Software Support for Metrology
Best Practice Guide No. 11

Numerical analysis for algorithm design in metrology

M G Cox and P M Harris

April 2004

Software Support for Metrology Best Practice Guide No. 11

Numerical analysis for algorithm design in metrology

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

April 2004

ABSTRACT

When a metrology problem is formulated for computer solution, it is necessary to use, or design and use, a suitable solution algorithm. This best-practice guide is concerned with the numerical analysis of such algorithms. In particular, guidance is provided on avoiding the unnecessary and damaging loss of numerical precision that accompanies some computations.

Much of the advice is provided through the presentation of case studies. Such studies are in the areas of hardness measurement, interlaboratory comparisons, extraction of mass of material, DNA measurement and climate change. Each starts with an indication of the numerical aspects illustrated by the study and concludes with the lessons learnt.

The companion Software Support for Metrology Best-Practice Guide No. 4, Discrete Modelling and Experimental Data Analysis, contains valuable material concerned with linear and non-linear approximation, especially the use of least-squares methods, relevant model parametrization and the supporting linear algebra. In addition, for these areas it provides information on numerical methods and statements concerning numerical stability, etc. The current best-practice guide therefore concentrates on other aspects of numerical analysis that are also relevant to metrology, viz., formula evaluation, differentiation and integration. Some aspects that overlap with the scope of the companion guide are indicated. Although the current guide does not deal explicitly with large-scale computations such as partial differential equations or non-linear optimization with many variables, some of the messages given are also relevant to these problems.

ISSN 1471–4124

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

Chapter 1

Introduction and objectives

> I have little doubt that about 80 per cent of all the results printed from the computer are in error to a much greater extent than the user would believe ... [40].

So said Leslie Fox, an eminent UK numerical analyst, in a 1971 paper entitled 'How to get meaningless answers in scientific computation (and what to do about it)'. All the messages of that paper, which constitutes highly recommended reading, remain relevant today. This best-practice guide is concerned with obtaining reliable numerical solutions to scientific problems, and discusses in the context of metrology some of the issues considered by Fox.

Numerical analysis is defined here as 'the design and analysis of mathematical algorithms and their implementation on a computer'.

There are many extensive treatises on numerical analysis. Here parts of numerical analysis are covered that in the authors' experience are important in solving classes of problems in metrology. However, mainly areas are covered that in some sense are fundamental and small-scale, because to do otherwise would have meant the production of a much larger document. The areas addressed are formula evaluation, differentiation and integration. Thus, there is no coverage here, for example, of topics such as large-scale modelling and optimization problems, partial differential equations and integral equations. Certain fundamental numerical aspects of model fitting and function minimization are, however, addressed. It is believed that some of the messages relating to the areas covered can be interpreted by the reader in the context of these larger problems. Moreover, topics that are well-covered by the companion Software Support for Metrology Best-Practice Guide No. 4, Discrete Modelling and Experimental Data Analysis [2], are generally not addressed here, although there is inevitably some overlap. That guide contains valuable material concerned with linear and non-linear approximation, especially the use of least-squares methods, relevant model parametrization and the supporting linear algebra. In addition, for these areas it provides information on numerical methods and statements concerning numerical stability, etc.

1.1 Why should the metrologist be concerned with numerical analysis?

The intended audience of this guide consists of metrologists and others involved in metrology who need to carry out numerical and statistical calculations in order to provide all types of measurement results. By its nature, much of the material will also be relevant to other scientific disciplines. However, the examples and case studies relate to metrology.

Why should the metrologist be concerned with numerical analysis? Surely, (a) most of the fundamental aspects of numerical analysis are understood and (b) reliable mathematical software libraries exist to assist? The answer is 'yes' to both questions. There are problems, however. Regarding (a), although these aspects are indeed understood by numerical analysis practitioners, not all metrologists may feel sufficiently confident to design algorithmic solutions for even simple problems. Unfortunately, in solving such problems, aspects frequently arise that endanger the provision of reliable answers. Further, a piece of software that has been well used on modest problems may not deliver reliable solutions for larger problems or for somewhat different values for one or more of its input parameters or data items. Regarding (b), inadequate attention to the input quantities to a library routine or to the way library routines are combined to provide an overall solution can again cause problems with furnishing acceptable results. Most importantly, attention to some numerical analysis principles can save considerable time and effort later. The authors have seen many instances relating to this point. The overriding influence is the fact that computers operate to a (relatively) limited numerical precision that can distort the results of a computation at the practical level, unless the computation is carried out with care.

The numerical precision (of the order of 16 decimal digits) to which most floating-point chips operate can be inadequate when a poor algorithm is used. In contrast, this precision is such that the results obtained using a good-quality algorithm will almost always be fit for purpose, even for the most demanding metrology applications (under the sometimes strong assumption that the input data is valid for the problem). In some cases the situation can be rescued by using extended-precision arithmetic. Doing so should not be regarded as a general remedy. It may perhaps extend the set of problems that can be solved with a certain algorithm, but does not address the fundamental limitations of the algorithm.

The reader is invited to take a forward look at the case study in section 3.2 concerned with measurement of the hardness of a material using an indenter. There, the formula (3.5), viz.,

$$B = \frac{0.204F}{\pi D(D - \sqrt{D^2 - d^2})}$$

is given for the Brinell hardness B in N/mm^2 as a function of the test force F in N, the diameter D in mm of the indenter and the mean diameter d in mm of the indentation. For some values of the quantities on the right-hand side of the formula, considerable numerical accuracy can be lost in forming B. On the

other hand, the use of the *mathematically equivalent* formula

$$B = \frac{0.204F(D + \sqrt{D^2 - d^2})}{\pi D d^2}$$

behaves numerically very differently, not suffering accuracy problems.

As another instance [43], to give the flavour of 'what can go wrong', consider the evaluation of the integral

$$y_r = \frac{1}{e}\int_0^1 x^r e^x dx, \tag{1.1}$$

where r is a positive whole number. Integrals of such type arise when determining the moments of probability density functions, of importance in uncertainty evaluation [7]. y_r satisfies the recurrence relation[1]

$$y_0 = 1 - 1/e, \qquad y_r = 1 - r y_{r-1}, \qquad r = 1, 2, \ldots \tag{1.2}$$

the use of which gives the values in the second row of table 1.1.

r	0	1	2	$\cdots$	17	18	19	20	21
y_r^{fwd}	0.632	0.368	0.264	$\cdots$	0.057	−0.030	1.560	−30.192	635.040
y_r^{bck}	0.632	0.368	0.264	$\cdots$	0.053	0.050	0.048	0.046	0.044

Table 1.1: The use of recurrence relations (1.2) and (1.3) to evaluate the integral (1.1).

The early values in the table are correct numerically, at least to the number of digits quoted, but matters go badly wrong subsequently. A simple analysis can be used to understand the reason for this behaviour. Since $y_0 = 1 - 1/e$ cannot be held exactly on the computer, because e is an irrational number (cf. section 2.1.1), the recurrence is started with a slightly perturbed value $\widetilde{y}_0$, say. Let $\widetilde{y}_r$ denote the computed value of y_r and

$$\delta y_r = \widetilde{y}_r - y_r.$$

Then, the values $\widetilde{y}_r$ delivered by the recurrence relation (1.2) satisfy

$$y_r + \delta y_r = 1 - r(y_{r-1} + \delta y_{r-1}),$$

from which

$$\delta y_r = -r\delta y_{r-1}.$$

Letting

$$\widetilde{y}_0 = y_0 + \epsilon$$

to start the recurrence gives

$$\delta y_0 = \epsilon$$

[1] The recurrence is derived by integrating by parts the expression for y_r in formula (1.1):

$$y_r = \frac{1}{e}\left([x^r e^x]_0^1 - \int_0^1 r x^{r-1} e^x dx\right) = \frac{1}{e}(e - r e y_{r-1}) = 1 - r y_{r-1}.$$

and hence

$$\delta y_1 = -\epsilon, \qquad \delta y_2 = 2\epsilon, \qquad \delta y_3 = -6\epsilon, \qquad \delta y_4 = 24\epsilon, \qquad \text{etc.},$$

i.e.,

$$\delta y_r = (-)^r r!\epsilon.$$

When $r = 18$, $r!$ is of the order of 10^{16}. Thus, if y_0 is held on the computer to an accuracy of approximately 10^{-16} (which would be the case for a 16-decimal computer—section 2.6), all accuracy has been lost in the value of y_{18} and hence in subsequent values of y_r.

Because of the exceedingly rapid rise in the magnitude of the error, the use of extended precision would only postpone the onset of instability.

The recurrence relation amplifies errors, as a consequence of the multiplication by r when forming y_r from y_{r-1}. However, if y_{r-1} is formed from y_r using the simple rearrangement

$$y_{r-1} = (1 - y_r)/r \tag{1.3}$$

of the recurrence (1.2), involving a *division* by r, it can be expected that the errors will be attenuated.

Suppose accurate values of $y_0, \ldots, y_{20}$ are required. Start with the (very poor) approximation $y_{30} = 0$. The use of the *backward recurrence relation* (1.3) gives the values in the third row of table 1.1, which are all correct to the digits shown (and many more beyond). The reason why such a poor starting approximation is adequate is due to the extreme error damping effect. By starting at a suitable point (or beyond), *any* initial approximation can be used.

Various reports on activity in the UK Software Support for Metrology programme (SS*f*M) that relate to the quality of numerical solutions to metrology problems are available. They cover areas such as geometric form assessment, data fusion, experimental data analysis, calibration and regression [1, 2, 12, 26, 27], and many more. The more recent reports are obtainable from the SS*f*M website.[2]

1.2 Objectives of this guide

There are three main stages in developing numerical software for use in metrology [3]:

1. Draw up a specification of the problem to be solved
2. Design an algorithm to solve the problem defined by the specification
3. Implement the algorithm as software.[3]

The concern of this best-practice guide is predominantly the second stage, one that has too often been ignored or where the difficulties have been underestimated, judging by some of the products used by metrologists worldwide, a

[2] http://www.npl.co.uk/ssfm/download/nplreports.html

[3] The term 'algorithm' is used to denote a prescription or recipe relating to a particular task to produce some results (output data) given some input data. An algorithm often constitutes an English-language step-by-step procedure that typically includes mathematics. The term 'software' is used to denote a computer implementation of an algorithm. The user provides the input data and expects to receive the corresponding results for the task.

number of which have been tested in the SS*f*M programme [4, 17, 24, 25, 33]. It is an area where attention to detail can make the difference between obtaining valid and invalid results.

An issue is that defects exposed by software testing may be wrongly associated with the software implementation when in fact the implementation of the specified algorithm is valid: the defects are inherent in the underlying algorithm.

There are three major considerations concerning algorithm design: accuracy, robustness, and efficiency.

Accuracy relates to how well a problem is solved according to some appropriate measure by (a sound software implementation of) an algorithm. An appropriate stance to take in designing an algorithm is that the solution provided should be at least adequate or provide relevant information if it encountered difficulty. An algorithm can be near perfect for relatively innocuous input data, but degrade seriously when it is 'stretched', e.g., for 'real-life' data. An example is the evaluation [29],[44, p12] of the standard deviation s of a sample $x_1, \ldots, x_N$, viz.,

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N} x_i^2 - N\bar{x}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2, \qquad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{1.4}$$

The Microsoft Excel standard deviation function, for instance, has been tested [25] and demonstrated to work satisfactorily for some data sets and not for others. It would appear to use the left-most expression for s.[4] The instances where it fails to deliver an accurate solution are for high-accuracy metrology data. It has the unhelpful property that the more accurate the data, measured by the standard deviation, the less accurate is the result (in terms of its relative error). The second of the two formulae is stable [25],[44, p12].

The issue of determining the influence of *perturbations in the data* on the solution provided is another important issue. That consideration comes under the banner of sensitivity or perturbation analysis. It has much in common with the law of propagation of uncertainty in uncertainty evaluation, which makes use of *sensitivity coefficients* formed from partial derivatives to scale the standard uncertainties associated with the input quantities to a model [7, 32]. See section 2.1.

Robustness relates to the ability of the algorithm to operate correctly within and near the boundary of its domain of application. An ideal algorithm would not provide an erroneous solution. For instance, as part of an uncertainty evaluation, it might be required to form the triangular factor R in the Cholesky factorization $V = R^{\mathrm{T}}R$ [44, chapter 10] of an uncertainty matrix (covariance matrix) V. If V is not positive definite, the calculation cannot proceed, but it might be possible to *repair* the matrix [56, p322] and factorize the repaired form. Perhaps V failed to be positive definite because of rounding errors in forming its elements from other quantities. The user could benefit from being informed of this possibility. The 'repair' process would consist of making 'smallest possible' changes to V such that it becomes positive definite. If the magnitudes of these changes are consistent with the rounding errors incurred in forming V, the repair process can be considered reasonable.

[4] This conclusion was drawn on the basis that when that formula was implemented it gave exactly the same results as those from Excel on the data sets used.

Efficiency relates to the amount of computation (and sometimes the memory requirements) needed to deliver the solution (sometimes to a stipulated numerical accuracy). With GHz-speed PCs on metrologists' desks, this might not be seen as a major problem. However, metrologists are constructing and analyzing ever more sophisticated models of sensors, electric fields, and so forth, and waiting an excessive time for a solution to be returned inhibits *human* efficiency.

Also, accuracy and efficiency are intimately linked in the many iterative and recursive processes used in metrology computation. Algorithms for such processes use tests for convergence of a numerical algorithm; at one extreme an inadequate convergence criterion can prevent the delivered result from meeting the stipulated accuracy, and at the other unnecessarily long computation times will result.

This guide is concerned with the elaboration of algorithm-design principles that are capable of application to a range of metrology problems. They are applied, through examples and case studies, to several areas to illustrate the main issues. Some of the considerations relate to

- The use of sensible scaling, normalization and other data transformations
- The avoidance of loss of information, e.g., deleterious arithmetic operations such as damaging subtractive cancellation, and adding a small number to a large number
- The advantages of stable problem parametrization and formulae.

These considerations arise in many of the problems encountered by metrologists, including calibration curve fitting, the solution of measurement models, uncertainty evaluation, experimental design, instrument-design optimization and chemometrics.

It is always important that an algorithm is seen in the context of being fit for purpose. In many circumstances it will not be necessary that it provides the maximum accuracy possible, but an accuracy that meets a metrological requirement and ideally somewhat beyond that in order to retain a degree of 'numerical safety'. Although this attitude is sound for a well-defined set of problems within one metrology area, an important consideration is that many algorithms have broad applicability in several areas of metrology including those above. The purpose to which these generic algorithms will be put is not known *a priori*; it might range from low-accuracy applications in biometrology to high-accuracy nanotechnology work. As a consequence, such an algorithm should be capable, when implemented, of delivering as much accuracy as possible, unless of course, this objective resulted in unacceptable development effort or excessive computation times. Thus, the attitude taken to algorithm design for metrology is that where sensibly possible the maximum accuracy attainable is provided. If, however, there is a penalty for doing so, for example in terms of a significant increase in processing time, variants should be considered that return a solution that meets a stipulated task-related accuracy. This last-mentioned point relates to the 'natural requirement' that the algorithm (and its software implementation) contribute an effect that is negligible compared to other sources of uncertainty. Otherwise, the effect needs to be quantified.

It is emphasized that even with GHz-speed PCs and the availability of high-precision arithmetic capabilities, these advances in hardware and software are

far from adequately addressing the above accuracy and efficiency considerations. However much accuracy is used, an unstable algorithm will break down or deliver unreliable results for some problems. With regard to efficiency, the gains from exploiting structure are typically a factor of order n or even n^2, where n is the problem size (for example the number of parameters in a model), sometimes of the order or hundreds or thousands [12]. The resulting speed-up by a factor of tens of thousands to millions cannot be matched by improvements in processor speed. These gains apply across all computer hardware. Such speed-ups are necessary to realize the full potential of sophisticated metrology systems such as multilateration co-ordinate measuring machines.

The emphasis of this best-practice guide is more on accuracy and robustness rather than efficiency, the former being the more fundamental consideration.

Acknowledgments

The UK Department of Trade and Industry is responsible for the UK National Measurement System (NMS). Four of the five case studies relate to collaborative work undertaken within various programmes of the NMS. The study concerning interlaboratory comparisons (section 3.3) was carried out with Eulogio Pardo (now at the University of Granada) as part of the Software Support for Metrology programme. The study concerning extraction of mass of material (section 3.4) was undertaken jointly with Steve Ellison and Mike Griffiths (Laboratory of the Government Chemist) as part of the Valid Analytical Measurement programme. The study concerning the determination of DNA concentration (section 4.2) was carried out with Sarantis Kamvissis (NPL) and Michael Burns, Steve Ellison, Mike Griffiths and Jacquie Keer (Laboratory of the Government Chemist) as part of the Biotechnology programme. The study relating to climate change (section 5.7) was carried out jointly with Nigel Fox and Emma Woolliams (NPL) as part of the Optical Radiation Metrology programme.

Sven Hammarling and David Sayers of NAG Ltd. reviewed an advanced draft of this guide, and Sven Hammarling provided material on error analysis.

Chapter 2

Fundamentals

Software is widely used in metrology to assist with the solution of measurement problems formulated in mathematical terms. That software will constitute an implementation of an algorithm. There will be a set of 'inputs' to the algorithm and a set of 'outputs' from it. Although the concern here is with numerical effects, there will be inevitable overlap with mathematical and statistical aspects. The concepts here are discussed in greater detail in recommended texts [43, 44].

2.1 Mathematical, statistical and numerical propagation

For any item of numerical software that implements a mathematical algorithm (in the NAG Library[1], EUROMETROS[2], etc.), three types of 'propagation of effects' can be considered when proceeding from generic input quantities X to the corresponding output quantities Y, formally specified as $Y = f(X)$, say, of the computation:

1. Mathematical propagation
2. Statistical propagation
3. Numerical propagation.

Mathematical propagation constitutes perturbation theory, the influence of changes Δx in a realization (input data) x of X to effect a change Δy in the corresponding instance (result or solution) y of Y.[3] It gives the *sensitivity* of the solution to the input data. Mathematical propagation is largely an exercise in differentiation, in that an indication of the sensitivity of the solution to the data is $\|\Delta y\|/\|\Delta x\|$, where $\|z\|$ is a measure of the 'size' of z, which can be obtained, at least approximately, in terms of derivatives.[4] Automatic differentiation (AD) [11] is a valuable tool here (section 4).

[1] http://www.nag.co.uk

[2] http://www.eurometros.org

[3] X denotes a scalar quantity or (more commonly) a set of quantities, and similarly for Y.

[4] $\|z\|$ denotes the absolute value of z if z is a scalar (single-valued) quantity or some norm, e.g., the 2-norm (square root of the sum of the squares of the components) if z has a number of components.

A *problem* is said to be ill-conditioned if $\|\Delta y\|/\|\Delta x\|$ is large.

Statistical propagation constitutes the propagation of uncertainties or the propagation of distributions through the algorithm to provide, e.g., the uncertainty matrix (uncertainty matrix) $V(y)$ associated with y given the uncertainty matrix $V(x)$ associated with x. There is much current activity in this area [32].

Numerical propagation relates to the numerical stability of an algorithm. Given data x, an *algorithm* (operating in terms of finite precision) produces a solution $y + \delta y$, say, that will differ from the exact solution y (which would be obtained using infinite precision) by the amount δy. Alternatively, $y+\delta y$ can be regarded as the *exact* solution to the problem with certain data $x+\delta x$. How big is δy? How big is δx? These are questions that numerical propagation, especially the associated error analysis (section 2.1.1) sets out to answer. The topics of *forward error analysis* (section 2.1.2) and *backward error analysis* (section 2.1.3) are used to address these questions.

An *algorithm* is ill-conditioned if $\|\delta y\|$ is large compared with $\|\Delta y\|$ (when $\delta x = \Delta x$).

2.1.1 Error analysis

To understand properly the effect of computer rounding errors, representation errors and truncation errors on a computation, it is necessary to perform an error analysis.

A *rounding error* generally occurs for each basic arithmetic operation performed. The operations of addition, subtraction, multiplication and division constitute these basic arithmetic operations. The strictly numerical parts of all computations can be expressed at the lowest level in terms of these operations.

A *representation error* occurs when a *number* is approximated (represented) on the computer. For instance, in representing e on the computer by 2.718 3, the magnitude of the representation error is bounded by 0.000 05. This result is a consequence of 2.718 3 correctly representing e to four decimal places.

