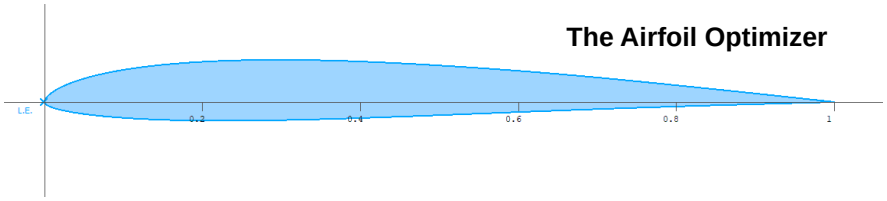
Xoptfoil-JX



Reference Guide

v1.70 beta

Xoptfoil-JX is a modified version of Xoptfoil version 1.11.1 Daniel Prosser. The original code and program is located at https://sourceforge.net/projects/xoptfoil/ .

Xoptfoil-JX can be found on https://github.com/jxjo/Xoptfoil-JX . Comments and contributions are warmely welcome

Some Remarks on Xoptfoil-JX

The intention of Xoptfoil-JX is to support the development of high-class airfoils which are ready for CAD or advanced aerodynamic calculations of a model plane in xflr5 or flow5.

The changes and enhancements made to the original Xoptfoil focus on a subset of Xoptfoil:

- Particle Swarm optimization
- Hicks-Henne shape functions
- Particle swarm optimization strategy
- 'Match foil' as a special mode for testing the optimization algorithm

Note: All the other optimization variations of the original Xoptfoil, were not tested in Xoptfoil-JX. If you want to evaluate these options, for example the genetic algorithm, you should better use Xoptfoil.

The extensions made within Xoptfoil-JX are in particalur

- Target values for operating points e.g. cd-value for a better control of optimization results
- Target values for the geometry parameters 'thickness' and 'camber' of an
- New shape type 'camber-thick' which optimizes solely the geometry parameters of an airfoil
- Advanced curvature constraints to avoid artefacts when using Hicks-Henne shape functions
- Smoothing of an airfoil as preparation for optimization
- Dynamic weighting of target based operating points to achieve a balanced, reproducible optimization result
- Automatic retry of particles when they fail due to geometric constraints

Note: In this Reference Guide only the parameter subset of the original Xoptfoil which is supported and tested in Xoptfoil-JX – as well as the extension parameters of Xoptfoil-JX to the original software are documented.

A more detailed description of the functionality of Xoptfoil-JX can be found in the Users Guide (to be written).

We are happy to get feedback on Github using 'issues'.

Have fun!

Jochen Günzel

2021

Running Xoptfoil-JX

Command line parameters

The optimizer is started as a shell command using

```
Xoptfoil-JX -i <input_file> -o <output_prefix> -a <airfoil_file> -r <reynolds>
```

Following options are supported

`-i <input_file>`	mandatory	Defines the 'input file' which holds all the parameters used for optimization.
`-o <output_prefix>`	mandatory	The 'output prefix' will be the name of the generated airfoil as well as the filename prefix for the intermediate data files used for the visualization of the optimization process.
`-a <airfoil_file>`	optional	The filename of an airfoil .dat-file which will be used as the seed airfoil for optimization. Equals to parameter 'airfoil_file' in namelist '&optimization_options'
`-r <reynolds>`	optional	The Reynolds number of all operating point without an explicit Reynolds number set. Equals to parameter 're_default' in namelist '&operating_conditions'
`-h`	optional	Show command line help

Tip:

Put the command to run Xoptfoil-JX in a little batch file (shell script) so you can start the optimization with a double click in the Explorer.

This batch file first looks for Xoptfoil-JX in the default directory – otherwise Xoptfoil-JX is searched via the PATH setting (Windows):

```
@echo off
set Airfoil=myNewAirfoil

rem Is xoptfoil-jx in the default directory?
if exist ..\..\windows\bin\xoptfoil-jx.exe (
  ..\..\windows\bin\xoptfoil-jx -i %Airfoil%.inp -o %Airfoil%
) else (
rem no - use normal windows search path
  xoptfoil-jx -i %Airfoil%.inp -o %Airfoil%
)
```

It's a good advice to name the input file like `<output_prefix>.inp` so the newly created airfoil and its corresponding input file belong together.

Together with Xfoil_worker, a little tool coming with Xoptfoil-JX, it is straightforward to automate the airfoils design process including setting flaps for the new airfoil or calculating complete polar sets which can be directly imported into Xflr5.

Parameters of the input file

The parameters of the input file define an optimization. They are grouped in sections called ‘namelists’ which start with ‘&namelist_name’ and end with an ‘/’ character.

Some namelists with certain parameters are mandatory. Other namelists are optional and only needed for advanced optimization tweaking as for example parameters for ‘Xfoil’ (aerodynamic evaluation).

The more advanced options, which only should be used with some understanding of their purpose, are indicated in the following table with ‘*advanced’.*
Based on the narrower focus of Xoptfoil-JX most of the parameters have a default value which fits well for the most optimizations. This helps to work with more compact input files.

Note: Parameters (of the original Xoptfoil) which shouldn’t be changed due to the focus of Xoptfoil-JX are not listed in this Reference.

&optimization_options	mandatory	General parameters of the optimization case
airfoil_file	mandatory	= '<my_seed_foil_file>' The filename of the airfoil which will be the seed airfoil for the optimization. The filename can alternatively be set with the “-a” command line option of Xoptfoil-JX. If the optimization use case is not to tweak an existing airfoil but to build a new airfoil using target values for the operating points (airfoil design “by polar”) it is highly recommend to use one of the “artificial” seed airfoils which are part of the Xoptfoil-JX installation. The usage of these seed airfoils are somehow a precondition to get perfect results. There are 3 seed airfoils which differ in the number of surface reversals on the top and bottom side of the airfoil JX-Seed.dat No reversals on top and bottom side JX-Seed-Rearload.dat One reversal on bottom side JX-Seed-S.dat One reversal on top side Depending on the type of target airfoil you won’t to achieve, the usage of one of these airfoils will fulfil the important “curvature reversal constraint’
shape_functions	optional (JX)	= 'hicks-henne' (Default) Hicks-Henne functions will be applied to the seed airfoil to modify the shape of the airfoil. The optimization with Hicks-Henne functions is extremely powerful in finding the optimum airfoil shape for a given optimization case. But also need special care and guidance in the definition of operating points and curvature constraints to avoid artefacts in the airfoil surface or bad aerodynamic properties outside the defined operating points. = 'camb-thick' In this case the seed airfoil will be modified during optimization by the 6 airfoil parameters

