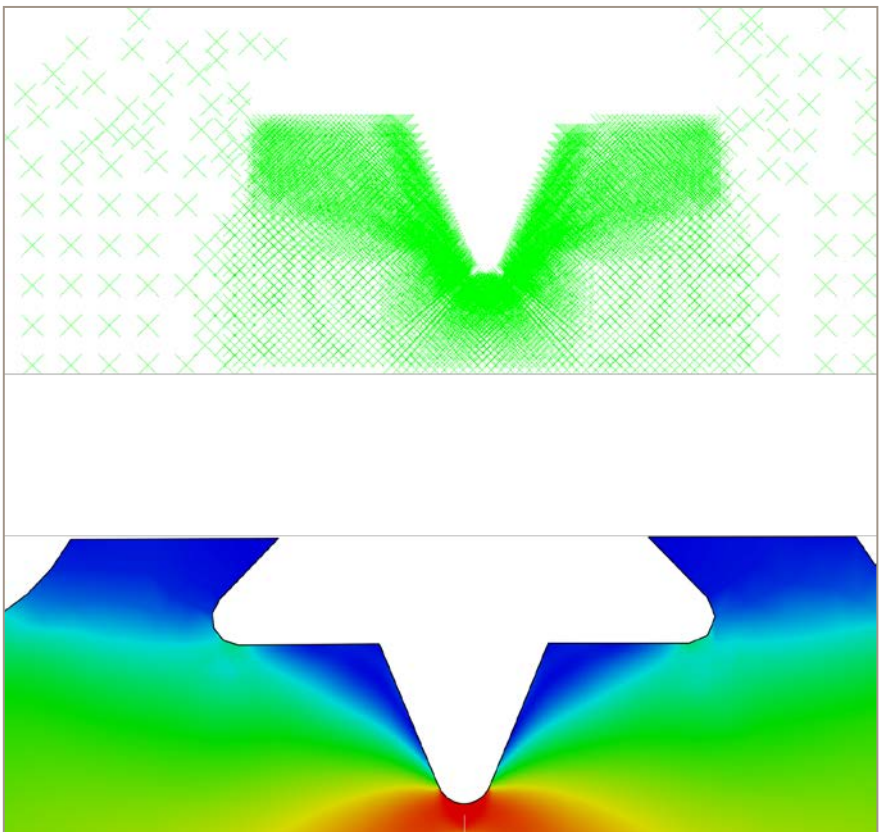
SINTEF

2019:00006 Unrestricted

Report

Visualization of user-defined elements in Abaqus

Author(s)
Vidar Osen



SINTEF Industry
Materials Integrity and Welding
2019-01-02

SINTEF

SINTEF Industri
SINTEF Industry
Address:
Postboks 4760 Torgarden
NO-7465 Trondheim
NORWAY
Switchboard: +47 73593000

info@sintef.no

Enterprise /VAT No:
NO 919 303 808 MVA

Report

Visualization of user-defined elements in Abaqus

KEYWORDS:
Abaqus
Visualization
User-defined elements

VERSION
2.0

DATE
2019-01-02

AUTHOR(S)
Vidar Osen

CLIENT(S)
The Research Council of Norway, Equinor, Gassco, TechnipFMC, EDF Induction, POSCO

CLIENT'S REF.
Kimberly Mayes, RCN

PROJECT NO.
MK102006704

NUMBER OF PAGES:
7

ABSTRACT
Abaqus is a software system for finite element analysis. It comes with standard elements which can be used for solving problems based on established theory. In addition to the standard elements, Abaqus offers a possibility for the users to write their own elements, so-called user-defined elements. A significant drawback when running calculations with user-defined elements is that the Abaqus' post-processor is not able to visualize user-defined elements, these will be invisible in the post-processor. In addition, there is no easy way to extract results from the user-defined elements in the same way as for the standard elements.

This report contains a short description of a method which makes it possible to visualize and extract data from user-defined elements in Abaqus. The method is relatively straightforward and easy to implement. No preparation is necessary during creation of the finite element model. It requires the user-defined element to persistently store the results of interest (i.e. to a file/database) during the analysis. After the finite element calculation has finished, a python-script will read these results and make them available for visualization in Abaqus.

