A brief introduction on how to read an *ABAQUS*®input file

Matheus C. Fernandes, Katia Bertoldi, and James R. Rice

October 9, 2015

1. Introduction

When generating a model using the *ABAQUS*®GUI, you create a set of instructions for the interpreter to generate what is called an input file (that has a file extension `.inp`). When executing the model (or submitting the 'Job'), you submit this input file to the solver which interprets how to run the job based on the set of instructions contained in the file.

In this manual, we will go over the different parts of what you will find in the `.inp` file and how you can interpret each part of the code. In particular, this can be helpful for understanding the model that you have generated or when you want to read the input file and generate a model yourself (or use the same model to do an analysis in a different program).

2. Example Description

In this document we will go over a simple input file for a three bar truss system as seen in fig. 1.

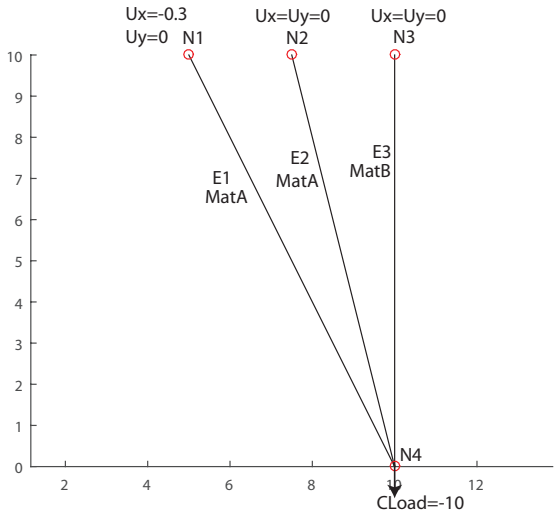

Figure 1: Structure generated from this example input file. The nodes are indicated in red and the line elements in black.

3. General Syntax

The first thing you will notice when you open an `.inp` file is that there are asterisks everywhere in the file. That is because double asterisks `**` (at the left) indicate that the line is a comment. All of the lines that contain comments are ignored by the computer and serve only one purpose – to inform a person reading the `.inp` file. An important note on comments is that you cannot have any empty lines in your input file. If you want an empty line you must make that line a comment by using double asterisks. A single asterisk `*` on the other hand, indicates the beginning of a command, namely `*HEADING` indicates that the line after that will contain the title of the output files created by *ABAQUS*®. Note that the interpreter is not sensitive to capitalization, thus, in general, you do not have to worry about case in input files. The exception is with the use of parameter, (not used in this example), where parameter names are case sensitive.

Furthermore, an important thing to note about *ABAQUS*®is that it does not have a built-in system of units, thus all input data must be specified in a consistent manner. For the example covered in this manual, the units used would be consistent for Steel with forces measured in [lbf] and lengths in [in].

4. More Specific Syntax

4.1 Heading Information

The `*HEADING` command will appear on any output file created by *ABAQUS*®. The heading passed on to the interpreter will then generate output files `.odb` with this title. Therefore, the script seen below would generate an output file named `Three Bar Truss: ES128 Example Problem.odb`. Also note that we have created a large commented section before the actual command to give the reader information about the code. All lines in the code are either commented or contain definition/ values.

```
1  **
2  **
3  ************************************************
4  **
5  ** heading information
6  **
7  ************************************************
8  **
9  *HEADING
10 Three Bar Truss: ES128 Example Problem
11 **
```

4.2 Node Definitions

The node definition gives the location of the nodal points. In this example, these points correspond to the "joints" of the truss system. The first entry is the node number, which is a reference point, and the next entries are the x,y, and z coordinates, respectively. Note that for this 2D problem the z component is omitted. Thus, the order for the example script below is: node number, x coordinate, and y coordinate.

```
**
*NODE
1, 5.0, 10.0
2, 7.5, 10.0
3, 10.0, 10.0
4, 10.0, 0.0
**
```

4.3 Element Definitions

Next, the elements (corresponding to truss bars in this case) are defined. Also, the type of element, from the element library available within *ABAQUS*®, is defined. The first number shown is the element number, and the next two are the two nodes which that element joins. The set of elements is given the name "BARS." Note that this is a 2D element type (T2D2). For different element types you will encounter more or less number of nodes, depending on how many nodes are contained in the given element you specify.

```
**
*ELEMENT, TYPE=T2D2, ELSET=BARS
1, 4, 1
2, 4, 2
3, 4, 3
**
```

4.4 Defining Sets

In *ABAQUS*®we can create sets for different parts of the geometry in order to assign these parts boundary conditions or specific material property. The following example illustrates the creation of two element sets using the `*ELSET` command. The first set is named "MateralA" and is composed of elements 1 and 2. The second element set is named "MaterialB" and only contains element 3.

```
**
*ELSET, ELSET=MATERIALA
1,    2,
*ELSET, ELSET=MATERIALB
3
**
```

4.5 Material Definitions

In order to run the different modules in *ABAQUS*®, you will need to define sets of material properties specific to that model. For the sake of this example, we will focus on running a fully linear elastic problem with a global cross-section area of 0.1, but we will define two different materials for the element sets we created. Here, material A will have a Young's modulus of 1000.0 and material B will have a Young's modulus of 1500.0. Note that we did not define a Poisson ratio, as this is irrelevant to the 1 dimensional problem.

```
**
*SOLID SECTION, ELSET=MATERIALA, MATERIAL=MAT1
0.1
*MATERIAL, NAME=MAT1
*ELASTIC
1000.0
**
*SOLID SECTION, ELSET=MATERIALB, MATERIAL=MAT2
0.1
*MATERIAL, NAME=MAT2
*ELASTIC
1500.0
**
```

4.6 Defining Step

Now we describe the loading, in this case as a single `*STEP`. Since this involves linear elastic material and we are content to neglect any effects of geometry change, due to deformation, on the writing of the equilibrium equations, our problem is a completely linear one. We indicate that `NLGEOM=NO` to indicate that this is a linear step.

```
**
*STEP, NAME=STEP-1,NLGEOM=NO
*STATIC
**
```

4.7 Boundary Conditions

Next, come statements about prescribed displacements. The node number is given first, and then the first and last degree of freedom that is prescribed at the node – displacements UX, UY, and UZ are zero in this case. It was not really necessary to mention UZ, since the problem is set up as 2D. For node 1, however, we prescribe a displacement of -0.3 in the x direction and 0 in the y direction.

```
**
*BOUNDARY
1, 1, 1, -0.3
1, 2, 2, 0.0
2, 1, 2, 0.0
3, 1, 2, 0.0
**
```

4.8 Loads

The following specifies that a concentrated load, `<Load>`, which was defined above as -10000 in the units adopted, is applied in the y direction at node 4 (i.e., 10000 is applied in the negative y direction).

```
**
*CLOAD
4, 2, -10.0
**
```

4.9 Output Requests

Note about output: For large analyses, it is important to consider output requests carefully, as they can create very large files. The following option is used to write output to the output database file `.odb` (for plotting in *ABAQUS*®/Viewer).

```
**
*OUTPUT, FIELD, VARIABLE=PRESELECT
**
```

The following options are used to provide tabular printed output of element in the data file `.dat`.

```
**
*EL PRINT, ELSET=BARS
S, E, COORD
*NODE PRINT
U, COORD
**
```

4.10 End Step

The final statement tells *ABAQUS*®that loading step is over.

```
**
*END STEP
```

Credits

This example was originally created by Prof. James R. Rice as an example problem for the Harvard University ES128 class, and has been modified in this document by Prof. Katia Bertoldi and Matheus C. Fernandes.