PW0 - Introduction: Understanding and developing the code

1 Getting started with the software

1.1 Modeling tools

It is reminded that the pre- and post-processing phases can be performed by using GMSH (mesh generator) provided by Christophe Geuzaine and Jean-François Remacle `http://gmsh.info/`
Problem solving will be achieved thanks to the -so far- 2D finite element code provided by Jeremy Bleyer, available on the Moodle platform.

1.2 Installing softwares

Using slide 2 of *Intro_ code.pdf* and *Tips_ Meshio_ gmsh.pdf*,

1. Install python,
2. Install meshio,
3. Install gmsh.

1.3 Understanding and running the code

1. Read *Intro_ code.pdf* while exploring the code.
2. Modify the function `call_gmsh` in *wombat/input.py*
   (see slide 12 of *Intro_ code.pdf* and *Tips_ Meshio_ gmsh.pdf*).
3. Run the main file *wing.py* to test the installation.
4. (optional) If you prefer figures to be displayed outside Spyder, see appendix A.

2 Parametric study

1. **Problem geometry**. Change the wing geometry (file *wing.geo*). Choose different lengths and shapes for the wing geometry. Run the code for different geometries, save pictures and comment.
2. **Solid properties**. The constitutive material is linear elastic isotropic and homogeneous. Its characteristics are Young's modulus E, Poisson ratio ν and density ρ (that does not intervene in this preliminary example). Run the code for various properties, save pictures and comment.
3. **Higher mesh density**. Change the mesh density (file *wing.geo*). Choose different characteristic lengths (lc1, lc2,...) in order to refine the mesh where stresses should be the highest a priori. Run the code for different mesh density, save pictures and comment.
4. **Lower mesh density**. Do the reverse: change the mesh density to get the lowest possible number of elements. Run the code, inspect the file *wing.msh* and find the connectivity of elements 2 and 5. You can also inspect the object `mesh` in Spyder.

3 Developing the code

Enhance the wombat library by writing a script that computes the Von Mises equivalent stress:

$$\sigma_{VM} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_{xx}-\sigma_{yy})^2+(\sigma_{xx}-\sigma_{zz})^2+(\sigma_{yy}-\sigma_{zz})^2+6(\sigma_{xy}+\sigma_{xz}+\sigma_{yz})^2} \tag{1}$$

Plot the obtained values on an isovalues graph as it is done for the components of the stress tensor.

Write as well a portion of script that reports the maximum value of the Von Mises Stress and that indicates the element where this maximum value is encountered.

Tip: modify the file *res_ treat.py* (and look at the already implemented functions for inspiration)

A Displaying figures outside iPython shell, and saving them

If you prefer figures to be displayed outside iPython shell, then

- In Spyder: go to *Tools/Preferences/iPython Console/Graphics* and set *Backend* to "automatic" (to plot figures in independant windows).
- Close and lauch Spyder again (for the change to be accounted for).
- In *wombat/post_ process.py*, in the method `Figure.show`, set `block=False` (to plot all figures at once).

To save figures automatically, you must load pyplot in the main file (`import matplotlib.pyplot as plt`) and then use the following commands:

`plt.figure(nfig)` (so that the figure `nfig` is the current figure)
`plt.savefig("name.format", bbox_inches="tight")`

(format can be e.g. png, pdf ... as in the “save as” button). For instance, to save figure 1 that displays the mesh and shape of the solid, in pdf format:

```
plt.figure(1)
plt.savefig("Wing_mesh_and_shape.pdf", bbox_inches="tight")
```