NPL Report CMSC 08/00

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

TESTING SPREADSHEETS AND OTHER PACKAGES USED IN METROLOGY

TESTING FUNCTIONS FOR LINEAR REGRESSION

BY

M G COX, M P DAINTON and P M HARRIS

October 2000

Testing Spreadsheets and Other Packages Used in Metrology
Testing Functions for Linear Regression

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 linear regression (applied to the straight-line linear regression problem) 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 *straight-line linear regression*, defined as follows: given discrete data (x_i, y_i), $i = 1, \ldots, m$, and the straight-line linear model

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

the straight-line linear regression problem is to solve

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

or, equivalently,

$$\min_{\mathbf{b}} \left\| A\mathbf{b} - \mathbf{y} \right\|_2^2,$$

where the observation matrix A is

$$A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix},$$

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

The straight-line linear regression problem is a special case of the *linear regression* or *linear least-squares* problem

$$\min_{\mathbf{b}} \left\| A\mathbf{b} - \mathbf{y} \right\|_2^2 \equiv \min_{\mathbf{b}} \sum_{i=1}^{m} \left(\mathbf{a}_i^{\mathrm{T}}\mathbf{b} - y_i\right)^2,$$

where A is an $m \times n$ regression or observation matrix made up of row vectors $\mathbf{a}_i^{\mathrm{T}}$, $i = 1, \ldots, m$, with $m \geq n$, and $\mathbf{b}$ is an $n \times 1$ vector of regression parameters b_j to be determined. The linear regression problem is important to metrology because of its application to discrete modelling of measurement data.

The software packages covered in this work are Excel, MathCAD, S-PLUS, Matlab, the NAG Fortran library and the IMSL Fortran library. The functions considered from these packages address the general linear regression problem, but their testing is restricted to the special case of the straight-line model given before.

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 linear regression problem 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 mechanism used by NIST to generate these certified results is to undertake calculations using multiple or extended precision. Data is read in as multiple precision numbers and all calculations are made to a very high precision (accurate to 500 digits). The results are also output in multiple precision, and only then rounded to fifteen significant digits, which is the computational precision to which the majority of users will be working. The results represent what would be achieved if calculations were to be made without round-off or other errors. In [10, 11, 12] the certified results are compared with the results returned by test software by computing the base-10 logarithm of the relative error (or the base-10 logarithm of the absolute error in cases where the certified result is zero). The testing methodology applied in this work differs from the approach described in [10, 11, 12] both in the way reference data sets and results are generated and in the comparison of reference and test results. The approach to data generation does not require software for solving the straight-line regression problem but instead relies on *data generators* (see Sections 2.5 and 5). Furthermore, a *performance measure* is used to compare reference and test results that accounts for the loss of accuracy in the test results caused by the natural ill-conditioning of the problem represented by the data set as well as the relative error (see Sections 2.4 and 4).

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 measure 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 linear regression 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 that used to compute the problem degree of difficulty K in the definition of the performance measure: see Section 2.4), 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 linear regression problem 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:

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

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, $K(\mathbf{x})$ 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 linear regression problem 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 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 linear regression problem is given in Sections 6 and 7.

---

[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.

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 linear regression function tested in this work are those contained within the following packages:

- the IMSL Fortran library [15].
- the NAG Fortran library [16],
- Matlab [17],
- Microsoft Excel [18],
- S-PLUS 4.0 [19], and
- MathCAD [20].

3.1 IMSL Library

DLSQRR

Solve a linear least-squares problem without iterative refinement.

```
CALL DLSQRR (NRA, NCA, A, LDA, B, TOL, X, RES, KBASIS)
```

Parameters

NRA	Number of rows of A. (Input)
NCA	Number of columns of A. (Input)
A	NRA by NCA matrix containing the coefficient matrix of the least-squares system to be solved. (Input)
LDA	Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
B	Vector of length NRA containing the right-hand side of the least-squares system. (Input)
TOL	Scalar containing the nonnegative tolerance used to determine the subset of columns of A to be included in the solution. (Input)
	If TOL is zero, a full complement of min(NRA, NCA) columns is used.
X	Vector of length NCA containing the solution vector with components corresponding to the columns not used set to zero. (Output)
RES	Vector of length NRA containing the residual vector B – A*X. (Output)
KBASIS	Scalar containing the number of columns used in the solution. (Output)

3.2 NAG Library

F04JGF

Finds the solution of a linear least-squares problem, $Ax = b$, where A is a real m by n ($m \geq n$) matrix and b is an m element vector. If the matrix of observations is not full rank, then the minimal least-squares solution is returned.

```
SUBROUTINE F04JGF (M, N, NRA, B, TOL, SVD, SIGMA, IRANK, WORK,
          LWORK, IFAIL)
INTEGER    M, N, NRA, IRAMK, LWORK, IFAIL
real       A(NRA, N), B(M), TOL, SIGMA, WORK(LWORK)
LOGICAL    SVD
```

Parameters

M — On entry: *m*, the number of rows of *A* (M ≥ N).

N — On entry: *n*, the number of columns of *A* (1 ≤ N ≤ M).

A — On entry: the *m* by *n* matrix *A*.

On exit: if SVD is returned as .FALSE., A is overwritten by details of the *QU* factorisation of *A*. If SVD is returned as .TRUE., the first *n* rows of A are overwritten by the right-hand singular vectors, stored by rows; and the remaining rows of the array are used as workspace.

NRA — On entry: the first dimension of the array A as declared by the (sub)program from which F04JGF is called (NRA ≥ M).

B — On entry: the right-hand side vector *b*.

On exit: the first *n* elements of B contain the minimal least-squares solution vector *x*. The remaining *m* – *n* elements are used for workspace.

TOL — On entry: a relative tolerance to be used to determine the rank of *A*. TOL should be chosen as approximately the largest relative error in the elements of *A*. For example, if the elements of *A* are correct to about 4 significant figures then TOL should be set to about 5×10^{-4}. If TOL is outside the range (ε, 1.0), where ε is the machine precision, then the value ε is used in place of TOL. For most problems this is unreasonably small.

SVD — On exit: SVD is returned as .FALSE. if the least-squares solution has been obtained from the *QU* factorisation of *A*. In this case *A* is of full rank. SVD is returned as TRUE. If the least-squares solution has been obtained from the singular value decomposition of *A*.

SIGMA — On exit: the standard error, i.e., the value $\sqrt{r^{\mathrm{T}}r/(m-k)}$ when $m > k$, and the value zero when $m = k$. Here *r* is the residual vector $b - Ax$ and *k* is the rank of *A*.

IRANK — On exit: *k*, the rank of the matrix *A*. It should be noted that it is possible for IRANK to be returned as *n* and SVD to be returned as .TRUE.. This means that the matrix *U* only just failed the test for non-singularity.

WORK — Workspace.

LWORK — On entry: the dimension of the array WORK as declared in the (sub)program from which F04JGF is called (LWORK ≥ 4×N).

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 or M < N or NRA < M or LWORK < 4×N; IFAIL = 2 if the *QR* algorithm has failed to converge to singular values in 50×N iterations (this failure can only happen when the singular value decomposition is employed, but even then it is not likely to occur).

3.3 Matlab

\ (Backslash or left matrix divide.)

A\B is the matrix division of A into B, which is roughly the same as INV(A)*B, except it is computed in a different way. If A is an N-by-N matrix and B is a column vector with N components, or a matrix with several such columns, then X = A\B is the solution to the equation A*X = B computed by Gaussian elimination. A warning message is printed if A is badly scaled or nearly singular. A\EYE(SIZE(A)) produces the inverse of A.

If A is an M-by-N matrix with M < or > N and B is a column vector with M components, or a matrix with several such columns, then X = A\B is the solution in the least squares sense to the under- or overdetermined system of equations A*X = B. The effective rank, K, of A is determined from the QR decomposition with pivoting. A solution X is computed which has at most K nonzero components per column. If K < N this will usually not be the same solution as PINV(A)*B. A\EYE(SIZE(A)) produces a generalized inverse of A.

3.4 Excel

LINEST

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

$$y = m_1x_1 + m_2x_2 + \ldots + m_nx_n + b$$

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.

3.5 S-PLUS

LM

Returns an object that represents a fit of a linear model. The data to be fitted is stored in an S-PLUS data frame, and is called by the *formula* argument.

lm(*formula, data, weights, subset, na.action, method, model, x, y, contrasts, ...*)

REQUIRED ARGUMENTS

formula formula to be fitted, of the form:

$$y \sim f(x, a_1, \ldots, a_n)$$

where y denotes the column of the data frame in which the response data is stored, x denotes the column of the data frame in which the predictor data is stored, and $a_1, \ldots, a_n$ are the parameters to be estimated.

