A brief introduction to the use of Abaqus: the input file

Course of Spacecraft Structures

AY 2017/2018

Riccardo Vescovini

Politecnico di Milano, Department of Aerospace Science and Technology

The ingredients of the finite element model

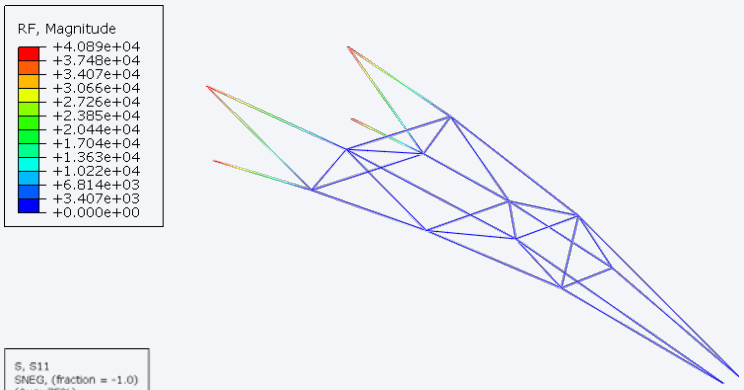
- Discretized geometry
  - finite elements and nodes
  - different elements available: beam, thin/thick shell, continuum shell, brick,…

  → the collection of elements and nodes defines the mesh

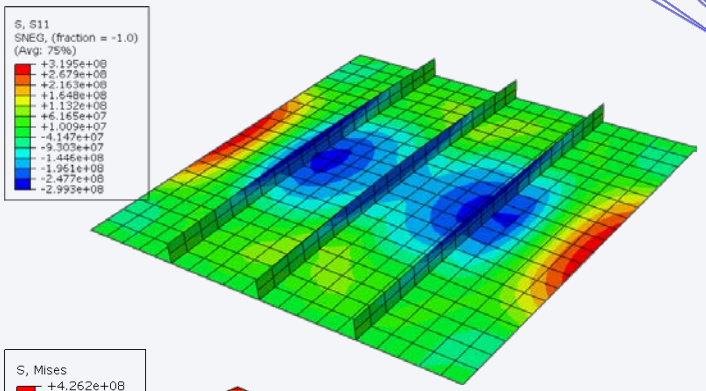
- Element section properties
  - for many elements, the geometry is not completely defined by the coordinates of the nodes (e.g. the dimension of a beam section)

- Material data
  - isotropic, anisotropic,…

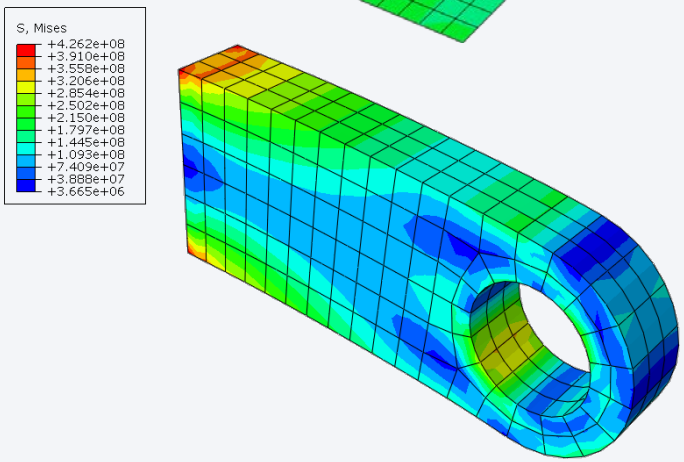
The ingredients of the finite element model



