NPL Report CISE 27/99

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

TESTING SPREADSHEETS AND OTHER PACKAGES USED IN METROLOGY

TESTING THE INTRINSIC FUNCTIONS OF EXCEL

BY

H R COOK, M G COX, M P DAINTON

and P M HARRIS

September 1999

Testing Spreadsheets and Other Packages Used in Metrology
Testing the Intrinsic Functions of Excel

H R Cook, M G Cox, M P Dainton and P M Harris
Centre for Information Systems Engineering

September 1999

ABSTRACT

The aim of the work reported here is to contribute to the development of an infrastructure, comprising supporting information and guidelines, to ensure that the use of software, particularly spreadsheets and proprietary software packages, within metrology is made as effective as possible. This is to be achieved by reporting the results of the objective testing of the intrinsic and in-built functions included within spreadsheets and other proprietary software packages that are popular in metrology applications.

We describe the application of a general methodology for testing the numerical accuracy of scientific software to specific functions taken from the Microsoft Excel spreadsheet package. Each stage of the methodology, from the provision of a specification for each function tested through the definition of performance parameters and measures to the presentation and interpretation of the test results, is presented. In this way, and by stating any assumptions made in the application of the methodology, the testing undertaken is made as objective as possible.

This report constitutes one of the deliverables of Project 2.1 “Testing Spreadsheets and Other Packages Used in Metrology” within the UK Department of Industry’s National Measurement System *Software Support for Metrology* Programme 1998–2001.

ISSN 1361-407X

Authorised by Dr Dave Rayner,
Head of the Centre for Information Systems Engineering

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

Contents

1. Introduction

The National Measurement System (NMS) Software Support for Metrology (SSfM) programme is a UK government-supported programme designed to tackle a wide range of generic issues concerning the use of mathematics and software in metrology. One of the topics addressed in the programme is the validation of software used in metrology. Validation here covers

- establishing the numerical correctness of the software, and
- ensuring that sound software engineering principles are employed when designing and implementing the software.

Numerical correctness, which is the issue addressed in this work, is important to metrology because a poor software implementation can lead to inaccuracy that could have been avoided, and as a consequence the accuracy of measurement results can be compromised. The aim of the work reported here is to contribute to the development of an infrastructure, comprising supporting information and guidelines, to ensure that the use of software, particularly spreadsheets and proprietary software packages, within metrology is made as effective as possible. This is to be achieved by reporting the results of the objective testing of the intrinsic and in-built functions included within spreadsheets and other proprietary software packages that are popular in metrology applications.

The work is primarily aimed at *users* of software packages within metrology applications. It is intended that the results presented will help users to understand whether for a particular application the functions used are fit for purpose, and to understand the limitations (if any) of those functions. In addition, however, it is anticipated that the work will be useful to the *developers* of the software, for example by highlighting any weaknesses identified in the functions tested.

The reports [1, 2] provide a review of

- the extent to which software packages including spreadsheets are used within metrology, and
- the effort put into validating and testing these packages.

The report [1] is a general survey of the status of the mathematics and software used to support each metrology area in the UK NMS, and [2] reports on existing work worldwide on spreadsheet quality and reliability with recommendations on its applicability in the metrology field.

Microsoft Excel for Windows 95 is identified as the “off the shelf” software package most commonly-used in the NMS metrology programmes. The main reasons for its widespread use include:

- its integration within the Microsoft Office software suite,
- the large range of intrinsic functions available,
- the convenience of its graphical user-interface, and
- its general usability enabling results to be generated quickly.

It is accepted that spreadsheets are a useful and popular tool for processing and presenting data. In addition to simple data manipulation and display, some of the applications of spreadsheet packages listed in [1] include regression, performing statistical calculations on data (such as mean, standard deviation, t and F tests, etc.), evaluating measurement uncertainties and experimental design.

Although there is some awareness of the potential limitations of spreadsheets and other software packages arising from the use of inaccurate or unstable numerical algorithms, relatively little testing and validation is performed on such software. Often, there is a tendency to *rely* upon well-established packages and to perform only a small number of checks using alternative methods of calculation. In contrast, bespoke software, possibly written using the same packages, is usually tested in accordance with software development procedures, for example, by comparing calculated results with reference results or results determined using other software.

Specific needs relating to the testing and validation of spreadsheet models are identified in [2]. These include:

1. The requirement for information about the numerical accuracy of the intrinsic functions within spreadsheets. In particular, the need to identify the parameter ranges over which the functions may be used with confidence, and guidelines on how the particular functions should be used in the various stages of processing data.
2. The requirement for guidelines concerning the effective design of spreadsheet applications so that the design, implementation, testing and maintenance of these applications may be properly controlled.

It is the first of these requirements that this report addresses by presenting the results of testing undertaken on the in-built functions of the spreadsheet package Microsoft Excel for Windows 95 Version 7.0a. This version of the spreadsheet package is used widely in metrology applications. Future work will report on other versions, such as Microsoft Excel 97, as well as software packages other than Microsoft Excel that underpin metrology applications..

Some related work can be found in the literature (see [2] and the references therein). Notably, the paper [3] reports on the numerical accuracy of statistical functions in the version Microsoft Excel 97. It is stated there that the computation of some discrete distributions (Binomial, Poisson, Hypergeometric) fails for probabilities in the central range between 0.01 and 0.99 and for parameter values that cannot be judged as too extreme. Also, it is demonstrated that the computation of some continuous distributions and their quantiles (Normal, Chi-squared, *F* and *t*) give unreliable results in the tails of the distribution.

The report is organised as follows. In Section 2 we discuss the methodology underlying the testing carried out, and how it has been applied to the Microsoft Excel spreadsheet. The methodology is described in detail in [7], and relies on the use of reference data sets and corresponding reference results to undertake black-box testing, as well as other approaches including self-consistency and continuity checks. Sections 3, 4 and 5 contain details of the implementation of the methodology for the testing of Microsoft Excel. Section 3 contains specifications of the Microsoft Excel functions tested. Section 4 describes the particular tests implemented and the (so-called) performance measures that are used to quantify the performance of each function. In Section 5 we present the results of the testing, and provide some interpretation of these results. Section 6 contains our conclusions.

2. Methodology

A general methodology for evaluating the numerical accuracy of the results produced by scientific software is described in [4, 5, 6, 7] and illustrated by the case study [8]. 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 to be tested. The reference data sets and results are generated in a manner 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 requirements and is fit for its intended purpose.

In addition to black-box testing of software using reference data sets and results, other complementary tests that do not require the availability of reference data are also useful and have been employed in this work. These tests include (see [7]):

a) consistency checks where, for example, we check the consistency of a forward calculation followed by an inverse calculation such as in comparing values of $\sin^{-1}\{\sin x\}$ against x;

b) continuity checks where, for example, for functions that use different evaluation methods over sub-ranges of the function arguments(s) we investigate the continuity of the evaluation methods across the sub-range boundaries;

c) spot checks against tabulated values;

d) checks of solution characterisations.

The methodology for testing that has been applied, and which is described in [7], comprises the following six stages:

1. specification of the test software,
2. implementation of the test software,
3. specification of reference data sets,
4. specification of performance measures and testing requirements,
5. generation of reference pairs, and
6. presentation and interpretation of performance measures.

The first two of these stages are usually carried out as part of the software development process, although in practice with varying amounts of formality. Stages 3 to 6 constitute the approach to software testing advocated in [7] with their application by a software *developer* presented in the case study [8]. The application of the methodology described here, however, is from the perspective of a *user* (of a proprietary software package). A user will usually not be part of the software development process and, consequently, stages 1 and 2 above are replaced by

1. *documenting* the specification of the test software, and
2. *interfacing* to the test software.

Recording the results of these stages is, nevertheless, important because it serves to define the basis of the testing undertaken and to make clear any assumptions made about the test software. For example, the GROWTH function in Microsoft Excel provides an instance (Section 2.1) in which the specification provided by the software supplier is incomplete and it is necessary to augment it to make it unambiguous, perhaps following discussion with the test software supplier or perhaps following initial testing of some assumed interpretation of the incomplete specification. These considerations may subsequently affect the interpretation of the results of the testing, and their inclusion ensures that the testing process is made as objective as possible.

In addition to the test software itself, software is used for

a) generating reference data sets and corresponding reference results,

b) evaluating and presenting quality metrics and performance measures, and

c) applying complementary software tests (such as those listed above).

The generation of reference pairs, i.e., reference data sets and corresponding reference results, is described in Section 2.5. Generally, reference data sets are generated outside the Microsoft Excel environment. A Microsoft Excel macro is used to load the data into a worksheet and

apply the worksheet function that is tested to the data. Reference results are either generated with the data (using a *data generator*) and are treated in the same way, or are calculated by dynamic calls to *reference software*.

The evaluation of quality metrics and performance measures and their presentation, as well as the application of complementary software tests, is undertaken under the control of Microsoft Excel macros. The advantage of using macros to automate the testing process, rather than performing software tests via manual interactive use of the Microsoft Excel package, is that a significant number of tests can be undertaken quickly.

In the remainder of this section we give an overview of the implementation of each of these stages in the testing of Microsoft Excel worksheet functions. Further specific details of the implementation of the methodology are then presented in Sections 3, 4 and 5.

2.1 Documenting the specification of the test software

For testing of software to be meaningful, it is necessary to have a full and unambiguous specification of the task carried out by the software. If the specification is inconsistent with the task carried out by the software, testing in accordance with the specification might yield the conclusion that the software was deficient, when in fact it might be acceptable.

Each Microsoft Excel worksheet function is provided with a *help-file* that indicates the purpose of the function, and includes descriptions of the inputs, outputs and optional arguments required by the routine. This information is usually sufficient to enable a user to make effective use of the function. However, no information about the nature of the numerical algorithms employed is generally provided, and consequently a black-box approach to testing, as described in [7], is appropriate.

The information provided in the help-file is regarded in this work as providing the basis of a *specification* of the function against which the function is tested. As an example, consider the GROWTH worksheet function. The help-file states that the function

> "Fits an exponential curve to the data *known_y's* and *known_x's*, and returns the *y*-values along that curve for the array of *new_x's* that you specify".

The syntax for using the function is, furthermore, defined as

GROWTH(*known_y's*, *known_x's*, *new_x's*, *const*)

where the inputs to the function are

- *known_y's*, the set of *y*-values in the relationship $y = bm^x$,
- *known_x's*, the (optional) set of *x*-values in the relationship $y = bm^x$ corresponding to the *y*-values *known_y's*,[1]
- *new_x's*, the new *x*-values for which GROWTH returns corresponding *y*-values,
- *const*, a logical value specifying whether to force the constant *b* in the fitted relationship $y = bm^x$ to equal 1,

and the outputs of the function are

- *new_y's*, the corresponding *y*-values that GROWTH returns for the *new_x's*,

Although the inputs and outputs are well-defined, it is necessary to apply some interpretation regarding the purpose of the function in order to make it an unambiguous specification. Specifically, we assume that the computed model is a *least-squares* fit to the user's data. We

[1] If *known_x's* is omitted, it is assumed to be the set of values {1, 2, 3, …*n*}, where *n* is the number of *y*-values specified by *known_y's*.

believe this assumption is reasonable given that the related regression function TREND, that is concerned with linear regression models, explicitly refers to the method of least-squares.

The specification that we have developed here, based on the help-file and including the additional interpretation about the purpose of the function, now makes clear the requirements for generating reference data sets and corresponding results for testing the GROWTH function. These are to generate reference data sets consisting of values for the inputs (*known_y's*, *known_x's*, *new_x's*, *const*) and corresponding reference results consisting of values for the outputs *new_y's*, where the outputs are computed from the inputs according to the purpose of GROWTH given above

The specification of each intrinsic function tested is constructed from its help-file in a similar manner. Section 3 contains a summary of these specifications for all the functions tested.

2.2 Interfacing to the test software

The implementations of the functions tested in this work are the *worksheet* functions contained within Microsoft Excel for Windows 95 Version 7.0a [9]. Throughout, the functions are accessed directly via a Microsoft Excel worksheet. The Microsoft Excel worksheet is either created manually or generated automatically using a Microsoft Excel macro written or recorded in Microsoft Excel's macro-language VBA (Visual Basic for Applications). In either case, the created worksheet and its use of the worksheet function are intended to mimic the way users themselves can be expected to access the function.

2.3 Specification of reference data sets

The aim of the software testing undertaken within the framework of the Software Support for Metrology programme is to verify whether the software is fit for purpose within the context of the National Measurement System. Reference data sets are required to reflect, as far as possible, the type of data sets that would commonly be encountered in practical measurement processes. It is an important consideration that when a function is tested, the range of inputs over which it is tested should reflect and include those of its intended use.

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.

For example, consider the problem addressed by the Microsoft Excel function LINEST of finding the least-squares best-fit straight-line model

$$y = ax + b$$

to given data $\{(x_i, y_i): i = 1, \ldots, m\}$. Then, possible performance parameters include:

- the size m of data set,
- the reference value for the gradient a of the straight line model,
- the reference value for the intercept b of the straight line model, and
- the size of measurement error used to define the data values y_i, $i = 1, \ldots, m$.

Each of these parameters may be varied in turn, with the remaining parameters taking nominal values, and the performance of LINEST investigated as a function of each parameter. Where data sets are found for which the performance of the function is poor, these data sets can be described using the above performance parameters and related to properties of the user's

metrology application. In this way, users are provided with information about the circumstances in which it is safe to use the function and when it is not.

Performance parameters are also used when applying the complementary software tests. For example, the consistency check based on the identity

$$\sin^{-1}\{\sin(x + 2n\pi)\} - x = 0,$$

is applied with m equispaced values of x in the range $[-\pi/2, +\pi/2]$, where m and n are regarded as performance parameters for the test.

Section 4 lists the performance parameters for each Microsoft Excel function and complementary test.

2.4 Specification of performance measures and testing requirements

Quality metrics are used to quantify the performance of the test software for the sequences and 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". Generally, the quality metrics measure the departure of the test results returned by the test software from the reference results. The departure may be measured in (at least) three ways [7]:

1. the *absolute* difference between test and reference results, i.e.,

$$d_A(\mathbf{x}) = \left\| \mathbf{y}^{\text{test}} - \mathbf{y}^{\text{ref}} \right\|, \tag{1}$$

   where $\mathbf{x}$ denotes the input reference data set, and $\mathbf{y}^{\text{test}}$ and $\mathbf{y}^{\text{ref}}$ are, respectively, the test and reference results,

2. the *relative* difference between the test and reference results, i.e.,

$$d_R(\mathbf{x}) = \frac{\left\| \mathbf{y}^{\text{test}} - \mathbf{y}^{\text{ref}} \right\|}{\left\| \mathbf{y}^{\text{ref}} \right\|}, \tag{2}$$

3. a *performance measure* that accounts for factors such as the computational precision and conditioning of the problem. In [7] the following performance measure is derived:

$$P(\mathbf{x}) = \log_{10}\left(1 + \frac{1}{\kappa(\mathbf{x})\eta} \frac{\left\| \mathbf{y}^{\text{test}} - \mathbf{y}^{\text{ref}} \right\|}{\left\| \mathbf{y}^{\text{ref}} \right\|} \right), \tag{3}$$

   where $\kappa(\mathbf{x})$ measures the condition of the problem defined by the data set $\mathbf{x}$ [7], and η is the computational precision[2]. The performance measure $P(\mathbf{x})$ indicates the number of figures of accuracy lost by the test software over and above what software based on an optimally stable algorithm would produce.

For example, in the testing of the GROWTH worksheet function, the reference results $\mathbf{y}^{\text{ref}}$ take the form of the residual values associated with the fitted exponential model evaluated at the new x-values *new_x's* (Section 2.1), where the residual values are derived from the model values *new_y's* returned by the GROWTH function and the data values *known_y's*. It is shown in [7] that the condition number $\kappa(\mathbf{x})$ for the problem of computing $\mathbf{y}^{\text{ref}}$ is unity, and so the performance measure takes the form

[2] 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.

$$P(\mathbf{x}) = \log_{10}\left(1 + \frac{1}{\eta}\frac{\left\|\mathbf{y}^{\text{test}} - \mathbf{y}^{\text{ref}}\right\|}{\left\|\mathbf{y}^{\text{ref}}\right\|}\right). \tag{4}$$

There is no general rule for deciding which measure should be used; in any one particular instance measure (1) or (2) or (3) above can be appropriate. Some indication regarding choice can, however, be given, by way of examples.

For an oscillatory (periodic) function such as $\cos x$, which lies between -1 and $+1$, it is natural to expect that the best absolute accuracy returned by test software would be of the order of the computational precision η. If, however, the argument x did not lie in the fundamental interval from $-\pi$ to $+\pi$ the cosine routine would first have to carry out *argument reduction* before applying an evaluation formula. Suppose that x was of the order of 10^n, for some integer n. Then the subtraction of an appropriate multiple of 2π would ensure that the reduced argument lay within the required interval. However, this initial process is essentially one of subtractive cancellation which implies that n figures are irrecoverable. As a consequence, it cannot be expected that the accuracy of the test software would be of order η but instead of order $10^n\eta$. (Only if the test software and the reference software carried out argument reduction in an identical manner could the better accuracy be expected.)

It might be expected that the consideration for $\sin x$ would be similar and, indeed, in respect of argument reduction this is the case. However, a high-quality implementation of $\sin x$ would give good *relative* accuracy for sufficiently small $|x|$. The reason is that such an implementation would approximate $\sin x$ by a formula of the form $xf(x)$, where $f(x)$ is a finite series-approximation (in Chebyshev form, say) or a rational approximation to $\sin x/x$. This form is capable of delivering $\sin x$ to a relative accuracy of the order of η for x lying in the fundamental interval.

The function $\exp x$ is illustrative of yet a different phenomenon. This function is always positive and it might be expected that for all x high-quality software would return a result with good relative accuracy. This is indeed the case. In particular, even a simple algorithm based on the Taylor expansion of $\exp x$ about the origin is capable of delivering a relative accuracy of the order of η for positive values of x. The qualification is necessary since for positive x all terms in the Taylor series are positive, and thus no subtractive cancellation and hence loss of significant figures can occur. However, for negative x, although the magnitudes of the individual terms are identical to those for the corresponding positive value, the terms alternate in sign. Since for large negative x, $\exp x$ is very small, it is inevitable that serious subtractive cancellation and hence severe loss of significant figures would take place if this approach were used.[3,4]

It is therefore clear, even through these simple examples, that the measure to be employed should be chosen to depend on the particular function and even the range of its argument.

For the complementary software tests, there are equivalent quality metrics to those described above. Suppose a consistency check is based on the identity

$$h(x) - x = 0,$$

where

$$h(x) = g(y) \qquad \text{and} \qquad y = f(x),$$

[3] Such a possibility is avoided from the use of the identity $\exp(-x) = 1/\exp x$.

[4] In practice, the Taylor-series expansion is not used by high-quality software because it is inefficient and has certain other deficiencies.

with *g* being the formal inverse of *f*. For example, for the consistency check based on the identity

$$\sin^{-1}\{\sin(x+2n\pi)\}-x=0,$$

we have

$$g(y)=\sin^{-1}\{y\} \qquad \text{and} \qquad f(x)=\sin(x+2n\pi).$$

Then,

$$d_A(x)=|h(x)-x| \tag{5}$$

is an *absolute* measure of consistency between the functions *f* and *g*, and

$$d_R(x)=\frac{|h(x)-x|}{|x|} \tag{6}$$

is a *relative* measure of consistency.

It is less easy to use the metrics (5) and (6) to make *quantitative* statements about the test software, although *qualitative* judgements are possible. However, care is necessary: the result that the quality metric $d_A(x)$ given by (5) is zero does not imply that the individual functions *f* and *g* are working correctly, only that they form a consistent pair of functions for forward and inverse calculations. Consequently, it is important to supplement such checks with other tests, e.g., testing the individual functions *f* and *g* against other (possibly reference) implementations of the functions.

Section 4 lists the quality metrics for each Microsoft Excel function and complementary test.

2.5 Generation of reference pairs

A *reference pair*, i.e., a reference data set and corresponding reference results, may be produced in two ways [7]:

1. Start with a reference data set and apply reference software to produce corresponding reference results.
2. Start with some reference results and apply a data generator to them to produce a corresponding reference data set.

In the testing reported on here, both approaches to generating reference pairs are adopted.

For the testing of regression functions, including the LINEST, LOGEST, TREND and GROWTH worksheet functions, for which a mathematical characterisation of the solution exists, *data generators* are used to construct reference data sets with known solutions, i.e., solutions specified *a priori*. The basis of the data generators are *null-space methods* [7] that use the solution characterisation to construct families or classes of data sets possessing nominally the same solution.

In the application of the complementary tests, particularly consistency checks, reference values are available from analytic considerations. For example, for the consistency check based on the identity

$$\sin^{-1}\{\sin(x+2n\pi)\}-x=0,$$

we can regard "*x*" as the reference data set, "$\sin^{-1}\{\sin(x + 2n\pi)\}$" as the test value returned by the software, and "*x*" as the reference value. This is consistent with the form of the quality metric for this test given in Section 2.4.

For many of the intrinsic functions tested, the test results returned by the functions are compared with values obtained from other implementations of functions. The software employed for this purpose is the IMSL Intrinsic Function Library supplied with the Digital Visual Fortran 90 software package [10]. This particular library was chosen for convenience and because the functions included within it are based on reliable and established numerical algorithms developed over many years. Of course there are other sources of high-quality software that could be used for this purpose: they include the NAG library, ACM Transactions on Mathematical Software, and Applied Statistics. It is proposed that as part of future work the sources of reference software used for testing will be expanded to include a number of these libraries.

2.6 Presentation and interpretation of performance measures

Having applied the test software to a reference data set to obtain a test result, the test result is compared with the reference result corresponding to the data set by computing a quality metric or performance measure such as is defined by (1), (2) or (3) (Section 2.4). The quality metrics are presented as functions of each (one or more) performance parameter (Section 2.3) either in tabular form or as a graph against the performance parameter, i.e., as a performance profile. It is one of the benefits of Microsoft Excel that the presentation of results in either form is made very easy, and consequently automating the presentation of the results as part of the testing procedure is straightforward to accomplish.

To use the testing results, for example, where presented in the form of a performance profile, the user needs to

1. decide the range of values of the performance parameter that correspond to the application, and hence identify that part of the performance profile appropriate to the application, and
2. decide whether the values of the quality metric over the identified range of the performance profile meet the accuracy requirements of the application.

In addition, by examining the performance over the complete range of the performance parameter, statements can be made about the general performance of the test software with respect to this parameter.

For example, one of the performance parameters used in the testing of the GROWTH worksheet function is the signal-to-noise ratio. A graph is provided showing values of the performance measure $P(\mathbf{x})$ given by (4) against this parameter. The graph can be used

- to identify values of the signal-to-noise ratio for which the performance of the GROWTH function meets a specified accuracy requirement, e.g., the results are to be accurate to a specified number of significant figures, or
- to assess the performance of the GROWTH function for data sets characterised by a particular range of signal-to-noise ratio values.

For the complementary software tests, quality metrics, such as those defined by (5) and (6) (Section 2.4), are presented (in tabular or graphical form) as a function of the input argument for various choices of the performance parameters (Section 2.3). For example, for the consistency check based on the identity

$$\sin^{-1}\{\sin(x + 2n\pi)\} - x = 0,$$

with $x \in [-\pi/2, +\pi/2]$, we show a plot of the quality metric $d_R(x)$ against x for various choices of the integer n. In addition to interpreting the value of the quality metric over sub-ranges of x of interest, considerations include the extent to which the quality metric varies smoothly with x, and the identification of the presence and position of extreme points and discontinuities in the metric.

3. Specifications of the Intrinsic Functions

In this section we list the Microsoft Excel functions tested as part of this work, and provide specifications for these functions. All the functions tested are *worksheet* functions, and the descriptions of each are based on the on-line help documentation provided with Microsoft Excel. We use the on-line documentation since this is the source of information, because of its easy access and availability, that is probably employed most by users of Microsoft Excel. The descriptions include the purpose of the functions as well as any documented restrictions on their arguments. Where information about the algorithm implemented by the function is provided, this is included in the description.

The functions are grouped into the following classes:

1. regression functions,
2. mathematical and trigonometric functions, and
3. statistical distributions.

This grouping reflects the intended purpose of the functions as well as the methods used to test them (Section 4). We include the AVERAGE and STDEV worksheet functions within the class of regression functions because their outputs can be related to the problem of finding the least-squares best-fit constant to the set of sample values and, consequently, the generation of reference data sets and corresponding reference results can be performed in a similar way to the other regression functions listed.

Specifications for the functions tested from each of the above classes are listed in, respectively, Sections 3.1, 3.2 and 3.3. As indicated above, these specifications are extracted from, or paraphrase, the Microsoft Excel on-line help documentation. Our comments on the specifications are included in Section 3.4.

The particular implementations of the functions tested in this work are those contained within Microsoft Excel for Windows 95 Version 7.0a [9].

3.1 Regression Functions

AVERAGE

Returns the average (arithmetic mean) of its arguments.

STDEV

Estimates the standard deviation *s* based on a sample of the population. The function uses the formula

$$s = \sqrt{\frac{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}{n(n-1)}}, \tag{7}$$

where $\{x_i: i = 1, \ldots, n\}$ is the sample.

LINEST

Uses the least-squares method to calculate the best-fit linear function

$$y = m_1 x_1 + m_2 x_2 + \ldots + m_n x_n + b \tag{8}$$

to given data $\{(y_i, x_{1i}, x_{2i}, \ldots, x_{ni}): i = 1, 2, \ldots\}$. Returns values for the parameters $m_1, m_2, \ldots, m_n$ and b.

LOGEST

Calculates the best-fit exponential function

$$y = bm_1^{x_1} m_2^{x_2} \ldots m_n^{x_n}$$

to given data $\{(y_i, x_{1i}, x_{2i}, \ldots, x_{ni})$: $i = 1, 2, \ldots\}$. Returns values for the parameters $m_1, m_2, \ldots, m_n$ and b.

TREND

Uses the least-squares method to calculate the best-fit linear function

$$y = m_1 x_1 + m_2 x_2 + \ldots + m_n x_n + b$$

to given data $\{(y_i, x_{1i}, x_{2i}, \ldots, x_{ni})$: $i = 1, 2, \ldots\}$ and evaluates the fitted function at given points. Returns values of the fitted function y at the given points.