OPTIONAL ARGUMENTS

data data-frame containing the *x* and *y* data, and the data for the *subset* and *weights* arguments, if required. If this is missing, then the variables in the formula should be on the search list. This may also be a single number to handle some special cases - see below for details.

weights vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the squared residuals. The length of weights must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the subset argument.

subset expression saying which subset of the rows of the data should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), or a numeric vector indicating which observation

numbers are to be included, or a character vector of the row names to be included. All observations are included by default.

na.action a function to filter missing data. This is applied to the *model.frame* after any subset argument has been used. The default (with *na.fail*) is to create an error if any missing values are found. A possible alternative is *na.omit*, which deletes observations that contain one or more missing values.

method the least squares fitting method to be used; the default is "qr". The method "*model.frame*" simply returns the model frame.

model logical flag: if TRUE, the model frame is returned in component model.

x logical flag: if TRUE, the model matrix is returned in component *x*.

y logical flag: if TRUE, the response is returned in component *y*.

qr logical flag: if TRUE, the QR decomposition of the model matrix is returned in component *qr*.

contrasts a list giving contrasts for some or all of the factors appearing in the model formula. The elements of the list should have the same name as the variable and should be either a contrast matrix (specifically, any full-rank matrix with as many rows as there are levels in the factor), or else a function to compute such a matrix given the number of levels.

... additional arguments for the fitting routines (see *lm.fit* and the functions it calls). Two possibilities are *singular.ok*=T to instruct the fitting to continue in the presence of over-determined models, and tolerance (default 1e−07) to change the tolerance for determining when models are over-determined.

3.6 MathCAD

LINFIT

linfit(vx,vy,F) returns a vector containing the coefficients used to create a linear combination of the functions in **F** which best approximates the data in **vx** and **vy**.

Arguments:

vx is a vector of data values. These correspond to the *x* values. The elements must be in ascending order. Use the **sort** function if they are not.

vy is a vector of data values. These correspond to the *y* values. The number of elements is the same as **vx**.

F is a function that returns a vector whose elements are functions making up a linear function. Or in the case of a single linear function, **F** is a scalar.

4. Specification of Performance Parameters and Measures

Performance parameters for the linear regression problem include:

- the number *m* of points in the data set,
- the size *s* of measurement noise simulated in the generated data, and
- the location (i.e., median x^{med}) of the data *x*-values.

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

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

where $K(\mathbf{b})$ measures the problem degree of difficulty, and η is the machine precision [4]. When considering the problem degree of difficulty it is necessary to incorporate both the relative condition $\kappa_1(\mathbf{b})$ for the straight-line regression problem represented by the reference data set as well as the condition $\kappa_2(\mathbf{b})$ associated with the process of generating the reference data set. This can be done by defining an "overall" problem degree of difficulty $K(\mathbf{b})$ by

$$K(\mathbf{b}) = \kappa_1(\mathbf{b})\kappa_2(\mathbf{b}).$$

$\kappa_1(\mathbf{b})$ is calculated as the relative condition number of the column-scaled observation matrix A (and its value is validated by an empirical sensitivity analysis of the each regression problem). The necessity of including $\kappa_2(\mathbf{b})$ in the evaluation of the problem degree of difficulty $K(\mathbf{b})$, and the estimation of $\kappa_2(\mathbf{b})$, is considered in Section 5 as part of the discussion of the data generation procedure.

The problem degree of difficulty is to be interpreted in the following way: $\log_{10} K(\mathbf{b})$ can be expected to bound the number of significant figures of accuracy lost by a reference algorithm for solving the linear regression problem. The performance measure $P(\mathbf{b})$ 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 number of sample points, $K(\mathbf{b})$ is of the order of unity,
- for the sequence generated by varying the noise in the data, $K(\mathbf{b})$ is of the order of unity, and
- for the sequence generated by varying the median of the data x-values, $K(\mathbf{b})$ varies from approximately unity to 10^{14}.

This information is useful when interpreting graphs of the performance measure against a performance parameter: see Section 6.

Performance parameter	*Nominal value*	*Range*
Number m of points	50	[3, 100]
Size s of measurement noise	0.1	$[10^{-16}, 10^{2}]$
Median x^{med} of data x-values	0	$[1, 10^{7}]$

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 reference results. We also discuss the necessity for including a contribution to the problem degree of difficulty $K(\mathbf{b})$ (Section 4) arising from the process of data generation. It is important to bear in mind that the procedure does not make use of reference software but instead implements an approach for generating a reference data set (x_i, y_i), $i = 1, \ldots, m$, for which the reference result $\mathbf{b}^{\text{ref}}$ is specified *a priori* and for which the performance parameters (number of points, size of measurement noise and median of data x-values) take prescribed values.

The basis of the approach is the observation that the solution to the straight-line linear regression problem defined in Section 1 is characterised by the condition

$$A^{\mathrm{T}}\mathbf{r} = \mathbf{0}, \tag{1}$$

where $\mathbf{r}$ denotes the residual vector

$$\mathbf{r} = A\mathbf{b} - \mathbf{y} \tag{2}$$

at the solution [14][2]. Since

$$A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix},$$

equation (1) implies two conditions, one that

$$\sum_{i=1}^{m} r_i = 0,$$

i.e., the sum of the residuals is zero, and the other that

$$\sum_{i=1}^{m} x_i r_i = 0,$$

i.e., the first moment (with respect to the abscissa values) of the residuals is zero.

Let N be a matrix whose columns form a basis for the null space of A^{T}, i.e.,

$$A^{\mathrm{T}}N = \mathbf{0}.$$

Then, for a vector

$$\mathbf{v} = N\mathbf{u},$$

for *any* choice of the vector $\mathbf{u}$, the replacement of $\mathbf{y}$ by $\mathbf{y} + \mathbf{v}$ will leave the solution vector $\mathbf{b}$ *unchanged*. This result can be seen as follows. The condition (1) and the model (2) imply the *normal equations*

$$A^{\mathrm{T}}A\mathbf{b} = A^{\mathrm{T}}\mathbf{y}.$$

But, from the definition of the matrix N,

$$A^{\mathrm{T}}(\mathbf{y} + \mathbf{v}) = A^{\mathrm{T}}\mathbf{y} + A^{\mathrm{T}}N\mathbf{u} = A^{\mathrm{T}}\mathbf{y}.$$

[2] In fact, the characterisation (1) and (2) applies for the *general* linear regression problem but we only consider here its application to the generation of reference pairs for the *straight-line* linear regression problem.

Thus, from any one data set any number of further data sets having the same solution can readily be constructed by choosing vectors **u**.

A procedure for generating a reference pair for the straight-line regression problem based on the above characterisation is now given. This is the procedure that has been employed for the generation of reference pairs for the testing described in this report. The procedure requires that a reference solution $\mathbf{b}^{\text{ref}}$ and values for the performance parameters (number m of points, size s of measurement noise and median x^{med} of the data x-values) are prescribed.

1. Form m values x_i such that the median of the set $\{x_i: i = 1, \ldots, m\}$ is the prescribed value x^{med}. This is done by choosing m values equispaced in an interval centred on x^{med}.
2. Evaluate the observation matrix
$$A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix}.$$
3. Evaluate the observation vector $\mathbf{y}_0$ given by $\mathbf{y}_0 = A\mathbf{b}^{\text{ref}}$. The data defined by **x** and $\mathbf{y}_0$ satisfies the linear model defined by $\mathbf{b}^{\text{ref}}$ and, consequently, the solution to the straight-line linear regression problem for this data is $\mathbf{b}^{\text{ref}}$.
4. Form the null space matrix N for A^{T}. This is obtained from the singular value decomposition (SVD) of the matrix A^{T} [14].
5. Form the vector **v** given by $\mathbf{v} = N\mathbf{u}$, where the elements of **u** are chosen to be random numbers from a (standardised) Gaussian distribution with zero mean and unit standard deviation.
6. Form the residual vector **r** by scaling the elements of **v** so that the standard deviation of the set $\{r_i: i = 1, \ldots, m\}$ is s.
7. Form the observation vector **y** given by $\mathbf{y} = \mathbf{y}_0 + \mathbf{r}$. The data defined by **x** and **y** is such that the solution to the straight-line linear regression problem for this data is the same as for the data defined by **x** and $\mathbf{y}_0$, i.e., $\mathbf{b}^{\text{ref}}$.

It is straightforward to apply the above procedure to generate sequences of data sets by varying any of the performance parameters listed in Table 1. We now analyse the procedure and, in particular Step 7 of the procedure, in order to understand the effect on the problem degree of difficulty of the data generation process.

