PreVABS Documentation

Version 1.3.0

Su Tian, Xin Liu and Wenbin Yu

Mar 10, 2021

TABLE OF CONTENTS

CHAPTER

ONE

DOWNLOAD AND INSTALLATION

1.1 Prerequisites

Download and install VABS (with valid license). In the instructions below, `VABS_DIR` refers to the path to the VABS 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;

1.2 Install binary

Download PreVABS binary 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 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`, `~/.profile`, or `~/bash_login`;

CHAPTER

TWO

HOW TO RUN PREVABS

PreVABS is a command line based program which acts as a general-purpose preprocessor and postprocessor 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;

2.1 Quick start

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 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. 2.1. Three files are generated in the same location at this moment, a VABS input file `mh104.sg`, a Gmsh geometry file `mh104.geo`, a Gmsh mesh file `mh104.msh`, and a Gmsh option file `mh104.opt`. The geometry file is used to inspect errors when meshing cannot be accomplished. The latter three 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/).

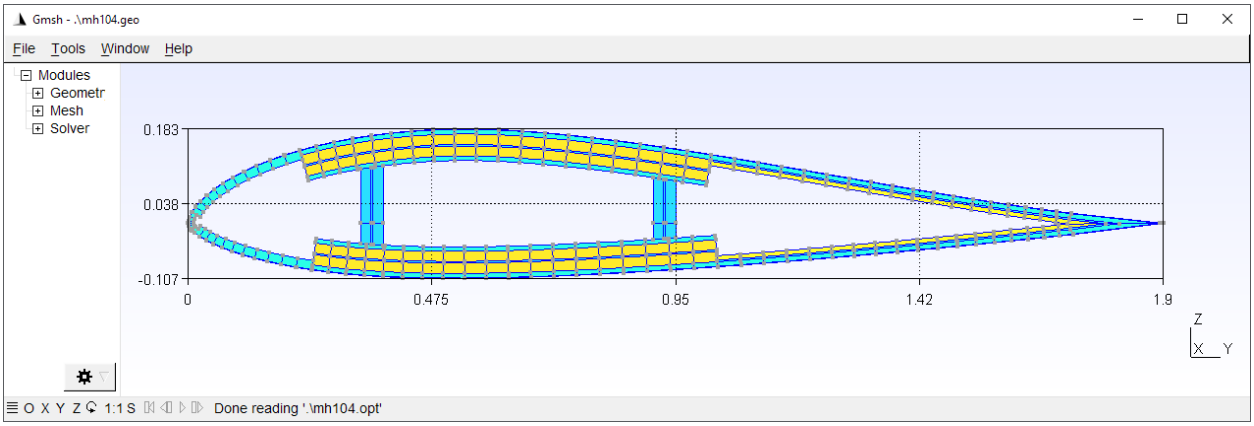

Figure 2.1: Cross section with meshes generated by PreVABS and visualized by Gmsh. Example: `examples\ex_airfoil\mh104.xml`.

2.2 Command line options

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.

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

Table 2.1: Command line options

Option	Description
-h	Build cross section and generate VABS/SwiftComp 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
-vabs	Use VABS (Default)
-sc	Use SwiftComp
-f	Initial failure strength analysis (SwiftComp only)
-fe	Initial failure envelope (SwiftComp only)
-fi	Initial failure indices and strength ratios (SwiftComp only)

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

2.3 Running cases

Some possible use cases are given below.

2.3.1 Case 1: Build cross section from parametric input files

```
prevabs -i <cross_section_file_name.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.

2.3.2 Case 2: Carry out homogenization without visualization

```
prevabs -i <cross_section_file_name.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 `VABS` *cross_section_vabs.dat*.

