pySailingVLm Documentation

Grzegorz Gruszczyński, Zuzanna Wieczorek

Apr 29, 2023

CONTENTS

The Vortex lattice method, also called VLM, is a numerical method used in computational fluid dynamics. VLM models a surface on aircraft as infinite vortices to estimate the lift curve slope, induced drag, and force distribution. The VLM is the extension of Prandtl's lifting-line theory that is capable of computing swept and low aspect ratio wings. [SHR+19].

In this thesis, the method was used to determinate aerodynamic parameters such as CL, CD and lift-to-drag ratio for modeling behaviour of sailing yacht with different initial wind conditions. The main purpose of this work was to update existing VLM code to be faster and more efficient. To achieve this goal authors have changed the structure of data and applied some compilations for specific functions in Python code using Numba which translates a subset of Python and NumPy code into fast machine code. Apart from this, the code has gained some bug fixes and new features such as cambered sail, pressure coefficient colormap, validation tests, command line script and jupyter notebook examples. Moreover, code has been packaged and now is available at The Python Package Index (PyPI) - a repository of software for the Python programming language. #TODO pySailingVLM at PyPI

Keywords: Vortex Lattice Method, VLM, sailing, yacht, Python, code, visualisation, yacht engineering

Inside documentation:

Part I

Description

CHAPTER

ONE

THEORY

1.1 Introduction

Vortex Lattice Method is build on the theory of ideal flow, known as Potential flow. Ideal flow is a simplication of the real flow which can be seen in nature. This approximation is suffiecient from the engineering point of view. VLM do not take into account any turbulence, dissipation and viscous effects.

1.2 Potential flow theory

Potenttial Flow Theory treats external flows around bodies as invicid and irrational [Tec05] .The viscous effects are limitted to a thin layer next to the body (boundary layer). Becouse of this and separation phenomena such fluid can be modelled only with small angle of attack. In irrational flow fluid particles are not rotating.

In Potential Flow Theory, because velocity must satisfy the conservation of mass equation, the Laplace Equation is qoverning:

$$\nabla^2 \phi = 0 \tag{1.1}$$

where $\phi$ is a potentinal function defined as continous function which satisfies the conservation of mass and momentum (incompressible, inviscid and irrational flow).

CHAPTER

TWO

VORTEX LATTICE METHOD

The Vortex Lattice Method is a panel method where wing or other configuration is modelled by a large number of elementary, quadrilateral panels lying either on the actual aircraft surface, or on some mean surface, or combination thereof. Sources, vortices and doublets are attached to each panel. Such singularities are defined by specifying functional variation across the panel and its value is set by determinating strength parameters. Such parameters are known after solving appropriate boundary condition equations. When singularity strengts are determinated, the pressure, velocity can be computed[JJB09].

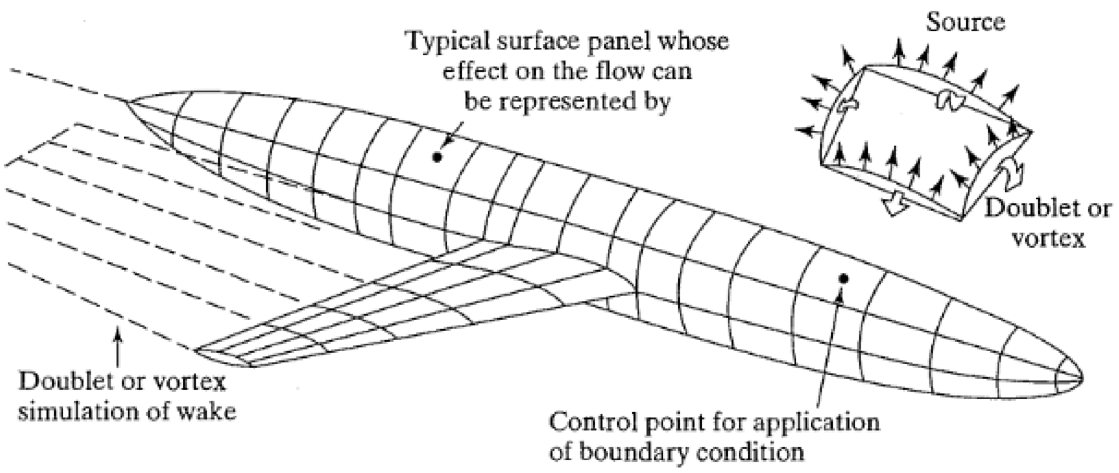

Typical surface panel whose effect on the flow can be represented by
Source
Doublet or vortex
Doublet or vortex simulation of wake
Control point for application of boundary condition

Fig. 2.1: Representation of an airplane flowfield by panel (or singularity) methods. Figure taken from [JJB09] (Fig. 7.22 page 350).

2.1 Vortices

Element which creates circulation around its origin is called a vortex. According to German scientist Hermann von Helmholtz who developed vortex theorems for inviscid incompressible flows [JK01]:

- The strength of vortex filament is constant along its length
- A vortex filament cannot start or end in a fluid (it must form or closed path or extend to infinity)
- The fluid that forms a vortex tube continues to form a vortex tube and the strength of the vortex tube remains constant as the tube moves about.