Example: model of a cargo crane using truss elements



Example: model of a stiffened plate using shell elements



Example: model of a lug using continuum elements

The ingredients of the finite element model

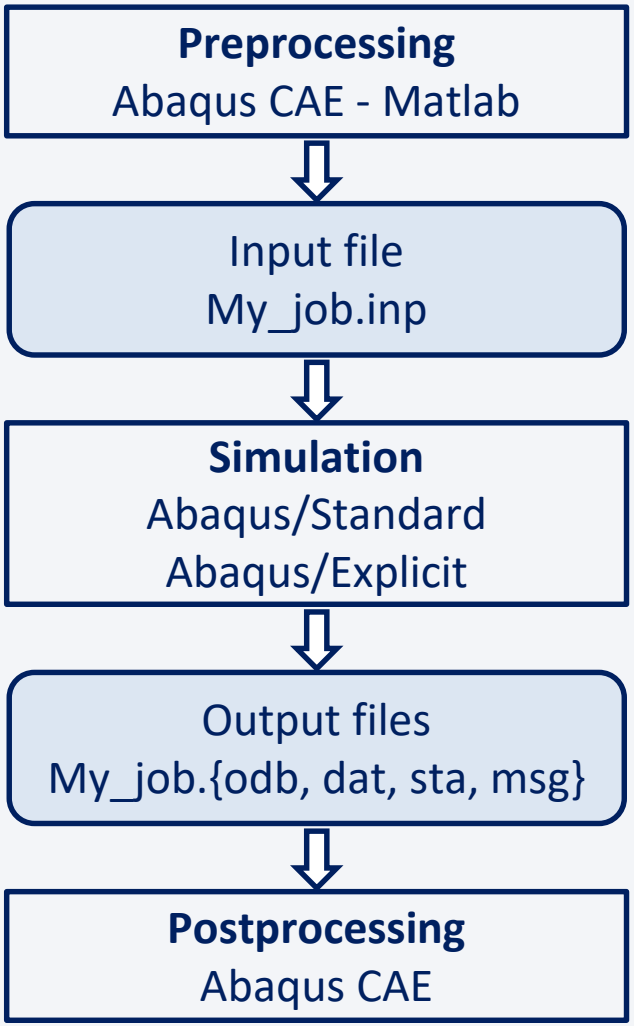
- Loads and boundary conditions
  - point loads, pressure loads, body forces, thermal loads
  - concentrated nodes applied in correspondence of a node
  - distributed loads should be reported to the nodes
  - BCs should prevent unrestrained rigid body motion (check error/warning message)

- Analysis type
  - different type of simulations available: static, eigenvalue, dynamic,...
  - is a linear approach appropriate?
  - are nonlinearities (geometric and/or material) present?
  - are dynamic effects important?

The ingredients of a finite element model

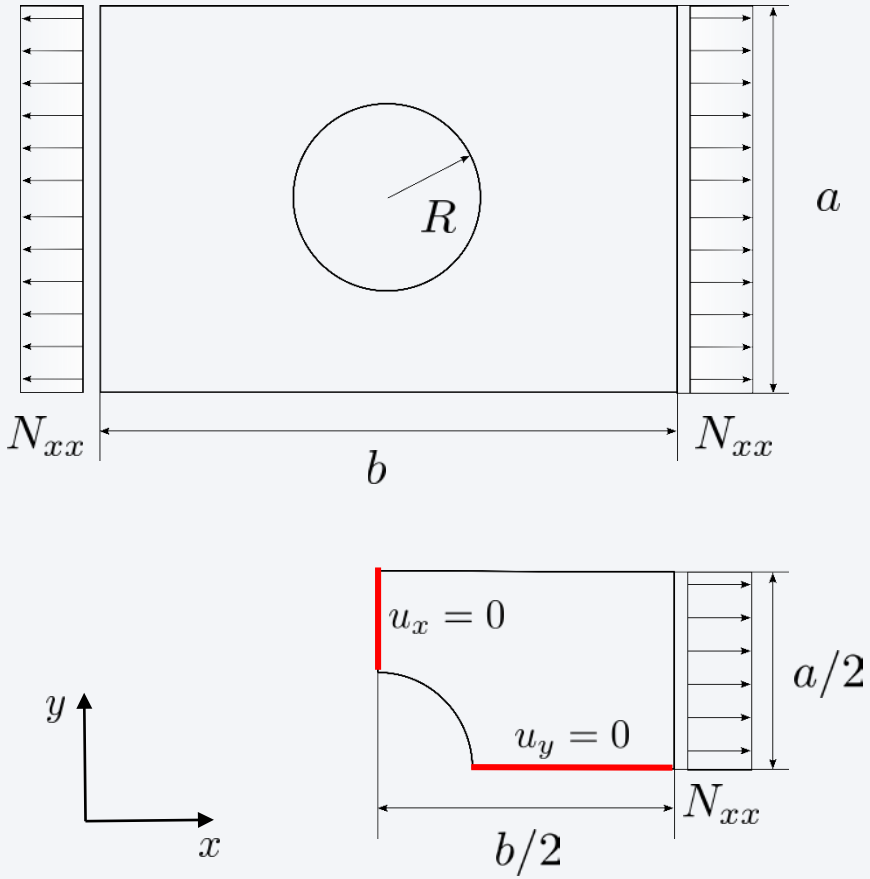
- Output
  - displacements, stresses, reaction forces,...
  - in a limited set of nodes/element or in the overall model
  - with different frequencies (for multi-step analysis)
  - in binary and/or ASCII files

Steps of the procedure



Example

- Open-hole test: membrane with center hole, loaded in traction (1/4 of the structure due to the double symmetry of the problem)



Input data

a = 100 mm
b = 150 mm
R = 25 mm
t = 1 mm
E = 72 GPa
v = 0.3
N_{xx} = 20 N/mm

The input file (generated by the CAE)

