NPL Report CMSC 05/00

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

TESTING SPREADSHEETS AND OTHER PACKAGES USED IN METROLOGY

TESTING THE INTRINSIC FUNCTIONS OF MATHCAD

BY

J BARRETT and M P DAINTON

September 2000

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

J Barrett and M P Dainton

Centre for Mathematics and Scientific Computing

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

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

© Crown Copyright 2000
Reproduced by permission of the controller of HMSO

ISSN 1471-0005

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

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

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

The reports [1, 2] provide a review of

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

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

The MathCAD software package is identified in [1] as the second most popular “off the shelf” software package used in the NMS metrology programmes, following Microsoft Excel. It is used for data analysis, modelling, general mathematics, and algebraic manipulation.

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

In this report we present the results of testing undertaken on the in-built functions of the MathCAD Professional Version 8.0 software package [10]. The report is a companion document to [8] and [9] that describe, respectively, similar testing of the Microsoft Excel and S-PLUS software packages.

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

performance of each function. In Section 5 we present the results of the testing, and provide some interpretation of these results. Section 6 contains our conclusions.

2. Methodology

A general methodology for evaluating the numerical accuracy of the results produced by scientific software is described in [3, 4, 5, 6] and illustrated by the case study [7] and its application to testing the Microsoft Excel [8] and S-PLUS [9] software packages. The basis of the approach is the design and use of reference data sets and corresponding reference results to undertake black-box testing of the software to be tested. The reference data sets and results are generated in a manner consistent with the functional specification of the test software, and the results returned by the test software for the reference data sets are then compared objectively with the reference results using quality metrics or performance measures. Finally, the performance measures are interpreted in order to decide whether the test software meets requirements and is fit for its intended purpose.

In addition to black-box testing of software using reference data sets and results, other complementary tests that do not require the availability of reference data are also useful and have been employed in this work. These tests include (see [6]):

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

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

c) spot checks against tabulated values;

d) checks of solution characterisations.

The methodology for testing that has been applied, and which is described in [6], comprises the following six stages:

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

The first two of these stages are usually carried out as part of the software development process, although in practice with varying amounts of formality. Stages 3 to 6 constitute the approach to software testing advocated in [6] with their application by a software *developer* presented in the case study [7]. 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.

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

a) generating reference data sets and corresponding reference results,

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

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

The generation of reference pairs, i.e., reference data sets and corresponding reference results, is described in Section 2.5. Reference pairs are generated using software implemented in Matlab [11] and employed in other testing work, such as in [7], or using the IMSL libraries supplied with the Digital Visual Fortran 90 software package [12]. The reference data and results are saved in data files that are subsequently imported into MathCAD worksheets during the testing procedure (Section 2.2).

The evaluation of quality metrics and performance measures (Section 2.4), and their presentation as graphs (Section 2.6), is undertaken using MathCAD functions. This approach enables a significant number of tests to be undertaken automatically and quickly.

In the remainder of this section we give an overview of the implementation of each of the six stages in the testing of MathCAD intrinsic functions. In the description that follows, knowledge of the MathCAD system and its terminology is assumed.

2.1 Documenting the specification of the test software

All MathCAD intrinsic functions have an on-line help-file that describes the purpose of the function, together with details of inputs, outputs and any optional arguments, such as information on how to alter tolerance values for iterative methods. Most functions also have a "Quicksheet Example" facility providing easy access to a typical example showing the use of the function. Generally, there is no information about the nature of the numerical algorithms used, and consequently a black-box approach to testing, as described in [6], is especially appropriate.

The information provided in the help-file is regarded in this work as providing the basis of a *specification* of the function against which the function is tested. As an example, consider the GENFIT function. Quoting the help-file verbatim, it is stated that the function provides

> "…A vector containing the parameters that make a function f of x and n parameters u_1, ..., u_n best approximate the data in **vx** and **vy**",

and the inputs to the function are described thus:

- **vx** is a vector of data values. These correspond to the x values.
- **vy** is a vector of real data values which correspond to the y values. The number of elements is the same as **vx**.
- **vg** is either an n element vector of guess values for the n parameters, or **vg** is a scalar in the case where $n = 1$.
- **F** is a function that returns an $n + 1$ element vector containing the function f and its partial derivatives with respect to its n parameters. When $n = 1$, **F** is a scalar.
- n is an integer.

The inputs and outputs are well-defined but it is necessary to apply some interpretation regarding the purpose of the function in order to make it an unambiguous specification. The output is described as a vector containing the parameters that make the function the "best approximate" to the input data. It is natural to assume that this means the computed model is a *least-squares* fit to the data with errors only in the data ordinates (y values), and this is the assumption we shall make for the purposes of testing.

The specification given above, which includes the additional interpretation about the purpose of the function, now makes clear the requirements for generating reference data sets and

corresponding results for testing the function. These are to generate reference data sets consisting of values for the inputs (data **vx**, **vy**, an approximate solution **vg,** the required function and its associated derivatives **F**) together with corresponding reference results for the output computed from the inputs in accordance with the specification of the GENFIT function described above.

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

2.2 Interfacing to the test software

The implementations of the functions tested in this work are the MathCAD worksheet functions contained within MathCAD Professional Version 8.0 [10]. Functions are accessed directly via a MathCAD worksheet that is created manually, and this is intended to mimic the way users themselves can be expected to access the functions. A worksheet is created to contain reference data and results as well as auxiliary information necessary for computing quality metrics (this is in the form of a table: see below), "equations" for calling the function to be tested and for computing quality metrics, and graphics to display the results of the testing.

Where it is necessary to import data into MathCAD this is done in the form of tables as follows:

1. Choose **input table** from the **component** option on the **insert** menu. This sets up the space for a table to be displayed.

2. Name the **table**.

3. Choose **import** from the pop-up menu, and select the data file that is required for importing. When the data file is opened the data will be inserted into the table defined and named in 2 above.

In cases where the amount of data to be imported was large, this was done by dividing the data into a number of smaller sub-blocks and performing the above procedure for each sub-block. (It was found that the integrity of the data was compromised when importing data representing an **input table** with greater than 421 rows.)

2.3 Specification of reference data sets

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

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

For example, consider the problem addressed by the MathCAD function SLOPE for finding the least-squares best-fit gradient for a linear model

$$y = ax + b$$

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

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

Each of these parameters may be varied in turn, with the remaining parameters taking nominal values, and the performance of SLOPE investigated as a function of each parameter. Where data sets are found for which the performance of the function is poor, these data sets can be described using the above performance parameters and related to properties of the user's metrology application. In this way, users are provided with information about the circumstances in which it is safe to use the function and when it is not.

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

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

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

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

2.4 Specification of performance measures and testing requirements

Quality metrics are used to quantify the performance of the test software for the sequences of reference data sets to which the test software is applied. Furthermore, by relating the values of these metrics to the requirements of the user, it is possible to assess objectively whether the test software meets these requirements and is therefore "fit for purpose". Generally, the quality metrics measure the departure of the test results returned by the test software from the reference results. The departure may be measured in (at least) three ways [6]:

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

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

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

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

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

3. a *performance measure* that accounts for factors such as the computational precision and conditioning of the problem. In [6] the following performance measure is derived:

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

where $\kappa(\mathbf{x})$ measures the condition of the problem defined by the data set $\mathbf{x}$ [6], 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.

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

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

where

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

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

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

we have

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

Then,

$$d_A(x) = |h(x) - x| \tag{4}$$

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

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

is a *relative* measure of consistency, and

$$P(x) = \log_{10}\left(1 + \frac{d_R(x)}{\eta}\right) \tag{6}$$

is a *performance measure* for consistency.

It is less easy to use the metrics (4), (5) and (6) to make *quantitative* statements about the test software, although *qualitative* judgements are possible. However, care is necessary: the result that the quality metric $d_A(x)$ given by (4) is zero does not imply that the individual functions f and g are working correctly, only that they form a consistent pair of functions for forward and inverse calculations. Consequently, it is important to supplement such checks with other tests, e.g., testing the individual functions f and g against other (possibly reference) implementations of the functions.

Section 4 lists the quality metrics for each MathCAD function tested and each complementary test.

[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.

2.5 Generation of reference pairs

For the testing of regression functions, including the GENFIT, INTERCEPT, SLOPE, LINFIT, REGRESS and INTERP functions (Section 3.1), for which a mathematical characterisation of the solution exists, *data generators* are used to construct reference data sets with known solutions, i.e., solutions specified *a priori*. The basis of the data generators are *null-space methods* [6] that use the solution characterisation to construct families or classes of data sets possessing nominally the same solution.

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

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

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

For many of the intrinsic functions tested, the test results returned by the functions are compared with values obtained from other implementations of the functions. The software employed for this purpose are the IMSL libraries supplied with the Digital Visual Fortran 90 software package [12]. These particular libraries were chosen for convenience and because the functions included within them are based on reliable and established numerical algorithms developed over many years. Of course there are other sources of high-quality software that could be used for this purpose: for example, the NAG library. It is proposed that as part of future work the sources of software used for testing will be expanded to include other such libraries.

2.6 Presentation and interpretation of performance measures

Having applied the test software to a reference data set to obtain a test result, the test result is compared with the reference result corresponding to the data set by computing a quality metric or performance measure such as is defined by (1), (2) or (3) (Section 2.4). The quality metrics are presented as functions of each (one or more) performance parameter (Section 2.3) either in tabular form or as a graph against the performance parameter, i.e., as a performance profile. Plotting of results in MathCAD is straightforward, and thus the presentation of results as part of the testing procedure can be automated.

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

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

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

For example, one of the performance parameters used in the testing of the REGRESS function is the signal-to-noise ratio. A graph is provided showing values of the performance measure $P(\mathbf{x})$ given by (3) against this parameter. The graph can be used

- to identify values of the signal-to-noise ratio for which the performance of the REGRESS function meets a specified accuracy requirement, e.g., the results are to be accurate to a specified number of significant figures, or

- to assess the performance of the REGRESS function for data sets characterised by a particular range of signal-to-noise ratio values.

For the complementary software tests, quality metrics, such as those defined by (4), (5) and (6) (Section 2.4), are presented (in tabular or graphical form) as a function of the input argument for various choices of the performance parameters (Section 2.3). For example, for the consistency check based on the identity

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

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

3. Specifications of the Intrinsic Functions

In this section we list the MathCAD functions tested as part of this work, and provide specifications for these functions. All the functions tested are MathCAD *worksheet* functions, and the descriptions of each are verbatim quotations of the on-line help documentation provided with MathCAD. Where it is appropriate, our interpretations of these descriptions are included as footnotes. The functions are grouped into the following classes:

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

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

Specifications for the functions tested from each of the above classes are listed in, respectively, Sections 3.1, 3.2 and 3.3. The particular implementations of the functions tested in this work are those contained within MathCAD Professional Version 8 [10].

3.1 Regression Functions

MEAN

mean(A) returns the arithmetic mean of the $m \times n$ array A:

$$\text{mean}(A)=\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}A_{ij}.$$

STDEV

stdev(v) returns the square root of the variance of the elements in **v**.

Arguments:

v is an $m \times 1$ vector.

GENFIT

genfit(vx,vy,vg,F) returns a vector containing the parameters that make a function *f* of *x* and *n* parameters $u_1, \ldots, u_n$ best approximate the data in **vx** and **vy**.

Arguments:

vx is a vector of data values. These correspond to the *x* values.

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

vg is either an *n* element vector of guess values for the *n* parameters, or **vg** is a scalar in the case where $n = 1$.

F is a function that returns an $n + 1$ element vector containing the function *f* and its partial derivatives with respect to its *n* parameters. When $n = 1$, **F** is a scalar.

INTERCEPT and **SLOPE**

slope(vx,vy) returns the slope of the line that best fits data in **vx** and **vy**.

intercept(vx,vy) returns the intercept of the line that best fits data in **vx** and **vy**.

Arguments:

vx is a vector of real data values. The values in **vx** correspond to the *x* values.

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

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**.[2]

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.

REGRESS and **INTERP**

regress(vx,vy,*k*) returns a vector[3] which **interp** uses to find[4] the *k*th order polynomial that best fits the *x* and *y* data values in **vx** and **vy**.

interp(vs,vx,vy,*x*) returns interpolated *y* value corresponding to *x*.

Arguments:

vx is a vector of real data values in ascending order. These correspond to the *x* values.

[2] Note: it is assumed that the elements of **vy** are arranged or sorted in a way corresponding to that in which **vx** has been sorted.

[3] i.e., of parameter values.

[4] i.e., evaluate.

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

vs is a vector[5] generated by **regress**.

k is a positive integer specifying the order of the polynomial you want to use to fit the data. Usually, $k < 5$.[6]

x is the value of the independent variable at which you want to evaluate the regression curve.

3.2 Mathematical and Trigonometric Functions

|

If *z* is a real or complex number, |*z*| returns the absolute value (or modulus or magnitude) of *z*. If **v** is a real or complex vector, |***v***| returns the magnitude (or Euclidean norm or length) of **v**. If **M** is a real or complex square matrix, |***M***| returns the determinant of **M**.

ACOS

acos(*z*) returns the inverse cosine of *z* in radians, where *z* can be real or complex. The result is between 0 and π if *z* is real. For complex *z*, the result is the principal value.

ACOSH

acosh(*z*) returns the inverse hyperbolic cosine of *z*, where *z* can be real or complex. The result is the principal value for complex *z*.

ARG

arg(*z*) returns the argument of *z*, where *z* can be real or complex. The result is between –π and π, including π.

ASIN

asin(*z*) returns the inverse sine of *z* in radians, where *z* can be real or complex. The result is between –π/2 and π/2 if *z* is real. For complex *z*, the result is the principal value.

ASINH

asinh(*z*) returns the inverse hyperbolic sine of *z*, where *z* can be real or complex. The result is the principal value for complex *z*.

ATAN

atan(*z*) returns the inverse tangent of *z* in radians, where *z* can be real or complex. The result is between –π/2 and π/2 if *z* is real. For complex *z*, the result is the principal value.

ATAN2

atan2(*x*,*y*) returns the angle (in radians between –π and π, excluding –π) from the *x*-axis to the line containing the origin (0, 0) and the point (*x*, *y*).

ATANH

atanh(*z*) returns the inverse hyperbolic tangent of *z*, where *z* can be real or complex. The result is the principal value for complex *z*.

---

[5] i.e., of parameter values.

[6] This implies an assumption that most users will only require polynomial fits up to 5th order for typical purposes.

COS

cos(*z*) returns the cosine of *z*, where *z* is in radians and can be real or complex.

COSH

cosh(*z*) returns the hyperbolic cosine of *z*, where *z* can be real or complex.

EXP

exp(*z*) returns the value of the exponential function e^z, where *z* can be real or complex.

LN

ln(*z*) returns the natural logarithm of nonzero *z* (to base *e*), where *z* can be real or complex. For complex *z*, the principal value is returned, i.e., ln(*z*) = ln(|*z*|) + *i*arg(*z*).

LOG

log(*z*) returns the common logarithm of nonzero z to base 10, where *z* can be real or complex. For complex *z*, the result is the principal value.

log(*z*,*b*) returns the logarithm of nonzero *z* to base b. For complex *z*, the result is the principal value.

SIN

sin(*z*) returns the sine of *z*, where *z* is in radians and can be real or complex.

SINH

sinh(*z*) returns the hyperbolic sine of *z*, where *z* can be real or complex.

TAN

tan(*z*) returns the tangent of *z*, where *z* is in radians and can be real or complex.

TANH

tanh(*z*) returns the hyperbolic tangent of *z*, where *z* can be real or complex.

3.3 Statistical Distributions

Functions are provided for evaluating:

1. Probability densities. These give the probability that a random variable will take on a particular value.
2. Cumulative probability distributions. These give the probability that a random variable will take on a value less than or equal to a specified value, and are obtained by integrating (or summing where appropriate) the corresponding probability density over an appropriate range.
3. Inverse cumulative probability distributions. These functions take a probability *p* as an argument and return a value such that the probability that a random variable will be less than or equal to that value is *p*.

Beta distribution

dbeta(*x*,*s1*,*s2*) returns the probability density for the beta distribution

$$\frac{\Gamma(s1+s2)}{\Gamma(s1)\Gamma(s2)}x^{s1-1}(1-x)^{s2-1}, \quad s1, s2 > 0, \quad 0 < x < 1.$$

pbeta(*x*,*s1*,*s2*) returns the cumulative probability distribution.

qbeta(*p*,*s1*,*s2*) returns the inverse cumulative probability distribution

Binomial distribution

dbinom(*k*,*n*,*p*) returns the probability density for the binomial distribution

$$\frac{n!}{k!(n-k)!}p^k(1-p)^{n-k}, \quad 0 \le k \le n, \quad 0 \le p \le 1.$$

pbinom(*k*,*n*,*p*) returns the cumulative probability distribution.

qbinom(*p*,*n*,*r*) returns the inverse cumulative probability distribution.

Chi-squared distribution

dchisq(*x*,*d*) returns the chi-squared distribution

$$\frac{\exp\left(-\frac{x}{2}\right)}{2\Gamma\left(\frac{d}{2}\right)}\left(\frac{x}{2}\right)^{\frac{d}{2}-1}, \quad d > 0, \quad x > 0.$$

pchisq(*x*,*d*) returns the cumulative probability distribution.

qchisq(*p*,*d*) returns the inverse cumulative probability distribution

Exponential distribution

dexp(*x*,*r*) returns the probability density for the exponential distribution

$$r\exp(-rx), \quad r > 0, \quad x > 0.$$

pexp(*x*,*r*) returns the cumulative probability distribution.

qexp(*p*,*r*) returns the inverse cumulative probability distribution.

F distribution

dF(*x*,*d1*,*d2*) returns the probability density for the F distribution

$$\frac{d1^{\frac{d1}{2}}\,d2^{\frac{d2}{2}}\,\Gamma\left(\frac{d1+d2}{2}\right)}{\Gamma\left(\frac{d1}{2}\right)\Gamma\left(\frac{d2}{2}\right)}\frac{x^{\frac{1}{2}(d1-2)}}{(d2+d1x)^{\frac{1}{2}(d1+d2)}}, \quad d1, d2 > 0, \quad x > 0.$$

pF(*x*,*d1*,*d2*) returns the cumulative probability distribution.

qF(*p*,*d1*,*d2*) returns the inverse cumulative probability distribution

Gamma distribution

dgamma(*x*,*s*) returns the probability density for the gamma distribution

$$\frac{x^{s-1}e^{-x}}{\Gamma(s)}, \quad s > 0, \quad x \ge 0.$$

pgamma(*x*,*s*) returns the cumulative probability distribution.

qgamma(*p*,*s*) returns the inverse cumulative probability distribution.

Hypergeometric distribution

dhypergeom(*m*,*a*,*b*,*n*) returns the probability density for the hypergeometric distribution

$$\frac{\begin{bmatrix} b \\ n-m \end{bmatrix}\begin{bmatrix} a \\ m \end{bmatrix}}{\begin{bmatrix} a+b \\ n \end{bmatrix}}, \quad \max\{0, n-b\} \le m \le \min\{n, a\}.$$

phypergeom(*m*,*a*,*b*,*n*) returns the cumulative probability distribution.

qhypergeom(*p*,*a*,*b*,*n*) returns the inverse cumulative probability distribution.

Log-normal distribution

dlnorm(*x*,μ,σ) returns the probability density for the log-normal distribution

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{1}{2\sigma^2}(\ln x-\mu)^2\right\}, \quad \sigma>0, \quad x>0.$$

plnorm(*x*,μ,σ) returns the cumulative probability distribution.

qlnorm(*p*,μ,σ) returns the inverse cumulative probability distribution

Normal distribution

dnorm(*x*,μ,σ) returns the probability density for the normal distribution

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}, \quad \sigma>0.$$

pnorm(*x*,μ,σ) returns the cumulative probability distribution.

cnorm(*x*) returns the cumulative probability distribution with mean 0 and variance 1.

qnorm(*p*,μ,σ) returns the inverse cumulative probability distribution.

Poisson distribution

dpois(*k*,λ) returns the probability density for the Poisson distribution

$$\frac{\lambda^k}{k!}e^{-\lambda}, \quad \lambda>0, \quad k=0,1,2,\ldots$$

ppois(*k*,λ) returns the cumulative probability distribution.

qpois(*p*,λ) returns the inverse cumulative probability distribution.

T distribution

dt(*x*,*d*) returns the probability density for the t distribution

$$\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)\sqrt{\pi}d}\left(1+\frac{x^2}{d}\right)^{-\frac{1}{2}(d+1)}, \quad d>0.$$

pt(*x*,*d*) returns the cumulative probability distribution.

qt(*p*,*d*) returns the inverse cumulative probability distribution.

Weibull distribution

dweibull(*x*,*s*) returns the probability density for the Weibull distribution

$$sx^{s-1}\exp\left(-x^s\right), \quad s>0, \quad x>0.$$

pweibull(*x*,*s*) returns the cumulative probability distribution.

qweibull(*p,s*) returns the inverse cumulative probability distribution.

4. Specification of Performance Parameters and Measures

4.1 Regression Functions

For the testing of the intrinsic regression functions within MathCAD, we follow the methodology of generating reference data sets and results using data generators described in [6]. Consequently, reference data sets with corresponding reference results are readily constructed, and a quantitative quality metric is explicitly available [6, 7]. We present here for each regression function tested, the performance parameters used to define the reference data sets and the performance measure used to compare the results returned by the function tested with the reference results.

MEAN

Performance parameters for this function include

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

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

$$P(\bar{x}) = \log_{10}\left(1 + \frac{1}{\kappa(\bar{x})\eta}\frac{\left|\bar{x} - \bar{x}^{\text{ref}}\right|}{\left|\bar{x}^{\text{ref}}\right|}\right), \tag{7}$$

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

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

and η is the machine precision.