PREPARED BY
Vidar Osen

SIGNATURE


CHECKED BY
Philippe Mainçon

SIGNATURE


APPROVED BY
Magnus Eriksson

SIGNATURE


REPORT NO.	ISBN	CLASSIFICATION	CLASSIFICATION THIS PAGE
2019:00006	978-82-14-06813-9	Unrestricted	Unrestricted

Document history

VERSION	DATE	VERSION DESCRIPTION
1.0	2018-12-21	Internal draft

VERSION	DATE	VERSION DESCRIPTION
2.0	2019-01-02	Issue for client review

Table of contents

1 Introduction

1.1 Typical problems when dealing with user-defined subroutines in Abaqus

When running a calculation with Abaqus' standard elements, the user can choose which results to store in the output database file (odb). The instruction to Abaqus will look like this in the input (inp) file:

```
*Output, field
*Element Output
S, SINV
```

This will tell the Abaqus solver to put the stress tensor (S) and the stress invariants (SINV) as field data into the odb file. These results will then be available to the post-processor and can be used to colorize the model (contours), generate curves and export data.

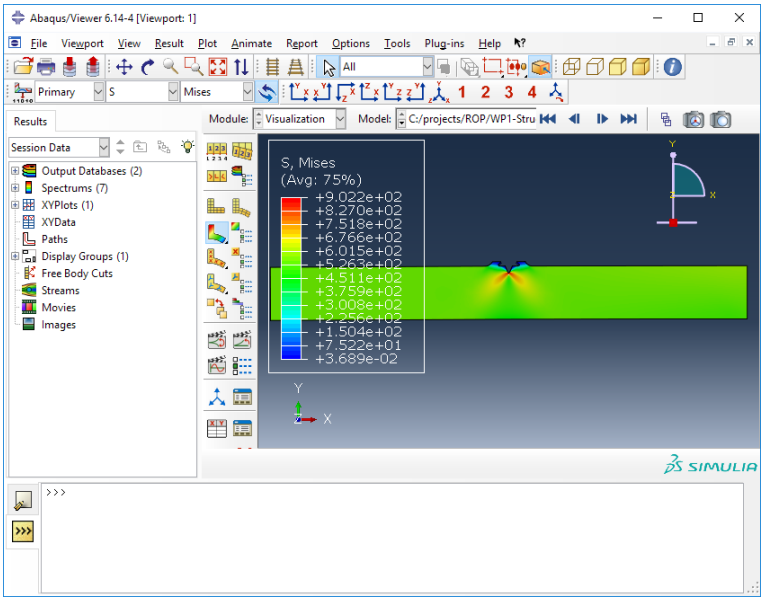


Figure 1: Typical view of results from Abaqus model with standard elements.

If a user-defined *material* is used, the user must provide an implementation of the material's constitutive behaviour. Examples of user-defined material subroutines are UMAT() (for mechanical constitutive behaviour) and UMATHT() (for thermal constitutive behaviour). The implementation will be called for each integration point in the elements. The user-defined material implementation can read and write results and other data in each integration point by using so-called solution-dependent state variables (SDV, STATEV). These values can be stored in the odb file the same way as for Abaqus' standard results and are available for visualization and post-processing. Here is an example on how to put the first three SDVs into the odb file:

```
*Output, field
*Element Output
S, SINV, SDV1, SDV2, SDV3
```

If a user-defined *element* type is used, the user must provide an implementation of the element behaviour. An example of a user-defined element subroutine is UEL(). The implementation will be called for each user-defined element of this type. As for the user-defined materials, state variables (SDV, SVARS) are available for the implementation to read and write results for the element. These state variables are stored in memory per element.

The challenge is now: Abaqus cannot write results from user-defined elements to the odb file. If one tries the above command (`*Element Output`), the Abaqus solver will give you this warning:

```
***WARNING: THE *ELEMENT OUTPUT OPTION IS NOT SUPPORTED FOR USER
ELEMENTS
```

A probable reason for this is that the element output relies on the configuration of the integration points (number of points and their positions in the element). This is an internal issue for the element and Abaqus does not have this information for the user-defined elements.

When opening an odb file with results from user-defined elements, the elements will not be visible in the viewer and no results are available for post-processing.