A *truncation error* results from representing a *function* approximately on the computer. For instance, if Taylor's series is used to approximate $f(x + h)$ as

$$f(x + h) \approx f(x) + hf'(x) + (h^2/2)f''(x),$$

the mean value theorem can be used to give the truncation error as $(h^3/6)f'''(x+\theta h)$, $\theta \in (0, 1)$, from which a bound for the truncation error can often be determined.

The general aim of an error analysis is to decide whether or not a method is stable for the problem at hand, through quantifying the size of δx or δy above. For example, in section 2.5 the computation of the mean of two numbers is discussed and it is natural to ask under what circumstances a particular method for computing the mean is satisfactory. Generally, an error analysis can be considered in terms of the mathematical propagation of errors, as in section 1.1, where an integral is evaluated by recursion. It can also be considered, as predominantly in this guide, in the presence of floating-point arithmetic.

2.1.2 Forward error analysis

Strictly, numerical propagation involves tracking the effects of floating-point operations on every intermediate quantity formed during the entire computational

algorithm in order to assess their impact on the result. This process is termed *forward error analysis*. In practice, there is often a critical part of the algorithm at which 'accuracy is lost', the remainder of the algorithm being relatively innocuous in this regard. This guide is generally concerned with such aspects rather than complete, formal error analysis.

When undertaking an error analysis it is natural to try to obtain a bound for the error in the computed solution to the problem to be solved. The bound would relate to $\|\delta y\|$ (in an absolute sense) or $\|\delta y\|/\|y\|$ (in a relative sense). For example, if the requirement is to solve the system of linear equations $A\mathbf{y} = \mathbf{x}$ and the computed solution is denoted by $\widehat{\mathbf{y}}$, a bound for the relative error $\|\widehat{\mathbf{y}} - \mathbf{y}\|_2/\|\mathbf{y}\|_2$ might be sought. Such a bound is called a forward error bound. For some problems it is quite reasonable and feasible to try to determine such a bound, but in other cases the analysis may lead to pessimistic results, or may even be inappropriate.

Consider the analysis of the method of Gaussian elimination [44, chapter 9] for the solution of the linear equations $A\mathbf{y} = \mathbf{x}$. Linear equations can be very ill-conditioned, or indeed even singular, and an error analysis to obtain a forward error bound would have to reflect both the sensitivity of the problem to perturbations in the data, and the effect of the rounding errors. Thus, typically, a forward error analysis of a system of linear equations would be unduly pessimistic.

An *algorithm* is *forward stable* (for x) if the result it produces has an error of similar magnitude to that produced by a backward stable algorithm [44, p10].

2.1.3 Backward error analysis

With a forward error analysis the objective is to track the errors at the various stages of a calculation, and bound the error associated with the results. With a *backward error analysis*, the results obtained are in a sense 'accepted' and the question asked, 'To what set of data do the results correspond?' A bound for $\|\delta x\|$ is sought.

This consideration is highly relevant to the metrologist. Typically his problems are such that he will provide estimates of the values of the input quantities and he will have standard uncertainties associated with these estimates that quantify their reliability. If the magnitudes of the above perturbations are negligible compared with (or at worst comparable to) the corresponding uncertainties, it can be concluded that the problem actually solved corresponds to a problem within the set of problems represented by the above estimates and the associated uncertainties.

An algorithm is *backward stable* (for x) if $\|\delta x\|$ is small, the definition of 'small' being context-dependent [44, p8].

2.2 The role of numerical software libraries

Enormous investment has been committed in producing high-quality numerical software libraries, such as the NAG Library [39]. The formulation of a solution to a problem in a way that maximizes the use of library routines, for which it can generally be assumed they represent the state-of-the-art in numerical software solutions, is often the most economical and potentially the most reliable

approach. The metrologist can then concentrate on the parts of the software that call and interconnect these routines.

It is at this level where failure to pay attention to detail can destroy the gains achieved by using the library components. The benefit (over the use of less reliable software) achieved by using quality library components can be sacrificed by employing a version of just one formula that has undesirable properties [30].

Sometimes the 'interconnecting' software appears to be innocuous. Instances include the normalization and scaling of variables, and the provision of the normal matrix $X^{\mathrm{T}}X$ rather than the design matrix X to a 'least-squares solver'. The *use* of the algorithm can induce ill-conditioning.

The art of problem-solving in metrology and other disciplines is to subdivide the problem, particular if it is of appropriate magnitude, into a set of interrelated functional modules, where as many as possible of the modules correspond to a 'standard mathematical computation': least-squares polynomial regression, zero of a function, solution of a constant-coefficient ordinary differential equation, etc.

Also, in general, metrologists would benefit from the use of software that gives an indication of the quality of the solution [43, 47].

2.3 Do-it-yourself sensitivity analysis

Carrying out a sensitivity analysis of a problem can be very revealing. If the solution is an explicit formula containing the key parameters of the problem, the effect of particular values for these parameters can be quantified, perhaps analytically. More generally, an explicit formula is not available. Rather, a software implementation of an algorithm exists. A 'do-it-yourself' sensitivity analysis can be considered in this case.

First, use the software implementation to provide the solution corresponding to some required or representative input data. Then, make small random perturbations to the values of the input quantities, perhaps those that mimic measurement uncertainties when appropriate, or based on probabilistic information concerning the values of the input quantities, and compute the corresponding solution. Compare the two solutions or, better, several solutions based on random perturbations.

One advantage of this approach is that it applies to the specific software implementation of an algorithm, i.e., the total solution, and not just to the (mathematical) algorithm. Thus, it is relevant to the software actually used for the computation. This point is, however, a disadvantage if an understanding of the properties of the algorithm alone is required.

For some problems, all the results might be perturbed to a comparable extent. For others, some of the results might be perturbed more than others, thus indicating the relative sensitivity of the specific results.

The sensitivity of the results with respect to particular items of input data can be studied by perturbing only some of the input data (e.g., one at a time).

It is essential that the perturbations are made in the correct context. For instance suppose the elements of a matrix are function of a parameter λ. It would be misleading to perturb the elements of the matrix individually. λ should be perturbed and the elements evaluated as functions of this perturbed value.

Some tools are available to assist with perturbation analysis [13, 57]. These tools implement facilities to assess the impact of rounding on a computation, by treating the roundings as a sample drawn randomly from a set of possible similar computations differing only in one of two ways:

1. In the input data, which is randomly perturbed slightly from the given data [13].

2. In the arithmetic operations, which are randomly perturbed slightly [57].

For either approach the computation is carried out a number of times, each time with different randomized perturbations. The consequent set of results, in the case of a scalar result, is regarded as a sample whose mean estimates the required result and standard deviation a measure of the spread of the results. If there is more than one output quantity (a vector), it can be used to provide estimates of these results and the associated uncertainty matrix. For the case of randomly perturbed data, there is a similarity with the propagation of distributions [23], certainly if the random perturbations respect the distributions that are appropriate for the values of the problem input quantities of which the prescribed data are realizations.

Both approaches have been criticized by Kahan[5] on several grounds:

1. Rounding errors are not random (nor are their accumulated effect on a computation [44]). Neither are they uncorrelated.

2. The approaches fail to mimic important properties that actual rounding errors possess.[6]

3. The approaches usually work but, when they fail, they do so in just those situations when a warning is needed that the computation has gone astray.

Even if these criticisms do not deter the use of the tools, there will an inevitable learning curve associated with their use, as well as the logistical considerations in integrating the tool with the user's computational environment. Users may well wish to carry out their own sensitivity analysis, since this can often be done easily.

For an example, see the case study on climate change (section 5.7).

Automatic differentiation (AD) can provide a means to determine problem sensitivity (section 2.4).

2.4 Condition of function evaluation

Formula or function evaluation is considered in detail in chapter 3. Here the condition of function evaluation is considered. The treatment uses automatic differentiation, covered more fully in chapter 4.

[5] www.cs.berkeley.edu/ wkahan/improber.pdf

[6] For instance, the difference $a - b$ between two floating-point numbers a and b of the same sign is computed *exactly* if $1/2 \leq a/b \leq 2$ [55], whereas a probabilistic approach would assign an error in all cases. Thus, in forming a difference table, for example, many of the entries will be determined exactly in terms of the relevant values in the previous column.

Consider a function $f(x)$ to be evaluated at the point $x = x_0$. The *relative condition number* [44, p9],[52] is[7]

$$k = \left| x_0 \frac{f'(x_0)}{f(x_0)} \right|.$$

If this expression is considered in the light of a *scaled* problem for which x_0 and $f(x_0)$ are of the order of unity, $|f'(x_0)|$ is essentially the condition number. This quantity constitutes a *sensitivity coefficient* within uncertainty evaluation [7].

Making early use of a simple automatic differentiation (AD) facility [11] (section 4.1), the determination of k can be illustrated using a MATLAB script:

```
h = 1e-100;
z = f(x0 + sqrt(-1)*h);
f0 = real(z)
g0 = imag(z/h)
k = abs(x0*g0/f0)
```

In this script, $\sqrt{-1}h$ denotes the (imaginary) finite-difference step, z the function f evaluated at x_0 perturbed by the imaginary quantity ih, where $i = \sqrt{-1}$, f_0 the function value at x_0, g_0 the value of $f'(x_0)$, and k the required relative condition number.

That this (AD) process works, to within a numerical error that is negligible for almost all practical purposes, follows from the Taylor series approximation

$$f(x_0 + ih) = f(x_0) + ihf'(x_0) + O(h^2),$$

to $f(x)$ at $x = x_0$, and taking real and imaginary parts.

To illustrate, consider the function $f(x) = x^{1/10}$ with $x_0 = 2$. The use of the MATLAB inline function

```
f = inline('x^(1/10)');
```

followed by the execution of the above script gives

```
f0 =
    1.0718
g0 =
    0.0536
k =
    0.1000
```

Thus the relative condition number is 0.1, implying that a *gain* in numerical accuracy of one significant decimal digit is achieved by the operation. This result means that x_0 need be known to an accuracy that is somewhat less than that required for the corresponding function value $f(x_0)$. To demonstrate this aspect, suppose x_0 has a value of 2.00 and the associated standard uncertainty $u(x_0)$

[7] In a backward error sense, let the approximate solution $\widehat{y}$ satisfy $f(x + \delta x)$. Then, if f is twice continuously differentiable, $\widehat{y} - y = f(x + \delta x) - f(x) = f'(x)\delta x + f''(x + \xi\delta x)(\delta x)^2/2$, $0 < \xi < 1$. Since $(\widehat{y} - y)/y = (xf'(x)/f(x))(\delta x/x) + O((\delta x)^2)$, the quantity $|xf'(x)/f(x)|$ measures, for small x, the relative change in the output for a given relative change in the input.

is 0.02. According to the above result, a relative perturbation of 1 % in x_0 (equal to the standard uncertainty $u(x_0)$) would perturb $f(x_0)$ by of the order of 0.1 %. In fact, $f(1.98) = 1.070\ 7$ and $f(2.02) = 1.072\ 8$, confirming the prediction.

The practical significance of this result is as follows. Suppose the target standard uncertainty associated with $y = f(x) = x^{1/10}$ were 0.1 %. Then, x needs to be measured with an associated standard uncertainty of 1 %. An awareness of such issues is valuable because of its usefulness in designing a measurement.

On the other hand, were $f(x) = x^{10}$, the condition number would be 10 rather than 0.1. In consequence, to achieve a target standard uncertainty associated with y of 0.1 % would require x to be measured with an associated standard uncertainty of 0.01 %, one hundredth of the value in the first example.

Results of this type can also be obtained approximately using the do-it-yourself sensitivity analysis of section 2.3.

2.5 Floating-point arithmetic

Two simple illustrations of floating-point arithmetic are given. The first is the arithmetic mean of two numbers. The second is the bisection algorithm, which has the arithmetic mean of two numbers at its core.

2.5.1 Arithmetic mean of two numbers

Taking the arithmetic mean of two numbers may superficially be viewed as an innocent operation that is free from surprises. However, it is possible that in floating-point arithmetic the arithmetic mean of two numbers is not contained within the (closed) interval (i.e., the interval between the two numbers including the numbers themselves).

If it is invalid to make such a monotonicity assumption within a piece of software that has to make certain decisions, the flow of control could be routed along an incorrect path with damaging results. The function $\sin x$ is strictly increasing in the interval $0 \leq x < \pi/2$, but is it safe to assume that the computed values of this function are also strictly increasing within this interval? Again, such an assumption embodied in software could have unexpected and potentially dangerous consequences.

Decimal arithmetic, operating to two significant decimal digits (2S arithmetic), is used to illustrate the point. The arithmetic mean y of two numbers a and b is

$$y = (a + b)/2, \tag{2.1}$$

and is conventionally computed in the sequence

1. $t = a + b$.
2. $y = t/2$.

The floating-point counterpart of this sequence (where $\mathrm{fl}(x)$ denotes the value of x computed using floating-point arithmetic) is

1. $\widehat{t} = \mathrm{fl}(a + b)$.

2. $\widehat{y} = \text{fl}(\widehat{t}/2)$.

For $a = 37$ and $b = 86$,

1. $\widehat{t} = \text{fl}(37 + 86) = \text{fl}(123) = 120$ (rounding to 2S).

2. $\widehat{y} = \text{fl}(120/2) = 60$.

The exact mean is $y = 61.5$. $\widehat{y}$ lies in the interval $[a,\ b]$ and hence satisfies the monotonicity property, although it is not equal to the closest representable number to the exact mean, viz., 61 or 62 (which are equally close).

For $a = 52$ and $b = 54$,

1. $\widehat{t} = \text{fl}(52 + 54) = \text{fl}(106) = 110$.

2. $\widehat{y} = \text{fl}(110/2) = 55$.

$\widehat{y}$ does not lie in the interval $[a,\ b] \equiv [52,\ 54]$ and hence does not satisfy the monotonicity property. The exact mean is $y = 53$.

Now consider $a = 0.999\ 9$ and $b = 0.999\ 7$, using 4S arithmetic:

1. $\widehat{t} = \text{fl}(0.999\ 9 + 0.999\ 7) = \text{fl}(1.999\ 6) = 2.000$.

2. $\widehat{y} = \text{fl}(2.000/2) = 1.000$.

The exact mean is $y = 0.999\ 8$. Again, $\widehat{y}$ does not lie in the interval $[a,\ b]$ and hence does not satisfy the monotonicity property and, as before, is different from the closest representable number, viz., 0.999 8, the exact answer.

Intermediate (and final) floating-point values are not necessarily exactly representable, which means that an error is committed, the error being propagated into the next step in the sequence. For the arithmetic mean computation, *growth* has occurred, viz., at the first step where, if a and b take the same sign, t will be larger in magnitude than the magnitude of either of them. If they were of different sign, t would be smaller in magnitude than the magnitude of either of them, and growth would not occur. This simple observation provides a clue to a method of computation that does not suffer this way.[8]

Express y in the *mathematically identical* form

$$y = a + (b - a)/2, \tag{2.2}$$

i.e., as one of the numbers (a), plus a correction ($(b - a)/2$) to it, and use this form as a basis for the computation. When a and b are 'close', no 'growth' as such will arise with the use of this form.

The computational sequence is

1. $t_1 = b - a$.

2. $t_2 = t_1/2$.

3. $y = a + t_2$.

[8]When implemented in IEEE arithmetic (binary as oppose to decimal arithmetic), the computation is both forward and backward stable: the calculated value of $(a + b)/2$ has a small error, and is the exact result for slightly perturbed values of a and b.

The sequence now contains three steps instead of two. That is the price to pay (in this case) for numerical stability (here monotonicity).[9]

The (obvious) floating-point counterpart of the sequence is

1. $\widehat{t_1} = \mathrm{fl}(b - a)$.
2. $\widehat{t_2} = \mathrm{fl}(\widehat{t_1}/2)$.
3. $\widehat{y} = \mathrm{fl}(a + \widehat{t_2})$.

For the first example,

1. $\widehat{t_1} = \mathrm{fl}(86 - 37) = 49$ (representable exactly in 2S).
2. $\widehat{t_2} = \mathrm{fl}(49/2) = \mathrm{fl}(24.5) = 24$ (rounding to 2S).
3. $\widehat{y} = \mathrm{fl}(37 + 24) = 61$ (representable exactly in 2S).

In the above, $\mathrm{fl}(49/2)$ was taken as 24. Equally, it could have been taken as 25, since both these values for $\widehat{y}$ are 'equidistant' from the exact value of $y = 24.5$. The use of 25 would deliver $\widehat{y} = 62$. The two possible results, 61 and 62, are the closest numbers (one each side) in the arithmetic used (2S) to the exact value of 61.5.

For the second example,

1. $\widehat{t_1} = \mathrm{fl}(54 - 52) = 2$.
2. $\widehat{t_2} = \mathrm{fl}(2/2) = 1$.
3. $\widehat{y} = \mathrm{fl}(52 + 1) = 53$,

yielding the exact result.

For the third example,

1. $\widehat{t_1} = \mathrm{fl}(0.999\ 7 - 0.999\ 9) = -0.000\ 200\ 0$.
2. $\widehat{t_2} = \mathrm{fl}(-0.000\ 200\ 0/2) = -0.000\ 100\ 0$.
3. $\widehat{y} = \mathrm{fl}(0.999\ 9 - 0.000\ 100\ 0) = \mathrm{fl}(0.999\ 8) = 0.999\ 8$,

again an exact result.

Although the loss of monotonicity in forming the arithmetic mean using formula (2.1) can occur in decimal arithmetic, as demonstrated above, it cannot occur in binary arithmetic, as implemented on most computers, and especially on most PCs. However, the general principle of representing a result as a value

[9] How far should one go with such detailed considerations? Does it matter? In some situations, the answer is 'yes' (e.g., in health-critical and safety-critical areas). Perhaps a result that failed a monotonicity assumption could take a program along a different path, which led to a bad decision (or perhaps, worse, along a path that had not been encountered previously and insufficiently tested) [58]. Fitness-for-purpose is the issue. There is an even better form for the formula (2.2), which as it stands is appropriate for values of the same numerical sign. If, however, a and b take opposite signs, the formula is *worse* than the original! Thus, a better form is

$$y = \begin{cases} a + (b - a)/2, & \mathrm{sign}(a) = \mathrm{sign}(b), \\ (a + b)/2, & \text{otherwise.} \end{cases}$$

.

plus an increment, as in expression (2.2), is generally sound in the design of numerical algorithms: doing so avoids unnecessary 'growth'. In a case where the input data and the results are of comparable size, as here, it is generally better from the viewpoint of numerical errors to use an algorithm for which the intermediate quantities are also of that magnitude. For a and b of the same sign, the straight mean has some growth; the incremental form does not.

2.5.2 The bisection algorithm

The *bisection algorithm* is commonly used to determine a zero of a continuous function $f(x)$ given an interval $[a,\ b]$ that definitely contains a zero, i.e., when the signs of $f(a)$ and $f(b)$ are different. f is evaluated at the midpoint $c = (a+b)/2$ of the interval. An interval of one half the length is given by replacing the endpoint whose function values has the same sign as $f(c)$ by c. The process is repeated until the length of the interval is no greater than a prescribed threshold.

This basic idea can be improved in two ways. First, the incremental form $c = a + (b - a)/2$ should be used when a and b have the same sign. Second, the stopping rule is flawed. The threshold might not be attainable, if too small a value were stipulated. The process should be terminated if the threshold is achieved *or* the *calculated* midpoint of the current interval is not *strictly* interior to the interval. As a consequence, either the requested threshold is met or the best-possible accuracy for the arithmetic employed is attained.

When an interval contains a zero, the bisection method will not fail to find it (assuming the above considerations concerning termination are taken into account). However, on the average it is among the slowest of all methods for determining zeros. When an interval contains an odd number of zeros, the bisection method will find one of them.

The worst-case performance of the bisection algorithm is optimal. For a given length of interval and a specified (attainable) accuracy, the algorithm will always take the same number of function evaluations (or a smaller number on those relatively few occasions when (the computed value of) the function is zero at one of the points of evaluation). This number of function evaluations is approximately $\log_2 |b - a|/t$, where a and b are the interval endpoints and t is an attainable positive absolute tolerance. On *average* the performance is not striking. At NPL the bisection algorithm has been used in many applications where reliability rather than speed is the dominant consideration.

2.6 Floating-point error analysis

The considerations of section 2.5 provide a lead-in to simple floating-point error analysis. The few basic concepts in floating-point error analysis can be stated informally as follows:

Floating-point representation. Let F denote the set of floating-point numbers that can be held on the computer concerned. For IEEE arithmetic the range of F is approximately from 10^{-308} to 10^{308}, together with their negative counterparts.

