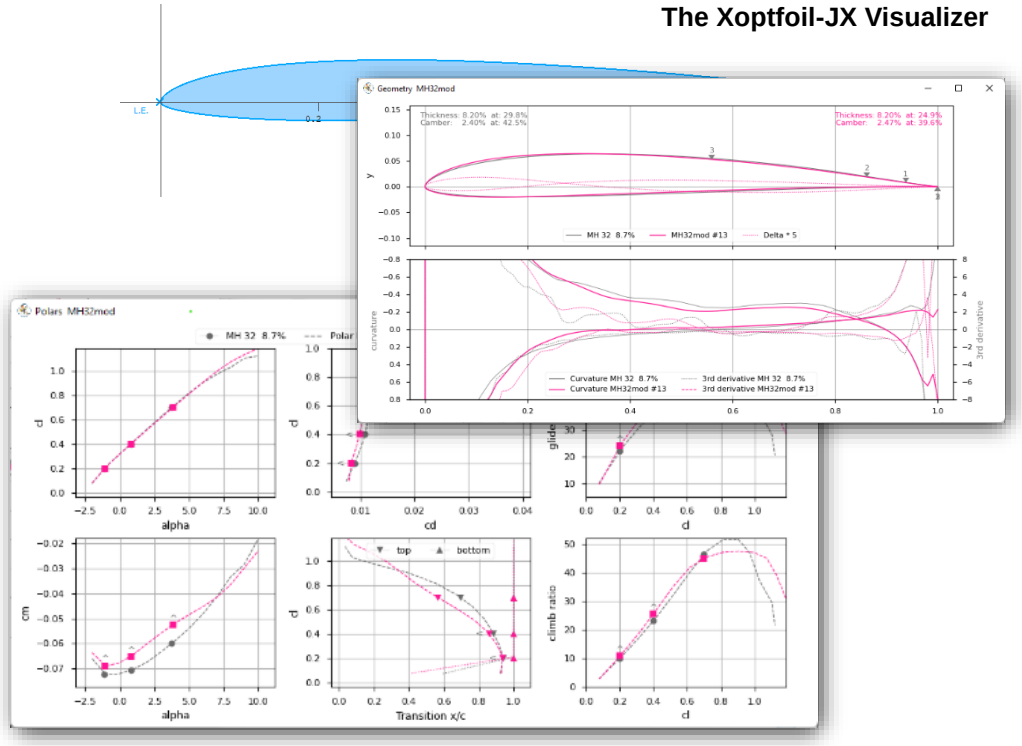
Visualizer-JX

The Xoptfoil-JX Visualizer



v1.71.4

Xoptfoil_visualizer-JX is part of the Xoptfoil-JX package.

Xoptfoil-JX is maintained on https://github.com/jxjo/Xoptfoil-JX .

Xoptfoil-JX is a modified version of Xoptfoil version 1.11.1 Daniel Prosser.

About the Visualizer

'Xoptfoil_visualizer-JX' is used to visualize the data generated by the airfoil optimizer Xoptfoil-JX during or after an optimization run. The visualizer is for display purposes only – there is no interactive user interface.

During optimization Xoptfoil-JX creates several temporary data files with data of the optimization progress, coordinate data of intermediate airfoil designs and data from the aerodynamic calculation by Xfoil.

The visualizer listens for changes within these files and updates the plots if new data arrived from Xoptfoil-JX. The intermediate files will be created in a temporary subdirectory within the current working directory.

Running the Visualizer

The visualizer is written in Python, a script based programming language. To run the script, the Python runtime environment has to be installed on the computer.

Windows only

Tip: For convenience you'll find a ready-to-run Windows EXE-file in the .\windows\bin subdirectory. Using this EXE-version there is no need to install Python.

For the first trails this version is absolutely fine.

If you want to step deeper into airfoil optimization, it's a good advice to use the Python version of the visualizer. The program will start faster and you are able to tweak the visualizer to your needs – for example changing the size of the output windows (see Appendix)

Installing Python

There a several distributions of the Python environment which may be a little confusing. The recommendation is to go for the official installation on https://www.python.org/.