STDEV

Performance parameters for this function include

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

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

$$P(s) = \log_{10}\left(1 + \frac{1}{\kappa(s)\eta}\frac{\left|s - s^{\text{ref}}\right|}{\left|s^{\text{ref}}\right|}\right), \tag{8}$$

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

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

and η is the machine precision [6].

GENFIT

The function GENFIT was tested using two different models as follows:

(a) Gaussian model

$$f(x,\mathbf{a}) = A\exp\left\{\frac{-(x-x_0)^2}{2\sigma^2}\right\},$$

with $\mathbf{a} = (A, x_0, \sigma)^{\mathrm{T}}$. This model is nonlinear in its parameters **a**.

Performance parameters for specifying reference data sets for this function and model include:

- the peak height A,
- the location of the data x-values,[7]
- the peak width σ, and
- the noise size *ns*.

(b) Sine-wave model

$$f(x,\mathbf{a}) = A\sin\{\omega x + \phi\},$$

where $\mathbf{a} = (A, \omega, \phi)^{\mathrm{T}}$. This model is nonlinear in its parameters **a**.

Performance parameters for specifying reference data sets for this function and model include:

- the wave amplitude A,
- the wave velocity ω,
- the phase ϕ, and
- the noise size *ns*.

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

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

where $\kappa(\mathbf{a})$ measures the relative conditioning of the problem, and η is the machine precision [6]. Here, $\kappa(\mathbf{a})$ is computed as the condition number of the (column-normalised) Jacobian matrix J for the associated least-squares problem evaluated at the reference solution, i.e., the (i, j)th element of J is the derivative of the model f (Gaussian or sine-wave) with respect to the jth regression parameter evaluated at the ith data point with $\mathbf{a} = \mathbf{a}^{\mathrm{ref}}$. We also use a performance measure that relates to a *single* regression parameter a taken from the vector **a** of regression parameters that takes the form

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

[7] The interest here is in investigating the affect of changing the location of the data x-values *relative* to the peak position x_0 (which is kept fixed). An alternative approach would be to choose the peak position as the performance parameter and keep the location of the data x-values fixed.

where η is defined above and $\kappa(a)$ is the relative conditioning of the problem for determining the parameter a.

INTERCEPT and **SLOPE**

The model for testing the INTERCEPT and SLOPE is the straight-line model

$$f(x,\mathbf{a}) = a_1 + a_2 x,$$

with $\boldsymbol{a} = (a_1, a_2)$. This model is linear in its parameters $\boldsymbol{a}$. The function INTERCEPT returns a value for the parameter a_1, and the function SLOPE a value for a_2.

Performance parameters for these functions include

- the number of points in the data set,
- the location of the data x-values, and
- the noise size ns.

The performance measure $P(a)$ (see (10)) was used to measure the departure of each computed regression coefficient a returned by the test software from the reference coefficient a^{ref}.

LINFIT

The function LINFIT was tested using the polynomial model

$$f(x,\mathbf{a}) = a_1 + a_2 x + \ldots + a_n x^{n-1},$$

with $\boldsymbol{a} = (a_1, \ldots, a_n)$. This model is linear in its parameters $\boldsymbol{a}$.[8]

Performance parameters for specifying reference data sets for this function and model include:

- the order n of the polynomial, and
- the noise size ns.

The performance measure $P(\mathbf{a})$ (see (9)) was used to measure the departure of each computed regression coefficients a returned by the test software from the reference coefficient a^{ref}.

REGRESS and **INTERP**

The model for testing the REGRESS and INTERP functions is the polynomial model

$$f(x,\mathbf{a}) = a_1 + a_2 x + \ldots + a_n x^{n-1},$$

with $\boldsymbol{a} = (a_1, \ldots, a_n)$. This model is linear in its parameters $\boldsymbol{a}$.[9] The function INTERP is used to evaluate the fitted model computed by REGRESS at the data x-values from which the residual deviations, i.e., the differences between the data y-values and the model values, may be calculated.

Performance parameters for these functions include

- the location of the data x-values,
- the order n of the polynomial, and

[8] This particular choice of polynomial parameterisation was used because it is the parametrisation (a) commonly used by metrologists, and (b) probably used by the MathCAD functions REGRESS and INTERP (note that no explicit information about parametrisation is included in the documentation for these functions). Alternative ways of representing polynomials, which possess better numerical properties, are discussed in [13].

[9] See footnote 8. Strictly, the testing of REGRESS and INTERP using the computed residuals **e** as reference and test results is independent of the polynomial parametrisation implemented.

- the noise size *ns*.

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

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

where η is the machine precision [6].

4.2 Mathematical and Trigonometric Functions

The following consistency tests were carried out:

T1. $\sin(\text{asin}(x)) = x, -1 \le x \le 1.$

T2. $\text{asin}(\sin(x + 2n\pi)) = x, -\pi/2 \le x \le +\pi/2.$

T3. $\cos(\text{acos}(x)) = x, -1 \le x \le 1.$

T4. $\text{acos}(\cos(x + 2n\pi)) = x, 0 \le x \le \pi.$

T5. $\tan(\text{atan}(x)) = x.$

T6. $\text{atan}(\tan(x + 2n\pi)) = x, -\pi/2 < x < \pi/2.$

T7. $\text{atan2}(x, y) = \text{atan}(y/x), x > 0.$

T8. $\text{atan2}(x, y) = \text{atan}(y/x) + \pi, x < 0.$

T9. $\text{atan2}(\cos(x + 2n\pi), \sin(x + 2n\pi)) = x, -\pi < x \le +\pi.$

T10. $\sinh(\text{asinh}(x)) = x.$

T11. $\text{asinh}(\sinh(x)) = x.$

T12. $\cosh(\text{acosh}(x)) = x, x \ge 1.$

T13. $\text{acosh}(\cosh(x)) = x.$

T14. $\tanh(\text{atanh}(x)) = x, -1 < x < +1.$

T15. $\text{atanh}(\tanh(x)) = x.$

T16. $\sin^2(x) + \cos^2(x) = 1.$

T17. $\tan(x) = \sin(x)/\cos(x).$

T18. $\sinh(x) = (\exp(x) - \exp(-x))/2.$

T19. $\cosh(x) = (\exp(x) + \exp(-x))/2.$

T20. $\tanh(x) = \sinh(x)/\cosh(x).$

T21. $\exp(\ln(x)) = x.$

T22. $\ln(\exp(x)) = x.$

T23. $\log(x) = \ln(x)/\ln(10).$

T24. $\log(x, \exp(1)) = \ln(x).$

T25. $\log(x, 10) = \log(x).$

T26. $\log(y^x, y) = x.$

In each case, the performance measure $P(x)$ given by (6),

$$P(x) = \log_{10}\left(1 + \frac{d_R(x)}{\eta}\right),$$

is used to make judgements about the test software in the following ways:

1. By highlighting cases for which the value of the performance measure is significantly greater than zero (equivalently, cases for which the relative measure $d_R(x)$ of consistency is significantly greater than η).
2. By identifying discontinuities in value and/or slope of the performance measure as a function of its argument, e.g., performance parameter.
3. By noting trends in the values of the performance measure, e.g., periodic behaviour of the performance measure.

4.3 Statistical Distributions

The results returned by individual functions are compared with those returned by the corresponding functions from the IMSL Fortran 90 Math Library (version 3.0) provided with the Digital Visual Fortran (version 5.0D) software package [12]. Tables of reference values were generated and imported into the MathCAD environment where direct comparisons were made.

In each case, the performance measure $P(x)$ given by (6),

$$P(x) = \log_{10}\left(1 + \frac{d_R(x)}{\eta}\right),$$

is used to make judgements about the test software in the ways described in Section 4.2.

5. Presentation and Interpretation of Results

5.1 Regression Functions

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

MEAN and **STDEV**

Figures 1–6 show plots of the performance measures given by (7) and (8) for the sample mean and sample standard deviation, respectively, against the following performance parameters:

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

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

Figures 1–3 show that there is no significant degradation of the performance of the function MEAN over the ranges of the performance parameters for which the function was tested. In all three graphs, the largest value of the performance measure is approximately unity, indicating that the accuracy obtained for this function is very close to that which would be obtained from a numerically optimal algorithm for calculating the sample mean.

The results for the function STDEV are shown in Figures 4–6. Although some degradation of the performance of the function may be observed with increasing reference sample mean, the

values of the performance measure are not significant, and the performance is similar to that observed for the function MEAN (Figure 1) on which the function STDEV depends. Indeed, over all ranges of the performance parameters for which the function was tested, it is observed that the largest value of the performance measure is approximately two. These results imply that even in fairly extreme cases only at most two significant figures of accuracy is lost over and above that which would be obtained from an optimal algorithm.

It is helpful to compare these results with those obtained from the testing of the equivalent functions within the Microsoft Excel spreadsheet package [8]. In [8] it was concluded that the Excel function for the sample standard deviation implemented a *numerically* unreliable, though *mathematically* correct, formula and that *pre-processing* of the sample values [13] prior to application of the Excel function was necessary in some circumstances to obtain accurate results. The testing of the MathCAD STDEV function, on the other hand, leads us to conclude that a numerically reliable method of evaluation has been implemented in this case and pre-processing of the data is therefore unnecessary.

Parameter	*Nominal value*
reference sample mean	+1.437575400258079
number of points	50
reference sample standard deviation	0.1

Table 1 Nominal values for the performance parameters for the testing of the MEAN and STDEV worksheet functions.

GENFIT

Figures 7–18 and 19–30 show performance profiles obtained from the testing of the function GENFIT using, respectively the Gaussian and sine-wave models. The performance measure plotted is that defined by equation (10) applied to each calculated model parameter. Nominal values for the performance parameters for the testing using the two models are given in Tables 2 and 3, respectively. The models were chosen as they present moderately difficult non-linear regression problems, and are encountered in a number of metrology applications.

For testing using the Gaussian model, we see that there is some degradation in the performance of the function GENFIT as the amplitude A is decreased (Figures 7–9), the peak width σ is increased (Figures 10–12), the location (median) of the x-data is moved relative to the peak location x_0 (Figures 13–15), and the amount of data noise is increased (Figures 16–18). Generally, we would expect the regression problem to be more difficult, i.e., to have a larger conditioning, when the peak is less well defined by the data. This happens when there is a greater amount of data noise or when there are fewer points defining the peak, e.g., as happens when the distribution of data remains fixed but the peak width is increased or less of the peak is covered by the data. The observed degradation for the ranges of data sets considered implies the performance of the function is worse than would be expected accounting for the change in conditioning.

For testing using the sine-wave model against the performance parameters amplitude A, wave velocity ω and phase angle ϕ (Figures 19–27), we see that the performance measure takes values that are either approximately equal to five or approximately fifteen. For the performance parameter A, the performance measure is approximately fifteen for a single reference data set (corresponding to a small wave amplitude); for the performance parameter ω, the performance measure is approximately five for about half of the reference data sets (corresponding to wave velocities less than five) and approximately fifteen for reference data sets with wave velocities greater than five; for the performance parameter ϕ, the performance measure is approximately five over the complete range of ϕ values. Where the performance measure is five, the software

is losing about five figures of accuracy compared to a reference algorithm. The fact that the performance measure assumes a *consistent* value suggests that the observed accuracy of the results may be due to truncation errors arising from the convergence criterion used to terminate an iterative algorithm implemented by the software. One way to ascertain whether this explanation is valid would be to adjust the termination criterion and observe any change in the performance value: it does not appear to be possible to make such an adjustment to the MathCAD function. Where the performance measure is fifteen, the software is losing essentially all figures of accuracy compared to a reference algorithm. In such cases the software has converged to a (local) solution that is different from the reference solution.[10] Such behaviour may be misleading for the casual user. Finally, in the same way as for the Gaussian model, there is degradation in the performance of the software as the amount of data noise is increased (Figures 28–30).

Parameter	*Nominal value*
peak height A	1.0
peak location x_0	1000
peak width σ	1.0
location (median) of data x-values	1000
range covered by x-values	4.0
noise size *ns*	0.001

Table 2 Nominal values for the performance parameters for the testing of GENFIT using a Gaussian model function.

Parameter	*Nominal value*
wave amplitude A	1.0
wave velocity ω	1.0
phase ϕ	$\pi/2$
noise size *ns*	0.001

Table 3 Nominal values for the performance parameters for the testing of GENFIT using a sine-wave model function.

INTERCEPT and **SLOPE**

Figures 31–36 show plots of the performance measure (10) for the MathCAD linear regression functions INTERCEPT and SLOPE against the performance parameters

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

Nominal values for the performance parameters are presented in Table 4.

[10] This is even though for all reference data sets initial estimates for the solution parameters are assigned by common procedure.

It can be seen that there is no significant degradation in the performance of the algorithms implemented by the functions INTERCEPT and SLOPE for changes in the number of points in the data set (Figures 31 and 32) and the noise size (Figures 35 and 36). The largest value of the performance measure for these performance parameters is approximately 2.5, indicating that at most three significant figures of accuracy are lost compared to a reference algorithm.

However, Figures 33 and 34 show that there is some degradation in the performance of the functions as the location of the data *x*-values with respect to the origin is changed. The performance measures for both the intercept a_1 and gradient a_2 values become larger as the data is moved away from the origin. For example, in the case of the intercept parameter, the performance measure increases from (approximately) two to six as the data is shifted by 10^3. It can be expected that pre-processing the input data (by shifting the data so that the mean of the data *x*-values is approximately zero) would be beneficial to the accuracy of the results returned [13].

Parameter	*Nominal value*
number of points	421
location (median) of data *x*-values	0
noise size *ns*	250

Table 4 Nominal values for the performance parameters for the testing of the INTERCEPT and SLOPE functions.

LINFIT

Figures 37–39 show performance profiles obtained from the testing of the function LINFIT using a polynomial model. The profiles show the behaviour of the performance measure (9) against noise size for a range of polynomial orders. Nominal values for the performance parameters are given in Table 5.

No significant loss of accuracy is observed, with the results for a 10^{th} order polynomial showing that a maximum of approximately two figures of accuracy is lost compared with a reference algorithm (Figure 39). A performance of this kind is expected to satisfy the requirements of most metrology applications.

Parameter	*Nominal value*
order *n* of the polynomial	2, 4 and 10
noise size *ns*	1.0

Table 5 Nominal values for the performance parameters for the testing of the LINFIT function.

REGRESS and **INTERP**

Performance profiles obtained from the testing of the functions REGRESS and INTERP are illustrated in Figures 40–45. The profiles show the behaviour of the performance measure (11) against the location of the data *x*-values and the noise size for a range of polynomial orders. Nominal values for the performance parameters are give in Table 6.

Whilst there is no significant degradation in the performance of the software with the noise size, the location of the data *x*-values does appear to affect the performance of the functions, with the magnitude of the effect depending on the polynomial order. It is a well-known phenomenon that the performance of polynomial regression software can exhibit such

behaviour if the algorithms implemented work in terms of a polynomial function expressed in powers of an unnormalised *x*-variable. It is expected that pre-processing the input data (by shifting the data so that the mean of the data *x*-values is approximately zero) would be beneficial to the accuracy of the results returned by these functions [13]. Further improvements in performance would be obtained from using an alternative polynomial representation, such as in terms of Chebyshev basis functions. Note that data normalisation (shifting and scaling) is performed when using a Chebyshev basis.

Parameter	*Nominal value*
location (median) of data *x*-values	1.0
order *n* of the polynomial	2, 4 and 10
noise size *ns*	1.0

Table 6 Nominal values for the performance parameters for the testing of the REGRESS and INTERP functions.

5.2 Mathematical and Trigonometric Functions

Figures 46–51 in Appendix B show plots of the performance measure (6) for a selection of the consistency checks described in Section 4.2 for the mathematical and trigonometric functions. We have only displayed those results that are significant (in terms of the values of the performance measure). Note that some care is needed in interpreting these performance profiles in the neighbourhood of the origin, since the measure (6) is based on the *relative* departure between the test and reference results. We note that:

- for many of the graphs (interlaced) subsets of the results appear to exhibit systematic behaviour, e.g., T19 (Figure 50) and T20 (Figure 51), and
- for T15 (Figure 49) and T20 (Figure 51) the value of the performance measure becomes large towards the ends of the interval for *x* over which the checks are applied.

The systematic behaviour observed might be explained by the fact that an innate sensitivity associated with the consistency checks is not being fully accounted for in the performance measure. For some input values the floating point numbers involved are such that the consistency check is exact; for other values the problem sensitivity will indicate an inflation of the computational precision η that manifests itself as a non-zero performance value. This effect is a consequence of the nature of floating point arithmetic, and may be significant depending on the intended use of the functions. It is the inflation of η that is observed in the graphs presented.

For T15, it should be noted that the use of a "zero of a function" technique for evaluating the inverse of a function, e.g., the use of "tanh" and a bisection algorithm for evaluating "atanh", can be expected to yield the maximum possible precision for this consistency check. In addition, the results for T15 are similar in appearance to those obtained by implementing the same consistency check using S-PLUS [9] for which the same performance measure (again not accounting for the problem sensitivity) is used.

5.3 Statistical Distributions

Figures 52–55 in Appendix C show plots of the performance measure (6) for the comparison of a selection of the MathCAD statistical distributions against the equivalent IMSL functions. Again, only significant results (in terms of the values of the performance measure) are shown. We note that there are:

- systematic departures between equivalent functions, e.g., for PBETA (Figure 52) and DNORM (Figure 55), and

- discontinuities in the comparisons of equivalent functions, e.g., for PBETA (Figure 52) and DNORM (Figure 55).

It is noted that the results for the PBETA function (Figure 52) are similar in appearance to those obtained by implementing the same comparison in the Microsoft Excel spreadsheet package using Excel's equivalent function for evaluating the beta distribution. Again, it is probable that there is an innate conditioning associated with these tests that is manifesting itself as systematic behaviour of this kind. Furthermore, no account is being taken of the accuracy of the results returned by the IMSL functions, which makes quantitative interpretation of the results difficult.

Since for many applications in metrology only a few significant figures of accuracy are required when evaluating these distributions, the differences reported here are expected to be sufficiently small.[11] However, the results for the DNORM function (Figure 55) are quite concerning, as the Gaussian probability distribution function is key to many uncertainty evaluations undertaken in metrology applications.

6. Conclusions

In this report we have described the application of a general methodology [6] for testing the numerical accuracy of scientific software to specific functions taken from the MathCAD software package. Each stage of the methodology, from documenting a specification for each function tested through the definition of performance parameters and measures to the presentation and interpretation of the test results, has been described. In this way, and by stating any assumptions made in the application of the methodology, the testing undertaken is made as objective as possible given the nature of the testing.

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

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

[11] However, note that deciding fitness for purpose (in terms of numerical accuracy of results) needs to account for the way the results returned by the software are subsequently used. Some of the issues that affect how users decide whether software is fit for purpose within the context of a metrology application are considered in [8, 13].

Some of the highlights of the test results reported in this work are as follows:

- The functions MEAN and STDEV returns results to an accuracy that is very close to that for numerically optimal algorithms for the sample mean and standard deviation calculations.
- The performance of the linear regression functions (LINFIT, INTERCEPT and SLOPE, and REGRESS and INTERP) is generally good, although some degradation in performance as a function of the location of the data *x*-values is observed. This is attributed to the model parametrisation employed, and may be avoided by pre-processing of the input data [13].
- The performance of the nonlinear regression function GENFIT for the two models considered (Gaussian and sine-wave) tends to degrade for reference data sets for which the solution is not well defined by the data, e.g., in cases where the amount of noise is large, or where the data does not fully reflect the main features of the function being fitted. For both models degradation beyond that which is accounted for by problem conditioning is observed. Additionally, for the sine-wave model, behaviour is observed such that the results either reproduce the reference solution to an accuracy (probably) dictated by a termination criterion, or approximate the reference solution very poorly (having converged to a local solution that is different from that anticipated). Since the latter occurs without warning, this can be very misleading for the user.
- The results of consistency checks for the mathematical and trigonometric functions are generally good: a small number of examples of systematic behaviour in the values of the performance measure for these checks have been indicated. The systematic behaviour may be due to the sensitivity of the consistency checks not being accounted for in the performance measures, and further work will be necessary to address this issue.
- The results of the comparison of the statistical distribution with other equivalent software are generally good: a small number of examples of systematic behaviour in the values of the performance measure for these tests have been indicated.

7. Acknowledgements

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

We would also like to acknowledge Professor M. G. Cox and Dr P. M. Harris for their considerable input into this report by way of supervision and reviewing of the work.

8. References

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] H.R. Cook, M.G. Cox, M.P. Dainton and P.M. Harris. *Testing spreadsheets and other packages used in metrology: A case study*. 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.

[8] 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.

[9] 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.

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

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

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

[13] 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.