For the sake of argument, let the reference values for the regression parameters be $\mathbf{b}^{\text{ref}} = (1, 1)^{\text{T}}$, and suppose data x-values **x** are chosen to lie between −1 and +1 with median $x^{\text{med}} = 0$. We form an observation matrix A corresponding to **x**, a vector of model values

$$\mathbf{y}_0 = A\mathbf{b}^{\text{ref}},$$

and a residual vector **r** in the null-space of A^{T}. Then, the data y-values **y** are formed from

$$\mathbf{y} = \mathbf{y}_0 + \mathbf{r}.$$

This construction ensures that the solution to the straight-line linear regression problem for the data (**x**, **y**) is $\mathbf{b}^{\text{ref}}$. Since the observation matrix A is well-conditioned, as a result of the choice of data x-values, the residual vector **r** is accurately determined. In addition, for a sensibly scaled problem, as we have here, there is negligible loss of accuracy in the computation of **y** from $\mathbf{y}_0$ and **r**.

Now suppose we wish to generate, for the same reference solution, reference data with data x-values $\mathbf{x}_k$ with median $x^{\text{med}} = 10^k$. We form $\mathbf{x}_k$ by shifting the elements of $\mathbf{x}$ by 10^k, and forming an observation matrix A_k corresponding to $\mathbf{x}_k$, a vector of model values

$$\mathbf{y}_0 = A_k \mathbf{b}^{\text{ref}},$$

and a vector of data y-values

$$\mathbf{y}_k = \mathbf{y}_0 + \mathbf{r}.$$

Here, the scaling of the problem, i.e., the relative sizes of the elements of $\mathbf{y}_0$ and $\mathbf{r}$, is such that the residual vector $\mathbf{y}_k - \mathbf{y}_0$ for the data *computed* in this way will differ from the intended residual vector $\mathbf{r}$ in approximately the last $\log_{10}(x^{\text{med}}) = k$ significant figures.

To see this, consider the addition using six-digit decimal arithmetic of $y_0 = 273.184$ and $r = 0.426587$:

$$y_k = 273.184 + 0.426587 = 273.610587 \text{ which rounds to } 273.611.$$

Then, $y_k - y_0 = 273.611 - 273.184 = 0.427000$ which differs from 0.426587 in the last three significant figures, i.e. by approximately $\log_{10}(273)$.

In this sense, the reference data sets generated for each k represent different, or *perturbed*, problems although each is intended to have the same reference solution. This means that the "true" numerical condition of the problem defined for each k needs to reflect both the perturbation caused by the *process* of data generation as well as the natural conditioning of the problems so generated. Note that the latter contribution is described by $\kappa_1(\mathbf{b})$ and is computed from knowledge of the observation matrix A (see Section 4), whereas the former is represented by the term $\kappa_2(\mathbf{b})$ in the formula for the problem degree of difficulty $K(\mathbf{b})$ given in Section 4. $\kappa_2(\mathbf{b})$ depends on x^{med} and is estimated from

$$\kappa_2(\mathbf{b}) = \max\left\{x^{\text{med}}, 1\right\}.$$

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 (number of data points, noise size and median of the data x-values), and are arranged in the following way:

- IMSL subroutine **DLSQRR**: Figures 1–3,
- NAG subroutine **F04JGF**: Figures 4–6,
- Matlab "backslash" function: Figures 7–9,
- Excel function **LINEST**: Figures 10–12,
- S-PLUS function **LM**: Figures 13–15, and
- MathCAD function **LINFIT**: 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 straight-line linear regression problem.

For the sequences of data sets generated by varying the number of sample points and the amount of simulated measurement noise, the performance is generally good for all the test software. The values of the performance measure for IMSL, NAG, Matlab and S-PLUS are approximately unity or better, indicating that results having an accuracy comparable to that of reference software are returned. The values of the performance measure for MathCAD vary

between zero and three for the first of these sequences, indicating that up to three figures of accuracy in the test results are lost, and are approximately unity for the second sequence, indicating that accurate test results are returned. The values of the performance measure for Excel are consistently of the order of three, indicating that for these sequences of data sets about three figures of accuracy are lost compared to reference software.

For the sequence of data sets generated by varying the median of the data *x*-values, the packages appear to form two groups. For the first group, comprising IMSL, NAG, Matlab and S-PLUS, the values of the performance measure are less than one except for a small number of data sets with an exceptionally high problem degree of difficulty for which the values of the performance measure increase to a maximum value of approximately three. For the second group, comprising Excel and MathCAD, the performance profile exhibits a linear trend, indicating that these functions are losing a number of significant figures of accuracy over and above that which would be expected from reference software.

Implementation	Performance Parameter		
	Number m of points	Size s of noise	Median x^{med}
IMSL	< 1	< 1	< 1 (except for a small number of data sets for which $K(\mathbf{b})$ is exceptionally large and $P < 3$)
NAG	< 1	< 1	< 1 (except for a small number of data sets for which $K(\mathbf{b})$ is exceptionally large and $P < 3$)
Matlab	≈ 1 (except for a very small number of data sets for which $K(\mathbf{b})$ is approximately 2)	≈ 1 (and all < 1.5)	< 1 (except for a small number of data sets for which $K(\mathbf{b})$ is exceptionally large and $P < 3$)
Excel	≈ 3	≈ 3	Roughly a linear increase from < 1 for $x^{\text{med}} = 1$ to 8 for $x^{\text{med}} = 10^7$
S-PLUS	< 1	≈ 1 (and all < 1.5)	< 1 (except for a small number of data sets for which $K(\mathbf{b})$ is exceptionally large and $P < 3$)
MathCAD	< 3	≈ 1 (and all < 1.5)	Roughly a linear increase from < 1 for $x^{\text{med}} = 1$ to 8 for $x^{\text{med}} = 10^7$

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

We are able to conclude, therefore, that for this sequence of reference data sets the functions provided by IMSL, NAG, Matlab and S-PLUS behave very well, losing very few significant figures of accuracy compared to reference software (and only then in cases where the problem degree of difficulty is exceptionally high). On the other hand, the performance of the functions provided by MathCAD and Excel appears to degrade as a function of the location of the data *x*-values. The loss of accuracy of the test results returned by these packages is probably attributable to the algorithms they implement.

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 straight-line linear regression 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 results obtained and conclusions drawn from the testing undertaken must be interpreted in the context of the straight-line regression function only, and do not necessarily reflect on other functions of the software package or library from which that function is taken. The "black-box" testing described here has been carried out in such a way that the functions have been used without taking account of information elsewhere, 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. Furthermore, it allows for the consistent testing of the functions given the differing quality and quantity of information provided with each.

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 show that the IMSL, NAG, Matlab and S-PLUS packages provide reliable software for straight-line linear regression. However, some degradation in the performance of the functions provided by Excel and MathCAD is observed for a sequence of reference data sets generated by varying the median of the data *x*-values, and for these data sets the Excel and MathCAD functions lose a number of significant figures of accuracy over and above that which would be expected from reference software.

It is proposed that in subsequent work on linear regression software testing consideration be given to the issue discussed in Section 5, *viz*, that concerned with the deleterious effect on problem condition of the data generation procedure. One way of handling this aspect would be to generate null-space perturbations **r** in the ordinates **y** in such a manner that the formation of the graded data sets

$$(\mathbf{x}_k, \mathbf{y}_k) = \left(\mathbf{x} + 10^k, A_k\mathbf{b}^{\text{ref}} + \mathbf{r}\right)$$

did not result in the loss of up to *k* figures in $\mathbf{y}_k$.

The formation of $A_k\mathbf{b}^{\text{ref}}$ is exact for the *k*-values used and that of **r** can be expected to be close to full machine accuracy since it is obtained using a matrix *N* in the null space of A^{T}, a perfectly conditioned matrix. As discussed in Section 5, the problem is the addition of an accurate **r** to an accurate $A_k\mathbf{b}^{\text{ref}}$ where

$$\|\mathbf{r}\| << \|A_k \mathbf{b}^{\text{ref}}\|.$$

Suppose therefore that **r** could be produced in such a way that the addition were carried out exactly. Some discussion of an approach to achieve this, based on starting with an observation matrix with integral elements and integral reference parameters $\mathbf{b}^{\text{ref}}$, is given in [2, Section 16.6.3].

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 the calculation of standard deviation*. Report to the National Measurement System Policy Unit, Department of Trade and Industry, from the UK Software Support for Metrology Programme. NPL Report CISE 07/00, 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 II. *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] G.H. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, USA, 1996. Third edition.

