NPL Report CMSC 48/04

Report to the National Measurement System Directorate, Department of Trade and Industry

From the Software Support for Metrology Programme

Testing algorithms for free-knot spline approximation

By
P M Harris and I M Smith

March 2004

Testing algorithms for free-knot spline approximation

P M Harris and I M Smith
Centre for Mathematics and Scientific Computing

March 2004

ABSTRACT

A case study is presented for testing and validating contributions to EUROMETROS. It concerns the application of numerical software testing and algorithm testing techniques to software implementations of (approximate) algorithms for solving the free-knot spline approximation problem. Knot placement strategies are used as the basis of the approximate algorithms. The strategies include selecting knots (a) uniformly-spaced, (b) based on the distribution of the data abscissa values, and (c) using a knot-insertion strategy. A number of testing techniques are applied, including an analysis of the algorithms, using software implementations of the algorithms, using data generators for the algorithms, and simulation. The results of the testing are presented and described.

ISSN 1471–0005

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

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

Contents

1 Introduction

There are many well-defined mathematical and statistical problems that arise in metrology. However, for some of these determining their solution may be difficult, and for this reason an approximate or nearby problem is often introduced, leading to a more tractable problem that is easier or quicker to solve. Algorithms specified in standards or implemented in software libraries, such as EUROMETROS [3], may implicitly or explicitly provide an approximate solution or the solution to a nearby problem. *Algorithm testing* [5] is concerned with understanding the effect of solving an approximate problem by measuring the departure of the solution so obtained from the solution to the correct problem. It is also about understanding, for a given application, whether the solution to the approximate problem is fit for purpose. In this context, fitness for purpose would mean that inferences derived from the solution to the approximate problem would be the same as those derived from the solution to the correct problem.

Empirical models are important to metrology in cases where the knowledge of the underlying physics for a measurement system is insufficient to characterise it completely. For empirical models depending on one variable, polynomial and particularly *polynomial spline curves*, when used with care, are generally very satisfactory for representing data. A polynomial spline curve is composed of a sequence of polynomial pieces joined together at points called *knots* and in such a way as to ensure smoothness of the complete curve. Spline curves provide a flexible class of functions that are effective for representing a wide variety of shapes. However, the knots generally have no physical meaning for the metrologist, and yet the effectiveness of a spline representation of data can depend critically on their number and positions. Consequently, metrology users require assistance with knot placement via appropriate algorithms and software.

Such assistance is provided in the form of *knot placement strategies* [13, 14]. The purpose of these strategies is to provide a 'sensible', if not necessarily 'optimal', set of knots. If we regard the problem of providing an optimal set of knots as the problem we set out to solve, viz., the 'correct' problem, then knot placement strategies provide the basis for algorithms designed to solve this problem *approximately*. If metrology users are to have confidence in such algorithms, it is important to investigate (and understand) the effect of using them as approaches to solving the correct problem. This is particularly so as *software* implementing a number of knot placement strategies is available to metrology users through EUROMETROS (see below). A requirement for such an investigation is that the correct problem referred to above is defined unambiguously, and a metric introduced to measure optimality. In this work this is done in terms of the *free-knot spline approximation problem* (section 2.2).

The Software Support for Metrology (SS*f*M) programme, through its projects on

discrete modelling and empirical models, has made available, through EUROMETROS, software to help metrology users work effectively with empirical models. In particular, the following software packages are available:

NPLFit Software for modelling experimental data using polynomial and polynomial spline curves. The software permits polynomial curves (of arbitrary order) and spline curves (of arbitrary order and specified general knots) to be fitted to data. Simple facilities for validating the choice of a fitted model are provided. In addition, basic operations on the fitted model, including evaluation, differentiation and integration may be undertaken, and the evaluation of the uncertainties associated with the results of these basic operations is also supported.

KPLFit Software implementing a number of knot placement strategies to help metrology users select a sensible set of knots to be used when representing experimental data by spline curves [13, 14]. The software provides as its output a set of knots that can be used as input to the spline fitting routines in NPLFit.

FeatureExtraction Software extending the functionality of NPLFit to include feature extraction from spline curves fitted to experimental data [15], viz., the determination of zeros, maxima and minima, points of inflexion, and peak widths, and the associated uncertainties.

The focus of the work reported here is on (a subset of) the knot placement strategies implemented as part of the software package KPLFit. The purpose of the work is to undertake *validation* of software implementations of those strategies, where the validation is to include:

Numerical software testing, viz., investigating the numerical accuracy of the results returned by software implementations of the knot placement strategies for the problems purportedly solved by those strategies [6, 12].

Algorithm testing, viz., investigating the extent to which the strategies implemented solve the free-knot spline approximation problem [5].

The work constitutes a case study for testing and validating contributions to EUROMETROS [3]. It accounts for related work on algorithm testing concerning the assessment of Type A1 surface texture reference artefacts [18] and free-form surface fitting to co-ordinate data [1].

The report is organised as follows. In section 2 the representation of polynomial spline curves in terms of B-splines is reviewed. The free-knot spline approximation problem is described, as are the knot placement strategies considered as the

basis for approximate algorithms for solving the free-knot spline approximation problem. Section 3 contains a summary of generic methods for algorithm testing. In sections 4, 5 and 6 the *tools* for applying the methods of algorithm testing to the algorithms considered in this study are presented. The tools include specifications of the 'computational aims' for the problems solved, software implementations of algorithms for solving the problems specified by the computational aims, and data generators for the computational aims. Section 7 contains the results of testing the numerical correctness of the above software implementations, and section 8 the results of applying the methods of algorithm testing. Section 9 contains our conclusions.

2 Data approximation by polynomial splines

In this section the representation of polynomial spline curves in terms of B-splines is reviewed. The free-knot spline approximation problem, which constitutes the 'correct' problem for this study, is described, as are the knot placement strategies that are the basis of approximate algorithms for solving the free-knot spline approximation problem.

2.1 Univariate polynomial splines

Let $I := [x_{\min}, x_{\max}]$ be an interval of the x-axis, and

$$x_{\min} < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{N-1} \leq \lambda_N < x_{\max}$$

a partition on I. Define sub-intervals $I_j,\ j = 0, \ldots, N$, of I by

$$I_j = \begin{cases} [x_{\min}, \lambda_1)\,, & j = 0, \\ [\lambda_j, \lambda_{j+1})\,, & j = 1, \ldots, N-1, \\ [\lambda_N, x_{\max}]\,, & j = N. \end{cases}$$

Then, a polynomial spline $s(x)$ of order n (degree $n-1$) on I is a piecewise polynomial of order n on $I_j,\ j = 0, \ldots, N$. The spline $s(x)$ is C^{n-k-1} at λ_j if card$(\lambda_\ell = \lambda_j, \ell \in \{1, \ldots, N\}) = k$.[1] So, for example, a spline $s(x)$ of order $n = 4$ for which the points λ_j are distinct (cardinality one) is a piecewise cubic polynomial of continuity class C^2, i.e., continuous in value, first and second derivatives, at the points λ_j.

The partition points $\boldsymbol{\lambda} = \{\lambda_j\}_1^N$ are the (interior) *knots* of s. To specify the complete set of knots needed to define s on I in terms of B-splines (below), the

[1] card(A) is used to denote the *cardinality* of the set A, i.e., the number of elements in the set.

knots $\boldsymbol{\lambda}$ are augmented by exterior knots $\{\lambda_j\}_{1-n}^{0}$ and $\{\lambda_j\}_{N+1}^{q}$, $q = N + n$, satisfying

$$\lambda_{1-n} \leq \cdots \leq \lambda_0 \leq x_{\min}, \qquad x_{\max} \leq \lambda_{N+1} \leq \cdots \leq \lambda_q.$$

For many purposes, a good choice [9] of exterior knots is

$$\lambda_{1-n} = \cdots = \lambda_0 = x_{\min}, \qquad x_{\max} = \lambda_{N+1} = \cdots = \lambda_q. \tag{1}$$

It readily permits derivative boundary conditions to be incorporated in spline approximants [8].

On I, s has the *B-spline representation* [7, 16]

$$s(x) := s(\mathbf{c}, \boldsymbol{\lambda}; x) = \sum_{j=1}^{q} c_j N_{n,j}(\boldsymbol{\lambda}; x), \tag{2}$$

where $N_{n,j}(\boldsymbol{\lambda}; x)$ is the *B-spline* [7, 16] of order n with knots $\{\lambda_k\}_{j-n}^{j}$ and $\mathbf{c} = (c_1, \ldots, c_q)^{\mathrm{T}}$ are the *B-spline coefficients* of s.[2] Each $N_{n,j}(\boldsymbol{\lambda}; x)$ is non-negative and has compact support $[\lambda_{j-n}, \lambda_j]$.[3] The B-spline basis $\{N_{n,j}(\boldsymbol{\lambda}; x)\}_{j=1}^{q}$ for splines of order n with knots $\boldsymbol{\lambda}$ is generally very well-conditioned [9].

2.2 Least-squares data approximation by splines

Two types of least-squares data approximation (or data modelling) by splines are considered:[4]

Fixed-knot approximation. Determination of the spline coefficients $\mathbf{c}$ for given data, spline order n and *fixed* knots $\boldsymbol{\lambda}$.

The fixed-knot spline approximation problem is stated as follows: given data $\{(x_i, y_i)\}_1^m$, spline order n, and knots $\boldsymbol{\lambda}$, find the spline coefficients $\mathbf{c}$ to solve

$$\min_{\mathbf{c}} \sum_{i=1}^{m} \{y_i - s(\mathbf{c}, \boldsymbol{\lambda}; x_i)\}^2 .$$

[2] Writing s and $N_{n,j}$ as functions of the *interior* knots $\boldsymbol{\lambda}$ (and not the endpoints of I and the exterior knots) is deliberate. This is done because in the context of data approximation by splines (section 2.2), the interior knots are regarded as *parameters* whose values are to be determined. Henceforth, it will be assumed that the endpoints $x_{\min}$ and $x_{\max}$ of I are given, and the exterior knots are set according to (1).

[3] A function f of a real variable x has *compact support* if it is zero outside an interval, denoted by supp(f), that is closed and bounded.

[4] It is assumed that the data approximation problems are *unweighted*. However, generalisations are possible to account for (possibly unequal) uncertainties associated with the (measured) ordinate values, mutual dependencies between the ordinate values, as well as uncertainties associated with both the ordinate and abscissa values [10].

In terms of the B-spline representation (2) for s, the following algebraic formulation

$$\min_{\mathbf{c}} \|\mathbf{e}\|^2, \quad \mathbf{e} = \mathbf{y} - A\mathbf{c}, \tag{3}$$

is obtained, where $\mathbf{y} = (y_1, \ldots, y_m)^{\mathrm{T}}$ and A is the $m \times q$ matrix with (i, j)th element $N_{n,j}(\boldsymbol{\lambda}; x_i)$. This is a *linear* least-squares problem, and is well understood, with highly satisfactory algorithms [9] and software implementations [2, 19] available.

Free-knot approximation. Determination of $\mathbf{c}$ *and* $\boldsymbol{\lambda}$ for given data, spline order n and number N of knots.

The free-knot spline approximation problem is formulated in a similar way to that for fixed knots (above), except that the knots are not specified *a priori* but are regarded as parameters whose values are to be determined. Writing

$$\mathbf{e}(\boldsymbol{\lambda}) = \mathbf{y} - A(\boldsymbol{\lambda})\mathbf{c},$$

it is required to solve

$$\min_{\mathbf{c},\boldsymbol{\lambda}} \|\mathbf{e}(\boldsymbol{\lambda})\|^2, \tag{4}$$

which is a *non-linear* least-squares problem, and a more difficult problem to solve.

In the description of the free-knot spline approximation problem given above, the number N of knots is *not* regarded as a parameter. This is because, provided the abscissa values $\{x_i\}_1^m$ are distinct, it is possible to choose N sufficiently large, and an appropriate knot set $\boldsymbol{\lambda}$, such that s *interpolates* the data, and the least-squares objective function takes its minimum possible value (of zero). In the context of the approximation (modelling) of data that arises from measurement (and is inexact), an interpolant is generally not a useful solution. Instead, *model validation* [4] can be used to decide a suitable value for N by, e.g., obtaining spline approximations for $N = 1, 2, \ldots$ knots, and studying the effect of N on the quality of the approximant [13, 14].

It has been noted (section 1) that the knots generally have no physical meaning for the metrologist, and yet the effectiveness of a spline approximant can depend critically on their number and positions. Consequently, for the purposes of this study, the free-knot spline approximation problem is regarded as the problem we set out to solve, viz., the 'correct' problem. A statement of the *computational aim* C_0 [5] for this problem is given in section 4.1. A software implementation of an algorithm for solving the free-knot spline approximation problem specified by C_0, and a data generator for C_0, are described in sections 5.1 and 6.1, respectively.

There are theoretical difficulties associated with existence, uniqueness and characterisation of a solution to the free-knot spline approximation problem [13, 14],

which makes solving (4) a much more difficult problem than the fixed-knot spline approximation problem (3), and therefore one that we often seek to avoid. In order to reflect the fact that for any given knot set the B-spline coefficients are given by solving a relatively simple, linear problem, formulation (4) can be expressed as

$$\min_{\boldsymbol{\lambda}} \left(\min_{\mathbf{c}} \|\mathbf{e}(\boldsymbol{\lambda})\|^2 \right).$$

This result is used as the basis of 'approximate' algorithms for solving the free-knot spline approximation problem. These algorithms apply a *knot placement strategy* to decide the positions of the knots, and then solve the fixed-knot spline approximation problem with these knots. Some knot placement strategies involve positioning the knots according to the distribution of the data abscissae (and possibly the data ordinates). Others 'iterate' with respect to $\boldsymbol{\lambda}$, solving a fixed-knot spline approximation problem at each iteration, and using the solutions so obtained to 'improve' the choice of $\boldsymbol{\lambda}$. Useful knot placement strategies have been proposed, implemented and used [13, 14], including manual methods, sequential knot-insertion strategies, sequential knot-deletion strategies, approaches based on removing trends, and 'theory-based' approaches. Those strategies that are the subject of this study are described in the following section. As for the free-knot spline approximation problem, statements of the computational aims corresponding to the approximate algorithms are given in section 4, software implementations of algorithms for solving the problems specified by the computational aims are given in section 5, and data generators are described in section 6.

2.3 Knot placement strategies

Three knot placement strategies are used as the basis of approximate algorithms for solving the free-knot spline approximation problem. These involve placement of the knots uniformly within the interval I, knot placement based on the distribution of the data abscissae, and a sequential knot-insertion strategy.

2.3.1 Uniformly-spaced knots

One of the simplest strategies for deciding the positions of a fixed number of knots is to select the knots to be uniformly-spaced between $x_{\min}$ and $x_{\max}$, the endpoints of the interval I. This corresponds to setting

$$\lambda_j = x_{\min} + j \left\{ \frac{x_{\max} - x_{\min}}{N+1} \right\}, \qquad j = 1, \ldots, N.$$

Such a choice makes minimal use of any available knowledge about the data, only that the data abscissa values are contained within the interval I.

2.3.2 Knot placement based on the distribution of the data abscissae

In the strategy described here, account is taken of the *distribution* of the data abscissa values $\{x_i\}_1^m$. The approach tends to give better knot positions (compared with uniformly-spaced knots) if the data is gathered in a manner which ensures that the local density of the data is greater in regions where the behaviour of function underlying the data is more marked, e.g., in the neighbourhood of peaks and valleys.

Consider the graph $x = F(\ell)$ given by the piecewise-linear function joining the points $\{(i, x_i)\}_1^m$. Then, the jth knot is set according to

$$\lambda_j = F(1 + (m-1)(j + n/2 - 1)/(q-1)), \qquad j = 1, \ldots, N.$$

The choice can be interpreted as placing the knots such that there is an approximately equal number of data points in each knot interval I_j, except that in the first and the last interval there are approximately $n/2$ times as many points. The strategy [2] has the property that when N is such that the data is interpolated, i.e., $N = m - n$, the choice of knots agrees with one of the recommended choices for spline interpolation [17]. It also has the property that the Schoenberg-Whitney conditions [23] are satisfied 'as well as possible' for a subset of the data: these are conditions on the data and a given knot set that guarantee (at least mathematically) that the solution to the fixed-knot spline approximation problem with the given knots is unique.[5]

2.3.3 Sequential knot-insertion strategies

In a sequential knot-insertion strategy, a succession of (fixed-knot) approximants is obtained, in which for each approximant a knot is inserted in the knot interval that gives rise to the greatest contribution to the least-squares measure. Previously inserted knots are retained undisturbed. Several variants are possible, e.g.:

- Start the process with a number of knots already in place, perhaps obtained from information specific to the application.
- Candidate positions for a new knot are
  - The continuum of points within the interval. The approach gives rise to the minimisation of a univariate function that may possess local minima.

[5] Numerically, however, the satisfaction of these conditions does not ensure that the solution is well-defined.

  - The subset within the interval of a discrete set of points chosen *a priori*, e.g., the data abscissae themselves or a uniformly-spaced set of x-values. The approach gives rise to a finite computation for the globally-best choice of knot, relative to the discretisation, with respect to previous knots.

- More than one knot can be inserted at a time. If candidate positions for the new knots are a discrete set of points (as in the bullet above), computation times rise rapidly with the number of 'simultaneous' knots so inserted, so in practice inserting only a small number, say two or three, might be feasible.

- Use a merit function other than the least-squares measure.

For the sequential knot-insertion strategy considered in this work:

1. The procedure starts with $N = 0$ knots.
2. Candidate positions for a new knot are the data abscissa values $\{x_i\}_2^{m-1}$,[6] excepting the positions of previously inserted knots.
3. A single knot is inserted at each iteration.
4. The merit function for knot-insertion is the least-squares measure.

3 Methods for algorithm testing

A framework has been described [5] for describing the activities of algorithm testing and numerical software testing, and how these activities contribute to the (overall) validation of the software implementation of an algorithm for solving the problem specified by a given computational aim. A number of approaches to algorithm testing, based on the methodologies of numerical software testing [6], are discussed. These include methods based on:

- an analysis of the required ('correct') computational aim C_0 and an approximate computational aim C_a,
- using software implementations of the computational aims,
- data generation software for the computational aims, and
- simulation.

[6] The abscissa values x_1 and x_m are not included since $x_{\min} = x_1,\ x_{\max} = x_m$ and the inserted knots are required to lie in the *open* interval $(x_{\min}, x_{\max})$ (section 2.1).

In this section a summary of these generic methods is provided.[7] The application of the methods to the validation of the software implementations of various algorithms for the free-knot spline approximation problem is covered in sections 4–8.

3.1 Methods based on the analysis of computational aims

Analysis of the required and approximate computational aims to understand the circumstances under which they are mathematically equivalent, and to derive expressions for the distance (in an appropriate norm) between the solutions to the problems specified by them, is an important approach to algorithm testing. The approach involves working with the computational aims, and the algorithms for solving the problems specified by them, as *mathematical objects* independent of any software implementation. The approach has the disadvantages of generally requiring a high degree of technical skill (and therefore relying on experts), being highly problem-specific, and consequently being difficult (if not impossible) to automate.

3.2 Methods based on using software implementations of the computational aims

In some circumstances *software* implementing the required and approximate computational aims will be available. A direct approach to comparing the computational aims using this software is as follows:

1. Start with a reference data set $\mathbf{x}$ (perhaps typical of one that is likely to be encountered in practice).[8]
2. Apply software for the required computational aim C_0 to the reference data set $\mathbf{x}$ to obtain reference results $\mathbf{y}_0$.
3. Apply software for the approximate computational aim C_a to the reference data set $\mathbf{x}$ to obtain test results $\mathbf{y}_a$.
4. Measure the difference between the computational aims C_0 and C_a by the distance (in an appropriate norm) between the reference results $\mathbf{y}_0$ and the test results $\mathbf{y}_a$.

The distance evaluated at step 4 will include contributions relating to the accuracy of the calculations of the reference and test results, and so may not immediately

[7] The notation 'C_a' for approximate computational aim is used for the discussion of the generic methods. In the application of the methods to the testing of knot placement strategies, the approximate computational aims will be denoted by 'C_1', 'C_2', etc.

[8] $\mathbf{x}$ is used to denote particular values for the input data for the problems specified by C_0 and C_a.

provide an accurate measurement of the departure of C_a from C_0. However, numerical software testing may be used to investigate whether the software used to calculate $\mathbf{y}_a$ and $\mathbf{y}_0$ is performing like reference software by quantifying the magnitude of the errors in the calculated results. In particular, in the case that the magnitudes of the errors are small compared with $\|\mathbf{y}_a - \mathbf{y}_0\|$, the latter can be used to measure the difference between the computational aims as required.

3.3 Methods based on data generation software for the computational aims

The generation of *reference pairs*, comprising reference data sets and corresponding reference results, appropriate to a specified computational aim is the basis of "black-box" testing of numerical software [6, 11, 12]. One approach to generating reference pairs is to use reference software (Section 3.2). Another approach is to use a *data generator*. This is software for constructing reference data sets with known solutions, i.e., solutions specified *a priori*, by solving the inverse problem to that defined by the computational aim. For a problem whose solution can be *characterised* mathematically, the implementation of a data generator is generally much simpler than the implementation of software for solving the forward problem defined by the computational aim. Data generators have been used in the numerical testing of software for a number of calculations [12].

In the context of algorithm testing, data generation software for the required and approximate computational aims may be used to compare the computational aims in the following manner:

1. Start with a reference solution denoted by $\mathbf{y}$.
2. Use data generation software appropriate to the required computational aim C_0 to generate reference data $\mathbf{x}_0$ corresponding to the reference solution $\mathbf{y}$.
3. Use data generation software appropriate to the approximate computational aim C_a to generate reference data $\mathbf{x}_a$ corresponding to $\mathbf{y}$.
4. Measure the difference between the computational aims C_0 and C_a by the distance (in an appropriate norm) between the data sets $\mathbf{x}_0$ and $\mathbf{x}_a$.

The process of data generation in steps 2 and 3 above is a one-to-many mapping, i.e., in each case there are *many* reference data sets $\mathbf{x}_0$ and $\mathbf{x}_a$ for the two computational aims that have the same reference solution $\mathbf{y}$. Consequently, at step 2 we choose a particular reference data set $\mathbf{x}_0$ that is 'close' (in an appropriate norm) to a data set $\mathbf{x}$ that is typical of one likely to be encountered in practice. Then, at step 3 we choose a reference data set $\mathbf{x}_a$ that is 'closest to' $\mathbf{x}_0$ (in the same norm).

If the distance between the reference data sets $\mathbf{x}_0$ and $\mathbf{x}_a$ constructed in this way is *zero*, the computational aims are equivalent for a problem specified by the reference pair $(\mathbf{x}, \mathbf{y})$. If the distance is *small* compared to the uncertainty associated with measurements of typical data $\mathbf{x}$, the computational aim C_a may be regarded as fit for purpose. Otherwise, we may question whether the approximate computational aim used in place of the required computational aim is adequate for its intended purpose.

The approach is illustrated in Figure 1. In this figure $D_0(\mathbf{y})$ represents the collection of data sets $\mathbf{x}_0$ for which the solution to the problem specified by the computational aim C_0 is $\mathbf{y}$. Similarly, $D_a(\mathbf{y})$ represents the collection of data sets $\mathbf{x}_a$ for which the solution to the problem specified by the approximate computational aim C_a is $\mathbf{y}$. $\mathbf{x}$ is a data set that is typical of the application: then, $\mathbf{x}_0$ is the data set in $D_0(\mathbf{y})$ closest to $\mathbf{x}$, and $\mathbf{x}_a$ the data set in $D_a(\mathbf{y})$ closest to $\mathbf{x}_0$. Finally, the distance d between $\mathbf{x}_0$ and $\mathbf{x}_a$ measures the difference between the computational aims for the data set $\mathbf{x}$.

The approach is similar to the technique of *backward error analysis* [24] used to investigate the numerical properties of algorithms implemented using finite precision arithmetic. An advantage of the approach is that software implementing algorithms to solve the problems specified by the computational aims is not required.