For any x in the range of F, let $\mathrm{fl}(x)$ denote the closest floating-point number to $x \in F$ (or one of the closest if x is halfway between two floating-

point numbers). Then

$$\mathrm{fl}(x) = x(1+e), \qquad |e| < \eta.$$

In words, the relative error in holding x on the computer is smaller than η ($\eta \approx 10^{-16}$ for IEEE arithmetic): see *computational precision* below. e is the *representation* error (section 2.1.1).[10]

Computational precision. The computational precision can be characterized by the *machine epsilon*, defined to be the distance from 1.0 to the next floating-point number [45, p41]. In MATLAB it is known as eps and takes the value 2.2×10^{16}, approximately. Somewhat more useful for stating the results of floating-point error analyses is the *unit roundoff*, the distance from 1.0 to the closest floating-point number *below* 1.0. It takes the value 1.1×10^{16}, approximately, half the machine epsilon, and is denoted in this guide by η.[11]

Model for floating-point arithmetic. For any $x,\ y \in F$, the results of the basic arithmetic operations $+,\ -,\ \times,\ /$ satisfy

$$\mathrm{fl}(x \circ y) = (x \circ y)(1+e), \qquad |e| \le \eta, \qquad \circ = +,\ -,\ \times,\ /.$$

Also, the square-root function satisfies

$$\mathrm{fl}(\sqrt{x}) = (\sqrt{x})(1+e), \qquad |e| \le \eta.$$

The symbol e_j, $j = 1, 2, \ldots$, is used to denote a quantity satisfying $|e_j| \le \eta$. It will be used generically in that, for instance, e_1 in one example will be different from e_1 in another example. To simplify the algebra, without making other than an academic sacrifice, terms of $O(e_j e_k)$ are ignored and thus products such as $(1+e_1)(1+e_2)(1+e_3)$ are replaced by $1+3e_4$, say. Such products arise frequently in floating-point error analysis.

Three basic rules resulting from the above considerations can be expressed:

1. The result of a single floating-point operation satisfies

$$\mathrm{fl}(x \circ y) = (x \circ y)(1+e), \qquad |e| \le \eta, \qquad \circ = +,\ -,\ \times,\ /.$$

2. The approximation

$$(1+k_1e_1)^{\pm 1} \ldots (1+k_qe_q)^{\pm 1} \approx 1 + \left(\sum_{j=1}^{q} k_j \right) e_{q+1},$$

where the k_j are positive or negative numbers of $O(1)$, holds. This rule corresponds to neglecting products of the e_j.

3. Each e_j is such that $|e_j| \le \eta$, the unit roundoff.

[10] e denoted the base of natural logarithms in sections 1.1 and 2.1.1.

[11] The symbol u is often used for the unit roundoff in modern numerical analysis texts [45]. However, since u is used extensively by metrologists to denote uncertainty, it is not used here to denote computational precision.

One aim of floating-point error analysis is to provide a bound for the forward error in the result of a floating-point computation. Thus, if y denotes the exact result and $\mathrm{fl}(y)$ the result obtained using floating-point arithmetic, an absolute error bound would take the form

$$|\mathrm{fl}(y) - y| \leq C\eta$$

and a relative error bound the form (with a different C)

$$\left|\frac{\mathrm{fl}(y) - y}{y}\right| \leq C\eta.$$

Another aim is to provide a bound in a backward-error sense (section 2.1.3). C denotes a factor that depends on the inputs to the computation, e.g., the parameters that enter a formula or the elements of a matrix or vector. y may be a vector-valued quantity or a matrix, in which case matrix or vector norms would be used in place of absolute value.

Wilkinson, an eminent numerical analyst of the Fox era, has stated [59, p26] that, if the magnitude of the error is bounded by $C\eta$, the error is typically of order $C^{1/2}\eta$. He accepted this as a rule of thumb, which has today become well known as such. The rule can be supported by assuming that the rounding errors are realizations of independent random variables and applying the central limit theorem. Rounding errors are not, however, random. See figure 3.4 in section 3.2.5 and the views of Kahan (section 2.3). Several eminent numerical analysts [44, p52] have modelled rounding errors in a statistical or probabilistic way. The situation is placated somewhat by the statement, 'There is no claim that ordinary rounding and chopping are random processes, or that successive errors are independent. The question to be decided is whether or not these particular probabilistic models of the processes will adequately describe what actually happens.' [46].

Chapter 3

Formula evaluation

The need to evaluate formulae is widespread in metrology. Formulae are provided in papers, reports, data sheets, procedures, specifications, national and international guides and standards, and elsewhere. They are used to represent both physical and empirical relationships between metrological quantities, e.g., the density of mercury as a function of temperature [5], the refractive index of air as a function of air density [8], and the amount of material extracted as a function of time (section 3.4).

Metrologists arguably have the right to expect that a formula provided in a standard or a scientific journal or some other reputable source is fit for purpose *as it stands*. In other words, the formula can be applied directly for manual computation with a hand-held calculator or implemented on a computer through the use of a spreadsheet or other software application. This is not always the case, however. Such use, even if implemented soundly, may yield a result having less numerical accuracy than might be expected, especially compared with those that would be obtained from the use of alternative mathematically equivalent formulae. In some instances the loss of numerical accuracy may be such that the results would need to be examined carefully to ensure they are adequate for the task in hand. It is not always obvious when such loss has occurred.

Reasons for loss of accuracy, illustrated in this best-practice guide, include

Damaging subtractive cancellation. An example is the evaluation of the sum of the geometric series having first term A, positive common ratio r and n terms from the formula $S = A(1 - r^n)/(1 - r)$, when r is close to unity [28]. In this case, the subtraction of the computed value of r^n from one usually results in a large relative error.[1]

Inadequate parametrization. Inherent ill-conditioning in a problem (which might not be great) is worsened by a poor parametrization An example is the monomial representation of a polynomial $p(x)$, i.e., as a linear combination of powers of x, the 'raw' variable, as might be provided by least-squares regression, say. The products of the monomial coefficients and the corresponding powers of x can be extremely large compared with

[1] The less economical formula $S = A(1 + r + r^2 + \cdots + r^{n-1})$, particularly if evaluated using $A(1 + r(1 + r(1 + \cdots)))$, involves no subtraction and yields a very small relative error for all positive values of r.

the value of p, and thus much accuracy is lost when evaluating p for various values of x.[2]

Poor normalization or scaling. Consider the normalization, i.e., transformation of an arbitrary interval $a \leq x \leq b$ to the standardized interval $-1 \leq y \leq 1$. Such a transformation is used in a number of areas of relevance to metrology, one of the commonest being in normalizing the range of the independent variable in data approximation by polynomials [15, 20]. For instance, the original interval might be [273, 300] K in temperature measurement. The transformation is needed to reduce the ill-conditioning of the monomial form of a polynomial if that form is to be used, or the Chebyshev form for which the interval $[-1,\ 1]$ is almost invariably used.[3] The normalization formula is

$$y = \frac{2x - a - b}{b - a} = \frac{(x - a) - (b - x)}{b - a}, \tag{3.1}$$

where $[a,\ b]$ is the interval in the original variable over which the approximation is required.

It is shown in section 3.1.1 that one of these forms is stable and the other unstable [18].

Error build-up. An example is the sum $x_1 + \cdots + x_n$ of a very large number of values x_i (as would arise, for instance, in forming the expectation (mean) of an approximation to a probability density function obtained from a Monte Carlo implementation of the propagation of distributions [23]). An accurate sum is important when calculating the standard deviation of a large number of comparable values (cf. section 1.2), in order that the determination of the differences of the individual items in the sample from the calculated mean have sufficiently small error. Methods other than 'conventional addition' are available for this purpose [31],[44, chapter 4].

Underflow. An example is the evaluation of the geometric mean y [21] of n positive numbers $x_1, \ldots, x_n$ from the formula $y = (x_1 \times \cdots \times x_n)^{1/n}$ when each x_i is smaller than unity and n is large. For instance, if each x_i is approximately 0.5 and n is 400, say, then in IEEE arithmetic, the answer is computed as zero, even though the correct result is approximately 0.5. The difficulty can be overcome by scaling the numbers or by forming $\log_e y = (1/n) \sum_{i=1}^{n} \log_e x_i$ and exponentiating the result. Approaches for dealing with an arithmetic mean can then be applied to the $\log_e x_i$.

Overflow. The same example as for underflow applies, where each x_i is now greater than unity, approximately 2.0, say.[4]

[2]The use of a Chebyshev-series representation, in a normalized variable, of p avoids this difficulty [18]. The contributions to the sum are then generally such that their magnitudes are comparable to or smaller than the absolute value of p.

[3]The Chebyshev form of a polynomial is $a_0 + a_1 T_1(y) + \ldots + a_{n-1} T_{n-1}(y)$, where $T_r(y)$ is the Chebyshev polynomial of the first kind of degree r in y. The Chebyshev polynomials can be evaluated using the three-point recurrence relation $T_0(t) = 1$, $T_1(t) = t$, $T_r(t) = 2tT_{r-1}(t) - T_{r-2}(t)$, $r = 2, 3, \ldots$.

[4]If overflow occurs, it is not strictly correct to say the result is inaccurate, but unavailable by this means.

There are many forms of computation that include such loss of accuracy in a more concealed form, e.g., changing the order of some arithmetic operations may influence substantially the effects of subtractive cancellation.

3.1 Subtractive cancellation, growth in intermediate quantities, etc.

Those familiar with the principles of uncertainty evaluation may recognize behaviour in the examples below that are reminiscent of the consequences of random and systematic effects that arise in determining a measurement result for a given model of measurement.

3.1.1 Variable normalization

Consider the first formula in expression (3.1). The application of the basic rules of floating-point arithmetic, with the indices of the error terms e_j indicating the order of operation, and since $2x$ is formed exactly in IEEE arithmetic,

$$\begin{aligned} \widehat{y} = \mathrm{fl}(y) &= \frac{((2x-a)(1+e_1)-b)(1+e_2)}{(b-a)(1+e_3)}(1+e_4) \\ &= \frac{(2x-a)(1+e_1)-b}{b-a}(1+3e_5). \end{aligned}$$

Thus,

$$\widehat{y}-y = \frac{2x-a}{b-a}e_1 + \frac{2x-a-b}{b-a}(3e_5)$$

and so

$$|\widehat{y}-y| \leq \left(\frac{|2x-a|}{b-a}+3|y|\right)\eta \leq \left(\frac{|2x-a|}{b-a}+3\right).$$

It is appropriate here to use the absolute error $|\widehat{y}-y|$ rather than the relative error $|\widehat{y}-y|/y$, since y can lie anywhere in the interval $[-1,\ 1]$.

Possible alternative orders of operation give factors $|2x-b|/(b-a)$ or $|a+b|/(b-a)$ in place of $|2x-a|/(b-a)$. All these forms are unstable in that for cases in which $a \gg 0$ or $b \ll 0$, the bound could be of arbitrary size. For instance, when $a = 1000$ and $b = 1001$, all these bounds are approximately 1000, implying a loss of up to three decimal digits for this 'simple' computation alone.

Consider the second formula in expression (3.1). Then, in terms of different e_j in general,

$$\widehat{y} = \frac{((x-a)(1+e_1)-(b-x)(1+e_2))(1+e_3)}{(b-a)(1+e_4)}(1+e_5)$$

and hence

$$\widehat{y}-y = \frac{x-a}{b-a}e_1 - \frac{b-x}{b-a}e_2 + \frac{2x-a-b}{b-a}(e_3-e_4+e_5),$$

giving

$$|\widehat{y}-y| \leq \left(\frac{x-a}{b-a}+\frac{b-x}{b-a}+3|y|\right)\eta = (1+3|y|)\eta \leq 4\eta.$$

This bound does not depend on a, b or x and is very small, and thus the second formula is highly stable regardless of the original interval.

3.1.2 Polynomial evaluation

Consider the evaluation of a polynomial of degree six [52, p125] in the region of a minimum of that polynomial. 201 uniformly spaced points of evaluation in the interval [0.992, 1.008] were used. Figure 3.1 shows a plot of the results obtained. The function is convex in shape, with essentially flat behaviour in the middle. There is irregular behaviour with a small amplitude, more apparent in the flatter part of the curve, which might be dismissed as of minor consequence, possibly on the basis of 'an inadequacy in the graphics software'. A closer inspection of this region indicates more than minor irregularities.

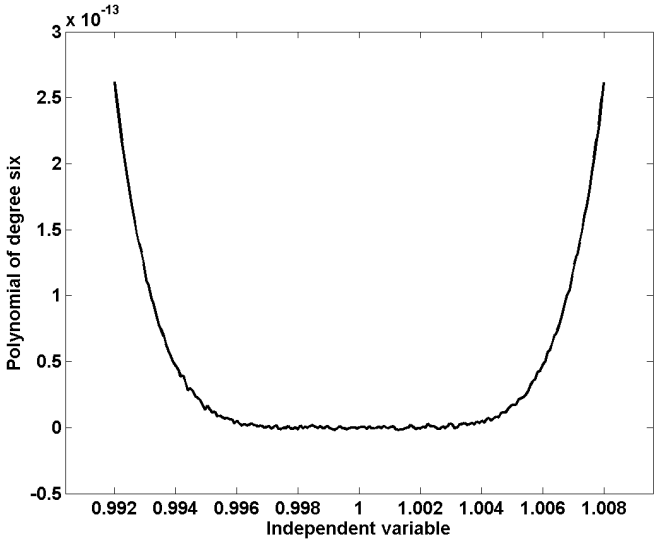

Figure 3.1: A polynomial of degree six evaluated over a specified interval.

The function was evaluated once more, but now over a reduced interval [0.995, 1.005] covering the 'flat region'. The result obtained is shown in figure 3.2. The behaviour of the function is without doubt highly erratic, and nothing like the smooth behaviour expected of a (low degree) polynomial. The irregularities correspond to large relative errors in the computed values of the polynomial. Superimposed on this figure is the polynomial evaluated in a stable manner (below).

In fact, the polynomial in monomial form is

$$p(x) = 1 - 6x + 15x^2 - 20x^3 + 15x^4 - 6x^5 + x^6. \tag{3.2}$$

This was the form that was evaluated in creating the 'jagged' figures. The evaluation of the form (3.2) near $x = 1$ suffers appreciable subtractive cancellation: individual terms in the polynomial (e.g., $15x^2$ and $-20x^3$) have magnitudes between one and twenty, yet the value of p is very much smaller, e.g., of the order of 10^{-8} at $x = 0.95$. So some nine significant decimal digits are lost for this value of x. For $x = 0.99$, some 13 digits are lost.

$p(x)$ above is in fact the expansion of $(x - 1)^6$, a form that is perfectly stable for evaluation. The smooth curve in figure 3.2 was formed using this representation.

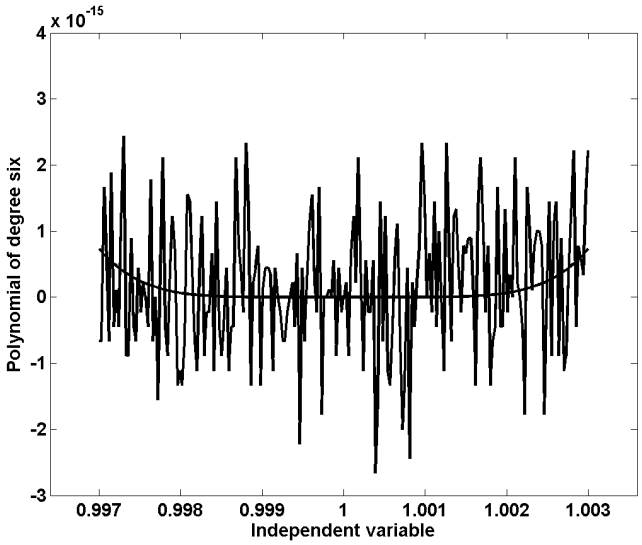

Figure 3.2: As figure 3.1, but evaluated over the central portion of the interval.

Imagine using the form (3.2) as input to (e.g., library) software for determining the minimum of a function. The noise induced into the values of the function by the use of representation (3.2) would cause a poor estimate of the minimum to be returned. It would also be likely to introduce difficulties for the minimization routine, which is probably designed to operate effectively for smooth continuously differentiable functions. This function has all these properties mathematically, but not numerically in the field of floating-point arithmetic.

Further imagine the determination of a zero of this function. According to the graph there are many 'zeros' induced by the noisy function values. More strictly, there are many pairs of adjacent points when the computed value of the function takes opposite signs.

There are several messages:

1. The jagged behaviour in the curve is a consequence of the finite arithmetic used. Greater numerical precision would have been needed to produce a smooth curve from the form (3.2) for the polynomial.

2. A stable representation of the function, a polynomial in this case, would have given the expected smooth curve, with no need to work with extended precision.

3. A false conclusion could be drawn from inspecting results such as these, e.g., that a metrological relationship was more complicated than it actually was. The results are contaminated by use of an unstable form for the function.

4. The use of this form of the function in conjunction with software to carry out certain computations, such as determining a zero or a minimum, may cause numerical difficulties.

3.1.3 Angle between two vectors

A further example [16] is the calculation of the angle θ between two unit vectors. Given vectors $\mathbf{a}$ and $\mathbf{b}$, each of unit length, mathematically equivalent formulae for evaluating θ (cf. figure 3.3) are

$$\theta = \cos^{-1}(\mathbf{a}^{\mathrm{T}}\mathbf{b}) \tag{3.3}$$

and

$$\theta = 2\sin^{-1}\left\|\frac{\mathbf{b}-\mathbf{a}}{2}\right\|. \tag{3.4}$$

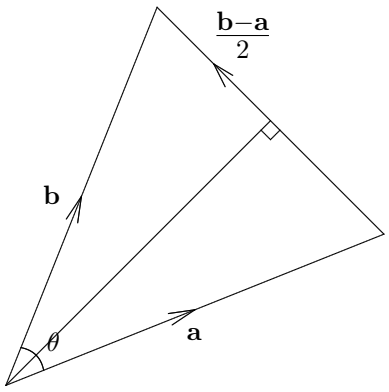

Figure 3.3: The angle between two unit vectors.

Which formula should be used? Formula (3.3) looks simpler and, on the face of it, might be appropriate. The difference in the behaviour of the formulae, especially for a very practical application in dimensional metrology (below), is, however, appreciable.

For vectors that are very nearly parallel, i.e.,

$$\mathbf{a}^{\mathrm{T}}\mathbf{b} = 1-\delta, \qquad 0 < \delta \ll 1,$$

the form (3.3) gives, approximately,

$$\cos\theta = 1 - \frac{\theta^2}{2} = \mathbf{a}^{\mathrm{T}}\mathbf{b} = 1-\delta,$$

i.e.,

$$\theta = \sqrt{2\delta}.$$

Since the smallest non-zero δ for which $1-\delta$ is representable is $\eta/2$, the smallest θ computable from formula (3.3) is $\sqrt{\eta}$, i.e., approximately 10^{-8} radians for IEEE arithmetic. Thus, formula (3.3) is unable to detect an angle smaller than 10^{-8} radians unless it is computed as zero, whereas the alternative formula (3.4) can be expected to return accurate values.

This example has serious consequences for those concerned with implementing the procedures described in the ISO International Standard [49]. This Standard is concerned with testing software for computing Gaussian best-fit geometric features to measured data that is used in industrial inspection and geometric

tolerancing applications. The Standard requires that the unit direction vector used in the parametrization of such geometric features as planes, cylinders and cones and returned by test software is compared with a reference solution by computing the angle between the test and reference vectors. The Standard defines acceptance of the test vector if this angle is smaller than 10^{-8} radians. It is clear from the above analysis that if formula (3.3) is used to evaluate the angle, this acceptance criterion can never be satisfied no matter how close are the test and reference vectors, unless the angle is computed as zero. On the other hand, there is no problem with undertaking the comparison if formula (3.4) is used.

The importance of using a stable formula is recognized in the standard, which includes information to the implementor to this effect.

3.2 CASE STUDY. Hardness measurement

3.2.1 Objective

The determination of the Brinell hardness number as part of the application of the Brinell hardness test.

3.2.2 What this case study illustrates

- The evaluation of a formula given in an international standard as part of a hardness test
- Subtractive cancellation associated with the use of this formula (chapter 3)
- An alternative, equivalent numerically stable formula
- Floating-point error analysis of the formulae.

3.2.3 The Brinell hardness test and the formula for the Brinell hardness number

The Brinell hardness test method consists of indenting the test material with a 10 mm diameter hardened steel or carbide ball subjected to a load of 3 000 kg. For softer materials the load is sometimes reduced to 1 500 kg or 500 kg to avoid excessive indentation. The full load is normally applied for 10 to 15 seconds in the case of iron and steel and for at least 30 seconds in the case of other metals. The diameter of the indentation left in the test material is measured with a low-powered microscope. The Brinell hardness number is calculated by dividing the load applied by the surface area of the indentation.

