PreVABS Version 0.6 User's Manual

Su Tian, Xin Liu and Wenbin Yu
Purdue University

March 3, 2019

Contents

Chapter 1

Installation and Execution

1.1 Installation

- Download and install VABSIII (with valid license). In the instructions below, `VABS_DIR` refers to the path to the VABSIII executable;
- Download and install Gmsh (`http://gmsh.info/`) following the official instructions. In the instructions below, `GMSH_DIR` refers to the path to the Gmsh executable;
- Download PreVABS from cdmHUB (`https://cdmhub.org/resources/1597/supportingdocs`). Unpack the package to any location. In the instructions below, `PREVABS_DIR` refers to the path to the PreVABS executable;
- Add those paths to executables to the system environment variable `PATH`;
  - On Windows:

    Open Environment Variables editor. Edit user variables for your account or edit system variables if you have the administrator access;

    Add `VABS_DIR`, `GMSH_DIR`, and `PREVABS_DIR` to the variable `PATH`.
  - On Linux:

    In the bash shell, type

    ```
    export PATH=$VABS_DIR:$GMSH_DIR:$PREVABS_DIR:$PATH
    ```

    To make this effective each time starting the bash, you can add this command to the bash startup file, which may be `~/.bashrc`, `~/.bash_profile, \lstinline` /.profile—, or `~/bash_login`;

1.2 Execution

- PreVABS is a command line based program which acts as a general-purpose preprocessor and post-processor based on parametric inputs necessary for designing a cross section;
- Download the examples package from cdmHUB (`https://cdmhub.org/resources/1597/supportingdocs`), and unpack it to any location;
- If you have already added the folder where you stored VABS, Gmsh and PreVABS to the system or user environment variable `PATH`, to execute PreVABS, you can open any command line tool (Command

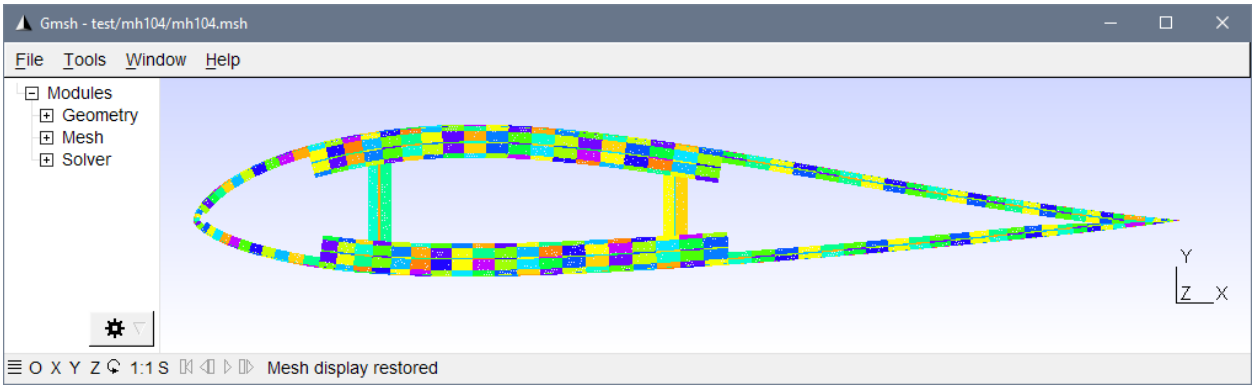

Figure 1.1: Cross section with meshes generated by PreVABS and visualized by Gmsh. Example: `examples\ex_airfoil\mh104.xml`.

Prompt or PowerShell on Windows, Terminal on Linux), change directory to the root of the PreVABS package, and type the following command:

- On Windows:

  ```
  prevabs -i examples\ex_airfoil\mh104.xml -h -v
  ```

- On Linux:

  ```
  prevabs -i examples\ex_airfoil/mh104.xml -h -v
  ```

- The first option `-i` indicates the path and name for the cross section file (`ex_airfoil\mh104.xml` for this case). The second option `-h` indicates the analysis to compute cross-sectional properties (this analysis is also called homogenization), where meshed cross section will be built and VABS input file will be generated. The last option `-v` is for visualizing the meshed cross section;
- PreVABS will read the parametric input files and generate the meshed cross section;
- Once finished, PreVABS will invoke Gmsh, a tool for visualization, to show the cross section with the corresponding meshes, as shown in Fig. 1.1. Three files are generated in the same location at this moment, a VABS input file `mh104_vabs.dat`, a Gmsh geometry file `mh104.geo`, and a Gmsh mesh file `mh104.msh`. The geometry file is used to inspect errors when meshing cannot be accomplished. The latter two files are generated only when visualization is needed;
- Then user can run VABS using the generated input file.

NOTE PreVABS and Gmsh are free and open source. The source codes of PreVABS and Gmsh are available on cdmHUB at `https://cdmhub.org/resources/1597`. You can make changes to both codes by modifying its source codes. However, VABS is a commercial code and you need to request the code and a valid license from AnalySwift (`http://analyswift.com/`).

1.3 Command line option

PreVABS is executed using command `PreVABS` with other options. If no option is given, a list of available arguments will be printed on the screen.

Command

```
prevabs -i <main_input.xml> [options]
```

Options

`-h` Build cross section and generate VABS input file for homogenization.

`-d` Read 1D beam analysis results and update VABS/SwiftComp input file for recovery.

`-v` Visualize meshed cross section for homogenization or contour plots of stresses and strains after recovery.

`-e` Execute VABS/SwiftComp. stress/strain/displacement distribution over the cross section.

`-vabs` Use VABS (Default).

`-sc` Use SwiftComp.

`-f` Initial failure strength analysis (SwiftComp only).

`-fe` Initial failure envelope (SwiftComp only).

`-f` Initial failure indices and strength rations (SwiftComp only).

NOTE When VABS is called by PreVABS (using the option `-e`), the actual name of the executable used here is `VABSIII`.

1.3.1 Case 1: Build cross section from parametric input files

```
prevabs -i cross_section.xml -h -v
```

In this case, parametric input files are prepared for the first time, and one may want to check the correctness of these files and whether the cross section can be built as designed. One may also want to try different meshing sizes before running the analysis.

1.3.2 Case 2: Carry out homogenization without visualization

```
prevabs -i <cross_section.xml> -h -e
```

The command will build the cross section model, generate the input, and run VABS to calculate the cross-sectional properties, without seeing the plot, since visualization needs extra computing time and resources. One can also make modifications to the design (change the parametric inputs) and do this step repeatedly. If you already have generated the input file *cross_section_vabs.dat*, and want to only run VABS, you can invoke VABS directly using `VABSIII` *cross_section_vabs.dat*.

1.3.3 Case 3: Predict 3D stress/strain and plot

```
prevabs -i cross_section.xml -d -e -v
```

After getting the results from a 1D beam analysis, one may want to find the local strains and stresses of a cross section at some location along the beam. This command will let PreVABS read those results, update the VABS input file, carry out recovery analysis, and finally draw contour plots in Gmsh Fig. 1.2. An example of the recover analysis can be found in Section 3.8.

NOTE Before any recovery run, a homogenization (with option `-h`) run must be carried out first for a cross section file. In other words, the file *cross_section.dat.opt* must be generated before the recovery run. Besides, results from the 1D beam analysis need to be added into the *cross_section.xml* file. Preparation of this part of data is explained in Section 2.8.

NOTE Plotted data are the nodal strains and stresses in the global coordinate system.

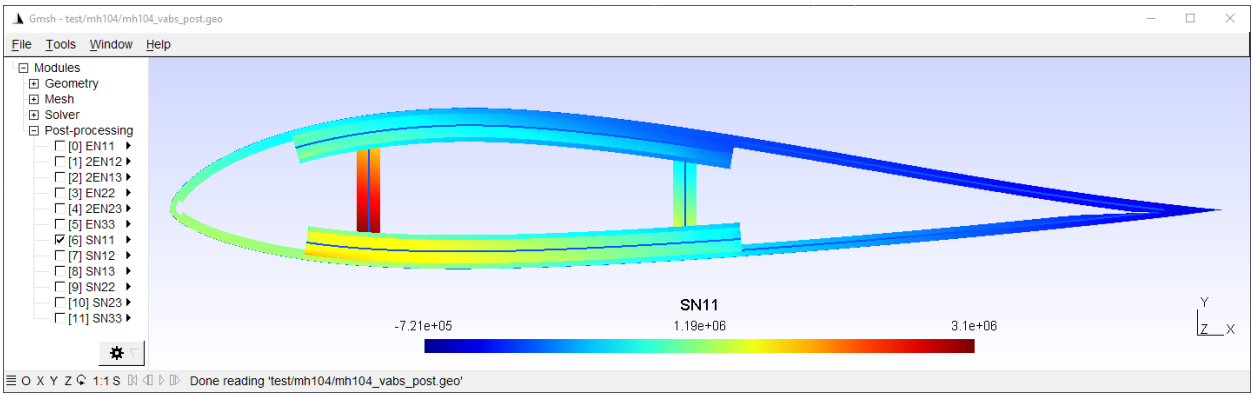

Figure 1.2: Visualization of strains and stresses in Gmsh.

Chapter 2

Prepare Cross Section

2.1 Introduction

In PreVABS, a cross section is defined through three aspects: geometry, material and overall configuration, as shown in Fig. 2.1. The geometry aspect comprises definitions of base points and base lines. The material aspect includes material properties, lamina thicknesses, layup stacking sequences, etc. The overall configuration defines the general size of the cross section, position of the coordinate origin, mesh size and element type of the finite element model.

In PreVABS, the key to preparing input files for a cross section is defining segments. A segment is a unique combination of a base line and a layup, as shown in Fig. 2.2. Another important concept is level, which collects segments into different groups based on how they connect to each other. For regions like the leading and trailing edges of an airfoil, it is sometimes difficult to handle. PreVABS treats these connections separately and thus requires separate declarations of these joints between segments. More details about these concepts can be found in Section 2.3, Section 2.4 and Section 2.5.

To prepare the cross section, five input files are needed. Definitions of base points, base lines, materials and layups are stored in four input files separately. A top level cross section file stores the definitions of segments and connections, overall configurations and references to other files. Except the base points file, which has a file extension .dat, all other files use an XML format.