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

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

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

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

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

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

Appendix

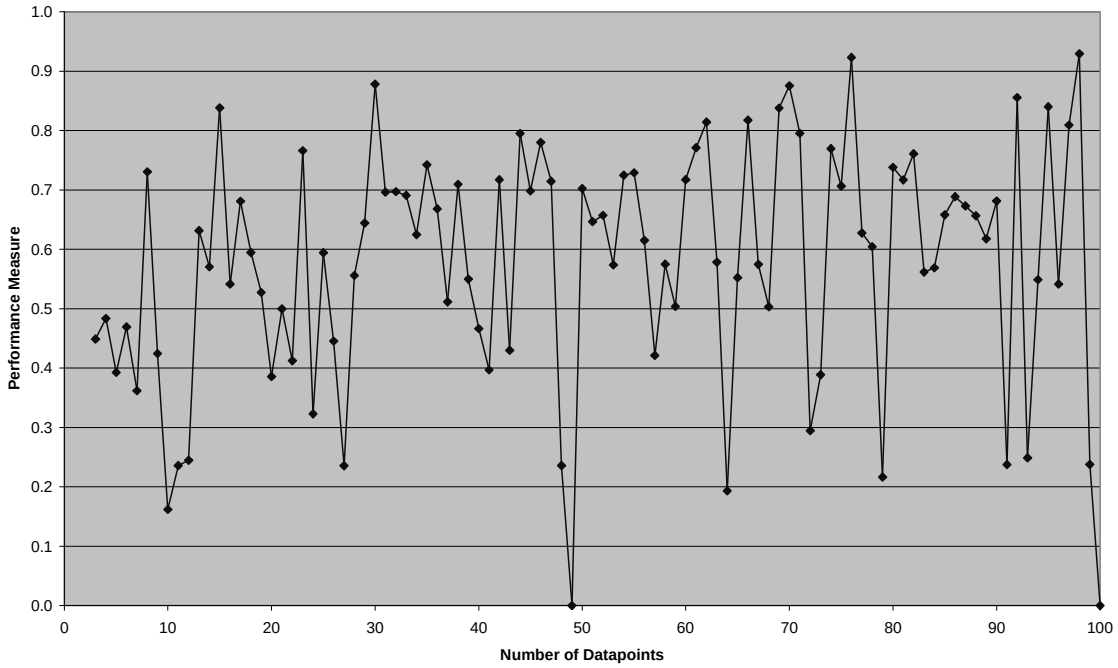

Figure 1: Plot of the performance measure *P*(**b**) for the computed regression coefficients against the number of data points for the **IMSL** subroutine **DLSQRR**.

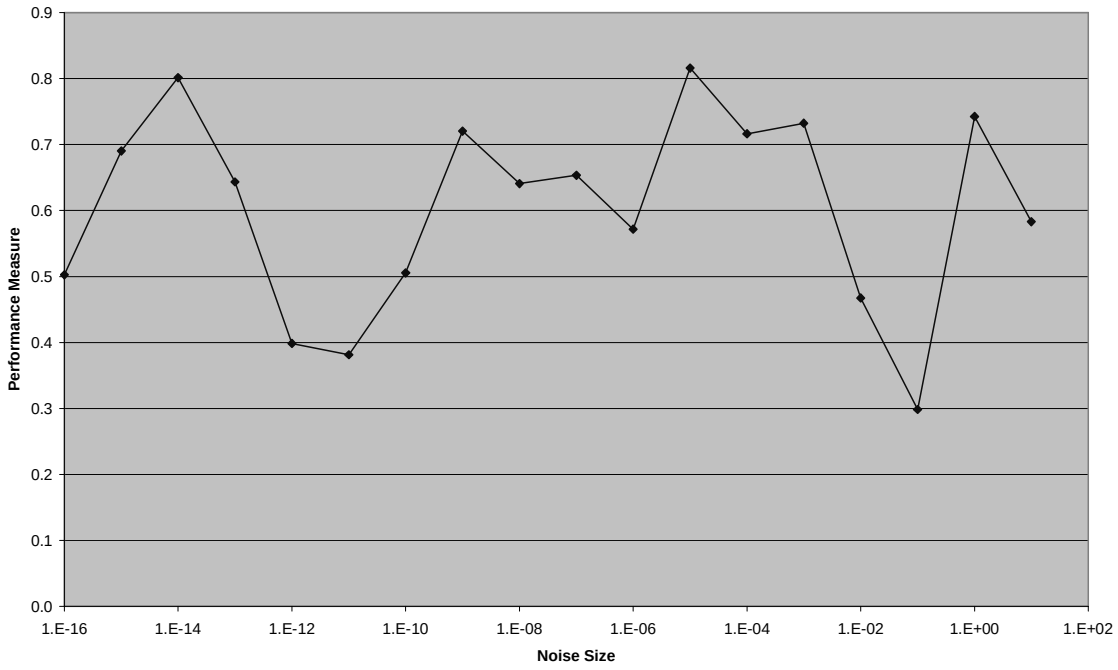

Figure 2: Plot of the performance measure *P*(**b**) for the computed regression coefficients against the noise size for the **IMSL** subroutine **DLSQRR**.

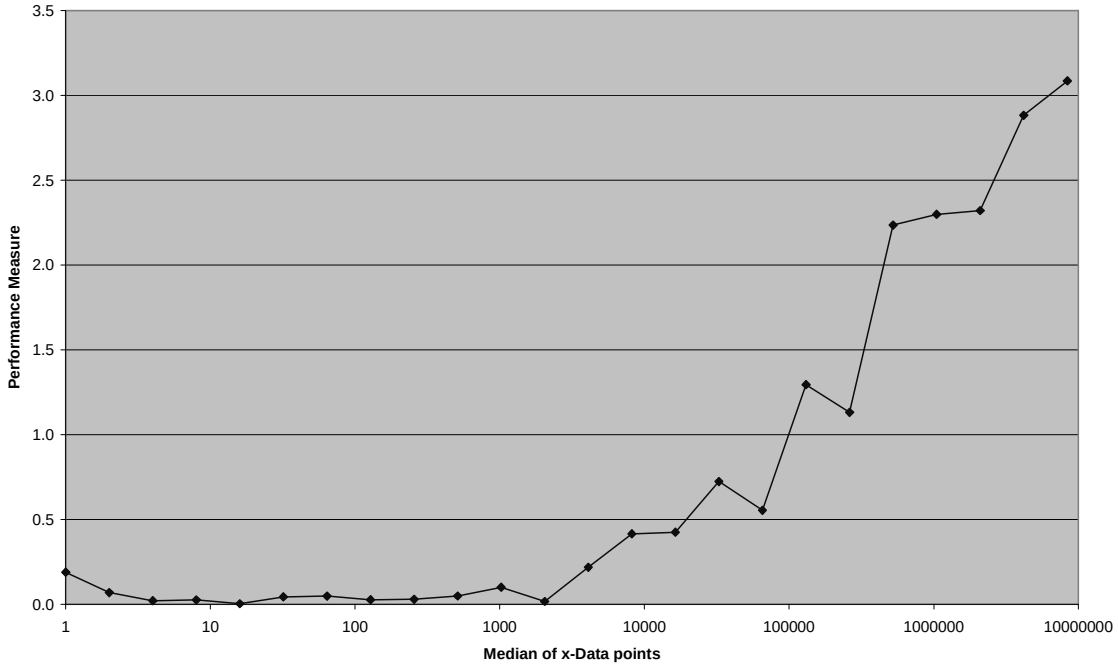

Figure 3: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the median of the data points for the **IMSL** subroutine **DLSQRR** .

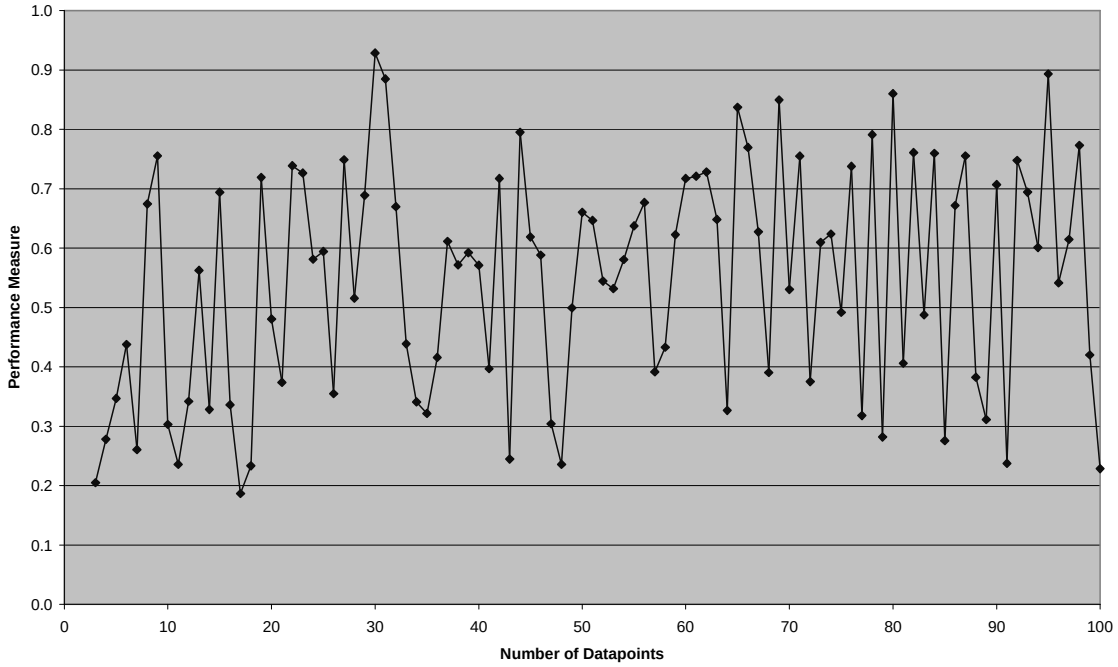

Figure 4: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the number of data points for the **NAG** subroutine **F04JGF**.

