Documentation for abaqusreader v0.5

1.0 Introduction

The abaqusreader package allows abaqus .inp decks to be read by Solfec. It does not translate the Abaqus .inp deck into a Solfec .py file. Instead, allows the .inp file to be read from the Solfec .py file and used to define a Solfec model.

- Section 2 shows the simplest use of the module, which is all that's needed in most cases.
- Section 3 provides an overview of the major capabilities and limitations.
- Section 4 provides a reference for the full functionality.

2.0 Basic Tutorial

This tutorial explains the example.py script in /inp/mesh/abaqusreader/, which can be run using:

```
solfec –w –v example.py
```

from within this directory.

NOTE:

This example assumes the solfec/scripts directory is on Python's path – see scipts/README.txt for information.

First, the AbaqusInput class is imported from the module:

```
from abaqusreader import AbaqusInput
```

Second, a SOLFEC object is created for the new model – this is needed by the reader so it can generate materials:

```
solfec = SOLFEC('DYNAMIC', 1E-3, 'out')
```

Third, a new AbaqusInput object is created using the SOLFEC object and the path to the .inp deck:

```
model = AbaqusInput('tests/MODEL04.inp', solfec)
```

(any slashes in the path to the input deck should be forward slashes, even on a Windows system.)

Note this hasn't actually generated any Solfec BODY objects, but it does generate MESH and BULK_MATERIAL objects which can be used to do so. These objects and all the "raw" data read from the input deck are accessible as properties of the AbaqusInput object. This makes the reader very flexible. The data is structured in a similar way to the Abaqus input deck, i.e. with Instances, Parts etc.

If no modification of the Instances in the model is needed, creating bodies is straight-forward - e.g. to create a Finite Element body for each Instance in the Assembly:

```
for inst in model.assembly.instances.values():      # .instances is a dict
    label = inst.name   # use Abaqus instance name
    mesh = inst.mesh    # solfec MESH object at the instance's position
    bulkmat = inst.material    # solfec BULKMATERIAL object
    bdy = BODY(solfec, 'FINITE_ELEMENT', mesh, bulkmat, label)
```

When an input deck is read in Solfec's WRITE mode, a summary of what has been read will be output to the terminal. This output is suppressed in READ mode.

3.0 Summary of Features & Limitations

- The input deck must be in Assembly / Instance format.
- Only mesh, instance position and material/section data is read; constraints, step information etc are ignored.
- A MESH object is created for each *INSTANCE, with the instance's positioning applied.
- *ELEMENT may be defined more than once per *INSTANCE. Any type of 3D solid element should be supported except for axi-symmetric and degenerate elements. 2nd-order elements will automatically be converted to Solfec's linear elements by removing mid-side nodes – this will generate a warning (to stdout).
- Surface IDs can be defined for the MESH objects by including one or more *NSET with a name SURF*x* (where *x* > 0) - see documentation for *NSET under Section 4.6. Other surfaces have a SURFID of 0.
- A BULK_MATERIAL may be created for each *MATERIAL, which must have *DENSITY and *ELASTIC defined in that order.
- Each *PART should have one *SOLID SECTION defining the material applied. Currently all elements are given the same VOLID (defaulting to 1) so the MESH object for an *INSTANCE may only have a single BULK_MATERIAL.
- Node and element numbers are not maintained between Abaqus and Solfec due to different conventions and the possible conversion from 2nd order to linear elements - see Section 4.1.

4.0 Reference

4.1 Node and element numbering

IMPORTANT:

Node and element numbers in Solfec will differ from those specified in the Abaqus input deck due to different conventions:

- Abaqus numbering starts at 1 and does not have to be consecutive.
- Solfec starts at zero and must be consecutive.

As mid-side nodes are deleted this can also affect the ordering. Hence, an arbitrary mapping between them is performed when the .inp file is parsed.

See reference for Part.nodemap and Part.elmap for access to this mapping.

4.2 Input format

- The model must be organized into an assembly of part instances.
- Comment lines (starting **) are ignored completely.
- The reader is fully case-insensitive – but note that all parameters (e.g. part names) are converted to uppercase.
- Keywords and parameters must be spelt out in full (which is how Abaqus/CAE outputs them)
- Keyword lines must be on a single line – comma-continuation across line-breaks is not supported.
- Single- or double-quotes in names – which are required in the Abaqus deck if the name has a space in it - are removed from solfec names.

4.3 Module Level Variables

__version__	Returns a string containing version information, "*major.minor[tag]*". At the moment the optional tag is always "dev" to indicate it is under development. Once the version reaches 1, changes to the major version number will indicate changes to the interface.

