NPL Report CMSC 07/00

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

TESTING SPREADSHEETS AND OTHER PACKAGES USED IN METROLOGY

TESTING FUNCTIONS FOR THE CALCULATION OF STANDARD DEVIATION

BY

M G COX, M P DAINTON and P M HARRIS

October 2000

Testing Spreadsheets and Other Packages Used in Metrology
Testing Functions for the Calculation of the Standard Deviation

M G Cox, M P Dainton and P M Harris
Centre for Mathematics and Scientific Computing

October 2000

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 software to functions for the calculation of the sample standard deviation taken from a number of spreadsheet, statistical and scientific software packages, including Microsoft Excel, MathCAD, S-PLUS, Matlab, and the NAG and IMSL Fortran libraries. Each stage of the methodology, from the provision of a specification for the 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 1471-0005

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

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

Contents

1. Introduction

The numerical correctness of scientific software, 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. 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. 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.

A general methodology for evaluating the numerical accuracy of the results produced by scientific software is described in [1, 2, 3, 4], and illustrated by the case study [5] and its application to testing the Microsoft Excel [6], MathCAD [7] and S-PLUS [8] software packages. In this work, and that described in the companion report [9], we focus on a particular numerical calculation and report the results of testing functions for this calculation taken from a number of spreadsheet, statistical and scientific software packages used in metrology. The calculation considered is that of the *sample standard deviation s*, defined by

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

for the sample $\{x_i: i = 1, \ldots, m\}$, where $\bar{x}$ is the sample mean,

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

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

The software packages covered in this work are Excel, MathCAD, S-PLUS, Matlab, the NAG Fortran library and the IMSL Fortran library.

Some related work can be found in the literature. The papers [10, 11, 12] report the results of assessing the accuracy of statistical procedures taken from Microsoft Excel 97, SAS, SPSS and S-PLUS in the areas of estimation, random number generation and the calculation of statistical probabilities. The sample standard deviation is covered in this work, the approach to testing being to compare results for data sets provided by NIST through its Statistical Reference Datasets (StRD) web-site with certified results for these data sets [13].

The report is organised as follows. In Section 2 we discuss the methodology underlying the testing carried out. The methodology is described in detail in [4], and relies on the use of reference data sets and corresponding reference results to undertake black-box testing of the functions. Sections 3, 4, 5 and 6 contain details of the implementation of the methodology. Section 3 contains specifications of the functions tested. Section 4 describes the performance parameters used to characterise the reference data sets and the performance measures used to quantify the performance of each function on these data sets. Section 5 contains the procedure

used to generate reference pairs, i.e., reference data sets and corresponding reference results. In Section 6 we present the results of the testing, and provide some interpretation of these results. Section 7 contains our conclusions.

2. Methodology

A general methodology for evaluating the numerical accuracy of the results produced by scientific software is described in [1, 2, 3, 4]. 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.

The methodology comprises the following six stages [4]:

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 [4] with their application by a software *developer* presented in the case study [5]. 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.

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. Specifications of functions for calculating the sample standard deviation tested in this work are given in Section 3.

2.2 Interfacing to the test software

Where possible the testing procedure is executed automatically using for each package its own intrinsic programming language: VBA (Visual Basic for Applications) for Excel, Fortran for the NAG and IMSL libraries, MathCAD's own programming syntax, etc. The test function is called in such a way as to mimic how a user might be expected to access the function. Each

reference data set, with its corresponding reference result and other information (such as the problem degree of difficulty used to compute the performance measure: see Section 4), is read from data files. Values of the performance parameter are calculated using functions provided by the intrinsic programming language, written to data files, and displayed using Excel's graphing facilities.

Further details of interfacing issues relating to Excel, MathCAD and S-PLUS are given in [6], [7] and [8], respectively. Note that the version of Excel considered in this work (Microsoft Excel 2000) is different from that tested in the report [6] (Microsoft Excel for Windows 95 Version 7.0a). Furthermore, with the exception of the NAG Fortran library, the testing was carried out on a PC running Windows NT: testing of the NAG Fortran library was carried out on a VAX computer running VMS.

2.3 Specification of reference data sets

Performance parameters are used to capture the properties of data sets that would be encountered in practice and to describe the range of admissible inputs to the test software. By varying an individual performance parameter sequences of data sets may be generated, with the sequence forming a *graded* sequence in cases where the performance parameter relates directly to the condition or "degree of difficulty" of the problem represented by the data. By investigating the performance of the test software for such graded sequences, it is possible to identify cases where the test software is based upon a poor choice of mathematical algorithm. Performance parameters for the calculation of the sample standard deviation are given in Section 4.

2.4 Specification of performance measures and testing requirements

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

In [4] the following performance measure is derived:

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

where $\mathbf{x}$ denotes the input reference data set, $\mathbf{y}^{\text{test}}$ and $\mathbf{y}^{\text{ref}}$ are, respectively, the test and reference results, $\kappa(\mathbf{x})$ measures the problem degree of difficulty defined by the data set $\mathbf{x}$, and η is the computational precision[1]. 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. A performance measure for the calculation of the sample standard deviation is given in Section 4.

2.5 Generation of reference pairs

In the general methodology, a *reference pair*, i.e., a reference data set and corresponding reference results, may be produced either using *reference software* or a *data generator* [4]. Reference software is software written to an extremely high standard to solve the problem given in the functional specification. Alternatively, data generators are used to construct

[1] 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.

reference data sets with known solutions, i.e., solutions specified *a priori*, and are based on *null-space methods* [3, 4] that use a solution characterisation to construct families or classes of data sets possessing nominally the same solution. In Section 5 we present the procedure used in this work for generating reference pairs.

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 performance measure (Section 2.4). The performance measure is presented as a function 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.

To use the results of the testing, for example, where these are 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. The presentation and interpretation of testing results for the calculation of the sample standard deviation is given in Sections 6 and 7.

3. Specifications of the Functions Tested

In this section we list the functions that have been tested as part of this work. We also provide a specification for each function, based on the on-line help documentation provided with the software package or library from which the function is taken. We believe it is appropriate to use the on-line documentation for providing these specifications, as this is the natural source of information for most users of these packages and libraries. It is also a convenient and straightforward way of obtaining details of inputs/outputs and mode of implementation.

The particular implementations of the sample standard deviation function tested in this work are those contained within the following packages:

- the IMSL Fortran library [14],
- the NAG Fortran library [15],
- Matlab [16]
- Microsoft Excel [17],
- S-PLUS 4.0 [18], and
- MathCAD [19].

3.1 IMSL Library

DUVSTA

Compute basic univariate statistics.

```
SUBROUTINE  UVSTA  (IDO,  NROW,  NVAR,  X,  LDX,  IFRQ,  IWT,  MOPT,
           CONPRM, CONPRV, IPRINT, STAT, LDSTAT, NRMISS)
```

Parameters

IDO — Processing option. (Input)

IDO = 0: This is the only invocation of UVSTA for this data set, and all the data are input at once.

IDO = 1: This is the first invocation, and additional calls to UVSTA will be made. Initialization and updating for the data in X are performed. The means are output correctly, but the other quantities output in STAT are intermediate quantities.

IDO = 2: This is an intermediate invocation of UVSTA, and updating for the data in X is performed.

IDO = 3: This is the final invocation of this routine. If NROW is not zero, updating is performed. The wrap-up computations for STAT are performed.

NROW — The absolute value of NROW is the number of rows of data currently input in X. (Input)

NROW may be positive, zero, or negative. Negative NROW means that the −NROW rows of data are to be deleted from some aspects of the analysis, and this should be done only if IDO is 2 or 3 and the wrap-up computations for STAT have not been performed. When a negative value is input for NROW, it is assumed that each of the −NROW rows of X has been input (with positive NROW) in a previous invocation of UVSTA. Use of negative values of NROW should be made with care and with the understanding that some quantities in STAT cannot be updated properly in this case. In particular, the minima, maxima, and ranges are not updated because of deletion. It is also possible that a constant variable in the remaining data will not be recognized as such.

NVAR — Number of variables (not including the weight or frequency variable, if used). (Input)

X — |NROW| by NVAR + m matrix containing the data, where m is 0, 1, or 2 depending on whether any column(s) of X correspond to weights and/or frequencies. (Input)

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)

IFRQ — Frequency option. (Input)

IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies.

IWT — Weighting option. (Input)

IWT = 0 means that all weights are 1.0. For positive IWT, column IWT of X contains the weights.

MOPT — Missing value option. (Input)

NaN (not a number from routine AMACH(6)) is interpreted as the missing value code and any value in X equal to NaN is excluded from the computations.

MOPT = 0: The exclusion is listwise. (The entire row of X is excluded if any of the values of the row is equal to the missing value code.)

MOPT = 1: The exclusion is elementwise. (Statistics for variables with nonmissing values are updated.)

CONPRM Confidence level for two-sided interval estimate of the means (assuming normality), in percent. (Input)

If CONPRM ≤ 0, no confidence interval for the mean is computed; otherwise, a CONPRM percent confidence interval is computed, in which case CONPRM must be between 0.0 and 100.0. CONPRM is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPRM = 100.0 – 2.0 * (100.0 – ONECL).

CONPRV Confidence level for two-sided interval estimate of the variances (assuming normality), in percent. (Input)

The confidence intervals are symmetric in probability (rather than in length). See also the description of CONPRM.