2.1.1 Vortex element

Vortex element in 2D is presented on figure 2.2. It is created by placing a rotating cylindrical core in two dimentional flowfield. If the cylinder has a radius R and a constant angular velocity of ω_y, then its motion results in a flow with circular streamlines [JK01].

The tangential velocity induced is describe by equation:

$$q_\theta(r) = \frac{\Gamma}{2\pi \cdot r} \tag{2.1}$$

where r is described as distance from the vortex.

If a vortex is located in free-strem with uniform velocity u_∞ then the total velocity at the distance r can be writtes as $u_\infty + q_\theta(r)$ [AH13].

2.1.2 Vortex segment

The velocity induced by a straight vortex segement (figure 2.3) in three dimentions with given vortex strength Γ at point $P(x,y,z)$ is qiven by equation:

$$q_{1,2} = \frac{\Gamma}{4\pi} \frac{\overrightarrow{r_1} \times \overrightarrow{r_2}}{|\overrightarrow{r_1} \times \overrightarrow{r_2}|^2} \left(\frac{\overrightarrow{r_1}}{||\overrightarrow{r_1}||} - \frac{\overrightarrow{r_2}}{||\overrightarrow{r_2}||} \right) \cdot \overrightarrow{r_0} \tag{2.2}$$

where

$$\overrightarrow{r_0} = \overrightarrow{r_1} - \overrightarrow{r_2} \tag{2.3}$$

and

$$\overrightarrow{r_0} = (x_2, y_2, z_2) - (x_1, y_1, z_1) \tag{2.4}$$
$$\overrightarrow{r_1} = P - (x_1, y_1, z_1) \tag{2.5}$$
$$\overrightarrow{r_2} = P - (x_2, y_2, z_2) \tag{2.6}$$

In case of a semi-infinite vortex line, the point 2 is infinitely far away thus formula reads [PS00]:

$$q_{1,2} = \frac{\Gamma}{4\pi} \frac{\overrightarrow{u_\infty} \times \overrightarrow{r_1}}{||\overrightarrow{r_1}||(||\overrightarrow{r_1}|| - \overrightarrow{u_\infty} \cdot \overrightarrow{r_1})} \tag{2.7}$$

Referencing do wyrazen matematycznych dziala w latexie, w htmlu nie, wiec rzeba tak pisac zdania zeby w latxie byly odnosniki a w html nie bylo to widoczne np tak jak w ostatnim wyrazeniu matematycznym 2.2.