4.4 AbaqusInput Objects

This both parses the input deck and provides containers for the objects used to representing it, described below.

Creation

```
obj = AbaqusInput(self, path, solfec=None, gid=0, volid=1)
```

Parameters:

path (str)	path to .inp file.

Optional parameters:

solfec	Solfec object, only needed if material properties are required.
gid (int)	Global surface ID for unspecified surfaces (default 0)
volid (int)	Volume identifier for all elements (default 1)

Properties

`.path`	str	Path to imported input deck
`.solfec`	SOLFEC object	Reference to SOLFEC object passed in.
`.parts`	dict	Dictionary of Part objects holding data from *PART cards. Key (str) is part name
`.assembly`	Assembly object	A single Assembly object holding data from a *ASSEMBLY card.
`.materials`	dict	Dictionary of ElasticMaterial objects holding data from *MATERIAL cards. Key (str) is material name. *Changed in v0.4: Dictionary values were BULK_MATERIAL objects, or None if no solfec object passed to AbaqusInput()*

4.5 Assembly Objects

This represents the data from a *ASSMEBLY card, of which there can only be one in each input deck.

`.name`	str	Name of assembly.
`.instances`	dict	Dictionary of Instance objects holding data from *INSTANCE cards. Key (str) is instance name.
`.elsets`	dict	Empty at present

4.6 Part Objects

This represents the data from a *PART card.

`.name`	str	Name of part.
`.materialname`	string	Name of corresponding material as read from *SOLID SECTION – see limitations there.

`.abqnodes`	dict	Dictionary of Abaqus nodal coordinates. Key (int) is Abaqus node number Value is a 3-tuple of floats giving x, y, z coordinates **NB:** See note on node/element numbers. *New in v0.4*
`.abqelements`	dict	Dictionary of Abaqus element data. Key (int) is Abaqus element number. Value is a tuple, where the first element is a string giving the element type, and subsequent elements are ints giving abaqus node numbers **NB:** See note on node/element numbers. *New in v0.4*
`.abqnodesets`	dict	Dictionary of Abaqus nodeset data for non-internal nodesets. Key (str) is set name. Value is a list of Abaqus node numbers **NB:** See note on node/element numbers. *New in v0.4*
`.nodes`	list	Flat list of nodal coordinates in format for `nodes` parameter of `MESH()`. **NB:** See note on node/element numbers.
`.elements`	list	Flat list of element data in format for `elements` parameter of `MESH()` The *volume identifier / volid* is set to the value given by the *`volid`* parameter to `AbaqusReader()`for all elements. **NB:** See note on node/element numbers.
`.surfids`	list	Flat list of surface ID information in format for `surfids` parameter of `MESH()`. Automatically generated from part-level *NSETs with a name of the form “SURFx” where x is a positive integer – see *NSET. Any faces for which surface IDs are not specifically defined (i.e. the `GID` in `MESH()`’s `surfids` parameter) are given the ID specified by the *`gid`* parameter to `AbaqusReader()`.
`.nodesets`	dictionary	Dictionary of nodeset data for non-internal nodesets in Solfec format. Key (string): *NSET name Value: list of ints giving the Solfec node numbers in the set
`.nodemap`	dictionary	Mapping between Abaqus and Solfec node numbers: Key (int): Abaqus node number – mid-side nodes won’t have an entry. Value (int): Solfec node number
`.elmap`	dictionary	Mapping between Abaqus and Solfec element numbers: Key (int): Abaqus element number Value (int): Solfec element number **NB:** See note on node/element numbers.

4.7 Instance objects

Represents the data from a *INSTANCE card

`.name`	str	Name of instance. Single- or double-quotes (needed in the Abaqus deck if the name has a space in it) are removed.
`.part`	Part	The corresponding Part object for this instance
*The following four properties control the position of the instance – see *INSTANCE in the Abaqus manual for how these are specified.*		
`.translate`	tuple	3-tuple of floats specifying translation to be applied to instance
`.point`	tuple	3-tuple of floats specifying position of point ‘a’ on rotation axis to be applied to instance
`.direction`	tuple	3-tuple of floats defining the direction of the rotation vector to be applied to instance. **NB:** This is not point ‘b’ as defined in the Abaqus manual but is in the correct format to to call: `ROTATE(shape, Part.point, Part.direction, Part.angle)`
`.angle`	float	Angle of rotation to be applied to instance
`.mesh`	MESH	Solfec MESH object corresponding to passing MESH() the .nodes, .elements and .surfids properties of the corresponding .part. Note the resulting MESH is translated and rotated to match the Instance position as defined above.
`.material`	BULK_MATERIAL	Solfec BULK_MATERIAL object - see *MATERIAL and *SOLID SECTION

4.8 ElasticMaterial objects

Represents the data from a *MATERIAL card followed by *DENSITY and *ELASTIC cards.