2.3.3 Case 3: Recover 3D stress/strain and plot

```
prevabs -i <cross_section_file_name.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. 2.2). An example of the recover analysis can be found in this example.

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: section-recover.

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

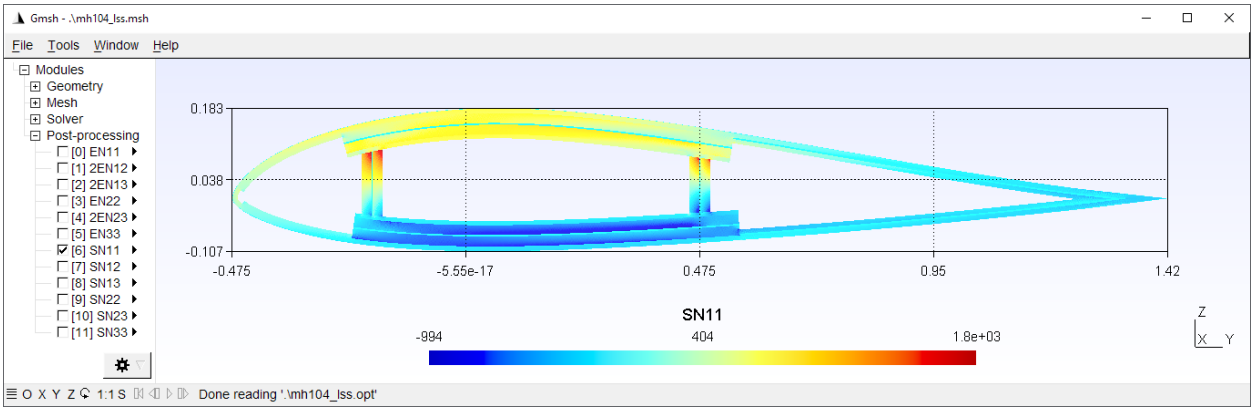

Figure 2.2: Visualization of strains and stresses in Gmsh.

CHAPTER

THREE

TUTORIAL

This tutorial provides a walkthrough for how to prepare input files for PreVABS given a cross section design.

3.1 The design of a cross section

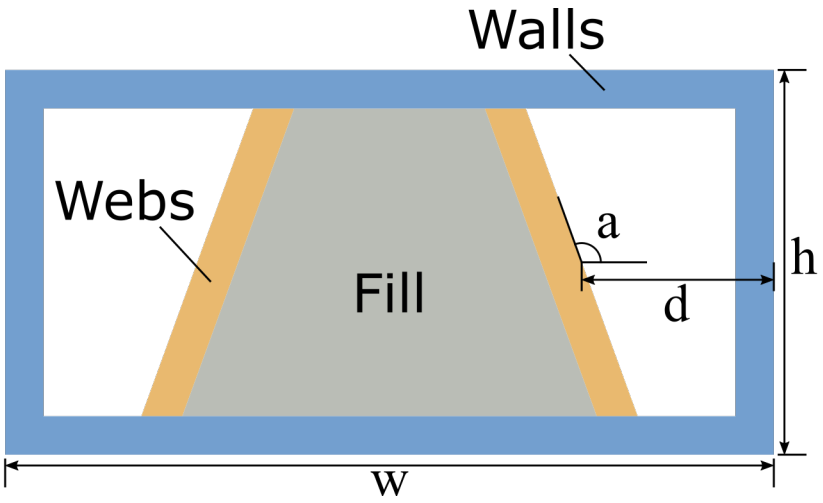

Figure 3.1: A box-beam cross section.

The box-beam cross section shown in Fig. 3.1 will be used in this tutorial. All four walls and two webs are made from composite laminates. The two webs are symmetric about a middle vertical line. They are inclined such that the upper portion of both webs are closer to each other than the lower portion. The central space enclosed by the top and bottom walls and two webs is filled with an isotropic material.

The overall shape is described using four parameters, with the following values:

- The width $w = 4$ m;
- The height $h = 2$ m;
- The distance $d = 1$ m;
- The angle $a = 100°$.

The materials used in this cross section are listed in Table 3.1 and Table 3.2. The layups used in this cross section are listed in Table 3.3.

Table 3.1: Isotropic material properties

Name	Density	E	ν
	$\mathrm{kg/m^3}$	10^3 Pa	
m0	1.00	25.00	0.30

Table 3.2: Orthotropic material properties

Name	Density	E_1	E_2	E_3	G_{12}	G_{13}	G_{23}	ν_{12}	ν_{13}	ν_{23}
	10^3 $\mathrm{kg/m^3}$	GPa	GPa	GPa	GPa	GPa	GPa			
m1	1.86	37.00	9.00	9.00	4.00	4.00	4.00	0.28	0.28	0.28
m2	1.83	10.30	10.30	10.30	8.00	8.00	8.00	0.30	0.30	0.30

Table 3.3: Layups

Component	Name	Material	Ply thickness	Orientation	Number of plies
			m	degree	
Walls	layup_1	m1	0.02	45	1
		m1	0.02	-45	1
		m2	0.05	0	1
		m1	0.02	0	2
Webs	layup_2	m2	0.05	0	1
		m1	0.02	0	3
		m2	0.05	0	1

3.2 Steps of preparing input files

In PreVABS, a cross section is composed of components, each of which can either be a laminate or a fill. The laminate type component is composed of segments with different layups. For a realistic cross section, there may be several different ways to decompose it, which may need different information in the input files to build the model.

Hence, the first step is to decide a way of decomposition of the cross section.

For this example, the first step is trivial. The cross section can be decomposed into three components: the walls, the webs, and the fill.