GROWTH

Calculates the best-fit exponential function

$$y = bm_1^{x_1} m_2^{x_2} \ldots m_n^{x_n}$$

to given data $\{(y_i, x_{1i}, x_{2i}, \ldots, x_{ni})$: $i = 1, 2, \ldots\}$ and evaluates the fitted function at given points. Returns values of the fitted function y at the given points.

3.2 Mathematical and Trigonometric Functions

ACOS

Returns the arccosine of a number. The number must lie in the range −1 to +1, and the returned angle is given in radians in the range 0 to π.

ACOSH

Returns the inverse hyperbolic cosine of a number. The number must be greater than or equal to +1.

ASIN

Returns the arcsine of a number. The number must lie in the range −1 to +1, and the returned angle is given in radians in the range −π/2 to +π/2.

ASINH

Returns the inverse hyperbolic sine of a number.

ATAN

Returns the arctangent of a number. The returned angle is given in radians in the range −π/2 to +π/2.

ATAN2

Returns the arctangent of the specified x- and y-coordinates. The arctangent is the angle from the x-axis to a line containing the origin (0, 0)) and the specified point (x, y). The returned angle is given in radians in the range −π to +π excluding −π.

ATANH

Returns the inverse hyperbolic tangent of a number. The number must lie in the range −1 to +1 (excluding −1 and +1).

COS

Returns the cosine of an angle given in radians.

COSH

Returns the hyperbolic cosine of a number.

EXP

Returns *e* raised to the power of a number, where *e* is the base of the natural logarithm.

LN

Returns the natural logarithm of a number. The number must be positive.

LOG

Returns the logarithm of a number to a specified base. The number must be positive.

LOG10

Returns the base-10 logarithm of a number. The number must be positive.

SIN

Returns the sine of an angle given in radians.

SINH

Returns the hyperbolic sine of a number.

TAN

Returns the tangent of an angle given in radians.

TANH

Returns the hyperbolic tangent of a number.

3.3 Statistical Distributions

BETADIST

Returns the cumulative beta probability density function BETADIST(x, α, β, A, B), where α and β are the parameters of the distribution and A and B are (optional) lower and upper bounds for the interval containing x.

Returns the "#NUM!" error value if

- $\alpha \leq 0$ or $\beta \leq 0$,
- $x < A$, $x > B$, or $A = B$.

BETAINV

Returns the inverse of the cumulative beta probability density function BETAINV(p, α, β, A, B), where α and β are the parameters of the distribution and A and B are (optional) lower and upper bounds for the interval containing x.

Returns the "#NUM!" error value if

- $\alpha \leq 0$ or $\beta \leq 0$,
- $p \leq 0$ or $p > 1$.

BETAINV uses an iterative technique for calculating the function. Given a probability value, BETAINV iterates until the result is accurate to within $\pm 3 \times 10^{-7}$. If BETAINV does not converge after 100 iterations, the function returns the "#N/A" error value.

BINOMDIST

Returns the individual term binomial distribution probability BINOMDIST(*x*, *n*, *p*, *cumulative*) given by

$$b(x,n,p)=\binom{n}{x}p^{x}(1-p)^{n-x},$$

or its cumulative distribution

$$B(x,n,p)=\sum_{y=0}^{x}b(y,n,p),$$

where *x* is the number of successful trials, *n* is the total number of independent trials, *p* is the probability of a successful trial, and *cumulative* is true if the cumulative distribution is required and false otherwise.

Returns the "#NUM!" error value if

- $x < 0$ or $x > n$,
- $p < 0$ or $p > 1$.

x and *n* are truncated to integers.

CHIDIST

Returns the one-tailed probability of the chi-squared distribution CHIDIST(*x*, ν), where ν is the number of degrees of freedom associated with the distribution.

Returns the "#NUM!" error value if

- *x* is negative,
- $\nu < 1$ or $\nu \geq 10^{10}$.

If ν is not an integer, it is truncated.

CHIINV

Returns the inverse of the one-tailed probability of the chi-squared distribution CHIINV(*p*, ν), where ν is the number of degrees of freedom associated with the distribution.

Returns the "#NUM!" error value if

- $p < 0$ or $p > 1$,
- $\nu < 1$ or $\nu \geq 10^{10}$.

If ν is not an integer, it is truncated.

CHIINV uses an iterative technique for calculating the function. Given a probability value, CHIINV iterates until the result is accurate to within $\pm 3 \times 10^{-7}$. If CHIINV does not converge after 100 iterations, the function returns the "#N/A" error value.

EXPONDIST

Returns the exponential distribution EXPONDIST(*x*, λ, *cumulative*) with probability density function

$$f(x,\lambda)=\lambda e^{-\lambda x}$$

and cumulative distribution function

$$F(x,\lambda)=1-e^{-\lambda x}.$$

where λ is the parameter of the distribution, and *cumulative* is true if the cumulative distribution is required and false otherwise.

Returns the "#NUM!" error value if

- $x < 0$ or $\lambda \leq 0$.

FDIST

Returns the *F* probability distribution FDIST(x, ν_1, ν_2), where ν_1 and ν_2 are the numbers of degrees of freedom in the numerator and denominator, respectively, associated with the distribution.

Returns the "#NUM!" error value if

- x is negative,
- If $\nu_1 < 1$ or $\nu_1 \geq 10^{10}$,
- If $\nu_2 < 1$ or $\nu_2 \geq 10^{10}$.

If ν_1 or ν_2 is not an integer, it is truncated.

FINV

Returns the inverse of the *F* probability distribution FINV(p, ν_1, ν_2), where ν_1 and ν_2 are the numbers of degrees of freedom in the numerator and denominator, respectively, associated with the distribution.

Returns the "#NUM!" error value if

- $p < 0$ or $p > 1$,
- $\nu_1 < 1$ or $\nu_1 \geq 10^{10}$,
- $\nu_2 < 1$ or $\nu_2 \geq 10^{10}$.

If ν_1 or ν_2 not an integer, it is truncated.

FINV uses an iterative technique for calculating the function. Given a probability value, FINV iterates until the result is accurate to within $\pm 3 \times 10^{-7}$. If FINV does not converge after 100 iterations, the function returns the "#N/A" error value.

FISHER

Returns the Fisher transformation FISHER(x) of x, given by

$$y = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right).$$

Returns the "#NUM!" error value if

- $x \leq -1$ or $x \geq +1$.

FISHERINV

Returns the inverse of the Fisher transformation FISHERINV(y) of y, given by

$$x = \frac{e^{2y}-1}{e^{2y}+1}.$$

GAMMADIST

Returns the gamma distribution GAMMADIST(x, α, β, *cumulative*), given by

$$f(x,\alpha,\beta) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{x/\beta},$$

where α and β are the parameters of the distribution, and *cumulative* is true if the cumulative distribution is required and false otherwise.

Returns the "#NUM!" error value if

- $x < 0$,
- $\alpha \le 0$ or $\beta \le 0$.

GAMMAINV

Returns the inverse of the gamma cumulative distribution GAMMAINV(p, α, β), where α and β are the parameters of the distribution.

Returns the "#NUM!" error value if

- $p < 0$ or $p > 1$,
- $\alpha \le 0$ or $\beta \le 0$.

GAMMAINV uses an iterative technique for calculating the function. Given a probability value, GAMMAINV iterates until the result is accurate to within $\pm 3 \times 10^{-7}$. If GAMMAINV does not converge after 100 iterations, the function returns the "#N/A" error value.

GAMMALN

Returns the natural logarithm of the gamma function $\Gamma(x)$, where

$$\Gamma(x) = \int_0^\infty e^{-u} u^{x-1} du.$$

Returns the "#NUM!" error value if

- $x \le 0$.

HYPGEOMDIST

Returns the hypergeometric distribution HYPGEOMDIST(x, n, M, N), given by

$$h(x, n, M, N) = \frac{\binom{M}{x}\binom{N-M}{N-x}}{\binom{N}{n}},$$

where x is the number of successes in the sample, n is the size of the sample, M is the number of successes in the population, and N is the size of the population.

Returns the "#NUM!" error value if

- $x < 0$ or x is greater than the lesser of n or M,
- x is less than the larger of 0 and $(n - N + M)$,
- $n < 0$ or $n > N$,
- $M < 0$ or $M > N$,
- $N < 0$.

LOGINV

Returns the inverse of the lognormal cumulative distribution function LOGINV(p, μ, σ) of x, where $\ln(x)$ is normally distributed with parameters μ and σ.

Returns the "#NUM!" error value if

- $p < 0$ or $p > 1$,
- $\sigma \leq 0$.

LOGNORMDIST

Returns the cumulative lognormal distribution LOGNOMDIST(x, μ, σ) of x, where $\ln(x)$ is normally distributed with parameters μ and σ.

Returns the "#NUM!" error value if

- $x \leq 0$,
- $\sigma \leq 0$.

NORMDIST

Returns the normal cumulative distribution NORMDIST(x, μ, σ, *cumulative*) with probability density function

$$f(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2},$$

where μ is the mean of the distribution, σ is its standard deviation, and *cumulative* is true if the cumulative distribution is required and false otherwise. Returns the "#NUM!" error value if

- $\sigma \leq 0$.

NORMINV

Returns the inverse of the normal cumulative distribution NORMINV(p, μ, σ), where μ is the mean of the distribution and σ is its standard deviation.

Returns the "#NUM!" error value if

- $p < 0$ or $p > 1$,
- $\sigma \leq 0$.

NORMINV uses an iterative technique for calculating the function. Given a probability value, NORMINV iterates until the result is accurate to within $\pm 3 \times 10^{-7}$. If NORMINV does not converge after 100 iterations, the function returns the "#N/A" error value.

NORMSDIST

Returns the standard normal cumulative distribution function NORMSDIST(z). The distribution has a mean of zero and a standard deviation of one, and probability density function

$$f(z,0,1) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}.$$

NORMSINV

Returns the inverse of the standard normal cumulative distribution NORMSINV(p). The distribution has a mean of zero and a standard deviation of one.

Returns the "#NUM!" error value if

- $p < 0$ or $p > 1$.

NORMSINV uses an iterative technique for calculating the function. Given a probability value, NORMSINV iterates until the result is accurate to within $\pm 3 \times 10^{-7}$. If NORMSINV does not converge after 100 iterations, the function returns the "#N/A" error value.

POISSON

Returns the Poisson distribution POISSON(x, λ, *cumulative*) with probability density function

$$f(x,\lambda) = \frac{e^{-\lambda}\lambda^{x}}{x!},$$

where λ is the mean of the distribution, and *cumulative* is true if the cumulative distribution is required and false otherwise.

Returns the "#NUM!" error value if

- $x \le 0$,
- $\lambda \le 0$.

If *x* is not an integer, it is truncated.

TDIST

Returns the Student's t-distribution TDIST(*x*, ν, *tails*), where ν is the number of degrees of freedom associated with the distribution and *tails* controls whether a one- ot two-tailed distribution is considered.

Returns the "#NUM!" error value if

- $\nu < 1$,
- *tails* is any value other than 1 or 2.

The ν and tails arguments are truncated to integers.

TINV

Returns the inverse of the two-tailed Student's t-distribution TINV(*p*, ν), where ν is the number of degrees of freedom associated with the distribution.

Returns the "#NUM!" error value if

- $p < 0$ or $p > 1$,
- $\nu < 1$.

If ν is not an integer, it is truncated.

TINV uses an iterative technique for calculating the function. Given a probability value, TINV iterates until the result is accurate to within $\pm 3 \times 10^{-7}$. If TINV does not converge after 100 iterations, the function returns the "#N/A" error value

WEIBULL

Returns the Weibull distribution WEIBULL(*x*, α, β, *cumulative*) with probability density function

$$f(x,\alpha,\beta) = \frac{\alpha}{\beta^{\alpha}} x^{\alpha-1} e^{-(x/\beta)^{\alpha}},$$

and cumulative distribution function

$$F(x,\alpha,\beta) = 1 - e^{-(x/\beta)^{\alpha}},$$

where α and β are parameters of the distribution, and *cumulative* is true if the cumulative distribution is required and false otherwise.

Returns the "#NUM!" error value if

- $x < 0$,
- $\alpha \le 0$ or $\beta \le 0$.

3.4 Remarks

3.4.1 Regression functions

The on-line help documentation for the STDEV worksheet function makes explicit reference to the formula (7) employed by the function. This is in contrast to many of the other functions that provide no details about the numerical algorithms or formulae used. Unfortunately, it is well known [4, 5, 6, 7] that this formula has the property that it suffers from subtractive cancellation for data sets for which the mean $\bar{x}$ is large compared to the standard deviation *s*, i.e., for which the coefficient of variation $s/\bar{x}$ is small. Furthermore, a floating-point error analysis of (7) shows that the number of incorrect significant figures in the results obtained from the formula is about twice that for the *mathematically* equivalent form

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

Similarly, for the case of a single independent variable x_1, the on-line documentation for the LINEST worksheet function gives the formulae on which the calculations of the parameters m_1 and b are based. Again, it is well known that these formulae can suffer from subtractive cancellation and, consequently, they should be avoided for general use. Even if the LINEST worksheet function does not implement these formulae, the documentation provided to the user is failing to promote good practice in terms of the numerical algorithms that might be used.

If we define

$$x_1 = x, \quad x_2 = x^2, \quad \ldots, \quad x_j = x^j, \quad x_n = x^n,$$

then the linear function (8) reduces to the *polynomial* model

$$y = a_1 x + a_2 x^2 + \ldots + a_n x^n + b.$$

The documentation for the TREND worksheet function suggests that in this way the function can be used for the purpose of evaluating the best-fit polynomial model to given data. However, especially for high powers and $x > 0$, the basis functions in this representation are similar in appearance, and consequently there is numerical instability when used in polynomial regression. The situation is improved if the data is transformed to the range [−1, +1], since there is then low correlation between even and odd powers of x and the power series representation may be safe for fits of low to moderate degree if solution methods based on matrix factorisations are used. Nevertheless, for general polynomial regression it is advisable to employ an alternative polynomial basis, e.g., Chebyshev polynomials, that exhibits better numerical properties.

Although it is not stated explicitly in the on-line help documentation, it is assumed here that the "best-fit" exponential function returned by the LOGEST and GROWTH worksheet functions is a *least-squares* fit to the user's data, i.e., the parameters m_1, m_2, …, m_n and b are determined to minimise the function

$$\sum_i \left\{ y_i - \left(b m_1^{x_{1i}} m_2^{x_{2i}} \ldots m_n^{x_{ni}} \right) \right\}^2.$$

We believe this assumption is reasonable given that the related functions LINEST and TREND explicitly refer to the least-squares method. The results of the testing reported in Section 5 can be used to confirm whether this assumption is valid. If it is not then we may conclude that the on-line help documentation is ambiguous in its description of the purpose of the LOGEST and GROWTH functions, as well as potentially misleading for users of these functions.

3.4.2 Statistical distributions

The amount and type of information about these functions varies from function to function. The descriptions sometimes include an explicit "mathematical" definition of the distribution in terms of its probability density and cumulative distribution functions. In cases where this is not provided, the user is therefore required to have sufficient knowledge to be able to make proper use of the function.

Details of the numerical algorithms and methods used are not provided, although where an iterative technique is employed, the convergence criterion is indicated. This criterion, *viz*., "… iterates until the result is accurate to within $\pm 3 \times 10^{-7}$" provides a *claim* about the software that can be validated by an appropriate performance test. An alternative criterion that the iteration is continued until successive iterates differ by less than 3×10^{-7} may be more realistic.

4. Specification of Performance Parameters and Measures

4.1 Regression Functions

For the testing of the intrinsic regression functions within Microsoft Excel, we follow the methodology of generating reference data sets and results using data generators described in [7]. Consequently, reference data sets with corresponding reference results are readily constructed, and a quantitative quality metric is explicitly available [7, 8]. We present here for each regression function tested, the performance parameters used to define the reference data sets and the performance measure used to compare the results returned by the function tested with the reference results.

AVERAGE

Performance parameters for this function include

- the reference sample mean,
- the number of points in the sample, and
- $\dfrac{\bar{x}^{\text{ref}}}{s}$, the inverse of the coefficient of variation, where s is the sample standard deviation.

The performance measure $P(\bar{x})$ used to measure the departure of the computed sample mean $\bar{x}$ returned by the test software from the reference sample mean $\bar{x}^{\text{ref}}$ is given by

$$P(\bar{x}) = \log_{10}\left(1 + \frac{1}{\kappa(\bar{x})\eta}\frac{\left|\bar{x} - \bar{x}^{\text{ref}}\right|}{\left|\bar{x}^{\text{ref}}\right|}\right), \tag{9}$$

where $\kappa(\bar{x})$ measures the relative conditioning of the problem,

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

and η is the machine precision.

STDEV

Performance parameters for this function include

- the reference sample mean,
- the number of points in the sample, and
- $\dfrac{\bar{x}}{s^{\text{ref}}}$, the inverse of the coefficient of variation, where $\bar{x}$ is the sample mean.

The performance measure $P(s)$ used to measure the departure of the computed sample standard deviation s returned by the test software from the reference sample standard deviation s^{ref} is given by

$$P(s) = \log_{10}\left(1 + \frac{1}{\kappa(s)\eta}\frac{\left|s - s^{\text{ref}}\right|}{\left|s^{\text{ref}}\right|}\right), \tag{10}$$

where $\kappa(s)$ measures the relative conditioning of the problem,

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

and η is the machine precision [7].

LINEST and **LOGEST**

Performance parameters for these functions include

- the regression coefficients,
- the number of points in the data set,
- the location and spread of the data x-values, and
- the size of measurement noise simulated in the generated data.

The performance measure $P(\mathbf{a})$ used to measure the departure of the computed regression coefficients $\mathbf{a}$ returned by the test software from the reference coefficients $\mathbf{a}^{\text{ref}}$ is given by

$$P(\mathbf{a}) = \log_{10}\left(1 + \frac{1}{\kappa(\mathbf{a})\eta}\frac{\left\|\mathbf{a} - \mathbf{a}^{\text{ref}}\right\|}{\left\|\mathbf{a}^{\text{ref}}\right\|}\right),$$

where $\kappa(\mathbf{a})$ measures the relative conditioning of the problem, and η is the machine precision [7]. Here, $\kappa(\mathbf{a})$ is computed as the condition number of the (column-normalised) Jacobian matrix J for the associated least-squares problem evaluated at the reference solution, i.e., the (i, j)th element of J is the derivative of the model y (linear for LINEST and exponential for LOGEST) with respect to the jth regression parameter evaluated at the ith data point $\mathbf{x}_i = (x_{1i}, x_{2i}, \ldots, x_{ni})^{\text{T}}$ with $\mathbf{a} = \mathbf{a}^{\text{ref}}$. We also use a performance measure that relates to a *single* regression parameter a taken from the vector $\mathbf{a}$ of regression parameters that takes the form

$$P(a) = \log_{10}\left(1 + \frac{1}{\kappa(\mathbf{a})\eta}\frac{\left|a - a^{\text{ref}}\right|}{\left|a^{\text{ref}}\right|}\right), \tag{11}$$

where $\kappa(\mathbf{a})$ and η are as defined above.

TREND and **GROWTH**

Performance parameters for these functions include

- the regression coefficients,
- the number of points in the data set,
- the location and spread of the data x-values,
- the location and spread of the data x-values at which the fitted function is evaluated, and
- the signal-to-noise ratio.

The performance measure $P(\mathbf{e})$ used to measure the departure of the computed residuals $\mathbf{e}$ returned by the test software from the reference residuals $\mathbf{e}^{\text{ref}}$ is given by

$$P(\mathbf{e}) = \log_{10}\left(1 + \frac{\left\|\mathbf{e} - \mathbf{e}^{\text{ref}}\right\|}{\eta\|\mathbf{y}\|}\right), \tag{12}$$

where η is the machine precision [7].

4.2 Mathematical and Trigonometric Functions

The results returned by the functions ACOS, ASIN, ATAN, COS, COSH, EXP, LN, SIN, SINH, TAN and TANH are compared with the corresponding Fortran intrinsic function provided with the Digital Visual Fortran (version 5.0.D) software package [11]. The results returned by the functions ACOSH, SINH and ATANH are compared with the corresponding function from the IMSL Fortran 90 Math/Library (version 3.0) [10] provided with the Digital Visual Fortran (version 5.0.D) software package [11].

In each case, the quality metric $d_R(x)$ given by (2) that measures the *relative* difference between the results returned by the two software implementations of the functions is examined to discern any significant differences between the implementations.

In addition, the following consistency and continuity checks are performed:

T1. SIN(ASIN(x)) = x, $-1 \le x \le +1$.

T2. ASIN(SIN($x \pm 2n\pi$)) = x, $-\pi/2 \le x \le +\pi/2$, n integral.

T3. COS(ACOS(x)) = x, $-1 \le x \le +1$.

T4. ACOS(COS($x \pm 2n\pi$)) = x, $0 \le x \le +\pi$, n integral.

T5. TAN(ATAN(x)) = x.

T6. ATAN(TAN($x \pm 2n\pi$)) = x, $-\pi/2 < x < +\pi/2$, n integral.

T7. ATAN2(x, y) = ATAN(y/x), $x > 0$.

T8. ATAN2(COS($x \pm 2n\pi$), SIN($x \pm 2n\pi$)) = x, $-\pi < x \le +\pi$, n integral.

T9. SINH(ASINH(x)) = x.

T10. ASINH(SINH(x)) = x.

T11. COSH(ACOSH(x)) = x, $x \ge +1$.

T12. ACOSH(COSH(x)) = x.

T13. TANH(ATANH(x)) = x, $-1 < x < +1$.

T14. ATANH(TANH(x)) = x.

T15. $\text{SIN}^2(x) + \text{COS}^2(x) = 1$.

T16. TAN(x) = SIN(x)/COS(x).

T17. SINH(x) = (EXP(x) – EXP($-x$))/2.

T18. COSH(x) = (EXP(x) + EXP($-x$))/2.

T19. TANH(x) = SINH(x)/COSH(x).

T20. EXP(LN(x)) = x.

T21. LN(EXP(x)) = x.

T22. LOG10(x) = LN(x)/LN(10).

T23. LOG(x, EXP(1)) = LN(x).

T24. LOG(x, 10) = LOG10(x).

T25. $\mathrm{LOG}(y^x, y) = x$.

In each case, the quality metric $d_R(x)$ given by (6) that is a relative measure of consistency is used to make judgements about the test software in the following ways:

1. By highlighting cases for which the value of the quality metric is significantly greater than the computational precision η.
2. By identifying discontinuities in value and/or slope of the quality metric as a function of its argument, e.g., performance parameter.
3. By noting trends in the values of the quality metric, e.g., periodic behaviour of the quality metric.

4.3 Statistical Distributions

The results returned by individual functions are compared with those returned by the corresponding functions from the IMSL Fortran 90 Math/Library (version 3.0) [10] provided with the Digital Visual Fortran (version 5.0.D) software package [11].

In each case, the quality metric $d_R(x)$ given by (2) that measures the *relative* difference between the results returned by the two software implementations of the functions is examined to discern any significant differences between the implementations.

In addition, the following consistency and continuity checks are undertaken:

S1. $\mathrm{EXPONDIST}(x, \lambda, \mathit{cum}=\mathit{false}) = \lambda e^{-\lambda x}$, $x > 0$, $\lambda \geq 0$.

S2. $\mathrm{EXPONDIST}(x, \lambda, \mathit{cum}=\mathit{true}) = 1 - e^{-\lambda x}$, $x > 0$, $\lambda \geq 0$.

S3. $2 \times \mathrm{FISHER}(x) = \mathrm{LN}\{(1 + x)/(1 - x)\}$, $|x| < 1$.

S4. $\mathrm{FISHERINV}(y) = (\mathrm{EXP}(2y) - 1)/(\mathrm{EXP}(2y) + 1)$.

S5. $\mathrm{GAMMALN}(x) = \mathrm{LN}(\mathrm{GAMMA}(x))$, $x > 0$.

S6. $\mathrm{LOGINV}(x, \mu, \sigma) = \mathrm{EXP}(\mu + \sigma \times \mathrm{NORMSINV}(x))$, $0 \leq x \leq 1$.

S7. $\mathrm{LOGNORMDIST}(x, \mu, \sigma) = \mathrm{NORMSDIST}\{(\mathrm{LN}(x) - \mu)/\sigma\}$.

S8. $\mathrm{NORMDIST}(x, \mu, \sigma) = \mathrm{NORMSDIST}\{(x - \mu)/\sigma\}$.

S9. $\mathrm{NORMINV}(x, \mu, \sigma) = \mu + \sigma \times \mathrm{NORMSINV}(x)$, $0 \leq x \leq 1$.

S10. $\mathrm{WEIBULL}(x, \alpha, \beta, \mathit{cum}=\mathit{false}) = \alpha x^{\alpha-1} \times \mathrm{EXP}\{-(x/\beta)^{\alpha}\}/\beta^{\alpha}$, $x > 0$, $\alpha, \beta \geq 0$.

S11. $\mathrm{WEIBULL}(x, \alpha, \beta, \mathit{cum}=\mathit{true}) = 1 - \mathrm{EXP}\{-(x/\beta)^{\alpha}\}$, $x > 0$, $\alpha, \beta \geq 0$.

In each case, the quality metric $d_R(x)$ given by (6) that is a relative measure of consistency is used to make judgements about the test software in the following ways:

1. By highlighting cases for which the value of the quality metric is significantly greater than the computational precision η.
2. By identifying discontinuities in value and/or slope of the quality metric as a function of its argument, e.g., performance parameter.
3. By noting trends in the values of the quality metric, e.g., periodic behaviour of the quality metric.

5. Presentation and Interpretation of Results

5.1 Regression Functions

We give here some typical results obtained from the testing of the Microsoft Excel regression functions whose specifications are given in Section 3.1. In Section 4.1 the performance parameters and performance measures for each regression function are listed. The performance measures are presented graphically as functions of the performance parameters in the figures contained in Appendix A.

AVERAGE and **STDEV**

Figures 1–6 show plots of the performance measures $P(\bar{x})$ and $P(s)$ given by (9) and (10) for, respectively, the sample mean $\bar{x}$ and the sample standard deviation s against the following performance parameters:

- the reference sample mean,
- the number of points in the sample, and
- the population standard deviation.

Nominal values for these performance parameters are given in Table 1.

