NPL Report CMSC 41/04

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

Testing Continuous Modelling Software

T J Esward, E G Johnson, and L Wright

March 2004

Testing Continuous Modelling Software

T J Esward, E G Johnson, and L Wright
Centre for Mathematics and Scientific Computing

March 2004

ABSTRACT

A methodology has been developed for testing the numerical correctness of software used for continuous modelling. This report describes the methodology and applies it to some commonly used software packages.

The methodology is partly based on the methodology for testing discrete modelling software developed as part of the first Software Support for Metrology programme, but has been further developed to address issues that are specific to continuous modelling software.

The applications of the methodology described in this report concern various aspects of some popular finite element packages. The methodology is applied to other, more specialised, software in a companion report, “Testing Continuous Modelling Software: Three Case Studies”.

ISSN 1471-0005

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

Approved on behalf of the Managing Director, NPL
by Dave Rayner, Head of the Centre for Mathematics and Scientific Computing

Contents

1 Introduction

It has been recognised in previous Software Support for Metrology (SS*f*M) programmes [1] that there is a growing need to test the numerical correctness of scientific software used in metrology. This recognition has led to the development of a testing methodology [1], and application of that methodology to a number of common functions in software packages used widely in metrology [2-7]. The results of this work have clearly demonstrated the importance of rigorous and impartial testing. However, the work so far has concentrated largely on functions used in the modelling and analysis of discrete data.

The aim of the work described here is to develop an equivalent methodology for testing continuous modelling packages, and to apply the methodology to test software packages commonly used in metrology. The applications described in this report concern various aspects of some finite element packages that are popular for metrology problems. The methodology is applied to other, more specialised, software packages in a companion report, "Testing Continuous Modelling Software: Three Case Studies" [8].

The work was motivated in part by the results of a survey of SS*f*M club members. The survey asked club members for details of their continuous modelling work, including details of their validation and testing procedures. Just under half of the respondents had never tested the correctness of the software they used, and none of the respondents had used benchmark problems for testing. Those who had tested the accuracy of software had either compared results with results obtained from another software package, compared results with an analytical solution, or used the files supplied with the software to test the accuracy of the software as well as its installation.

It is understandable that users of expensive software packages expect the packages to have been thoroughly tested prior to their release, but previous work [1-7,9] has shown that the algorithms implemented in software are not necessarily best practice or fit for purpose under some circumstances. The work described in the article by Hatton [9] concerns the testing of a range of software for solving earth science problems and found a startling number of errors and inaccuracies in commercial software. The work on testing discrete modelling software found that even software from reputable developers produces poor results for simple calculations under some circumstances.

It should be noted that throughout the following, the aim of the testing is to establish the numerical accuracy of the software rather than its usability, speed, or visualisation aspects.

1.1 Structure of this report

A brief discussion of work on testing methods for discrete and continuous modelling software is presented in section 2. This discussion includes existing methodologies and some useful sources of test problems.

In many ways, the testing of software for continuous and discrete modelling should not differ significantly. In both cases, input data are used to generate test results, which are then compared to reference results according to some metric. In both cases the software is treated as a black box. However, there are several aspects of continuous modelling software that require extensions and alterations to the software testing methodology for discrete modelling. These features will be discussed in section 3.

Section 4 discusses the methodology developed for continuous modelling software. This section also describes other techniques for testing continuous modelling software that are supplementary to the main methodology. Section 5 describes the results of applying the methodology to several popular finite element packages. Further examples of testing methods applied to continuous modelling software can be found in the case study report [8].

2 Key features of continuous modelling software

Continuous modelling software can be split into two broad classes: software that takes the physical problem as its starting point, and software that takes the mathematical formulation as its starting point. For example, many finite element packages intended for use by engineers model heat transfer through a structure but do not mention the pde that describes conduction, whereas software for solution of pdes and odes that has a more mathematical origin, such as the NAG routines in chapters D02 and D03 or the Matlab functions for pde and ode solution, tends not to describe the input data in physical terms.

Whilst these two classes of software do not need to be tested differently, care needs to be taken when defining test problems as physical problems to ensure that the definition is mathematically unambiguous and precise. Some definitions of test problems in physical terms may be difficult to rewrite in a suitable form for pde solvers. For instance, stress analysis of plastic materials is best described using a system of pdes to define the stress behaviour under load and a nonlinear material model to define the stress in terms of the displacements. Inserting the material model into the system of pdes leads to a complicated nonlinear system that is likely to be difficult to use as an input to a general pde solver. This difficulty can limit the range of tests that can be applied to some software packages.

Many continuous modelling methods are derived using the same broad procedure. This procedure can be summarised as follows:

1. Split the problem domain into small units (elements, volumes, areas, etc.), called the mesh. This step may involve approximating the true problem geometry.
2. Approximate the continuous differential equation on each of the small units to produce a local discrete approximation to the problem.
3. Assemble all of the discrete approximations into a large system of simultaneous equations (usually linear equations, but not always).
4. Solve the large system.

The procedure is used in finite difference methods, FE analysis, boundary element methods, and finite volume methods such as those used for computational fluid dynamics and electromagnetic modelling. In general, as the number of units in the mesh increases, the model results converge to the solution of the differential equation (although there are some exceptions to this claim: see section 7.1.3 of [20]).

The use of this procedure means that a choice has to be made when testing many continuous modelling software packages: either the test definition must include the mesh specification, or it must be specified that the mesh is to be refined until the results have converged to some degree of accuracy. If the test is to be used to compare several packages, it is preferable to specify the mesh as this ensures that the test is fair and unambiguous. If the mesh is not defined explicitly, an experienced user may be able to obtain better test results than an inexperienced user, which means that the user is being tested rather than the software. If the test is to compare the solution obtained from a single package to an analytical solution of a problem, it is more reasonable to attempt to obtain converged results because the accuracy of the software is of interest and so it is desirable to obtain the best results possible. As a compromise, it may be possible to identify a mesh that leads to a solution of known accuracy for the problem, and state that this mesh should be used in the test.

As the use of continuous modelling software has become more wide-spread, there has been a drive towards making software more user-friendly for users with little mathematical background. Of particular relevance to the testing of such software is the increased use of automated meshing tools and adaptive meshing in finite element software packages.

Automated meshing tools allow the user to define the basic geometry of the problem of interest and leave the software to generate a mesh that fills the domain of the problem, usually consisting of triangular or tetrahedral elements. These tools are particularly useful for creating detailed meshes of objects with complicated shapes such as are often used in engineering and manufacturing applications.

Adaptive meshing is a technique that alters the mesh as it calculates the solution. The basic procedure is:

- calculate a solution on a mesh,
- use *a posteriori* error estimation methods to estimate the accuracy of the solution throughout the domain,
- create a new mesh by increasing either the number of elements or the order of approximation in the regions where the error estimate is largest, and
- calculate a solution using the new mesh.

These steps are repeated until the error estimate throughout the domain drops below some user-defined tolerance level.

Both of these techniques aim to produce accurate results with the minimum of user input. Whilst this aim is very useful for the inexperienced user, it means that the user has little or no control over the mesh. If a software package gives the user no control over the mesh, the software cannot be tested using tests with a defined mesh, thus limiting the range of tests that can be applied to the software.

Some FE packages use iterative techniques to solve the large systems of equations as mentioned in step 4 above. The solutions generated by these techniques are considered to have converged if the maximum change in all of the solution values between two consecutive iterations is less than some tolerance parameter times the solution values. The user generally does not supply the tolerance parameter directly unless a solution of particularly high accuracy is required. This use of iterative schemes means that sometimes users have good control over the accuracy of the solutions to the large system of equations.

3 Existing test methods

As was mentioned in the introduction, extensive work on testing has taken place in the previous SS*f*M programme. One of the main results of this work was a methodology for testing spreadsheets and other packages used in metrology. This methodology views the software under test as a black box, which means that the first requirement of the methodology is a specification of the software function being tested. The specification takes the form of an unambiguous definition of the inputs and outputs of the software, and the mathematical function relating the two.

Once the specification has been recorded, reference input data sets and corresponding results can be developed. The main considerations in designing reference data sets are

1. The data sets should be capable of identifying deficiencies such as instabilities in the algorithms implemented by the software.
2. The data sets should have known properties such as condition number or "degree of difficulty".
3. The reference data sets should be reasonably similar to input data sets that are used in metrology applications.
4. The data sets should provide adequate coverage of the full space of input data.

It may not be possible to meet all of these needs under all circumstances. In particular, the degree of difficulty may be difficult to quantify for some problems. The methodology recommends the use of reference data generators wherever possible. Reference data generators produce input data of known "degree of difficulty" from reference results specified *a priori*. For instance, if a routine for calculating standard deviation were to be tested, the reference data set generator would produce data sets of known size, coefficient of variation, and standard deviation, and the coefficient of variation would be used to measure the "degree of difficulty". The methodology using reference data set generators can be summarised by the diagram in figure 1.

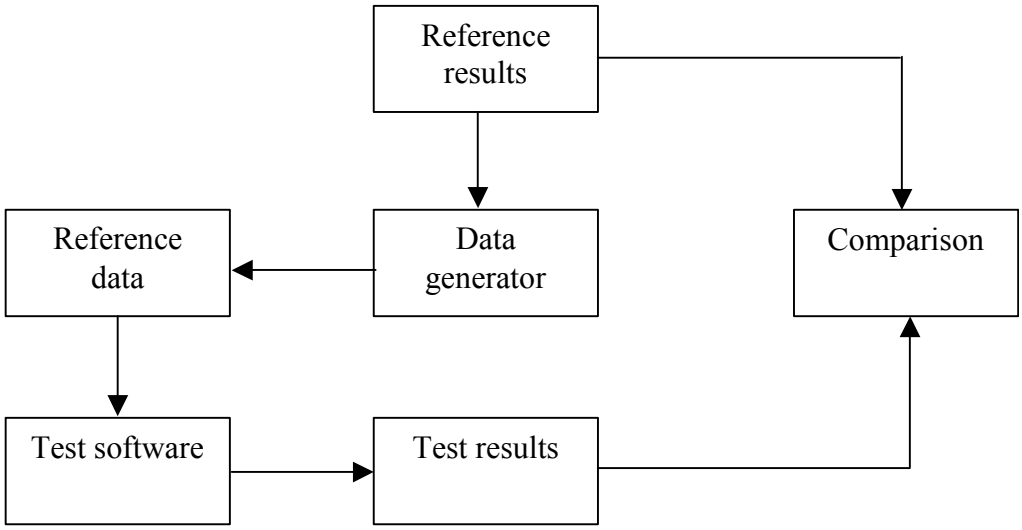

Figure 1: Testing methodology using reference data generators

Once the reference data and results have been obtained, the test results can be obtained by applying the software under test to the reference data. The comparison between the reference results and the test results then requires a performance metric. In general, this performance metric is either an absolute or relative measure of accuracy, or a

performance measure defined in terms of the difference between the test and reference results, the computational precision, and the condition of the problem. Computational precision, η, is a property of finite precision arithmetic, and for the commonly-used floating point arithmetic η is defined as the smallest positive representable number u such that the value $1 + u$, computed using the arithmetic, exceeds unity. For the many floating-point processors which today employ IEEE arithmetic, $\eta = 2^{-52} \approx 2 \times 10^{-16}$. The condition of a problem is a measure that describes the sensitivity of the exact solution to a problem to changes in the problem or its input data. If small changes in the problem or its input data lead to small changes in the solution, then the problem is well-conditioned. If small changes in the input data lead to large changes in the solution, then it is ill-conditioned for the data. Metrics will be discussed further in section 4.4

An alternative to the use of data set generators also mentioned in [1] is the use of reference software. Reference software is software that is known to be accurate, stable and reliable. Such software is generally difficult to develop and test, and invariably requires considerably more effort to develop than the corresponding data set generator. If reference software is used, the procedure is to use the same input data in the software under test and the reference software, and compare the two sets of results using some metric.

An example of an alternative to this black-box methodology is given by NAG's software testing procedures. As NAG develop software, its software testers have full source code access and so do not regard the software as a black box. The testing procedures test the implementation of a specific algorithm for calculating some function and so the software is specified by its inputs, outputs, and the algorithm that produces one from the other. The same testing procedures are applied to all NAG's software, independent of its purpose. The testing methods generally consider the test to be passed if the test result is accurate to computational precision.

The tests take a variety of forms, including

- checks to ensure that invalid input data causes a warning or error message,
- convergence tests for ordinary and partial differential equation (ode and pde) solvers, usually of the "halve the step size and re-run" type,
- comparisons with asymptotic solutions for any special values of the input data
- use of problems that are known to be exact for a given algorithm, as will be explained further below.

The results of routines for numerical solution of odes and pdes are by their nature approximate. However, generally an expression for the approximation error can be derived, for example from a Taylor expansion. This means that often a test problem can be developed for which the approximate numerical method produces exact results. As an example, consider the common three-point approximation to the second derivative of a function $f(x)$:

$$f''(x) \approx \left(f(x+\Delta x) + f(x-\Delta x) - 2f(x)\right)/\Delta x^2 .$$

This expression is exact for polynomials of degree 2 or less, so if an algorithm using this approximation is applied to the ode

$$f''(x) = 2a, \quad 0 \le x \le 1, \quad f(0) = c, \quad f(1) = a + b + c,$$

which has the exact solution $f(x) = ax^2 + bx + c$, then the results should be exact for any

values of the constants a, b, and c.

NAG choose input data according to similar criteria as those mentioned above as part of the SS*f*M methodology. To ensure that the software is tested using problems with a range of degrees of difficulty, NAG have a library of routines to generate matrices with known properties such as condition number for testing linear algebra routines.

3.1 Sources of test problems

There are several useful sources of test problems for general pde and ode solvers, and of problems for specific solution methods. DETEST [10] is a standard set of test problems for ode solvers, and many textbooks on solution of differential equations [11, 12] include test problems. There are good online resources for ode problems [13-15]. The Computational Differential Equations (CDE) forum [14] is compiling an online database of test problems and reference solutions for odes and pdes.

Finite element (FE) analysis is probably the most widely-used continuous modelling method, and hence test problems for FE software packages are useful to a wide range of applications. One of the first tests suggested for assessment of the formulations of finite elements was the patch test [16]. This test checks the consistency of the elements with the problem in question. Consistency of an approximation and a problem ensures that as the cell size tends to zero, the approximate solution converges to the solution of the problem, which is clearly a desirable property in an approximation method. It has been extended since its original proposal, and it has been shown [17] that the patch test can be applied to other continuous modelling methods such as finite difference and finite volume methods, and that it has properties that are beneficial for establishing convergence behaviour and error estimates.