The Brinell Hardness Test [48] utilises the formula

$$B = \frac{0.204F}{\pi D(D - \sqrt{D^2 - d^2})} \tag{3.5}$$

for the Brinell hardness B in N/mm^2 as a function of the test force F in N, the diameter D in mm of the indenter and the mean diameter d in mm of the indentation.

3.2.4 Considerations regarding numerical evaluation

Suppose $F = 1\ 500$ kg. Consider the numerical evaluation of the term $T = D - \sqrt{D^2 - d^2}$ in the denominator of formula (3.5) for the values $D = 10.00$ mm and $d = 0.84$ mm. To four significant decimal digits (4S), using guard digits before each intermediate result is rounded as in IEEE arithmetic (and suppressing the cumbersome 'fl' notation used in section 2.5):

$$D^2 - d^2 = 10.00^2 - 0.84^2 = 100.0 - 0.705\ 6 = 99.294\ 4 \approx 99.29\ \text{mm}^2,$$

$$\sqrt{D^2 - d^2} = \sqrt{99.29} \approx 9.964\ \text{mm}$$

and hence

$$T = D - \sqrt{D^2 - d^2} = 10.00 - 9.964 = 0.036\ 00\ \text{mm}, \tag{3.6}$$

the trailing zeros being spurious (used purely to provide 4S) as a consequence of subtractive cancellation. Finally,

$$B = 270.6\ \text{N/mm}^2, \tag{3.7}$$

only the first two digits of which would be meaningful because, evidently, there is a loss of (some two) significant digits as a consequence of the subtractive cancellation in forming T at stage (3.6). The loss is not too damaging here, since, for the number of digits used, two significant digits remain in the value of $D - \sqrt{D^2 - d^2}$, although it might be damaging if fewer digits were recorded at the intermediate stages of a *hand* calculation. The remainder of the calculation presents no problem, since only multiplications and divisions are required, each of which introduces only a small relative error in the final result.

Repeating the calculations with enough digits to deliver 4S in the result gives

$$B = 275.6\ \text{N/mm}^2, \tag{3.8}$$

confirming the loss of two digits in the result (3.7).

Suppose a smaller force were applied or the material under test were harder. Then, d would be very much smaller than D, with the consequence that the formula (3.5) would fare worse. For instance, for $D = 10.00$ mm and $d = 0.05$ mm, in 4S,

$$D^2 - d^2 = 10.00^2 - 0.05^2 = 100.0 - 0.002\ 500 \approx 100.0\ \text{mm}^2, \tag{3.9}$$

$$\sqrt{D^2 - d^2} = \sqrt{100.0} = 10.00\ \text{mm}$$

and hence

$$D - \sqrt{D^2 - d^2} = 10.00 - 10.00 = 0\ \text{mm}.$$

The resulting value of B would be infinite! Such an absurd result would be spotted during a hand calculation at the stage (3.9) and the number of digits increased accordingly to compensate. Indeed, working to 6S,

$$D^2 - d^2 = 10.00^2 - 0.05^2 = 100.000 - 0.002\ 500\ 00 = 99.997\ 5\ \text{mm}^2,$$

$$\sqrt{D^2 - d^2} = \sqrt{99.997\ 5} = 9.999\ 87\ \text{mm}$$

and hence

$$D - \sqrt{D^2 - d^2} = 10.00 - 9.999\ 87 = 0.000\ 130\ 000 \text{ mm}.$$

The resulting value, to four digits, is

$$B = 7.493 \times 10^4 \text{ N/mm}^2. \tag{3.10}$$

To four digits, obtained by holding more digits throughout the computation,

$$B = 7.792 \times 10^4 \text{ N/mm}^2. \tag{3.11}$$

Thus, even working to 6S, only one digit is correct in the above result.

3.2.5 An alternative formula and its use

To avoid having to use additional precision as in section 3.2.4 and record intermediate results to more digits, formula (3.5) can be recast to avoid damaging subtractive cancellation. By multiplying the numerator and denominator of expression (3.5) by $D + \sqrt{D^2 - d^2}$ and simplifying, the following *mathematically equivalent* formula is obtained:

$$B = \frac{0.204F(D + \sqrt{D^2 - d^2})}{\pi D d^2} \tag{3.12}$$

The only term now involving subtractive cancellation is $D^2 - d^2$, which would not be damaging even if d was only just smaller than D, since the result, after square-rooting, is *added* to D, both terms being positive.

The above calculations are now repeated, but for the alternative formula (3.12). For the values $D = 10.00$ mm and $d = 0.84$ mm, working to 4S, $B = 275.6$ N/mm^2, and, for the values $D = 10.00$ mm and $d = 0.05$ mm, working to 4S, $B = 7.792 \times 10^4$ N/mm^2, both of which agree with the values (3.8) and (3.11) obtained using a sufficient number of intermediate digits to permit the result to be rounded correctly to 4S.

Formulae (3.5) and (3.12) contains two main features for numerical consideration: the evaluation of $D^2 - d^2$ and that of $D - (D^2 - d^2)^{1/2}$ when $0 \leq d \leq D$. The other aspects are numerically innocuous. Section 3.2.6 provides floating-point error analyses of these formulae and a related formula.

3.2.6 Floating-point error analysis

This section provides support for the results observed in section 3.2.4 using floating-point error analysis. In particular, such an analysis quantifies the manner in which the error bounds for the formulae depend on the quantities that are 'input' to the formulae. Computed values are denoted by hats.

First, the formula

$$S = D^2 - d^2 \tag{3.13}$$

and the alternative form

$$S = (D - d)(D + d) \tag{3.14}$$

are analyzed using the concepts of section 2.6.

Now,

$$\text{fl}(D^2) = D^2(1+e_1), \qquad \text{fl}(d^2) = d^2(1+e_2),$$
$$\widehat{S} = \text{fl}(S) = (D^2(1+e_1) - d^2(1+e_2))(1+e_3).$$

So the relative error in $\widehat{S}$ is

$$\frac{\widehat{S}-S}{S} = \frac{(D^2(1+e_1) - d^2(1+e_2))(1+e_3) - S}{S} = \frac{D^2e_1 - d^2e_2 + (D^2-d^2)e_3}{S}.$$

Hence,

$$\left|\frac{\widehat{S}-S}{S}\right| \le \left(\frac{D^2+d^2+(D^2-d^2)}{D^2-d^2}\right)\eta = \frac{2D^2}{D^2-d^2}\eta, \tag{3.15}$$

which can be arbitrarily large, depending on the closeness of d to D.[5]

Now consider the formula (3.14):

$$\text{fl}(D-d) = (D-d)(1+e_4), \qquad \text{fl}(D+d) = (D+d)(1+e_5),$$
$$\widehat{S} = (D-d)(D+d)(1+e_4)(1+e_5)(1+e_6) = S(1+3e_7). \tag{3.16}$$

Thus,

$$\frac{\widehat{S}-S}{S} = 3e_7$$

and

$$\left|\frac{\widehat{S}-S}{S}\right| \le 3\eta,$$

a very small relative error, regardless of the values of D and d. Thus, formula (3.14) is unconditionally stable, whereas the degree of stability of formula (3.13) depends on the values of D and d.

A floating-point error analysis (appendix A) of the remainder of the computation of B using formula (3.5) gives

$$\frac{|\widehat{B}-B|}{B} \le \frac{(D+T)(D+1.5T)}{d^2}\eta \le 5\left(\frac{D}{d}\right)^2\eta, \qquad T = \sqrt{D^2-d^2}. \tag{3.17}$$

This result means that the relative error in the computed value $\widehat{B}$ does not exceed $5(D/d)^2\eta$.

In contrast, a floating-point error analysis in appendix A of formula (3.12) gives

$$\frac{|\widehat{B}-B|}{B} \le 9.5\eta.$$

This bound is independent of D and d. Moreover, it means that at most one decimal digit is lost in the calculation.

For each of 100 values of the quotient d/D, viz., $d/D = i/100$, $i = 1, \ldots, 100$, the Brinell hardness was calculated using the unstable and stable formulae, and the relative error (3.17) formed regarding the stable formula as providing the correct result. Figure 3.4 shows these values of the relative error divided by η.

[5]For the cases of concern here, in which $d \ll D$, the right-most part of expression (3.15) is essentially equal to 2η, marginally *superior* to the formula below (with 3η), which applies for *all* D and d.

It also shows as solid curves the relative error bounds, also divided by η, in the basic formula (3.5). The bounds are pessimistic quantitatively, but qualitatively have the 'shape' of the observed errors. The broken curve is the rule-of-thumb estimate of the floating-point error (section 2.6), i.e., corresponding to $\{(D + T)(D + 1.5T)\}^{1/2}\eta/d$. It is not an unreasonable estimate of the floating-point errors that occurred, although these errors have structure and are evidently not 'random'.

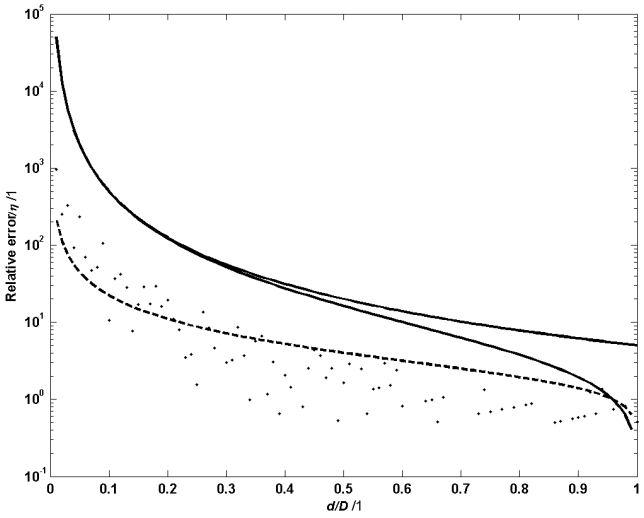

Figure 3.4: The relative error, normalized by the unit roundoff, in the basic formula (3.5) for Brinell hardness B as a function of the quotient of the indentation diameter d and the indenter diameter D. The dots show the relative error, using the stable formula as a reference, for 100 values of the quotient. The solid curves correspond to the bounds (3.17) for this relative error obtained using floating-point error analysis. The broken curve corresponds to the 'rule of thumb' estimate.

3.2.7 Lessons learnt

- Formulae can suffer subtractive cancellation that can compromise the results delivered (also see chapter 3)
- The advantage of a mathematically equivalent formula with superior numerical properties
- The use of floating-point error analysis to expose the difficulty and to provide error bounds for the use of the formulae that involve the problem parameters
- The extent to which the error bounds are realistic
- For almost any of the values of d and D likely to arise in practice, any of the above formulae should be adequate for practical purposes when the calculation is carried out using IEEE arithmetic or something comparable.

This is the case because the arithmetic operates with a sufficient number of significant digits *for this particular computation.* If, however, intermediate results are recorded to a limited number of significant digits, and then subsequently used for the remainder of the calculation, the result may not have the numerical accuracy expected. In any case it is *good practice* to use a stable formula. Then it can be used for all feasible values of its 'parameters' without undue concern. Moreover, recording intermediate results to limited precision would be less dangerous.

3.3 CASE STUDY. Interlaboratory comparisons

3.3.1 Objective

The determination of the total median as a location parameter for key comparison measurements.

3.3.2 What this case study illustrates

- The computational inefficiency arising from the use of full enumeration to provide the probabilities in a formula defining the total median
- The dangers of overflow and underflow in the direct use of published formulae for these probabilities
- The dangers of subtractive cancellation in forming these probabilities from these formulae
- The use of a recurrence relation and symmetry to overcome these three difficulties

3.3.3 The practical problem and requirements

In an interlaboratory key comparison of measurement standards it is required [6] to provide a key comparison reference value (KCRV) and unilateral and bilateral degrees of equivalence based on the measurements and the associated uncertainties provided by the participating national measurement institutes (NMIs). In the simplest type of key comparison [22] a single artefact, regarded as having temporal stability, is measured by the NMIs, and the measurements they provide are taken as statistically mutually independent. For a consistent set of measurements and associated standard uncertainties, a classical location parameter (the weighted mean has been recommended [22]) is used as the KCRV. When a test for consistency of the data and the estimated location parameter fails, other location parameters that are less influenced by discrepant data can be considered [22]. Among such estimators are the median and the total median. Neither of these uses the provided uncertainties, and thus may be more appropriate when those uncertainties are not all credible. The provided uncertainties are, however, used in forming the required degrees of equivalence.

3.3.4 The total median

Let $x_{(1)}, \ldots, x_{(n)}$ denote a sample $x_1, \ldots, x_n$ arranged in non-decreasing order. The *median* is given by

$$\theta = (x_{(\lfloor (n+1)/2 \rfloor)} + x_{(\lfloor (n+2)/2 \rfloor)})/2,$$

where $\lfloor r \rfloor$ is the largest integer no smaller than r.

The *total median* is

$$\theta = \sum_{i=1}^{n} p_i x_{(i)}, \tag{3.18}$$

where the probabilities p_i are derived from properties required of the estimator. It is an estimator of the population mean that retains the robustness property of the median, but has a smaller mean-squared error [35].

The total median θ is defined [35] as the mathematical expectation of the median according to the bootstrap. A bootstrap sample is $x_{i_1}, \ldots, x_{i_n}$, where the i_j are uniformly random integers between 1 and n. Thus, the mathematical expectation is the arithmetic mean of all n^n distinct such samples, there being n possible values for each of the n indices i_j. Although the bootstrap is generally regarded as an infinite algorithm, terminated to provide results to some (stochastic) degree of accuracy, it has an exact finite interpretation for the median (and for certain other order statistics).

The *total bootstrap* is defined as drawing, with replacement, all these n^n essentially distinct bootstrap samples from the given sample. Consider a small sample, viz., of size $n = 3$. There are $3^3 = 27$ such samples, viz., items (1, 1, 1), (2, 1, 1), (3, 1, 1), (1, 2, 1), (2, 2, 1), ..., (3, 3, 3), the probability of occurrence of each of which is identical (= 1/27). Consider calculating the median of each of these samples. From these 27 median values, each of which coincides with one of the x_i, $x_{(1)}$ is selected 7 times, $x_{(2)}$ 13 times and $x_{(3)}$ 7 times. Since the expectation is the sum of products of outcome and the probability of that outcome, the probabilities in this case are taken as 7/27, 13/27 and 7/27, and the total median is

$$\theta = \sum_{i=1}^{3} p_i x_{(i)} = \frac{7}{27} x_{(1)} + \frac{13}{27} x_{(2)} + \frac{7}{27} x_{(3)}.$$

In general, the probabilities p_i in expression (3.18) can be formed by full enumeration.[6]

3.3.5 Full enumeration to calculate the probabilities

The probabilities can in principle be calculated by counting for small values of n. However, the value of n^n increases too rapidly with n to permit full enumeration to be used for n much greater than about 10, for which $n^n = 10^{10}$. Random re-sampling using a more modest, say 10^6 samples, can be used to estimate the probabilities in general. The situation is unsatisfactory. At one extreme, a prohibitive amount of computation is required. At the other, the probabilities are calculated only approximately, with no guarantee of their degree of numerical accuracy.

[6] For n odd, p_i is the probability that $x_{(i)}$ is the median of a bootstrap sample (of size n with replacement) taken from the given sample, with an analogous statement for n even.

3.3.6 The use of an explicit formula

A formula [35],[37, p16],[51] has been provided for the probabilities in expression (3.18). The probabilities depend (only) on n and are given by

$$p_i = \sum_{j=0}^{\lfloor n/2 \rfloor}{}' \{B(j;\ n,\ (i-1)/n) - B(j;\ n,\ i/n)\}, \tag{3.19}$$

where

$$B(j;\ n,\ p) = {}^nC_j p^j (1-p)^{n-j} \tag{3.20}$$

is a binomial probability, and the prime on the summation symbol indicates that the last term is to be taken with weight one-half if n is even. The binomial probabilities are in principle easily calculated (but see below). The computational complexity is much improved, however, being $O(n^2)$, or even $O(n^3)$, depending on how the binomial probabilities are calculated (rather than $O(n^n)$ for full enumeration).

However, for large values of n, ${}^nC_j$ can be very large and $p^{n-j}(1-p)^j$ very small, although their product is of moderate size. There is thus a danger of overflow or underflow in the direct use of the formula.

For values of n greater than 41, the computed values of some of the probabilities are negative! The use of the formula for combining binomial probabilities to produce the probabilities in the formula for the total median is such that for values of i less than $n/2$, especially those for small i, the magnitudes of some of the (positive and negative) contributions to the sum can far outweigh the value of p_i. The effect becomes more pronounced as n increases. This effect is a further instance of damaging subtractive cancellation.

A concern is that if formula (3.19) fails catastrophically for $n > 41$, how reliable are the values of the p_i computed using it for $n \leq 41$? Although they are positive, to what extent are they accurate?

For larger n, there is the additional problem that overflow can occur when calculating the binomial coefficients $B(j;\ n,\ p)$, and no result is returned. This may be a good thing of course, being arguably superior to providing an erroneous result.

3.3.7 The use of a recurrence relation and symmetry

The use of a recurrence relation and symmetry overcomes all three difficulties (the expense of full enumeration, overflow and underflow dangers, and subtractive cancellation):

1. Formula (3.19) gives an economical calculation.

2. The simple recursion formula

$$B(j;\ r,\ p) = pB(j-1;\ r-1,\ p) + (1-p)B(j;\ r-1,\ p) \tag{3.21}$$

   generates binomial probabilities.[7] There is no growth: $B(j;\ r,\ p)$ is a *convex combination* of $B(j-1;\ r-1,\ p)$ and $B(j;\ r-1,\ p)$, since their multipliers p and $1-p$ sum to unity.

[7] The recursion is proved by substituting the right-hand side of expression (3.20) into the formula (3.21) and simplifying the result.

3. Since the set of values of p_i forms a symmetric sequence, only the lower 'half', which can be formed stably, need be evaluated and 'reflected' about the 'midpoint'.

3.3.8 Lessons learnt

- The use of a calculation method based on full enumeration is tractable for small problems (small sample sizes in this case), but computationally prohibitive for large problems.
- The direct use of a formula for the total median probabilities can give rise to quantities that overflow and others that underflow. In particular, the products of such quantities can be of modest size, but evaluating these products needs to be accomplished by other means.
- Subtractive cancellation in the use of the formula can impair the quality of some of the results obtained using the formula. In particular, spurious 'negative probabilities' can be produced.
- The use of a simple recurrence relation avoids the difficulties. Overflow and underflow will not occur, because intermediate quantities are comparable in size. The use of symmetry in the required results avoids the cancellation difficulty.

3.4 CASE STUDY. Extraction of mass of material

3.4.1 Objective

The determination of the total mass of extractable material given measurements of the mass extracted at various times.

3.4.2 What this case study illustrates

- Summation of an infinite series to a required numerical accuracy
- Optimization with respect to linear and non-linear parameters
- Accounting for structure in a non-linear problem
- Finding the globally best solution
- Determination of the parameters of a physical model.

3.4.3 The practical problem and requirements

Heterogeneous phase extraction is critical for many chemical measurements, and is a common feature of organic and other analyses. A paper on recent activity in the area is available [38].

The area presents difficulties because of the lack of sufficiently general physical models of extraction. A simple 'hot-ball' diffusion model that assumes uniform spherical particles of radius r with constant diffusion coefficient D and

essentially zero concentration and high solubility in the surrounding fluid is used as a basis for the study here. The model relates the mass $m = m(t)$ of the material extracted to time t. By fitting the model to measurements of m at a sequence of time values, an estimate of m_0, the total mass of extractable material can be obtained.

3.4.4 The diffusion model and data

The hot-ball diffusion model for a uniform sphere of initial mass m_0 and radius r is

$$m(t) = m_0 S(\mu t), \qquad S(\lambda) = 1 - \frac{1}{\pi^2} \sum_{k=1}^{\infty} \frac{1}{k^2} e^{-\lambda k^2}, \tag{3.22}$$

where

$$\mu = \pi^2 D / r^2,$$

$m(t)$ is the mass extracted after time t, and D is the diffusion coefficient.

The data consists of pairs $(t_i,\ m_i)$, $i = 1, \ldots, N$, where m_i is a measurement of the mass extracted up to time t_i.

3.4.5 Evaluating the model

In fitting the model (3.22) to such data, and in computing model values at various times t for given values of μ (or r and D) and m_0, it is necessary to decide how many terms should be included in the sum in formula (3.22) in order to approximate the infinite series sufficiently well.