Appendix A: Results for Regression Functions

MEAN

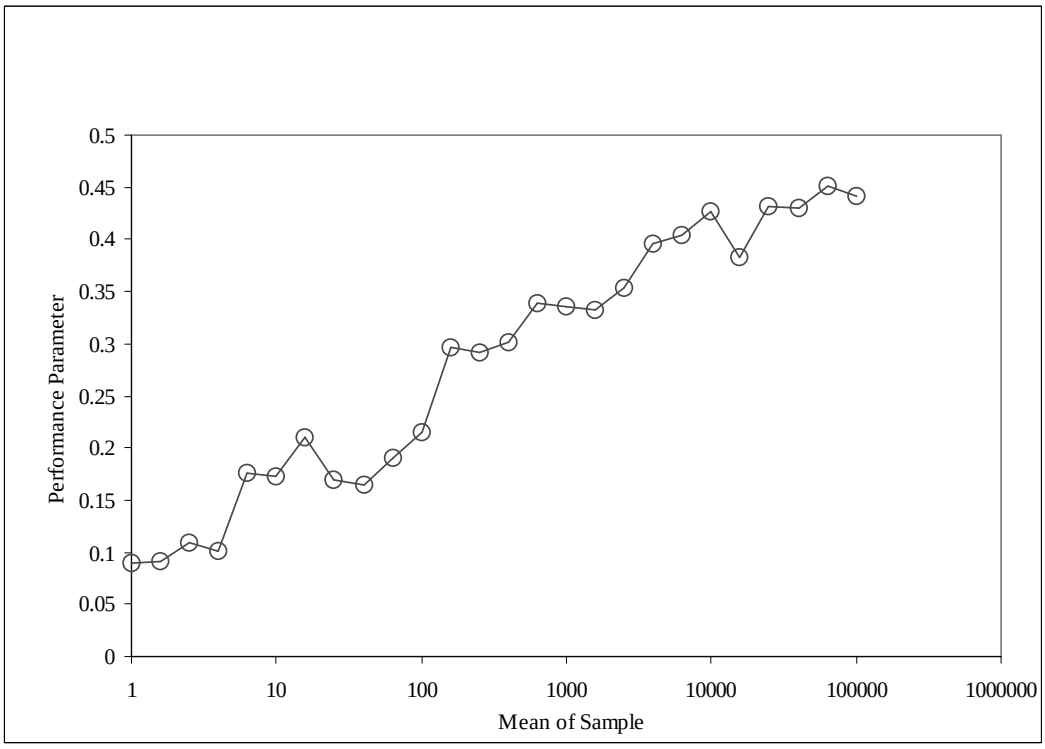

Figure 1: Plot of the performance measure $P(\bar{x})$ for the sample mean $\bar{x}$ against the reference sample mean.

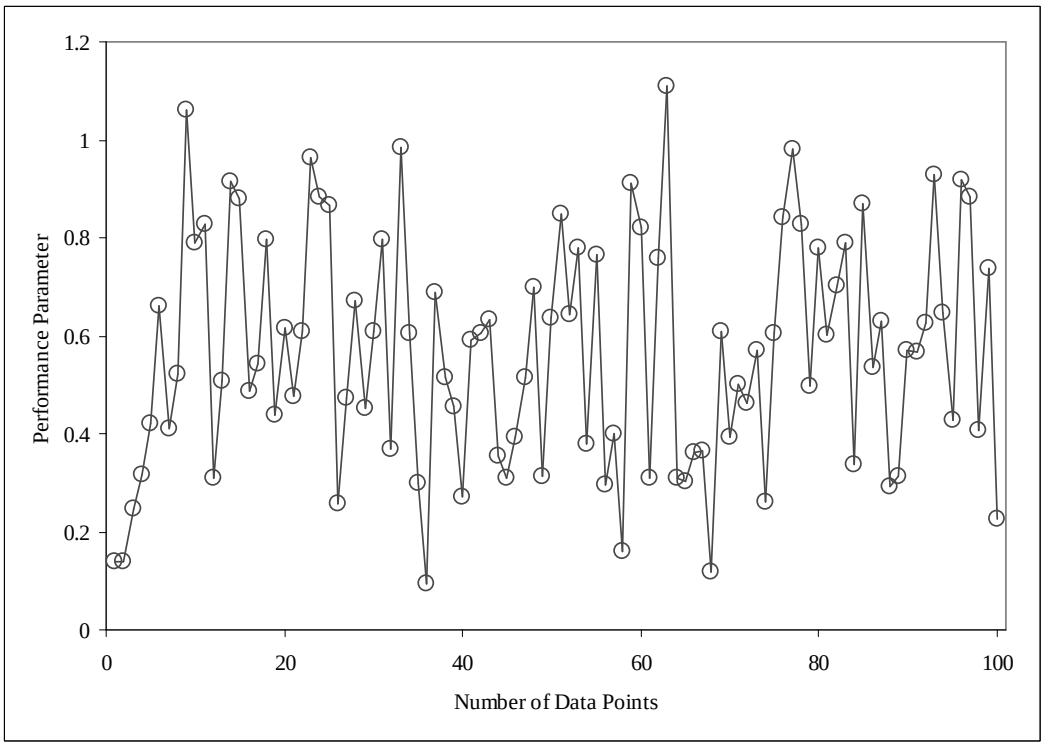

Figure 2: Plot of the performance measure $P(\bar{x})$ for the sample mean $\bar{x}$ against the number of data points.

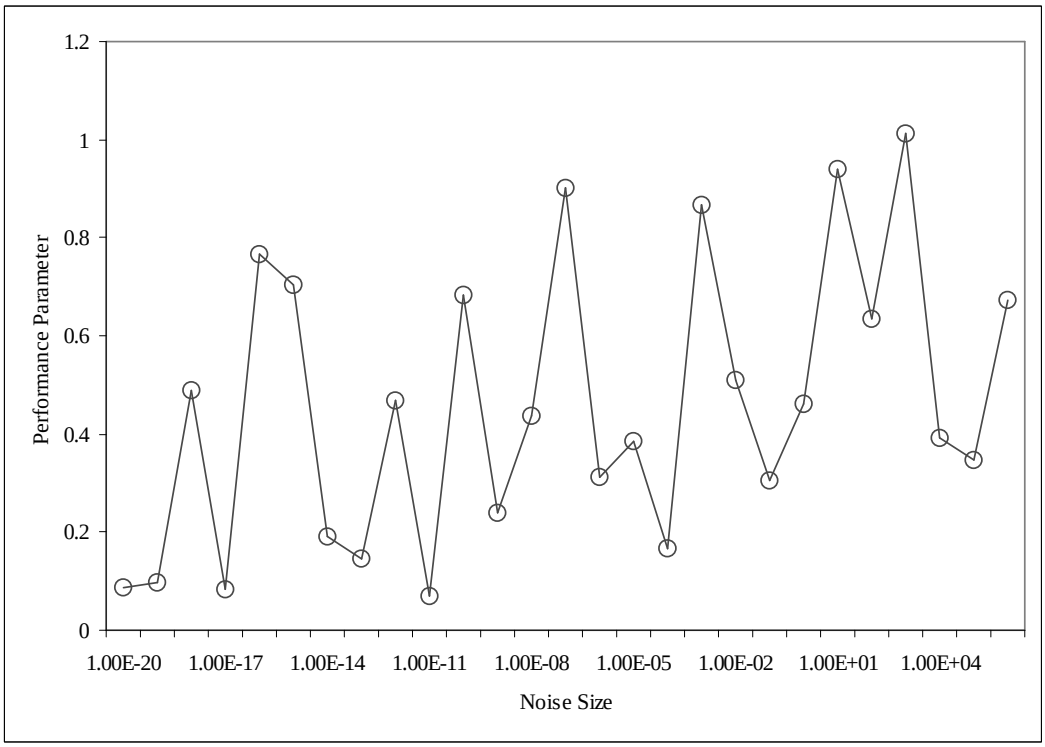

Figure 3: Plot of the performance measure $P(\bar{x})$ for the sample mean $\bar{x}$ against the reference sample standard deviation.

STDEV

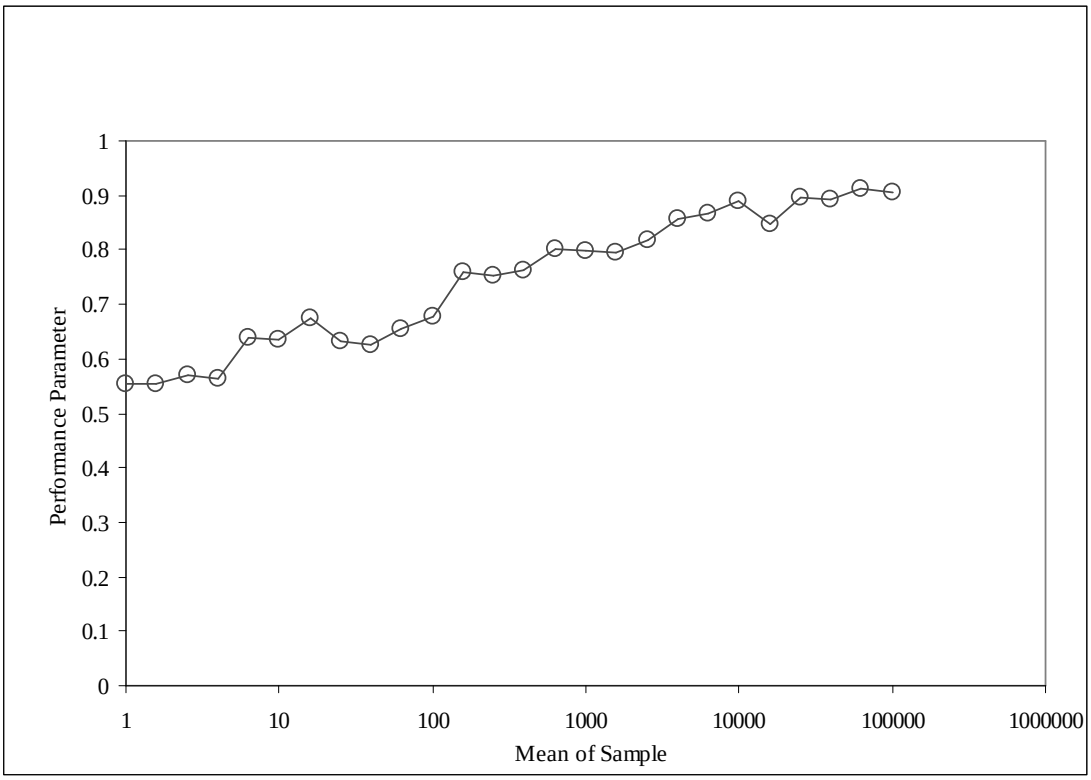

Figure 4: Plot of the performance measure $P(s)$ for the sample standard deviation s against the reference sample mean.

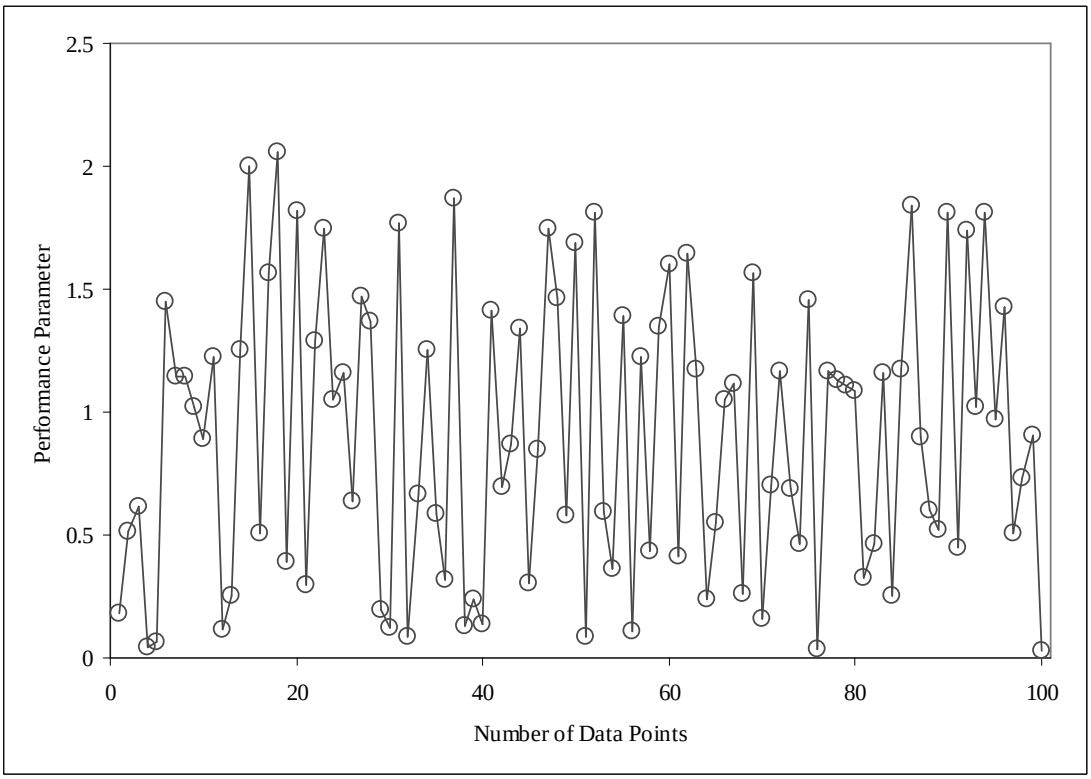

Figure 5: Plot of the performance measure $P(s)$ for the sample standard deviation s against the number of data points.

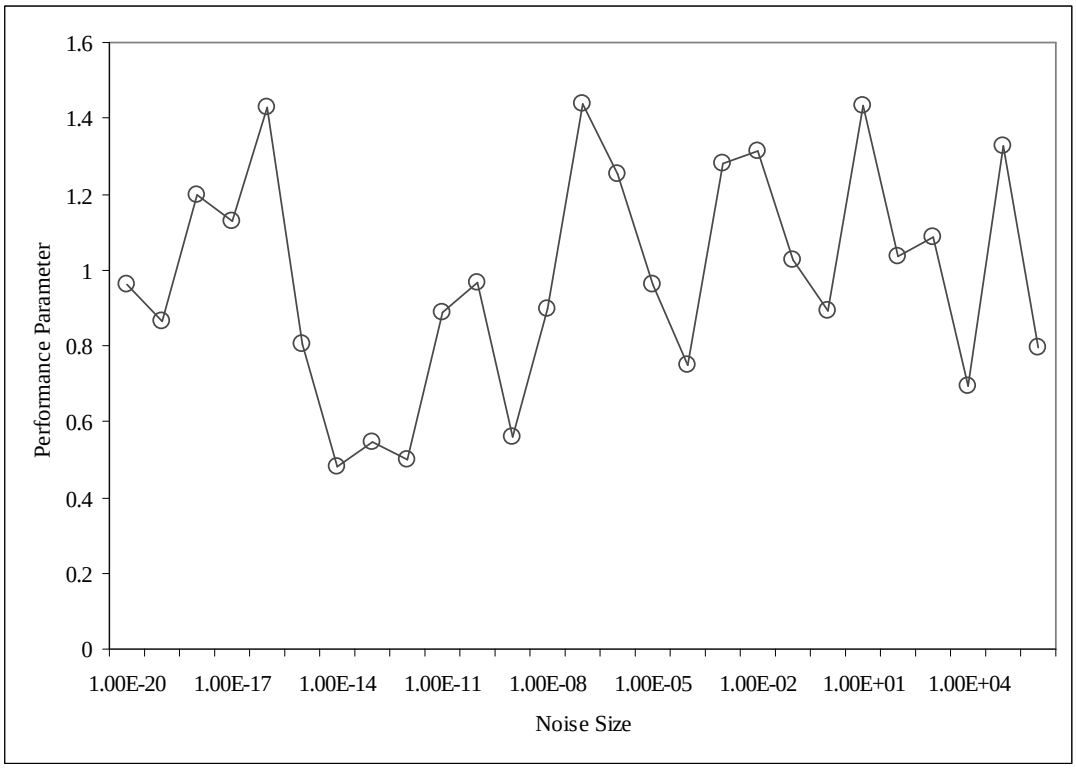

Figure 6: Plot of the performance measure $P(s)$ for the sample standard deviation s against the reference sample standard deviation.

GENFIT: Gaussian Model

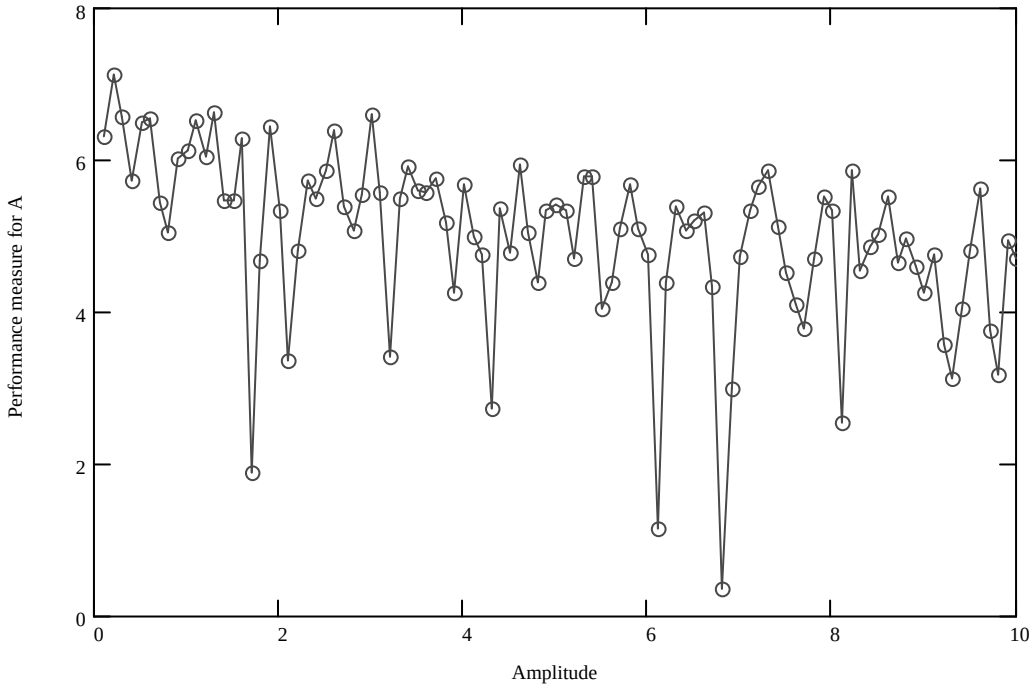

Figure 7: Plot of the performance measure $P(A)$ against amplitude.

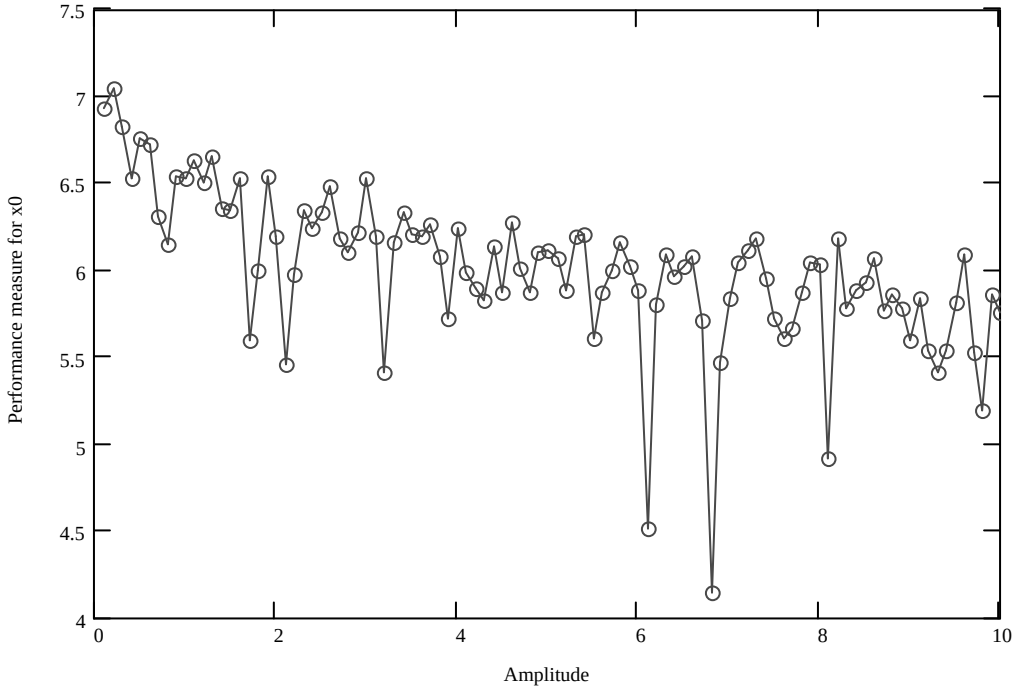

Figure 8: Plot of the performance measure $P(x_0)$ against amplitude.

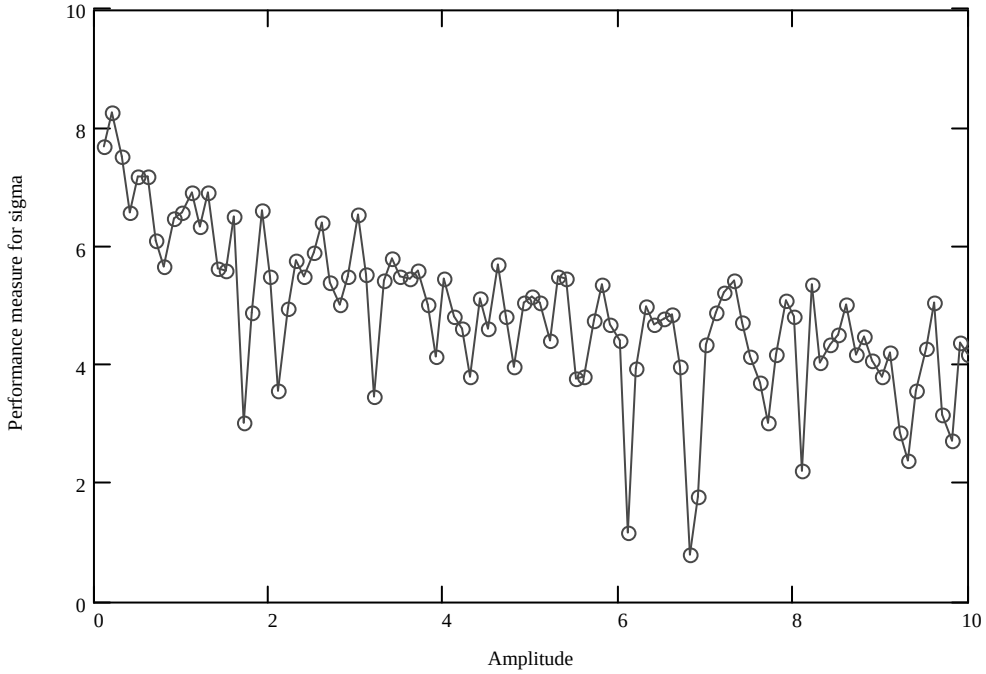

Figure 9: Plot of the performance measure $P(\sigma)$ against amplitude.