A standard set of tests to assess the accuracy of the individual element formulations was suggested by MacNeal and Harder [18]. This set of tests include tests of beam, membrane, plate, shell, and solid elements. All the tests are static stress analyses, and the reference results are taken from analytical solutions to the equivalent mathematical problems.

An excellent source of tests for finite element software is the NAFEMS benchmarks. NAFEMS [19], the National Agency for Finite Element Methods and Standards, was set up in 1983. NAFEMS has been operated as a not-for-profit company since 1990 and has members world-wide. The aim of NAFEMS is “to promote the safe and reliable use of finite element and related technology”. To achieve this, NAFEMS has developed several sets of benchmark problems to test and validate all types of finite element analyses, and these benchmarks are constantly being updated and reviewed as FE technology advances. In addition to its benchmarks, NAFEMS promotes best practice for all aspects of finite element analysis from choosing software to interpreting results. This promotion is undertaken through seminars, “How To …” guides, and a quarterly email newsletter to its members.

The benchmarks cover many areas of application, including non-linear materials, contact problems, damage and delamination simulations, vibration and acoustics problems, and thermal analyses. Many software houses now use NAFEMS benchmarks as a part of their testing procedures. Some examples from their standard benchmarks have been used to test popular FE packages in section 5 of this report.

In addition to these resources, many software manufacturers supply a set of test problems with their software, and it may be possible to use these problems for testing

other similar software packages. It should be borne in mind when using such problems that sometimes the problems are designed to test the installation of the software rather than its accuracy, and wherever possible the background to the test problem should be fully understood before it is used.

4 A methodology for testing continuous modelling software

The methodology described here aims to take relevant parts of the NAG approach to software testing and the methodology developed for discrete modelling software as part of SS*f*M-1 [1], as discussed in section 3, and use them to produce an approach to testing suitable for continuous modelling software. The methodology developed for discrete modelling software is largely suitable for testing continuous modelling software, but there are some features of continuous modelling software that make modifications to that methodology necessary.

The starting point for the methodology for testing continuous modelling software is the same as that for testing discrete modelling software. The software is to be treated as a black box, and the aim of the test is to assess the ability of the software to generate a particular result rather than to check the implementation of a specific algorithm. The most commonly used proprietary continuous modelling software is black-box, so this approach is reasonable.

As is the case for much modern software, many continuous modelling software packages are multi-purpose and can solve a range of different pdes, so the first step is to define the aspect of the software that is to be tested. One possible choice is to decide which generic pde or ode is to be solved, on the assumption that the full test problem will be defined by description of the solution domain, the values of any input data, and the boundary and initial conditions.

It was mentioned in section 2 that the majority of continuous modelling methods create a mesh, approximate the differential equation on the mesh to form a system of simultaneous equations, and then solve the system to obtain the results. The stages in this process which are most likely to produce numerical errors are the approximation of the differential equation within each element, and the assembly and solution of the system of equations. Thus there are two stages in the process that need to be tested. With this in mind, the testing methodology aims to test these two stages as independently as is possible. The methodology is illustrated in figure 2, and the details of the approach will be explained in the rest of this section.

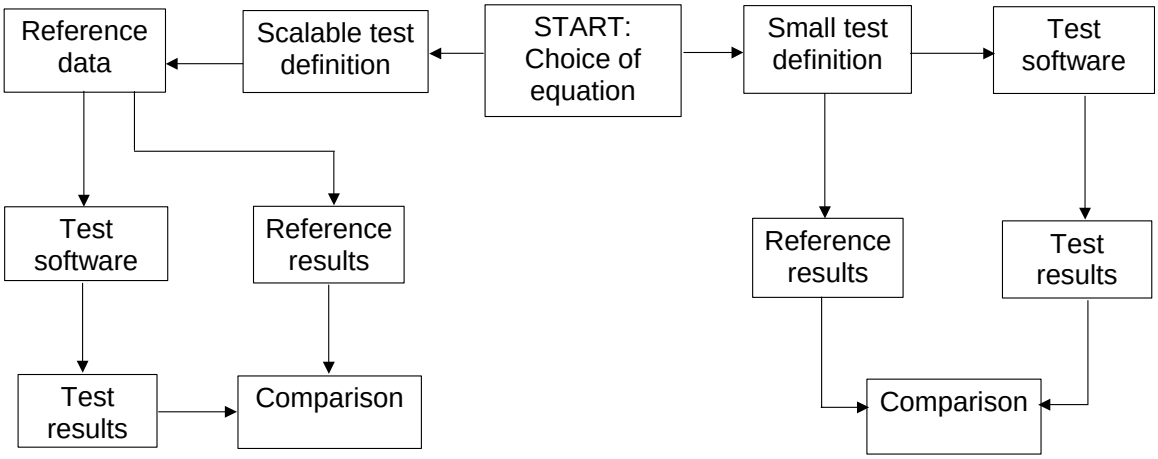

Figure 2: Testing methodology for continuous modelling software, reflecting extensions to the discrete testing methodology that are necessary due to the two-stage calculation process in continuous modelling and the importance of defining which equation is of interest before beginning.

The first step in defining a test is identifying the pde of interest. This may be defined in physical terms, provided the definition is clear and unambiguous, but would be better

expressed in mathematical terms. Initially, the boundary conditions and solution method need not be defined.

Next, the small-scale and scaleable tests need to be defined. These tests are discussed in more detail in sections 4.1 and 4.2. In general, the definition of the test will require a clear statement of the following:

- Pde of interest,
- Geometry of the domain,
- Material properties within the domain,
- Boundary conditions and loading conditions,
- Statement of meshing: either the mesh is explicitly defined or a statement such as "mesh to be refined until the results converge" is made.
- Statement of the results that are to be calculated including the accuracy to which they are to be stated.

This outline will apply to small-scale and scaleable tests. The reference results should be readily available for the problem described by the test definition. Generation of reference results is discussed further in section 4.3. Once the test results have been generated, a metric for the comparison stage must be chosen. Some possibilities are discussed in section 4.4.

There are other techniques complementary to this methodology that may be of benefit when testing. These techniques are outlined in section 4.5.

4.1 Small-scale tests

The testing of the formulation of individual elements can be separated from the testing of the solution of the full system by using well-conditioned tests that contain only a few elements. These tests are called "small-scale tests". If the number of elements is small and the problem is well-conditioned, the system of equations will be small and well-conditioned and so will not be likely to cause errors in the system solver routines. Thus any errors that occur will be due to element formulation rather than system solver error.

The NAFEMS benchmarks [19] and the problems described by MacLean and Harder [18] provide several examples of such tests. Often the example problems supplied with software are sufficiently simple to be suitable on grounds of size, but they may not be sufficiently challenging to make good test problems. Small-scale tests can also be generated for finite difference methods and for many finite-volume methods, but they are less suitable for boundary element methods, some finite volume methods, and any software that requires the use of an automatic mesh generation tool.

Care must be taken when defining small-scale tests. Clearly the mesh must be defined as part of the test definition. Since the test is to involve a small number of elements, it is unlikely that the solution will have converged to the solution of the differential equation, unless the problem is chosen such that the solution can be described exactly using the chosen elements.

As an example, consider a bar of length h made of a uniform isotropic elastic material with Young's modulus E and Poisson's ratio υ, fixed in x at $x = 0$ and displaced by an amount εh in the x direction at $x = h$. Then it can be shown that

$$u = \varepsilon x, \quad v = -\nu \varepsilon y, \quad w = -\nu \varepsilon z, \quad \sigma_{xx} = E\varepsilon,$$

where u, v, and w are the displacements in the x, y, and z directions and σ_{xx} is the axial stress, satisfies the relevant equilibrium conditions and material formulation. This solution has linear displacements in each of the coordinate directions and a constant stress throughout the domain. These features mean that a model of this problem using linear finite elements ought to produce the correct result independent of the number of elements in the mesh, making it a good example of a small-scale test that can be compared with an analytic solution.

4.2 Scalable tests

The second type of test contains more elements as it aims to test the assembly and solution algorithms of the package. The aim when generating the large-scale tests is to identify a function of some of the input data that effectively determines the solution, so that individual input data points can be varied without affecting the solution. These tests are called "scalable" tests. Ideally, this type of test would consist of a family of problems, all generating the same results, with different input data. The intention is to create a set of tests that explore the full range of the input data with reference results that are "normal" sized, i.e. typical of the results that the software might reasonably be expected to produce. This is a similar aim to the use of data set generators in [1].

As an example, consider the equation for heat conduction in a solid,

$$\rho c_p \frac{\partial T}{\partial t} = \lambda \nabla^2 T \, .$$

Provided that the geometry, initial conditions and boundary conditions are identical, two problems with different values of λ, ρ, and c_p will have the same solution if they have the same value of $\lambda/(\rho\ c_p)$, called the thermal diffusivity. Hence a family of problems with identical solutions can be generated by defining a suitable domain and boundary conditions and varying λ, ρ, and c_p whilst holding $\lambda/(\rho\ c_p)$ fixed. In this case, care needs to be taken to ensure that boundary conditions are chosen suitably, since often thermal boundary conditions depend on λ.

Reference results for scalable tests are almost always derived from analytic solutions to problems. This choice of reference results means that the discretisation error will always affect the difference between the test results and the reference results. One way to minimise the effects of the discretisation error is to identify a mesh that produces results that are converged to some accuracy for one package and to define the test problem to include that mesh specification. Another way to minimise the effects is to state in the test definition that the mesh is to be refined until the results have converged. Advice on estimating the size of the discretisation error is given in [20].

If several packages are to be tested and the mesh to be used is specified, the effects of the discretisation error are less important. If the same formulation is used in each package and the same mesh is used throughout, then the implementations of the model in the packages should produce the same discretisation error throughout. Hence comparison of the metric values will provide information about the relative performance of the packages but not about its absolute performance unless the mesh is expected to produce results to some known accuracy.

If reference results are not available for some problem, but the pde or ode depends on some function of the input data as described above, then it is still possible to investigate

the behaviour of the software package. The results of tests using input data that produce a fixed value of the function will illustrate the behaviour of the software as the input data vary without giving an absolute measure of the accuracy of the software.

4.3 Generation of reference problems

Reference problems must be defined clearly and unambiguously. In particular, careful consideration needs to be given to choice of reference result. Continuous modelling software produces results over the whole of the domain of interest, and accuracy of the results at one point does not guarantee accuracy elsewhere. This means that it is often worthwhile specifying a set of reference values rather than a single result, so that the overall performance of the software can be assessed. Similarly, some calculations result in more than one result at a point, for instance a static stress analysis usually generates six stress components and three displacements throughout the domain, so it is worth checking all results produced if reference values are available.

Some useful sources of reference problems were given in section 2.1. An alternative method of generating reference results that is common for continuous modelling problems is the use of independent methods to produce solutions to the same problem. In some cases, several different continuous modelling methods can be used to solve the same problem, and the derivations of these methods are independent and rely on different local approximations to formulate the problem. For instance, many problems that can be solved using finite difference methods can also be solved using finite element analysis, and each of these broad classes of method includes methods that use different local approximations. The existence of several independent methods means that if more than one method produces the same results to within some accuracy, then these results can be regarded as reference results. This approach has been used in two of the case studies described elsewhere [8].

Several problems have been encountered during the generation of the reference results used in this report and its companion report [8]. These problems have come from two main sources: ambiguity of terms and unclear statements of assumptions. Both sources cause problems when attempting to obtain analytic reference solutions from literature without going through the detailed derivation. These problems illustrate the importance of supplying an unambiguous statement of the test problem.

Ambiguity of terms causes problems when the input data required by the software and the symbols used in the notation in the literature seem to be identical but do not mean the same thing. This is a pitfall for a software user who is inexperienced in the area of physics under test. The best way of avoiding these problems is to consult an expert regarding the meaning of all symbols in any references, or to check the detailed derivation starting from the original pde.

One example of ambiguity of terms comes from one of the case studies described in [8]. Maxwell's equations for the propagation of electromagnetic waves can be written

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \qquad \nabla \times \mathbf{B} = \mu\varepsilon \frac{\partial \mathbf{E}}{\partial t} + \varepsilon \mathbf{J}$$
$$\nabla . (\varepsilon \mathbf{E}) = \rho \qquad \nabla . \mathbf{B} = 0$$

where **E** is the electric field measured in Vm^{-1}, **B** is the magnetic field measured in T, μ is the magnetic permeability measured in NA^{-2}, ε is the electric permittivity, measured in Fm^{-1}, **J** is the vector current density measured in Am^{-2}, and ρ is the charge density in Cm^{-3}.

However, because μ and ε generally have extremely small values, it is usual to write $\mu = \mu_0\mu_r$ and $\varepsilon = \varepsilon_0\varepsilon_r$, where μ_0 and ε_0 are the permeability and permittivity of free space respectively, and the subscript r denotes relative permeability or permittivity. These relative values always have a modulus greater than 1.0 which generally makes calculations involving them more accurate. However, the use of relative values introduces ambiguity since authors do not always include the subscripts or state that they are using relative values, and so care needs to be taken when using solutions including these quantities as reference solutions.

The generation of analytic solutions to differential equations almost invariably requires simplifications and assumptions to hold, which make the problem tractable to analytical methods but not descriptive of a real physical problem. If the authors do not state explicitly the assumptions that have been made in the generation of an analytic solution to a problem, the test problem cannot be defined such that those assumptions hold, and so the test problem and the analytic reference results are unlikely to compare well. This problem can be identified by checking the analysis that has generated the solution and ensuring that the assumptions and simplifications are clear.

In some cases it may be difficult or impossible to reproduce the assumptions required to derive the analytic solution in the test definition. For instance, many analytic solutions to stress problems are based on the assumption that the outer surface of the geometry is stress-free, but this is not a condition that can be enforced in most finite element packages. This difficulty may mean that the results most likely to be affected by the assumption cannot be used as reference results. For an example of this restriction, see section 5.2.2. The use of such assumptions may mean that only part of the analytic solution can be used as reference results.

4.4 Metrics

The comparison step of the testing methodology requires a metric in order to give an objective comparison between the test result and the reference results. Metrics have been discussed at length in the report on the methodology for testing discrete modelling software [1], and so the key points will be summarised here without going into detail.

