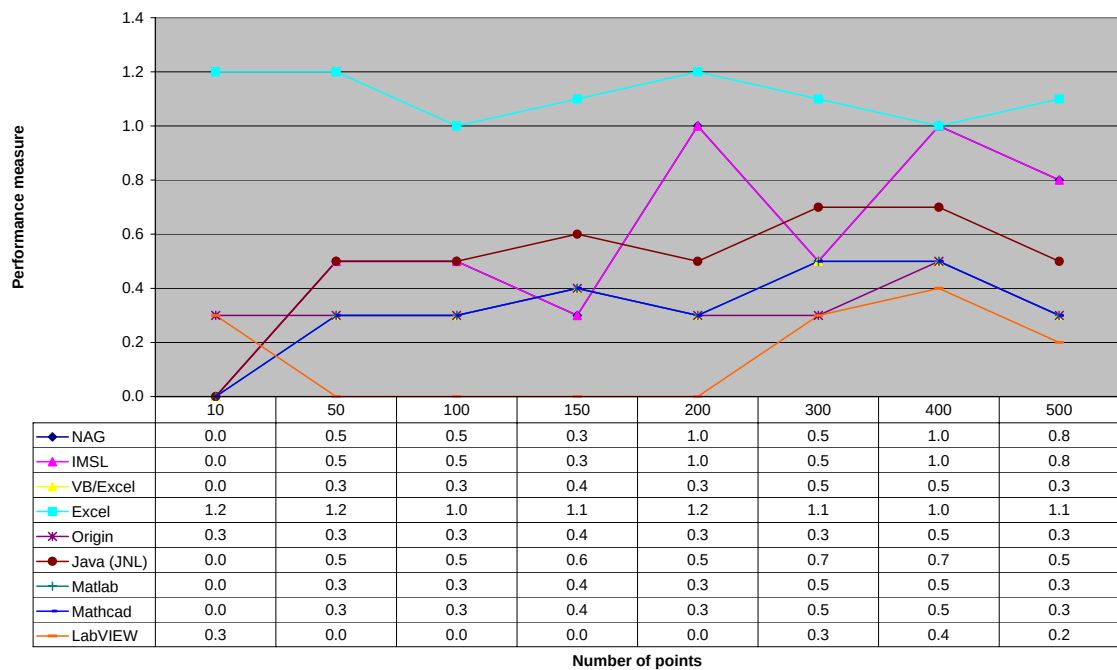


Figure 3: As Figure 1 except that the performance parameter is the sample size.

Appendix B: Results for sample standard deviation

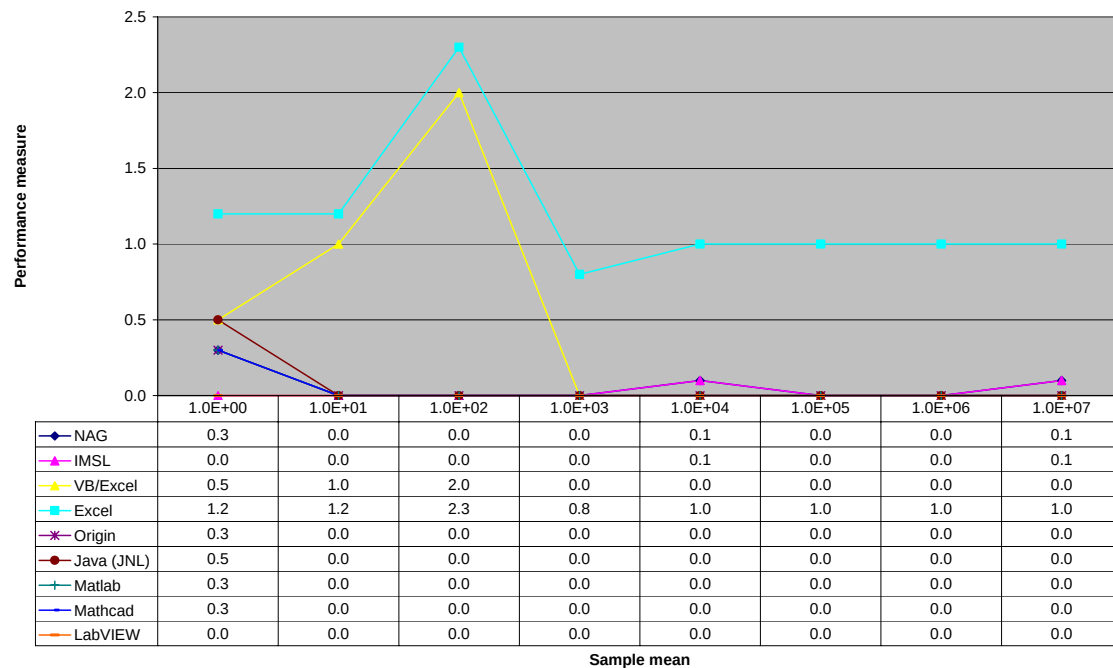


Figure 4: Values of the performance measure for test software for the computation of the sample standard deviation. The performance parameter is the reference value for the sample mean.

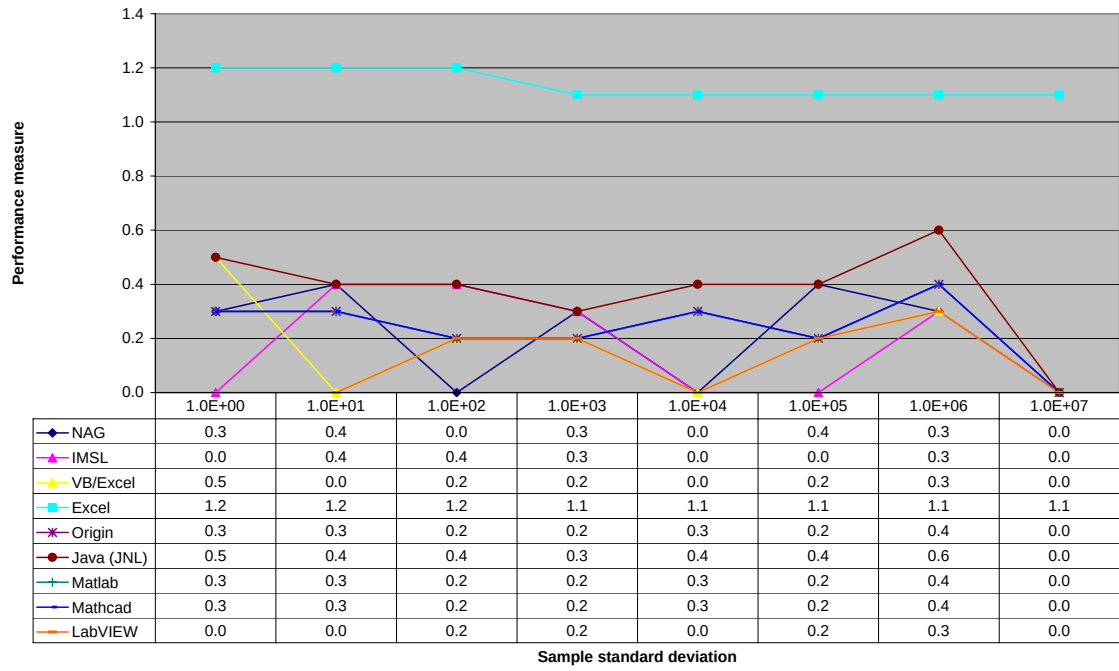


Figure 5: As Figure 4 except that the performance parameter is the reference value for the sample standard deviation.