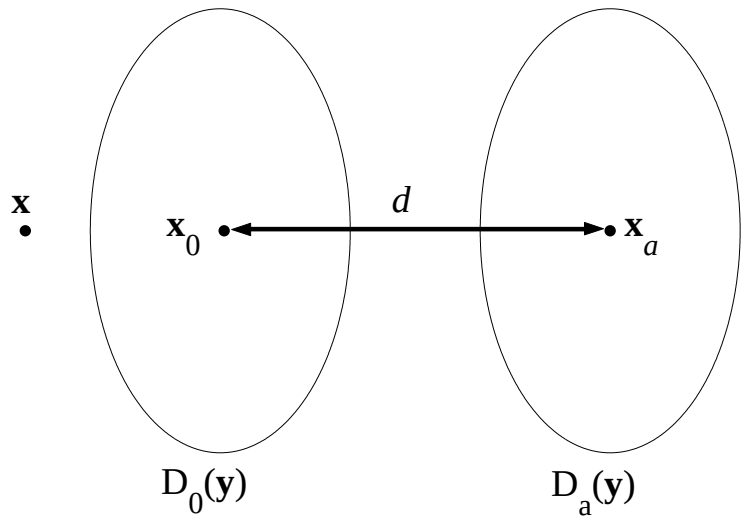

Figure 1: Comparing required and approximate computational using data generation software.

3.4 Methods based on simulation

The methods described in the previous sections provide point measures of the departure of C_a from C_0 for the single reference data set $\mathbf{x}$. It is recommended that the methods are applied for a range of reference data sets typical of those likely to be encountered in practice. This can mean sampling (perhaps uniformly) from the domain of all possible data sets that are likely to be encountered, including those close to the boundaries of this domain. In addition, and in order to obtain statistical measures of the departure and, in particular, to assess the fitness for purpose of C_a as a substitute for C_0, it is recommended that the methods are applied using reference data sets sampled in such a way as to reflect the available knowledge about the inexactness of the data.[9]

Let $\mathbf{X}$ and $\mathbf{Y}$ denote the input and output quantities for the problems specified by C_0 and C_a. Furthermore, let $g(\mathbf{X})$ denote the probability density function (PDF) for the value of $\mathbf{X}$, describing the available knowledge about the inexactness of $\mathbf{X}$, and $g_0(\mathbf{Y})$ and $g_a(\mathbf{Y})$ the PDFs for the value of $\mathbf{Y}$ returned as, respectively, the solutions to the problems specified by C_0 and C_a. Then, software for solving the problems specified by the two computational aims (Section 3.2) may be combined with simulation in the following manner:

1. Choose a number M of trials.
2. For $r = 1, \ldots, M$:
   (a) Construct the data set $\mathbf{x}^r$ as a random sample from the PDF $g(\mathbf{X})$.
   (b) Apply software for the required computational aim C_0 to the data set $\mathbf{x}^r$ to obtain reference results $\mathbf{y}_0^r$.
   (c) Apply software for the approximate computational aim C_a to the data set $\mathbf{x}^r$ to obtain test results $\mathbf{y}_a^r$.
3. Construct an approximation $\widehat{g_0}(\mathbf{Y})$ to the PDF $g_0(\mathbf{Y})$ for the value of $\mathbf{Y}$ from the set of reference results $\mathbf{y}_0^r,\ r = 1, \ldots, M$.
4. Construct an approximation $\widehat{g_a}(\mathbf{Y})$ to the pdf $g_a(\mathbf{Y})$ for the value of $\mathbf{Y}$ from the set of test results $\mathbf{y}_a^r,\ r = 1, \ldots, M$.
5. Compare $\widehat{g_0}(\mathbf{Y})$ and $\widehat{g_a}(\mathbf{Y})$.

The comparison at step 5 can involve comparing the expectations of the PDFs, or their standard deviations, or coverage intervals corresponding to a stipulated coverage probability, or (other) quantities derived from the PDFs.

[9] For example, for a regression problem, we might choose to simulate data by sampling the measurement deviations for the data from a normal (Gaussian) probability distribution with expectation zero and standard deviation σ, where σ describes the 'accuracy' of the data obtained from a typical instrument or measurement system.

4 Computational aims

In this section are specified:

- the computational aim C_0 of the free-knot spline approximation problem that it is required to solve, and
- the computational aims $C_1,\ C_2$ and C_3 of the *approximate* problems that are solved in place of the problem specified by C_0.

Each computational aim is presented as a *mathematical* relationship between input quantities (data, spline order n and number N of knots) and output quantities (spline coefficients and knot positions). In the case of $C_0,\ C_1$ and C_2, the relationship takes the form a spline approximation problem (with free or fixed knots). In the case of C_3, the relationship is in the form of an algorithm (a set of steps) involving the solution of a succession of fixed-knot spline approximation problems. In all cases, the relationship between input and output quantities is *implicit*. In particular, the relationships are not provided in the form of a set of formulae to be evaluated, or a set of equations to be solved, for the output quantities. It is not the purpose of this section to provide details of the algorithms and (software) implementations of the algorithms for solving the problems specified by the computational aims. Such details are provided in section 5. A discussion of the computational aims, including comments concerning those circumstances for which the computational aims are *mathematically* equivalent (section 3.1), is given in section 4.5.

4.1 Computational aim C_0

Given data $\{(x_i, y_i)\}_1^m$, spline order n and a number N of knots, find $\mathbf{c} = \mathbf{c}_0$ and $\boldsymbol{\lambda} = \boldsymbol{\lambda}_0$ that solve

$$\min_{\mathbf{c},\boldsymbol{\lambda}} \sum_{i=1}^{m} \{y_i - s(\mathbf{c}, \boldsymbol{\lambda}; x_i)\}^2 ,$$

where

$$s(\mathbf{c}, \boldsymbol{\lambda}; x) = \sum_{j=1}^{q} c_j N_{n,j}(\boldsymbol{\lambda}; x).$$

This is the free-knot spline approximation problem (section 2.2).

4.2 Computational aim C_1

Given data $\{(x_i, y_i)\}_1^m$, spline order n and a number N of knots, set $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1$ where

$$\lambda_{1,j} = x_{\min} + j \left\{ \frac{x_{\max} - x_{\min}}{N+1} \right\}, \qquad j = 1, \ldots, N, \tag{5}$$

and find $\mathbf{c} = \mathbf{c}_1$ that solves

$$\min_{\mathbf{c}} \sum_{i=1}^{m} \{y_i - s(\mathbf{c}, \boldsymbol{\lambda}_1; x_i)\}^2 .$$

This is the fixed-knot spline approximation problem with uniformly-spaced knots (sections 2.2 and 2.3.1).

4.3 Computational aim C_2

Given data $\{(x_i, y_i)\}_1^m$, spline order n and a number N of knots, set $\boldsymbol{\lambda} = \boldsymbol{\lambda}_2$ where

$$\lambda_{2,j} = F(1 + (m-1)(j + n/2 - 1)/(q-1)), \qquad j = 1, \ldots, N, \qquad (6)$$

and the graph $x = F(\ell)$ is given by the piecewise-linear function joining the points $\{(i, x_i)\}_1^m$, and find $\mathbf{c} = \mathbf{c}_2$ that solves

$$\min_{\mathbf{c}} \sum_{i=1}^{m} \{y_i - s(\mathbf{c}, \boldsymbol{\lambda}_2; x_i)\}^2 .$$

This is the fixed-knot spline approximation problem with knot placement based on the distribution of the data abscissae (sections 2.2 and 2.3.2).

4.4 Computational aim C_3

Given data $\{(x_i, y_i)\}_1^m$, spline order n and a number N of knots, perform the following steps:[10]

1 Set $X_1 = \{x_2, \ldots, x_{m-1}\}$.

2 Set $\boldsymbol{\mu}_0 = \{\}$.[11]

3 If $N = 0$, find $\mathbf{c} = \mathbf{c}_3$ that solves

$$\min_{\mathbf{c}} \sum_{i=1}^{m} \{y_i - s(\mathbf{c}, \boldsymbol{\mu}_0; x_i)\}^2 .$$

4 If $N > 0$, perform the following steps:

4.1 For $k = 1, \ldots, N$:

[10] For convenience $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are used to denote both *sets* of knot positions (which are subject to set operations, including 'union') and *vectors* of knot positions (which are used as parameters of the spline function s). The context will make clear which interpretation is to apply.

[11] This is the empty set used to denote the absence of (interior) knots.

4.2 Find $\mathbf{c} = \mathbf{c}_3$ and $\gamma = \gamma_k$ that solve

$$\min_{\mathbf{c},\gamma} \sum_{i=1}^{m} \left\{ y_i - s(\mathbf{c}, \boldsymbol{\mu}_{k-1} \cup \{\gamma\}; x_i) \right\}^2, \qquad \gamma \in X_{k-1}. \tag{7}$$

4.3 Set $X_k = X_{k-1} \backslash \gamma_k$.

4.4 Set $\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k-1} \cup \{\gamma_k\}$.

5 Set $\boldsymbol{\lambda}_3 = \boldsymbol{\mu}_N$.

4.5 Discussion of computational aims

The following observations apply to the computational aims specified above:

- For the case $N = 0$, the free-knot spline approximation problem with no (interior) knots, all four computational aims will possess the same solution, that obtained by solving the fixed-knot spline approximation problem with $\boldsymbol{\lambda}$ set equal to the empty vector.

- For the case $N = 1$, the problem specified by C_3 is a 'discrete' version of that specified by C_0. In this case, the problems specified by the two computational aims are to minimise the same (least-squares) objective function, but in the case of C_0 this is done with respect to a knot position that can take *any* value between $x_{\min}$ and $x_{\max}$, whereas for C_3 the knot position is restricted to take values within a discrete set (here, the set $\{x_i\}_2^{m-1}$). Consequently, the maximum (absolute) difference between the knot positions returned as solutions to the problems specified by C_0 and C_3 is bounded by the (maximum) spacing between consecutive x_i–values.

- The knot positions returned as solutions to the problems specified by C_0 and C_1 will only be the same if a particular condition applies, viz., that the solution to the problem specified by C_0 is such that the knots are uniformly spaced (and satisfy (5)). (Similarly, for C_0 and C_2 to be (mathematically) equivalent, the solution to the problem specified by C_0 must be such that the knots satisfy (6).) The amount by which the knots returned as a solution to C_0 fail to satisfy (5) (respectively, (6)) is a measure of the departure between C_0 and C_1 (respectively, C_2).

- In the case of uniformly-spaced x_i–values, the knot positions returned as solutions to the problems specified by C_1 and C_2 will *only* be the same for the cases $N = 0$ (above) and $N = 1$.

5 Software implementations of the computational aims

This section provides details of the algorithms and software implementations of those algorithms for solving the problems specified by the computational aims given in section 4. In section 5.5 an example is given to illustrate the results returned by the implemented software for a given data set. Section 7 is concerned with presenting the results of testing the numerical accuracy of the software implementations described here.

5.1 Software S_0

The free-knot spline approximation problem specified by computational aim C_0 in section 4.1 is a nonlinear least-squares problem. Software S_0 solves this problem using a Gauss-Newton algorithm [20].The Gauss-Newton algorithm is an iterative algorithm where at each iteration the linear least-squares problem

$$\min_{\delta\boldsymbol{\zeta}} \|\mathbf{r} + J\,\delta\boldsymbol{\zeta}\|^2$$

is solved, with $\mathbf{r}$ the vector of residual deviations

$$r_i = y_i - s(\mathbf{c}, \boldsymbol{\lambda}; x_i), \qquad i = 1, \ldots, m, \tag{8}$$

and J the $m \times (q + N)$ (Jacobian) matrix of partial derivatives, given by

$$J = \begin{bmatrix} \dfrac{\partial r_1}{\partial c_1} & \cdots & \dfrac{\partial r_1}{\partial c_q} & \dfrac{\partial r_1}{\partial \lambda_1} & \cdots & \dfrac{\partial r_1}{\partial \lambda_N} \\ \vdots & & \vdots & \vdots & & \vdots \\ \dfrac{\partial r_m}{\partial c_1} & \cdots & \dfrac{\partial r_m}{\partial c_q} & \dfrac{\partial r_m}{\partial \lambda_1} & \cdots & \dfrac{\partial r_m}{\partial \lambda_N} \end{bmatrix}, \tag{9}$$

both evaluated for the current estimates of the parameters $\boldsymbol{\zeta} = (\mathbf{c}^{\mathrm{T}}, \boldsymbol{\lambda}^{\mathrm{T}})^{\mathrm{T}}$. An orthogonal factorization method [21] is used to obtain the solution to the above linear least-squares problem [10]. The vector $\delta\boldsymbol{\zeta}$ contains the Gauss-Newton step which is used to update the current estimates of the parameters $\boldsymbol{\zeta}$ to provide new estimates. Initial estimates of the parameters are provided by the software S_1 (section 5.2) for solving the problem specified by the computational aim C_1, and the iteration continues until either the decrease in the residual sum of squares is less than 10^{-15} or $\max_k |\delta\zeta_k|$, (the maximum absolute change in the values of the parameters) is less than 10^{-15}.[12]

In an implementation of S_0 it is necessary to evaluate the Jacobian matrix J. Using (2), the derivative of the residual deviation r_i with respect to the spline coefficient

[12] The use of these fixed tolerances is appropriate for *scaled* problems.

c_k is

$$\frac{\partial r_i}{\partial c_k} = -N_{n,k}(\boldsymbol{\lambda}; x_i),$$

the value of the kth B-spline basis function with knots $\boldsymbol{\lambda}$ at $x = x_i$. Furthermore, the derivative with respect to λ_k is

$$\frac{\partial r_i}{\partial \lambda_k} = -\sum_{j=1}^{q} c_j \frac{\partial N_{n,j}(\boldsymbol{\lambda}; x_i)}{\partial \lambda_k}.$$

The evaluation of the derivatives of the B-spline basis functions with respect to the knots $\boldsymbol{\lambda}$ is considered in Appendix A.

5.2 Software S_1

The key computation undertaken by software S_1 to solve the problem specified by computational aim C_1 is to solve the fixed-knot spline approximation problem with uniformly-spaced knots $\boldsymbol{\lambda}_1$ defined by (5). This is equivalent to finding $\mathbf{c} = \mathbf{c}_1$ that solves the linear least-squares problem

$$\min_{\mathbf{c}} \|\mathbf{y} - K_1\mathbf{c}\|^2,$$

where K_1 is the $m \times q$ matrix K given by

$$K = \begin{bmatrix} N_{n,1}(\boldsymbol{\lambda}; x_1) & \cdots & N_{n,q}(\boldsymbol{\lambda}; x_1) \\ \vdots & & \vdots \\ N_{n,1}(\boldsymbol{\lambda}; x_m) & \cdots & N_{n,q}(\boldsymbol{\lambda}; x_m) \end{bmatrix} \tag{10}$$

evaluated at the knots $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1$. An orthogonal factorization method [21] is used to obtain the solution to the above linear least-squares problem [10].

5.3 Software S_2

As in section 5.2, the key computation undertaken by software S_2 to solve the problem specified by computational aim C_2 is to solve the fixed-knot spline approximation problem with knots $\boldsymbol{\lambda}_2$ defined by (6). This is equivalent to finding $\mathbf{c} = \mathbf{c}_2$ that solves the linear least-squares problem

$$\min_{\mathbf{c}} \|\mathbf{y} - K_2\mathbf{c}\|^2,$$

where K_2 is the matrix K given by (10) evaluated at $\boldsymbol{\lambda} = \boldsymbol{\lambda}_2$. An orthogonal factorization method [21] is again used to obtain the solution to the above linear least-squares problem [10].

5.4 Software S_3

The key computation undertaken by software S_3 to solve the problem specified by computational aim C_3 is to solve (7) at step 4.2. This can be done by solving a sequence of fixed-knot spline approximation problems of the form described in sections 5.2 and 5.3. Specifically, for each $\gamma \in X_{k-1}$, find $\mathbf{c} = \mathbf{c}_\gamma$ that solves the linear least-squares problem

$$\min_{\mathbf{c}} \|\mathbf{y} - K_\gamma \mathbf{c}\|^2,$$

where K_γ is the matrix K given by (10) evaluated at $\boldsymbol{\lambda} = \boldsymbol{\mu}_{k-1} \cup \{\gamma\}$, and evaluate the least-squares objective function

$$S_\gamma^2 = \sum_{i=1}^{m} \left\{ y_i - s(\mathbf{c}_\gamma, \boldsymbol{\mu}_{k-1} \cup \{\gamma\}; x_i) \right\}^2 .$$

Then, the minimiser $\mathbf{c} = \mathbf{c}_3$ and $\gamma = \gamma_k$ at step 4.2 of section 4.4 are those values $\mathbf{c}_\gamma$ and γ that correspond to the minimum value of S_γ^2.

5.5 Example to show application of software implementations

Section 6.1 describes data generation software G_0 for generating data for which the solution to the problem specified by computational aim C_0 is specified *a priori*. Figure 2 shows the results of applying each of the software implementations $S_0,\ S_1,\ S_2$ and S_3 to data generated by G_0 for a cubic (order 4) spline with specified spline coefficients and three knots $\boldsymbol{\lambda}^{(3)}$ given by

$$\boldsymbol{\lambda}^{(3)} = (-0.380, -0.004, 0.530)^{\mathrm{T}}.$$

Figure 3 shows results produced similarly for data generated by G_0 for a cubic spline with nine knots $\boldsymbol{\lambda}^{(9)}$ given by

$$\boldsymbol{\lambda}^{(9)} = (-0.810, -0.610, -0.390, -0.210, -0.003, 0.200, 0.380, 0.580, 0.810)^{\mathrm{T}}.$$

In each figure the top graph displays the solutions returned by the four software implementations. The spline curves are shown (as continuous curves) together with the positions of the knots associated with the curves (as crosses). (Note that by virtue of using data generation software for C_0, software S_0 will return the 'correct' spline and knot positions, i.e., the spline and knots specified *a priori*.) The lower graph shows the differences between the splines curves returned by the four software implementations and the 'correct' spline specified *a priori*.[13] (Again,

[13]The 'kinks' that are apparent in the curves representing the differences between the spline curves returned by the four software implementations and the 'correct' spline arise in regions where differences are taken of spline curves with knots that are close (but not identical). The curves representing the differences are 'smooth' in a mathematical sense.

by virtue of using G_0, the difference for the solution returned by S_0 is identically zero.)

Because the 'correct' knots are almost uniformly-spaced, the solution returned by S_1 is the most likely to be close to the solution to the free-knot spline approximation problem. We note that for both sets of data, software S_3, implementing the sequential knot-insertion strategy, returns knots that are far from being uniformly-spaced, and yet the spline approximant based on these knots (in black) does generally reproduce the 'shape' of the free-knot spline approximant (in blue). The solutions returned by S_1 and S_2 also generally reproduce this shape. For some applications the solutions returned by the approximate algorithms (based on the knot placement strategies considered) may well be acceptable; for others they will not. Algorithm testing is used to help users of such algorithms to make an objective decision about the acceptability of a particular algorithm for a particular application.

6 Data generators for the computational aims

This section provides details of data generation software for the problems specified by the computational aims given in section 4. Data generation software is software used to generate data for which the solution to a problem specified by a given computational aim is known *a priori* (section 3.3). Data generation software is used:

- for testing the numerical accuracy of the software implementations described in section 5 against their respective computational aims, and
- in algorithm testing, to compare those computational aims.

Some remarks about the data generation software G_3 for computational aim C_3 are made in section 6.5.

6.1 Data generator G_0

A solution to the problem specified by the computational aim C_0 is characterised by the condition

$$J_0^{\mathrm{T}}\mathbf{r}_0 = \mathbf{0},$$

where J_0 and $\mathbf{r}_0$ are, respectively, the Jacobian matrix J given by (9) and vector of residual deviations $\mathbf{r}$ given by (8), both evaluated at the solution $(\mathbf{c}_0^{\mathrm{T}}, \boldsymbol{\lambda}_0^{\mathrm{T}})^{\mathrm{T}}$.

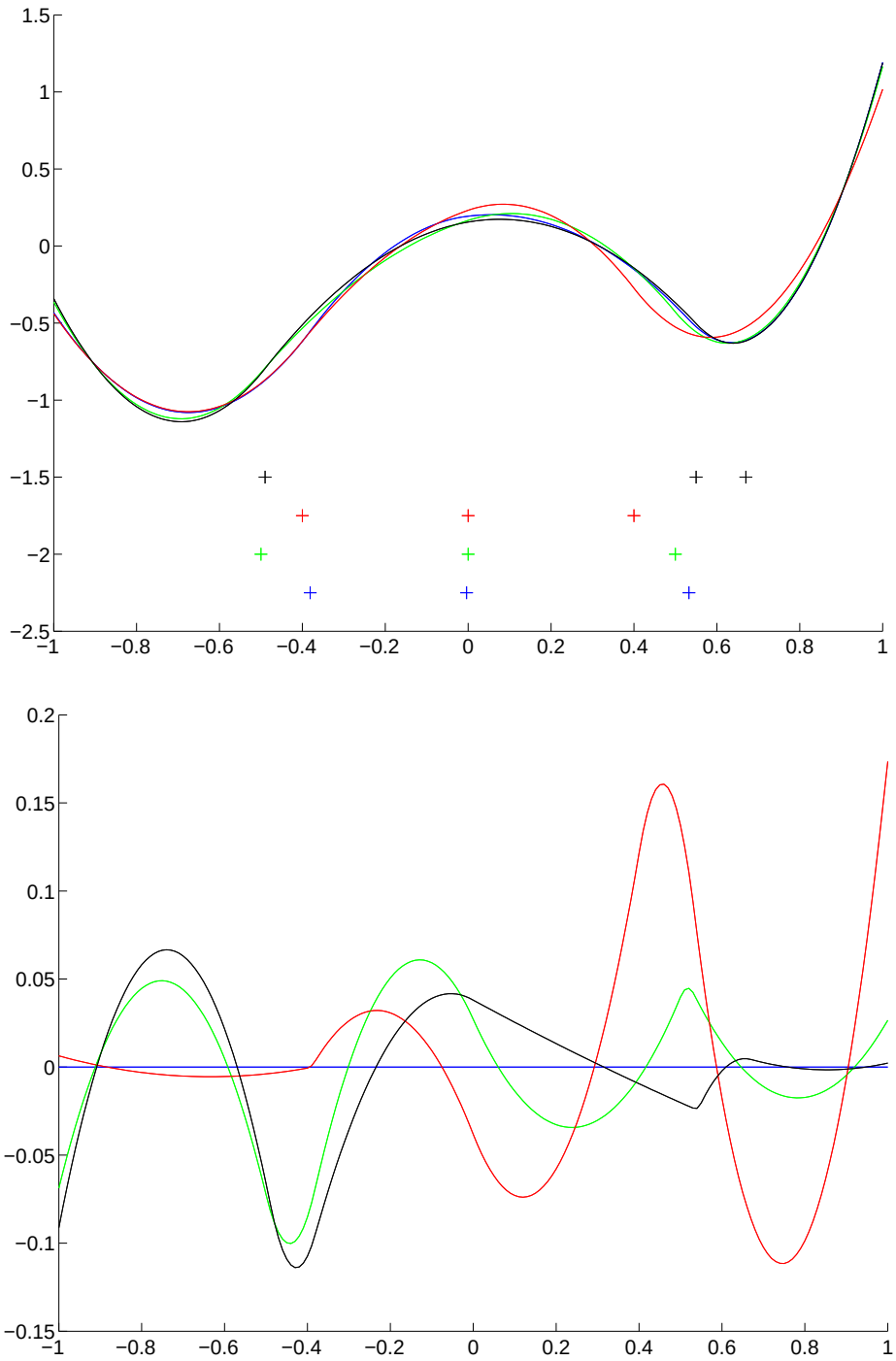

Figure 2: Results of applying S_0 (blue), S_1 (green), S_2 (red) and S_3 (black) to data generated by G_0 for a spline curve specified *a priori* with three knots. The top figure displays the solutions returned by the four software implementations, and the bottom figure shows the differences between those solutions and the 'correct' spline curve specified *a priori*.