Parameter	*Nominal value*
Sample mean	+1.437575400258079
Number of points	50
Population standard deviation	0.1

Table 1 Nominal values for the performance parameters for the testing of the AVERAGE and STDEV worksheet function.

The results for the worksheet function AVERAGE (Figures 1–3) show that there is no significant degradation in the performance of the function over the ranges of performance parameters for which the function was tested. The value of the performance measure is approximately unity, indicating that the accuracy of the results is very close to that which we expect to be delivered by an optimal algorithm.

The results for the worksheet function STDEV are shown in Figures 4–6. In Figure 4, the performance parameter $\bar{x}$ is plotted on a logarithmic scale with s essentially fixed. The graph shows that there is degradation in the performance of the function as the coefficient of variation $s / \bar{x}$ is decreased. The function loses a number of significant figures over and above that which would be lost by a stable algorithm in a manner essentially proportional to $\log(\bar{x} / s)$. This result is predicted by a detailed floating-point error analysis of the formula (7) implemented by the STDEV worksheet function (Section 3.1).

In Figure 6, s is the performance parameter with $\bar{x}$ held constant. Again, the graph shows that there is degradation in the performance of the function as s is reduced to a value of about 10^{-8}. The function loses a number of significant figures over and above that which would be lost by a stable algorithm in a manner essentially proportional to $\log(\bar{x} / s)$. Note that, because $\bar{x} \approx 1$, when s equals 10^{-8} the condition $\kappa(s)$ of the problem is approximately 10^{+8} (see Section 4.1). This means that we expect an optimal algorithm to lose about eight significant figures of accuracy for a data set defined by these parameters. The value of the performance measure, which indicates the *additional* number of significant figures lost, for this value of s can be seen

in Figure 6 to be approximately eight. Consequently, all sixteen significant figures are lost, and the result returned by the STDEV function for this data set has *no* correct significant figures. This remains the case as s is reduced, and so for s less than 10^{-8} the performance measure behaves like $\log(s)$. This is evident in Figure 6, and is a consequence of the way the performance measure is calculated.

The accuracy loss illustrated in Figures 4 and 6 can be avoided by subtracting the sample mean from all the values in the sample before applying the STDEV function. Mathematically, the standard deviations of the given and shifted samples are identical, but numerically that of the second can be determined more reliably. This is an example where *pre-processing* of the data is beneficial to the numerical performance of the algorithm.

LINEST

Figures 7–14 show plots of the performance measures $P(m)$ and $P(b)$ given by (11) for the regression coefficients m and b in the linear model

$$y = mx + b$$

against the following performance parameters:

- the gradient m,
- the number of points in the data set,
- the size of measurement noise simulated in the generated data, and
- the location of the data x-values.

Nominal values for the linear model parameters and the performance parameters are given in Table 2.

Parameter	*Nominal value*
m	15.4502985108624
b	−1.437575400258079
Number of points	50
Measurement noise	0.1
Location of x-values	0

Table 2 Nominal values for the linear model parameters and performance parameters for the testing of the LINEST worksheet function.

It can be seen from the figures that no significant degradation in performance of the algorithm implemented in the LINEST worksheet function is observed for changes in the gradient, number of data points and size of measurement noise within the ranges indicated in the figures. The performance measure for the gradient m appears to be approximately within the range 2 to 2.5, and that for the intercept b between 1.5 and 2.5 for most of the tests shown. This may be interpreted (see [7]) as indicating a loss of approximately two significant figures of accuracy over and above that which an optimal algorithm would be expected to loose.

However, in Figures 13 and 14 we see a significant degradation in performance of the algorithm as a function of the location of the data x-values. These figures show the performance measures $P(m)$ and $P(b)$ to behave essentially as a linear function of the centroid $\bar{x}$ of the data x-values when the latter is displayed logarithmically. The performance measure $P(m)$ for the gradient m increases from a value of about 2 to between 7 and 8, and the corresponding measure $P(b)$ for

the intercept *b* can be seen to increase from about 2 to approximately 14 for large values of the centroid. As before, the values of the performance measures may be interpreted as the number of significant figures of accuracy lost by the algorithm over and above that which an optimal algorithm would lose.

It is a well known phenomenon (see, for example, [7] and [8]), that the numerical performance of linear least-squares algorithms, such as that implemented in LINEST, can be severely affected if the data is not centred about the origin. This is another example where *pre-processing* of the data, or an equivalent re-parametrisation of the linear model in the form

$$y = m(x - \bar{x}) + b,$$

is beneficial to the numerical performance of the algorithm. When pre-processing is not undertaken, the expected degradation in the performance measure as seen in Figures 13 and 14 is usually observed. It can be speculated that the absence of any data pre-processing or model re-parametrisation is what is causing the poor performance of the LINEST worksheet function under these conditions. An important conclusion to be drawn is that pre-processing of the data by the user prior to the application of the Microsoft Excel intrinsic function LINEST is recommended.

LOGEST

Figures 15–20 show plots of the performance measures $P(m)$ and $P(b)$ given by (11) for the regression coefficients *m* and *b* in the exponential model

$$y = bm^x \tag{13}$$

against the following performance parameters:

- the number of points in the data set,
- the size of measurement noise simulated in the generated data, and
- the location of the data *x*-values.

Nominal values for the exponential model parameters and the performance parameters are given in Table 3.

Parameter	*Nominal value*
m	15.4502985108624
b	−1.437575400258079
Number of points	50
Measurement noise	0.1
Location of *x*-values	0

Table 3 Nominal values for the exponential model parameters and performance parameters for the testing of the LOGEST worksheet function.

The results shown in Figures 15–20 relate to reference data sets and corresponding reference results generated in accordance with the (assumed) specification of the LOGEST worksheet function given in Section 3.1, i.e., given data points (x_i, y_i), the parameters *m* and *b* are determined to minimise the function

$$\sum_i \left\{ y_i - bm^{x_i} \right\}^2. \tag{14}$$

The figures illustrate a very poor performance of the LOGEST algorithm. Values for the performance measures of approximately 12 are observed throughout most of the range of the performance parameters, suggesting that there are very few significant figures in the results returned by the function that are accurate. It is concluded that either (a) the function does not solve the problem specified above, or (b) the numerical processing undertaken by the function is seriously flawed.

Note that if we take logarithms of the exponential model (13), we obtain

$$\log y = x \log m + \log b,$$

which expresses the variable "log y" as a *linear* function of the variable x with parameters log m and log b. Consequently, given data (x_i, log y_i), the problem of determining parameters log m and log b to minimise the function

$$\sum_i \left\{\log y_i - \left(x_i \log m + \log b\right)\right\}^2 \tag{15}$$

is a *linear* least-squares problem that may be solved using the LINEST worksheet function. However, it is noted [8] that the solutions to these two least-squares problems, the first involving the exponential model and the second a transformation of this model, are not equivalent.

Testing of the LOGEST worksheet function was carried out according to this alternative specification of the function. Reference data sets and corresponding reference results were generated for the LINEST worksheet function, and the inverse transformation applied to give test data and results for the LOGEST function. Performance measures for the parameters returned by the LOGEST function were calculated as functions of the performance parameters as described above. The corresponding performance profiles are shown in Figures 21−26.

It is seen from the figures that the performance profiles obtained show the same general behaviour to those corresponding to the LINEST worksheet function (Figures 9−14). It is concluded, therefore, that the LOGEST worksheet function is *not* finding the least-squares best-fit exponential model to given data as assumed, but returns an approximate solution to this problem by solving a transformed version of the least-squares problem.

TREND and **GROWTH**

Figures 27−29 and 30−32 show plots of the performance measure $P(\mathbf{e})$ given by (12) for the residuals **e** for, respectively, the TREND and GROWTH worksheet functions against the following performance parameters:

- the number of points in the data set,
- the signal-to-noise ratio, and
- the location of the data x-values.

We believe it is reasonable to assume that the TREND and GROWTH intrinsic functions make use of the fitting routines LINEST and LOGEST, respectively, to obtain the fitted linear and exponential models, and then evaluate these fitted models at the required points. Because the outputs returned are the y-values of points along the fitted curve, testing of the functions is based on comparing the residual values (which depend on the model values) returned by the test software and reference values calculated for each reference data set [7, 8].

For the TREND worksheet function (Figures 27−29), we note that the performance profiles obtained show the same general behaviour as the LINEST worksheet function. There is a marked deterioration in performance as the location of the data x-values, defined by the centroid $\bar{x}$ is increased. When the number of data points and the signal-to-noise ratio are varied, the performance measure remains reasonably constant, with values of the measure

approximately unity. As for the LINEST worksheet function, it can be expected that pre-processing of the data, i.e., shifting the data by subtracting $\bar{x}$ from the data x-values, is beneficial to the numerical performance of the TREND worksheet function.

For the GROWTH worksheet function (Figures 30–32), the results shown relate to reference data sets and corresponding reference results generated in accordance with the (assumed) specification of the GROWTH worksheet function given in Section 3.1, i.e., the best-fit exponential model is computed to be the "true" least-squares fit. The figures illustrate a poor performance of the algorithm, as would be expected from the testing of LOGEST reported above. When the number of data points and the location of the data x-values are varied, the performance measure takes values of between 12 and 14, and indicates the results returned by the function have very few significant figures that are accurate. This is unlikely to be acceptable for most users. However, when the signal-to-noise ratio is increased (or, equivalently, the amount of measurement noise is decreased), the performance of the function is improved. This is consistent with the fact that the fitted model obtained by minimising (15) becomes a better approximation to the model obtained by minimising (14) as the amount of measurement noise is decreased, and the solutions are identical when there is no measurement error. Consequently, there may be circumstances, i.e., when the user's data is highly accurate, in which the performance of the GROWTH worksheet function is satisfactory, because the errors introduced by the function as quantified in Figure 31 are not significant compared with the user's requirements.

5.2 Mathematical and Trigonometric Functions

In the comparison of the Microsoft Excel mathematical and trigonometric worksheet functions with equivalent Fortran intrinsic or IMSL functions, the relative differences $d_R(x)$ between the results returned by the software implementations were generally found to be of the order of the computational precision η, and consequently not regarded as significant. The only exception is for the function ASINH for which for some values of its argument differences of the order of 10^{-13} were encountered. However, these differences are probably sufficiently small for most applications. However, see the conclusions section (Section 6).

Figures 33 and 34 show graphs of the (relative) differences, i.e., the values of the quality metric, for, respectively, the functions SINH and ASINH. We note that the results for ASINH show good agreement for positive values of its argument x with differences between the results returned by the implementations occurring for negative values of x (Figure 34).

Since the inverse hyperbolic sine function is an *odd* function, i.e.,

$$\sinh^{-1}(x) = -\sinh^{-1}(-x), \tag{16}$$

the results for the ASINH worksheet function, that show an asymmetry about the axis $x = 0$, are noteworthy. In fact, we find that the ASINH function does not satisfy (16), with (absolute) differences between ASINH(x) and −ASINH(−x) being typically of the order of 10^{-13}. If we compare the IMSL inverse hyperbolic sine function with −ASINH(−x), instead of ASINH(x), we find the results shown in Figure 34 are reflected about the $x = 0$ axis. Consequently, *if* we regard the IMSL implementation of this function as a reference implementation, the performance of the ASINH function is improved for negative values of x by using the relationship (16).

We can postulate the following explanation for the behaviour of Microsoft Excel's ASINH worksheet function described above. The problem of evaluating the ASINH function at x is to solve for y the equation

$$x = \frac{e^y - e^{-y}}{2}.$$

Multiplying by e^y, we obtain

$$(e^y)^2 - 2xe^y - 1 = 0,$$

which is a quadratic equation in e^y. The roots of this equation are given *mathematically* by

$$e^y = x \pm \sqrt{1+x^2},$$

and the positive root, which determines the required solution y, is

$$e^y = x + \sqrt{1+x^2}.$$

Now, for positive x, e^y is evaluated as the sum of two positive terms. However, for negative x, e^y is evaluated as the difference of two positive terms whose magnitudes become larger and closer as x becomes more negative. The evaluation of e^y for negative x is therefore subject to subtractive cancellation, and this may explain the performance observed for the ASINH function. Note that this performance can be avoided by the use of numerically stable methods for determining the roots of the quadratic equation.

Figures 35 and 36 show graphs of the quality metric $d_R(x)$ for the complementary software tests T9 and T10. We note that these consistency checks, that involve the SINH and ASINH worksheet functions, show differences for negative values of x and a general behaviour which is consistent with our comments above about the ASINH function.

5.3 Statistical Distributions

Figures 37–48 show plots of the quality metric $d_R(x)$ for the comparison of a selection of the Microsoft Excel statistical distributions tested against the equivalent IMSL functions. We note:

1. Many of the graphs show *systematic* behaviour of the quality metrics, e.g., for BETADIST (Figures 37, 39 and 40), CHIDIST (Figure 41), FDIST (Figure 43), GAMMADIST (Figure 45), and NORMSDIST (Figure 47).
2. Many of the graphs show *discontinuities* in the behaviour of the quality metrics, e.g., for BETADIST (Figures 37, 39 and 40), CHIDIST (Figure 41), FDIST (Figure 43), and GAMMADIST (Figure 45).
3. Many of the graphs show that the differences between the implementations are greatest at *critical* values of the argument, e.g., for the basic distributions the critical points depend on the parameters of the distribution, such as the number of degrees of freedom (Figure 41), and for the inverse distributions the critical points are typically the extreme probability points $p = 0$ and $p = 1$.

Since for many applications in metrology for which probably only a few significant figures of accuracy are required when evaluating these distributions, the differences reported here are sufficiently small. However, see the conclusions section (Section 6).

The values of the quality metrics for the supplementary tests S1–S11 are all regarded as small for most applications. Again, see the conclusions section (Section 6). However, some simple spot-checks on the consistency of the statistical distribution functions and their inverses indicate behaviour that is noteworthy. Figures 49 and 50 show values of the quality metric $d_R(x)$ for, respectively, the consistency checks

S12. CHIINV(CHIDIST(x, 10), 10) – x = 0, and

S13. CHIINV(CHIDIST(x, 100), 100) – x = 0.

In each case, the values of the quality metric are large. It should be noted that the use of a “zero of a function” technique for evaluating the inverse distributions, e.g., the use of CHIDIST and a bisection algorithm for evaluating CHIINV, can be expected to yield the maximum possible precision, limited only by the quality of the “forward” distribution, e.g., CHIDIST. It is therefore surprising to observe the results illustrated in Figures 49 and 50.

6. Conclusions

In this report we have described the application of a general methodology [7] for testing the numerical accuracy of scientific software to specific functions taken from the Microsoft Excel spreadsheet package. Each stage of the methodology, from documenting a specification for each function tested through the definition of performance parameters and measures to the presentation and interpretation of the test results, has been 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 nature of the testing.

It has not been the intention in this work to consider *all* the intrinsic functions included with Microsoft Excel, but to concentrate on specific functions that are relevant to the numerical processing of measurement data undertaken in metrology applications. The work has, therefore, concentrated on regression functions, mathematical and trigonometric functions, and statistical distributions (including their inverses), that underpin the requirements of metrology. Consequently, conclusions drawn from the testing undertaken of a particular function must be interpreted in the context of that function only, and not in the context of other functions or the Microsoft Excel software package as a whole. Furthermore, the tests described here have been carried out in such a way that the functions have been used without taking account of information elsewhere, e.g., as contained in publications on Microsoft Excel or information posted on the World Wide Web); only the documentation available on-line as part of the normal Microsoft Excel software "environment" was used. This mode of use is deliberate, since we believe it accords with that adopted by most users generally and within metrology in particular.

The test results are intended primarily to help users understand whether for a particular application the functions used are fit for purpose, and to understand the limitations (if any) of those functions. Where the testing had indicated ways in which a user may make better use of a given function, for example by pre-processing the data prior to its application, to overcome any such limitation, this information has been provided.

The testing indicates that the numerical performance of some of the regression functions can be poor. This is caused either

- by the use of a mathematical formula (as in the STDEV worksheet function) or a model parametrisation (as in the LINEST and TREND worksheet functions) that can exacerbate the natural ill-conditioning of the problem to be solved, or
- by the solution of a problem that approximates that intended to be solved (as in the LOGEST and GROWTH worksheet functions).

In such cases, users need to consider whether they are applying these functions to a problem for which the resulting numerical errors are significant compared to their requirements on numerical accuracy.

The accuracy requirements for mathematical and trigonometric functions can be high, because such functions typically underpin and are fundamental to more complicated calculations. For example, the use of the trigonometric functions "sin" and "cos" are fundamental to the Fourier analysis of measurement data, and of "$\tan^{-1}$" is important in the analysis of complex-valued data where the conversion from Cartesian to polar representations may be required. The testing reported here indicates that these high accuracy requirements are generally met, although the accuracy of the inverse hyperbolic sine function depends on the sign of its argument.

The accuracy requirements for statistical distributions (and their inverses) tend to be more modest with only a few figures of accuracy typically needed. For example, for evaluation of confidence intervals in statistical modelling applications, which requires the evaluation of the inverse of the Student's-*t* distribution, the result may be required to be accurate to at most two

or three significant figures. The testing reported here indicates that these accuracy requirements are generally met, although the results suggest that there are critical values for which the numerical performance of the functions is poorest.

There are, however, circumstances where modest accuracy is acceptable at a superficial level but unacceptable when considered in a broader context. Some examples follow.

1. In a number of metrology applications it is necessary to compute the value of one or more integrals (e.g., areas under curves). Often, such integrals cannot be determined analytically, in which case quadrature routines are used instead. Some of the most popular quadrature routines are *adaptive*: successive estimates of the required integral are computed, each such estimate based on applying integration rules to values of the integrand determined at the previous stage together with those formed at the current stage. A particular feature of the more sophisticated adaptive-quadrature routines is that they attempt to account for possibly tortuous behaviour of the integrand, including singularities in the value and derivatives. The testing of some of the Microsoft Excel functions reported here would indicate that the functions as implemented contain value discontinuities at a number of values of (one or more of) their arguments, e.g., see the results for the BETADIST worksheet function. Thus, if an integrand contains one such function, these discontinuities are inherited to a greater or lesser extent by it.

   An adaptive-quadrature routine inevitably performs less efficiently in the presence of discontinuities; moreover, the results may not be as reliable as those produced for smooth integrands, simply because of the greater difficulty of the integration problem. Furthermore, a much larger number of evaluations of the integrand will generally be required, with the consequence that an in-built or user-specified accuracy requirement may not be met.

2. In many branches of metrology, approximations to derivatives are required. One of the commonest circumstances is in estimating the sensitivity coefficients needed in the evaluation of uncertainties according to the ISO GUM [11]. Typically, the first derivative of a function f at x is estimated by the finite-difference formula

$$\frac{f(x+h)-f(x)}{h}$$

   or

$$\frac{f(x+h)-f(x-h)}{2h},$$

   where h is a small positive increment. The magnitude of h is generally chosen to balance the effects of cancellation error in forming, e.g., $f(x + h) - f(x)$ and truncation error (the finite-difference formulae constitute only the leading terms in a Taylor series expansion of $f'(x)$).

   If f is, or is based on, one of the tested Microsoft Excel functions which delivers only modest accuracy, clearly the above derivative approximations will deliver less accuracy. For instance, if a Microsoft Excel function was correct to six significant figures, the estimate derivative would be accurate to about three figures. The loss would be exacerbated in the neighbourhood of critical points.

3. The need to provide optimal solutions is commonplace across metrology, in such areas as fitting physical and empirical models to measurement data, parameter identification, experimental design, and in the design of instruments. The functions to be optimised and any constraints on the optimisation variables are usually expressed as mathematical functions. If these functions make use of the Microsoft Excel in-built functions which return only modest accuracy, the performance of optimisation software can be compromised. In particular, convergence to the solution can be strongly affected, as indicated by the following one-dimensional example.

Consider the minimisation of the function $f(x)$. In the neighbourhood of the minimum $x = x_0$, f, assuming it to be sufficiently smooth, can be expressed as

$$f(x) = f(x_0) + \frac{1}{2}(x - x_0)^2 f''(x_0), \qquad (17)$$

plus higher-order terms, since $f'(x_0) = 0$ at the minimum. Suppose that $f(x)$ is evaluated using the in-built functions to an absolute accuracy of order ε. Suppose also that the scale of x has been normalised such that $f''(x_0)$ is of order unity. Then, since for x near x_0, the difference $f(x) - f(x_0)$ is of order ε, it follows from (17) that $x - x_0$ is of order $\varepsilon^{1/2}$. Hence, in this case, the number of digits that are correct in determining the minimum x_0 is only half that in the function f.

The above examples illustrate the fact that even if the in-built functions provide sufficient accuracy to be used as "stand-alone" calculations, the situation may well be different if they form part of a larger computation. Therefore, it is necessary to view carefully the greater context in which the in-built functions are to be utilised. Thus the extreme importance of ensuring the correctness of the in-built functions, to a greater extent than a superficial assessment would indicate, is clear. In particular it is evident that sound numerical analysis is necessary to help ensure fitness for purpose of applications that utilise in-built functions.

7. Acknowledgements

This report constitutes one of the deliverables of Project 2.1 of the 1998–2001 NMS Software Support for Metrology Programme, and has been funded by the National Measurement System Policy Unit of the UK Department of Trade and Industry.

8. References

[1] D Rayner. *Initial report on status of software and mathematics in each metrology area*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 18/99, February 1999.

[2] M G Cox, M P Dainton, P M Harris and B A Wichmann. *Survey report on testing spreadsheets and other packages used in metrology*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 22/99, March 1999.

[3] Knusel, L. On the accuracy of statistical distributions in Microsoft Excel 97. *Computational Statistics and Data Analysis*, **26**(3), pp 375-7, 1998.

[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. *A methodology for testing spreadsheets and other packages used in metrology*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 25/99, September 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*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 26/99, September 1999.

[9] Microsoft Excel for Windows 95, Version 7.0a. Copyright© 1985-1996 Microsoft Corporation.

[10] IMSL Fortran 90 Math/Library (version 3.0) provided with Digital Visual Fortran (version 5.0.D, Professional Edition, Digital Equipment Corporation.

[11] *Guide to the Expression of Uncertainty in Measurement*, International Organisation for Standardisation, 1993, ISBN 92-67-10188-9.

Appendix A: Results for Regression Functions

AVERAGE

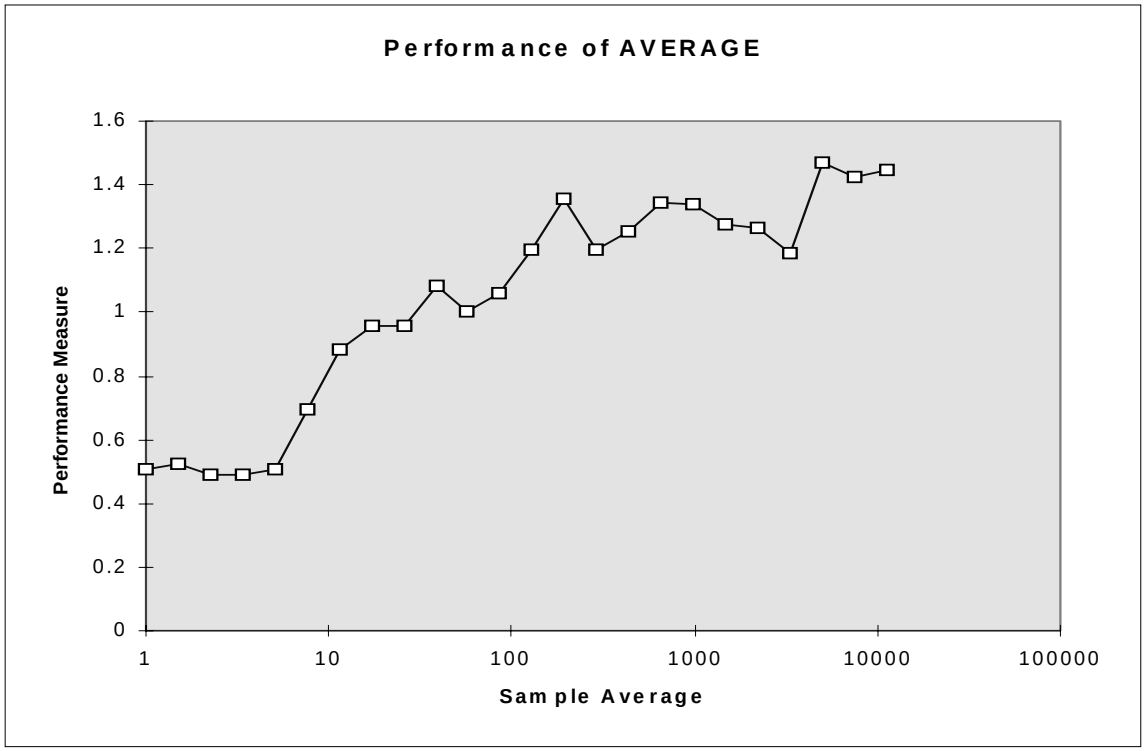

Figure 1 Plot of the performance metric $P(\bar{x})$ for the sample mean $\bar{x}$ against the performance measure $\bar{x}$.

