VSP2WOPWOP User Manual

Daniel Weitsman [*]
Pennsylvania State University, University Park, PA, 16802

Nomenclature

I. Introduction

VSP2WOPWOP is a code that was written in Python, which couples NASA's parametric aircraft geometry tool, Open Vehicle Sketch Pad (OpenVSP) with the acoustic prediction code PSU-WOPWOP. The benefit of OpenVSP is that it enables users to rapidly develop and modify blade geometries, particularly since they are parameterized in familiar terms, e.g., chord, twist, sweep. VSP2WOPWOP can then be used to analyze the degenerate blade geometry output from OpenVSP, in conjunction with airfoil cross section polars attained from MIT's XFoil panel code, to compute the blade loads using blade element momentum theory (BEMT). The program then writes out the blade geometry and loading information as binary patch and data input files for PSU-WOPWOP. VSP2WOPWOP was primarily developed to handle isolated rotors operating in hover, axial climb, and forward flight; however for more complex, perhaps multi-rotor vehicle configurations, VSP2WOPWOP can be utilized to write the necessary patch files for PSU-WOPWOP, after which the namelist file can be edited manually to reflect the case. The intent of this program is to (1) minimize the learning curve faced by new researchers working with PSU-WOPWOP and to (2) provide a framework for conducting parametric studies of blade geometries and operating conditions, where manual modification of individual test cases may not be practical.

II. Getting Started

In order to run VSP2WOPWOP, the user will need the degenerate blade geometry information output from OpwnVSP as a .csv file and an airfoil cross section polar obtained from MIT's XFoil panel code. The remainder of the case configuration is done in the input.py module. This section provides an overview of the workflow using VSP2WOPWOP.

A. VSP2WOPWOP Dependencies

Since VSP2WOPWOP is written in Python there are dependencies on several modules that must be installed in the user's environment. The most up to date releases of the following modules would be sufficient. These modules are commonly installed using conda or pip, please refer to the respective documentation for a more comprehensive set of installation instructions.

1) NumPy

[*]Research Assistant, Aerospace Engineering Department.

2) Matplotlib
3) SciPy

B. Blade Geometry

There are several important things to note about modeling the blade geometries in OpenVSP. Firstly, only a single blade actually needs to be modeled. Since the blade geometries are assumed to be identical and each blade experiences the same loads as it travels around the azimuth, the blade is duplicated in the namelist file via a change of base (CB). If a rotor consists of different blades then each blade would need to be modeled and analyzed separately, in which case the namelist file would need to be modified accordingly. Furthermore, in order for the degenerate blade geometry to be analyzed correctly, the blade must be oriented properly in the absolute coordinate system of OpenVSP. This orientation is illustrated in Fig. 1, whereby the blade spans in the direction of the positive y-axis, the positive x-axis points towards the trailing edge of the blade, and the positive z-axis is oriented upwards towards the suction side of the airfoil cross-sections.

The span and chord wise tessellation must be fine enough to capture the surface detail of the blade. A sensitivity study has not been conducted for this parameter but it was found that using approximately 50 and 25 points in the spanwise and chordwise directions, respectively yields satisfactory results. Additionally, if the blade has a twist distribution it is important to select a location along the blade span where the blade pitch would be referenced. The blade pitch at 75% span position (beta 3/4) should then be adjusted so that the pitch is zero at the reference location. This enables the user to interpret the collective and cyclic blade pitch settings computed in VSP2WOPWOP. After the geometry has been properly modeled, a degenerate geometry .csv file can be exported by navigating to the analysis tab of OpenVSP and selecting DegenGeom from the dropdown menu.

C. XfFoil Polar

XFoil polar files are referenced when computing blade loading. In hover, the lift and drag coefficients that correspond to the sectional angle of attack are directly extracted from the polars, whereas only the lift-curve slope is used in forward flight. VSP2WOPWOP automatically parses and extracts all the necessary quantities from the XFoil polar files. The number of cross section polars to use is left to the discretion of the user, adequate results have been obtained using a single polar generated with the flow parameters computed at 75% of the blade span. If the blade is composed of different airfoil cross section a polar can be generated for each and then their locations can be specified in the input.py module.