- Consider now the input file generated with the graphical interface after selecting "Write input" in the job creation phase. The content of the .inp file represents the one and only set of data which is used by the Abaqus for solving the problem

- With this regard, the input file can be created with any kind of pre-processor (not necessarily the Abaqus one), including the generation of the .inp file via script (for instance, in Matlab)

The input file (generated by the CAE)

- The content of the input file generated during the pre-processing phase can be examined by opening the .inp file with a text editor
- In Abaqus the symbol * denotes a keyword (Abaqus command); ** is a comment

```
*Heading
** Job name: run_01 Model name: Model-1
** Generated by: Abaqus/CAE 2016
*Preprint, echo=NO, model=NO, history=NO,
contact=NO
**
** PARTS
**
*Part, name=plate_with_hole
*Node
      1,   17.6776676,   17.6776714
      2,   49.9999924,          50.
      3,           0.,          50.
      4,           0.,          25.
      5,   49.9999924,           0.
      6,          25.,           0.
      7,          75.,           0.
```

Keyword *Part (alle the keyword below are referred to this part). When the file is written by hand, this keyword is not necessary

Keyword *node (all the entries here below are quantities referred to the node definition)

The syntax, in this case, is intuitive:
Id of the node, x-position, y-position.
(the z-position is not reported for a planar analysis, but in general wold be the fourth entry)

The input file (generated by the CAE)

- The content of the input file generated during the pre-processing phase can be examined by opening the .inp file with a text editor
- In Abaqus the symbol * denotes a keyword (Abaqus command); ** is a comment

```
*Heading
** Job name: run_01 Model name: Model-1
** Generated by: Abaqus/CAE 2016
*Preprint, echo=NO, model=NO, history=NO,
contact=NO
**
** PARTS
**
*Part, name=plate_with_hole
*Node
      1,   17.6776676,   17.6776714
      2,   49.9999924,          50.
      3,           0.,          50.
      4,           0.,          25.
      5,   49.9999924,           0.
      6,          25.,           0.
      .
      .
      .
```

note that the ID is an integer. A real number would cause an error!

The input file (generated by the CAE)

```
*Element, type=CPS4
 1,  1,  9, 37, 20
 2,  9, 10, 38, 37
 3, 10, 11, 39, 38
 4, 11,  2, 12, 39
 5, 20, 37, 40, 19
 6, 37, 38, 41, 40
 7, 38, 39, 42, 41
 8, 39, 12, 13, 42
 9, 19, 40, 43, 18
.
.
.
```

Keyword *element; defines the type of element with the parameter «type» (in this case CPS4), and the nodes composing the elements

Element 8, composed by nodes 39, 12, 13 and 42

The input file (generated by the CAE)

```
*Nset, nset=Set-1, generate
  1,  60,   1
*Elset, elset=Set-1, generate
  1,  44,   1
** Section: Section-1
*Solid Section, elset=Set-1,
material=Material-1
1.,
*End Part
```

keyword *Nset: create a set of nodes whose name is Set-1

keyword *Elset: create a set of elements whose name is Set-1

keyword *Solid Section: specifies the section properties. In this case this is done by specifying the name of the material in the parameter material=, and the thickness of the membrane in the first line

*end of the part

The input file (generated by the CAE)

```
*Assembly, name=Assembly
**
*Instance, name=plate_with_hole-1,
part=plate_with_hole
*End Instance

**
*Nset, nset=Set-1, instance=plate_with_hole-1
  3,  4, 15, 16, 17
*Nset, nset=Set-2, instance=plate_with_hole-1
  5,  6,  7, 27, 28, 29, 30, 31
*Elset, elset=_Surf-1_S3, internal,
instance=plate_with_hole-1, generate
 41,  44,   1
*Surface, type=ELEMENT, name=Surf-1
_Surf-1_S3, S3
*End Assembly
```

→ Define the assembly (=collection of parts)

→ One and only part composing the assembly

→ Specify the set of nodes Set-1

→ Specify the set of nodes Set-2

→ Specify the information (set of elements and surface) for applying the distributed load.

→ Terminate assembly definition

The input file (generated by the CAE)

```
*Assembly, name=Assembly
**
*Instance, name=plate_with_hole-1,
part=plate_with_hole
*End Instance

**
*Nset, nset=Set-1, instance=plate_with_hole-1
  3,  4, 15, 16, 17
*Nset, nset=Set-2, instance=plate_with_hole-1
  5,  6,  7, 27, 28, 29, 30, 31
*Elset, elset=_Surf-1_S3, internal,
instance=plate_with_hole-1, generate
 41,  44,   1
*Surface, type=ELEMENT, name=Surf-1
_Surf-1_S3, S3
*End Assembly
```

Note that in this case, and for any model which is not composed by several parts, no practical advantages exist in creating a model as an assembly of part instances.