IPRINT Printing option. (Input)

IPRINT = 0: No printing is performed.

IPRINT = 1: Statistics in STAT are printed if IDO = 0 or 3.

IPRINT = 2: Intermediate means, sums of squares about the mean, minima, maxima, and counts are printed when IDO = 1 or 2, and all statistics in STAT are printed when IDO = 0 or 3.

STAT 15 by NVAR matrix containing in each row statistics on all of the variables. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

The columns of STAT correspond to the columns of X, except for the columns of X containing weights or frequencies. (The columns beyond the weights or frequencies column are shifted to the left.)

I	STAT(I, *)
1	contains means
2	contains variances
3	contains standard deviations
4	contains coefficients of skewness
5	contains coefficients of excess (kurtosis)
6	contains minima
7	contains maxima
8	contains ranges
9	contains coefficients of variation, when they are defined. If the coefficient of variation is not defined for a given variable, STAT(9, *) contains a zero in the corresponding position.
10	contains numbers (counts) of nonmissing observations
11	is used only when CONPRM is positive, and, in this case, contains the lower confidence limit for the mean (assuming normality)
12	is used only when CONPRM is positive, and, in this case, contains the upper confidence limit for the mean (assuming normality)
13	is used only when CONPRV is positive, and, in this case, contains the lower confidence limit for the variance (assuming normality).

14 is used only when CONPRV is positive, and, in this case, contains the upper confidence limit for the variance (assuming normality).

15 is used only when weighting is used (IWT is nonnegative), and, in this case, contains the sums of the weights.

LDSTAT Leading dimension of STAT exactly as specified in the dimension statement in the calling program. (Input)

NRMISS Number of rows of data encountered in calls to UVSTA that contain any missing values. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

Rows with a frequency of zero are not counted.

3.2 NAG Library

G01AAF

Calculates the mean, standard deviation, coefficients of skewness and kurtosis, and the maximum and minimum values for a set of ungrouped data. Weighting may be used.

```
SUBROUTINE G01AAF(N, X, IWT, WT, XMEAN, S2, S3, S4, XMIN, XMAX,
          WTSUM, IFAIL)
INTEGER    N, IWT, IFAIL
Real       X(N), WT(N), XMEAN, S2, S3, S4, XMIN, XMAX, WTSUM
```

Parameters

N On entry: number of observations, n.

X On entry: sample observations x_i, $i = 1, \ldots, n$.

IWT On entry: indicates whether weights are to be supplied by the user or not. In the latter case, the weights will be assumed equal and assigned the value 1.0 in the routine. IWT = 0 indicates no user-supplied weights. IWT = 1 indicates user-supplied weights are required, and they will be supplied in the array WT.

On exit: IWT is used to indicate the number of valid observations.

WT On entry: if IWT = 1, the elements of WT must contain the weights associated with the observations, w_i, $i = 1, .., n$. If IWT = 0, the elements of WT need not be set.

On exit: if IWT = 1, the elements of WT are unchanged. If IWT = 0, each element of WT is assigned the value 1.0.

XMEAN On exit: the mean $\bar{x}$, where

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{W}, \quad W = \sum_{i=1}^{n} w_i .$$

S2 On exit: the standard deviation s_2, where

$$s_2 = \sqrt{\frac{\sum_{i=1}^{n} w_i \left(x_i - \bar{x}\right)^2}{d}}, \quad d = W - \frac{\sum_{i=1}^{n} w_i^2}{W}.$$

S3 On exit: the coefficient of skewness.

S4: On exit: the coefficient of kurtosis.

XMIN On exit: the smallest value in the sample.

XMAX On exit: the largest value in the sample.

WTSUM On exit: the sum of the weights, W. This will be N if IWT was 0 on entry.

IFAIL On entry: IFAIL must be set to 0, –1 or 1. For users not familiar with this parameter, the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error. IFAIL = 1 if N < 1; IFAIL = 2 if the number of valid observations for which $w_i > 0$ is 1 (in this case, standard deviation and coefficients of skewness and of kurtosis cannot be calculated); IFAIL = 3 if either the number of valid observations is 0, or at least one weight is negative.

3.3 Matlab

STD

For vectors, STD(X) returns the standard deviation. For matrices, STD(X) is a row vector containing the standard deviation of each column. For *N–D* arrays, STD(X) is the standard deviation of the elements along the first non-singleton dimension of X.

STD(X) normalizes by $(N - 1)$ where N is the sequence length. This makes STD(X).^2 the best unbiased estimate of the variance if X is a sample from a normal distribution.

STD(X, 1) normalizes by N and produces the second moment of the sample about its mean. STD(X, 0) is the same as STD(X).

STD(X, FLAG, DIM) takes the standard deviation along the dimension DIM of X. When FLAG = 0 STD normalizes by $(N - 1)$, otherwise STD normalizes by N.

3.4 Excel

STDEV

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

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

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

3.5 S-PLUS

The sample standard deviation is calculated as the square root of the variance returned by the intrinsic function VAR of S-PLUS.

VAR

Returns the variance of a vector, the variance-covariance matrix of a data matrix, or covariances between matrices or vectors. If *x* is a matrix, the result is a matrix such that the [i,j] element is the covariance of *x*[,i] and either *y*[,j] or *x*[,j]. If *x* is a vector, the result is a vector, with length equal to the number of columns of *y* (or length 1 if *y* is not supplied).

var(*x*, *y*, *na.method*="fail", *unbiased*=T, *SumSquares*=F)

REQUIRED ARGUMENTS

x numeric matrix or vector, or data frame. May be complex. If a matrix, columns represent variables and rows represent observations. If a data frame, non-numeric variables result in missing values in the result.

OPTIONAL ARGUMENTS

y numeric matrix or vector, or data frame. May be complex. If a matrix, columns represent variables and rows represent observations. If a data frame, non-

numeric variables result in missing values in the result. This must have the same number of observations as *x*.

na.method a character string specifying how missing values are to be handled. Options are

"fail" (stop if any missing data is found),

"omit" (omit rows with any missing data),

"include" (missing values in the input result in missing values in the output)

"available" (use available observations, see below). Only enough of the string to determine a unique match is required.

unbiased if TRUE, then variances are sample variances, e.g.

sum((x-mean(x))^2)/(N-1)

for a vector of length *N*, which is unbiased if the values in *x* are obtained by simple random sampling. If FALSE, the definition

sum((x-mean(x))^2)/N

is used instead.

SumSquares if TRUE, then unnormalized sums of squares are returned, with no division by either *N* or $(N-1)$ (and unbiased is ignored).

3.6 MathCAD

STDEV

stdev(v) returns the square root of the variance var(**v**) of the elements in **v**, where

$$\mathrm{var}(\mathbf{v}) = \frac{1}{m-1}\sum_{i=0}^{m-1}\left(v_i - \mathrm{mean}(\mathbf{v})\right)^2,$$

and mean(**v**) is the mean of the elements in **v**.

Arguments:

v is an $m \times 1$ vector.

4. Specification of Performance Parameters and Measures

Performance parameters for the sample standard deviation include:

- the reference sample mean $\bar{x}$,
- the reference sample standard deviation s^{ref}, and
- the number *m* of points in the sample.

Three sequences of reference data sets are used to test the functions, where each sequence is generated by setting two of the performance parameters equal to a nominal value and varying the other parameter within a specified range. The nominal values and ranges for each performance parameter are given in Table 1.

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

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

where $\kappa(s)$ measures the problem degree of difficulty,

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

and η is the machine precision [4].

The information contained in Table 1 and the formula for $\kappa(s)$ given above enable a "degree of difficulty" to be assigned to each reference data set. The degree of difficulty is to be interpreted in the following way: $\log_{10}\kappa(s)$ is the number of significant figures of accuracy lost by a reference algorithm for computing the sample standard deviation. The performance measure $P(s)$ then indicates the number of additional significant figures of accuracy lost by test software for this calculation [4].

The ranges of values for the performance parameters indicated in Table 1 are chosen so that the generated reference data sets have a wide range of degrees of difficulty, as follows:

- for the sequence of data sets generated by varying the reference sample mean, the degree of difficulty varies from 10^1 to 10^{16},
- for the sequence generated by varying the reference sample standard deviation, the degree of difficulty varies from 10^{16} down to 10^1, and
- for the sequence generated by varying the number of sample points, the degree of difficulty is fixed at (approximately) 15.

Note that data for which $\bar{x}/s >> 1$ is not uncommon in metrology applications and can arise, for example, in the form of replicated high-accuracy measurements [4]. The information given above is useful when interpreting graphs of the performance measure against a performance parameter: see Section 6.

Performance parameter	*Nominal value*	*Range*
Reference mean $\bar{x}$	15.450298510862456	$[1, 10^{16}]$
Reference standard deviation s^{ref}	1	$[10^{-15}, 10]$
Number m of points	100	[6, 400]

Table 1: Nominal values and ranges for each performance parameter

5. Generation of Reference Pairs

In this section we present the procedure used to generate reference pairs, i.e., reference data sets with corresponding results. The procedure does not make use of reference software but instead implements an approach for generating a reference data set **x** which has prescribed values for the performance parameters listed in Section 4, i.e., reference sample mean $\bar{x}$, reference sample standard deviation s^{ref} and number m of points in the sample (with m assumed to be even).

