Shellexpander: A preprocessor for the cross-section analysis software BECAS

Robert David Bitsche*
Technical University of Denmark, Department of Wind Energy

September 23, 2013

Shellexpander is a python program that generates input files for the cross-section analysis software BECAS [Blasques, 2011] based on information contained in an finite element shell model.

1 Version

This document describes version 1.3 of shellexpander.py. Call shellexpander with the option `--version` to display version information.

2 Concept

While shellexpander was developed for the analysis of wind turbine blades, it can be used for any thin-walled beam-like structure.

The input required by BECAS for each cross-section comprises a 2D-mesh of the cross-section and the corresponding material and orientation assignments. Shellexpander generates suitable BECAS input files for an arbitrary number of cross-sections based on the information read from an finite element shell model. It is assumed that the shell model uses layered shell elements. The nodes of the shell model may be offset from the mid-surface.

Each shell element is converted into a stack of three-dimensional solid elements by "expansion" in the local normal direction. The two-dimensional mesh for BECAS is then found by projecting the front faces of the three-dimensional elements to an appropriate plane. 4-node and 8-node quadrilateral shell elements result in 4-node and 8-node quadrilateral BECAS elements respectively.

*robi@risoe.dtu.dk

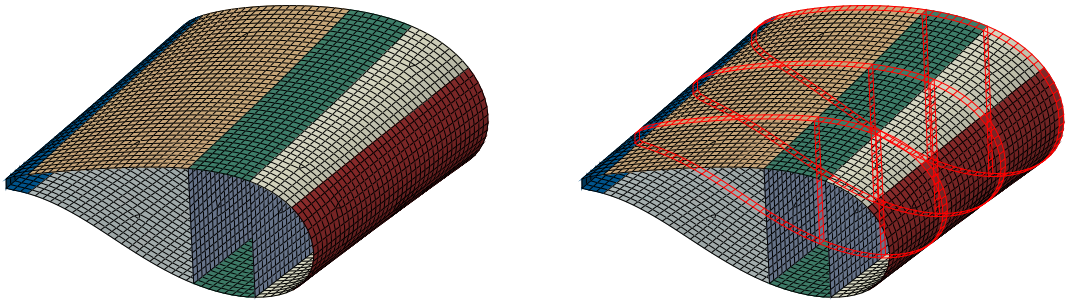

Figure 1: Left: Part of a finite element shell model of a wind turbine blade; Right: Three element sets defining three cross-sections that are analyzed by BECAS.

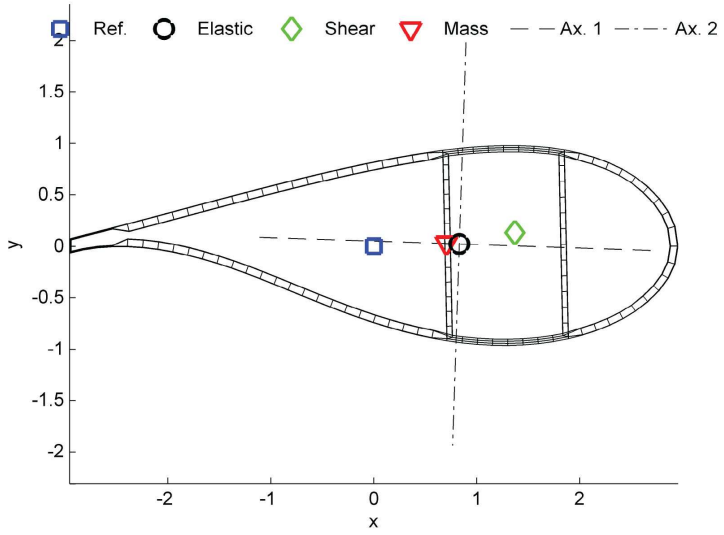

Figure 2: 2D mesh generated by shellexpander. The center of mass, shear center, elastic center and the elastic axes are also indicated.

As the process described above can lead to "unexpected" positions of mid-side nodes of the 8-node BECAS elements, all mid-side nodes are repositioned to the midpoints between the corresponding main nodes after the 2D mesh has been generated.

Any number of elements n can be used to discretize the shell thickness as defined by the `--layers` option. The minimum value for n is the largest number of layers of different material anywhere in the shell model.

Figure 1 shows part of a finite element shell model of a wind turbine blade. Figure 2 shows one of the 2D meshes generated by shellexpander based on the shell model.