		○ max. thickness and location of max. thickness ○ max. camber and location of max. camber ○ leading edge radius and its mixing distance This shape_functions type is quite convenient if only minor changes should be made to an existing airfoil for example to adopt the airfoil to different Re number situations. It avoids possible geometric problems like bumps. = `'camb-thick-plus'` Same as ‘`'camb-thick'` but the top side and the bottom side of the airfoil are treated separately when applying the thickness, camber modifications. This doubles the design variables and increases the solution space for optimization. *Experimental:* = `'hicks-henne+'` `(Default)` When applying the shapes to the seed airfoil, the x-coordinates are transformed by `xnew(i) = acos(1 - x(i))* 2 / pi` With this the airfoil looks like a part of a circle without high curvature at leading edge .
`min_bump_width`	optional *advanced Hicks-Henne*	= `0.x` `(Default 0.1)` Defines the minimum width of a Hicks-Henne bump applied to the airfoil surface. The default value of 0.1 is fine for the most requirements.
`initial_perturb`	optional *advanced Particle Swarm*	= `0.x` `(Default 0.003)` Is a measure of the speed the particles of the swarm will initially explore the solution space. A higher value will lead to more violations of constraints (e.g. curvature or thickness of an airfoil) but will increase the probability of finding difficult solutions. A lower value may speed up convergence but maybe not with the optimum solution (local optimum). When using shape function ‘camb-thick’ the initial perturb can be increased up to 0.3 because this shape type can not lead to violations of curvature constraints
`nfunctions_top` `nfunctions_bot`	optional *advanced Hicks-Henne*	= `X` `(Default 4)` The number of Hicks-Henne functions that wil be applied to the surface of the seed airfoil to modify the airfoil during optimization. A Hicks-Henne function is described with 3 variables: x-Position, width (see also `min_bump_width)` and height of the ‘bump’. For example 5 Hicks-Henne functions on the top side of the airfoil will result in 15 design variables which have to be optimized for just on side… To limit extensive time needed for optimization, the rule must be: As little functions as possible, as much as necessary to best results… For typical optimizations 4 Hicks-Henne functions are a good, balanced value. The first trials can be done with 3 functions. To squeeze the last 0.1% improvement out of the optimization, 5 functions may be appropriate.
`show_details`	optional	= `<Boolean>` `(Default .false.)` Setting `show_details = .true.` will show a number of details during optimization:

		○ current aerodynamic values of the operating points and their deviation to the target value – or the improvement to the seed airfoil ○ new weighting value of the operating points when dynamic weighting is active ○ existence, position and size of laminar separation bubbles at an operating point Tip Together with the graphical visualizer `xoptfoil_visualizer-jx.py` it helps a lot to get a better understanding what's going on during the optimization process. Especially it helps to have an early detection if optimization doesn't lead in the desired direction.
`&operating_conditions`	mandatory	Definition of the operating points
`noppoint`	mandatory	`= X` The number of operating which will be defined for this optimization case. The sum of the desired aerodynamic result of all operating points provides the value of the "famous" objective function. Note: Define as little operating points as possible, as much as necessary to best results When using `shape_functions = 'camb-thick'` 3 or 4 operating points may enough operating to get good and valid results. For `shape_functions = 'hicks-henne'` 8 to 12 operating points may be necessary to cover the whole cl range of the airfoil to avoid collapse of cd or artefacts in the airfoil surface. The optimization time will be directly proportional to the number of operating points. So it's a good advice to start with less operating points and increase later the number for final tuning.
`re_default`	optional JX	`= xxxxxx` Default value for Reynolds number which is taken if a `reynolds(i)` for an operating point is not defined. This is useful if all operating points do have the same reynolds number. This parameter equals to the command line argument '-r xxxxxx'
`re_default_as_resqrtcl`	optional JX	= `<Boolean>` `(Default .false.)` When set to `.true.` the value of the default Reynolds number `re_default` is interpreted as re*sqrt(cl) which equals to the Type 2 polar. This is convenient if your operating points are polar Type 2 based. In this case you don't have to calculate the Reynolds number at every operating point.
`dynamic_weighting`	optional JX	= `<Boolean>` `(Default .false.)` Tip: If dynamic weighting is activated, all operating points with a target value and all geometric targets will be balanced automatically during optimization to get best results. Fine balancing of operating points weighting can be a tedious work as target values like cd have quite a different volatility depending on their corresponding cl value. For example it is much to achieve a target value at cl = 0.8 than a target at cl = 0.2. Dynamic weighting tries to bring all operating points in a

		balanced manner to their targets. At a certain frequency of new improved designs the operating points are checked regarding their deviation to their targets. The individual weighting is then adjusted to bring the operating points with a higher deviation closer to their targets. For normal operation of dynamic weighting the initial weighting of an operating point should be left to “1” which is the default weighting. If you want to prioritize a certain operating point you can increase its initial weighting – for example to “1.5”. To exclude a single operating point from dynamic weighting its weighting has to be a negative value like “-2.5”. Dynamic weighting needs at least 3 operating points to take part of the dynamic process. Note: Dynamic weighting changes the “formula” of the objective function which may lead to improvement steps after re-weighting. So – do not trust all the “improvements” …

`dynamic_weighting_spec%...`
 	
`min_weighting`
`max_weighting`
`extra_punch`
`start_with_design`
`frequency`
	advanced	The data structure `dynamic_weighting_spec%...` allows some fine tweaking of dynamic weighing behavior. The values shown are the default values of these advanced parameters
		
		`= 0.7` lower value of dynamic range
		`= 1.4` upper value of dynamic range
		`= 1.4` extra punch factor if op point is far away from target
		`= 10` First time a new weighting is performed
		`= 20` new weighting will be performed every xx design

`use_flap`	optional	= `<Boolean>` `(Default .false.)` When activated flap deflection are applied during optimization. See also parameters `flap_selection(n)` and `flap_selection(n)` for an operating point.
`x_flap`	optional	= `0.x` `(Default 0.75)` Flap hinge point in x (between 0 and 1). Only used if `use_flap` is enabled.
`y_flap_spec`	optional	= `<'y/c' | 'y/t'>` `(Default 'y/c')` Method of specifying vertical hinge location. Either `y/c' (vertical location normalized by chord length) or `y/t' (vertical location normalized by local thickness at x_flap). Only used if `use_flap` is enabled.
`y_flap`	optional	= `x.y` `(Default 0.0)` Flap hinge point in y. If `y_flap_spec` is `y/c', `y_flap` refers to the vertical location of the hinge normalized by the chord length. If instead `y_flap` is `y/t', the position is normalized by the local thickness at x flap. In the latter case, for example, -1.0 places the hinge at the lower surface, and 1.0 places it at the upper surface. Only used if `use_flap` is enabled.

`op_mode(i)`
 	
`where i = 1..noppoint`
	optional	= `<'spec-cl' | 'spec-al'>` `(Default 'spec-cl')`
		
		Defines whether to use the lift coefficient (cl) or the angle of attack (alpha) to specify the operating point.

