Advances in Engineering Software 105 (2017) 9–16

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research paper

Abaqus2Matlab: A suitable tool for finite element post-processing

George Papazafeiropoulos[a], Miguel Muñiz-Calvente[b], Emilio Martínez-Pañeda[c,*]

[a] Department of Structural Engineering, National Technical University of Athens, Zografou, Athens 15780, Greece
[b] Department of Construction and Manufacturing Engineering, University of Oviedo, Gijón 33203, Spain
[c] Department of Mechanical Engineering, Solid Mechanics, Technical University of Denmark, Kgs. Lyngby DK-2800, Denmark

ARTICLE INFO

Article history:
Received 20 November 2016
Revised 4 January 2017
Accepted 17 January 2017
Available online 25 January 2017

Keywords:
Abaqus2Matlab
Post-processing
Finite Element Method
Weibull stress model
Inverse analysis

ABSTRACT

A suitable piece of software is presented to connect Abaqus, a sophisticated finite element package, with Matlab, the most comprehensive program for mathematical analysis. This interface between these well-known codes not only benefits from the image processing and the integrated graph-plotting features of Matlab but also opens up new opportunities in results post-processing, statistical analysis and mathematical optimization, among many other possibilities. The software architecture and usage are appropriately described and two problems of particular engineering significance are addressed to demonstrate its capabilities. Firstly, the software is employed to assess cleavage fracture through a novel 3-parameter Weibull probabilistic framework. Then, its potential to create and train neural networks is used to identify damage parameters through a hybrid experimental–numerical scheme, and model crack propagation in structural materials by means of a cohesive zone approach. The source code, detailed documentation and a large number of tutorials can be freely downloaded from www.abaqus2matlab.com.

1. Introduction

Partial Differential Equations (PDEs) govern the physics of most engineering systems. As analytical solutions are limited and generally restricted to idealized cases, the development of efficient and robust numerical methods marks a milestone in the solution of boundary value problems in structural mechanics, electromagnetism, heat transfer, mass diffusion and fluid dynamics, among many other disciplines. The Finite Element Method (FEM) has become the leading numerical technique for solving PDEs in the mechanical, civil, aeronautical and bioengineering industries. Among the wide range of packages available, Abaqus [1] is undoubtedly one of the most popular finite element tools for academics and practitioners.

However, practical applications often require considering non-linear conditions, where uncertainties hinder high fidelity numerical predictions. In such circumstances, the use of advanced analysis methodologies – such as inverse approaches, statistical tools or hybrid experimental–numerical techniques – has proven to compensate the lack of information, yielding results that are otherwise unobtainable. Matlab [2], a multi-paradigm computing environment, is generally considered to be the most powerful software in this regard due to its advanced capabilities in statistics, machine learning, neural networks, curve fitting, model-based calibration and optimization. Yet, a connection between the two most used packages in, respectively, finite element modeling and mathematical analysis, is still lacking. To fill this gap, a novel software tool is here proposed: *Abaqus2Matlab*, which allows to run Abaqus directly from Matlab and to post-process the results, providing a link between the two well-known packages in a non-intrusive and versatile manner. The present proposal enjoys the benefits of Matlab's user friendly and centralized environment, as opposed other powerful tools like Python, which require add-on libraries.

Abaqus2Matlab is distributed as source code with the aim of facilitating research. Numerous codes have been made freely available through the years, positively impacting the computational mechanics community. For instance, Sigmund and co-workers presented an efficient topology optimization implementation [3,4], Bordas and collaborators [5–7] described an object-oriented programming library for the extended finite element method (X-FEM) and meshless methods, Giner et al. [8] implemented the X-FEM in Abaqus through a user subroutine, Parks and Paulino [9] described the numerical implementation of the PPR potential-based cohesive zone model, Nguyen [10] proposed an open source program to generate zero-thickness cohesive elements and Martínez-Pañeda and Gallego [11] provided a user subroutine to effectively define the material property variation of functionally graded materials in Abaqus. Other open-source software that has recently contributed to scientific progress includes FReET [12], a code to conduct statistical, sensitivity and reliability assessment; FraMePID-3PB [13],

* Corresponding author.
E-mail address: mail@empaneda.com (E. Martínez-Pañeda).

http://dx.doi.org/10.1016/j.advengsoft.2017.01.006
0965-9978/

a tool to identify fracture parameters in concrete through inverse analysis; NiHu [14], an open source C++ library for the boundary element method; ESFM [15], a general framework for meshless methods; NOSA-ITACA [16], a finite element code for masonry structures; PCLab [17], an object-oriented Monte Carlo/Finite Element software; and, μMECH [18], an open source C/C++ library of analytical solutions to classical micromechanical problems.