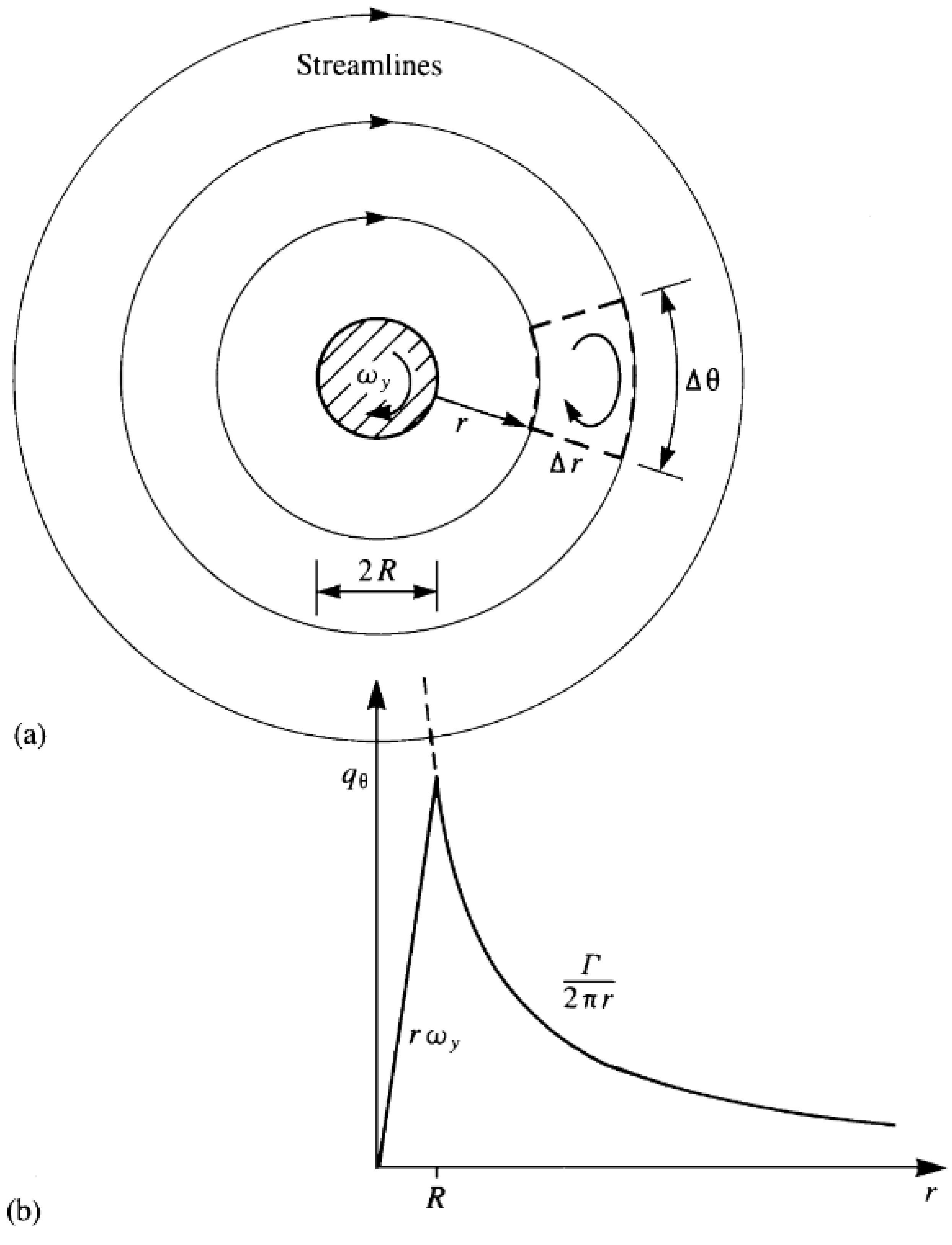


Fig. 2.2: Two dimentional flowfield around a cylindrical core rotating as a rigid body. Figure taken from [JK01] (Fig. 2.11 page 34).

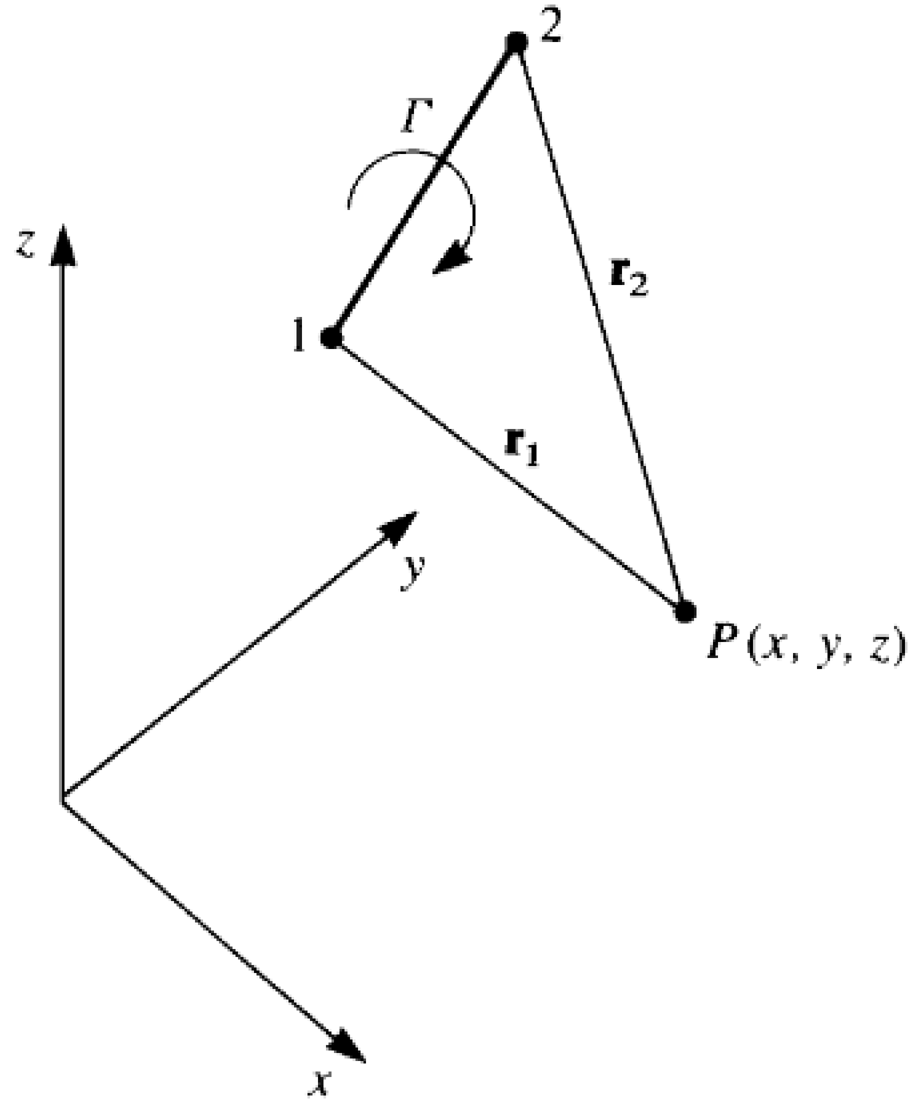


Fig. 2.3: Vortex segment in three dimentions. Figure taken from [JK01] (Fig. 2.16 page 40).