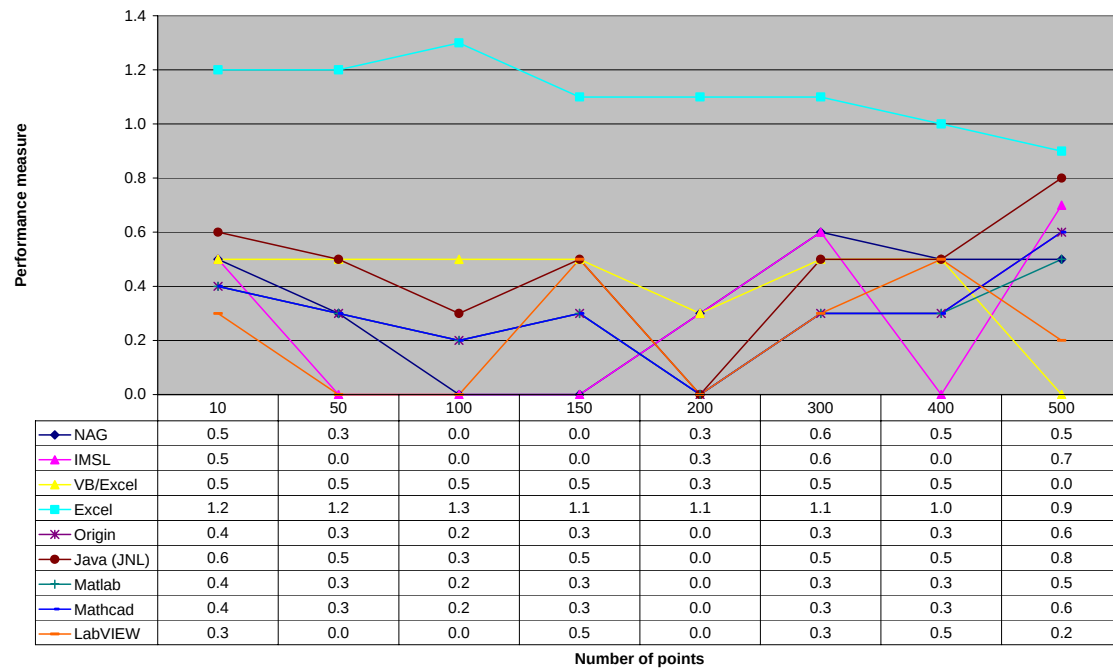


Figure 6: As Figure 4 except that the performance parameter is the sample size.

Appendix C: Results for Straight-Line Regression

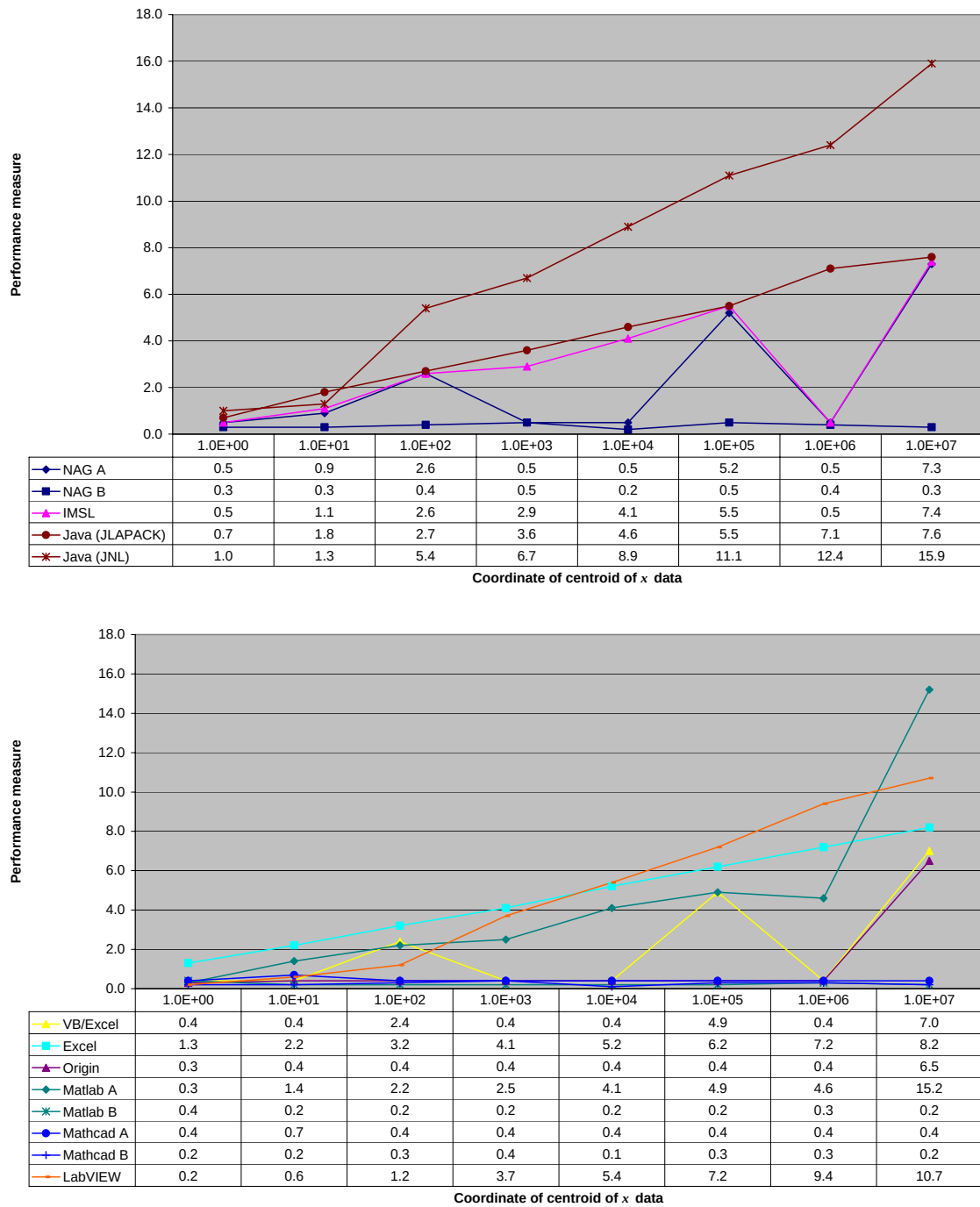


Figure 7: Values of the performance measure for test software for the computation of straight-line regression. The performance parameter is the coordinate of the centroid of the data x_i -values.²⁶

²⁶ For the particular reference data set for which the coordinate of the centroid of the data x -values is 10^7 , Matlab A provides a warning to the user about the reliability of the test result.

