A DEVELOPER'S GUIDE TO ABAQUS2MATLAB

Notes on the structure and usage of Abaqus2Matlab. Intended to complement the information provided in the article:

G. Papazafeiropoulos, M. Muñiz-Calvente, E. Martínez-Pañeda. Abaqus2Matlab: a suitable tool for finite element post-processing. *Advances in Engineering Software*

<u>Index</u>

1. Structure of Abaqus Results (*.fil) files

2. Reading of Abaqus results files with Abaqus2Matlab

3. Use of Abaqus2Matlab

If using Abaqus2Matlab for research or industrial purposes, please cite the aforementioned article.

Last update: January 2017

1. Structure of Abaqus Results (*.fil) files

A medium in which Abaqus analysis results can be transferred to other software for postprocessing or pre- and postprocessing is the results file. The Abaqus results file can be written in binary (default) or ASCII format. Generally, the manipulation of results files in asci format is easier than in binary format, since they can be transferred between different computer systems and read from many different postprocessing software without special settings. On the other hand, for large problems the results files in ascii format are significantly larger than the same files in binary format. Abaqus provides the **ascfil** facility to convert a results file from binary to ascii format. The discussion from now on will concern only Abaqus results files in ascii format.

1.1 Data item format

Any data item contained in a results file can be either integer, floating point number or character string. Integers begin with the character I, followed by a two-digit integer which shows the number of the digits of the integer, followed by the integer itself. If the number of digits of the integer has one digit, the first character after character I is a blank space. For example, integer number "*8*" would be written as "`I 18`" and integer number "*9999999999*" would be written as "`I109999999999`".

Floating point numbers begin with the character D, followed by the number in the format E22.15 or D22.15, depending on the precision (single or double respectively). For example, number "*0.5*" in double precision would be written as "`D 5.000000000000000D-01`".

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. For example, "*HOMOGENEOUS TENSION FOR ELEMENT 1*" would be written as "`AHOMOGENEAOUS TENSAION FOR AELEMENT A1       `". Note the seven trailing blank spaces after the last character ("1") in the last character string.

1.2 Record format

The results file is a sequential file, meaning that it contains and stores data records in a specific order. It must be read from the beginning, up to the location of the desired data. All data items are converted into equivalent character strings and written in series which are called (logical) records. Each single line of a results file contains a series of 80 string characters, which may contain whole or part of a record. In the latter case, after completely filling the first line in which a record begins, the record string continues at the subsequent lines till the end of the record. If a record string ends before the end of a line, then the next record starts immediately after the current record in the same line, with continuation in the subsequent lines as explained above. The beginning of each record is indicated by an asterisk (*). Within each record, the data items are arranged immediately behind each other, and therefore it is possible that the end of a line splits a data item, with its first characters belonging to a line and the remaining characters belonging to the next line. The last line of the results file, if partially completed, is filled with blank spaces until the end of the line. Then, a logical record consisting of 80 blanks is inserted as the next line, in order for the end-of-file to be handled correctly.

Each record has the format shown in Table 1:

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

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

The location number denotes the position in the record, where a series of consecutive data items is written. The number of data items in each series is denoted by the length number. The first series of data items (consisting of a single data item) is an integer showing the record length, i.e. the number of data items which the record contains. The second series of data items (also consisting of a single data item) is an integer showing the record type key. The record type keys are standard indicators set in Abaqus by convention, and denote the type of data which the record includes. The data items which actually provide useful information for the user (or attributes) are contained in a series of L-2 data items, at the 3rd (and last) position of a record. For example, record key 1900 (Record type: Element definition) for a CPE4R element with element number *2* and nodes *5*, *6*, *7*, and *8* would be written as follows:

```
*I 18I 41900I 12ACPE4R   I 15I 16I 17I 18
```

and record key 101 (Output variable identifier: U, i.e. displacements) for node *145* and displacements for the 6 degrees of freedom equal to (*0.2000000029802322*, *0.00*, *-0.07500000298023224*, *1.732049942016602*, *1.732049942016602 and 1.732049942016602*) would be written as:

```
*I 19I 3101I 3145D 2.000000029802322D-01D 0.000000000000000D+00D-
7.50000029802322          4D-02D          1.732049942016602D+00D
1.732049942016602D+00D 1.732049942016602D+00
```