CHAPTER

THREE

TECHNOLOGY

Part II

Getting Started

CHAPTER

FOUR

INSTALLATION

4.1 For Users

pySailingVLM package is available at PyPI, to install it do:

```
pip install pySailingVLM
```

4.2 For Developers

If you would like to dive into pySailingVLM code and test it on your own ,please do the folowing steps:

- Clone git project

```
git clone #TODO wsadzic tutaj link
```

- Go into project

```
cd #TODO project
```

- There is no requirements.txt file, for installing dependencies install a project in editable mode (i.e. setuptools "develop mode") from a local project path:

```
pip install -q -e .
```

4.2.1 Create package

Building package requires setup.cfg and pyproject.toml files located in main tree of pySailingVLM project. If you do not have the latest pip build package do:

```
pip install --upgrade build
```

Then:

```
pip install build
```

Build and install

```
python3 -m build
```

```
pip install dist/pySailingVLM-VERSION.tar.gz
```

CHAPTER

FIVE

USAGE

Note: First usage of pySailingVLm will produce numba warining. This behaviour is correct. Warining messages will disappear during second run of program.

Example warning:

/home/zuzanna/Documents/miniconda3/envs/sv_build_test_2/lib/python3.10/site-packages/numba/core/lowering.py:107: NumbaDebugInfoWarning: Could not find source for function: <function __numba_array_expr_0x7fbf333f1780 at 0x7fbf33361b40>. Debug line information may be inaccurate.

5.1 Run from command line

5.1.1 input file

In order to run pySailingVLM from command line you must provide a variable.py file. Example file is shown below. Modify it for your needs.

```
import os
import numpy as np
import time

# OUTPUT DIR
case_name = os.path.basename(__file__)  # get name of the current file
case_dir = os.path.dirname(os.path.realpath(__file__))  # get dir of the current file
time_stamp = time.strftime("%Y-%m-%d_%Hh%Mm%Ss")
output_dir_name = os.path.join("results_example_jib_and_mainsail_vlm", time_stamp)
file_name = 'my_fancy_results' # name of xlsx excel file

# SOLVER SETTINGS
n_spanwise = 4  # No of control points (above the water) per sail, recommended: 50
n_chordwise = 1 # No of control points (above the water) per sail, recommended: 50
interpolation_type = "linear"  # either "spline" or "linear"
LLT_twist = "real_twist"  # defines how the Lifting Line discretize the sail twist.
# It can be "sheeting_angle_const" or "average_const" or "real_twist"

# SAILING CONDITIONS
leeway_deg = 0.    # [deg]
heel_deg = 0.     # [deg]
SOG_yacht = 0.   # [m/s] yacht speed - speed over ground (leeway is a separate␣
↪variable)
```

(continues on next page)

(continued from previous page)

```
tws_ref = 1.     # [m/s] true wind speed
alpha_true_wind_deg = 0#10.   # [deg] true wind angle (with reference to course over␣
↪ground) => Course Wind Angle to the boat track = true wind angle to centerline +␣
↪Leeway
reference_water_level_for_wind_profile = 0.  # [m] this is an attempt to mimick the␣
↪deck effect
# by lowering the sheer_above_waterline
# while keeping the wind profile as in original geometry
# this shall be negative (H = sail_ctrl_point - water_level)
wind_exp_coeff = 0.  # [-] coefficient to determine the exponential wind profile
wind_reference_measurment_height = 10.  # [m] reference height for exponential wind␣
↪profile
rho = 1.225  # air density [kg/m3]
wind_profile = 'flat' # allowed: 'exponential' or 'flat' or 'logarithmic'
roughness = 0.05 # for logarithmic profile only

# GEOMETRY OF THE RIG
main_sail_luff = 0.5#10. # 12.4  # [m]
jib_luff = 10.0  # [m]
foretriangle_height = 11.50  # [m]
foretriangle_base = 3.90  # [m]
sheer_above_waterline = 0.  # [m]
boom_above_sheer = 0. #1.3  # [m]
rake_deg = 45+90.  # rake angle [deg]
mast_LOA = 0.0  # [m]

# INPUT - GEOMETRY OF THE SAIL
sails_def = 'main' # definition of sail set, possible: 'jib' or 'main' or 'jib_and_
↪main'
# sails_def = 'main'
main_sail_girths = np.array([0.00, 1./8, 1./4, 1./2, 3./4, 7./8, 1.00])
main_sail_chords = np.array([0.2]* len(main_sail_girths)) # np.array([4.00, 3.82, 3.
↪64, 3.20, 2.64, 2.32, 2.00])
main_sail_centerline_twist_deg = -2 + 0* main_sail_girths # 10 + 12. * main_sail_
↪girths  #

# First digit describing maximum camber as percentage of the chord.
# Second digit describing the distance of maximum camber from the airfoil leading␣
↪edge in tenths of the chord.
jib_sail_camber= 0*np.array([0.01, 0.01, 0.01, 0.01, 0.01])
jib_sail_camber_distance_from_luff = np.array([0.5, 0.5, 0.5, 0.5, 0.5]) # starting␣
↪from leading edge
main_sail_camber= 0*np.array([0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01])
main_sail_camber_distance_from_luff = np.array([0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]) #␣
↪starting from leading edge

jib_girths = np.array([0.00, 1./4, 1./2, 3./4, 1.00])
jib_chords = 1E-12* np.array([3.80, 2.98, 2.15, 1.33, 0.5])
jib_centerline_twist_deg =  0*(10+5)  + 0*15. * jib_girths # (10+5)  + 15. * jib_
↪girths #

# REFERENCE CSYS
# The origin of the default CSYS is located @ waterline level and aft face of the mast
# The positive x-coord: towards stern
# The positive y-coord: towards leeward side
# The positive z-coord: above the water
```