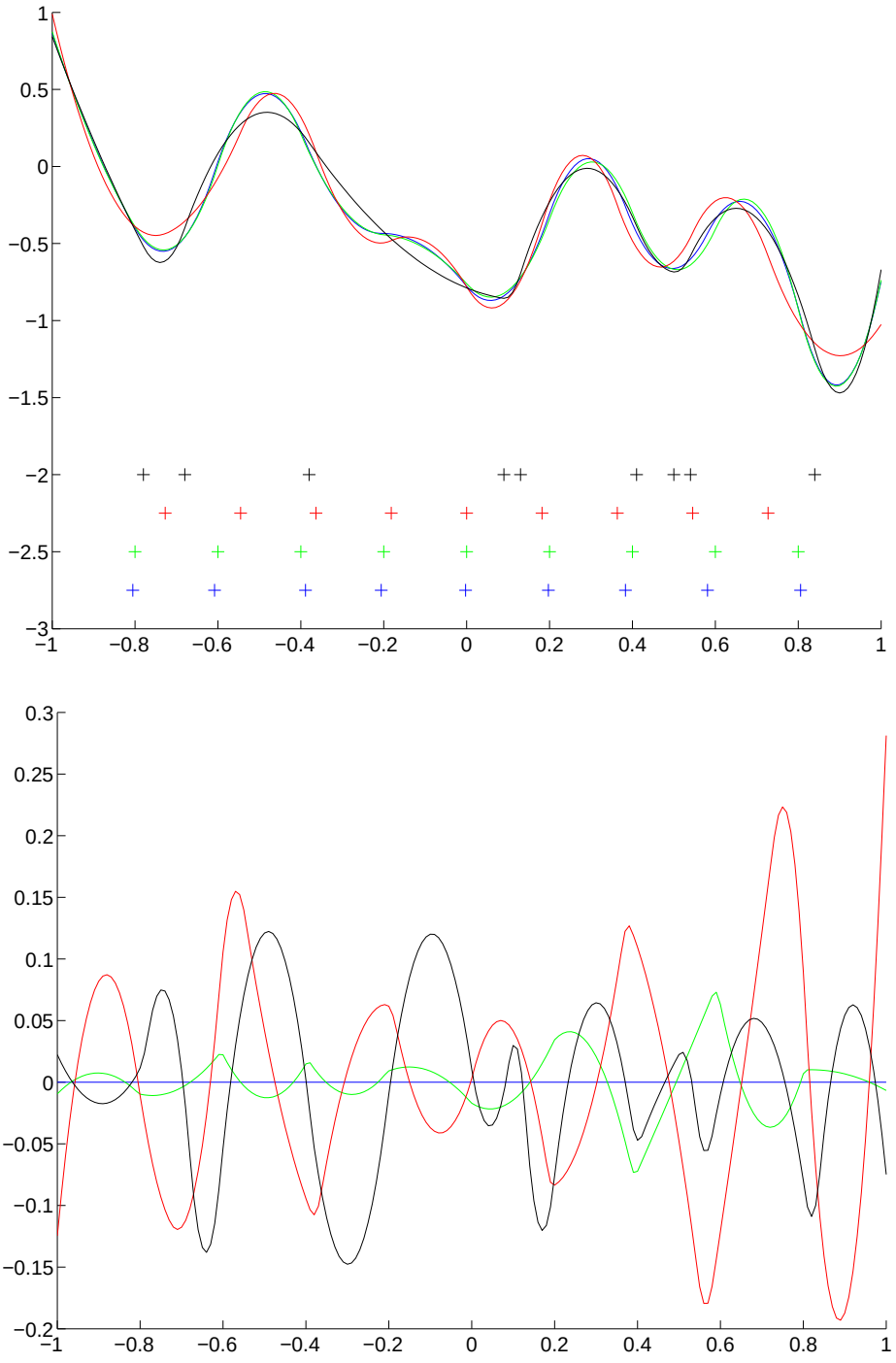

Figure 3: As figure 2, but for a spline curve specified *a priori* with nine knots.

Based on this characterisation, a procedure for generating data $\{y_{0,i}\}_1^m$ for which the solution to the problem specified by the computational aim C_0 is given *a priori* takes the following form.

Given values for $n,\ N,\ \{x_i\}_1^m,\ \mathbf{c}_0$ and $\boldsymbol{\lambda}_0$:

$G_0\mathbf{a}$ Form the spline values $y(x_i) = s(\mathbf{c}_0, \boldsymbol{\lambda}_0; x_i),\ i = 1, \ldots, m$.

$G_0\mathbf{b}$ Form the $m \times (q + N)$ Jacobian matrix J_0.

$G_0\mathbf{c}$ Form a set of target residuals $\mathbf{e}$.

$G_0\mathbf{d}$ Form the null space N_J for J_0^{T}.[14]

$G_0\mathbf{e}$ Form the residual vector $\mathbf{r}_0 = N_J\mathbf{u}$, where $\mathbf{u} = N_J^{\mathrm{T}}\mathbf{e}$.

$G_0\mathbf{f}$ Form the data values $y_{0,i} = y(x_i) + r_{0,i},\ i = 1, \ldots, m$.

The resulting data set will have a known solution (defined by $\mathbf{c}_0$ and $\boldsymbol{\lambda}_0$) to the problem specified by computational aim C_0 and be characterised by a known vector $\mathbf{r}_0$ of residual deviations that is 'close' (in a least-squares sense) to the vector $\mathbf{e}$ of target residual deviations. For example, the target residuals may be chosen to be random samples from a normal (Gaussian) probability distribution with expectation zero and standard deviation σ, and thus mimic measurement deviations that are likely to be encountered in practice when those deviations are dominated by random effects.

6.2 Data generator G_1

A solution to the problem specified by the computational aim C_1 is characterised by the condition

$$K_1^{\mathrm{T}}\mathbf{r}_1 = \mathbf{0},$$

where K_1 and $\mathbf{r}_1$ are, respectively, the matrix K given by (10) and the vector of residual deviations $\mathbf{r}$ given by (8), both evaluated at the solution $(\mathbf{c}_1^{\mathrm{T}}, \boldsymbol{\lambda}_1^{\mathrm{T}})^{\mathrm{T}}$.

A procedure for generating data $\{y_{1,i}\}_1^m$ for the problem specified by the computational aim C_1 is analogous to that described in section 6.1, with K_1 replacing J_0, $\mathbf{c}_1$ replacing $\mathbf{c}_0$, $\boldsymbol{\lambda}_1$ replacing $\boldsymbol{\lambda}_0$, etc.

[14] The null space N_J for J_0^{T} satisfies $J_0^{\mathrm{T}}N_J = 0$, and may be computed using matrix factorization techniques such as the singular value decomposition (SVD) [21].

6.3 Data generator G_2

A solution to the problem specified by the computational aim C_2 is characterised by the condition

$$K_2^{\mathrm{T}}\mathbf{r}_2 = \mathbf{0},$$

where K_2 and $\mathbf{r}_2$ are, respectively, the matrix K given by (10) and the vector of residual deviations $\mathbf{r}$ given by (8), both evaluated at the solution $(\mathbf{c}_2^{\mathrm{T}}, \boldsymbol{\lambda}_2^{\mathrm{T}})^{\mathrm{T}}$.

A procedure for generating data $\{y_{2,i}\}_1^m$ for the problem specified by the computational aim C_2 is analogous to that described in section 6.1, with K_2 replacing J_0, $\mathbf{c}_2$ replacing $\mathbf{c}_0$, $\boldsymbol{\lambda}_2$ replacing $\boldsymbol{\lambda}_0$, etc.

6.4 Data generator G_3

For $N = 1$ the data generation procedure is identical to that for computational aim C_0; see section 6.1.

For $N > 1$ the sequential nature of the knot insertion procedure means that a solution to the problem specified by the computational aim C_3 is characterised by N conditions as follows.

Given are values[15] for $n,\ N\ (>1),\ \{x_i\}_1^m,\ \mathbf{c}_N = (c_{N,1}, \ldots, c_{N,N+n})^{\mathrm{T}}$ and a set of knots $\tilde{\boldsymbol{\lambda}}_N = (\tilde{\lambda}_1, \ldots, \tilde{\lambda}_N)^{\mathrm{T}}$ arranged *in order of insertion*, i.e., $\tilde{\lambda}_1$ is the first knot to be inserted, $\tilde{\lambda}_2$ the second, etc.[16] Then, data $\{y_i\}_1^m$ for which the solution to the problem specified by C_3 is $\mathbf{c}_N$ and $\tilde{\boldsymbol{\lambda}}_N$ satisfies

$$y_i = y_{k,i} + r_{k,i}, \quad i = 1, \ldots, m, \quad k = 1, \ldots, N, \tag{11}$$

where

- $y_{k,i} = s(\mathbf{c}_k, \boldsymbol{\lambda}_k; x_i)$ in which $\mathbf{c}_k = (c_{k,1}, \ldots, c_{k,n+k})^{\mathrm{T}}$ are coefficients that, together with $\tilde{\boldsymbol{\lambda}}_k = (\tilde{\lambda}_1, \ldots, \tilde{\lambda}_k)^{\mathrm{T}}$, define a spline curve 'close to' the curve specified by $\mathbf{c}_N$ and $\boldsymbol{\lambda}_N$ (see $G_3\mathbf{b}$ below), and

- $\mathbf{r}_k$ is a vector of residual deviations satisfying

$$J_k^{\mathrm{T}}\mathbf{r}_k = \mathbf{0},$$

[15] To avoid excessive use of subscripts in this section, a notation different from that used for data generators G_0, G_1 and G_2 is adopted.

[16] The corresponding knot vector is $\boldsymbol{\lambda}_N$ obtained by arranging the elements $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_N$ into non-decreasing order.

where J_k is the $m \times (n+k+1)$ matrix of partial derivatives

$$J_k = \begin{bmatrix} \dfrac{\partial r_{k,1}}{\partial c_{k,1}} & \cdots & \dfrac{\partial r_{k,1}}{\partial c_{k,n+k}} & \dfrac{\partial r_{k,1}}{\partial \tilde{\lambda}_k} \\ \vdots & & \vdots & \vdots \\ \dfrac{\partial r_{k,m}}{\partial c_{k,1}} & \cdots & \dfrac{\partial r_{k,m}}{\partial c_{k,n+k}} & \dfrac{\partial r_{k,m}}{\partial \tilde{\lambda}_k} \end{bmatrix}, \tag{12}$$

evaluated at $(\mathbf{c}_k^{\mathrm{T}}, \boldsymbol{\lambda}_k^{\mathrm{T}})^{\mathrm{T}}$.

In terms of N_k, the null space of J_k^{T}, $\mathbf{r}_k$ may be written as

$$\mathbf{r}_k = N_k \mathbf{v}_k,$$

where $\mathbf{v}_k$ is an arbitrary (non-zero) vector of dimension $m - (n+k+1) \times 1$, and (11) becomes

$$\mathbf{y}_1 + N_1 \mathbf{v}_1 = \cdots = \mathbf{y}_N + N_N \mathbf{v}_N.$$

Defining

$$M = \begin{bmatrix} N_1 & -N_2 & & & \\ N_1 & & -N_3 & & \\ \vdots & & & \ddots & \\ N_1 & & & & -N_N \end{bmatrix} \text{ and } \mathbf{q} = \begin{bmatrix} \mathbf{y}_2 - \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N - \mathbf{y}_1 \end{bmatrix}, \tag{13}$$

this gives

$$M \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \mathbf{q}. \tag{14}$$

Letting N_M, the null space of M, be partitioned as

$$N_M = \begin{bmatrix} N_{M,1} \\ \vdots \\ N_{M,N} \end{bmatrix},$$

it follows that the solution to (14) can be written as

$$\begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{0,1} \\ \vdots \\ \mathbf{v}_{0,N} \end{bmatrix} + \begin{bmatrix} N_{M,1} \\ \vdots \\ N_{M,N} \end{bmatrix} \mathbf{p},$$

where $(\mathbf{v}_{0,1}^{\mathrm{T}}, \ldots, \mathbf{v}_{0,N}^{\mathrm{T}})^{\mathrm{T}}$ is a *particular* solution to (14), and $\mathbf{p}$ is an arbitrary vector.

Finally, finding the vector $\mathbf{r}_N$ of residual deviations closest (in a least-squares sense) to a given vector $\mathbf{e}$ reduces to solving the linear least-squares problem

$$\min_{\mathbf{p}} ||\mathbf{r}_N - \mathbf{e}||^2 \equiv ||N_N\mathbf{v}_N - \mathbf{e}||^2 \equiv ||N_N(\mathbf{v}_{0,N} + N_{M,N}\mathbf{p}) - \mathbf{e}||^2.$$

Based on the above characterisation, a procedure for generating data $\{y_i\}_1^m$ for which the solution to the problem specified by the computational aim C_3 is given *a priori* takes the following form.

G_3**a** Form the spline values $y(x_i) = s(\mathbf{c}_N, \boldsymbol{\lambda}_N; x_i),\ i = 1, \ldots, m$.

G_3**b** For $k = 1, \ldots, N-1$:

G_3**b**.1 Evaluate $\mathbf{c} = \mathbf{c}_k$ that minimises

$$\sum_{i=1}^{m} \{y(x_i) - s(\mathbf{c}, \boldsymbol{\lambda}_k; x_i)\}^2 .$$

G_3**b**.2 Form the spline values $y_{k,i}(x_i) = s(\mathbf{c}_k, \boldsymbol{\lambda}_k; x_i)$.

G_3**c** For $k = 1, \ldots, N$:

G_3**c**.1 Form the matrix J_k, as defined in (12), at $(\mathbf{c}_k^{\mathrm{T}}, \boldsymbol{\lambda}_k^{\mathrm{T}})^{\mathrm{T}}$.

G_3**c**.2 Form the null space N_k of J_k^{T}.

G_3**d** Form M and $\mathbf{q}$ as defined in (13) and find

$$\mathbf{v}_0 = \begin{bmatrix} \mathbf{v}_{0,1} \\ \vdots \\ \mathbf{v}_{0,N} \end{bmatrix}$$

such that

$$M\mathbf{v}_0 = \mathbf{q}.$$

G_3**e** Form the null space

$$N_M = \begin{bmatrix} N_{M,1} \\ \vdots \\ N_{M,N} \end{bmatrix}$$

of M.

G_3**f** Form a set of target residuals $\mathbf{e}$.

G_3**g** Find $\mathbf{p}$ that minimises $||N_N(\mathbf{v}_{0,N} + N_{M,N}\mathbf{p}) - \mathbf{e})||^2$.

G_3**h** Form the residual vector $\mathbf{r}_N = N_N(\mathbf{v}_{0,N} + N_{M,N}\mathbf{p})$.

G_3**i** Form the data values $y_{3,i} = y(x_i) + r_{N,i},\ i = 1, \ldots, m.$

The resulting data set will have a known solution (defined by $n,\ N,\ \mathbf{c}_N$ and $\boldsymbol{\lambda}_N$) to the problem specified by computational aim C_3 and be characterised by a known vector $\mathbf{r}_N$ of residual deviations that is 'close' (in a least-squares sense) to the vector $\mathbf{e}$ of target residual deviations.

6.5 Discussion of data generators

An issue for the problem specified by the computational aim C_0 is that the algorithm implemented by S_0 to solve the problem, and that implemented by G_0 to generate data for the problem, are both based on a characterisation of a *local* solution to the problem. A consequence of this is that (strictly) S_0 solves the problem specified by C_0 *approximately*. A further consequence is that S_0 may return a solution for data generated by G_0 that is differrent that the solution specified *a priori* for that data. Indeed, S_0 may return a 'better' solution, i.e., one with a lower value for the least-squares objective function, than that specified *a priori*.

A similar observation applies to the problem specified by the computational aim C_3. However, here, the algorithm implemented by S_3 returns the *global* solution to the problem[17] whereas the algorithm implemented by G_3 is based on a characterisation of a *local* solution to the problem.

To illustrate this (for the problem specified by C_3), consider using G_3 to generate data for which the solution to C_3 is a cubic spline with knots $\tilde{\boldsymbol{\lambda}} = (0, -0.6, 0.6)^{\mathrm{T}}$ specified *a priori*.[18] Figures 4 and 5 show the results of applying software S_3 to two sets of data generated using G_3. In each figure, the graph in the top-left shows the generated data set. The graph in the top-right shows how the least-squares objective function

$$S^2 = \sum_{i=1}^{m} \{y_i - s(\mathbf{c}_\gamma, \{\gamma\}; x_i)\}^2$$

varies with the position of the (single) knot γ, where $\mathbf{c}_\gamma$ are the spline coefficients that solve the fixed-knot spline approximation problem with knot γ. The position γ_1 of the first inserted knot is chosen to coincide with the minimum of this function. For both data sets, the minimum is at $\gamma = 0$ as required.

The graph in the bottom-left and bottom right show how the least-squares objective function varies with the position of the second and third inserted knots, respec-

[17] This is because the minimisation with respect to the spline coefficients for fixed knots possesses a single (global) solution, and that with respect to the (single) inserted knot is undertaken by examining all possible values for the position of the knot.

[18] The knot vector is $\boldsymbol{\lambda} = (-0.6, 0, 0.6)^{\mathrm{T}}$. The ordering of the knots in $\tilde{\boldsymbol{\lambda}}$ is used to indicate the *order* in which the knots are inserted, i.e., a knot at zero first, followed by a knot at -0.6, and finally a knot at 0.6. See section 6.4.

tively. For the first data set (figure 4) the minima of these functions are at $\gamma = -0.6$ and $\gamma = 0.6$, respectively, as software S_3 returns the required knots (in the correct order) for this data set. However, for the second data set (figure 5) the required knots are not returned. In particular, although for the determination of the third knot, a *local* minimum of the least-squares objective functions occurs at $\gamma = 0.6$, there is a 'better' minimum (close to zero), which is chosen by S_3.

Our experience is that the likelihood of S_3 returning a solution that is different from that specified *a priori* for G_3 increases with the number N of knots. Our approach to dealing with this in the validation described in this report is to 'reject' any data set generated by G_3 for which S_3 returns a different (local) solution to that specified *a priori*. Even for modest N this can mean that a large proportion of the data sets generated are rejected, e.g., as many as 97.5 % when $N = 4$. Furthermore, our experience of using the data generator G_3 is that it rarely returns a data set for a solution specified *a priori* for which the residual deviations are particularly 'random', i.e., particularly 'close' to a vector $\mathbf{e}$ of target residuals that are (usually) themselves chosen to be random. Consequently, it is difficult to use the generator to obtain data that is likely to be 'typical' of data encountered in practice.

7 Numerical software testing

In this section are presented the results of applying techniques of *numerical software testing* [6] to investigate the numerical accuracy of the software described in section 5 as implementations of algorithms for solving the problems specified by their respective computational aims given in sections 4. The approach is to apply 'black-box' testing of the software using the data generators described in section 6.

For given values of $\sigma,\ n,\ N,\ \{x_i\}_1^m,\ \mathbf{c}$ and $\boldsymbol{\lambda}$,[19] the data generator G_0 (section 6.1) is used to generate reference data $\{y_{0,i}\}_1^m$ for the problem specified by C_0, with target residuals $\mathbf{e}$ in G_0 chosen to be random samples from a normal (Gaussian) distribution with expectation zero and standard deviation σ. The reference solution corresponding to the reference data is characterised by the vector $\mathbf{r}$ of residual deviations. Software S_0 is applied to the reference data to give test results $\widehat{\mathbf{c}}$ and $\widehat{\boldsymbol{\lambda}}$ that define a solution characterised by the vector $\widehat{\mathbf{r}}$ of residual deviations. The test results are compared with the reference results using the following measures:

[19] For the examples discussed in sections 7 and 8, the order n is set to four (cubic splines), the number m of points to 201, and the data abscissa values x_i to be uniformly-spaced in the interval $[x_{\min}, x_{\max}]$ with $x_{\min} = -1$ and $x_{\max} = 1$. The values of the spline coefficients are chosen randomly, and the knots as described.

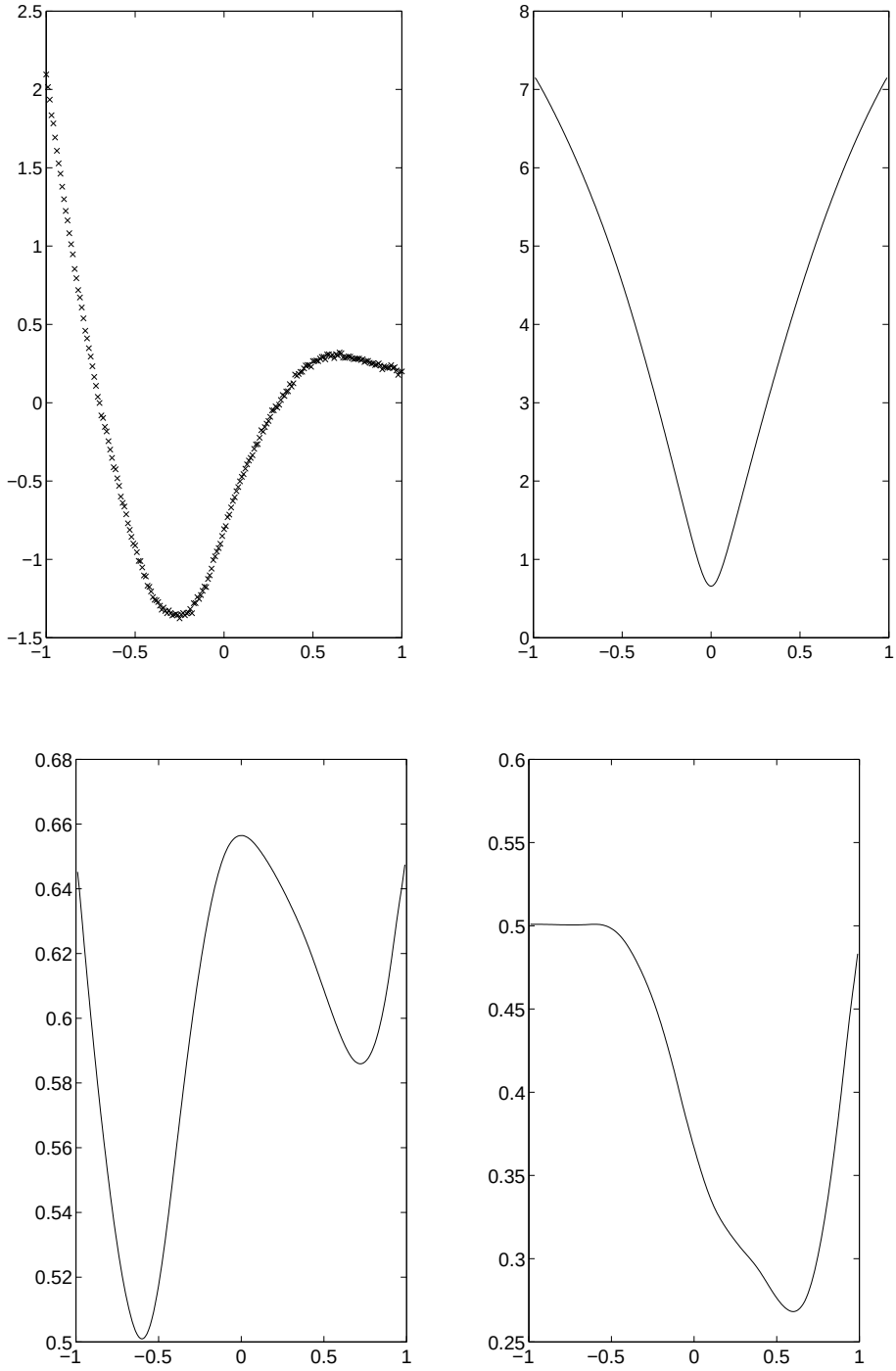

Figure 4: Applying software S_3 to a data set generated using G_3. The data set is shown in the top-left graph. The remaining graphs show how the least-squares objective function varies with the position of the first, second and third inserted knots. The solution returned corresponds to that specified *a priori*.

