Finite Element Method for Solids and Structures
Introduction to ABAQUS

Pankaj Pankaj
School of Engineering
The University of Edinburgh

Contents

1 The UNIX environment **2**

1.1 Logging in to a 'linux' PC 2

1.2 Basic UNIX commands 2

1.3 Directory structure 2

1.4 Available packages 3

1.5 Printing files 4

1.6 Exiting the UNIX window 4

2 More on ABAQUS **4**

2.1 ABAQUS keywords 5

2.2 Running ABAQUS 7

2.3 Post-processing 8

2.4 Plotting figures 8

2.5 Viewing a postscript file 8

3 Training exercise 1: Cantilever problem **8**

3.1 Creating the input file 8

3.2 Analysis of results 9

4 Training exercise 2: Column problem **11**

4.1 Problem description 11

4.2 Tasks to be completed 11

1 The UNIX environment

To use ABAQUS you will be working in the UNIX environment. This is provided by the 'linux' operating system available on the PC. The following subsections introduce the UNIX environment.

1.1 Logging in to a 'linux' PC

The computing lab PCs have 'Redhat linux' installed on them. If the PC you choose is in the Windows environment you will need to restart and choose 'linux'. You will be provided username and password to login. Once you have successfully logged in, you will be in a 'unix' environment. Now right click on the desktop and select 'new terminal'. This will give you a command window to work in.

1.2 Basic UNIX commands

You will need to know a few basic UNIX commands to access and manipulate files and directories.

The following commands will be useful for you to start off with (here *name* is substituted for the name of a file or directory as appropriate):

`ls`	list contents of directory
`ls -al`	long listing of directory, showing file size, date last modified, etc.
`more` *file*	browse through a text *file*
`cp` *file1 file2*	copy *file1* to *file2*
`mv` *file1 file2*	move *file1* to *file2* (changes the file name)
`mkdir` *dir*	create the directory *dir* in the current directory
`cd` *dir*	change working directory to the directory *dir*
`cd ~`	change working directory to your home directory (the directory you end up in when you first log on)
`rm -i` *file*	remove (delete) a *file* with interactive confirmation. **Warning**: files deleted are difficult or impossible to retrieve. If you omit the `-i` option on the `rm` command the *file* will be deleted without interactive confirmation (dangerous!)
`rm -ir` *dir*	remove directory *dir* with interactive confirmation
`pwd`	establish which directory you are currently in

Use `man` *command* to find out more about any particular UNIX command. Note that UNIX is case sensitive. The symbol * can be used as a wildcard (e.g. `abs*` refers to all files where the filename begins with the letters `abs`).

1.3 Directory structure

Creating a clear directory structure (or folder structure) will make keeping track of files a lot easier for you in the long run, especially if you continue to order your files sensibly. The directories should be ordered like a tree with many branches, where each branch evolves from a larger branch in a sensible manner (see below). Make use of the fact that you can use as many characters as you want to give a clear description of the contents of a file or directory (typing long names in commands is easy as you can use the tab key to complete command or filenames after typing the first few letters).

Every time you log on you will immediately be placed in your home directory `/home/`*username* (the "tree trunk"). The home directory can be abbreviated with the symbol ~ for purposes such as moving between directories or creating files. When you log in for the first time there will be no subdirectories under your home directory (the fattest "branches" in Figure 1).

A typical example of a sensible directory structure is shown in Figure 1.