At the same time, the order of creating components should also be decided. This is done by figuring out the dependence relationships between components. For this example, the exact shape of each web depends on the shape of the walls, and the shape of the fill depends on both the webs and the walls. Then the order of creation should be: first creating the walls, then the webs, and at last the fill, as shown in Fig. 3.2.

Figure 3.2: Order of components creation.

The next step is to prepare various input files based on all design parameters listed above. In this example, all files are in the XML format. A brief summary of these input files is listed below.

- A baseline file (*baselines.xml*), storing definitions of geometric elements.

- A material file (*MaterialDB.xml*), storing definitions of materials and laminae.
- A layup file (*layups.xml*), storing definitions of layups.
- A main cross section file (*box.xml*), storing definitions of components and other configurations of modeling and analysis.

All files can have arbitrary file names and be placed at any working directory, except the material database, which must be named as *MaterialDB.xml* and placed at the same location as where the PreVABS executable is.

3.2.1 Prepare geometric elements

As shown in Fig. 3.3, seven points are used to define the shape of the cross section. Points p1 to p4 define the walls, p5 and p6 define the webs, and p0 indicates the space which should be filled with some material. The origin of the coordinate system is placed at the centroid of the rectangular. Based on the design parameters w, h and d, coordinates of all points are found and listed in Table 3.4.

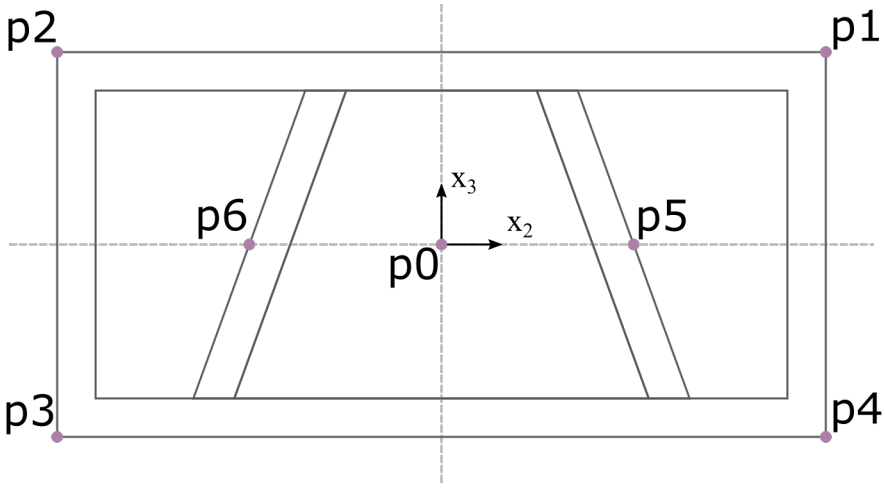

Figure 3.3: Key points defining the shape of the cross section.

Table 3.4: Key points

Name	Coordinate
p0	(0, 0)
p1	(2, 1)
p2	(-2, 1)
p3	(-2, -1)
p4	(2, -1)
p5	(1, 0)
p6	(-1, 0)

Base lines are created based on key points defined above. As shown in Fig. 3.4, three lines are created. Line 1 is defined by connecting the four points (p1 -> p2 -> p3 -> p4 -> p1). Line 2 and 3 are defined by a single point with an incline angle. For line 2, it is the point p5 with an angle of 100 degrees. For line 3, it is the point p6 with an angle of 80 degrees.

The direction of each base line is important. It is related with how the laminate is created for each segment, and how the local coordinate system is defined for each element.

The completed input file for geometry is shown in Listing 3.1.

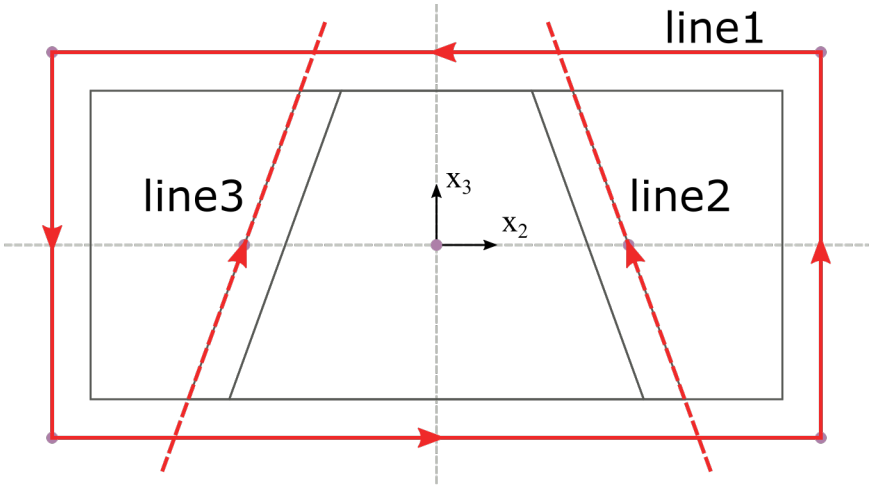

Figure 3.4: Base lines defining the shape of the cross section.

Listing 3.1: Input file for geometric elements (*baseline.xml*).