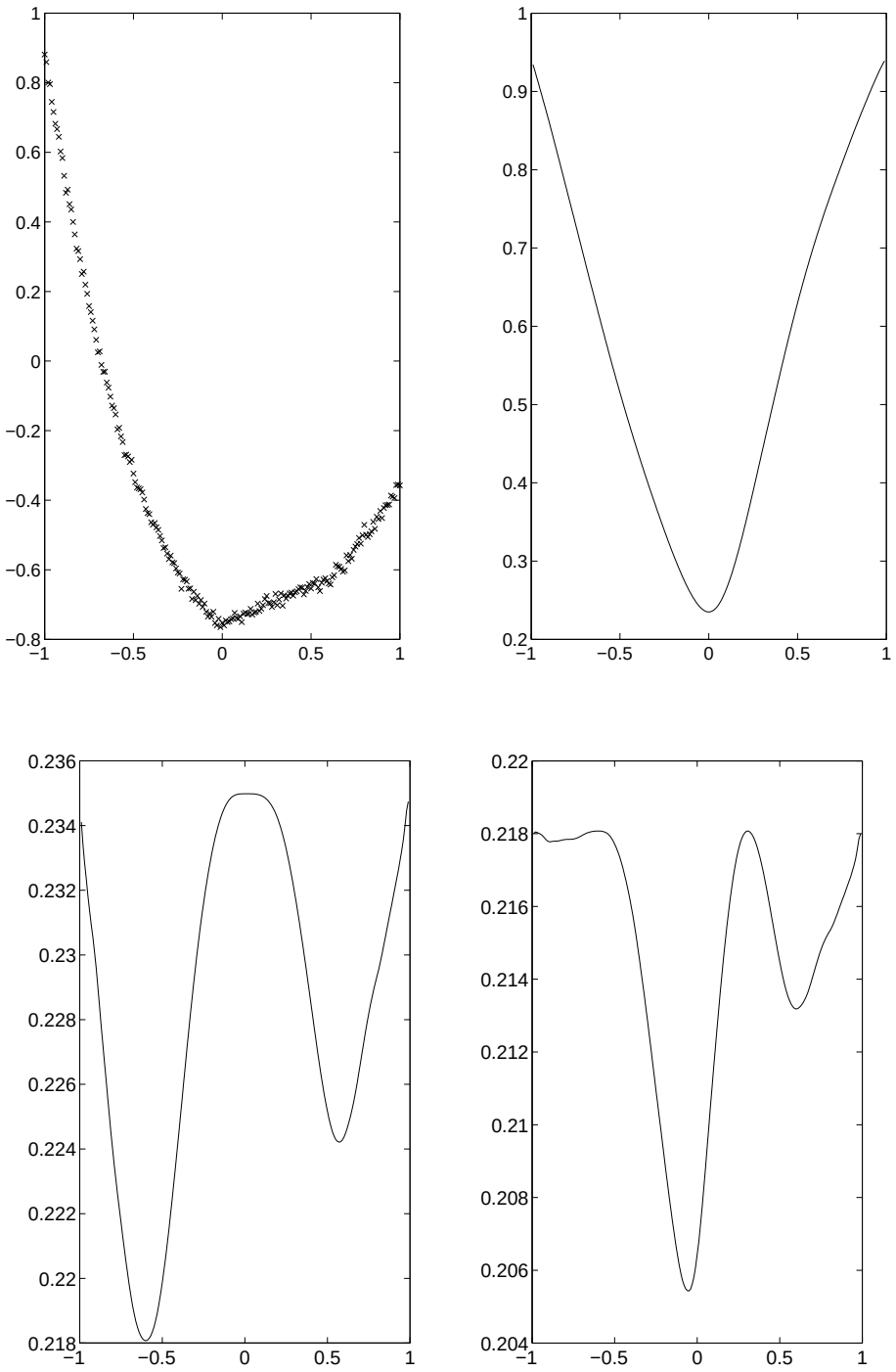

Figure 5: As figure 4, except that the solution returned does not correspond to that specified *a priori*.

Root-mean-square error, defined by

$$d = \frac{1}{\sqrt{m}} \|\widehat{\mathbf{r}} - \mathbf{r}\|.$$

Performance measure, defined by

$$P = \log_{10}\left(1 + \frac{d\sqrt{m}}{\|y_0\|\eta}\right).$$

Maximum (absolute) difference in knots, defined by

$$\Lambda = \max_{j=1,\ldots,N} |\widehat{\lambda}_j - \lambda_j|.$$

Here, P is a 'performance measure' that indicates the number of significant figures of accuracy lost by the (test) software compared with a reference implementation of software to solve the problem specified by C_0. The measure accounts for (a) the relative error in the results returned by the test software, (b) the 'degree of difficulty' of the computational aim C_0 for the reference data set, and (c) the computational precision η of the arithmetic used to compute the test results [6].

The above data generation procedure and tests are repeated a number of times for each of a number of values of σ in the range $[0, 0.1]$, and for the following cases: $N = 1$ knot,[20] $N = 3$ knots[21] and $N = 9$ knots.[22] Figures 6, 10 and 14 (appendix B) show the results of numerical software testing applied to software S_0 for these cases. Each figure contains three graphs that show, respectively, how the measures $d,\ P$ and Λ (above) vary with σ.

The above generation procedure and tests are repeated for:

- software S_1 applied to data $\{y_{1,i}\}_1^m$ generated using G_1 but with knots that satisfy (5): figures 7, 11 and 15,
- software S_2 applied to data $\{y_{2,i}\}_1^m$ generated using G_2 but with knots that satisfy (6): figures 8, 12 and 16, and

[20] With the knot located at

$$\boldsymbol{\lambda} = (-0.5146)^{\mathrm{T}}.$$

[21] With the knots located at

$$\boldsymbol{\lambda} = (-0.6103, 0.0396, 0.4035)^{\mathrm{T}}.$$

[22] With the knots located at

$$\boldsymbol{\lambda} = (-0.8055, -0.6082, -0.3889, -0.2061, -0.0025, 0.1973, 0.3833, 0.5812, 0.8058)^{\mathrm{T}}$$

.

- software S_3 applied to data $\{y_{3,i}\}_1^m$ generated using G_3 but with knots that form a subset of the data abscissa values: figures 9 and 13).[23]

The figures show that for the majority of the reference data sets and for all four software implementations, the root-mean-square error d takes values of the order of 10^{-15}, the performance measure P takes values less than one, and the maximum (absolute) differences Λ in the knots are zero. This indicates that for these reference data sets the software implementations are performing as reference software for the respective computational aims would be expected to perform. There are a few exceptions (figure 10) relating to the application of software S_0 to reference data sets each with a solution with three knots. It can be expected that these exceptions arise in cases where software S_0 has converged to a *local* solution (to the problem specified by C_0) that is different from that specified *a priori* (see the discussion in section 6.5). However, the results obtained from the application of software S_0 to reference data sets each with a solution with nine knots (figure 14), a seemingly 'harder' problem (with more scope for local solutions) than that for three knots, suggest that S_0 is a sufficiently good implementation of an algorithm for solving the problem specified by C_0, at least for the numbers of knots and range of values of σ considered.

8 Algorithm testing

In this section are presented the results of applying the techniques of *algorithm testing* described in section 3 to the problems specified by the computational aims given in section 4. These techniques involve the application of the software described in section 5 and the data generators described in section 6 to investigate the extent to which the 'approximate' computational aims $C_1,\ C_2$ and C_3 can be used to deliver solutions to the 'correct' computational aim C_0, the free-knot spline approximation problem, that are fit for purpose.

8.1 Methods based on using software implementations of the computational aims

Algorithm testing based on using software implementations of the computational aims (section 3.2) follows the procedure of numerical software testing described in section 7 except that *all* four software implementations are applied to reference data $\{y_{0,i}\}_1^m$ generated using the data generator G_0 for the 'correct' computational

[23] Results for the testing of software S_3 applied to data generated using G_3 for the case of nine knots are not presented. For this case, software S_3 invariably returns a 'local' solution for the reference data set that is different from that specified *a priori* for the data generator: see section 6.5.

aim C_0. Figures 17, 18 and 19 (appendix C) show the results for reference data sets each with a solution with a single knot and for values of σ in the range $[0, 0.1]$. Similarly, figures 20, 21 and 22 show the results for reference data sets each with a solution with three knots (and for values of σ in the range $[0, 0.1]$), and figures 23, 24 and 25 for reference data sets each with a solution with nine knots (and for values of σ in the range $[0, 0.02]$).[24]

Since $x_{\min}$ and $x_{\max}$ are identical for *all* the reference data sets, for a given N the knots returned by software S_1 (that locates the knots uniformly between $x_{\min}$ and $x_{\max}$) will be the same. Consequently, the measure Λ of the 'error' in the knots returned by S_1 is independent of the value of σ (figures 17, 20 and 23). Similarly, since the data abscissa values $\{x_i\}_1^m$ are identical for all the reference data sets, for a given N the knots returned by software S_2 (that locates the knots according to the distribution of x_i–values) will be same, and Λ is again independent of σ (figures 18, 21 and 24). Furthermore, for the case $N = 1$, both S_1 and S_2 return, for uniformly-distributed data abscissa values, the (same) knot at the midpoint of the interval $[x_{\min}, x_{\max}]$ (section 4.5). The 'error' Λ is, therefore, the (absolute) difference between this midpoint and the knot position returned by S_0. For this case, the solutions returned by S_1 and S_2 for each reference data set are identical (e.g., figures 17 and 18). As discussed in section 4.5, for the case $N = 1$, the problem specified by C_3 is a 'discrete' version of that specified by C_0. Consequently, for this case, Λ is bounded by the (maximum) separation between consecutive x_i–values. Consequently, in this study,

$$\Lambda < \frac{x_{\max} - x_{\min}}{m - 1} = \frac{2}{200} = 0.01.$$

Only in special cases will the knot placement strategies, that form the basis of the approximate algorithms considered here, return knots that correspond to the solution to the free-knot spline approximation problem.[25] Consequently, having found the 'wrong' knot(s), the fitted spline curves returned by $S_1,\ S_2$ and S_3 will differ from that returned by S_0. The measures d and P quantify the difference. The results show that, generally, only one or two significant figures in the values of the residual deviations (that characterise the solutions) can be expected to be correct.[26] For some applications (e.g., for those for which σ is large) this level of accuracy may be acceptable. The 'most accurate' results obtained in this study arise from the use of S_3 in the case of a single knot (figure 19). This is to be expected as it corresponds to the 'most accurate' determination of the knot(s).

Finally, the results shown in some of the figures suggest that that there is a tendency for the measures d and P to increase with σ (e.g., figure 17). However, the ranges

[24] A smaller range of values for σ was chosen for the case of $N = 9$ knots because of the difficulties experienced when using the data generators G_0 (and G_3) to generate data sets with solutions specified *a priori*, the difficulties being exacerbated as σ is increased (section 6.5).

[25] Some of these cases are identified in section 4.5.

[26] A value of 16 for P corresponds, roughly, to *no* figures of accuracy.

of values of d and P are small compared to their respective (mean) values and, consequently, the effect is not regarded as significant.

8.2 Methods based on using data generator software for the computational aims

For given values of $\sigma,\ n,\ N, \{x_i\}_1^m,\ \mathbf{c}$ and $\boldsymbol{\lambda}$ (the latter satisfying (5)), data generation software G_0 and G_1 is used to generate, respectively, data $\{y_{0,i}\}_1^m$ and $\{y_{1,i}\}_1^m$ such that $\mathbf{c}$ and $\boldsymbol{\lambda}$ is the solution for the problems specified by computational aims C_0 and C_1 applied to their respective data sets (section 3.3). The data $\{y_{0,i}\}_1^m$ is chosen to have residual deviations that are 'close to' values sampled randomly from a normal distribution with zero expectation and standard deviation σ. The data $\{y_{1,i}\}_1^m$ is chosen to be 'close to' the data $\{y_{0,i}\}_1^m$. The 'distance' between the data sets is measured by the root-mean-square error d, as defined in section 7.

Figures 26 and 29 (appendix D) show the results obtained for the cases $N = 1$ and $N = 3$, respectively, for a number of data sets for each of a number of values of σ in the range $[0, 0.1]$. Figures 27 and 30 show corresponding results for data generated using data generation software G_0 and G_2 with knots satisfying (6). Finally, figures 28 and 31 show corresponding results for data generation software G_0 and G_3 with knots that form a subset of the data abscissa values.

Generally, the 'distance' between the computational aims measured by d is zero (being, numerically, of the order of the computational precision η). The exception is in figure 31 corresponding to the comparison of C_0 and C_3 using $N = 3$ knots. In this case, the distance between the data sets (and, hence, between the computational aims) takes the *same* value for all values of σ. Furthermore, the value of the root-mean-square error is comparable to that observed when applying software S_3 to data generated using G_0 (section 8.1 and figure 22). Further work is required to explain this result and relationship.

8.3 Methods based on simulation

For given values of $\sigma,\ n,\ N,\ \{x_i\}_1^m,\ \mathbf{c},\ \boldsymbol{\lambda}$ and M, repeat the following steps for $r = 1, \ldots, M$ (section 3.4):

1. Construct the data set $(x_i, y_i^r),\ i = 1, \ldots, m$, by evaluating

$$y_i^r = s(\mathbf{c}, \boldsymbol{\lambda}; x_i) + e_i^r, \qquad i = 1, \ldots, m,$$

where e_i^r is a random sample from $N(0, \sigma^2)$, a normal (Gaussian) distribution with mean zero and standard deviation σ.

Implementation	$\sigma = 0.01$	$\sigma = 0.05$
S_0	0.002 9	0.013 7
S_1	0.001 3	0.006 5
S_2	0.001 4	0.006 8
S_3	0.001 7	0.009 1

Table 1: Standard deviations of the PDFs for the value y_0 at $x = 0$ of the values of the fitted spline curve returned by each software implementation for the two simulations corresponding to $\sigma = 0.01$ and $\sigma = 0.05$.

2. Apply software S_0 to the data to obtain spline coefficients $\mathbf{c}_0$ and knots $\boldsymbol{\lambda}_0$. Evaluate $y_0^r = s(\mathbf{c}_0, \boldsymbol{\lambda}_0; 0)$.

3. Apply software S_1 to the data to obtain spline coefficients $\mathbf{c}_1$ and knots $\boldsymbol{\lambda}_1$. Evaluate $y_1^r = s(\mathbf{c}_1, \boldsymbol{\lambda}_1; 0)$.

4. Apply software S_2 to the data to obtain spline coefficients $\mathbf{c}_2$ and knots $\boldsymbol{\lambda}_2$. Evaluate $y_2^r = s(\mathbf{c}_2, \boldsymbol{\lambda}_2; 0)$.

5. Apply software S_3 to the data to obtain spline coefficients $\mathbf{c}_3$ and knots $\boldsymbol{\lambda}_3$. Evaluate $y_3^r = s(\mathbf{c}_3, \boldsymbol{\lambda}_3; 0)$.

For the case $\sigma = 0.01, N = 3, \boldsymbol{\lambda} = (-0.5, 0, 0.5)^{\mathrm{T}}$ and $M = 10\ 000$, figure 32 (appendix E) shows, in the form of histograms (scaled frequency distributions), approximations to the PDFs for:

- the values of the positions of the knots returned by software S_0, and
- the value y_0 at $x = 0$ of the fitted spline curve returned by S_0.[27]

In this figure, the 'correct' positions $\boldsymbol{\lambda}$ of the spline underlying the simulation (defined above), and the 'correct' value at $x = 0$ of this spline, are indicated by vertical (dashed) lines. Figures 33–35 show corresponding results for the solutions returned by S_1, S_2 and S_3. Similarly, figures 36–39 show the results obtained from the four software implementations for the case $\sigma = 0.05$. Table 1 contains the standard deviations[28] of the PDFs for the value y_0 at $x = 0$ of the fitted spline curve returned by each software implementation for the two simulations based on the different values of σ.

[27] The abscissa value $x = 0$ is at the midpoint of the interval $[-1, 1]$ containing the values $\{x_i\}_1^m$.

[28] The standard deviation evaluated from the distribution can be used as a measure of dispersion or spread of values that can reasonably be attributed to the quantity y_0.

The results obtained using S_0 (figures 32 and 36) show distributions that are 'normal' in appearance, centred on the 'correct' values (thus, exhibiting no bias),[29] and with dispersions (e.g., table 1) that are larger for the larger value of σ. For this particular simulation, the knots of the spline underlying the simulation are uniformly-spaced. Consequently, S_1 will *always* return the 'correct' knots, and the PDFs for the values of the positions of the knots returned by S_1 are 'Dirac-delta' functions centred on the 'correct' positions (figures 33 and 37). The dispersion of values y_0 at $x = 0$ of the fitted spline curve returned by S_1 is about half that for the curves returned by S_1.

The knots returned by S_2 will not be uniformly-spaced, but will be the same for all simulated data sets, because the distribution of the data abscissa values is the same for all data sets. Consequently, the PDFs for the values of the positions of the knots returned by S_2 are also Dirac-delta functions, but *not* centred on the correct positions (figures 34 and 38). Using this knot strategy, the PDF for the value y_0 exhibits a bias, i.e., the PDF does not have an expectation equal to the 'correct' value. The bias remains approximately the same for the two choices of σ, but the dispersion of values y_0 is larger for the larger value of σ. The dispersion of values returned by S_2 is comparable to that for S_1.

Finally, the PDFs for the values of the positions of the knots returned by S_3 exhibit a bias compared to their 'correct' positions but are not Dirac-delta functions, i.e., different knot positions are returned for different simulated data sets (figures 35 and 39). As for S_2, the bias in the values of the positions of the knots yields a bias in the value y_0. The dispersion of values y_0 is somewhere between that for S_0 and S_1 (or S_2).

9 Conclusions

This report describes a case study for testing and validating contributions to EUROMETROS. It brings together work undertaken within the Software Support for Metrology (SS*f*M) programmes on

- algorithms and software for discrete modelling (in particular, using polynomial spline curves),
- numerical software testing,
- algorithm testing, and
- disseminating algorithms and software through EUROMETROS.

[29] The expectation evaluated from the distribution, and its difference from the 'correct' value, can be used to quantify bias.

The case study concerns the application of numerical software testing and algorithm testing techniques to software implementations of approximate algorithms for solving the free-knot spline approximation problem. The free-knot spline approximation problem is a non-linear problem, and there are theoretical difficulties associated with the existence, uniqueness and characterisation of solutions to the problem. For these reasons, approximate algorithms, implementing various knot placement strategies and based on solving (sequences of) linear problems, are often used in place of the free-knot spline approximation problem. Software implementations of a number of such approximate algorithms are contained within a software package available from EUROMETROS.

The case study has focussed on three such approximate algorithms. The algorithms are based on knot placement strategies that select knots to be uniformly-spaced, based on the distribution of the data abscissa values, and using a (sequential) knot-insertion strategy. The aim has then been to undertake *validation and testing* of software implementations of the algorithms using a number of techniques, including an analysis of the algorithms, using software implementations of the algorithms, data generators for the algorithms, and simulation. The 'ingredients' for the application of these techniques have been described, viz., for each algorithm:

- a specification of the computational aim,
- information about the software implementation, and
- a procedure for generating data for which the solution to the problem specified by the computational aim is prescribed *a priori*.

Data generators are a key ingredient in the application of the techniques. The case study has highlighted the effect of basing data generators (for non-linear problems) on the characterisations of *local* solutions, and how this can lead to difficulty with the interpretation of results in cases where a software implementation returns a different (perhaps 'better') local solution than that specified *a priori*. It has also highlighted the potential for substantial *complexity* of a data generator, such as for the algorithm based on the knot-insertion strategy. There are very many problems arising in metrology for which the design and implementation of a data generator is straightforward. However, in cases where the complexity of the data generator is comparable to, or exceeds, that of the problem to be solved, the benefits of software and algorithm testing techniques employing such a generator may be less clear.

Numerical software testing techniques have been used to show that the software implementations provide, for the examples considered, numerically reliable solutions to the problems specified by their respective computational aims. The testing employs reference data and corresponding reference results, obtained using the data generators, to undertake 'black-box' testing of the software implementations.

Several metrics, including a performance measure, are used to quantify the accuracy of the test results returned by the software.

An analysis of the computational aims has identified some of the circumstances under which the free-knot spline approximation problem and the approximate algorithms used to solve that problem have the same solution. Generally, these circumstances are 'trivial' (e.g., relating to the case $N = 0$) or quite restrictive regarding the problem to be solved. Consequently, it is expected that, in practice, the various algorithms will return different solutions. The algorithm testing techniques considered are intended to *quantify* the differences between the solutions returned.

Methods of algorithm testing based on using software implementations of the computational aims are similar in the way they are applied to methods of numerical software testing, excepting that the software implementations are applied to the *same* data sets for which the solutions to the free-knot spline approximation problem are known. The results show that, for the examples considered, generally only one or two significant figures in the values of the residual deviations that characterise the solutions returned by the approximate algorithms can be expected to be correct. For some applications this level of agreement may be acceptable. In particular, from the user's point of view there is no 'right' answer: what is required is *a* function that explains the data adequately.

Methods of algorithm testing based on data generator software for the computational aims have been more difficult to apply for the reasons indicated above. They seek to measure the 'distance' between the collections of data sets for which the problems specified by (different) computational aims have the same solution. The results show that, for the examples considered, this distance is zero, excepting for the comparison of the free-knot spline approximation problem and the algorithm based on the knot-insertion strategy when $N = 3$. Consequently, there are collections of data sets for which, for example, the free-knot spline approximation problem and the approximate algorithm based on selecting uniform knots have the same solution. However, such collections may not exist for the free-knot spline approximation problem and the approximate algorithm based on the knot-insertion strategy. Further work is required to explain this result.

Finally, methods of algorithm testing based on simulation seek to make 'statistical' comparisons between the algorithms by comparing the probability density functions (PDFs) for the values of the output quantities,[30] or for the value of a derived quantity,[31] returned by the algorithms. Such comparisons are useful in metrology for which considerations of 'uncertainty' are important. If the difference between the results returned by two algorithms is small compared to the uncertainty associated with the difference, the difference can be safely ignored and the use

[30] Here, spline coefficients and knot positions.

[31] Here, a value of the fitted spline model.

of either algorithm regarded as fit for purpose. The results show that, for the examples considered, *bias* in the results returned by the approximate algorithms can be obtained. Furthermore, the magnitude of the bias, and of the dispersion of values of the output and derived quantities, depends on the uncertainty associated with the measurements. For the (particular) examples considered, for which the knot positions are chosen to be uniformly spaced, the approximate algorithm that positions the knots uniformly returns results that are comparable to (perhaps 'better' by design than) those returned for the free-knot spline approximation problem. However, the other approximate algorithms return results that exhibit a bias and, consequently, for some applications will not be fit for purpose.

Acknowledgements

The work described here was supported by the National Measurement System Directorate of the UK Department of Trade and Industry as part of its NMS Software Support for Metrology programme.

We thank our colleague Maurice Cox for his comments on this report.

References

[1] P. M. Harris A. B. Forbes and I. M. Smith. Correctness of free form surface fitting software. In D. G. Ford, editor, *Laser Metrology and Machine Performance VI*, pages 263–272, Southampton, 2003. WIT Press.

[2] G. T. Anthony and M. G. Cox. The National Physical Laboratory's Data Approximation Subroutine Library. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*, pages 669–687, Oxford, 1987. Clarendon Press.

[3] R. M. Barker. Good Practice Guide No. 5. Guide to EUROMETROS: a manual for users, contributors and testers. Technical report, National Physical Laboratory, Teddington, UK, 2004. Available for download from the SS*f*M website at www.npl.co.uk/ssfm/download/bpg.html.

[4] R. M. Barker, M. G. Cox, A. B. Forbes, and P. M. Harris. Best Practice Guide No. 4. Discrete modelling and experimental data analysis. Technical report, National Physical Laboratory, Teddington, UK, 2004. Available for download from the SS*f*M website at www.npl.co.uk/ssfm/download/bpg.html.

[5] R. M. Barker, M. G. Cox, P. M. Harris, and I. M. Smith. Testing algorithms in standards and METROS. Technical Report CMSC 18/03, National Physical Laboratory, 2003.

[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. Technical Report CMSC 25/99, National Physical Laboratory, Teddington, UK, 1999.

[7] M. G. Cox. The numerical evaluation of B-splines. *J. Inst. Math. Appl.*, 10:134–149, 1972.

[8] M. G. Cox. The incorporation of boundary conditions in spline approximation problems. In G. A. Watson, editor, *Lecture Notes in Numerical Analysis 630: Numerical Analysis*, pages 51–63, Berlin, 1978. Springer-Verlag.

[9] M. G. Cox. Algorithms for spline curves and surfaces. In L. Piegl, editor, *Fundamental Developments of Computer-Aided Geometric Modelling*, pages 51–76, London, 1993. Academic Press.

[10] M. G. Cox, A. B. Forbes, P. M. Harris, and I. M. Smith. The classification and solution of regression problems for calibration. Technical Report CMSC 24/03, National Physical Laboratory, Teddington, UK, 2003.

[11] M. G. Cox and P. M. Harris. Design and use of reference data sets for testing scientific software. *Analytica Chimica Acta*, 380:339 – 351, 1999.

[12] M. G. Cox, P. M. Harris, E. J. Johnson, P. D. Kenward, and G. I. Parkin. Testing the numerical correctness of software. Technical Report CMSC 34/04, National Physical Laboratory, Teddington, UK, 2004.