		Tip: Use `'spec-cl'` whenever possible. There are only a few cases when `'spec-al'` is the right option like optimizing maximum cl value. Quite often `'spec-al'` based operating points will lead to deformed airfoils.

`op_point(i)`
 	
`where i = 1..noppoint`
	mandatory	= `x.y`
		
		Specifies either the angle of attack (alpha in degrees) or the lift coefficient (cl) for this operating point, depending on whether `op_mode` is `'spec-al'` or `'spec-cl'` for this operating point.
		
		Note Xfoil calculations are not accurate above stall, so do not specify too large a lift coefficient or angle of attack.
		
		Tip: Do also define operating points with a small or negative lift (negative alpha) to achieve a well-shaped bottom side of the airfoil.

`optimization_type(i)`
 	
`where i = 1..noppoint`
	mandatory	Specifies the optimization objective for this operating point.
		
		= `'target-drag'`
		
		The most important type. The operating point should a have certain cd value which is defined with `target_value(i)`
		
		= `'target-lift'`
		
		The operating point should a have certain lift coefficient cl which is defined with `target_value(i)`. In this case `op_mode` must be `'spec-al'`.
		
		= `'target-moment'`
		
		The operating point should a have certain moment coefficient cm which is defined with `target_value(i)`. For this optimization type `op_mode` must be `'spec-al'`.
		
		= `'min-drag'`
		
		Minimize the drag coefficient cd.
		
		= `'max-lift'`
		
		Maximize the lift coefficient cl. In this case `op_mode` must be `'spec-al'`
		
		= `'max-glide'`
		
		Maximize the glide ratio cl/cd. When `op_mode` is set to `'spec-al'` this type equals to `'min-drag'`
		
		= `'min-sink'`
		
		Minimize the sink rate cl**1.5/cd.
		
		= `'max-xtr'`
		
		Move the laminar-turbulent transition location as far towards the trailing edge as possible. The mean value of upper and lower side is taken as the objective.
		
		= `' max-lift-slope '`
		= `' min-lift-slope '`
		= `' min-glide-slope '`

		… are deprecated

`target_value(i)`
 	
`where i = 1..noppoint`
	mandatory	= `x.y`
		
		The target value to achieve when one of the target optimization types is used.
		
		A negative value for this parameter will be taken as a (positive) factor to the value of the seed airfoil.
		
		Tip: When tweaking existing airfoils these “negative factors” are perfect to freeze the polar of the airfoil in ranges where you do not want any changes through the optimization.

`reynolds(i)`
 	
`where i = 1..noppoint`
	optional	`= xxxxxx (Default re_default)`
		
		Defines the Reynolds number for this operating point. If this value is omitted the default Reynolds number `re_default` will be taken for this operating point.

`weighting(i)`
 	
`where i = 1..noppoint`
	optional	= `x.y` `(Default 1.0)`
		
		The weighting of this operating point towards the other operating points. The higher the weighting the more influence an operating point will have within the objective function.
		
		Tip: When having target based operating points, “dynamic weighting” allows to forget this parameter in most cases.
		→ see `dynamic_weighting`

`flap_selection(i)`
 	
`where i = 1..noppoint`
	optional	= `<'specify' | 'optimize'>` `(Default 'specify')`
		
		If `'specify'` the flap angle will have a fixed value of `flap_degrees..`
		In case of `'optimize'` the flap angle will be optimized at this operating point. `flap_degrees` will be taken for the seed airfoil to get initial values.

`flap_degrees(i)`
 	
`where i = 1..noppoint`
	optional	= `x.y` `(Default 0.0)`
		
		Specifies the flap angle in degrees at this operating point. A positive angle means flap downward.
		The optimized flap angle will be constrained by `max_flap_degress` and `min_flap_degress`.
		
		`use_flap` must be enabled.

`ncrit_pt(i)`
 	
`where i = 1..noppoint`
	optional
	
	advanced Hicks-Henne
		= `x.y` `(Default ncrit)`
		
		Specifies ncrit for this operating point. If this parameter is omitted the default value from `&xfoil_run_options` (default ncrit = 9) will be used.

`&geometry_targets`	optional	Definition of the operating points
`ngeotargets`	optional JX	`= X` Number of geometric targets that shall be applied Same as operating points, geometric targets are part of the objective function. So the optimizer will try to satisfy geometric targets equal to aerodynamic aspects of the operating points. Tip: Use geometry targets instead of thickness or camber constraints as they work together more natural with aerodynamic targets..

`target_type(i)` `where i = 1..ngeotargets`	optional JX	Specifies the optimization objective for this geometry target = `'thickness'` Final thickness of the airfoil. = `'camber'` Final camber of the airfoil.
`geo_target(i)` `where i = 1..ngeotargets`	optional JX	= `x.y` The target value to achieve. A negative value for this parameter will be taken as a (positive) factor to the value of the seed airfoil. Tip: When tweaking existing airfoils these “negative factors” are perfect to freeze the geometry to the current values.
`weighting_geo(i)` `where i = 1..ngeotargets`		= `x.y` `(Default 1.0)` Weighing of this target within the objective function. Same meaning as ‘weighting(N)’ for an operating point. Tip: When having target based operating points, “dynamic weighting” allows to forget this parameter in most cases. → see `dynamic_weighting`
`&constraints`	optional	Specifies geometric constraints for the optimization
`check_geometry`	optional JX	= `<Boolean>` `(Default .true.)` Disabling all geometry checks may improve optimization time. in case of Hicks-Henne shape functions optimization time could be worsen because Xfoil may have too many operating points not converging because of strange geometry. Parameter `symmetrical` won’t be influenced by `check_geometry`.
`min_thickness` `max_thickness`	optional	= `x.y` `(Default 0.01)` The minimum / maximum thickness of a design as a fraction of the chord (for example, 0.08 means 8% thickness. Tip: Better use geometric targets to control geometry.
`min_camber` `max_camber`	optional	= `x.y` `(Default -0.1)` The minimum / maximum camber of a design as a fraction of the chord (for example, 0.02 means 2% camber. Tip: Better use geometric targets to control geometry.
`max_flap_degress`	optional	= `x.y` `(Default 15.0)` Maximum flap angle, in degrees, for `` `optimize' `` flap selection type in `&operating_conditions`. Positive degrees corresponds to a downward deflection.
`min_flap_degress`	optional	= `x.y` `(Default -5.0)` Minimum flap angle, in degrees, for `` `optimize' `` flap selection type in `&operating_conditions`.

		Negative degrees corresponds to a upward deflection.
`symmetrical`		= `<Boolean>` `(Default .false.)` Whether to only generate symmetrical airfoils. If true, only the top surface of the seed airfoil will be modified, and the bottom surface will be replaced by a mirrored version of the top surface. The seed airfoil does not need to be symmetrical as it will be also mirrored in the beginning.