Shellexpander does not operate on all shell elements found, but only on those contained in the element sets given as arguments in the command line. This allows to disregard certain parts of the model. More importantly, if more than one element set is given, nodes

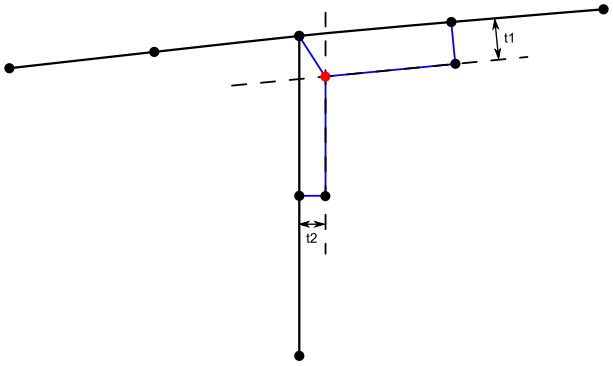

Figure 3: If a node is considered a corner, the offset node is found by intersecting two offset surfaces.

appearing in more than one element set are assumed to be "corners" and are therefore treated specially:

- Based on the respective element sets, two normal directions and two thicknesses are computed at the node.
- The offset node is found by intersecting two offset surfaces as shown in Figure 3.
- If a node appears in more then 2 element sets, a warning message is issued and only the first two normals (and thicknesses), corresponding to the first two element sets specified, are used.

For a standard wind turbine blade it may be a good idea to treat the intersection of the shear webs and the airfoil and the trailing edge as "corners".

As layup and thickness information in the finite element shell model is usually defined on an element basis, thickness discontinuities occur - see Figure 4 (left). In order to facilitate mesh generation, shellexpander removes these discontinuities by defining a continuous, node-based thickness distribution. This is achieved by defining the thickness at a node as the minimum, maximum or average thickness of any shell element attached to this node. An example using the minimum shell thickness is shown in Figure 4 (right).

In addition to the algorithm described above the node-based thickness distribution can be controlled by defining dominant element sets (see the `--dom` option). If dominant element sets are specified, the node-based thickness distribution at thickness discontinuities is governed by the dominant elements. For a wind turbine blade it may be a good idea to define load carrying parts (e.g. the caps) as dominant, because even small thickness changes of elements representing these parts may have a non-negligible effect on the cross section stiffness properties.

As shellexpander defines a continuous, node-based thickness distribution and uses the same number of elements to discretize the shell thickness everywhere, the 2D mesh generated by shellexpander and the underlying shell model do not describe exactly the same cross-section. In order to minimize the deviations between the shell model and the 2D mesh generated by shellexpander the following guidelines are suggested:

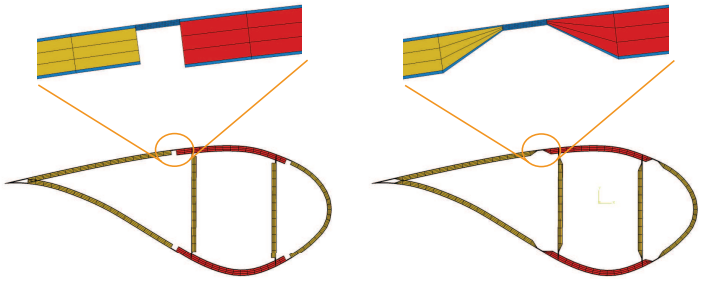

Figure 4: Left: Thickness discontinuities; Right: Continuous, node-based thickness distribution (minimum thickness).

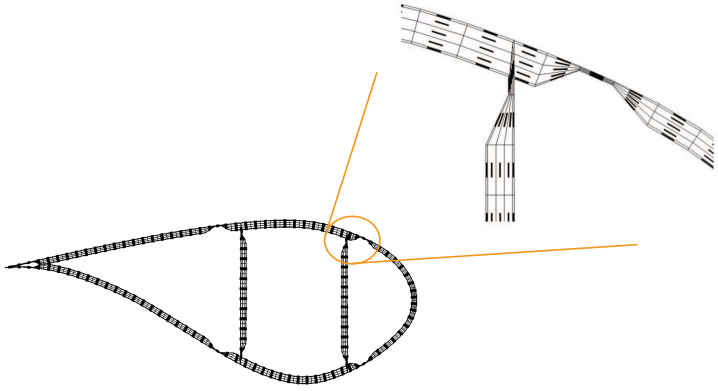

Figure 5: BECAS fiber plane angle.