There are two key points that need to be considered when choosing a metric for testing continuous modelling software that are less important when considering discrete modelling software. The first point is that the reference results are likely to be a set of results rather than a single value, because it is often important to look at the whole solution rather than a point value, as was mentioned in section 4.3.

The second point is that the test results are the solution to a discrete approximation to a differential equation, and so there will always be a difference between the test results and any reference results derived from an analytic solution to the differential equation. This difference can be estimated and reduced but it cannot be eliminated altogether, which means that differences between the reference and test results will be due to discretisation errors as well as numerical errors. Whilst this does not preclude the use of analytic solutions as reference results, it does mean that the existence of discretisation errors should be borne in mind when interpreting test results using a metric.

The methodology for testing discrete modelling software [1] defined three types of metric for measuring the departure of the test results from the reference results:

- the difference between the test results and the reference results, an absolute measure of accuracy,

- the number of figures of agreement, a relative measure of accuracy, and
- a performance measure that accounts for factors including the computational precision of the arithmetic used to generate the reference and test results and the condition of the problem.

Wherever possible, the metrics that follow should be applied to results that are all of approximately the same magnitude. If the metrics are applied to a set of results that span several orders of magnitude, then the metric may be dominated by the error in the results of large magnitude. If it is not possible to scale the results to avoid this problem, the results should be looked at individually.

Let $\mathbf{y}^{\text{test}}$ be the test results generated for input data $\mathbf{x}$, $\mathbf{y}^{\text{ref}}$ the corresponding reference results, and $\Delta\mathbf{y} = \mathbf{y}^{\text{test}} - \mathbf{y}^{\text{ref}}$. If

$$RMS(\mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i^2} \quad \text{for} \quad \mathbf{y} = \{y_i, i = 1,2,...n\}$$

is the root-mean-square value of a vector, then the simplest absolute measure of accuracy is $RMS(\Delta\mathbf{y})$.

Let $M(\mathbf{x})$ be the number of correct significant figures in the reference result corresponding to reference data $\mathbf{x}$. Then

$$N(\mathbf{x}) = \begin{cases} \min\left\{ M(\mathbf{x}), \log_{10}\left(1 + \dfrac{RMS(\mathbf{y}^{\text{ref}})}{RMS(\Delta\mathbf{y})}\right)\right\} & RMS(\Delta\mathbf{y}) > 0 \\ M(\mathbf{x}) & RMS(\Delta\mathbf{y}) = 0 \end{cases} \qquad (1)$$

gives a measure of the number of figures of agreement between the test results and the reference results. It may be useful to visualise

$$N(\mathbf{x};\mathbf{s}) = \begin{cases} \min\left\{ M(\mathbf{x};\mathbf{s}), \log_{10}\left(1 + \dfrac{|y^{\text{ref}}(\mathbf{s})|}{|\Delta y(\mathbf{s})|}\right)\right\} & |\Delta y(\mathbf{s})| > 0 \\ M(\mathbf{x};\mathbf{s}) & |\Delta y(\mathbf{s})| = 0 \end{cases}$$

where $\mathbf{s}$ denotes position in the domain and all results are now compared component by component. $N(\mathbf{x}; \mathbf{s})$ will give an idea of how the accuracy of the results varies with position.

A performance metric aims to quantify the number of figures of accuracy lost by the test software over and above what software based on an optimally stable algorithm would lose. It can be shown [1] that one suitable measure is given by

$$P(\mathbf{x}) = \log_{10}\left(1 + \frac{1}{\kappa(\mathbf{x})\eta}\frac{RMS(\Delta\mathbf{y})}{RMS(\mathbf{y}^{\text{ref}})}\right), \qquad (2)$$

where $\kappa(\mathbf{x})$ represents the relative condition of the problem and η is the computational precision. Clearly, this measure is not suitable for tests where $RMS(\mathbf{y}^{\text{ref}}) = 0$. The results of such tests should be analysed using an absolute metric instead. One expression for $\kappa(\mathbf{x})$ is

$$\kappa(\mathbf{x}) = \frac{\left\|\mathbf{y}^{\mathrm{ref}}(\mathbf{x}+\delta\mathbf{x})\right\|}{\left\|\mathbf{y}^{\mathrm{ref}}(\mathbf{x})\right\|} \Bigg/ \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|}, \tag{3}$$

where δ denotes a small change in the quantity it precedes. If the model is given by $\mathbf{y} = \mathbf{f}(\mathbf{x})$ an alternative expression for $\kappa(\mathbf{x})$ is

$$\kappa(\mathbf{x}) \leq \frac{\|J(\mathbf{x})\|\|\mathbf{x}\|}{\|\mathbf{y}\|}, \quad J(\mathbf{x}) = \left[\frac{\partial f_i}{\partial x_j}\right].$$

The calculation of $\kappa(\mathbf{x})$ can be difficult for continuous modelling software. It may be possible to calculate $\kappa(\mathbf{x})$ for problems with analytic solutions where $\mathbf{f}$ is known explicitly, but this may not be possible in all cases. If an analytic solution does not exist for the problem, it can be difficult to calculate the partial derivatives analytically, and perturbing and rerunning for every value of $\mathbf{x}$ of interest may be too computationally expensive. If the source code of the software is available, it may be possible to evaluate the partial derivatives using automatic differentiation techniques, but this will not always be possible for black box software.

If the aim of the test is to compare two pieces of software (package A and package B producing test results $\mathbf{y}_A$ and $\mathbf{y}_B$, say) then an alternative performance metric is given by

$$P_B^A(\mathbf{x}) = \log_{10}\left(\frac{\|\Delta\mathbf{y}_A\|}{\|\Delta\mathbf{y}_B\|}\right),$$

and if $\|\Delta\,\mathbf{y}\| = 0$ for either package, $\eta\|\mathbf{y}\|$ is used instead of $\|\Delta\,\mathbf{y}\|$. This quantity is only ever a relative measure of accuracy, but it can be used to rank different software packages. The value represents the number of figures in accuracy lost by using package A instead of package B. If $P > 0$ then B is more accurate than A, and if $P < 0$ then A is more accurate than B.

4.5 Other testing issues

A number of tests complementary to the main methodology were described as part of the report on testing of discrete modelling software [1]. Similar tests exist for continuous modelling software. Many of the methods described elsewhere [20] for model validation can be adapted for use as complementary tests

Many problems have properties such as symmetry or periodicity, and the results of such test problems can be checked to ensure that these properties are reproduced in the test results. For example, it is often possible to design a test problem so that the solution should have some kind of rotational or reflectional symmetry. If a test is defined such that the results should be symmetric but the symmetry is not enforced as part of the problem definition, a complementary test would be to check for symmetry in the results.

In some cases it may not be possible to define a scalable test or a small test for a pde, but a reference solution may be available for a particular choice of input data, problem domain, and boundary conditions. If this is the only type of reference result that is available, then it must be remembered when interpreting the test results that a single calculation often says very little about the overall performance of the software.

It was explained in section 4.2 that results from scalable tests could be used to explore

the behaviour of the software even in the absence of reference results, provided that the value of the function of the input data on which the test results depend is held constant. In some cases, this is true even for sets of tests that allow that function value to vary. For instance, consider the example given in section 4.2:

$$\rho c_p \frac{\partial T}{\partial t} = \lambda \nabla^2 T \, .$$

If the time t is rescaled by a factor α, where α is defined in section 4.2, so that $t^* = \alpha t$, then

$$\frac{\partial T}{\partial t^*} = \nabla^2 T \, ,$$

and so $T(\mathbf{x}, t^*)$ is independent of the input data. This result means that results from models using different values of α can be compared if the time scales of each are rescaled by an appropriate factor.

The methodology described in this report is designed to test the calculation steps carried out by continuous modelling software. There are various other tools commonly in use in continuous modelling that have been deliberately omitted from consideration, principally pre-processors and post-processors. Pre-processors help a user to construct a model and often include complex geometric generation and manipulation commands and automatic meshing tools. Post-processors are used to display results of calculations and usually include tools for generation of line plots, contour plots, and visual displays of displaced shapes for stress and vibrational analyses. These tools have not been considered initially because it was felt that the focus of the work should be on the calculation stages as it is often easier for users to tell if their pre- or post-processing software is producing errors. Future work could test these types of software tools, but identification of performance metrics could be difficult in some cases. Validation and testing of post-processing software is explored in the SS*f*M report "Visualisation, Uncertainties, and Continuous Modelling" [21].

5 Testing popular finite element packages

The methodology outlined in section 4 was applied to several commonly-used packages. The scalable tests were derived from analytic solutions to the pdes under certain simplifying assumptions, and the small-scale tests were taken from the basic set of NAFEMS benchmark tests. Whilst these tests only cover a small range of the capabilities of the packages, and the tests are not necessarily typical of the everyday usage of each package, they provide a useful comparison because the problem definitions are independent of the packages and do not play to the strengths of any one package.

5.1 Choice of packages

Three different software packages were tested: Abaqus version 6.3.1 [22], Ansys versions 5.4 and 7.1 [23], and Pafec version 8.7 [24]. These packages were the most popular choices of finite element software reported in the responses to the survey mentioned in section 1. Two different versions of Ansys were tested using the small-scale tests because both were readily available and comparison between the two could have shown how subsequent releases of a software package can improve and correct previous shortcomings. All packages are described more fully in the Appendix to the SS*f*M guide to use of finite element and finite difference software [25].

The original plan included testing of Femlab [26], but various features of the package made this impossible. Femlab has an automatic meshing tool that cannot be turned off and so it was not possible to test using the NAFEMS benchmarks or the thermal scalable test, all of which require specific meshes. Femlab does not allow point loads to be defined, so the scalable linear elastic test could not be used. Models of axisymmetric vibrational problems require an additional module that was not available under the licence held at NPL, so the scalable free vibration problem could not be used. This meant that none of the chosen tests could be used to test Femlab. This illustrates some of the difficulties in applying tests to software packages: packages do not always have capabilities appropriate for the tests.

5.2 Choice of problems

Three sets of tests were chosen, each covering a broad class of problem, namely static linear elastic stress analysis, thermal analysis, and vibrational analysis. Each set of tests consisted of one scalable test and several NAFEMS benchmarks. A brief outline of the tests will be given here, with a more detailed description of the scalable tests being given in the appendix. The NAFEMS tests will not be described in detail as they are only available to NAFEMS members.

Throughout the test definitions, units have not been supplied for the scalable tests. This is because one of the reasons for choosing such problems is that their results are independent of the units in which the problem is defined. Units are stated in the NAFEMS definitions of the small-scale tests, however, since they may have been derived from reference software rather than analytic solutions and so the rescaling necessary when making a change of units may affect the reference result. The units for the small-scale tests are not supplied in the index because they are irrelevant to the test results presented.

5.2.1 Small-scale tests

As was mentioned above, all the small-scale tests were taken from the set of NAFEMS benchmarks. The background to the benchmarks, published separately [27], describes the key attributes of the tests that make them suitably challenging problems for software packages. This section of the report gives broad descriptions of the tests and explains the motivation behind the tests.

1. Linear elastic test LE1: Elliptic membrane

The test models one quarter of a two-dimensional elliptic membrane with a central elliptic hole under a constant pressure loading on its outer edge. Four meshes are specified, two using triangular elements and two using quadrilateral elements. Two mesh densities are specified for each type of element, the finer mesh being a halving of the coarser mesh in each direction. The triangular elements are created by connecting one pair of diagonally opposite corners of the quadrilaterals, with the pair to be connected being specified. The required result is a stress value at a node, the reference value of which is derived from an analytic solution.

The background to this problem mentions three attributes of the test problem that cause difficulties with software packages. The first attribute is the problem geometry: some packages may not be able to describe the ellipse correctly. The second is that the analytic solution has a strong stress concentration at the point where the test result is calculated, and some element formulations may not pick this up correctly. The third is that the reference result is a nodal stress. Stresses are generally calculated at points within the elements and then extrapolated to the nodes. If this extrapolation algorithm has not been well chosen, the accuracy of the nodal stress results will decrease.

2. Linear elastic test LE3: Hemisphere - Point loads

The test models one quarter of a hemispherical shell under point loads. Two meshes are specified, both using quadrilateral elements, with the fine mesh having approximately half the element size of the coarse mesh. The required result is a displacement at a node, the reference value of which is derived from an analytic solution.

The background to the test explains that the test is formulated to test shell elements under bending loads rather than membrane stretching. The test also includes rigid body displacements which can cause numerical problems for some element formulations.

3. Linear elastic test LE7: Axisymmetric cylinder/sphere – Pressure

The test models a slice through a cylindrical shell topped by a hemispherical shell under a uniform internal pressure load. Two meshes are specified, the finer mesh having half the element size of the coarse mesh, both of which use axisymmetric shell elements. The required result is a stress value at a node, the reference value of which is derived from an analytic solution.

The background to the test explains that the main aspect under test in this problem is the application of the uniform pressure load. If the pressure load is not interpreted correctly by the software, the quality of the results suffers. It is important to ensure that the approximate loads applied at individual points are kinematically equivalent to the distributed pressure that the boundary condition of the continuous problem requires.

4. Thermal test T1: Membrane with hot-spot

The test models one quarter of a thin plate loaded by a strain caused by differential thermal expansion. The differential thermal expansion is caused by a hot spot at the centre of the plate. The thermal strain distribution is given over the entire plate, and the mesh is specified using quadrilateral elements. The required result is a stress value at a node, the reference value of which is derived from an analytic solution.

The background to the test describes how errors can be caused by the discontinuity in temperature at the edge of the hot spot, and by using the same order of elements for approximation of the displacements and the temperatures.

5. Thermal test T2: One dimensional heat transfer with radiation

The test is a steady-state thermal model of a uniform bar with boundary conditions of a fixed temperature at one end and radiative losses to a fixed ambient temperature at the other end. Two meshes are defined, one with one-dimensional beam elements and one with two-dimensional plane elements. It is assumed that there is no heat conduction in the directions perpendicular to the length of the bar. The required result is the temperature at one end of the bar, the reference value of which is obtained from an analytic solution.

The background to this test explains that the radiation boundary conditions are nonlinear, which means that the solution cannot be found by solving a simple linear system. The test checks the ability of the software to find a solution iteratively from an initial guess. It should be noted that this test is expected to be removed from subsequent issues of the NAFEMS benchmarks as it was found to be insufficiently challenging for most FE packages.

