ISSUE 14 • AUTUMN 2003

Counting on IT

MATHEMATICS AND SCIENTIFIC COMPUTING

The Wilkinson Prize for Numerical Software

In honour of the outstanding contributions of James Hardy Wilkinson to the field of numerical software, NPL, the Numerical Algorithms Group and Argonne National Laboratory award every four years a prize of US $1000 for numerical software. The 2003 prize presentation took place in Sydney in July 2003 at the International Congress for Industrial and Applied Mathematics.

Wilkinson spent his pioneering days at NPL, where he laid the basis for reliable numerical methods for linear algebra and floating-point error analysis. These concepts are central to today's quality software for solving linear and non-linear algebraic systems and least-squares problems.

The winner of the 2003 Prize was Jonathan Shewchuk of the University of California at Berkeley for *Triangle, two-dimensional mesh generator and Delaunay triangulator. Triangle* generates high-quality unstructured triangular meshes. It also generates two-dimensional Delaunay triangulations, constrained Delaunay triangulations, Voronoi diagrams, and convex hulls. The software is used in discrete meshing for the discretisation of partial differential equations, in solving classes of geometric problems, and the graphing of surfaces. It has thousands of users, and is downloaded on average more than 30 times a day. *Triangle* has been licensed for inclusion in eleven commercial software packages.

The speed and accuracy of *Triangle* is a result of novel algorithms for extended precision floating-point arithmetic and the use of adaptive computation controlled by forward error analysis.

NPL sponsors the Wilkinson Prize, through the SS*f*M programme.

For further information contact
Maurice Cox, extension 6096
e-mail: maurice.cox@npl.co.uk

Jonathan Shewchuk (second left) of the University of California at Berkeley receives the 2003 Wilkinson Prize for *Triangle* from the sponsors ANL (represented by Jorge Moré, far left), NAG (Steve Hague, far right) and NPL (Maurice Cox).

Contents

NPL National Physical Laboratory

The National Physical Laboratory is operated on behalf of the DTI by NPL Management Limited, a wholly owned subsidiary of Serco Group plc

Testing Algorithms in Standards and METROS

Certain mathematical and computational analyses must be undertaken in order to implement a number of specification standards and guides in metrology. Many other computations are fundamental to metrology, although they are not always specified in standards, and these form the basis of software libraries such as METROS.

Many mathematical and statistical problems that arise in metrology can be posed in such a way that they have a unique solution. Algorithms in standards or software libraries, however, may provide only an *approximate* solution, or the solution to a 'nearby' problem. An approximate or nearby problem is often introduced because it is (more) tractable, i.e. it *can* be solved, it is *easier* to solve, or it is *quicker* to solve.

Choosing to solve an approximate problem is one reason why software may not deliver correct results. Another reason is that the implementation of software for solving the approximate problem is poor or faulty. The distinction is illustrated in figure 1, which shows the relationship between software S_a implemented to solve the problem specified by the computational aim C_0.

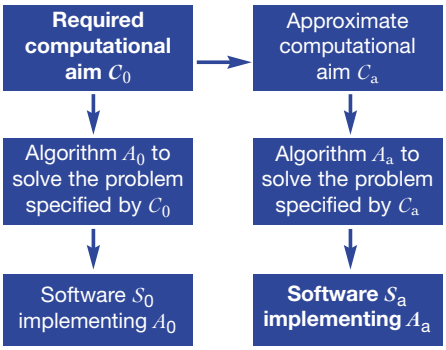


Figure 1: Objects in the process of developing mathematical software: computational aims (mathematics), algorithms (pseudo-code) and software (source or object code).

Algorithm testing [1] is concerned with understanding the effect of solving an approximate problem, by measuring the departure of the solution so obtained from the solution to the actual problem. In a project undertaken as part of the SSfM programme, a number of approaches, based on the methodologies of numerical software testing, have been proposed to test the *fitness for purpose* of algorithms used in metrology. The emphasis on fitness for purpose is an important one. It is reasonable to ignore the error introduced by solving an approximate problem if that error is negligible compared with the effect of other contributions to the uncertainty associated with the measurement result, e.g. inexact knowledge about the state of the measurement system or instrument, the environment, etc. On the other hand, account must be taken of the approximation error in cases where it is appreciable compared with the effect of these other contributions.