Given values for $\bar{x}$, s and $m = 2n$, the following steps are undertaken.

Firstly, generate a set of n values y_i, $i = 1, \ldots, n$, satisfying

$$y_i \geq 0, \quad i = 1,\ldots,n,$$

and

$$\frac{1}{n}\sum_{i=1}^{n} y_i = \left(\frac{2n-1}{2n}\right)\left(s^{\text{ref}}\right)^2,$$

i.e., the values y_i are non-negative and have a prescribed sample mean. This is done by noting that that the mean of a data set is equal to the least-squares best-fit constant to the data, and to use the *null-space method* described in [3, 4] for generating a data set for which the least-squares best-fit constant is prescribed *a priori*. The null-space method is used to construct a vector $\mathbf{r} = (r_1, r_2, \ldots, r_n)^{\mathrm{T}}$ of (pseudo-)random numbers such that the n values

$$y_i = \left(\frac{2n-1}{2n}\right)\left(s^{\text{ref}}\right)^2 + r_i, \quad i = 1,\ldots,n,$$

have the required mean. The elements of the vector **r** are chosen to be close (in a least-squares sense) to a vector of Gaussian distributed numbers so as to ensure the elements in the generated sample are sensibly distributed [4]. Furthermore, the standard deviation of this Gaussian distribution is chosen so that the generated samples of values y_i are non-negative.

Secondly, define a set of m values x_i, $i = 1, \ldots, m$, by

$$x_{2i-1} = +\sqrt{y_i} \quad \text{and} \quad x_{2i} = -\sqrt{y_i}, \quad i = 1,\ldots,n.$$

Then, the values x_i, $i = 1, \ldots, m$, satisfy

$$\text{mean}\left(\{x_i\}\right) = 0,$$

and

$$\text{std}\left(\{x_i\}\right) = \sqrt{\frac{\sum_{i=1}^{m} x_i^2}{m-1}} = \sqrt{\frac{2\sum_{i=1}^{n} y_i}{2n-1}} = s^{\text{ref}}.$$

Finally, define a (new) set of m values x_i, $i = 1, \ldots, m$, by

$$x_i := x_i + \bar{x}, \quad i = 1,\ldots,m,$$

and randomise the order of the data values. Then, since the sample standard deviation is invariant to a translation of the data values, the values x_i, $i = 1, \ldots, m$, satisfy, as required,

$$\text{mean}\left(\{x_i\}\right) = \bar{x},$$

and

$$\text{std}\left(\{x_i\}\right) = s^{\text{ref}}.$$

6. Presentation and Interpretation of Results

The results are presented in the Appendix in Figures 1-18. The figures show the performance measure P plotted against each performance parameter (reference mean, reference standard deviation and number of data points), and are arranged in the following way:

- IMSL subroutine **DUVSTA**: Figures 1–3,
- NAG subroutine **G01AAF**: Figures 4–6,
- Matlab function **STD**: Figures 7–9,
- Excel function **STDEV**: Figures 10–12,

- S-PLUS function **VAR**: Figures 13–15, and
- MathCAD function **STDEV**: Figures 16–18.

Table 2 provides a (simplified) quantitative interpretation of the results shown in the Figures. The table lists the number of significant figures of accuracy lost by each test function for the tests performed compared with a reference algorithm for the sample standard deviation.

The results indicate that the IMSL, NAG, Matlab, S-PLUS and MathCAD functions give results that are accurate for the ranges of reference data sets considered. For the IMSL, NAG, Matlab and S-PLUS functions, the performance measure takes values that are consistently less than unity, indicating that no additional significant figure of accuracy is lost compared with a reference implementation; for MathCAD there are isolated data sets for which the performance measure is approximately two. We conclude that these packages provide reliable software for the calculation of the sample standard deviation.

The results for the Excel function show that this function loses additional figures of accuracy compared with a reference implementation. Figure 10 shows a clear trend in the performance of the test software as a function of the reference sample mean (and hence the degree of difficulty). For values of the reference sample standard deviation down to 10^{-7}, this trend is also apparent in Figure 11. Notice that when the reference sample standard deviation is 10^{-7}, both $\log_{10} \kappa(s)$ and $P(s)$ are approximately eight and so *all* figures of accuracy are lost in the test result. For values of the reference sample standard deviation smaller than 10^{-7}, no further significant figures of accuracy can be lost, and $P(s)$ behaves like $\log_{10} s$: this is a consequence of the way $P(s)$ is calculated rather than being indicative of the performance of the test software. Unlike the other packages, as the sample size is varied (Figure 12), the value of the performance measure is consistently greater than unity, indicating that the software loses significant figures of accuracy over and above the number predicted by the problem difficulty. We conclude that Excel's function does not implement a reliable algorithm for the calculation of the sample standard deviation.

Implementation	Performance Parameter		
	Mean $\bar{x}$	Standard deviation s^{ref}	Number of points m
IMSL	< 1	< 1	< 1
NAG	< 1	< 1	< 1
Matlab	< 0.1	< 0.1 (except for a small number of data sets for which s^{ref} is small and $P < 1$)	< 0.1
Excel	Roughly a linear increase with $\log \bar{x}$ from < 1 for $\bar{x} = 1$ to 8 for $\bar{x} = 10^{16}$	Roughly quadratic in $\log s$ from 2 for $s^{\text{ref}} = 10^{-15}$ or 10^{1} rising for intermediate values to 8 for $s^{\text{ref}} = 10^{-7}$	Rising from 1 for very small sample sizes to 2.5 $m = 50$ and maintaining this value for larger m
S-PLUS	< 1	< 1	< 1
MathCAD	Roughly linear increase with $\log \bar{x}$ from < 1 for $\bar{x} = 1$ to 1.5 for $\bar{x} = 10^{16}$	< 1	< 2.5

Table 2: Number of significant figures of accuracy lost for the tests performed compared with a reference algorithm for the sample standard deviation.

7. Conclusions

In this report we have described the application of a general methodology [4] for testing the numerical accuracy of software to functions for the calculation of the sample standard deviation taken from a number of spreadsheet, statistical and scientific software packages. Each stage of the methodology, from documenting a specification for the 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.

The work reported here has focussed on functions for the calculation of the sample standard deviation. Consequently, conclusions drawn from the testing undertaken of a particular function must be interpreted in the context of that function only, and not to reflect on other functions of the software package or library from which that function is taken. 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 or information posted on the World Wide Web; only the documentation available on-line as part of the normal 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.

The test results indicate that the IMSL, NAG, Matlab, S-PLUS and MathCAD packages provide reliable software for the calculation of the sample standard deviation. However, this is not the case for the Excel spreadsheet package that appears to implement an algorithm whose performance degrades as a function of the problem degree of difficulty. For general use of Excel's standard deviation function, users are recommended to pre-process the data (using mean-centring of the data) prior to application of the Excel function [6, 20].

A further test was carried out. The formula in Section 3.4 that is stated by the Excel documentation to be used in the Excel function STDEV was implemented within Excel and the results obtained compared with those from STDEV. For all data sets used for this purpose the results were identical. Therefore, it would appear that STDEV does indeed utilise this formula. A floating-point error analysis [21] of this formula shows that its use would be expected to give rise to a loss of figures over and above the use of a stable formula such as that in Section 1. Furthermore, this loss can be expected to be proportional to $\log_{10}(\bar{x}/s)$, a behaviour that is apparent in Figure 10.

In contrast, it seems likely that the other implementations tested had implemented either the formula in Section 1 or a formula or procedure of comparable numerical stability. IMSL, NAG and S-PLUS perform comparably well. There are *some* variations: Matlab performs exceptionally well, and MathCAD suffers a small loss of accuracy.

8. 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.

The authors would like to thank their colleagues Jessica Barrett and Paul Kenward for their help in carrying out the testing described here.

9. References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] H.R. Cook, M.G. Cox, M.P. Dainton and P.M. Harris. *Testing spreadsheets and other packages used in metrology: Testing the intrinsic functions of Excel*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 27/99, September 1999.

[7] J. Barrett and M.P. Dainton. *Testing spreadsheets and other packages used in metrology: Testing the intrinsic functions of MathCAD*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 05/00, September 2000.

[8] J. Barrett and M.P. Dainton. *Testing spreadsheets and other packages used in metrology: Testing the intrinsic functions of S-PLUS*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 06/00, September 2000.

[9] M.G. Cox, M.P. Dainton and P.M. Harris. *Testing spreadsheets and other packages used in metrology: Testing functions for linear regression*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 08/00, October 2000.

[10] B.D. McCullough and Berry Wilson. On the accuracy of statistical procedures in Microsoft Excel 97. *Computational Statistics and Data Analysis*, **31**, pp 27-37, 1999.

[11] B.D. McCullough. Assessing the reliability of statistical software: Part I. *The American Statistician*, **52**(4), pp 358-366, 1998.

[12] B.D. McCullough. Assessing the reliability of statistical software: Part I. *The American Statistician*, **53**(2), pp 149-159, 1999.

[13] J. Rogers, J. Filliben, L. Gill, W. Guthrie, E. Lagergren and M. Vangel. StRD: statistical reference datasets for assessing the numerical accuracy of statistical software. *NIST TN #1396*, National Institute of Standards and Technology, USA.