[13] M. G. Cox, P. M. Harris, and P. D. Kenward. Fixed- and free-knot univariate least-squares data approximation by polynomial splines. Technical Report CMSC 13/02, National Physical Laboratory, Teddington, UK, 2002.

[14] M. G. Cox, P. M. Harris, and P. D. Kenward. Fixed- and free-knot univariate least-squares data approximation by polynomial splines. In I. J. Anderson J. Levesley and J. C. Mason, editors, *Algorithms for Approximation IV*, pages 330–345, Huddersfield, UK, 2002. University of Huddersfield.

[15] M. G. Cox, P. M. Harris, P. D. Kenward, and I. M. Smith. Extracting features from experimental data. Technical Report CMSC 22/03, National Physical Laboratory, Teddington, UK, 2003.

[16] C. de Boor. On calculating with B-splines. *J. Approx. Theory*, 6:50–62, 1972.

[17] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.

[18] A. B. Forbes, P. M. Harris, and R. K. Leach. The comparison of algorithms for the assessment of Type A1 surface texture reference artefacts. Technical Report CMSC 33/03, National Physical Laboratory, Teddington, UK, 2003.

[19] B. Ford, J. Bentley, J. J. du Croz, and S. J. Hague. The NAG Library 'machine'. *Software - Practice and Experience*, 9:56–72, 1979.

[20] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, USA, 1996. Third edition.

[22] P. M. Harris. *Knot Placement in Spline Approximation Problems*. PhD thesis, Brunel University, 1991.

[23] I. J. Schoenberg and Anne Whitney. On Pólya frequency functions III. *Trans. Am. Math.*, 74:246–259, 1953.

[24] J. H. Wilkinson. *Notes in Applied Science No. 32. Rounding Errors in Algebraic Processes*. Her Majesty's Stationery Office, London, 1963.

A Differentiating a spline with respect to its knots

In this appendix is considered the problem of computing the first derivatives of a B-spline basis function with respect to its (interior) knots. The calculation is necessary for the implementation of software S_0 for the computational aim C_0 (section 5.1). The calculation is also necessary for implementations of the data generators G_0 and G_3 for the computational aims C_0 and C_3 (sections 6.1 and 6.4, respectively).

Recall the B-spline basis function $N_{n,k}(\boldsymbol{\lambda}; x)$ is a function of the vector of knots $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)^{\mathrm{T}}$ and the (independent) variable x that satisfy, respectively, the constraints

$$x_{\min} < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N < x_{\max}, \tag{15}$$

and

$$x_{\min} \leq x \leq x_{\max}, \tag{16}$$

for some $x_{\min} < x_{\max}$ (see section 2.1). Let $\boldsymbol{\lambda}^*$ denote a vector of given *values* for the knots $\boldsymbol{\lambda}$ that satisfy (15), and x^* a *value* of x satisfying (16). The problem is to compute (if it exists) the derivative with respect to λ_j of the B-spline basis function $N_{n,k}(\boldsymbol{\lambda}; x)$ evaluated at $(\boldsymbol{\lambda}^*, x^*)$.

Now the normalized B-spline basis function $N_{n_k}(\boldsymbol{\lambda}; x)$ is defined as a certain divided difference of the truncated power function [7, 16]

$$N_{n,k}(\boldsymbol{\lambda}; x) = (\lambda_k - \lambda_{k-n})\, [\lambda_{k-n}, \ldots, \lambda_k]\, (\cdot - x)_+^{n-1},$$

where

$$(x - \lambda)_+^n = \begin{cases} 0, & x < \lambda, \\ (x - \lambda)^n & x \geq \lambda. \end{cases}$$

Now, since $\lambda_{k-n} < \lambda_k$,[32] it follows (by the properties of divided differences) that

$$N_{n,k}(\boldsymbol{\lambda}; x) = [\lambda_{k-n+1}, \ldots, \lambda_k]\, (\cdot - x)_+^{n-1} - [\lambda_{k-n}, \ldots, \lambda_{k-1}]\, (\cdot - x)_+^{n-1},$$

and so

$$\left.\frac{\partial}{\partial \lambda_j} N_{n,k}(\boldsymbol{\lambda}; x)\right|_{(\lambda^*, x^*)} = d_{kj} - d_{k-1,j}$$

where

$$d_{kj} = \left.\frac{\partial}{\partial \lambda_j}\, [\lambda_{k-n+1}, \ldots, \lambda_k]\, (\cdot - x)_+^{n-1}\right|_{(\lambda^*, x^*)}.$$

Therefore, the required derivative is expressed in terms of the quantities d_{kj} and $d_{k-1,j}$ which are themselves derivatives.

[32] Otherwise, $N_{n,k}$ is identically zero, and not a basis function.

Clearly, if $j < k-n+1$ or $j > k$, $d_{kj} = 0$. So, let $k-n+1 \leq j \leq k$ and suppose that $n \geq 3$ and $\lambda^*_{j-1} < \lambda^*_j < \lambda^*_{j+1}$.[33] Then,

$$\begin{aligned} d_{kj} &= \lim_{\delta \to 0} \frac{1}{\delta} \Big(\Big[\lambda^*_{k-n+1}, \ldots, \lambda^*_{j-1}, \lambda^*_j + \delta, \lambda^*_{j+1}, \ldots, \lambda^*_k \Big] (\cdot - x^*)^{n-1}_+ \\ &\qquad - \Big[\lambda^*_{k-n+1}, \ldots, \lambda^*_{j-1}, \lambda^*_j, \lambda^*_{j+1}, \ldots, \lambda^*_k \Big] (\cdot - x^*)^{n-1}_+ \Big) \\ &= \lim_{\delta \to 0} \Big[\lambda^*_{k-n+1}, \ldots, \lambda^*_{j-1}, \lambda^*_j, \lambda^*_j + \delta, \lambda^*_{j+1}, \ldots, \lambda^*_k \Big] (\cdot - x^*)^{n-1}_+ \\ &= \Big[\lambda^*_{k-n+1}, \ldots, \lambda^*_{j-1}, \lambda^*_j, \lambda^*_j, \lambda^*_{j+1}, \ldots, \lambda^*_k \Big] (\cdot - x^*)^{n-1}_+. \end{aligned}$$

It follows that d_{kj} is the value at x^* of the $(k+1)$th *un-normalized* B-spline of order n defined by the knots $\boldsymbol{\lambda}^*$ with the cardinality of λ^*_j increased by one. Defining the new knot vector $\boldsymbol{\tau}$ by

$$\tau_\ell = \begin{cases} \lambda^*_\ell & \text{for} \quad \ell \leq j, \\ \lambda^*_{\ell-1} & \text{for} \quad \ell > j, \end{cases}$$

then

$$d_{kj} = \frac{\hat{N}_{n,k+1}(\boldsymbol{\tau}; x^*)}{\tau_{k+1} - \tau_{k-n+1}},$$

where $\hat{N}_{n,k}$ denotes the kth normalized B-spline of order n with knots $\boldsymbol{\tau}$.

The conditions that $n \geq 3$ and λ^*_j is a simple knot ensure that $\hat{N}_{n,k}$ is a continuous function, and so there is no difficulty when evaluating the function for any $x = x^*$. When these conditions do not apply, however, care is required with the above analysis. For example, suppose there are *equalities* in the constraint (15) at the point $(\boldsymbol{\lambda}^*, x^*)$; in particular, suppose

$$\lambda^*_{j-2} < \lambda^*_{j-1} = \lambda^*_j = \lambda^*_{j+1} < \lambda^*_{j+2}.$$

Then, there is no non-zero perturbation δ in λ_j about λ^*_j for which the constraint (15) remains valid, and so $N_{n,k}$ is not differentiable with respect to λ_j at the given point. Similarly, perturbations in λ^*_{j-1} and λ^*_{j+1} are constrained to be from below ($\delta \to 0^-$) and from above ($\delta \to 0^+$), respectively. It is important to note the use of these one-sided limits because it affects the evaluation of $\hat{N}_{n,k+1}$ in these cases. A treatment of these special cases is available [22].

[33] In other words, the basis function is (piecewise) quadratic or higher order, and λ^*_j is a 'simple' knot of cardinality one in the knot vector $\boldsymbol{\lambda}^*$. For the purposes of this study, these conditions are assumed to apply, although for more general problems the conditions may be relaxed.

B Results of numerical software testing

This appendix contains the results (section 7) of applying techniques of *numerical software testing* to investigate the numerical accuracy of the software described in section 5. The following figures are included:

Figure 6 Software S_0 applied to data generated using data generator G_0: one knot.

Figure 7 Software S_1 applied to data generated using data generator G_1: one knot.

Figure 8 Software S_2 applied to data generated using data generator G_2: one knot.

Figure 9 Software S_3 applied to data generated using data generator G_3: one knot.

Figure 10 Software S_0 applied to data generated using data generator G_0: three knots.

Figure 11 Software S_1 applied to data generated using data generator G_1: three knots.

Figure 12 Software S_2 applied to data generated using data generator G_2: three knots.

Figure 13 Software S_3 applied to data generated using data generator G_3: three knots.

Figure 14 Software S_0 applied to data generated using data generator G_0: nine knots.

Figure 15 Software S_1 applied to data generated using data generator G_1: nine knots.

Figure 16 Software S_2 applied to data generated using data generator G_2: nine knots.

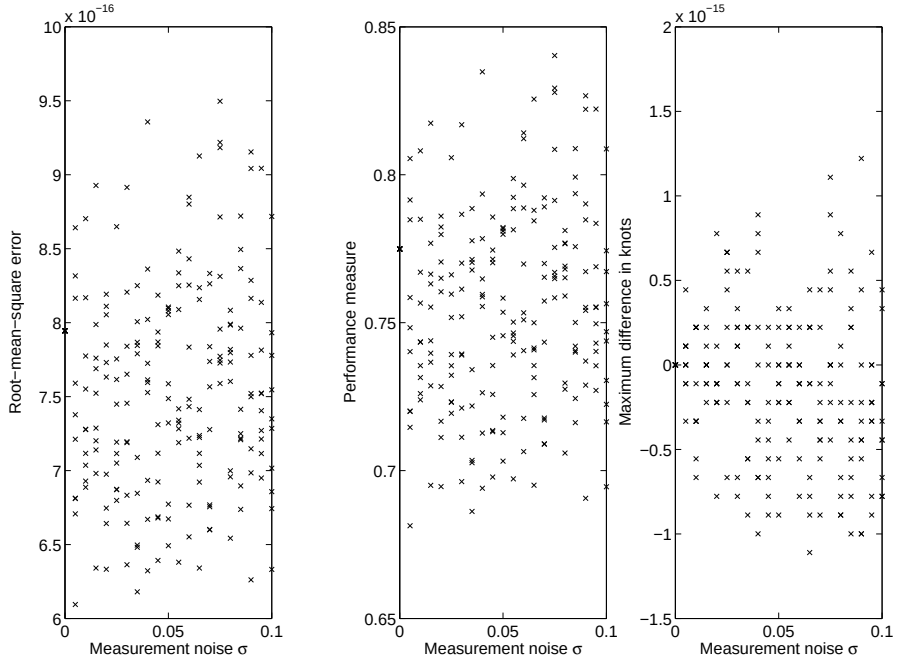

Figure 6: S_0 applied to data generated using G_0: one knot.

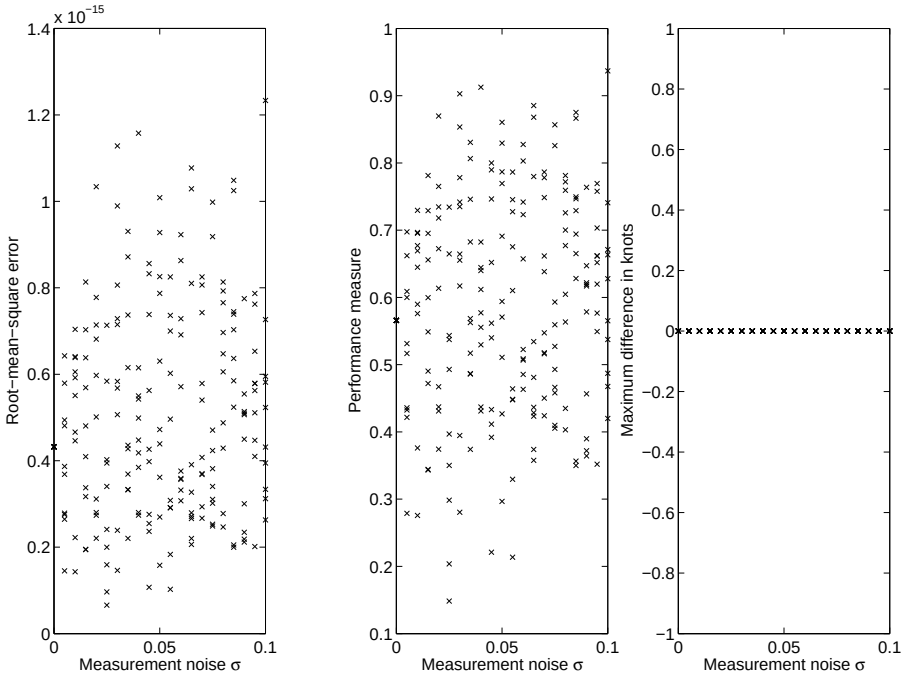

Figure 7: S_1 applied to data generated using G_1: one knot.

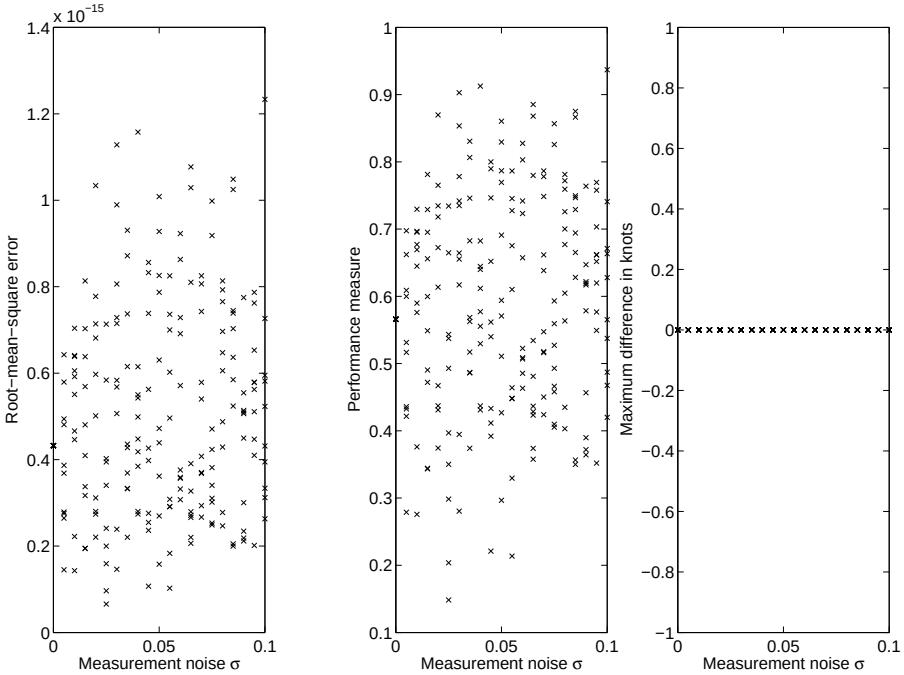

Figure 8: S_2 applied to data generated using G_2: one knot.

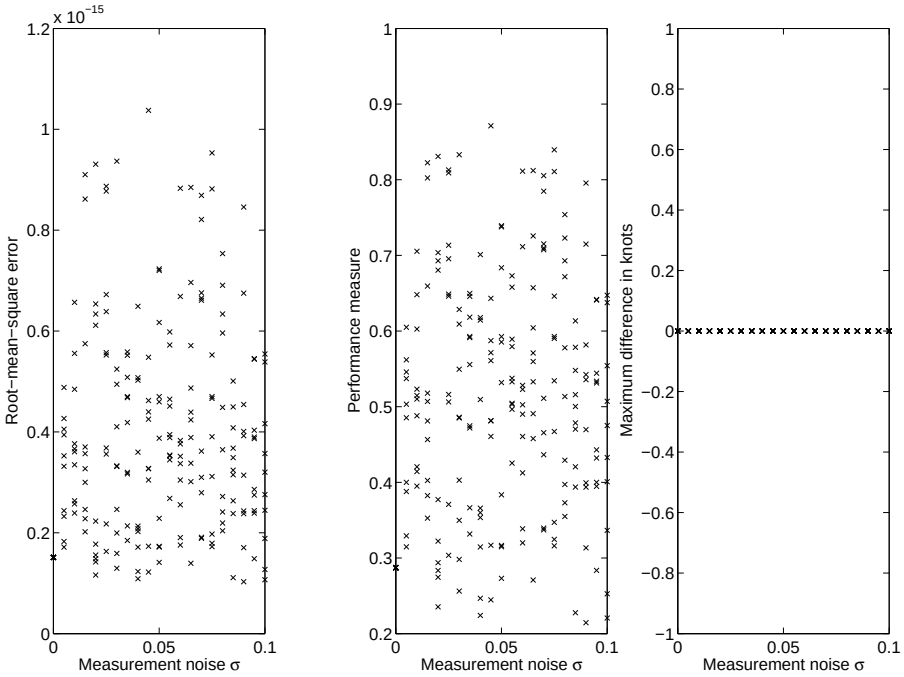

Figure 9: S_3 applied to data generated using G_3: one knot.

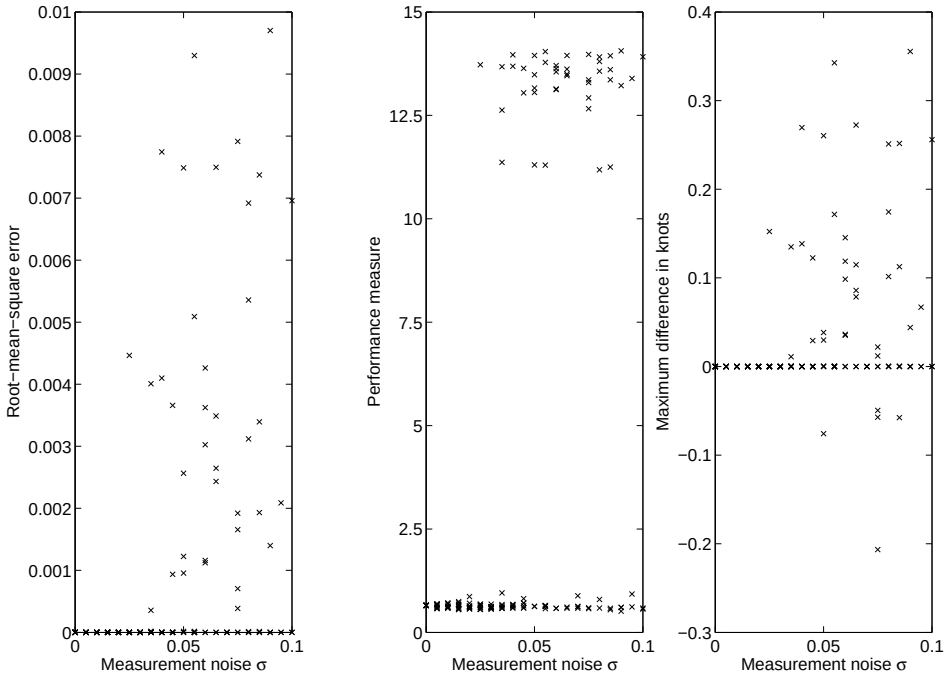

Figure 10: S_0 applied to data generated using G_0: three knots.

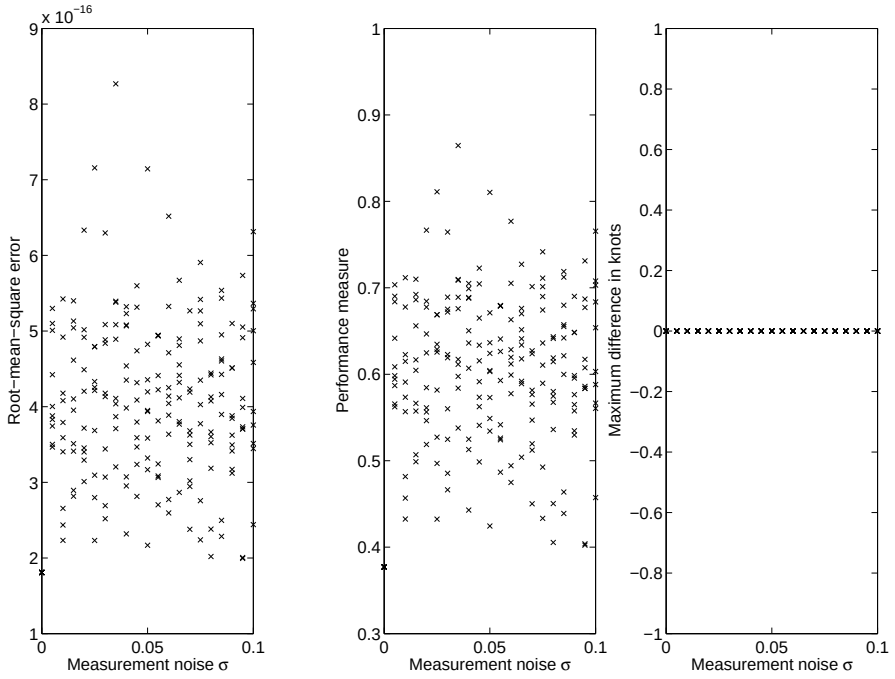

Figure 11: S_1 applied to data generated using G_1: three knots.

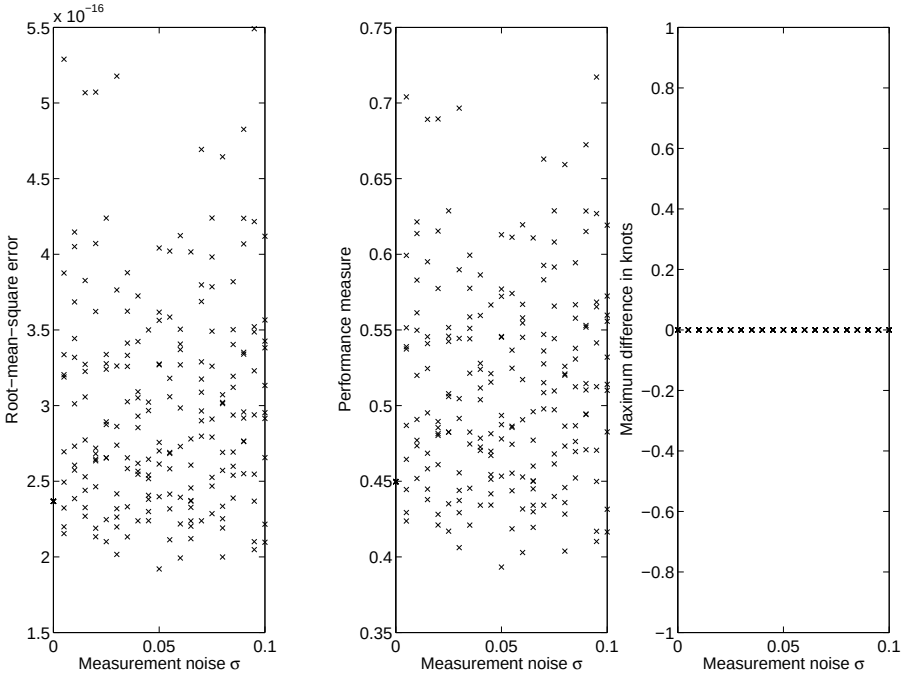

Figure 12: S_2 applied to data generated using G_2: three knots.