This (probabilistic) interpretation of fitness for purpose is illustrated in figures 2 and 3, in which y and y_a, respectively, are the solutions to the problems specified by C_0 and C_a for the same measurement data. The inexactness of the solutions, arising from that associated with the measurement data, is described by probability density functions (pdfs) $g(Y)$ and $g(Ya)$, respectively (figure 2). In terms of these pdfs, we can answer questions about the 'average' departure of y_a from y, as well as determining the probability of obtaining a solution to the problem specified by C_0 that is further from y than is y_a.

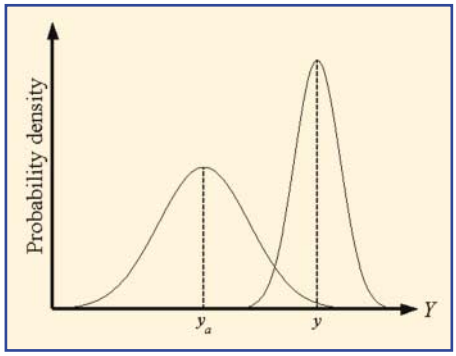


Figure 2: Probability density functions $g(Y)$ and $g(Y_a)$ for, respectively, the values of the solutions to C_0 and C_a.

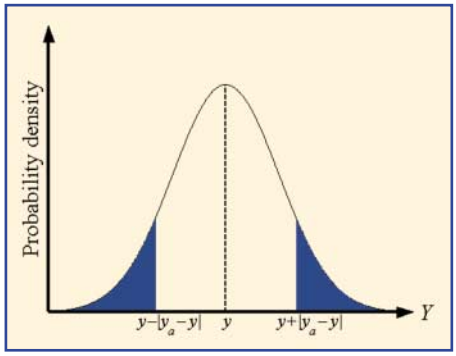


Figure 3: A measure of the fitness for purpose for solving C_a in place of C_0. The area of the shaded regions represents the probability of obtaining a solution to C_0 that is further from y than is y_a.

To determine the latter, consider the evaluation of Prob($|Y - y| \geq |y_a - y|$). This problem corresponds to finding the sum of the areas under the curve $g(Y)$ to the left of $y - |y_a - y|$ and to the right of $y + |y_a - y|$ (figure 3). A small probability suggests that y_a, the solution obtained for C_a, is *not* a likely solution to C_0, accounting for the inexactness of the measurement data.

Reference

[1] R M Barker, M G Cox, P M Harris and I M Smith. Testing algorithms in standards and METROS. NPL Report CMSC 18/03, March 2003.

For further information contact
Peter Harris, extension 6961
e-mail: peter.harris@npl.co.uk

Club Members' Page

Letter to the editor from Valery Granovsky

Measuring Instruments Including Software: Traditional Metrology Approach

Generally, guidance on the development and validation of metrology software makes no distinction between software used for modelling or processing measurement data and software to be used within a measuring instrument (MI). However, the metrological requirements on the software may be quite different in the two cases. Software for handling measurement data can be tested or validated on its own. For example, in the work on numerical software testing within the SS*f*M programme, each numerical function within packages like MatLab or the NAG library is tested in isolation.

Software that forms part of an MI needs to be treated differently: it must be tested or validated as part of the MI. Although the instrument designer would find it appropriate to conduct software testing in accordance with SS*f*M Best Practice Guide No. 1, the essential requirement from a metrological perspective is that it works properly in combination with the MI hardware. There is a need to describe, check, and certify the MI as a complete system fit for the purpose of metrology. This is the traditional metrological approach to calibrating MIs. Only if it is impossible to use this traditional approach should the MI be validated as a set of individual parts.

In conclusion, seen purely from the narrow metrological perspective, we believe that the traditional procedures for calibrating an MI that is entirely implemented in hardware should also be used for calibrating MIs that incorporate software. Calibration of the MI as a whole allows its metrological characteristics to be evaluated, including those of its software.