`check_curvature`
`max_curv_reverse_top`
`max_curv_reverse_bot`
`curv_threshold`
	deprecated	This parameters moved to `&curvature` namelist.

`&curvature`	optional Hicks-Henne	Specifies options to control surface curvature
`check_curvature`	optional *Hicks-Henne*	= `<Boolean>` `(Default .true.)` When enabled the curvature, which is the 2nd derivative of the airfoil polyline and the 3rd derivative are checked against constraints. One important attribute of the airfoils shape is the existence of curvature reversals either on the top or the bottom side. A reversal means the shape is changing from concave to convex or vice versa which is a fundamental characteristics of an airfoil. Tip: See the remarks regarding airfoil and reversals for the parameter `airfoil_file` Note: Disabling `check_curvature` will switch off all of the checks in this namelist.
`auto_curvature`	optional	= `<Boolean>` `(Default .true.)` “Auto curvature” tries set curvature parameters in a best possible way. To do this, the parameters are applied to the seed airfoil and decreased iteratively until they are constrained by the values of `max_curv_reverse` or `max_te_curvature`. The new, optimized values are shown when starting the program.
`curv_threshold`	optional	= `x.y` `(Default 0.1)` Threshold of the curvature curve along x/c to detect curve reversals. If the threshold is too high, reversals won’t be detected. A too low threshold on airfoils with a bad surface quality will lead to a lot of “false” reversals.
`max_curv_reverse_top` `max_curv_reverse_bot`	optional	= `n` `(Default 0)` The maximum number of reversals the surface of the airfoil should have on top and bottom side. One reversal on top side is typical for airfoils for flying wings (“reflexed airfoil”). Higher cambered airfoils will have a reversal on the bottom side.
`spike_threshold`	optional	= `x.y` `(Default 0.4)` Threshold to detect reversals of the 3rd derivative of the airfoils surface. The 3rd derivative is like a magnifier to detect bumps of the curvature (2nd derivative). The other use case: Airfoils with only 3 or 4 decimals in the .dat-file will have a “zick-zack” 3rd derivative curve – this “zicks” are called spikes. The smaller the value the more spikes will be detected. Xoptfoil-JX built-in smoothing algorithm allows greatly to reduce

		the number of spikes which is the precondition for advanced curvature control.
`max_spikes_top` `max_spikes_bot`	optional *advanced*	= `n` `(Default 0)` Maximum number of spikes on top and bottom side. See `spike_threshold` for more explanation. if `auto_curvature` is active, the best value will be set automatically.
`do_smoothing`		= `<Boolean>` `(Default .false.)` Xoptfoil-JX smoothing of the airfoil surface uses a modified “Chaikin corner cutting algorithm” which is quite restrained in modifying the shape but efficient in reducing spikes (see `spike_threshold` for further explanation). Even if not activated explicitly, smoothing will be done if the quality of the seed airfoil is not good enough (too many spikes) as a base for Hicks-Henne optimization.
`max_te_curvature`	optional *advanced*	= `y` `(Default 10.0)` Maximum value of the curvature at trailing edge. In the current Xoptfoil Hicks-Henne shape functions implementation, the last panel is forced to become the original trailing edge value which may lead to a thickened trailing edge with steep last panel(s). An artefact of a deformed trailing edge can also be seen in a lot of existing airfoils. These “mini spoiler” have an influence on Xfoils results and are difficult to see in the original coordinates. The origin is quite often in the “Xfoil inverse design” where the airfoil shape is generated from the pressure distribution. Tip: Have a look at the curvature plot in the visualizer to assess the situation. A value of 0.1 – 1.0 leads to well-formed trailing edges. The high default value of 10 will switch off the curvature check more or less. The `auto_curvature` mode will automatically set the trailing edge curvature to the lowest (best) value based on the seed airfoil.
`max_spikes_top` `max_spikes_bot`	optional *advanced*	= `n` `(Default 0)` Maximum number of spikes on top and bottom side. See `spike_threshold` for more explanation. if `auto_curvature` is active, the best value will be set automatically.

`highlow_threshold`
`max_curv_highlow_top`
`max_curv_highlow_bot`
	deprecated	…

`&initialization`	optional	Specifies options for initialization the designs of the particle swarm.
`feasible_init`	optional *advanced*	= `<Boolean>` `(Default .true.)` If enabled, tries to create valid initial designs for each particle of the swarm. This process increases the overall expense of the optimization but also improves the results because more of the initial designs will be good ones.
`feasible_init_attempts`	optional	= `n` `(Default 1000)`

	advanced	Number of attempts to try to make an initial design feasible.
&polar_generation	optional	Generate a polar set in xfoil-format for each new design
generate_polar	optional	= <Boolean> (Default .true.) If enabled, a polar will be generated for each new design. The polar will be displayed in the visualizer and can be red directly into Xflr5 or Flow5 (“Import Xfoil Polar”). The polars will be stored in the subdirectory <airfoil-name>_polars.
type_of_polar	optional	= < 1 | 2 > (Default 1) Defines the type of polar -either “Type 1” (fixed speed) or “Type 2” (fixed Lift).
polar_reynolds	optional	= xxxxx, yyyyy, zzzzzz, … (Default re_default) List of Reynolds numbers, polars should be generated. If polar_reynolds is omitted re_default is checked to get a valid Reynolds number. Note: Within Xoptfoil-JX only a single polar can be generated for each new design. Xfoil_worker allows to generate a complete set of polars.
op_mode	optional	= <'spec-cl' | 'spec-al'> (Default 'spec-al') Defines whether the sequence defined in op_point_range is based on alpha or cl.
op_point_range		= <start>,<end>,<increment> (Default -2,10,0.5)) Defines the sequence of operating points with the polar(s)
&matchfoil_options	optional	Options for testing the optimization algorithm
match_foils	optional	= <Boolean> (Default .false.) If .true., the optimizer will try to match the seed airfoil shape to another known airfoil shape, and no aerodynamic calculations will be performed. Xfoil is never run. It's about a million times faster than aerodynamic optimization, which makes it good for testing things.
match_foil_file		= '<filename>' (Default .false.) File name of the airfoil .dat-file which is to be matched by the optimizer. If the file is not in the working directory, you must include the relative path. Tip: A special test case is to take the seed airfoil as the match foil. If the two airfoils are equal the optimizer is initially “told” that they differ. it's a little unfair to the particle team ...
&xfoil_paneling_options	optional	Xfoil options for paneling – see also org. Xfoil documentation
repanel	optional *advanced*	= <Boolean> (Default .false.) If enabled, each time a new design will be evaluated by Xfoil the airfoil will be repaneled in advance. If the seed airfoil already has a good quality of the curvature (e.g. little number of “spikes”) it is