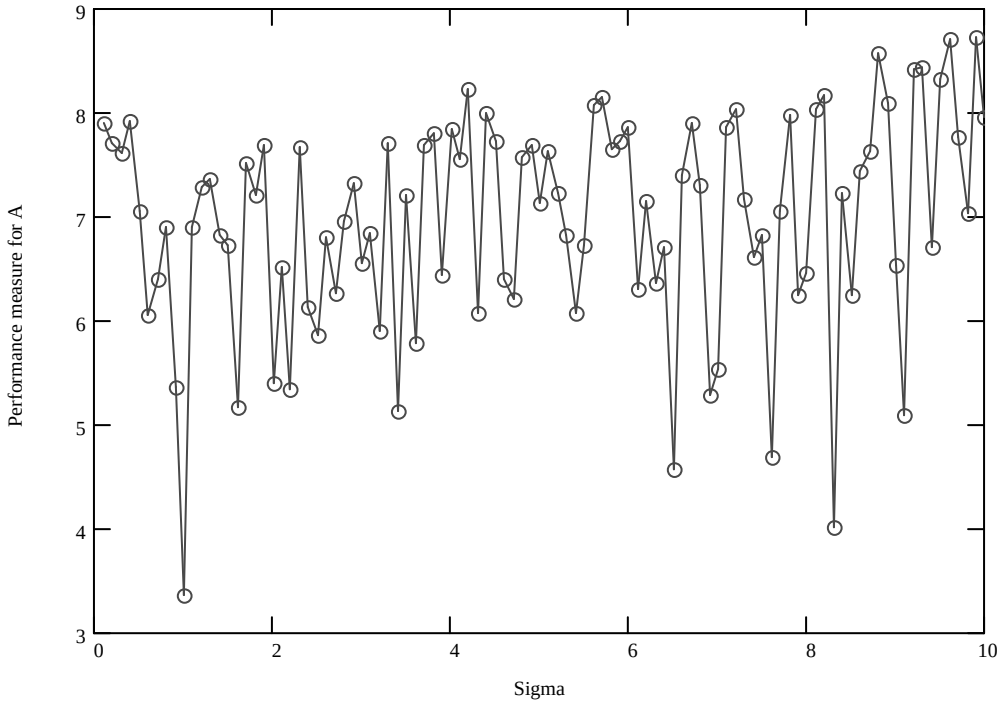

Figure 10: Plot of the performance measure $P(A)$ against peak width.

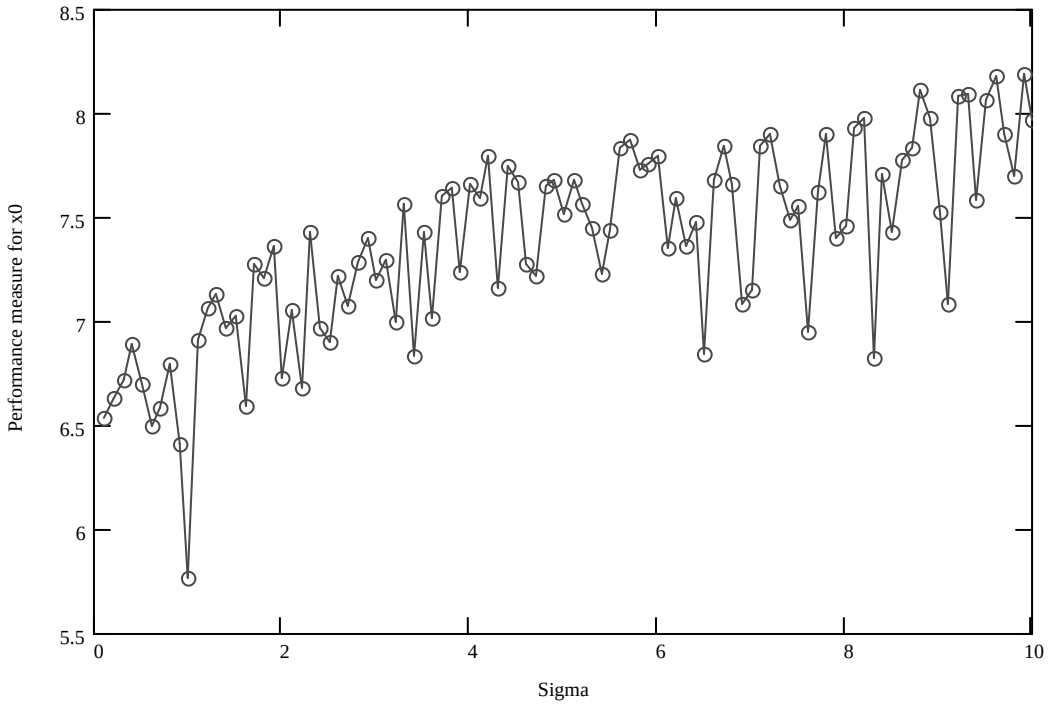

Figure 11: Plot of the performance measure $P(x_0)$ against peak width.

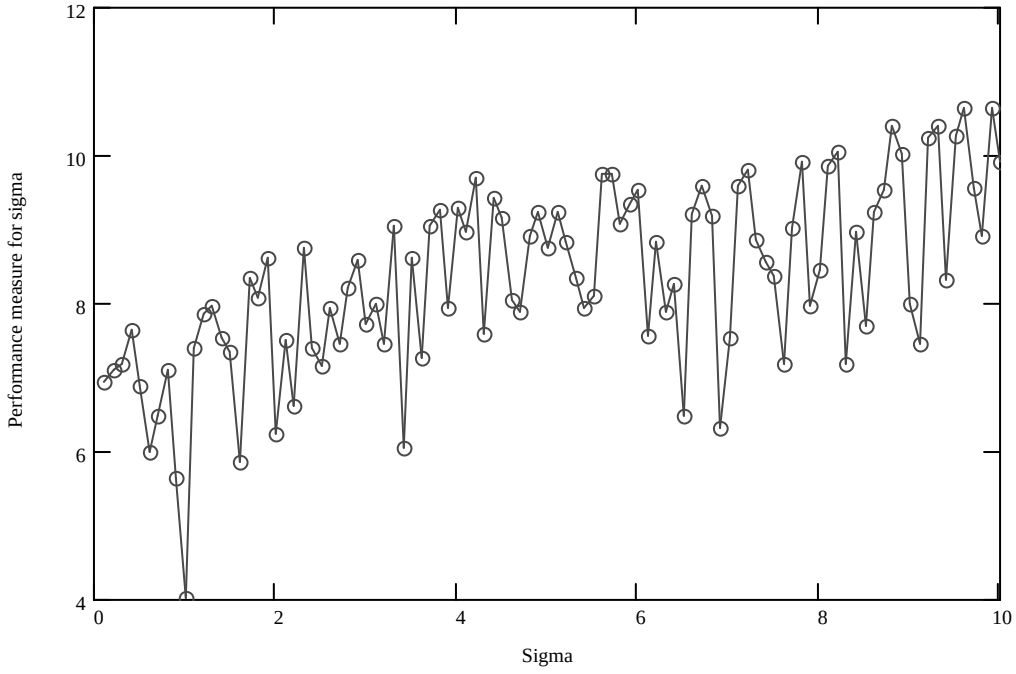

Figure 12: Plot of the performance measure $P(\sigma)$ against peak width.

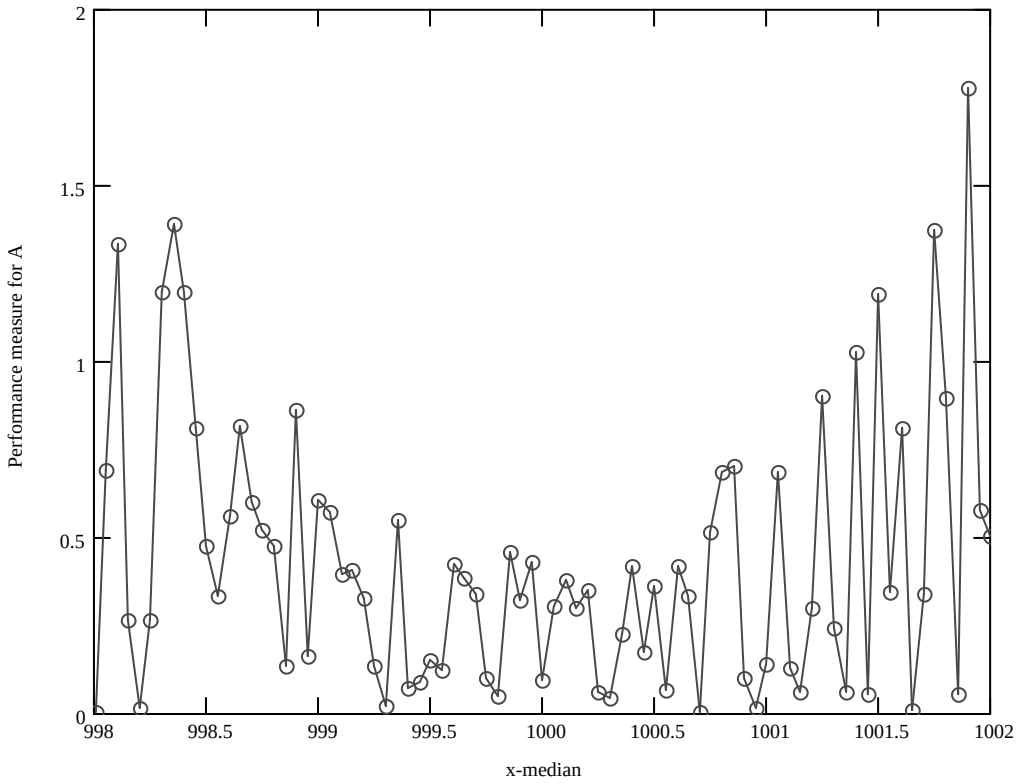

Figure 13: Plot of the performance measure $P(A)$ against data location.

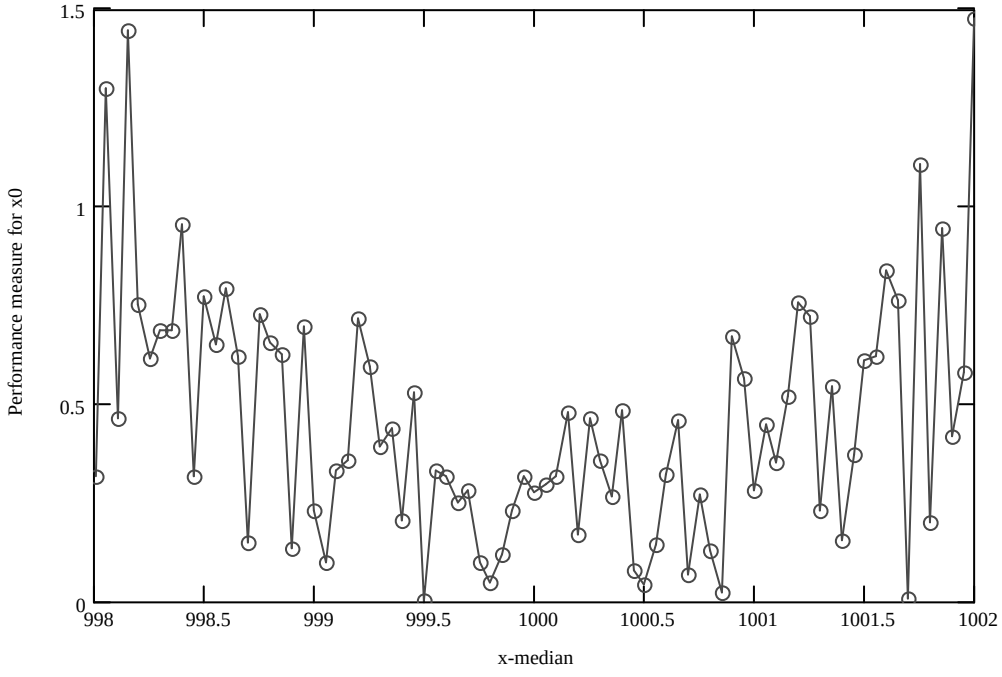

Figure 14: Plot of the performance measure $P(x_0)$ against data location.

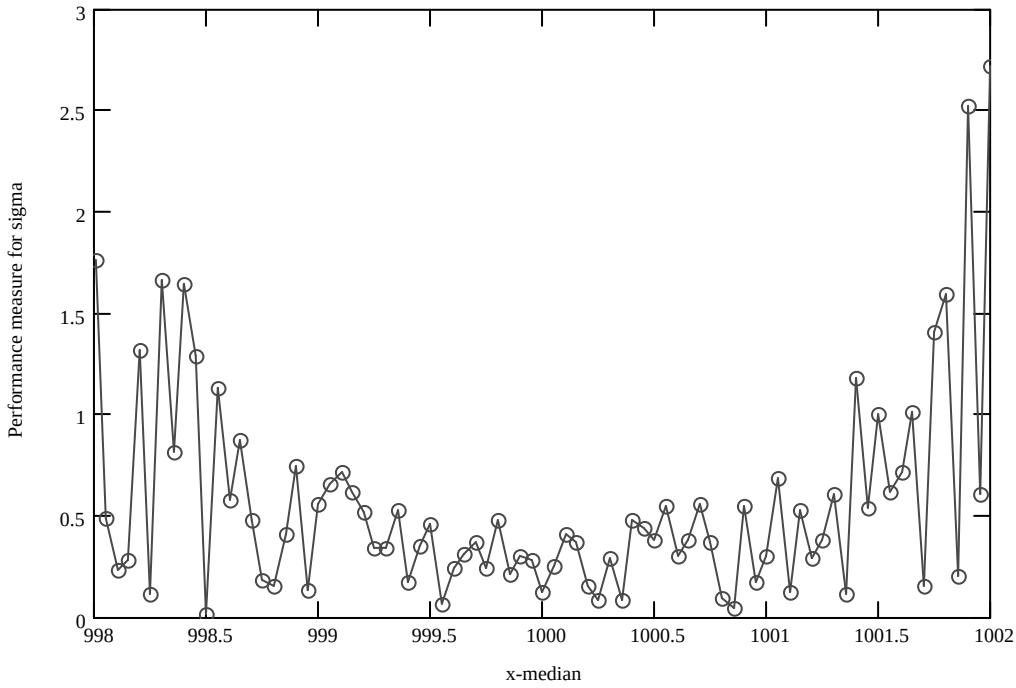

Figure 15: Plot of the performance measure $P(\sigma)$ against data location.

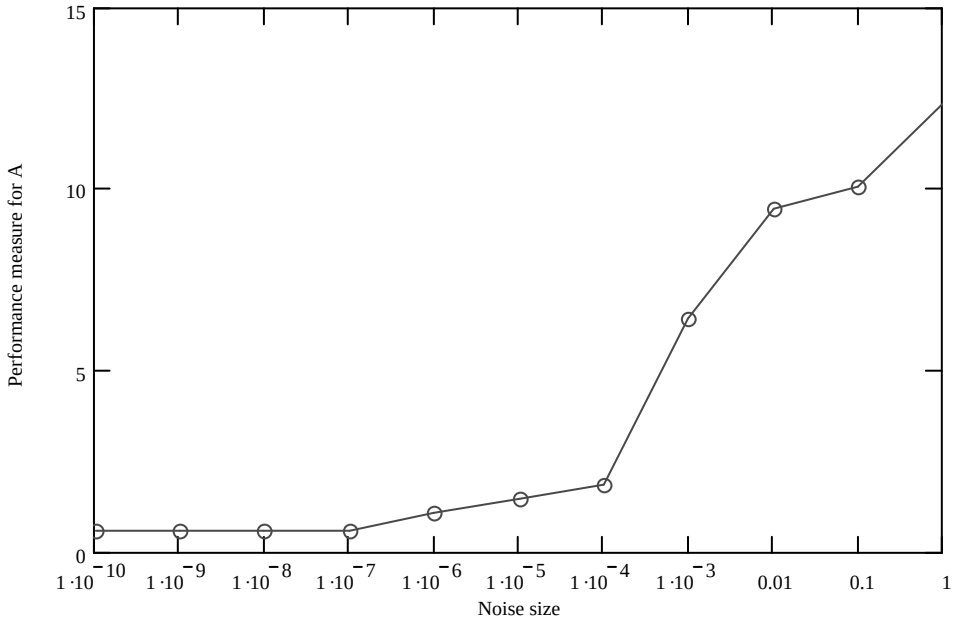

Figure 16: Plot of the performance measure $P(A)$ against noise size.

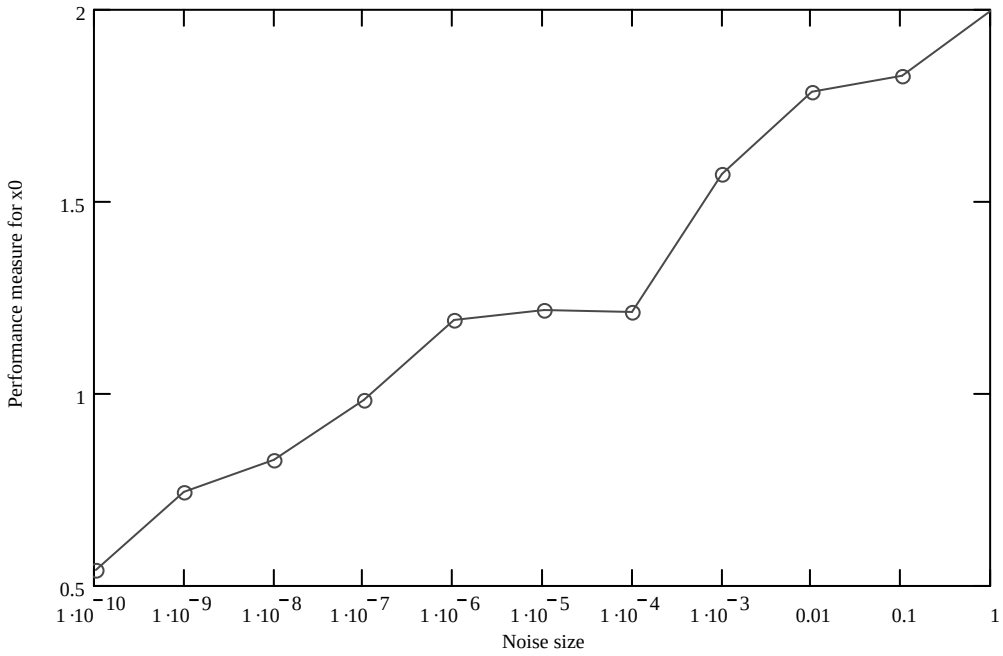

Figure 17: Plot of the performance measure $P(x_0)$ against noise size.

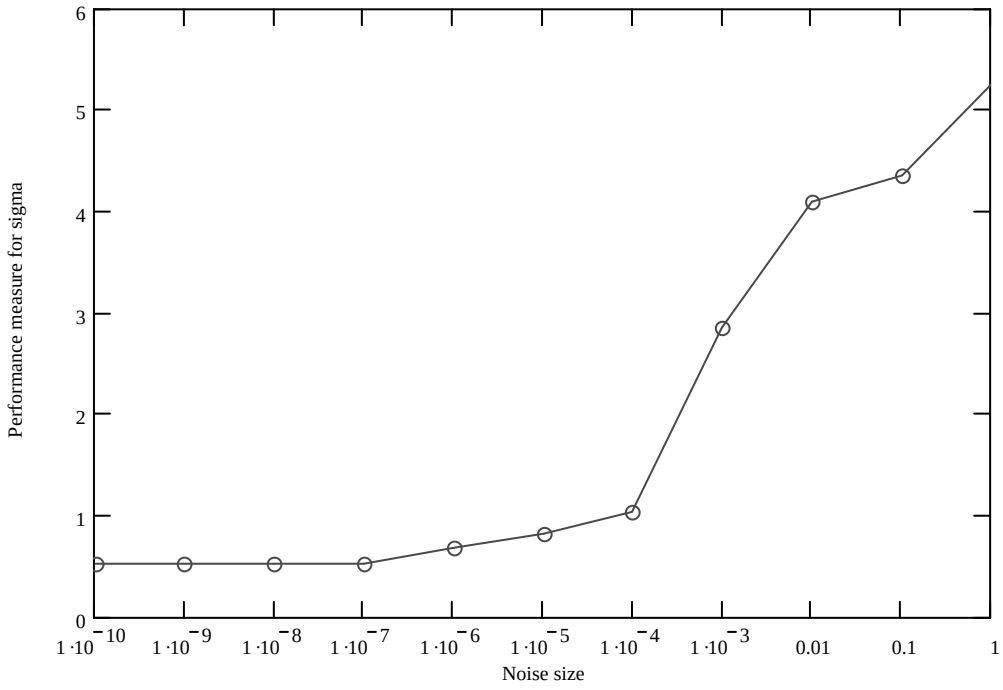

Figure 18: Plot of the performance measure $P(\sigma)$ against noise size.