6. Thermal test T4: Two dimensional heat transfer with convection

The test is a steady-state thermal model of a uniform plate with zero internal heat generation. The boundary conditions are insulation along one edge, a fixed temperature along another edge, and convective heat loss (with a specified ambient temperature and surface convective heat transfer coefficient) on the other two edges. Two meshes are defined, one with quadrilateral elements and one with triangular elements created by joining a specified pair of diagonally opposite nodes within each element. The required result is the temperature at a node, the reference value of which is derived from an analytic solution.

The background to the test states that the solution to this problem contains steep temperature gradients and quasi-singular behaviour. The background also explains that software packages are not expected to be able to compute this value accurately due to the prescribed mesh being inadequate for description of quasi-singular behaviour, but that the problem is suitable for intercomparison of packages.

7. Vibration test FV4: Cantilever with off-centre point masses

The test models a simple beam, fixed at one end, attached to two point masses through rigid links. The beam undergoes free vibration. Two meshes are defined using different types of beam element. The required result is the determination of the first six natural frequencies of the system. The source of the reference solution is not stated in the test documentation.

The main attributes of the system that make it a difficult test problem are the lumped masses and rigid links and their effects. The offset masses cause the flexibility axis

of the system and the inertial axis to be non-coincident. The masses also cause coupling between the torsional and flexural behaviour. The system also has close eigenvalues which can be difficult to calculate.

8. Vibration test FV5: Deep simply-supported beam

The test models a beam of square cross-section, fixed at one end, undergoing free vibration. Two meshes are specified using different formulations of beam element. The required results are the first nine natural frequencies of the beam. The source of the reference solution is not stated in the test documentation.

The symmetry of this test problem means that the reference results include several pairs of repeated eigenvalues. Repeated eigenvalues can be difficult to calculate accurately. Additionally, some of the deformed shapes that correspond to the eigenvalues are purely extensional, and the eigenvalues corresponding to such shapes can be missed if an iterative method is used for eigenvalue determination.

9. Vibration test FV52: Simply-supported "solid" square plate

The test models a square plate in three dimensions. The plate is fixed along its lower outer edges and undergoes free vibration. Two meshes are specified, both using three-dimensional brick elements. One mesh has 16 elements using a second-order approximation and the other mesh uses 192 elements and a first-order approximation. The required results are the first ten natural frequencies of the system. The source of the reference solution is not stated in the test documentation.

The main attribute of the problem that makes it a challenging test is the existence of rigid body modes. Rigid body modes correspond to zero eigenvalues, and their existence can cause the calculation of higher frequencies to be inaccurate.

5.2.2 Scalable tests

Three scalable tests were developed from analytic solutions to problems. The full derivation of the analytic solutions is not given here. The full list of input data that were used in the tests is given in the appendix. A mesh definition was supplied as part of the test definition of each of the scalable tests. Each of these meshes has been shown to produce converged results to the quoted degree of accuracy. As was stated in section 5.2, no units are specified for these problems.

1. Three-dimensional elastic bending beam

The first scalable problem is derived from an analytic solution to a elastic bending beam problem. The test problem for the software is illustrated in figure 3. The beam is loaded with a force P at one end and is fixed at the other end so as not to move in any direction in the plane $x = 0$. The beam is 0.1 thick in the y and z directions. The mesh consists of three-dimensional second-order brick elements measuring 0.025 in each direction, i.e. a 40 by 4 by 4 element mesh.

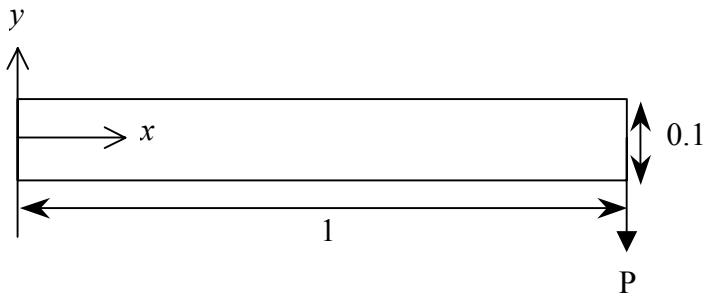

Figure 3: Illustration of the elastic beam scalable problem.

The generation of an analytic solution requires a number of assumptions about the stress distribution within the beam that cannot generally be imposed in the majority of finite element codes. The assumptions are:

$$\sigma_{yy} = \sigma_{zz} = \sigma_{yz} = 0$$
$$\sigma_{xx} = \frac{-P(1-x)y}{I} \qquad (4)$$
$$I = 0.1^4/12,$$

where the origin of the coordinate system is taken to lie at the centre of the beam face that is held fixed. Additionally, the boundary conditions imposed to achieve the analytic solution are less strict than those on the finite element model. As was stated above, the finite element model fixed the whole of the plane $x = 0$, but the analytic solution only requires that

$$u = v = w = \frac{\partial u}{\partial x} = \frac{\partial v}{\partial x} = 0 \quad \text{on } x = y = z = 0 \qquad (5)$$

where u, v, and w are the displacements in the x, y, and z directions respectively. The assumptions (4) cannot be imposed directly on a model using three-dimensional elements, although use of beam elements will apply them directly. The conditions (5) are not sufficient to produce a solution in a three-dimensional model, and so the stricter conditions mentioned above have been applied instead.

It is likely that these differences in boundary conditions will affect the results in the region of $x = 0$. It is also likely that the use of a point load P will affect the results in the region of $x = 1$. For these reasons, the results of interest are only examined for $0.05 \le x \le 0.95$ since it is expected that the effects should have died away in that region. The results were examined as a function of x to check that this was the case.

The results of interest for this test were the axial stress σ_{xx} and the displacements u and v along the line $y = -0.05$, $z = 0$, $0.05 \le x \le 0.95$, and the displacement v and the axial stress σ_{xx} along the line $y = z = 0$, $0.05 \le x \le 0.95$. These quantities can be calculated from the formulae

$$\sigma_{xx} = \frac{P(1-x)y}{I}$$

$$u = \frac{-Pxy(2-x)}{2EI} + \frac{P}{EI}\left[(1+\nu)\left(0.05^2 y - \frac{y^3}{3}\right) + \nu\frac{y^3}{6}\right] \qquad (6)$$

$$v = \frac{\nu P(1-x)y^2}{2EI} + \frac{Px^2(3-x)}{6EI},$$

where I is given in (4) above, E is the Young's modulus of the beam material and ν is Poisson's ratio. It is clear from (6) that the displacement results will not change for a fixed value of P/E, and so the load and the Young's modulus were taken to be the input data. The stress results were divided by P before comparison. A condition number can be calculated for these results, but it is constant for constant values of P/E and so will not be used here.

2. Axisymmetric transient thermal test

The second scalable test was an axisymmetric transient thermal test. The full geometry and the section used in the model are shown in figure 4.

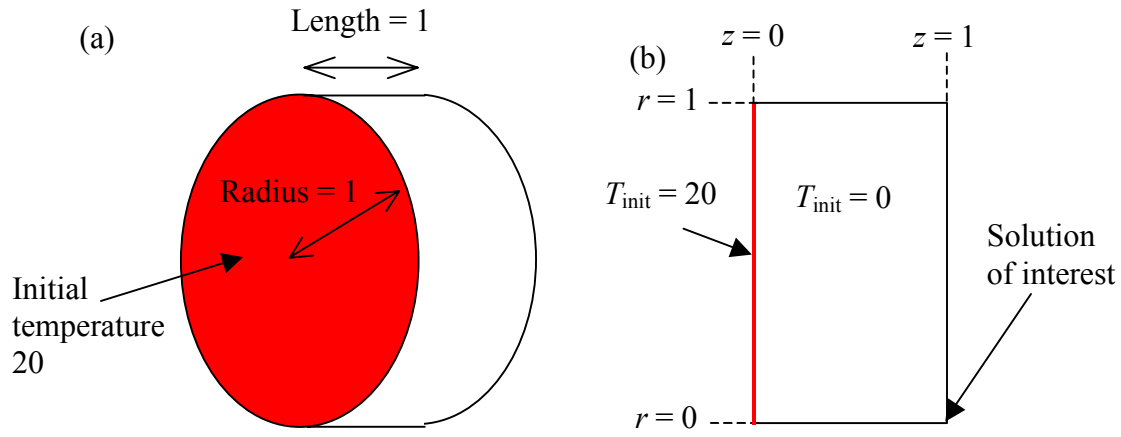

Figure 4: Full geometry (a) and modelled cross-section (b) of the scalable thermal test.

The test simulates a cylindrical uniform sample of material with radius 1 and width 1. The sample is perfectly insulated on all sides so that no heat is lost from the sample and the initial conditions are that the nodes on the face $z = 0$ are at a temperature of 20 and the rest of the sample is at 0. The mesh consists of 10 axisymmetric second-order elements in each direction so that each element measures 0.1 units by 0.1 units. The solution of interest is the temperature at the centre of the rear face, i.e. $r = 0$, $z = 1$.

The analytic solution to this problem is given by

$$T(t) = \frac{1}{3}\left(1 + 2\sum_{m=1}^{\infty}\{-1\}^m \exp\left[-\alpha(m\pi)^2 t\right]\right) \qquad (7)$$

where t is time, $\alpha = \lambda/(\rho c_p)$ is the thermal diffusivity, and the input data are λ the thermal conductivity, ρ the density, and c_p the specific heat capacity. Hence if the input data are varied individually but α is held fixed, the solution will not change.

3. Axisymmetric free vibration

The final test is an axisymmetric free vibration test. This test simulates the vibration of

a disc of unit radius and thickness 0.02 units whose outer edge is clamped. The mesh consists of 100 second-order axisymmetric elements in the radial direction and three second-order axisymmetric elements in the through-thickness direction. The results of interest were the first two natural frequencies. The analytical solution for the frequencies of the vibration are given by [28]

$$f_n = \frac{\pi h \beta_n^2}{4a^2}\sqrt{\frac{E}{3\rho\left(1-\nu^2\right)}}, \tag{8}$$

where h is the thickness of the disc, a is the radius, ρ is the density, E is the Young's modulus, ν is the Poisson's ratio, and the β_n are constants. The values of β_n corresponding to the first two frequencies are 1.015 and 2.007.

It is clear from this expression that the results will remain constant if ρ, E, and ν are varied, but $E/(\rho\{1-\nu^2\})$ is held fixed. Hence ρ, E, and ν were taken as the input data. It is possible to calculate a condition number for these results. However, the quantities ρ and E vary over a much wider range than the quantity ν (which only ever lies between 0 and 0.5 for almost all materials) which causes the condition number to be influenced much less by ν than it is by ρ and E. A suitable scaling to solve this problem has not been found, and so the condition number has not been considered when investigating the test results.

In two of the three scalable tests, it is expected that the quantities that have been chosen to be held fixed will be formed during the assembly of the system of equations. For example, consider a transient thermal problem. Transient heat flow is described by

$$\rho c_p \frac{\partial T}{\partial t} = \nabla.\left(\lambda \nabla T\right),$$

with appropriate boundary conditions. If U^n represents a discrete approximation to the temperature at time $n\Delta t$, where Δt is the time step, and the spatial derivatives are approximated using some function G, and all material properties are regarded as being constant then a simple discrete approximation can be written

$$U^{n+1} = U^n + \frac{\lambda}{\rho c_p} G\left(U^n\right).$$

This expression means that, provided that there is no dependence of the boundary conditions on the input data, the approximate solution is dependent on $\lambda/(\rho c_p)$ but not λ, ρ, and c_p independently. Similarly, it is expected that the quantity P/E will occur during the solution procedure for the beam problem.

5.3 Implementation issues

This section describes some of the problems experienced during implementation of the small-scale tests and scalable tests. The majority of the tests were implemented without any problems, but some minor problems were encountered with a few of the NAFEMS benchmarks.

In one case, the test could not be implemented in any meaningful way. The test in question was LE3, the hemispherical shell under point loads. This test requires the use of non-planar shell elements to define its geometry correctly. In Abaqus this can be done by defining the nodal geometry as a set of points in space and defining a normal vector to the surface at each of the nodes. In Pafec, however, all shell elements had to

be defined as flat and so the hemispherical geometry could not be described sufficiently accurately to give meaningful results.

Two issues were encountered when implementing the tests in Ansys. The first problem was a minor difficulty in implementing radiation boundary conditions in test T2. In two dimensions, it was necessary to define a separate set of elements to model the surface radiative heat losses, rather than defining the losses as nodal conditions or as conditions on the surfaces of the two-dimensional elements. In addition, it seemed to be impossible to run the simulation using Kelvin as the temperature unit and so the test had to be run using the Celsius scale and corrected afterwards.

The second issue occurred during the implementation of test FV4 in Ansys. Ansys does not seem to have a "rigid link" type element as is required to link the lumped masses to the main beam. Instead, beam elements with a large thickness (2.5 m), a high Young's modulus (100 times that of the main beam), and a low density (10^{-10} kg m^{-3}) were used. These elements would be near-rigid and near-weightless and so should not affect the results adversely.

In addition to these problems, one set of tests was initially run with the wrong mesh definition: the Ansys implementations of test LE1 using triangular elements were run with the wrong orientation for the triangles. The results of these runs will be discussed in section 5.4, since they illustrate the consequences of a small error in implementation.

The one-dimensional version of test T2 also created problems in Pafec. It was not possible to implement radiative heat loss directly. Instead, an element was defined whose thermal conductivity was temperature-dependent. Some problems were experienced with the use of this element until it was realised that the average element temperature was used to calculate the conductivity, rather than the nodal temperature.

None of the scalable tests caused any problems with implementation.

5.4 Test results

The full list of reference results and test results for the small-scale tests are given in the appendix. The scaleable test results are too large to list in full. This section summarises the results graphically and discusses the points of interest raised by the tests.

5.4.1 Small-scale tests

Throughout the following, the results plotted are values of N, an estimate of the number of figures of agreement between the test result and the reference result. This enables comparisons between different tests to be made as well as comparisons between different packages on the same test. Note that the larger the value of N is, the better the agreement is between the test result and the reference result.

Figure 5 shows the results of test LE1, and illustrates a number of interesting points. The first point is that it is likely that the triangular element used in this test has been reformulated between Ansys versions 5.4 and 7.1. This is the only test where differences between the two versions of Ansys occurred, which means that the differences here are almost certainly due to the altered element formulation rather than any improvements in the system solver routines. Another interesting point is the difference between the results using the correct and incorrect meshes in Ansys. Using the incorrect meshes loses at least one figure of agreement. This illustrates the importance of correct mesh definition as part of test problem definition, and of the importance of taking care when implementing a test. It is also interesting to note that in

several cases the results obtained using the quadrilateral elements are not as good as those obtained using a fine triangular mesh.