	Hicks-Henne	not necessary and helpful to repanel the airfoil all the time.
`npan`	optional *advanced*	= `n` `(Default 200)` At the very beginning of an optimization the seed airfoil will be repaneled using Xfoils PANGEN routine to get standardized point distribution of the surface. The number of data points (npan) is fixed to 200. Afterwards the seed airfoil will be 'normalized' to have the leading edge at x,y = 0,0 and the trailing edge at x,y = 1,0. In case of a trailing edge gap the mean value of top and bottom surface is applied. Typically there is no real leading edge point when Xfoils repaneling is done. In this case an additional real leading edge point at 0,0 will be inserted into the data points. Therefore most Xoptfoil generated airfoil .dat files will have 201 data points also 200 points are used for repaneling.
`cvpar`	optional *advanced*	= `y` `(Default 1.0)` Panel bunching parameter. Increasing this number will cause more panels to be bunched in regions of high curvature. 1.0 is the default.
`cterat`	optional *advanced*	= `y` `(Default 0.0)` Ratio of trailing-edge to leading-edge panel density. Note: If set to 0.15 which is the default of Xfoil, the curvature at trailing edge panels tends to flip away and have tripple value (bug in Xfoil). With a very small value the panel gets wider and the quality better.
`cttrat`	optional *advanced*	= `y` `(Default 0.2)` Ratio of regular panel density to refined area panel density..
`&xfoil_run_options`	optional	Xfoil options to control aerodynamic calculations
`ncrit`	optional	= `y` `(Default 9.0)` Transition-triggering parameter. A higher number represents cleaner (less turbulent) freestream conditions. This is the default value for the calculations at operating points. It may be overwriten by `ncrit_pt(i)`
`xtript` `xtripb`	optional	= `0.x` `(Default 1.0)` Trip location for turbulent transition on top / bottom side of the airfoil. For free transition, set this to 1.
`bl_maxit`	optional	= `n` `(Default 40)` Maximum number of Xfoil viscous iterations. A higher value increases the probability that the iteration may converge – but also increases computation time. If Xfoil fails, it typically indicates a strange shape design or an operating point with a (too) high angle of attack or cl-Value. During optimization the iteration fails are indicated with a "red **x**" particles result line.
`vaccel`	optional	= `0.0y` `(Default 0.005)` Xfoil viscous convergence acceleration parameter. If you get too many iteration fails you may try to reduce `vaccel` a little.
`fix_unconverged`	optional	= `<Boolean>` `(Default .true.)`

		If enabled, up to 3 retries with a little increased Reynolds number and reinitialized boundary layer are performed to reach convergence at an operating point.
`reinitialize`	optional	= `<Boolean>` `(Default .false.)` If enabled, the boundary layer is reinitialized at each operating point. If disabled: As the retry mechanism usually works quite well, this time consuming option is switched off beside some certain cases where reinitialization may be useful.
`&particle swarm options`	optional	Options to particle swarm optimization
`pso_pop`	optional	= `n` `(Default 40)` Number of particles. These are randomly initialized throughout the design space and then use swarming behavior to converge on the global optimimum. Search cost increases linearly with population size. As the particle swarm needs iterations to learn (from each other) it is not helpful to increase the population to reduce iterations needed. Typically around 40 seem to work well.
`pso_tol`	optional	= `y` `(Default 0.0001)` Tolerance in max radius of particles before triggering a stop condition. When all particles are within this radius of the population center, the particle swarm optimization process will stop even if the max number of iterations has not been reached. The current max. radius is displayed during optimization. Tip: It may be more efficient to limit optimization time with `pso_tol` than with the number of iterations. Have a look at the optimization history in the visualizer.
`pso_maxit`	optional	= `n` `(Default 600)` Maximum number of iterations allowed before the particle swarm optimization is stopped. Using the `quick' convergence profile, the optimization will usually converge in less iterations. However, if you use the `exhaustive' profile, it likely will not converge within the default 700 iterations, but the airfoil design will probably not improve much after that (see pso_tol)

`pso convergence profile`	optional (JX)	= `<'exhaustive' | 'quick' | 'quick_camb_thick'>`
		
		`(Default 'exhaustive' for hicks-henne`
		`'quick_camb_thick' for camb-thick)`
		
		This setting adjusts some parameters in the particle swarm algorithm which affect how quickly the optimization converges. The `'quick'` profile works well and converges quickly. The `'exhaustive'` profile usually is able to find a bit better designs than the quick prole, but it also takes much longer to converge (e.g., 700 iterations instead of 150). The `'quick_camb_thick'` profile is especially for shape functions `'camb_thick'` and results in lowest iterations needed.
		
		Tip: Use `'quick'` for the first evaluations and switch to `'exhaustive'` in a later phase of airfoil fine-tuning.

Appendix A: Airfoil curvature and geometric quality

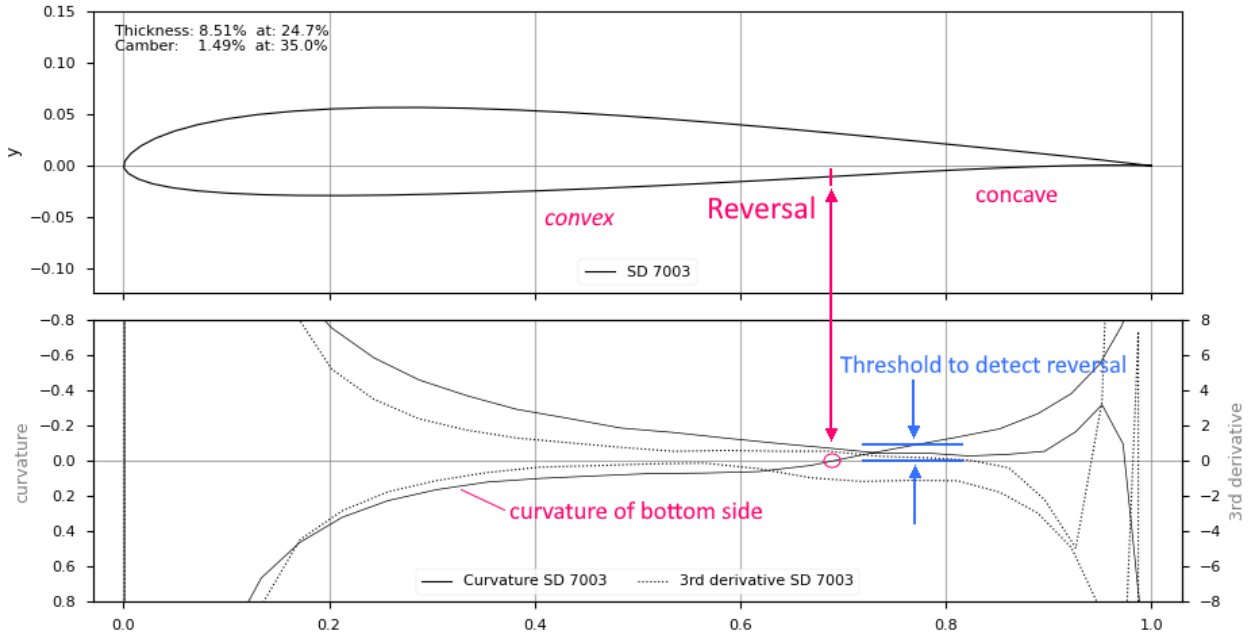
Curvature and Reversals

The curvature, which is the 2nd derivative of the airfoils contour line, is one of the most important aspects to describe the properties of an airfoil. The curvature highly impacts the pressure and velocity distribution along the chord.