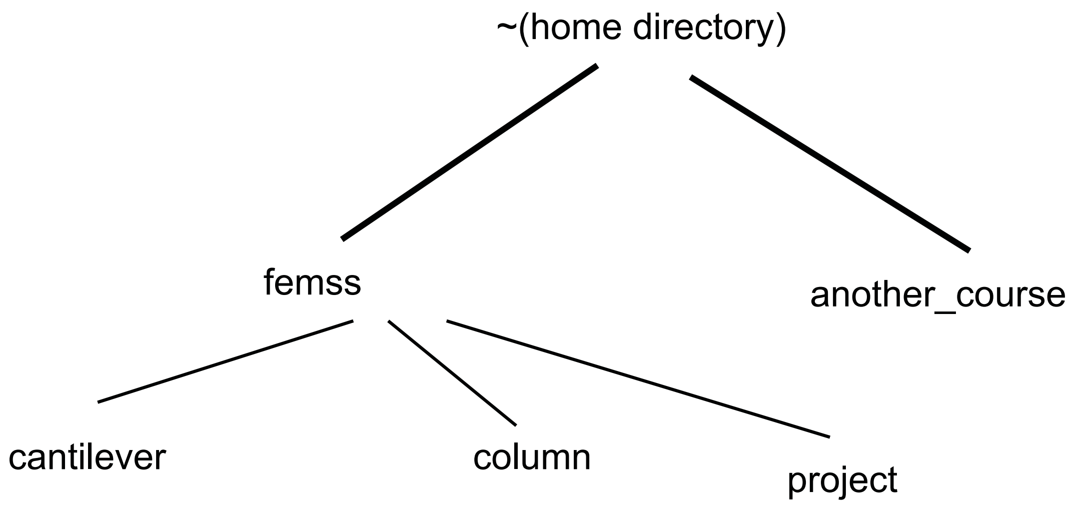

Figure 1: Directory tree structure. All directory and file names are case sensitive

In the example, two subdirectories have been created under the home directory: `femss` and `another_course`. All files relating to ABAQUS may be stored in the `femss` subdirectory. However, further subdivisions are advisable so that individual files can be found easily. Thus a sub-subdirectory has been created for each problem you will be working on. The complete path (which defines some of the characteristics of a file) for the file `test1.inp` in the `cantilever` sub-subdirectory would be

```
/home/civil#/femss/cantilever/test1.inp
```

The path can be obtained by using the `pwd` command in the appropriate directory. The abbreviated path would be

```
~/femss/cantilever/test1.inp
```

1.4 Available packages

The list below gives an outline of some of the packages available to you in linux. It is not exhaustive! You will, for example, need to use the text editor `emacs` for creating and editing the input files used to run ABAQUS (there are other text editors such as `gedit` and `kedit` and `pico`, however `emacs` is the simplest). It is also good practice to save each new version of a file, i.e. when you make any significant changes to the file, under a new filename. Thus if you are developing an ABAQUS input file, for example, the first version might be called `cantil1.inp`, the second version `cantil2.inp`, and so on. You can then use old versions of the input file at any stage if required.

1. `emacs`: Type `emacs` *filename* in the command window. For example if you want to create a file called `test1.inp` type `emacs test1.inp`. If `test1.inp` already exists then the file will

become available for editing, else a new file called `test1.inp` will be created. You can now type or edit and save the file.

2. `pico`: This can also be used to to create or edit files. Commands to be used within pico are shown at the bottom of the screen. Type ^ (the ctrl character) together with a capital letter to perform a particular command, e.g. ^X to exit *pico* and save the file currently being edited. Make sure you save your text file regularly in case the system suddenly crashes and you loose any changes you have made to the file (^O).

3. `ggv`: is the command for invoking the package 'ghostscript' used to view postscript figures. You can store the final version of all your ABAQUS figures in postscript files (a graphics language). You can use ghostscript to check what a figure (e.g. `fig1.ps`) would look like if you printed it, by typing

   ```
   ggv fig1.ps
   ```

1.5 Printing files

The best way to print files from a linuxronment is by using the `lpr` command. For example to print the created file `trial1.inp` use the command
`lpr -P pstlg trial1.inp`
from the command window. Here `pstlg` is the name of the printer in TLG lab. If you are in another lab, a sticker on the printer will give you the name. Postscript files (that you will use for figures) can be printed in the same way. A number of other binary files are created by ABAQUS. Do not try to print these.