```
<baselines>
  <basepoints>
    <point name="p0">0  0</point>
    <point name="p1">2  1</point>
    <point name="p2">-2  1</point>
    <point name="p3">-2  -1</point>
    <point name="p4">2  -1</point>
    <point name="p5">1  0</point>
    <point name="p6">-1  0</point>
  </basepoints>
  <baseline name="line1" type="straight">
    <points>p1,p2,p3,p4,p1</points>
  </baseline>
  <baseline name="line2" type="straight">
    <point>p5</point>
    <angle>100</angle>
  </baseline>
  <baseline name="line3" type="straight">
    <point>p6</point>
    <angle>80</angle>
  </baseline>
</baselines>
```

3.2.2 Prepare materials and layups

Material data are stored in the material database. As stated above, this file must be named as *MaterialDB.xml* and placed at the directory where the PreVABS executable is. The format of this file is shown in Listing 3.2. Arrangement of data under the `<elastic>` element are different for different material types, which can be isotropic, orthotropic, or anisotropic.

Another part of the file is the lamina data. A lamina is a unique combination of material and ply thickness. For this example, according to the layup table (:Table 3.3) given above, there are two laminae. One has material m1 and thickness 0.02, and another one has material m2 and thickness 0.05. It is the lamina, instead of the material, that will be used to define each layer of the layup.

Listing 3.2: Input file for materials (*MaterialDB.xml*).