In a data record which contains complex values (e.g. in a steady-state analysis), all the real components of the data record are written first and all the imaginary components follow immediately. For example, record key 101 (Output variable identifier: U, i.e. displacements) for node *1* and complex displacements for the 6 degrees of freedom equal to (*-1.621881950939540e^{-16}+0.50939i*, *0.004367975320916413+0.67975i*, *-1.558539209401511e^{-15}+0.055i*) would be written as:

```
*I 19I 3101I 11D-1.621881950939540D-16D 4.367975320916413D-03D-
1.558539209401511          D-15D          0.509390000000000D+00D
0.679750000000000D+00D 0.055000000000000D+00
```

1.3 Output

The types of output that can be written to the results file are the following:

- element output, nodal output, energy output, modal output, contact surface output, and section output
- element matrix output
- substructure matrix output
- cavity radiation view factor matrices

It is possible that a model is defined as an assembly of part instances, the nodes and/or the elements of which have repeated numbering definitions. In this case the local node and element numbers are converted internally into global node and element numbers, which are unique for the model being analyzed. The output in the results file is given in terms of these global identities. A map between user-defined numbers and internal numbers is printed to the data file (*.dat) if any results file output that includes node and element numbers is requested.

Set and surface names that appear in the results file are given along with their corresponding assembly and part instance names, separated by underscores. For example, if *Set1* is the name of a set or surface of part *Part1*, which is instanced in the assembly *Assembly1*, then this set appears with the name Assembly1_Part1_Set1 in the results file.

1.4 Generation of Abaqus Results (*.fil) files

Abaqus results files can be produced in a variety of ways. The overall implementation which includes the generation of the results file(s) depends on the information flow between Abaqus and other pre- and postprocessing software. In order to retrieve the results of an analysis in an easy to handle form, results files in ascii format must be generated. This can be achieved by determining specific execution procedures, which can involve input (*.inp), restart (*.res), and other types of files which can be found in the Abaqus Documentation. In each of the input files involved, specific options with specific parameters have to be defined. In addition, the results file generation procedures differ between Abaqus/Standard and Abaqus/Explicit. The execution procedures for Abaqus/Standard and Abaqus/Explicit, the required files as well as the options in the input files of the single or restart analysis are shown in Table 2. Four procedures are presented, which combine Abaqus/Standard and Abaqus/Explicit finite element programs with either single or restart analysis, resulting thus in four different cases. The abaqus ascfil utility serves to convert results files from binary to ascii format. This is particularly useful when Abaqus/Explicit is used for the analyses, in which the results files generated can be only in binary format. In the case of a restart analysis, the *FILE FORMAT, ASCII and *FILE OUTPUT options for Abaqus/Standard and Abaqus/Explicit respectively have to be specified either in the initial or in the restart input files.

Finite element program	Execution command	Files required	Input file options		Result files generated
Abaqus/Standard - single analysis	abaqus job=1	1.inp	Required	Optional	1.fil (ascii)
			*FILE FORMAT, ASCII	*CONTACT FILE	
				*EL FILE	
				*ENERGY FILE	
				*MODAL FILE	
				*NODE FILE	
				*SECTION FILE	
Abaqus/Standard - restart analysis	abaqus job=2 oldjob=1	2.inp	Required	Optional	2.fil (ascii)
		1.mdl	*POST OUTPUT	*CONTACT FILE	
		1.odb	*FILE FORMAT, ASCII	*EL FILE	
		1.stt		*ENERGY FILE	
		1.prt		*MODAL FILE	
		1.res		*NODE FILE	
				*SECTION FILE	
Abaqus/Explicit - single analysis	abaqus job=1	1.inp	Required	Optional	1.fil (binary)
			*FILE OUTPUT	*CONTACT FILE	
				*EL FILE	
				*ENERGY FILE	
				*MODAL FILE	
				*NODE FILE	
				*SECTION FILE	
Abaqus/Explicit - restart analysis	abaqus job=2 oldjob=1	2.inp	Required	Optional	2.fil (binary)
		1.abq	*RESTART,READ	*CONTACT FILE	
		1.mdl	*FILE OUTPUT	*EL FILE	
		1.odb		*ENERGY FILE	
		1.stt		*MODAL FILE	
		1.pac		*NODE FILE	
		1.prt		*SECTION FILE	
		1.res			
		1.sel			
-	abaqus ascfil job=1	1.fil (binary)	-		1.fin (ascii)

