

Abaqus-VABS Interface Manual

Su Tian, Bo Peng and Wenbin Yu

March 7, 2019

Contents

- 1 INSTALLATION** **5**
- 1.1 Manual Installation 5

- 2 AIRFOIL AUTOMATION** **7**
- 2.1 Introduction 7
- 2.2 Preparation of Input Files 7
- 2.2.1 Control file 7
- 2.2.2 Head file 8
- 2.2.3 Airfoil main file 8
- 2.2.4 Shape file 9
- 2.2.5 Material file 11
- 2.2.6 Layup file 12
- 2.2.7 Foot file 13
- 2.3 Generation and Homogenization of the Airfoil Cross-Section 13
- 2.4 Dehomogenization 13
- 2.5 Visualization 14

1 INSTALLATION

1.1 Manual Installation

First of all, locate the home directory of Abaqus on your machine, for example:

C:\SIMULIA\Abaqus\6.13-4

which is defined as `$ABA_HOME`;

1. **Copy the code to a custom folder under `$ABA_HOME`.**

Create a new folder named `customApps` under `$ABA_HOME`;

Download file `Abaqus_VABS_GUI_code.zip` to `$ABA_HOME\customApps\` and unzip the file.

2. **Add the code to the `PYTHONPATH` of Abaqus.**

Open file `abaqus.aev` in a text editor, which should be located at `$ABA_HOME\SMA\site\`;

Edit the parameter `PYTHONPATH`;

Insert `$ABA_HOME/customApps/Abaqus_VABS_GUI`: just before `.:$PYTHONPATH` at the end; (Pay attention to the direction of the slash.)

Save.

3. **Add a custom command to Abaqus.**

Open file `abaqus.app` in a text editor, which should be located at `$ABA_HOME\SMA\site\`;

Append `vabs cae -custom VABSGUI` at the end of the command list;

Save.

4. **Run Abaqus-VABS GUI.**

In the command prompt, type `abaqus vabs` to enter the GUI.

User can still use original Abaqus by typing `abaqus cae` or other predefined commands.

2 AIRFOIL AUTOMATION

2.1 Introduction

Cross-sections with airfoil shape usually contain several parts, like surfaces, webs and fillings, and the surface part has several segments and each segment has dozens or hundreds of layers, which makes it very difficult to build the model by hand as what we have shown in the previous tutorials. The Abaqus-VABS GUI will help users to draw the cross-section and prepare the VABS input file automatically from airfoil data. Thus the main topic of this function is the preparation of data files. Once those files are ready, the whole process will be done by one click. A sample set of files are also given in the folder `airfoil` which the users can use as template to adapt for their own cross-sections.

2.2 Preparation of Input Files

User needs to provide 6 files: **control**, **head**, **main**, **shape**, **material** and **layup**, which are all XML files. All files should be placed in the same folder. Now we will describe each file in details.

2.2.1 Control file

This is the input file if user wants to generate meshed airfoil cross-section and run VABS through common line.

Listing 2.1: Example control file. (Control.xml)

```
<control>
  <name>test</name>
  <recover_name>test_recover</recover_name>
  <head>Head.xml</head>
  <airfoil>Airfoil.xml</airfoil>
  <foot>Foot.xml</foot>
  <input_only>0</input_only>
</control>
```

The **control** element is the root element, below which there are 4 subelements. The **name** element contains the project name defined by users. This name will be used for every file generated during the whole process, the Abaqus `.cae` file and VABS `.dat` input file, to name a few. The **recover_name** element contains the name used during recovering of local displacements, strains and stress. This element can be omitted when doing homogenization. The **head** element contains the name of the head file, which will be explained below. The **airfoil** element contains the name of the airfoil main file, which will be explained below. The **foot** element contains the name of the foot file, which will be explained below. This element can be omitted when doing homogenization. For all these file names, `.xml` can be omitted. The last **input_only** element

contains an integer, 0 or 1, which indicates whether user wants to run VABS or not. 1 means generating VABS input file only, without running VABS.