(continues on next page)

(continued from previous page)

```
# To shift the default CSYS, adjust the 'reference_level_for_moments' variable.
# Shifted CSYS = original + reference_level_for_moments
# As a results the moments will be calculated around the new origin.

# yaw_reference [m] - distance from the aft of the mast towards stern, at which the␣
↪yawing moment is calculated.
# sway_reference [m] - distance from the aft of the mast towards leeward side. 0 for␣
↪symmetric yachts ;)
# heeling_reference [m] - distance from the water level,  at which the heeling moment␣
↪is calculated.
reference_level_for_moments = np.array([0, 0, 0])  # [yaw_reference, sway_reference,␣
↪heeling_reference]

# GEOMETRY OF THE KEEL
# to estimate heeling moment from keel, does not influence the optimizer.
# reminder: the z coord shall be negative (under the water)
center_of_lateral_resistance_upright = np.array([0, 0, -1.0])  # [m] the coordinates␣
↪for a yacht
```

5.1.2 Run script

To run script with variables.py located in working directory:

```
pySailingVLM
```

If variables.py is located in different directory, you must sepecify its location by providing additional option for pySailingVLM script:

```
pySailingVLM --dvars path_to_foler_with_variables.py
```

More information is available inside script help:

```
pySailingVLM --help
```

5.1.3 Output

pySailingVLM produces output files: data in xlsx file, matplotlib figures and pressure coefficient colormap in html extention (interactive plot). It is saved in the location specified in varaibles.py.

5.2 Run from python

```
from sailing_vlm.runner import aircraft as a
import numpy as np
chord_length = 1.0
half_wing_span = 5.0
AoA_deg = 10.
sweep_angle_deg = 0.
tws_ref = 1.0
```

(continues on next page)

(continued from previous page)

```
V = np.array([tws_ref, 0., 0.])
rho = 1.225
gamma_orientation = -1.0
n_spanwise = 32  # No of control points (above the water) per sail, recommended: 50
n_chordwise = 8 #?
b = a.Aircraft(chord_length, half_wing_span, AoA_deg, n_spanwise, n_chordwise, V, rho,
 ↪ gamma_orientation)
b.Cxyz
>>> (0.023472780216173314, 0.8546846987984326, 0.0)
```

5.3 jupyter notebook

Patrz dołącznoy plik notebooka.

CHAPTER

SIX

EXAMPLES

6.1 Example 1

6.2 Example 2

CHAPTER

SEVEN

VALIDATION

7.1 Flat_plate

7.2 Sweep

CHAPTER

EIGHT

CONTENT WITH NOTEBOOKS

You can also create content with Jupyter Notebooks. This means that you can include code blocks and their outputs in your book.

```
# varaibles.py for jupyter
import os
import numpy as np
import time

# OUTPUT DIR
case_dir = os.path.abspath('') # current directory
time_stamp = time.strftime("%Y-%m-%d_%Hh%Mm%Ss")
output_dir_name = os.path.join("results_example_jib_and_mainsail_vlm", time_stamp)
file_name = 'my_fancy_results' # name of xlsx excel file

# SOLVER SETTINGS
n_spanwise = 4  # No of control points (above the water) per sail, recommended: 50
n_chordwise = 1 # No of control points (above the water) per sail, recommended: 50
interpolation_type = "linear"  # either "spline" or "linear"
LLT_twist = "real_twist"  # defines how the Lifting Line discretize the sail twist.
# It can be "sheeting_angle_const" or "average_const" or "real_twist"

# SAILING CONDITIONS
leeway_deg = 0.    # [deg]
heel_deg = 0.     # [deg]
SOG_yacht = 0.   # [m/s] yacht speed - speed over ground (leeway is a separate␣
↪variable)
tws_ref = 1.     # [m/s] true wind speed
alpha_true_wind_deg = 0#10.   # [deg] true wind angle (with reference to course over␣
↪ground) => Course Wind Angle to the boat track = true wind angle to centerline +␣
↪Leeway
reference_water_level_for_wind_profile = 0.  # [m] this is an attempt to mimick the␣
↪deck effect
# by lowering the sheer_above_waterline
# while keeping the wind profile as in original geometry
# this shall be negative (H = sail_ctrl_point - water_level)
wind_exp_coeff = 0.  # [-] coefficient to determine the exponential wind profile
wind_reference_measurment_height = 10.  # [m] reference height for exponential wind␣
↪profile
rho = 1.225  # air density [kg/m3]
wind_profile = 'flat' # allowed: 'exponential' or 'flat' or 'logarithmic'
roughness = 0.05 # for logarithmic profile only

# GEOMETRY OF THE RIG
```

(continues on next page)