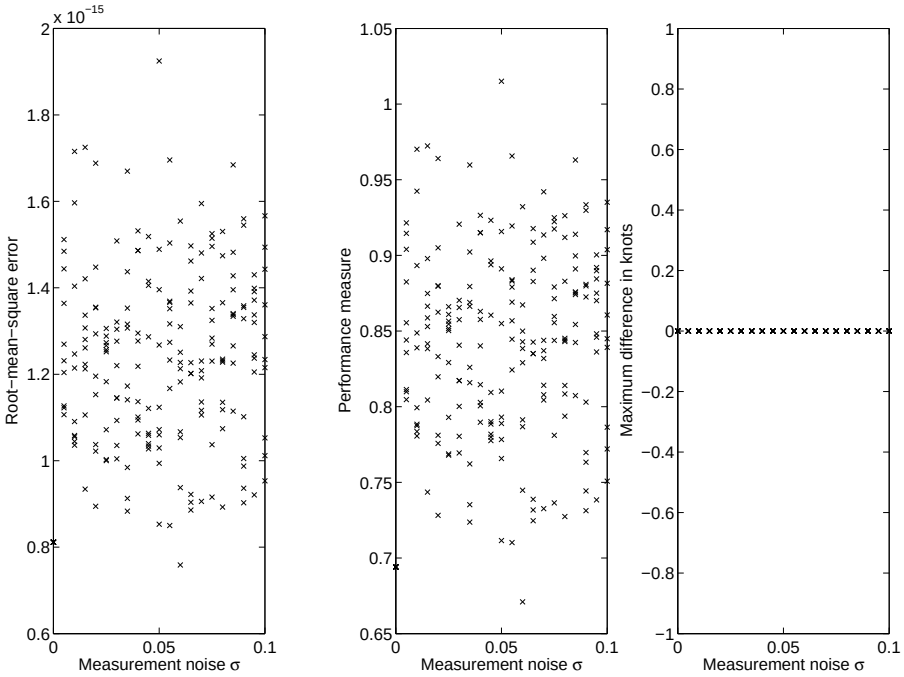

Figure 13: S_3 applied to data generated using G_3: three knots.

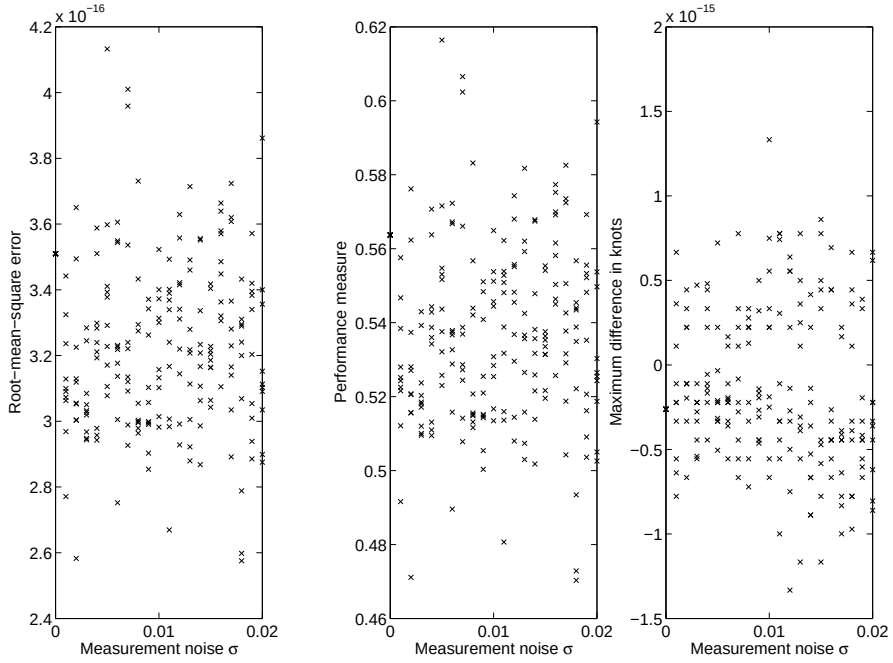

Figure 14: S_0 applied to data generated using G_0: nine knots.

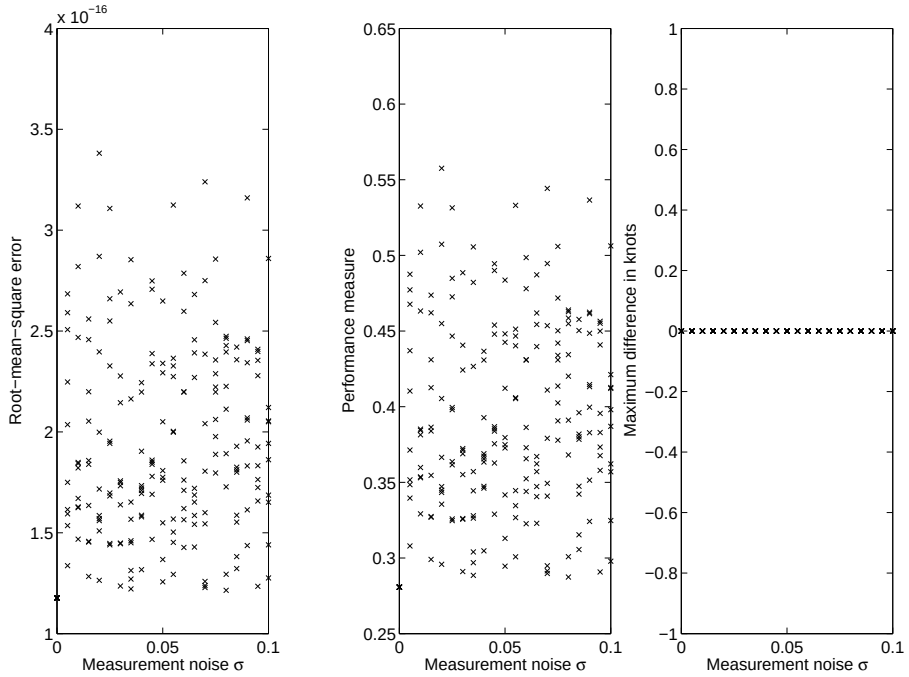

Figure 15: S_1 applied to data generated using G_1: nine knots.

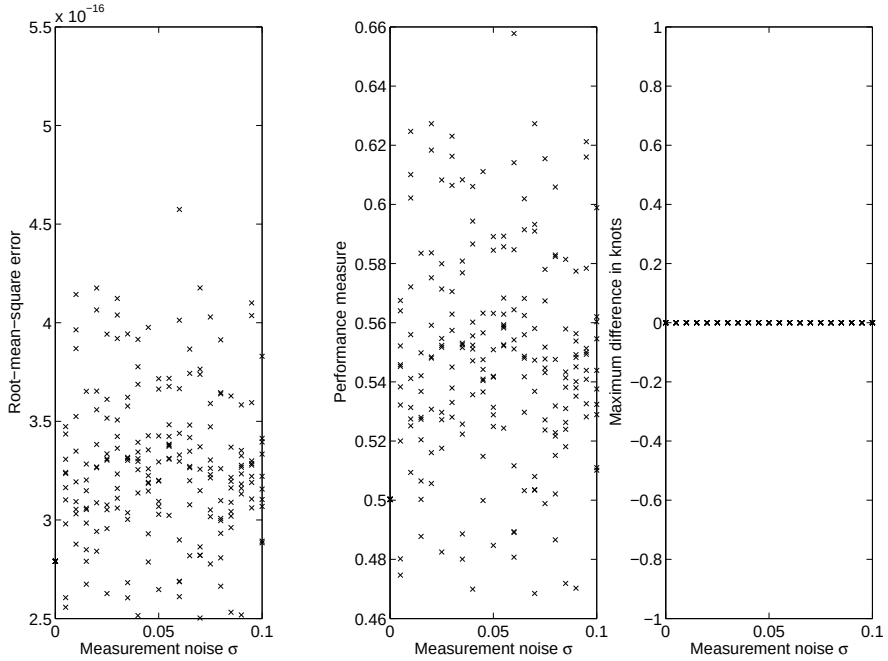

Figure 16: S_2 applied to data generated using G_2: nine knots.

C Results from using software implementations of the computational aims

This appendix contains the results (section 8.1) of the *algorithm testing* obtained using software implementations of the computational aims. The following figures are included:

Figure 17 Software S_1 applied to data generated using data generator G_0: one knot.

Figure 18 Software S_2 applied to data generated using data generator G_0: one knot.

Figure 19 Software S_3 applied to data generated using data generator G_0: one knot.

Figure 20 Software S_1 applied to data generated using data generator G_0: three knots.

Figure 21 Software S_2 applied to data generated using data generator G_0: three knots.

Figure 22 Software S_3 applied to data generated using data generator G_0: three knots.

Figure 23 Software S_1 applied to data generated using data generator G_0: nine knots.

Figure 24 Software S_2 applied to data generated using data generator G_0: nine knots.

Figure 25 Software S_3 applied to data generated using data generator G_0: nine knots.

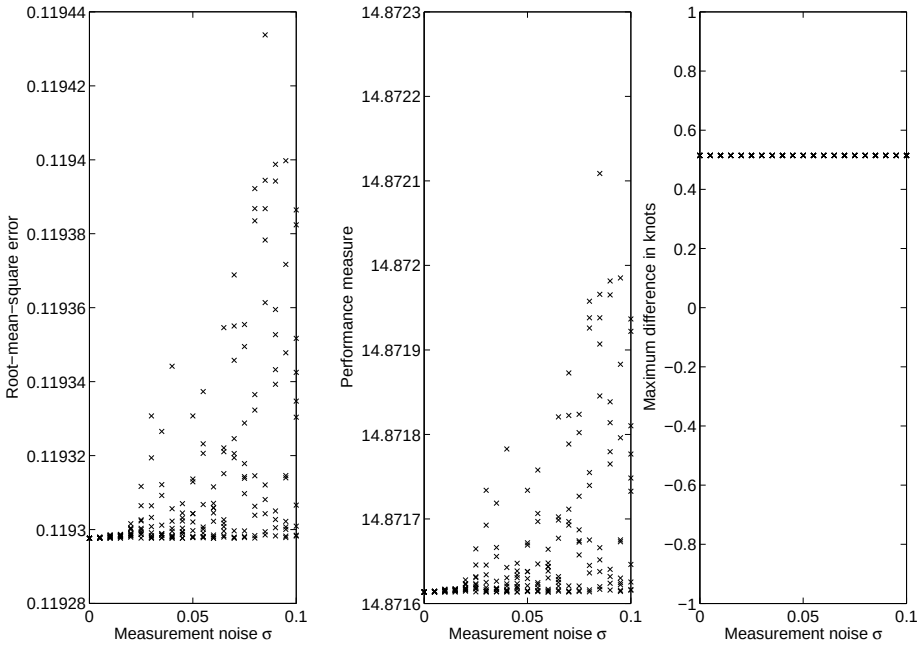

Figure 17: S_1 applied to data generated using G_0: one knot.

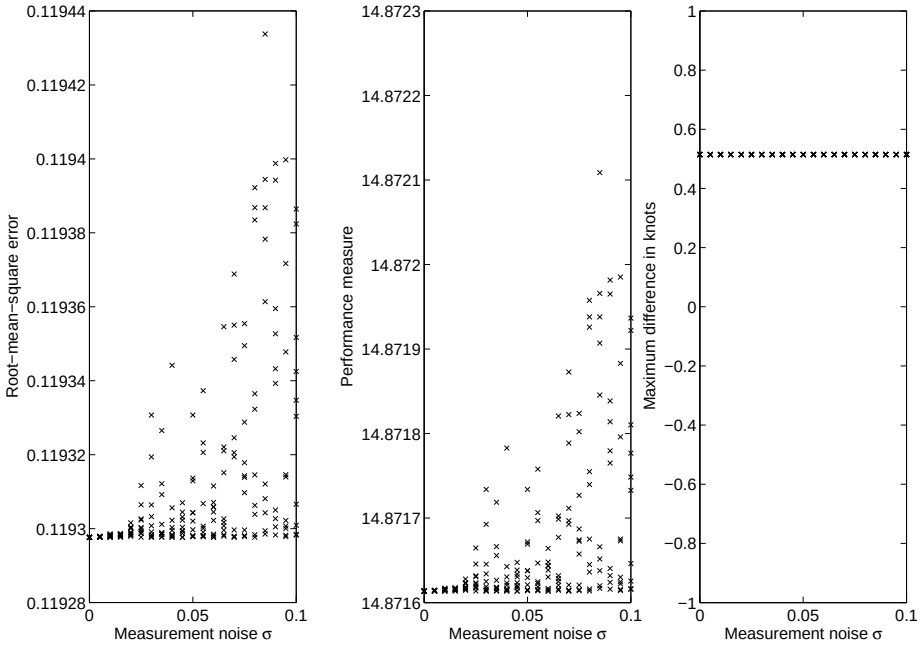

Figure 18: S_2 applied to data generated using G_0: one knot.

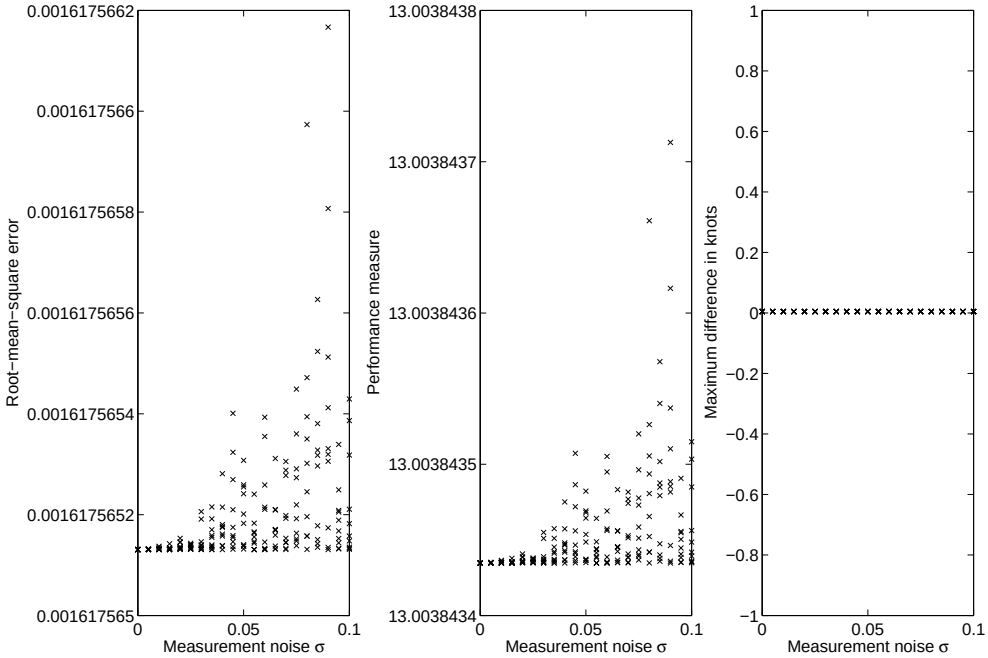

Figure 19: S_3 applied to data generated using G_0: one knot.

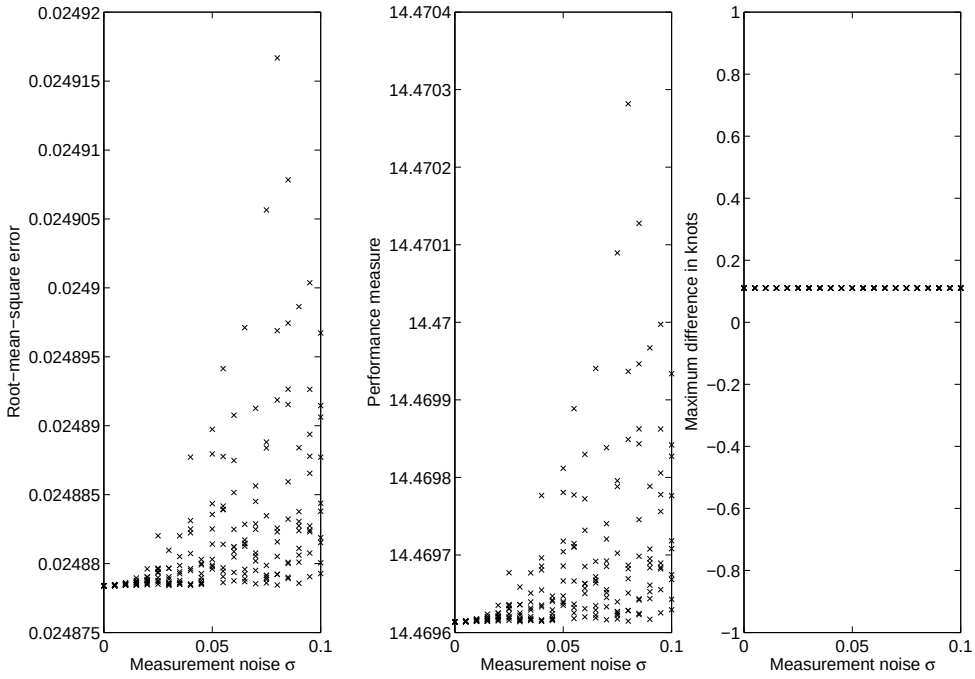

Figure 20: S_1 applied to data generated using G_0: three knots.

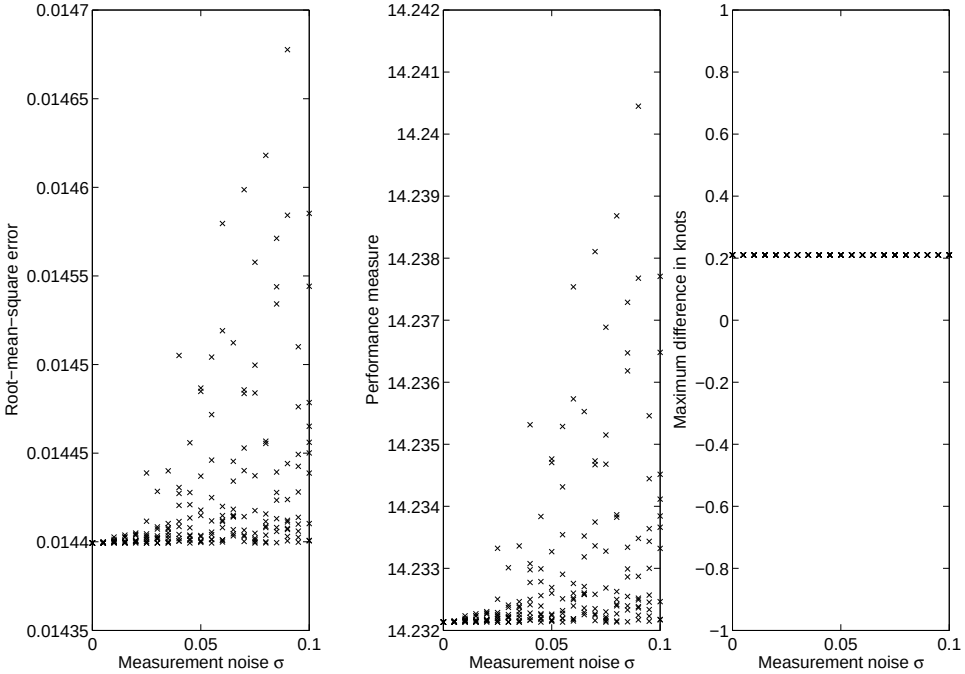

Figure 21: S_2 applied to data generated using G_0: three knots.

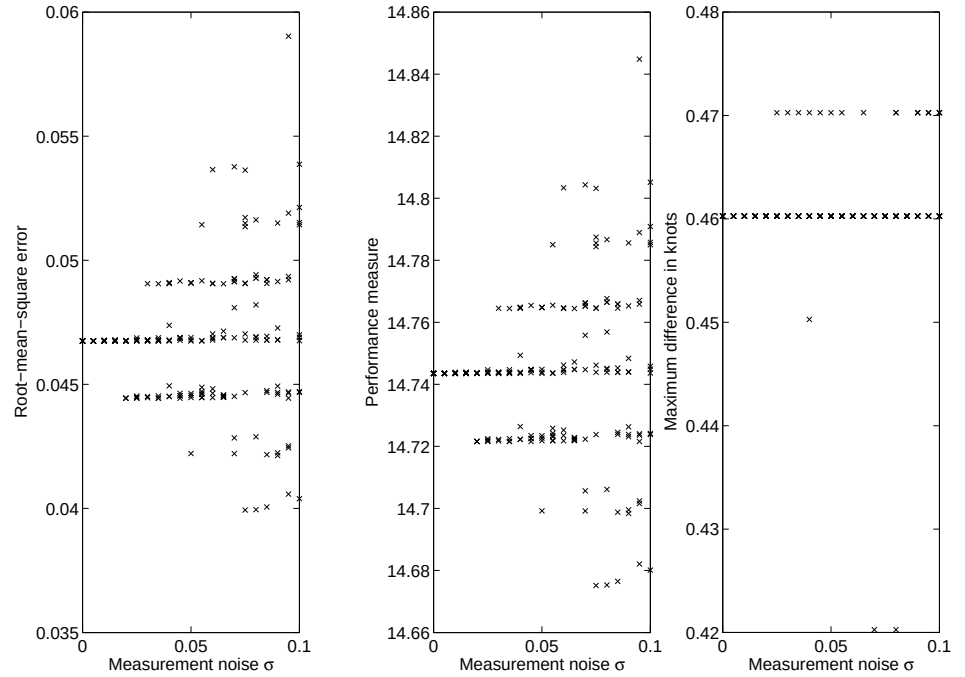

Figure 22: S_3 applied to data generated using G_0: three knots.

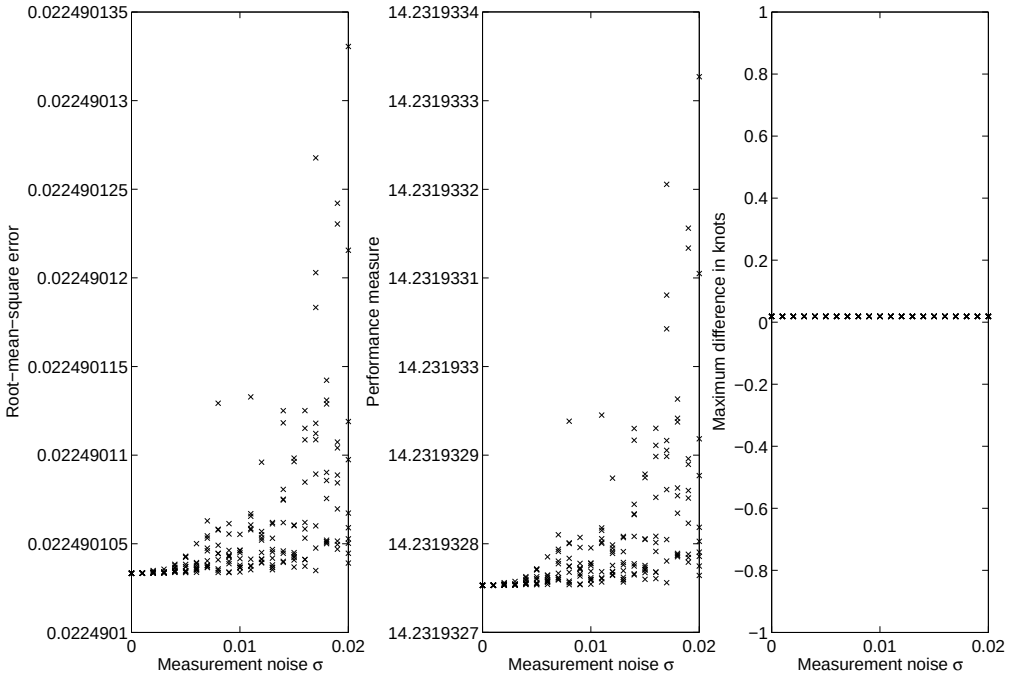

Figure 23: S_1 applied to data generated using G_0: nine knots.

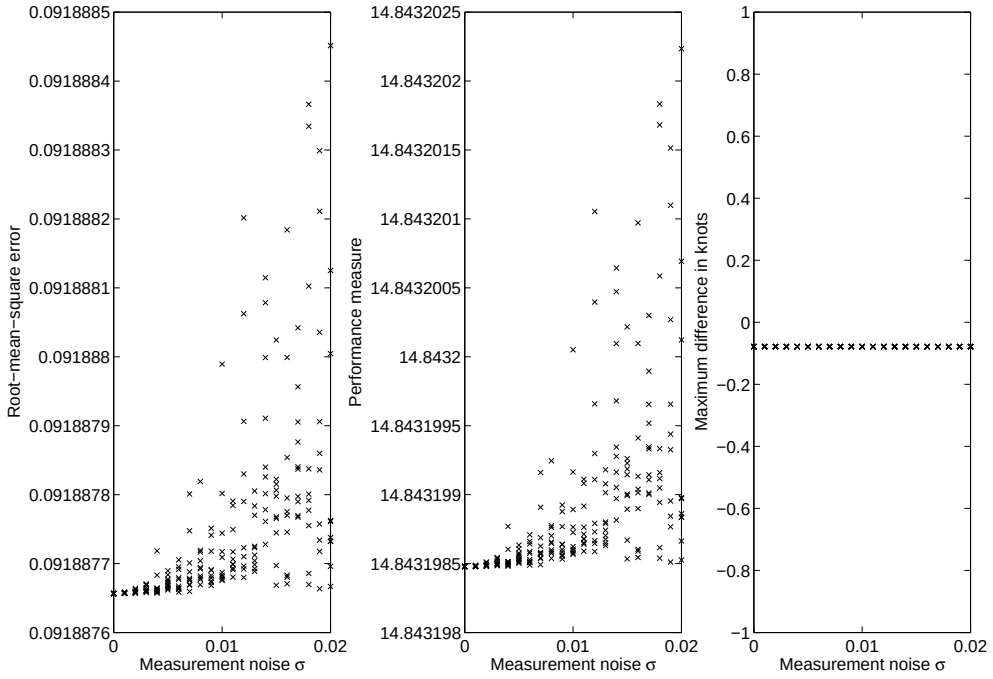

Figure 24: S_2 applied to data generated using G_0: nine knots.