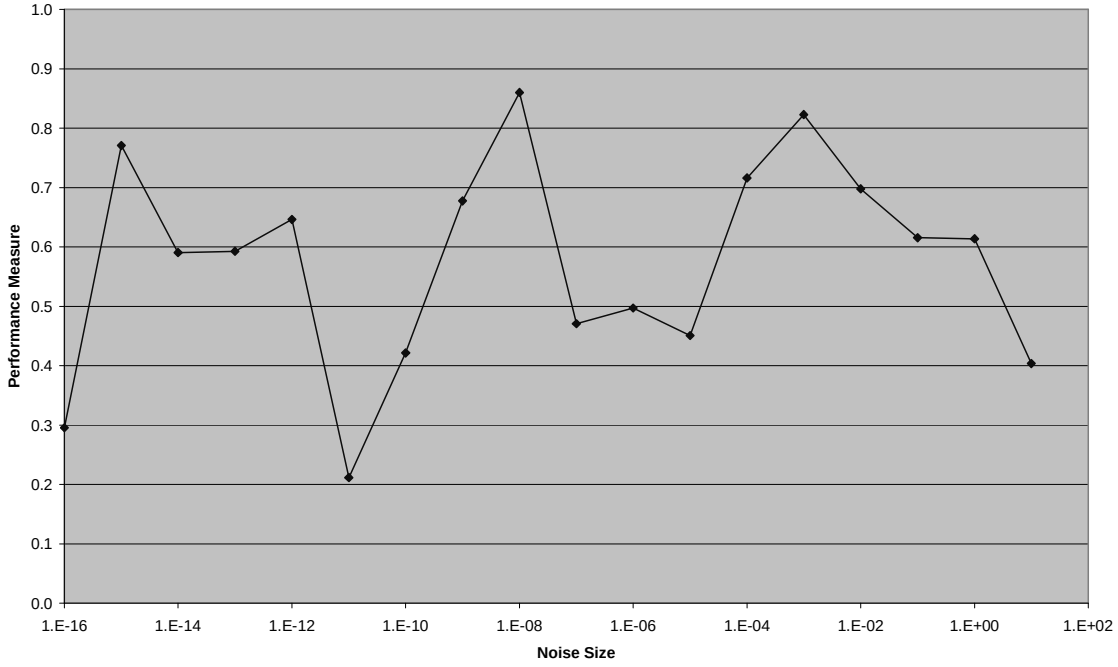

Figure 5: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the noise size for the **NAG** subroutine **F04JGF**.

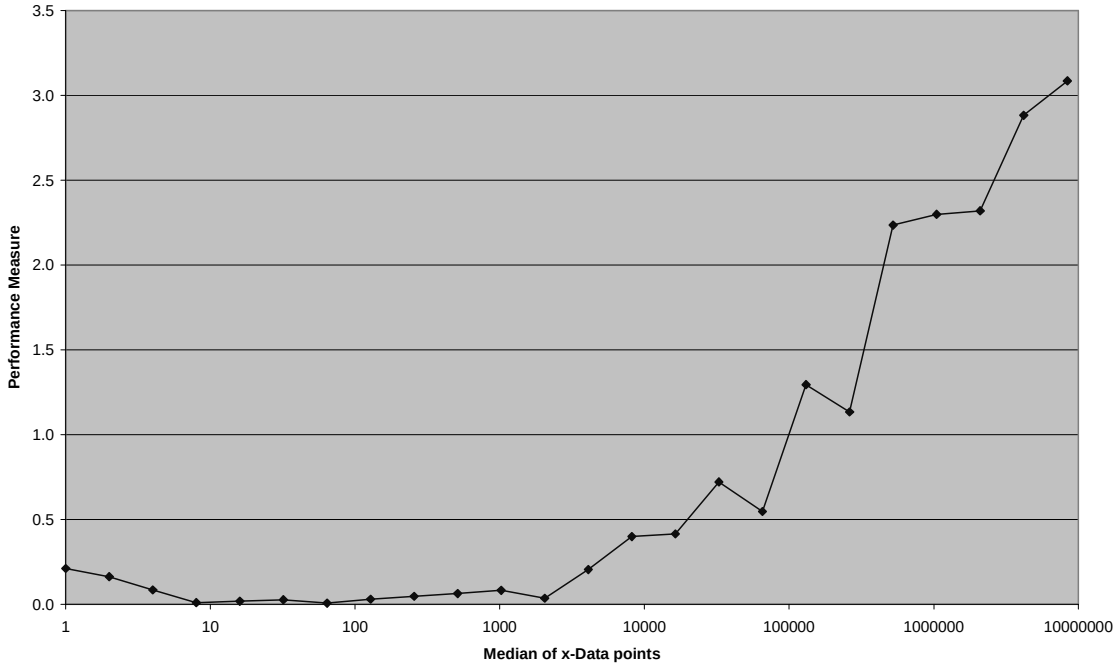

Figure 6: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the median of the data points for the **NAG** subroutine **F04JGF**.

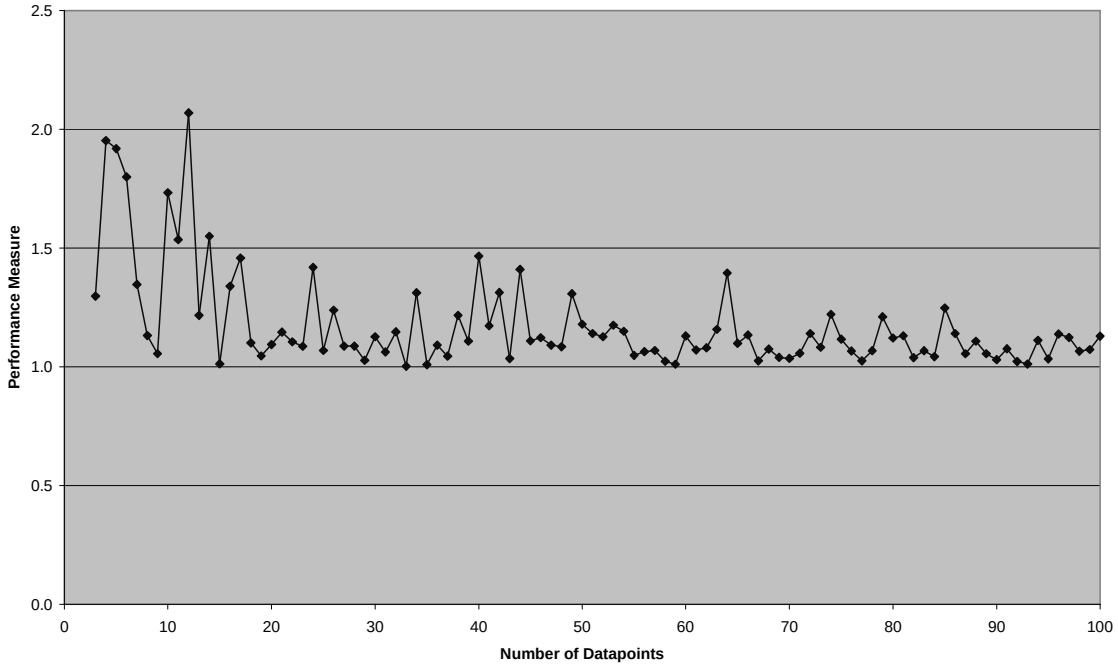

Figure 7: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the number of data points for the **Matlab** intrinsic function \ (backslash).