2.2 Coordinate systems

There are three coordinate frames used in PreVABS, a basic frame $\mathbf{z}$, a cross-sectional frame $\mathbf{x}$, and an elemental frame $\mathbf{y}$, as shown in Fig. 2.3. Here, z_1, x_1 and y_1 are parallel to the tangent of the beam reference line and pointing out of the paper. The basic frame is where base points are defined. The cross-sectional and

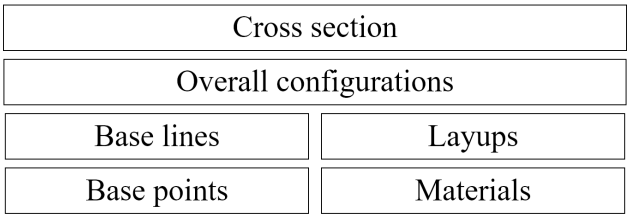

Figure 2.1: Cross section definition in PreVABS.

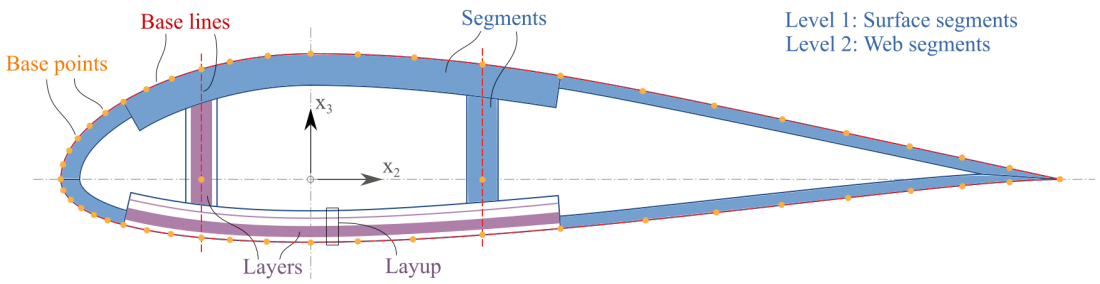

Figure 2.2: Basic components in a typical cross section.

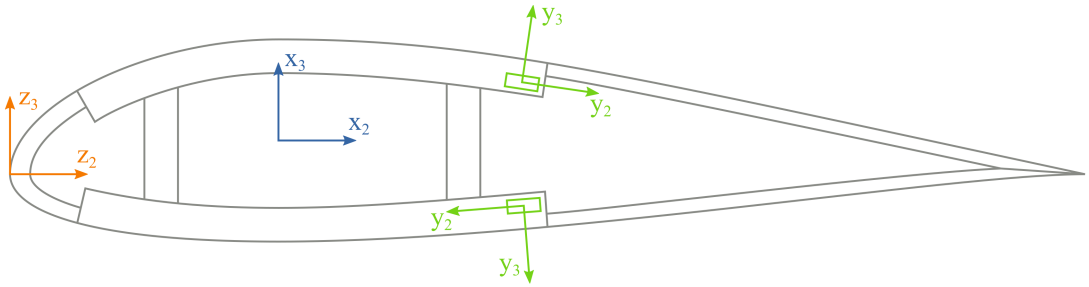

Figure 2.3: The basic, cross-sectional and elemental frames in a cross section.

elemental frames have the same definitions as those in VABS. User can define the topology of a cross section in the basic frame **z** and use manipulations like translation, scaling and rotation to generate the actual geometry in **x**. For an airfoil cross section, airfoil surface data points downloaded from a database having chord length 1 are in the frame **z**, and they are transformed into the frame **x** through translation (re-define the origin), scaling (multiplied by the actual chord length), and rotation (attack angle) if necessary, as shown in Fig. 2.4. More details about this transformation can be found in Section 2.7 below. In PreVABS, the definition of the elemental frame **y** follows the rule that the positive direction of y_2 axis is always the same as the direction of the base line, and then y_3 is generated based on y_1 and y_2 according to the right-hand rule. More details about the base line can be found in Section 2.3 below.

2.3 Base points and lines

2.3.1 Base points

Base points are used to draw base lines, which form the skeleton of a cross section. They can be either actually on the base lines, or not, as reference points, for example the center of a circle. Points that are directly referred in the definitions of base lines are called key points, such as starting and ending points of a

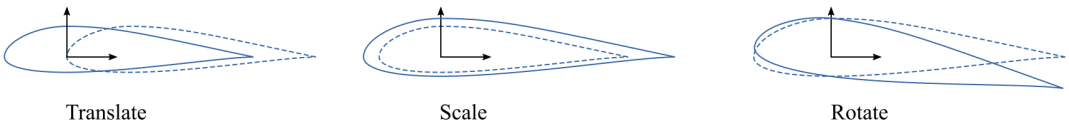

Figure 2.4: Three manipulations to transform a cross section.

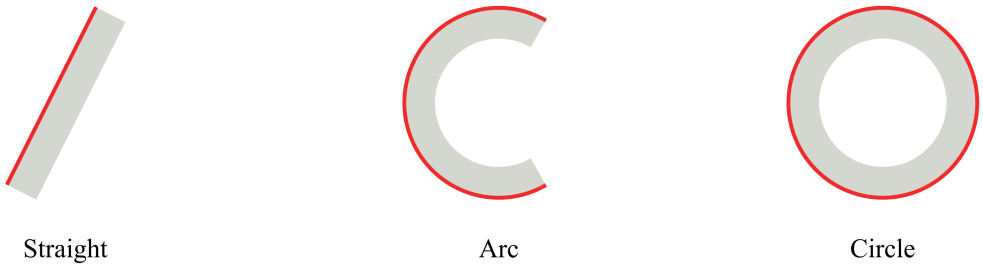

Figure 2.5: Three types of base lines in PreVABS.

line or an arc, or the center of a circle. The rest are normal points. The coordinates provided in the input file are defined in the basic frame **z** and then transformed into the cross-sectional frame **x**, through processes like translating, scaling and rotating. If none of those operations are needed, then those data also define the position of each point in the frame **x**.

The file storing these data is a plain text file, with a file extension `.dat`. This block of data has three columns and n rows, where n is the number of base points. The three columns are arranged as

```
label z2 z3
```

where the first column is the labels/names for each point, and the second and the third columns are the z_2 and z_3 coordinates, respectively. Some notes for this input file are:

- Three columns are separated by spaces;
- `label` can be the combination of any letters, numbers and underscores "_";
- Each key points name must be unique;
- Meaningful names for key points are highly recommended;
- Names of normal points can be less meaningful, even identical;
- The order of the point list is important for those points belonging to a sublist defining the direction of a base line, but it can be placed in any part of the big list.

2.3.2 Base lines

Base lines form the skeleton of a cross section. PreVABS can handle three types of base lines, straight, arc and circle, as shown in Fig. 2.5. Spline are approximated by straight lines. Some types have several ways to define the base line. In the end, all curved base lines are converted into a chain of short straight lines in PreVABS. User can provide those short straight lines directly for spline, arc and circle. Or, for arc or circle, user can use simple rules to draw the shape first and then PreVABS will discretize it.

Data for base lines are stored in an XML formatted file. The general arrangement of data is shown in Listing 2.1. All base line definitions are stored in the root element `<baselines>`, which has one attribute `basepoints` indicating the name of base point file used in this cross section. Each `<baseline>` element is the definition of a base line. Each one has a unique `name` and a `type`, which can be `straight`, `arc` or `circle`. Inside the `baseline` element, the `straight` and `arc` types have several different ways of definition, and thus the arrangements of data are different, which will be explained in details below.

Listing 2.1: Base line input file template

```
<baselines basepoints="basepoints">
  <baseline name="name1" type="straight">...</baseline>
  <baseline name="name2" type="arc">...</baseline>
```

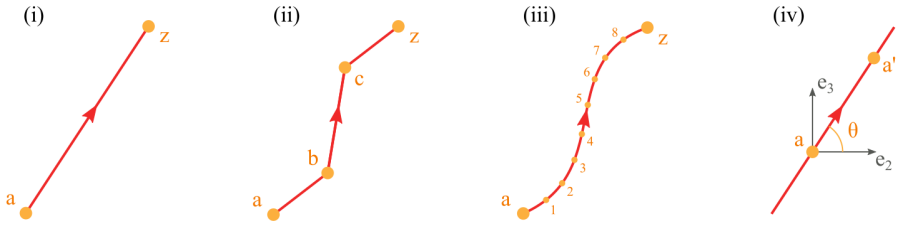

Figure 2.6: Different ways of defining straight base lines in PreVABS.

```
    <baseline name="name3" type="circle">...</baseline>
    ...
</baselines>
```

Straight

For the straight type, the basic idea is to provide key points for the chain of straight lines and the direction of the base line is defined by the order of the point list, as shown in Fig. 2.6. Spline is also defined in this way. Some notes are:

- All key points are placed in the `<points>` element;
- Key points are separated by comma;
- Successive points can be included using two key points separated by colon;
- Space is not allowed;
- Above two methods can be used in combination.

NOTE Use `type="straight"` for splines.

Besides, a simple straight line like (i) in Fig. 2.6 can also be defined using one key point and an angle, which is shown as (iv) in the same figure. In this case, PreVABS will calculate the second key point (a') and generate the base line. Some notes are:

- Two elements are used here, `<point>` and `<angle>`. The former element stores the user-provided key point "a", and the latter element stores the angle "theta";
- The positive angle (degree) is defined from the positive z_2 axis, counterclockwise;
- The PreVABS-computed second key point will always be "not lower" than the user-provided key point, which means the base line will always be pointing to the upper left or upper right, or to the right if it is horizontal;
- The two key points do not indicate the two ends of the base line; or, in other words, base line defined in this way is infinite.

For the four base lines in Fig. 2.6, the sample input file is shown in Listing 2.2, assuming that there is a base point file named `basepoints.dat` in the same folder.

Listing 2.2: Sample input file for the base lines shown in Fig. 2.6

```
<baselines basepoints="basepoints">
  ...
  <baseline name="i" type="straight">
    <points>a,z</points></baseline>
  <baseline name="ii" type="straight">
```

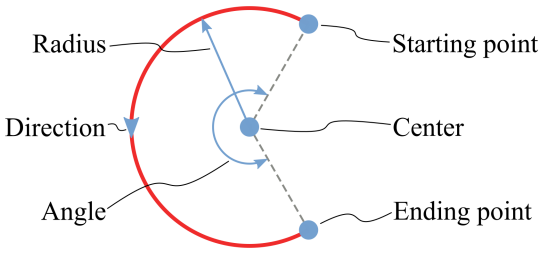

Figure 2.7: Items in an arc.

```
    <points>a,b,c,z</points></baseline>
  <baseline name="iii" type="straight">
    <points>a:z</points></baseline>
  <baseline name="iv" type="straight">
    <point>a</point><angle>theta</angle></baseline>
  ...
</baselines>
```

Arc

A real arc can also be created using a group of base points, in which case the straight type should be used. The arc type provides a parametric way to build this type of base line, then PreVABS will discretize it. To uniquely define an arc, user needs to provide at least four of the following six items: center, starting point, ending point, radius, angle and direction, as shown in Fig. 2.7. Two possible ways of defining an arc are given in Fig. 2.8, where the first one is defined by center, starting point, ending point and direction, and the second one is defined by center, starting point, angle and direction.