Contact: Valery Granovsky: +7 (812) 238 8167
e-mail: elprib@online.ru

Programme Manager's response:

The need to treat general metrology software differently from software embedded in MIs is well understood in SS*f*M. Best Practice Guide No. 12, (test and measurement software), addresses the former, and Best Practice Guide No. 1, (validation of measurement software), addresses the problem of validating the latter. Although, in SS*f*M, we agree that this latter type of software must be validated in combination with the hardware, the fact that software can exhibit classes of fault that are very unlikely to occur in hardware means that we need to combine validation techniques. We see software validation techniques as complementing but not replacing traditional MI validation techniques. We believe validation should focus on fitness for purpose and the purpose is generally not just metrological.

Automatic Differentiation

A wide variety of mathematical and scientific problems involve solving systems of nonlinear equations, or finding the maximum or minimum of a nonlinear function. For example, in data fitting, the estimates of model parameters are found by minimising a goodness-of-fit measure (figure 4). Finding the solution to these problems usually depends on being able to linearise the functions involved, and to achieve this we have to calculate their derivatives.

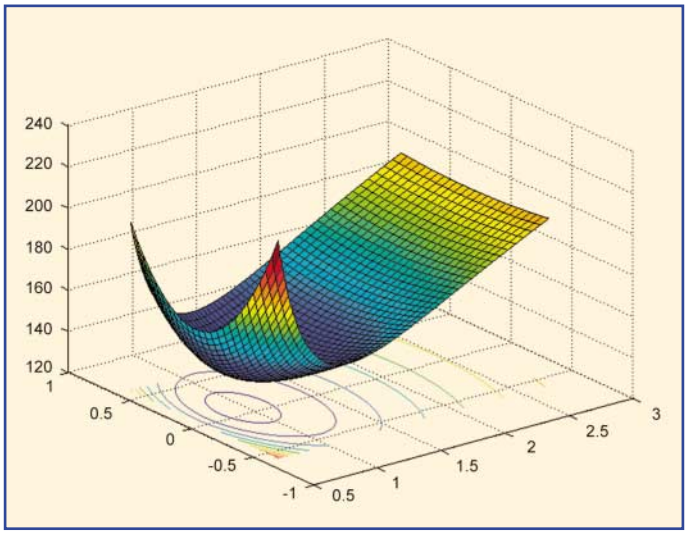


Figure 4. Finding estimates of parameters from data involves minimising a goodness-of-fit measure.

If $f(x)$ is a function of a variable x, its derivative f' is also a function of x and represents the slope of f at x. For a function of a number of variables $f(x_{1,\ldots,}x_n)$, the partial derivative of f with respect to x_j represents the slope of f in the direction of the jth axis. Using basic calculus rules, e.g. $(f+g)'(x) = f'(x) + g'(x)$, the derivatives of complex functions can be expressed in terms of the derivatives of their simpler, component functions. However, even for only moderately complicated functions, the hand calculation of derivatives can lead to pages of tedious algebra with an increasing likelihood of a mistake entering into the computation. If the function evaluation is encoded in a software component, it is natural to ask if it is possible for the software to compute the derivatives automatically.

Until recently, the standard method of evaluating derivatives numerically was to use finite differences. The main and simple idea of the method is to approximate the derivative by evaluating f at two nearby points. Typical of finite-difference formulae are the *forward-difference* formula $f'(x) \approx (f(x+h) - f(x))/h$ and the *central-difference* formula $f'(x) \approx (f(x+h) - f(x-h))/2h$. Here h is a "step" selected in an appropriate way. These formulae generally calculate only approximate estimates. If h is chosen to be too small, cancellation errors are likely. If h is too large, then the approximation error will be large.