1.6 Exiting the UNIX window

It is very important to ensure that you log off when you finish working as your work can be accessed by others if you don't. You may also be blocking a machine that may be needed by someone else. Stop all jobs you have started (by issuing commands or clicking on icons). To exit click the 'redhat' icon and choose logoff.

2 More on ABAQUS

ABAQUS is a powerful package capable of solving a large variety of problems using the finite element method. Details of what all ABAQUS is capable of can be found in the manuals available in the lab. Online help is available from: `http://www.see.ed.ac.uk/it/online/abaqus-681/v6.8/index.html`

The ABAQUS input file (for example, like one discussed in the lectures) is a means of communicating a correctly defined problem to the ABAQUS package. It contains a complete description of the numerical model, and can be edited using a simple text editor (e.g. `emacs`).

The input file is composed of a number of option blocks that contain data describing the model. Each option block begins with a **keyword** line. This usually consists of a keyword and several either optional or requisite parameters separated by a comma. Keywords (the ABAQUS input file commands) always begin with an asterisk (∗). Often the option block also contains some data lines, defining comma separated data used in connection with the keyword. The complete ABAQUS input file may be split into two distinct parts. The first section contains model data and includes all the information required to define the structure being analysed. The second section contains history data that define what happens to the model: the sequence of loading for which the response of the structure is required. The history may consist of several steps, each defining a separate part of the simulation.

Before you begin defining a model, you need to decide on a consistent system of **units**. One possible consistent set of units is given in Table 1.

Quantity	**SI**
Length	m
Force	N
Mass	kg
Time	s
Stress	Pa (N/m^2)
Density	kg/m^3

Table 1: Consistent units

You also need to decide which **coordinate system** to use. The default coordinate system in ABAQUS is a right-handed, rectangular (Cartesian) system.

2.1 ABAQUS keywords

This section gives a brief overview of many of the most important keywords used in ABAQUS input files. The parameters shown are either optional (o) or requisite (r). A default value or name is shown in square brackets [ ]. Vertical dots (⋮) indicate that the data line should be repeated as often as necessary.

Keyword *Data lines*	**Parameter**	**Description**
**		All following text on this line is treated as a comment
***BOUNDARY**		Specify boundary conditions
First line		Node number or node set
		First degree of freedom constrained
		Last degree of freedom constrained
		Magnitude of displacement applied to node/node set
		(Only for non-zero boundary conditions specified as history data)
⋮		
***CLOAD**		Specify concentrated forces and moments
First line		Node number or node set
		Degree of freedom
		Magnitude of load
⋮		
***EL PRINT**		Define data file (.dat) requests for element variables
	ELSET (o)	Name of element set. If parameter omitted, output all elements
	FREQUENCY (o)	Set FREQUENCY=0 to suppress the output
	POSITION (o)	Set POSITION=INTEGRATION POINT [default] or POSITION=AVERAGED AT NODES
	TOTALS (o)	Set TOTALS=YES to print the total of each column in the table
First line		Identifying key for variables to be printed

		in a table for this element set
⋮		
∗**ELASTIC**		Specify elastic material properties
First line		Young’s modulus
		Poisson’s ratio
∗**ELEMENT**		Define elements by giving their nodes
	TYPE (r)	Element type
	ELSET (o)	Name of element set
	INPUT (o)	Name of file containing data lines
First line		Element number
		First node number forming the element
		Second node number forming the element
		Etc., up to 15 node numbers on this line
⋮		
∗**ELGEN**		Generate incremental elements
	ELSET (o)	Name of element set
First line		Master element number
		Number of elements to be defined in first row generated
		Increment in node numbers of corresponding nodes from element to element in the row [1]
		Increment in element numbers in the row [1]
		Number of rows to be defined [1]
		Increment in node numbers of corresponding nodes from row to row
		Increment in element numbers of corresponding elements from row to row
⋮		
∗**END STEP**		End the definition of a step
∗**HEADING**		Print a heading on the output
First line		The heading
∗**MATERIAL**		Begin the definition of a material
	NAME (r)	Name of material (referenced in element property option)
∗**NFILL**		Fill in nodes in a region between two bounds
	BIAS (o)	Ratio of adjacent distances between nodes along line (Nodes concentrated towards first node set for BIAS $<$ 1)
	NSET (o)	Name of node set
First line		Name of node set defining first bound of region
		Name of node set defining second bound of region
		Number of intervals between bounding nodes
		Increment in node numbers from the first bound set [1]
⋮		
∗**NGEN**		Generate incremental nodes
	NSET (o)	Name of node set
First line		Number of first end node
		Number of second end node
		Increment in numbers between each node along line [1]
⋮		