For preparing the input file, some notes are:

- Element tags used for arc are `<center>`, `<start>`, `<end>`, `<radius>`, `<angle>` and `<direction>`;
- Not used elements should be deleted;
- `<center>`, `<start>` and `<end>` store the key point labels from the base point file;
- `<angle>` stores the angle in degrees greater than 0 and less than 360;
- `<direction>` can only be `cw` for clockwise or `ccw` for counterclockwise. `ccw` is the default.

NOTE `<radius>` is ignored in the current version, and will be supported in a future version.

A sample input file for the two arcs shown in Fig. 2.8 is provided in Listing 2.3.

Listing 2.3: Sample input file for the base lines shown in Fig. 2.8

```
<baselines basepoints="basepoints">
  ...
  <baseline name="left" type="arc">
    <center>c</center>
    <start>s</start>
    <end>e</end>
    <direction>ccw</direction>
  </baseline>
  <baseline name="right" type="arc">
    <center>c</center>
```

Figure 2.8: Two possible cases of defining an arc.

```
      <start>s</start>
      <angle>a</angle>
      <!-- here the direction is the default value 'ccw' -->
      </baseline>
    ...
</basepoints>
```

Circle

Defining a circle is simpler than an arc. User only need to provide a center with radius or another point on the circle. The corresponding element tags are `<center>`, `<radius>` and `<point>`. A sample input file demonstrating the two methods is presented in Listing 2.4.

Listing 2.4: Sample input file for circles

```
<baselines basepoints="basepoints">
  ...
  <baseline name="circle1" type="circle">
    <center>c</center>
    <radius>r</radius>
  </baseline>
  <baseline name="circle2" type="circle">
    <center>c</center>
    <point>p</point>
  </baseline>
  ...
</baselines>
```

2.4 Materials and layups

PreVABS uses the keyword material for the physical properties attached to any materials, while lamina for material plus thickness, which in a sense is fixed by manufacturers. This can be thought as the basic commercially available "material", such as a composite preprag. A layer is a stack of laminas with the same fiber orientation. The thickness of a layer can only be a multiplier of the lamina thickness. Layup is several layers stacked together in a specific order. This relationship is illustrated in Fig. 2.9. More details can be found below.

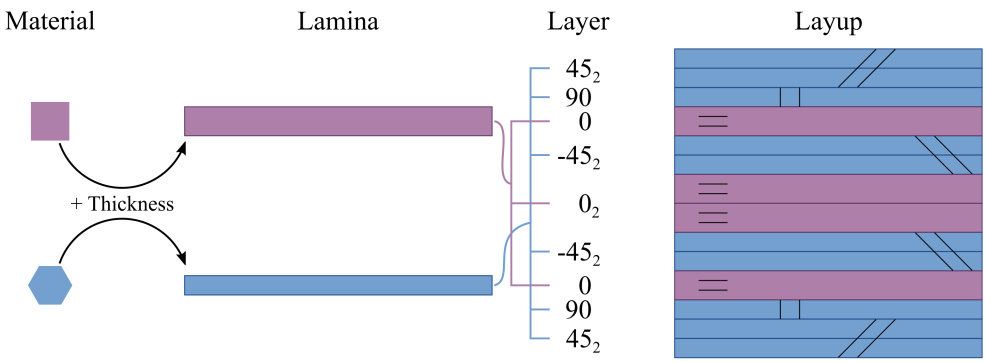

Figure 2.9: Relationship between materials, lamina, layer, and layup.

2.4.1 Materials and laminae

Both materials and laminae are stored in one XML file, under the single root element `<materials>`. A template of this file is shown in Listing 2.5. Each material must have a name and type. Under each `<material>` element, there are a `<density>` element and an `<elastic>` element. The arrangement of elastic properties is different for different types:

- `isotropic` material has 2 constants: `<e>` and `<nu>`;
- `orthotropic` material has 9 constants: `<e1>`, `<e2>`, `<e3>`, `<g12>`, `<g13>`, `<g23>`, `<nu12>`, `<nu13>` and `<nu23>`;
- `anisotropic` material has 21 constants: `<c11>`, `<c12>`, `<c13>`, `<c14>`, `<c15>`, `<c16>`, `<c22>`, `<c23>`, `<c24>`, `<c25>`, `<c26>`, `<c33>`, `<c34>`, `<c35>`, `<c36>`, `<c44>`, `<c45>`, `<c46>`, `<c55>`, `<c56>` and `<c66>`. These constants are defined in Eq. (2.1).

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{12} \\ \sigma_{13} \\ \sigma_{22} \\ \sigma_{23} \\ \sigma_{33} \end{Bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ c_{12} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ c_{13} & c_{23} & c_{33} & c_{34} & c_{35} & c_{36} \\ c_{14} & c_{24} & c_{34} & c_{44} & c_{45} & c_{46} \\ c_{15} & c_{25} & c_{35} & c_{45} & c_{55} & c_{56} \\ c_{16} & c_{26} & c_{36} & c_{46} & c_{56} & c_{66} \end{bmatrix} \begin{Bmatrix} \epsilon_{11} \\ 2\epsilon_{12} \\ 2\epsilon_{13} \\ \epsilon_{22} \\ 2\epsilon_{23} \\ \epsilon_{33} \end{Bmatrix} \tag{2.1}$$

Listing 2.5: A template for the material and lamina input file

```
<materials>
  ...
  <material name="iso1" type="isotropic">
    <density>...</density>
    <elastic>
      <e>...</e>
      <nu>...</nu>
    </elastic>
  </material>
  <material name="orth1" type="orthotropic">
    <density>...</density>
    <elastic>
      <e1>...</e1>
      <e2>...</e2>
      <e3>...</e3>
```

```
        <g12>...</g12>
        <g13>...</g13>
        <g23>...</g23>
        <nu12>...</nu12>
        <nu13>...</nu13>
        <nu23>...</nu23>
      </elastic>
    </material>
    <material name="aniso1" type="anisotropic">
      <density>...</density>
      <elastic>
        <c11>...</c11>
        <c12>...</c12>
        <c13>...</c13>
        <c14>...</c14>
        <c15>...</c15>
        <c16>...</c16>
        <c22>...</c22>
        <c23>...</c23>
        <c24>...</c24>
        <c25>...</c25>
        <c26>...</c26>
        <c33>...</c33>
        <c34>...</c34>
        <c35>...</c35>
        <c36>...</c36>
        <c44>...</c44>
        <c45>...</c45>
        <c46>...</c46>
        <c55>...</c55>
        <c56>...</c56>
        <c66>...</c66>
      </elastic>
    </material>
    ...
    <lamina name="lamina1">
      <material>orth1</material>
      <thickness>...</thickness>
    </lamina>
    ...
  </materials>
```

2.4.2 Layups

In general, there are three ways to define a layup, explicit list, stacking sequence code and ply (lamina) percentage code. For the explicit list, a laminate is laid onto the base line from the first layer in the list to the last one, in the direction given by the user. For the stacking sequence and ply percentage code, the layup starts from left to the right. User should pay attention to the relations among the base line direction, elemental frame $\mathbf{y}$ and fiber orientation θ_3 of each layer, as shown in Fig. 2.10. Change of direction of the base line will change the elemental frame $\mathbf{y}$ as defined, which will further require the user to change the fiber orientations accordingly, even though nothing changes physically. All layup information are included in one XML file. A template of this file can be found in Listing 2.6.

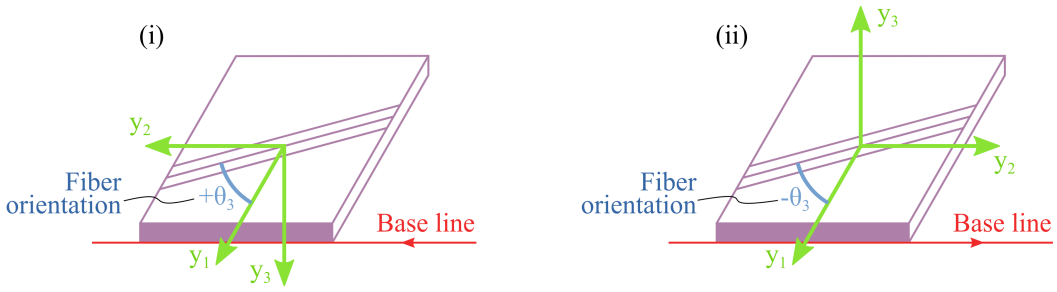

Figure 2.10: Relations among the base line direction, elemental frame **y** and fiber orientation (Note y_2 is parallel to the base line direction).

Listing 2.6: A template for the layup input file

```
<layups>
  <layup name="layup1" method="explicit list">
    <layer lamina="lamina1">...</layer>
    ...
  </layup>
  <layup name="layup2" method="stack sequence">...</layup>
  <layup name="layup3" method="ply percentage">...</layup>
  ...
</layups>
```

Explicit list

This method requires user to write down the lamina name, fiber orientation and number of successive laminas with the same fiber orientation, layer by layer. A template for one layer is shown below.

```
<layer lamina="lamina_name">angle:stack</layer>
```

`lamina_name` is defined in the material and lamina file. `angle` is the fiber orientation defined as shown in Fig. 2.10. `stack` is the number of successive laminae with the same fiber orientation. Some notes are:

- Default values are 0 for `angle` and 1 for `stack`;
- Omitted values are replaced by default values;
- If there is only one number presented in the `<layer>` element, then it is read in as `angle`, not `stack`, which is 1 by default;
- Empty input equals to `0:1`.

An example for the layup shown in Fig. 2.9 is given in Listing 2.7 and Listing 2.8.

Stacking sequence code

This method requires users to provide one lamina name and the stacking sequence code. A template is shown below.

```
<layup name="..." method="stack sequence">
  <lamina>...</lamina>
  <code>...</code>
</layup>
```

Code	Complete Sequence
`[0/90]2s`	0, 90, 90, 0, 0, 90, 90, 0
`[(45/-45):2/0:2]s`	45, -45, 45, -45, 0, 0, 0, 0, -45, 45, -45, 45

Some notes are:

- `method` is required in this case, which should be `stack sequence` or `ss` for short;
- The element `<code>` stores the sequence of fiber orientations. Some rules and examples are given below.
  - All fiber orientations should be put between a pair of square brackets `[]`;
  - Different fiber orientations are separated by slash `/`;
  - After the right bracket, user can add `ns` to indicate symmetry of the layup, where `n` is the number of the symmetry operations needed to generate the complete layup;
  - Successive laminae with the same fiber orientation can be expressed using colon like `angle:stack`, where `angle` is the fiber orientation and `stack` is the number of plies;
  - If a group of fiber orientations is repeated, user needs to close them in a pair of round brackets `()`.