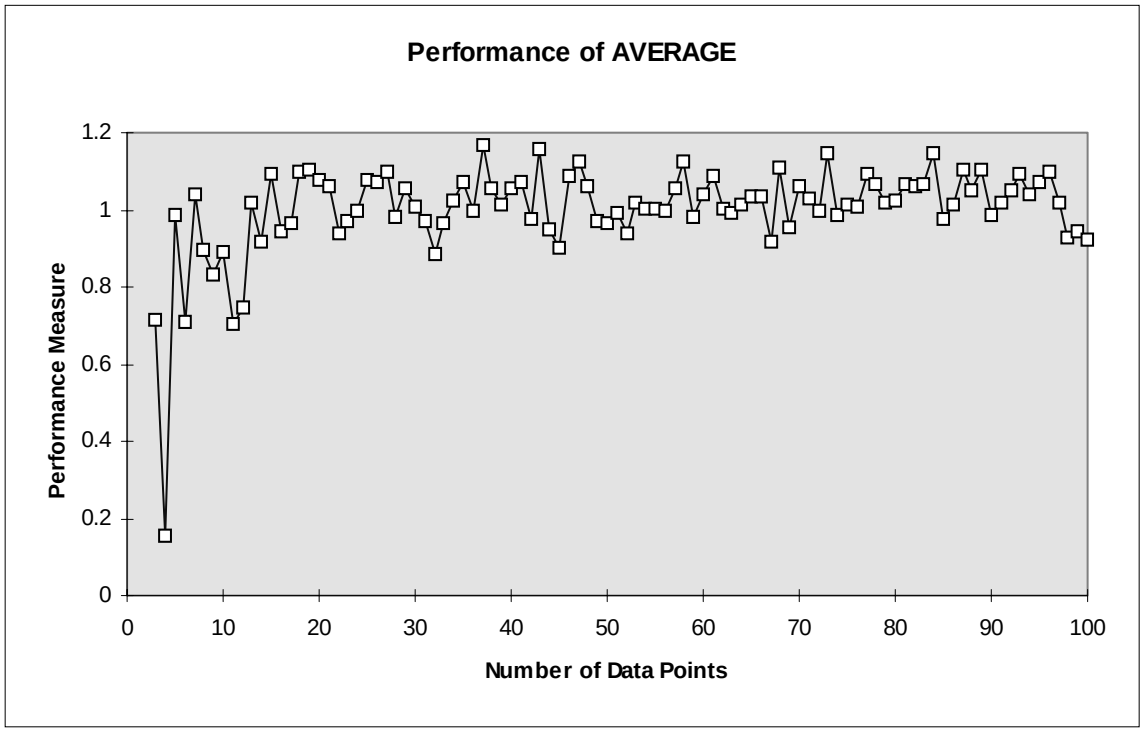

Figure 2 Plot of the performance metric $P(\bar{x})$ for the sample mean $\bar{x}$ against the number of data points.

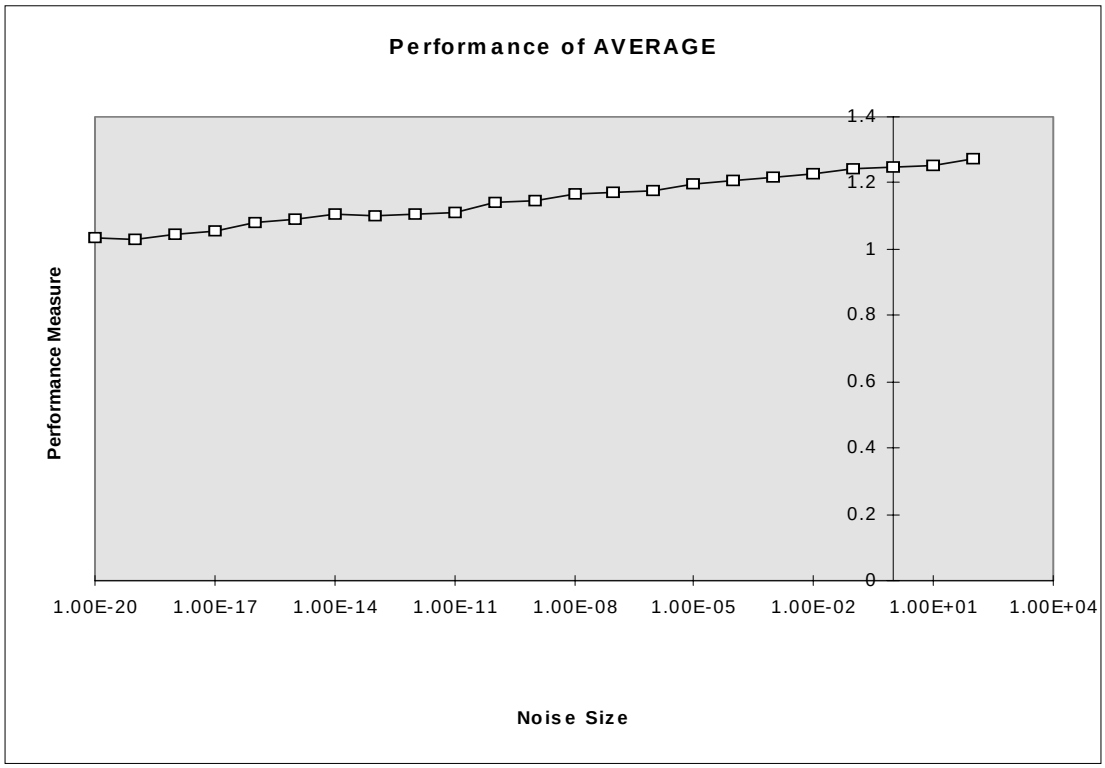

Figure 3: Plot of the performance metric $P(\bar{x})$ for the sample mean $\bar{x}$ against the noise size.

STDDEV

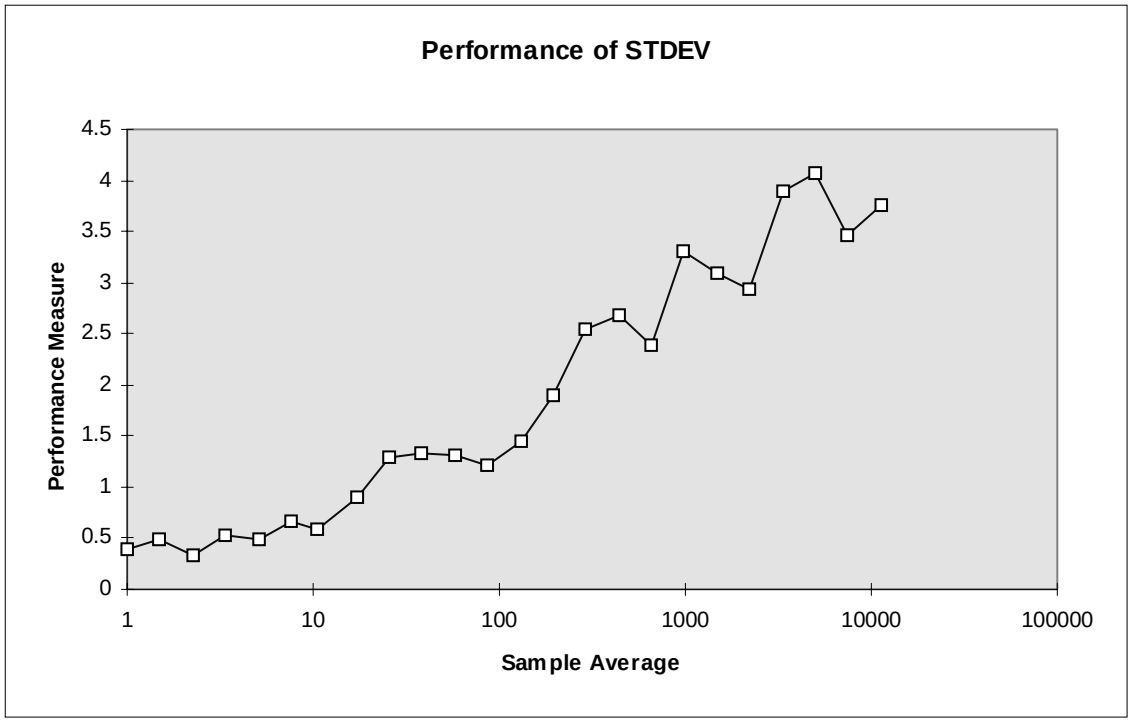

Figure 4: Plot of the performance metric $P(s)$ for the sample standard deviation s against the performance measure $\bar{x}$.

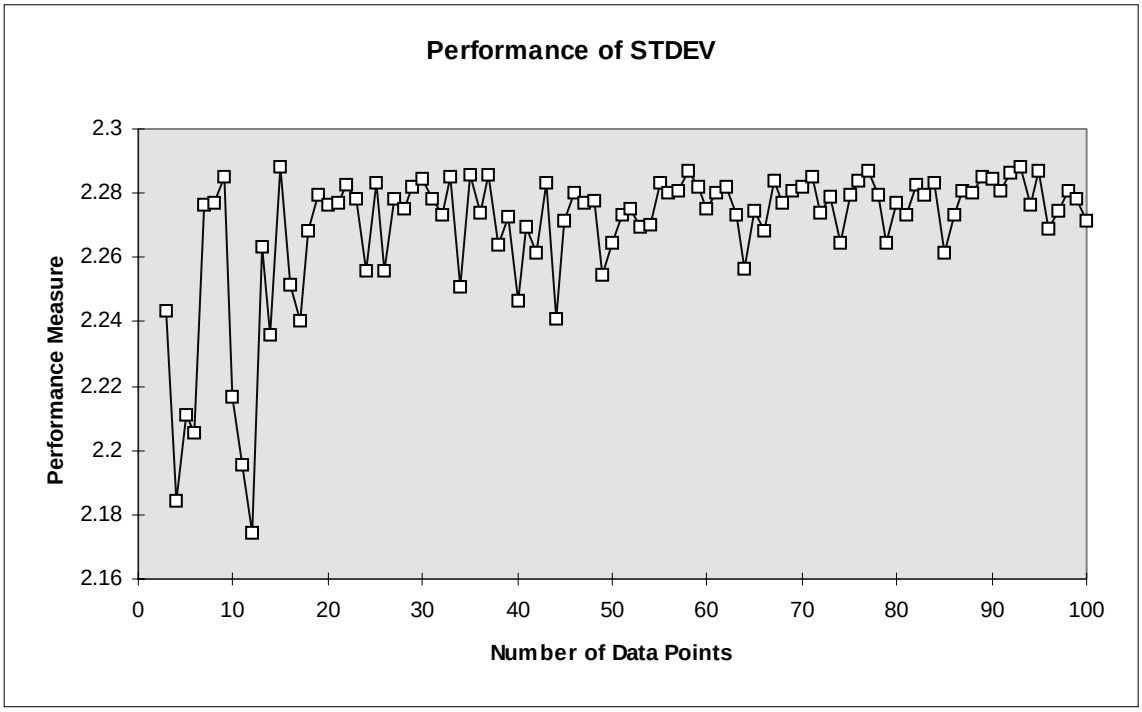

Figure 5: Plot of the performance metric $P(s)$ for the sample standard deviation s against the number of data points.

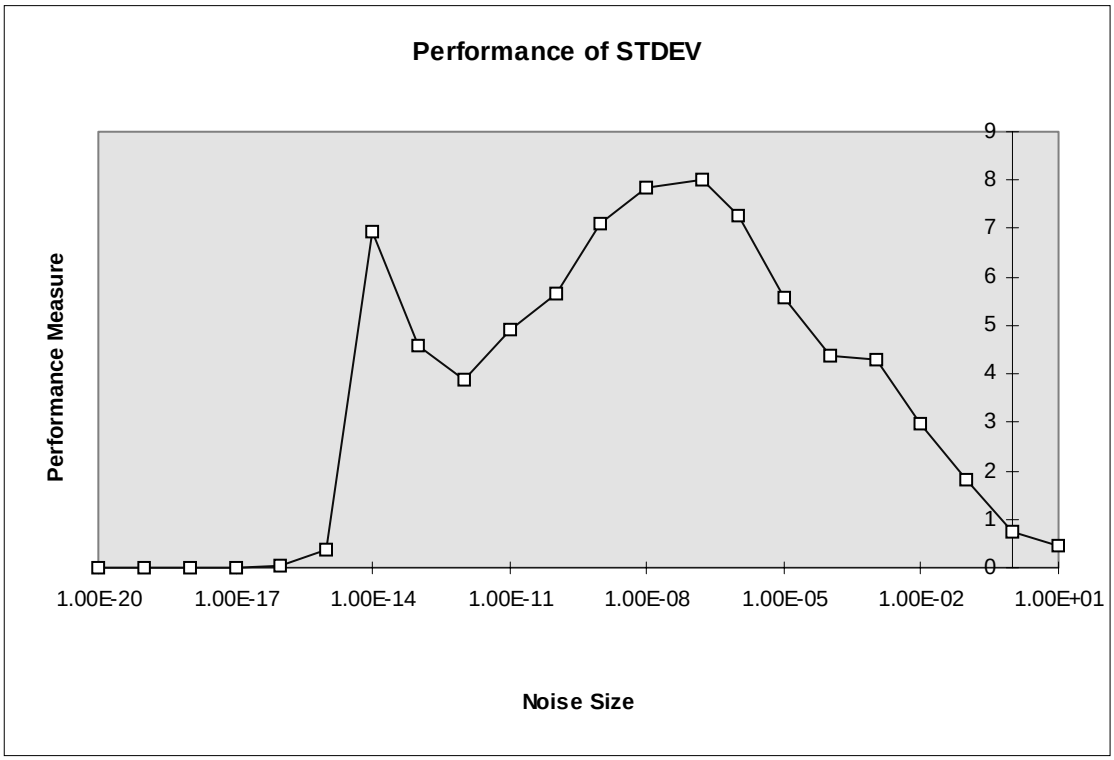

Figure 6: Plot of the performance metric $P(s)$ for the sample standard deviation s against the reference value.

LINEST

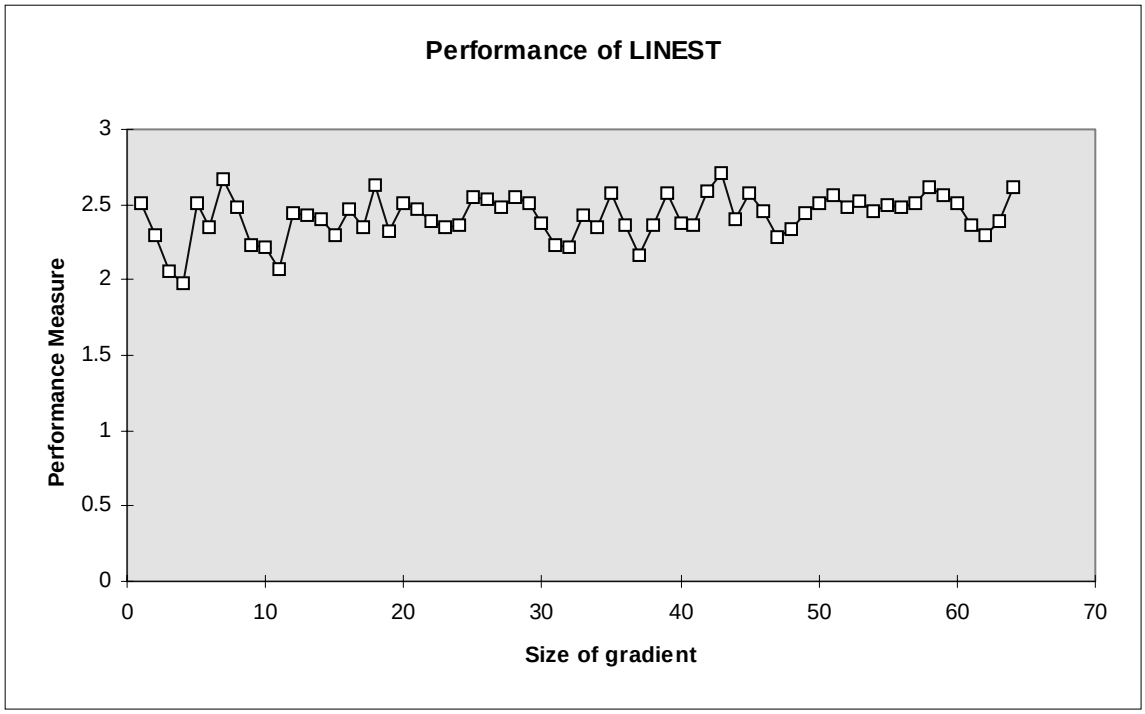

Figure 7 Plot of the performance measure $P(m)$ for the gradient m against the performance parameter m.

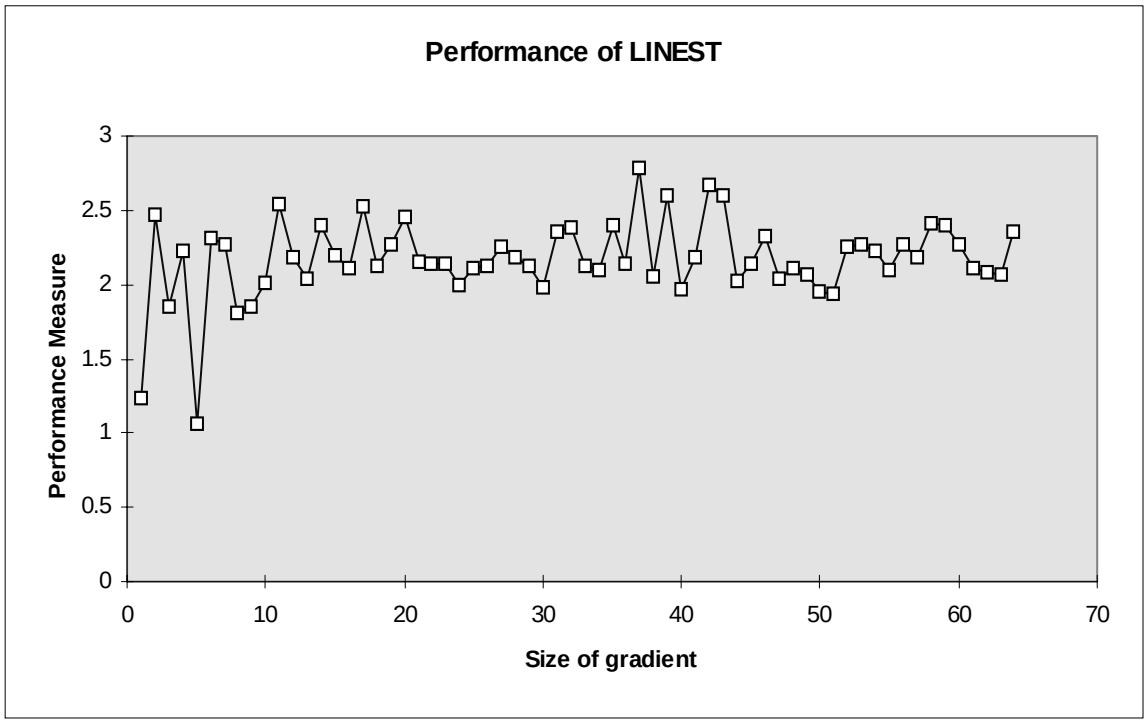

Figure 8 Plot of the performance measure $P(b)$ for the intercept b against the performance parameter m.

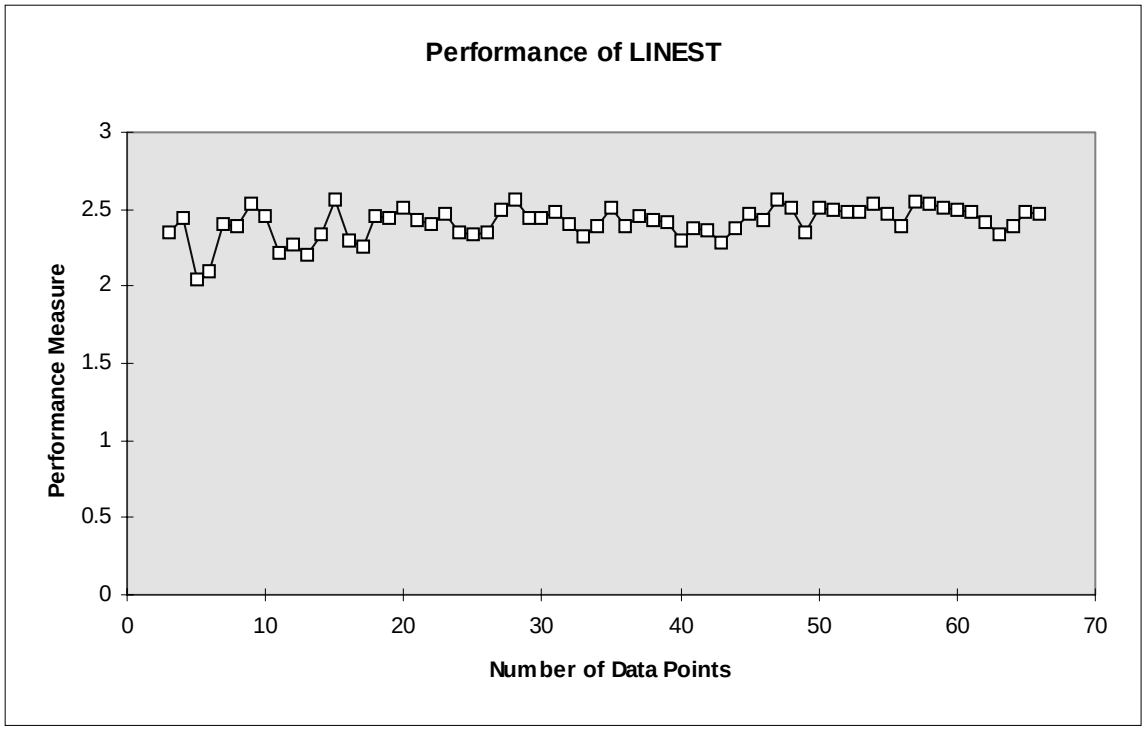

Figure 9 Plot of the performance measure $P(m)$ for the gradient m against the number of data points.

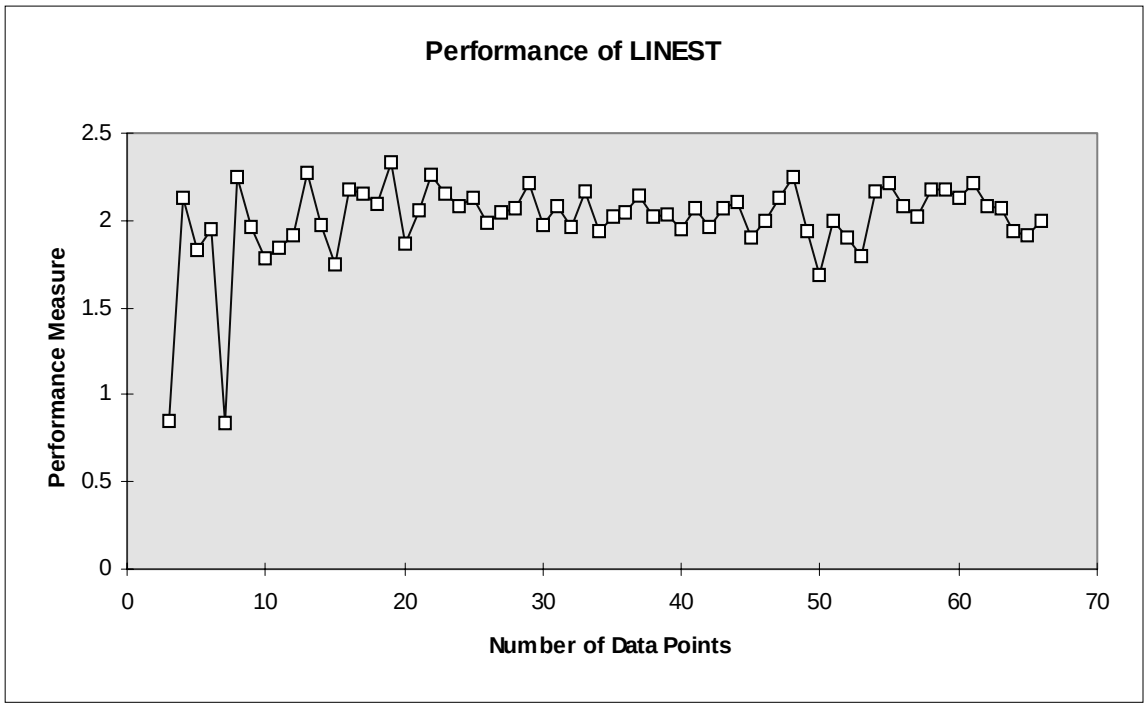

Figure 10 Plot of the performance measure $P(b)$ for the intercept b against the number of data points.

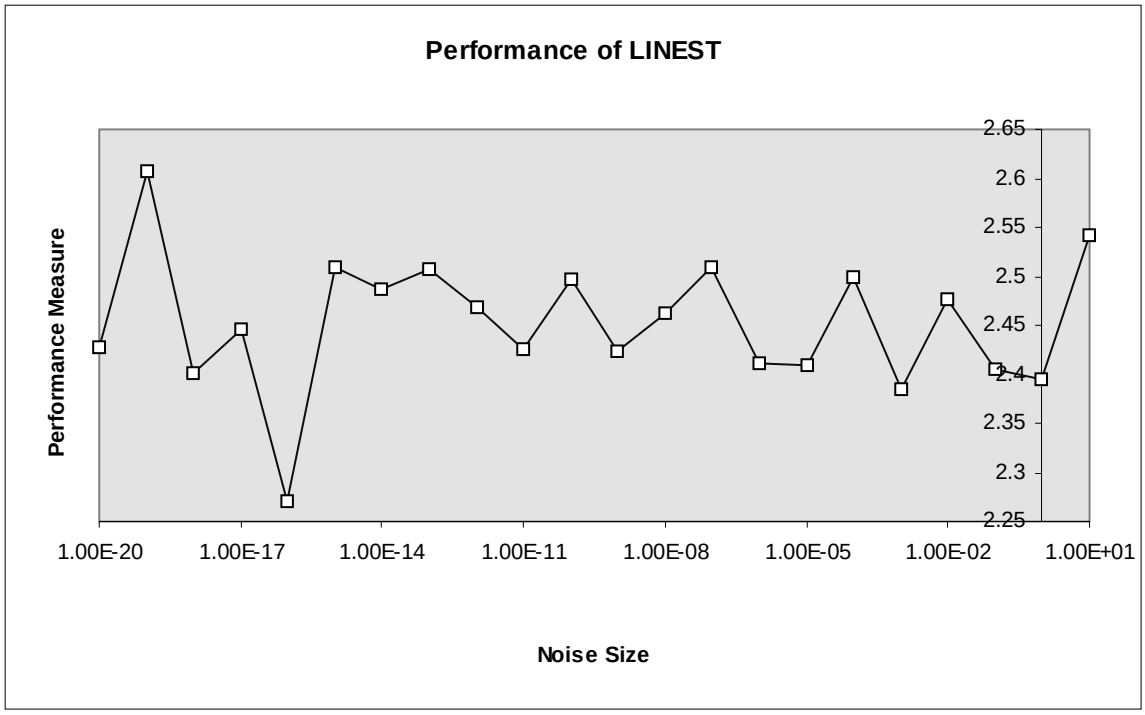

Figure 11 Plot of the performance measure $P(m)$ for the gradient m against the amount of measurement noise.

