JAVAPROP – USERS GUIDE

Initial Creation: June 2009 – Last Revision: August 2018 - Martin Hepperle

Table of Contents

1. Introduction

JAVAPROP is a simple tool for the design and the analysis of propellers and wind turbines. Within its limits, it is applicable to aeronautical as well as to marine applications. The implemented classical blade element design and analysis methods are based on a coupling of momentum considerations with two-dimensional airfoil characteristics. It is therefore possible to consider different airfoil sections and the impact of their characteristics on rotor performance.
This document describes the pure application of JAVAPROP and is not a complete description of the underlying theory. For a description of the theoretical background, the reader is referred to the references cited on page 35.

As a relatively simple tool the capabilities of JAVAPROP should not be exploited beyond reasonable limits. The underlying blade-element-momentum theory is valid for the design and analysis of many propellers as long as

- the disc loading of the propeller is not too high (thrust coefficient $T_C \lessapprox 2$), which excludes static operation conditions,
- the number of blades is small (say below 15) so that no strong interaction due to overlap and thickness occurs,
- three-dimensional effects are small (no winglets, no highly curved blades), and
- compressible flow effects are small and mostly two-dimensional ($M_{tip} < 1.0$).

2. Symbols and Coefficients

Symbols

In the field of propellers and windmills a variety of definitions are used to describe geometry, operating conditions and performance. JAVAPROP follows mostly the traditional American notation as described in the following tables.
Note that in some publications coefficients of the same name (e.g. T_C) are used, which follow their own definitions – so be careful when comparing results.

Symbol	Description	Unit
D	diameter	m
D_{sp}	spinner or hub diameter	m
$R = \frac{D}{2}$	radius	m
RPM	revolutions per minute	1/min
$n = \frac{RPM}{60}$	revolutions per second	1/s
P	power	W
T	thrust	N
v_∞	axial inflow speed (flight speed, wind speed)	m/s
$S = \pi \cdot R^2 = \pi \cdot \frac{D^2}{4}$	disc area	m^2
ρ	density of medium	kg/m^3
$\Omega = 2 \cdot \pi \cdot n$	angular speed	1/s
$AF = N_{blades} \cdot \frac{100000}{16} \cdot \frac{1}{2} \cdot \int_{\frac{r}{R}=0.15}^{\frac{r}{R}=1} \frac{c}{R} \cdot \left(\frac{r}{R}\right)^3 \cdot d\frac{r}{R}$	Activity Factor of propeller	-

Coefficients

Description	Definition	Conversions

Description	Definition	Conversions
thrust coefficient (propeller)	$C_T = \frac{T}{\rho \cdot n^2 \cdot D^4}$	$= \frac{\pi}{8} \cdot T_C \cdot \left(\frac{v_\infty}{n \cdot D}\right)^2$ $= \frac{\pi}{8} \cdot \eta \cdot P_C \cdot \left(\frac{v_\infty}{n \cdot D}\right)^2$ $= \eta \cdot \frac{C_P}{\left(\frac{v_\infty}{n \cdot D}\right)}$
thrust coefficient (propeller)	$T_C = \frac{T}{\frac{\rho}{2} \cdot {v_\infty}^2 \cdot S}$	$= \frac{T}{\frac{\rho}{2} \cdot {v_\infty}^2 \cdot \pi \cdot R^2}$ $= \frac{8}{\pi} \cdot \frac{C_T}{\left(\frac{v_\infty}{n \cdot D}\right)^2}$ $= \frac{8}{\pi} \cdot \eta \cdot \frac{C_P}{\left(\frac{v_\infty}{n \cdot D}\right)^3}$
power coefficient (propeller)	$C_P = \frac{P}{\rho \cdot n^3 \cdot D^5}$	$= \frac{\pi}{8} \cdot P_C \cdot \left(\frac{v_\infty}{n \cdot D}\right)^3$ $= \frac{\pi}{8} \cdot \frac{1}{\eta} \cdot T_C \cdot \left(\frac{v_\infty}{n \cdot D}\right)^3$ $= \frac{1}{\eta} \cdot \frac{C_T}{\left(\frac{v_\infty}{n \cdot D}\right)}$ $= \left(\left(\frac{v_\infty}{n \cdot D}\right) \cdot \frac{1}{C_S}\right)^5$
power coefficient (propeller, wind turbine)	$P_C = \frac{P}{\frac{\rho}{2} \cdot {v_\infty}^3 \cdot S}$ $= C_{P\ \text{wind turbine}}$	$= \frac{8 \cdot P}{\rho \cdot v_\infty^3 \cdot \pi \cdot D^2}$ $= \frac{P}{\frac{\rho}{2} \cdot {v_\infty}^3 \cdot \pi \cdot R^2}$ $= \frac{8}{\pi} \cdot \frac{C_P}{\left(\frac{v_\infty}{n \cdot D}\right)^3}$ $= \frac{8}{\pi} \cdot \frac{1}{\eta} \cdot \frac{C_T}{\left(\frac{v_\infty}{n \cdot D}\right)^2}$
efficiency (propeller)	$\eta = \frac{T \cdot v}{P}$	$= \frac{T_C}{P_C}$ $= \frac{C_T}{C_P} \cdot \left(\frac{v_\infty}{n \cdot D}\right)$
advance ratio (propeller)	$J = \frac{v_\infty}{n \cdot D}$	$= \pi \cdot \lambda$
advance ratio (propeller)	$\lambda = \frac{v_\infty}{\Omega \cdot R}$	$= \frac{1}{\pi} \cdot \frac{v_\infty}{n \cdot D}$
tip speed ratio (wind mill)	$X = \frac{\Omega \cdot R}{v_\infty}$	$= \frac{1}{\lambda}$

3. Propellers

How to design a propeller

JAVAPROP contains a powerful direct inverse design module. Inverse design means that you specify only a few basic parameters and JAVAFOIL produces a geometry which automatically has the maximum efficiency for the selected design parameters. The beautiful thing is that JAVAPROP creates an optimum propeller with just 5 design parameters plus a selection of airfoil operating points along the radius. You can later modify this design to adapt to additional off-design conditions.

The following cards are relevant for the design of a propeller:

- Design,
- Airfoils,
- Options.

The Design Card

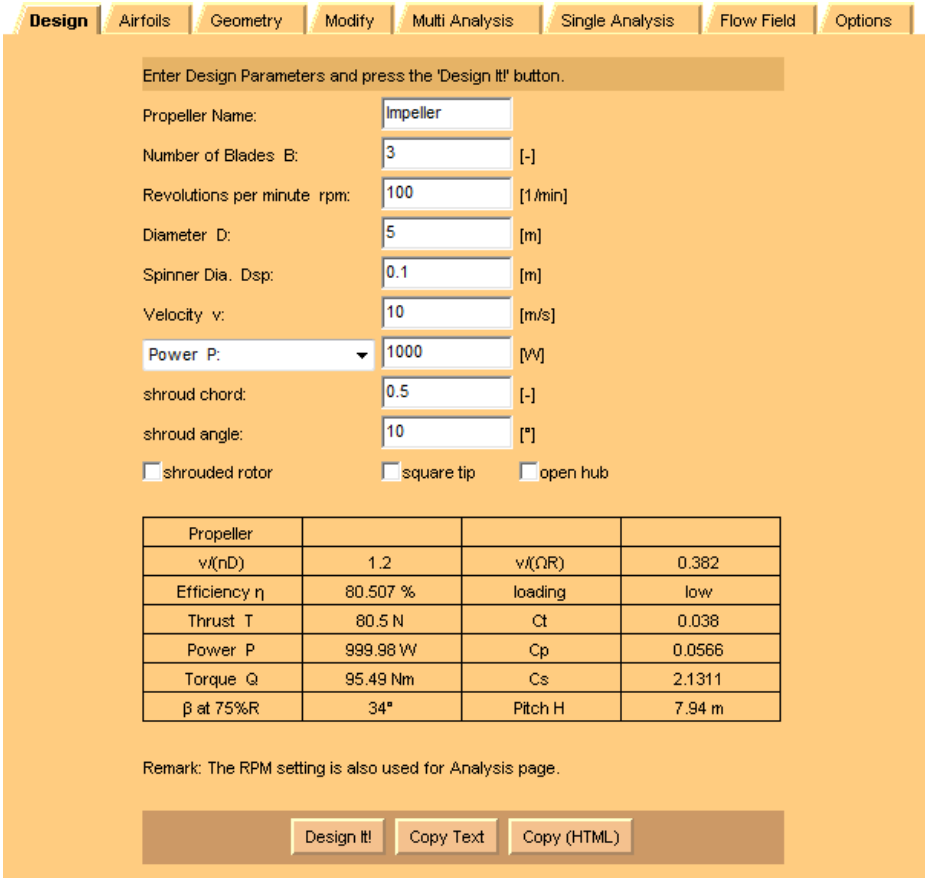


Figure 1: Design card after a design has been performed.

The design card holds most of the parameters which are required for a design.
It is possible to perform a design for either

- power (the propeller will consume the specified power),
- thrust (the propeller will produce the specified thrust), or
- torque (the propeller will consume the specified torque).

Shrouded propeller option

The thrust distribution along a propeller with free tips drops to zero at the tips. If a shroud is added to the propeller, this "tip loss" is suppressed.
Additionally, the profile section of the shroud may be inclined to form a convergent or divergent nozzle. Thus the shroud accelerates or decelerates the flow through the propeller. JAVAPROP models this effect by a rather crude approximation of the real flow field. The shroud is replaced by a vortex ring whose circulation strength depends on the angle of the shroud section. The purpose of this model is to support the blade design by providing an approximation of the local inflow velocity for obtaining a more reasonable blade angle distribution for the shrouded case.
The shroud is characterized by two parameters: its relative cord length c/R and its profile inclination angle α. To simplify the model, the profile section is assumed to be symmetrical – and thin. If you use a cambered section (to adapt to the local flow field and to avoid flow separation) you can adapt the inclination angle accordingly.