Examples

Ply/Lamina percentage code

Listing 2.7: Example material and lamina input file for the layup shown in Fig. 2.9

```
<materials>
  <material name="square" type="orthotropic">
    <density>...</density>
    <elastic>...</elastic>
  </material>
  <material name="hexagon" type="orthotropic">
    <density>...</density>
    <elastic>...</elastic>
  </material>
  <lamina name="la_square_15">
    <material>square</material>
    <thickness>1.5</thickness>
  </lamina>
  <lamina name="la_hexagon_10">
    <material>hexagon</material>
    <thickness>1.0</thickness>
  </lamina>
</materials>
```

Listing 2.8: Example layup input file for the layup shown in Fig. 2.9

```
<layups>
  <layup name="layup_el" method="explicit␣list">
    <layer lamina="la_hexagon_10">45:2</layer>
    <layer lamina="la_hexagon_10">90</layer>
    <layer lamina="la_square_15"></layer>
    <layer lamina="la_hexagon_10">-45:2</layer>
```

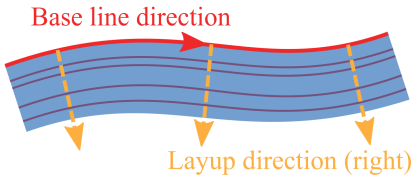

Figure 2.11: A typical segment in PreVABS and the relation between base line direction and layup direction.

```
      <layer lamina="la_square_15">0:2</layer>
      <layer lamina="la_hexagon_10">-45:2</layer>
      <layer lamina="la_square_15"></layer>
      <layer lamina="la_hexagon_10">90</layer>
      <layer lamina="la_hexagon_10">45:2</layer>
    </layup>
    <layup name="layup_ss" method="stack␣sequence">
      <lamina>la_square_15</lamina>
      <code>[(45/-45):2/0:4/90]2s</code>
    </layup>
  </layups>
```

2.5 Segments and connections

Segment is the key to a cross section in PreVABS. A schematic plot is shown in Fig. 2.11. A segment is a unique combination of base line and layup. The base line provides a reference for the position and direction of the layup. Layers can be laid to the left or right of the base line. The direction is defined as one's left or right, assuming one is walking along the direction of the base line.

Connection or joint is another key aspect. There are three types of connections considered in PreVABS, as shown in Fig. 2.12, two types of "V" shape and one of "T" shape. "V" shape connection is the one that two segments are joint end to end, while in "T" shape connection, two segments are joint end to side. Two "V" shape connections produce different geometries when the thicknesses of two connected segments are different. The first type tries to remove the step in the joint region, while the second type uses a bisector as the interface of two segments, which will create a step. By default, all connections have the "V2" shape. In the current version of PreVABS, the "V1" type is used for four specific groups of cross sections, "box", "L", "C" and "Z". In this case, user needs to specify the name of the cross section as "box", "L", "C" or "Z". More details about the `name` and other attributes of a cross section can be found in Section 2.7.

Level of a segment is used for dealing with structures with "T" shape connections, such as the web in an airfoil cross section. Since PreVABS does not require users to define connections types explicitly, this "T" shape joint is realized by putting segments into groups with different building priorities. The smaller the level number is, the higher the priority will be. Group with the highest priority will be created first. Then segments in the lower group will be built and connected to the previous group using the "T" type joint.

All segments and connections information are stored in a main input file. The complete format will be discussed later since the overall configurations are also included in this file, which will be explained in Section 2.7. A template for segments and connections are given in Listing 2.9. All segments are placed in a single element `<segments>`. Each `<segment>` element has two attributes, `name` and `level`, and two child elements, `<baseline>` and `<layup>`. The `<layup>` element has another attribute `direction` (see Fig. 2.11). Some notes are:

- `level` can be any positive integers;

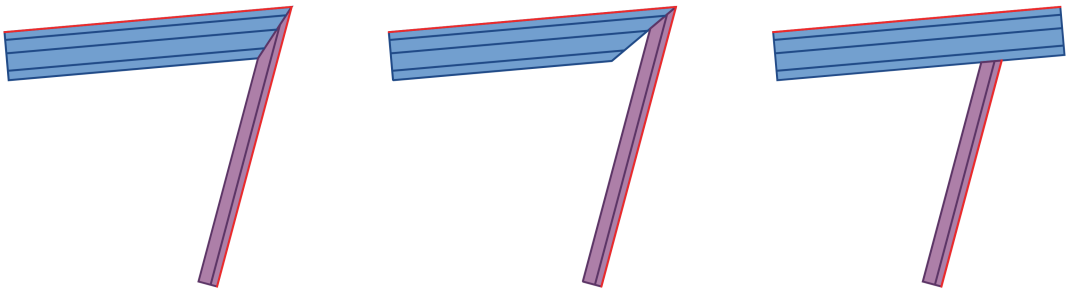

Figure 2.12: Three types of connections that can be created in PreVABS.

- The first `level` number must be 1 always;
- `level` numbers do not need to be consecutive. For example, the first `level` is 1 and the next `level` number can be 10;
- `direction` of a `layup` element can only be `left` or `right`;

Some extra notes regarding to the usage of connection and level are:

- In the current version of PreVABS, `<connections>` is used for defining leading and/or trailing edge in an airfoil only, and works only if the type of the cross section is `airfoil`.
- Defining leading and/or trailing edge in `<connections>` is for handling the complex (sharp, narrow, highly different layups) shapes of these two joints. If the shape is not complex, then users do not need to make these definitions, and the running time of PreVABS will be less.
- Creating shear webs in an airfoil cross section is the main purpose of the *Level* attribute. Hence, similar to this, a "web" in a `general` type cross section can also be defined this way.

Listing 2.9: A template for the segments and connections in a main input file

```
<segments>
  <segment name="segment1" level="1">
    <baseline>baseline1</baseline>
    <layup direction="right">layup1</layup>
  </segment>
  <segment name="segment2" level="10">
    <baseline>baseline2</baseline>
    <layup direction="left">layup2</layup>
  </segment>
  ...
  </segments>
  <connections>
  <connection name="leading_edge">
    <segment>segment1</segment>
    <segment>segment2</segment>
  </connection>
  <connection name="trailing_edge">
    <segment>segment3</segment>
    <segment>segment4</segment>
  </connection>
  ...
```

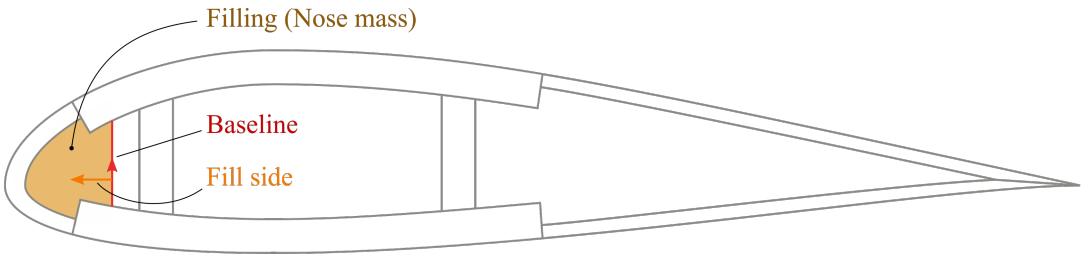

Figure 2.13: Definition of a filling as a nose mass in an airfoil type cross section.

```
</connections>
```

2.6 Other components

Besides segments, user can use one material to fill a region. A typical usage is a nose mass in an airfoil type cross section. A schematic plot is shown in Fig. 2.13. In PreVABS, fillings will be created after all segments are finished. Similar to a segment, a filling is defined by a baseline, a material (not layup), and the filling side of the material. The baseline defines part of the boundaries of the region. The rest are found by the program automatically.

Fillings are also defined in the main input file. A template is shown in Listing 2.10. All fillings are placed in a single element `<fillings>`. Each `<filling>` element has one attribute, `name`, and two child elements, `<baseline>` and `<material>`. The `<material>` element has an extra attribute, `fillside`, which can only be `left` or `right`. These have the same meaning as the `direction` attribute of a `<layup>` element in a `<segment>` definition.

NOTE In the current version, filling can only be used to create nose mass of an airfoil type cross section, and it is always filled to the left of the baseline.

Listing 2.10: A template for the fillings in a main input file

```
<fillings>
  <filling name="filling1">
    <baseline>baseline1</baseline>
    <material fillside="left">material1</material>
  </filling>
  ...
</fillings>
```

2.7 Overall configurations

The overall configurations contain two parts. First, user needs to tell PreVABS what the rest input files are, the base line file, the material file, and the layup file. Second, user needs to set the overall geometry and meshing parameters, including three transformation operations, mesh size and element type. The three transformation operations have been discussed in Section 2.2. Although the two frames $\mathbf{z}$ and $\mathbf{x}$ have some distances between them, the translate operation will actually move the geometry, instead of the coordinate system. Also, since the scale operation is done in the second place, the absolute distances of translation are defined in the initial frame $\mathbf{z}$. The rotation angle is counted from the positive direction of x_2 and follows the right-hand rule.

These overall configurations are also stored in the main input file. A template is shown in Listing 2.11. All included files are placed in the `<include>` element. The geometry and meshing configurations are placed in the `<general>` element. Some notes are:

- `path` in the `<include>` element is the relative path to the main input file;
- File extensions `.xml` can be omitted;
- `e2` and `e3` are distances before scaling. Default is $(0.0, 0.0)$ if omitted;
- Default for `<scale>` is 1.0 if omitted;
- Default for `<rotate>` is 0.0 if omitted;
- `<mesh_size>` is defined globally. The default size is the smallest thickness of layers used in the cross section;
- `<element_type>` can only be `linear` or `quadratic` (default).

NOTE Only triangular element is available in the current version.