Because of the accuracy difficulties associated with finite differences, attention is now being paid to the *complex step method*, which is similar to finite differences, but uses complex arithmetic. We recall that a complex number z is of the form $z = x + iy$ where x and y are real and $i = \sqrt{-1}$. All the arithmetic operations for real numbers also apply to complex numbers. Most real-valued functions $f(x)$ occurring in science also have complex-valued counterparts, and can be written in the form $f(z) \approx \mathrm{Re}\, f(z) + i\mathrm{Im}\, f(z)$ where the real-valued functions Re f and Im f are known as the *real* and *imaginary* parts. Taking $z = x + ih$ where x is real and h is real and small, the derivative is approximated by $f'(x) \approx \mathrm{Im}\, f(x+ih)/h$. Unlike the use of a finite difference formula, h can be chosen to be *very* small without the risk of cancellation errors. The complex step is very straightforward to implement in languages that support complex arithmetic.

"Attention is now being paid to the **complex step method**"

The term *automatic differentiation* (AD) generally applies to techniques that produce, from a function evaluation software component, a computational scheme, also implemented in software, that calculates the derivatives. The execution of any program, no matter how complex, is built up from a sequence of elementary arithmetic operations (e.g. add, multiply) or elementary function evaluations (e.g. sin, exp). This implies that in any program, a function can be split into

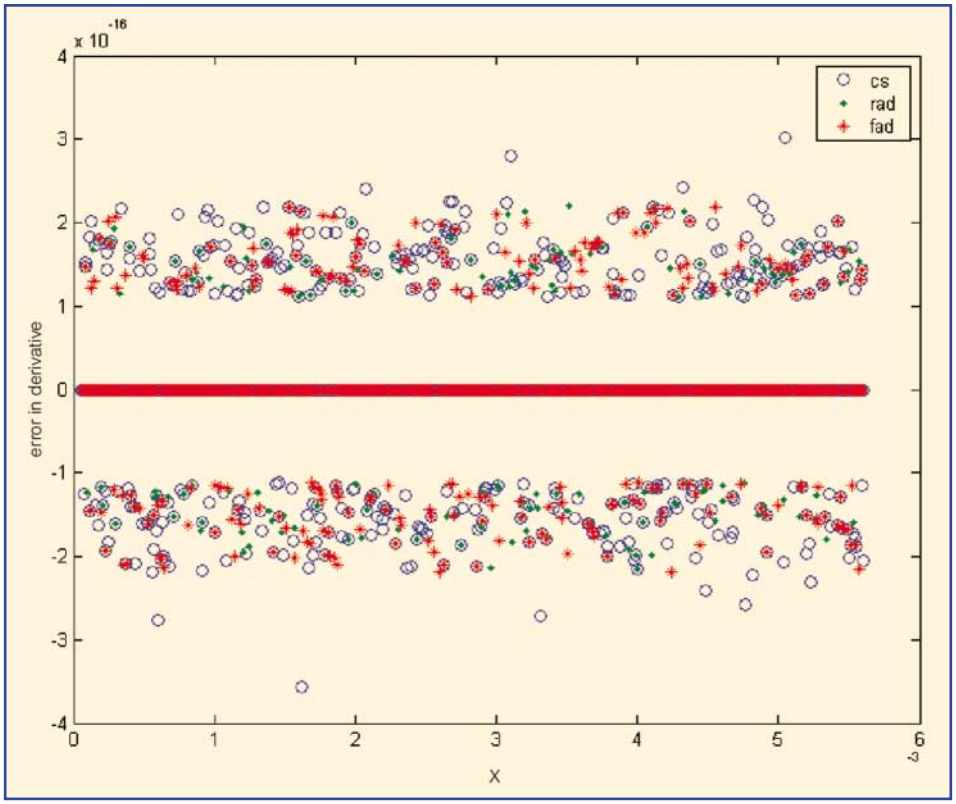


Figure 5. Error in computing the derivative of $f(x) = \sin(1/x)$ for CS, FAD and RAD. The band at zero indicates that the computed derivatives agreed with the evaluated analytical derivative to machine precision.

atomic operations that involve no more than two variables at a time. In automatic differentiation, the rules of calculus are applied to these atomic operations and are combined appropriately according to the algorithmic specification of the function.

The AD approach can be implemented in two ways. In *forward automatic differentiation* (FAD), the derivative information is accumulated in the same order as for the function evaluation. In *reverse automatic differentiation* (RAD), the algorithm first performs a forward sweep for the function evaluation, storing, at the same time, the information required for the derivative computations. Then, in a second sweep, the algorithm steps in the reverse direction to that of the function generation and uses the information saved from the forward sweep to calculate the gradient. For many problems, RAD can be considerably more efficient than FAD.