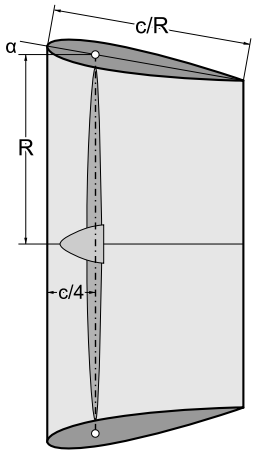


Figure 2: Shroud with relative chord length c/R and inclination angle α.

Note that the thrust calculated by JAVAPROP is only the propeller thrust; under static conditions a good shroud design can double this figure due to the suction forces on the shroud. The thrust of the shroud drops with forward speed and will turn into drag at higher speeds.
The drag of the shroud is not included in the thrust and has to be subtracted from the propeller net thrust to obtain the gross thrust of the complete arrangement.
A detailed modeling of such configurations is beyond the targeted conceptual design capabilities of JAVAPROP.

Square tip option

The optimum design procedure creates blades with rounded tips. As this is not always practical the option "square tip" produces a tip with finite chord length by simple extrapolation of the last section.

Open hub option

JAVAPROP uses the so called "tip loss factor" derived by Prandtl to model the effect of the free blade tip and the finite number of blades. A similar correction can be applied if the hub is "open", i.e., the blade is not attached tightly to a cylindrical hub. Such a situation represents a propeller with an open hole instead of a closed hub. Depending on the given diameter of the hub, resp. spinner, such an open hub leads to a local loss of lift. It is also affecting the propeller design method. Such a correction can also be used when a blade is abruptly

changing its chord length towards the hub so that a relatively wide gap between hub and blade root exists.

Loading

The optimum design procedure works well for lightly or medium loaded propellers. While the term loading is not clearly defined, JAVAPROP sorts the design into three categories depending on the thrust coefficient T_C :

- $T_C > 1$: highly loaded,
- $T_C > 0.25$: medium load,
- $T_C \leq 0.25$: lightly loaded.

If the loading is very high, the theory will become more and more inaccurate.

The Airfoils Card

In addition to the basic parameters on the Design card, airfoils have to be selected and their operating point must be specified on the Airfoils card. JAVAPROP comes with a few built-in airfoil sections. For each section the tables of lift and drag coefficients versus angle of attack are shown as graphs on the Airfoils card.

For the design it is necessary to assign airfoils to four radial stations – JAVAPROP interpolates linearly between these design sections. You define lift- and drag coefficient by selecting a design angle of attack for each station. Note that the absolute maximum efficiency is obtained when the airfoil sections are operated at their individual maximum L/D. For real world propellers, which must also be useable at low speed off-design conditions, it is usually better to select angles of attack, which are lower than the point of maximum L/D. This is especially true for the inner sections towards the root, which see a large variation of angle of attack with forward speed.

Suppress airfoil drag option

Sometimes it is of interest to see the effect of the airfoil drag on the results. Instead of preparing special polars with drag coefficients of zero this option allows to override the drag coefficient contained in the polars.

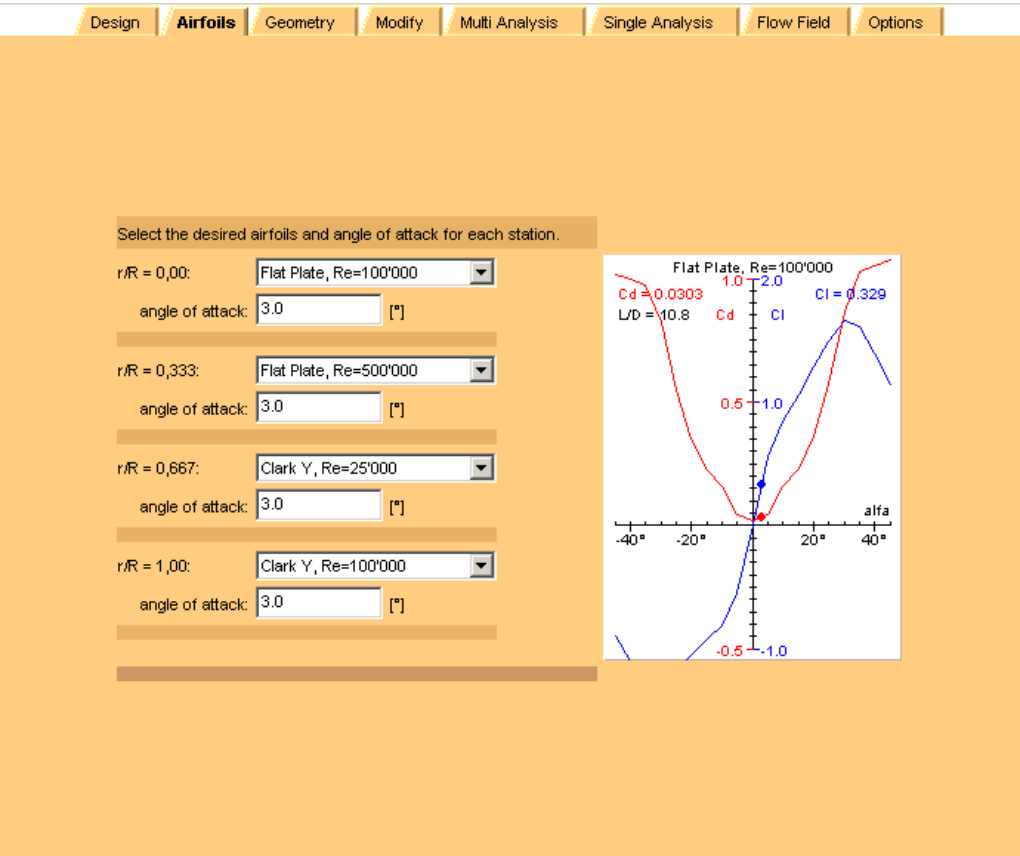


Figure 3: Airfoils card with four design sections and a graph of the lift and drag coefficients versus angle of attack.

How to use your own airfoils

JAVAPROP comes with a set of canned airfoils and associated polars. These are sufficient for first steps, for understanding the main design parameters and also for many typical applications.
Nevertheless, some users may want to add their own airfoil data. This is possible in case of a local installation by storing additional files in the installation directory of JAVAPROP[1].

Airfoil Polars

The polar data files must be named "af_#.polar.EXT", where the # character represents a serial number and the extension EXT is either "afl" or "xml". JAVAPROP will search for files beginning with "af_1.polar", trying the extension ".afl" first. If no matching file is found JAVAPROP tries the alternative file name ending in "xml". If a matching file was found it is read and the airfoil index is incremented until no corresponding file is found (because no more files exist or the file names have a gap in their numbering). Note that file names must be composed from lowercase characters if your system is case sensitive like Linux systems.

XML Format

The polar data files in "xml" format are in my standard XML format. These can be created with JAVAFOIL's Polar card by saving the polar with the extension ".polar.xml". The file must contain at least the three variables "alpha", "Cl" and "Cd" declared in the <variables> block. JAVAPROP will read only the first <configuration> found in the file, therefore it is recommended to perform the analysis for just one Reynolds number close to the Reynolds

[1] This option is not available when JAVAPROP is run via WEBSTART or as an APPLET in a browser, because these applications are not allowed to access your computers file system for security reasons.

numbers occurring during the operation of the propeller. You can have as many data points in the regime from -180° to +180°, but it is usually sufficient to provide polars with a range from -45° to +45° in steps of 2.5 degrees. JAVAPROP adds data points at +/- 90° automatically if not supplied. In order to achieve realistic results it makes sense to select a NACA standard roughness and no perfect surface finish. Note that JAVAPROP only uses a single polar, so you should create a polar for one single typical Reynolds number for the desired radial station.

AFL Format

Polar data files in "afl" format are plain text files with exactly 5 header lines, followed by data points describing the airfoil polars. The first line contains the airfoil name and will appear in the dropdown list boxes of the Airfoils card.

```
Tabulated Airfoil 1
This is an airfoil polar file for JavaProp. It can have up to 1000 data triples.
This format will be changed to my airfoil-polar-XML form in a future release.
-------------------------------------------------------------
alpha         cl         cd         cm
-180.00000  0.00000  0.49786  -0.13940
-175.00000  0.19970  0.27181  -0.07611
… some polar points omitted…
178.00000  -0.08022  0.00005  -0.00001
179.00000  -0.04013  0.00001  -0.00000
180.00000  -0.00000  0.00000  -0.00000
```

Figure 4: Example of a tabulated airfoil polar data set. Some data points have been omitted for clarity.

It is recommended to use the XML format, the AFL format exists only for backwards compatibility.

Airfoil Geometry

In addition to the polar data you can also supply a set of coordinates representing the geometry of your airfoil. The airfoil geometry is optional and only used for exporting the propeller blade to CAD systems in IGES or DXF format. You can also see the airfoil shape in the cross sections on the geometry card. If you do not supply a geometry JAVAPROP uses the default section shape which is a plain flat plate.

For this feature you supply another file using a similar naming scheme. Coordinate files are named "af_#.geometry.xml". The serial number # must match the polar file, but as geometry is optional it is not necessary to supply coordinates for each polar file. If a matching airfoil geometry was found and read, the string "(*)" is appended to the airfoil name.
The coordinate files use my standard XML format which can be written by JAVAFOIL's Geometry card.