***NODE**		Specify nodal coordinates
	INPUT (o)	Name of file containing data lines
	NSET (o)	Name of node set
First line		Node number
		First coordinate
		Second coordinate
		Third coordinate
⋮		
***NODE PRINT**		Define data file (.dat) requests for nodal data
	FREQUENCY (o)	Set FREQUENCY=0 to suppress the output
	NSET (o)	Name of node set. If parameter omitted, output all nodes
	TOTALS (o)	Set TOTALS=YES to print the total of each column in the table
First line		Identifying key for variables to be printed in a table for this node set
⋮		
***SOLID SECTION**		Specify element properties for solid elements
	ELSET (r)	Name of element set for which mat. behaviour is defined
	MATERIAL (r)	Name of defined material
First line		Thickness of element [1]
***STATIC**		Static stress/displacement analysis
***STEP**		Begin a step
	PERTURBATION (o)	Include this parameter to indicate a linear perturbation step

2.2 Running ABAQUS

Once you have created the input file you can execute the ABAQUS program. If your input file is called `trial1.inp` the basic analysis program and is executed by typing:

```
abaqus job=trial1 interactive
```

During the analysis, several files are created, for example:

- `trial1.023`: A temporary communications file that is deleted when the analysis is completed
- `trial1.msg`: File containing potential error and warning messages
- `trial1.dat`: Printed output file (data determined by the print options in the input file)
- `trial1.com`: Internal commands executed
- `trial1.odb`: Binary file containing all data for ABAQUS/Viewer (post processing package)
- `trial1.sta`: Status file containing step summaries

Some of these files are normal text files (e.g. `trial1.dat`, `trial1.msg`, `trial1.sta`) and can be viewed using any text editor (e.g. `pico`). The binary files created by ABAQUS (e.g. `trial1.odb`) cannot be viewed using a text editor (do not try and print such files). The analysis usually runs as a background process completed in two stages. The preprocessor within ABAQUS reads the input file, rechecks the data, and sets up the mesh. Unless any problems are detected, the solver then solves the numerical problem defined in the input file.

2.3 Post-processing

After running ABAQUS using file `trial1.inp` you will have created a file called `trial1.dat`. This file contains the results in numerical form. You can open this file using an editor to check if the results (displacements, reactions etc.) are as you would expect. If there is a problem with your input file then the `dat` file will also indicate where the error is likely to be.

If your program runs successfully a file with extension `odb` will also be created. This file is in binary format, so that you cannot view the file using a normal text editor. You can view the results from this file in a graphical form by using the postprocessor, ABAQUS/Cae. type

```
abaqus cae
```

in the command window and a new window will appear. Choose 'open database' and then in the file filter box at the bottom select 'output database (*.odb)'. You now choose the `odb` file you have created. Abaqus/Cae can be used to display the mesh, the deformed mesh, contour plots etc.

2.4 Plotting figures

You can save any figures or graphs that you create in viewer by choosing 'to file' on the print dialog box and choosing a name for the file to save (always use a `.ps` extension for 'postscript' files, `.eps` for 'encapsulated postscript' files and `.tiff` for 'tiff' files). These files can be printed or copied into reports.