When using the official installation, two additional libraries have to be installed on your computer. After having Python installed successfully type these two commands on the command line:

```
- pip install matplotlib

- pip install numpy
```

Now the visualizer should be ready to run.

If You want to run the visualizer anywhere on your file system, the following environment variable has to be set in order Python is able to find the visualizer.

```
PYTHONPATH='my_Xopfoil_JX-directory'\windows\bin
```

For testing, start a command shell and type `Xoptfoil_visualizer-JX.py` The visualizer should start …

Command line options

Command line options ease the repeated use of the visualizer during optimization sessions. Using command line options no additional input is needed each time the visualizer is re-started in the command shell.

Note for Windows: Depending on your Python installation you may have to set up once, Python scripts accept command line options. Just add the two characters `%*` at the end of following registry entries:

```
HKEY_CLASSES_ROOT\Applications\python.exe\shell\open\command

HKEY_CLASSES_ROOT\py_auto_file\shell\open\command
```

Two command line options are available:

`-c <my_case_name>`	Sets the optimization case name to visualize to `<my_case_name>` e.g. 'SD7003mod'
`-o [1,2,3,4]`	Equals to the menu option when starting the visualizer. `1` Plot a single design `1-0` Plot design #0 `2-0` Plot design #1 `3-0` Plot design #2 `2` Animate all designs `3` Monitor ongoing optimization progress For example, '-o 3' will immediately start the monitoring of your current optimization case.

Tip: Run the visualizer with a little batch job to start the visualization directly from the Explorer with the desired command line options.

This is an example for the EXE-version of the visualizer assuming you already added the Xoptfoil-JX windows\bin subdirectory to your search PATH:

```
@echo off
set Airfoil=SD7003mod
xoptfoil_visualizer-jx -c %Airfoil% -o 3
```