- Use the `--dom` option for structurally important parts like, for example, load carrying caps.
- Use a sufficiently high number of elements to discretize the shell thickness (`--layers` option). A sufficiently high number of elements should result in a mesh where all finite elements in one stack have similar thickness. This minimizes error introduced at nodes where the layup changes. Numbers higher than 16 are usually not required.
- Using an even number for the number of elements to discretize the shell thickness (`--layers` option) is the better choice more often than not.

3 Material Orientation

The BECAS fiber plane angle is automatically determined based on the orientation of the generated 2D elements as shown in Figure 5.

The material orientation angle found in the shell model is used as BECAS fiber angle. No checks or conversions with respects to local coordinate systems are performed. That is, the user is responsible for a consistent material orientation definition.

4 Local Coordinate Systems

The two-dimensional mesh for BECAS is found by projecting the front faces of the three-dimensional elements to an appropriate plane.

By default, this plane is the x-y-plane. Alternatively, the x-y-plane of a local coordinate system can be used. A local coordinate system can be defined at each cross-section using the `--cline` option. The procedure defining these local coordinate systems is described in the following paragraphs.

The textfile given by the `--cline` option must contain 6 real numbers per line and as many lines as cross-sections that should be analyzed (see `--sec` option). The first 3 real numbers in each line are the x-, y- and z-coordinate of the points $\vec{A}_i$ defining the "centerline" of the beam. The real numbers 4 to 6 define an "additional" node $\vec{B}_i$ for each cross-section. The points $\vec{A}_i$ and $\vec{B}_i$ must be located in the respective cross-section defined by the element sets given using the `--sec` option. If the mean z-coordinate of the element nodes defining a cross-section and the z-coordinate of $\vec{A}_i$ differ by more than 2%, a warning message is issued.

The local coordinate system is defined by three unit vector $\vec{e}_1$, $\vec{e}_2$ and $\vec{e}_3$, where $\vec{e}_3$ is the normal of the cross-section plane. The origin of the local coordinate system is the centerline point $\vec{A}_i$. $\vec{e}_3$ is computed as the tangent of the cubic[1], interpolating spline connecting the centerline points $\vec{A}_i$.

A preliminary vector ${}^*\vec{e}_1$ is defined as:

$$ {}^*\vec{e}_1 = \vec{B}_i - \vec{A}_i \tag{1} $$

Vector $\vec{e}_2$ is then defined as:

$$ \vec{e}_2 = \vec{e}_3 \times {}^* \vec{e}_1 \tag{2} $$

Finally, vector $\vec{e}_1$ is defined as:

$$ \vec{e}_1 = \vec{e}_2 \times \vec{e}_3 \tag{3} $$

5 Use

Shellexpander is a command line program. Enter `shellexpander.py --help` to display a short help message and a list of the available command line options.

Element sets are used to define an arbitrary number of cross-section that should be analyzed by BECAS as shown in Figure 1 (right). Each element set should contain a single "slice" of the model. The element sets must have consecutive names. For example, if the element sets are called `BECAS_SECTION_01`, `BECAS_SECTION_02`, and so forth, then BECAS should be called with the option `--sec BECAS_SECTION`.

Instead of typing a long command line, the list of arguments can also be stored in a file. An argument starting with an @ character is treated as a filename, and is replaced by the

[1] If only three cross-sections are defined, a quadratic spline is used. If only two cross-sections are defined, a linear function is used.

arguments the file contains. The arguments in the file must be given one per line.
For example: `shellexpander.py @options.txt`

6 Demonstration Examples

The directory `test` contains two demonstration examples. See the file `README.TXT` for further instructions.

7 Limitations

Currently, shellexpander has the limitations listed below. Some of these limitations may be removed in future versions.

- The input file containing the finite element shell model must be in ABAQUS[2] format.
- Only 4-node and 8-node quadrilateral shell elements can be used. The input file may contain other element types, as long as they do not appear in the element sets defining cross-sections that should be analyzed.
- If the ABAQUS input file uses the part/instance concept, only one part and one instance may be used.

8 Software Requirements

Shellexpander requires Python 3.2 or newer and the NumPy and SciPy modules. Python can be downloaded from http://www.python.org. The NumPy module and the SciPy module are available from http://www.scipy.org.

9 Licensing

Shellexpander is distributed as part of BECAS [Blasques, 2011] and covered by the BECAS license.

References

J. P. Blasques. User's manual for BECAS - a cross section analysis tool for anisotropic and inhomogeneous beam sections of arbitrary geometry. Technical Report Risø-R-1785(EN), Risø DTU, National Laboratory for Sustainable Energy, 2011.

[2] http://www.simulia.com