When preparing the input file by script, these commands could be removed. They are generated by the Abaqus CAE, as this is the way Abaqus organizes the model.

Interlude: assemblies and parts

Assembly

An assembly is a collection of positioned part instances. An analysis is conducted by defining boundary conditions, constraints, interactions, and a loading history for the assembly.

Part

A part is a finite element idealization of an object. Parts are the building blocks of an assembly and can be either rigid or deformable. Parts are reusable; they can be instanced multiple times in the assembly. Parts are not analyzed directly; a part is like a blueprint for its instances.

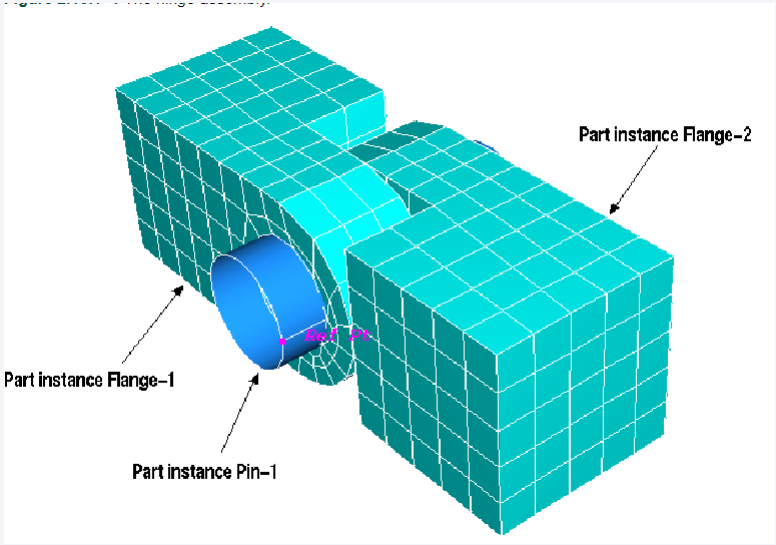
Part instance

A part instance is a usage of a part within the assembly. All characteristics (such as mesh and section definitions) defined for a part become characteristics for each instance of that part—they are inherited by the part instances. Each part instance is positioned independently within the assembly.

Interlude: assemblies and parts

- In this case the advantages of building the assembly as a collection of parts is clear. The Abaqus code would be:

```
*ASSEMBLY, NAME=Hinge
   *INSTANCE, NAME=Flange-1, PART=Flange
    <positioning data>
   *END INSTANCE
   *INSTANCE, NAME=Flange-2, PART=Flange
    <positioning data>
   *END INSTANCE
   *INSTANCE, NAME=Pin-1, PART=Pin
    <positioning data>
   *END INSTANCE
   *ELSET, ELSET=Top
    ...
   *NSET, NSET=Output
    ...
*END ASSEMBLY
```



The mesh of the flange is used twice

The input file (generated by the CAE)

```
**
** MATERIALS
**
*Material, name=Material-1
*Elastic
72000., 0.3
** ----------------------------------------------
------------------
**
** STEP: Step-1
**
*Step, name=Step-1, nlgeom=NO, perturbation
*Static
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Displacement/Rotation
*Boundary
Set-1, 1, 1
** Name: BC-2 Type: Displacement/Rotation
*Boundary
Set-2, 2, 2
```

Define a linear elastic material
(Young's modulus and Poisson's ratio)

Specify the solution procedure
*Step: specify the analysis step
*Static: static analysis (linear)

Set the boundary conditions:
the node Set-1 is constrained from 1 (x-direction) to 1 (x-direction)

The input file (generated by the CAE)

```
**
** LOADS
**
** Name: Load-1   Type: Surface traction
*Dsload, follower=NO, constant resultant=YES
Surf-1, TRVEC, 20., 1., 0., 0.
**
** OUTPUT REQUESTS
**
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=PRESELECT
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=PRESELECT
*End Step
```

→ Specify the distributed load (see manual for additional information)

→ Output request (post-processing).
The option preselect is intended to require a set of default output (displacement, stresses,...)

