Use of the Method of the Solutions Manufactured for the software checking

Summary:

This document presents the use of the Method of the Solutions Manufactured for the software validation of *Code_Aster*. The principle of this method is detailed on a simple example of thermal diffusion process. This example allows us to specify the interest of this approach and particularly to present the approach of validation which can be put in work. Lastly, the data-processing establishment of the Method of the Manufactured Solutions, based on the use of calculation symbolic system, is detailed.

Contents

1 Introduction

The checking of a scientific software consists in making sure of the absence of error in the programming of established modeling. A classical and highly reliable approach consists in comparing the result of the software with that of an analytical solution. Unfortunately, for the complex and/or non-linear problems, the analytical solutions become very difficult to obtain. It is within this framework that the Method of the Manufactured Solutions is particularly interesting.

The Method of the Manufactured Solutions is a method of checking of scientific software. It is about a systematic method of obtaining of analytical solutions to problems which can be very complex and non-linear. Its principle is simple: one gives oneself the solution explicitly to be sought in the form of an analytical expression and, starting from this solution, one builds the data (blockings, loadings) necessary to obtaining of this solution.

It is interesting to note that this method is very usually used in mechanics of the fluids. Thus "American Institute of Aeronautics and Astronautics" (AIAA) included it in its standards of software quality assurance in 1998 [bib1] This practice is much rarer in mechanics of the solids; thus "American Society of Mechanical Engineers" (ASME) integrated it in its standard only in 2006 [bib2], that is to say much more recently. That is not related to a particular difficulty, studies pushed in mechanics of the solids having been carried out thanks to this method [bib3].

In the continuation of this document, we will expose the principle of this method on an example the simple then applicable approach of validation. We will detail then the data-processing implementation in *Code_Aster*.

2 Principle general and use for the checking

2.1 Method of the Solutions Manufactured in thermics

The principle of this method is very simple; we will illustrate it on a linear example of thermics 2D. Above all, we point out the equations of a linear problem of thermics. In the field Ω, the field of temperature is searched $T(x,y)$ such as:

$$\begin{cases} -\lambda \Delta T = s \text{ dans } \Omega \\ T = T_d \text{ sur } \partial\Omega_d \\ \lambda \vec{\nabla} T \cdot \vec{n} = q_n \text{ sur } \partial\Omega_n \end{cases} \quad (1)$$

where T_d and q_n are respectively the temperature and the flow imposed on parts of the border.

The stages of the Method of the Manufactured Solutions are the following ones:

- An arbitrary field of study 2D is chosen Ω comprising the borders $\partial\Omega_d$ with conditions of Dirichlet and $\partial\Omega_n$ with conditions of Neumann. By choosing, one intends "to choose the form of the field". Thus we choose the form of the figure 1.
- One chooses an arbitrary solution, say $T(x,y) = x^3 + y^3$
- By reinjecting the preceding expression in (1), the expressions are deduced:
  - source term;

$$s(x,y) = -\lambda \Delta T = -\lambda (6x + 6y) \quad (2)$$

  - term of Dirichlet;

$$T_d(x,y) = x^3 + y^3 \quad (3)$$

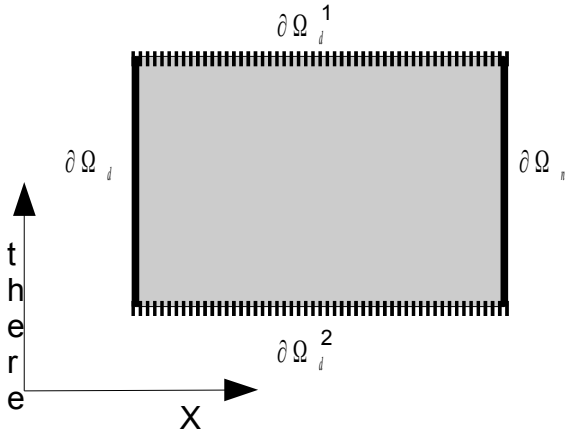
  - term of Neumann:

$$q_d(x,y) = \lambda \vec{\nabla} T(x,y) \cdot \vec{n} \quad (4)$$

If one wants to impose this condition on the edge $\partial\Omega_n$, then one the expression of the normal is known $\vec{n}$ with knowknowing $\vec{n}=[1,0]^T$. One thus obtains:

$$q_d(x,y)=\lambda\vec{\nabla}T(x,y)\cdot\vec{n}=3\lambda x^2 \quad (5)$$

Arrived at this stage, one has the necessary information with the modeling of the studied problem. It is nevertheless essential to pass the following remark: to obtain the sought solution, it is necessary well to affect boundary conditions on all the edge $\partial\Omega$ field of study. With the choice, following a partition of the border, that can be a mixture of conditions of Dirichlet and Neumann. The obligation to impose conditions on all the border is in particular due to the fact that an absence of boundary condition is implicitly a condition of null Neumann. However the equation 3 watch which one does not have the choice of the value of this condition. In the example present, we make the choice to impose a condition of Neumann on $\partial\Omega_n$ and of the conditions of Dirichlet on $\partial\Omega_d$, $\partial\Omega_d^1$ and $\partial\Omega_d^2$.