Table 2: Procedures used in Abaqus for the generation of results (*.fil) files.

2. Reading of Abaqus results files with Abaqus2Matlab

This section describes the way an Abaqus result file is read, in order to obtain the numerical data in an easy to use form. A segment of a results file is shown in Figure 1. As mentioned in a previous section, each single line of a results file contains a series of 80 string characters, which may contain whole or part of a record. The segment shown in Figure 1 contains three records. Before the first record, the last 74 characters of the last record appear. After the third record, the first 98 characters of the next record appear.

```
...
4D-02D 1.732049942016602D+00D 1.732049942016602D+00D 1.732049942016602D+00*I 19I
41901I 3262D 1.147152855992317D-01D-1.638304144144058D-01D 7.500000298023224D-0
2D 1.732049942016602D+00D 1.732049942016602D+00D 1.732049942016602D+00*I 19I 419
01I 3263D 1.285575181245804D-01D-1.532088816165924D-01D 7.500000298023224D-02D 1
.732049942016602D+00D 1.732049942016602D+00D 1.732049942016602D+00*I 19I 41901I
3264D 1.414213627576828D-01D-1.414213627576828D-01D 7.500000298023224D-02D 1.732
049942016602D+00D 1.732049942016602D+00D 1.732049942016602D+00*I 19I 41901I 3265
D 1.532088816165924D-01D-1.285575181245804D-01D 7.500000298023224D-02D 1.7320499
...
```

Figure 1: Segment of the contents of an Abaqus results file.

The way Abaqus2matlab reads the segment of the results file presented in Figure 1 will be illustrated. For this purpose, the code used for reading the segment will be shown and explained line by line.

The function Fil2str (<u>fil</u> file <u>to</u> <u>str</u>ing conversion), the code of which is shown in Figure 2, opens the Abaqus results file for reading only, reads the data in this file by considering it as a string and concatenating lines horizontally, so that the cell array C contains a single line string. It is reminded that all lines starting with "%" are not executed and are treated as comments. Special characters as delimiters, whitespaces or end of line characters are not specified. The concatenation in a single line during execution of textscan does not happen in previous versions of Matlab, and therefore the newline and carriage return characters of the string A contained in the 1 x 1 cell array C have to be deleted (replaced with nothing) by applying two strrep (<u>str</u>ing <u>rep</u>lacement) commands consecutively, as shown in lines 19 & 21 of the code shown in Figure 2, in order to yield a single line string containing all information of the results file. A single line output string is necessary, since this is the only way to manipulate whole records easily, avoiding interruptions due to continuation to subsequent lines. After the application of Fil2str function, all lines of the segment in Figure 1 will be arranged in a single line as shown in Figure 3.

```matlab
function Rec = Fil2str(ResultsFileName)
% Open the results file for reading
fileID = fopen(ResultsFileName,'r');
% Read data from results file as a string and assign them to a cell array
% Concatenate each line without specifying delimiter, whitespace 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'),'');
```