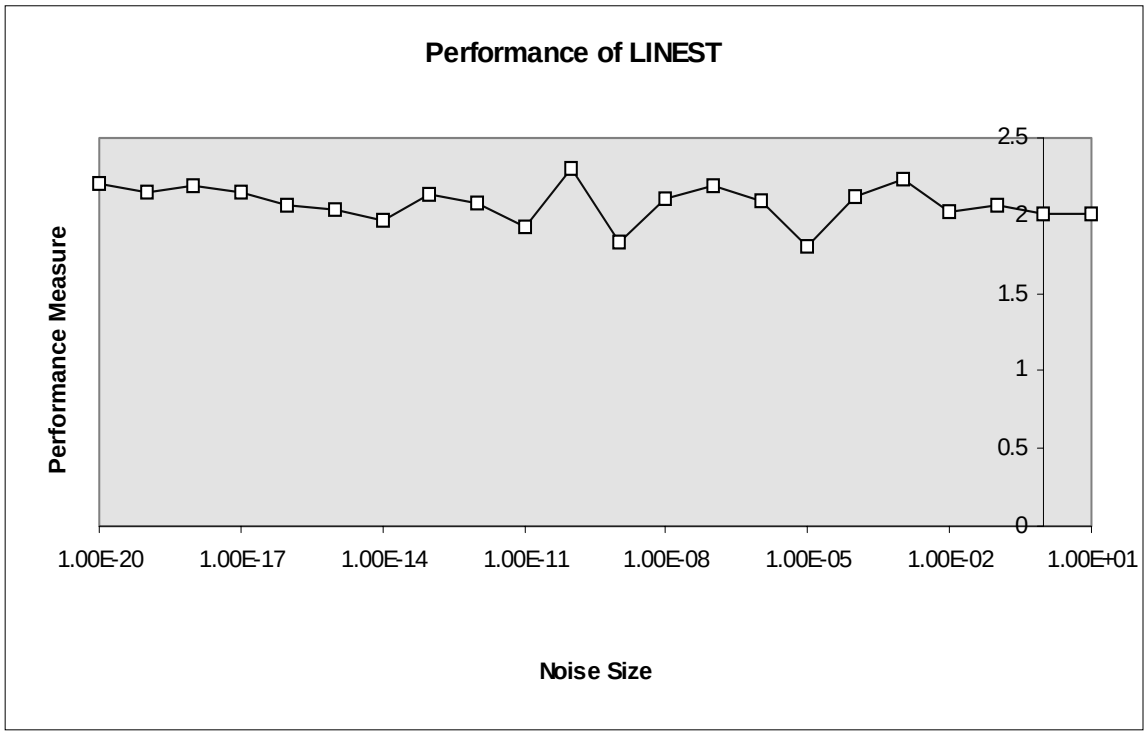

Figure 12 Plot of the performance measure $P(b)$ for the intercept b against the amount of measurement noise.

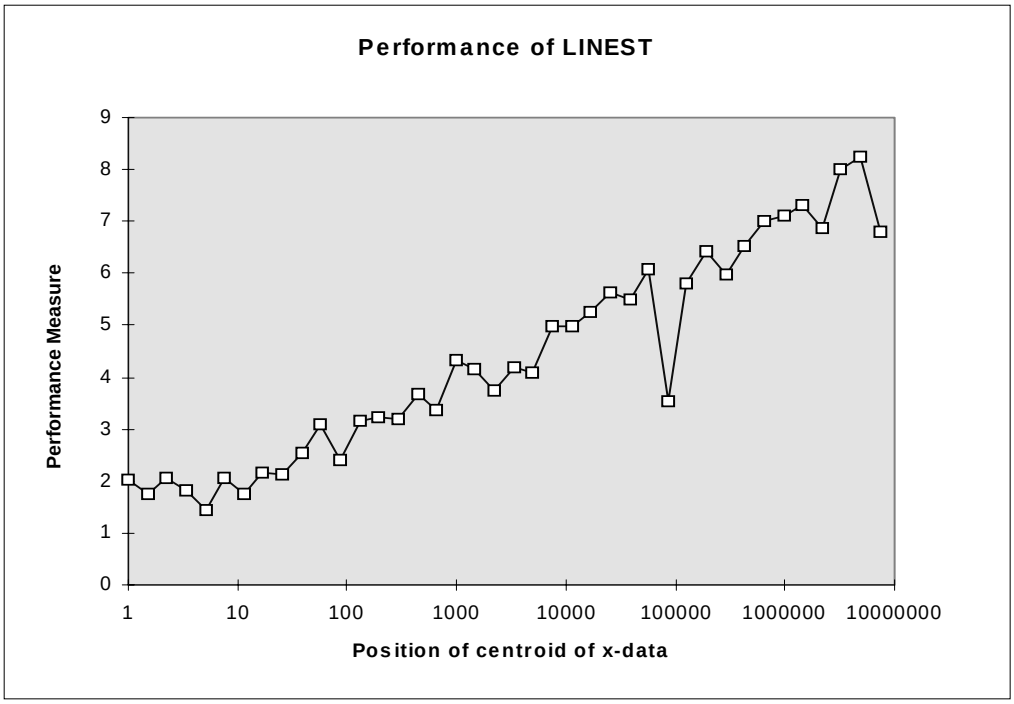

Figure 13 Plot of the performance measure $P(m)$ for the gradient m against the location of the data x-values.

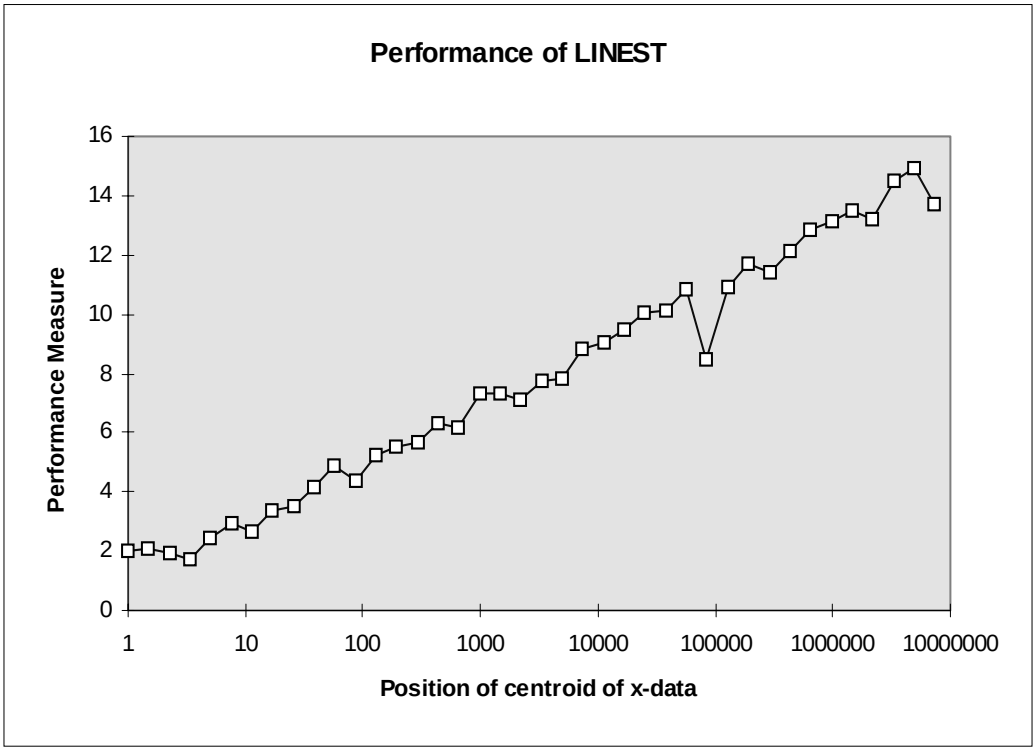

Figure 14 Plot of the performance measure $P(b)$ for the intercept b against the location of the data x-values.

LOGEST

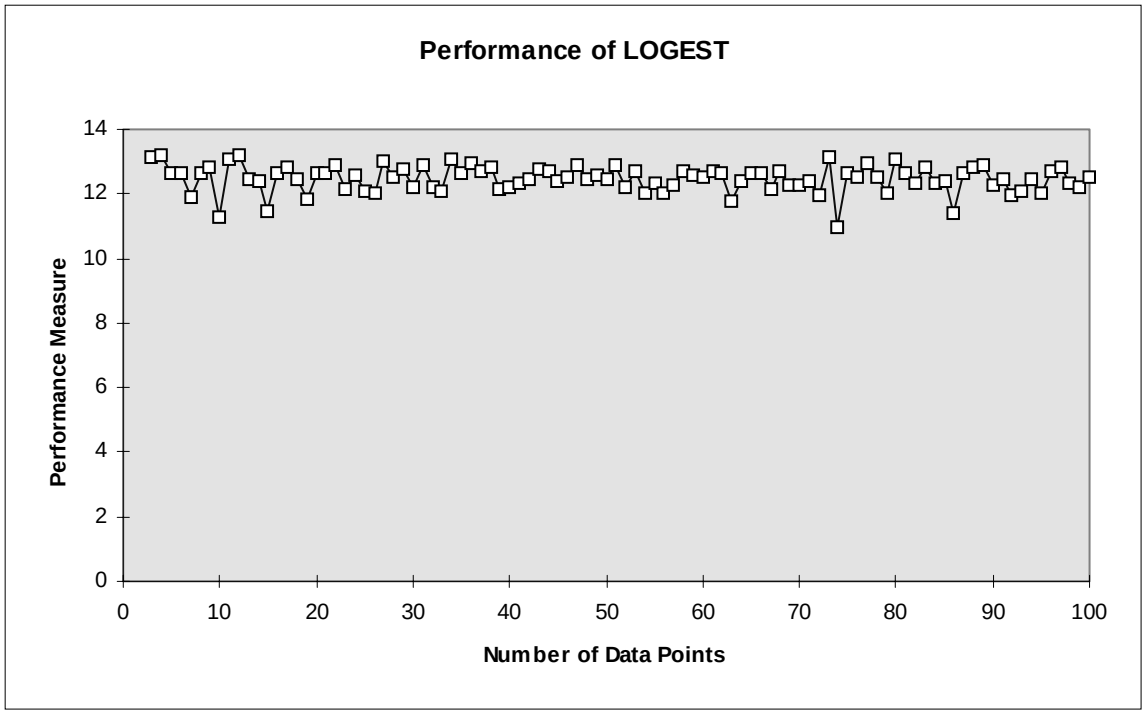

Figure 15 Plot of the performance measure $P(m)$ against the number of data points.

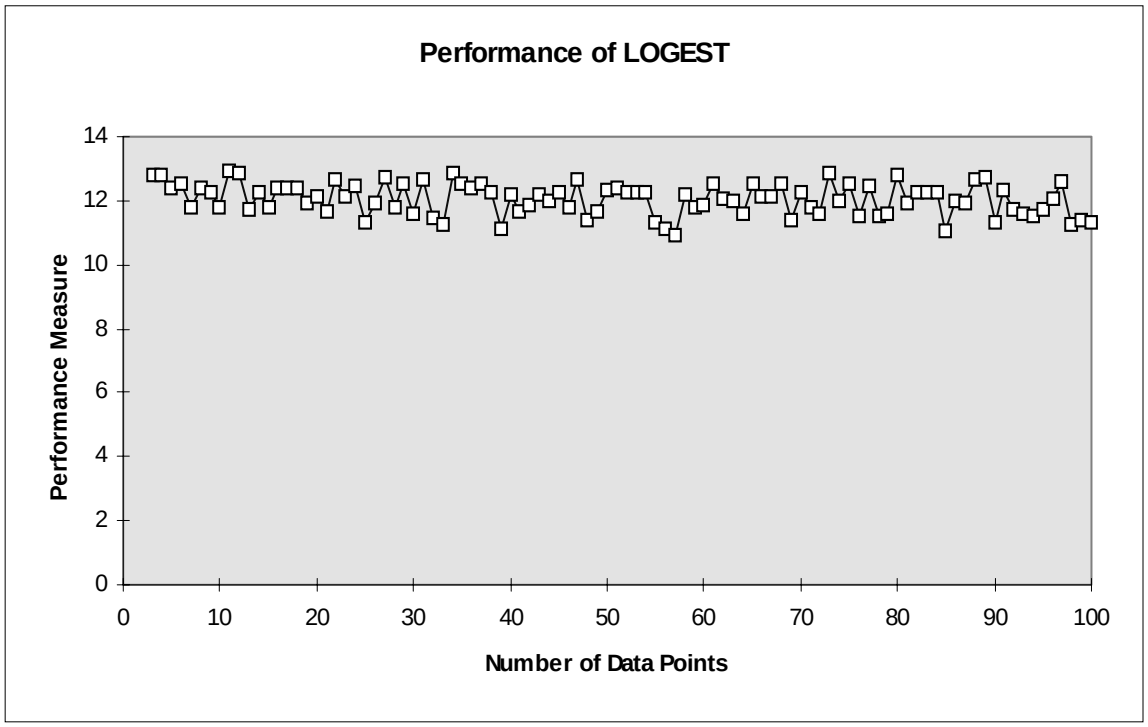

Figure 16 Plot of the performance measure $P(b)$ against the number of data points.

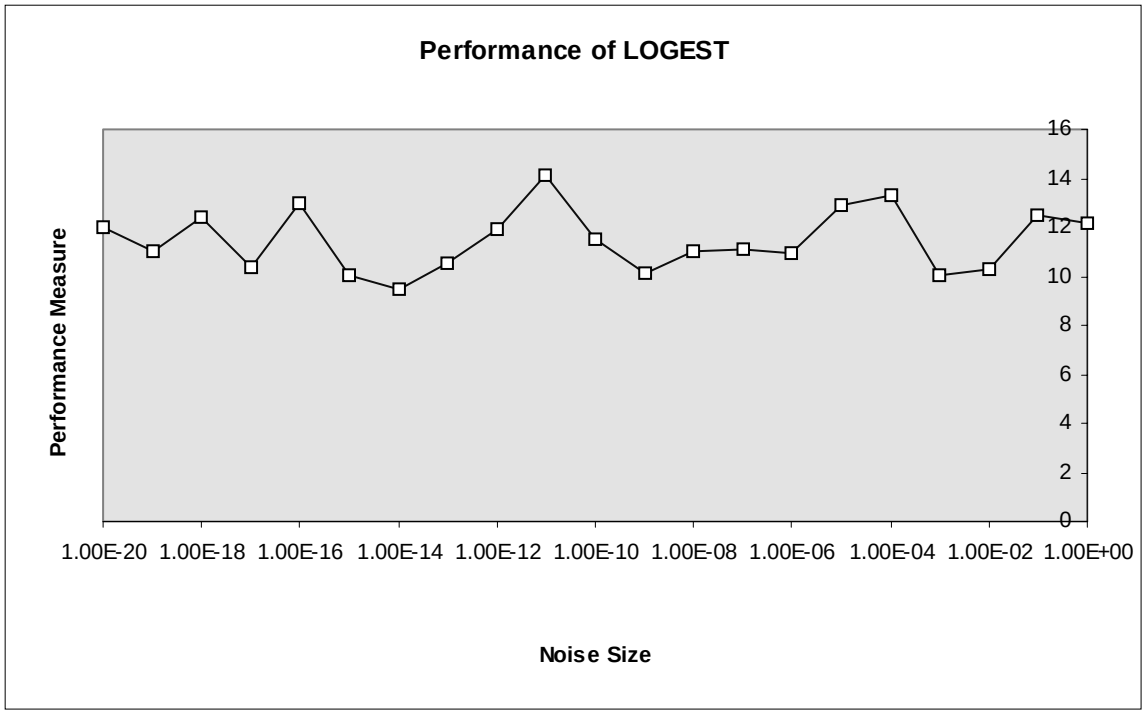

Figure 17 Plot of the performance measure $P(m)$ against the amount of measurement noise.

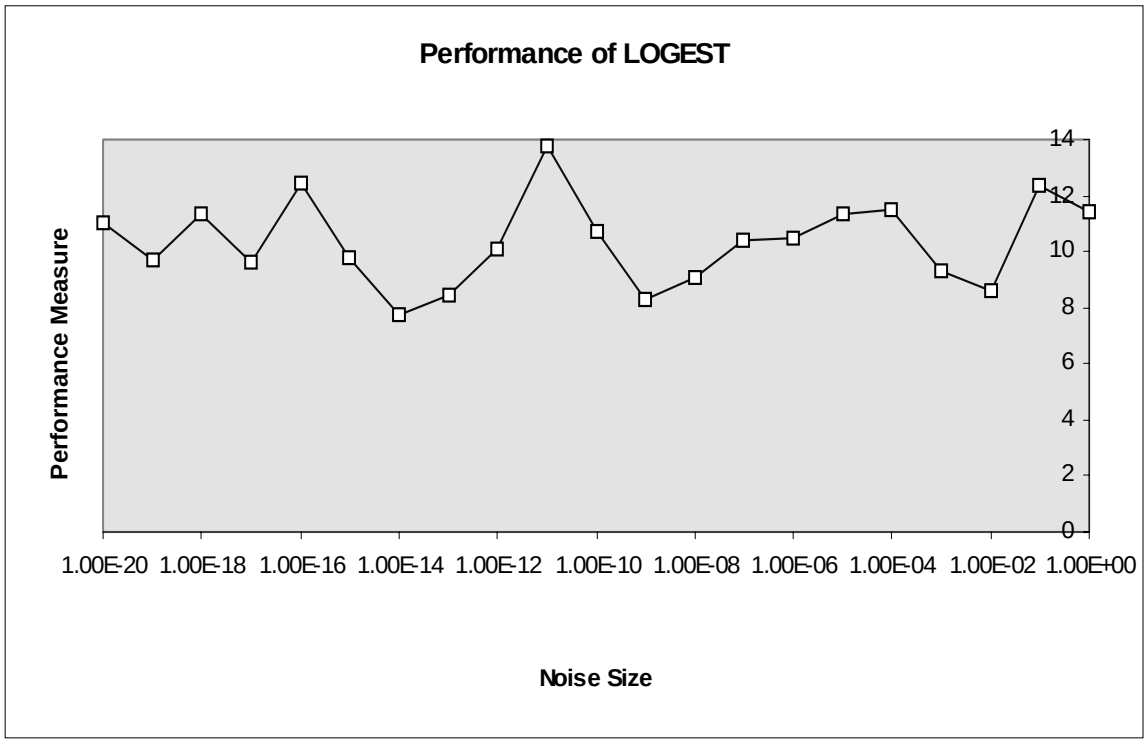

Figure 18 Plot of the performance measure $P(b)$ against the amount of measurement noise.

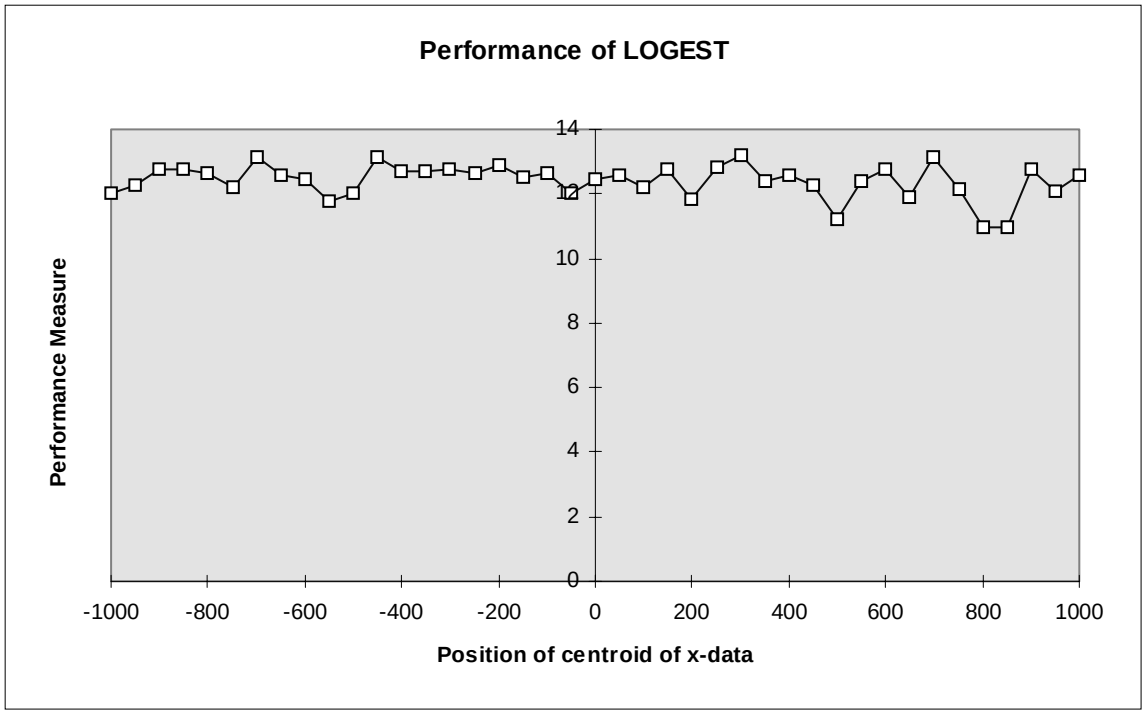

Figure 19 Plot of the performance measure $P(m)$ against the location of the data x-values.

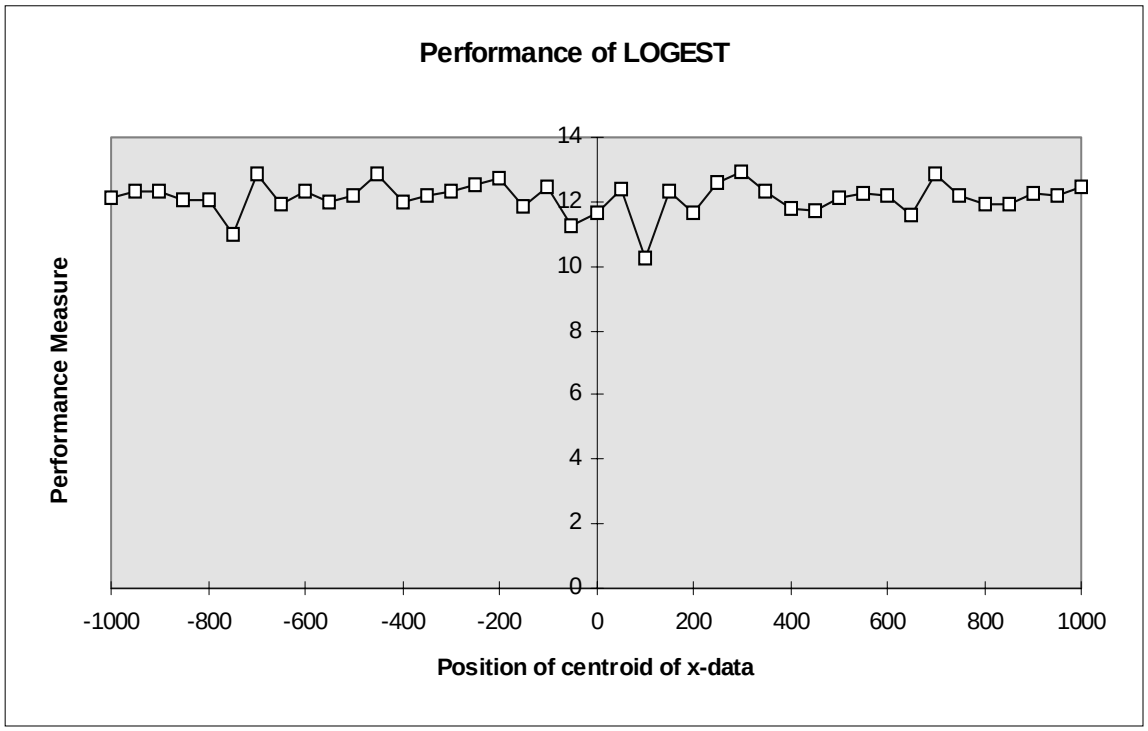

Figure 20 Plot of the performance measure $P(b)$ against the location of the data x-values.

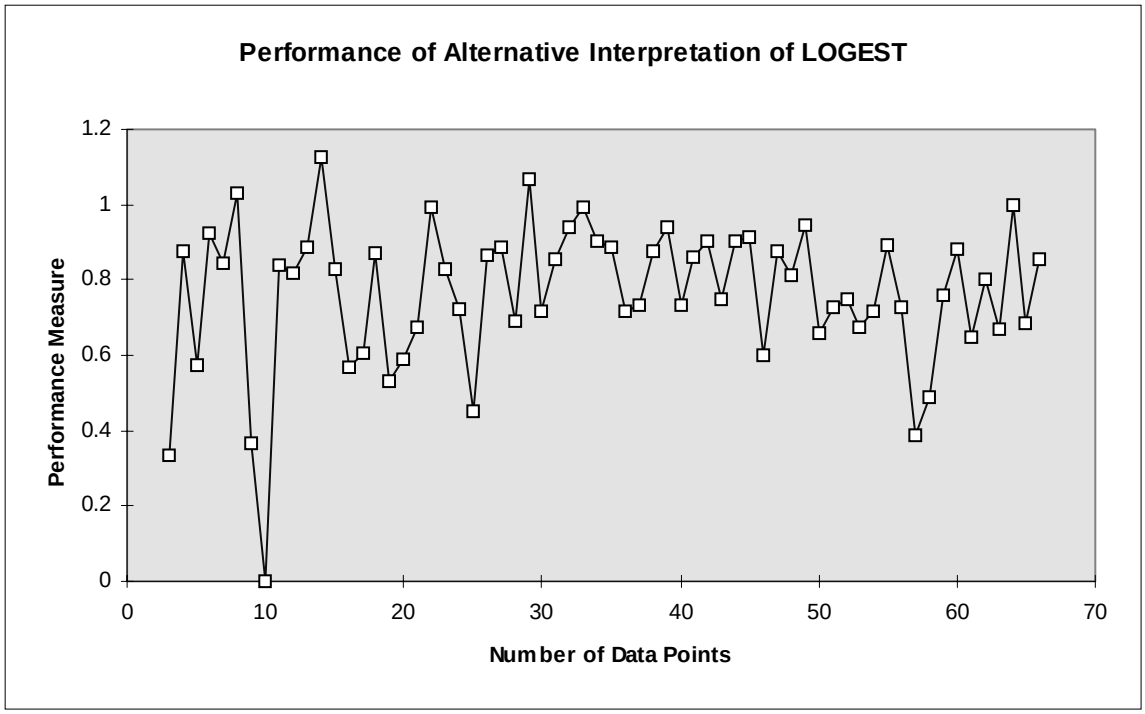

Figure 21 Plot of the performance measure $P(m)$ against the number of data points for the alternative test of LOGEST.

