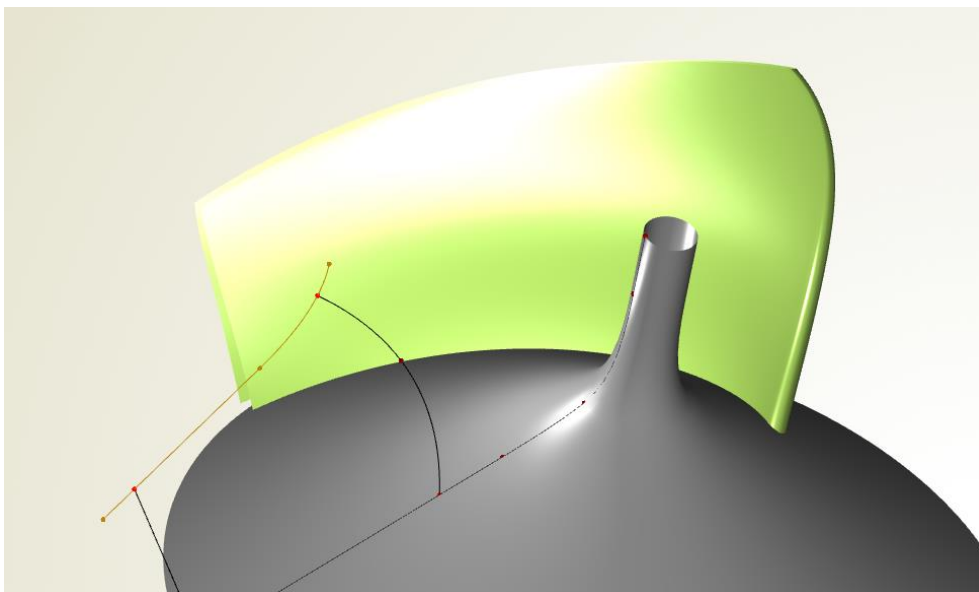


Impeller Design

This tutorial gives you a brief overview of modeling an impeller blade, e.g. for water pumps and turbochargers. It requires knowledge about the *meta surface* techniques of CAESES. Please check out the meta surface introductory tutorials for a better understanding.

The design details of the hub and shroud contours are described as well as things to consider for the leading and trailing edge contours. The blade design is fully parametric, and based on angle distributions (theta/beta). In this tutorial, the thickness is applied normal to a generated camber surface. The thickness function for each section is smartly controlled by spanwise functions.

Finally, the blade is merged with hub and shroud to demonstrate some of the *brep* capabilities, such as extending a surface into the hub and shroud region, and Boolean Operations. This is an issue when setting up the CFD domain of a blade model where a closed and periodic geometry is required, as well as inlet and outlet geometries.



CAESES Project

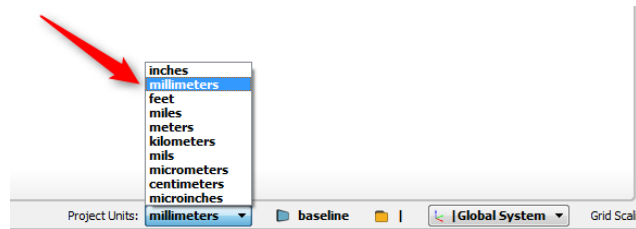
The resulting model can be found in the section *samples > blade design* of the documentation browser.

1

Getting Started

In this preparation step, we make sure that we use the correct project units. In addition, we get in touch with the zx-view which is the main 2D view for designing the blade's hub and shroud contours.

- Switch the project units to millimeters.



✓ Projects units are relevant when it comes to geometry exports at a later stage.

- Check out the zx-view button at the 3D view which always brings you back to the main hub and contour view in the upcoming steps of this tutorial.



✓ In addition to the zx-view button, you can use the camera positions in the toolbar (upper right corner of the GUI) in order to store and reuse your favorite views.