Between the very high curvature at the leading edge and the low value at the trailing edge some airfoils do have a reversal of the curvature where the curvature changes from positive to negative – or vice versa.

Such a curvature reversal is typical for certain type of airfoils like “reflected airfoils” for flying wings having a curvature reversal on the top side of the airfoil.

For airfoil optimization the existence of curvature reversals help to classify an airfoil into a certain category – so reversals are typically used as a geometric constraint for the optimization: Should the airfoil have a reversal on the top or bottom side? If yes, how many?



Reversals are detected with a simple algorithm: Going from the leading edge to the trailing, did the curvature change its sign? Is the new value higher than the “curvature threshold” value? Yes – curvature found.

It’s important to understand this detection of curvature reversals as it may become tricky with real life airfoils when we do not have the perfect quality of the airfoils contour line…

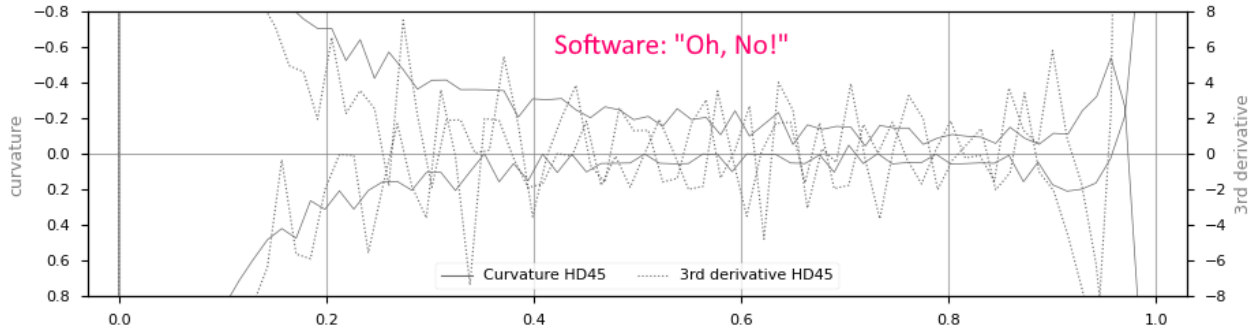
Real life airfoils – About bumps and spikes

When optimizing an airfoil, a typical approach is to take an existing airfoil and to tweak it until it has the desired aerodynamical properties.
Except some extreme outliers a typical airfoil looks perfect and smooth on the screen:

But oh! When starting the optimizer, the program reports something like “Detected 13 reversals on top side”.

What is the reason for this?

The software does have a different look on the airfoil. This is, what the “software sees” when the airfoil is evaluated for optimization:

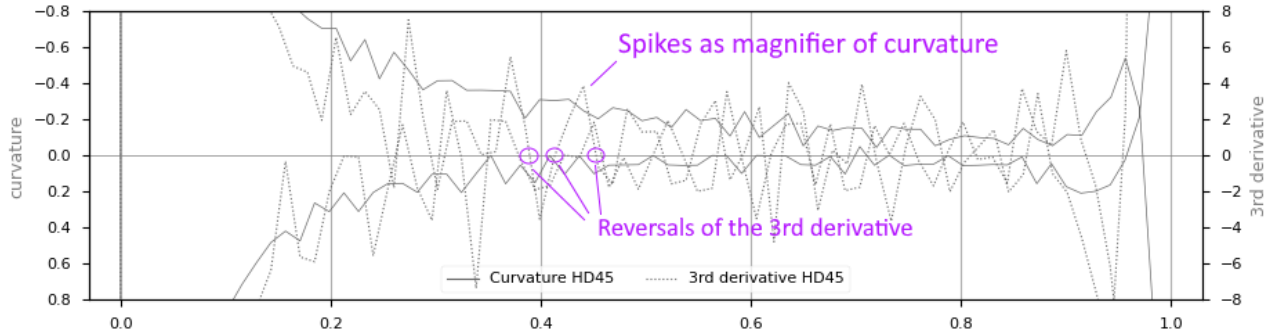


The curvature line isn’t smooth anymore but a zick-zack line of up and downs. A common reason for this curvature artefact is just because the airfoil coordinates do have only 5 decimals in the “.dat – file” which is not exact enough to calculate a smooth curvature (2nd derivative) out of these points.

It is obvious that this “zick-zack” makes it much more difficult to distinguish between a “zick-zack” and a real reversal of curvature. On way to solve this, is “smoothing” the airfoils geometry… (see below).

Note: Xfoils aerodynamic calculation is quite robust regarding these numerical inaccuracies of the coordinates. In some cases these inaccuracies lead to even better aerodynamic characteristics (typical: higher maximum cl) compared to a real smooth airfoil (high angle of attack, improved maximum cl).