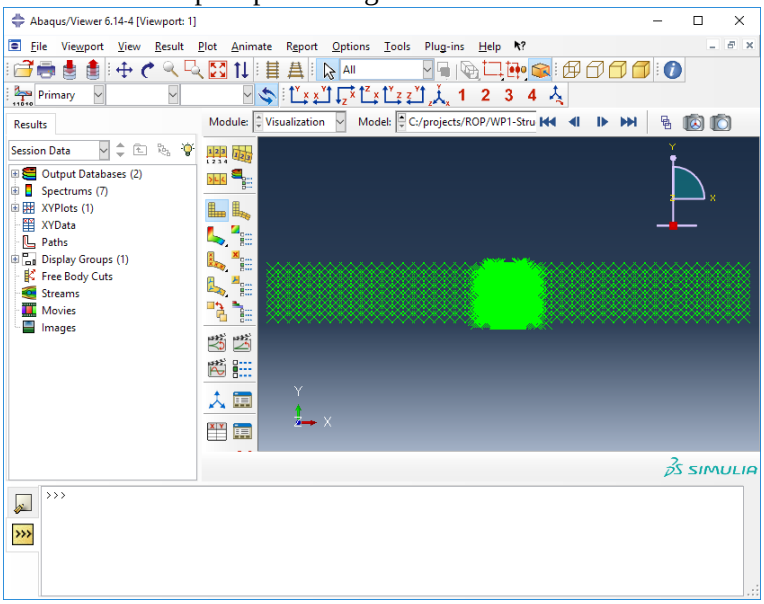


Figure 2: User-defined elements are not visible in Abaqus Viewer. The elements are indicated by their nodes.

One option to get information from the state variables in user-defined elements is to write them to the data output file (dat). This is a text file and the results will be printed in a space-separated table. For instance, this will make Abaqus print the three first state variables as a table in dat file:

```
*El Print, elset=Part-1-1.COH
SDV1, SDV2, SDV3
```

Thus, data can be output for interesting elements of the model. By writing a script, it is then possible to extract the data and to perform processing and visualisation with third party or custom made software. This does not atake advantage of the Abaqus post-processor's capabilities, and in particular, is difficult to use when combing standard and user-defined elements in the same analysis.

In the following, a method for achieving full visualization support in Abaqus with user-defined elements will be described.

2 Solution

2.1 Requirements

This solution is based on using standard Abaqus element as placeholder for the results from the user-defined element. This means that it must be possible to describe the stress field from the user-defined elements with the Gauss quadrature scheme used in some standard Abaqus element. Our own user-defined elements are, for

the time being, using the same integration schemes as standard Abaqus elements and we have been able to reuse the Gauss point values directly without any reinterpolation.

The user-defined elements need to output their results in a persistent way (for instance to a file or a database) during the analysis in a format that is readable for the merge processing later.

2.2 Description

Figure 3 provides a flowchart for the process. Here is a short description of the steps:

- Run the analysis with user-defined elements in the standard way. We will call the odb file from these calculation "the result odb". The user-defined elements should write their results to a separate file, named "<job>-uel.db" in the chart.
- Copy the input file. We will call this copy "the standard input file". Make these changes to the standard input file:
    - Change the user-defined elements to a standard Abaqus element type. These standard elements will be called "standard elements". See the requirements for this element type in section 2.1 Requirements.
    - Replace all the *STEP commands with a dummy *STEP command.
    - Other changes might be necessary to be able to run the standard input file with the standard elements. For instance, a valid *MATERIAL block must probably be linked to the elements.
- Run Abaqus on the standard input file with the option -datacheck to generate an odb file with a model containing the standard elements instead of the user-defined elements.
- Copy the standard odb file from previous step. We will call this copy "the visualization odb file".
- Open the visualization odb file in writable mode. Open the result odb file (containing the real calculation with the user-defined elements). Loop through all steps/increments in this result odb file, create similar steps/increments in the visualization odb and copy all results from the result odb. The results can be given freely chosen names and is not restricted to something like SDV1 etc.

After this, it should be possible to open the visualization odb in Abaqus Viewer with all the elements and their results visible. All post-processing functionality in the viewer should be available for all the results.