Listing 2.11: A template for the overall configurations in a main input file

```
<include>
  <baseline>path/baseline_file_name</baseline>
  <material>path/material_file_name</material>
  <layup>path/layup_file_name</layup>
</include>
<general>
  <translate>e2  e3</translate>
  <scale>scaling_factor</scale>
  <rotate>angle</rotate>
  <mesh_size>a</mesh_size>
  <element_type>quadratic</element_type>
</general>
```

A top level cross section file stores all information discussed in this and previous sections. A template for this file is shown in Listing 2.12. The root element `<cross_section>` has two attributes, `name` (required) and `type` (optional), and four child elements, `<include>`, `<general>`, `<segments>` and `<connections>`. Some notes are:

- `name` will be used in file names of PreVABS outputs. For example, the `name` in the template is `cs1`, then the generated VABS input file name will be `cs1_vabs.dat` and Gmsh file name will be `cs1.msh`;
- `type` can only be `general` (default) or `airfoil`;

Listing 2.12: A template for the cross section file

```
<cross_section name="cs1" type="airfoil">
  <include>...</include>
  <general>...</general>
  <segments>...</segments>
  <connections>...</connections>
</cross_section>
```

2.8 Recovery

Once the 1D beam analysis is finished, those results can be used to recover local 3D strains and stresses at every point of the cross section. As stated in the VABS manual, the following data are required to carry out

the recover analysis at a selected location along the beam:

- 1D beam displacements;
- rotations in the form of a direction cosine matrix;
- sectional forces and moments;
- distributed forces and moments, and their first three derivatives with respect to x_1.

NOTE The current version of PreVABS only performs recovery for Euler-Bernoulli model and Timoshenko model. If you want to perform recovery for other models such as the Vlasov model, please refer to the VABS manual to modify the input file yourself.

These data are included in a `<recover>` element, added into the `<cross_section>` element in the main input file. The nine entries in the direction cosine matrix are flattened into a single array in the `<rotations>` element. This matrix is defined in Eq. (2.2). Note that displacements and rotations are needed only for recovering 3D displacements. A template for this part of data is shown in Listing 2.13. Numbers in this template are default values. Once this file is updated correctly, the VABS recover analysis can be carried out as shown in Section 1.3.

$$\mathbf{B}_i = C_{i1}\mathbf{b}_1 + C_{i2}\mathbf{b}_2 + C_{i3}\mathbf{b}_3 \quad \text{with} \quad i = 1, 2, 3 \tag{2.2}$$

where $\mathbf{B}_1$, $\mathbf{B}_2$, and $\mathbf{B}_3$ are the base vectors of the deformed triad and $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$ are the base vectors of the undeformed triad.

Listing 2.13: A template for the recover data in a cross section file

```
<cross_section name="cs1" type="airfoil">
  ...
  <recover>
    <displacements>0 0 0</displacements>
    <rotations>1 0 0 0 1 0 0 0 1</rotations>
    <forces>0 0 0</forces>
    <moments>0 0 0</moments>
    <distributed>
      <forces>0 0 0</forces>
      <forces_d1>0 0 0</forces_d1>
      <forces_d2>0 0 0</forces_d2>
      <forces_d3>0 0 0</forces_d3>
      <moments>0 0 0</moments>
      <moments_d1>0 0 0</moments_d1>
      <moments_d2>0 0 0</moments_d2>
      <moments_d3>0 0 0</moments_d3>
    </distributed>
  </recover>
</cross_section>
```

Chapter 3

Examples

3.1 Material Database

Table 3.1: Material properties

Isotropic										
Name	Density	E	ν							
mtr_1	1068.69	206.843	0.49							
Orthotropic										
Name	Density	E_1	E_2	E_3	G_{12}	G_{13}	G_{23}	ν_{12}	ν_{13}	ν_{23}
mat_1	1.353	20.59	1.42	1.42	0.87	0.87	0.696	0.30	0.30	0.34
Uni-directional FRP	1.86	37.0	9.00	9.00	4.00	4.00	4.00	0.28	0.28	0.28
Double-bias FRP	1.83	10.3	10.3	10.3	8.00	8.00	8.00	0.30	0.30	0.30
Gelcoat	1.83	10^{-8}	10^{-8}	10^{-8}	10^{-9}	10^{-9}	10^{-9}	0.30	0.30	0.30
Nexus	1.664	10.3	10.3	10.3	8.00	8.00	8.00	0.30	0.30	0.30
Balsa	0.128	0.01	0.01	0.01	0.0002	0.0002	0.0002	0.30	0.30	0.30

3.2 Box beam

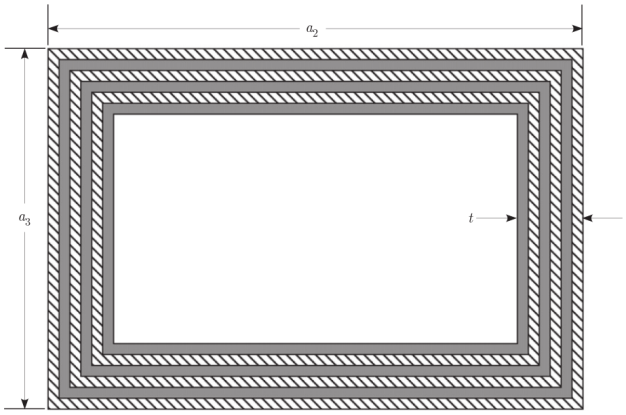

Figure 3.1: Cross section of the box beam [3].

This example is a thin-walled box beam whose cross section is depicted in Fig. 3.1. The width $a_2 = 0.953$ in, height $a_3 = 0.530$ in, and thickness $t = 0.030$ in. Each wall has six plies of the same composite material and the same fiber orientation of 15°. Material properties and layup scheme are listed in Table 3.1 and Table 3.2, respectively. Cross-sectional properties are given in Table 3.3 and compared with those in Ref. 3. The tiny differences are due to different meshes. Complete input files can be found in `examples\ex_box\`, including `box.xml`, `basepoints.dat`, `baselines.xml`, `materials.xml`, and `layups.xml`.

Table 3.2: Layups

Name	Layer	Material	Ply thickness (in)	Ply angle (deg)	Number of plies
layup1	1	mat_1	0.05	-15	6

Table 3.3: Results

Component	Value	Reference [3]
S_{11} (10^6 lb)	1.437	1.437
S_{22} (10^6 lb)	0.090	0.090
S_{33} (10^6 lb)	0.039	0.039
S_{14} (10^6 lb $\cdot$ in)	0.107	0.107
S_{25} (10^6 lb $\cdot$ in)	-0.052	-0.052
S_{36} (10^6 lb $\cdot$ in)	-0.056	-0.056
S_{44} (10^6 lb $\cdot$ in^2)	0.017	0.017
S_{55} (10^6 lb $\cdot$ in^2)	0.066	0.066
S_{66} (10^6 lb $\cdot$ in^2)	0.173	0.173

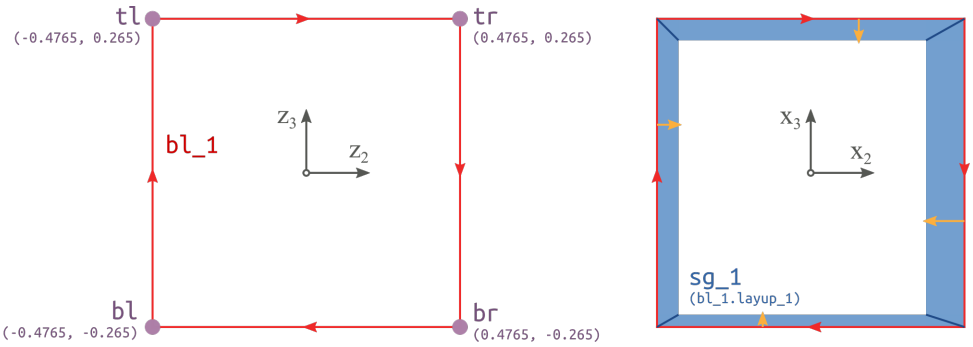

Figure 3.2: Base points, base lines and segments of the box beam cross section.

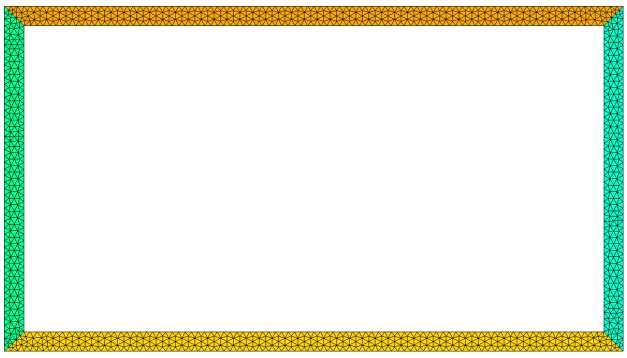

Figure 3.3: Meshed cross section viewed in Gmsh.

3.3 Pipe

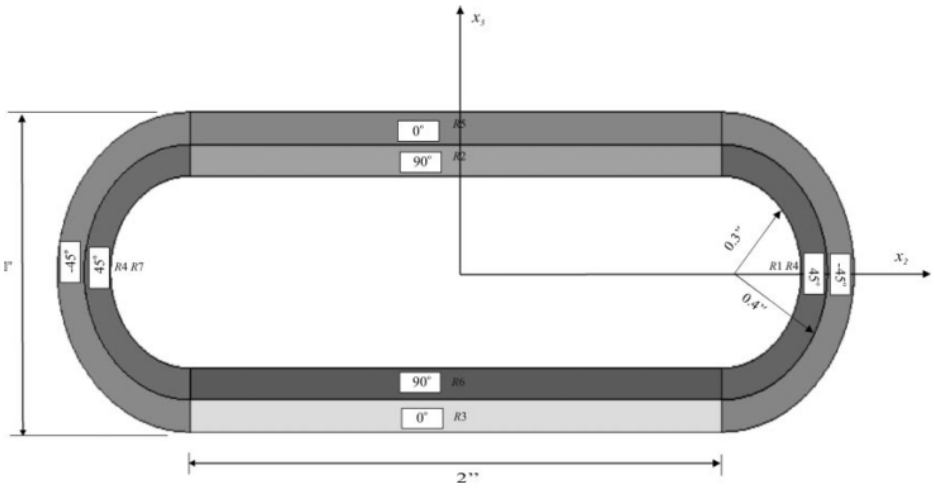

Figure 3.4: Cross section of the pipe [2].

This example has the cross section as shown in Fig. 3.4 [2]. This cross section has two straight walls and two half circular walls. $r = 1.0$ in. and other dimensions are shown in the figure. Each wall has the layup having two layers made from one material. Fiber orientations for each layer are also given in the figure. Material properties and layups are given in Table 3.1 Table 3.4. Cross-sectional properties are given in Table 3.5 and compared with the results from Ref. 2. The tiny differences are due to different meshes. Complete input files can be found in `examples\ex_pipe\`, including `pipe.xml`, `basepoints.dat`, `baselines.xml`, `materials.xml`, and `layups.xml`.