Note that the number of coordinate points of each section is reduced to 61 points by a spline interpolation in order to minimize memory consumption. It is recommended to open the trailing edge to a finite thickness of 0.5% of the chord length to avoid surface generation problems in CAD software as well as manufacturing issues. This opening can be performed using the Modify card.

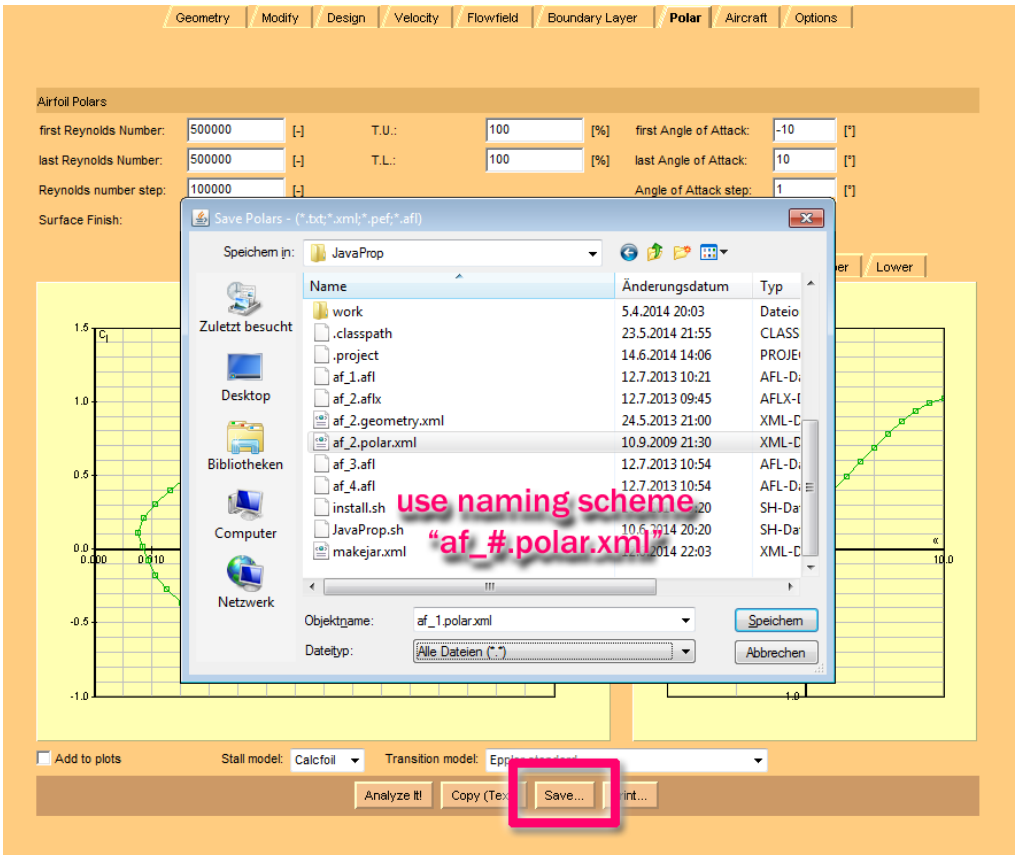


Figure 5: Exporting an airfoil polar for a single Reynolds number from JAVAFOIL.

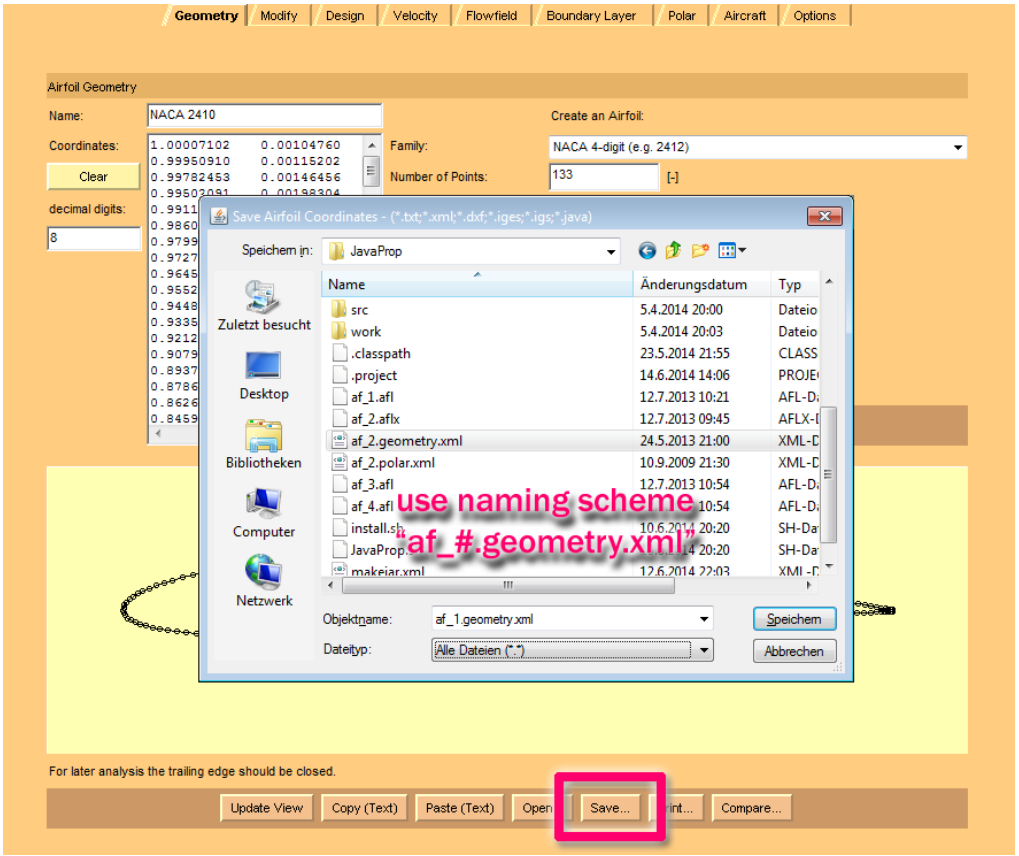


Figure 6: Exporting the airfoil geometry from JAVAFOIL.

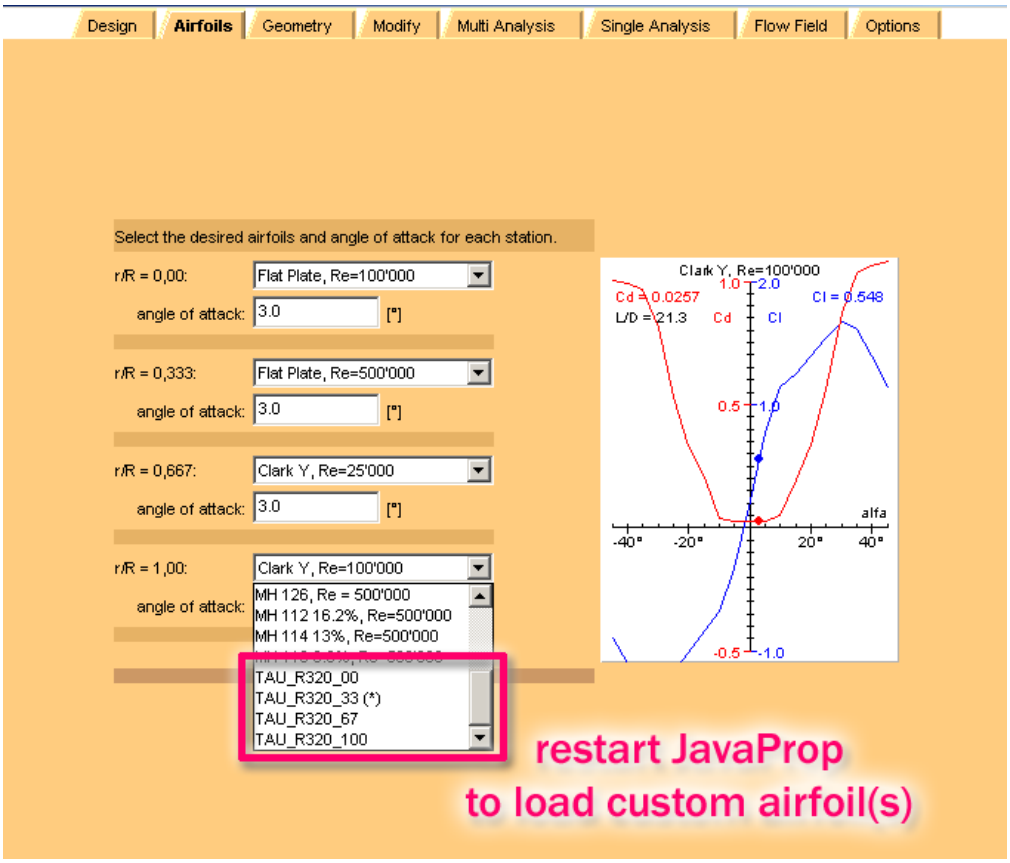


Figure 7: Using the airfoil polar in JAVAPROP.

Note that the polars must also include any stall delay effects due to 3D effects on the rotating blades. In general the three-dimensional flow past the rotating rotor blade airfoil changes the airfoil characteristics, especially in the stall region. The inertial forces acting on the boundary layer tend to delay stall so that it occurs at higher angles of attack and higher lift coefficients. JAVAPROP does not modify the given polars for this effect because there is no general method to do so. Many stall delay models exist and each fits only a limited class of cases.
In any case, the pitching moment coefficients contained in the airfoil data are not used by JAVAPROP.