One way to 'sum' the series would be to add successive terms until a term was reached that was smaller than a prescribed absolute precision δ. However, this approach provides no guarantee of delivering a solution to that precision. To illustrate this point, consider the sum in expression (3.22) evaluated at $t = 0$, viz.,

$$S = \sum_{k=1}^{\infty} \frac{1}{k^2}. \tag{3.23}$$

Suppose a result correct to a numerical precision of two decimal places is required, i.e., the absolute error is to be no greater than $\delta = 0.005$. The smallest value of k such that $1/k^2 \leq 0.005$ is 15, since $1/15^2 = 0.0044$, whereas $1/14^2 = 0.0051$. Thus, the series (3.23) would be replaced by

$$S = \sum_{k=1}^{K} \frac{1}{k^2} \tag{3.24}$$

with $K = 15$. The value of this sum is 1.58, to two decimal places, whereas the correct value is 1.64, and so in error by 0.06, an order of magnitude bigger than the required precision. The approach is therefore inadequate. A correct approach would be to sum the series in such a way that the *sum of the remaining terms* is no greater than 0.005.

It is shown in appendix B that if an absolute numerical precision of δ is required in the numerical value of $m(t)/m_0$, expression (3.22) can be replaced by

$$m(t) = m_0 S_K(\mu t), \qquad S_K(\lambda) = 1 - \frac{1}{\pi^2} \sum_{k=1}^{K} \frac{1}{k^2} e^{-\lambda k^2}, \tag{3.25}$$

where for any t the number K of terms in the sum is chosen to be the smallest integer satisfying

$$\frac{1}{K+1/2}e^{-\lambda(K+1/2)^2} \leq \delta. \tag{3.26}$$

Thus, the sum in expression (3.25) is calculated by incorporating successive terms until an index $k = K$ is reached satisfying inequality (3.26).

The result (3.26) reduces to $1/(K+1/2)) \leq \delta$ when $t = 0$, giving $K = 200$ to ensure that $\delta \leq 0.005$. Thus, 200 rather than (the inadequately established) 15 terms are needed. Note that $t = 0$ is the 'worst case' in terms of the rate of convergence of the series. This statement is a consequence of the fact that the larger the value of t the faster the series converges because of the exponential term in expression (3.22). The sum (3.23) is in fact $\pi^2/6 = 1.644\ 934\ 066\ 848\ldots$ [44, p18]. As pointed out by Higham [44, p19], it is far preferable to sum a convergent series 'backwards', because adding smaller and smaller numbers to the partial sum fails to account for their full numerical precision, many of the digits 'falling off the end'. He states that summing the series (3.24) in FORTRAN 90 in single-precision with $K = 4096$ (the first value of k for which the sum changes no more when adding successive terms) gives the value 1.644 725 32, which agrees with the correct value to only four significant digits from a possible nine. Summing these 4096 terms backwards gives the improved value 1.644 934 06, correct to eight significant digits.

In the context of this application, a reliable way to proceed is to evaluate the left-hand side of inequality (3.26) for successive values until a value of K was reached that satisfied the inequality. Then the series would be summed backwards from this value of K.[8]

The approach of appendix B can be followed for certain other series. However, there is no approach that is generally applicable, a case-specific analysis generally being required to support the choice of the number of terms to be included.

3.4.6 Fitting the model

The problem to be solved is formulated as one of least squares:

$$\min_{m_0,\mu} \sum_{i=1}^{N} (m_i - m(t_i))^2.$$

There are two adjustable parameters, m_0 and μ. The fitting of $m(t)$ to data is an instance of a non-linear least-squares optimization problem [2].

The hot-ball diffusion model is intrinsically non-linear in (one of) its parameters, and consequently some numerical methods may have difficulty in locating the (globally) best solution. An approach that avoids this difficulty uses the fact that there is just one non-linear parameter in the model.

Consider first 'traditional' methods for solving a non-linear least-squares problem [2]. Such a problem can be described as follows. Let $f_i(\mathbf{a})$, $i = 1, \ldots, N$, denote a set of N functions of n ($\leq N$) parameters $\mathbf{a} = (a_1, \ldots, a_n)^{\mathrm{T}}$.

[8] There would be more efficient ways to proceed if computation time was a consideration. One way would be to replace the inequality sign in expression (3.26) by an *equality*, regarding the resulting expression as an equation to be solved for (a generally non-integer value) K', say. The smallest integer greater than or equal to K' is then taken as K.

For the current purposes,

$$f_i(\mathbf{a}) = w_i(y_i - f(t_i; \mathbf{a})),$$

i.e., the weighted difference (residual) between a data ordinate value y_i and a model $f(t; \mathbf{a})$ at the corresponding abscissa value $t = t_i$. w_i denotes the weight, which would normally be chosen equal to the reciprocal of the standard uncertainty associated with y_i. The t_i are taken as accurately known. A modification, not required here, can be used to handle the case where there is uncertainty associated with the t_i [27].

The problem is to determine values $\widehat{\mathbf{a}}$ of $\mathbf{a}$ such that the sum-of-squares function (weighted residual sum of squares)

$$\phi(\mathbf{a}) = \sum_{i=1}^{N} f_i^2(\mathbf{a}) \tag{3.27}$$

is a minimum. A straightforward extension permits an uncertainty matrix associated with the data values y_i to be incorporated [2].

Most numerical approaches for solving this problem start with an initial approximation $\mathbf{a}_0$ to $\widehat{\mathbf{a}}$ and determine a sequence of approximations (iterates) $\mathbf{a}_r$, $r = 1, 2, \ldots$ to $\widehat{\mathbf{a}}$. A test for convergence is applied to each approximation so generated, and the process terminated when the test is satisfied.

A basic solution procedure is the Gauss-Newton algorithm. At the $(r+1)$st iteration, $r = 0, 1, \ldots$, this algorithm uses, in place of the $f_i(\mathbf{a})$, linear approximations to these functions. These approximations are given by the Taylor series expansions of first order of the $f_i(\mathbf{a})$ about $\mathbf{a} = \mathbf{a}_r$. The resulting sub-problem is one of linear least squares (a sum of squares of linear functions) that can be solved by standard methods of linear algebra. Full details of recommended methods are available [2], [44, chapter 19].

An initial approximation to the adjustable parameters is often available from the physical context of the application. Non-linear least-squares algorithms typically require the user to specify the partial derivative of first order of the model function, i.e., with respect to the adjustable parameters. These algorithms solve a sequence of linearized problems until (generally) the solution is obtained [2].

3.4.7 Accounting for problem structure

Consider a model containing one non-linear parameter and $n - 1$ linear parameters. An example of such a model, in which $n = 3$, is

$$a_1 + a_2 e^{-a_3 x}. \tag{3.28}$$

The parameters a_1 and a_2 occur linearly and a_3 non-linearly.

Problems involving such a model can be solved more readily than when more general models are involved. A reason is that they can be reduced to problems involving just the non-linear parameter. A consequence is that a detailed examination of the space of that parameter will permit the globally best solution to be found. Generally, the Gauss-Newton and the Levenberg-Marquardt algorithms applied to the n-parameter problem will provide a local solution. If the initial approximation to the parameter values is sufficiently close to the solution values, the required solution will be obtained. Details of

these and other procedures for linear and non-linear least-squares modelling problems are available [2]. For numerical stability these procedures work with the Jacobian matrix[9] J at each iteration, solving the resulting linear system using the QR or SVD algorithm [44].

The approach is discussed through the use of extraction information typified by the data [38] of figure 3.5. Consider a particular numerical value for the non-linear parameter a_n. Because all the remaining parameters occur linearly, the problem can be solved for them using linear least squares. This linear problem generally has a unique solution. This solution yields the smallest value of the residual sum of squares ϕ *given* the chosen value of a_n. Since in this setting the linear parameters depend on a_n, they can be written as functions of a_n, viz., as $a_j(a_n)$, $j = 1, \ldots, n-1$. Hence, the weighted residual sum of squares ϕ (3.27) can be (re-)expressed (not necessarily explicitly) as a formula in terms of the single parameter a_n.[10]

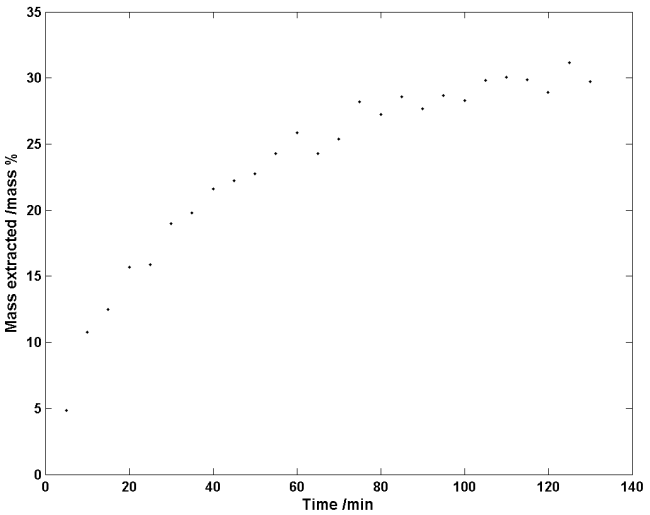

Figure 3.5: Extraction data.

Suppose a different choice were made for a_n, and the values of the linear parameters and the residual sum of squares formed for this choice. Consider this process performed for a large number of choices of a_n. In particular, suppose an interval for a_n were chosen in which it was expected that its optimal value would lie. Partition this interval into a large number M of points (1000, say) and carry out the process for all such values. Express the result as M pairs of values of a_n and the corresponding residual sum of squares. A graph of these M points would show the residual sum of squares as a function of a_n. In particular, if the interval chosen contained the optimal value of a_n, the function underlying these points would have a minimum within the interval. Figure 3.6 shows the

[9] J is the matrix of partial derivatives of the model with respect to the model parameters, evaluated at the current approximation to the values of these parameters.

[10] Generally, ϕ is a function of (or formula involving) the n model parameters.

graph of the residual sum of squares as a function of a_3 for the exponential model (3.28) applied to the data of figure 3.5. (The use of the actual physical model is considered below.)

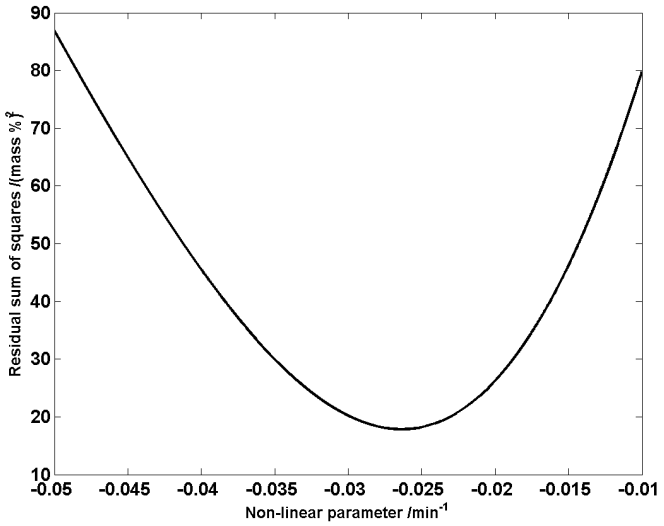

Figure 3.6: Residual sum of squares as a function of the (non-linear) rate constant for the extraction data of figure 3.5.

There may be more than one minimum, as can arise for a non-linear problem, in which case the minimum of concern would be the global minimum, i.e., that corresponding to the smallest residual sum of squares in the interval.

If the interval contained no minimum, i.e., the residual sum of squares was increasing or decreasing throughout the region, the selected interval is clearly inappropriate and another should be chosen. Even if the selected interval contained one or more minimum, there remains a possibility that the global minimum lay outside the interval. The choice of the interval should be made by a practitioner with knowledge of the area. The choice could perhaps be automated, should sufficient information be made available to an algorithm. In any case, it is desirable for the practitioner to be 'in the loop', especially to examine a graph of the behaviour of the residual sum of squares as a function of the non-linear parameter. Such knowledge can be informative in the context of the application.

Once an interval containing a minimum had been identified, the pair of points from the M are selected that are as close as possible to each other and that bracket the minimum. These points lie immediately to the left and the right of the point for which the residual sum of squares is smallest among the M values. Then, a 'bracketing' algorithm can be applied to minimise to the required numerical precision the residual sum of squares as a function of a_n. High-quality algorithms are available [36] for this purpose.

The commonest and arguably the most reliable bracketing algorithm is the *bisection algorithm*. See section 2.5.

It would be unnecessary to apply this bracketing algorithm if the length of the sub-interval defined by the chosen pair of points was less than the numerical precision required in a_n. Such an occurrence would be the consequence of selecting an initial interval that contained the minimum sought. For any initial interval, there would be a value of M such that the resulting sub-interval (which would have length a factor $2/M$ of the original interval) was less than any required numerical accuracy.

Once the minimum had been obtained to the required numerical precision, the resulting value of a_n and the corresponding values of the linear parameters represent the required set of parameter values. Figure 3.7 shows the exponential model (3.28) obtained this way for the data of figure 3.5.

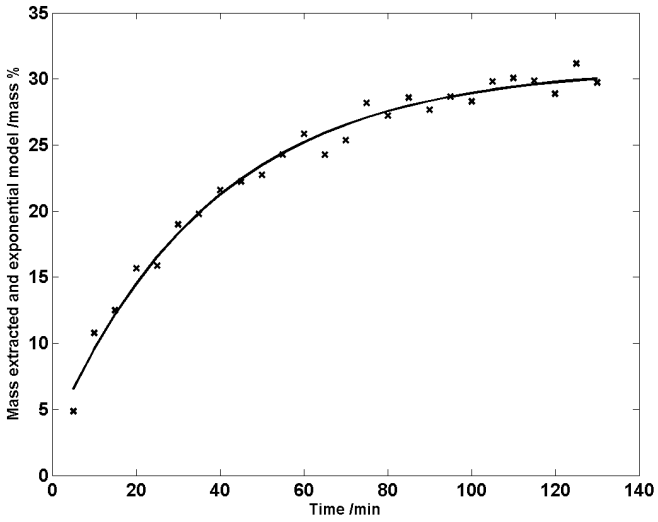

Figure 3.7: The exponential model with optimised rate constant for the extraction data of figure 3.5.

3.4.8 Lessons learnt

- The 'natural' way to sum a series may not yield a stipulated accuracy. An example that makes the point very clearly [40] is as follows. In the summation of the series

$$S = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots,$$

it is perfectly correct to stop when the next term is smaller than the tolerable error, because the truncation error is smaller than the first neglected term. This stopping rule would fail catastrophically for the apparently similar series

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots,$$

whose sum is infinite!

- Providing a bound for the remaining terms in a series and using this bound to decide when to terminate the summation provides a basis for summing the series to a prescribed numerical precision. Values of this sum can be used in various ways, e.g., as here, as part of the iterative solution (the problem Jacobian matrix depends on such values, and the accuracy of the elements of this matrix will influence the convergence of the iteration).
- Conventional algorithms for non-linear least-squares problems may not yield global solutions
- Accounting for problem structure can provide robust, global solutions. It also shows how the quality of the solution (in terms of residual sum of squares) depends on the value of the non-linear parameter of the problem. Generally, taking such account can be valuable in terms of efficiency of solution and, more importantly, often reducing the dimensionality and hence improving the understanding of a problem.

Chapter 4

Differentiation

The need for derivatives (this section) and integrals (section 5) arise frequently in metrology.

The modelling and analysis of non-linear problems usually requires the calculation of the partial derivatives of the component functions with respect to the problem variables. For all but the simplest models, these calculations are time-consuming and error-prone. Automatic differentiation (AD) techniques aim to provide efficient numerical evaluation of the derivatives of the component functions solely on the basis of software to evaluate the component functions. Descriptions of the main AD techniques and a summary of their advantages and disadvantages are available [10, 11]. Of particular interest is the complex-step method (applied indicatively in section 2.4), which uses complex arithmetic to evaluate derivatives. It is particularly simple to implement in software languages such as FORTRAN 90 and MATLAB that support complex arithmetic. Also of value is finite-difference formulae, because of their relative ease of implementation, but there are risks associated with their use.

Consider the evaluation at a point x_0 of the derivative with respect to x of an analytic function $f(x)$. Such derivatives are required in many contexts. One of the most important is in scientific computation when it is required to form the $m \times n$ *Jacobian matrix* J having $\partial f_j/\partial x_k$ as element j, k. J arises for example in non-linear least-squares regression in minimizing a residual sum of squares. Then, $f_j(\mathbf{x})$ is the jth of m residuals, and x_k is the kth of n variables or parameters $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$ on which the residuals depend. The need for J also arises in sensitivity analyses, uncertainty evaluation and simulation.

Such an evaluation can be carried out in several ways, including

1. Coding by hand the analytic derivative $f'(x)$ and evaluating it at x_0,

2. Using a symbolic differentiation package (such as Maple [14]) to form the analytic derivative $f'(x)$ and evaluating it at x_0,

3. Using a numerical differentiation package (such as ADIFOR [9]) to provide $f'(x_0)$,

4. Applying an appropriate finite-difference approximation (see, e.g., [42, p339]) to estimate $f'(x_0)$ using values of $f(x)$ in the neighbourhood of $x = x_0$.

With the exception of the last approach all these approaches will if correctly used provide the required value $f'(x_0)$ to good accuracy. The term 'good accuracy' is deliberate. There is no guarantee that the representation of the derivative will be such that its evaluation will not cause problems such as damaging subtractive cancellation. However, the evaluation is essentially of an *analytic* form rather than involving a numerical approximation, and is thus potentially capable of providing a good result. The hand-coding approach is of course error-prone (and for this reason often checked using finite differences). In a number of circumstances it is not convenient to utilize a symbolic or a numerical differentiation package, particularly because a compact and self-contained code is desired. Therefore, the concentration here is on the use of finite differences and approaches related to them.

4.1 Finite differences and related approaches

Typical of finite-difference formulae are the forward-difference formula [42, p339]

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h} \tag{4.1}$$

and the central-difference formula [42, p340]

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}. \tag{4.2}$$

Here h is a 'step' selected in an appropriate way. It is typically chosen to balance truncation error (the error given by ignoring higher-order terms in the Taylor-series expansion from which the formula is derived—cf. section 2.1.1) and subtractive-cancellation error (the error incurred when forming differences between function values at closely-neighbouring points). The truncation error in (4.1) is $h|f''(\xi_1)|/2$, where $\xi_1 \in [x_0,\ x_0 + h]$, and that in (4.2) is $h^2|f'''(\xi_2)|/6$, where $\xi_2 \in [x_0 - h,\ x_0 + h]$. For appropriately scaled problems [42, p341], an optimal choice of h would be proportional to $\eta^{1/2}$ for (4.1) and for (4.2) proportional to $\eta^{1/3}$. Here, η is the unit roundoff defined in section 2.6. For such problems, the use of (4.1) can be expected at best to lose 'half a wordlength of accuracy' and the use of (4.2) to lose one-third of the available figures. Not only can this loss of accuracy prove troublesome, particularly in its influence on the convergence rates of iterative methods for solving non-linear least-squares regression problems, e.g., but determining a value of h in order to achieve the above 'best accuracies' is not trivial.

Recently, Squire and Trapp [54] have revisited work by Lyness and Moler [50] in which a result from complex-variable theory is used to approximate derivative values. The requirement is that the programming system in which the derivatives are to be computed supports complex types, and in particular the basic functions such as square root, sine and exponential are available for complex as well as real arguments. FORTRAN 77, FORTRAN 90 and MATLAB are examples of such programming systems.

Consider the evaluation of $f(x + iy)$, where $i = \sqrt{-1}$. Suppose that this function is evaluated at a point where $x = x_0$ and y is small. It is to be expected that as $y \to 0$, $\Re f(x_0 + iy) \to f(x)$ and $\Im f(x_0 + iy) \to 0$, and indeed

this is the case. However, the *manner* in which $\Im f(x_0+iy) \to 0$ is very pertinent to this discussion. Replace y by h and consider

$$\lim_{h\to 0} \frac{\Im f(x_0+ih)}{h}. \tag{4.3}$$

It is straightforward to show by Taylor-series expansion that, after taking imaginary parts,

$$\frac{\Im f(x_0+ih)}{h} = f'(x_0) - \frac{h^2}{6} f'''(x_0) + O(h^4) \tag{4.4}$$

and hence that

$$f'(x_0) = \frac{\Im f(x_0+ih)}{h} + \frac{h^2}{6} f'''(x_0) + O(h^4). \tag{4.5}$$

It therefore follows that $f'(x_0)$ can be estimated by choosing a 'sufficiently small' value of h and forming

$$f'(x_0) \approx \frac{\Im f(x_0+ih)}{h} \tag{4.6}$$

(also see section 2.4). Unlike the use of a finite-difference formula, h can be chosen to be *very* small with no concern about the loss of significant figures. The only restriction is that h must not be chosen so small that it (or the terms involving it) underflows, i.e., in floating-point arithmetic is replaced by zero. The value $h = 10^{-100}$ is used in NPL software, which is suitable for all but pathologically scaled problems.