(continued from previous page)

```
main_sail_luff = 0.5#10. # 12.4   # [m]
jib_luff = 10.0   # [m]
foretriangle_height = 11.50   # [m]
foretriangle_base = 3.90   # [m]
sheer_above_waterline = 0.   # [m]
boom_above_sheer = 0. #1.3   # [m]
rake_deg = 45+90.   # rake angle [deg]
mast_LOA = 0.0   # [m]

# INPUT - GEOMETRY OF THE SAIL
sails_def = 'main' # definition of sail set, possible: 'jib' or 'main' or 'jib_and_
↪main'
# sails_def = 'main'
main_sail_girths = np.array([0.00, 1./8, 1./4, 1./2, 3./4, 7./8, 1.00])
main_sail_chords = np.array([0.2]* len(main_sail_girths)) # np.array([4.00, 3.82, 3.
↪64, 3.20, 2.64, 2.32, 2.00])
main_sail_centerline_twist_deg = -2 + 0* main_sail_girths # 10 + 12. * main_sail_
↪girths   #

# First digit describing maximum camber as percentage of the chord.
# Second digit describing the distance of maximum camber from the airfoil leading␣
↪edge in tenths of the chord.
jib_sail_camber= 0*np.array([0.01, 0.01, 0.01, 0.01, 0.01])
jib_sail_camber_distance_from_luff = np.array([0.5, 0.5, 0.5, 0.5, 0.5]) # starting␣
↪from leading edge
main_sail_camber= 0*np.array([0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01])
main_sail_camber_distance_from_luff = np.array([0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]) #␣
↪starting from leading edge

jib_girths = np.array([0.00, 1./4, 1./2, 3./4, 1.00])
jib_chords = 1E-12* np.array([3.80, 2.98, 2.15, 1.33, 0.5])
jib_centerline_twist_deg =  0*(10+5)  + 0*15. * jib_girths # (10+5)  + 15. * jib_
↪girths #

# REFERENCE CSYS
# The origin of the default CSYS is located @ waterline level and aft face of the mast
# The positive x-coord: towards stern
# The positive y-coord: towards leeward side
# The positive z-coord: above the water
# To shift the default CSYS, adjust the 'reference_level_for_moments' variable.
# Shifted CSYS = original + reference_level_for_moments
# As a results the moments will be calculated around the new origin.

# yaw_reference [m] - distance from the aft of the mast towards stern, at which the␣
↪yawing moment is calculated.
# sway_reference [m] - distance from the aft of the mast towards leeward side. 0 for␣
↪symmetric yachts ;)
# heeling_reference [m] - distance from the water level,  at which the heeling moment␣
↪is calculated.
reference_level_for_moments = np.array([0, 0, 0])  # [yaw_reference, sway_reference,␣
↪heeling_reference]

# GEOMETRY OF THE KEEL
# to estimate heeling moment from keel, does not influence the optimizer.
# reminder: the z coord shall be negative (under the water)
center_of_lateral_resistance_upright = np.array([0, 0, -1.0])  # [m] the coordinates␣
↪for a yacht standing in upright position
```

```
from pySailingVLM.rotations.csys_transformations import CSYS_transformations
from pySailingVLM.yacht_geometry.hull_geometry import HullGeometry
from pySailingVLM.results.save_utils import save_results_to_file
from pySailingVLM.solver.panels_plotter import display_panels_xyz_and_winds
from pySailingVLM.results.inviscid_flow import InviscidFlowResults
from pySailingVLM.solver.coefs import get_vlm_Cxyz
from pySailingVLM.solver.vlm import Vlm
from pySailingVLM.runner.sail import Wind, Sail

from pySailingVLM.solver.panels_plotter import plot_cp
```

```
csys_transformations = CSYS_transformations(
    heel_deg, leeway_deg,
    v_from_original_xyz_2_reference_csys_xyz=reference_level_for_moments)

w = Wind(alpha_true_wind_deg, tws_ref, SOG_yacht, wind_exp_coeff, wind_reference_
 ↪measurment_height,
         reference_water_level_for_wind_profile, roughness, wind_profile)
w_profile = w.profile

s = Sail(n_spanwise, n_chordwise, csys_transformations, sheer_above_waterline, rake_
 ↪deg, boom_above_sheer,
        mast_LOA, main_sail_luff, main_sail_girths, main_sail_chords, main_sail_
 ↪centerline_twist_deg,
        main_sail_camber, main_sail_camber_distance_from_luff, foretriangle_base,␣
 ↪foretriangle_height,
        jib_luff, jib_girths, jib_chords, jib_centerline_twist_deg, jib_sail_camber,
        jib_sail_camber_distance_from_luff, sails_def, LLT_twist, interpolation_type)
sail_set = s.sail_set
hull = HullGeometry(sheer_above_waterline, foretriangle_base, csys_transformations,␣
 ↪center_of_lateral_resistance_upright)
myvlm = Vlm(sail_set.panels, n_chordwise, n_spanwise, rho, w_profile, sail_set.
 ↪trailing_edge_info, sail_set.leading_edge_info)

inviscid_flow_results = InviscidFlowResults(sail_set, csys_transformations, myvlm)

inviscid_flow_results.estimate_heeling_moment_from_keel(hull.center_of_lateral_
 ↪resistance)
```

```
Applying initial_sail_twist_deg to main_sail -  Lifting Line, mode: real_twist
```

```
# TODO: by matplotlib byl interaktywny w notebooku
print("Preparing visualization.")
display_panels_xyz_and_winds(myvlm, inviscid_flow_results, myvlm.inlet_conditions,␣
 ↪hull, show_plot=True)
df_components, df_integrals, df_inlet_IC = save_results_to_file(myvlm, csys_
 ↪transformations,
                                    inviscid_flow_results, sail_set, output_dir_name,␣
 ↪file_name)

print(f"-------------------------------------------------------------")
print(f"Notice:\n"
      f"\tThe forces [N] and moments [Nm] are without profile drag.\n"
      f"\tThe the _COG_ CSYS is aligned in the direction of the yacht movement␣
 ↪(course over ground).\n"
```