Design parameters on the Options card

Finally the density of the fluid from the Options card is used for the design. A propeller designed for a low density medium (e.g. high altitude) must have blades of a wider chord length than a design for a high density medium.
This difference is also visible when a hydroprop is designed – the density of water is roughly 1000 higher than the density of air. Therefore a propeller for underwater operation would have blades of only 1/1000 the chord length of an aircraft propeller – if the diameter and the design lift coefficients were the same. Note that the cavitation phenomenon limits the airfoils design lift coefficient to rather lower values than the high values useable for the design of aircraft propellers (close to maximum L/D).

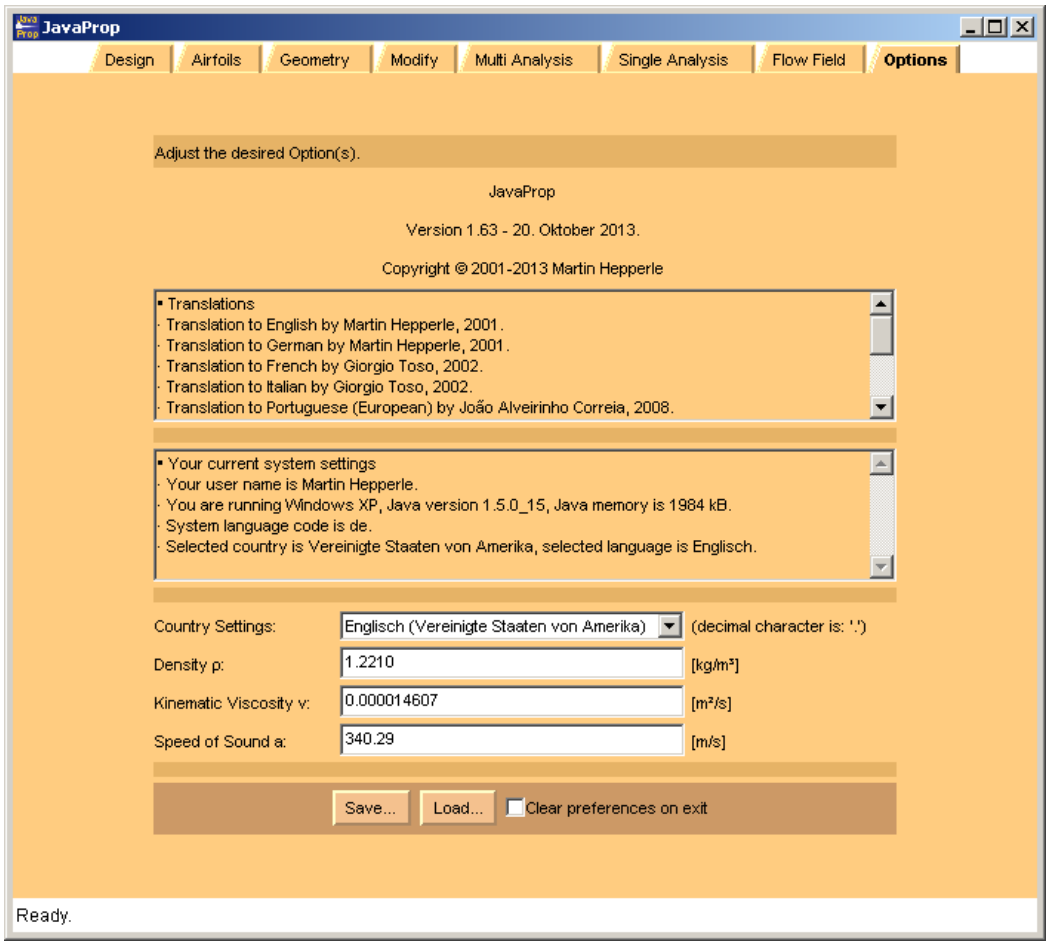


Figure 8: The Options card holds the density of the medium.

The Geometry card

This card (Figure 9) presents the geometry of the current propeller in form of a table and a three view sketch. It also presents the distribution of the pitch to diameter ratio H/D over the radius of the propeller.

The data table presents the following columns:

- "r/R" – the radius station, normalized by propeller radius,
- "c/R" – the corresponding chord length at each station, normalized by propeller radius,
- "β" – the blade angle at the station in degrees,
- "H/D" – the local pitch to diameter ratio,
- "r" – the radius of the station in millimeters,
- "c" – the local chord length in millimeters,
- "H" – the local pitch height in millimeters,
- "t" – the local blade thickness in millimeters,
- "Airfoil" – the airfoil at each station as selected on the "Airfoils" card.

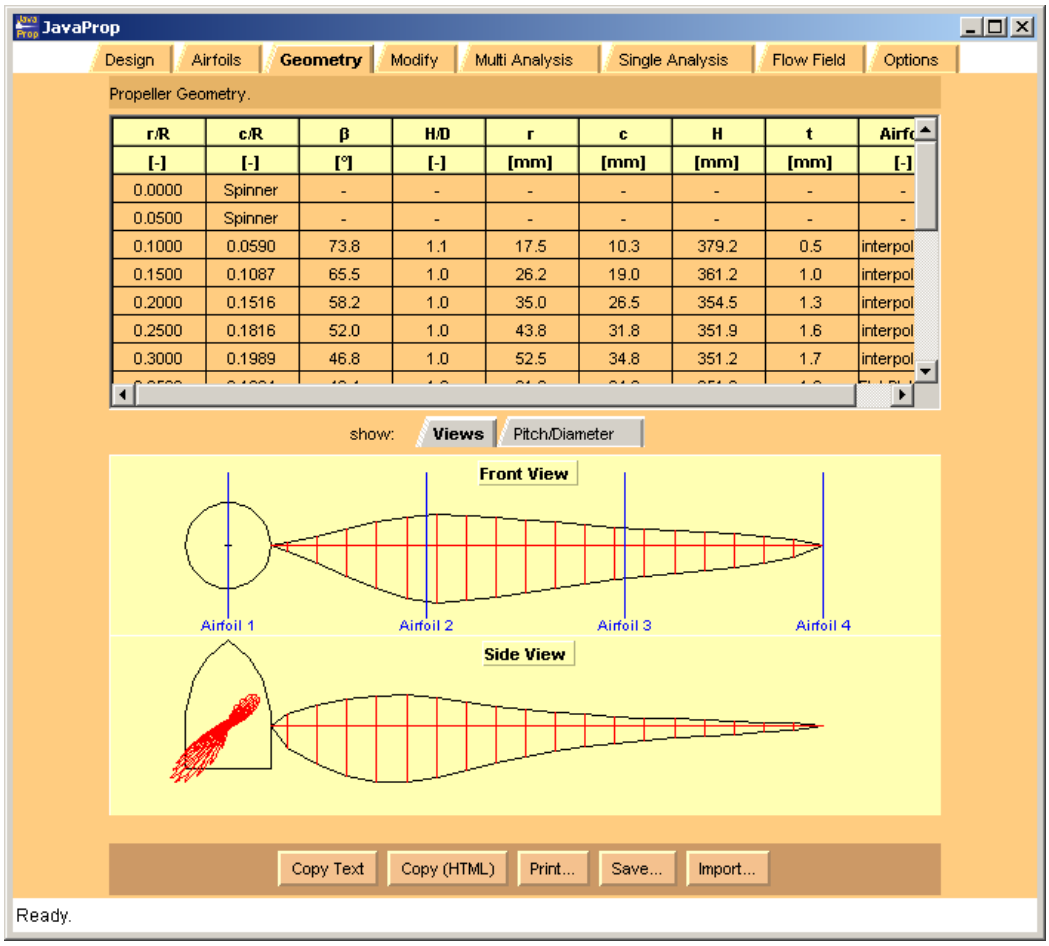


Figure 9: Geometry card with current propeller.

The geometry card also allows you to copy or export the geometry in form of text files or as a 3D geometry, either in AutoCAD DXF or IGES format. This option is intended mainly for illustration purposes. Note also that JAVAPROP selects the export format by file name extension so that you have to use one of the proposed extensions.
You can e.g. import the IGES file into CATIA and create a smooth surface spanning the sections as shown in Figure 10. Before creating the blade loft, some modifications in the root and tip area might be appropriate.
The IGES export function creates accurate, interpolated airfoil section curves as well as a surface. The surface uses the airfoil coordinate points as control points so that the surface does not pass through these points. Therefore the surface is not 100% accurate, but may be smoother. It is up to you to decide whether you want to create an accurate loft through the sections or use the smoothed surface.