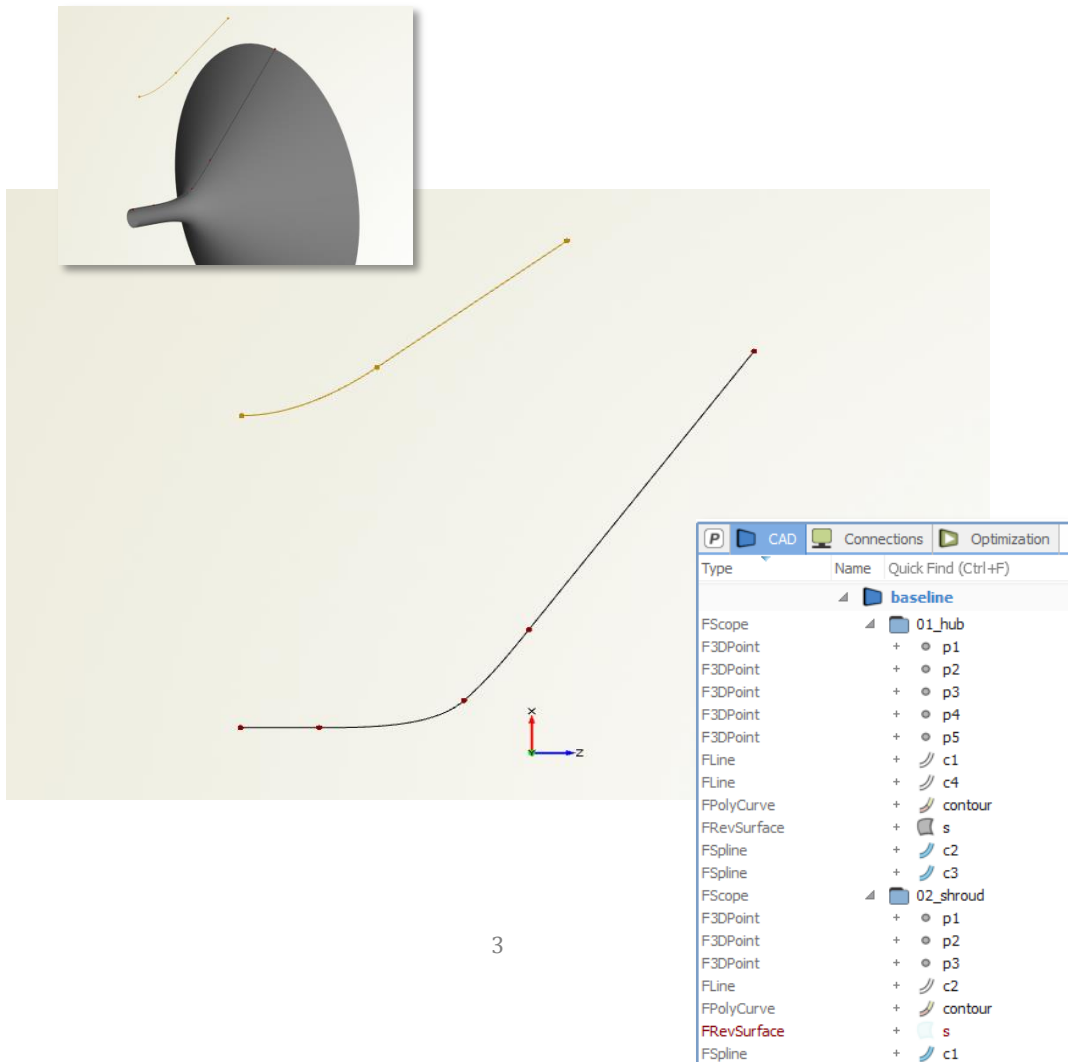
2

Hub and Shroud Design

First, you need to create 2 planar curves in the zx-view, one for the lower and one for the upper stream surface, i.e. for hub and shroud.

✓ In CAESES, the flow direction is assumed to be in the positive z-direction. The orientation of your hub and shroud contour curves needs to be running in positive z-direction.

- ▶ Make sure that you are in the zx-view, and model your curves in this 2D system!
- ▶ You can make use of any curve type of CAESES for this 2D design.
- ▶ Create surfaces of revolution that are based on the hub and shroud contour.
- ▶ Put hub and shroud geometry into separate scopes for a clearer structure of your project.
- ▶ You should introduce parameters for main dimensions, and use them in your points and curves to create everything in a parametric manner.
- ▶ Don't forget to save your project ☺ ("save as" ...)

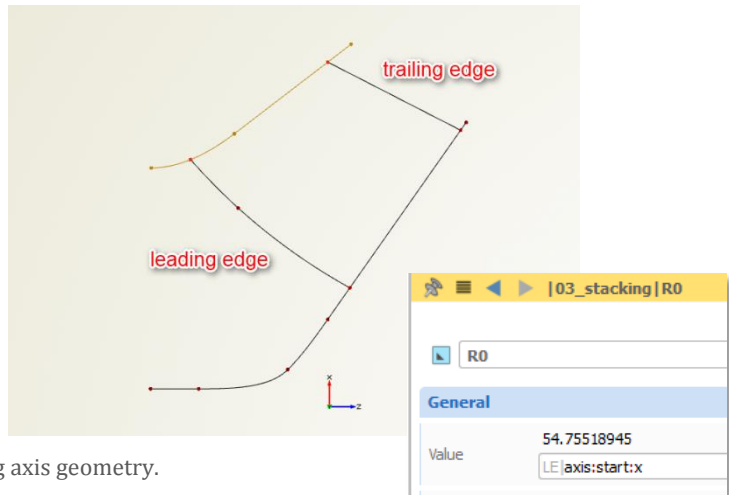


3

Stacking Axis

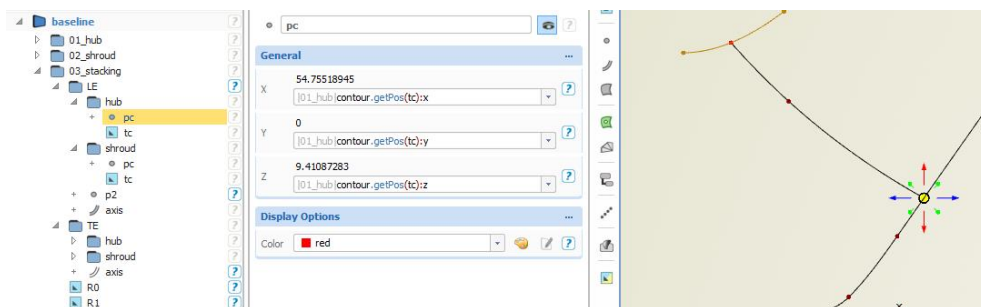
The stacking axis (or leading edge contour curve) is the next step in the design process. Optionally, a trailing edge contour can also be provided:

- ▶ Both the leading and the trailing edge curve can have an arbitrary shape.
- ▶ These curves need to intersect with the hub and shroud contours.
- ▶ Typically, both curves are in the zx -plane i.e. they are 2D curves.
- ▶ The trailing edge curve is optional.
- ▶ Instead of a 2D curve, you can also design a 3D stacking axis for the leading edge. This is common for the design of axial blades.
- ▶ Store the radii (i.e. the x -values) for $R0$ and $R1$ of each axis – you will need them later on.
- ▶ Create a new scope for the stacking axis geometry.



✓ Here is a convenient way of locating a point on the hub or shroud surface:

Just select the hub contour curve and choose *CAD > points > 3D point*. This automatically creates a point on the curve. You can change the location on the curve by using the parameter “tc” which gets also created automatically. With this, you can create 2 points, one on the hub and one on the shroud curve, respectively. These 2 points are then input for the leading edge contour (e.g. a straight connection, or the start and end position of a spline curve).