The result (4.6) should be compared with the Taylor-series expansion from which approximation (4.2) is derived, viz.,

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} - \frac{h^2}{6} f'''(x_0) + O(h^4). \tag{4.7}$$

In place of the central difference, $\Im f(x_0+ih)/h$, a quantity involving no subtractive cancellation, is used. Moreover, the dominant error term, that involving h^2, is equal and opposite in the two cases. An interesting observation therefore is that in exact arithmetic the arithmetic mean of the formulae (4.2) and (4.6) is accurate to $O(h^4)$.

The success of this approach, i.e., the use of approximation (4.6), depends on the integrity of the built-in complex-valued functions. Many scientific computations rely on their integrity for real arguments and a significant number, particularly in electrical science, on the same for complex-valued arguments.

To summarise, if a scientific computation requires the value of $f'(x_0)$, it can be obtained by

1. Setting h to be a small number (10^{-100} is suggested), but this value could be taken as a default that can be overridden if necessary),
2. Forming $f(x_0+ih)$,
3. Taking $f(x_0) = \Re f(x_0+ih)$,
4. Taking $f'(x_0) = \Im f(x_0+ih)/h$.

Step 3 can of course be omitted, but provides $f(x_0)$ at little extra cost.

A simple illustration of the use of this approach was given in section 2.4. It and classical finite differences are used in the case study in section 4.2.

4.2 CASE STUDY. Determination of DNA concentration

4.2.1 Objective

The calibration of a measurement system for determining DNA concentration and its use to provide the concentrations of DNA for a number of samples.

4.2.2 What this case study illustrates

- The determination of a calibration curve from response measurements for a set of standards
- The mathematical representation of the calibration curve
- The use of the calibration curve to provide concentrations from response measurements for a set of samples
- The numerical propagation of uncertainties through the calibration curve model
- The development of algebraic expressions for the various uncertainty components as part of a formal 'uncertainty budget'
- The use of finite differences and automatic differentiation to validate the results.

4.2.3 The practical problem

PCR (Polymerase Chain Reaction) has the ability to replicate large quantities of a specific DNA species from a small starting amount. Fluorescence probes are used to monitor the accumulation of PCR products in real time. Two probes are used: one probe is labelled with a fluorescent dye that hybridises to the endogenous gene, and the other with a fluorescent dye that hybridises to the transgene.

A fluorescence baseline is established above which DNA amplification is regarded as taking place. Instrument-based software assigns a C_t (Cycle Threshold) value, the cycle number where the amplification curve and fluorescence baseline meet.

The response is the ΔC_t value,[1] the difference between the C_t values of the transgene and the endogenous gene.

Further steps in the analysis that are carried out by instrument-based software involve

1. Formulation of a calibration curve, obtained by applying straight-line regression to the logarithm of the specified DNA concentrations of the standards and the corresponding mean ΔC_t values.
2. Calculation of the unknown DNA sample concentrations from the calibration curve based on given mean ΔC_t values.

[1] Strictly, the ΔC_t value is a mean ΔC_t value, because of the repeated measurements on which it is based.

The instrument-based software may not evaluate valid uncertainties. In particular, it may not make use of all the available uncertainty information. Consideration may not be given to uncertainties associated with measurement data when establishing the calibration curve. The curve would typically be obtained simply by minimizing the sum of squared differences between the measurement and corresponding fitted model response values (ordinary least-squares regression), even though both the response and stimulus values in general have different associated uncertainties.

The uncertainties so obtained would be valid *for the manner in which the model parameters were obtained*. A solution that respected the actual uncertainty structure of the data should ideally be used [27], but such a solution is not implemented in the instrument manufacturer's software.

The problem is to propagate uncertainties that have been obtained from the supplier of the DNA standards and from replicate response measurements through the calibration curve-fitting process. The result will be uncertainties associated with the calibration curve parameters that can subsequently be used as a basis for evaluating the uncertainties associated with the concentrations of unknown samples when the calibration curve is subsequently used to estimate these concentrations.

4.2.4 Formulating the problem mathematically

The data consists of points (t_i, y_i), $i = 1, \ldots, m$, where t_i and y_i denote respectively estimates (measurements) of the values of the concentration of the ith DNA standard and the corresponding cycle number (the mean ΔC_t value corresponding to t_i). Associated with the t_i and the y_i are standard uncertainties $u(t_i)$ and $u(y_i)$. A typical data set is shown in table 4.1 and figure 4.1.

	DNA concentration standard		Response to standard	
i	t_i	$u(t_i)$	y_i	$u(y_i)$
1	0.100	0.025	10.58	0.11
2	0.500	0.050	8.19	0.11
3	1.000	0.100	6.74	0.11
4	2.000	0.150	5.63	0.11
5	5.000	0.300	4.38	0.11

Table 4.1: DNA concentration standards, the corresponding responses $y_i \equiv (\text{mean } \Delta C_t)_i$, and the associated standard uncertainties.

The instrumental response y to DNA concentration t is observed to be a straight line in $\log_{10} t$. Thus, the calibration curve can be expressed as

$$y - \bar{y} = b(x - \bar{x}), \qquad x = \log_{10} t, \tag{4.8}$$

where $(\bar{x}, \bar{y})$ denotes a point on the curve and b the gradient of the curve. This form is just one of several parametrizations of the curve. It is a sound one because it avoids unnecessary damaging cancellation effects that could occur from the use of a form such as that involving an intercept and a gradient.[2]

[2] A form such as the representation (4.8) is good numerical practice, but for the particular data in table 4.1 this is issue is of little concern.

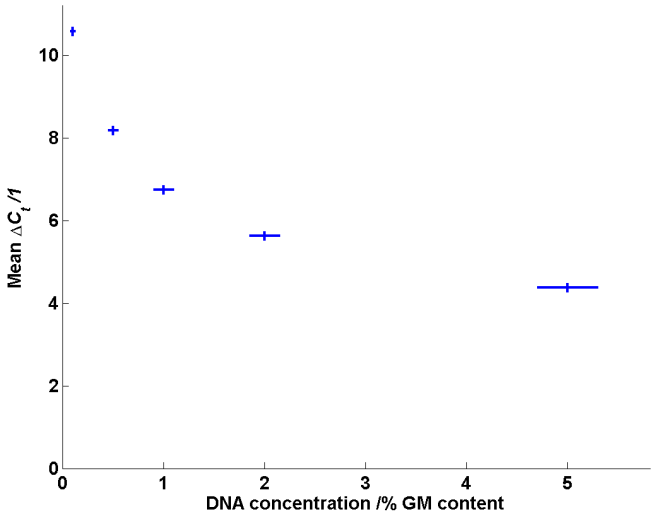

Figure 4.1: Measurements of mean ΔC_t for five DNA concentration standards and the associated standard uncertainties. The centre of each cross indicates a data point (t_i, y_i), the length of the horizontal arm of the cross is $2u(t_i)$ and that of the vertical arm $2u(y_i)$.

4.2.5 Obtaining the calibration curve

Let $\widehat{b}$ denote the unweighted least-squares estimate of b. The unweighted least-squares solution [53, p 514] is

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i, \qquad \bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i, \tag{4.9}$$

identical to the Cartesian co-ordinates of the centroid of the data, and

$$\widehat{b} = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{m}(x_i - \bar{x})^2}, \tag{4.10}$$

giving, for the above data, the calibration curve shown in figure 4.2.

4.2.6 Propagating the uncertainties

Since for this study a 'formal uncertainty budget' was required, algebraic expressions were derived for the standard uncertainties and covariances associated with estimates of the values of the model parameters. Generically, let $u(\alpha)$ denote the standard uncertainty associated with α and $u(\alpha,\ \beta)$ the covariance associated with α and β. The following expressions were obtained from the application of the law of propagation of uncertainty [7] (only the non-zero covariances are stated):

$$u(x_i) = \frac{u(t_i)}{t_i \log_{10} e}, \qquad u^2(\bar{x}) = \frac{1}{m^2}\sum_{i=1}^{m} u^2(x_i), \qquad u^2(\bar{y}) = \frac{1}{m^2}\sum_{i=1}^{m} u^2(y_i),$$

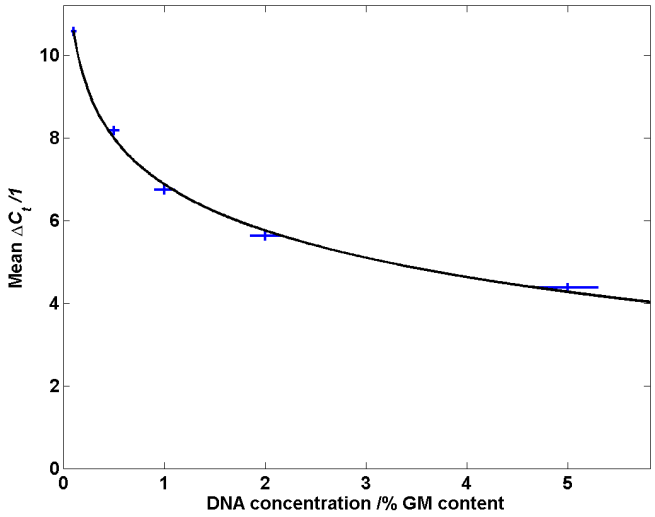

Figure 4.2: The calibration curve for the measurements of mean ΔC_t for five DNA concentration standards, using ordinary least squares.

$$u^2(\widehat{b}) = \frac{1}{S_{xx}^4} \sum_{i=1}^{m} \left\{ T_i^2 u^2(x_i) + (x_i - \bar{x})^2 S_{xx}^2 u^2(y_i) \right\}, \tag{4.11}$$

$$u(\bar{x}, \widehat{b}) = \frac{1}{mS_{xx}^2} \sum_{i=1}^{m} T_i u^2(x_i), \qquad u(\bar{y}, \widehat{b}) = \frac{1}{mS_{xx}} \sum_{i=1}^{m} (x_i - \bar{x}) u^2(y_i), \tag{4.12}$$

where

$$S_{xx} = \sum_{i=1}^{m} (x_i - \bar{x})^2, \qquad S_{xy} = \sum_{i=1}^{m} (x_i - \bar{x})(y_i - \bar{y}), \qquad S_{yy} = \sum_{i=1}^{m} (y_i - \bar{y})^2, \tag{4.13}$$

$$T_i = (y_i - \bar{y}) S_{xx} - 2(x_i - \bar{x}) S_{xy}. \tag{4.14}$$

These uncertainties are needed when the calibration curve is used to evaluate the standard uncertainties associated with estimates of the values of the DNA concentration of unknown samples.

Table 4.2 shows the values produced using these formulae for the above data. $u(\bar{y}, b)$ is zero because, for $u(y_i)$ that are all equal, as here, the right-hand sum in expression (4.12) becomes

$$u^2(y_1) \sum_{i=1}^{m} (x_i - \bar{x}),$$

which vanishes because of the definition (4.9) of $\bar{x}$.

Given (a) the estimates $\bar{x}$, $\bar{y}$ and $\widehat{b}$ of the values of the calibration curve parameters and the associated uncertainties and (b) a response y_0 and an asso-

$\bar{x}$	$u(\bar{x})$	$\bar{y}$	$u(\bar{y})$	$\widehat{b}$	$u(\widehat{b})$	$u(\bar{x},\widehat{b})$	$u(\bar{y},\widehat{b})$
–0.060	0.026	7.105	0.048	–3.73	0.25	–0.0048	0

Table 4.2: Estimates of the values of the calibration curve parameters $\bar{x}$, $\bar{y}$ and $\widehat{b}$ and the standard uncertainties and the (generally) non-zero covariances associated with these estimates.

ciated uncertainty $u(y_0)$ for an unknown sample, the estimated logged concentration corresponding to y_0 from the calibration curve is

$$x_0 = \bar{x} + \frac{y_0 - \bar{y}}{\widehat{b}} \tag{4.15}$$

and the corresponding concentration is

$$t_0 = 10^{x_0}. \tag{4.16}$$

The standard uncertainty $u(x_0)$ associated with x_0 can be obtained by applying the law of propagation of uncertainty [7, Clause 5.2.2]:

$$u^2(x_0) = u_1^2(x_0) + u_2^2(x_0) + u_3^2(x_0),$$

where

$$\begin{aligned} u_1^2(x_0) &= \sum_{i=1}^{m} \left\{ \frac{1}{m} - \frac{(x_0 - \bar{x})T_i}{\widehat{b}S_{xx}^2} \right\}^2 u^2(x_i), \\ u_2^2(x_0) &= \sum_{i=1}^{m} \left\{ \frac{1}{m} + \frac{(x_0 - \bar{x})(x_i - \bar{x})}{S_{xx}} \right\}^2 \frac{u^2(y_i)}{\widehat{b}^2}, \\ u_3^2(x_0) &= \frac{1}{\widehat{b}^2} u^2(y) \end{aligned} \tag{4.17}$$

are the variances (squared standard uncertainties) associated with the contributions to $u^2(x_0)$ resulting, respectively, from the concentration standards, the system responses to those standards, and the system response to the sample.

Finally, the standard uncertainty associated with the sample concentration t_0 is

$$u(t_0) = u(x_0)10^{x_0} \log_e 10 = t_0 u(x_0)/\log_{10} e. \tag{4.18}$$

Table 4.3 shows the results obtained. The mean ΔC_t values for the samples and the associated standard uncertainties are shown in columns 2 and 3 and the required sample DNA concentrations and the associated standard uncertainties determined from the calibration curve in columns 4 and 5.

Figure 4.3 shows the sample concentrations determined from the calibration curve. In that figure, the semi-widths of the shaded regions are equal to the standard uncertainties associated with the estimated sample DNA concentrations. The gradient of the curve is a major factor influencing these uncertainties. The decreasing gradient of the calibration curve with increasing DNA concentration yields increasingly greater uncertainties. Were the uncertainties associated with the mean ΔC_t values for the samples not identical, there would be a further effect from that source.

	Response		DNA sample concentration	
i	y_i	$u(y_i)$	t_i	$u(t_i)$
6	9.05	0.11	0.262	0.039
7	7.09	0.11	0.878	0.083
8	6.32	0.11	1.411	0.119
9	5.66	0.11	2.130	0.182
10	4.38	0.11	4.676	0.511

Table 4.3: The mean measurement responses $y_i \equiv (\text{mean } \Delta C_t)_i$, and the required DNA concentration values for the samples obtained from the calibration curve, and the associated standard uncertainties. The indices $i = 6, \ldots, 10$ are used to distinguish values from those related to the standards, with indices $i = 1, \ldots, 5$ in table 4.1.

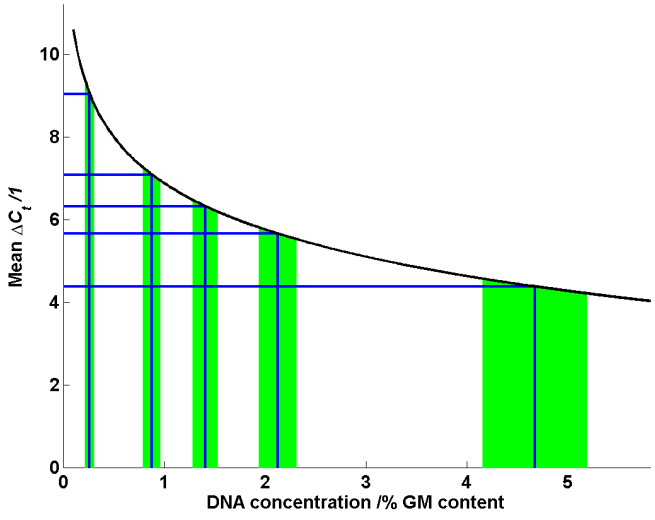

Figure 4.3: Sample concentrations determined from the calibration curve, and the associated standard uncertainties.

4.2.7 Numerical validation of the results

It requires care to produce formulae such as those in section 4.2.6 for the uncertainties associated with estimates of the values of the calibration parameters and the required DNA concentrations. There is a distinct possibility of an algebraic error in deriving such formulae. To validate these particular formulae, the uncertainty matrix

$$V_{\widehat{\mathbf{b}}} = \begin{bmatrix} u^2(\bar{x}) & 0 & u(\bar{x}, \widehat{b}) \\ 0 & u^2(\bar{y}) & u(\bar{y}, \widehat{b}) \\ u(\bar{x}, \widehat{b}) & u(\bar{y}, \widehat{b}) & u^2(\widehat{b}) \end{bmatrix}$$

associated with the calibration curve parameters $\widehat{\mathbf{b}} = (\bar{x}, \bar{y}, \widehat{b})^{\mathrm{T}}$ was evaluated *numerically* for choices of values of the input data, including those in table 4.1. Specifically, if

$$V_{\mathbf{d}} = \mathrm{diag}(u^2(t_1), \ldots, u^2(t_m), u^2(y_1), \ldots, u^2(y_m))$$

denotes the uncertainty matrix associated with the input data

$$\mathbf{d} = (t_1, \ldots, t_m, y_1, \ldots, y_m)^{\mathrm{T}},$$

then

$$V_{\widehat{\mathbf{b}}} = J V_{\mathbf{d}} J^{\mathrm{T}},$$

where J is the (Jacobian) matrix containing the partial derivatives of first order of expressions (4.9) and (4.10) with respect to $\mathbf{d}$ evaluated for the data [32].

J was evaluated using finite-difference approximations (4.1) and (4.2). Care was taken to select sensible step sizes to ensure that the resulting approximations were adequate. The Jacobian matrix was consequently confirmed correct to approximately half of the available digits when the forward-difference formula (4.1) was used and two thirds of the available digits when the central-difference formula (4.2) was used, as predicted by the theory (section 4.1). The complex-step method of section 4.1 was also used for this purpose. Its use confirmed that J was correct in all its elements to all digits or all but one digit. The use of this method required only a fraction of the (human) time compared with the use of the finite-difference formulae, primarily because the 'default' step size of 10^{-100} (section 4.1) proved, as expected, to be perfectly adequate.

The uncertainties associated with estimates of the values of the sample DNA concentrations were confirmed similarly. In that case the standard uncertainty $u(t_0)$ associated with the estimate t_0 of the sample concentration is given by

$$u^2(t_0) = \mathbf{c}^{\mathrm{T}} V_{\widehat{\mathbf{b}}} \mathbf{c},$$

where $\mathbf{c}$ is the vector of partial derivatives of first order with respect to $\mathbf{b}$ of expression 4.16, making use of expressions 4.9, 4.10 and 4.15, evaluated at $\mathbf{b} = \widehat{\mathbf{b}}$.

The complex-step method of AD cannot provide higher-order derivatives, were they required. In such cases, some of the other methods [10] can be used.

4.2.8 Lessons learnt

- A sound parametrization of the calibration curve confers advantages

- Formal uncertainty evaluation can be difficult algebraically
- The use of finite differences or automatic differentiation can be directly helpful in validating algebraic results
- The complex-step method is readily programmed in-line (cf. section 4.1) and can therefore be used without having to integrate the user's software with AD software
- When algebraic derivatives of first order are not specifically required, the complex-step method offers a viable alternative that is efficient and less error-prone.

Chapter 5

Integration

Integration (in one variable) is concerned with the evaluation of the definite integral

$$I = \int_a^b f(x)\,dx$$

or the determination of the indefinite integral

$$F(x) = \int_a^x f(t)\,dt, \qquad a \le x \le b,$$

given values a and b and information about $f(x)$.[1]

Three types of information about $f(x)$ can be considered:

1. $f(x)$ specified mathematically or at least in the form of a procedure for returning the value of f given a value for x in the interval $[a,\ b]$
2. $f(x)$ specified exactly at prescribed points $x_1, \ldots, x_m$, where $a \le x_1 < \cdots < x_m \le b$
3. As type 2, but $f(x)$ specified inexactly as measurements y_i at the points x_i, $i = 1, \ldots, m$.

Most of the literature is concerned with type 1 or 2, the second often corresponding to uniformly spaced points in the interval $a \le x \le b$. Integration problems that fall into these categories can be solved accordingly. The introduction to Chapter D01—Quadrature in the NAG Library provides useful background.[2]

The concentration here is on type 3. It is commonplace in metrology to provide measurements of a function and their associated uncertainties, and it is required to analyse them in a manner required by the application. One such analysis is the determination of I or $F(x)$. There is much less literature on this problem.