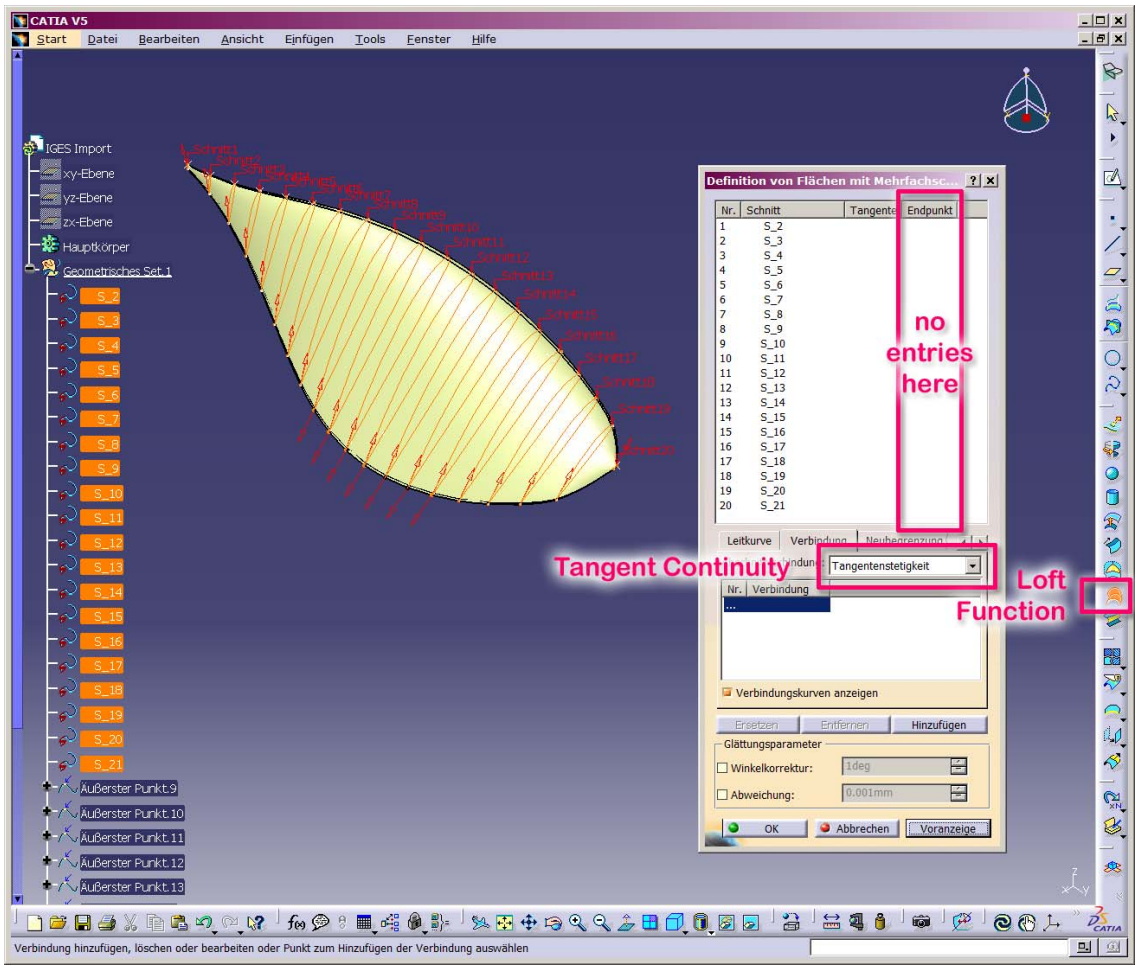


Figure 10: Creating a blade surface in CATIA V5 by lofting sections imported from a JavaProp IGES file.

Additionally, this card also offers the option to import a given propeller geometry from a table of values (Figure 11). The table must contain the planform as well as the blade angle in at least 3 columns in the sequence “r/R”, “c/R”, and “β”. Note that the blade angle must be specified in degrees. An example data set can be produced by copying the current propeller in text format to the clipboard and then opening the “Geometry Import” form. On opening, the import form automatically pastes the content of the clipboard into its text field. This also allows for manual modifications of the current propeller. You can copy and paste your prepared data via your systems clipboard. During the import process, JAVAPROP tries to be smart and skips non-numeric data, but it is a good idea to stick to the proposed format.

When the data has been parsed successfully, JAVAPROP performs an analysis of the geometry using the parameters flight speed, diameter and rotational speed as currently set on the Design card. The table on the design card is then updated with the resulting performance data. Note that if you would perform a new design on the Design card, the imported propeller would be overwritten by the new design.

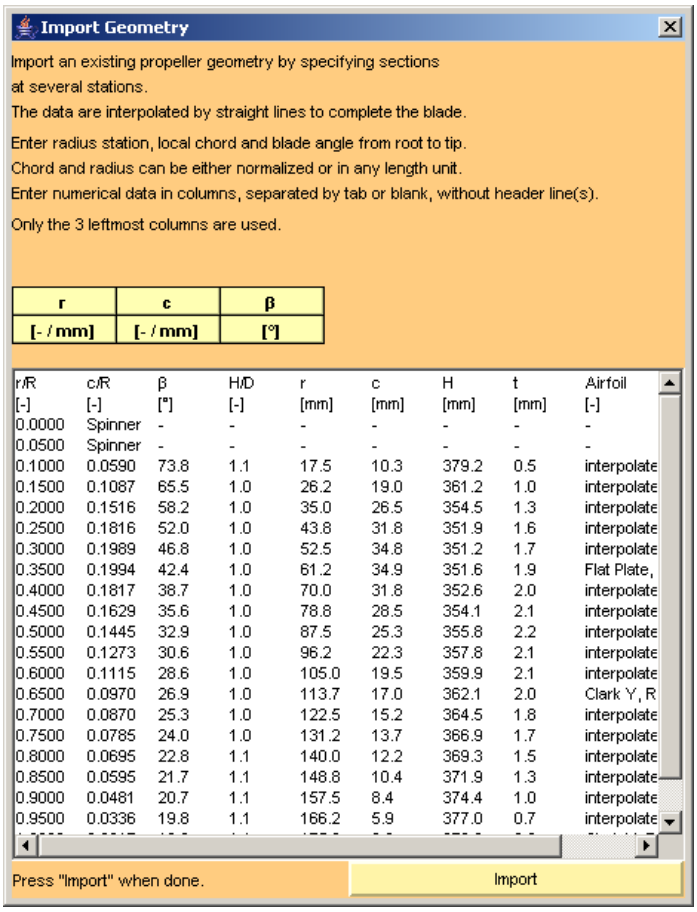


Figure 11: Geometry import form with example data.

The Modify card

This card offers tools to modify the current propeller blade or the inflow conditions as used for design and analysis. All modifications are performed in sequence from top to bottom. Usually you want to use only one type of modification and you should take care to reset it when you apply another, but different type of modification.

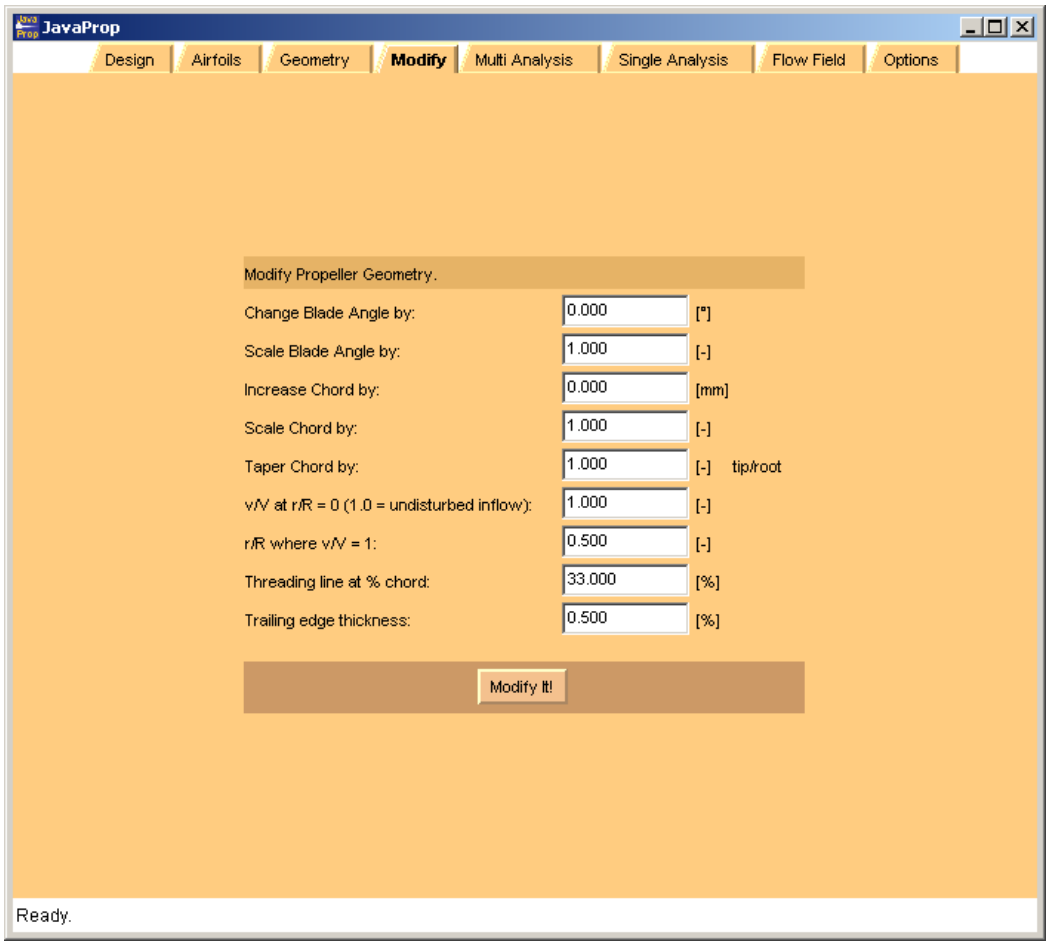


Figure 12: Modifications of the blade geometry can be performed using the Modify card.

The following modifications are available:
Change blade angle by a given angle – this option changes the blade angle like a variable pitch propeller by rotation of the complete blade. You can enter positive or negative values and check the result on the Geometry card.
Scale blade angle by a factor – this modification multiplies the blade angle at each station by the given factor, thus changing the internal twist of the blade.
Increase chord by a given length – adds or removes the given length from the chord length at each station. You can also specify negative values, but the chord length must not fall below zero.
Scale chord by a factor – applies the given scale factor to the chord length at each station.
Taper chord by a taper ratio – similar to the constant scaling factor option, but applies a variable scaling factor. For a given ratio "tip/root" of 0.5, the blade tip will be narrowed to 50% of its current chord while the root will maintain its current chord length.
v/v_∞ *at* $r/R = 0$ and r/R *where* $v/v_\infty = 1$ – can be used to define a linear variation of the inflow velocity profile $v/v_\infty = f(r/R)$. Such an inflow profile can be used to represent the influence of an axisymmetric body in front of or behind the propeller. The given inflow velocity profile is used for design and analysis. The two values define a linear variation of the axial inflow velocity v. The case without inflow has a constant value of v_∞ and is obtained if "v/v_∞ at $r/R = 0$" is set to 1.0 and "r/R where $v/v_\infty = 1$" can be arbitrary.