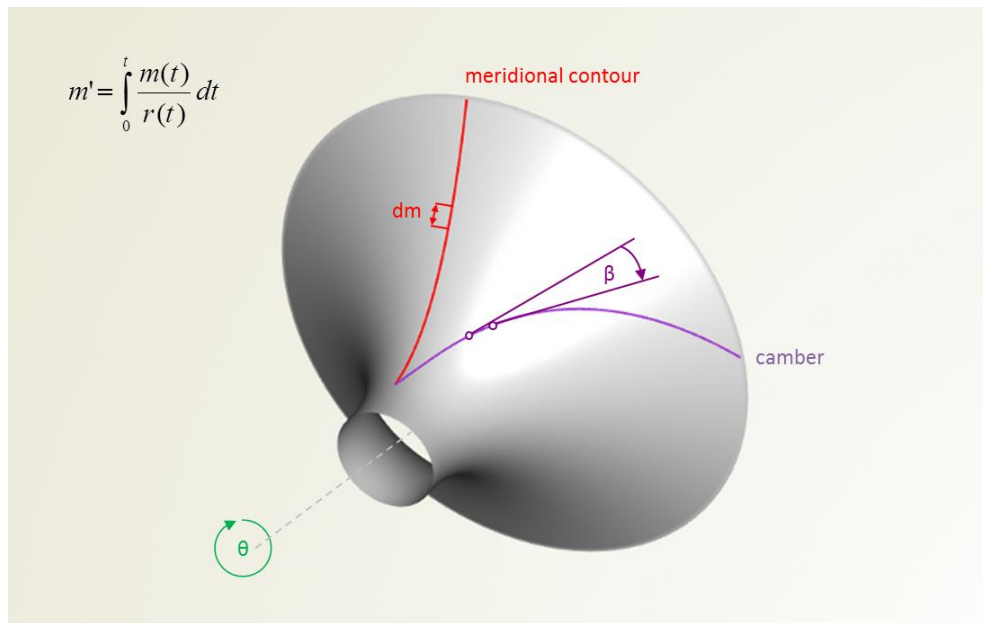


4

Camber Design: Stream Section

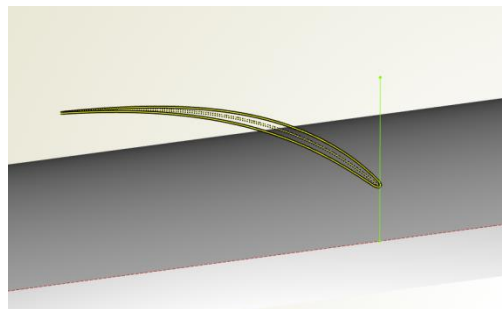
For generating a camber curve in the 3D space, we can use the *stream section*. The stream section is a curve type that transforms a 2D profile design into the 3D space. You can find this curve in the menu *CAD > blade > stream section*.

In this step, the stream section gets explained briefly before the tutorial is continued. See the illustration below for an overview of the design situation.



The stream section is based on a hub and shroud contour, a stacking axis, camber and optional thickness information. Camber data can be either designed and given in the (m', θ) or in the (m', β) system. In addition, the 2D systems $(m, r^* \theta)$ and (m', θ) as well as $(z, r^* \theta)$ are offered. All the transformations between the different 2D systems and the 3D space are done by the stream section curve.

On the next page, the settings of the stream section are discussed.



Design Mode

You can define your own 2D profile and provide it to the stream section (design mode “custom MRT” and “custom MPT”). Alternatively, you can use the camber and thickness capabilities of the stream section for which you have to choose “default”.

Definition

The default is a 3D definition (“XYZ”), but you can also switch to a (m, r, θ) view, in order to see how the 2D profile looks like.

Stream Lines

This is a list with the two curves for hub and shroud. Typically, you have only 2 curves, but you can also provide more curves. See step 2 of this tutorial where these curves are described.

The screenshot shows the 'FStreamSection' dialog box in CAESES. The title bar indicates the current project is '04_camber | testing | streamsection'. The dialog is divided into several sections:

- General:**
 - Design Mode: default
 - Definition: XYZ
 - Stream Lines: [01_hub | contour, 02_shroud | contour]
- Stacking:**
 - Axis: |03_stacking | LE | axis
 - Radius: 54.75518945
 - Theta: 0
- Camber Data:**
 - Mode: theta
 - Curve: |meanline_theta
 - Factor: 100
 - TE Cut Contour: |03_stacking | TE | axis
- Thickness Distribution:**
 - Curve: (empty)
 - Blend Position LE: 0.01
 - Blend Position TE: 0.99

Stacking: Axis

See again step 3 of this tutorial: For impeller design, this is usually a 2D curve that intersects with the hub and shroud contour. For axial blade design, this can also be a 3D-shaped curve.

Stacking: Radius

This is the absolute radius for the location of the stream section.

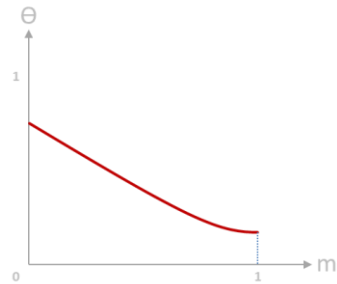
Stacking: Theta

This is an optional theta value which is added in order to rotate the stream section around the z-axis. Mostly used in combination with the camber mode “beta”, see below.

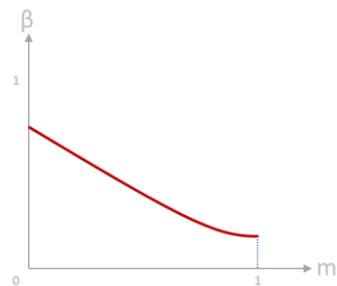
Camber Data: Mode

For impellers, the blade camber curve can be designed based on two angle distributions. The first option generates a curve by means of the wrap angle, while the second option uses a piecewise local angle on the camber to develop the curve.

- Option “theta”: 2D theta angle distribution given as a curve in the (m', θ) system:



- Option “beta”: 2D beta angle distribution given in the (m', β) system:



These distributions can be designed with any curve type of CAESES (e.g. line, fspline, bspline, interpolation curve). In the next step for generating a blade shape with a meta surface, the parameters of such a distribution are parameterized using a spanwise function.

Camber Data: Curve

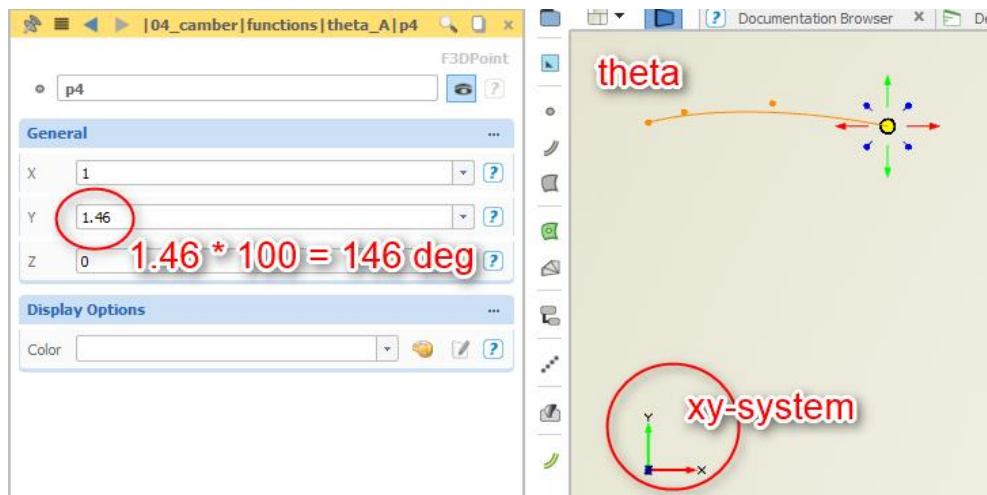
All angle distributions are modeled in the xy-system of CAESES, in the x-range $[0,1]$, i.e. from 0% to 100%:

This means that x corresponds to m' , while y corresponds to *theta* or *beta*!

The distributions are typically normalized so that they can be nicely viewed in the 3D window. Such a curve is designed with ordinate values that lie approximately in the range $[0,2]$.

At a later stage, the ordinate values of such a function are multiplied by a factor to reach the real values.