GENFIT: Sine-wave model

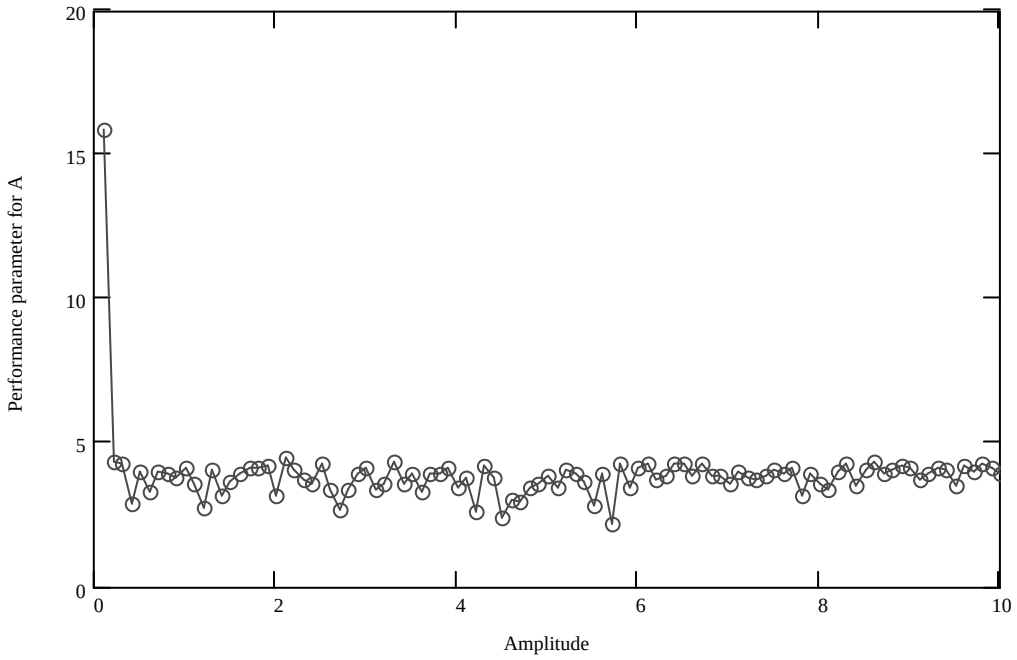

Figure 19: Plot of the performance measure $P(A)$ against amplitude.

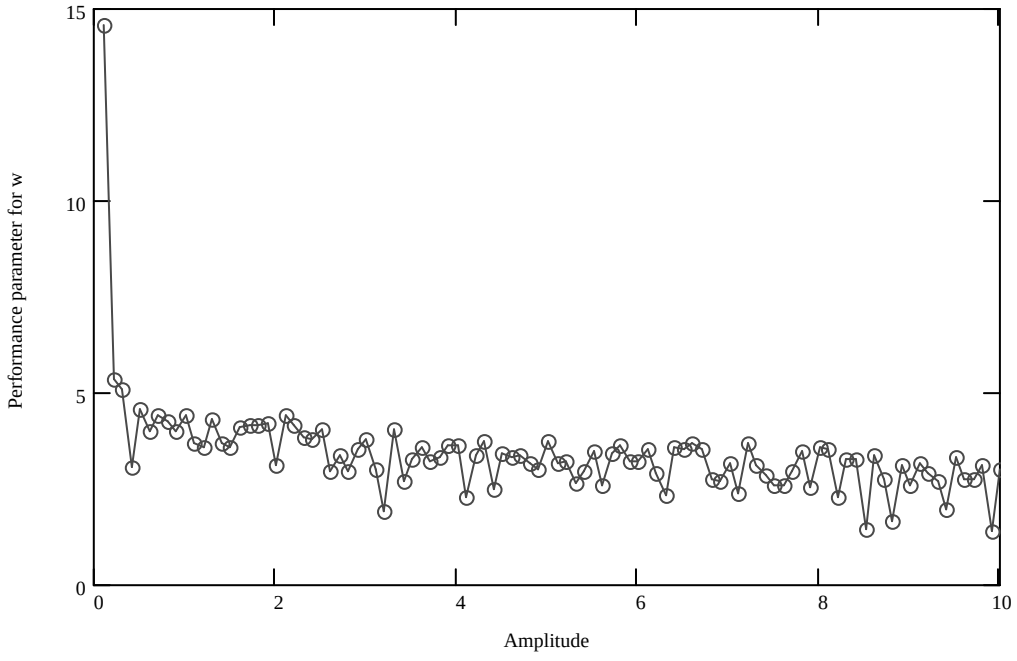

Figure 20: Plot of the performance measure $P(\omega)$ against amplitude.

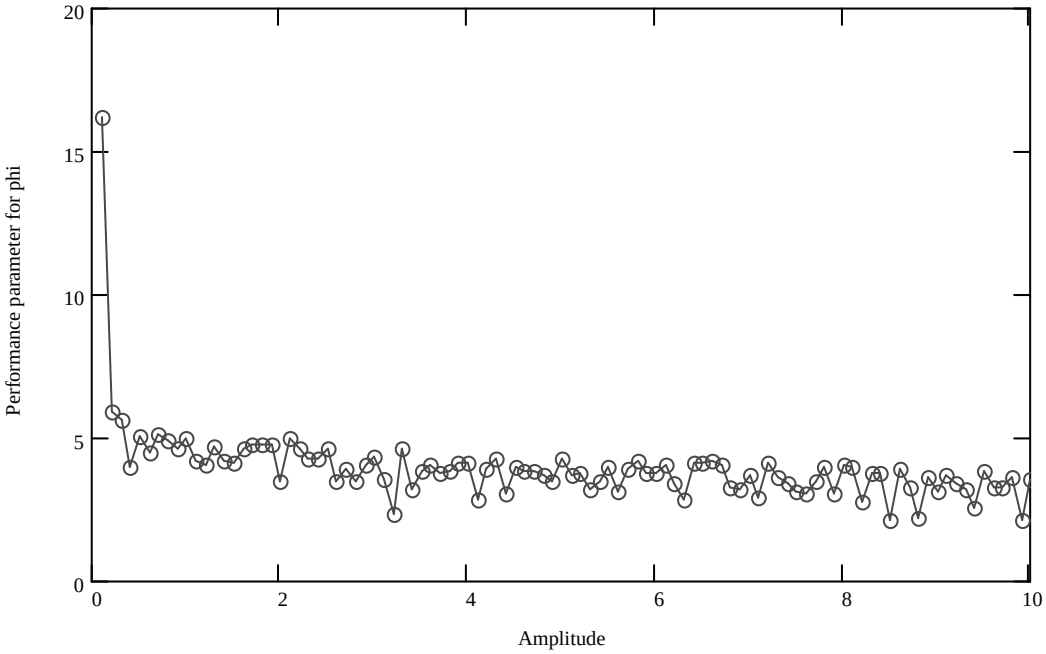

Figure 21: Plot of the performance measure $P(\phi)$ against amplitude.

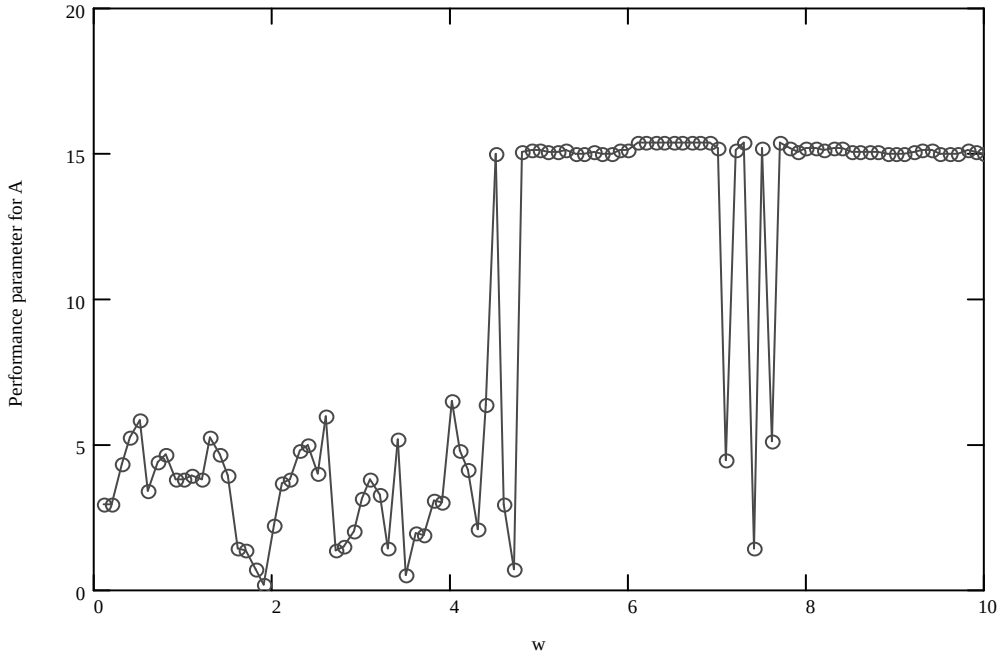

Figure 22: Plot of the performance measure $P(A)$ against wave velocity.

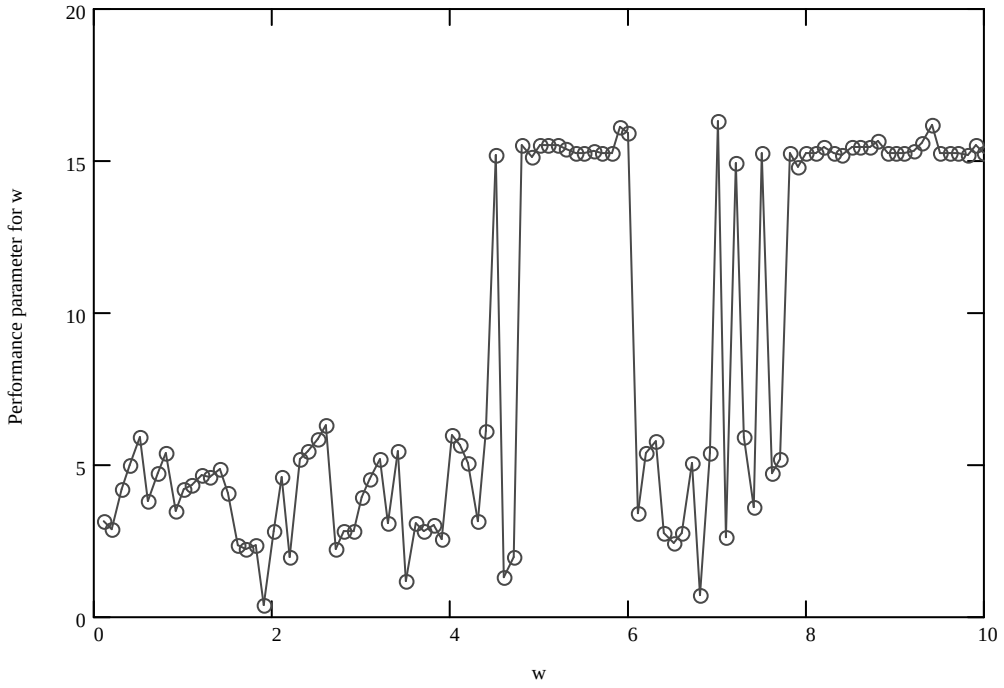

Figure 23: Plot of the performance measure $P(\omega)$ against wave velocity.

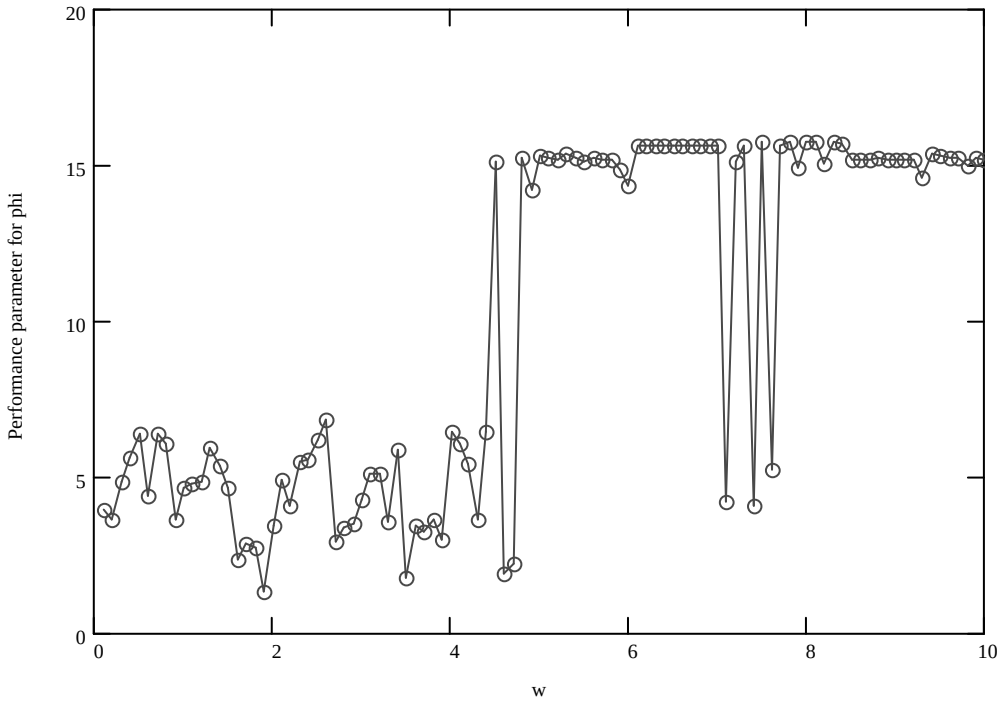

Figure 24: Plot of the performance measure $P(\phi)$ against wave velocity.

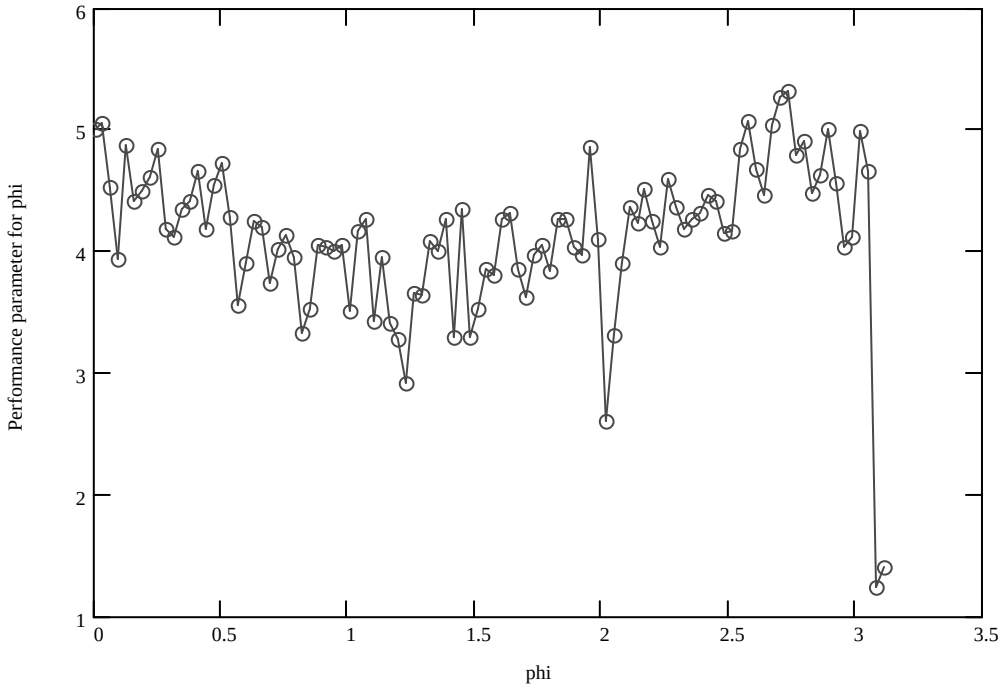

Figure 25: Plot of the performance measure $P(A)$ against phase angle.

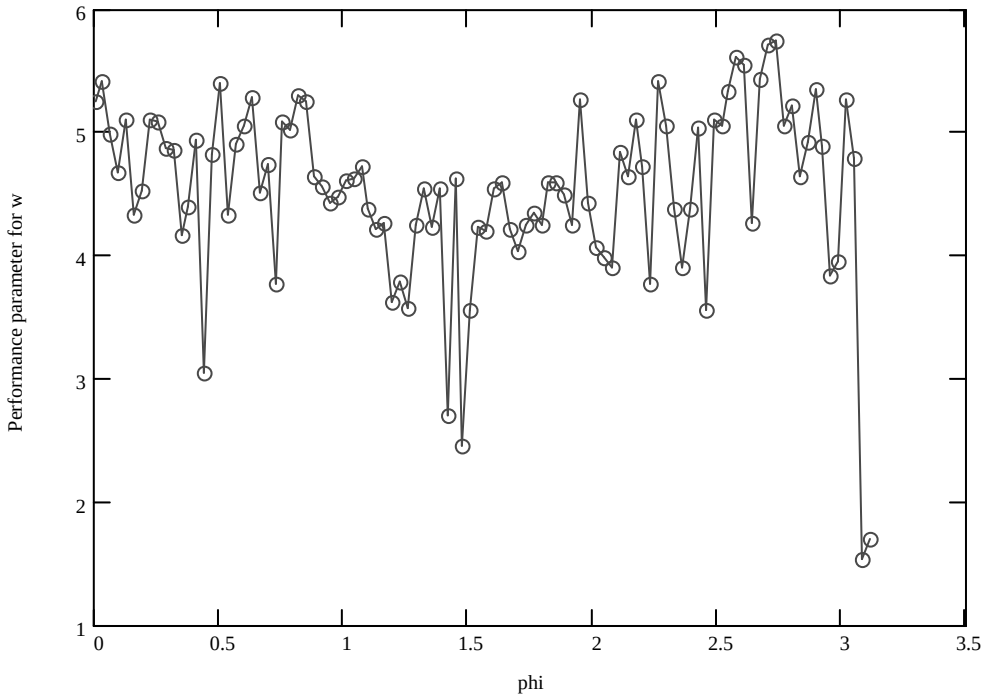

Figure 26: Plot of the performance measure $P(\omega)$ against phase angle.

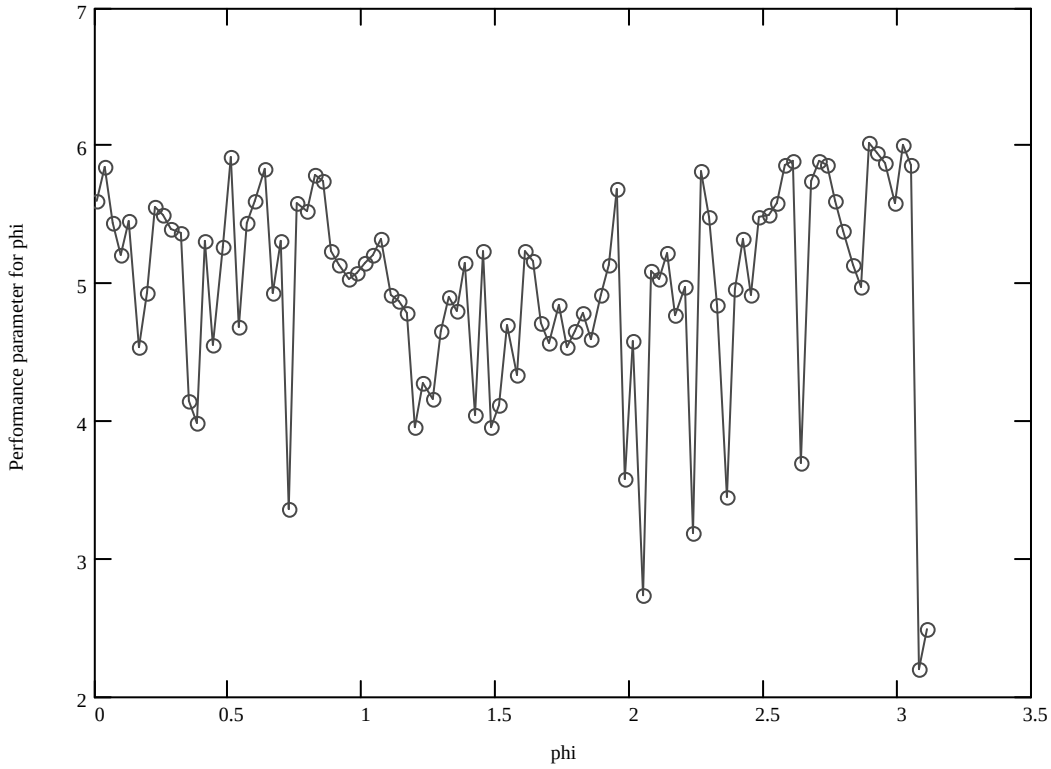

Figure 27: Plot of the performance measure $P(\phi)$ against phase angle.

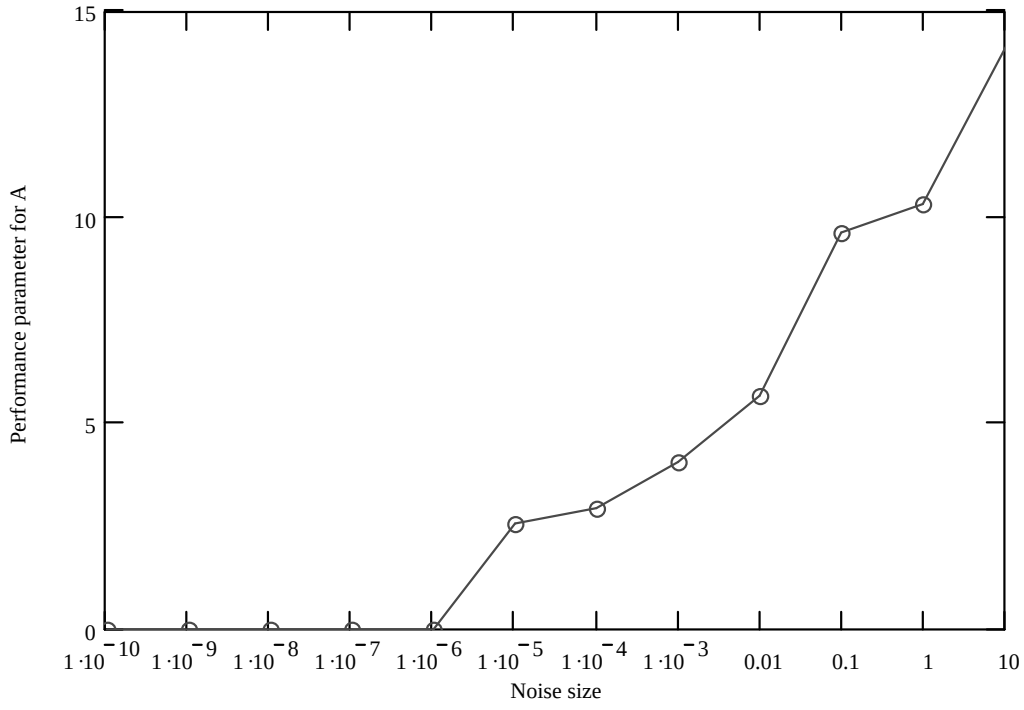

Figure 28: Plot of the performance measure $P(A)$ against noise size.