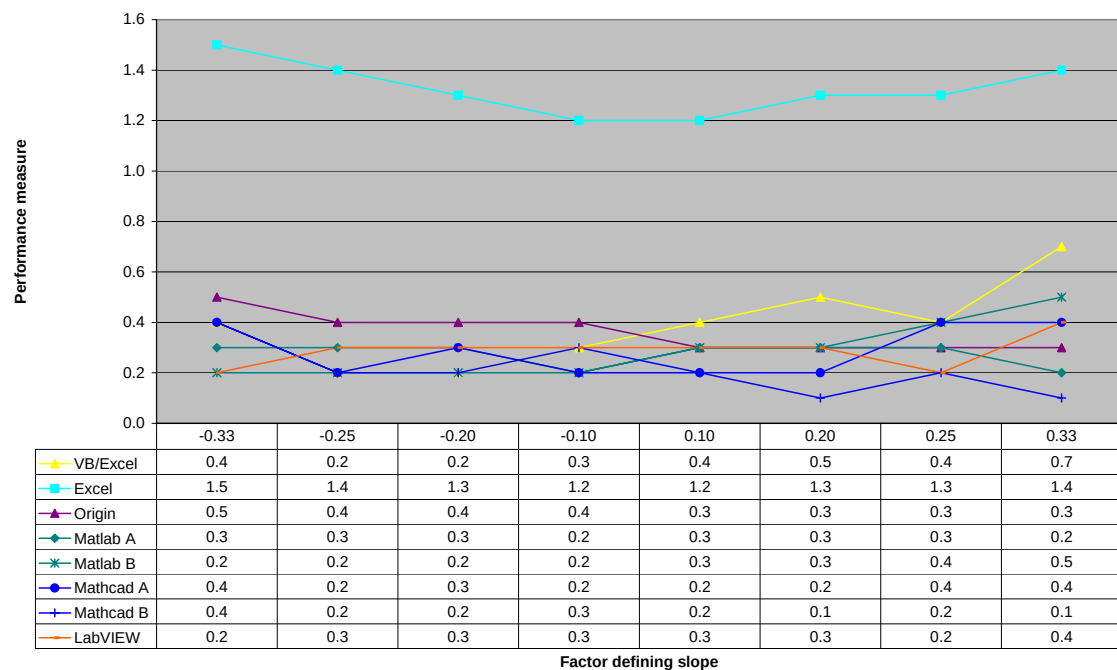
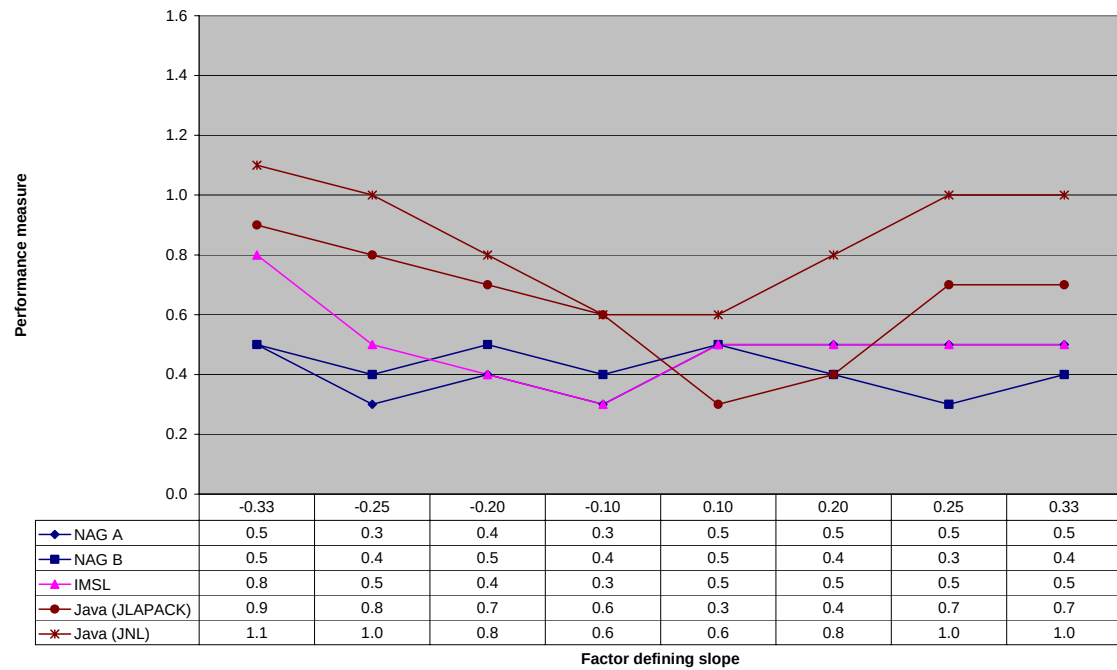


Figure 8: As Figure 7 except that the performance parameter is the factor defining the slope of the solution line.