D. Configuring the Input Module

The input.py module is where the user configures a single case or an entire parametric study, it is also most likely the only module that needs to modify. This section will provide a description of each parameter in this module and

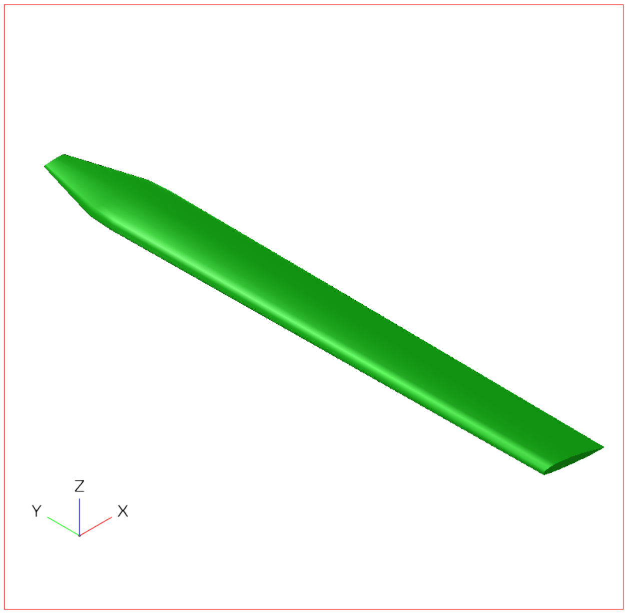

Fig. 1 Blade geometry orientation in OpenVSP.

describes how VSP2WOPWOP may be leveraged to conduct parametric studies of variating operating conditions and blade geometries. The module is organized such that the directories containing the input files (DegenGeom .csv and XFoil polars) as well as the location to where the user wishes to write the case files to are specified in the first two cells. In the following cell, the degenerate blade geometry files are specified in a comma-delimited list object (dataFileNames). There are several other variables that must be specified as lists because they are iterated over in parametric studies. Therefore, if several variations of the blade geometry are included in the dataFileNames list, separate cases would be configured for each variant.

1) I/O
   - *dirDataFile*: Absolute path to the directory containing the DegenGeom .csv and XFoil polar files.
   - *dirPatchFile*: Absolute path to the directory where the user wishes the generated patch and data files to be written to.
   - *geomFileName*: Desired name of the geometry patch file that contains both the blade geometry and compact lifting line information.
   - *loadingFileName*: Desired name of the compact loading data file.
2) Blade geometry configuration and general settings
   - *dataFileNames*: Comma-delimited list object containing the names of the degenerate geometry .csv files. VSP2WOPWOP will iterated over each geometry variant in this list. If only a single blade geometry is being analyzed it must still be included in brackets.
   - *operMode*: This variable specifies the operational mode, which concerns parametric studies. If this value is

set equal to 1, VSP2WOPWOP will run in "design mode", whereby each operating condition corresponds to a single variant of the blade geometry (*dataFileNames*). This mode is geared towards conducting parametric studies of the blade geometry, where the operating conditions may vary to account for alterations of the blade design. When using this mode, it is important to ensure that the number operating conditions corresponds to the number of DegenGeom files, therefore the lengths of the lists specifying the operating conditions (*Vx, Vz, alphaShaft, T, omega*) must be equal to that of *dataFileNames* or have a single value, if it is constant for all the blade geometry variants. When this variable is set equal to 2, VSP2WOPWOP runs in "analysis mode", which is tailored for conducting parametric studies of operating conditions. This mode essentially sweeps through all the combination of operating conditions specified in *Vx, Vz, alphaShaft, T, and omega*. When configuring a case consisting of a single blade geometry operating under a single flight condition either of the two modes could be used.

- *savePickle*: Set this value equal to 1, if the user wishes to save the main dictionary (*MainDict*), which contains all of the input and computed geometric and loading quantities as a .pkl file.

3) Airfoil cross section polar configuration

   - *airfoilPolarFileName*: This is a nested comma-delimited list containing the names of the XFoil polar files. If the blade is composed of different airfoil cross sections their corresponding polars are specified in the inner list. If running a parametric study and different polars are generated for each variation of the geometry or operating condition they can be specified in the outer list.
   - *XsecLocation*: This is a list that specifies the radial location corresponding to each airfoil polar specified in the inner list of *airfoilPolarFileName*. This quantity is non-dimensionalized by the blade radius.
   - *aStart*: This parameter is the starting angle of attack, specified in degree, from which to calculate the lift-curve slope. This starting angle of attack must be included in the XFoil polar output file, otherwise an error would arise.
   - *aLength*: This is the range of angles of attack, specified in degree, over which to evaluate the lift-curve slope.
   - *check*: If this variable is set to 1, all of the airfoil polars along with the region over which the lift-curve slope was evaluated would be plotted.

4) Operational condition configuration

   - *Nb*: Number of blades
   - *Vx*: Comma-delimited list specifying the forward flight velocities, specified in m/s.
   - *Vz*: Comma-delimited list specifying the climb or descent velocities, specified in m/s. A positive velocity corresponds to climb, whereas a negative velocity signifies that the rotor is descending.
   - *alphaShaft*: Comma-delimited list of shaft tilt angles, specified in degrees. A negative tilt angle signifies

that the rotor disk is tilted forward.