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

[15] The NAG Fortran Library Manual (Mark 19), Numerical Algorithms Group.

[16] Matlab Version 5.3.0.10183 (R11). Copyright ©1984-1999, The MathsWork Inc.

[17] Microsoft Excel 2000. Copyright © 1985-1999 Microsoft Corporation.

[18] S-Plus 4.0 Release 3 for Windows Copyright ©1988-1997, MathSoft Inc.

[19] MathCad Version 8, Professional. Copyright © 1986-1998, MathSoft Inc.

[20] M.G. Cox and P.M. Harris. *Guidelines to help users select and use software for their metrology applications*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 04/00, September 2000.

[21] T.F. Chan and J.G. Lewis. Computing standard deviations: accuracy. *Comm. ACM*, **22**(9), 1979.

Appendix

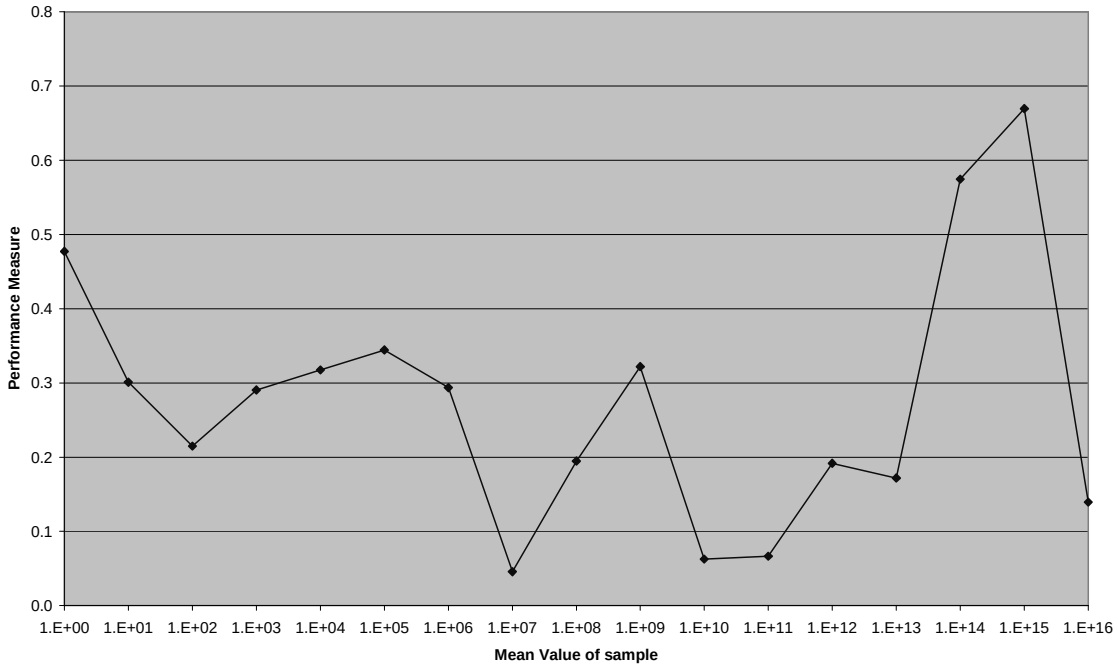

Figure 1: Plot of the performance measure $P(s)$ for the standard deviation against the mean for the **IMSL** subroutine **DUVSTA**.

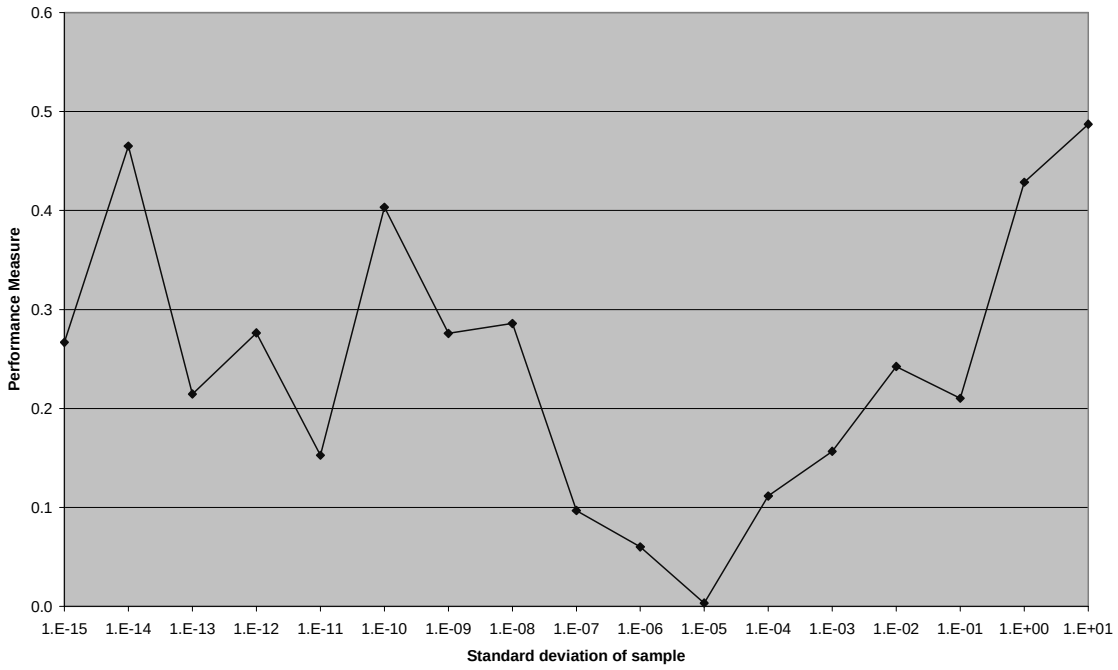

Figure 2: Plot of the performance measure $P(s)$ for the standard deviation against the standard deviation of the sample for the **IMSL** subroutine **DUVSTA**.

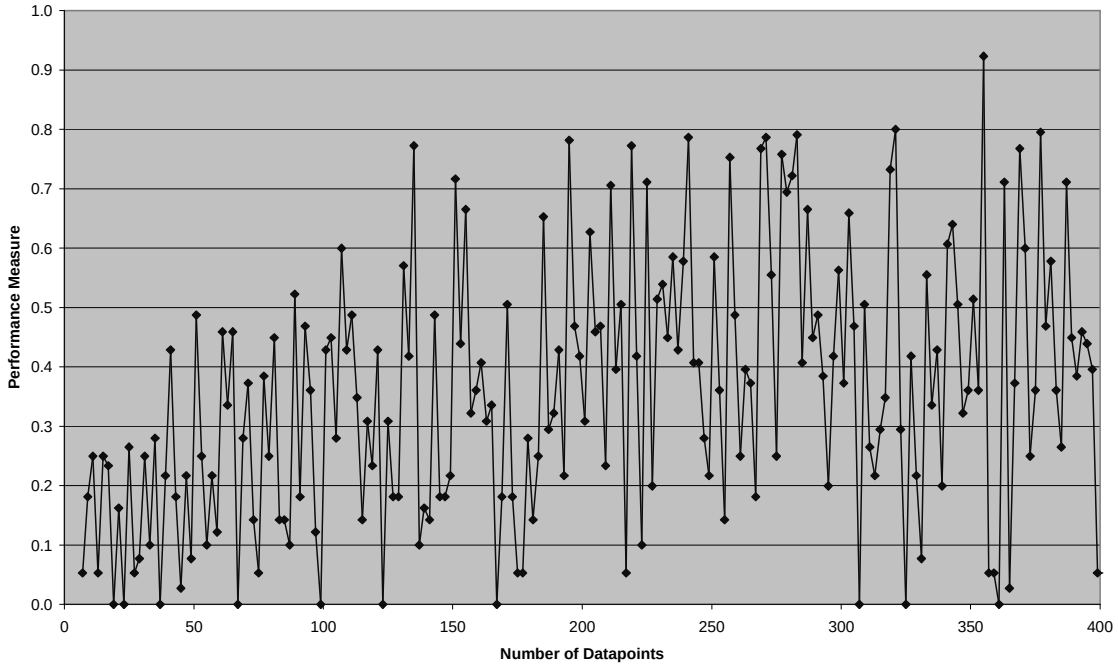

Figure 3: Plot of the performance measure $P(s)$ for the standard deviation against the sample size for the **IMSL** subroutine **DUVSTA**.