2.5 Viewing a postscript file

Use the 'ghostscript' program `ggv` to check that your postscript (*name*`.ps` and *name*`.eps`) figures look the way you want them to be printed out. To see the file `trial1.ps` type

```
ggv trial1.ps
```

3 Training exercise 1: Cantilever problem

3.1 Creating the input file

1. Log in to the linux PC (see Section 1.1) and create your own directory structure. You may wish to follow the example provided in Section 1.3 or adopt something similar. Start by creating the subdirectories (see `mkdir` command in Section 1.2) required in your home directory. You can always add to these at a later stage. Then move into one of these subdirectories to create further subdivisions and proceed along the branches of your tree. It might be helpful for you to make a quick sketch such as Figure 1, indicating how you wish to structure your directories and files, before implementing such a structure.

2. At an appropriate location in your directory create the ABAQUS input file for solving the cantilever problem discussed in the lecture. You may, for example, create a file in the sub-subdirectory called `cantilever`. You will need to move in to this directory using the `cd` command (see Section 1.2) and then use a text editor, e.g. `emacs`, discussed in Section 1.4. The input file is given below:

   ```
   *HEADING
   ```

```
 CANTILEVER WITH CONTINUUM ELEMENTS--END SHEAR, CPS4I, 4 X 4 MESH
*NODE
1,0.,0.
17,6.,0.
401,0.,.2
417,6.,.2
*NGEN,NSET=FIX
1,401,100
*NGEN,NSET=END
17,417,100
*NFILL
FIX,END,4,4
*ELEMENT,TYPE=CPS4I
1,1,5,105,101
*ELGEN,ELSET=EALL
1,4,4,1,4,100,4
*MATERIAL,NAME=A1
*ELASTIC
1.E7,0.
*SOLID SECTION,MATERIAL=A1,ELSET=EALL
.1
*PREPRINT, ECHO=YES, MODEL=YES, HISTORY=YES
*STEP,PERTURBATION
*STATIC
*BOUNDARY
FIX,1,2
*CLOAD
17,2,-.5
417,2,-.5
*EL PRINT
COORD
S
E
*EL PRINT, POSITION=AVERAGED AT NODES
S
*NODE PRINT
U
RF
*END STEP
```

3. Make sure you have typed the input file correctly and you understand it well (see your lecture notes). Now you are ready for analysis. Run ABAQUS using the command discussed in Section 2.2. Use the text editor to look at the *name*`.dat` and *name*`.msg` files created. If there are mistakes in the *name*`.inp` file the *name*`.msg` and the *name*`.dat` files will indicate errors. Correct any errors and run ABAQUS again.

3.2 Analysis of results

The following tasks should be performed after successfully running ABAQUS on the cantilever problem. You will require your *name*`.dat` file for tasks A–D and *name*`.odb` file for task E. The `.dat` file can be viewed using a text editor and the `.odb` file is used with ABAQUS/Cae.

(A) *Preliminary checks*

(a) Sketch the mesh and write down the element and node numbering from the output file.

(b) Check that the material properties are as expected and note these below your sketch.

(c) Check that the boundary conditions have been applied to the correct nodes and that the restraints (degree of fixity) are as intended. Mark the nodes (and degrees of freedom) restrained on the above sketch.

(d) Check the loading: is it on the correct elements and nodes? Is the value of the applied load correct? Mark the loading as applied on your sketch.

(B) *Displacements*

(a) Calculate the vertical movement at the free end $\frac{PL^3}{3EI}$ that you would have expected from beam bending. Compare it with the corresponding nodal displacements in the output.

(b) Look at the horizontal displacements at the free end and explain these? Consider the slope $\frac{PL^2}{2EI}$ at the end of a cantilever (carrying an end force). Are the horizontal displacements about the right magnitude for your explanation?