See the screenshot: The function is designed with an end point where the ordinate value is $y=1.46$. This corresponds to an angle of 146° , since we will specify a factor of "100" in the next step.



Camber Data: Factor

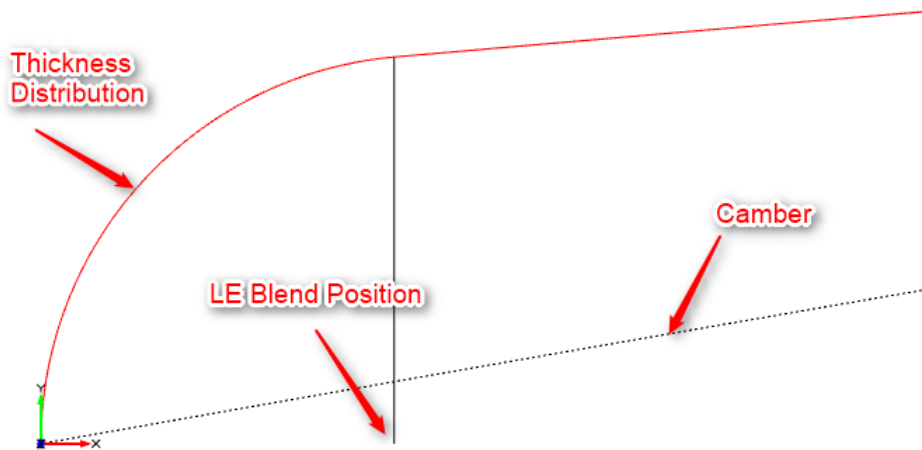
This is the factor for the ordinate values of the camber input curves from above. In the screenshot, this factor is "100".

TE Cut Contour

This is an optional 2D contour that describes the trailing edge of the blade.

Thickness Distribution: Curve

Optional 2D curve given in the xy-system – in the x-range $[0,1]$ – that defines a thickness distribution. If no curve is set, only the mean camber curve gets generated in the 3D space. The thickness is internally applied in the $(m,r*\theta)$ system. Note that other options are possible to apply a thickness distribution, which is also discussed in this tutorial (e.g. normal to a 3D camber surface).



Thickness Distribution: Blend Position LE/TE

These are the x-locations of the blend points of the leading and trailing edge. This information is used to split the section internally. With this, suction and pressure side can be generated independently from the leading and trailing edge.

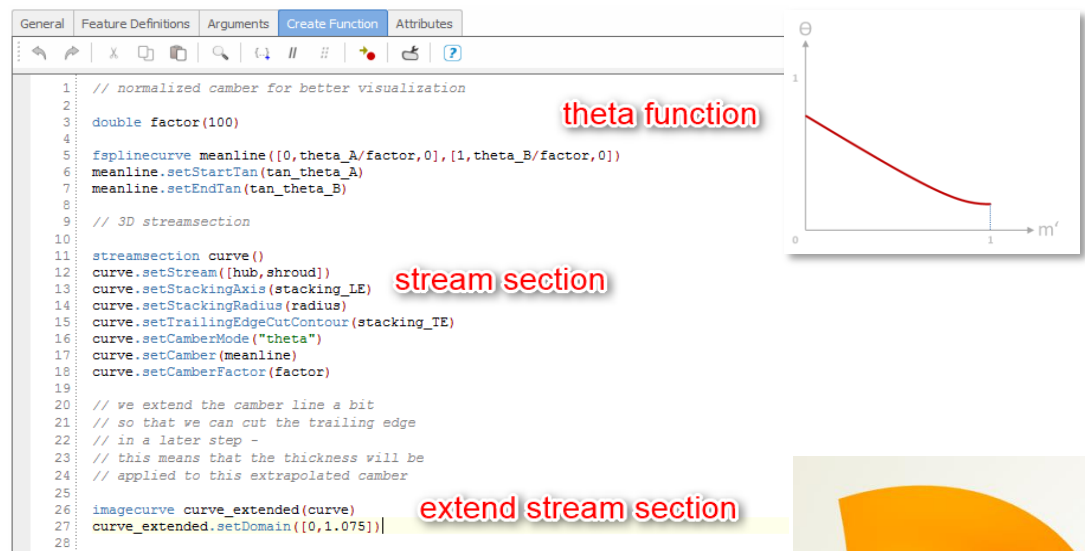
On the next page, we continue with the overall blade design process. Let's go...

5

Generate a 3D Camber Surface

For generating a parametric camber surface, the stream section from the previous step is used in a *meta surface*. Note that there are tutorials available that explain meta surfaces in detail, so please check them out beforehand. You really need to understand meta surfaces first, and then CAESES blade design is real fun!

In the following example, a smooth *fspline* curve (called “meanline”) represents the theta angle distribution. The start and end values as well as the tangents of this curve are controlled by the user, this is input to the feature definition, as well as hub and shroud and stacking information. The stream section is created and all attributes are set by using commands. Finally, the stream section is extended slightly to allow cutting it at a later stage.



The screenshot shows the CAESES software interface with the 'Create Function' tab selected. The code editor contains the following code:

```

1 // normalized camber for better visualization
2
3 double factor(100)
4
5 fsplinecurve meanline([0,theta_A/factor,0],[1,theta_B/factor,0])
6 meanline.setStartTan(tan_theta_A)
7 meanline.setEndTan(tan_theta_B)
8
9 // 3D streamsection
10
11 streamsection curve()
12 curve.setStream([hub,shroud])
13 curve.setStackingAxis(stacking_LE)
14 curve.setStackingRadius(radius)
15 curve.setTrailingEdgeCutContour(stacking_TE)
16 curve.setCamberMode("theta")
17 curve.setCamber(meanline)
18 curve.setCamberFactor(factor)
19
20 // we extend the camber line a bit
21 // so that we can cut the trailing edge
22 // in a later step -
23 // this means that the thickness will be
24 // applied to this extrapolated camber
25
26 imagecurve curve_extended(curve)
27 curve_extended.setDomain([0,1.075])
28

```

Annotations in red text highlight specific parts of the code and plots:

- theta function**: Points to the `fsplinecurve meanline` line.
- stream section**: Points to the `streamsection curve()` line.
- extend stream section**: Points to the `curve_extended.setDomain` line.

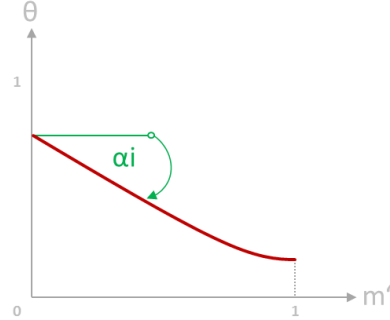
The top plot shows a graph of θ versus m' , with a red curve representing the theta function. The bottom plot shows a 3D model of a blade section with a highlighted orange stream section.