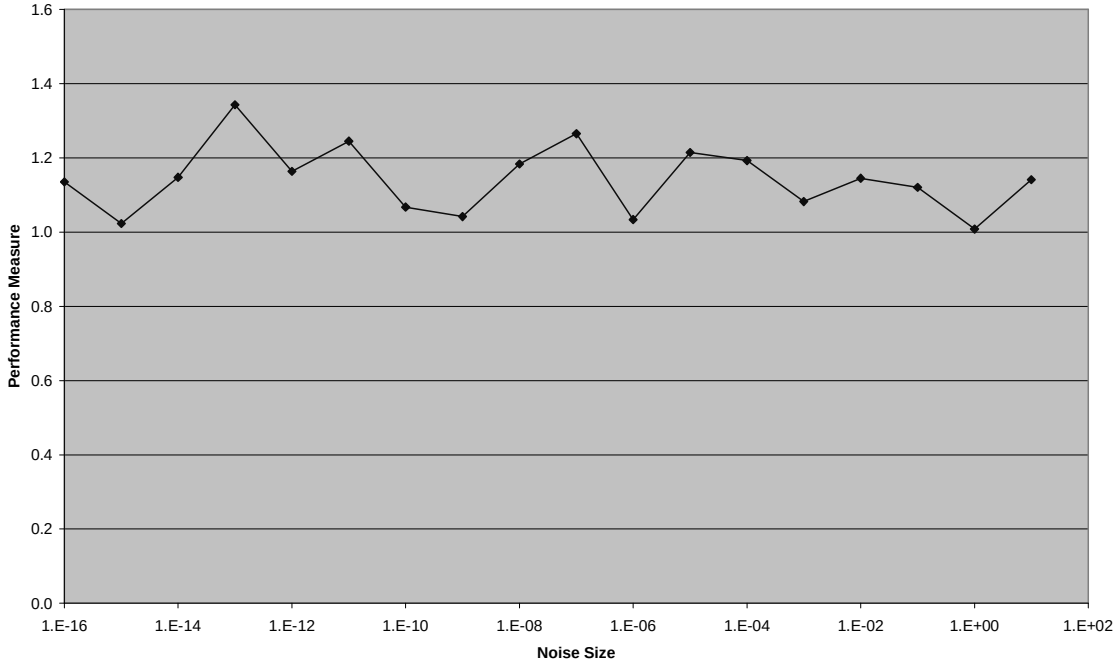

Figure 8: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the noise size for the **Matlab** intrinsic function \ (backslash).

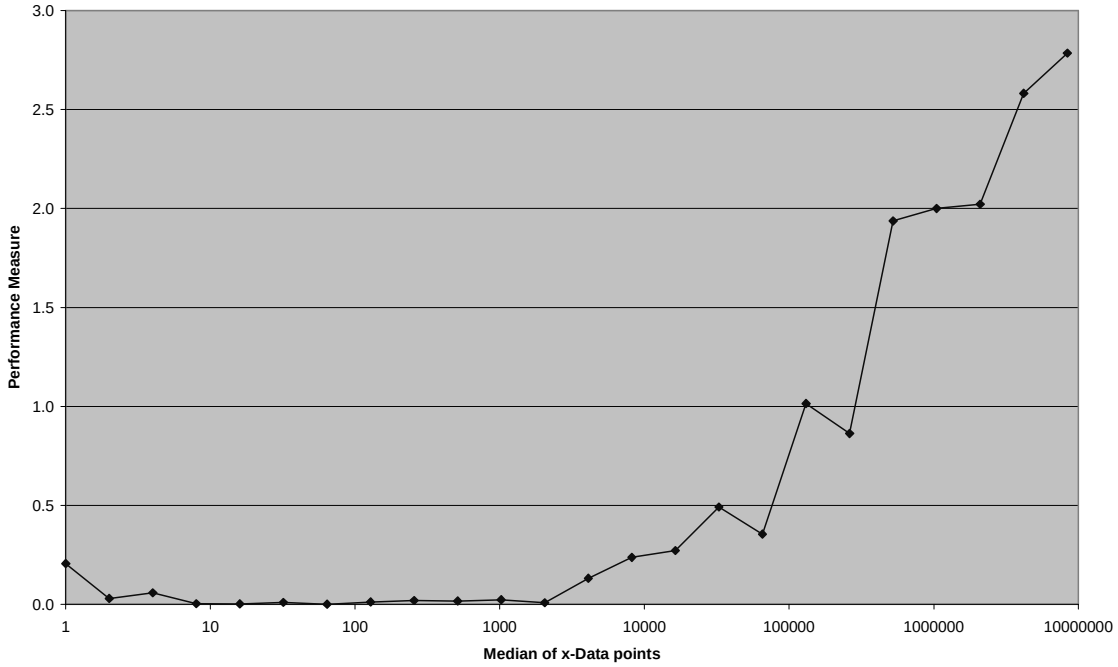

Figure 9: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the median of the data points for the **Matlab** intrinsic function \ (backslash).

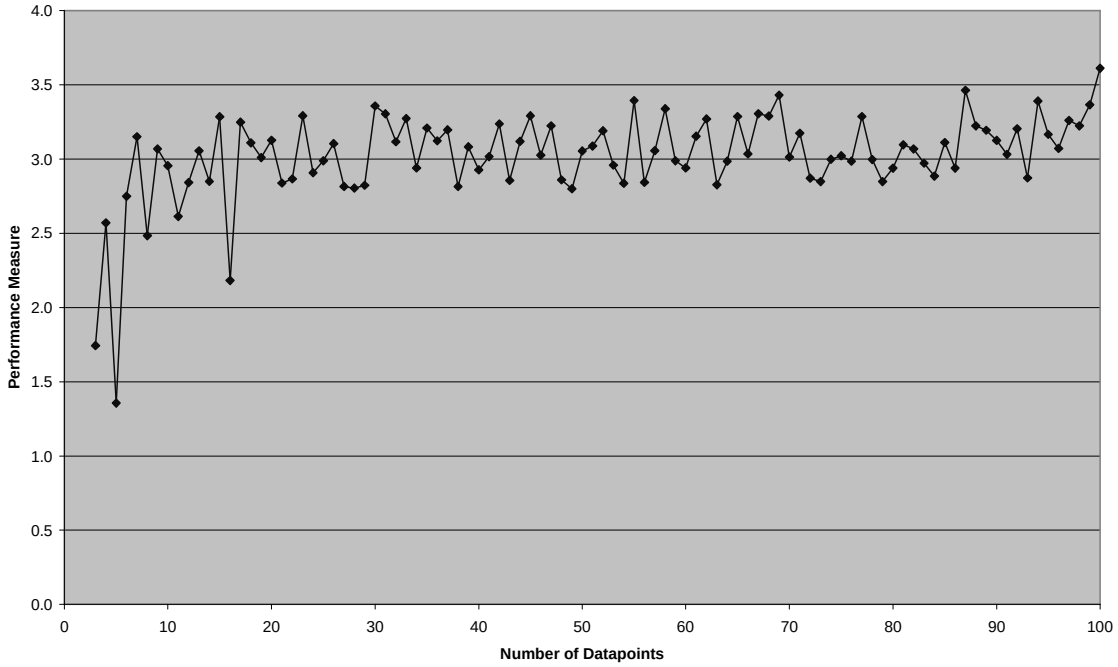

Figure 10: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the number of data points for the **Excel** intrinsic function **LINEST**.

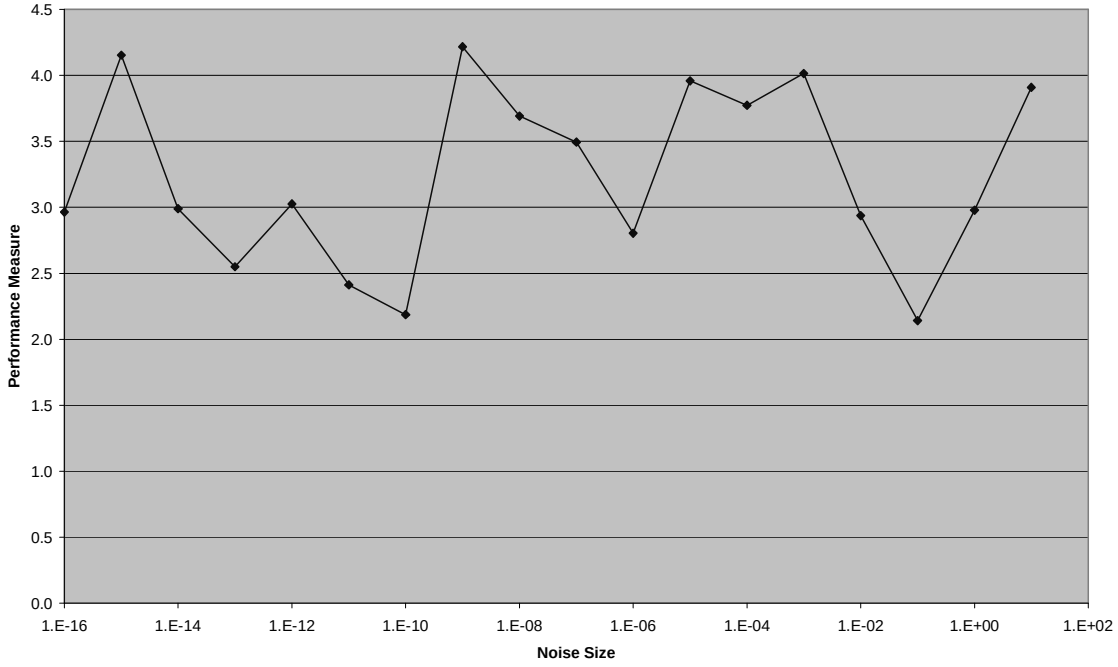

Figure 11: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the noise size for the **Excel** intrinsic function **LINEST**.