The input file: a simple template (1/2)

- In many cases, unless the model is relatively complex, the input file created by Abaqus can be simplified by removing parts, instances and templates

```
*node, nset = my_nodes
1,   17.6776676,   17.6776714

*element, type = CPS4, elset = my_els
1,  1,  9, 37, 20

*material, name = my_material
*elastic
72000., 0.3

*solid section, elset = Set-1, material = my_material
1.,

*nset, nset = edge_1
3,  4, 15, 16, 17

*step
*static

*boundary
edge_1, 1, 1

*cload
```

: arbitrary names

The input file: a simple template (2/2)

- The simpler way for requesting the output is:

```
*output, field, variable = preselect
*end step
```

→ The parameter field specifies that the output has to be written on the binary .odb file

- In many cases, it is necessary to specify the kind of output, and the where the output has to be written. Thus, the following commands can be used

```
*output, field
*element output, elset = eglobal
s, e
*node output, nset = nglobal
u,
*el print, elset = eglobal
s, e
*node print, elset = nglobal
u
```

→ *element output: element quantities to be written to the .odb file

→ *el print: element quantities to be written to the ASCII .dat file
s,e: stresses and strains

The output files

- .odb: binary file containing the results (to be opened with Abaqus CAE or Abaqus Viewer)
- .dat: information about the model definition; tabular output of results
- .msg: diagnostic messages about the progress of the solution
- .sta: status of the solution process (can be monitored when the solution is running)

Remarks: the input file

- Defining the type of element in the *element keyword is mandatory. Some examples of commonly used elements:

  - T3D2: truss in 3D with 2 nodes
  - B33: beam in 3D and cubic shape functions (Euler-Bernoulli)
  - B31: beam in 3D and linear shape functions (Timoshenko)

  - S4: shell with four nodes
  - S4R: shell with four nodes and reduced integration

  - C3D8: solid element with 8 nodes
  - C3D20: solid element with 20 nodes

Remarks: the input file

- Each element of the model has to be associated to a section definition. The definition of the section is different depending on the kind element adopted:
  - Truss → *solid section
  - Beam → *beam section
  - Shell → *shell section
  - Continuum element → *solid section

Remarks: running the analysis

- After generating the .inp file with a text editor (Notepad, Crimson Editor, ...), save it into a folder
- Open the dos command prompt and move to the folder of .inp file

- A preliminary check is suggested to verify the model
  - `abaqus j=job_name datacheck interactive`
  - warning/errors messages are written in the .dat file

- Running the analysis
  - `abaqus j=job_name interactive continue`
  - `abaqus j=job_name interactive` (if datacheck is not performed)

- Analysis of the results
  - ASCII file in the .dat file
  - graphical post-processing of the .odb using Abaqus viewer: `abaqus viewer`

Remarks: generating the input file from Matlab

- In Matlab, the fprintf function can be used to write the Abaqus input file. Recall the syntax:

fprintf(fileID, formatSpec, A1, ..., An)

  - fileID: file identifier
  - formatSpec: format of the output field

  - A1,...,An: numeric or character arrays

Remarks: generating the input file from Matlab

- Assume the matrix of node IDs and coordinates has been computed

```
nodes =
    1.0000         0         0         0
    2.0000   10.0000         0         0
    3.0000   20.0000         0         0
    4.0000         0    2.5000         0
    5.0000   10.0000    2.5000         0
    6.0000   20.0000    2.5000         0
```

- The Abaqus input file can be written using the following Matlab code:

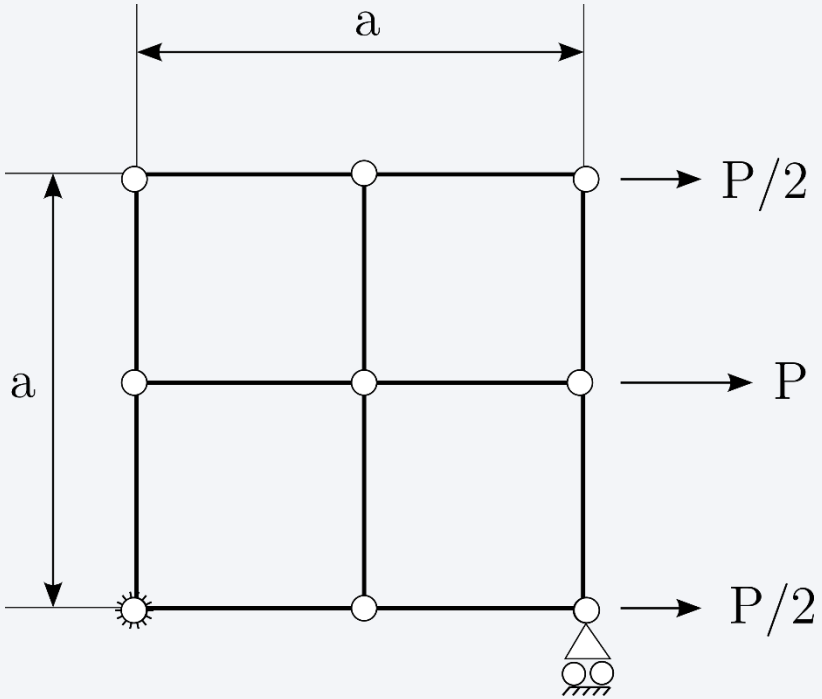
```
fileID = fopen('input_abaqus.inp','w');
fprintf( fileID, '%20s\n',’*node, nset = nglobal');
fprintf( fileID, '%2d, %6.2f, %6.2f, %6.2f\n', nodes' );
fclose( fileID );
```