Things you have to do to generate a 3D camber surface:

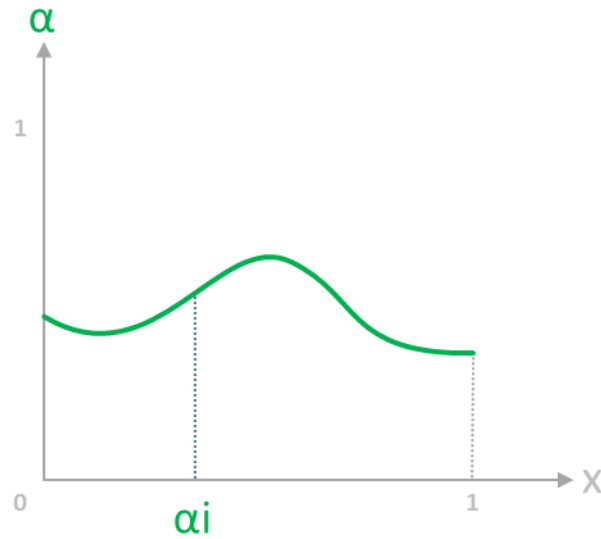
- ▶ Create a feature definition for the stream section curve.
- ▶ Include the beta or theta angle distribution in this feature definition, if possible. Feel free to set up your own parametric description for such a curve, it is up to you!
- ▶ Create a curve engine for this feature definition and the final generating curve.
- ▶ Create a meta surface that uses the curve engine. The radius is the running parameter which generates the surface. Generate the shape in between the hub and shroud radii.
- ▶ Create functions for the 2D theta/beta parameters for having full control of the 3D shape. This is the fun part, you will enjoy all the possibilities – see the next page for more information!

How to Control the 3D Shape

In the previous step, there is an example for a possible theta-distribution. The illustration next to this text shows such a function. The curve is smooth and gets controlled by few parameters, and parameter α_i is picked for demonstration purposes. This value is scalar input to the feature definition. The curve itself is usually modeled in the xy-space, where x can be interpreted as m' , and y represents θ .



For an entire 3D blade shape, we have i sections, and for each section a value α_i . As a consequence, we can easily set up another function which gives the parameter distribution of our parameter in spanwise direction:



This is the information that is used by a meta surface to generate the 3D shape, along with the 2D camber or profile information. Such a parameter curve is input for a *curve engine* object. The spanwise definition of a parameter is then controlled by the user to easily modify the blade shape. Since smooth definitions are used in general, this leads to feasible and smooth shapes, using a low total number of parameters for the entire blade.

This meta surface technique is applied for all the 2D blade profile parameters, so that, finally, there are several parameter curves that control the entire blade in spanwise direction. Great!

6

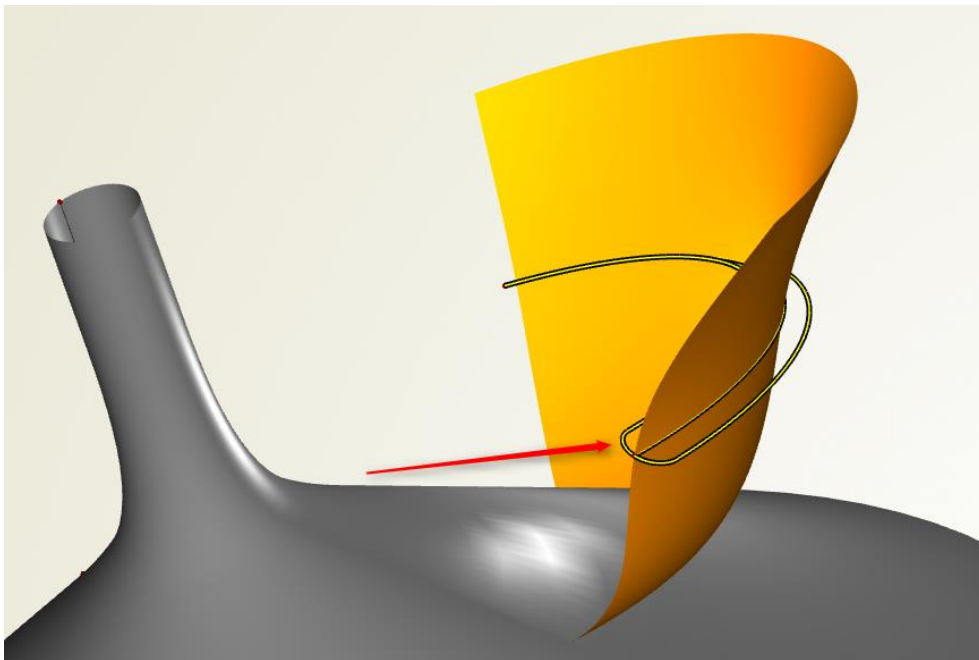
Thickness Distribution – Introduction

Instead of using the thickness capabilities of the stream section, we can apply a user-defined thickness function normal to the camber surface.

For a section on the camber surface at a specific location, a thickness distribution is defined. The thickness parameters are again controlled by user-defined spanwise distributions, and another meta surface is created again. This allows you to modify your blade shape with smart parameters and a reduced degree of freedom, while still generating only smooth and feasible shapes.

The picture below shows the creation of a 3D section based on a feature definition that is shown on the next page. Input for the feature definition is as follows:

- ▶ Camber surface
- ▶ V-location of the section on the camber surface, where v runs in the interval $[0,1]$
- ▶ Thickness values for the leading and trailing edge



```

1 // applies a thickness distribution to a meanline
2 // that is represented by a surface curve on the camber surface
3
4 // surface curve on camber surface
5
6 line domaincurve([0,vp,0],[1,vp,0])
7 domaincurve.setVisible(false)
8
9 surfacecurve meanline(camber, domaincurve)
10
11 // leading edge: almost flat -
12 // set it to 1 for a rounded LE and e.g 0.001 for a flat LE
13
14 parameter flatfactor(1)
15
16 parameter length(meanline.getLength())
17 parameter t2(thickness_A/2)
18 parameter x(t2/length*flatfactor)
19
20 // thickness definition is modeled in a normalized view
21 // so that it can be nicer visualized
22
23 parameter thicknessfactor(10)
24
25 translation dLE(x,0,0)
26
27 ellipse LE()
28 LE.setAxisA(x)
29 LE.setAxisB(t2/thicknessfactor)
30 LE.setStartAngle(90)
31 LE.setEndAngle(180)
32 LE.setTransformation(dLE)
33
34 fsplinecurve smoothjoin(LE:start,[1,thickness_B/(2*thicknessfactor),0])
35 smoothjoin.setStartTan(0)
36 smoothjoin.setEndTan(0)
37
38 polycurve tc([LE,smoothjoin])
39 tc.setAutomaticOrientation(true)
40
41 offsetcurve section()
42 section.setCurve(meanline)
43 section.setThicknessCurve(tc)
44 section.setThicknessFactor(thicknessfactor)
45 section.setBothSides(true)
46 section.setBlendPositionLE(LE:start:X)

```

In this feature, a surface curve gets created on the given camber surface. The input value “vp” specifies the location on the surface.