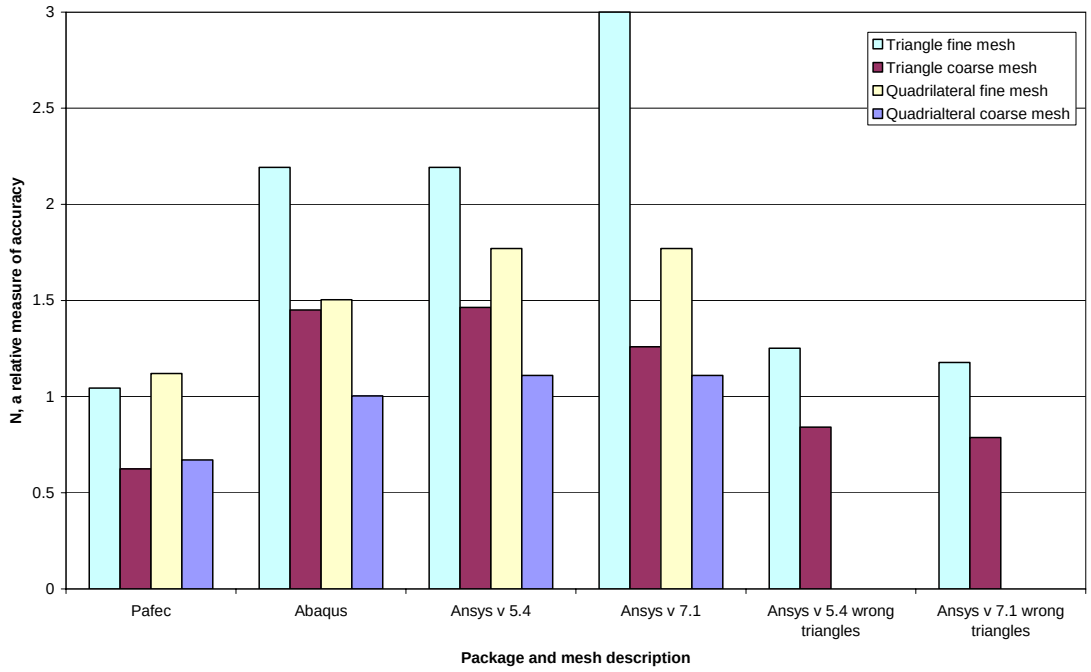

Figure 5: Number of figures of agreement between test results and reference results for test LE1 for the different packages and meshes.

For the remainder of the tests, only one set of results will be showed for Ansys since the results from versions 5.4 and 7.1 were identical. Figure 6 shows the results of the other linear elastic tests. In general, Abaqus performed better than the other two packages. No results are shown for Pafec for test LE3 with the fine mesh due to the problem described in section 5.3.

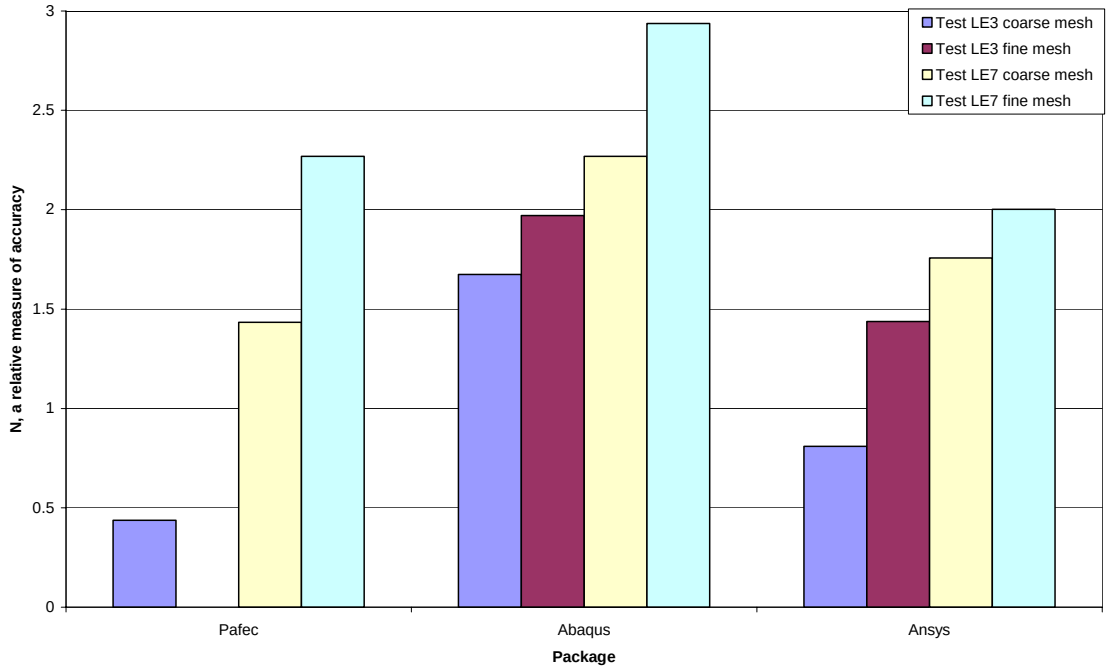

Figure 6: Number of figures of agreement between test and reference results for tests LE3 and LE7.

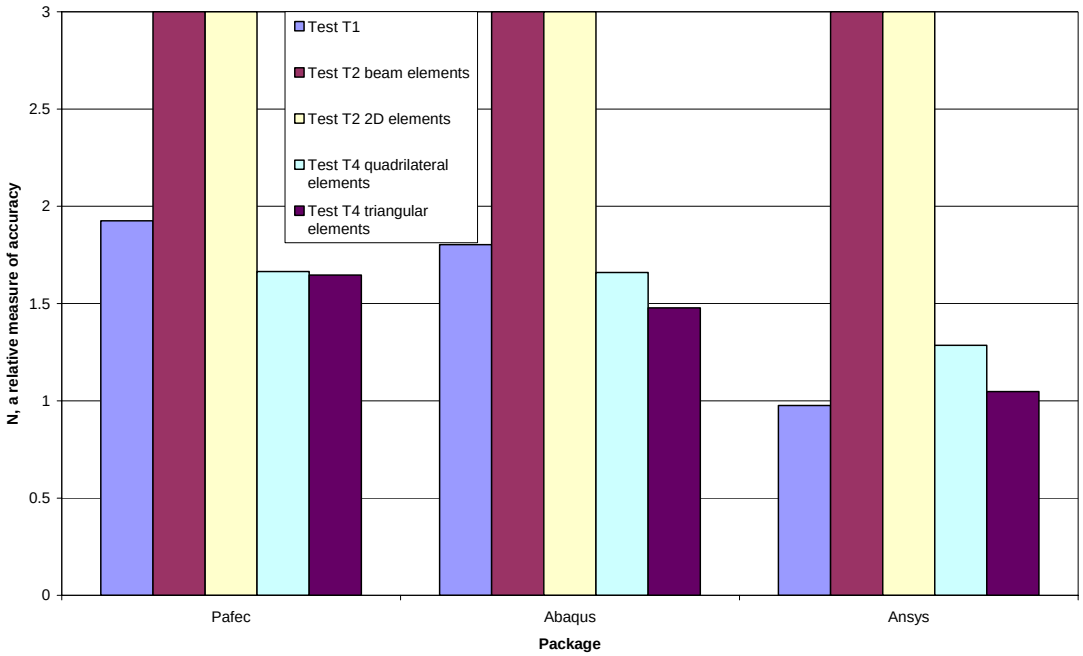

Figure 7: Number of figures of agreement between test and reference results for the small-scale thermal tests.

Figure 7 shows the results of the thermal tests. The maximum possible value of N for each of the thermal tests is 3. All three packages produced exact agreement with the reference results for the radiation test, as was expected. Initially an error occurred in the Pafec solution with one-dimensional elements, caused by the problem described in section 5.3, but this was eliminated. Ansys produced slightly less accurate results than the other two packages for tests T1 and T4.

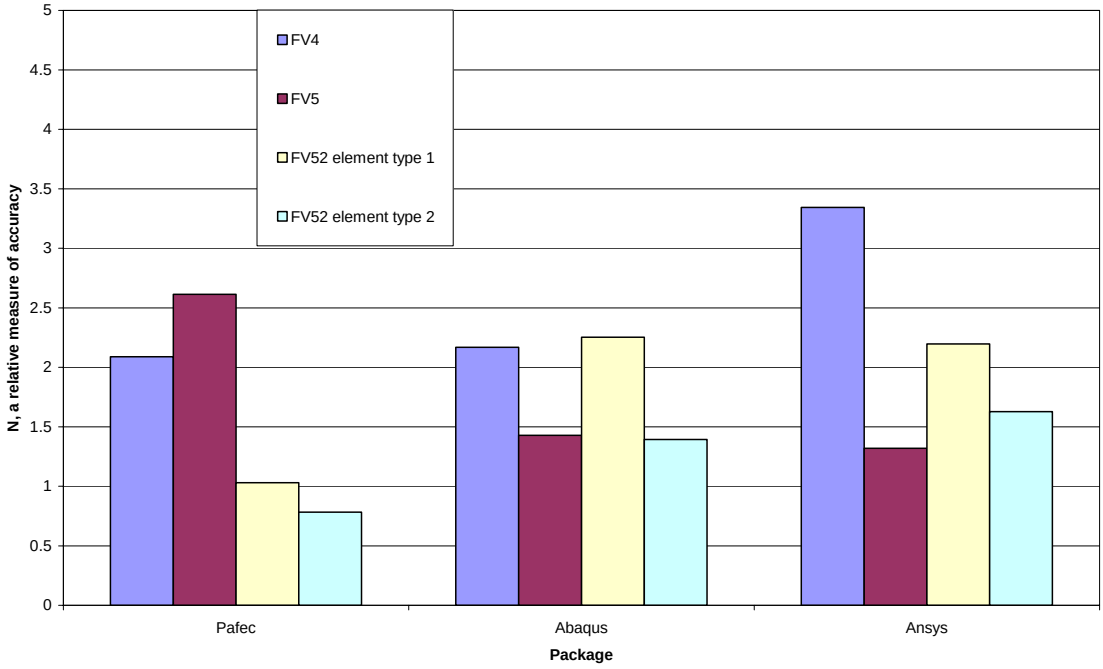

Figure 8: Number of figures of agreement between test results and reference results for the small-scale vibration tests, treating each frequency as a component of a vector.

Figure 8 shows the results of the vibration tests. These values of N are calculated by

regarding the frequencies to be determined in each test as components of a vector and calculating *N* that way. For test FV4, the maximum achievable value of *N* is 4, and for tests FV5 and FV52 the maximum value is 5. Figure 8 shows that the agreement is not perfect in any of the tests. The frequency-by-frequency variations of each test are shown in figures 9 to 12.

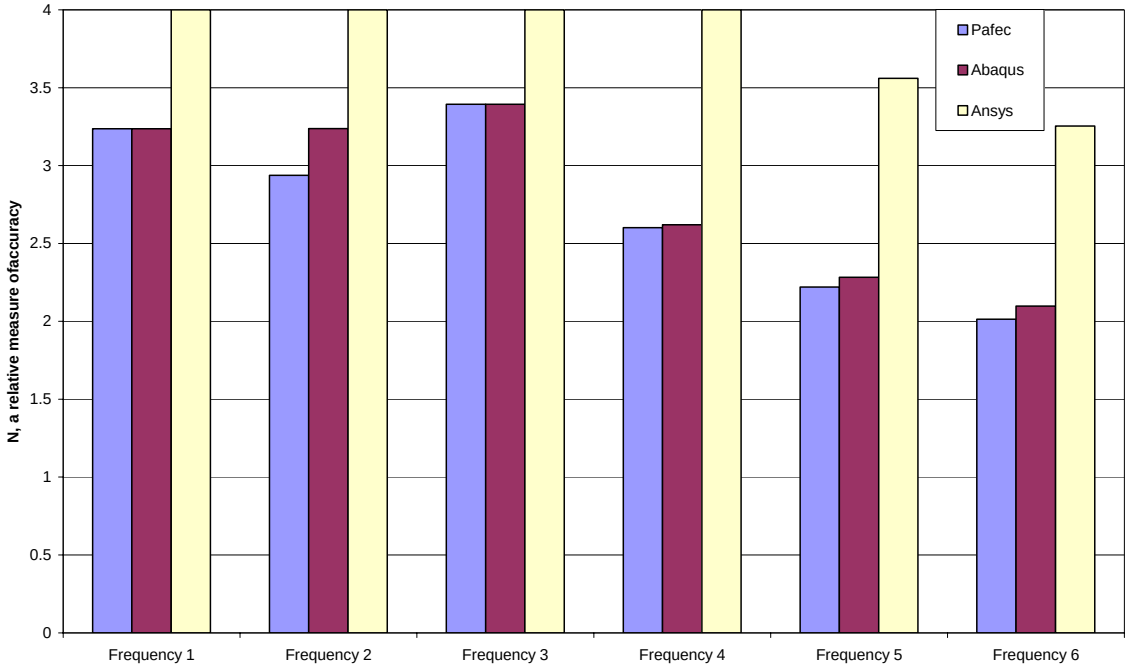

Figure 9: Number of figures of agreement between test results and reference results for test FV4, treating each frequency separately.