Table 3.4: Layups

Name	Layer	Material	Ply thickness (in)	Ply angle (deg)	Number of plies
layup_1	1	mat_1	0.1	0	1
	2	mat_1	0.1	90	1
layup_2	1	mat_1	0.1	-45	1
	2	mat_1	0.1	45	1

Table 3.5: Results

Component	Value	Reference [2]
S_{11} (10^6 lb)	10.385	10.389
S_{22} (10^6 lb)	0.786	0.784
S_{33} (10^6 lb)	0.331	0.329
S_{14} (10^6 lb $\cdot$ in)	0.098	0.098
S_{25} (10^6 lb $\cdot$ in)	-0.008	-0.008
S_{36} (10^6 lb $\cdot$ in)	-0.051	-0.052
S_{44} (10^6 lb $\cdot$ in^2)	0.688	0.687
S_{55} (10^6 lb $\cdot$ in^2)	1.881	1.882
S_{66} (10^6 lb $\cdot$ in^2)	5.385	5.390

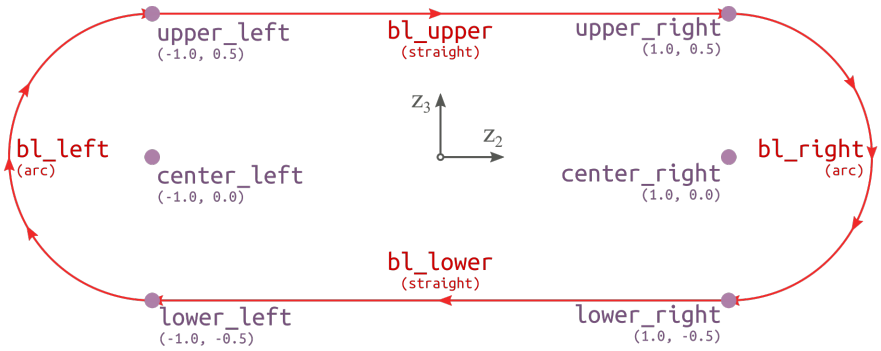

Figure 3.5: Base points and base lines of the pipe cross section.

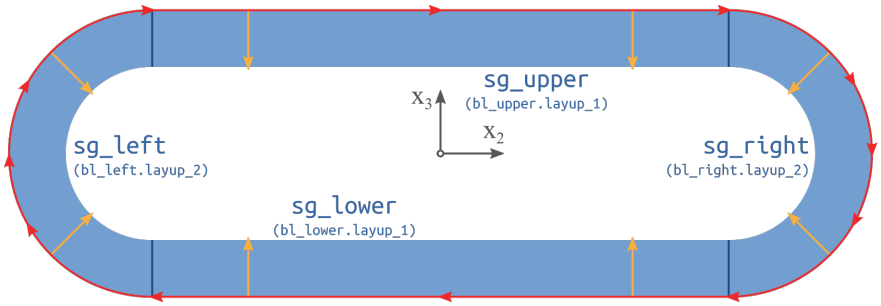

Figure 3.6: Segments of the pipe cross section.

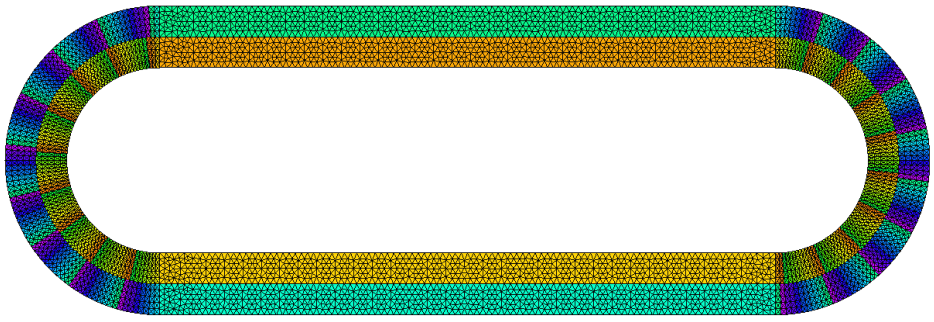

Figure 3.7: Meshed cross section viewed in Gmsh.

3.4 Circular tube

This example has a cross section of a simple circular shape with radius $r = 10$ m. This cross section geometry can be defined easily by a center and a radius. Material properties are given in Table 3.1. The layup is defined using the stacking sequence code $[\pm 45_2/0_2/90]_{2s}$. The result is given in Table 3.7. Complete input files can be found in `examples\ex_tube\`, including `tube.xml`, `basepoints.dat`, `baselines.xml`, `materials.xml`, and `layups.xml`.

Table 3.6: Layups

Name	Material	Stacking sequence
layup1	Nexus	$[\pm 45_2/0_2/90]_s$

Table 3.7: Results ($\times 10^9$ unit)

1 107.673	−0.000	−0.000	−0.000	−0.000	−0.000
−0.000	235.199	−0.000	0.000	0.000	0.000
−0.000	−0.000	235.199	0.000	0.000	0.000
−0.000	0.000	0.000	40 425.095	0.000	0.000
−0.000	0.000	0.000	0.000	48 194.408	−0.001
−0.000	0.000	0.000	0.000	−0.001	48 194.409

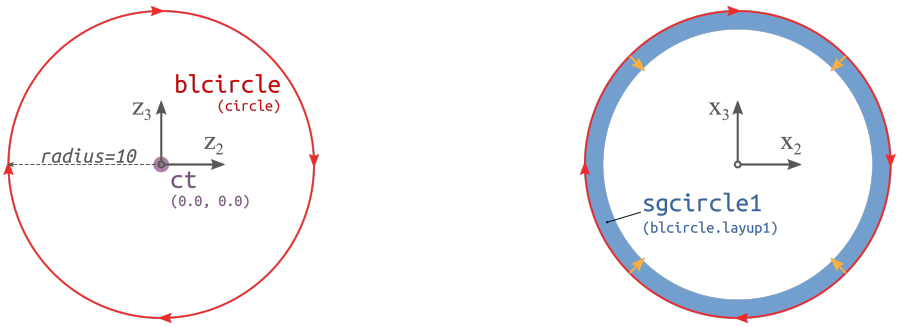

Figure 3.8: Base points, base lines and segments of the tube cross section.

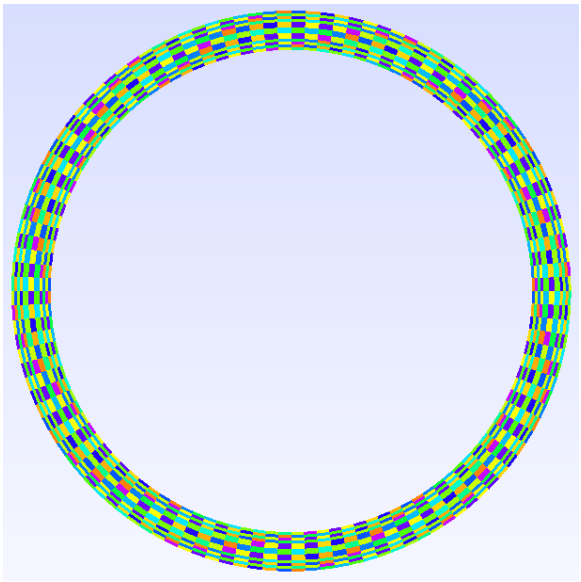

Figure 3.9: Meshed cross section viewed in Gmsh.

3.5 Channel

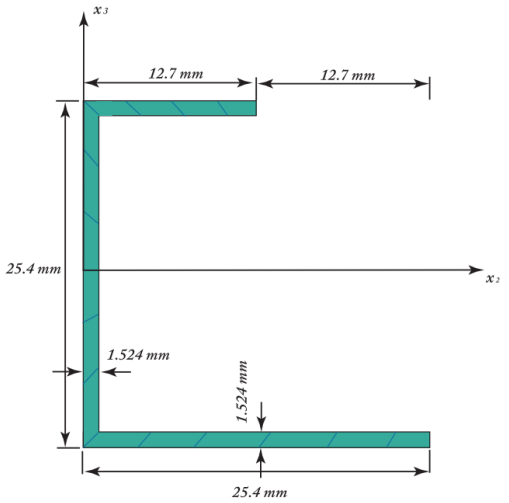

Figure 3.10: Cross section of the channel [1].

This example has a cross section of a highly heterogeneous channel. This cross section geometry can be defined as shown in Fig. 3.10 [1]. The isotropic material properties are given in Table 3.1. The layup is defined having a single layer with the thickness 0.001524 m. The result is shown in Table 3.9 and compared with those in Table 3.10 from Ref. 1. Complete input files can be found in `examples\ex_channel\`, including `channel.xml`, `basepoints.dat`, `baselines.xml`, `materials.xml`, and `layups.xml`.

Table 3.8: Layups

Name	Layer	Material	Ply thickness (in)	Ply angle (deg)	Number of plies
layup_1	1	mtr_1	0.001524	0	1

Table 3.9: Results ($\times 10^6$ unit)

19.056 207	0.000 000	0.000 000	0.000 000	−0.047 793	−0.132 477
0.000 000	2.803 959	0.241 721	0.021 279	0.000 000	0.000 000
0.000 000	0.241 721	2.145 841	−0.007 663	0.000 000	0.000 000
0.000 000	0.021 279	−0.007 663	0.000 209	0.000 000	0.000 000
−0.047 793	0.000 000	0.000 000	0.000 000	0.002 011	0.000 910
−0.132 477	0.000 000	0.000 000	0.000 000	0.000 910	0.001 946

Table 3.10: Results from Chen et al. [1] ($\times 10^6$ unit)

19.030 000	0.000 000	0.000 000	0.000 000	−0.047 780	−0.132 500
0.000 000	2.791 000	0.236 400	0.021 220	0.000 000	0.000 000
0.000 000	0.236 400	2.137 000	−0.007 679	0.000 000	0.000 000
0.000 000	0.021 220	−0.007 679	0.000 209	0.000 000	0.000 000
−0.047 780	0.000 000	0.000 000	0.000 000	0.002 010	0.000 910
−0.132 500	0.000 000	0.000 000	0.000 000	0.000 910	0.001 944

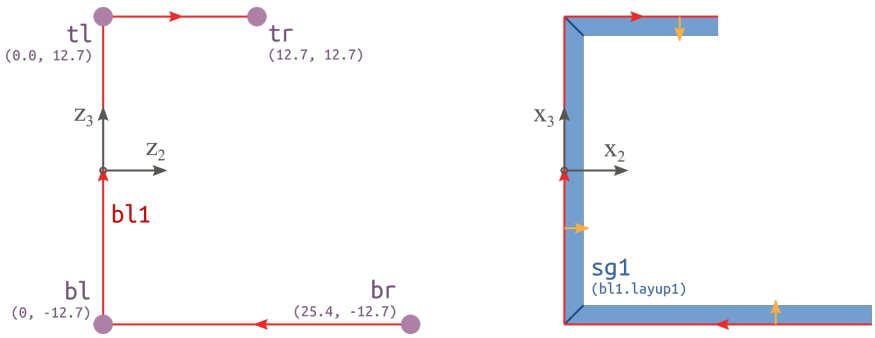

Figure 3.11: Base points, base lines and segments of the channel cross section.