The present manuscript is organized as follows. The software framework and architecture are explained in the following section. Then, Section 3 provides usage instructions through simple examples. Section 4 shows the capabilities of the toolbox by addressing two relevant engineering applications; namely, probabilistic analysis of cleavage fracture and inverse identification of damage parameters through neural networks. Finally, the work is summarized in Section 5.

2. Abaqus2Matlab

The main internal characteristics of *Abaqus2Matlab* are described below. The structure of Abaqus results (*.fil) file is briefly described in the first place, as it is necessary to understand how the presented software stores Abaqus results. The reading procedure is then detailed and insight is given into the software architecture.

2.1. Creating and processing Abaqus' results (*.fil) file

The results (*.fil) file can be used to transfer Abaqus analysis results to other packages. The aforementioned file can be written in binary or ASCII format, depending on the need for porting results between dissimilar operating systems. ASCII format is chosen in the present approach due to its versatility.

2.1.1. Generation of Abaqus results (*.fil) file

The Abaqus results file is obtained in ascii format by defining specific options in the input (*.inp) or restart (*.res) files. The results file generation procedure differs between Abaqus/Standard and Abaqus/Explicit, *FILE FORMAT, ASCII must be specified in the former and *FILE OUTPUT in the latter. The reader is referred to Abaqus documentation for more details.

2.1.2. Output

The following output types can be written to the results file: element, nodal, energy, modal, contact surface, element matrix, substructure matrix and cavity radiation factor. Nodes and elements are numbered globally in models that have been defined as an assembly of part instances. A map between user-defined numbers and internal numbers is printed to the data file (*.dat) if any output requested includes node and element numbers. Set and surface names that appear in the results file are given along with their corresponding assembly and part instance names, separated by underscores.

2.1.3. Record format

The results (*.fil) file is a sequential file that must be read up to the location of the desired data. All data items are converted into equivalent character strings and written in (logical) records. Each single line contains a series of 80 string characters, which may contain the full record or part of it. In the latter case, after completely filling the first line, the record string continues at subsequent lines. The beginning of each record is indicated by an asterisk (*) and the data items are arranged immediately behind each other within each record. Each record has the format shown in Table 1.

The location number denotes the position in the record where a series of consecutive data items are written. The number of data items in each series is denoted by the length number. The first data item is an integer denoting the number of data items in the record. The second one defines the record type key, an indicator denoting the type of data. And finally the attributes are contained in a series of $L-2$ data items, at the 3rd position of a record.

Table 1
Format of a record written in an Abaqus results file.

Location	Length	Description
1	1	Record length (L)
2	1	Record type key
3	($L-2$)	Attributes

```
function Rec = Fil2str(ResultsFileName)
% Open the results file for reading
fileID = fopen(ResultsFileName,'r');
% Read data as a string and assign them to a cell array
% Concatenate each line without specifying delimiter, white
% space or end of line characters
try
C = textscan (fileID, '%s', 'CollectOutput', '1', ...
        'delimiter','','whitespace','','endofline','');
catch
C = textscan (fileID, '%s', 'CollectOutput', 1, ...
        'delimiter','','whitespace','','endofline','');
end
% Close the results file
fclose(fileID);
% Assign A
A = C{1}{1};
% Remove newline characters
A1 = strrep(A,sprintf('\n'),'');
% Remove carriage return characters
Rec = strrep(A1,sprintf('\r'),'');
```

Listing 1. Function Fil2str.m to read Abaqus results (*.fil) file.

2.1.4. Data item format

Integer numbers are denoted by the character I, followed by a two digit integer which shows the number of the digits of the integer with the value of the integer following. On the other hand, floating point numbers begin with the character D, followed by the number in the format E22.15 or D22.15, depending on the precision. And character strings begin with the character A, followed by eight characters. If the length of a character string is less than 8, then the trailing positions are filled with blank spaces. If the length of a character string is larger than 8, then the character string is written in consecutive character strings, eight characters at a time.

2.2. Reading Abaqus results files with Abaqus2Matlab

A function named `Fil2str` is defined in Matlab to read the Abaqus results (*.fil) file by considering the data as a string and concatenating lines horizontally, as shown in Listing 1.

The function is programmed so as to allow compatibility between different MATLAB versions. The information from the results file is stored in a cell array `C` containing a single line string. That single line string subsequently enters an *ad hoc* function that depends on the results that the user wishes to post-process. Thus, more than 50 different functions are already available in *Abaqus2Matlab*, covering the vast majority of results types that can be obtained in Abaqus; new record functions can be easily generated from the existing template. An appropriate naming convention is adopted, where each function is defined by the word `Rec` followed by the record key of the particular type of results. Record keys for each specific set of results can be found in Abaqus documentation. For example, nodal coordinates (record key 1901) are obtained through function `Rec1901.m`, whose code is shown in Listing 2.