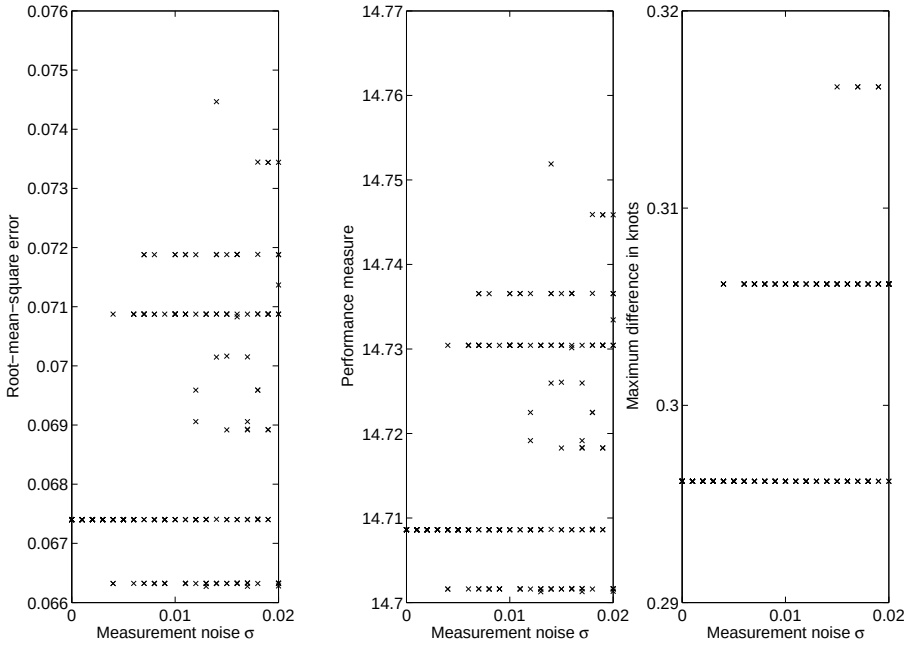

Figure 25: S_3 applied to data generated using G_0: nine knots.

D Results from using data generator software for the computational aims

This appendix contains the results (section 8.2) of the *algorithm testing* obtained using data generator software for the computational aims. The following figures are included:

Figure 26 Comparing data generated using data generators G_0 and G_1: one knot.

Figure 27 Comparing data generated using data generators G_0 and G_2: one knot.

Figure 28 Comparing data generated using data generators G_0 and G_3: one knot.

Figure 29 Comparing data generated using data generators G_0 and G_1: three knots.

Figure 30 Comparing data generated using data generators G_0 and G_2: three knots.

Figure 31 Comparing data generated using data generators G_0 and G_3: three knots.

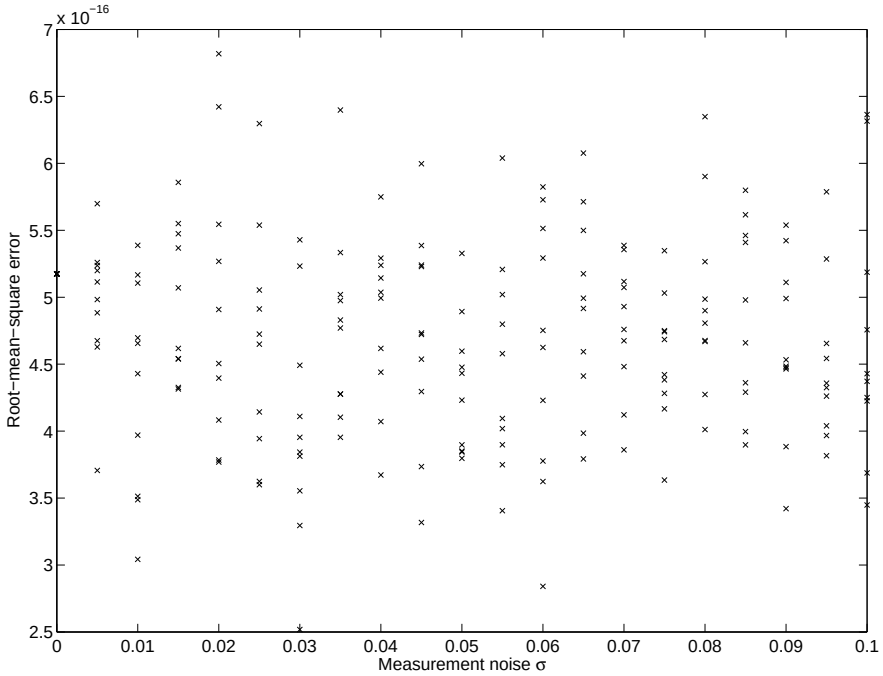

Figure 26: Comparing data generated using G_0 and G_1: one knot.

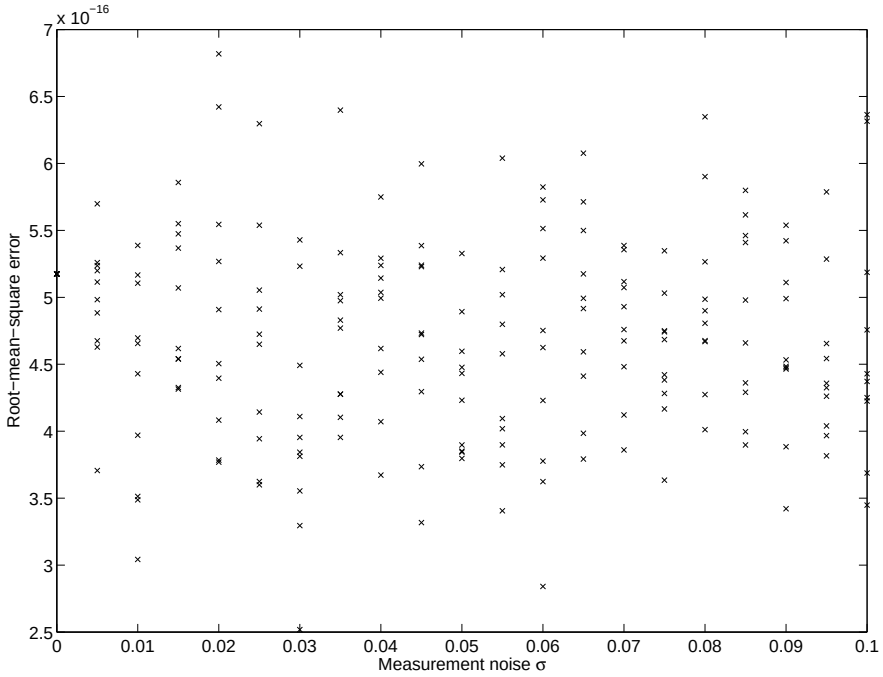

Figure 27: Comparing data generated using G_0 and G_2: one knot.

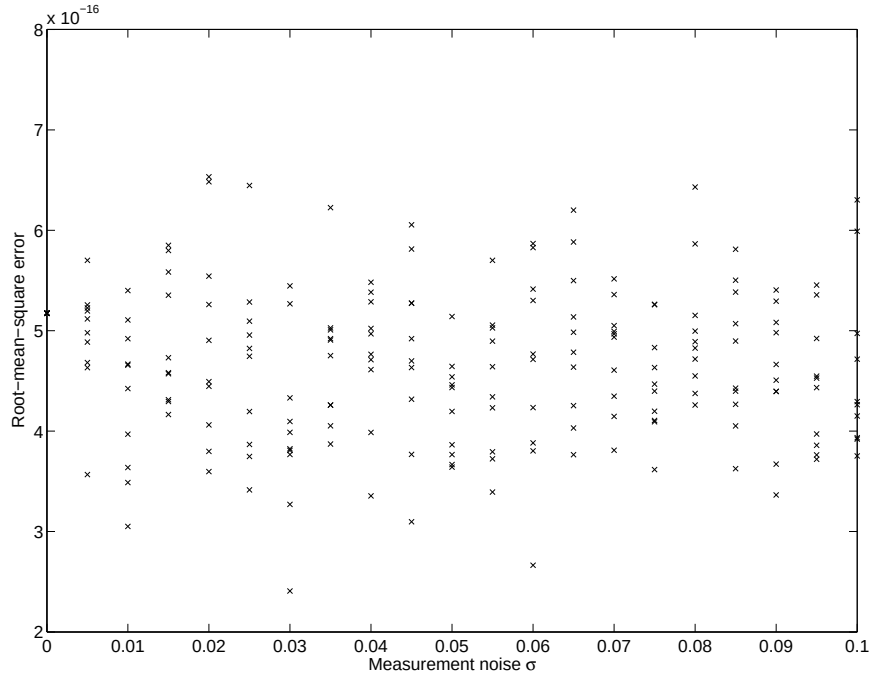

Figure 28: Comparing data generated using G_0 and G_3: one knot.

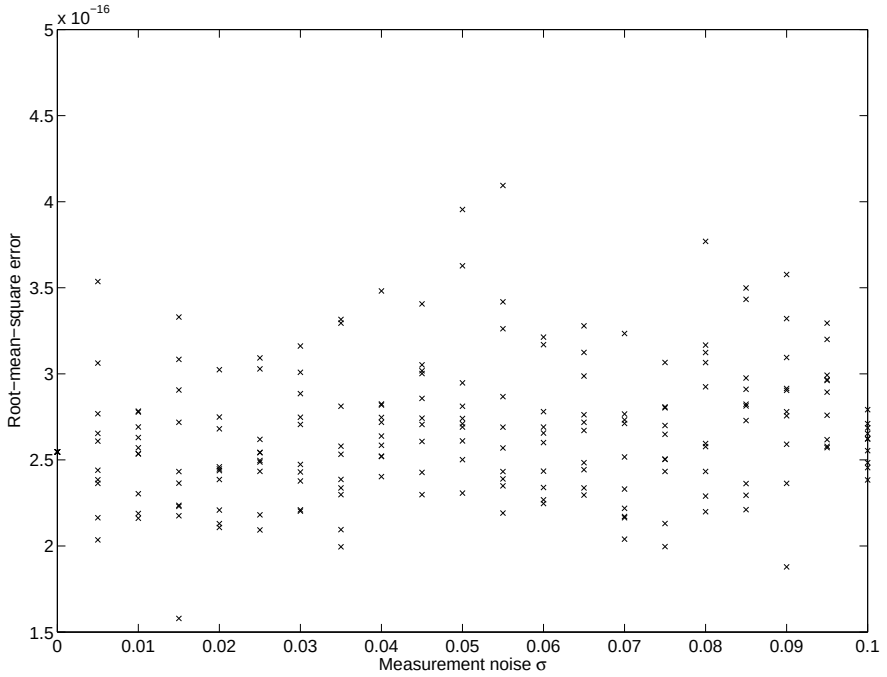

Figure 29: Comparing data generated using G_0 and G_1: three knots.

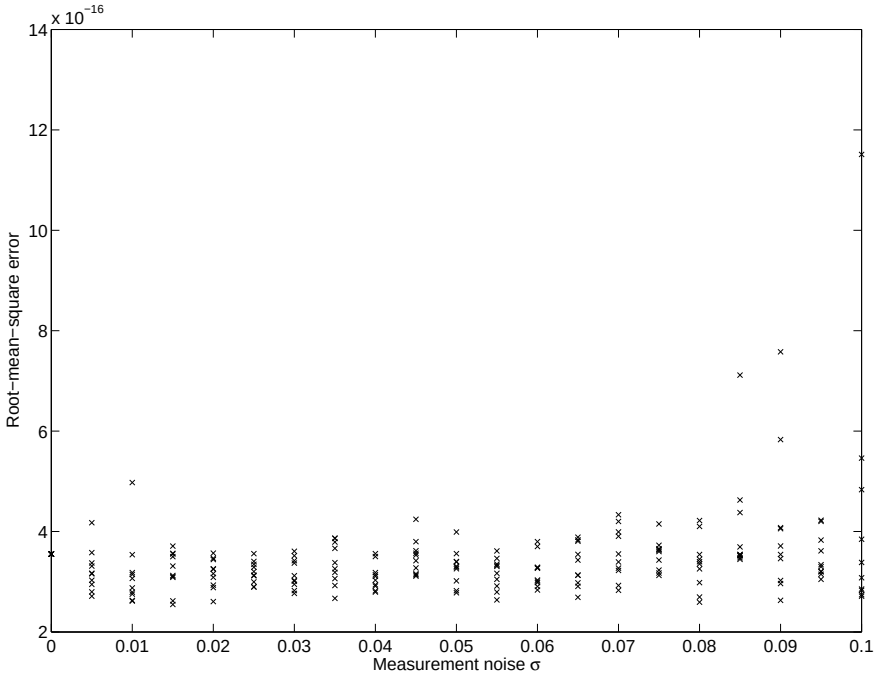

Figure 30: Comparing data generated using G_0 and G_2: three knots.

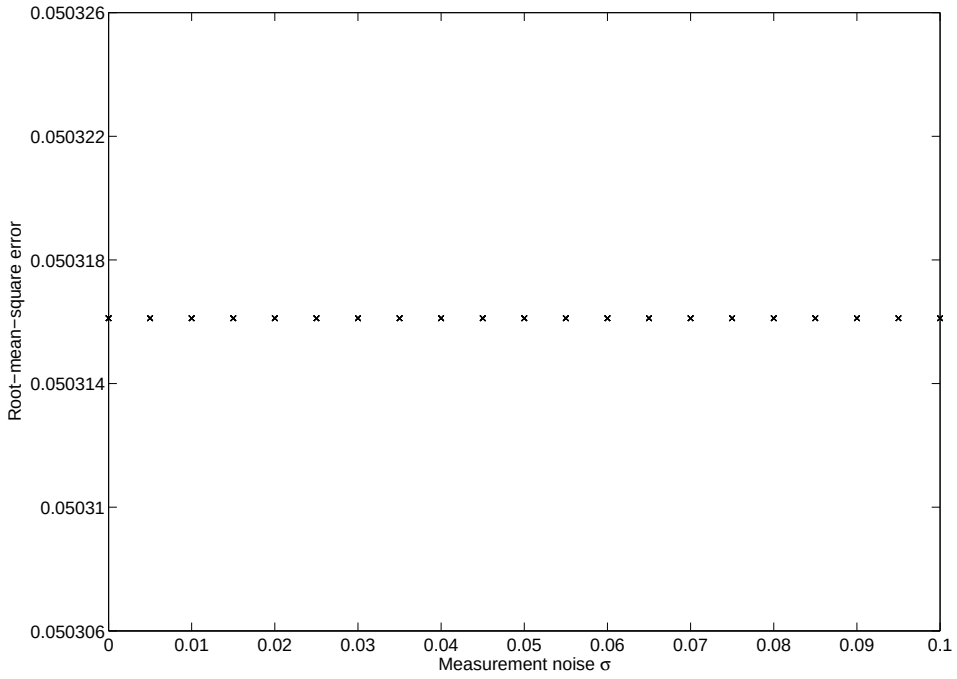

Figure 31: Comparing data generated using G_0 and G_3: three knots.

E Results from simulation

This appendix contains the results (section 8.3) of the *algorithm testing* obtained using simulation. The following figures are included:

Figure 32 Approximate PDFs for the values of the (three) knot positions of the fitted spline curve, and the value at $x = 0$ of the curve, returned by software S_0. Data is simulated with $\sigma = 0.01$.

Figure 33 Approximate PDFs for the values of the (three) knot positions of the fitted spline curve, and the value at $x = 0$ of the curve, returned by software S_1. Data is simulated with $\sigma = 0.01$.

Figure 34 Approximate PDFs for the values of the (three) knot positions of the fitted spline curve, and the value at $x = 0$ of the curve, returned by software S_2. Data is simulated with $\sigma = 0.01$.

Figure 35 Approximate PDFs for the values of the (three) knot positions of the fitted spline curve, and the value at $x = 0$ of the curve, returned by software S_3. Data is simulated with $\sigma = 0.01$.

Figure 36 Approximate PDFs for the values of the (three) knot positions of the fitted spline curve, and the value at $x = 0$ of the curve, returned by software S_0. Data is simulated with $\sigma = 0.05$.

Figure 37 Approximate PDFs for the values of the (three) knot positions of the fitted spline curve, and the value at $x = 0$ of the curve, returned by software S_1. Data is simulated with $\sigma = 0.05$.

Figure 38 Approximate PDFs for the values of the (three) knot positions of the fitted spline curve, and the value at $x = 0$ of the curve, returned by software S_2. Data is simulated with $\sigma = 0.05$.

Figure 39 Approximate PDFs for the values of the (three) knot positions of the fitted spline curve, and the value at $x = 0$ of the curve, returned by software S_3. Data is simulated with $\sigma = 0.05$.

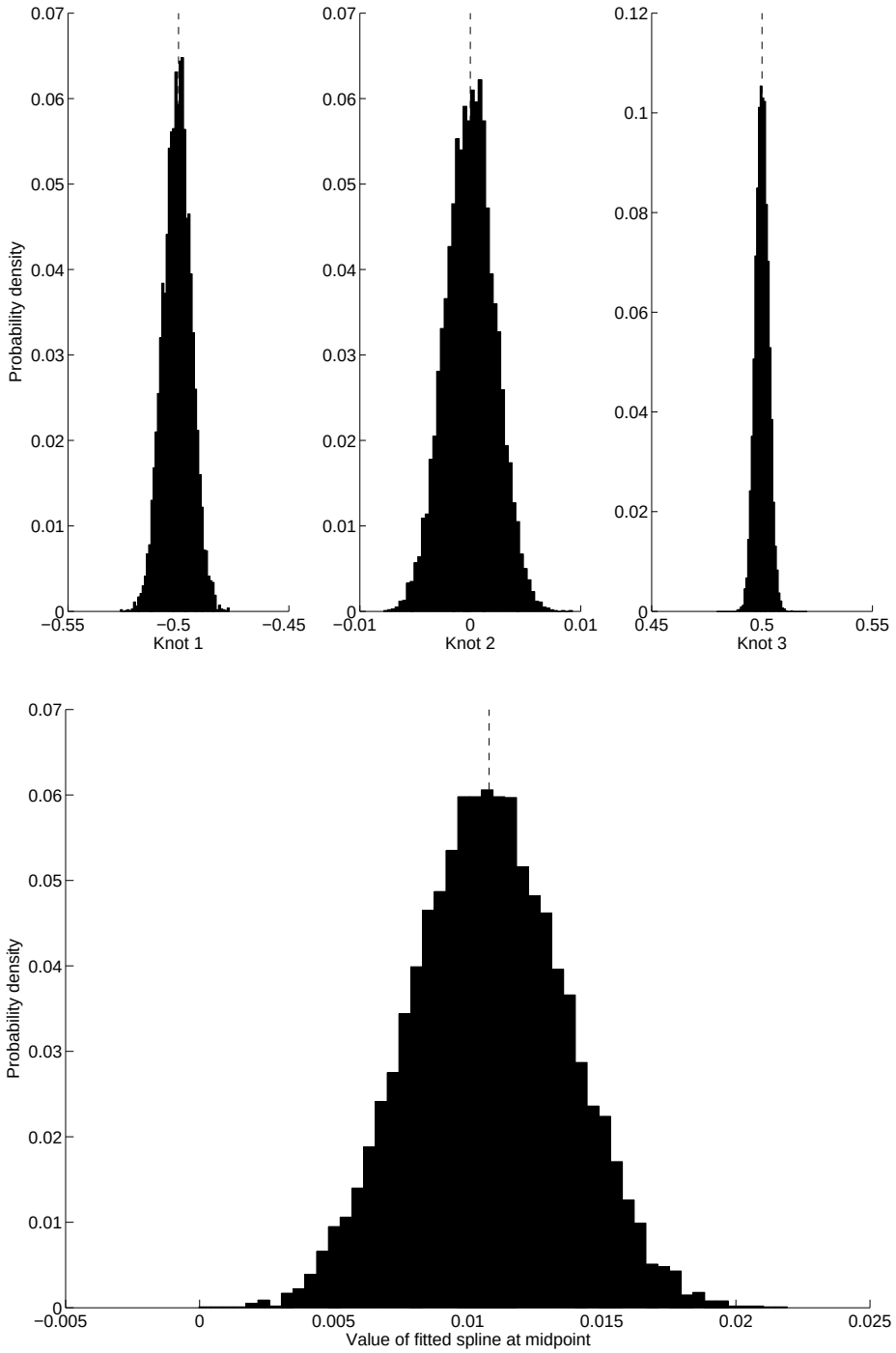

Figure 32: Approximate PDFs for the values of the (three) knot positions (above) and the value at $x = 0$ of the fitted curve (below) returned by simulation and software S_0. Data is simulated with $\sigma = 0.01$. The knots of the spline underlying the simulation, and the value at $x = 0$ of the spline, are shown as vertical (dashed) lines.

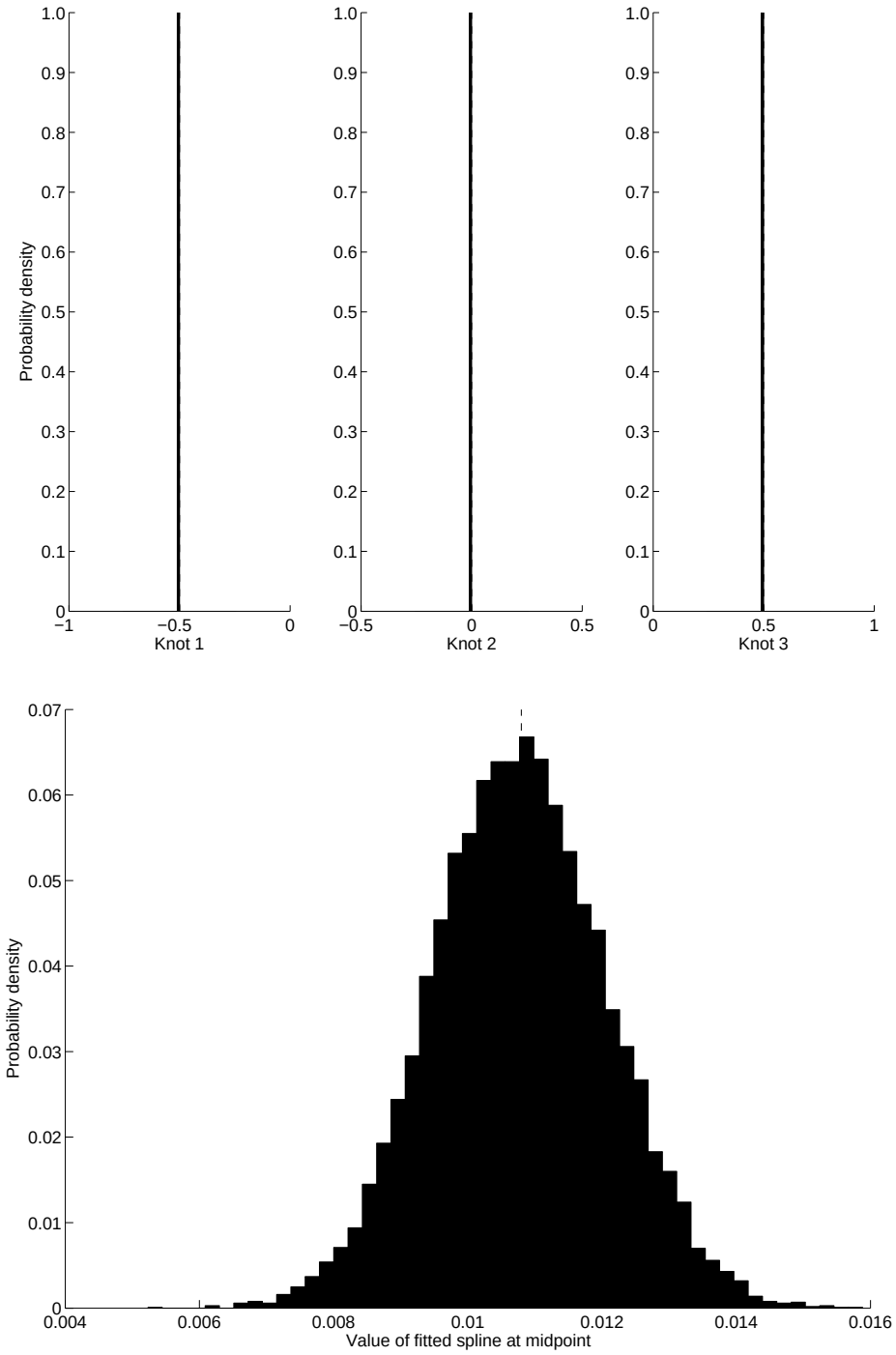

Figure 33: As figure 32, but using software S_1.

