

Version 4.0 beta

The **AirfoilEditor** serves as a fast airfoil viewer and an advanced geometry editor including Xoptfoil2-based optimization.

The App provides three operating modes:

View:

* Browse and view airfoils in subdirectories
* Analyse curvature of airfoil surface
* Show polars generated using XFOIL

Modify:

* Repanel and normalize airfoils
* Adjust thickness, camber, high points, and trailing edge gap
* Blend two airfoils
* Set flap
* Generate airfoil replicas using Bezier curves.

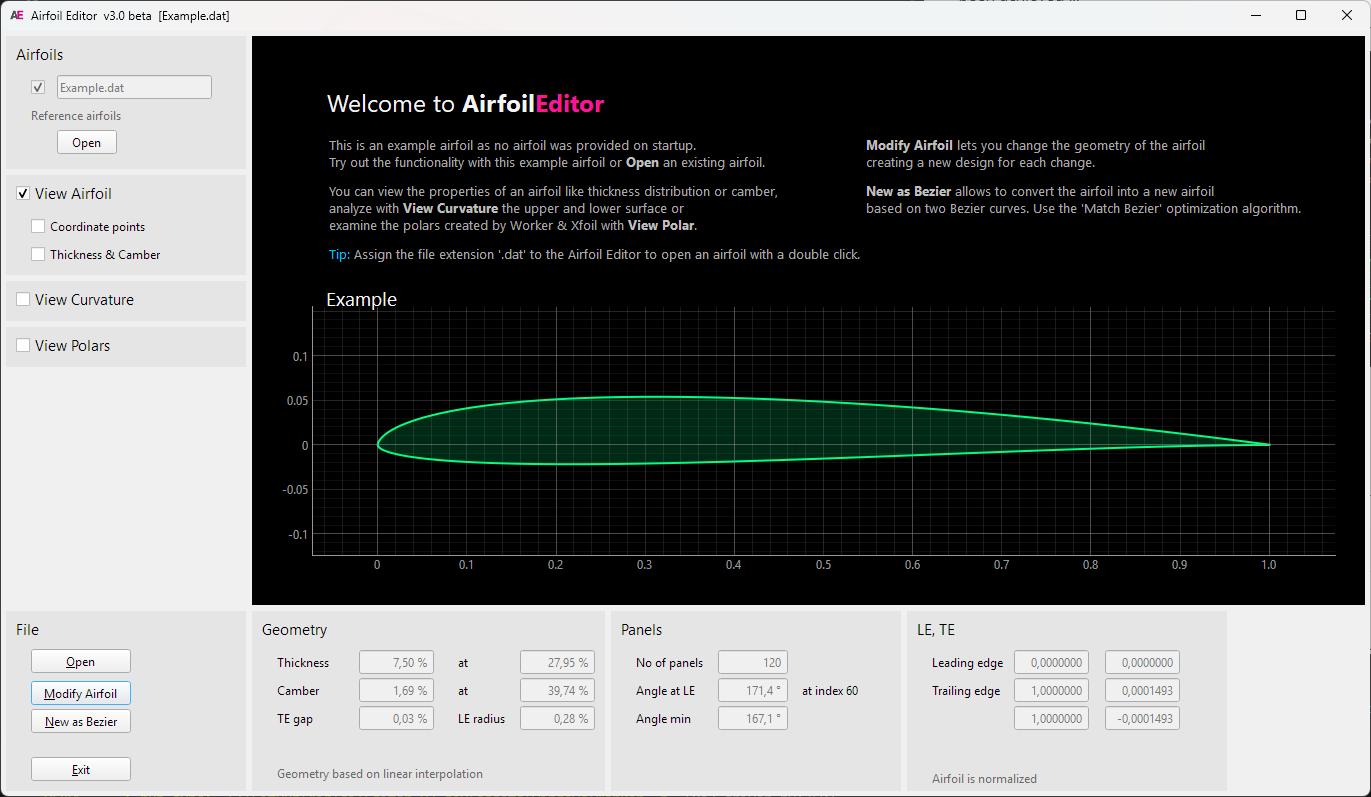
Optimize:

* User Interface for Xoptfoil2
* Graphical definition of polar based objectives
* View results while optimizing

The app was initially developed to address artefacts found in other tools like Xflr5 when using xfoil geometry routines. The aim has been an intuitive, user-friendly experience that encourages exploration.