- The content of the input_abaqus.inp file is:

```
*node, nset = nglobal
 1,   0.00,   0.00,   0.00
 2,  10.00,   0.00,   0.00
 3,  20.00,   0.00,   0.00
 4,   0.00,   2.50,   0.00
 5,  10.00,   2.50,   0.00
 6,  20.00,   2.50,   0.00
```

Exercise 1

- Realize the finite element model here below, by manually preparing the input file using the template reported before
- Write the nodal displacements and the stresses to the .odb and .dat files



Input data

a = 100 mm

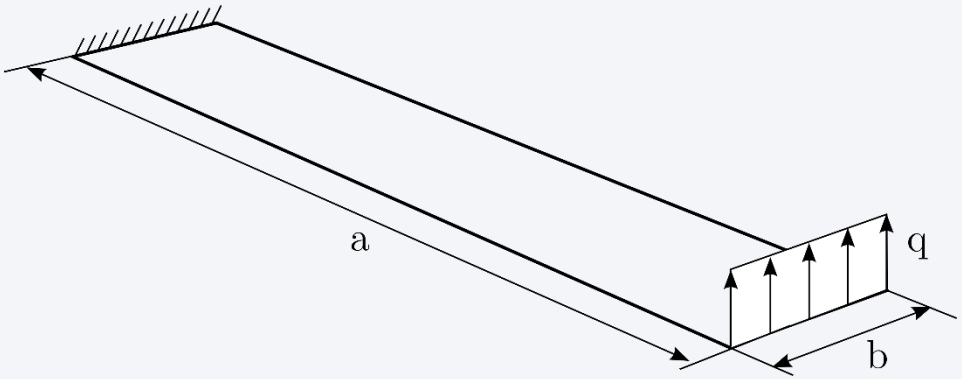
t = 1 mm

E = 72 GPa

v = 0.3

P = 10 N

Exercise 2

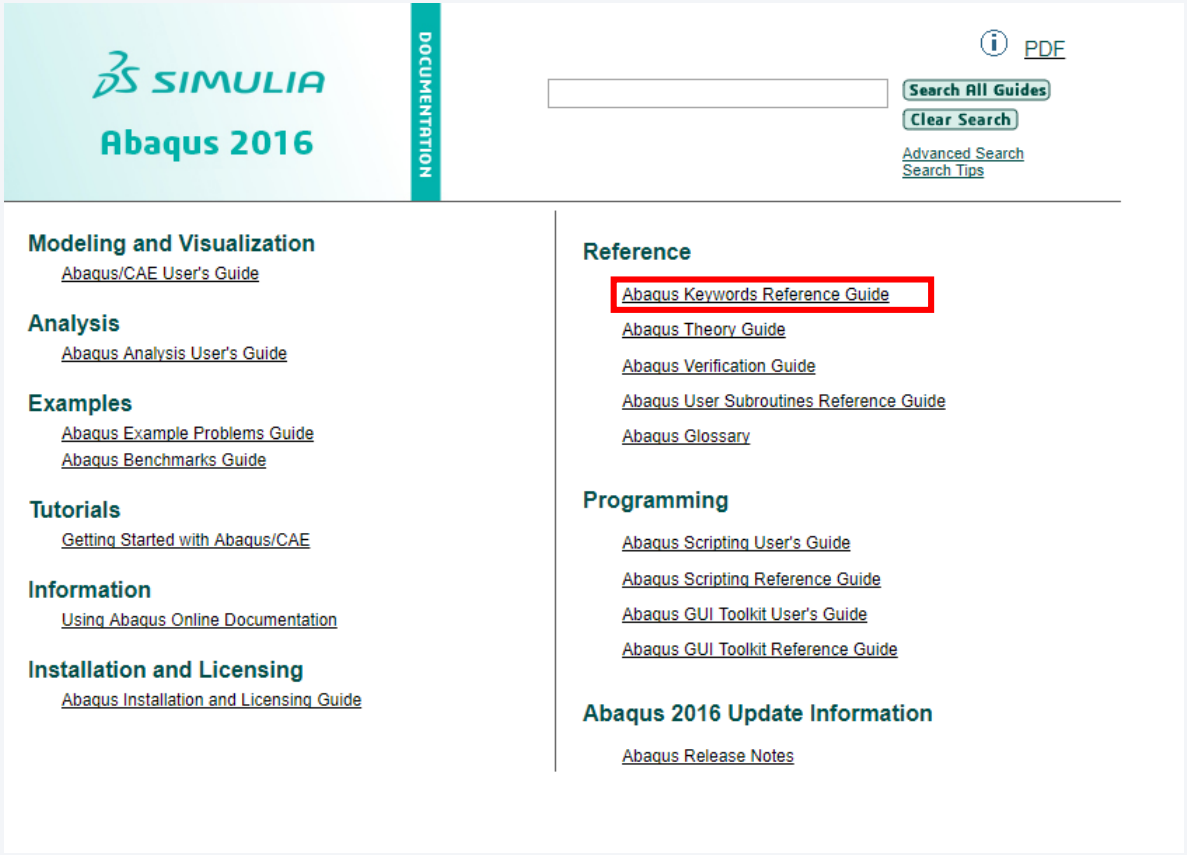


Input data

a = 1000 mm
b = 200 mm
t = 1.5 mm
q = 0.02 N/mm
E = 72 GPa
v = 0.3

- Use shell elements S4R (and *shell section to define the section property)
- Load discretization: the distributed load should be reported to the nodes
- Requests
  - Determine the displacement at the tip and compare it with the analytical solution (write this result to the .dat file)
  - Determine the maximum stress at the section at x = 350 mm and compare it with the analytical solution (write this result to the .dat file)
  - Study the convergence of the solution by considering different mesh sizes

Using the Abaqus Keyword Reference guide



Example: *node keyword

*NODE
Specify nodal coordinates.

This option is used to define a node directly by specifying its coordinates. Nodal coordinates given in this option are in a local system if the *SYSTEM option is in effect when this option is used.

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CFD Abaqus/CAE

Type: Model data

Level: Part, Part instance, Assembly

Abaqus/CAE: Mesh module

Reference:

Link(s) for additional information

- "Node definition," Section 2.1.1 of the Abaqus Analysis User's Guide

Optional parameters:

Optional: to be defined if needed, but not mandatory

INPUT

Set this parameter equal to the name of the alternate input file containing the data lines for this option. See "Input syntax rules," Section 1.2.1 of the Abaqus Analysis User's Guide, for the syntax of such file names. If this parameter is omitted, it is assumed that the data follow the keyword line.

NSET

Set this parameter equal to the name of the node set to which these nodes will be assigned. Node sets created or modified with this option will always be sorted.

SYSTEM