Drawing 1: Field of study

Once, the preceding equations obtained, one passes to the phase modeling. One starts by choosing a discretization finite elements V_h (linear, quadratic). One models in his software of finite elements preferred the field Ω_h which one applies the loadings $s_h(x,y)$, $T_{dh}(x,y)$ and $q_{dh}(x,y)$. After resolution, the approximate solution is obtained $T_h(x,y)$.

2.2 Techniques of checking

Thanks to the MSM, various checks are realizable.

2.2.1 To choose a reference solution within the space of approximation $T\in V_h$

It is the simplest checking because one must then obtain $T(x,y)=T_h(x,y)$. It should be noted that, because of digital resolution, one will not have perfect equality but the difference must be about the precision machine $\sim 1.E-15$. In a concrete way, if linear elements are used, will have then to be chosen a linear solution.

2.2.2 To choose a reference solution out of the space of approximation $T\notin V_h$

In the case of figure where $T\notin V_h$, there is not any more the equality of the preceding paragraph. On the other hand, one can rather easily measure the speed of convergence of T_h towards T when $h\to 0$.

It is specified that the speed of convergence, with the smoothness h grid, solution calculated towards the analytical solution, in a given standard, is defined as greatest reality $\alpha>0$ such as $\|T(x,y)-T_h(x,y)\|<C*h^{\alpha}$ where C is independent of h.
To measure the speed of convergence, one proceeds as follows:

- One carries out increasingly fine grids of the field Ω
- For each grid, one determines the solution T_h
- For each grid, one measures in a quite selected standard the error $e_h = \|T(x,y) - T_h(x,y)\|$
- One computes the asymptotic command of convergence of e_h towards 0 when $h \to 0$ by the formula $-\log(\frac{e_{h/2}}{e_h})/\log(2)$
- One checks about convergence is in agreement with the theory.

For example, for a solution T sufficient regular, while using normalizes it L_2, the order of convergence of a linear element is at least of 2 tandis the order of convergence of a quadratic element is at least of 3.

2.3 Data-processing aspects

2.3.1 Calculation symbolic system

With the reading of large the principle of the method, it is obvious that the passage of the analytical expression of the solution sought with the data input of the problem finite elements to solve requires many operations of derivation and algebra. To facilitate and automate this stage, one rests features of calculation symbolic system. They are provided by the module Sympy [bib4] of the language Python. It is about a module completely written in Python of a great ease of use.

To facilitate the implementation of the MSM a class TensorModule Python was developed which defines the tensor object like its methods (produced simply and doubly contracted, trace, determinant, …). We also developed the main operators applying to these tensors (gradient, symmetrical gradient, divergence, Laplacian). On the basis of this class, we defined another HookTensor module which simply makes it possible to establish various tensors of Hook (isotropic, anisotropic, orthotropic with nautical angles).

These classes are placed in the repertoire `bibpyt/Utilitai` sources of Code_Aster.

2.3.2 Implementation in Code_Aster

In this part, we describe as the MSM is put in work in the process control language of Code_Aster. With this intention, we base ourselves on the example of preceding linear thermics and detail the principal stages necessary. Implementation the precise is carried out in the CAS-test `tplp107a`.

```
importation sympy
```

One imports the module of calculation Sympy symbolic system, to use his features.

```
X, Y=sympy.symbols ('XY');
```

One defines the variables of space which will be used for calculation symbolic system.

```
T=sympy.Function ('You);
S=sympy.Function (');
```

The name of the functions is defined; T will be obviously the temperature and f the source term.

```
T=100* (X ** 6+Y ** 6);
```

One gives the expression of the manufactured solution which one seeks to find.

```
S=-Lambda* (sympy.diff (sympy.diff (T, X), X) +sympy.diff (sympy.diff (T, Y),
Y));
```

One deduces the expression from the source term of the equation (2).

```
SS=FORMULE (NOM_PARA= ('X', 'Y'), VALE=str (S));
```

This order makes it possible to transform the preceding expression into object formulates of Code_Aster which could be evaluated in the software in the following way:

```
CLIMIT=AFFE_CHAR_THER_F (MODELE=MO,
                         SOURCE=_F (GROUP_MA=' SURFACE',
                                    SOUR=SS,),
                         );
```

3 Bibliography

1 American Institute for Aeronautics and Astronautics, *Guide for the Checking and Validation of Computational Fluid Dynamics Simulations*, 1998.

2 American Society of Mechanics Engineers, *Guide for Checking and Validation in Computational Solid Mechanics*, 2006.

3 Eric Chamberland, André Fortin, Michel Fortin, *Comparison of the performance of nap finite element discretizations for broad deformation elasticity problems*, Computers & Structures, Vol. 88, No 11-12, pp. 664-673, 2010

4 Module Sympy de Python, http://code.google.com/p/sympy/