(C) *Reactions*

(a) Look at the reactions at the restrained nodes. Is equilibrium satisfied? Do the vertical reactions add up to the total load P?

(b) Why do you think horizontal reactions occur? Do these add up to what you would expect? Is horizontal equilibrium satisfied?

(c) Calculate the bending moment which these forces represent? Does it match the value you get from simple statics on the cantilever?

(D) *Stresses*

(a) Look at the stresses half way along the cantilever. Do the horizontal normal stresses match the value you would calculate from bending theory – plot the horizontal stress across this section? Using simple approximations and averages show that values that have been averaged at the nodes match up with those given at the integration points.

(b) Plot the shear stress τ_{xy} half way along the cantilever. Does the shear stress distribution match that of simple bending theory? Check whether the maximum value is about right ($\tau_{max} = 1.5\tau_{average}$)?

(c) Find the stresses in element 1 at integration point 1 using strains at the same point. Do the evaluated stresses match those in the `.dat` file?

(E) *Post-processing using ABAQUS/Cae*

(a) Draw a deformed mesh superimposed on an undeformed mesh. Comment on the shape – is the shape what you would expect for bending?

(b) Create a filled contour plot of the horizontal normal stresses S11. Comment on their variation from from top to bottom at the fixed end and at the midsection. How do these values compare with values in the `dat` file?

(c) Create a filled contour plot of the horizontal shear stresses S12. Comment on their variation from from top to bottom and along the length of the cantilever. How do these values compare with values in the `dat` file?

(d) Plot an x-y graph showing horizontal normal stress across the beam at $x = 3$m. How does this plot compare with your previously plotted results?

(e) Plot the shear stress across the beam at $x = 3$m. How does this plot compare with your previously plotted results?

4 Training exercise 2: Column problem

Use another subdirectory for this exercise (e.g. column). You are required to analyse a short wide column in compression.

4.1 Problem description

A short wide concrete column 1600 mm high, 400 mm wide, and 200 mm thick, is subjected to a uniformly distributed axial load at the top. The concrete has a modulus of elasticity (E) of 2.0×10^4 MPa (N/mm^2) and a Poisson's ratio (ν) of 0.25.

Analyse the column using a simple mesh four elements wide and ten elements high. Choose a 4 noded elements. Fix the five nodes at the bottom and apply a uniform downward load of 10.0 MPa to the top of the column.

Use a text editor to create your input file. Then run ABAQUS. Finally, examine the *name*`.dat` and *name*`.odb` files to find out how the column responded.

4.2 Tasks to be completed

Perform the following tasks for the column described above. All tasks should be included in your submission.

(A) *Pre-analysis tasks*

(a) Sketch the column and include the element and node numbers in your sketch.

(b) Create an input file to analyse the problem.

(B) *Post-analysis tasks*

(a) Look at the reactions at the restrained nodes. Do the axial reactions add up to the total load you had applied?

(b) Calculate the axial displacement at the top of the column that you would have expected from simple bar shortening $\frac{PL}{EA}$? Compare it with the corresponding nodal displacements in the output.

(c) Look at the transverse displacements at the top. Can you explain these? Are they about the right magnitude for your explanation?

(d) Plot the deformed shape of the column. Explain why there are transverse reactions at the base of the column and why the column appears "squeezed". (For a clue, look at the deflected shape).

(e) Examine axial displacements half way down. Show through calculations that they are what you would expect.

(f) Examine stresses away from loading and supports. Do the stresses match the value you had applied?

(g) Look at the stresses near the support. Why is there significant transverse stress near the support?

(C) *Modifications*

(a) Repeat the analysis with only one of the bottom nodes restrained against transverse translation. Highlight the key differences in reactions, deformation and stresses in comparison to the previous exercise and comment on them. Plot a deformed mesh superimposed on an undeformed mesh to answer the question. Look at the vertical displacement at the top and half way down. Are the values closer to the theoretical values than before?