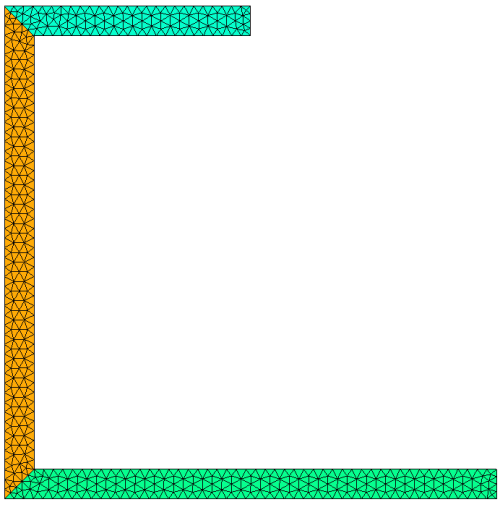

Figure 3.12: Meshed cross section viewed in Gmsh.

3.6 I beam

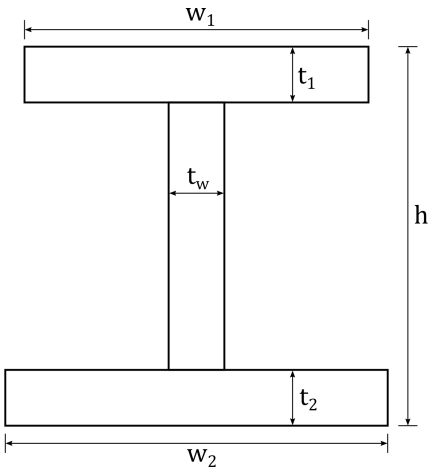

Figure 3.13: Cross section of the I-beam.

This example has an I-shape cross section. The dimensions shown in Fig. 3.13 are $w_1 = 2.0$ m, $w_2 = 3.0$ m, $h = 3.0$ m, $t_1 = 0.11$ m, $t_2 = 0.065$ m, $t_w = 0.08$ m. Materials and layups are given in Table 3.1 and Table 3.11. The effective stiffness matrix is listed in Table 3.12. Complete input files can be found in `examples\ex_ibeam\`, including `ibeam.xml`, `basepoints.dat`, `baselines.xml`, `materials.xml`, and `layups.xml`.

Table 3.11: Layups

Name	Layer	Material	Ply thickness (in)	Ply angle (deg)	Number of plies
layup1	1	iso5_1	0.03	90	2
	2	iso5_2	0.05	0	1
layup2	1	iso5_3	0.015	0	3
	2	iso5_4	0.02	90	1
layupweb	1	iso5_5	0.02	0	4

Table 3.12: Results ($\times 10^6$ unit)

2 749.000 000	−0.000 000	−0.000 000	−0.000 000	−1 945.000 000	−0.002 779
−0.000 000	1 362.000 000	0.000 431	1 645.000 000	0.000 000	−0.000 000
−0.000 000	0.000 431	0.047 290	0.000 520	0.000 000	−0.000 000
−0.000 000	1 645.000 000	0.000 520	1 990.000 000	0.000 000	0.000 000
−1 945.000 000	0.000 000	0.000 000	0.000 000	5 376.000 000	−0.000 527
−0.002 779	−0.000 000	−0.000 000	0.000 000	−0.000 527	1 173.000 000

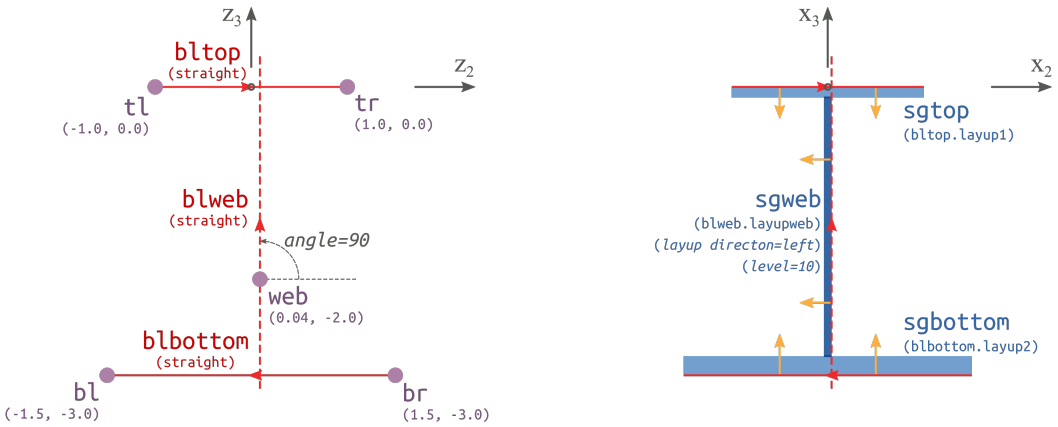

Figure 3.14: Base points, base lines and segments of the I-beam cross section.

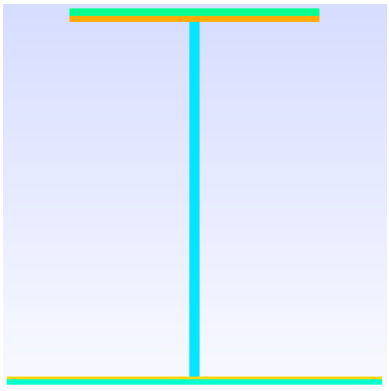

Figure 3.15: Meshed cross section viewed in Gmsh.

3.7 Airfoil

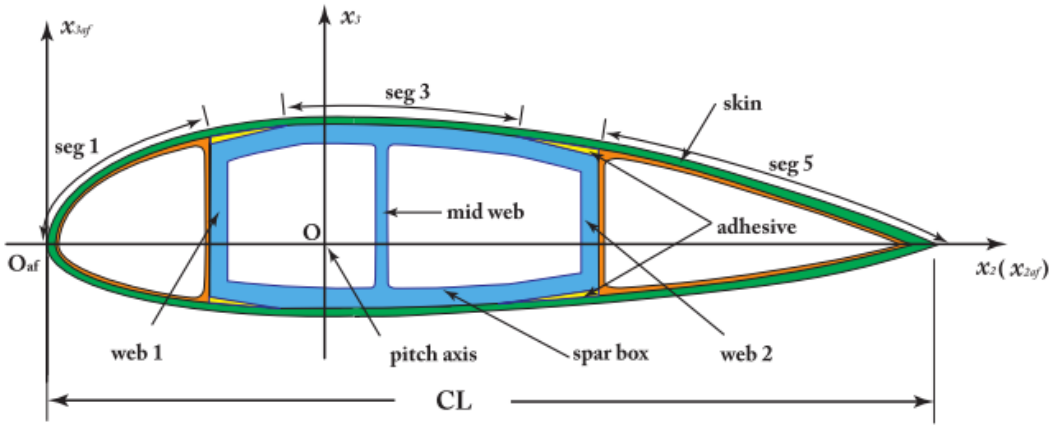

Figure 3.16: Sketch of a cross section for a typical wind turbine blade. [1]

This example demonstrates the capability of building a cross section having an airfoil shape, which is commonly seen on wind turbine blades or helicopter rotor blades. This example is also studied in Chen et al. [1]. A sketch of a cross section for a typical wind turbine blade is shown in Fig. 3.16. The airfoil is MH 104 `http://m-selig.ae.illinois.edu/ads/coord_database.html#M`. In this example, the chord length $CL = 1.9$ m. The origin O is set to the point at 1/4 of the chord. Twist angle θ is 0°. There are two webs, whose right boundaries are at the 20% and 50% location of the chord, respectively. Both low pressure and high pressure surfaces have four segments. The dividing points between segments are listed in Table 3.13. Materials are given in Table 3.1 and layups are given in Table 3.14. A complete 6×6 stiffness matrix is given in Table 3.16. Complete input files can be found in `examples\ex_airfoil\`, including `mh104.xml`, `basepoints.dat`, `baselines.xml`, `materials.xml`, and `layups.xml`.

Table 3.13: Dividing points

Between segments	Low pressure surface	High pressure surface
	(x, y)	(x, y)
1 and 2	$(0.004053940, 0.011734800)$	$(0.006824530, -0.009881650)$
2 and 3	$(0.114739930, 0.074571970)$	$(0.126956710, -0.047620490)$
3 and 4	$(0.536615950, 0.070226120)$	$(0.542952100, -0.044437080)$

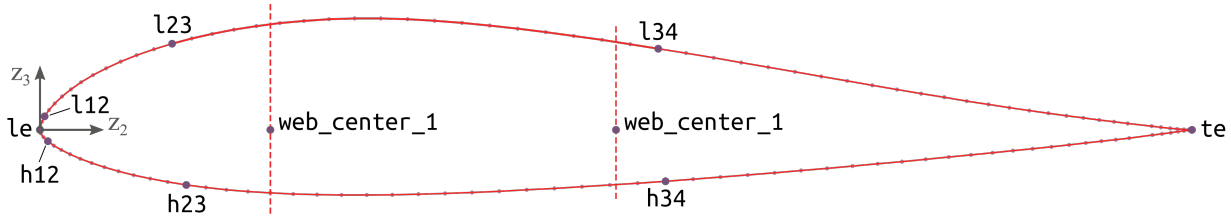

Figure 3.17: Base points of the airfoil.

NOTE The errors between the result and the reference are caused by the difference of modeling of the trailing edge. If reduce the trailing edge skin to a single thin layer, then the difference between the trailing edge shapes is minimized, and the two resulting stiffness matrices are basically the same, as shown in Fig. 3.21.