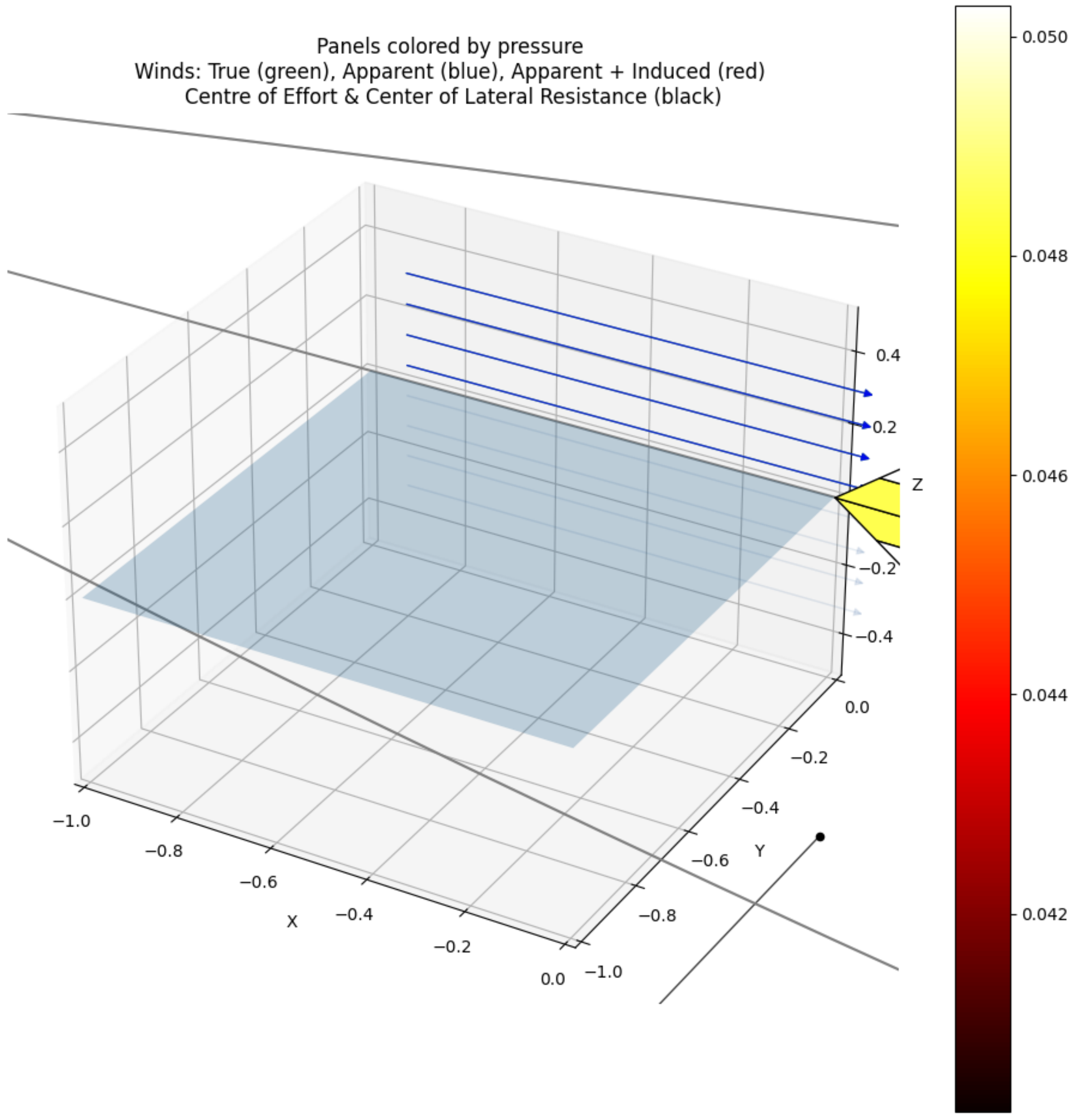
(continues on next page)

(continued from previous page)

```
      f"\tThe the _COW_ CSYS is aligned along the centerline of the yacht (course␣
↪over water).\n"
      f"\tNumber of panels (sail sail_set with mirror): {sail_set.panels.shape}")

df_integrals
```

```
Preparing visualization.
```



```
------------------------------------------------------------
Notice:
        The forces [N] and moments [Nm] are without profile drag.
```