An important feature of the solution is that it allows to visualise an arbitrary number of fields (stress components, temperature, pressure, etc…) not limited to the fields available in the standard Abaqus element.

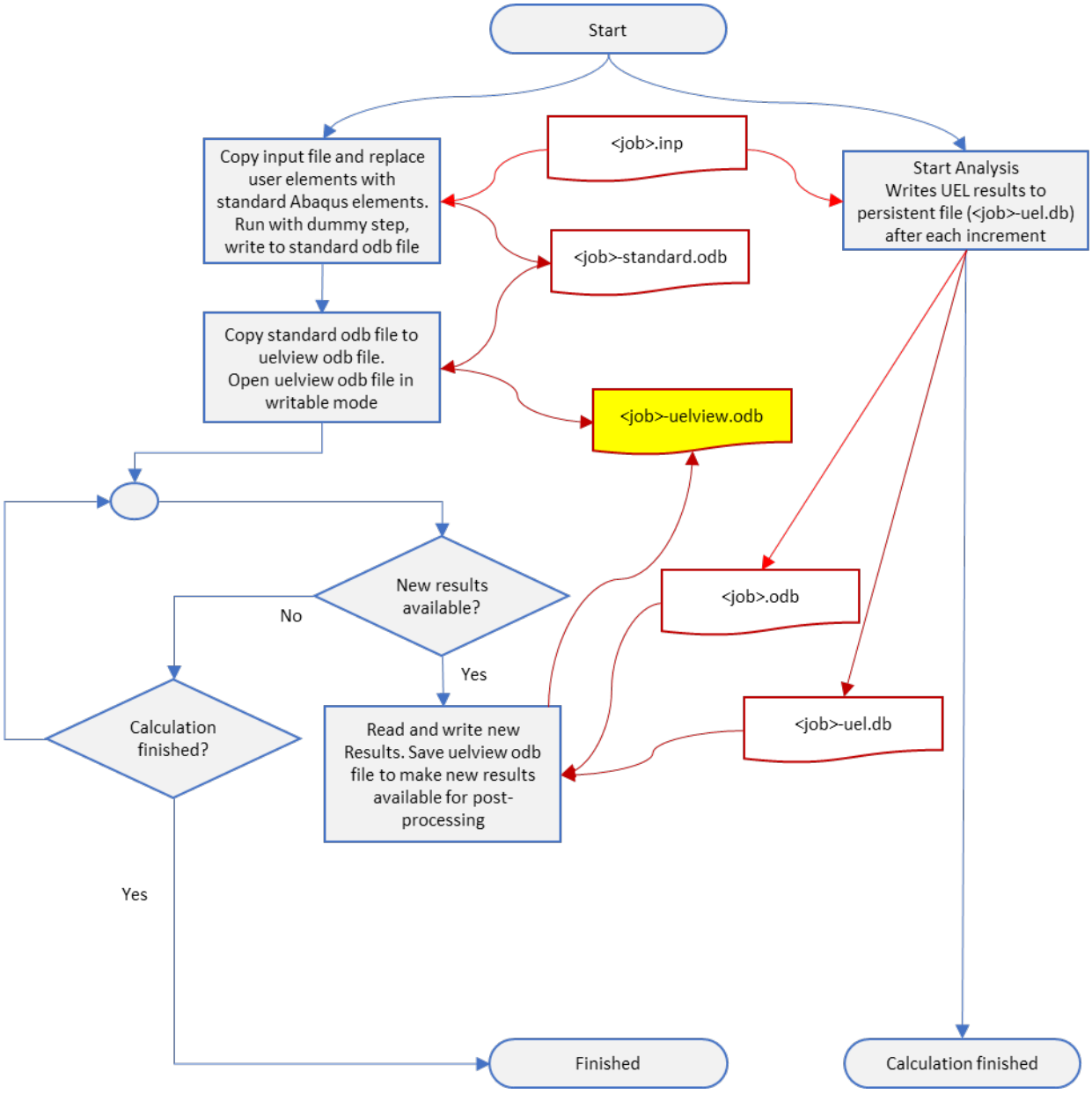


Figure 3 Flowchart for handling of UEL data

Technology for a better society

www.sintef.no