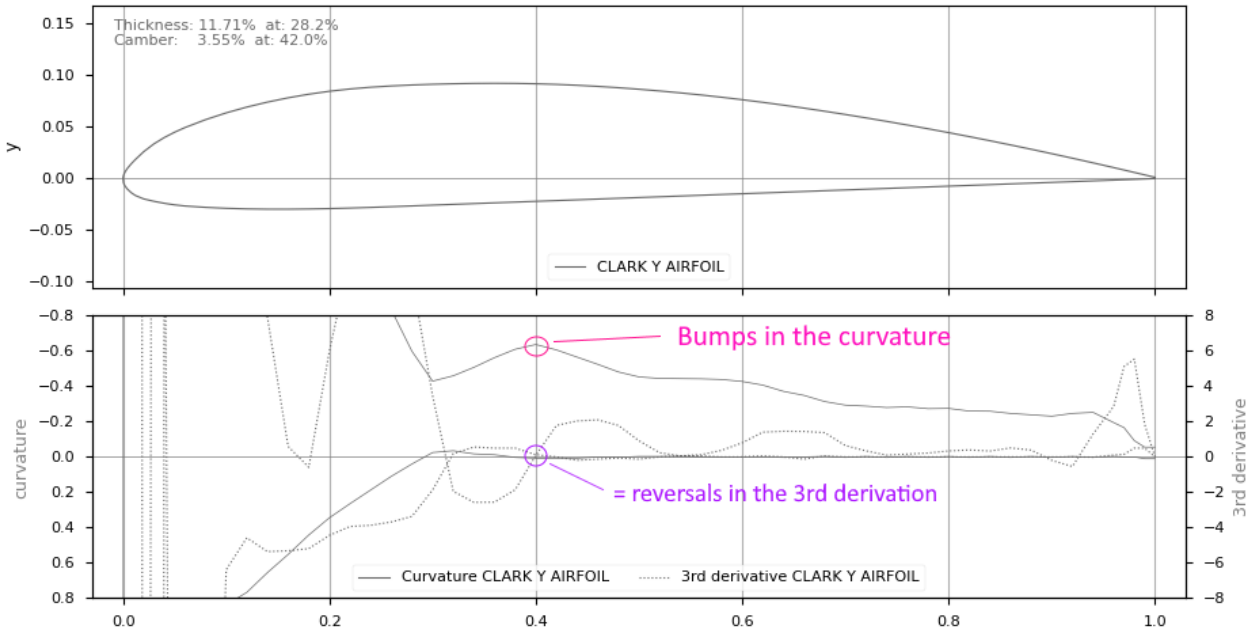
For detection of these geometric artefacts the 3rd derivation of the airfoils contour line is used. The 3rd derivative acts as a magnifier of the curvature. Smaller up and downs of the curvature will become reversals of the 3rd derivate with high peak values. These 3rd derivative peaks are called “Spikes”. The number of spikes detected serves as a measure of the quality of the airfoils contour line:



Beside these numerical artefacts because of too few decimals, there may also be bumps and discontinuities of the curvature which have their reason in

- humans changing some airfoil coordinates manually
- humans with software e.g. doing inverse airfoil design in Xflr5
- algorithms or polynomial expressions like the NACA airfoil family.
- optimization with Hicks-Henne shape functions

Here's an example of a curvature bump at the classic CLARK Y airfoil.

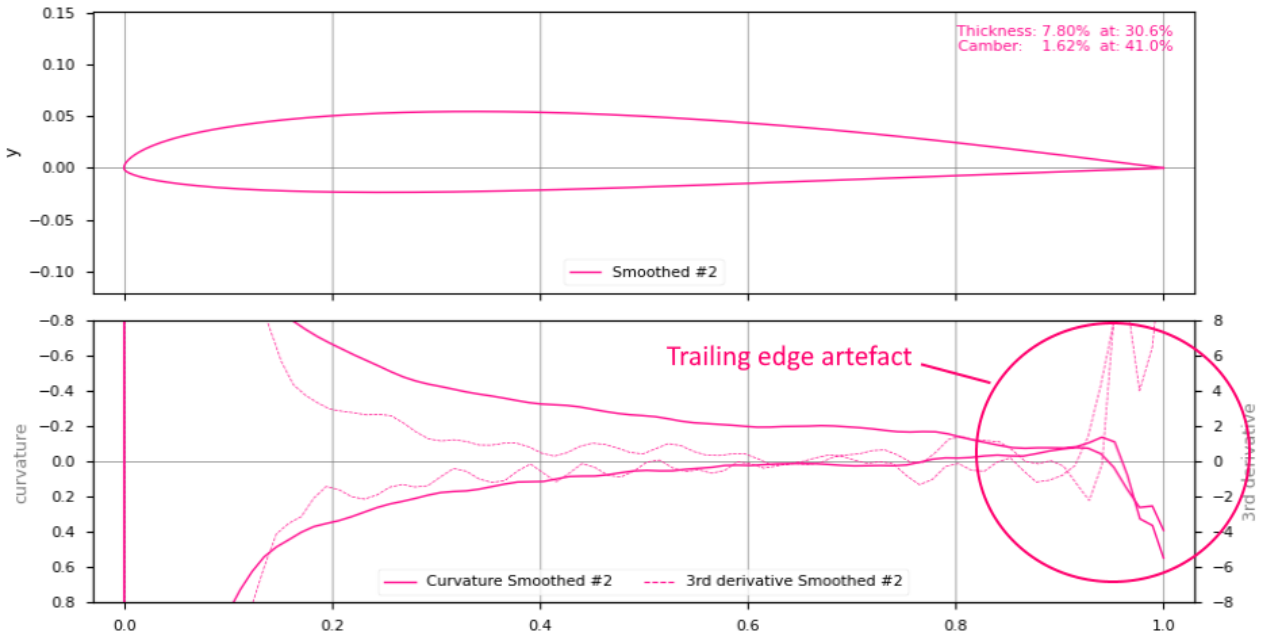


General Note: Because airfoil shape modification during optimization is either "additive" (applying Hicks-Henne shape functions to the airfoil) or "deforming" (changing e.g. camber or thickness of the airfoil) the resulting airfoil will still have the artefacts of the original airfoil. The quality of the curvature cannot be improved by optimization.

Trailing edge spoiler and artefacts

Beside "bumps" and "spikes" on the contour line there is finally a typical third artefact which has to be taken in account: The trailing edge artefact. The bandwidth of this artefact can be from quite small (can be ignored) to "amazing" (having some influence on the airfoils aerodynamic characteristics). Again the curvature helps to detect and identify these trailing edge artefacts.

This is a typical example which can be found on in a similar way on many airfoils. Near the trailing edge the top side is changing with a reversal from convex to concave:



There may be another reason for trailing edge artefacts:

When making optimizations with Hicks-Henne shape functions, it is almost certain, that the particle swarm will find “strange” ways in moving the point of laminar-turbulent transition as far as possible towards the trailing edge – leading to very high curvature values close to leading edge.

This misbehavior can be suppressed by controlling the maximum value of the curvature near the trailing edge with the parameter “`max_te_curvature`”.

Note: As already stated for “bumps” and “spikes” the final curvature at trailing edge cannot be smaller than the value of the seed airfoil. Therefore check the seed airfoil if it is really a good base for the optimization task.

Auto setting of curvature parameters

There is a special option called “`auto_curvature`” which tries to set the thresholds and constraints for curvature control to the best value for perfect airfoil shape quality.

When this option is activated, the seed airfoil will be assessed and the thresholds for curvature and 3rd derivation set to the lowest value to detect constraints safely.

During optimization each design will be checked regarding curvature reversals, bumps (reversals of the 3rd derivative) and trailing edge artefacts if the parameter “`check_urvature`” is also activated.