```
<materials>
  <material name="m0" type="isotropic">
    <density>1.0</density>
    <elastic>
      <e>25.0e3</e>
      <nu>0.3</nu>
    </elastic>
  </material>
  <!-- ========================================================== -->
  <material name="m1" type="orthotropic">
    <density>1.86E+03</density>
    <elastic>
      <e1>3.70E+10</e1>
      <e2>9.00E+09</e2>
      <e3>9.00E+09</e3>
      <g12>4.00E+09</g12>
      <g13>4.00E+09</g13>
      <g23>4.00E+09</g23>
      <nu12>0.28</nu12>
      <nu13>0.28</nu13>
      <nu23>0.28</nu23>
    </elastic>
  </material>
  <lamina name="la_m1_002">
    <material>m1</material>
    <thickness>0.02</thickness>
  </lamina>
  <!-- ========================================================== -->
  <material name="m2" type="orthotropic">
    <density>1.83E+03</density>
    <elastic>
      <e1>1.03E+10</e1>
      <e2>1.03E+10</e2>
      <e3>1.03E+10</e3>
      <g12>8.00E+09</g12>
      <g13>8.00E+09</g13>
      <g23>8.00E+09</g23>
      <nu12>0.30</nu12>
      <nu13>0.30</nu13>
      <nu23>0.30</nu23>
    </elastic>
  </material>
  <lamina name="la_m2_005">
    <material>m2</material>
    <thickness>0.05</thickness>
  </lamina>
</materials>
```

Layup information is stored in a separate file. Based on the layup table, the input file can be prepared with the format shown in Listing 3.3. The order of the layers defines the laying sequence from the base line. The number in the `<layer>` element stands for the orientation. If there is a colon, then the number behind it stands for the number of plies in this layer. If there is no number at all, then by default this layer has only one ply with orientation of 0 degree.

Listing 3.3: Input file for layups (*layups.xml*).