For the thickness function, an ellipse is created for the leading edge (LE), and a smooth *fspline curve* connects the leading edge with a zero tangent. The end tangent is zero, too. For a better visualization, the function is scaled down with a factor.

The two curves are put into a *poly curve* so that a single curve is available.

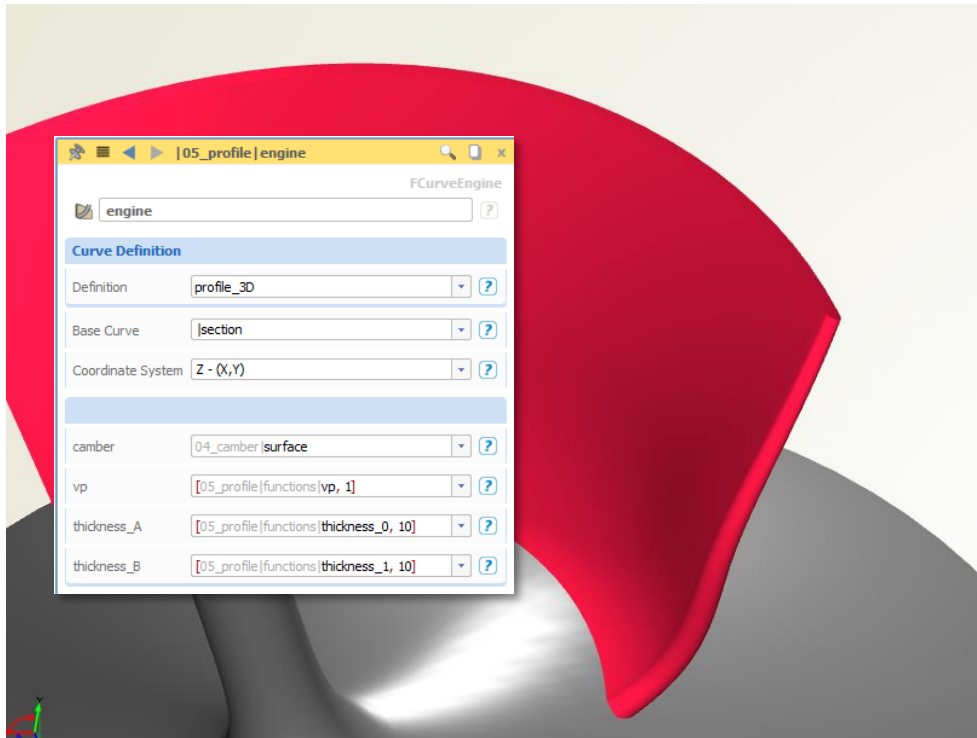


The offset curve applies the thickness to the 3D surface curve. This is done in normal direction with regards to the underlying input surface. Leading edge blend position and the factor are provided to this curve. That’s it! Now let’s take a look at the surface generation:

7

Thickness Distribution – Surface Generation

Similar to the camber surface, the final blade surface is once again generated by using a meta surface.



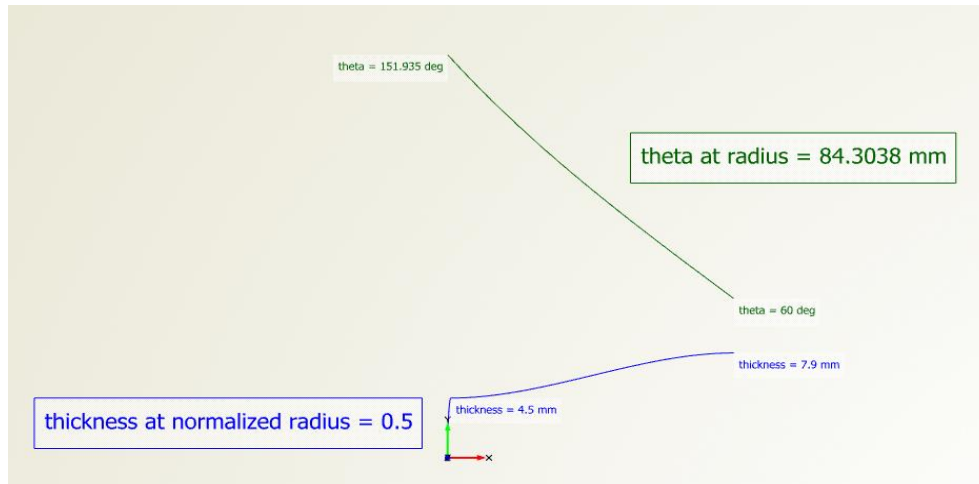
This is what you need to do:

- ▶ Create a *curve engine* for the feature definition with the final section that includes the thickness information (i.e. the final 3D offset curve).
- ▶ In the screenshot, the input “vp” is the running parameter that generates the surface. Note that we run in v-direction of an input camber surface. Typically, one would use a linear input function for it (i.e. a line with start = [0,0,0] and end = [1,1,0]).
- ▶ Create radial distributions for the thickness parameters which smartly control the 3D shape of the blade. In the screenshot, there are spanwise distributions for the leading edge and trailing edge thickness values. In this example, the values are scaled with a factor of “10” because the functions are again scaled down for better visualization.