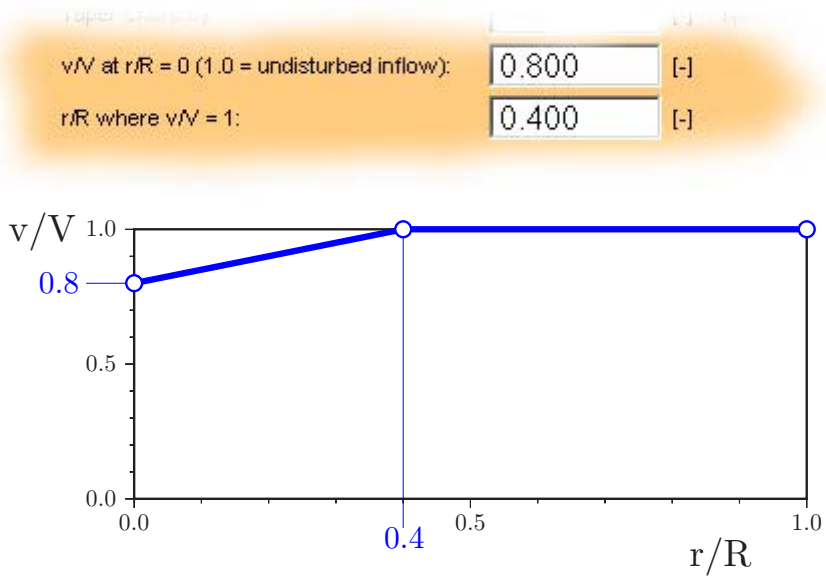


Figure 13: Modified inflow profile with a reduced axial flow speed towards the axis of rotation.

Threading line at % chord – the airfoil sections are threaded onto a straight line, which is normally located close to the center of gravity of each section to minimize structural loads. This value is typically between 30-40% of the chord and can be specified here.
Note: if the local helical Mach number is higher than 0.85, local sweep is automatically added to the blade so that the Mach number in the section normal to leading edge does not exceed 0.85. This is currently hardwired into the code.
Trailing edge thickness – the trailing edge of the airfoil sections can be thickened to allow for manufacturing constraints. Typical values are in the order of 0.5-1%. The resulting airfoil shapes are used on the Geometry card for display and export functions.

How to analyze a propeller

JAVAPROP can analyze propellers at arbitrary operating points. The propellers can be created by the design module of JAVAPROP or by importing a given geometry. There are two cards available for analysis:

- Multi-Analysis,
- Single-Analysis.

Both cards differ in their analysis range and in the level of detail of their output but apply the same analysis method.

The Multi-Analysis card

The Multi-Analysis card is used to analyze the propeller over its complete useable operating range from static operation up to the beginning of the windmilling regime at high speeds. The output of this card consists of the global propeller data like thrust, power or efficiency versus advance speed.
The coefficients shown on this card are generally applicable performance parameters. On the other hand the absolute values like thrust or power are calculated using these coefficients plus additional data taken from the

- Design card (diameter, and one of n, P, T or Q),
- Options card (density).

You can change any of these values and perform an additional analysis to study their effect. As long as you do not modify the geometry, the coefficients will always be the same, but the absolute values will change.

The four different cases for the calculation of the absolute values represent:

- n=given – constant speed propeller, P, T, Q vary with air speed,
- P=given – n is adjusted so that the propeller consumes the given power,
- T=given – n is adjusted so that the propeller produces the given thrust,
- Q=given – n is adjusted so that the propeller consumes the given torque.

In order to analyze for example a constant speed propeller at different speeds of rotation n, you would just change the value of n on the design card and then perform and additional Multi-Analysis. Careful: do not perform a new design on the Design card – this would create a new blade shape.

Note that none of these cases exactly represents a propeller operating on a given engine because for simplicity no engine performance curve model is used in JAVAPROP. While the "constant speed" (n=given) is a realistic operating procedure, the constant P, T or Q methods are somewhat artificial, but can be used to get an overview of the basic characteristics. Note that all these dimensional results are obtained from the single set of thrust and power coefficients versus advance ratio.

The point corresponding to the data on the Design card is marked in the graphical output by a black circle.
The scaling of the graphs is determined by the first set of analysis data. If you want to combine the results of several analyses into the graphs, you must therefore run the case with the maximum data extents first.

The output of the Multi-Analysis card contains a table with the following columns:

Symbol	Description
$v_\infty / (n \cdot D)$	advance ratio
$v_\infty / (\Omega \cdot R)$	advance ratio
C_T	thrust coefficient
C_P	power coefficient
C_S	speed-power coefficient
P_C	power coefficient
η	efficiency
η^*	maximum possible efficiency
stalled	relative disc area swept by stalled airfoils
v	flight speed resp. wind speed
n	speed of rotation
P	power
T	thrust
Q	torque

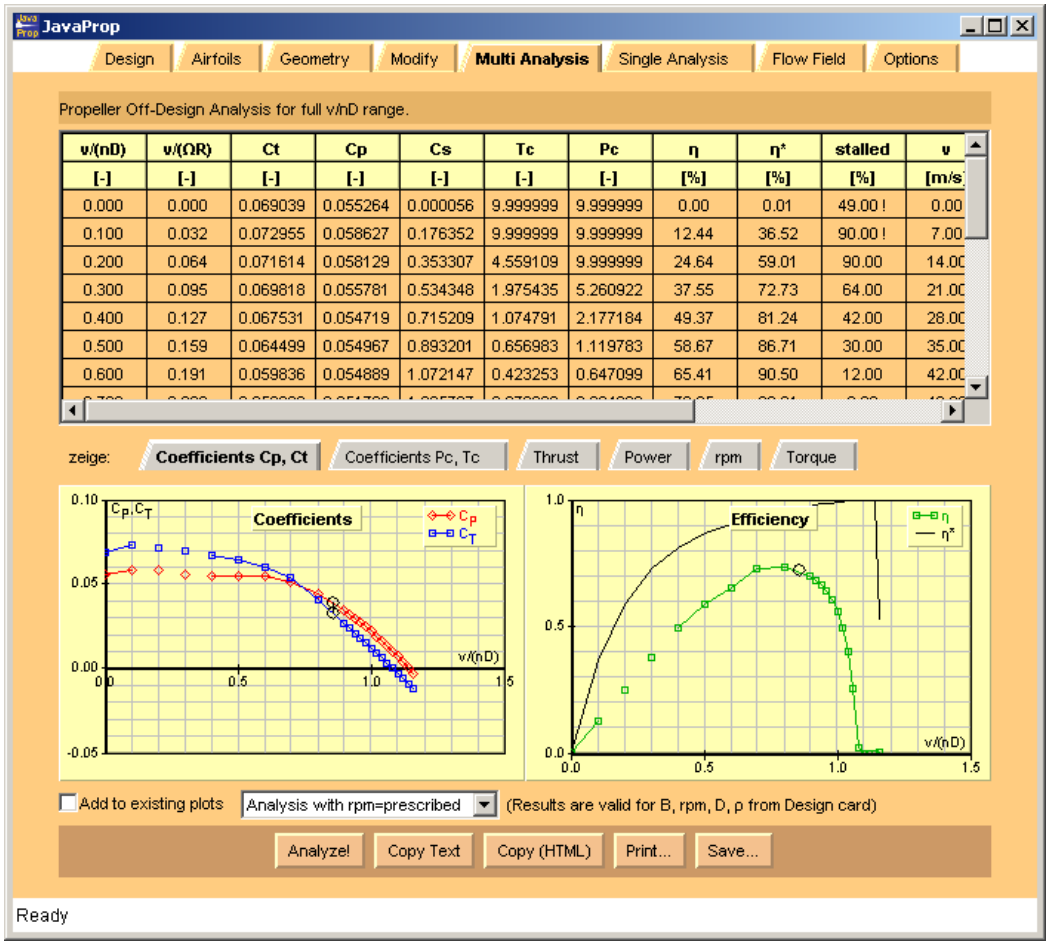


Figure 14: The Multi-Analysis card produces global propeller coefficients over a range of operating conditions.