New in v0.4

`.name`	str	Name of material
`.density`	float	Density
`.Ym`	float	Young’s modulus
`.poisson`	float	Poisson’s ratio
`.bulkmat`	BULK_MATERIAL	Solfec BULK_MATERIAL object - see *MATERIAL and *SOLID SECTION

4.6 Supported Abaqus Keywords & Properties

Top level

The following keywords are supported at the top level of the input deck:

*HEADING	Required to be the first (non-comment) line of the input deck. No access is provided to heading data.
*PART	See details. Access through `AbaqusImport.parts` or `Instance.part`
*ASSEMBLY	See details. Access through `AbaqusImport.assembly`
*MATERIAL	See details.

	Access as a BULK_MATERIAL object through `AbaqusImport.materials` or `Instance.material` Access name through `Part.materialname`

*PART

No supported parameters

Supported keywords:

*NODE	**NB:** See note on node/element numbers. Access through `Part.nodes`
*ELEMENT	If convert=True: Can be any 3D solid element type with 4, 6, 8, 10, 15, or 20 nodes with any options such as hybrid formulation or reduced integration, i.e. any element type in Section 25.1.1 of the Abaqus Analysis User's manual should be supported. 2nd-order elements are automatically converted to 1st-order elements by removing mid-side nodes – a warning is output for each *ELEMENT block where this occurs. **NB:** See note on node/element numbers. If convert=False: Any Abaqus element type will be read. Access through `Part.elements`
*NSET	See details. Access through `Part.nsets`
*SOLID SECTION	Only one *SOLID SECTION should be defined for a part – this means each Part must be of a single material. See details.

*SOLID SECTION

Supported parameters

Material	Access through `Part.materialname`

*NSET

NB: Only part-level node-sets are supported.

Supported parameters:

Name	Used as key of `Part.nsets`
Internal	If this is found the *NSET is ignored
Generate	(not actually tested but should work ok)

NB: the following parameters are unsupported and will raise an error: Elset, Instance

Defining BODY surface IDs:

If a *NSET has the name of the form "SURFx" where x is an integer > 0 (or something such as 0001) which converts to the same), then the set is assumed to define a surface. The corresponding element faces of the instances MESH object will be given a surfid=x.

The surfaces defined by such sets do not need to be continuous, so for example a nodeset may be used to define a "surface" having the same ID on opposite faces of a body.

*NSETs may be conveniently defined in CAE by creating a new geometry-based Set. These will then be updated if the geometry is modified.

*ASSEMBLY

No supported parameters.

Supported keywords:

*INSTANCE	Access through Assembly`.instances`

*INSTANCE

Supported parameters:

Name	Access as Instance`.name`
Part	Access as Instance`.part`

Both translational and rotational positioning of instances is supported.

*MATERIAL

Supported parameters:

Name	Access as `BULK_MATERIAL.label`

Supported keywords:

*DENSITY	**NB:** 1) Only single values are supported (not temperature or field dependence) 2) *DENSITY and *ELASTIC must occur immediately after *MATERIAL, in that order.
*ELASTIC	See above.

If density, young's modulus and poission's ratio are not found then an error is raised.

5.0 History

Contains major changes and API changes only.

V0.4:

- AbaqusInput(convert=True|False) option added to allow decks to be parsed, but not converted to solfec format meshes. Was required to support Atkins work. Some refactoring.

V0.3:

- Passing a solfec object is now optional, volid and gid can be passed.

V0.2:

- Added support for any type of 3D solid element with 4, 6, 8 10, 15 or 20 nodes. 2nd-order elements are converted (with a warning) to linear elements by discarding mid-side nodes.
- Added support for non-consecutive Abaqus node and element numbers, through arbitrary mapping of these to Solfec node/element IDs. Part.nodemap and Part.elmap added.

V0.1:

The abaqusreader module was based on the abaqusread module which shipped with Solfec v0.673, with the following major changes:

- Refactored to provide a flexible object-orientated interface to the contents of deck instead of actually creating bodies etc.
- Added printonce() function and additional debug information in WRITE mode only.
- Ignores all comment lines
- Changed actual parsing to be case-insensitive and much closer to Abaqus specification. Resolved problems with some parameter names and capitalisation.
- Added functionality to use *NSETS to define surface IDs of MESH objects
- Code formatting, variable names etc changed to be closer to PEP8 (with 2-space indentations).