The app, developed in Python with the Qt UI framework, runs on Windows, Linux, and MacOS. Linux and MacOS users are required to compile the underlying programs for polar viewing and airfoil optimization; refer to “Installation” for details.

Neu



# Basic Concepts

## Geometry of an Airfoil

The **AirfoilEditor** utilises various strategies to represent the geometry of an airfoil.

* 'Linear interpolation' – Using the point coordinates from the airfoils '.dat' file, intermediate points are calculated through linear interpolation. Used for quick previews and simple tasks.
* 'Cubic spline interpolation' – A cubic spline is created from the airfoil's point coordinates, enabling precise interpolation of intermediate points.
* 'Bezier curve' – An airfoil is modeled using two Bezier curves, one for the upper surface and one for the lower surface. Nelder-Mead optimization is used to fit these Bezier curves to an existing airfoil profile.

Spline interpolation is applied to determine the position of the actual leading edge, which can vary from the coordinate-based leading edge defined as the point with the smallest x-value. Airfoil normalization iteratively rotates, stretches, and shifts the airfoil so its leading edge based on the spline is at (0,0) and trailing edge at (1,0).

For thickness and camber geometry operations, the airfoil is divided into two separate splines that represent the thickness and camber distributions. To shift the high point of thickness or camber, a mapping spline - similar to that in xfoil - is applied to the thickness or camber spline. The airfoil is then reconstructed from the adjusted thickness and camber spline.

This method is also used to adjust the highpoints of both the upper and lower surfaces of the airfoil, allowing for separate modification of each side.

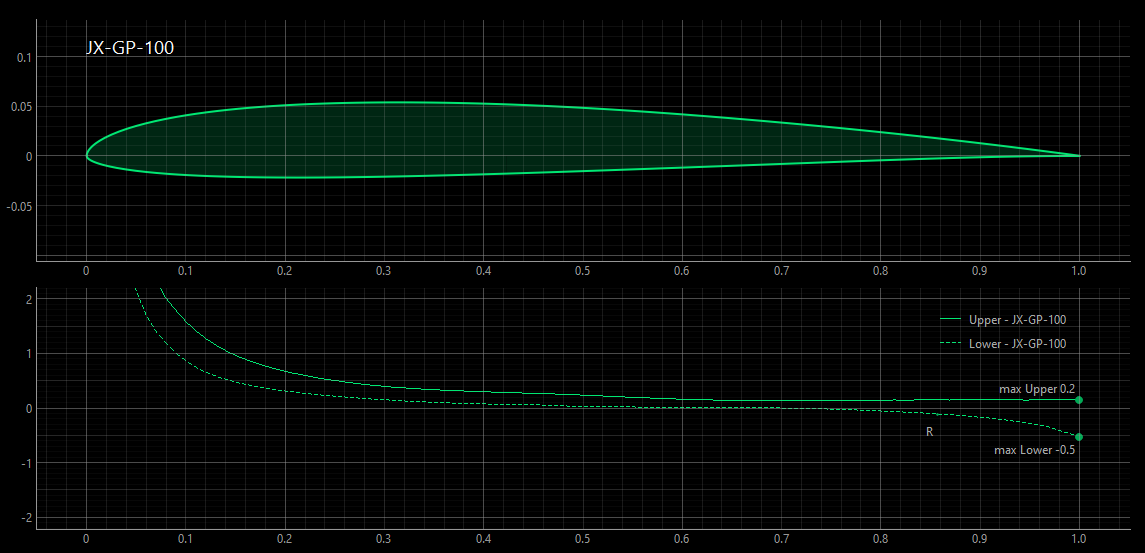
Neu

Ein Bild, das Screenshot, Diagramm, Astronomie enthält.