2.2.2 Head file

This file contains flags and parameters which will be written at the beginning of the VABS input file.

Listing 2.2: Example head file. (Head.xml)

```
<head>
  <timoshenko>0</timoshenko>
  <recover>0</recover>
  <thermal>0</thermal>
  <curve>0</curve>
  <oblique>0</oblique>
  <trapeze>0</trapeze>
  <vlasov>0</vlasov>
  <k>0.0 0.0 0.0</k>
  <cos>1.0 0.0</cos>
</head>
```

The **head** element is the root element, below which there are 9 subelements. To learn more details about each flag and parameter, please refer to the VABS manual.

2.2.3 Airfoil main file

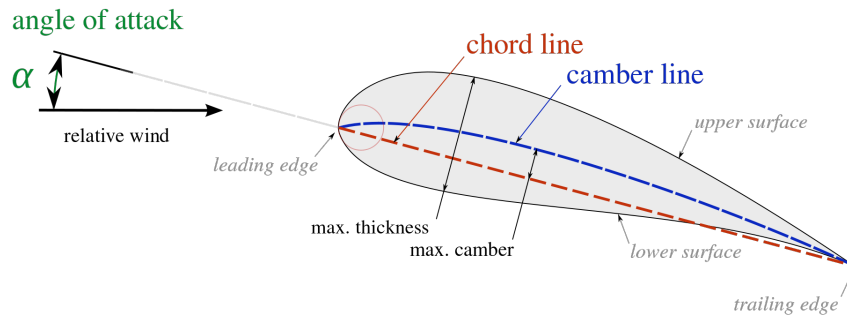


Figure 2.1: Typical airfoil. Source: Wikipedia.

This is the main input that will be used in the VABS GUI.

Listing 2.3: Example airfoil main file. (Airfoil.xml)

```
<project type="airfoil">
  <shapes>shape_filename</shapes>
  <materials>material_filename</materials>
  <layups>layup_filename</layups>
  <pitch_axis_yz>coord_y coord_z</pitch_axis_yz>
  <twisted_angle>angle</twisted_angle>
  <chord_length>length</chord_length>
  <flip>No</flip>
</project>
```

<project> This is the root element, which has an attribute `type="airfoil"`. Since the 'Read File' function mainly deals with cross-sections with airfoil shape, this attribute can be omitted for the current version. There are seven sub-elements below `<project></project>`.

<shapes> This element contains the name of the shape file, with file extensions ".xml" being appended or omitted.

<materials> This element contains the name of the material file, with file extensions ".xml" being appended or omitted.

<layups> This element contains the name of the layup file, with file extensions ".xml" being appended or omitted.

<pitch_axis.yz> This element indicates the normalized location of the pitch axis in the Y-Z plane, which is the cross-section plane. The two numbers are separated by spaces.

<twisted_angle> This element contains the angle in degree between the actual chord line and the horizontal line. In Fig.2.1, the "angle of attack" is what we mean by "twisted angle" here.

<chord_length> This element stores the actual length of the chord line.

<flip> This element is used to indicate whether user want to switch the orientation of the airfoil. By default, the leading edge is on the left.

2.2.4 Shape file

This file stores the shape data of the surfaces, webs and fillings, including the geometry information like coordinates of points and the segment division information like the dividing point number and the layup id that will be assigned to this segment. The structure is shown below.