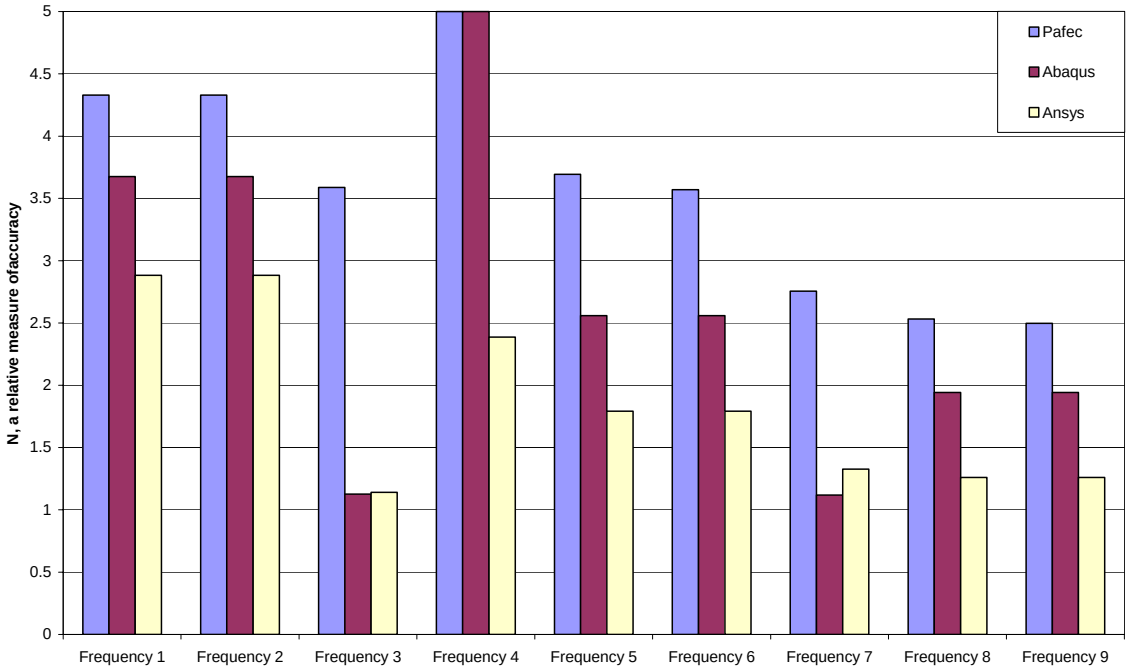

Figure 10: Number of figures of agreement between test results and reference results for test FV5, treating each frequency separately.

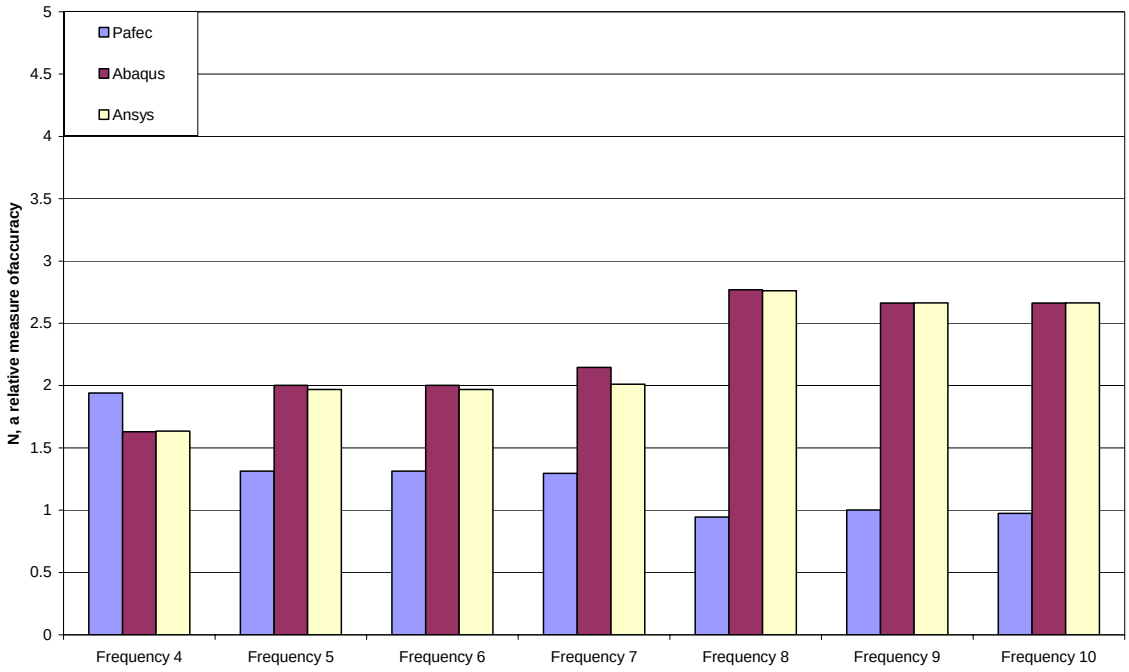

Figure 11: Number of figures of agreement between test results and reference results for test FV52 using a coarse mesh of high-order elements, treating each frequency separately.

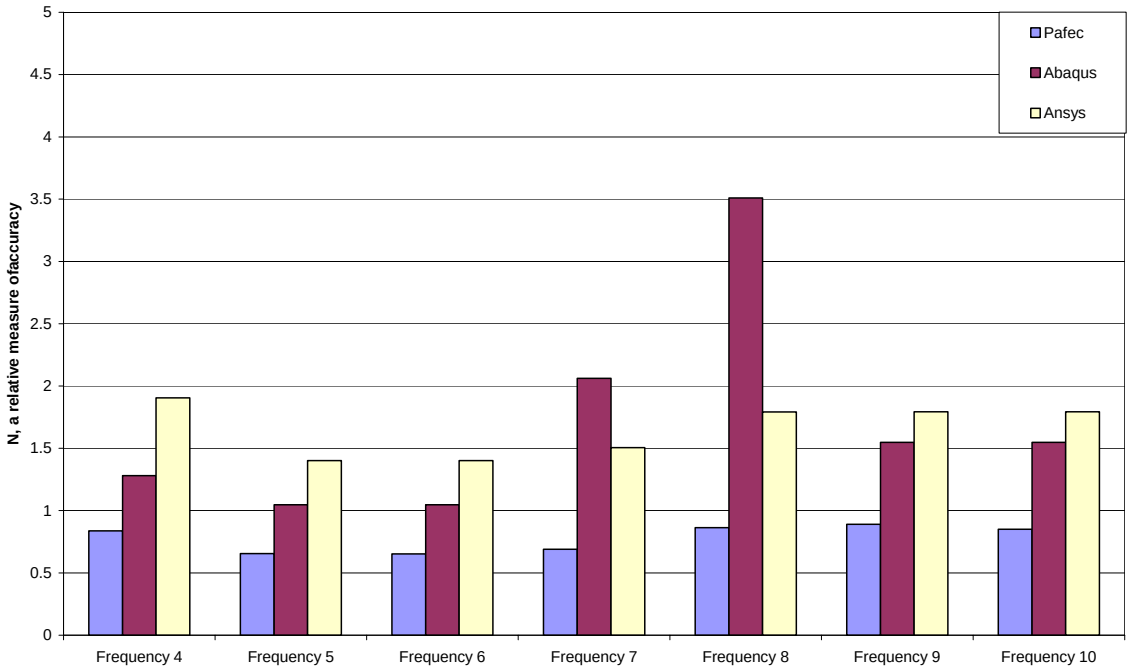

Figure 12: Number of figures of agreement between test results and reference results for test FV52 using a fine mesh of low-order elements, treating each frequency separately.

Figure 9 shows a decrease in the accuracy of the results with an increase in frequency number for all three packages. This is generally to be expected. The natural frequencies of a structure are calculated as the eigenvalues of the matrix produced by considering the stiffness and loading of its locally-approximated equivalent. These eigenvalues are usually calculated using an iterative procedure that calculates a single eigenvalue, updates the matrix, and proceeds to the next eigenvalue. As the procedure progresses,

the updated matrix can become less well-conditioned for some choices of algorithm and so the calculations can become less accurate.

The Ansys results show good agreement throughout, and so the alternative to rigid links as explained in section 5.3 has been successful. It is interesting to note that the solution to this approximation of the original test problem is closer to the reference result than the implementations of the "true" test problem. All of the packages distinguished between frequencies 1 (1.723 Hz) and 2 (1.726 Hz) successfully, which was one of the challenges of the test.

Figure 10 shows an interesting variation in accuracy with frequency. Abaqus and Ansys are inaccurate for frequencies 3 and 7, but Pafec is accurate for these two frequencies. One possible reason for this is suggested by examining the mode shapes: frequencies 3 and 7 correspond to torsional mode shapes where the beam is not displaced horizontally or vertically, but does rotate about its own axis. It would seem that Abaqus and Ansys have had trouble calculating the frequencies corresponding to these modes. Overall, figure 10 indicates that Ansys has not performed particularly well for this test. It is not clear why this might be the case.

Figures 11 and 12 show Pafec as being less accurate than Abaqus and Ansys for test FV52. All three packages successfully identified the rigid body modes (and the corresponding zero eigenvalues), but it is possible that determination of these zero values has affected the accuracy of the subsequent calculations. The rigid body modes correspond to the first three frequencies and are not shown in figures 11 and 12.

One surprising trend shown in figure 11 is the increase in accuracy of the Ansys and Abaqus results with increasing frequency. It is not clear why this is happening, although it is only a slight tendency and may not be present if higher frequencies were calculated.

Comparison of figures 11 and 12 indicates that the less dense mesh with the higher order approximation generally produces better results than the lower-order approximation with a denser mesh, although neither mesh produces perfect results. It is not entirely clear which aspect of this problem is so challenging.

5.4.2 Scalable tests

The beam test and the thermal test both produce vector results, either because the results vary with position (beam test) or they vary with time (thermal test). These test results will largely be examined by considering the RMS difference between the reference result and the test result, although some plots of difference versus position will be shown. The frequency tests produce two results and so the relative measure of accuracy N will be used for comparison for each.

Throughout the following, "Job number", as used on the horizontal axis of many of the plots, identifies the values of the input data. The tables linking job number with the input data used are listed in full in the Appendix.

Almost without exception, each package produced the same RMS difference between the reference result and test result for all the beam tests, independent of the input data. For example, the difference between the analytic expression for y displacement at the centre of the bar and the values calculated by Pafec had an RMS value of 3.39×10^{-6} for all the values of Young's modulus and load that were used. The RMS differences for the results and the corresponding values of N, calculated from (3) where the reference results are non-zero, are given in table 1. No value of N is given for the axial stress at the centre of the beam because the reference results are uniformly zero.

Package	Test result	Location	RMS difference	*N*
Abaqus	Axial stress	Centre of beam	3.64	-
Ansys	Axial stress	Centre of beam	0.00	-
Abaqus	Axial stress	Base of beam	10.2	2.5
Ansys	Axial stress	Base of beam	8.60	2.6
Abaqus	*y* displacement	Centre of beam	6.66×10^{-6}	3.1
Ansys	*y* displacement	Centre of beam	3.35×10^{-6}	3.4
Pafec	*y* displacement	Centre of beam	3.35×10^{-6}	3.4
Abaqus	*y* displacement	Base of beam	2.39×10^{-5}	2.6
Ansys	*y* displacement	Base of beam	2.45×10^{-5}	2.6
Pafec	*y* displacement	Base of beam	2.26×10^{-5}	2.6
Abaqus	*x* displacement	Base of beam	9.4×10^{-5}	1.1
Ansys	*x* displacement	Base of beam	9.4×10^{-8}	4.1
Pafec	*x* displacement	Base of beam	4.5×10^{-8}	4.4

Table 1: RMS differences between the test results and reference results, and the corresponding *N* values where the reference results are non-zero. Directions are as indicated in figure 3.

It is clear from the results in Table 1 that the calculated horizontal displacements for Abaqus are unexpectedly poor. No reason has been identified for this as yet. In general the displacements are in better agreement than the stresses.

The only sets of results not to appear in table 1 are the stresses calculated by Pafec. The RMS differences for these results varied from test to test. A plot of the variation in RMS difference with test number is shown in figure 13. The variation does not appear to have a pattern to it, and it is not clear what has caused the variation.

It is interesting to plot the difference between the reference results and test results as a function of position. Figure 14 shows a typical variation for stress results. The results shown are the difference between the reference results and test results for the axial stress calculated on the base of the beam by Abaqus, but the other packages showed similar behaviour. Note that the vertical scale is logarithmic. It should also be noted that in figures 13 and 14, in contrast to figures 5 to 12, the smaller the value the better because RMS differences are being plotted rather than number of figures of agreement.

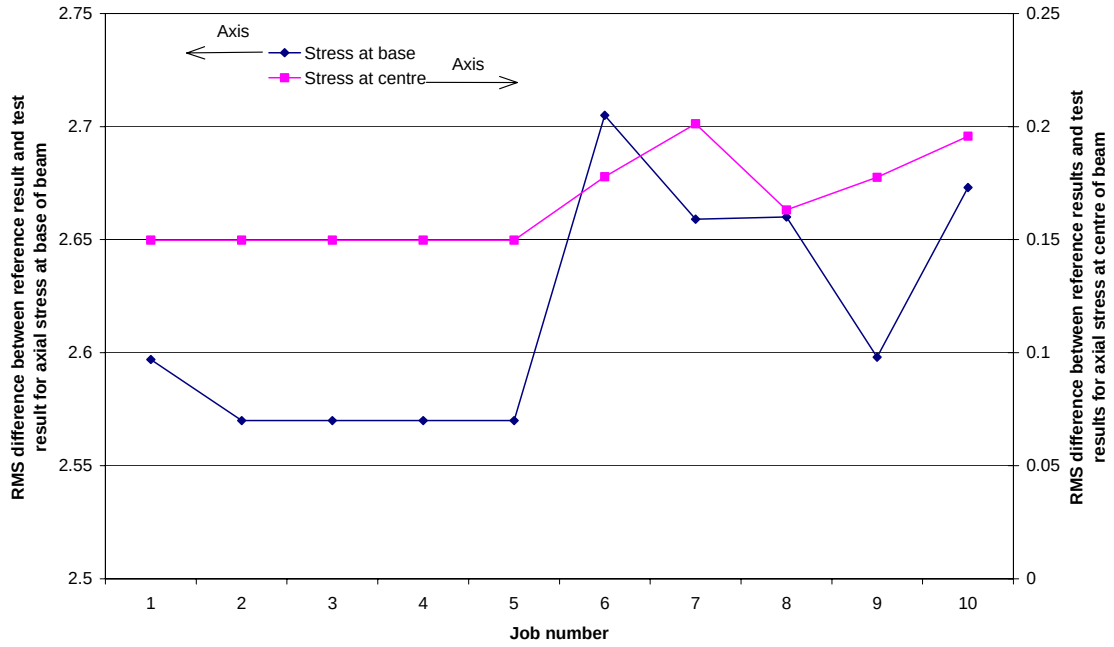

Figure 13: RMS differences between test result stresses produced by Pafec and reference results. See table A1 in the appendix for input data corresponding to the job number.