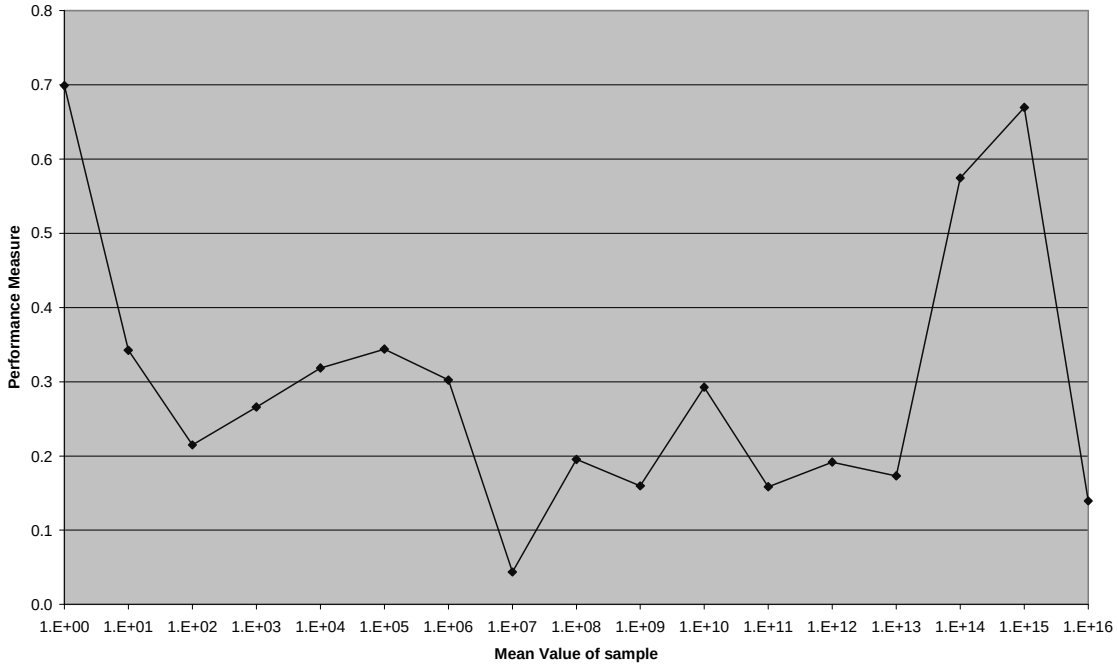

Figure 4: Plot of the performance measure $P(s)$ for the standard deviation against the mean for the **NAG** subroutine **G01AAF**.

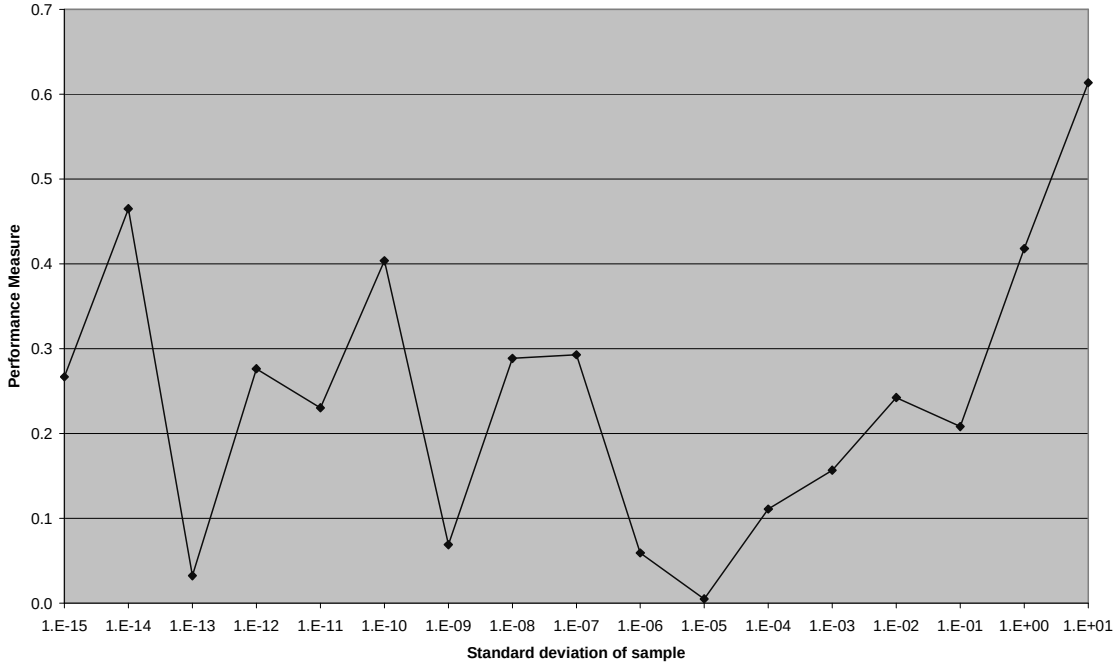

Figure 5: Plot of the performance measure $P(s)$ for the standard deviation against the standard deviation of the sample for the **NAG** subroutine **G01AAF**.

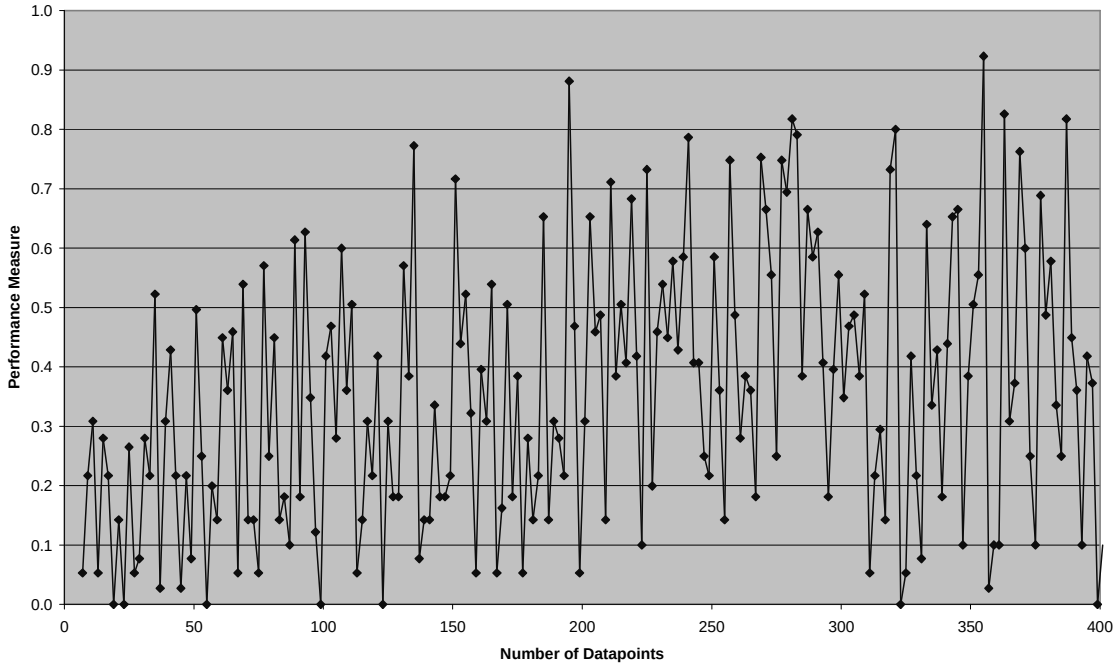

Figure 6: Plot of the performance measure $P(s)$ for the standard deviation against the sample size for the **NAG** subroutine **G01AAF**.

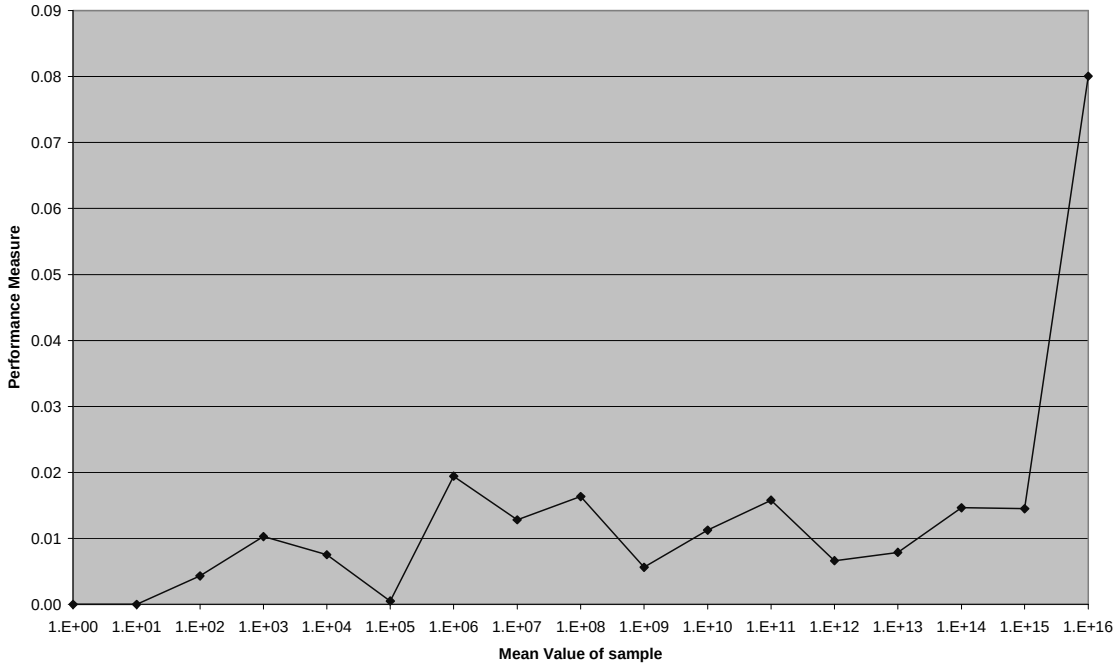

Figure 7: Plot of the performance measure $P(s)$ for the standard deviation against the mean for the **Matlab** intrinsic function **STD**.