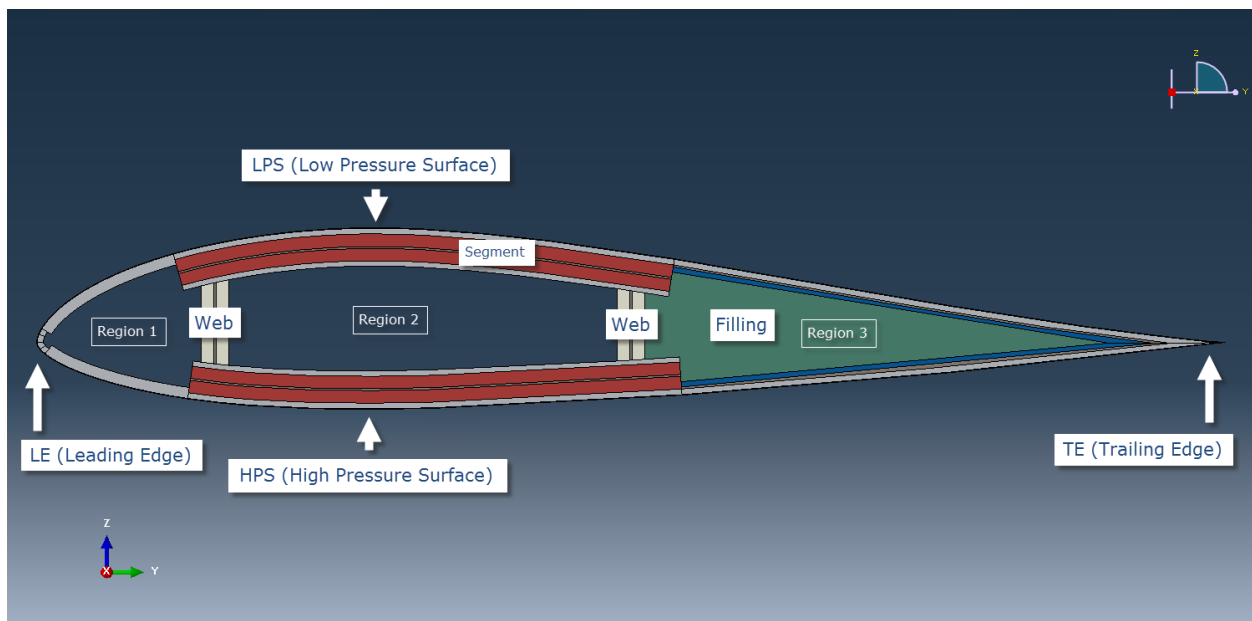


Figure 2.2: A schematic figure for the example airfoil, defining different parts, segments and regions.

Listing 2.4: Example shape file. (Shapes.xml)

```
<assembly type = "airfoil">
```