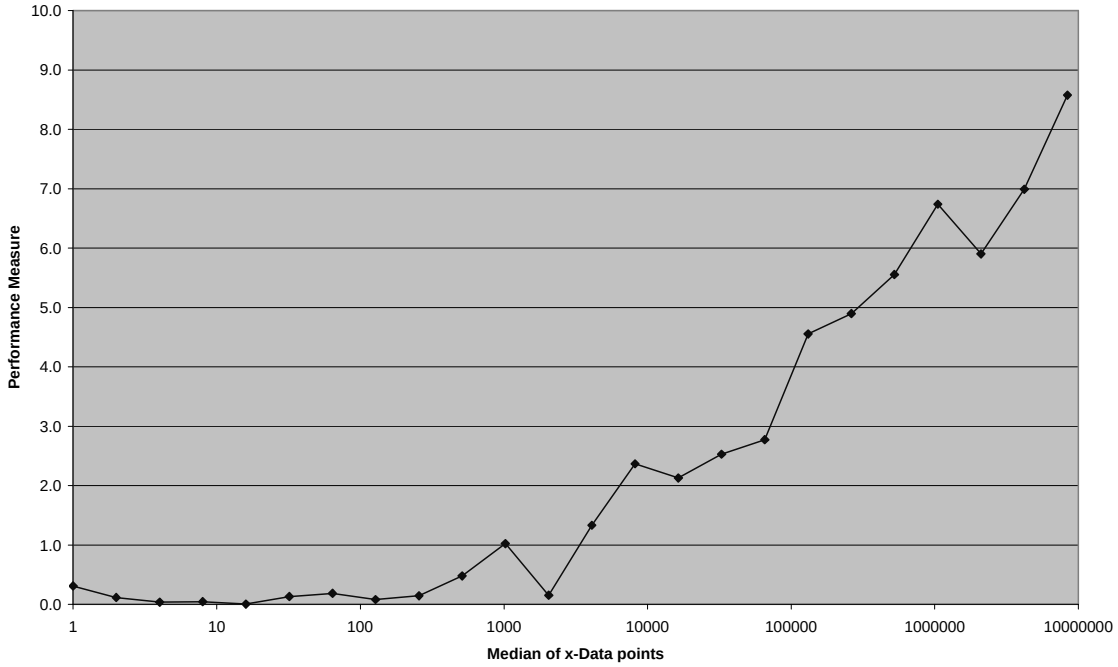

Figure 12: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the median of the data points for the **Excel** intrinsic function **LINEST**.

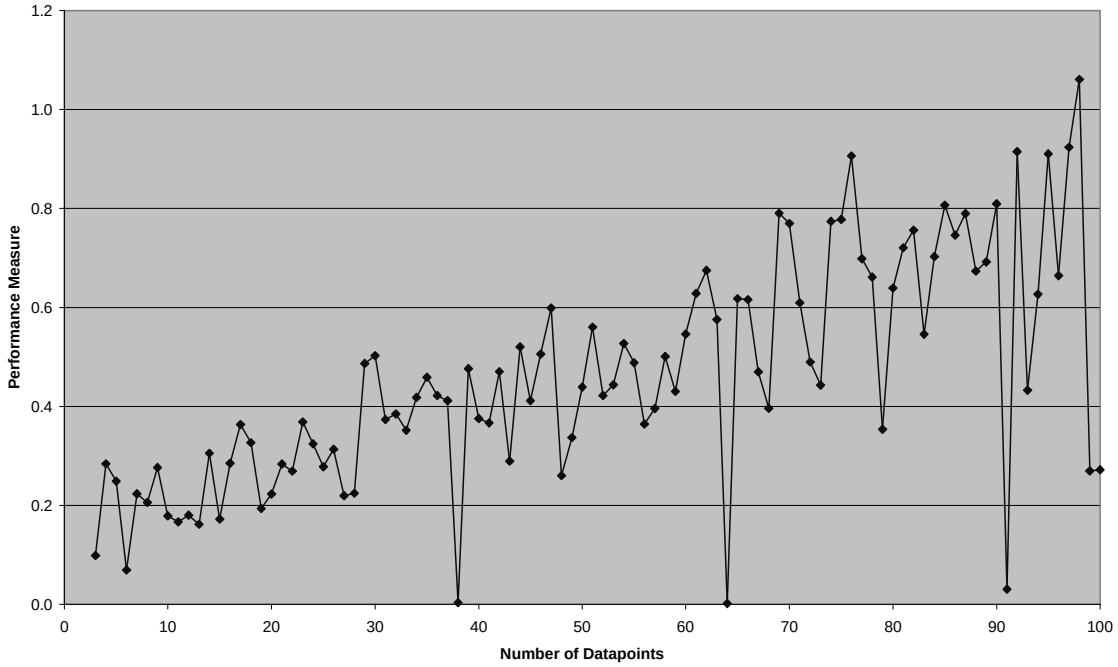

Figure 13: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the number of data points for the **SPLUS** intrinsic function **LM**.

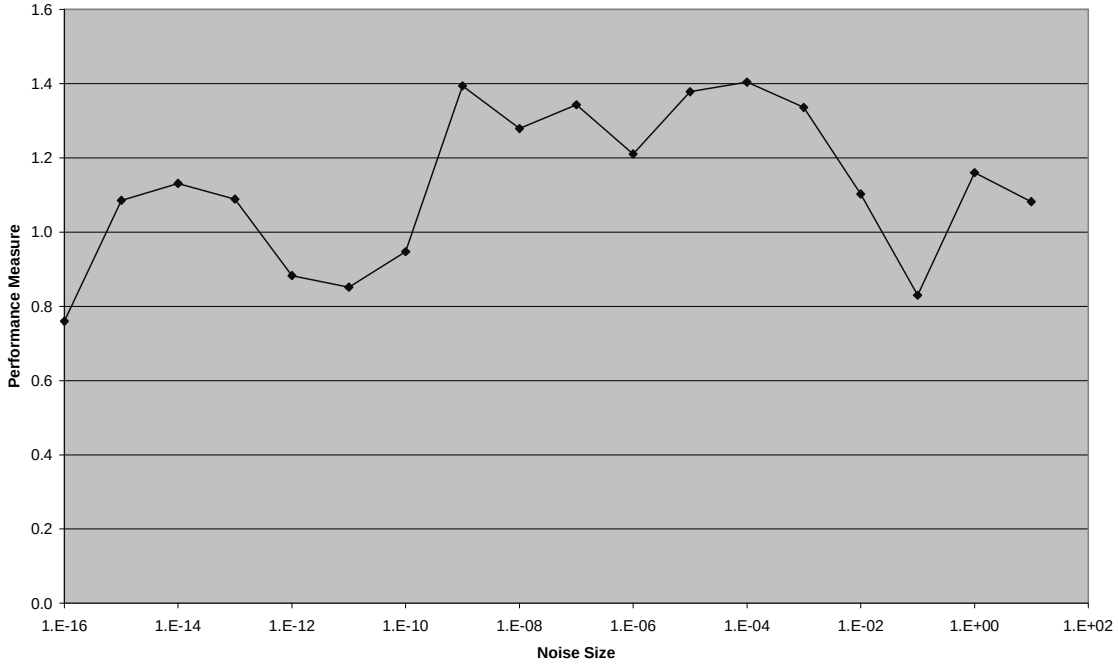

Figure 14: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the noise size for the **SPLUS** intrinsic function **LM**.

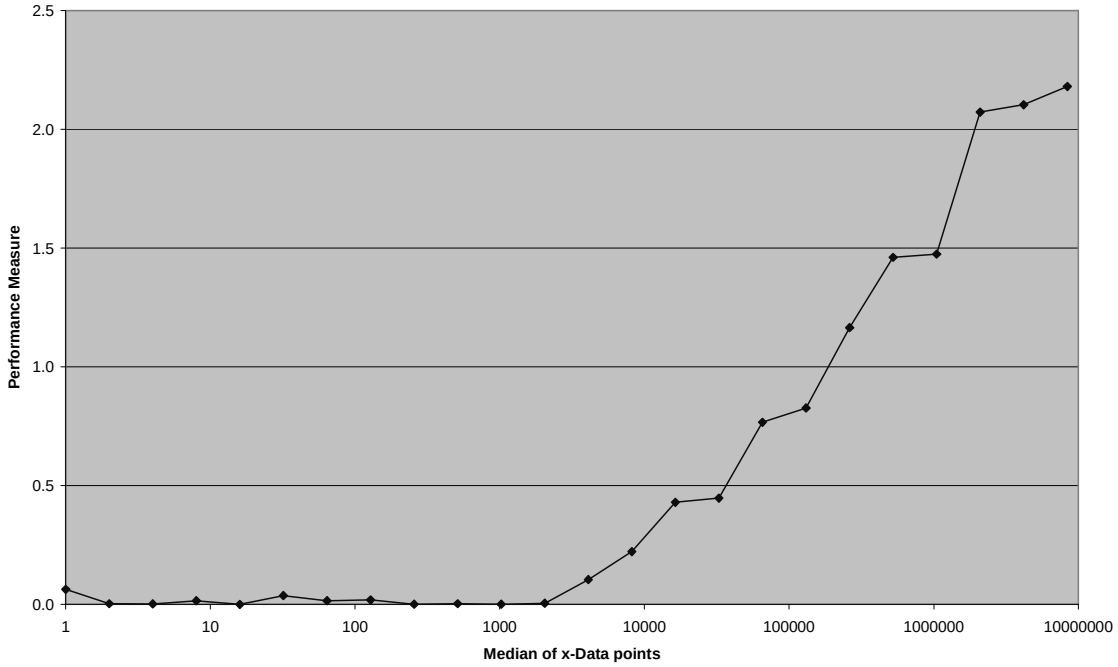

Figure 15: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the median of the data points for the **SPLUS** intrinsic function **LM**.