8

Sectional Information

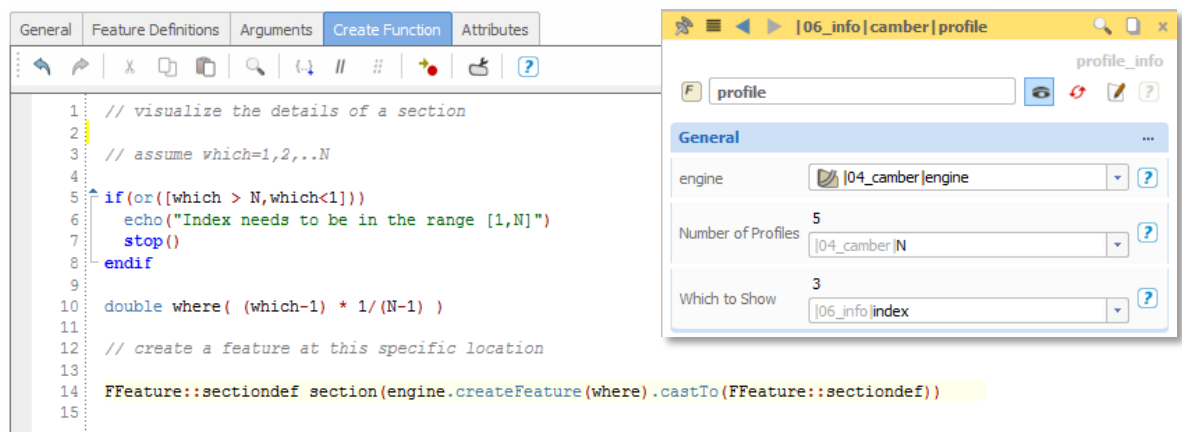
You can also use labels for sending some information from the feature definitions to the 3D window. Check out the example impeller that is provided with CAESES. It contains some labels for showing thickness and angle information to the user. You need to switch to the xy-plane and zoom into the range [0,1] in order to see it.



The screenshot shows information for a section at a specific radius ("0.5").

The labels are created in the feature definitions of the camber and thickness-based profiles. They can then be accessed "from outside" by creating a feature from the used curve engines. Creating a feature based on the engine is a generic possibility to access meta surface data, and it is helpful for checking out sectional blade data at a specific radius.

This is already a bit advanced though... ☺ we just want to let you know...



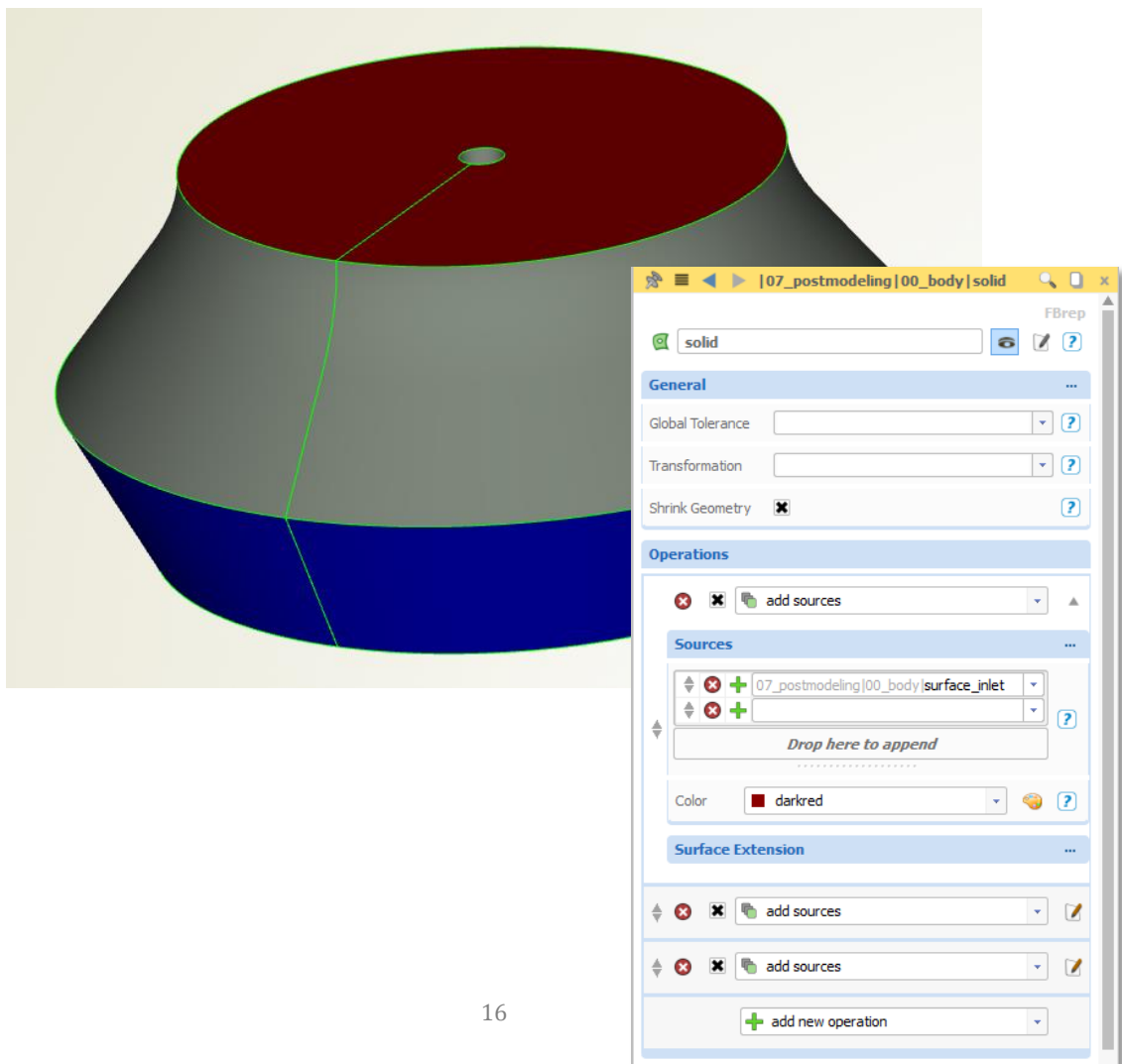
9

Merge Blade with Hub and Shroud Geometry

Usually, for meshing and CFD calculations only one periodic blade of the entire device is required. Furthermore, the blade needs to be intersected with hub and optionally shroud, and sometimes a hub fillet needs to be added. Often a closed and watertight geometry is mandatory. All these tasks can be solved by using the *brep* objects of CAESES.