- *T*: Comma-delimited list of the thrust, specified in newtons.
- *omega*: Comma-delimited list of rotational rates, specified in rpm.
- *thetaInit*: Initial guess for the collective pitch setting. This is an arbitrary value that is needed to initialize the trim procedure, however if the trim solution does not converge this quantity will likely need to be changed to a value closer to the actual collective pitch setting. It should also not be set equal to 0.
- *loadPos*: This parameter is non-dimensionalized by the sectional chord length and specifies the position from the blade's leading edge where the compact lifting line is located. The integrated sectional blade loads are prescribed on this line.
- *tipLoss*: Set equal to 1 in order to include Pardtl's tip loss formulation. This only applies to cases where the rotor is operating in hover.
- *rho*: Density, specified in kg/m^3 and is written to the namelist file.
- *c*: Speed of sound, specified in m/s and is written to the namelist file.
- *Ib*: Blade mass moment of inertia about the flap hinge, specified in $kg * m^2$, and is only applicable for forward flight cases.
- *nuBeta*: Non-dimensionalized rotating flap frequency, which is dependent on the blade's non-rotating natural frequency and its hinge offset. This quantity is also only applicable for forward flight cases.

5) Broadband noise model configuration
   - *BBNoiseFlag*: Set equal to 1 in order to include the broadband noise prediction.
   - *BBNoiseModel*: Set equal to 1 in order to use Pegg's [1] and 2 for Brooks, Pope, and Marcolini's (BPM) [2] semi-empirical broadband models. Note: if Pegg's model is selected the corresponding namelist (&PeggIn) is currently not written to the namelist file and must be included manually- 8/20.
6) Namelist file configuration
   - *nmlWrite*: Set equal to 1 in order to write out the namelist file.
   - *NmlFileName*: Desired name for the namelist file.
7) Observer namelist configuration
   - *nRev*: Number of rotor revolutions to simulate.
   - *nt*: Desired sample rate, specified in Hz.
   - *obsType*: Specifies the type of observer grid to use. Set equal to 1 for a single observer, 2 for a rectangular grid, and 3 for a spherical grid. Then define the dimensions of these grids in the respective subsequent cells.
8) Single observer
   - *xLoc*: x-position of the observer (ahead/behind of the rotor), expressed in meters.

- *yLoc*: y-position of the observer (to the side of the rotor), expressed in meters.
- *zLoc*: z-position of the observer (above/below of the rotor), expressed in meters.

9) Rectangular observer grid
   - *nbx*: Number of observers along the x-direction (ahead/behind of the rotor).
   - *xMin*: Minimum observer position along the x-direction, expressed in meters.
   - *xMax*: Maximum observer position along the x-direction, expressed in meters.This variable can be specified as a list when conducting parametric studies.
   - *nby*: Number of observers along the y-direction (to the side of the rotor).
   - *yMin*: Minimum observer position along the y-direction, expressed in meters.
   - *yMax*: Maximum observer position along the y-direction, expressed in meters. This variable can be specified as a list when conducting parametric studies.
   - *nbz*: Number of observers along the z-direction (above/below the rotor).
   - *zMin*: Minimum observer position along the z-direction, expressed in meters.
   - *zMax*: Maximum observer position along the z-direction, expressed in meters. This variable can be specified as a list when conducting parametric studies.

10) Spherical observer grid
   - *radius*: Radial position of the observers from the rotor hub, express in meters. This variable can be specified as a list when conducting parametric studies.
   - *nbTheta*: Number of in-plane observers.
   - *thetaMin*: Minimum angle of in-plane observer, expressed in degrees.
   - *thetaMax*: Maximum angle of in-plane observer, expressed in degrees. This variable can be specified as a list when conducting parametric studies.
   - *nbPsi*: Number of out-of-plane observers.
   - *psiMin*: Minimum angle of out-of-plane observer, expressed in degrees.
   - *psiMax*: Maximum angle of out-of-plane observer, expressed in degrees. This variable can be specified as a list when conducting parametric studies.

E. Running VSP2WOPWOP

F. Loading Computed Parameters

III. Theory

A. Hover

B. Forward Flight

IV. Validation

V. Program Structure

1) input.py

VI. Modules

VII. Appendices

1) Define all variables in MainDict

References

[1] Pegg, R., "A Summary and Evaluation of Semi-Empirical Methods for the Prediction of Helicopter Rotor Noise," Tech. Rep. Technical Memorandum 80200, NASA, December 1979.

[2] Brooks, P. D., T.F., and Marcolini, M., "Airfoil Self-Noise and Prediction," Tech. Rep. Reference Publication 1218, NASA, July 1989.