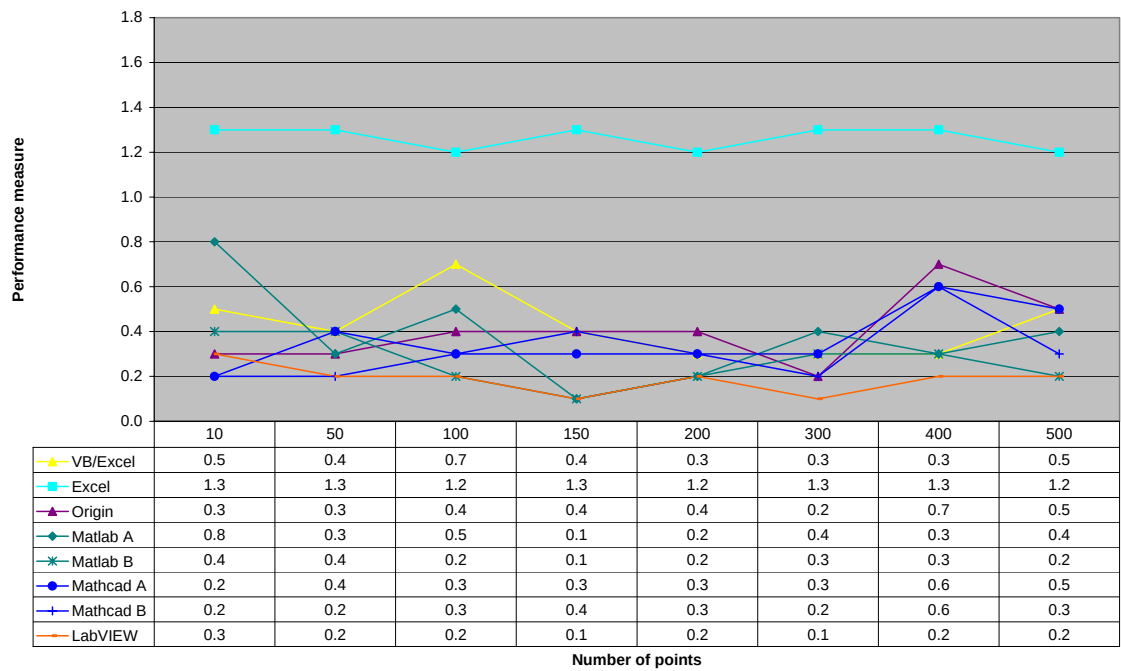
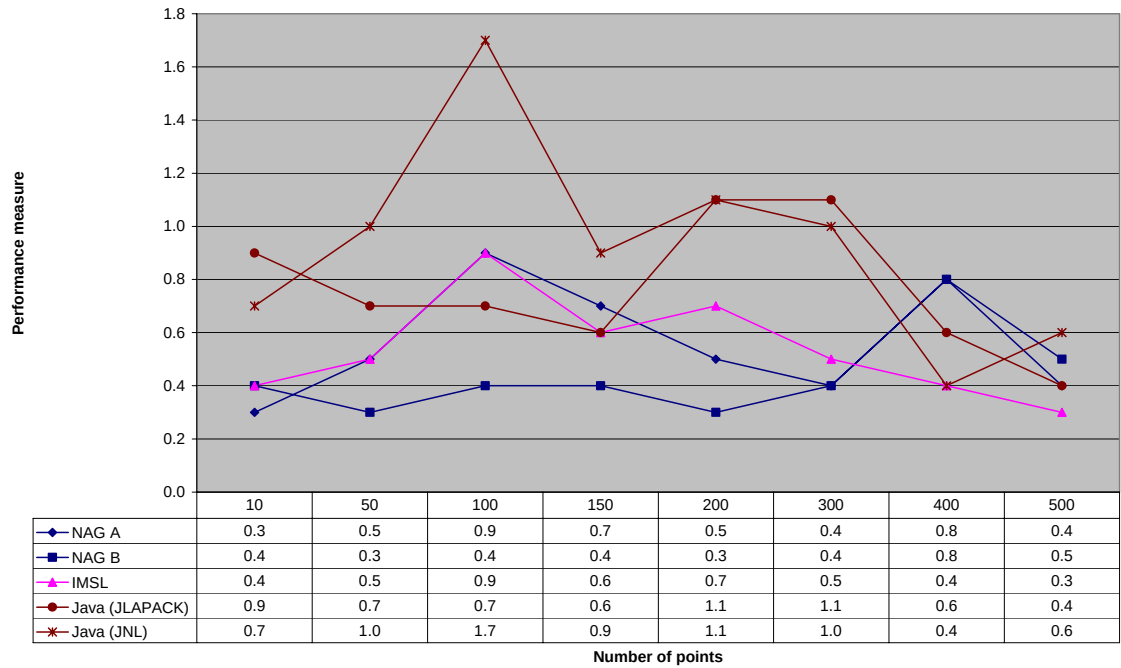


Figure 9: As Figure 7 except that the performance parameter is the number of data points.

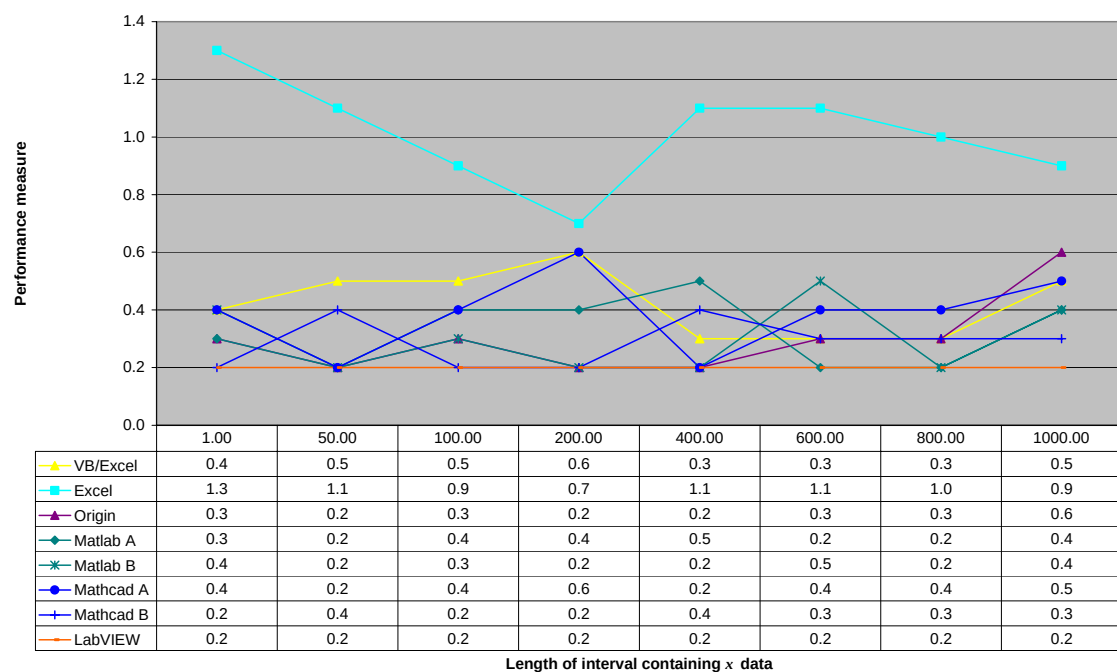
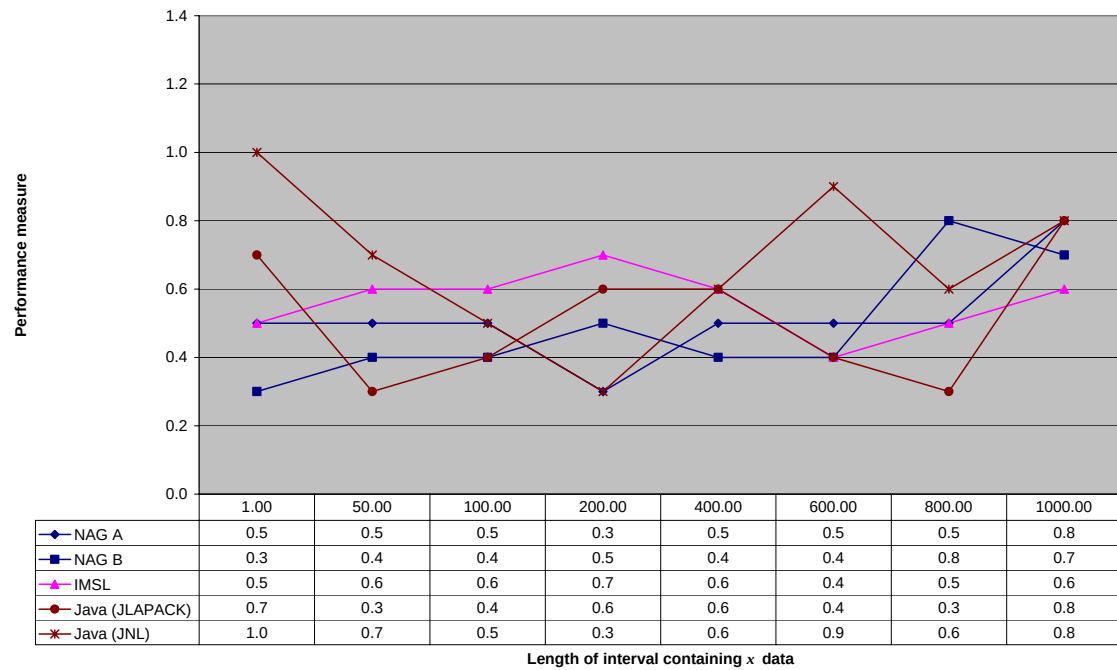