Some general remarks are made on determining I or $F(x)$ and on the effects of uncertainties associated with the measurements. Then, specific solutions are provided as part of a case study.

[1] Integration involving more than one variable is not considered here.
[2] www.nag.co.uk

5.1 Motivation

Consider the determination of I when measurements y_i, $i = 1, \ldots, m$, of $f(x)$ are made at a uniform spacing $h = (b-a)/(m-1)$ in $[a,\ b]$, i.e., at abscissae $x_i = a + (i-1)h$. I may be approximated by the formula

$$\widetilde{I} = h \sum_{i=1}^{m-1} \frac{y_i + y_{i+1}}{2}, \tag{5.1}$$

known as the *trapezoidal rule*. This rule corresponds to determining the piecewise-linear function joining the points $(x_1, y_1), \ldots, (x_m, y_m)$ and then forming the area under this function. The partial sum, given by replacing m by r, where $2 \leq r \leq m$, gives an approximation to $F(x_r)$. How good are these approximations? What are the advantage and the effect of using a more sophisticated rule? Two main influences can be considered. One is the use of the approximation $\widetilde{I}$ rather than the exact integral. The other is the effect of uncertainties associated with the y_i. Approaches for general functions of the y_i for handling the second influence are available [7, 32], but are treated specifically here, because there is little point in attempting to evaluate the integral much more accurately than is warranted by the associated uncertainties.

A simple means for validating the use of the trapezoidal rule (and some other rules) is to obtain a further approximation based on the use of every other data point (with appropriate modifications for m even).[3] If the difference is judged negligible compared with the uncertainty associated with $\widetilde{I}$, the trapezoidal rule can be regarded as adequate.[4]

Figure 5.1 shows an example of the application of the trapezoidal rule to a spectral peak. The solid curve depicts the underlying function. The points joined by straight-line segments represent 11 specified values y_i at a spacing of 10 nm in the interval 600 nm to 700 nm. The areas of the ten trapezoids represent the contributions from the trapezoidal rule to the approximation $\widetilde{I}$ to the integral I representing the area under the curve over this interval.

[3] This is a weak form of validation, in that it would be more meaningful to consider the rule for the original spacing as being used to validate the rule applied to half the spacing.

[4] Take the simplest case where the y_i are mutually independent and the standard uncertainty associated with y_i is denoted by $u(y_i)$. The application of the law of propagation of uncertainty [7] to expression (5.1) yields a standard uncertainty $u(\widetilde{I})$ associated with $\widetilde{I}$ given by

$$u^2(\widetilde{I}) = h^2 \left(\frac{1}{4}u^2(y_1) + \sum_{i=2}^{m-1} u^2(y_i) + \frac{1}{4}u^2(y_m) \right),$$

which, in the case where the uncertainties $u(y_i)$ are identical, becomes

$$u^2(\widetilde{I}) = (m - 3/2)h^2 u^2(y_1).$$

This expression becomes more meaningful by setting $h = (b-a)/(m-1)$, whence

$$u(\widetilde{I}) = (b-a)\frac{(m-3/2)^{1/2}}{m-1}u(y_1) \approx \frac{b-a}{\sqrt{m}}u(y_1)$$

for large m. Since the length $b - a$ of the interval of integration is fixed, the uncertainty decreases with the number m of measurements as $1/\sqrt{m}$. Such behaviour is similar to that for the mean of a set of m mutually independent measurements having the same associated uncertainty.

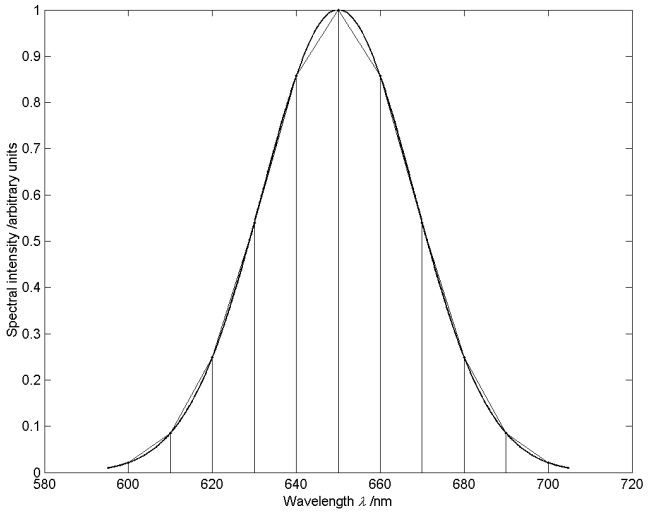

Figure 5.1: The trapezoidal rule for approximate integration applied to a spectral peak.

If the measurements are provided at non-uniformly spaced values of x, viz., $x_1, \ldots, x_m$ (assumed increasing), a slight generalization of the trapezoidal rule is

$$\widetilde{I} = \sum_{i=1}^{m-1} (x_{i+1} - x_i)(y_i + y_{i+1})/2.$$

The term $x_i - x_{i-1}$ is the width of the ith trapezoid and $(y_i + y_{i+1})/2$ its mean height. Thus, the formula represents the sum of the areas of the trapezoids constituting the area of the piecewise-linear approximation to the function.

More sophisticated rules are possible. For instance, rather than approximating the underlying function by a straight line over each interval, a polynomial of higher order can be used. Once these polynomial pieces have been constructed, their integration over each interval and their summation provides the required approximation to the definite integral of the function. Again, values of the indefinite integral correspond to partial sums.

There are various choices of order for the polynomial pieces and various ways of constructing them. A class of approaches is discussed in a general setting below.

The requirement is to design an algorithm to determine (an approximation to) the area under a curve or function 'defined' by measurement data. Unless further information is available, such as the asymptotic behaviour of the function underlying the data, this requirement will be interpreted as the area under the curve defined by the *span* of the data, i.e., between the smallest and largest abscissa values. Thus, any approach will be more suited for functions that decay to zero for small and large values of the abscissa and for which data representative of all but the insignificant tails is available. Spectral data, which is widespread in metrology, often has this characteristic.

Further requirements are:

1. Estimate the numerical accuracy of the approximation to the area returned by the algorithm.
2. Evaluate the uncertainty associated with the approximation to the area.
3. Compare the numerical accuracy and the associated uncertainty in the context of fitness for purpose of the result.

5.2 Outline design

1. Regard the measurements as representing a curve, the area under which is sought.
2. Represent each interval between adjacent measurements by a polynomial piece.
3. Form the definite integral of each polynomial piece over the interval to which it applies to provide an approximation to the area under the part of the curve relating to that interval.
4. Sum these definite integrals to obtain an approximation to the area under the complete curve. Use partial sums if values of the indefinite integral are required.
5. Compare the results obtained using different approaches to obtaining the polynomial pieces and using polynomials of different orders.
6. Apply the law of propagation of uncertainty to evaluate the uncertainty due to measurement associated with the approximations obtained.

5.2.1 Curve defined by measurement

Data points $(x_i,\ y_i)$, $i = 1, \ldots, m$, with $x_1 < \cdots < x_m$, are provided. The x_i are regarded as exact. The y_i are regarded as measurements of a smooth unknown function $f(x)$ at the abscissae x_i. The measurements y_i are regarded as inexact, to be statistically mutually independent and to have associated standard uncertainties $u(y_i)$. (It is possible to extend the treatment here to cases where mutual dependencies exist among the y_i.) If the $u(y_i)$ are provided, as assumed here, the standard uncertainty in the determined area can be evaluated.

5.2.2 Representation by polynomial pieces

The function underlying the data is represented over each interval $(x_i,\ x_{i+1})$, $i = 1, \ldots, m-1$, between adjacent abscissae by a polynomial. One of the simplest cases is the use of an interpolating polynomial of order two (degree one), i.e., the straight line joining the points $(x_i,\ y_i)$ and $(x_{i+1},\ y_{i+1})$.

If a polynomial of higher degree is used to represent the function over the interval $(x_i,\ x_{i+1})$, additional information is required to define it. For instance, for a cubic polynomial, two further piece of information are required. Unless the interval of concern is the first or the last, it is appropriate to use also the points either side, viz., $(x_{i-1},\ y_{i-1})$ and $(x_{i+2},\ y_{i+2})$. For the first interval, since no point 'to the left' of x_1 is available, two points to the right are used, i.e., the

cubic polynomial interpolating points 1 to 4 is formed and used to represent the function over the interval from x_1 to x_2. An analogous statement applies to the last ($(m-1)$st) interval.

Polynomials of odd degree use information symmetrically, i.e., except possibly for intervals near the ends of the range, the same number of additional points each side of the interval of concern.

Polynomials of even degree cannot by their nature use information symmetrically, so it is used in as balanced a way as possible. For instance, to provide an interpolating quartic polynomial for the interval $(x_i,\ x_{i+1})$, points with indices $i-2,\ldots,i+2$ or with indices $i-1,\ldots,i+3$ could be used. Both quartics are 'equally good' in the absence of further information, and their mean over the interval may be taken to induce some symmetry into the calculation.

In the end intervals or intervals close to the range endpoints, information can be used 'as symmetrically as possible'. Thus, an equal number of points either side of the interval of concern can be used (with an adaptation as above for polynomials of even order), unless some of these points are 'unavailable' on one side, in which case as many points as needed are taken from the other side.

Table 5.1 shows the points used by some of the integration rules (in fact those used in the case study below).

Rule	Points used to construct the polynomial piece over interval $(x_i,\ x_{i+1})$	
Trapezoidal	$i, i+1,$	$1 \le i < m-1$
Cubic	$1,\ldots,4,$	$i = 1,$
	$i-1,\ldots,i+2,$	$2 \le i \le m-2,$
	$m-3,\ldots,m,$	$i = m-1$
Quartic I	$1,\ldots,5,$	$1 \le i \le 2,$
	$i-2,\ldots,i+2,$	$3 \le i \le m-2,$
	$m-4,\ldots,m,$	$i = m-1$
Quartic II	$1,\ldots,5,$	$i = 1,$
	$i-1,\ldots,i+3,$	$2 \le i \le m-3,$
	$m-4,\ldots,m,$	$m-2 \le i \le m-1$
Quintic	$1,\ldots,6,$	$1 \le i \le 2,$
	$i-2,\ldots,i+3,$	$3 \le i \le m-3,$
	$m-5,\ldots,m,$	$m-2 \le i \le m-1$

Table 5.1: The points used to construct the polynomial pieces over each interval when using an integration rule based on polynomials.

5.3 Integration of the polynomial pieces

For each of the $m-1$ intervals the integral of the constructed polynomial is formed and used as a contribution to I or to $F(x_i)$, $i = 1,\ldots,m$. (The approaches considered here can be extended to cases where $F(x)$ is to be evaluated at points that do not coincide with the x_i.) Let I_i denote the integral over the interval $x_i \le x \le x_{i+1}$ of the polynomial piece for that interval. Then

$$I = \int_a^b f(x)\,dx \approx \sum_{i=1}^{m-1} I_i, \qquad F(x_j) = \int_a^{x_j} f(t)\,dt \approx \sum_{i=1}^{j-1} I_i.$$

The approach is quite general. Many existing rules constitute special cases. If the x_i are uniformly spaced the use of polynomial pieces of order two gives the trapezoidal rule, while order three gives Simpson's rule. If the x_i are generally spaced, polynomial pieces of order two gives the generalized trapezoidal rule, whereas order four gives the Gill-Miller rule [41].

5.4 Approximation errors

Interpolating polynomials of some order can be used to provide an approximation $\widetilde{I}'$ to I and then higher order used to provide a second approximation $\widetilde{I}''$. The difference between these approximations is an indication of the error in $\widetilde{I}'$. Ideally, further orders should also be used.

5.5 Uncertainties

All the rules considered are linear in the ordinates y_i. Thus, the law of propagation of uncertainty [7] based on a first-order Taylor series expansion can be applied, making no further approximation, to evaluate the uncertainty associated with the results.

5.6 Other remarks

More sophisticated approaches embody modelling the data [2, 34]. The data is 'replaced' by a mathematical model that adequately explains the data and has a degree of smoothness that ideally is consistent with the underlying function and the uncertainties associated with the data. Tests of conformity of the model and data are carried out to give a degree of assurance regarding these issues. The model is integrated over the range of the data, the value obtained used as an approximation to the required integral. If the model is linear in its parameters, the uncertainty associated with the approximate integral can be established as above.

5.7 CASE STUDY. Climate change

5.7.1 Objective

The determination of integrated radiance (total energy) across the spectrum from radiance measurements at a set of wavelengths.

5.7.2 What this case study illustrates

1. The approximation of the area under a curve defined by measurement data
2. The use of different integration methods to obtain such approximations
3. Estimation of the numerical accuracy of the approximations obtained
4. Evaluation of the uncertainties associated with the approximations

5. Analysis of linear models not specified explicitly

6. The use of problem structure to avoid long computer times

5.7.3 The practical problem

The Earth's Radiation Budget (ERB) is the difference between (a) the incoming radiation from the sun and (b) the outgoing reflected and scattered solar radiation plus the thermal infrared emission to space. Measurements relating to ERB are made from satellites and help to provide better understanding of climate change, including gobal warming.

The total radiant energy associated with ERB is the difference between the total energies associated with (a) and (b). A difference that is significant compared with the uncertainty associated with the difference can be regarded as evidence for climate change.

The total radiant energy associated with (a) is the energy given by integrating across the relevant spectral region (the wavelength-dependent) incoming radiation from the sun. A similar statement applies to (b).

The task of determining the contributory integrals and the associated uncertainties is that addressed generically in this section. The data considered here relates to check measurements using a source provided by NPL of the on-board instrumentation used for measuring the incoming radiation from the sun. Figure 5.2 shows the spectral radiance measurements obtained. They correspond to 151 non-uniformly spaced wavelength values spanning the spectral range 350 nm to 3 000 nm. It is more complicated than the single smooth peak above, but has the property that it exhibits a strong decay to zero for small and large wavelengths.

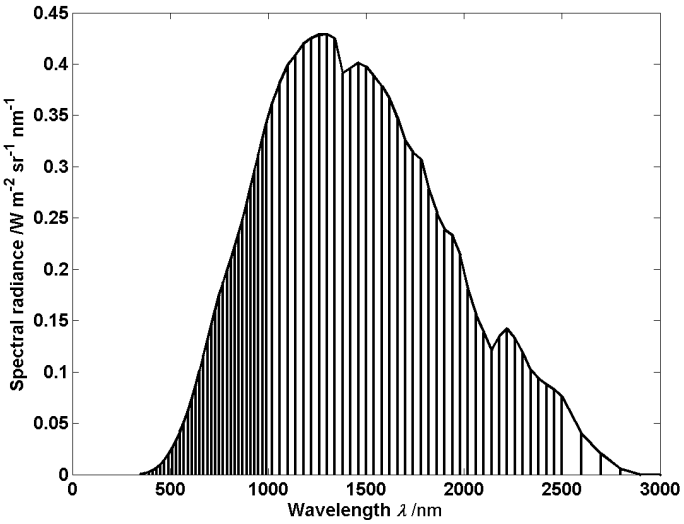

Figure 5.2: Data for the spectral irradiance of a lamp and the piecewise-linear function joining the data points. The vertical lines delineate the panels used in the application of the integration rules.

5.7.4 Newton representation by polynomial pieces

Polynomials can conveniently be represented in their Newton form [19]. The straight line over $(x_i,\ x_{i+1})$ is expressed as

$$y = y_i + y_{i,i+1}(x - x_i), \tag{5.2}$$

where $y_{i,i+1}$ denotes the divided difference of first order defined by

$$y_{i,i+1} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}.$$

It is immediately verified that expression (5.2) returns the values y_i when $x = x_i$ and y_{i+1} when $x = x_{i+1}$. The form (5.2) is the Newton form of a polynomial of order two (degree one).

A cubic polynomial, applying to an interval $(x_i,\ x_{i+1})$ other than the first and the last can be expressed in the Newton form

$$\begin{aligned} y &= y_{i-1} + y_{i-1,i}(x - x_{i-1}) + y_{i-1,i,i+1}(x - x_{i-1})(x - x_i) \\ &+ y_{i-1,i,i+1,i+2}(x - x_{i-1})(x - x_i)(x - x_{i+1}), \end{aligned} \tag{5.3}$$

where $y_{i-1,i,i+1}$ and $y_{i-1,i,i+1,i+2}$ are divided differences of orders two and three, respectively, defined recursively from those of lower orders using

$$y_{j,\ldots,k} = \frac{y_{j+1,\ldots,k} - y_{j,\ldots,k-1}}{x_k - x_j}.$$

5.7.5 Integrating the polynomial pieces

There are several ways to determine the integral of each polynomial piece over the interval to which it applies. A poor way, in general, would be to construct the Taylor form of the polynomial, e.g., its expansion in powers of x about the left-hand endpoint of the interval, and to integrate that form. The coefficients in this form can indeed be obtained from the divided differences, but subtractive cancellation tends to reduce their accuracy. A good way is to evaluate the polynomial piece from its Newton representation at the n Chebyshev points in the interval, after normalizing the latter to the range $[-1,\ 1]$, and to form the Chebyshev-series representation from those values [19]. Another way, used here, is to evaluate the Newton form at n uniformly spaced points in the interval.[5] Then an integration rule that is exact for polynomials of order n can be applied to those values. These rules are known as Newton-Coates' rules.

5.7.6 Evaluation of the uncertainty

Appendix C provides an approach to evaluating the standard uncertainty associated with the approximation to the integral. It exploits the fact that the integration rule is linear in the ordinates, the quantities that are subject to uncertainty. By perturbing each ordinate in turn (by an arbitrary amount) and re-applying the rule, the sensitivity coefficients required by an application of the law of propagation of uncertainty can be computed essentially exactly. The appendix also shows how this laborious calculation can be made efficient.

[5] The endpoints are already known, so only $n - 2$ 'internal' evaluations are needed. The same statement applies to the Chebyshev form.

5.7.7 Results

The various rules indicated were applied to the full set of 151 points illustrated in figure 5.2. In addition, the trapezoidal rule was applied to every other point (76 points). Appendix C describes the manner in which the uncertainties associated with these results were evaluated. The results, together with the associated standard uncertainties, are given in table 5.2, from which it is seen that, for instance, the trapezoidal rule (based on 151 points) and the Gill-Miller rule give values of 515.29 and 515.28 Wm^{-2}, agreeing closely compared with the associated standard uncertainties. In the absence of standard uncertainties there would not be a direct means of stating whether such closeness is adequate. The trapezoidal rule with half the number of points gives 515.78 Wm^{-2} with an associated standard uncertainty of 1.02 Wm^{-2}. The use of higher-order rules also gave good consistency with the trapezoidal rule and the Gill-Miller rule. 'Quartic 1' in the table refers to the use of quartic interpolation with two points to the left and one to the right of 'sufficiently interior' intervals, and 'Quartic 2' with one to the left and two to the right. All the rules based on the full set of points delivered approximations between 515.27 and 515.30 Wm^{-2}, with associated standard uncertainties between 0.74 and 0.76 Wm^{-2}. In terms of the climate change work, it would be possible to state with a high degree of assurance that the value of the integral is 515.3 Wm^{-2} with an associated standard uncertainty of 0.8 Wm^{-2} (or a relative standard uncertainty of 0.2 %).

Number of points	Rule	Approximate integral /Wm^{-2}	Standard uncertainty /Wm^{-2}
151	Trapezoidal	515.290 7	0.740 1
76	Trapezoidal	515.783 4	1.020 1
151	Cubic (Gill-Miller)	515.280 7	0.745 1
151	Quartic I	515.292 3	0.753 8
151	Quartic II	515.269 7	0.746 7
151	Quintic	515.303 7	0.757 7

Table 5.2: The application of various quadrature rules to the data of figure 5.2. More digits are quoted than justified in terms of the standard uncertainty for purposes of numerical comparison only.

5.7.8 Lessons learnt

1. The problem can be decomposed into the sum of a number of simpler problems. Each sub-problem relates to an interval between points and its 'solution' forms a contributory part to the required solution. It involves forming an interpolating polynomial and integrating it, taking advantage at this stage of integration rules for uniformly spaced points.

2. The Newton form of a polynomial is easy to construct, and for this problem has advantages over some other forms. In particular, it avoids the subtractive cancellation that can arise when forming the Taylor coefficients.

3. The use of alternative integration rules, viz., rules using different polynomial orders, provide a means of validating the results.