In terms of software engineering, AD can be implemented using *operator overloading* and *source-to-source transformation*. In the operator-overloading approach (for languages such as Fortran 90 that support it), the basic arithmetic operations and intrinsic functions are re-assigned to calculate the corresponding derivatives in addition to the function values so that, at the end of the computation, the derivative information is also available. In source-to-source transformation, the source code for the function evaluation is analysed and extended to produce source code to evaluate the derivatives. Generally, the implementation of source code transformation requires much more effort than the operator overloading approach.

When comparing the behaviour of the finite difference approach (FD), the complex step method (CS) , FAD and RAD on the function $f(x) = \sin(1/x)$ with derivative, $f'(x) = -(1/x^2)\cos(1/x)$, the finite difference method is seen to be much less accurate than the other three approaches.

Figure 5 illustrates the error for the CS, FAD and RAD methods.

Further information on automatic differentiation can be found in report CMSC 26/03 *Automatic differentiation techniques and their application in metrology*, by R Boudjemaa, M G Cox, A B Forbes and P M Harris.

For further information contact
Alistair Forbes, extension 6348
e-mail: alistair.forbes@npl.co.uk

SSfM-3 Formulation Update

The draft for public comment of the Software Support for Metrology programme for 2004-2007 (SSfM-3) is now available. This document is the result of six months' consultation and development.

The consultation phase culminated in a presentation to DTI and the Measurement Advisory Committee Working Group (MAC WG) at their annual review meeting on 22 May. It had been intended that the cost of the programme at that stage would be 170% of the target of £3M, but there had been so many worthwhile ideas put forward that the programme as presented was over 200% of the target.

The draft programme for public comment was reduced to 115% of the target based on the scores the MAC WG members gave to each topic presented. These scores were used in combination with scores from the completed questionnaires, received from 62 different respondents during the consultation, and with information on requests for joint projects from other National Measurement System (NMS) programmes. All this information was used to select the most strongly supported set of topics, combining or discarding the other topics as appropriate.

These selected topics were then grouped appropriately into themes, sub-themes and projects in the Public Comment Document, now available on both the DTI and SSfM websites. Each theme was provided with an aim, background and rationale, and a more detailed aim was given for each sub-theme. The deliverables and rationale were identified for each project. The projects were classified according to type of activity:

- Development
- Research
- Maintenance
- International liaison
- Support for international traceability and regulation
- Knowledge transfer
- Programme management

Figure 6 shows the make-up of the programme by these activity types.

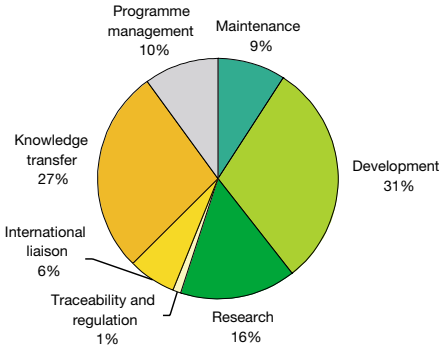


Figure 6. Composition of SSfM-3 by activity type

The themes selected for SSfM-3 in the draft programme are as follows:

- Modelling tools and techniques
- Uncertainties and statistical techniques
- Software development, testing and validation
- Applications and supporting techniques
- Generic knowledge transfer
- Programme management

Figure 7 gives the division of the programme by theme, showing that there is a good balance between the themes.

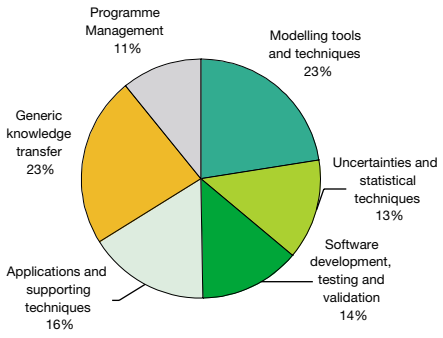


Figure 7. Composition of SSfM-3 by theme

The first three themes continue with topics that have been central to the first two SSfM programmes, whereas the fourth theme tackles the following newer areas:

- Data curation
- Signal processing
- Internet-enabled metrology
- Support for soft metrology

The public comment period ended on 29 September and the comments are being considered in producing the final draft of the programme. The formulation will be brought to a conclusion in the "Appraisal" meeting, on 6 November, when DTI and the MAC WG will decide what to cut from the final draft programme to create the final programme. They will also decide the projects to be put out to competitive tender.