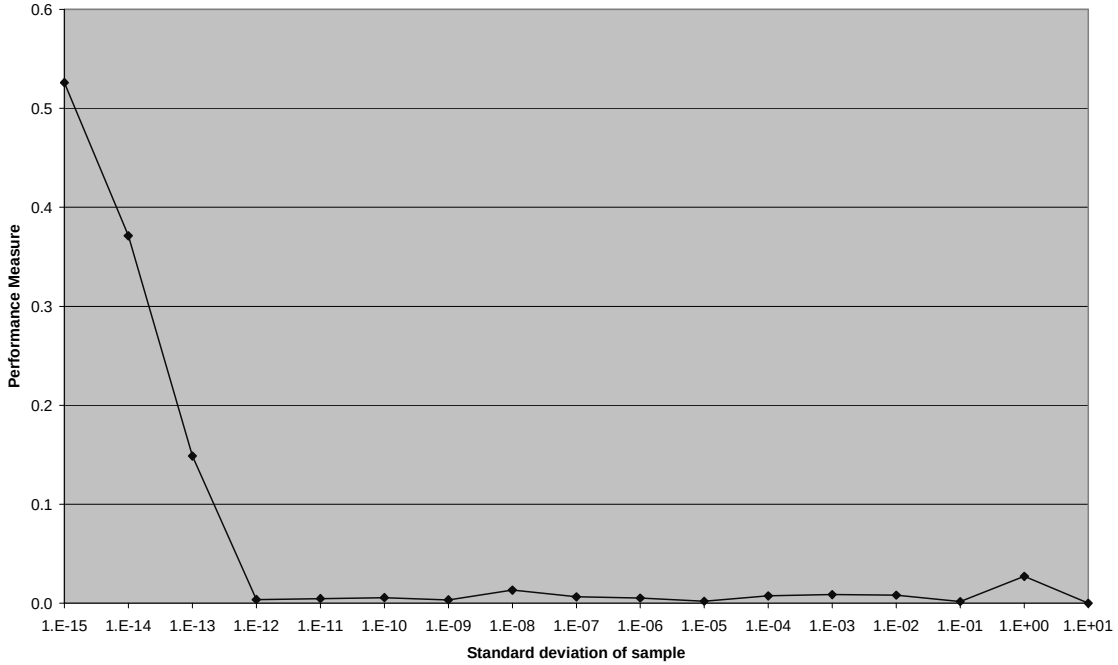

Figure 8: Plot of the performance measure $P(s)$ for the standard deviation against the standard deviation of the sample for the **Matlab** intrinsic function **STD**.

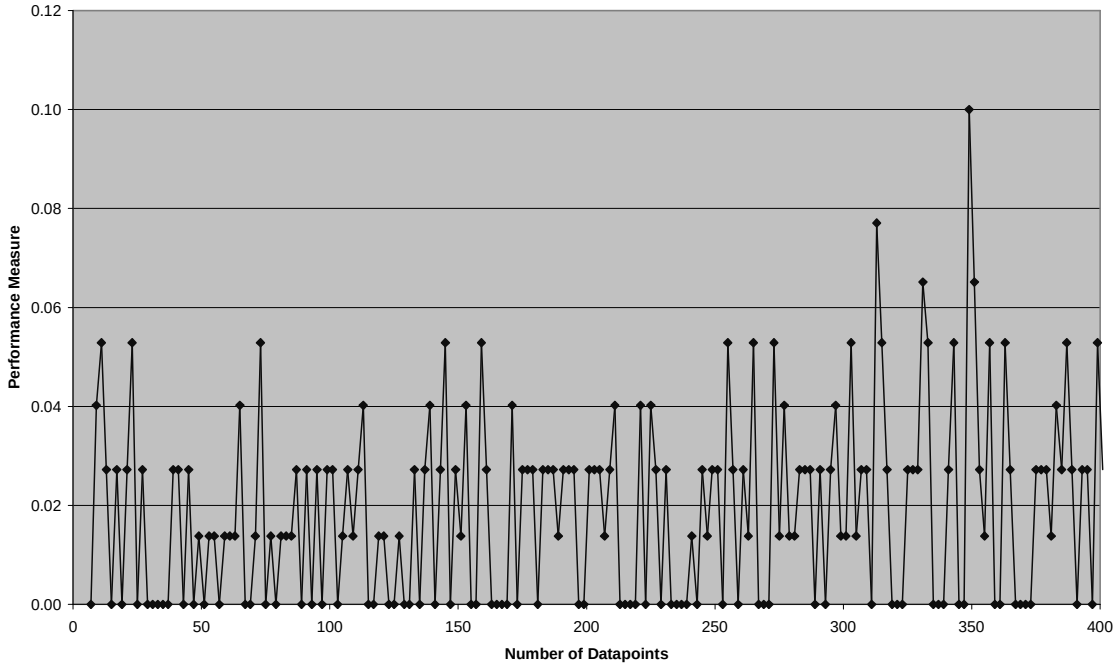

Figure 9: Plot of the performance measure $P(s)$ for the standard deviation against the sample size for the **Matlab** intrinsic function **STD**.

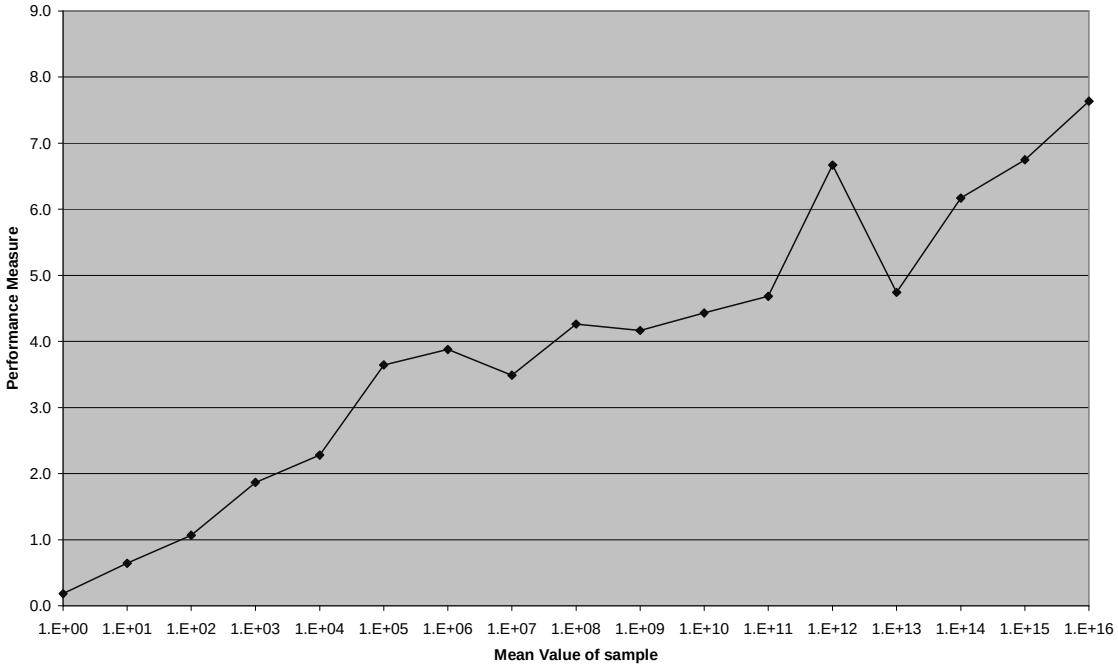

Figure 10: Plot of the performance measure $P(s)$ for the standard deviation against the mean for the **Excel** intrinsic function **STDEV**.

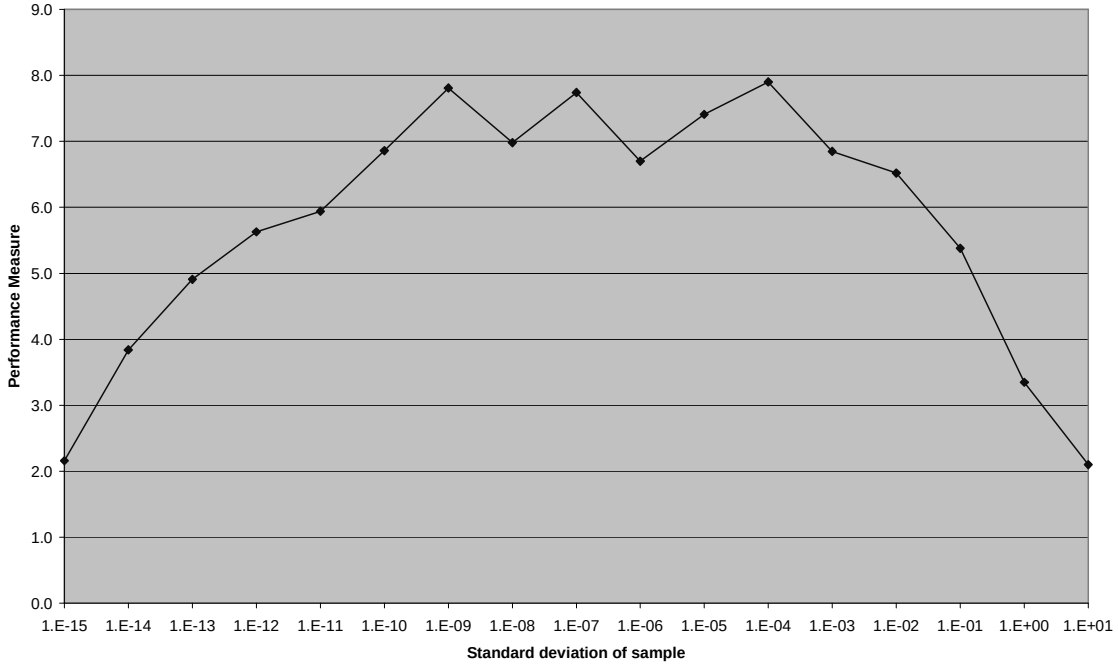

Figure 11: Plot of the performance measure $P(s)$ for the standard deviation against the standard deviation of the sample for the **Excel** intrinsic function **STDEV**.