KI-generierte Inhalte können fehlerhaft sein.

**Curvature**

On of the major views on an airfoil in the AirfoilEditor is the curvature of the airfoils surface. It allows a quick assessment of the surface quality and to detect artefacts like a 'spoiler' at the trailing edge which is quite common.

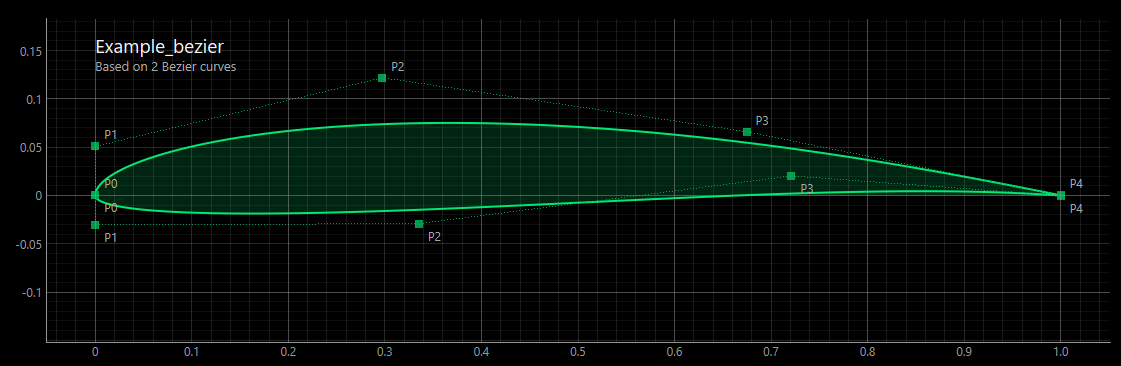


Tip

For further information regarding airfoil geometry, please refer to the documentation of Xoptfoil2.

**Bezier based airfoils**

Beside .dat files the **AirfoilEditor** seamlessly handles .bez files defining a Bezier based airfoil.

[](https://github.com/jxjo/AirfoilEditor/blob/4.0_dev/images/Bezier.png)

While a ‘normal’ airfoil is defined by coordinate points, a Bezier based airfoil is defined by two Bezier curves for upper and lower side.

A Bezier curve itself is defined by control points. One significant benefit of utilising a Bezier curve is its ability to provide a consistently smooth curvature along the airfoil surface.

The **AirfoilEditor** supports a manual mode, where the control points can be moved with the mouse to create the desired airfoil and an automatic mode with a match function:

The match function fits the Bezier curve to an existing airfoil as accurately as possible. For this a simplex optimization (Nelder Mead) is performed to

* minimize the norm2 deviation between the Bezier curve and a target airfoil
* align the curvature of the Bezier curve at leading and trailing to the curvature of the target.
* ensure the curvature at leading edge on upper and lower side is equal

Neu Match Bezier

Ein Bild, das Screenshot, Text, Diagramm, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Polars of an Airfoil

To generate the polars of an airfoil the **AirfoilEditor** uses the **Worker** tool of the [Xoptfoil2 project](https://jxjo.github.io/Xoptfoil2). On of the Worker actions is the multi-threaded creation of a polar set using Xfoil.

For polar generation the auto\_range feature of the **Worker** is applied which optimizes the alpha range of the polar to show the complete T1 polar from cl\_min to cl\_max of the airfoil. For T2 polars (constant lift) the range starts right above cl=0.0 to cl\_max.

New

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Polars on Demand

Within the app, a polar is generated on demand, specifically at the time it needs to be displayed, and this occurs asynchronously in a background task. Each polar is stored in an individual file using the Xfoil polar format.

This method enables the sequential review of airfoils or airfoil designs, displaying the polars without requiring additional user input.