The final programme will be presented, together with the supporting business case, to the Minister for approval, in time for the programme to start in April 2004.

For further information contact:
Dave Rayner, extension 7040,
e-mail: dave.rayner@npl.co.uk

Measuring Properties of Optical Fibres

Optical fibres are used to carry signals in many applications. For example: links between telephone sub stations, LANs, cable TV and CCTV. Attenuation uniformity and loss of a fibre can be measured using an Optical Time Domain Reflectometer (OTDR). It is important that the OTDR is calibrated to give accurate results. In some cases it would be convenient if this could be done in situ.

NPL has been taking a lead in the calibration of a number of different instruments over the internet. A demonstration of the technology, based on previous work by NPL and Adelard to develop a generic internet solution for any instrument, has been applied to the calibration of OTDRs.

The generic solution consists of a server connected via a network to a PC that contains some client software. The PC is connected to the instrument to be calibrated, and to any other equipment necessary for the calibration. In this case, the OTDR and associated instruments are connected as shown in the diagram.

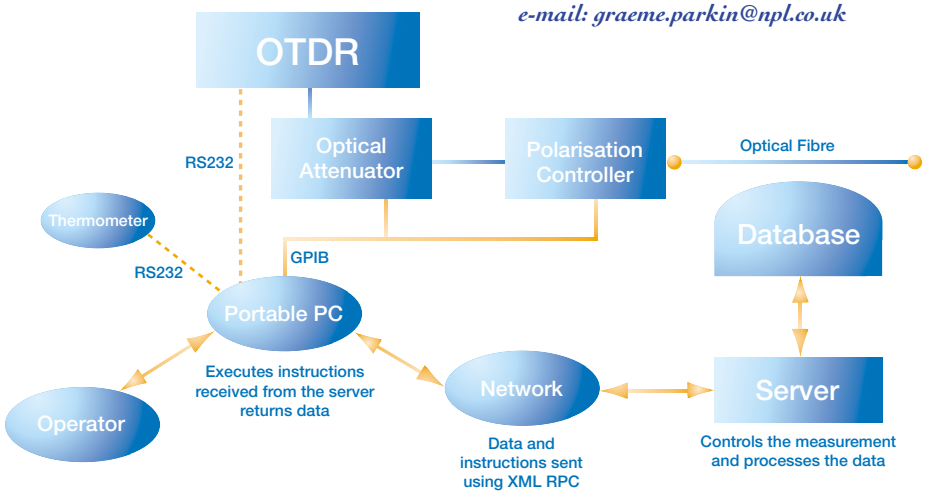
Software in the server controls the calibration by sending instructions for the operator, and commands for the OTDR and attached instruments, to the client. The client software delivers these instructions and commands, and returns the responses to the server for processing. All measurement data is stored in the server in a MySQL database, and all calculations are done on the server.

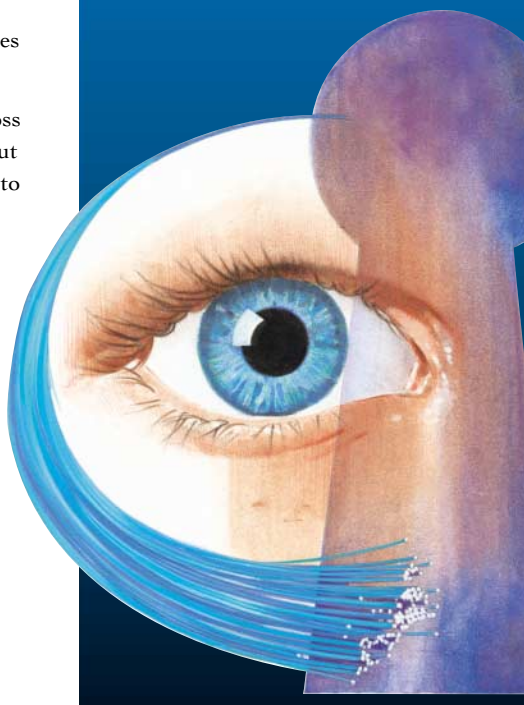
The client software is written in Visual Basic, and the server is written in PHP, while the commands sent from the server to the client are written in VBScript.