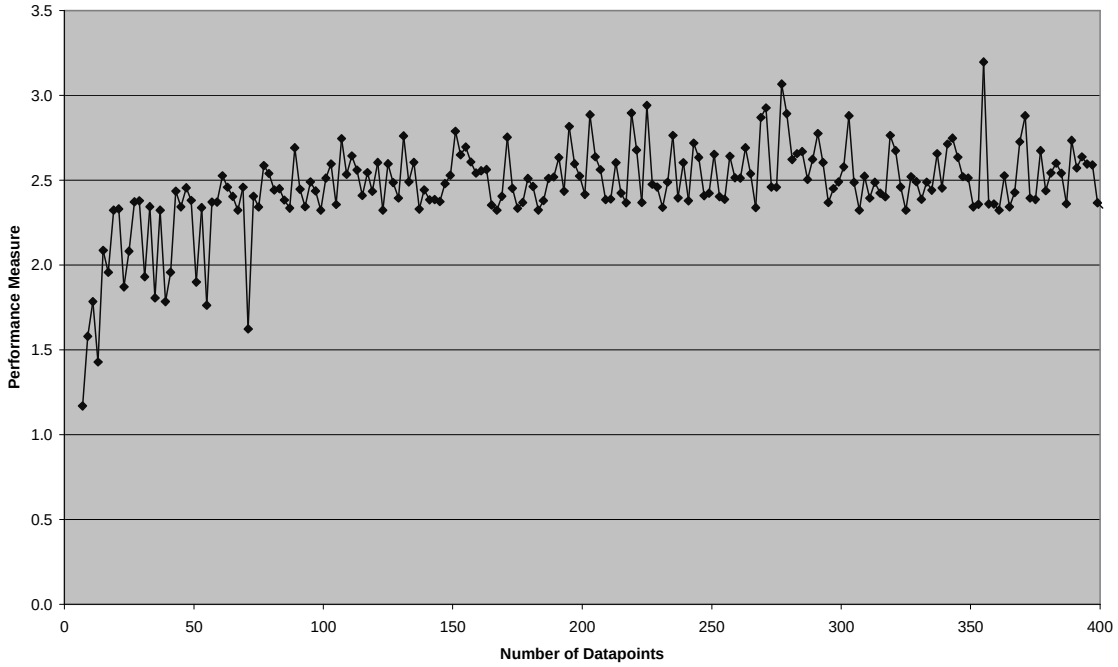

Figure 12: Plot of the performance measure $P(s)$ for the standard deviation against the sample size for the **Excel** intrinsic function **STDEV**.

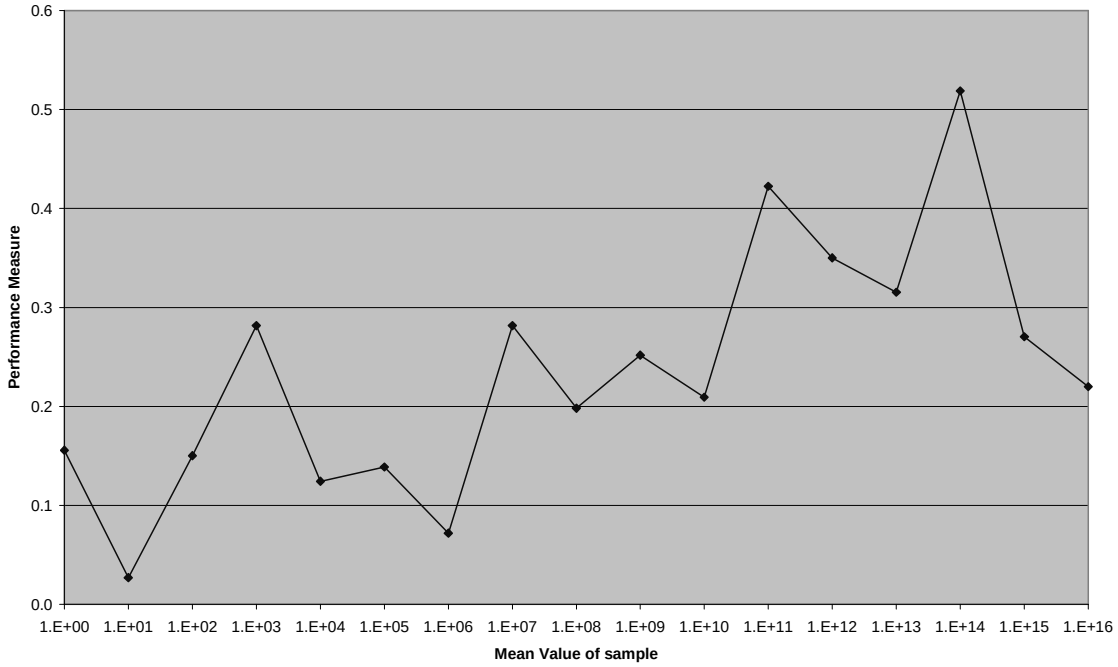

Figure 13: Plot of the performance measure $P(s)$ for the standard deviation against the mean for the **SPLUS** intrinsic function **VAR**.

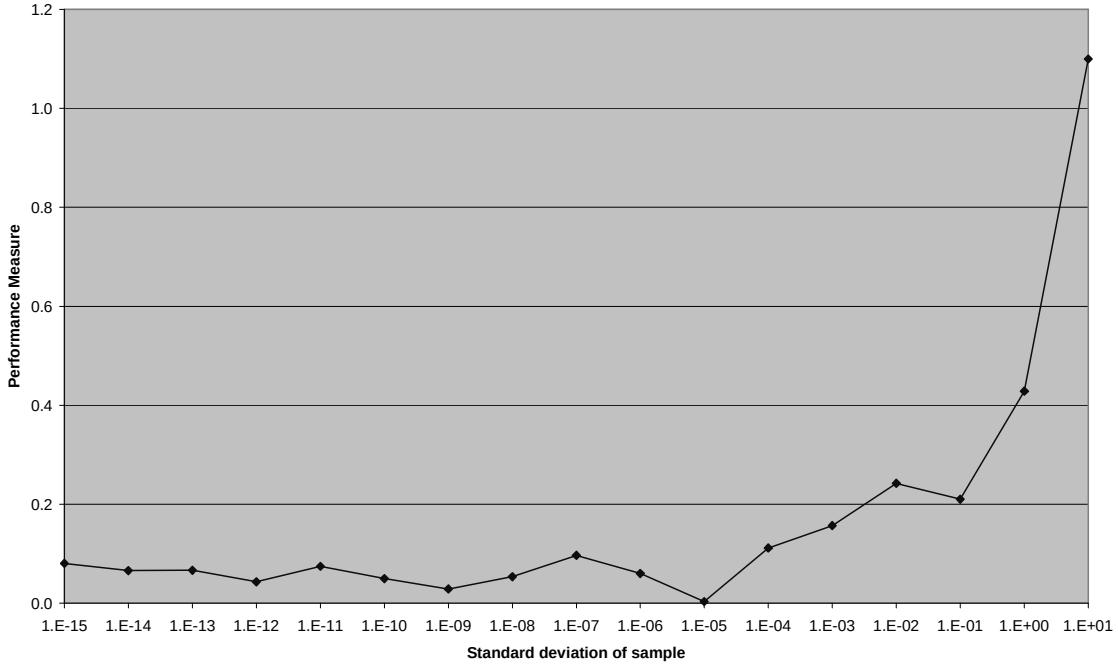

Figure 14: Plot of the performance measure $P(s)$ for the standard deviation against the standard deviation of the sample for the **SPLUS** intrinsic function **VAR**.