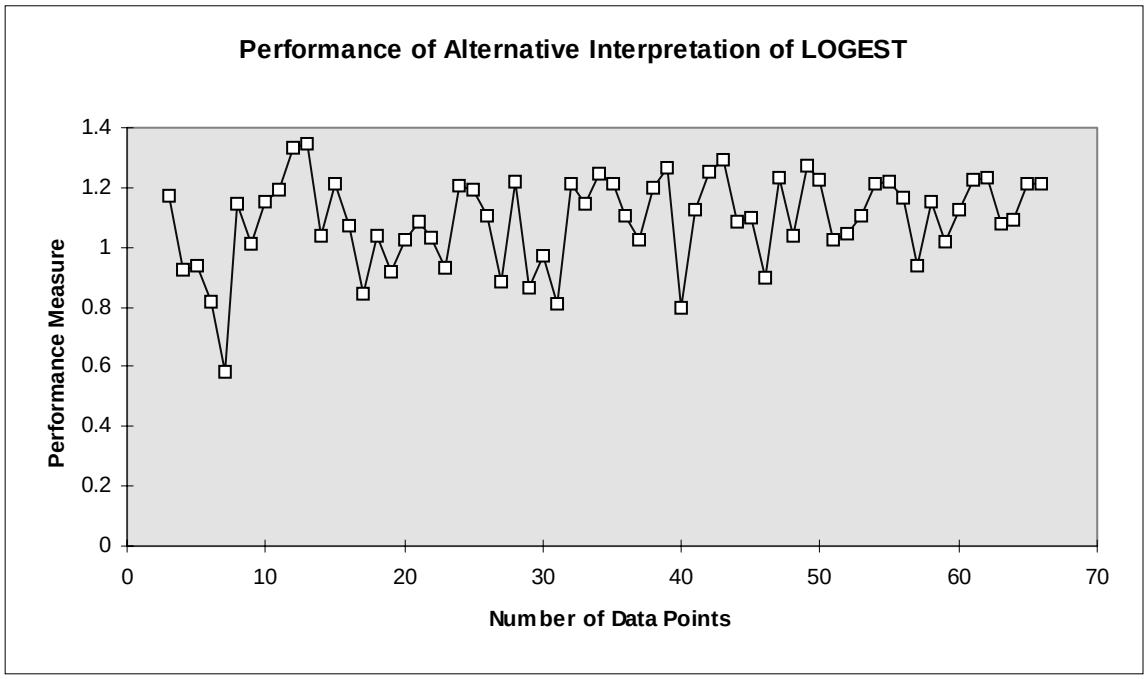

Figure 22 Plot of the performance measure $P(b)$ against the number of data points for the alternative test of LOGEST.

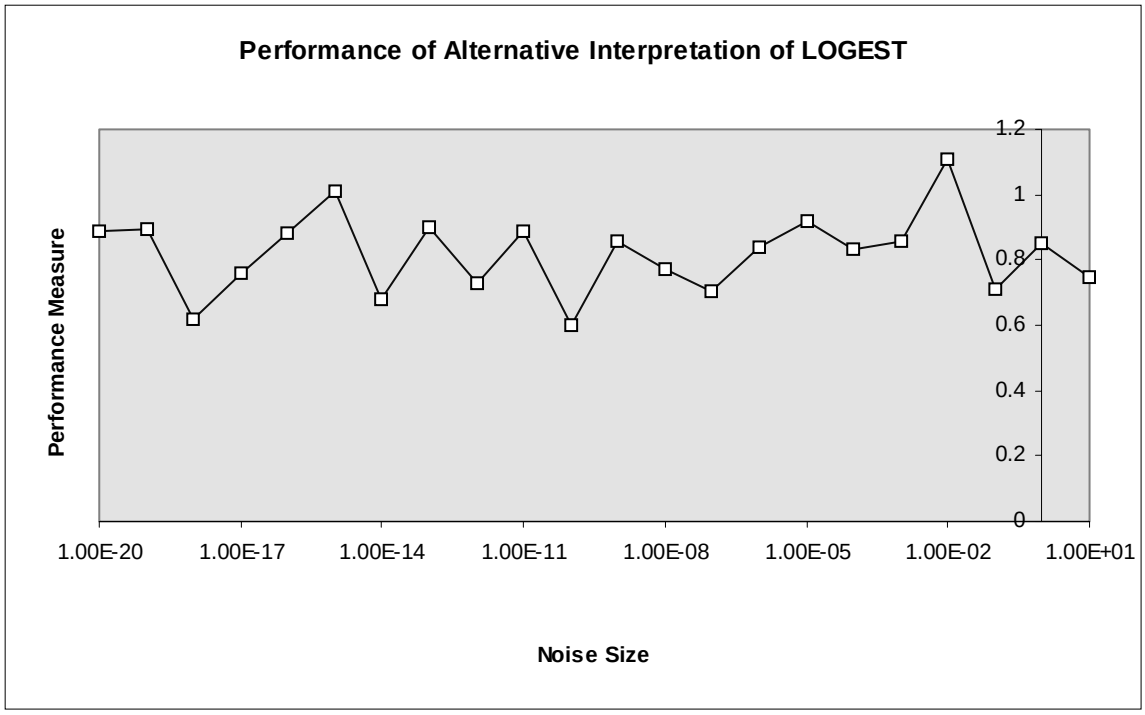

Figure 23 Plot of the performance measure $P(m)$ against the amount of measurement noise for the alternative test of LOGEST.

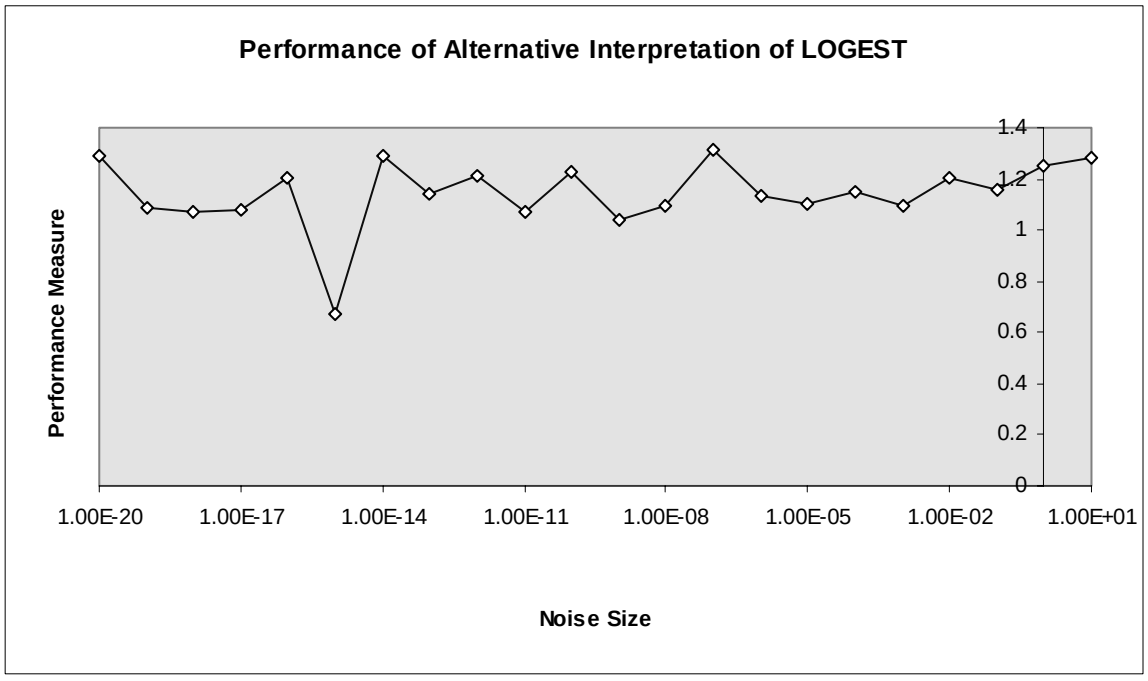

Figure 24 Plot of the performance measure $P(b)$ against the amount of measurement noise for the alternative test of LOGEST.

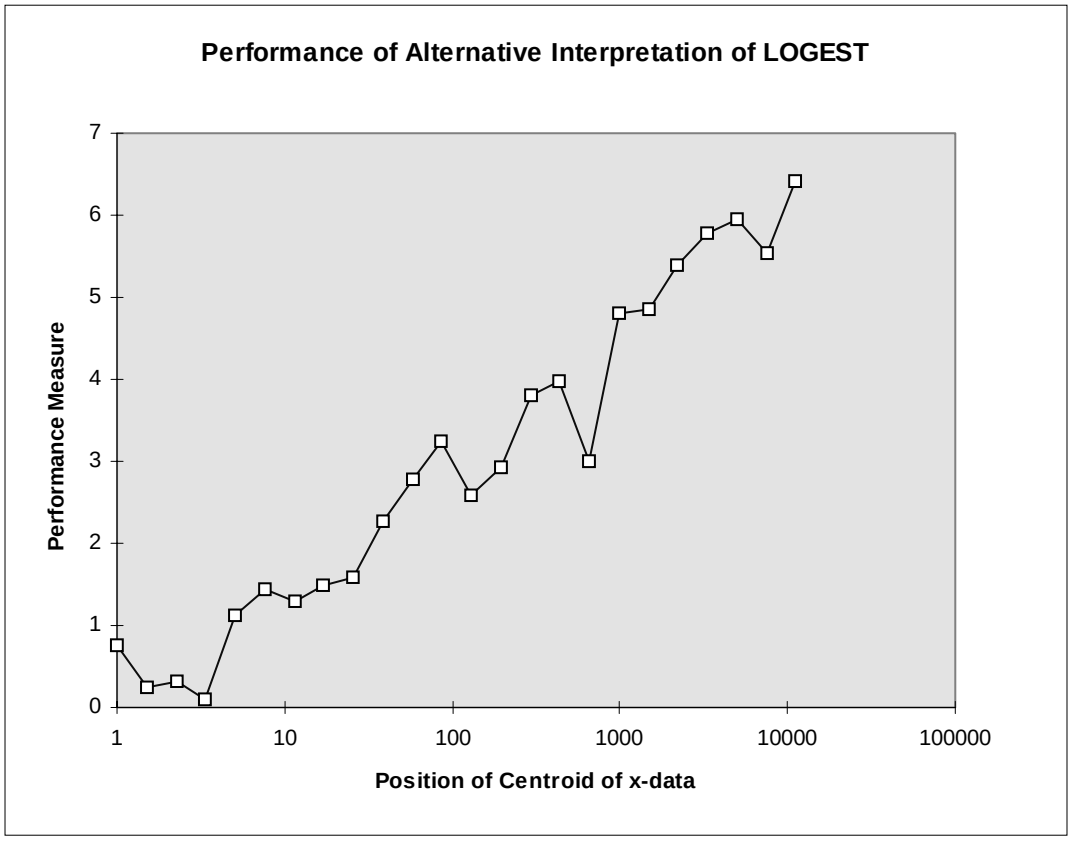

Figure 25 Plot of the performance measure $P(m)$ against the location of the data x-values for the alternative test of LOGEST.

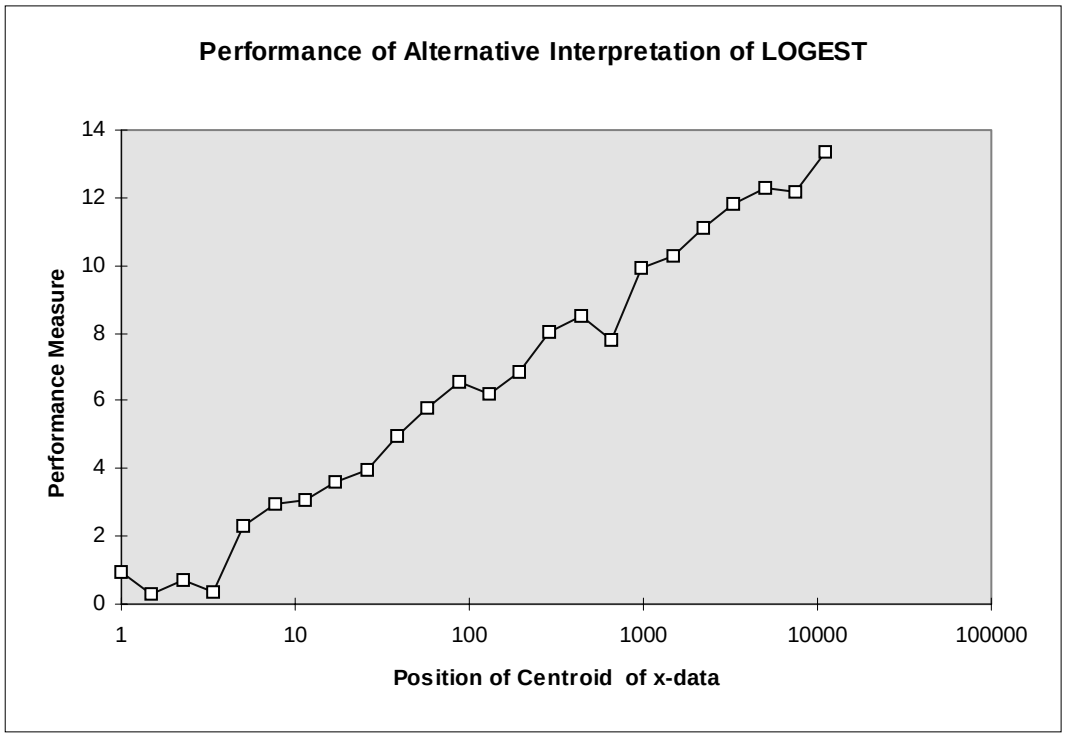

Figure 26 Plot of the performance measure $P(b)$ against the location of the data x-values for the alternative test of LOGEST.

TREND

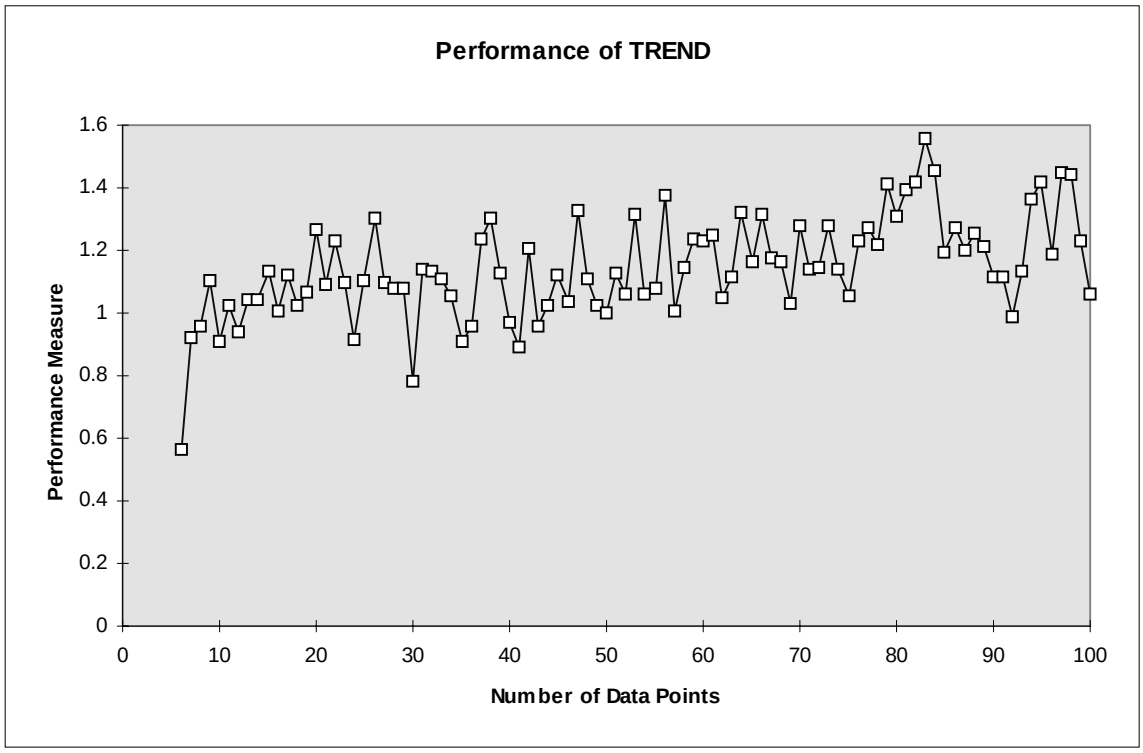

Figure 27 Plot of the quality metric *P*(**e**) for the residuals **e** against the number of data points.

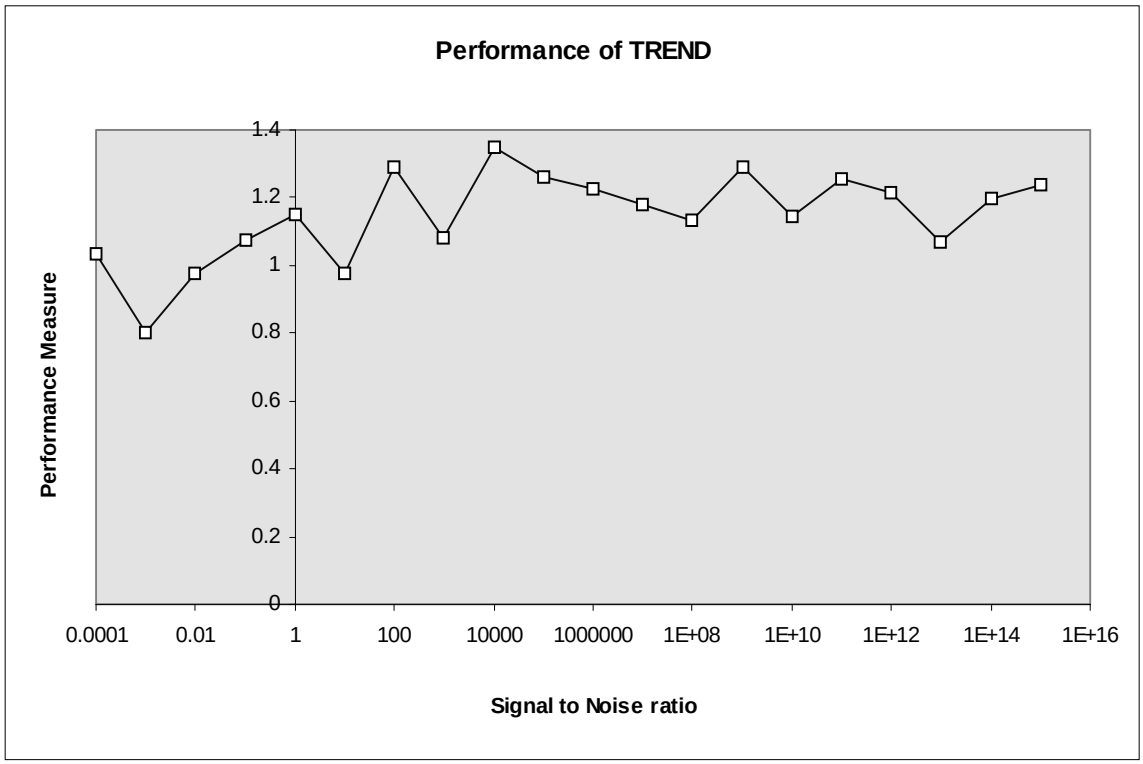

Figure 28 Plot of the quality metric *P*(**e**) for the residuals **e** against the signal to noise ratio.

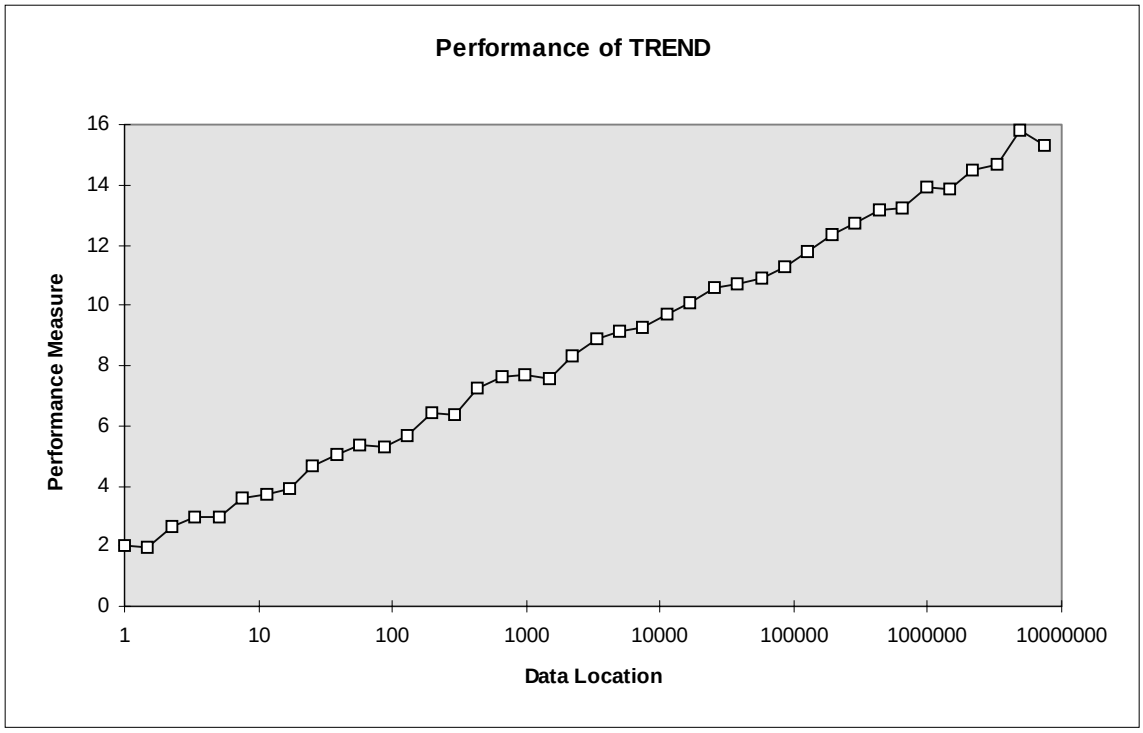

Figure 29 Plot of the quality metric $P(\mathbf{e})$ for the residuals **e** against the location of the data x-values.

GROWTH

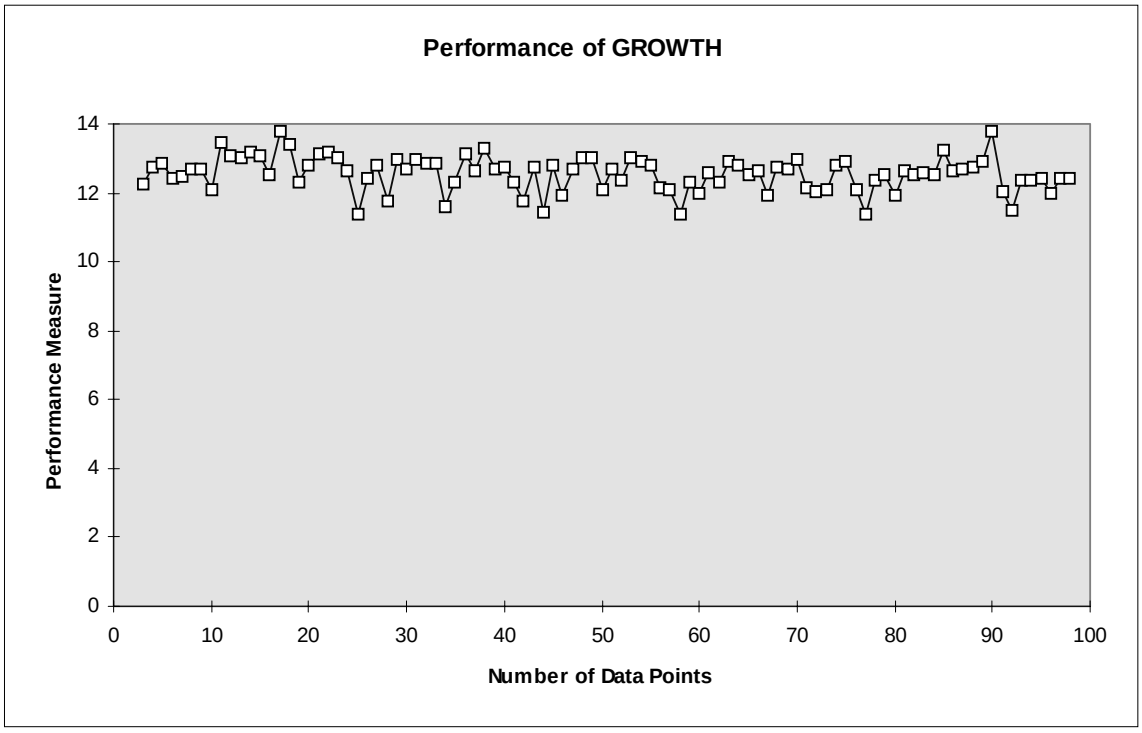

Figure 30 Plot of the quality metric $P(\mathbf{e})$ for the residuals **e** against the number of data points.