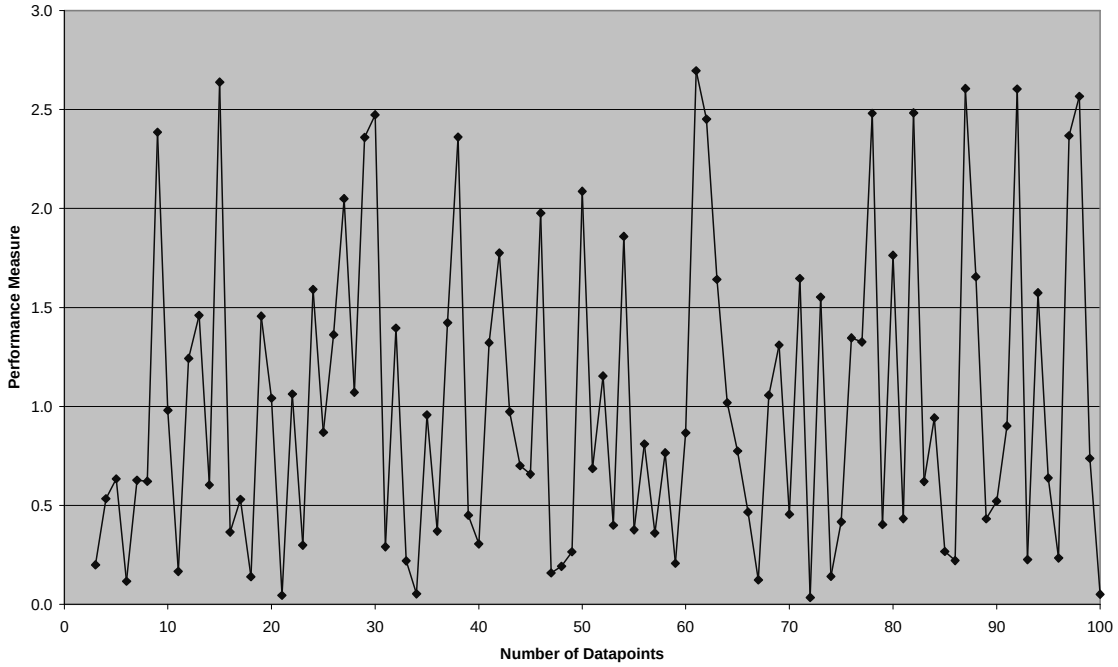

Figure 16: Plot of the performance measure *P*(**b**) for the computed regression coefficients against the number of data points for the **MathCAD** intrinsic function **LINFIT**.

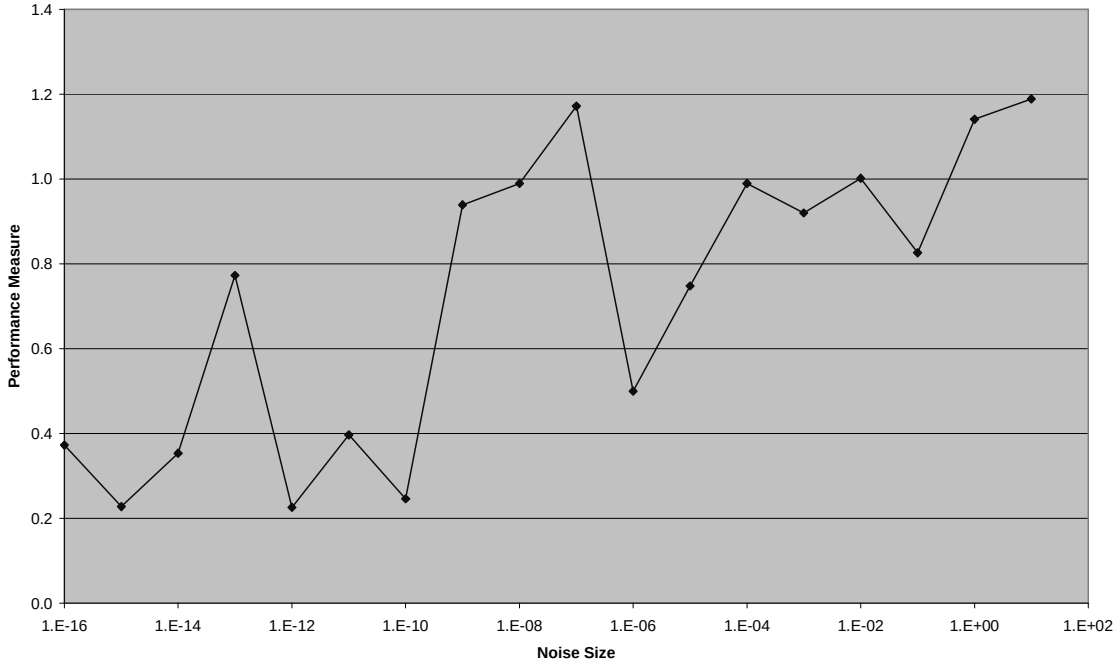

Figure 17: Plot of the performance measure *P*(**b**) for the computed regression coefficients against the noise size for the **MathCAD** intrinsic function **LINFIT**.

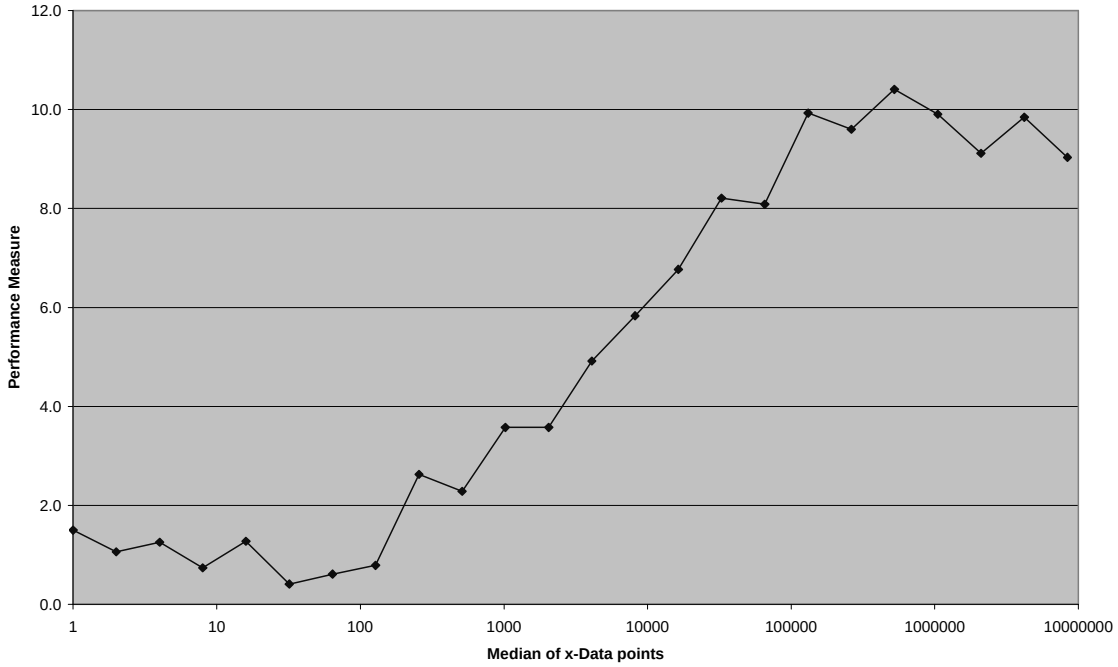

Figure 18: Plot of the performance measure $P(\mathbf{b})$ for the computed regression coefficients against the median of the data points for the **MathCAD** intrinsic function **LINFIT**.