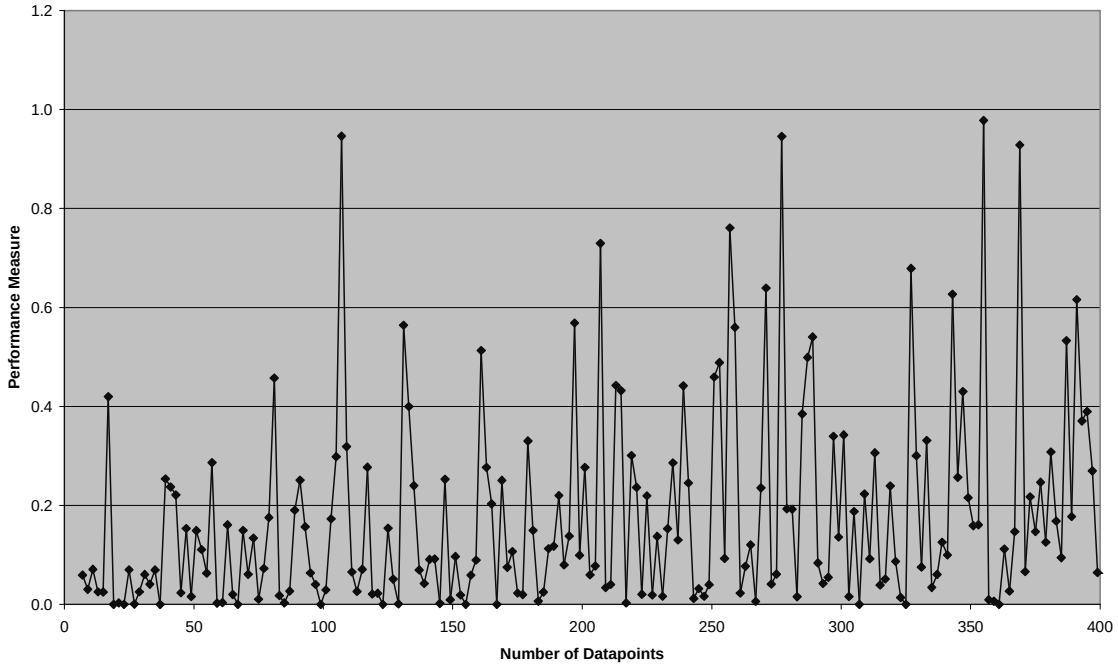

Figure 15: Plot of the performance measure $P(s)$ for the standard deviation against the sample size for the **SPLUS** intrinsic function **VAR**.

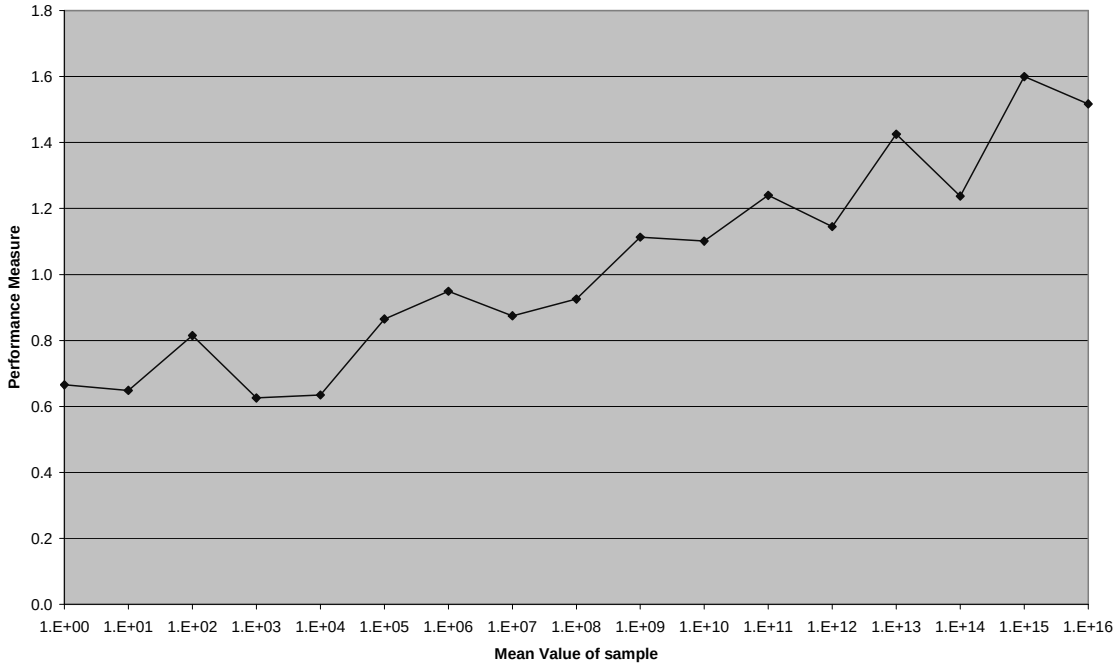

Figure 16: Plot of the performance measure $P(s)$ for the standard deviation against the mean for the **MathCAD** intrinsic function **STDEV**.

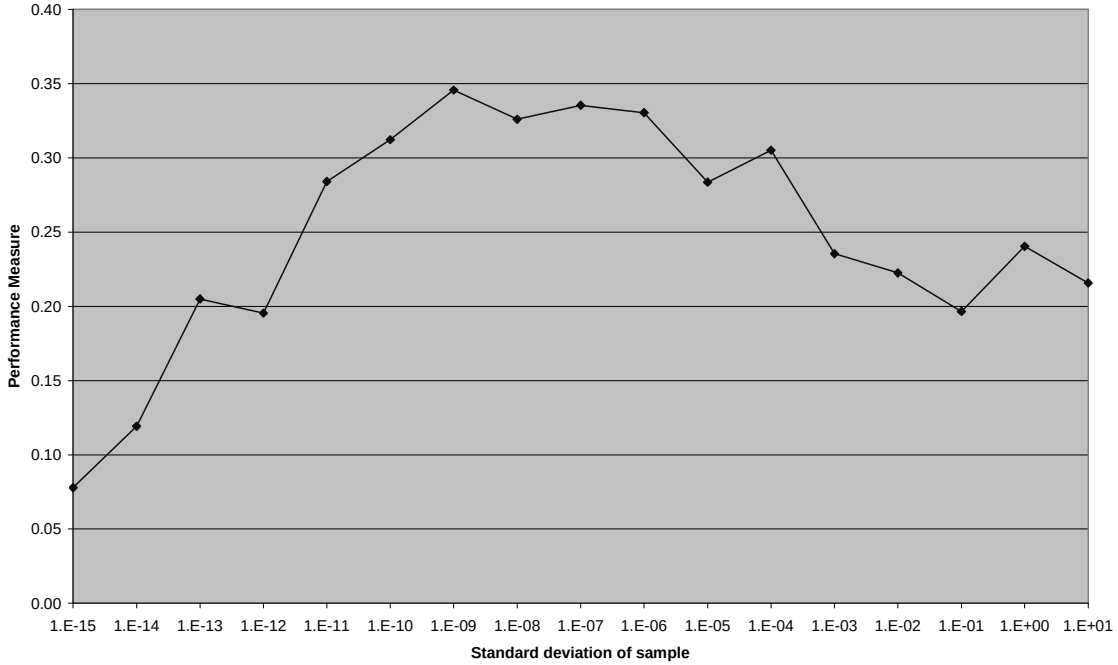

Figure 17: Plot of the performance measure $P(s)$ for the standard deviation against the standard deviation of the sample for the **MathCAD** intrinsic function **STDEV**.

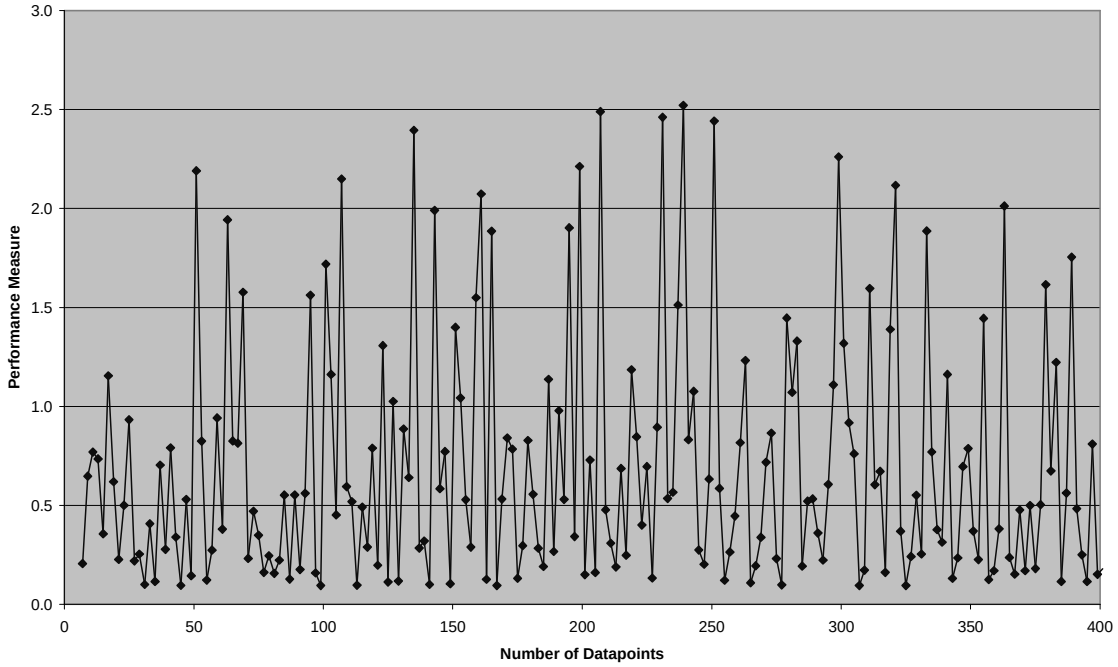

Figure 18: Plot of the performance measure $P(s)$ for the standard deviation against the sample size for the **MathCAD** intrinsic function **STDEV**.