Output Windows of the Visualizer

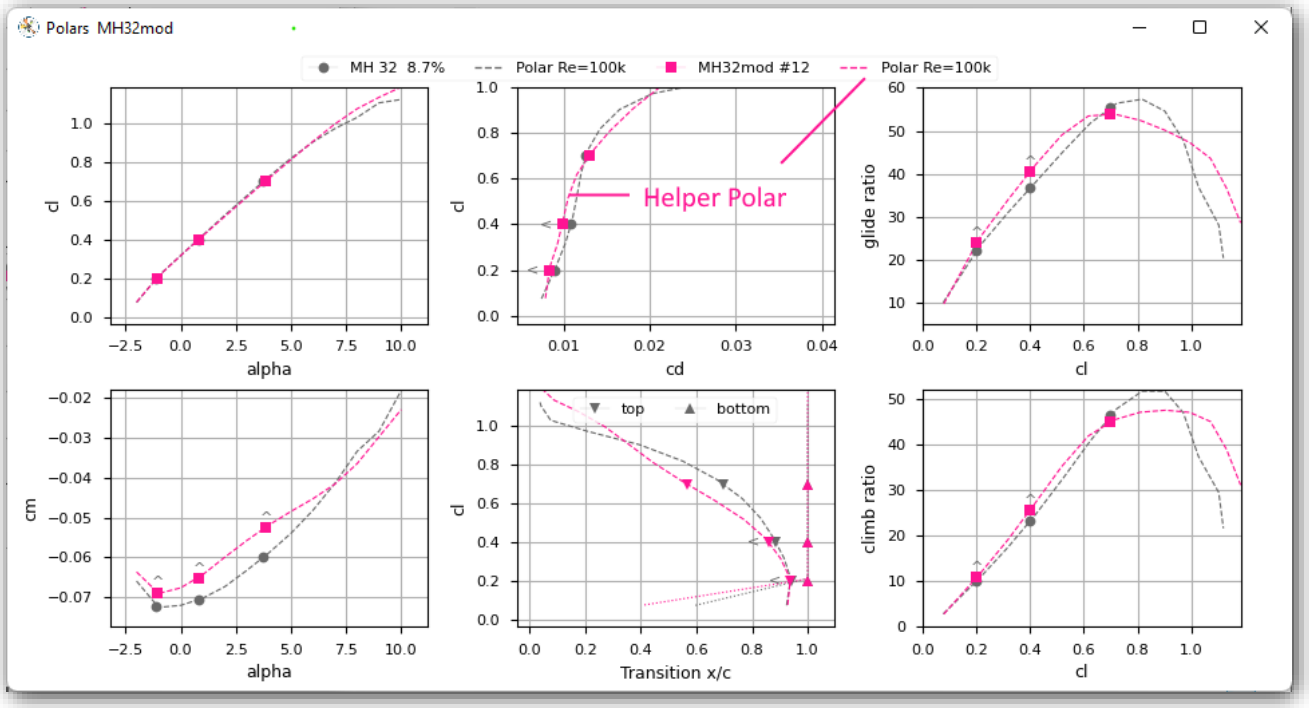
The Polar window

To get an earlier indication whether the optimized airfoil will have the desired aerodynamic properties the polar window of the visualizer shows the results of the aerodynamic Xfoil calculation.

Because the operating points could be defined for different Reynolds numbers, thus not belonging to a single polar, the operating points are not connected with lines.

Tip: To support a better and faster estimation of the current result, the optimizer can generate a complete polar for each new design. To do this, activate parameter `generate_polar` in the namelist `&polar_generation`.

Especially the glide ratio view is helpful for an early detection of optimizations resulting in poor polars.



When doing flap optimization, the actual flap angle at the operating point is shown in the cl(cd) polar.

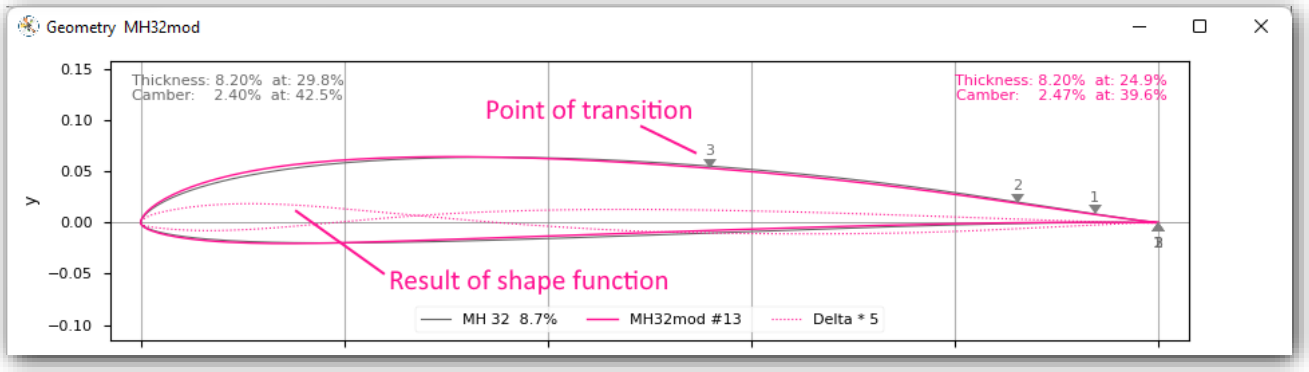
The Geometry window

The geometry window shows two plots with information about the geometry of the seed and the current design airfoil.

The upper plot shows the outline of both the seed and the current airfoil.

In addition two ‘Delta’ lines, one for the top side, one for the bottom are ploted . ‘Delta’ Is the difference between the y-coordinate of seed and design airfoil. By its nature ‘Delta’ is equal to the shape function values the optimizer is currently applying. As ‘Delta’ is scaled by factor 5, the work of the optimizer applying shapes can be watched quite convenient.