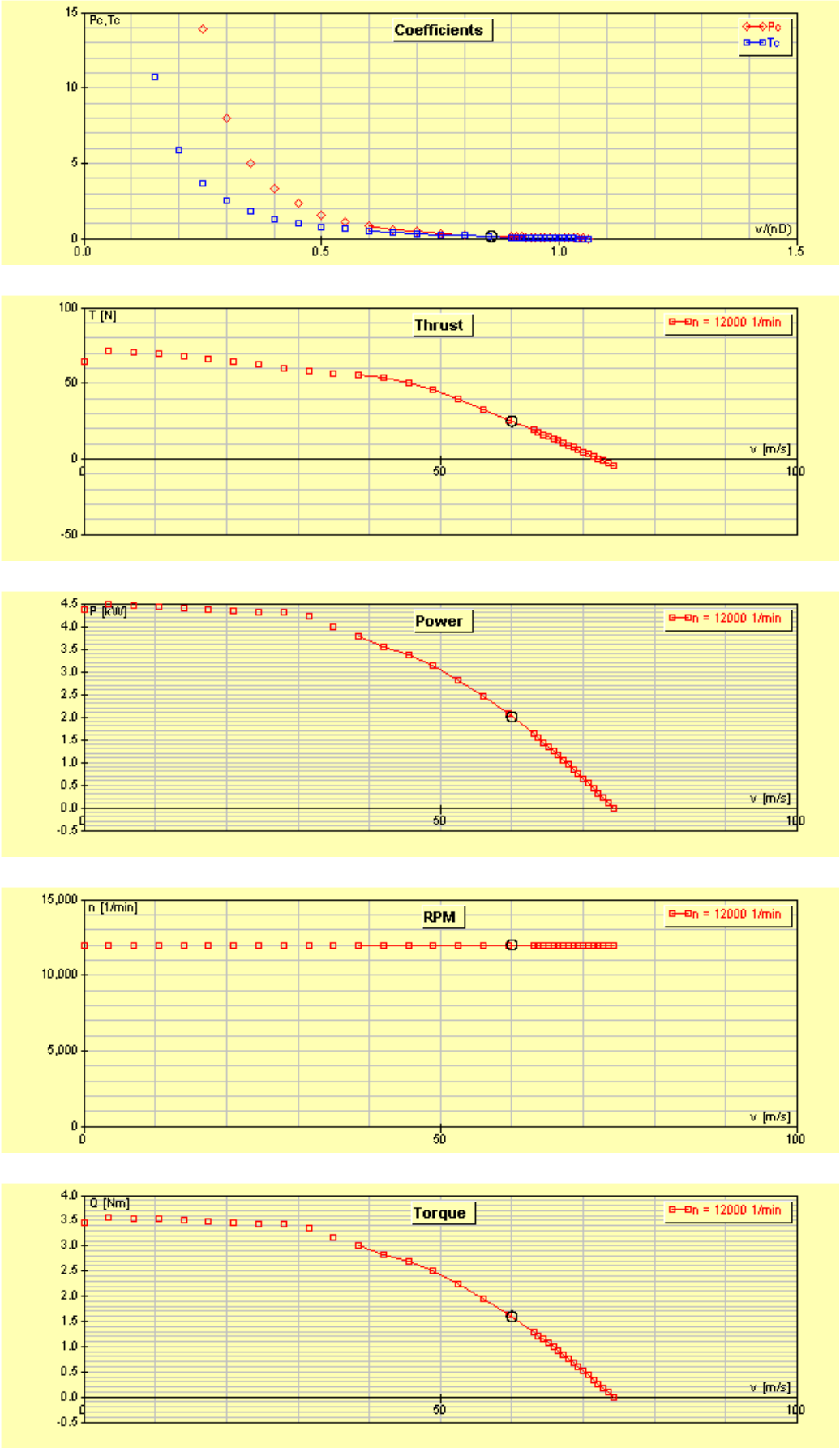


Figure 15: The individual graphs on the Multi-Analysis card present thrust, power, RPM and torque versus flight speed for the selected operating mode (in this case RPM=constant).

The Single-Analysis card

The Single-Analysis card is used to analyze the propeller at a single, arbitrary operating point. This point is specified by the flight speed v_∞, rotational speed n and diameter D on the Design card, which define an advance ratio

The output of the Single Analysis card is more detailed than that of the Multi-Analysis card. It consists of distribution of local aerodynamic data along the radius of the blade and includes coefficients related to structural loads (shear force and bending moment) as well. All results are available in a table and some specific data is presented in form of individual graphs.

The first page of graphs shows parameters relevant for airfoil aerodynamics, the second page displays the distributions of thrust and power as well as the local efficiency, the third graph page presents the structural loads, whereas the fourth page shows properties of the wake. Note that the drag coefficient may be zero if you selected the corresponding option on the airfoils card. In this case the lift over drag ratio L/D will be infinite.

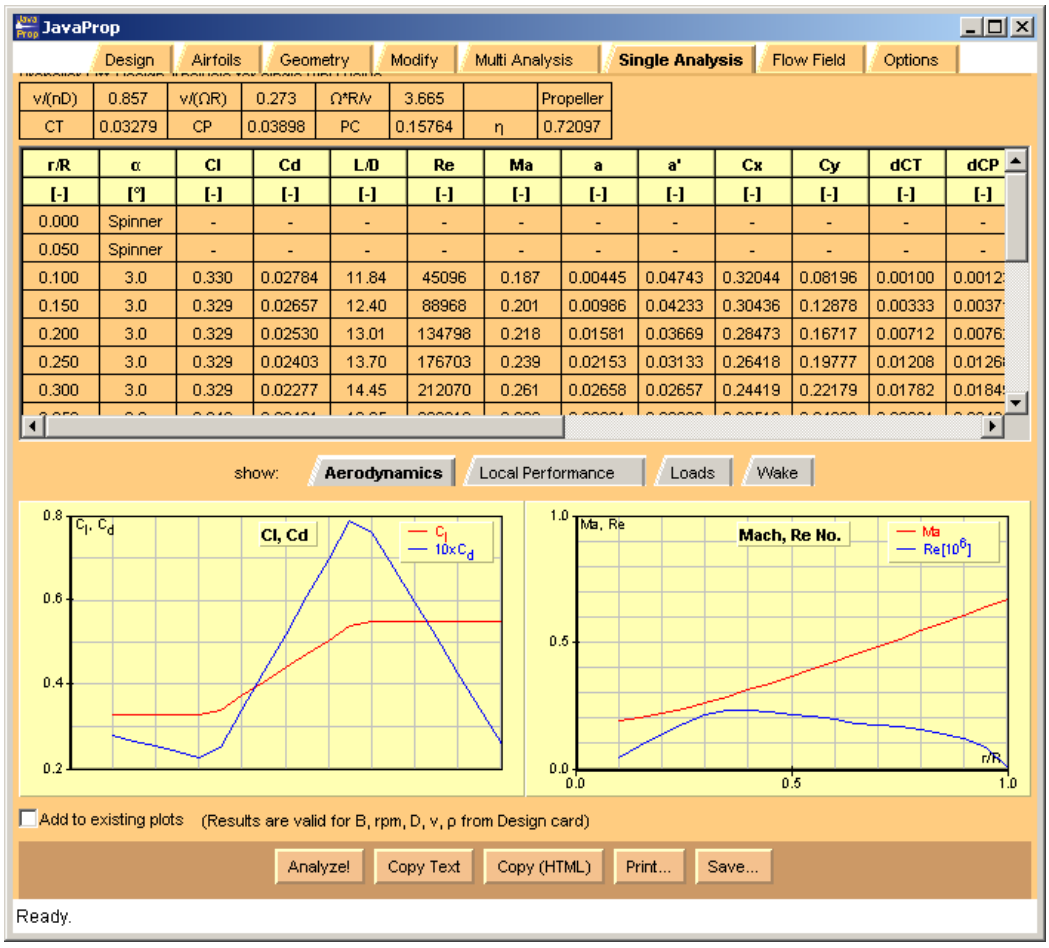


Figure 16: The Single-Analysis card shows detailed results for a single operating condition. The first set of graphs contains parameters related to airfoil section aerodynamics.

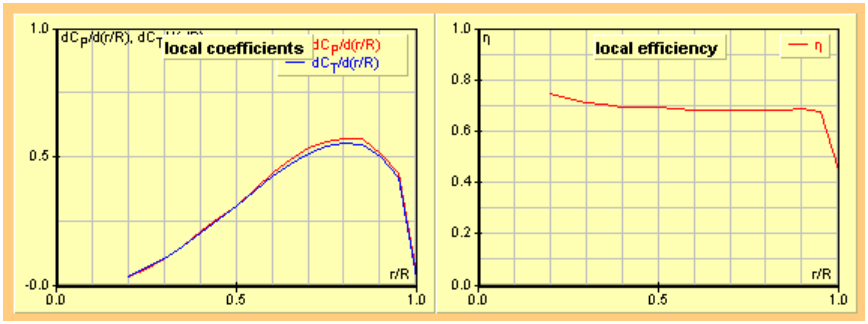


Figure 17: The second set of graphs contains local power and thrust coefficients as well as the local efficiency.

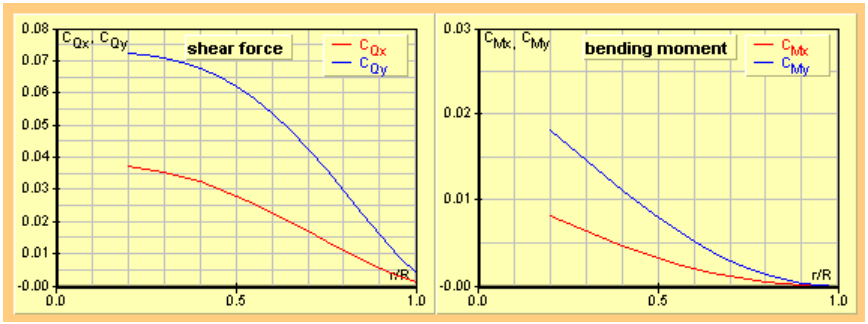


Figure 18: The third set of graphs shows shear force and bending moment.

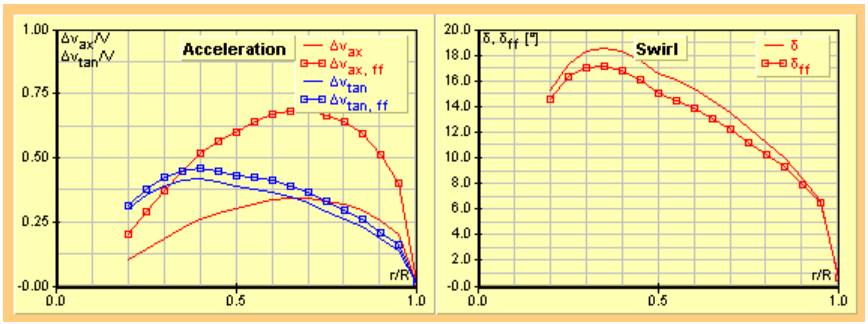


Figure 19: The fourth set of graphs shows the velocities in the propeller plane due to the propeller slipstream and the local swirl angle in the propeller plane as well as far behind. The curves with square symbols represent data in the ideal far wake, including contraction.