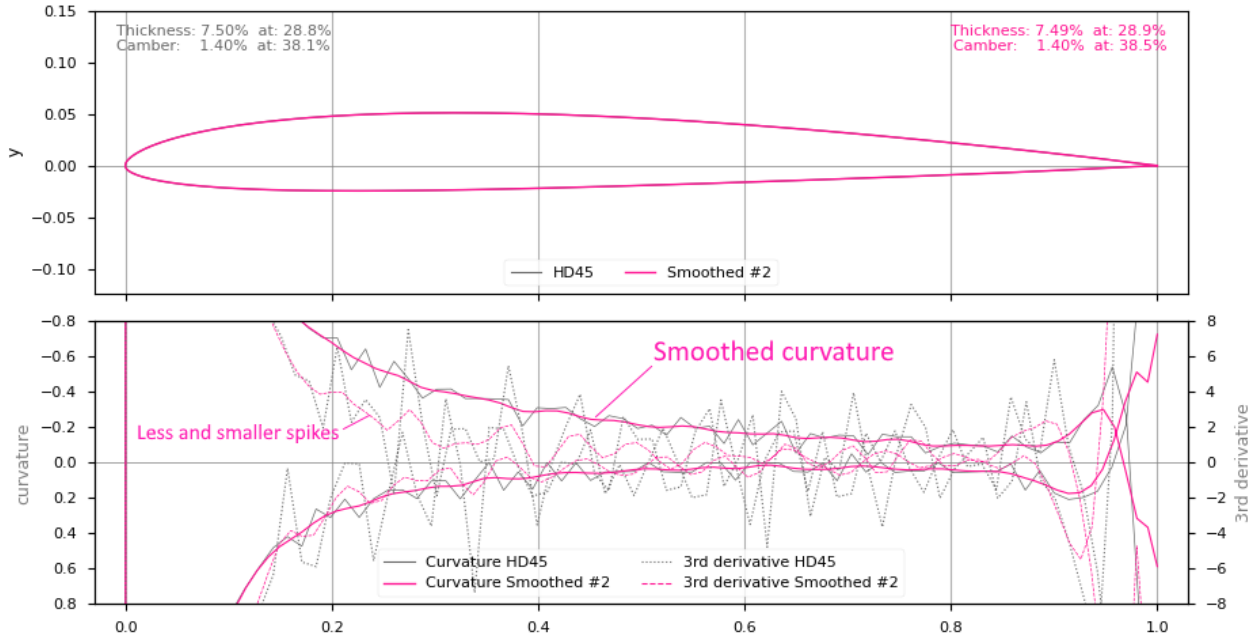
Tip: Switch off “`auto_curvature`” only if you have a good understanding of parameters envolved.

Improving the airfoil surface quality: Smoothing

Xoptfoil-JX has a built-in smoothing algorithm which greatly can minimize “spikes” and smooth the curvature especially for airfoils which have their origin in “low decimals .dat files” (see explanation above)

The implemented smoothing algorithm (‘Chaikin Corner Cut’) is quite defensive in changing the airfoils surface. It is able to smooth discontinuities between neighboring data points but it is not able to remove real bumps in the surface of the airfoil. It can be compared to very fine sand paper applied to the airfoil.

Here is an example of an airfoil before and after smoothing was applied:



When “`check_curvature`” is active – which is the default for Hicks-Henne shape functions – smoothing will be applied automatically if the assessment of the seed airfoil detects too many “spikes” either on top or on bottom side of the airfoil.

Tip: You may use the tool “Xfoil_Worker” if you just want to smooth an airfoil without doing optimizations.

Well suited seed airfoils

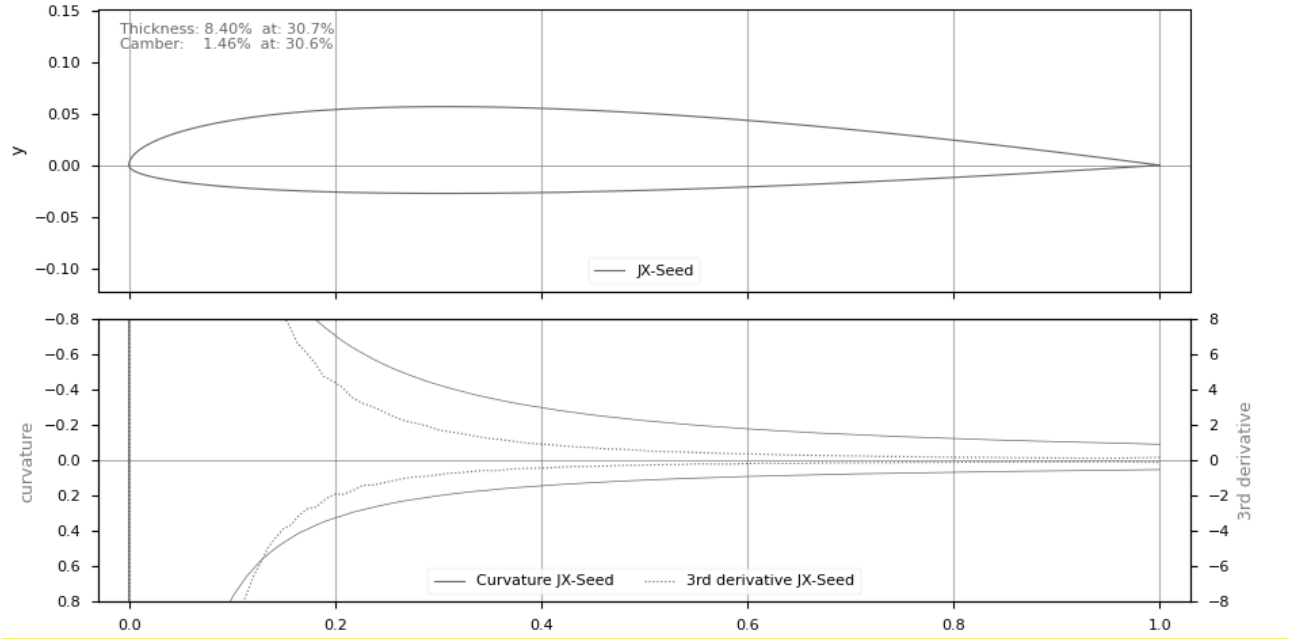
If it’s not the task to tweak an existing airfoil for certain aerodynamic aspects but rather to design a new airfoil, it is highly recommended to use a geometric “clean” airfoil as the base for optimization.

For Hicks-Henne based optimizations there are some well prepared “artificial” airfoils in the .\examples subdirectory:

- JX-Seed.dat Having no reversal either on top or bottom side.
- JX-Seed-Rearload.dat Having one reversal on bottom side
- JX-Seed-S.dat Having on reversal on top side.

Take one of these depending on the “max_reversal”-constraint the new airfoil should satisfy.

These airfoils are created with respectively one spline for top and bottom side having minimum control points leading to a very smooth curvature:



Appendix B: Operating points and use of targets

Choosing the right operating points

to be written

Polar by design using targets

to be written

Appendix C: Examples

Tweaking an airfoil

to be written

Advanced airfoil design from scratch

to be written