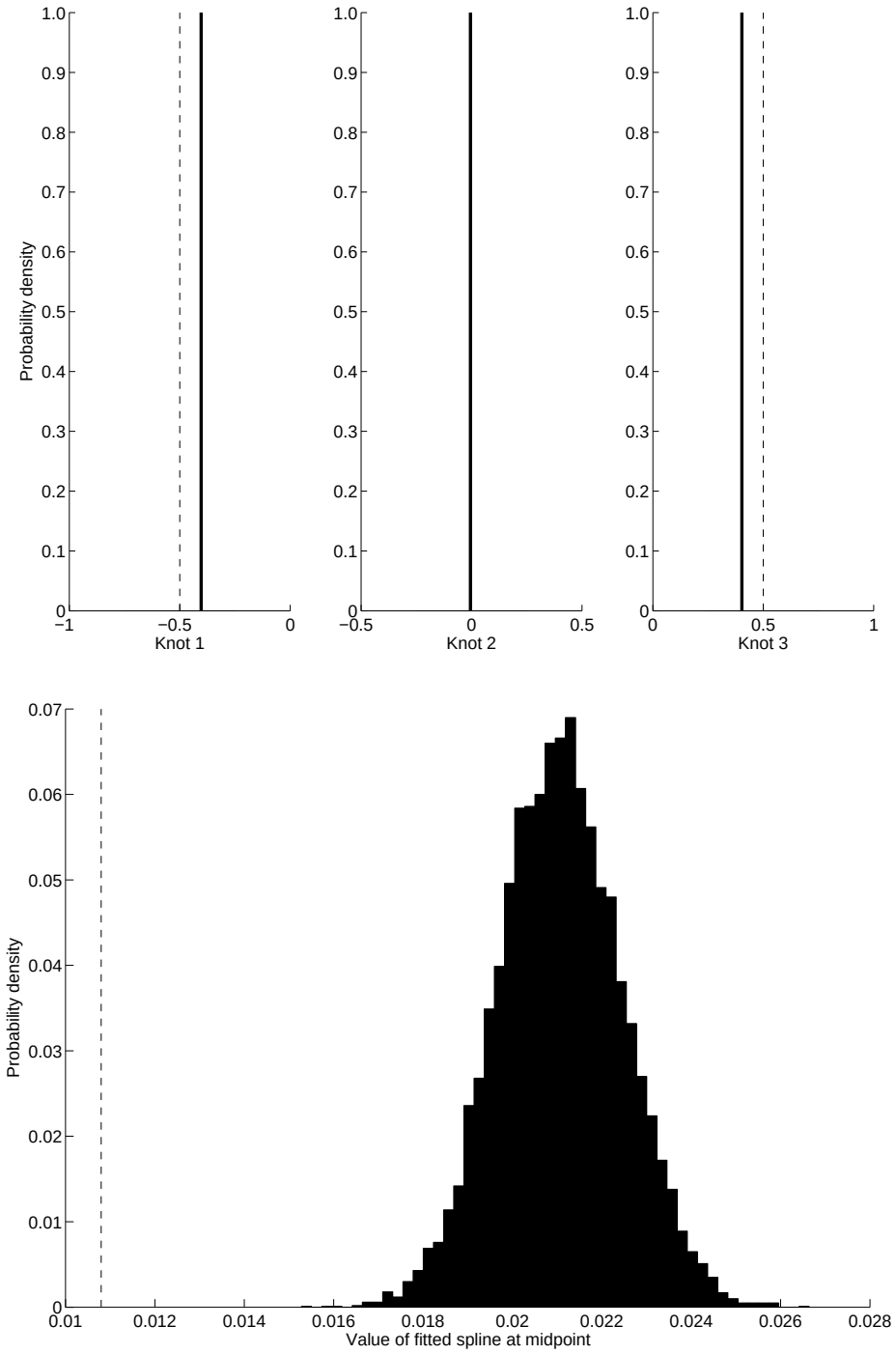

Figure 34: As figure 32, but using software S_2.

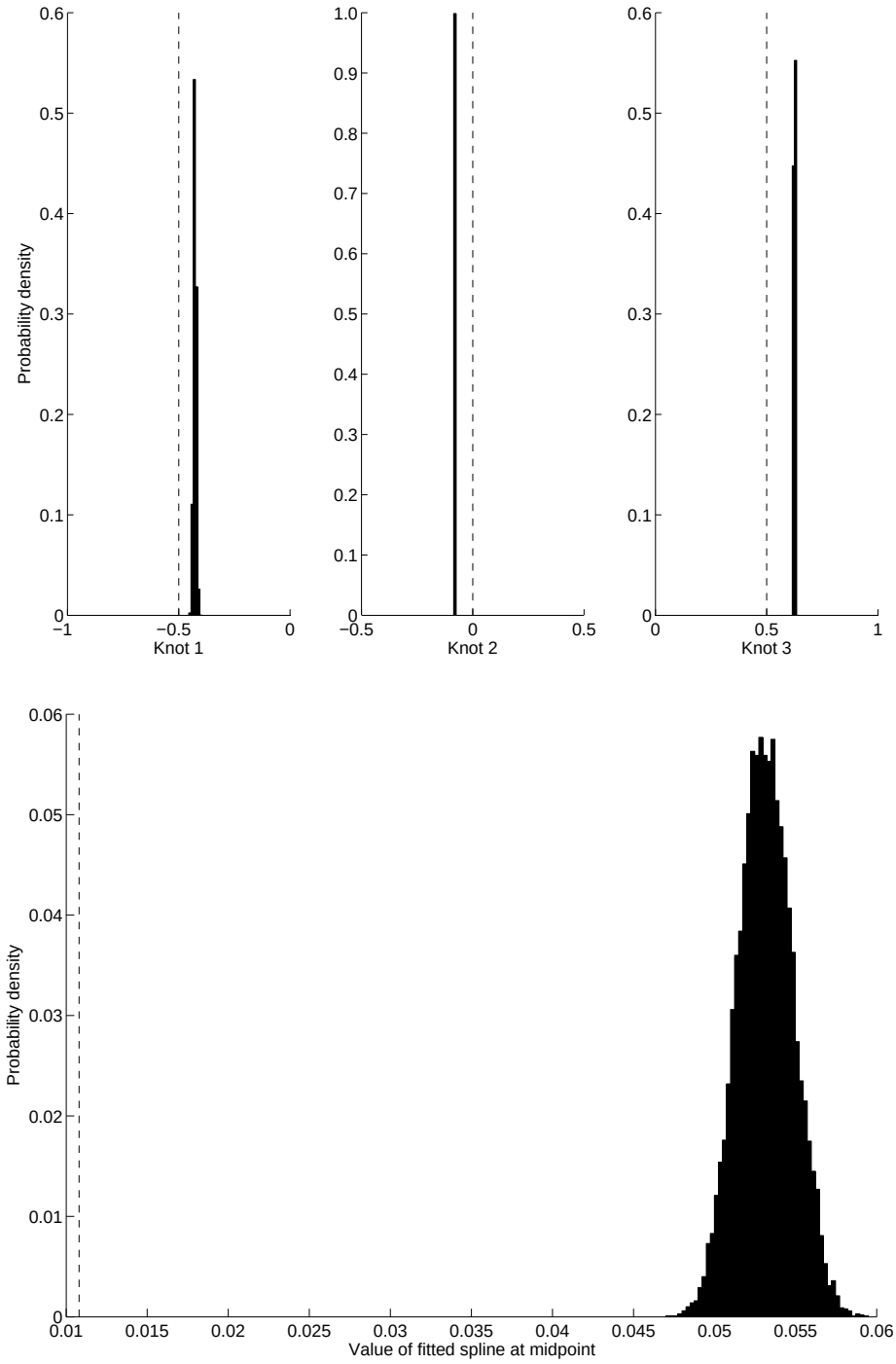

Figure 35: As figure 32, but using software S_3.

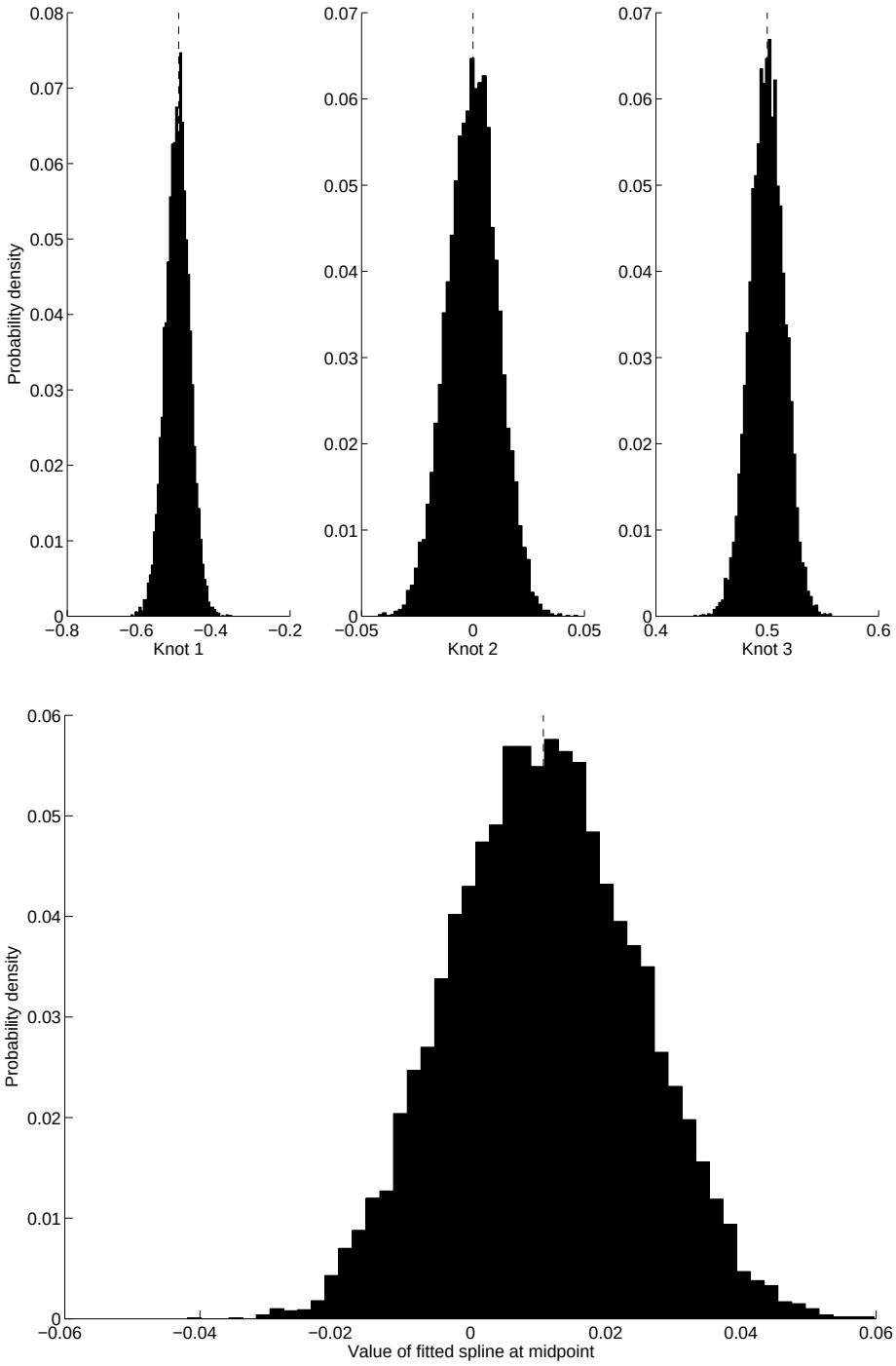

Figure 36: As figure 32, but for data simulated with $\sigma = 0.05$.

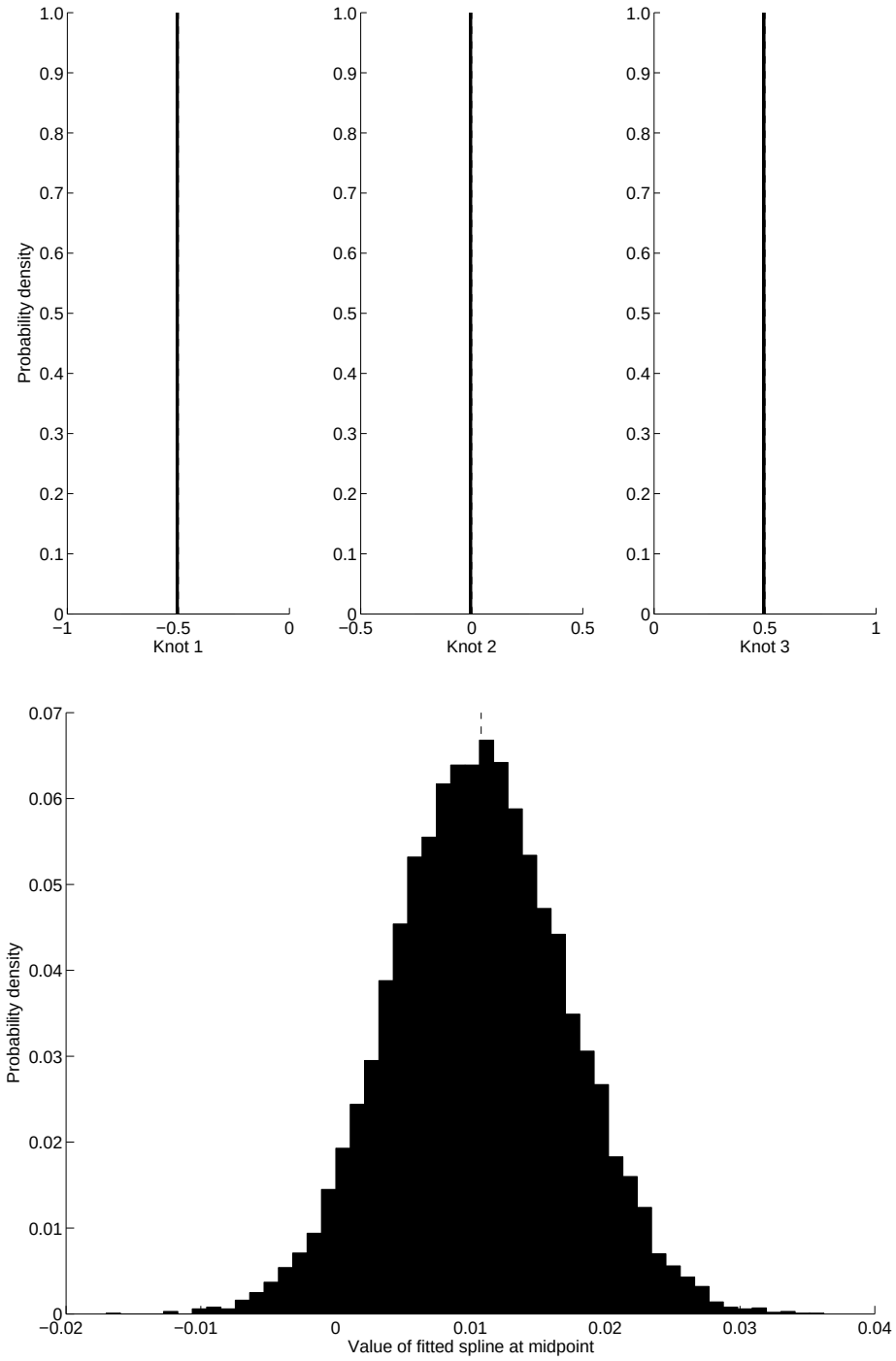

Figure 37: As figure 36, but using software S_1.

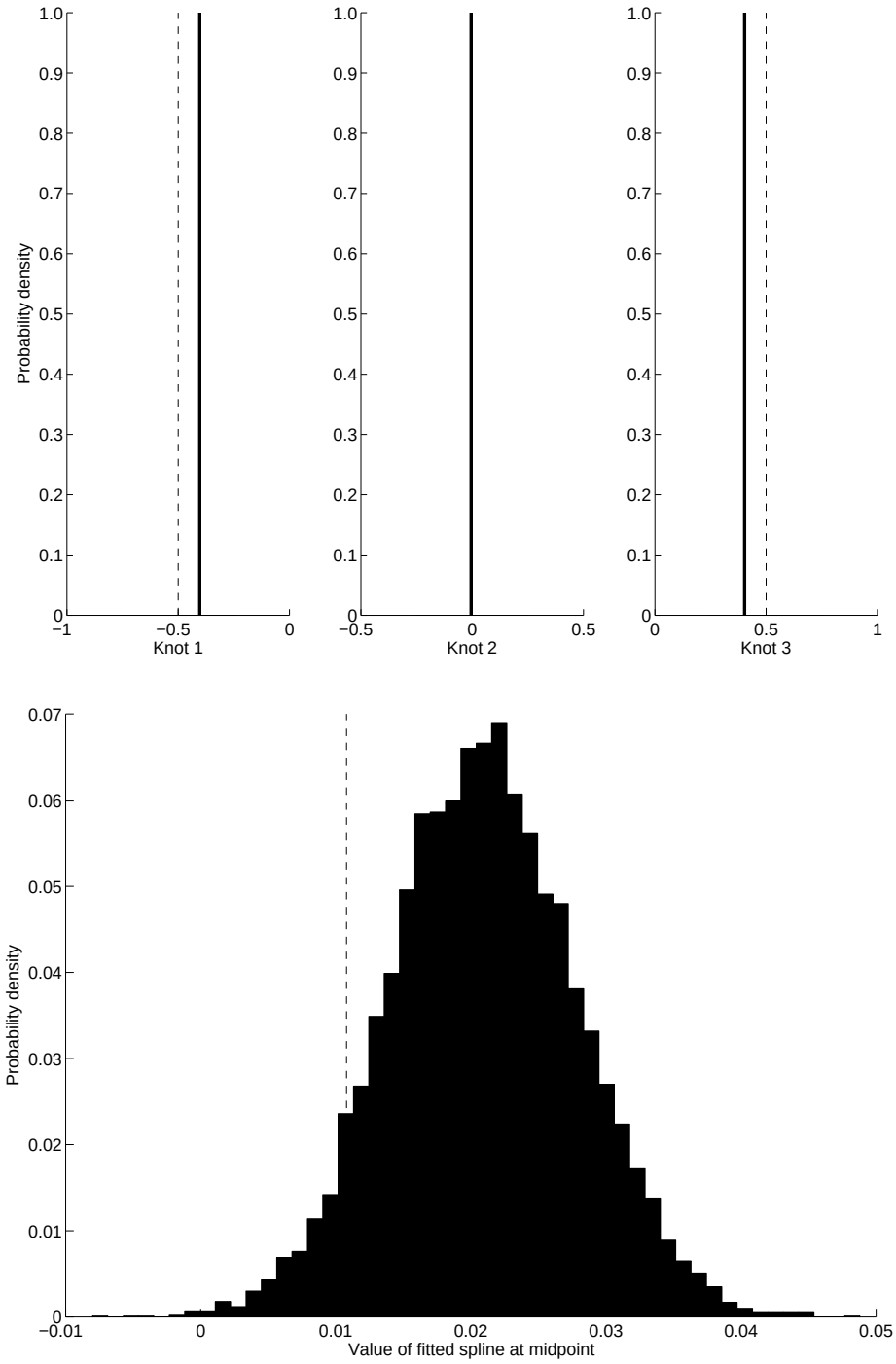

Figure 38: As figure 36, but using software S_2.

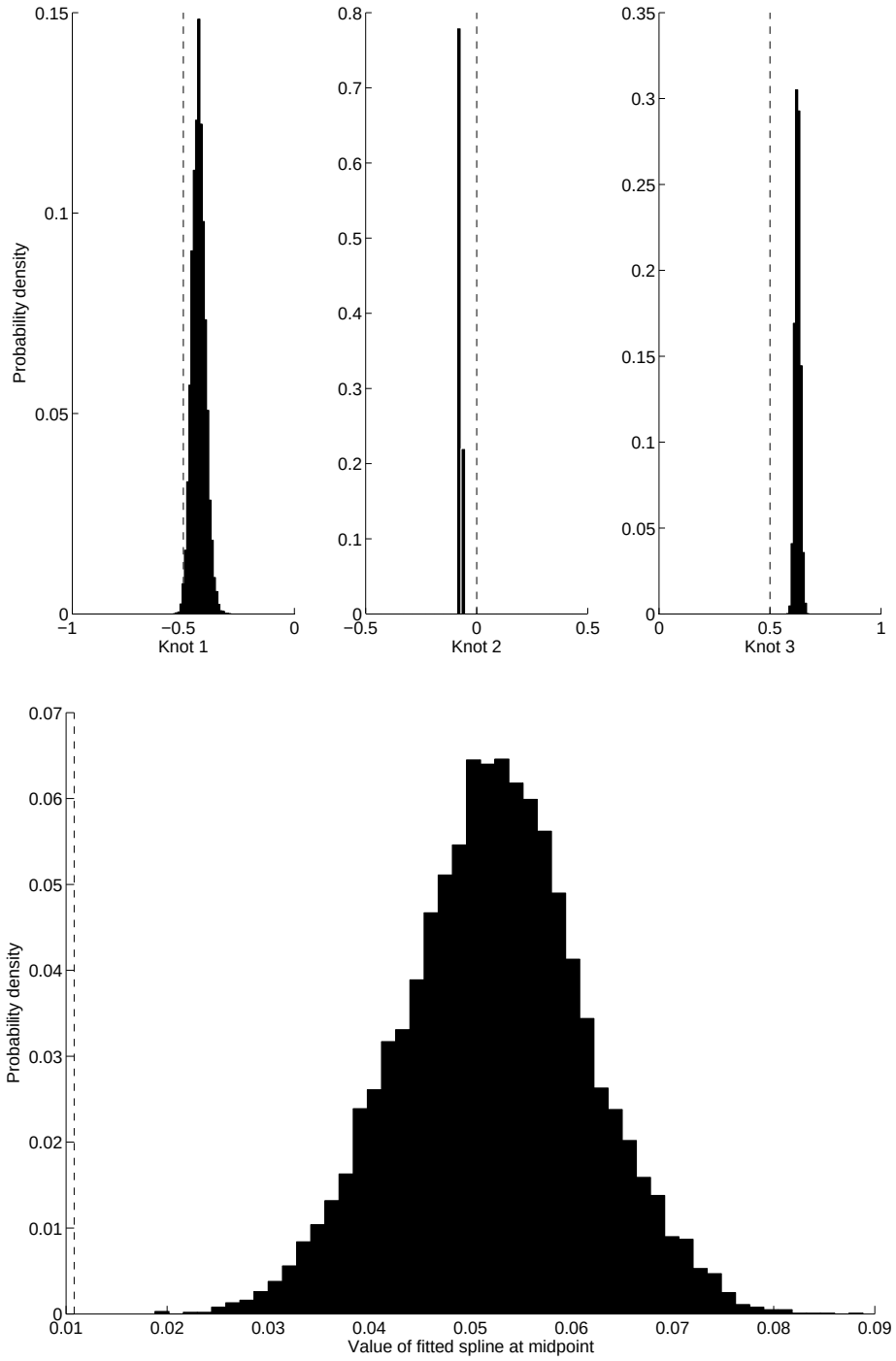

Figure 39: As figure 36, but using software S_3.