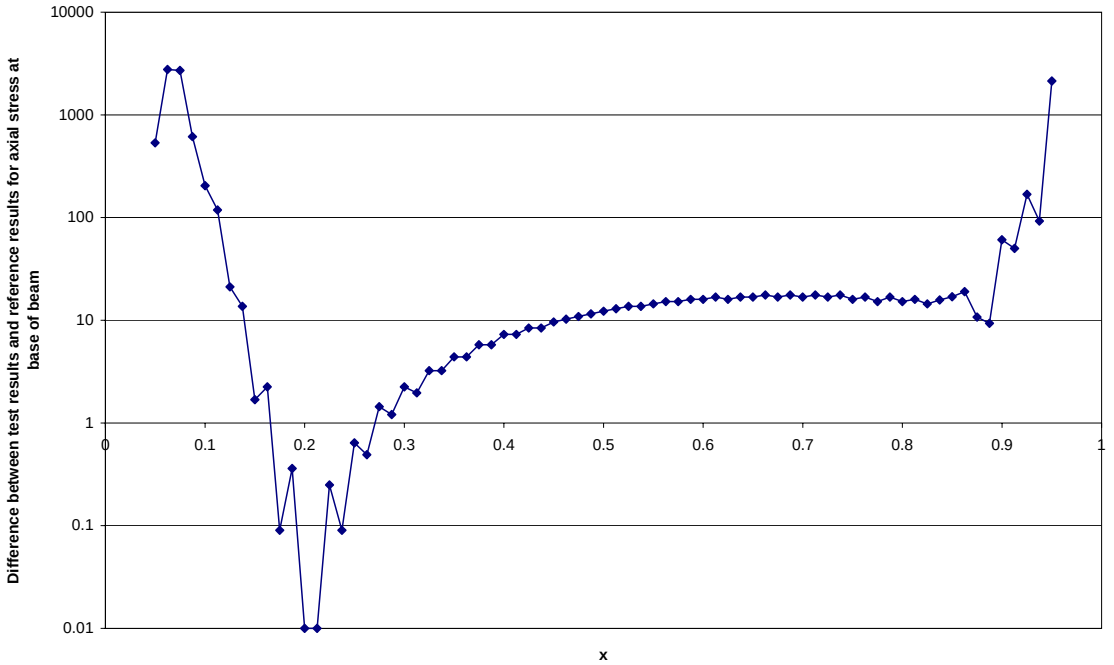

Figure 14: Difference between the reference result for axial stress and the value calculated by Abaqus as a function of x.

As was mentioned in section 5.2.2, the boundary conditions applied to the FE model are not the same as those used to derive the analytic solution, and this is what has caused the large differences for small values of x. In addition, the use of a point load leads to a stress concentration at $x = 1$, and this has caused large differences between the reference results and the test results in the region of $x = 0.95$. These differences would become worse as x tended towards 0 or 1, which illustrates the usefulness of choosing the reference results carefully.

For the thermal tests, as with the beam test, for each package the RMS difference between reference results and test results hardly varied as the input data changed. For Abaqus the difference was constant for all values of the input data, at 4.54×10^{-4}. For Ansys, the RMS difference was 1.89×10^{-3} for all the tests. These values equate to N values, calculated as in (3), of about 2.8 for Abaqus and about 2.2 for Ansys.

The Pafec results, although close to constant, showed some variation with input data. Their variation with input data is shown in figure 15. There does not seem to be a pattern to this variation. The variation does not seem to relate to the extreme values in any of the parameters, nor to extreme values of the ratio or product of any pair of the parameters. It may be due to accumulated rounding error, since Pafec uses an explicit time-stepping algorithm to solve thermal problems rather than an iterative algorithm.

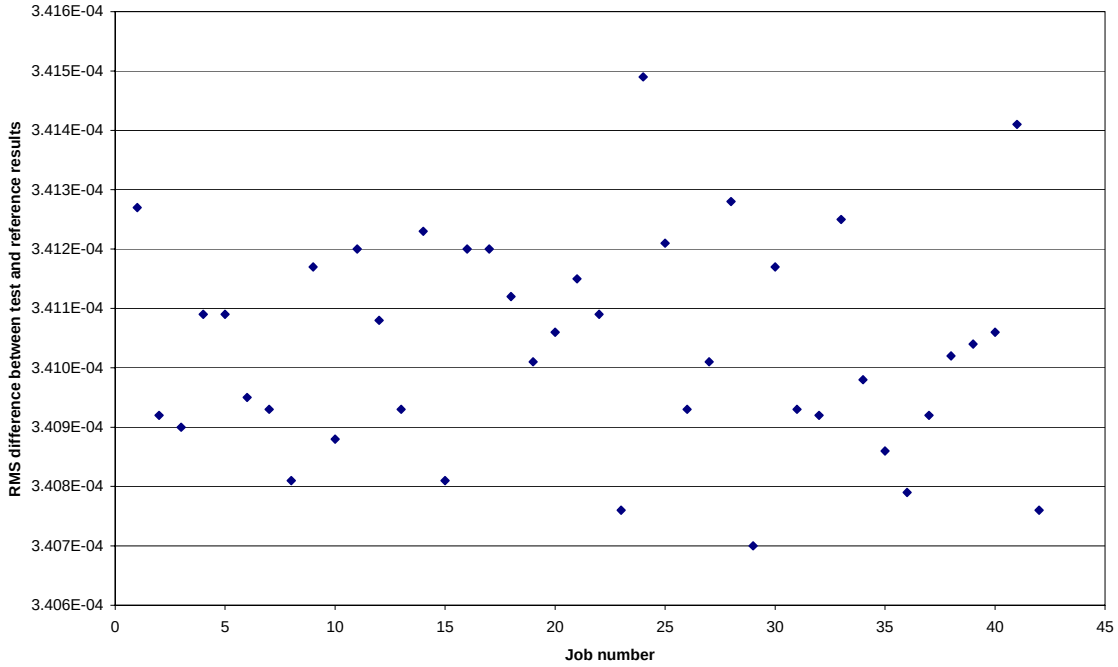

Figure 15: RMS difference between Pafec thermal test results and reference test results. See table A2 in the appendix for input data corresponding to the job number.

The results of the scalable vibration tests are shown in figures 16 and 17. The two figures show a relative measure of accuracy for each combination of parameters. Figure 16 shows the results for the first frequency and figure 17 shows the results for the second frequency. It was assumed that the reference results were known to four figures of accuracy since the constants β are given to four figures. In these figures, the number of figures of agreement between the test result and the reference result is plotted so the larger the value, the better the agreement between the two.

The three packages produced almost identical results. Each package produced a single value for the tests where ν was held constant whilst E and ρ were varied. Varying ν, E and ρ simultaneously led to more variation in the test results, but there was still little change in results. Typically the test results and reference results agreed to two or three figures for both frequencies. Since the three packages produced such similar results, it is possible that the differences between test result and reference result are caused by assumptions made during the generation of the reference result not being implemented during the implementation of the test.

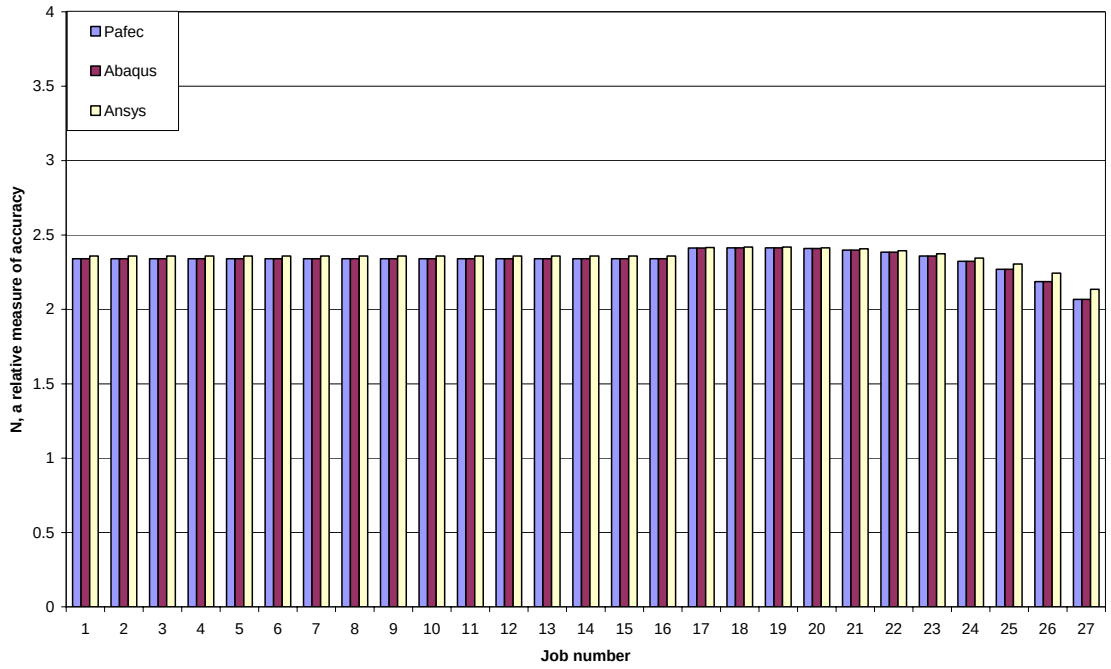

Figure 16: Relative accuracy results for the first frequency of the scalable frequency tests. See table A3 in the appendix for input data corresponding to the job number.

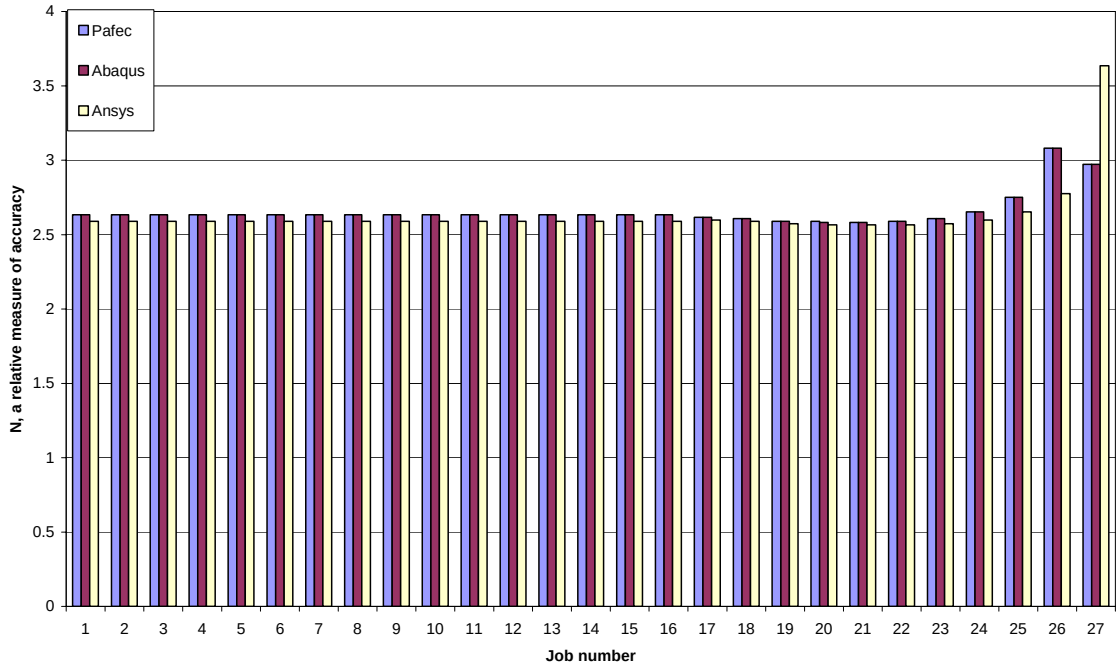

Figure 17: Relative accuracy results for the second frequency of the scalable frequency tests. See table A3 in the appendix for input data corresponding to the job number.

5.5 Summary

Several interesting points were highlighted during the testing of three popular FE packages. These points included:

- Two versions of the same software produced different results for one test. This is almost certainly due to a reformulation of the element used in the test, and illustrates the importance of retesting software whenever a new version is released.
- Two tests were run with a slightly incorrect mesh definition, resulting in inaccurate results. This illustrates the importance of defining a mesh as part of the test definition wherever possible, and checking that the mesh supplied has been implemented correctly.
- Several difficulties were encountered during the implementation of the tests: not all the packages had suitable elements for all the tests, and in some cases the definition of appropriate boundary condition behaviour was difficult. Suitable alternative elements and conditions were identified, but this shows the importance of defining problems clearly. Since the aim of the test formulation was clear in all cases, alternative methods of producing the same results could be found.
- Some of the small tests identified specific weaknesses of the packages. This is a useful feature of well-designed small scale tests such as the NAFEMS benchmarks.
- The scalable tests identified a seemingly random error behaviour for one of the packages. Where two of the packages had a fixed RMS difference between reference results and test results for a given problem, independent of the input data, the third package's RMS difference varied with no clear pattern. The results were no less accurate for the third package, but the variation was unexpected.

6 Conclusions

A methodology has been developed for testing continuous modelling software. The methodology is partially based on the methodology for testing discrete modelling software developed as part of the first SS*f*M (1998-2001) programme [1], but the new methodology takes several key features of most continuous modelling software into account. The main key feature considered is that most continuous modelling techniques construct an approximate solution to the continuous model by splitting the solution domain into small units, approximating the solution within each of the small units, and solving the system of equations resulting from the union of all of the local approximations.

The new methodology is a two level process. The first level tests the formulation of the local approximations to the larger system, by using tests that can be solved accurately using only a few units. It may not be possible to construct small-scale tests for all continuous modelling methods. In particular, it may not be possible to apply this type of test to boundary element software.

The second level tests the solution procedures for the large system of equations. Wherever possible, this level is based on the idea of reference data generators. If a problem with an analytic solution can be identified such that the solution will not vary if some function of the input data is held constant, then the individual input data can be varied, provided that the function remains constant. This idea can be used to develop problems of varying difficulty that should produce the same results. Such problems are called scalable tests.

As well as the methodology outlined above, other tests can be used to test continuous modelling software. Reference solutions for unique problems may be available. It may be possible to generate reference results using an independent calculation method. It is possible to use some model validation methods [20] for software testing.

The methodology has been applied to several popular finite element packages. In general, all of the packages produced good agreement with the scalable test reference results. Some of the results of the small-scale tests (which were generally more challenging) were less accurate. Several interesting points were raised during the testing, some of which will be considered for further work in the numerical software testing project of the third SS*f*M (2004-2007) programme.

7 References

[1] H Cook, M Cox, M P Dainton and P M Harris. A Methodology for Testing Spreadsheets and Other Packages Used in Metrology. NPL Report CISE 25/99, September 1999.