Figure 10: As Figure 7 except that the performance parameter is the length of the interval containing the data x_i -values.

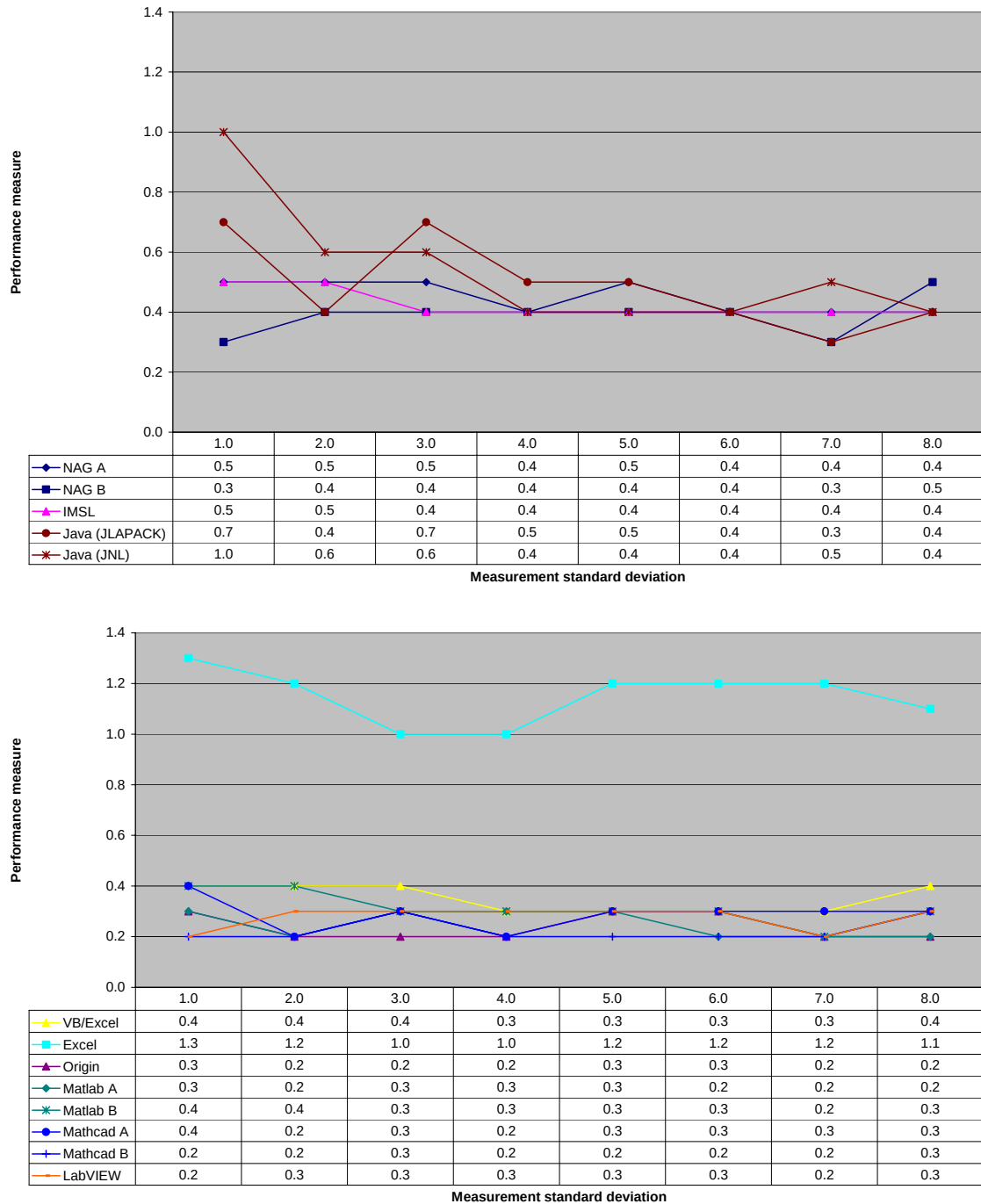


Figure 11: As Figure 7 except that the performance parameter is the measurement standard deviation.

Appendix D: Results for Polynomial Regression

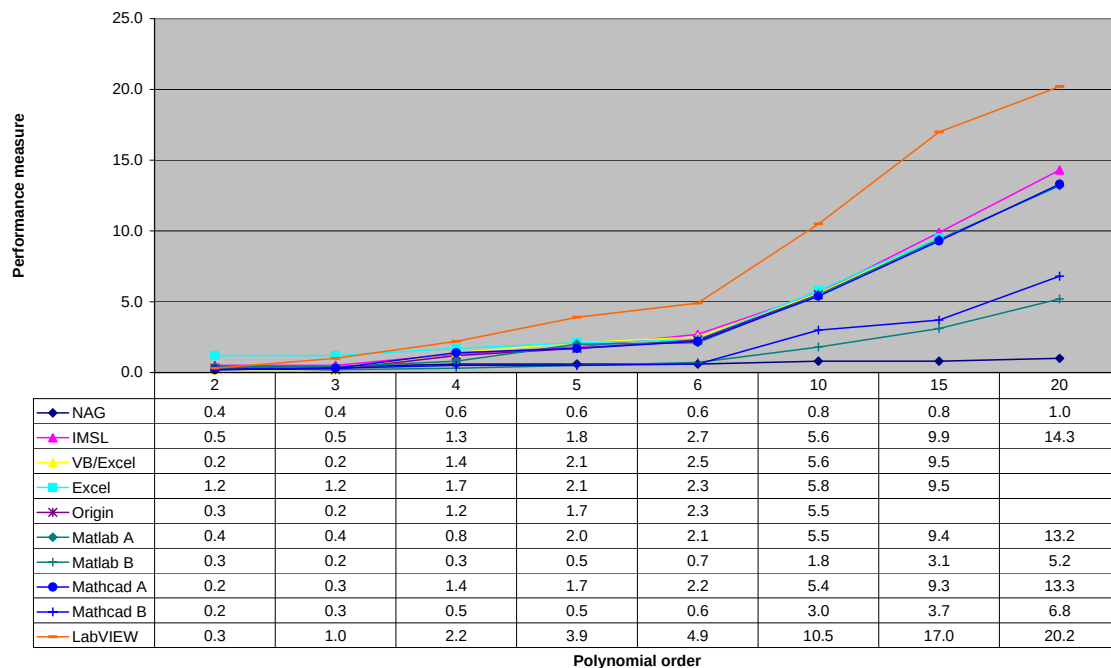


Figure 12: Values of the performance measure for test software for the computation of polynomial regression. The performance parameter is the order (degree + 1) of the polynomial.²⁷

²⁷ 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. The maximum order permitted by Excel is 17. The maximum order permitted by Origin is 10.