```
<layups>
  <layup name="layup1">
    <layer lamina="la_m1_002">45</layer>
    <layer lamina="la_m1_002">-45</layer>
    <layer lamina="la_m2_005">0</layer>
    <layer lamina="la_m1_002">0:2</layer>
  </layup>
  <layup name="layup2">
    <layer lamina="la_m2_005"></layer>
    <layer lamina="la_m1_002">0:3</layer>
    <layer lamina="la_m2_005"></layer>
  </layup>
</layups>
```

3.2.3 Create components

There are two types of components in PreVABS, laminate and fill. For this example, based on the decomposition we did at the begining of this section, there are three laminate-type components and one fill-type component. Besides, the sequence of creating components is important as stated at the beginning of this section. This is defined by declaring which components are needed before creating the current one.

Each laminate-type component is composed of one or more segments. Each segment is a unique combination of a base line and a layup. The layup can be created on either side of the base line. This is controlled by an attribute `direction`, which can be either left or right. This can be understood as the left- or right-hand side when walking along the base line, as shown in Fig. 3.5.

As for the ordering, the walls should be completed first, since its inner boundary is needed to trim the base lines of both webs. Hence, each web component depends on the walls component, as shown in Listing 3.4.

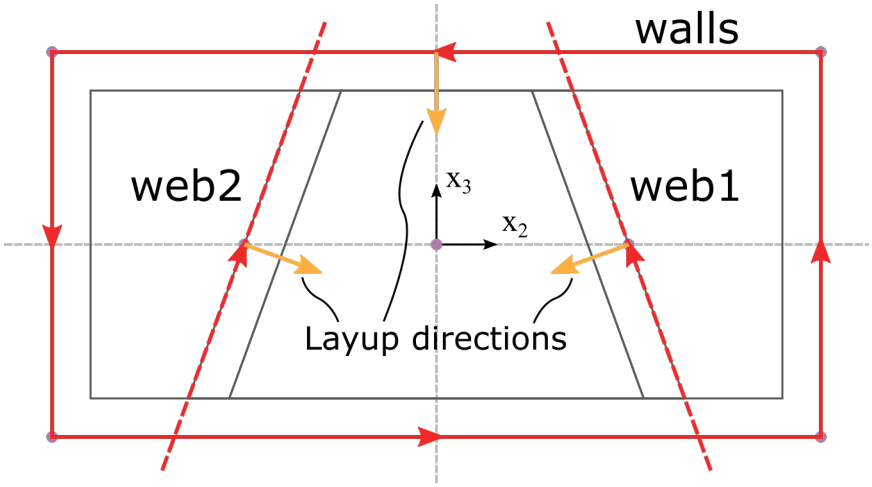

Figure 3.5: Layup directions for each segment.

Listing 3.4: Input elements for the laminate-type components