Figure 2: Matlab code of the function Fil2str.m.

```
...4D-02D 1.732049942016602D+00D 1.732049942016602D+00D 1.732049942016602D+00*I 1 9I 41901I 32
62D 1.147152855992317D-01D-1.638304144144058D-01D 7.500000298023224D-02D 1.7320499420166O 2D+0
0D 1.732049942016602D+00D 1.732049942016602D+00*I 1 9I 41 901I 3263D 1.285575181245804D-01D-1.
532088816165924D-01D 7.500000298023224D-02D 1 .732049942016602D+00D 1.732049942016602D +00D 1.
732049942016602D+00*I 19I 41901I3264D 1.414213627576828D-01D-1.414213627576828D-01D 7.50000029
8023224D-02D 1.732049942016602D+00D 1.732049942016602D+00D 1.732049942016602D+00*I 19I 41901I 3
265D 1.532088816165924D-01D-1.285575181245804D-01D 7.500000298023224D-02D 1.732049 9...
```

Figure 3: Single line string extracted from the data in Figure 1.

The single line string, after being produced by Fil2str function, enters another suitable function specified by the user, depending on the type of the results to be extracted from this string. The string shown in Figure 3 contains node definition data (which are identified by the record key 1901 in Abaqus) and a function which can read node definitions from the string must be used. Of course, the string may contain more than one types of data (such as nodal displacements, for example), but there is not a unique function which can extract all types of data from a string. For each type of data to be extracted, the corresponding function has to be used. Abaqus2matlab contains 61 different functions which can read 61 different types of results from a single line string that has been produced from a results file. To avoid confusion, there is a standard naming convention of these functions. For example, in order to extract node definition results (record key 1901 as mentioned above) the function Rec1901.m has to be used, namely, the name of the function is comprised of "Rec" followed by the record key of the results to be read. In the string shown in Figure 3, the results correspond to record key 1901. Therefore, the function Rec1901 has to be used to read these results. The code of all such functions follows a similar logic, which does not differ significantly from the logic described for the code of the function Rec1901.m. The application of this function is explained in the following.

```
function out = Rec1901(Rec)
ind = strfind(Rec,'I 41901'); % record key for node output (1901)
if isempty(ind)
    out=[];
    return;
end
nextpos=numel('I 41901')+1;
% Initialize
NodeNum=zeros(numel(ind),1);
% Initialize record length matrix
NW=zeros(numel(ind),1);
for i=1:numel(ind)
    % find the record length (NW)
    Rec2=Rec(ind(i)-7:ind(i));
    indNW=strfind(Rec2,'*'); % record starts with *
    % ensure that the record exists and that the record type key is at
    % location 2
    if isempty(indNW) || indNW>3
        ind(i)=NaN;
        continue;
    end
    % number of digits of record length
    ind1=indNW+2;
    ind2=indNW+3;
    a1=str2num(Rec2(ind1:ind2));
    % Record length (NW)
    ind1=ind1+2;
    ind2=ind2+a1;
    NW(i)=str2num(Rec2(ind1:ind2));
end
NodeCoords=zeros(numel(ind),max(NW)-4);
for i=1:numel(ind)
    % number of digits of node number
    ind1=ind(i)+nextpos;
    ind2=ind(i)+nextpos+1;
    a1=str2num(Rec(ind1:ind2));
    % Node number
    ind1=ind1+2;
    ind2=ind2+a1;
    NodeNum(i)=str2num(Rec(ind1:ind2));
    % Node coordinates
```

```
42      for j=1:NW(i)-4
43          % node coordinate
44          ind1=ind2+2;
45          ind2=ind2+23;
46          NodeCoords(i,j)=str2num(Rec(ind1:ind2));
47      end
48  end
49  % Assemply of matrices for output
50  out=[NodeNum NodeCoords];
51  end
```

Figure 4: Matlab code of the function Rec1901.m

The Rec1901.m function works as follows. In order to accelerate matrix storage in Matlab, preallocation of the results matrix has to be made, especially for large output. In order to preallocate the results matrix, the record length has to be known. To find the record length, the positions of the record key in ascii form (“`I 41901`”) are found first using the `strfind` function (line 2). The position of the record key is meant to be the position of its first character (i.e. the character `I`). These positions for the example string in Figure 3 are [235 391 546]. After this, a typical check is made if the array `ind` is empty (i.e. if no string “`I 41901`” is found). In positive case, the function is exited giving as output an empty matrix (lines 3 – 6). This case can be encountered if in the results file no nodal definition data are written for some reason.