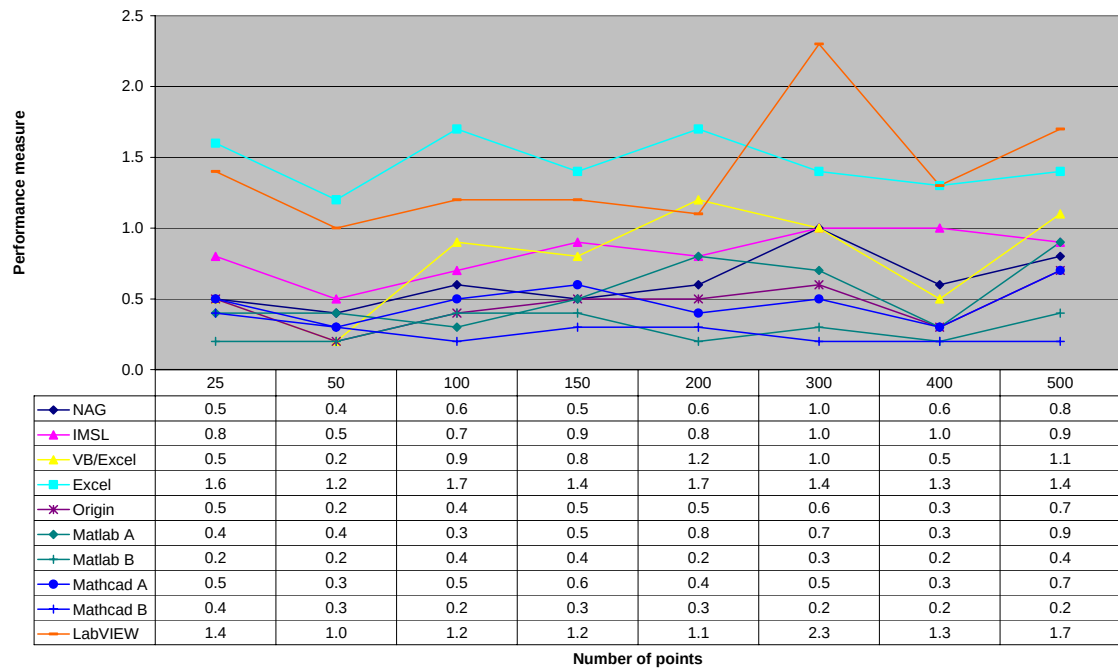


Figure 13: As Figure 12 except that the performance parameter is the number of data points.

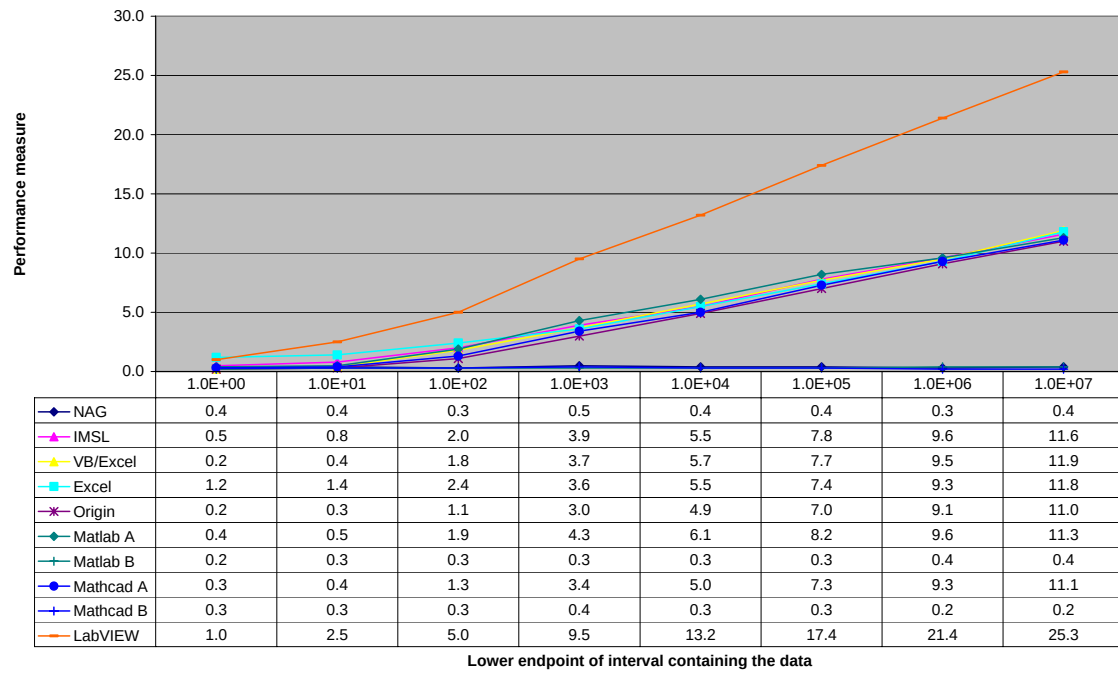


Figure 14: As Figure 12 except that the performance parameter is the lower endpoint defining the interval containing the data x_i -values.²⁸

²⁸ 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.