Set SYSTEM=R (default) to give coordinates in a rectangular Cartesian coordinate system. Set SYSTEM=C to give coordinates in a cylindrical system. Set SYSTEM=S to give coordinates in a spherical system. See Figure 14.10–1.

The SYSTEM parameter is entirely local to this option. As the data lines are read, the coordinates given are transformed to rectangular Cartesian coordinates immediately. If the *SYSTEM option is also in effect, these are local rectangular Cartesian coordinates, which are then immediately transformed to global Cartesian coordinates.

Example: *node keyword

Data lines to define the node: → `Data to be defined in the rows after the keyword definition`

First line:

1. Node number.
2. First coordinate of the node.
3. Second coordinate of the node.
4. Third coordinate of the node.
5. First direction cosine of the normal at the node (optional).
6. Second direction cosine of the normal at the node (optional). For nodes entered in a cylindrical or spherical system, this entry is an angle given in degrees.
7. Third direction cosine of the normal at the node (optional). For nodes entered in a spherical system, this entry is an angle given in degrees.

The normal will be used only for element types with rotational degrees of freedom. See Part VI, "Elements," of the Abaqus Analysis User's Guide.

Repeat this data line as often as necessary.

Example: *cload keyword

*CLOAD
Specify concentrated forces and moments.

This option is used to apply concentrated forces and moments at any node in the model. The *CLOAD option can also be used to specify a fluid reference pressure for incompressible flow in an Abaqus/CFD analysis and to specify concentrated buoyancy, drag, and inertia loads in an Abaqus/Aqua analysis.

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CFD Abaqus/CAE Abaqus/Aqua

Type: History data

Level: Step

Abaqus/CAE: Load module

References:

- "Concentrated loads," Section 34.4.2 of the Abaqus Analysis User's Guide
- "Abaqus/Aqua analysis," Section 6.11.1 of the Abaqus Analysis User's Guide
- "Analysis of models that exhibit cyclic symmetry," Section 10.4.3 of the Abaqus Analysis User's Guide
- "Defining ALE adaptive mesh domains in Abaqus/Explicit," Section 12.2.2 of the Abaqus Analysis User's Guide

Applying concentrated loads

Data lines to define concentrated loads for specific degrees of freedom:

First line:

1. Node number or node set label.
2. Degree of freedom.
3. Reference magnitude for load.

Repeat this data line as often as necessary to define concentrated loads.