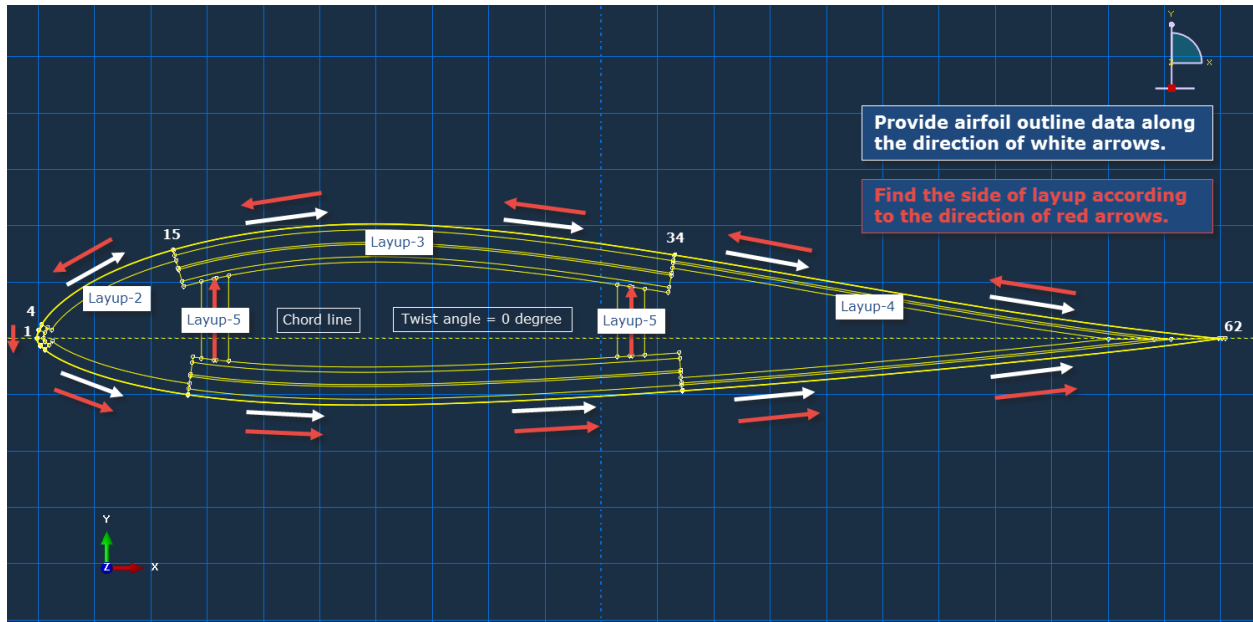


Figure 2.3: A schematic figure for the example airfoil, defining data points and layups.

```

<part structure = "surface">
  <baseline type = "spline" format = "lednicer">
    <lps>
      0 0
      0.00035937 0.0029595
      0.00162747 0.00696192
      ... ..
      9.97E-01 1.33E-04
      1.00E+00 0.00E+00
    </lps>
    <hps>
      0 0
      0.00062671 -0.00294872
      0.0030035 -0.00629665
      ... ..
      0.9970425 -0.00036119
      1 0
    </hps>
  </baseline>
  <layup side = "left">
    <lps>
      1 3 1
      ...
    </lps>
    <hps>
      1 3 1
      ...
    </hps>
  </layup>
</part>
<part structure = "web">
  <baseline>
    <y_z_angle>0.15 0.0 90.0</y_z_angle>
    ...

```

```

    </baseline>
    <layup side = "both">
    5
    ...
    </layup>
  </part>
  <part structure = "filling">
<region material = "6">3</region>
  ...
  </part>
</assembly>

```

There is a root element `<assembly></assembly>`, whose attribute “type” can be omitted for the current version. The child element is `<part></part>` and the attribute “structure” for each part tells what the part is, surface, web or filling. See Fig.2.2. For laminate-like part, surface and web, there are two sub-elements, `<baseline></baseline>` and `<layup></layup>`.

The “baseline” element of the “surface” part contains two sub-elements, `<lps></lps>` and `<hps></hps>`, which store the raw data of the airfoil low pressure surface and high pressure surface, respectively. Two attributes, “type” and “format” may be used in the future version and can be omitted here. The arrangement of data for both “lps” and “hps” is from (0,0) to (1,0), which is from the leading edge to the trailing edge. See Fig.2.3. The “baseline” element for the “web” part contains any number of sub-elements `<y_z_angle></y_z_angle>`, each of whom stands for a web. Here a web is represented by a straight line. The first two numbers are the normalized position of the baseline on the chord line and the last number is the angle in degree measured from the chord line segment near the leading edge to the upper part of the web baseline.

The “layup” element has an attribute “layup”, which can have the value “left”, “right” or “both”. The “side” is defined according to the direction of the surface baseline (Fig.2.3), which starts from the trailing edge, goes along the low pressure surface baseline to the leading edge and back to the trailing edge along high pressure surface baseline. So basically, if the “side” is “left”, then the plies are laid towards the inside of the airfoil. Thus if the surface baseline is the outmost profile and the leading edge is on the left, then the “side” should be “left”. The value “both” means doing the layup on both sides of the baseline, but will not double the layup. Or more precisely, this means that the baseline is at the middle of the layup. For the current version, this case can only handle symmetric layup. Otherwise, user needs to translate the baseline to one side and change some values in the file accordingly. The “layup” element for the “surface” part contains two sub-elements `<lps></lps>` and `<hps></hps>`. In each of them, there are several lines of data and each line represents a segment and has three numbers. The first two numbers are the starting and ending points for the segment and the last one is the layup id that will be defined in the layup file. The first number of the first line must be 1 and the second number of the last line has to be the total number of points for low pressure surface or high pressure surface. The “layup” element for “web” part is much simpler. Each line only contains the layup id for the web.