```
<component name="walls">
  <segment>
```

(continues on next page)

(continued from previous page)

```
    <baseline>line1</baseline>
    <layup>layup1</layup>
  </segment>
</component>
<component name="web1" depend="walls">
  <segment>
    <baseline>line2</baseline>
    <layup>layup2</layup>
  </segment>
</component>
<component name="web2" depend="walls">
  <segment>
    <baseline>line3</baseline>
    <layup direction="right">layup2</layup>
  </segment>
</component>
```

The fill-type component is defined by a point and a material. Here, the point p0 is used to indicate that the material should be filled in the space in the middle enclosed by the walls and both webs. At the same time, the dependent components are also decided, as shown in Listing 3.5.

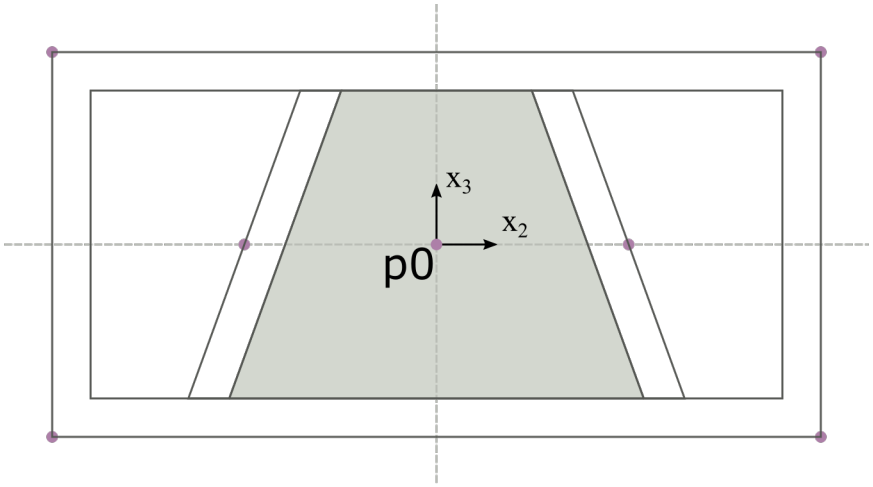

Figure 3.6: The fill-type component.

Listing 3.5: Input elements for the fill-type component

```
<component name="fill" type="fill" depend="walls,web1,web2">
  <location>p0</location>
  <material>m0</material>
</component>
```

3.2.4 Set other configurations and complete the main input file

Besides the definitions of components, the main input file of the cross section contains several other required or optional settings.

The first part is the <include> settings, which is required. This contains names of the base lines and layups files.

The second part is the <analysis> settings, which is optional. This contains configurations used by VABS for the cross-sectional analysis. For this example, the <model> setting is set to 1, which means that the Timoshenko beam model will be used and the 6x6 stiffness matrix will be calculated.

The last part is the <general> settings, which is optional. This part contains global configurations for the shape and meshing of the cross section. Here, the global mesh size is set to 0.02.

The completed main input file for the cross section is shown in Listing 3.6.

Listing 3.6: Input file for the cross section

```
<cross_section name="box">
  <include>
    <baseline>baselines</baseline>
    <layup>layups</layup>
  </include>
  <analysis>
    <model>1</model>
  </analysis>
  <general>
    <mesh_size>0.02</mesh_size>
  </general>
  <component name="walls">
    <segment>
      <baseline>line1</baseline>
      <layup>layup1</layup>
    </segment>
  </component>
  <component name="web1" depend="walls">
    <segment>
      <baseline>line2</baseline>
      <layup>layup2</layup>
    </segment>
  </component>
  <component name="web2" depend="walls">
    <segment>
      <baseline>line3</baseline>
      <layup direction="right">layup2</layup>
    </segment>
  </component>
  <component name="fill" type="fill" depend="walls,web1,web2">
    <location>p0</location>
    <material>m0</material>
```

(continues on next page)

(continued from previous page)