4. The evaluation of the standard uncertainties, when possible, associated with the delivered approximations to the integral provides a powerful means of confirming whether the closeness of the approximations is adequate.

5. The law of propagation of uncertainty applies exactly to the integration rules considered, since the approximation to the integral is a linear function of them.

6. The linear function is not available explicitly, but the required standard uncertainty can be obtained by reusing the integration rule.

7. Exploiting the structure of the problem, viz., that each polynomial piece depends only on 'local' ordinates can be used to provide the required standard uncertainty in a time $O(m)$ rather than $O(m^2)$, where m is the number of measurements.

Bibliography

[1] G. T. Anthony, H. M. Anthony, B. Bittner, B. P. Butler, M. G. Cox, R. Drieschner, R. Elligsen, A. B. Forbes, H. Gross, S. A. Hannaby, P. M. Harris, and J. Kok. Chebyshev best-fit geometric elements. Technical Report DITC 221/93, National Physical Laboratory, Teddington, UK, 1993.

[2] R. M. Barker, M. G. Cox, A. B. Forbes, and P. M. Harris. SS*f*M Best Practice Guide No. 4. Discrete modelling and experimental data analysis. Technical report, National Physical Laboratory, Teddington, UK, 2004. `www.npl.co.uk/ssfm/download/bpg.html#ssfmbpg4`.

[3] 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, Teddington, UK, 2003.

[4] R. E. Beckett, M. G. Cox, M. P. Dainton, P. M. Harris, E. G. Johnson, and G. I. Parkin. Testing methods of Java libraries. Technical Report CMSC 35/04, National Physical Laboratory, Teddington, UK, 2004.

[5] H. Bettin and H. Fehlauer. Density of mercury—measurements and reference values. *Metrologia*, 41:S16–S23, 2004.

[6] BIPM. Mutual recognition of national measurement standards and of calibration and measurement certificates issued by national metrology institutes. Technical report, Bureau International des Poids et Mesures, Sèvres, France, 1999.

[7] BIPM, IEC, IFCC, ISO, IUPAC, IUPAP, and OIML. Guide to the Expression of Uncertainty in Measurement, 1995. ISBN 92-67-10188-9, Second Edition.

[8] K. P. Birch and M. J. Downs. An updated Edlén equation for the refractive index of air. *Metrologia*, 30:155–162, 1993.

[9] C. Bischof, A. Carle, P. Khademi, and A. Mauer. The ADIFOR 2.0 system for the automatic differentiation of Fortran 77 programs. Technical Report ANL/MCS-P481-1194, Argonne National Laboratory, Argonne, Illinois, USA, 1994.

[10] R. Boudjemaa, M. G. Cox, A. B. Forbes, and P. M. Harris. Automatic differentiation and its applications to metrology. In Patrizia Ciarlini, M. G. Cox, F. Pavese, D. Richter, and G. B. Rossi, editors, *International*

Conference on Advanced Mathematical and Computational Tools in Metrology VI, Torino, September, 2003. In press.

[11] R. Boudjemaa, M. G. Cox, A. B. Forbes, and P. M. Harris. Automatic differentiation techniques and their application in metrology. Technical Report CMSC 26/03, National Physical Laboratory, Teddington, UK, 2003.

[12] R. Boudjemaa and A. B. Forbes. Parameter estimation methods for data fusion. Technical Report CMSC 38/04, National Physical Laboratory, Teddington, UK, 2004.

[13] Françoise Chaitin-Chatelin and Valérie Frayssé. *Lectures on finite precision computations*. SIAM, Philadelphia, 1996.

[14] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. Maple V Library Reference Manual. Technical report, University of Waterloo, Waterloo, Ontario, Canada, 1991.

[15] C. W. Clenshaw and J. G. Hayes. Curve and surface fitting. *J. Inst. Math. Appl.*, 1:164–183, 1965.

[16] 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.

[17] H. R. Cook, M. G. Cox, M. P. Dainton, and P. M. Harris. Testing spreadsheets and other packages used in metrology. A case study. Technical Report CMSC 26/99, National Physical Laboratory, Teddington, UK, 1999.

[18] M. G. Cox. Piecewise Chebyshev series. *Bull. Inst. Math. Appl.*, 22:163–166, 1986.

[19] M. G. Cox. Reliable determination of interpolating polynomials. *Numerical Algorithms*, 5:133–154, 1993.

[20] M. G. Cox. Constructing and solving mathematical models of measurement. In P. Ciarlini, M. G. Cox, F. Pavese, and D. Richter, editors, *Advanced Mathematical Tools in Metrology II*, pages 7–21, Singapore, 1996. World Scientific.

[21] M. G. Cox. Graded reference data sets and performance profiles for testing software used in metrology. In P. Ciarlini, M. G. Cox, F. Pavese, and D. Richter, editors, *Advanced Mathematical Tools in Metrology III*, pages 43–55, Singapore, 1997. World Scientific.

[22] M. G. Cox. The evaluation of key comparison data. *Metrologia*, 39:589–595, 2002.

[23] M. G. Cox, M. P. Dainton, A. B. Forbes, P. M. Harris, P. M. Schwenke, B. R. L Siebert, and W. Wöger. Use of Monte Carlo simulation for uncertainty evaluation in metrology. In P. Ciarlini, M. G. Cox, E. Filipe,

F. Pavese, and D. Richter, editors, *Advanced Mathematical Tools in Metrology V. Series on Advances in Mathematics for Applied Sciences Vol. 57*, pages 93–104, Singapore, 2001. World Scientific.

[24] M. G. Cox, M. P. Dainton, and P. M. Harris. Testing spreadsheets and other packages used in metrology. Testing functions for linear regression. Technical Report CMSC 08/00, National Physical Laboratory, Teddington, UK, 2000.

[25] M. G. Cox, M. P. Dainton, and P. M. Harris. Testing spreadsheets and other packages used in metrology. Testing functions for the calculation of standard deviation. Technical Report CMSC 07/00, National Physical Laboratory, Teddington, UK, 2000.

[26] M. G. Cox, A. B. Forbes, P. M. Fossati, P. M. Harris, and I. M. Smith. Techniques for the efficient solution of large scale calibration problems. Technical Report CMSC 25/03, National Physical Laboratory, Teddington, UK, 2003.

[27] 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.

[28] M. G. Cox and P. M. Harris. Overcoming an instability arising in a spline approximation algorithm by using an alternative form of a simple rational function. *Bull. Inst. Math. Appl.*, 25:228–232, 1989.

[29] 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.

[30] M. G. Cox and P. M. Harris. Guidelines to help users select and use software for their metrology applications. Technical Report CMSC 04/00, National Physical Laboratory, Teddington, UK, 2000.

[31] M. G. Cox and P. M. Harris. Software specifications for uncertainty evaluation. Technical Report CMSC 40/04, National Physical Laboratory, Teddington, UK, 2004. CMSC 10/01 revised.

[32] M. G. Cox and P. M. Harris. SSfM Best Practice Guide No. 6. Uncertainty evaluation. Technical report, National Physical Laboratory, Teddington, UK, 2004. `www.npl.co.uk/ssfm/download/bpg.html#ssfmbpg6`.

[33] M. G. Cox, P. M. Harris, E. G. 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.

[34] M. G. Cox, P. M. Harris, P. D. Kenward, and Emma Woolliams. Spectral characteristic modelling. Technical Report CMSC 27/03, National Physical Laboratory, Teddington, UK, 2003.

[35] M. G. Cox and E. Pardo. The total median and its uncertainty. In P. Ciarlini, M. G. Cox, E. Filipe, F. Pavese, and D. Richter, editors, *Advanced Mathematical Tools in Metrology V. Series on Advances in*

Mathematics for Applied Sciences Vol. 57, pages 106–117, Singapore, 2001. World Scientific.

[36] T. J. Dekker. Finding a zero by means of successive linear interpolation. In B. Dejon and P. Henrici, editors, *Constructive Aspects of the Fundamental Theorem of Algebra*, London, 1969. Wiley Interscience.

[37] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability 57. Chapman and Hall, New York, 1993.

[38] S. L. R. Ellison, V. J. Barwick, P. Norris, and M. Griffiths. Complete curve fitting of extraction profiles for estimating uncertainties in recovery estimates. *Analyst*, 128:493–498, 2003.

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

[40] L. Fox. How to get meaningless answers in scientific computation (and what to do about it). *IMA Bull.*, 7:296–302, 1971.

[41] P. E. Gill and G. F. Miller. An algorithm for the integration of unequally spaced data. *Comput. J.*, 15:80–83, 1972.

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

[43] S. J. Hammarling. An introduction to the quality of computed solutions. Technical report, Numerical Algorithms Group Ltd., 2004. To appear.

[44] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 1996.

[45] N. J. Higham. Testing linear algebra software. In R. F. Boisvert, editor, *The Quality of Numerical Software: Assessment and Enhancement*, pages 109–124, London, 1997. Chapman and Hall.

[46] T. E. Hull and J. R. Swenson. Tests of probabilistic models for propagation of rounding errors. *Comm. ACM*, 9:108–113, 1966.

[47] IFIP. *Accuracy and reliability in scientific computing*. SIAM, Philadephia, 2004. IFIP WG 2.5, Project 68.

[48] ISO. ISO 6506-1. Metallic materials—Brinell hardness test—Part 1: Test method, 1999.

[49] ISO. ISO 10360-6. Geometrical product specifications (GPS) – acceptance test and reverification test for coordinate measuring machines (CMM). Part 6: Computation of Gaussian associated features, 2002. International Standards Organization, Geneva.

[50] J. N. Lyness and C. B. Moler. Numerical differentiation of analytic functions. *SIAM J. Numer. Anal.*, 4:202–210, 1967.

[51] J. S. Maritz and R. G. Jarrett. A note on estimating the variance of the sample median. *J. Amer. Statist. Assoc.*, 73:194–196, 1978.

[52] J. R. Rice. *Matrix Computations and Mathematical Software*. McGraw-Hill, New York, 1981.

[53] J. R. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, Belmont, Ca., USA, second edition, 1995.

[54] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Rev.*, 40:110–112, 1998.

[55] P. H. Sterbenz. *Floating-Point Computation*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[56] G. Strang and K. Borre. *Linear Algebra, Geodesy and GPS*. Wiley, Wellesley-Cambridge Press, 1997.

[57] J. Vignes and R. Alt. An efficient stochastic method for round-off error analysis. In W. L. Miranker and R. A. Toupin, editors, *Accurate Scientific Computations 1985*, pages 183–205, Berlin, 1985. Springer-Verlag.

[58] B. Wichmann, G. Parkin, and R. Barker. SS*f*M Best Practice Guide No. 1. Validation of software in measurement systems. Technical report, National Physical Laboratory, Teddington, UK, 2004. `www.npl.co.uk/ssfm/download/bpg.html#ssfmbpg1`.

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

Appendix A

Floating-point error analysis of the Brinell hardness test formulae

Consider the floating-point error analysis of the Brinell hardness test formulae (3.5) and (3.12), applying the rules of section 2.6.

The critical part of formulae (3.5) is the expression

$$E = D - T, \qquad T = \sqrt{D^2 - d^2}.$$

Using the stable form $(D-d)(D+d)$ for $D^2 - d^2$, and expression (3.16),

$$E = D - T, \qquad T = ((D-d)(D+d))^{1/2},$$

$$\begin{aligned} \widehat{S} &= (D^2 - d^2)(1 + 3e_7), \\ \widehat{T} = \mathrm{fl}(\widehat{S}^{1/2}) &= T(1 + 3e_7)^{1/2}(1 + e_8) = T(1 + 2.5e_9), \\ \widehat{E} &= (D - T(1 + 2.5e_9))(1 + e_{10}). \end{aligned} \tag{A.1}$$

Hence,

$$\frac{\widehat{E} - E}{E} = \frac{(D-T)e_{10} - 2.5Te_9}{D - T},$$

and so

$$\frac{|\widehat{E} - E|}{E} \leq \frac{D + 1.5T}{D - T}\eta.$$

The remainder of the computation to form B involves just multiplications and divisions, five in all, and thus

$$\widehat{B} = \mathrm{fl}\left(\frac{0.204F}{\pi D\widehat{E}}\right) = \frac{0.204F}{\pi D\widehat{E}}(1 + 5e_{11}).$$

So $\widehat{B}$ has the same relative error bound as $\widehat{E}$, inflated by the factor $1+5\eta$, which can safely be ignored. Thus,

$$\frac{|\widehat{B} - B|}{B} \leq \frac{D + 1.5T}{D - T}\eta = \frac{(D+T)(D+1.5T)}{d^2}\eta, \qquad T = \sqrt{D^2 - d^2}.$$

Since $T \leq D$, this bound can be expressed as

$$\frac{|\widehat{B} - B|}{B} \leq 5\frac{D^2}{d^2}\eta.$$

For the alternative formula (3.12), using expression (A.1)

$$\mathrm{fl}(D + \widehat{T}) = T(1 + 3.5e_{12}).$$

Hence, since the remaining operations to form B involve purely multiplications and divisions, six in all,

$$\widehat{B} = B(1 + 9.5e_{13}),$$

and so

$$\frac{|\widehat{B} - B|}{B} \leq 9.5\eta.$$

Appendix B

Summing the series for the hot-ball diffusion model

The inequality (3.26) relating to the number of terms $K = K(\lambda)$ in the finite series (3.25) for $S_K(\lambda)$ is derived. Its use ensures that the approximation of $S(\lambda)$, defined by the infinite series (3.22), by $S_K(\lambda)$ has an absolute error no greater than δ. Then K is to be chosen such that

$$S(\lambda) - S_K(\lambda) \leq \delta,$$

i.e., such that

$$\sum_{k=K+1}^{\infty} \frac{1}{k^2} e^{-\lambda k^2} \leq \delta. \tag{B.1}$$

An upper bound B_λ for the left-hand side of this inequality is now established. Thus, if a number K of terms in the series is taken such that

$$B_\lambda \leq \delta,$$

then *a fortiori* inequality (B.1) will be satisfied.

Let $k > 1/2$, $\lambda > 0$ and $F(x)$ be a twice continuously differentiable function with $F''(x) > 0$. Then

$$\int_{k-1/2}^{k+1/2} F(x)dx > F(k). \tag{B.2}$$

To prove this result, first expand $F(x)$ about $x = k$:

$$F(x) = F(k) + (x-k)F'(k) + \frac{1}{2}(x-k)^2 F''(\xi_x)dx,$$

for some ξ_x depending on x. Thus,

$$\begin{aligned}
\int_{k-1/2}^{k+1/2} F(x)dx &= \left[(x-k)F(k) + \frac{1}{2}(x-k)^2 F'(k)\right]_{k-1/2}^{k+1/2} \\
&+ \int_{k-1/2}^{k+1/2} \frac{1}{2}(x-k)^2 F''(\xi_x)dx \\
&= F(k) + \frac{1}{2}(x-k)^2 F''(\xi_x)dx.
\end{aligned}$$

But the integrand in the right-hand integral is positive throughout the interval $[k-1/2,\ k+1/2]$, except at $x=k$, where it is zero. Hence, the value of this integral is positive, and thus the inequality (B.2) indeed holds.

Now, since $\exp(-\lambda x^2)$ is a decreasing function of x for $x>0$ and $\lambda>0$,

$$\begin{aligned}\int_{K+1/2}^{\infty}\frac{1}{x^2}e^{\lambda x^2}dx &\leq \int_{K+1/2}^{\infty}\frac{1}{x^2}e^{\lambda(K+1/2)^2}dx \\ &= e^{\lambda(K+1/2)^2}\int_{K+1/2}^{\infty}\frac{1}{x^2}dx = \frac{1}{K+1/2}e^{-\lambda(K+1/2)^2}.\end{aligned}$$

So, in order to establish how many terms should be taken in the sum, carry out the summation process until the current number K of terms satisfies

$$\frac{1}{K+1/2}e^{-\lambda(K+1/2)^2}\leq\delta.$$

Similar concepts would apply to a range of other summation formulae. A certain amount of analysis, as above, would be required in any one instance. However, once the result has been implemented faithfully, the software will provide the sum to the prescribed accuracy.

Appendix C

Uncertainty evaluation for quadrature rules

Consider input quantities $\mathbf{Y} = (Y_1, \ldots, Y_N)^{\mathrm{T}}$, an output quantity Z and a linear model

$$Z = \mathbf{a}^{\mathrm{T}}\mathbf{Y} = \sum_{i=1}^{N} a_i Y_i \tag{C.1}$$

relating these quantities. Let $\mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$ denote best estimates of $\mathbf{Y}$ and $\mathbf{u}(\mathbf{y}) = (u(y_1), \ldots, u(y_N))^{\mathrm{T}}$ the associated standard uncertainties. The measurement result is

$$z = \mathbf{a}^{\mathrm{T}}\mathbf{y} = \sum_{i=1}^{N} a_i y_i.$$

The law of propagation of uncertainty [7] states that the standard uncertainty $u(z)$ associated with z is given by

$$u(z) = \|\mathbf{a}^{\mathrm{T}}\mathbf{u}(\mathbf{y})\| = \left(\sum_{i=1}^{N} a_i^2 u^2(y_i) \right)^{1/2}.$$

The y_i will denote the measurements (ordinates) at x_i of a function whose definite integral z over the span of the x_i is required. A simple quadrature rule can readily be expressed explicitly in the form (C.1). For example, the trapezoidal rule for ordinates at a uniform spacing h has $\mathbf{a} = h(1/2, 1, \ldots, 1, 1/2)^{\mathrm{T}}$ (section 5.1). Further, the trapezoidal rule for general spacing, with abscissae $x_1, \ldots, x_m$ has

$$\mathbf{a} = (h_1, h_1 + h_2, \ldots, h_{m-2} + h_{m-1}, h_{m-1})^{\mathrm{T}},$$

where $h_i = x_{i+1} - x_i$, $i = 1, \ldots, m-1$, are the interval lengths. Other rules for uniform spacing can also generally be cast in this way. However, the determination of $\mathbf{a}$ for a general rule is not straightforward.

An approach is given that can readily be applied in such circumstances. In terms of its computational complexity, it is not efficient, taking a number of arithmetic operations that is proportional to m^2. However, this aspect should

not be a drawback for many problems. For instance, for the data of figure 5.2, the computational time is less than 1 second on a 1 GHz PC.

Since the sensitivity coefficients are equal to the weights it is preferable to use a quadrature rule whose weights are as close to being equal as possible.

Define

$$Z_r = I_r(Y_r) = I((y_1, \ldots, y_{r-1}, Y_r, y_{r+1}, \ldots, y_m)^{\mathrm{T}}).$$

Then,

$$I_r(y_r) = I(\mathbf{y}) = \mathbf{a}^{\mathrm{T}}\mathbf{y}$$

and, for any non-zero δy_r,

$$I_r(y_r + \delta y_r) = \mathbf{a}^{\mathrm{T}}\mathbf{y} + a_r \delta y_r.$$

It follows that

$$a_r = \frac{I_r(y_r + \delta y_r) - I(\mathbf{y})}{\delta y_r}.$$

C.1 Speeding up the calculation

The whole process takes a time proportional to m^2 (m applications of a rule requiring a time proportional to m), which can be prohibitive. However, the problem has structure in the sense that each contribution I_i to the integral I depends only on y-values having indices 'close to i'. Thus, perturbations can be made *simultaneously* to a number of the y_i and the consequent changes in the I_i used to form the corresponding sensitivity coefficients. For instance, suppose cubic interpolating polynomials are used, and consider a simple case with $m = 10$ ordinates. Table C.1 shows the indices j of the sub-integrals I_j that are affected by changes in the indices i of the y_i.

Index of	Indices of affected sub-integrals								
y-value changed	1	2	3	4	5	6	7	8	9
1	†	†							
2	†	†	†						
3	†	†	†	†					
4		†	†	†	†				
5			†	†	†	†			
6				†	†	†	†		
7					†	†	†	†	
8						†	†	†	†
9							†	†	†
10								†	†

Table C.1: Indices j of the sub-integrals I_j affected by changes in the indices i of the y_i

Simultaneous changes are therefore possible to the ordinates in each of the following rows:

$$\begin{array}{lll} y_1, & y_5, & y_9 \\ y_2, & y_6, & y_{10} \\ y_3, & y_7, & \\ y_4, & y_8, & \end{array}$$

Thus, only four rather than $m = 10$ 're-integrations' are required. By a simple extension of this approach, for cubics, only four re-integrations are required for any value of m, and, for interpolating polynomials of order n, only n re-integrations are required for any value of m. Thus, the overall cost is proportional to nm rather than m^2, or simply m for a fixed-order rule.