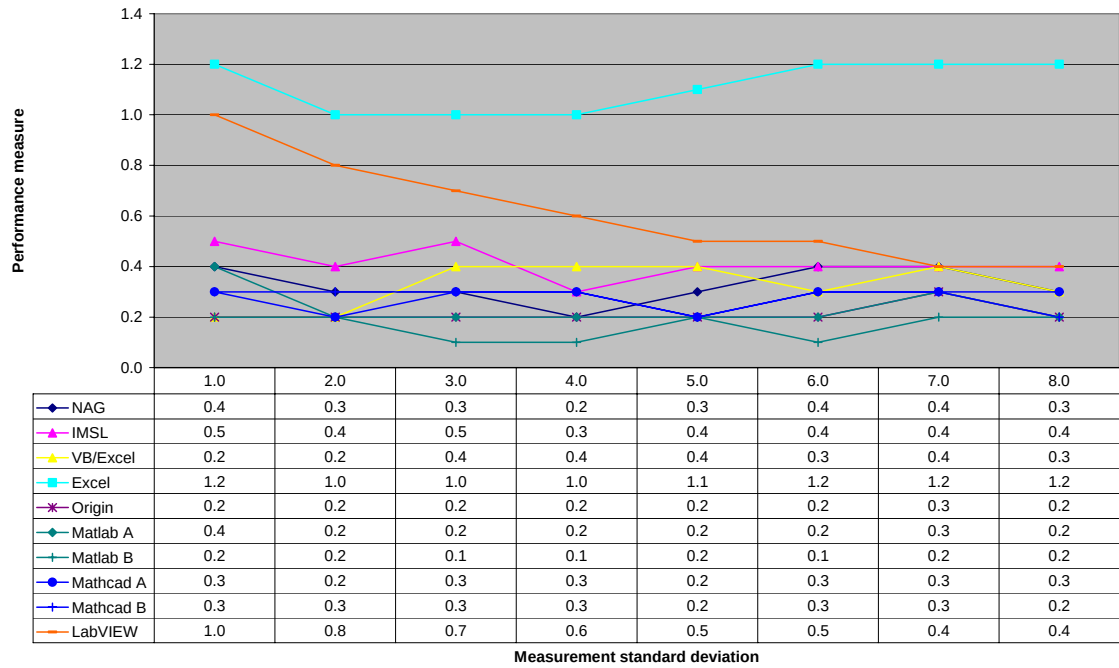


Figure 15: As Figure 12 except that the performance parameter is the measurement standard deviation.

Appendix E: Results for LabVIEW Polynomial Regression

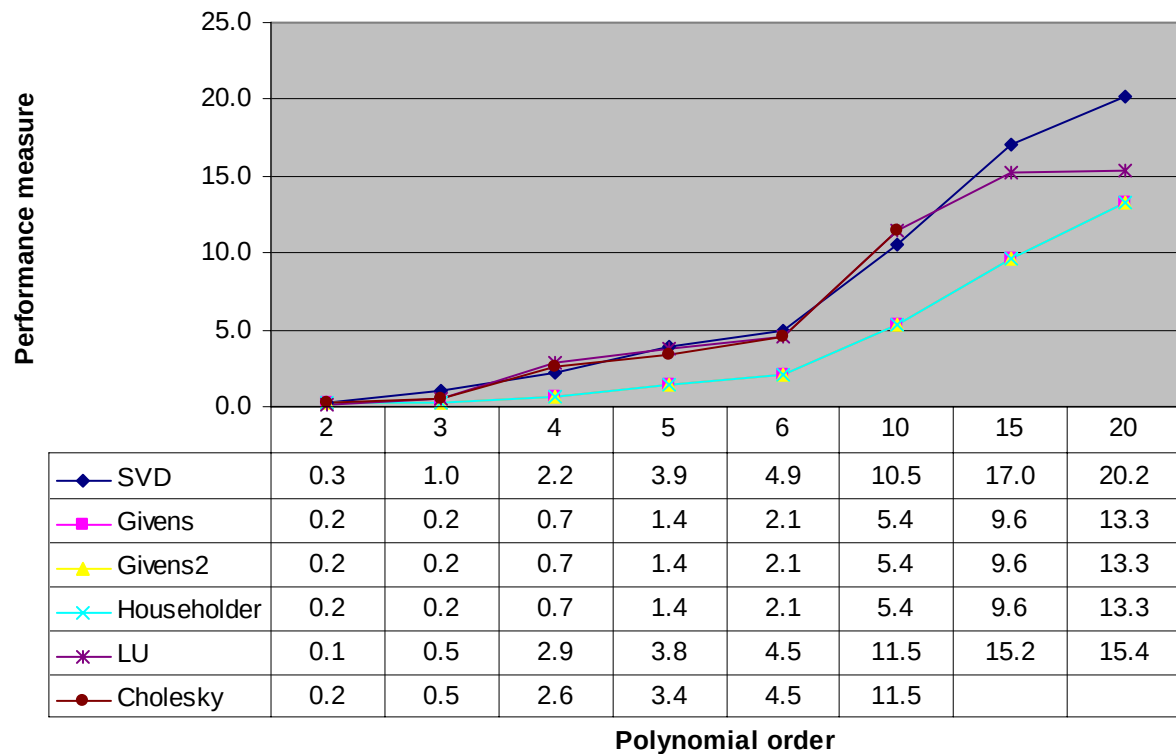


Figure 16: Values of the performance measure for test software for the computation of polynomial regression. The performance parameter is the order (degree + 1) of the polynomial.²⁹

²⁹ For the last two reference data sets in this sequence (corresponding to orders 15 and 20), an error is returned when using the Cholesky factorisation.

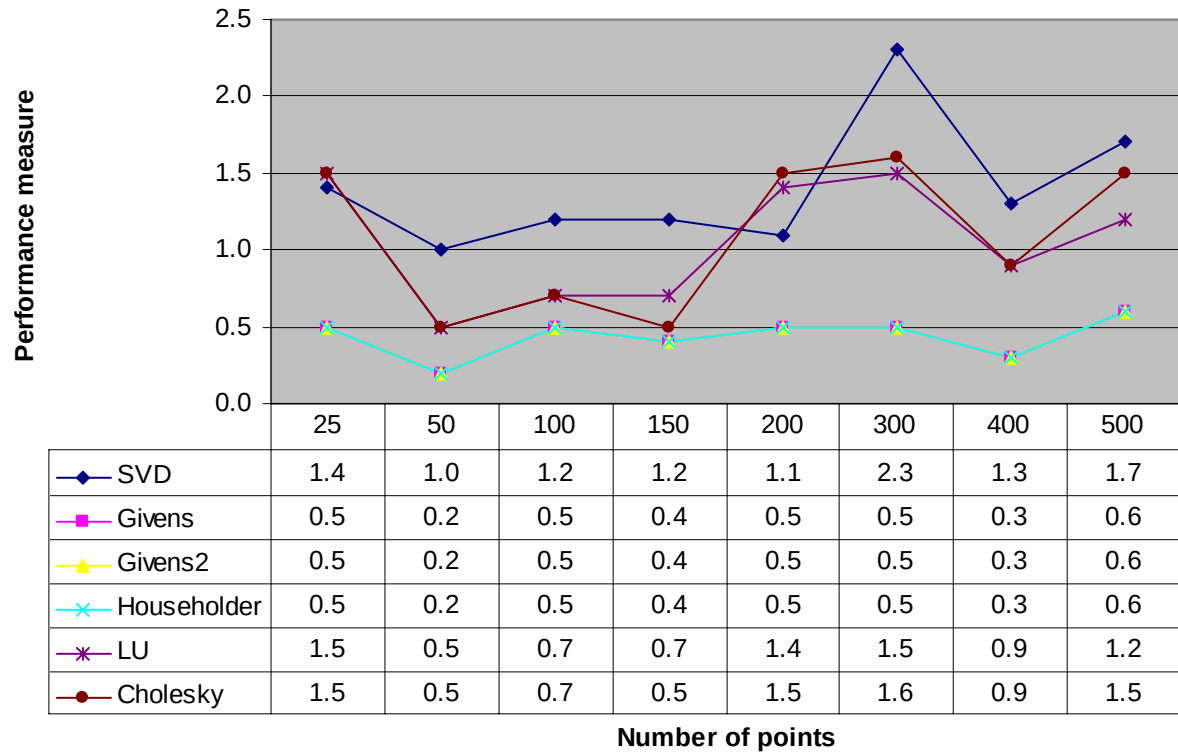


Figure 17: As Figure 16 except that the performance parameter is the number of data points.