The current client software can have any number of RS232 or GPIB interfaces, and further interfaces are being considered. The demonstration of the OTDR calibration involves two RS232 and two GPIB interfaces being used at the same time.

The same client software will be used across the various internet calibration services, but the server software will change according to what is being calibrated.

For further information contact
Graeme Parkin, extension 7104
e-mail: graeme.parkin@npl.co.uk

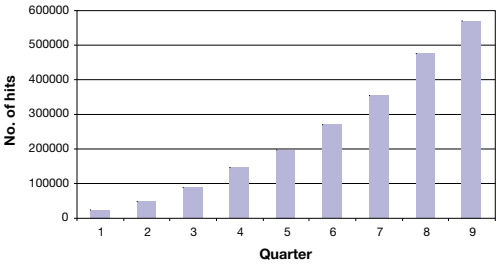




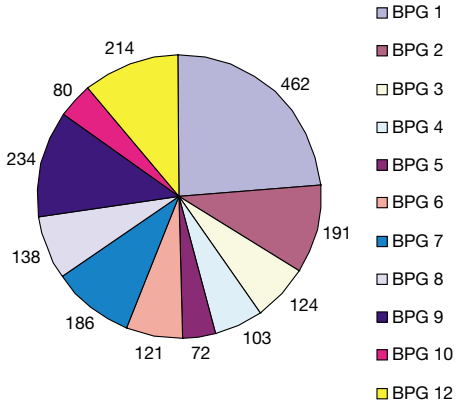
SSfM website exceeds half a million hits

The growth in the number of accesses to the SSfM-2 website continues: we reached the 500,000 mark in April 2003 and had already exceeded 600,000 by August.

The total number of accesses to our documents area during this period was in excess of 88,000. Best Practice Guide No. 1, Measurement System Validation, continues to enjoy its usual popularity.



Cumulative hits for SSfM since April 2001



Number of downloads per best practice guide, quarter 9

The website remains the primary method for distributing output from the programme.

For further information contact
Bernard Chorley, extension 7050
e-mail: bernard.chorley@npl.co.uk

Forthcoming Events

10-11 November 2003, NPL Measurement Software Design and Development, course for software developers

2 December 2003, NPL Testing Numerical Correctness of Scientific Software Course for scientists and engineers

19-20 January 2004, Uncertainty Evaluation and Associated Statistical Modelling, one-and-a-half-day advanced training course

27-28 January 2004, Scientific Computing in Fortran 90/95, two-day course

Visit www.npl.co.uk/training/schedule.html

Contact ssfm@npl.co.uk or Jan Kane on +44 20 8943 7100

Centre for Mathematics and Scientific Computing (CMSC) Making contact

You can contact any of the experts directly by using the direct dial number plus the extension or via e-mail.

Direct line +44 20 8943 + (extension)

Head of CMSC
Dave Rayner ext 7040 e-mail: dave.rayner@npl.co.uk

Software Support for Metrology Club
Wendy Johnson ext 6106 e-mail: ssfm@npl.co.uk
SSfM website: www.npl.co.uk/ssfm

If you have a general enquiry or do not know who you should contact please call our general enquiries number and we will be pleased to help you.

General CMSC enquiries
+44 20 8943 7100 (direct line)
+44 20 8977 7091 (facsimile)
Website: www.npl.co.uk/scientific_software

General NPL Helpline
For enquiries to NPL outside the scope of CMSC, please use:
+44 20 8943 6880 (NPL Helpline)
+44 20 8943 6458 (Helpline facsimile)

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