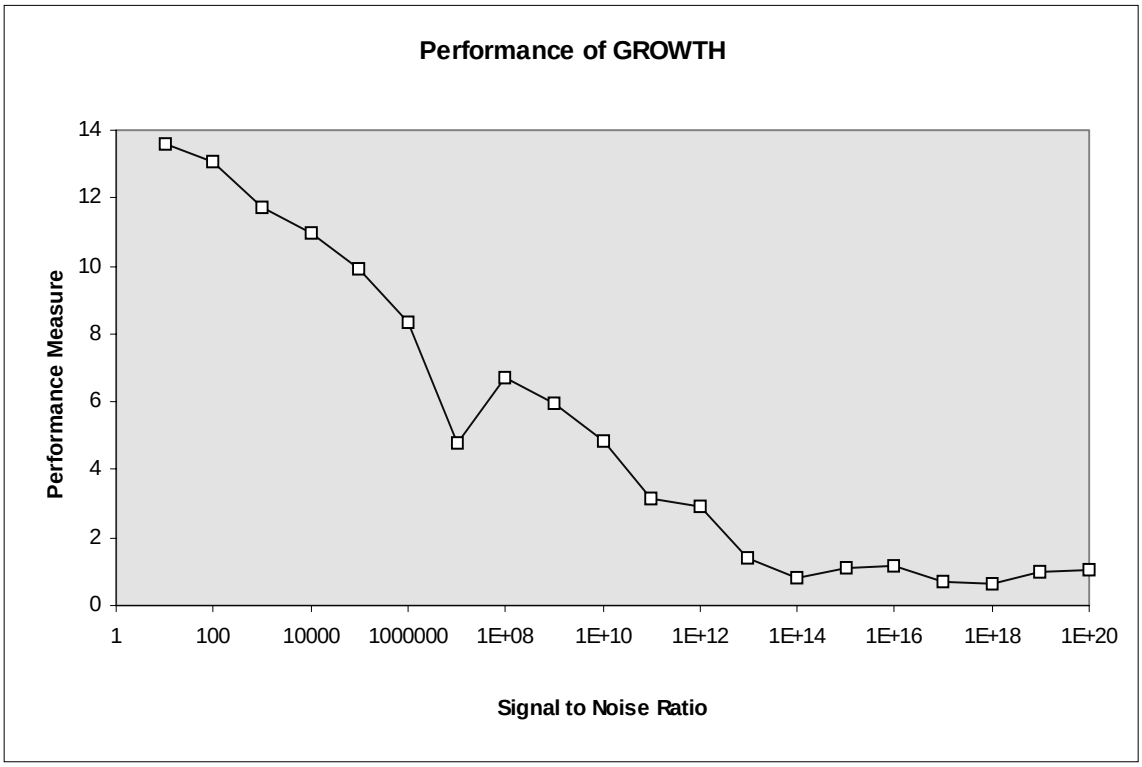

Figure 31 Plot of the quality metric $P(\mathbf{e})$ for the residuals **e** against the signal to noise ratio.

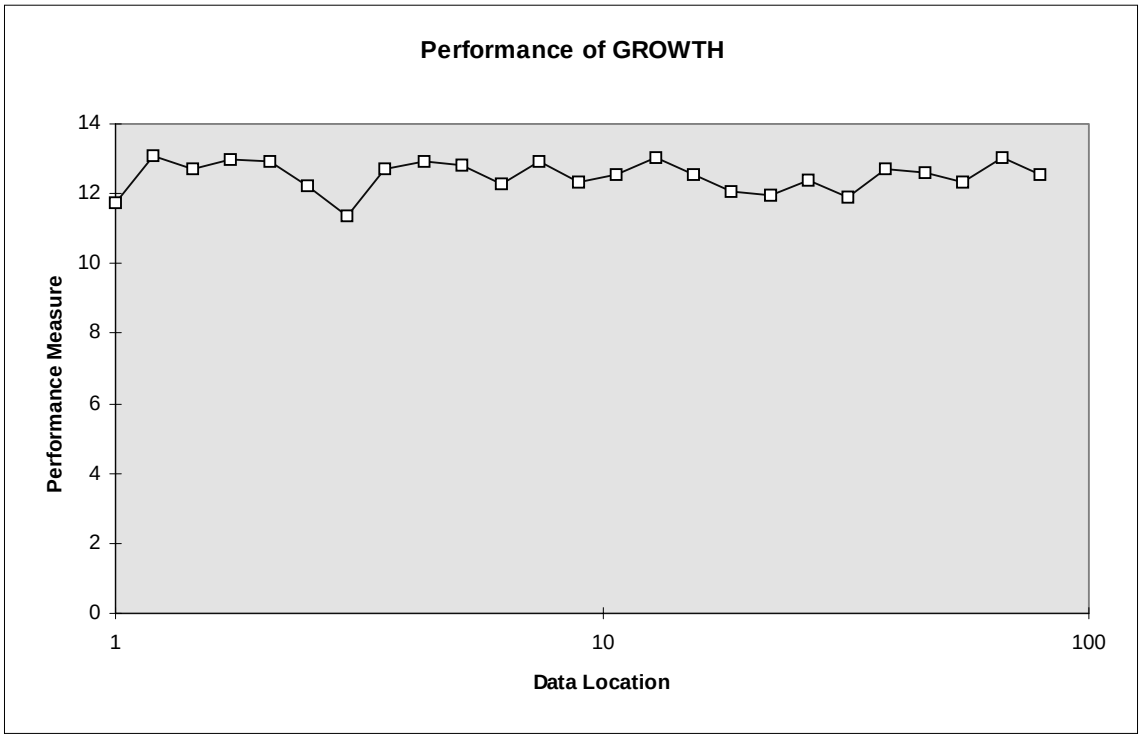

Figure 32 Plot of the quality metric $P(\mathbf{e})$ for the residuals **e** against the location of the data x-values.

Appendix B: Results for Mathematical and Trigonometric Functions

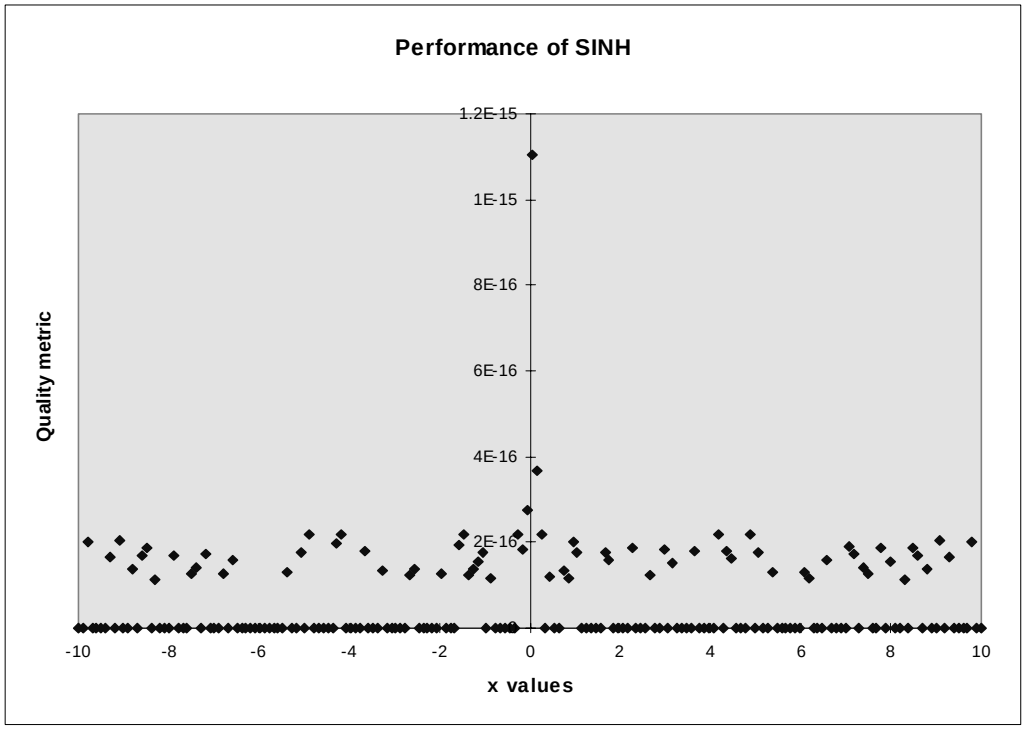

Figure 33 Graph of the quality metric $d_R(x)$ for the comparison of the SINH worksheet function and the equivalent Fortran intrinsic function.

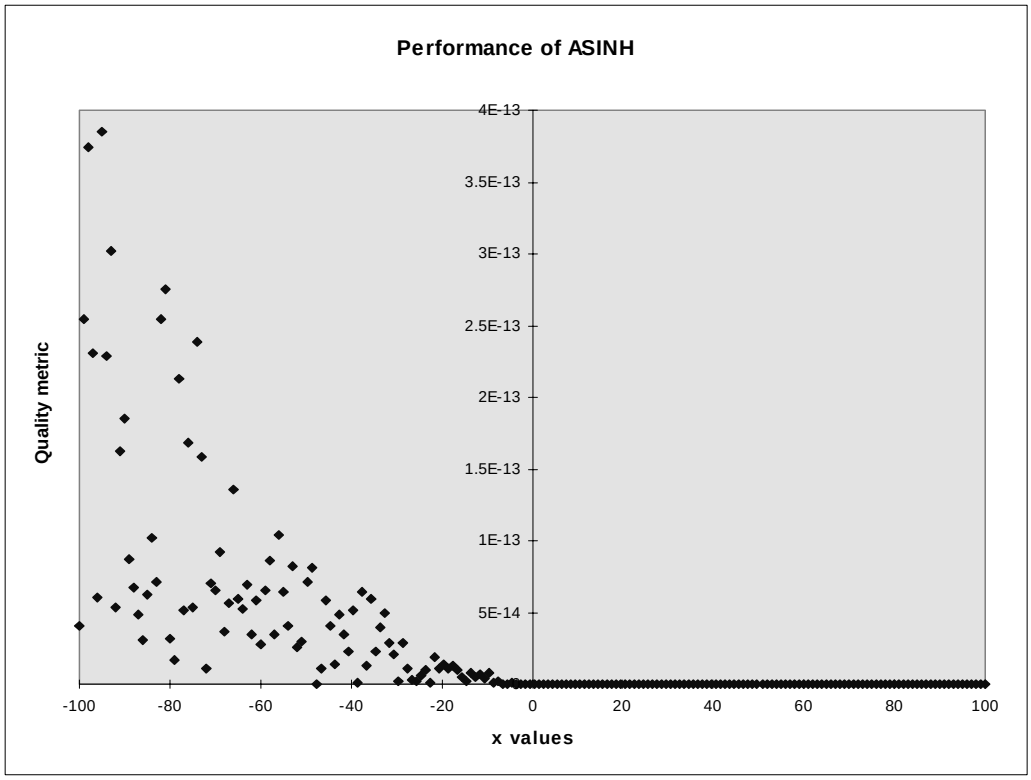

Figure 34 Graph of the quality metric $d_R(x)$ for the comparison of the ASINH worksheet function and the equivalent IMSL function.

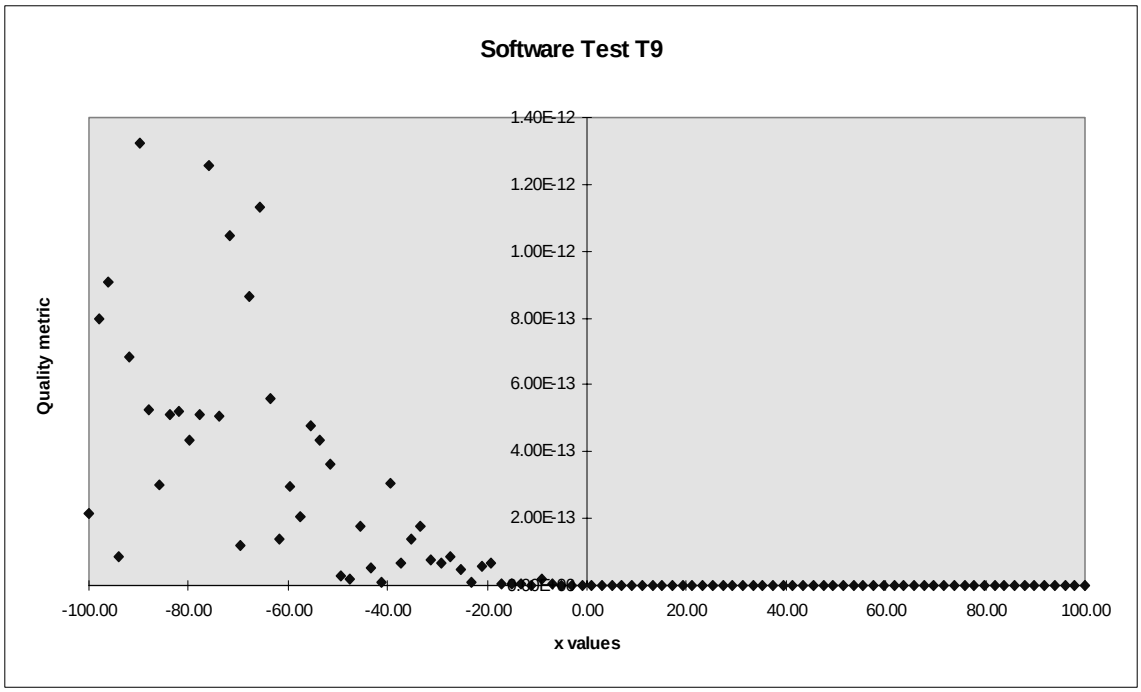

Figure 35 Graph of the quality metric $d_R(x)$ for the software test T9: SINH(ASINH(x)) = x.

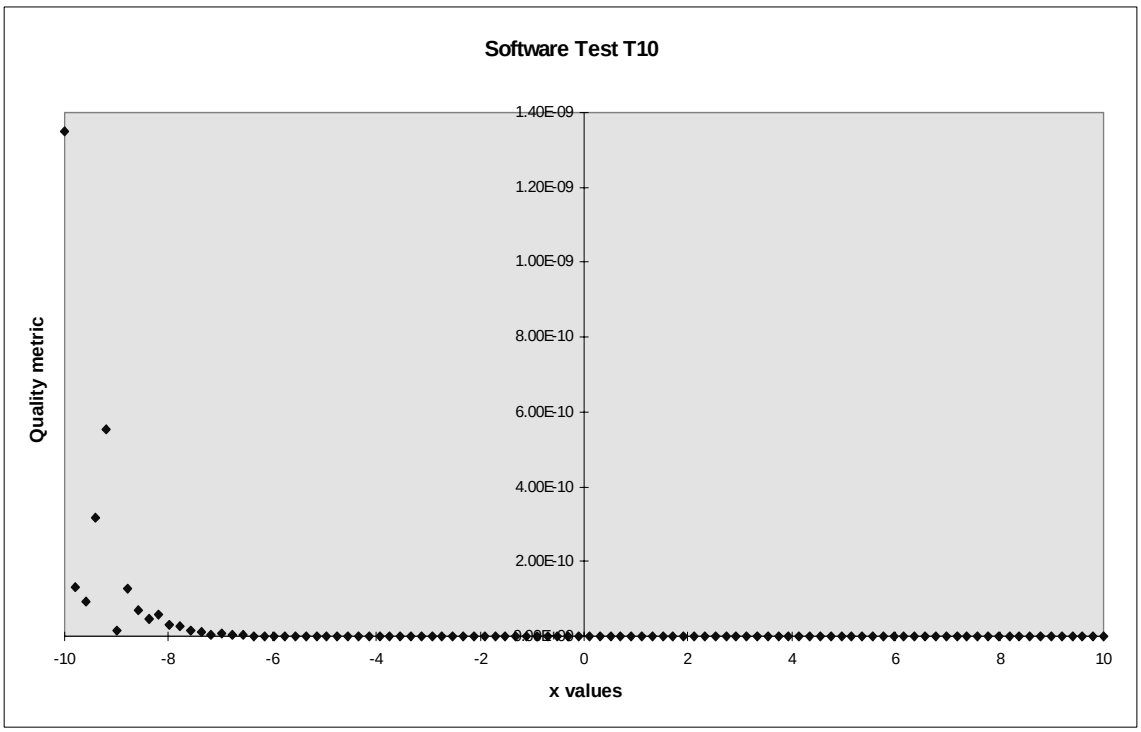

Figure 36 Graph of the quality metric $d_R(x)$ for the software test T10: ASINH(SINH(x)) = x.

Appendix C: Results for Statistical Distributions

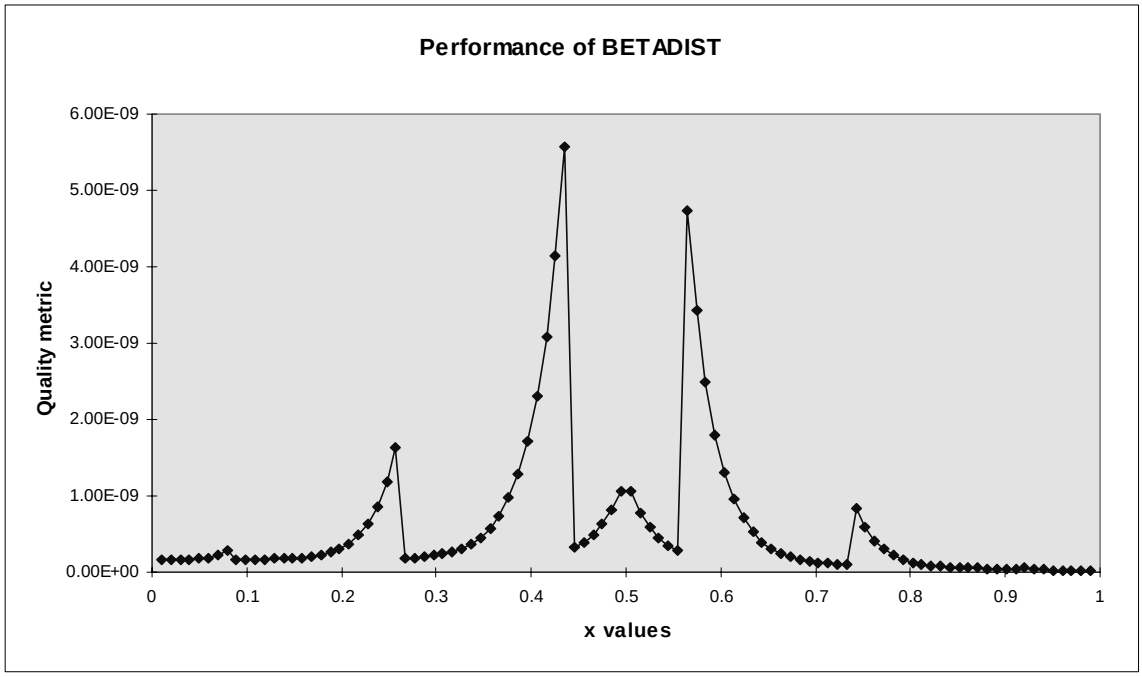

Figure 37 Graph of the quality metric $d_R(x)$ for the comparison of the BETADIST(x, 0.5, 0.5, 0, 1) worksheet function and the equivalent IMSL function.

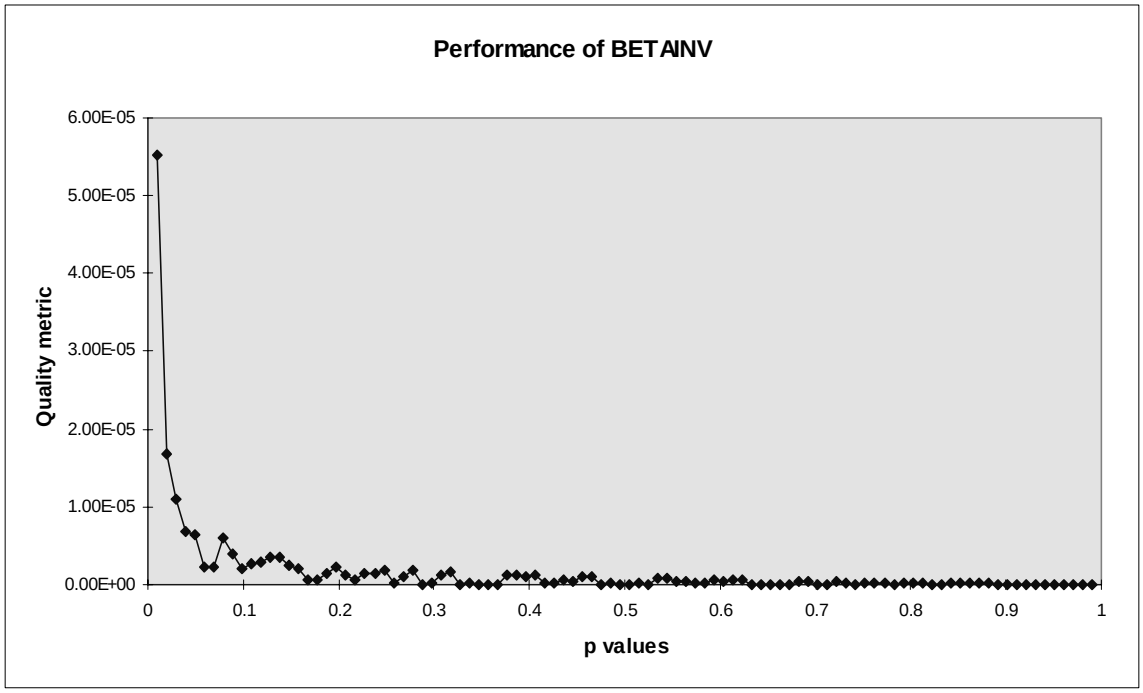

Figure 38 Graph of the quality metric $d_R(x)$ for the comparison of the BETAINV(p, 0.5, 0.5, 0, 1) worksheet function and the equivalent IMSL function.

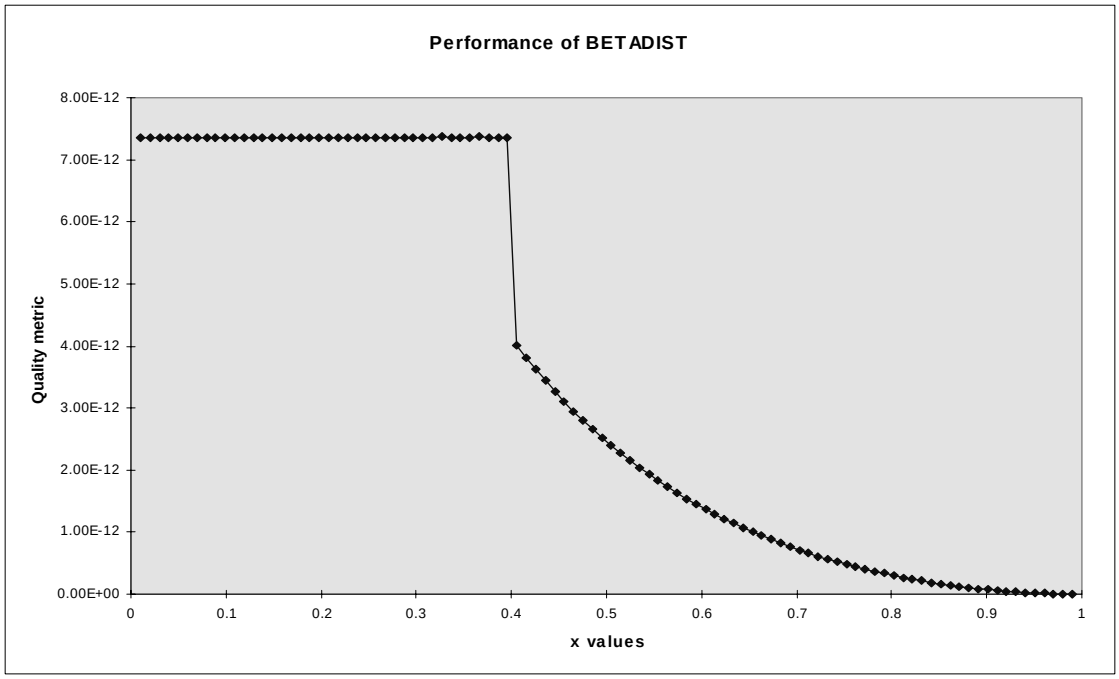

Figure 39 Graph of the quality metric $d_R(x)$ for the comparison of the BETADIST(x, 1, 2, 0, 1) worksheet function and the equivalent IMSL function.

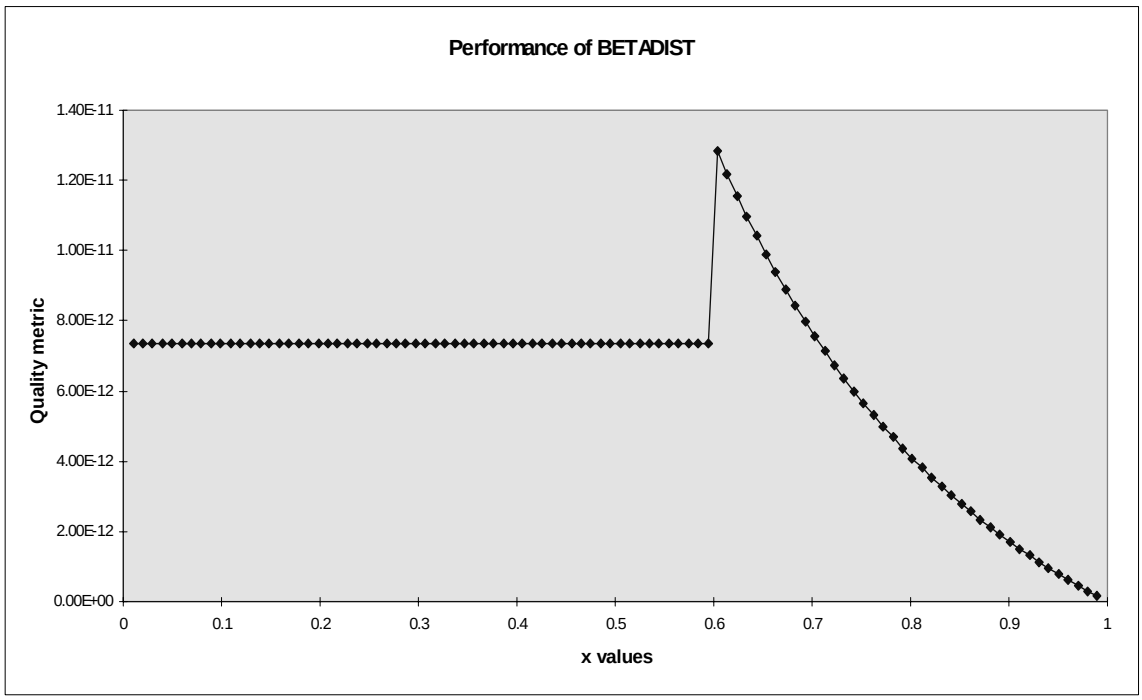

Figure 40 Graph of the quality metric $d_R(x)$ for the comparison of the BETADIST(x, 2, 1, 0, 1) worksheet function and the equivalent IMSL function.