For the “filling” part, such as foam, has only one type of sub-element, `<region></region>`. The number stored between the two tags is region id, which is defined as follows. If there are two webs, then the inner space of the airfoil is divided into three regions. Starting from the leading edge, those regions are labeled as “1”, “2” and “3”. The attribute “material” stores the material id, which is defined in the material file. Here we make the assumption that the materials used for the fillings are homogeneous, could not generally anisotropic, with material properties given in the global coordinates.

2.2.5 Material file

The root element is `<materials></materials>` and each `<material></material>` sub-element store one material. For the current version, we can only deal with materials with density and elastic properties.

Listing 2.5: Example material file. (Materials.xml)

```

<materials>
  <material type = "ENGINEERING CONSTANTS">
    <id>1</id>
    <name>iso5_1</name>
    <density>1.860000E+03</density>
    <e1>3.7000E+10</e1>
    <e2>9.0000E+09</e2>
    <e3>9.0000E+09</e3>
    <g12>4.0000E+09</g12>
    <g13>4.0000E+09</g13>
    <g23>4.0000E+09</g23>
    <nu12>0.28</nu12>
    <nu13>0.28</nu13>
    <nu23>0.28</nu23>
  </material>
  <material>
    ...
  </material>
  ...
</materials>

```

Each material has a “type” attribute, which has the same definition as Abaqus, ISOTROPIC, ENGINEERING CONSTANTS, ORTHOTROPIC or ANISOTROPIC. For each material, user needs to provide a unique “id” and “name”. The “density” is optional. When omitted, it will use the default value 1.0. The components of elastic properties for each type are the same as those in Abaqus, and the arrangement of components will not be a problem. For ISOTROPIC, we have 2 components, “e” and “nu”. For ENGINEERING CONSTANTS, we have 9 components, “e1”, “e2”, “e3”, “g12”, “g13”, “g23”, “nu12”, “nu13” and “nu23”. For ORTHOTROPIC, we have 9 components, “d1111”, “d1122”, “d2222”, “d1133”, “d2233”, “d3333”, “d1212”, “d1313” and “d2323”. For ANISOTROPIC, we have 21 components, “d1111”, “d1122”, “d2222”, “d1133”, “d2233”, “d3333”, “d1112”, “d2212”, “d3312”, “d1212”, “d1113”, “d2213”, “d3313”, “d1213”, “d1313”, “d1123”, “d2223”, “d3323”, “d1223”, “d1323” and “d2323”.

2.2.6 Layup file

The root element is <layups></layups> and each <layup></layup> sub-element stores one layup.

Listing 2.6: Example layup file. (Layups.xml)

```

<layups>
  <layup>
    <id>1</id>
    <name>layup1</name>
    <data>
      0.0003810000000 3 0
      0.0005099999960 4 0
      0.0095400001224 2 20
    </data>
  </layup>
  ...
</layups>

```

For each layup, user need to provide a unique “id” and “name”. In the <data></data> sub-element, each line is a ply, where the first number is the thickness, the second one is the material id defined in materials.xml and the last one is the fiber orientation angle in degrees.

2.2.7 Foot file

This file contains the results from the global analysis of a beam. The root element is `<foot></foot>`. There are 9 subelements, containing global displacements, rotations, sectional forces, sectional moments, distributed forces, distributed moments, axial strains, twist and curvatures and derivatives of twist. For more details, please refer to the VABS manual.

Listing 2.7: Example foot file. (Foot.xml)

```

<foot>
  <displacement>0.0 0.0 0.0</displacement>
  <rotation>
    1.0 0.0 0.0
    0.0 1.0 0.0
    0.0 0.0 1.0
  </rotation>
  <sectional_force>1.0e3 0.0 0.0</sectional_force>
  <sectional_moment>1.0e3 0.0 0.0</sectional_moment>
  <distributed_force>
    0.0 0.0 0.0
    0.0 0.0 0.0
    0.0 0.0 0.0
    0.0 0.0 0.0
  </distributed_force>
  <distributed_moment>
    0.0 0.0 0.0
    0.0 0.0 0.0
    0.0 0.0 0.0
    0.0 0.0 0.0
  </distributed_moment>
  <axial_strain>1.0e-3</axial_strain>
  <twist_curvature>1.0e-3 0.0 0.0</twist_curvature>
  <twist_d1d2d3>0.0 0.0 0.0</twist_d1d2d3>
</foot>

```

2.3 Generation and Homogenization of the Airfoil Cross-Section

In the Windows Command Prompt, change directory to where script `runvabs.py` is located. Then type:

```
abaqus cae noGui=runvabs.py -- Control.xml
```

If all files are created properly, then Abaqus will create the meshed cross-section first and then run VABS.