Thereafter, the record length matrix is initialized, having number of rows equal to the number of elements in `ind` array. It is known that the record length is written one position before the record key number and therefore the pointer goes back from the position of the record key by a default number of 7 characters and stores these characters in string `Rec2`. After this, the string `Rec2` is searched for “`*`”, to determine the positions where the records start. If there is not an asterisk, then this means that the record does not start within these seven characters, and consequently the string “`I 41901`” does not signify a record key (it could be the number of a node in an element definition for example). Another point to be noted is that `indNW` (which shows the location of the asterisk (`*`) within the seven characters preceding the string “`I 41901`”) cannot be larger than 3; this would mean that the first data item of the record includes less than 7-3=4 characters, which is not possible, since if this occurs, only the number of digits of the record length will be known, and not the record length itself (three characters include the character `I` followed by at most two numerical characters). In any of the two cases, `ind` is set equal to NaN, so that results in the corresponding positions are not read. After having ensured that `indNW` shows the position of the beginning of a record, the number of digits of the record length is read using the function `str2num`, which converts a string into a number. In a similar way, the record length is read (with indexing based on the number of its digits given previously) and assigned to array `NW`.

Having formed the array `NW`, its maximum value is taken to set the number of columns of the output matrix at preallocation, denoted as `out`. The number of rows of this matrix is set to be equal to the number of elements of `ind`. After this, the elements of `ind` (i.e. position of the second data item of records giving node definition results) are scanned and for each element the number of digits of node number is determined first, then the node number, and finally the nodal coordinates, by the insertion of a for loop within each record definition, intended to scan the three coordinates (x,y,z) of each node. Finally, the node numbers and the node coordinates are concatenated horizontally to form the output array `out`.

3. Use of Abaqus2Matlab

Before using the Abaqus2Matlab toolbox, the user has to be aware of the various source codes contained and how they are organized. Knowledge of the source codes will enable the user to use the toolbox more effectively to perform the desired postprocessing of the Abaqus results. In this section, after a description of how the various files are organized in the toolbox, detailed instructions are given for the use of Abaqus2matlab.

3.1. Organization of source code

The source code files and folders used in the toolbox are the following:

3.1.1. A function named Fil2str.m that converts the contents of the results file into a one-row string from which the desired output is retrieved, as already mentioned in previous sections.

3.1.2. A folder named OutputAnalysis which contains the functions available for the processing of the results of analysis type (e.g. node definitions, element connectivity, eigenfrequencies and eigenvalues, etc). A table of variables available for analysis output requests is shown in Table 3. The first column (with title “record type”) describes the variable which is written in the Abaqus results file for the corresponding record key shown in the second column. In the third column the output variable identifier is written. The output variable identifier is the identifying key for the variables to be written to the results (.fil) or selected results (.sel) file. The keys are defined in the sections 4.2.1. (“Abaqus/Standard output variable identifiers”) and 4.2.2. (“Abaqus/Explicit output variable identifiers”) of the Abaqus Analysis User's Guide (version 6.13). In the fourth column, the Matlab function suitable for the extraction of the corresponding variable from the Abaqus results file is shown.

ANALYSIS RECORD TYPE	RECORD KEY	OUTPUT VARIABLE IDENTIFIER	FUNCTION
Element definitions	1900	-	Rec1900.m
Node definitions	1901	-	Rec1901.m
Modal	1980	-	Rec1980.m

Table 3: List of variables available in Abaqus2Matlab for analysis output requests

3.1.3. A folder named OutputNodes which contains the functions available for the processing of the results of nodal type (e.g. node displacements, concentrated forces, nodal temperatures, etc). A table of variables available for nodal output requests is shown in Table 4, in which the various data are presented in the same way as in Table 3. The variables are ordered according to the output variable identifiers’ names, alphabetically.