### Flapped Polars

A polar can be ‘flapped’, meaning the airfoil has temporary flaps set before XFOIL computes the polar data.

A ‘flapped polar’ is convenient when different airfoils should be compared having set a certain flap angle.

In difference, a flap can be configured for an individual airfoil and saved as a separate airfoil file. This method is typically applied when the modified airfoil needs to be used in additional software, such as Xflr5 (see ‘Modification of an Airfoil’)

# 1. View Mode

Upon launch, AirfoilEditor opens in ‘View Mode’, which serves as the app’s default mode.

The ‘View Mode’ provides an overview of the geometric properties and polars of an airfoil. Since all airfoil parameters are read-only, there is no risk of making unintended changes to the airfoil file.

Using the mouse wheel on the airfoil selection combo box allows for quick browsing of the airfoils within a subdirectory to locate a specific airfoil efficiently.

The inclusion of additional reference airfoils enables comparison between the current airfoil and other airfoils.

New

Ein Bild, das Text, Screenshot, Diagramm enthält.

KI-generierte Inhalte können fehlerhaft sein.

View Mode with airfoil and polar diagram including a ‘reference airfoil’

# 2. Modify Mode

To change the geometry of an airfoil, access the 'Modify Mode' by pressing the 'Modify' button.

In the ‘Modify Mode’ a lot of airfoil parameters can be changed either by entering new values in the data fields or by moving helper points in the diagram.

New

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Airfoil Designs

A key feature of the **AirfoilEditor** is that every modification creates a new ‘Design’ version of the airfoil. Such a ‘Design’ airfoil is saved in a subdirectory related to the original airfoil. This allows to leave the ‘Modify Mode’, re-enter later and find all the Designs of the last session.

At every time you may step through the created Designs and compare the changes and the effects of the modifications on the polar.

As the polar(s) of each Design is created automatically, it becomes very easy to see how airfoil modifications relate to polar changes.

New

Ein Bild, das Screenshot, Diagramm, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

Tip

Adjust the camber highpoint position and observe its impact on polars at different Reynolds numbers. This approach helps you understand airfoil geometry interactively…

## Setting Flap

One of the possible modifications is to set a trailing edge flap – either permanently or just to assess to the influence of a flap setting on the polar of the airfoil.

new

Ein Bild, das Screenshot, Text, Diagramm, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

A little remark: As a flap may not be set on an already ‘flapped’ airfoil, the app remembers the initial unflapped design airfoil. This enables multiple sequential flap settings to be applied during a design session.

## Bezier based Airfoils

Also, Bezier based airfoils can be modified in the ‘Modify Mode’. As the geometry of such an airfoil is defined by two Bezier curves for the upper and lower side, the typical geometry parameters like ‘thickness’ cannot be changed directly.

Instead, the control points of the Bezier curves can be moved with mouse directly in the diagram.

Each modification results in a new 'Design' with newly generated polars. This allows for observation of how adjustments to the Bezier curve impact the polar.

## Match Bezier

# 3. Optimization Mode

In ‘Optimization Mode’ the **AirfoilEditor** acts as a wrapper of [Xoptfoil2](https://jxjo.github.io/Xoptfoil2).

Xoptfoil2 is a particle swarm based airfoil optimizer which supports three different ‘shaping methods’ to modify the airfoil during optimization:

* Hicks-Henne shape functions
* Bezier curve defining the shape
* Geometry parameters like maximum thickness and its position

The **AirfoilEditor** covers all steps needed for airfoil optimization with Xoptfoil2:

* Define an optimization case with the objectives and boundary conditions
* Run, control and watch an optimization
* Analyse the results
* Improve the specifications and re-run

In comparison to using Xoptfoil2 as a command-line tool, the user interface greatly streamlines the process of defining and entering operating points being objectives of the optimization.

Multiple versions of an optimization case can be managed, making it easier to select the best version for your airfoil project at the end of the optimization sessions.