(continues on next page)

(continued from previous page)

```
        The the _COG_ CSYS is aligned in the direction of the yacht movement␣
↪(course over ground).
        The the _COW_ CSYS is aligned along the centerline of the yacht (course␣
↪over water).
        Number of panels (sail sail_set with mirror): (8, 4, 3)
```

```
                              Quantity     Value
0               F_main_sail_total_COG.x  0.000019
1               F_main_sail_total_COG.y  0.003322
2               F_main_sail_total_COG.z -0.000019
3                   F_sails_total_COG.x  0.000019
4                   F_sails_total_COG.y  0.003322
5                   F_sails_total_COG.z -0.000019
6                   F_sails_total_COW.x  0.000019
7                   F_sails_total_COW.y  0.003322
8                   F_sails_total_COW.z -0.000019
9                    M_keel_total_COG.x -0.003322
10                   M_keel_total_COG.y  0.000019
11                   M_keel_total_COG.z  0.000000
12            M_keel_total_COW.x (heel) -0.003322
13           M_keel_total_COW.y (pitch)  0.000019
14   M_keel_total_COW.z (yaw - JG sign) -0.000000
15             M_keel_total_COW.z (yaw)  0.000000
16              M_main_sail_total_COG.x -0.000567
17              M_main_sail_total_COG.y  0.000005
18              M_main_sail_total_COG.z  0.000733
19                  M_sails_total_COG.x -0.000567
20                  M_sails_total_COG.y  0.000005
21                  M_sails_total_COG.z  0.000733
22           M_sails_total_COW.x (heel) -0.000567
23          M_sails_total_COW.y (pitch)  0.000005
24  M_sails_total_COW.z (yaw - JG sign) -0.000733
25            M_sails_total_COW.z (yaw)  0.000733
26                        M_total_COG.x -0.003889
27                        M_total_COG.y  0.000024
28                        M_total_COG.z  0.000733
29                 M_total_COW.x (heel) -0.003889
30                M_total_COW.y (pitch)  0.000024
31        M_total_COW.z (yaw - JG sign) -0.000733
32                  M_total_COW.z (yaw)  0.000733
```

```
AR = 2 * main_sail_luff / main_sail_chords[0]
S = 2* main_sail_luff * main_sail_chords[0]

C_xyz = get_vlm_Cxyz(myvlm.force, np.array(w_profile.get_true_wind_speed_at_h(1.0)),␣
↪rho, S)
C_xyz
```

```
(0.0003111088732027318, 0.05423878234346646, 8.297465605756411e-20)
```

```
# nie dziala na razie to tak jak zaplanowalam
# cel: by funcja wywołana z notebooka pokazywala na ekranie rysunek
# a wywolana ze skryptu zapisywala do pliku
```

(continues on next page)

(continued from previous page)

```
# aktualnie w obu przypadkach bedzie zapisywac do pliku html, mozna
# go otworzyc interaktywnie w przegladarce
plot_cp(sail_set.zero_mesh, myvlm.p_coeffs, output_dir_name)
```

There is a lot more that you can do with outputs (such as including interactive outputs) with your book. For more information about this, see the Jupyter Book documentation

CHAPTER

NINE

BIBLIOGRAPHY

BIBLIOGRAPHY

[AH13] T.Larsson A. Helmstad. *An Aeroelastic Implementation for Yacht Sails and Rigs, Degree project in Naval Architecture*. KTH Engeneering Sciences, 2013.

[JK01] A. Plotkin J. Katz. *Low-Speed Aerodynamics, Second Edition*. Cambridge University Press, 2001.

[JJB09] Russell M. Cummings John J. Bertin. *Aerodynamics for engineers, Fifth Edition*. Pearson Education International, 2009.

[PS00] W. F. Phillips and D. O. Snyder. Modern adaptation of prandtl's classic lifting-line theory. *JOURNAL OF AIRCRAFT*, 37(4):662–670, July - August 2000. URL: http://arc.aiaa.org, doi:DOI: 10.2514/2.2649.

[SHR+19] A. Septiyana, K. Hidayat, A. Rizaldi, M. L. Ramadiansyah, R. A. Ramadhan, P. A. P. Suseno, E. B. Jayanti, N. Atmasari, and A. Rasyadi. Analysis of aerodynamic characteristics using the vortex lattice method on twin tail boom unmanned aircraft. *7TH INTERNATIONAL SEMINAR ON AEROSPACE SCIENCE AND TECHNOLOGY - ISAST 2019*, 2226(April):536, 24-25 September 2019. URL: https://pubs.aip.org/aip/acp/article/2226/1/020003/814546/Analysis-of-aerodynamic-characteristics-using-the, doi:10.1063/5.0002337.

[Tec05] Prof. A.H. Techet. Potential flow theory. In *Lecture: 2.016 Hydrodynamics*. Massachusetts Institute of Technology, 2005. MIT. URL: https://web.mit.edu/2.016/www/handouts/2005Reading4.pdf.