NODAL RECORD TYPE	RECORD KEY	OUTPUT VARIABLE IDENTIFIER	FUNCTION
Nodal Acceleration	103	A	Rec103.m
Concentrated Electrical Nodal Charge	120	CECHG	Rec120.m
Concentrated Electrical Nodal Current	139	CECUR	Rec139.m
Nodal Point Load	106	CF	Rec106.m
Concentrated Flux	206	CFL	Rec206.m
Nodal Coordinate	107	COORD	Rec107.m
Fluid Cavity Volume	137	CVOL	Rec137.m
Electrical Potential	105	EPOT	Rec105.m
Motions (in Cavity Radiation Analysis)	237	MOT	Rec237.m
Normalized Concentration (Mass Diffusion Analysis)	221	NNC	Rec221.m
Temperature	201	NT	Rec201.m
Fluid Cavity Pressure	136	PCAV	Rec136.m
Pore or Acoustic Pressure	108	POR	Rec108.m
Electrical Reaction Charge	119	RCHG	Rec119.m
Electrical Reaction Current	138	RECUR	Rec138.m
Nodal Reaction Force	104	RF	Rec104.m
Residual Flux	204	RFL	Rec204.m
Internal Flux	214	RFLE	Rec214.m
Reactive Fluid Volume Flux	109	RVF	Rec109.m
Reactive Fluid Total Volume	110	RVT	Rec110.m
Total Force	146	TF	Rec146.m
Nodal Displacement	101	U	Rec101.m
Nodal Velocity	102	V	Rec102.m
Viscous Forces Due to Static Stabilization	145	VF	Rec145.m

Table 4: List of variables available in Abaqus2Matlab for nodal output requests.

3.1.4. A folder named OutputElements which contains the functions for the processing of the element results (results at the element integration points or results regarding whole elements, e.g. total strains, section forces and moments, etc). A table of variables available for element output requests is shown in Table 5, in which the various data are presented in the same way as in Table 3. The variables are ordered according to the output variable identifiers' names, alphabetically.

ELEMENT RECORD TYPE	RECORD KEY	OUTPUT VARIABLE IDENTIFIER	FUNCTION
Creep Strain (Including Swelling)	23	CE	Rec23.m
Mass Concentration (Mass Diffusion Analysis)	38	CONC	Rec38.m
Concrete Failure	31	CONF	Rec31.m
Coordinates	8	COORD	Rec8.m
Unit Normal to Crack in Concrete	26	CRACK	Rec26.m
Total Strain	21	E	Rec21.m
Total Elastic Strain	25	EE	Rec25.m
Energy (Summed over Element)	19	ELEN	Rec19.m
Energy Density	14	ENER	Rec14.m
Mechanical Strain Rate	91	ER	Rec91.m
Whole element volume	78	EVOL	Rec78.m
Film	33	FILM	Rec33.m
Total Fluid Volume Ratio	43	FLUVR	Rec43.m
Pore Fluid Effective Velocity Vector	97	FLVEL	Rec97.m
Gel (Pore Pressure Analysis)	40	GELVR	Rec40.m
Heat Flux Vector	28	HFL	Rec28.m
Total Inelastic Strain	24	IE	Rec24.m
Logarithmic Strain	89	LE	Rec89.m
Nominal Strain	90	NE	Rec90.m
Nodal Flux Caused by Heat	10	NFLUX	Rec10.m
Plastic Strain	22	PE	Rec22.m
Pore or Acoustic Pressure	18	POR	Rec18.m
Radiation	34	RAD	Rec34.m
Stress	11	S	Rec11.m
Saturation (Pore Pressure Analysis)	35	SAT	Rec35.m
Section Strain and Curvature	29	SE	Rec29.m
Section Force and Moment	13	SF	Rec13.m
Stress Invariant	12	SINV	Rec12.m
Strain Jump at Nodes	32	SJP	Rec32.m
Principal stresses	401	SP	Rec401.m
Average Shell Section Stress	83	SSAVG	Rec83.m
Element Status	61	STATUS	Rec61.m
Section Thickness	27	STH	Rec27.m
Thermal Strain	88	THE	Rec88.m