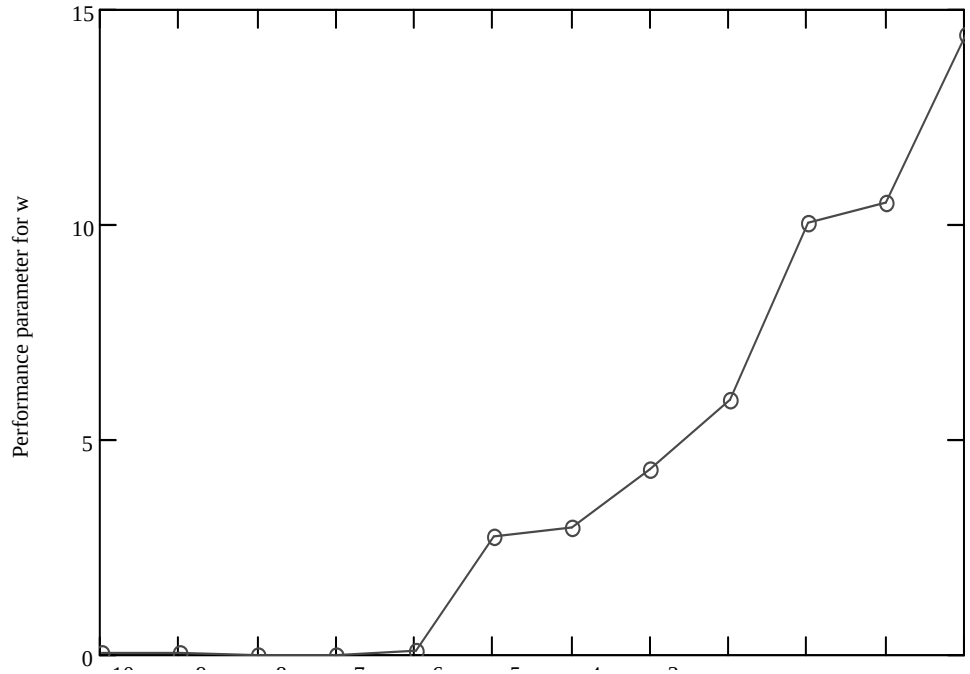

Figure 29: Plot of the performance measure $P(\omega)$ against noise size.

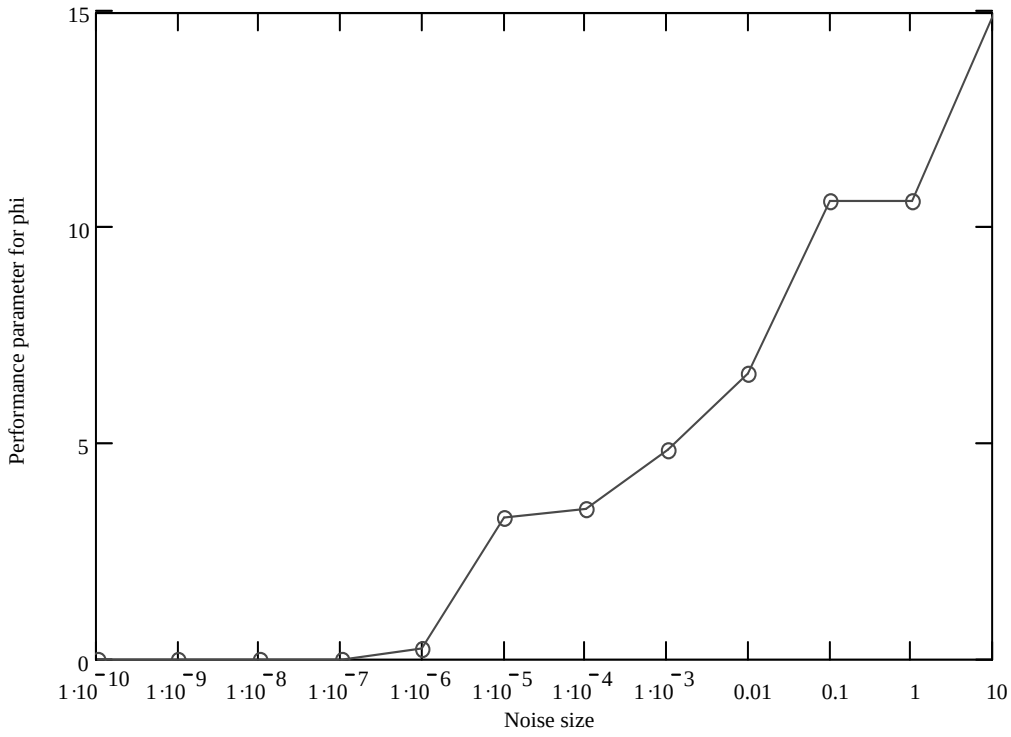

Figure 30: Plot of the performance measure $P(\phi)$ against noise size.

INTERCEPT and SLOPE

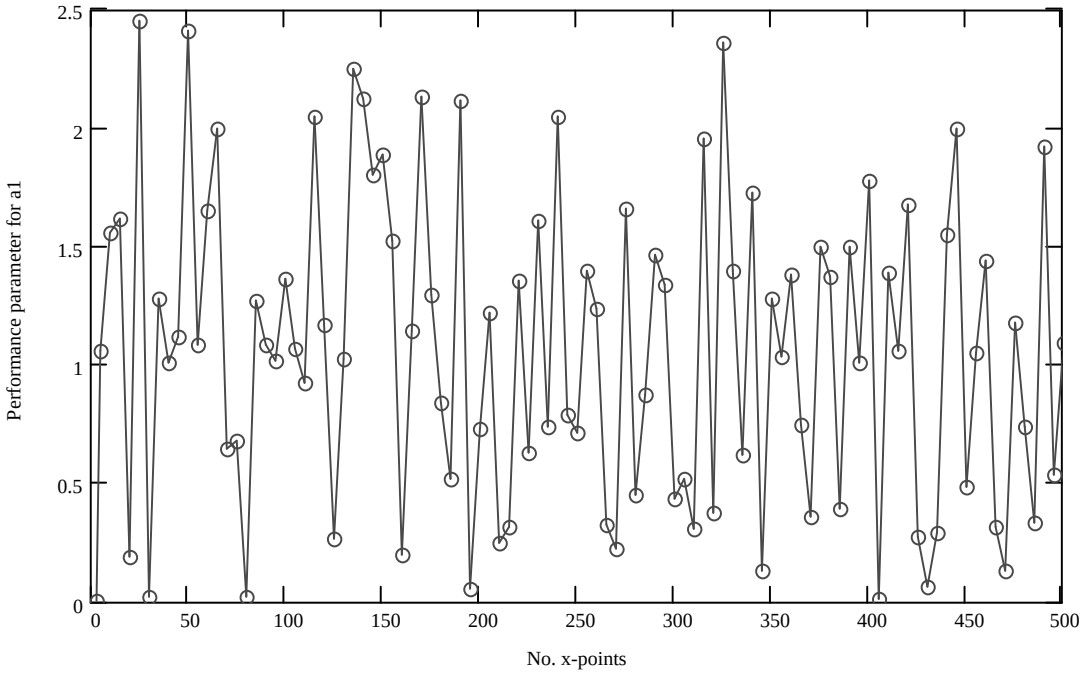

Figure 31: Plot of the performance measure $P(a_1)$ against number of points.

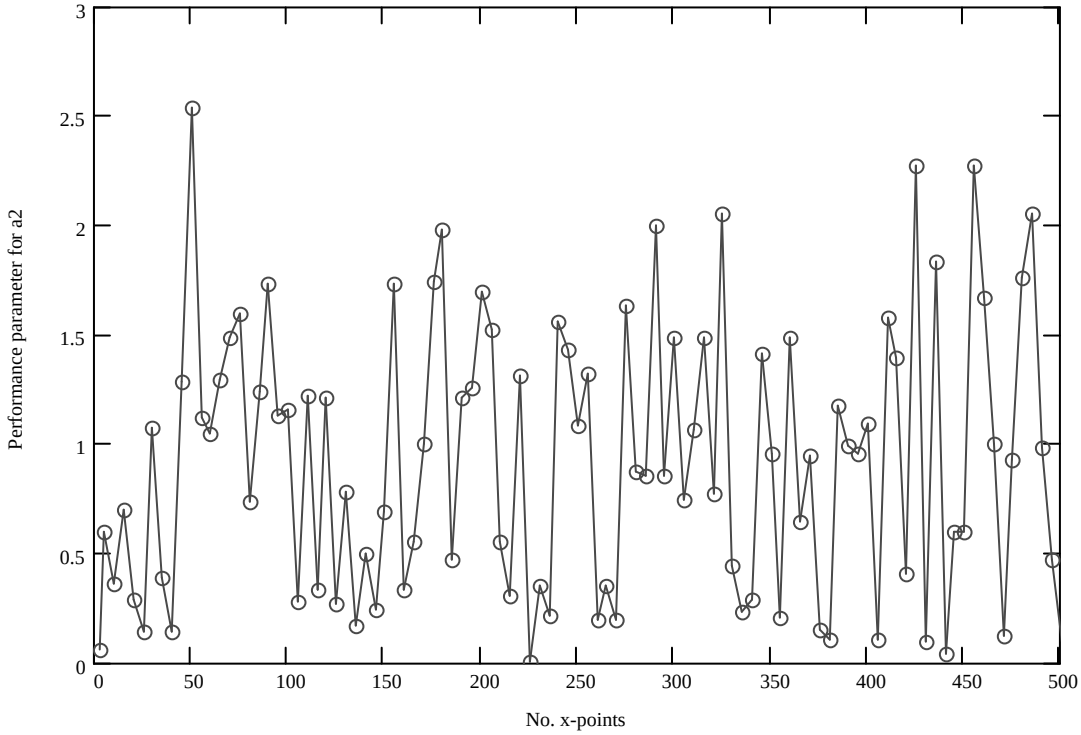

Figure 32: Plot of the performance $P(a_2)$ against number of points.

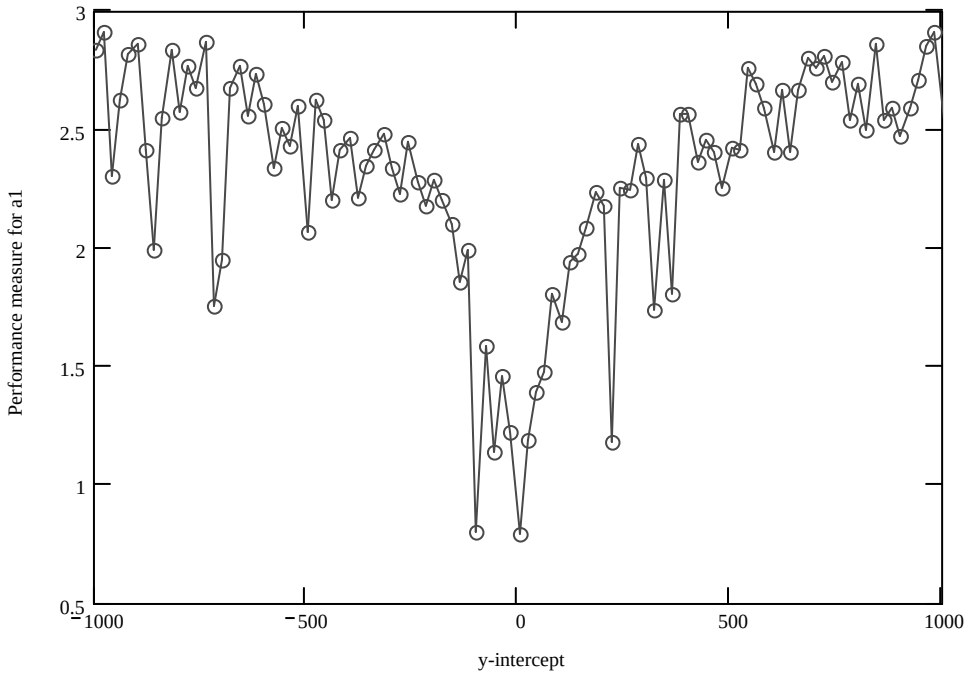

Figure 33: Plot of the performance measure $P(a_1)$ against data location.

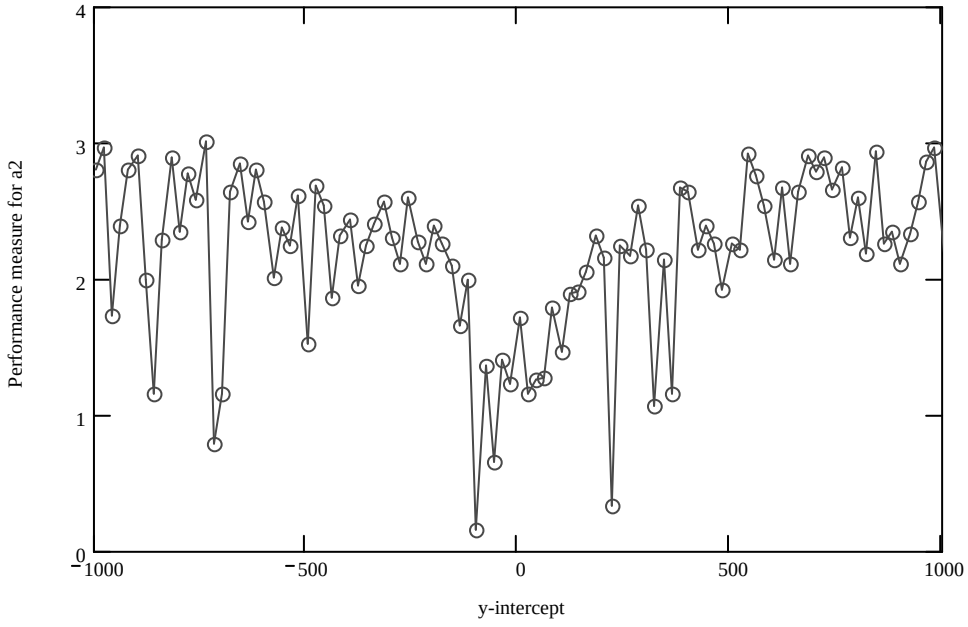

Figure 34: Plot of the performance measure $P(a_2)$ against data location.

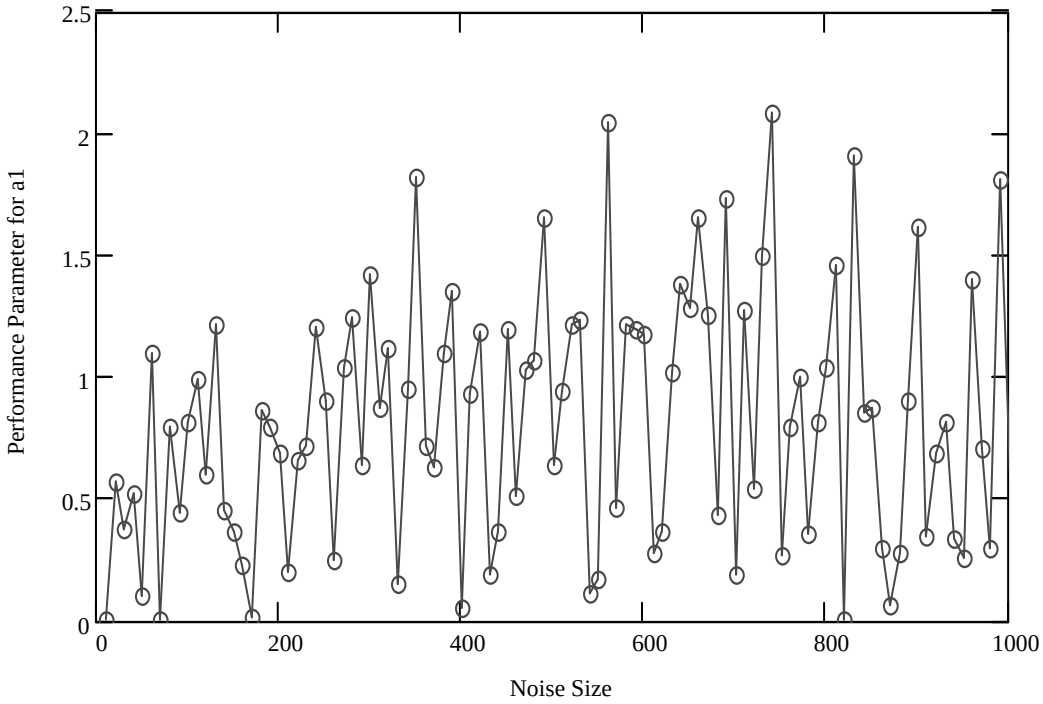

Figure 35: Plot of the performance measure $P(a_1)$ against noise size.

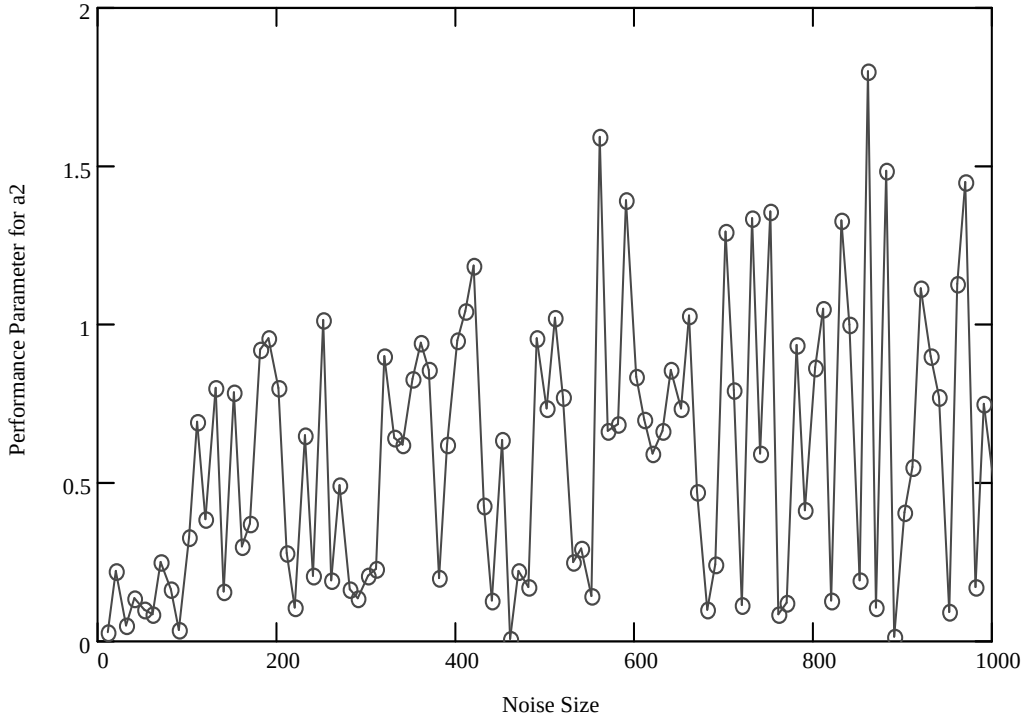

Figure 36: Plot of the performance measure $P(a_2)$ against noise size.

LINFIT

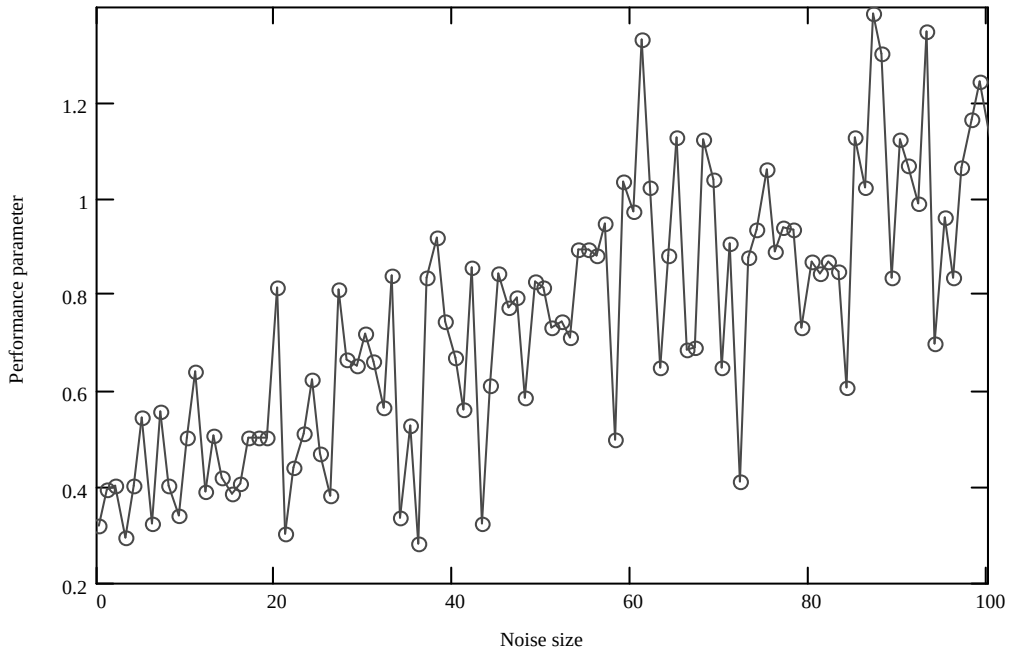

Figure 37: Plot of the performance measure *P*(**a**) against noise size, second order polynomial.