Note

Before you start your own airfoil optimizations with the AirfoilEditor, you should fully understand the key concepts of Xoptfoil2 and the special terms like ‘seed airfoil’ or ‘operating point’.   
Please read carefully the chapters [Getting Started](https://jxjo.github.io/Xoptfoil2/docs/getting_started) and [Airfoil Optimization](https://jxjo.github.io/Xoptfoil2/docs/airfoil_optimization) of the Xoptfoil2 documentation. You will find the example of ‘Getting Started’ is ready to go in the **AirfoilEditor** making it easy to watch and modify your first optimization.

New

Ein Bild, das Text, Elektronik, Screenshot, Display enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Setting up an Optimization Case

The main task when setting up a new optimization case is to define the ‘operating points’ on a (virtual) polar and to choose the type of objective for each of this operating points.

Within the polar diagram of the **AirfoilEditor** operating points can be added, deleted or moved with the mouse. A little dialog allows to enter additional specifications for the selected operating point.

New

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

If a different polar (eg Reynolds Number) is defined for an operating point, this polar will be automatically added to the list of polars and displayed in the diagram.

An individual weighting is visualized by the size of the symbol in the diagram.

In the lower data panel of the **AirfoilEditor** nearly all of the numerous options of Xoptfoil2 can be modified according to the needs of the optimization.

The button ‘Input File’ opens a text editor with the current Xoptfoil2 input file which would be used for the optimization. The input file may be tweaked with this editor (or an external editor) to cover special situations.

Once the definition of the optimization case is finished, the optimization is ready to go.

## Run an Optimization

When an optimization is started, the diagram area of the AirfoilEditor is automatically maximized to have full view of what is happening during the optimization.

As the Xoptfoil2 optimization is a background task, you may change the view settings, pan and zoom the diagram to your needs while the optimization is running.

New

Ein Bild, das Text, Screenshot, Diagramm enthält.

KI-generierte Inhalte können fehlerhaft sein.

When the optimization is finished a new, final airfoil will be created.

You may have a look at the numerous Designs of the optimization process, analyse the properties of the final airfoil and if necessary, change the objectives of the optimization and re-run the optimization.

Doing such iterations it is very helpful to create a new ‘version’ when changing the parameters of the optimization. This allows to roll back to a former version which might have been better.

# Installation

## Windows Easy Setup

A pre-build Windows App including Worker.exe for polar generation and Xoptfoil2.exe for airfoil optimization is available in the [releases section](https://github.com/jxjo/AirfoilEditor/releases)

## Windows Advanced Setup

If you already have installed the actual Python version on your PC, it is advantageous to run the AirfoilEditor directly as a Python script. This will startup the app much faster than using the .exe file

1. Download python sources from [releases](https://github.com/jxjo/AirfoilEditor/releases) or clone the repository
2. Install

pip3 install pyqt6

pip3 install numpy

pip3 install pyqtgraph

pip3 install termcolor

1. Run ‘python AirfoilEditor’

## Linux and MacOS

1. Download python sources from [releases](https://github.com/jxjo/AirfoilEditor/releases) or clone the repository
2. Install

pip3 install pyqt6

pip3 install numpy

pip3 install pyqtgraph

pip3 install termcolor

1. To use polar generationand airfoil optimization you have to compile the Worker.exe and Xoptfoil.exe of the [Xoptfoil2 project](https://jxjo.github.io/Xoptfoil2) and copy the program either into /usr/local/bin or into the sub folder ./assets/linux of the AirfoilEditor. In the second case the execute permission has to be set with ‘chmod +x Worker’ and ‘chmod +x Xoptfoil2’.
2. Run ‘python AirfoilEditor’

Tip

For Windows: Use the "Open with ..." Explorer command to connect the AirfoilEditor.exe to the file extension .dat. Later a double click on an airfoil dat-file will open the **AirfoilEditor** and you can browse through the files in the directory (if you are using the Python version, create a little batch job to open an airfoils dat-file)