Table 5: List of variables available in Abaqus2Matlab for element output requests

3.1.5. A folder named MatlabExamples, which contain Matlab scripts for the verification of the Fil2str.m and the various RecX.m functions (where X is the record key). All the functions provided with this toolbox and associated with obtaining analysis, element or nodal results are verified to ensure that they work correctly and they are not error-prone. In the verification process, an appropriate Abaqus input file (in which the option for the extraction of the desired results in an ascii results file (.fil) is specified), is run by Abaqus. After the Abaqus analysis terminates and the results file is created in the Abaqus working directory, it is processed appropriately by Abaqus2Matlab to obtain the requested results. Finally, the results are presented and checked with regard to their class and size. The verification of Abaqus2Matlab toolbox was made using Abaqus 6.13.

3.1.6. A folder named AbaqusInputFiles which contains the input files which are run by Abaqus for the verification described in section 3.1.5. Each Abaqus input file is named with a number (let it be Y) which is the record key of the corresponding output variable identifier, followed by the extension ".inp". The Abaqus input file Y.inp is run by Abaqus and produces results which are retrieved (after Abaqus completes the analysis) by the function RecY.m.

3.1.7. A folder named help which contains all Matlab source codes which are intended to print the contents of the Abaqus input files contained in the folder AbaqusInputFiles.

3.1.8. A folder named html which contains all the html files of the documentation of Abaqus2Matlab, including the html files produced by publishing the verification examples. All the verification examples contained in the folder MatlabExamples and the Abaqus input files contained in the folder help are published by Matlab in this folder and are accessible through the documentation.

3.2. Instructions for use of Abaqus2Matlab

To use Abaqus2Matlab, follow the instructions below:

3.2.1. Ensure that Abaqus license server is running.

3.2.2. Open the file named Documentation.m in Matlab and run it (press F5). This action virtually sets up all files and folders contained in the Abaqus2matlab toolbox, including the documentation. It is noted that the files generated during Abaqus analyses will be placed one level up (outside) of the toolbox folder. The command "S=pwd" finds the directory containing the file Documentation.m, wherever may it be. The command "addpath(genpath(S))" does the setup.

3.2.3. To extract an arbitrary Abaqus analysis result from an Abaqus results file, initially the record key and the output variable identifier have to be specified. These can be obtained from Table 3 for an analysis-type output, Table 4 for a node-type output, and Table 5 for an element-type output.

3.2.4. The syntax of each RecX.m function has to be known (especially regarding its output). To view the syntax of an arbitrary RecX.m function type "doc RecX" or "help RecX" (where X is the record key found in step 3.2.3. above) in the Matlab command window. The first option shows the function manual in a Matlab browser, whereas the second option shows the function manual in the

Matlab command window. In the manual of each function the necessary options to be included in the corresponding Abaqus input file are shown.

3.2.5. Create the Abaqus input file and place it in the folder of the Abaqus2Matlab toolbox (at least at the same level as the Documentation.m script and anyway not outside the toolbox folder.

3.2.6. Run the Abaqus input file by typing in the Matlab command window “!abaqus job=X”, then enter. After the analysis terminates, the results file X.fil is generated in the same directory as the X.inp file. The results file is then read by Abaqus2Matlab to extract the requested results.

3.2.7. Type in the Matlab command window “Rec=Fil2str('X.fil')”. The variable Rec is a one-row string containing the information included in the X.fil file.

3.2.8. Type in the Matlab command window “out=RecX(Rec)”. The variable out contains the requested results, extracted from the X.fil results file. It will be generally a double or cell array. For more information about the identity and/or physical meaning of each element contained in this array, one can refer to the manual of the function RecX.m, mentioned in section 3.2.4. above, or section 5.1.2 (Results file output format) of the Abaqus 6.13 Analysis User's Guide.