```
  </component>
</cross_section>
```

3.3 Execution and results

Once all input files are prepared, the cross section can be created and homogenized using the following command:

```
prevabs -i box.xml -h -v -e
```

If everything works successfully, Gmsh will be called and the cross section will be plotted as shown in Fig. 3.7 (the display of element edges is turned off in the figure for clarity), and VABS homogenization analysis will be carried out and effective beam properties can be found in the file `box.sg.K`. The effective Timoshenko stiffness matrix is listed in Table 3.5.

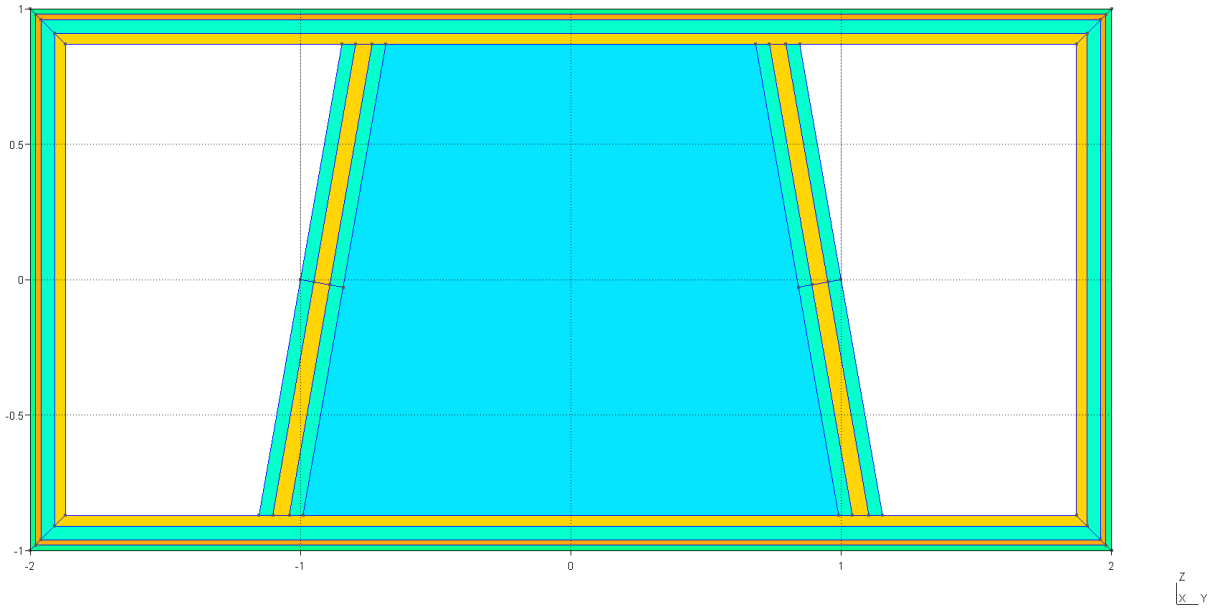

Figure 3.7: The cross section created by PreVABS and plotted by Gmsh.

Table 3.5: Timoshenko stiffness matrix in the file *box.sg.K*

3.991E+10	-1.662E+05	-4.037E+01	4.204E+07	-4.358E+03	2.867E+01
-1.662E+05	6.743E+09	3.945E+04	-2.918E+08	-1.577E+07	-4.150E+03
-4.037E+01	3.945E+04	6.165E+09	8.220E+03	-2.897E+03	-8.700E+06
4.204E+07	-2.918E+08	8.220E+03	1.972E+10	2.721E+05	3.303E+02
-4.358E+03	-1.577E+07	-2.897E+03	2.721E+05	2.173E+10	-3.025E+04
2.867E+01	-4.150E+03	-8.700E+06	3.303E+02	-3.025E+04	6.728E+10

CHAPTER

FOUR

CHANGE LOG

VERSION 1.3

- 1.3.0
  - Added a new capability to create base points using normalized parametric locations on a base line.
  - Added a new capability to assign local mesh size for filling components.

VERSION 1.2

- 1.2.0
  - Added a new capability to create layups from sublayups.
  - Added a new capability to create segments using normalized parametric locations on a base line.

VERSION 1.1

- 1.1.1 (10/28/2020)
  - Fixed the problem of unable to read recover analysis result files on Linux.
  - Fixed a problem when the segment is too short while the laminate is too thick.
- 1.1.0 (10/15/2020)
  - Added a new format for the input file. Now baselines and layups data are merged into the main input file, to reduce the number of input files needed.
  - Added a new material type 'lamina' accepting four numbers for elastic properties.
  - Added default small numbers for elastic properties for isotropic materials.
  - Updated the fill-type component for non-structural mass.

VERSION 1.0 (07/01/2019)

- Added capability to create quadratic triangle elements. In the main input file, under the xml element <*general*>, create a sub element <*element_type*> and set it to *quadratic* or *2*. Default is *linear*.
- Added a new output file **.txt* storing all running messages.
- Changed one of the layup methods tag name from 'explict list' ('el') to 'layer list' ('ll').
- Fixed the crashing issue caused by zero number of layers.
- Fixed the bug that sectional loads were read and written to the distributed loads when doing recovery using VABS.
- Fixed the bug that local lamina data does not overwrite global data.

VERSION 0.6 (07/01/2018)

- Added xml elements in the main input file for various options of VABS/SwiftComp execution.
- Added xml elements in the main input file for failure analysis of SwiftComp.
- Added a <*basepoints*> element in the baseline file. Now for simple shapes, users do not need to use an extra basepoint file, which can still be included for long point list.
- Added a material database file along with the executable. Now PreVABS will look for materials in this file by default.
- Changed Gmsh library to dynamic/shared library.

VERSION 0.5 (01/31/2018)

- Added the capability to create nose mass in an airfoil type cross section.
- Added the post-processing function to visualize the recovered strains and stresses in Gmsh.

VERSION 0.4 (12/04/2017)

- Added the capability to read stacking sequence code.
- Added the parameter to set the number of straight lines to approximate an arc or circle.
- Changed the Gmsh input file name to **.msh*, where * is the cross section name.

VERSION 0.3 (11/27/2017)

- Updated the manual.
- Changed the default behaviour of the command with input file specified only to preparing VABS input (without running VABS).
- Changed the element tag <*origin*> to <*translate*>. Now the two numbers in this element moves base points and base lines, instead of the origin.
- Changed the element tag <*rotation*> to <*rotate*>.
- Changed the 'level' of a segment from an element to an attribute, with default 1.
- Changed the 'layup_direction' of a segment from an element to an attribute of the layup, with default 'right'.
- Changed the material type 'engineering constants' to 'orthotropic'.
- Set the default element type as 'quadratic'.
- Set the default mesh size to the smallest layer thickness.

CHAPTER

FIVE

INTRODUCTION TO XML

1. The content of an XML file is composed of elements. Each element is marked by a pair of tags `<tag>...</tag>`.
2. The hierarchical structure of elements is important. At the same level of hierarchy, the arrangement order of elements is not important.
3. The tag names are keywords, which should not be changed.
4. Each element can have multiple attributes, which are `name="value"` pairs. The names are also keywords. The values must be surrounded by double quotes.

CHAPTER

SIX

REFERENCES

BIBLIOGRAPHY

[YU2012] Yu, W., Hodges, D. H. and Ho, J. C. (2012) 'Variational asymptotic beam sectional analysis – An updated version', International Journal of Engineering Science, 59, pp. 40–64.

[YU2005] Yu, W. and Hodges, D. H. (2005) 'Generalized Timoshenko Theory of the Variational Asymptotic Beam Sectional Analysis', Journal of the American Helicopter Society, 50(1), pp. 46–55.

[CHEN2010] Chen, H., Yu, W. and Capellaro, M. (2010) 'A critical assessment of computer tools for calculating composite wind turbine blade properties', Wind Energy, 13(6), pp. 497–516.