2.4 Dehomogenization

Before running, make sure that several modifications of files are properly made:

1. Add a new recover name in the control file;
2. Add the name of foot file in the control file;
3. Change the recover flag from 0 to 1 in the head file;
4. Add or edit the foot file so that all parameters required by the analysis model are set properly.

Then in the Windows Command Prompt, change directory to where script `runvabs.py` is located. Then type:

```
abaqus cae noGui=runvabs.py -- Control.xml
```

If all files are set properly, then VABS will run the dehomogenization.

2.5 Visualization


In the Windows Command Prompt, type:

```
abaqus vabs
```

In the opened Abaqus GUI, there should be an extra VABS toolset. (Figure 2.4)



Figure 2.4: VABS toolset in Abaqus.

Set the work directory to where those recovered files are located. Then click the **Visualization** button . In the **Visualization** dialog box, select the VABS input file with the recover name. Click **OK**.

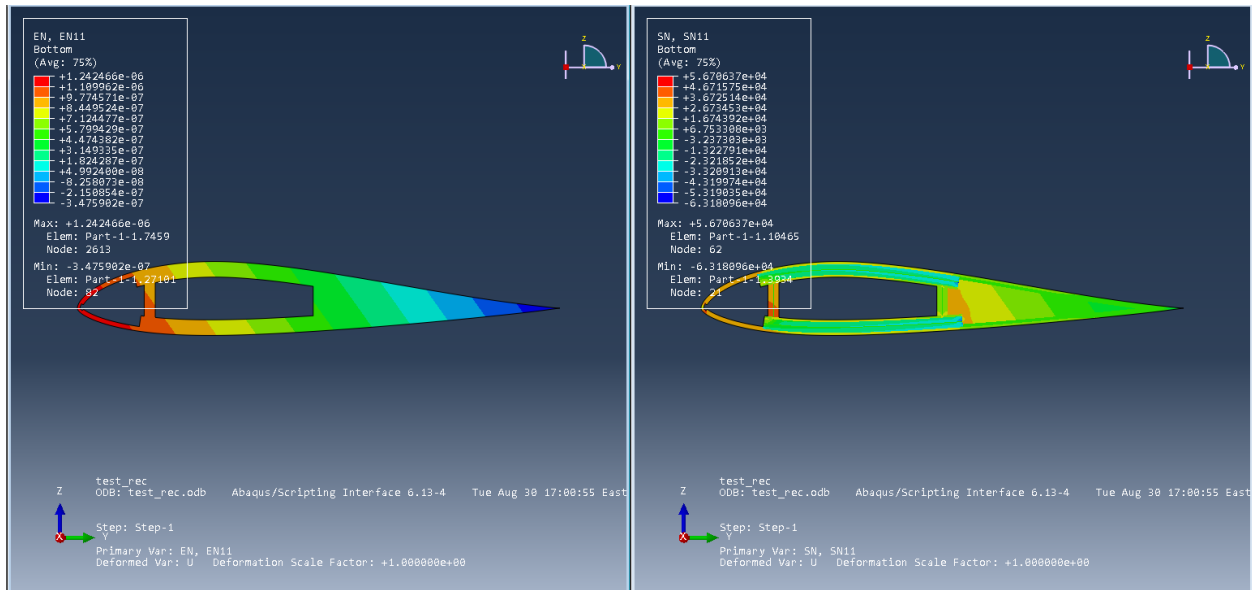


Figure 2.5: Visualization.