Available from http://www.npl.co.uk/ssfm/download/documents/cise25_99.pdf

[2] M G Cox, M P Dainton, and P M Harris. Testing Spreadsheets and Other Packages Used in Metrology: Testing Functions for Linear Regression. NPL Report CMSC 08/00, October 2000.

Available from http://www.npl.co.uk/ssfm/download/documents/cmsc08_00.pdf

[3] M G Cox, M P Dainton, and P M Harris. Testing Spreadsheets and Other Packages Used in Metrology: Testing Functions for the Calculation of the Standard Deviation. NPL Report CMSC 07/00, October 2000.

Available from http://www.npl.co.uk/ssfm/download/documents/cmsc07_00.pdf

[4] J Barrett and M P Dainton. Testing Spreadsheets and Other Packages Used in Metrology: Testing the Intrinsic Functions of S-Plus. NPL Report CMSC 06/00, September 2000.

Available from http://www.npl.co.uk/ssfm/download/documents/cmsc06_00.pdf

[5] J Barrett and M P Dainton. Testing Spreadsheets and Other Packages Used in Metrology: Testing the Intrinsic Functions of Mathcad. NPL Report CMSC 05/00, September 2000.

Available from http://www.npl.co.uk/ssfm/download/documents/cmsc05_00.pdf

[6] H Cook, M Cox, M P Dainton and P M Harris. Testing Spreadsheets and Other Packages Used in Metrology - Testing the Intrinsic Functions of Excel. NPL Report CISE 27/99, September 1999.

Available from http://www.npl.co.uk/ssfm/download/documents/cise27_99.pdf

[7] H Cook, M Cox, M P Dainton and P M Harris. Testing Spreadsheets and Other Packages Used in Metrology, A Case Study. NPL Report CISE 26/99, September 1999.

Available from http://www.npl.co.uk/ssfm/download/documents/cise26_99.pdf

[8] T J Esward, K Lees, D Sayers, and L Wright. Testing Continuous Modelling Software: Three Case Studies. NPL Report CMSC 42/04. March 2004.

[9] L Hatton. The T Experiments: Errors in Scientific Software. IEE Computational Science & Engineering, v.4 no.2, 27-38, January 1997.

Available from http://www.leshatton.org/IEEE_CSE_297.html

[10] W H Enright and J D Pryce. Two FORTRAN packages for assessing initial value methods. ACM Trans. Math. Soft., v.13 no.1, 1-27, 1987.

[11] E Hairer, S P Nørsett, and G Wanner. Solving Ordinary Differential Equations I: Nonstiff Problems. Springer Series in Comput. Mathematics, Vol. 8, Springer-Verlag, Second revised edition 1993.

[12] E Hairer and G Wanner. Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer Series in Comput. Mathematics, Vol. 14, Springer-Verlag, Second revised edition 1996.

[13] http://www.unige.ch/math/folks/hairer/testset/testset.html

[14] http://www.md.chalmers.se/Centres/Phi/cdeforum/cgi-bin/view.cgi

[15] http://www.scicomp.uni-erlangen.de/SW/diffequ.html#testprob

[16] G P Bazeley, Y K Cheung, B M Irons, and O C Zienkiewicz. Triangular elements in plate bending: Conforming and nonconforming solutions. Proc. 1st Conf. on Matrix Methods in Structural Mechanics, AFFDLTR-CC-80. 547-576, 1965.

[17] O C Zienkiewicz and R L Taylor. The finite element patch test revisited. A computer test for convergence, validation, and error estimates. Computer methods in applied mechanics and engineering 149, 223-254, 1997..

[18] R H MacNeal and R L Harder. A Proposed Standard Set of Problems to Test Finite Element Accuracy. Finite Elements in Analysis and Design, vol. 1 No. 1, pp 3-20, 1985.

[19] http://www.nafems.org/

[20] T J Esward, G J Lord, and L Wright. Model Validation in Continuous Modelling. NPL Report No. CMSC 29/03, December 2003.

Available from http://www.npl.co.uk/ssfm/download/documents/cmsc29_03.pdf

[21] L Wright. Visualisation, Uncertainties, and Continuous Modelling. NPL Report No. CMSC 39/04, February 2004.

Available from http://www.npl.co.uk/ssfm/download/documents/cmsc39_04.pdf

[22] http://www.abaqus.co.uk

[23] http://www.ansys.com

[24] http://www.vibroacoustics.co.uk/

[25] T J Esward and L Wright. Guide to the use of finite element and finite difference software. NPL report No. CMSC 30/03, December 2003.

Available from http://www.npl.co.uk/ssfm/download/documents/cmsc30_03.pdf

[26] http://www.uk.comsol.com/femlab/

[27] G A O Davies, R T Fenner and R W Lewis (Editors). Background to Benchmarks. NAFEMS, Birniehill, East Kilbride, Glasgow, 1993. ISBN 1-874376-10-7.

[28] P M Morse and K U Ingard. Theoretical Acoustics, McGraw-Hill, 1968.

Appendix A: Detailed test definitions and results

This appendix gives the full details of the scaleable tests and of the test results produced by the various packages. The NAFEMS tests are not described explicitly: readers who use FE on a regular basis are recommended to consider joining NAFEMS to gain access to their wide range of benchmarks and guidance material.

Scalable test input data

The first problem is a static elastic beam deforming under load. The input data are Young's modulus and the applied load P. The range of values that were used are listed in table A1. Poisson's ration is to be held fixed at 0.3.

The second problem is an axisymmetric transient thermal problem. The input data are λ, the thermal conductivity, ρ, the density, and $c_{p,}$ the specific heat capacity. The values of these three quantities for each of the tests is given in table A2.

The third test simulates the normal modes of vibration of a circular steel plate which is clamped at the edge. The input data are taken to be ρ, E, and v, and the values taken in the various tests are shown in table A3. Later values of E are given to 8 significant figures as this is how they were input.

Detailed test results

Table A4 gives the reference results and test results for all of the NAFEMS small-scale tests for Pafec, Abaqus, and Ansys. Ansys results are only shown for version 7.1. Results for 5.4 were identical apart from for test LE1 with triangular elements, for which Ansys v5.4 gave 89.4 for the coarse mesh and 93.3 for the fine mesh.

Job number	Load P (N)	E (Pa)	Job number	Load P (N)	E (Pa)
1	1	2×10^{6}	6	10^{5}	2×10^{11}
2	10	2×10^{7}	7	10^{6}	2×10^{12}
3	10^{2}	2×10^{8}	8	10^{7}	2×10^{13}
4	10^{2}	2×10^{9}	9	10^{8}	2×10^{14}
5	10^{2}	2×10^{10}	10	10^{9}	2×10^{15}

Table A1: Input data for the beam test.

Test number	λ	ρ	c_p	Test number	λ	ρ	c_p
1	1.01×10^{2}	2.8×10^{-10}	8.0×10^{11}	22	1.01×10^{10}	2.8×10^{5}	8.0×10^{4}
2	1.01×10^{2}	2.8×10^{-9}	8.0×10^{10}	23	1.01×10^{9}	2.8×10^{4}	8.0×10^{4}
3	1.01×10^{2}	2.8×10^{-8}	8.0×10^{9}	24	1.01×10^{8}	2.8×10^{4}	8.0×10^{3}
4	1.01×10^{2}	2.8×10^{-7}	8.0×10^{8}	25	1.01×10^{7}	2.8×10^{3}	8.0×10^{3}
5	1.01×10^{2}	2.8×10^{-6}	8.0×10^{7}	26	1.01×10^{6}	2.8×10^{3}	8.0×10^{2}
6	1.01×10^{2}	2.8×10^{-5}	8.0×10^{6}	27	1.01×10^{5}	2.8×10^{2}	8.0×10^{2}
7	1.01×10^{2}	2.8×10^{-4}	8.0×10^{5}	28	1.01×10^{4}	2.8×10^{2}	8.0×10^{1}
8	1.01×10^{2}	2.8×10^{-3}	8.0×10^{4}	29	1.01×10^{3}	2.8×10^{1}	8.0×10^{1}
9	1.01×10^{2}	2.8×10^{-2}	8.0×10^{3}	30	1.01×10^{1}	2.8×10^{0}	8.0×10^{0}
10	1.01×10^{2}	2.8×10^{-1}	8.0×10^{2}	31	1.01×10^{0}	2.8×10^{-1}	8.0×10^{0}
11	1.01×10^{2}	2.8×10^{0}	8.0×10^{1}	32	1.01×10^{-1}	2.8×10^{-1}	8.0×10^{-1}
12	1.01×10^{2}	2.8×10^{1}	8.0×10^{0}	33	1.01×10^{-2}	2.8×10^{-2}	8.0×10^{-1}
13	1.01×10^{2}	2.8×10^{2}	8.0×10^{-1}	34	1.01×10^{-3}	2.8×10^{-2}	8.0×10^{-2}
14	1.01×10^{2}	2.8×10^{3}	8.0×10^{-2}	35	1.01×10^{-4}	2.8×10^{-3}	8.0×10^{-2}
15	1.01×10^{2}	2.8×10^{4}	8.0×10^{-3}	36	1.01×10^{-5}	2.8×10^{-3}	8.0×10^{-3}
16	1.01×10^{2}	2.8×10^{5}	8.0×10^{-4}	37	1.01×10^{-6}	2.8×10^{-4}	8.0×10^{-3}
17	1.01×10^{2}	2.8×10^{6}	8.0×10^{-5}	38	1.01×10^{-7}	2.8×10^{-4}	8.0×10^{-4}
18	1.01×10^{2}	2.8×10^{7}	8.0×10^{-6}	39	1.01×10^{-8}	2.8×10^{-5}	8.0×10^{-4}
19	1.01×10^{2}	2.8×10^{8}	8.0×10^{-7}	40	1.01×10^{-9}	2.8×10^{-5}	8.0×10^{-5}
20	1.01×10^{2}	2.8×10^{9}	8.0×10^{-8}	41	1.01×10^{-10}	2.8×10^{-6}	8.0×10^{-5}
21	1.01×10^{2}	2.8×10^{10}	8.0×10^{-9}				

Table A2: Input data for the thermal test.

Test number	E	ρ	ν	Test number	E	ρ	ν
1	2.09×10^{11}	7.8×10^{3}	0.33	15	2.09×10^{6}	7.8×10^{-2}	0.33
2	2.09×10^{8}	7.8×10^{0}	0.33	16	2.09×10^{7}	7.8×10^{-1}	0.33
3	2.09×10^{9}	7.8×10^{1}	0.33	17	2.3451812×10^{11}	7.8×10^{3}	0.01
4	2.09×10^{10}	7.8×10^{2}	0.33	18	2.3395522×10^{11}	7.8×10^{3}	0.05
5	2.09×10^{12}	7.8×10^{4}	0.33	19	2.3219616×10^{11}	7.8×10^{3}	0.10
6	2.09×10^{13}	7.8×10^{5}	0.33	20	2.2926439×10^{11}	7.8×10^{3}	0.15
7	2.09×10^{14}	7.8×10^{6}	0.33	21	2.2515991×10^{11}	7.8×10^{3}	0.20
8	2.09×10^{15}	7.8×10^{7}	0.33	22	2.1988273×10^{11}	7.8×10^{3}	0.25
9	2.09×10^{16}	7.8×10^{8}	0.33	23	2.1343284×10^{11}	7.8×10^{3}	0.30
10	2.09×10^{17}	7.8×10^{9}	0.33	24	2.0581023×10^{11}	7.8×10^{3}	0.35
11	2.09×10^{2}	7.8×10^{-6}	0.33	25	1.9701493×10^{11}	7.8×10^{3}	0.40
12	2.09×10^{3}	7.8×10^{-5}	0.33	26	1.8704691×10^{11}	7.8×10^{3}	0.45
13	2.09×10^{4}	7.8×10^{-4}	0.33	27	1.7822814×10^{11}	7.8×10^{3}	0.49
14	2.09×10^{5}	7.8×10^{-3}	0.33				

Table A3: Input data for the vibration test.

Test ID	Reference result	Mesh description	Pafec	Abaqus	Ansys
LE1	92.7	Coarse quad	67.5	82.5	84.9
LE1	92.7	Coarse tri	63.8	89.3	87.3
LE1	92.7	Fine quad	85.1	89.7	91.1
LE1	92.7	Fine tri	83.5	93.3	92.7
LE3	0.185	Coarse	0.078	0.181	0.151
LE3	0.185	Fine	-	0.183	0.178
LE7	25.86	Coarse	26.85	26.00	25.40
LE7	25.86	Fine	26.00	25.89	25.60
T1	50.0	-	49.4	49.2	22.1
T2	927	1-D	927	927	927
T2	927	2-D	927	927	927

Test ID	Reference result	Mesh description	Pafec	Abaqus	Ansys
T4	18.3	Quad	17.9	17.9	19.3
T4	18.3	Tri	17.9	17.7	16.5
FV4	1.722	-	1.722	1.722	1.723
FV4	1.726	-	1.725	1.726	1.727
FV4	7.412	-	7.410	7.410	7.413
FV4	9.953	-	9.947	9.948	9.972
FV4	18.166	-	18.045	18.060	18.160
FV4	27.034	-	26.693	26.740	26.972
FV5	42.657	-	42.647	42.658	42.705
FV5	42.657	-	42.647	42.658	42.705
FV5	77.522	-	77.522	71.261	71.504
FV5	125	-	125	125.00	125.52
FV5	148.71	-	148.34	148.72	150.75
FV5	148.71	-	148.35	148.72	150.75
FV5	232.69	-	232.69	213.89	221.60
FV5	287.81	-	285.39	287.84	301.06
FV5	287.81	-	285.46	287.84	301.06
FV52	45.897	Coarse	46.43	44.796	44.806
FV52	109.44	Coarse	115.02	110.54	110.63
FV52	109.44	Coarse	115.03	110.54	110.63
FV52	167.89	Coarse	176.84	169.10	169.54
FV52	193.59	Coarse	218.41	193.92	193.93
FV52	206.19	Coarse	228.98	206.64	206.64
FV52	206.19	Coarse	230.65	206.64	206.64
FV52	45.897	Fine	53.724	48.435	45.318
FV52	109.44	Fine	140.5	120.23	113.96
FV52	109.44	Fine	140.78	120.23	113.96
FV52	167.89	Fine	210.87	169.36	173.30
FV52	193.59	Fine	224.44	193.53	196.77
FV52	206.19	Fine	236.7	200.19	209.57
FV52	206.19	Fine	240.5	200.19	209.57

Table A4: Reference and test results for the NAFEMS small-scale tests