Table 3.14: Layups

Name	Layer	Material	Ply thickness (in)	Ply angle (deg)	Number of plies
layup_1	1	iso5_3	0.000381	0	1
	2	iso5_4	0.00051	0	1
	3	iso5_2	0.00053	20	18
layup_2	1	iso5_3	0.000381	0	1
	2	iso5_4	0.00051	0	1
	3	iso5_2	0.00053	20	33
layup_3	1	iso5_3	0.000381	0	1
	2	iso5_4	0.00051	0	1
	3	iso5_2	0.00053	20	17
	4	iso5_1	0.00053	30	38
	5	iso5_5	0.003125	0	1
	6	iso5_1	0.00053	30	37
	7	iso5_2	0.00053	20	16
layup_4	1	iso5_3	0.000381	0	1
	2	iso5_4	0.00051	0	1
	3	iso5_2	0.00053	20	17
	4	iso5_5	0.003125	0	1
	5	iso5_2	0.00053	0	16
layup_web	1	iso5_1	0.00053	0	38
	2	iso5_5	0.003125	0	1
	3	iso5_1	0.00053	0	38

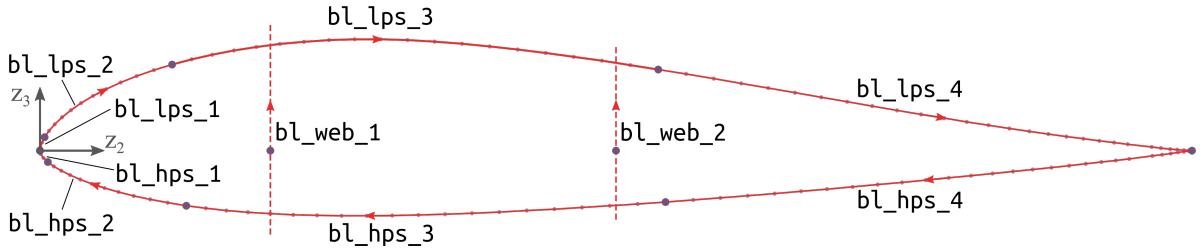

Figure 3.18: Base lines of the airfoil.

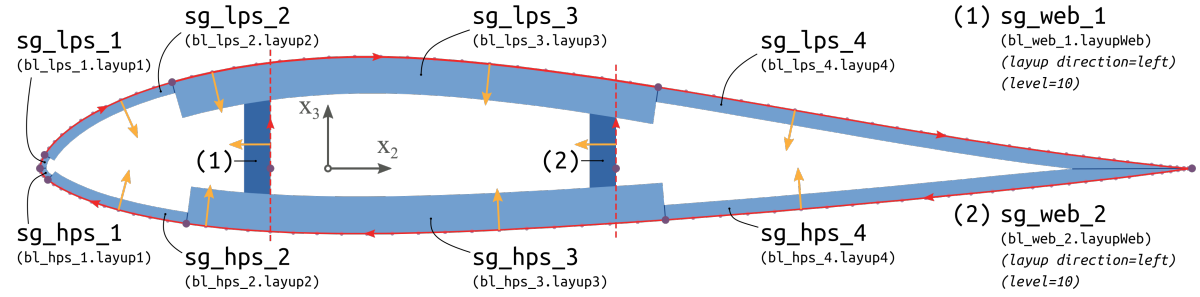

Figure 3.19: Segments of the airfoil.

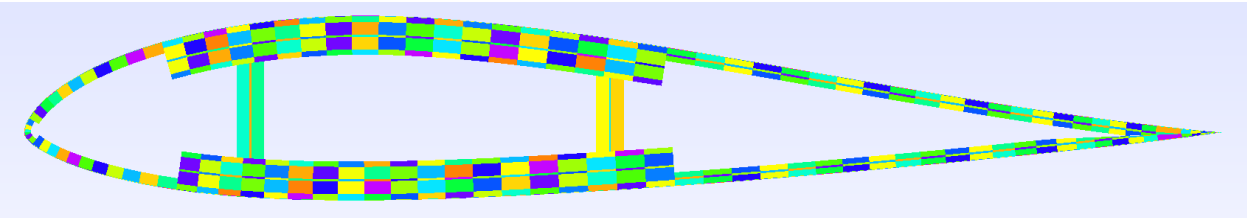

Figure 3.20: Meshed cross section viewed in Gmsh.

Table 3.15: Results ($\times 10^6$ unit)

2 387.225 489	6.318 173	10.681 638	−30.955 567	70.128 140	−549.880 179
6.318 173	438.834 409	−8.411 246	−21.634 829	15.691 493	−1.514 755
10.681 638	−8.411 246	25.108 801	−1.441 441	0.105 437	−6.706 095
−30.955 567	−21.634 829	−1.441 441	20.673 327	−2.185 282	1.112 912
70.128 140	15.691 493	0.105 437	−2.185 282	21.484 502	−9.228 771
−549.880 179	−1.514 755	−6.706 095	1.112 912	−9.228 771	476.310 792

Table 3.16: Results from reference ($\times 10^6$ unit) [1]

2 389.000 000	1.524 000	6.734 000	−33.820 000	−26.270 000	−473.600 000
1.524 000	433.400 000	−3.741 000	−0.293 500	15.270 000	0.383 500
6.734 000	−3.741 000	27.430 000	−0.045 920	−0.000 687	−4.742 000
−33.820 000	−0.293 500	−0.045 920	21.670 000	−0.062 790	1.430 000
−26.270 000	15.270 000	−0.000 687	−0.062 790	19.700 000	12.090 000
−473.600 000	0.383 500	−4.742 000	1.430 000	12.090 000	440.600 000

PreVABS					
2064472125.300	1511727.417	9718692.930	-29031396.199	66725788.529	-248509051.850
1511727.417	502135612.680	-279898.188	-20205923.352	27828620.058	1475369.601
9718692.930	-279898.188	20021647.481	-8727653.954	541726.919	-1738542.290
-29031396.199	-20205923.352	-8727653.954	13509499.286	-2247533.833	4604438.521
66725788.529	27828620.058	541726.919	-2247533.833	21337888.526	-6943938.047
-248509051.850	1475369.601	-1738542.290	4604438.521	-6943938.047	174131287.290
Reference					
2064518914.600	1508938.476	9687610.464	-29015161.717	66726945.698	-248650654.250
1508938.476	502009501.810	-300747.609	-20203039.227	27821546.775	1473462.671
9687610.464	-300747.609	20005103.430	-8714306.591	539559.226	-1733729.206
-29015161.717	-20203039.227	-8714306.591	13500297.561	-2246590.047	4620635.345
66726945.698	27821546.775	539559.226	-2246590.047	21338145.826	-6943939.608
-248650654.250	1473462.671	-1733729.206	4620635.345	-6943939.608	174159526.640
Difference = (PreVABS - Reference) / Reference					
-0.002%	0.185%	0.321%	0.056%	-0.002%	-0.057%
0.185%	0.025%	-6.933%	0.014%	0.025%	0.129%
0.321%	-6.933%	0.083%	0.153%	0.402%	0.278%
0.056%	0.014%	0.153%	0.068%	0.042%	-0.351%
-0.002%	0.025%	0.402%	0.042%	-0.001%	0.000%
-0.057%	0.129%	0.278%	-0.351%	0.000%	-0.016%

Figure 3.21: Comparison of stiffness matrices after modifying the trailing edge.

3.8 Airfoil (Recover)

This example continues from the previous one to demonstrate the dehomogenization analysis. It is assumed that a 1D beam analysis has been finished, and data of global deformations and loads have been added to the main input file correctly (Section 2.8. Suppose that the results in Table 3.17 are used and default values are kept for others. All files generated from the VABS homogenization analysis are kept in the same place as input files. Final visualization is required and the contour plot of one of the stress components is shown in Fig. 3.22. Complete files can be found in `examples\ex_airfoil_r\`, including `mh104.xml`, `basepoints.dat`, `baselines.xml`, `materials.xml`, `layups.xml`, `mh104_vabs.dat`, `mh104_vabs.dat.ech`, `mh104_vabs.dat.K`, `mh104_vabs.dat.opt`, `mh104_vabs.dat.v0`, `mh104_vabs.dat.v1S`, and `mh104_vabs.dat.v22`.

Table 3.17: Sectional forces and moments

Quantity	Value
F_1, N	1
F_2, N	2
F_3, N	3
M_1, Nm	4
M_2, Nm	5
M_3, Nm	6

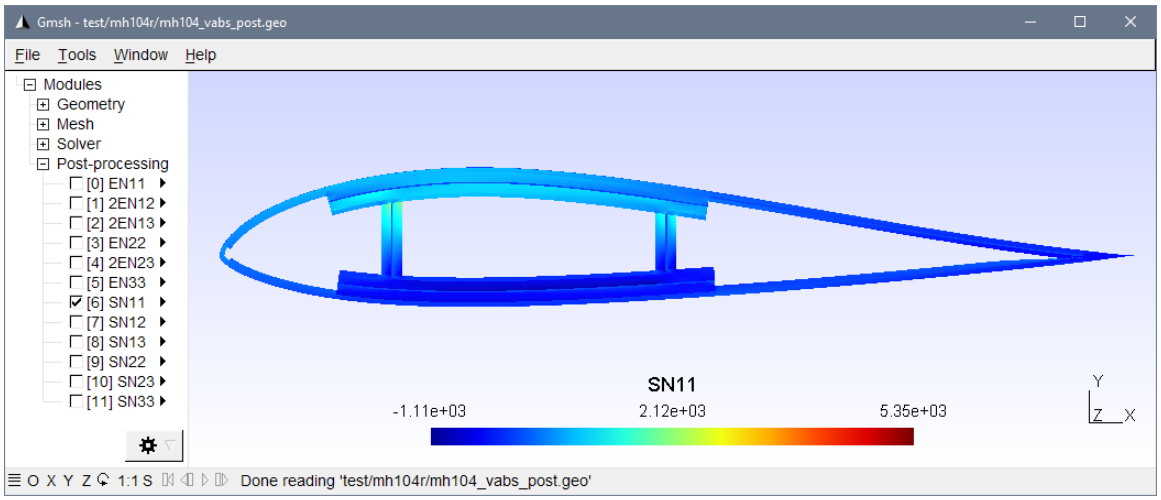

Figure 3.22: Contour plot of recovered nodal stress SN11.

Bibliography

[1] H. Chen, W. Yu, and M. Capellaro. A critical assessment of computer tools for calculating composite wind turbine blade properties. *Wind Energy*, 13(6):497–516, 2010.

[2] W. Yu and D. H. Hodges. Generalized timoshenko theory of the variational asymptotic beam sectional analysis. *Journal of the American Helicopter Society*, 50(1):46–55, 2005.

[3] W. Yu, D. H. Hodges, and J. C. Ho. Variational asymptotic beam sectional analysis – an updated version. *International Journal of Engineering Science*, 59:40–64, 2012.