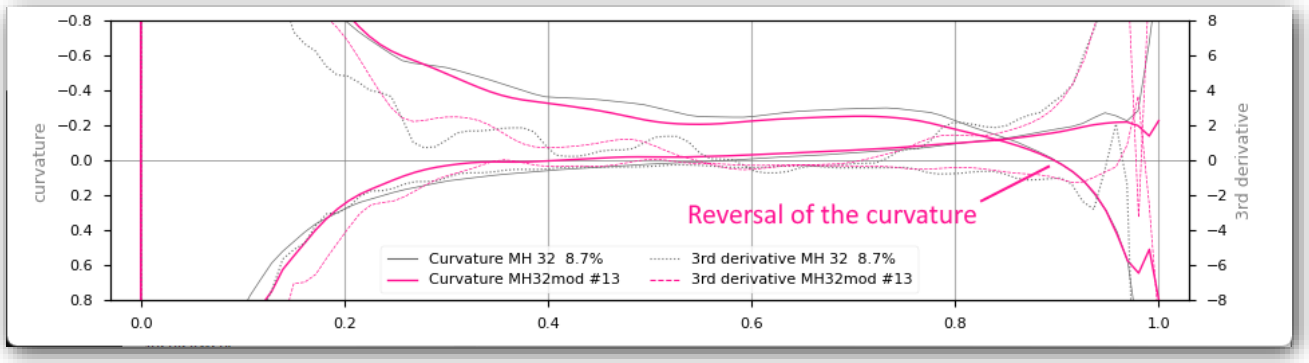
Small markers with a number indicate the point of laminar-turbulent transition of the operating points.



The lower plot is a little more advanced and focus on the curvature, which is the 2nd derivation of the airfoil contour. The curvature view allows to get a quick assessment of the airfoils surface properties and quality.

Maybe the most important aspect is the identification of ‘curvature reversals’ in the airfoils contour line.

Note: Please have a look in the Xoptfoil-JX documentation for explanations of the curvature and the 3rd derivative.



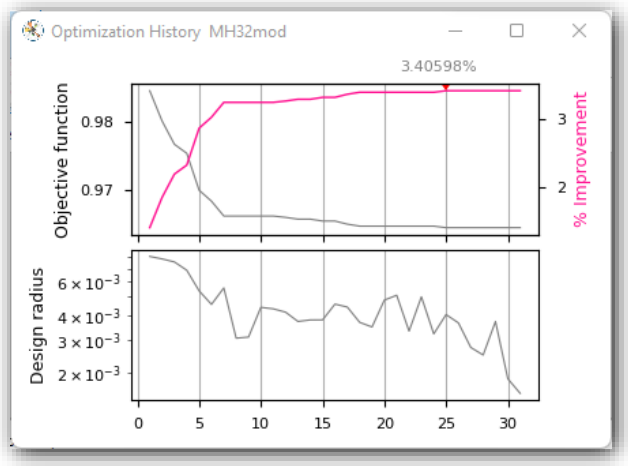
The Optimization History window

This little tool window gives some useful information about the ongoing optimization process.

The upper plot shows the course of the objective function. Always starting with ‘1’ and – hopefully – improving to lower values. The ‘Improvement’ value is just the difference between the current value of the objective function and ‘1’.

The lower plot gives information about the “activity” of the particle swarm along optimization. The “Design radius” is an indicator of the average speed, the particles searching through the solution space.

In a well defined optimization case the radius will decrease with ongoing optimization, especially when no further improvement can be achieved.



The optimization will be stopped when

- the “radius” is decreased to the value of parameter `pso_tol` (particle swarm tolerance in namelist `&pso_options`)
- the number of iterations hit its limit defined with `pso_maxit` (maximum number of iterations in namelist `&pso_options`)

Appendix: Adapt size of display windows

Size and layout of the typical three visualizer windows are adjusted to make best use of a monitor with a resolution of 1920x1024.

When using the Python script version ‘Xoptfoil_visualizer-JX.py’ for visualization, it is easy to adapt size and position of each single window to your needs.

Just use any text editor to open and edit ‘Xoptfoil_visualizer-JX.py’ and search for the string ‘setGeometry’ to find something like

```
plt.get_current_fig_manager().window.setGeometry(100,30,1300,550)
```

The first two numbers are the x,y position and the second two numbers are the size of the output window in pixels. Just overwrite the numbers with your desired values, save it and restart the visualizer – ready.

It is as easy as that.