Solid Body from Hub and Shroud Contour

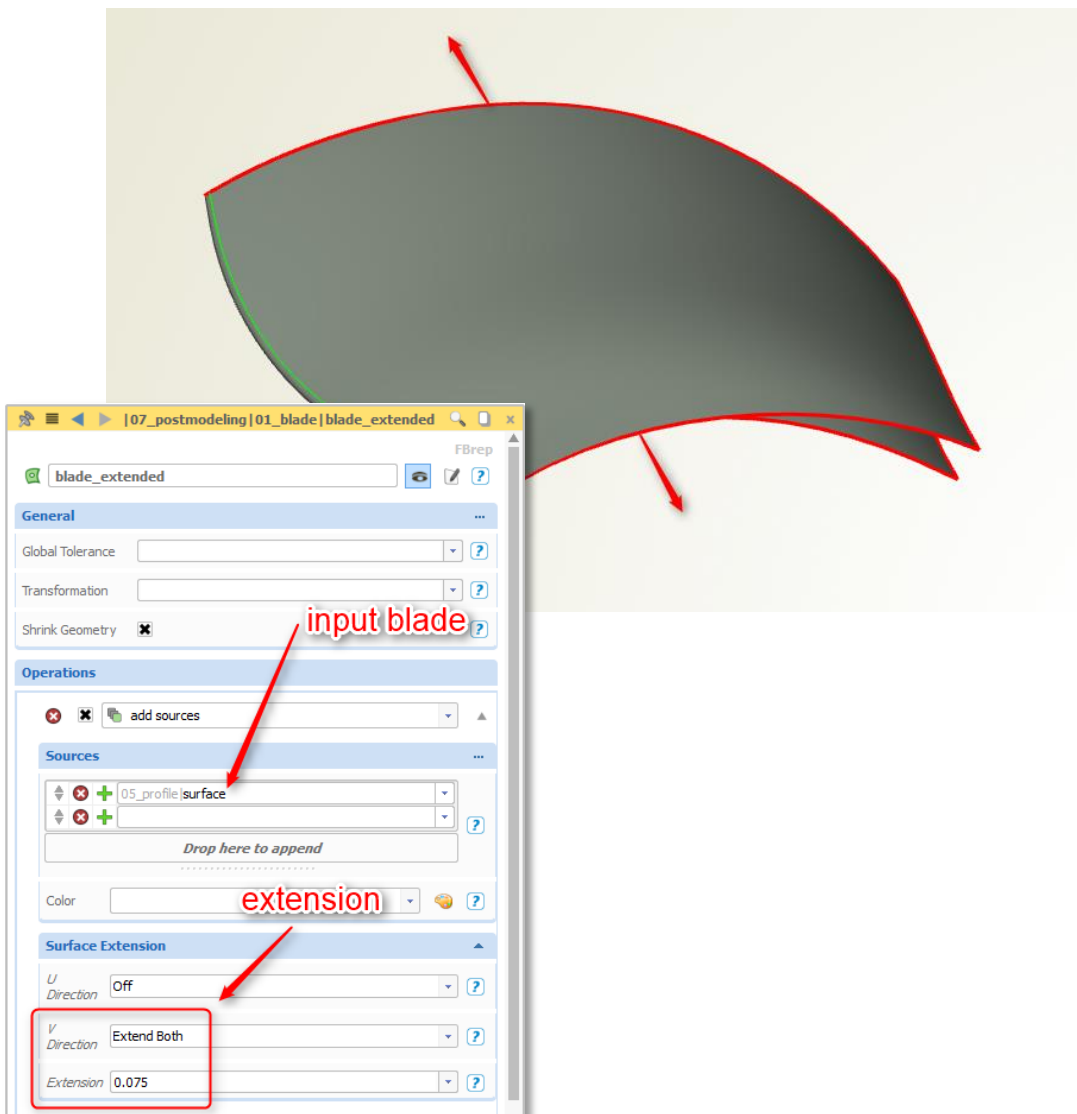
In order to intersect the blade with the hub for merging the two, we can use Boolean Operations. The first body we need for this is made of the hub and shroud contour: Single surfaces are created and put into a brep. In addition, the faces are colored which is helpful to indicate the inlet and outlet areas. This information will be kept through all Boolean Operations.



Extending the Blade

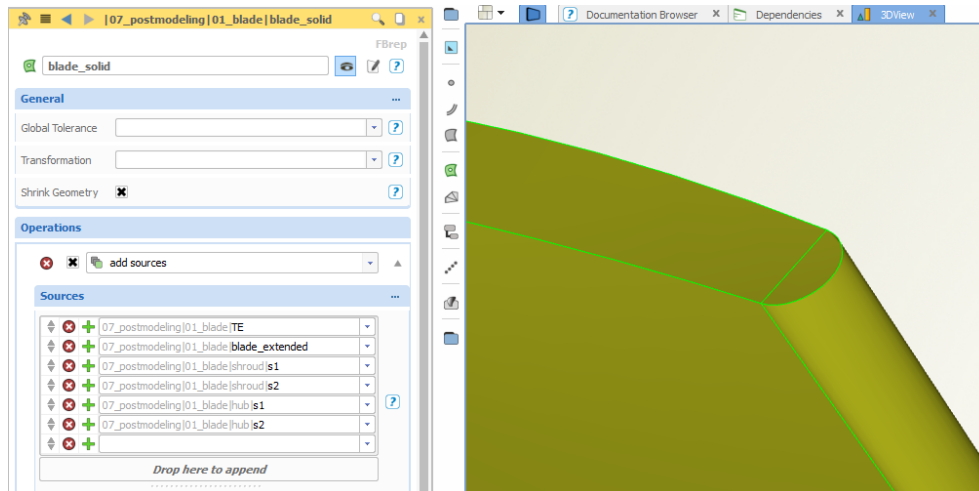
In order to create a solid body from the blade, we extend the surfaces at the hub and shroud region, just to make sure we really intersect with the hub and shroud geometry. For this purpose, we create a new brep and put the blade surface into it. Then, we activate the surface extensions for both sides (v_{min} and v_{max} of the input surface).

Note that we need only a small value to enter, such as “0.075”. This value is based on the parameter domain of the surface which is $u=[0,1]$ and $v=[0,1]$. This means that it is no absolute value but more a relative one.

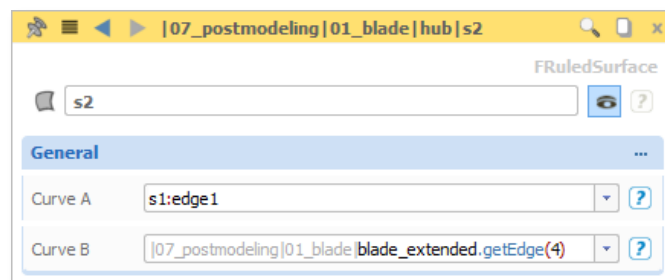


Closing the Blade

For Boolean Operations, we need closed geometries. In order to close the open blade shape, there are many possibilities in CAESES. One quick option is to create lids for the open regions. We can use *ruled surfaces* for this purpose. You find this surface type in the menu *CAD > surfaces > ruled surface*. Simply enter the opposite boundary curves for creation of each face. In the leading edge region, we have another small ruled surface.



Note that we use the edges of the extended surface (brep) for setting up the lids. In the screenshot, the object "blade_solid" is another brep that contains the extended blade surface plus the other ruled surfaces.



Boolean Operation

As an example, we create a final brep, and put the solid hub and shroud body into it. Add a Boolean Operation and enter the solid blade as source. Choose “Difference” in order to remove the blade from the outer body. As you can see below, the colors of the final brep are still kept, and you can export everything using the colored STL format, for example.