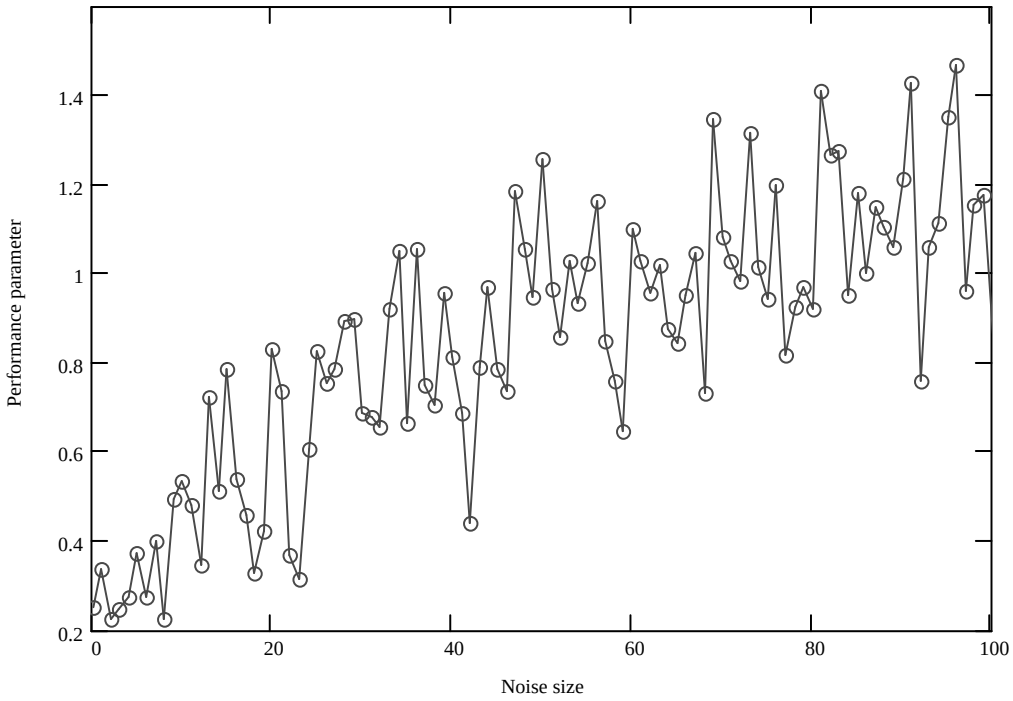

Figure 38: Plot of the performance measure *P*(**a**) against noise size, fourth order polynomial.

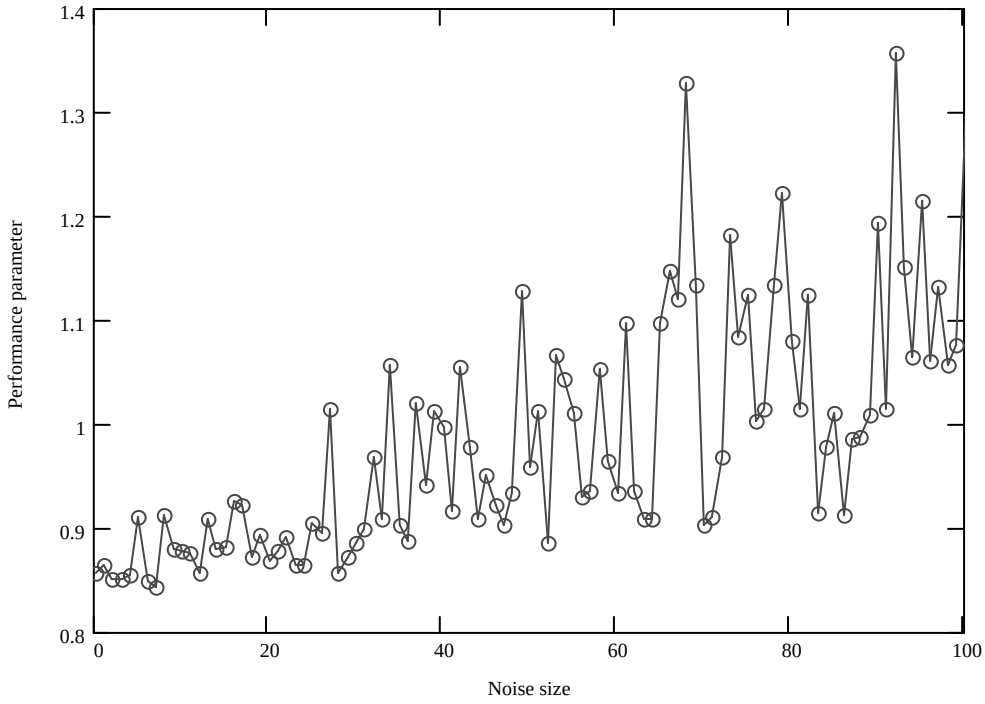

Figure 39: Plot of the performance measure $P(\mathbf{a})$ against noise size, tenth order polynomial.

REGRESS and INTERP

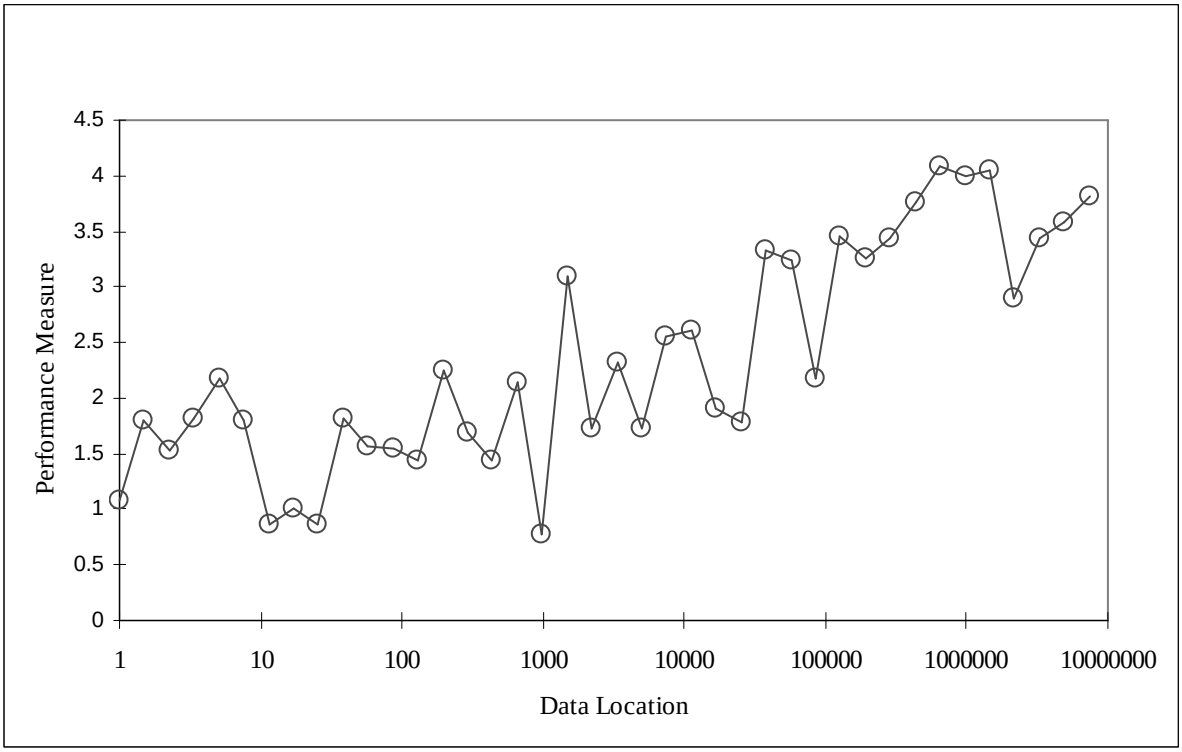

Figure 40: Plot of the performance measure $P(\mathbf{e})$ against data location, second order polynomial.

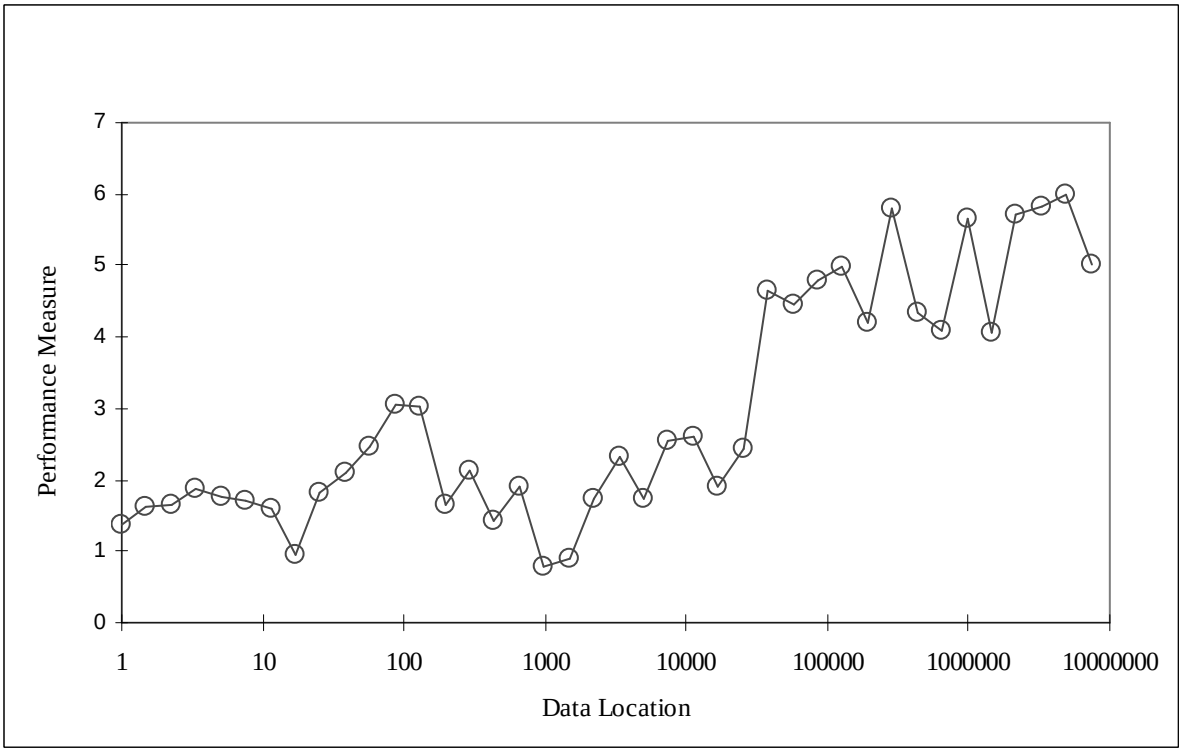

Figure 41: Plot of the performance measure $P(\mathbf{e})$ against data location, fourth order polynomial.

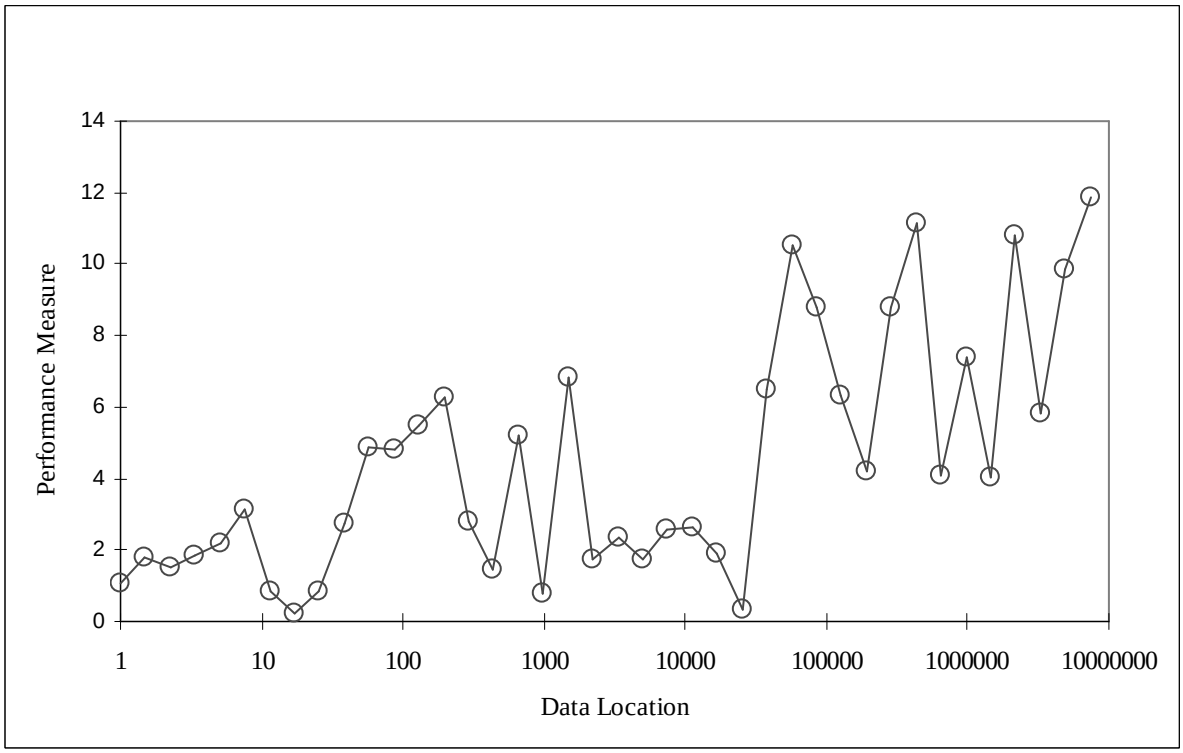

Figure 42: Plot of the performance measure $P(\mathbf{e})$ against data location, tenth order polynomial.

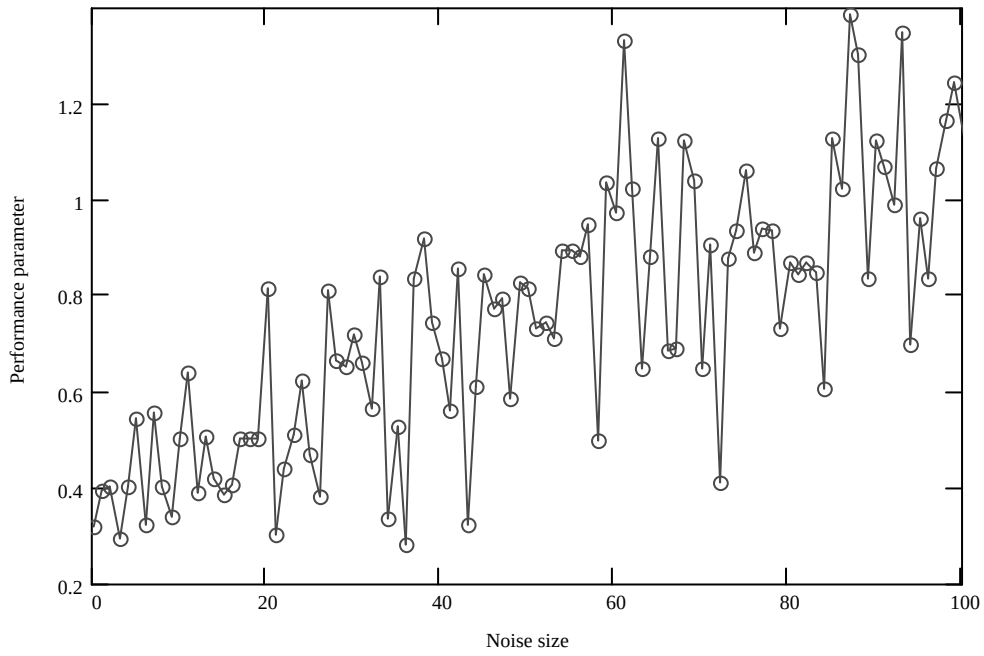

Figure 43: Plot of the performance measure $P(\mathbf{e})$ against noise size, second order polynomial.

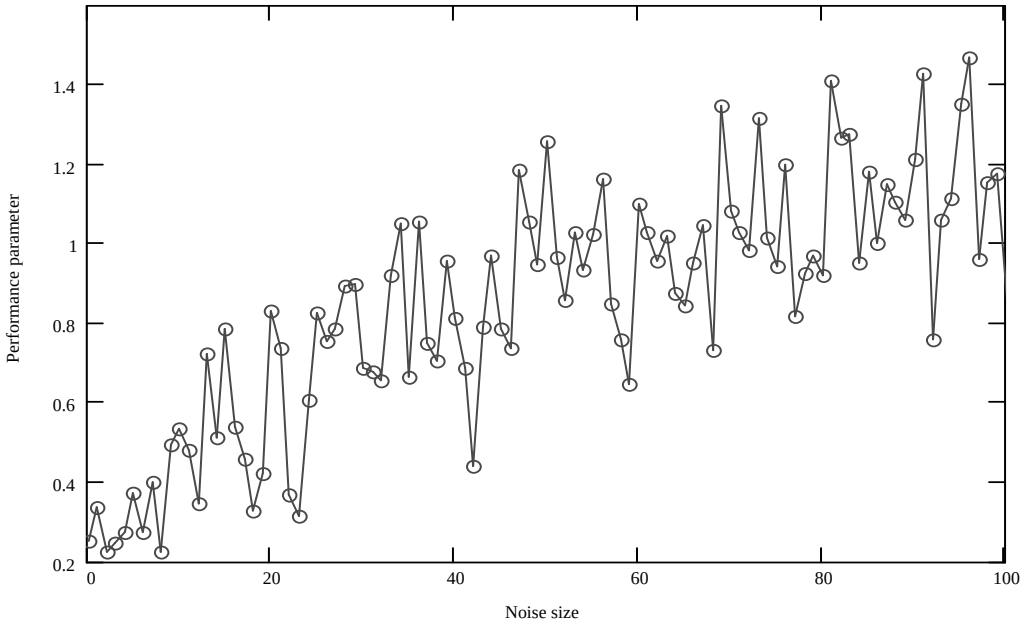

Figure 44: Plot of the performance measure *P*(**e**) against noise size, fourth order polynomial.

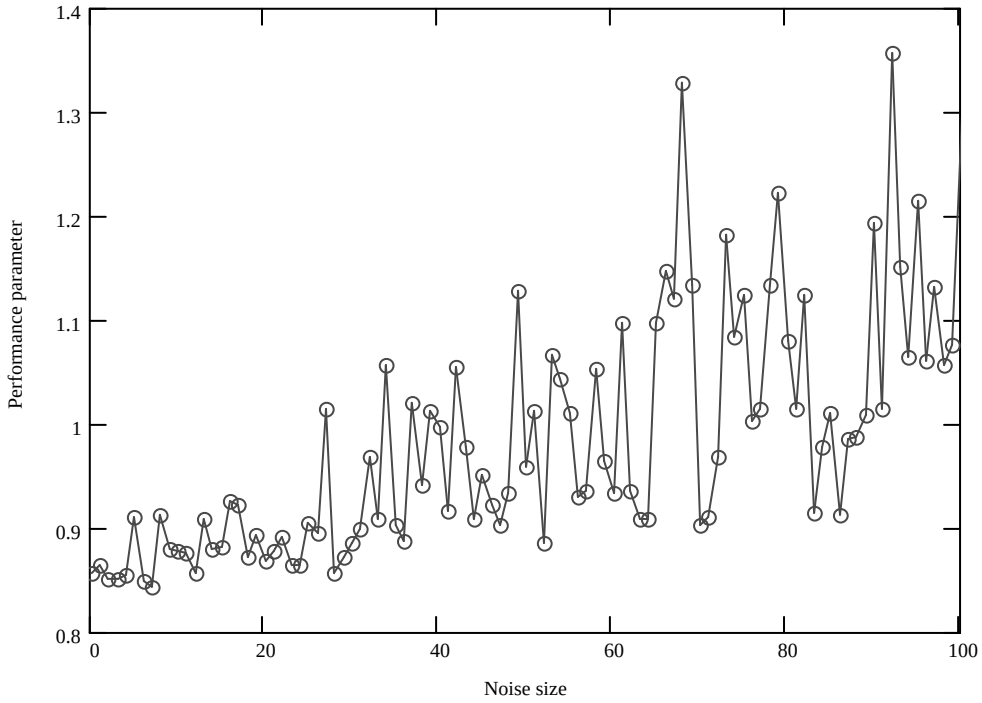

Figure 45: Plot of the performance measure *P*(**e**) against noise size, tenth order polynomial.