Example: *shell section keyword

***SHELL SECTION**
Specify a shell cross-section.

This option is used to specify a shell cross-section.

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Type: Model data

Level: Part, Part instance

Abaqus/CAE: Property module

References:

- "Shell elements: overview," Section 29.6.1 of the Abaqus Analysis User's Guide
- "Using a shell section integrated during the analysis to define the section behavior," Section 29.6.5 of the Abaqus Analysis User's Guide

Required parameter: → `Mandatory!`

ELSET

Set this parameter equal to the name of the element set containing the shell elements for which the section behavior is being defined.

Required, mutually exclusive parameters: → `Choose between these two options`

COMPOSITE

Include this parameter if the shell is made up of several layers of material.

MATERIAL

Set this parameter equal to the name of the material of which the shell is made.

Example: *shell section keyword

- In this case the data to insert depend on the choice of the parameters

Data line to define a homogeneous shell (the MATERIAL parameter is included):

First (and only) line:

1. Shell thickness. This value is ignored if the NODAL THICKNESS or SHELL THICKNESS parameters are included.
2. Number of integration points to be used through the shell section. The default is five points if Simpson's rule is used and three points if Gauss quadrature is used. The number of integration points must be an odd number for Simpson's rule and is equal to the number of temperature degrees of freedom at a node of the element if this section is associated with heat transfer or coupled temperature-displacement elements. The maximum number of points for Simpson's rule is 99, and in the case of heat transfer or coupled temperature-displacement elements it is 19. This number must be at least 2 and less than or equal to 15 for Gauss quadrature. For Simpson's rule it must be at least 3, except in a pure heat transfer analysis, where the number of integration points can be 1 for a constant temperature through the shell thickness.

Data lines to define a composite shell (the COMPOSITE parameter is included):

First line:

1. Positive scalar value defining layer thickness or the name of a distribution ("Distribution definition," Section 2.8.1 of the Abaqus Analysis User's Guide) that defines spatially varying layer thicknesses. A distribution for composite layer thickness can be used only for conventional shell elements (not continuum shell elements). The layer thickness is modified if the NODAL THICKNESS or SHELL THICKNESS parameter is included.
2. Number of integration points to be used through the layer. The default is three points if Simpson's rule is used and two points if Gauss quadrature is used. The number of integration points must be an odd number for Simpson's rule, and it determines the number of temperature degrees of freedom at a node of the element if this section is associated with heat transfer or coupled temperature-displacement elements. The maximum number of points for Simpson's rule is 99, and in the case of heat transfer or coupled temperature-displacement elements it is 19. This number must be less than or equal to 15 for Gauss quadrature.
3. Name of the material forming this layer.
4. Name of the orientation to be used with this layer, an orientation angle, ϕ, or the name of a distribution ("Distribution definition," Section 2.8.1 of the Abaqus Analysis User's Guide) that defines spatially varying orientation angles. If the name of an orientation is used, the orientation cannot be defined with distributions. Orientation angles (in degrees) are measured positive counterclockwise relative to the orientation definition given with the ORIENTATION parameter. If the ORIENTATION parameter is not included, ϕ is measured relative to the default shell local directions (see "Orientations," Section 2.2.5 of the Abaqus Analysis User's Guide).
5. Name of the ply. Required only for composite layups defined in Abaqus/CAE.

Final remarks: creating the model

- No built-in system of units is provided. A consistent system has to be defined
  - m, N, kg, s, Pa, kg/m^3
  - mm, N, t, s, MPa, t/mm^3

- Preliminary analysis
  - perform preliminary calculations to compare with FEM results is always a good practice
  - nice contours do not mean good results

- Beginning with simple models
  - initial models do not need to be highly accurate
  - it is easier to identify errors on simple models
  - even in the context of nonlinear problems, an initial linear analysis can provide useful information

Final remarks: creating the model

- Element size
  - the use of very fine mesh has to be avoided, if not necessary
  - good practice to begin with a coarse mesh, and refine step by step
  - regular shapes guarantee, in commercial finite element codes, a better accuracy. Ideally, quadrangular element should be square
  - abrupt mesh transitions or warped elements should be avoided

Final remarks: verification phase

- Pay attention to the visualization options available on the post-processor (e.g. animations to view the results of static analyses are meaningless!)
- Check the satisfaction of the boundary conditions
- The finite element model is stiffer than the real structure
- To avoid interpretation errors, check how the results are plotted. Are the stresses, for instance, averaged at the nodes?
- Unexpected stress gradients can be due to modeling errors

Final remarks: possible sources of errors

- Singularities in the stiffness matrix (can the structure undergo a rigid body motion?)
- Nodes not connected to the elements
- Coincident nodes
- "Double" elements (responsible for unexpected local stiffening)