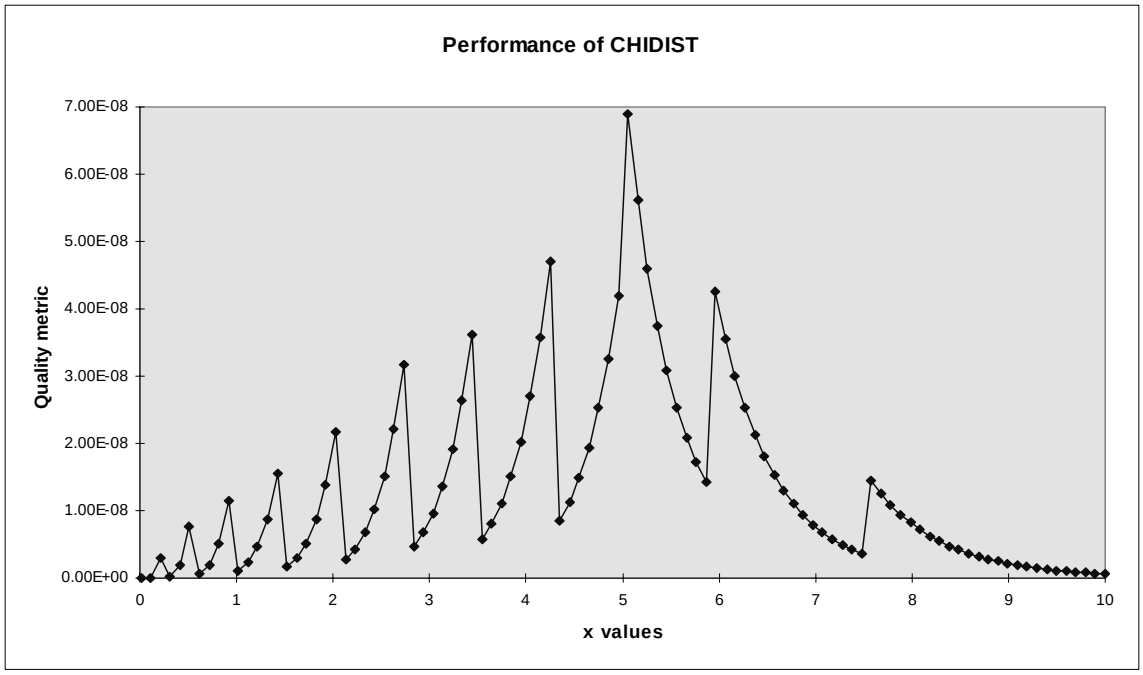

Figure 41 Graph of the quality metric $d_R(x)$ for the comparison of the CHIDIST(x, 5) worksheet function and the equivalent IMSL function.

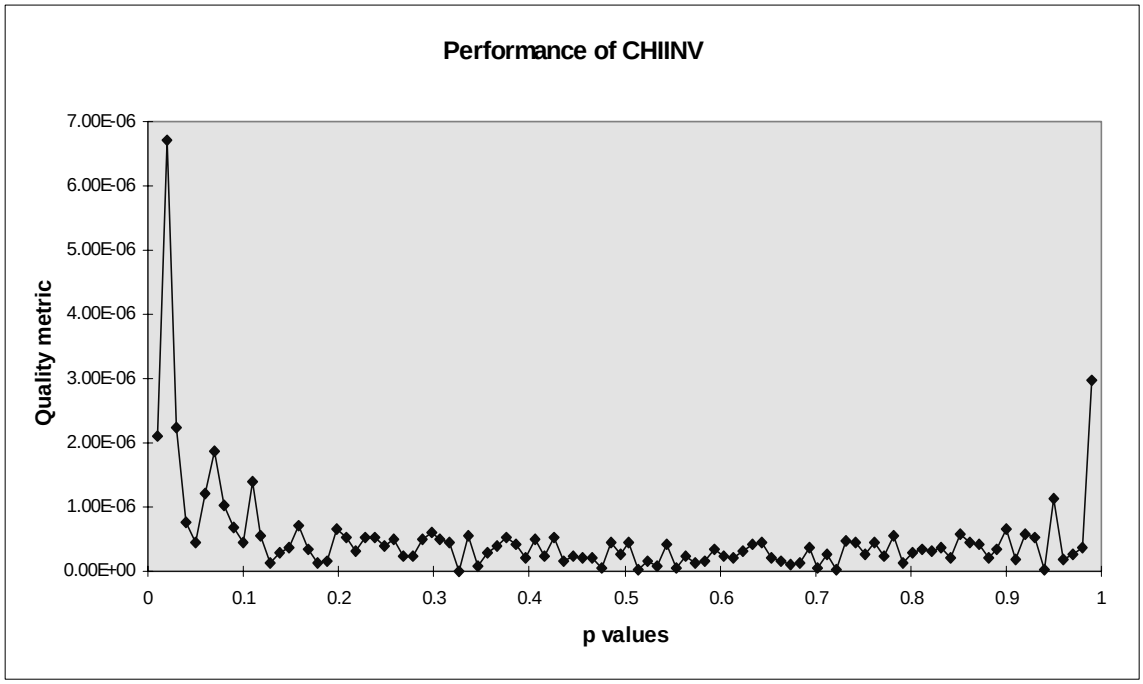

Figure 42 Graph of the quality metric $d_R(x)$ for the comparison of the CHIINV(p, 5) worksheet function and the equivalent IMSL function.

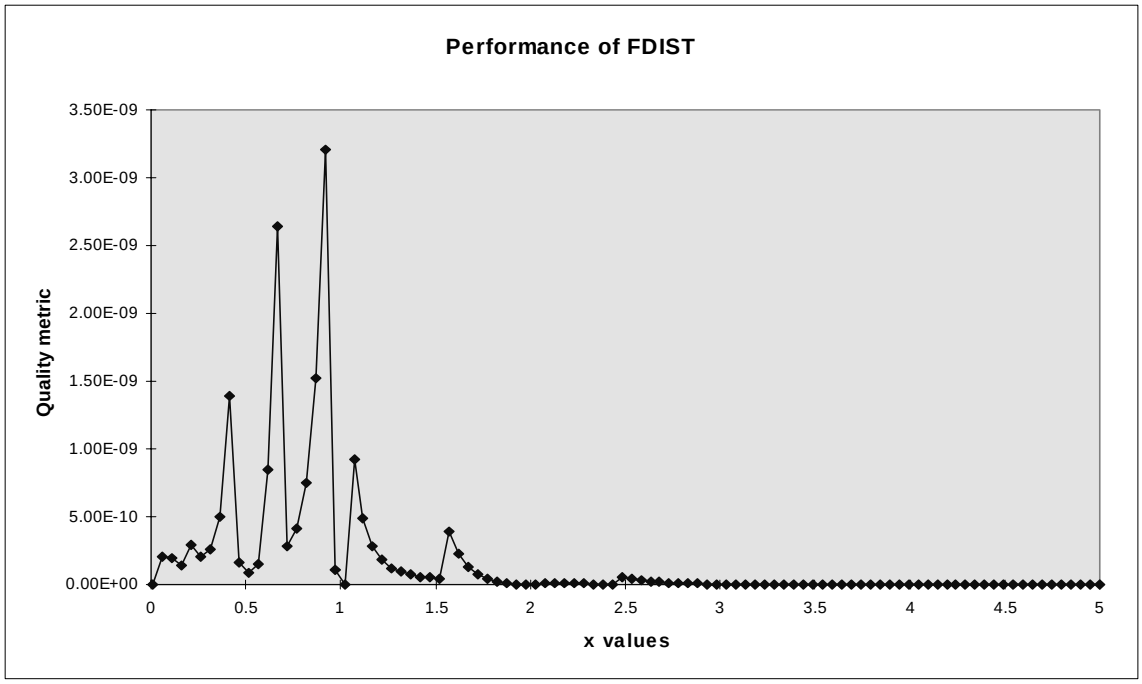

Figure 43 Graph of the quality metric $d_R(x)$ for the comparison of the FDIST(x, 15, 25) worksheet function and the equivalent IMSL function.

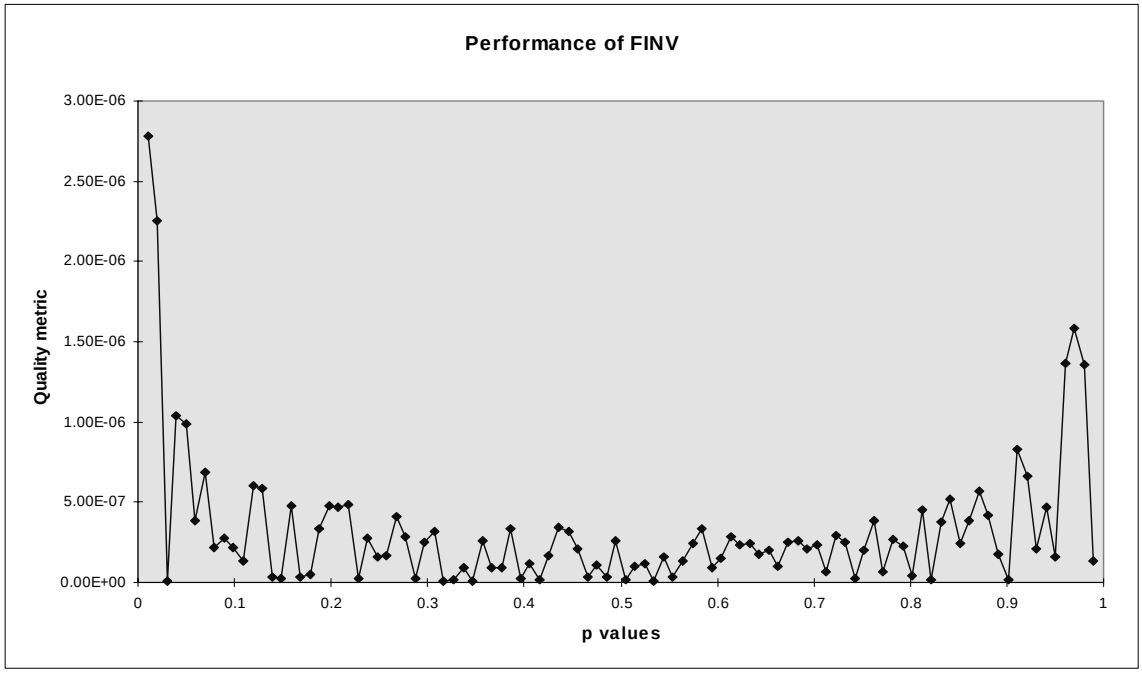

Figure 44 Graph of the quality metric $d_R(x)$ for the comparison of the FINV(p, 15, 25) worksheet function and the equivalent IMSL function.

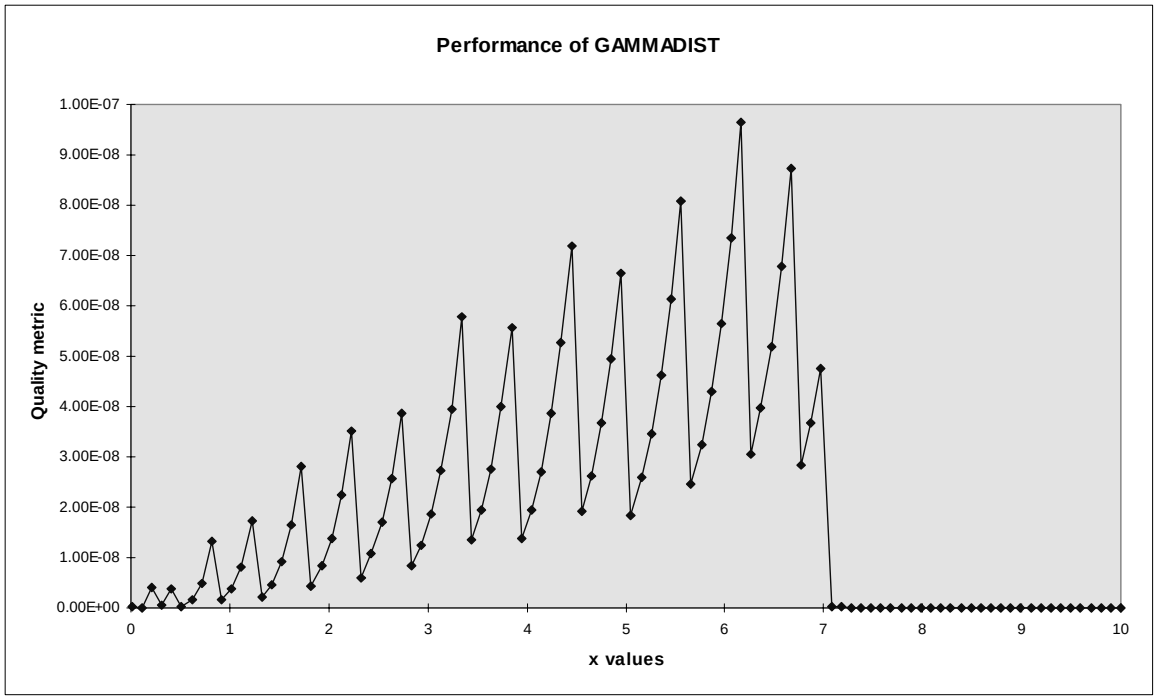

Figure 45 Graph of the quality metric $d_R(x)$ for the comparison of the GAMMADIST(x, 7, 1, *true*) worksheet function and the equivalent IMSL function.

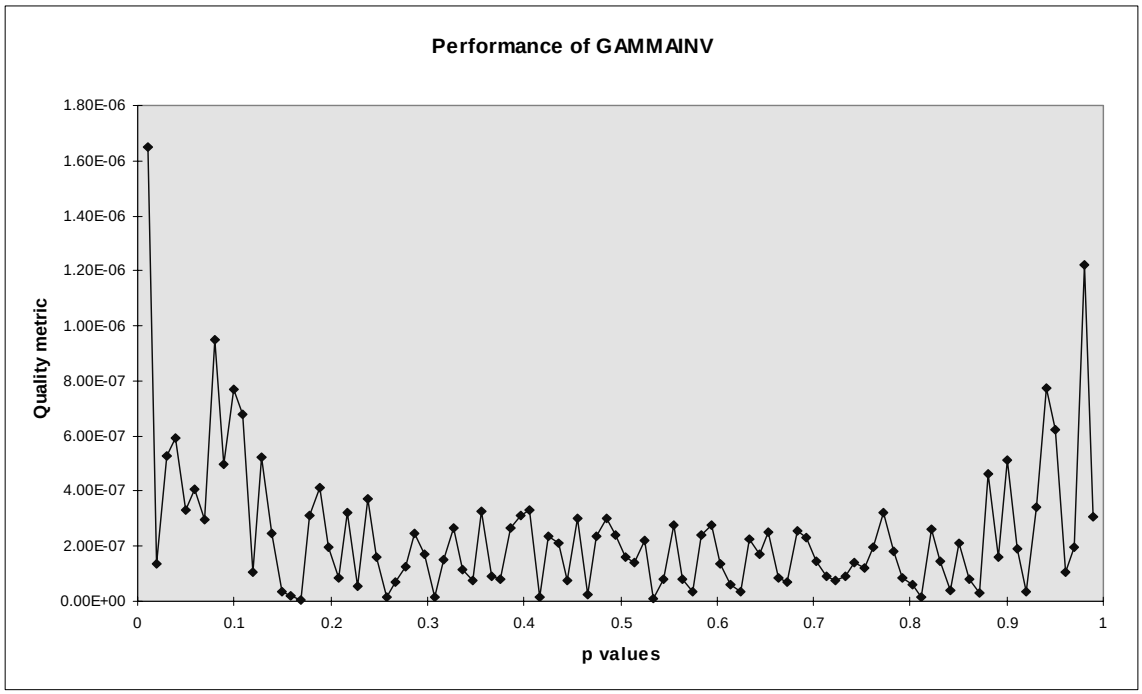

Figure 46 Graph of the quality metric $d_R(x)$ for the comparison of the GAMMAINV(p, 7, 1) worksheet function and the equivalent IMSL function.

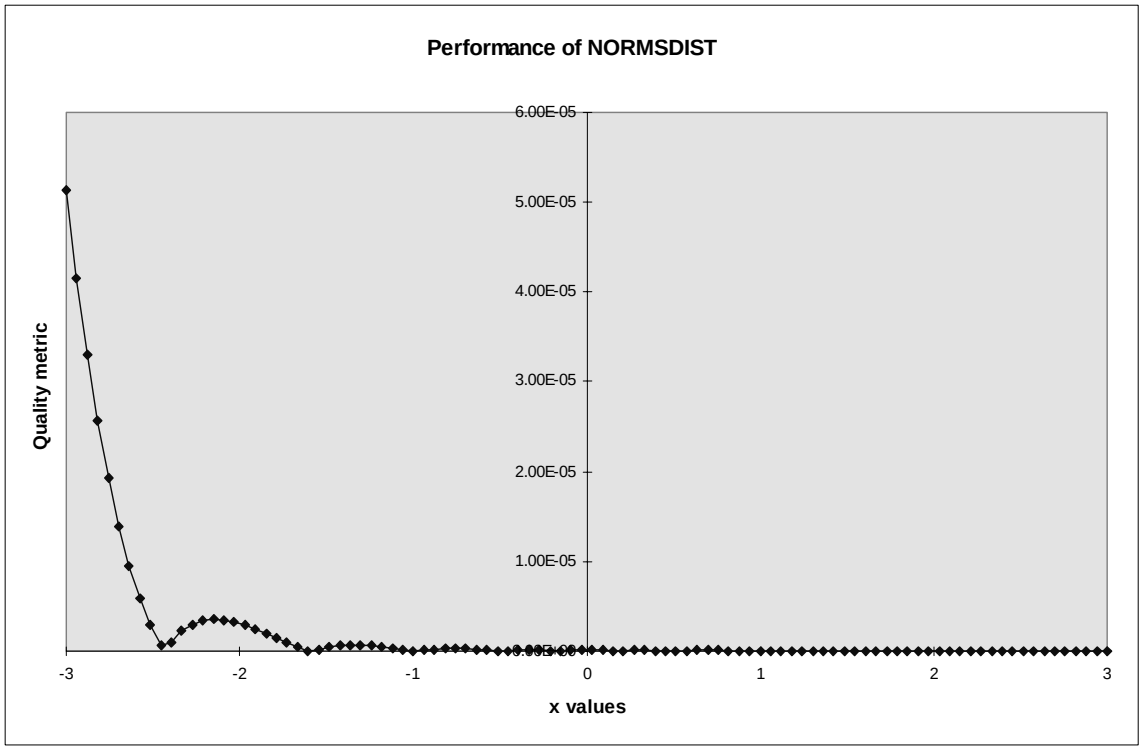

Figure 47 Graph of the quality metric $d_R(x)$ for the comparison of the NORMSDIST(x) worksheet function and the equivalent IMSL function.

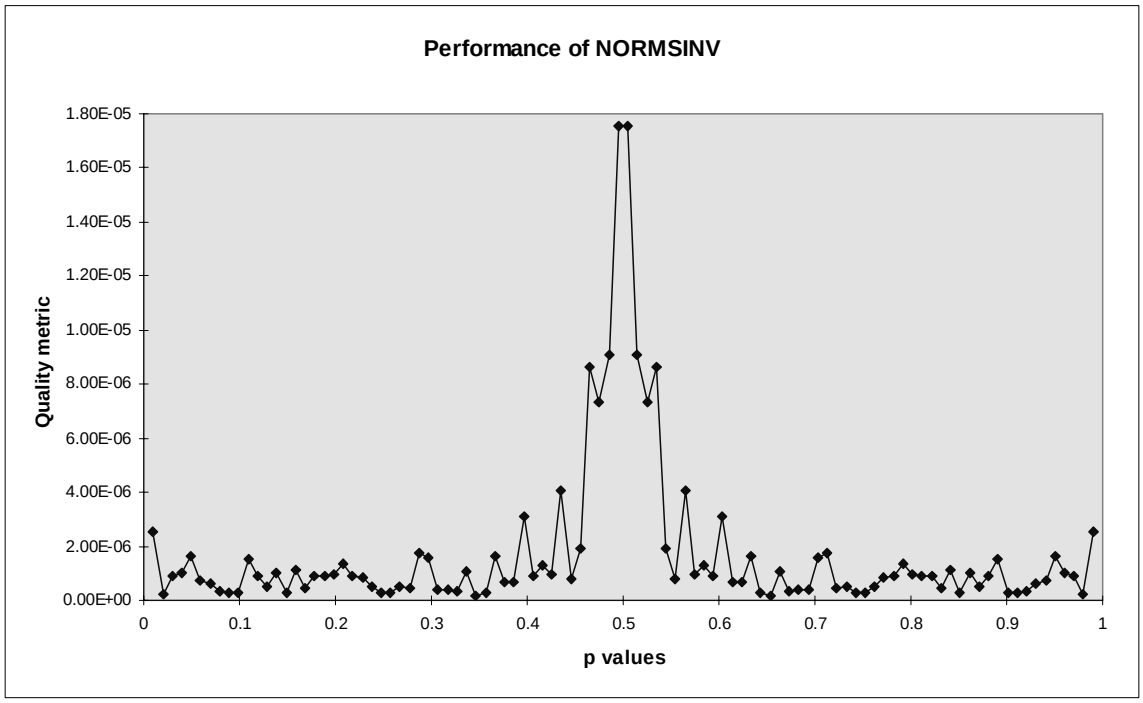

Figure 48 Graph of the quality metric $d_R(x)$ for the comparison of the NORMSINV(p) worksheet function and the equivalent IMSL function.

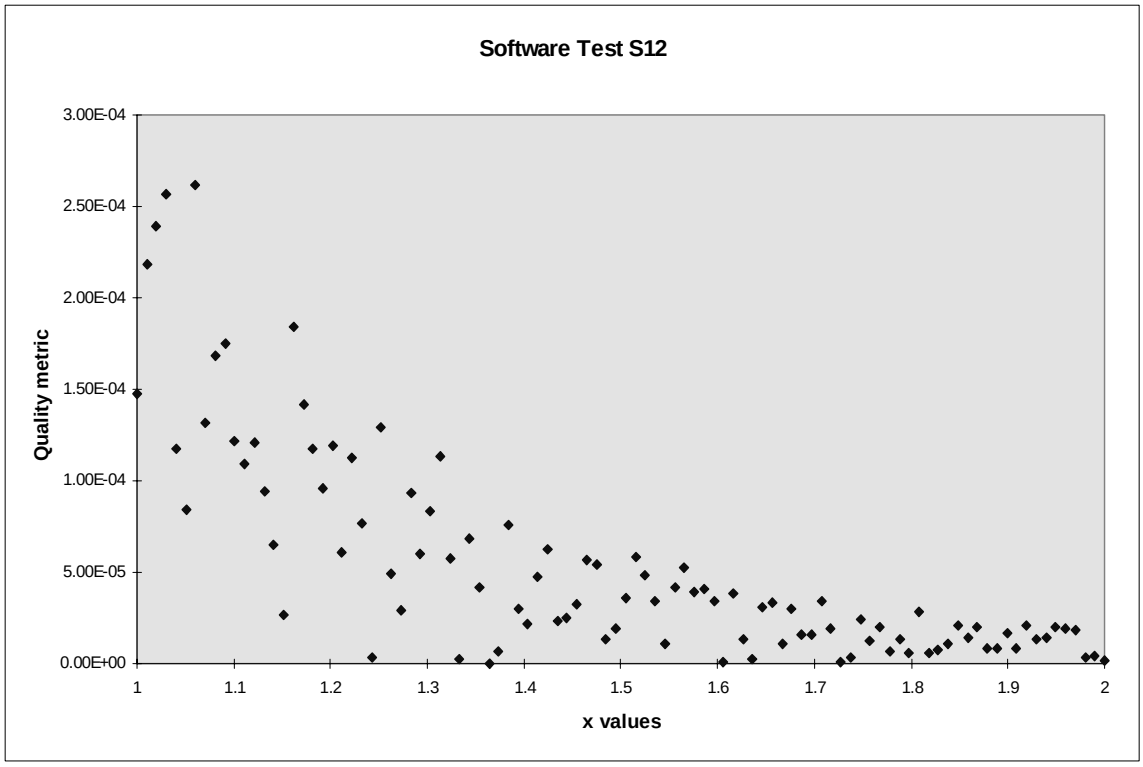

Figure 49 Graph of the quality metric $d_R(x)$ for the software test S12: CHIINV(CHIDIST(x, 10), 10) = x.

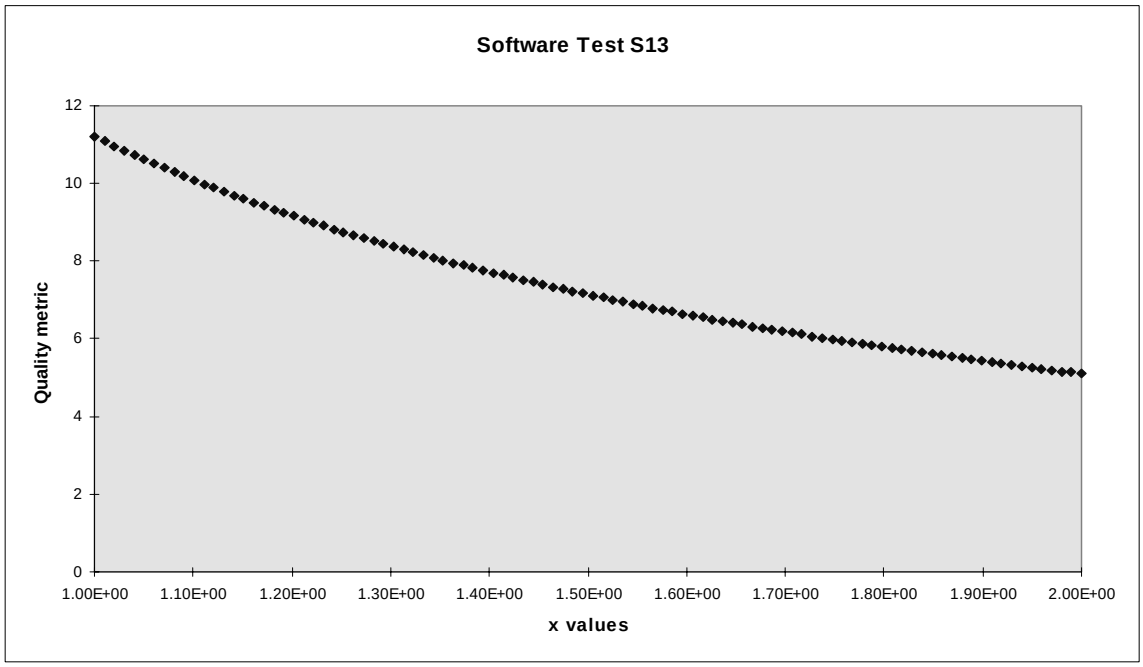

Figure 50 Graph of the quality metric $d_R(x)$ for the software test S13: CHIINV(CHIDIST(x, 100), 100) = x.