Appendix B: Results for Mathematical Functions

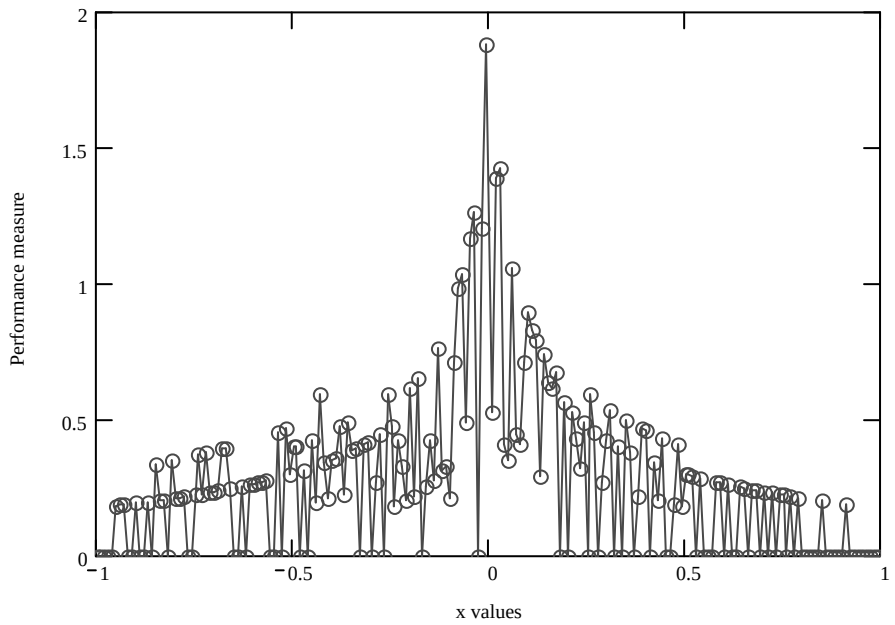

Figure 46: Plot of the performance measure for software test T3.

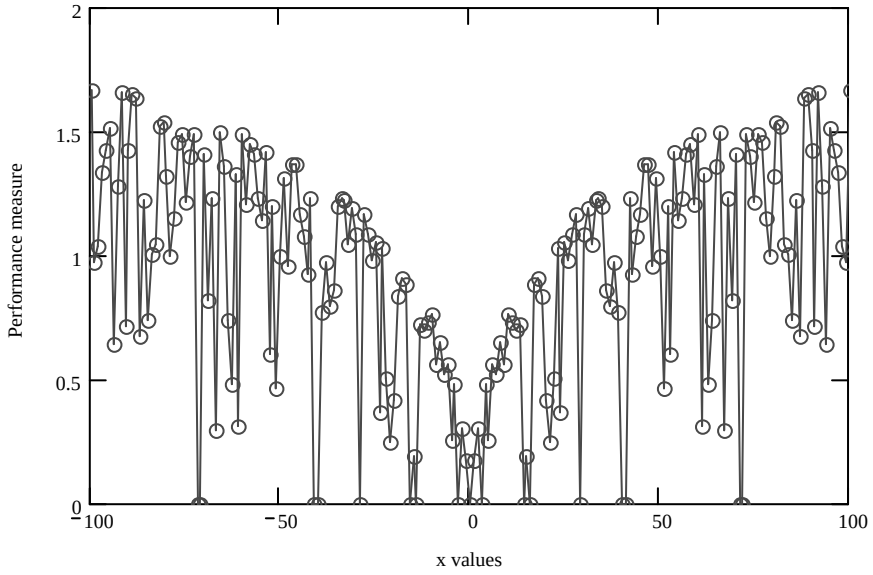

Figure 47: Plot of the performance measure for software test T5.

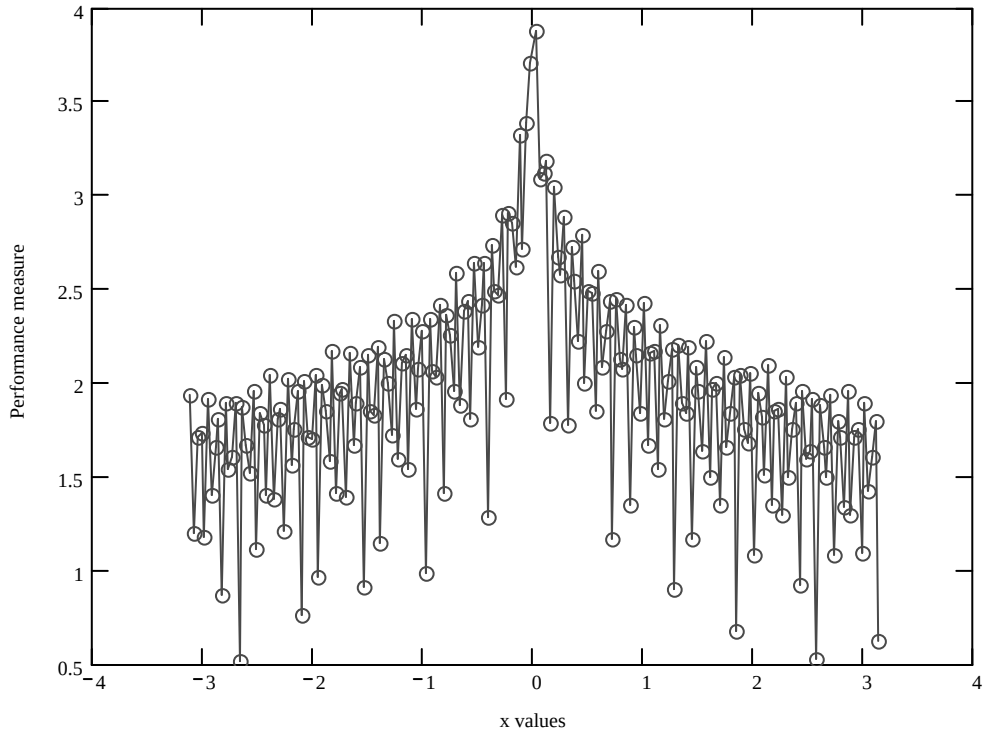

Figure 48: Plot of the performance measure for software test T9.

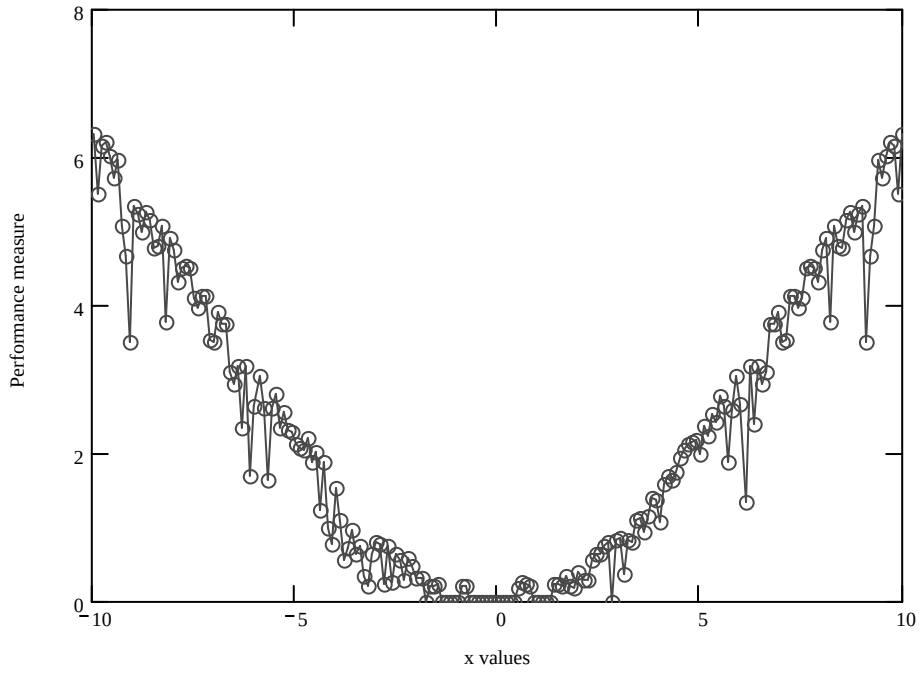

Figure 49: Plot of the performance measure for software test T15.

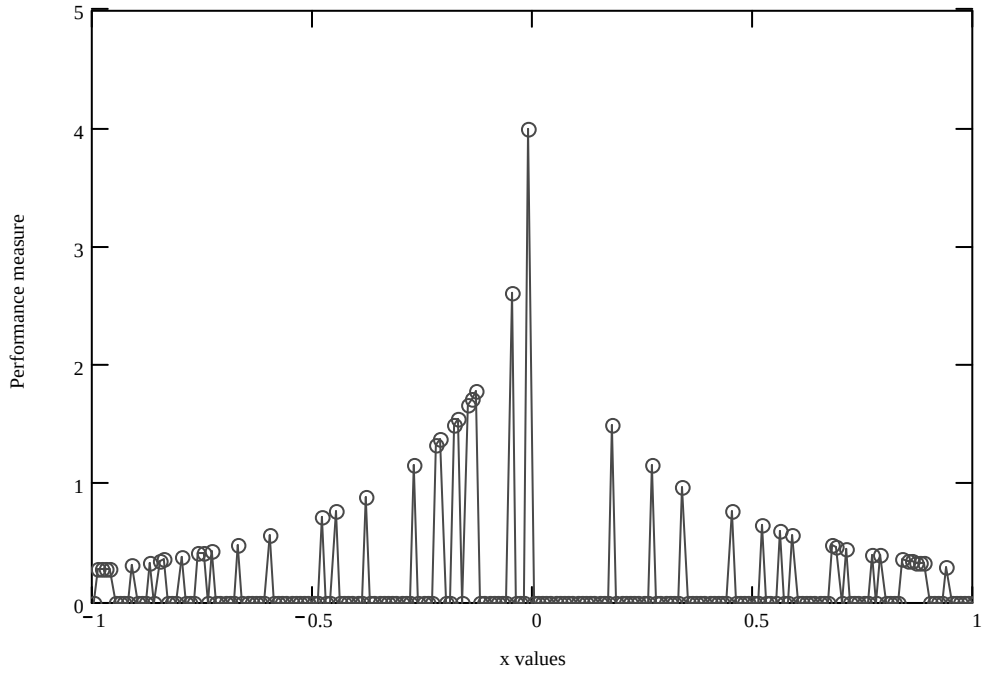

Figure 50: Plot of the performance measure for software test T19.

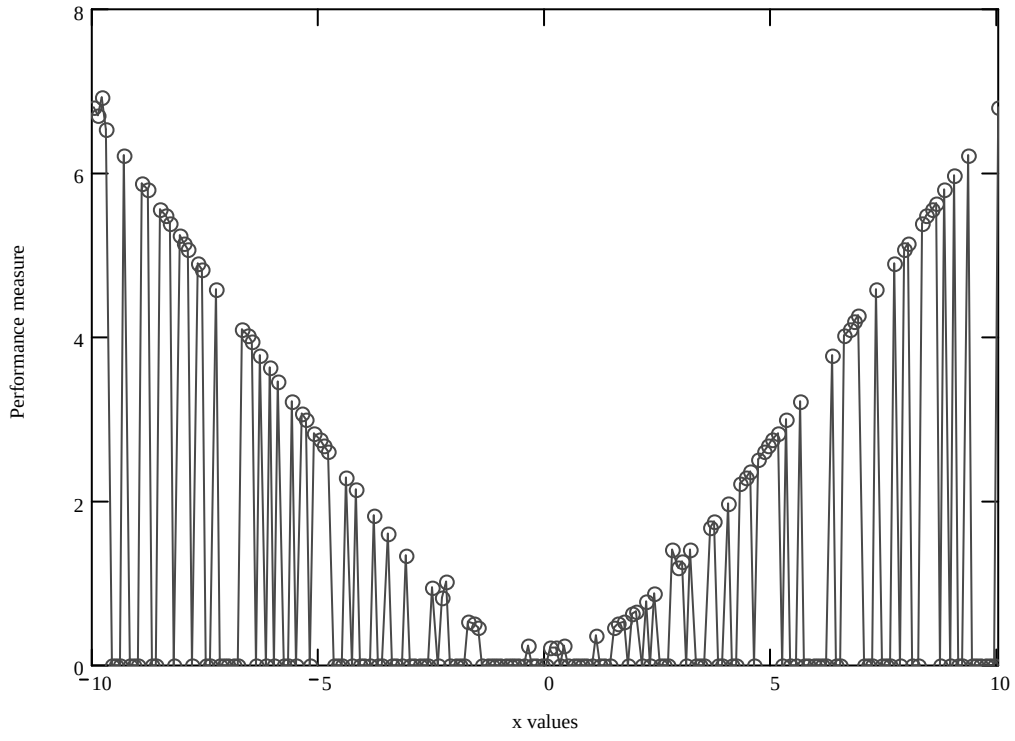

Figure 51: Plot of the performance measure for software test T20.

Appendix C: Results for Statistical Distributions

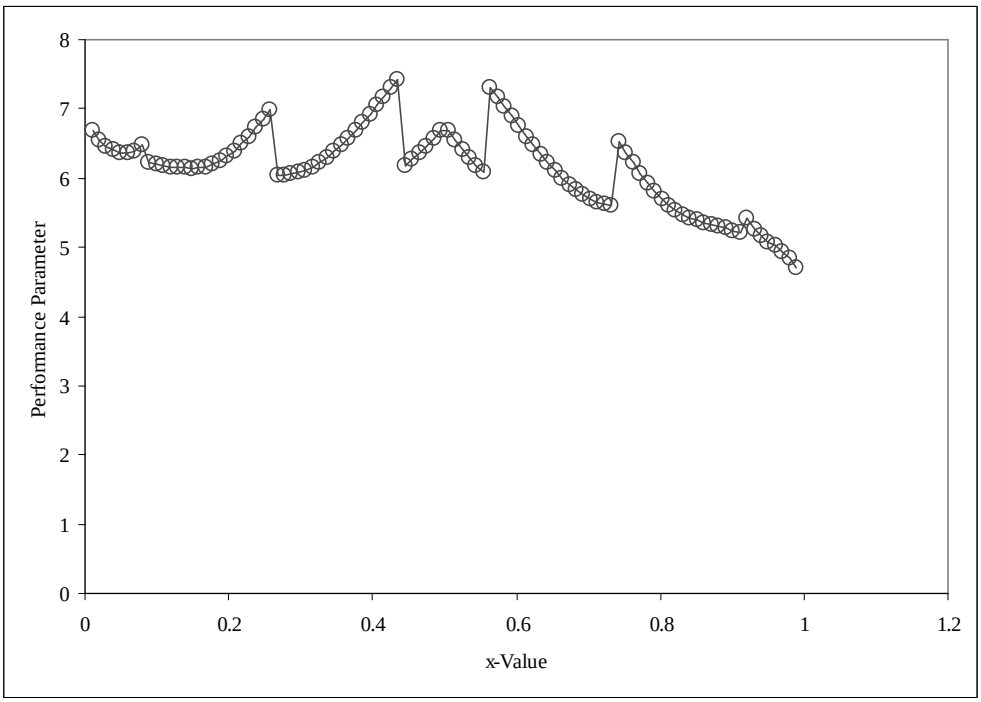

Figure 52: Plot of the performance measure for the comparison of PBETA(*x*, 0.5, 0.5) with the equivalent IMSL function.

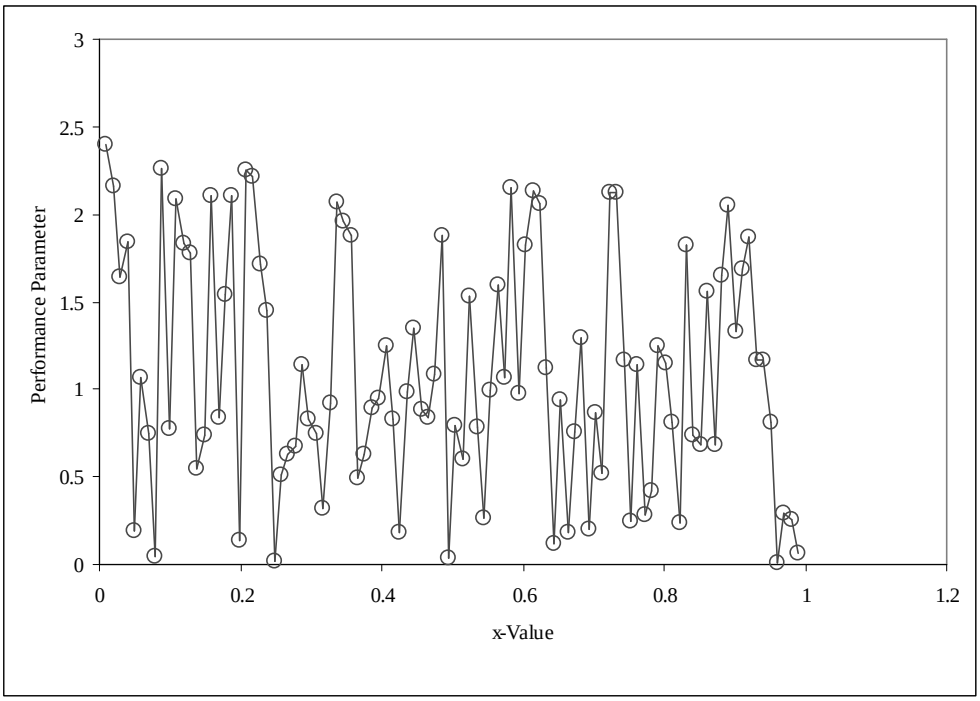

Figure 53: Plot of the performance measure for the comparison of QBETA(*x*, 0.5, 0.5) with the equivalent IMSL function.

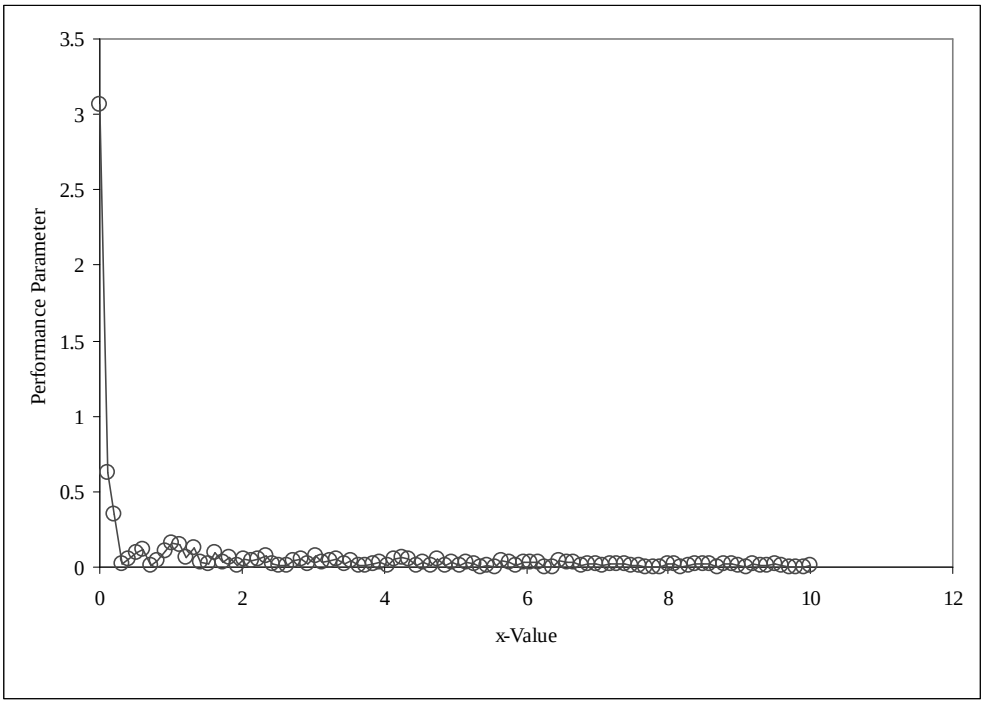

Figure 54: Plot of the performance measure for the comparison of DCHISQ(*x*, 0.5, 0.5) with the equivalent IMSL function.

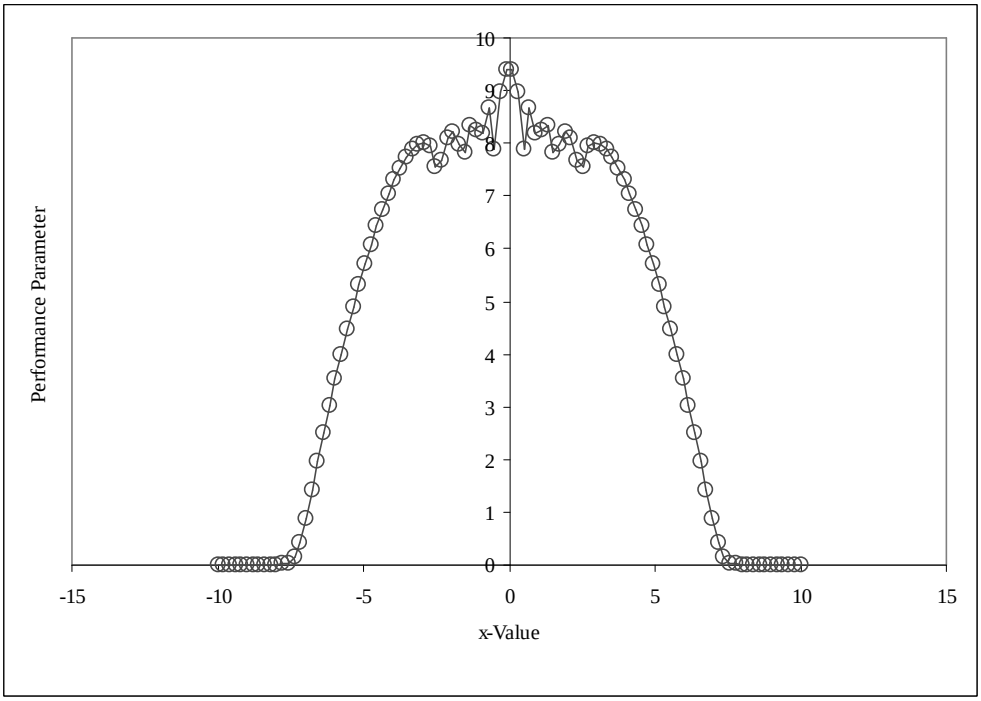

Figure 55: Plot of the performance measure for the comparison of DNORM(*x*, 0.5, 0.5) with the equivalent IMSL function.