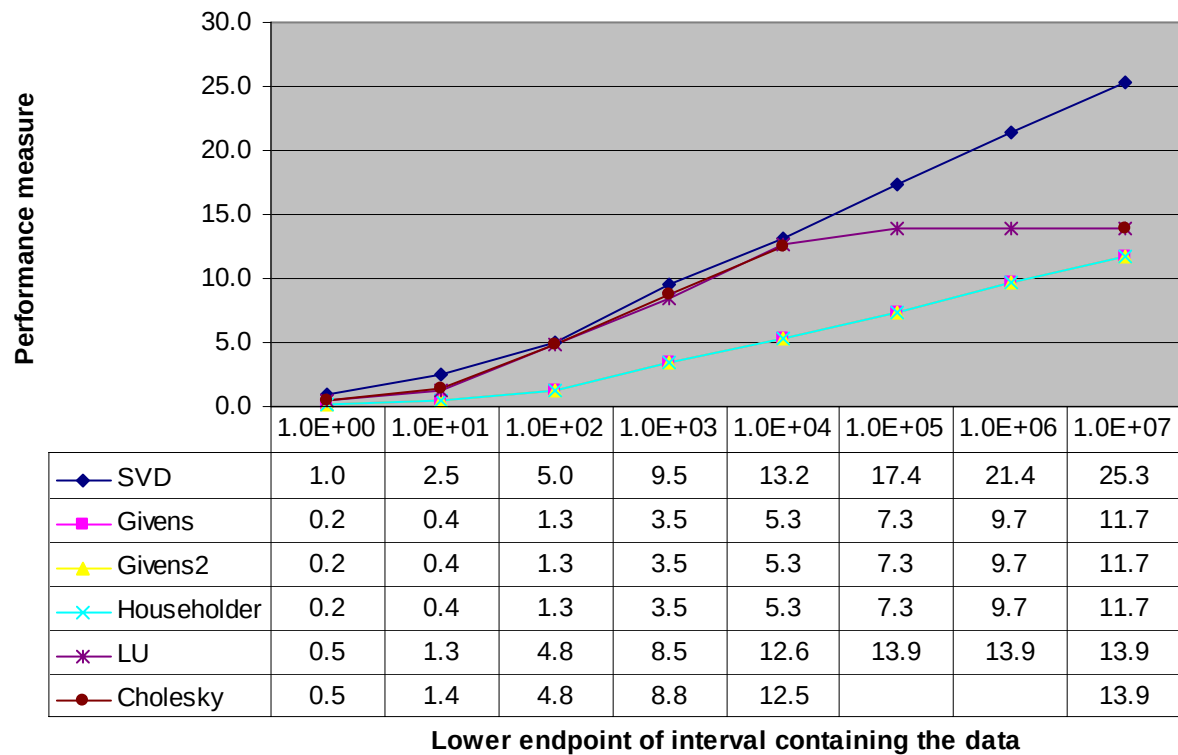


Figure 18: As Figure 16 except that the performance parameter is the lower endpoint defining the interval containing the data x_i -values.³⁰

³⁰ For the reference data sets in this sequence corresponding to lower endpoints 10^5 and 10^6 , an error is returned when using the Cholesky factorisation.

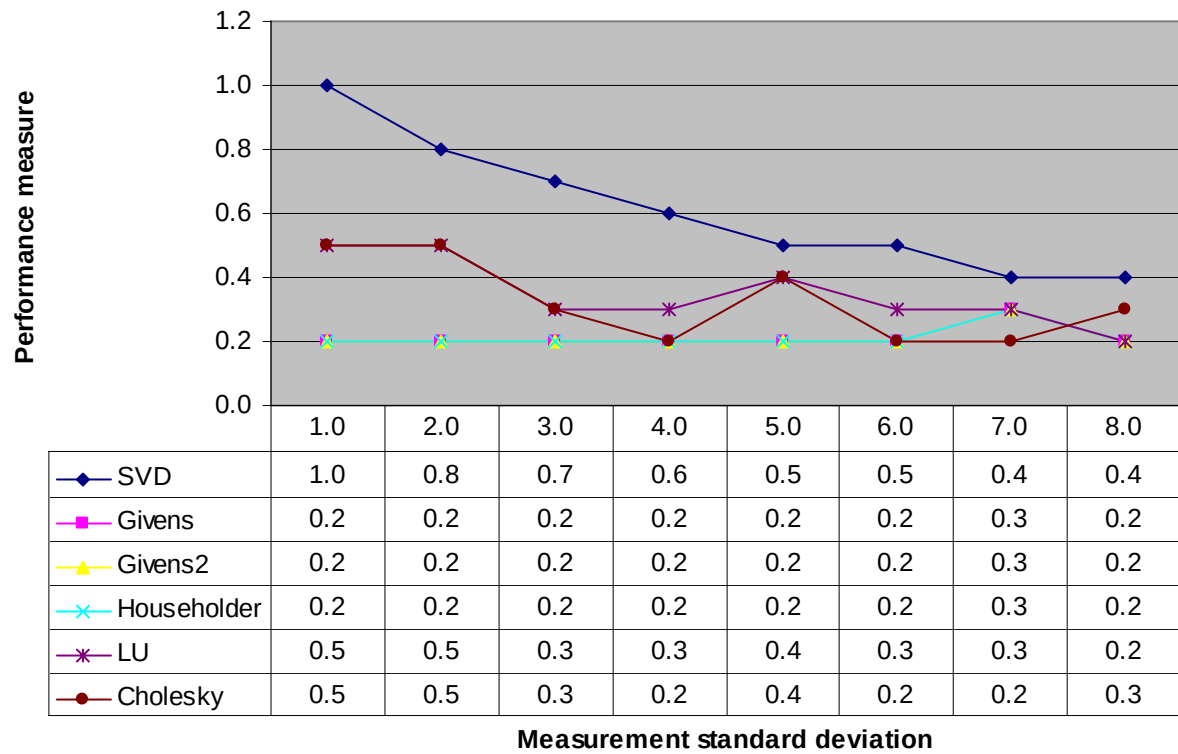


Figure 19: As Figure 16 except that the performance parameter is the measurement standard deviation.