Symbol	Description
r/R	relative radius
α	angle of attack in degrees
C_ℓ	lift coefficient
C_d	drag coefficient
L/D	lift to drag ratio
Re	Reynolds number
Ma	Mach number
a	axial induction factor in the propeller plane (the mean axial velocity through the propeller plane is $v_\infty \cdot (1 + a \cdot F)$)
a'	tangential induction factor in the propeller plane (the mean tangential velocity in the propeller plane is $\Omega \cdot r \cdot (1 - a' \cdot F)$)
$\Delta v_{ax} / v_\infty$	axial flow speed increment immediately after the propeller $\Delta v_{ax} = v_\infty \cdot a \cdot F$ (far behind the propeller the incremental velocity has twice this value)
$\Delta v_{tan} / v_\infty$	tangential flow speed increment immediately after the propeller $\Delta v_{tan} = \Omega \cdot r \cdot 2 \cdot a' \cdot F$ (far behind the propeller the tangential velocity has a slightly increased value)
C_x	local tangential (in-plane) force coefficient at a blade station $C_x = \frac{F_x}{\frac{\rho_\infty}{2} \cdot v_{eff}^2 \cdot c \cdot dr}$
C_y	local thrust force coefficient at a blade station $C_y = \frac{F_y}{\frac{\rho_\infty}{2} \cdot v_{eff}^2 \cdot c \cdot dr}$
dC_T	local thrust coefficient of all blades for a ring element, $C_T = d(r/R) \cdot \sum dC_T$
dC_P	local power coefficient of all blades for a ring element, $C_P = d(r/R) \cdot \sum dC_P$
η	local efficiency at a blade station $\frac{v_\infty}{n \cdot D} \cdot \frac{dC_T}{dC_P}$
δ	swirl angle immediately behind the propeller in degrees
δ_{ff}	swirl angle far behind the propeller in degrees
$C_{Q\,x}$	tangential (in-plane) shear force coefficient (integrated from tip to root)
$C_{M\,x}$	tangential (in-plane) bending moment coefficient (integrated from tip to root)
$C_{Q\,y}$	normal (out of plane) shear force coefficient (integrated from tip to root)
$C_{M\,y}$	normal (out of plane) bending moment coefficient (integrated from tip to root)
Δp_t	total pressure increment immediately after the propeller

Table 1: Description of the tabular results on the Single Analysis card.

Definition of shear force and bending moment coefficients

In order to determine the loads on the propeller blades the local aerodynamic forces represented by the coefficients C_x and C_y are integrated along the blade from tip to root.

These coefficients are defined similar to the thrust and torque coefficients of the propeller.

Shear force due to out-of-plane axial force (thrust)

$$Q_y = C_{Q,y} \cdot \rho \cdot n^2 \cdot D^4$$

Shear force due to in-plane tangential force (torque/r)

$$Q_x = C_{Q,x} \cdot \rho \cdot n^2 \cdot D^4$$

Bending moment due to out-of-plane axial force (thrust)

$$M_y = C_{M,y} \cdot \rho \cdot n^2 \cdot D^5$$

Bending moment due to in-plane tangential force (torque/r)

$$M_x = C_{M,x} \cdot \rho \cdot n^2 \cdot D^5$$

Note that besides these aerodynamic forces and moments propellers are also largely affected by inertial loads. Torsional loads due to airfoil pitching moment or local sweep are not calculated by *JavaProp*.

Some simple validation checks

A quick validity check is the fact the axial shear force coefficient at the root must be equal to the thrust coefficient divided by the number of blades;

$$Q_y = \frac{C_T}{n_{blades}}$$

Another plausibility check is the center of thrust of each blade, which is the radial position of a single force representing the thrust of the blade. This replacement force is acting at

$$\left(\frac{r}{R}\right)_{\text{center of thrust}} = 2 \cdot \frac{C_{M\,y}(0)}{C_{Q\,y}(0)}.$$

Most propellers have their thrust force located between 60 and 70% of the radius.

What happens behind the propeller?

The propeller trails a vortex wake which induces an additional axial acceleration inside the wake. This wake system consists of helical wake sheets which roll up into hub and tip vortices and finally dissipate into heat.

Due to this wake system, the axial velocity increment caused by the propeller is doubled far behind the propeller. This also leads to a contraction of the stream tube because the static pressure does not change and the enclosed mass flow rate is the same. This contraction leads to an increase of the swirl in the slipstream because the tangential velocity because the swirl momentum must be maintained. The same effect can be observed on a dancer on the ice moving her extended arms towards the body. Without contraction the tangential velocity would stay the same. In case of a wind turbine, the opposite effects can be seen: the stream tube expands and the flow is further decelerated.

The flow field around the propeller

JAVAPROP has no capability to accurately predict the flow around the propeller. For this purpose more elaborate and time consuming methods would be required. However, a simple momentum theory model provides some basic results to show the main features of the flow through the propeller. This model includes the contraction of the stream tube as well as swirl losses. The results are presented on the Flow Field card in form of a contour plot of the axial velocity ratio (upper half of graph) and the stream tube boundary (lower half). It can be seen that half of the acceleration of the flow occurs at the propeller while the remaining half occurs due to the vortices in the slipstream.

If you analyze a wind turbine, you will notice that the stream tube is expanding, i.e. the flow is decelerated by the turbine. You will also notice that the contraction or expansion is usually relatively small. It becomes more important when a propeller is highly loaded as during takeoff.

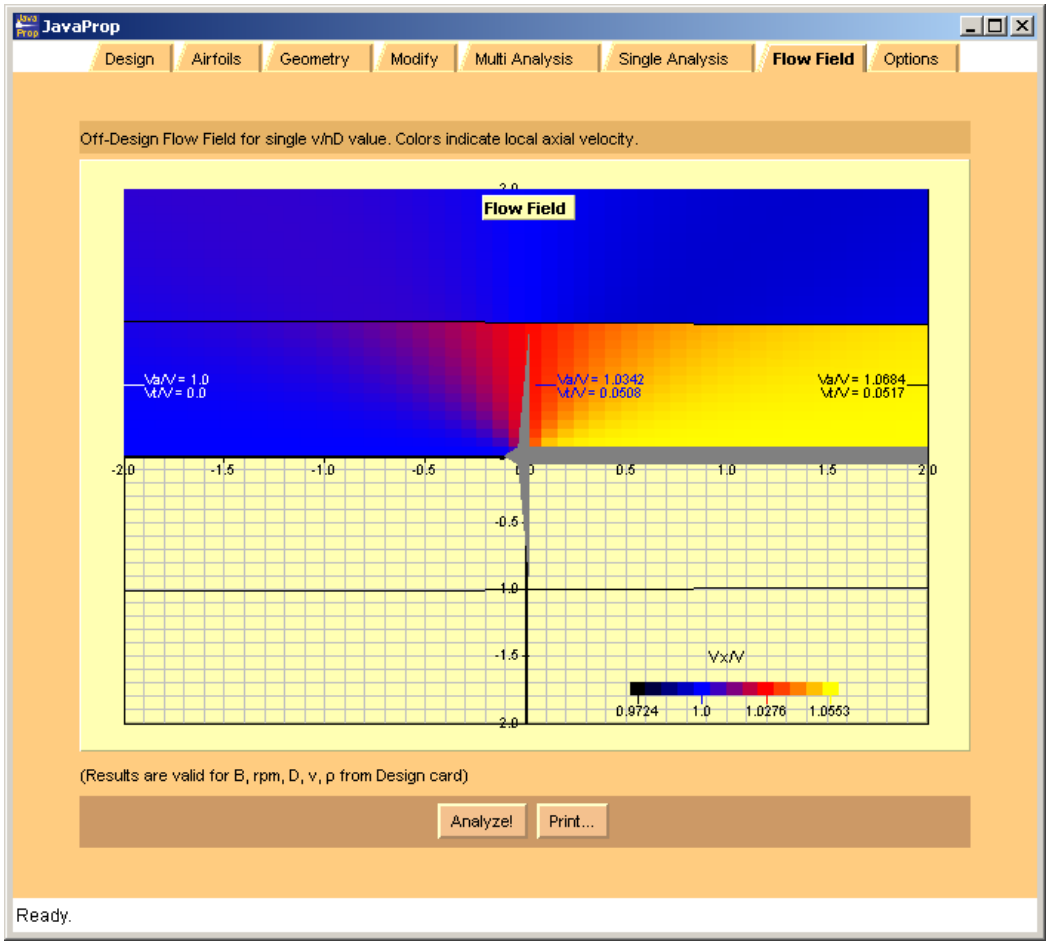


Figure 20: The Flow Field card produces a picture of the flow through the propeller.

4. Wind Turbines

While originally being designed as a propeller design and analysis tool, JAVAPROP can also be used for wind turbines. Some differences must be considered, though. Figure 21 shows the general power and thrust curves of a rotor, covering a wide speed range. In the case of propellers, only the left hand side of the graph is of interest, for wind turbines it is the right hand side. The transition between propeller and wind turbine state is fluent. Any fixed pitch propeller running at constant speed of rotation will eventually reach the windmilling state. When the air speed is increased further, it will act as a wind turbine, albeit a relatively poor one. This is because the airfoils on a wind turbine operate a negative lift and hence must be applied "upside down". Note also, that between the propeller and the wind turbine regimes there is a small range of advance ratios where the propeller already produces drag but still consumes power. This is a not very useful condition as the propeller merely creates entropy (heat). This effect is cause by friction and induced losses due to the radial lift distribution and cannot be avoided. Luckily this is only a very narrow band.

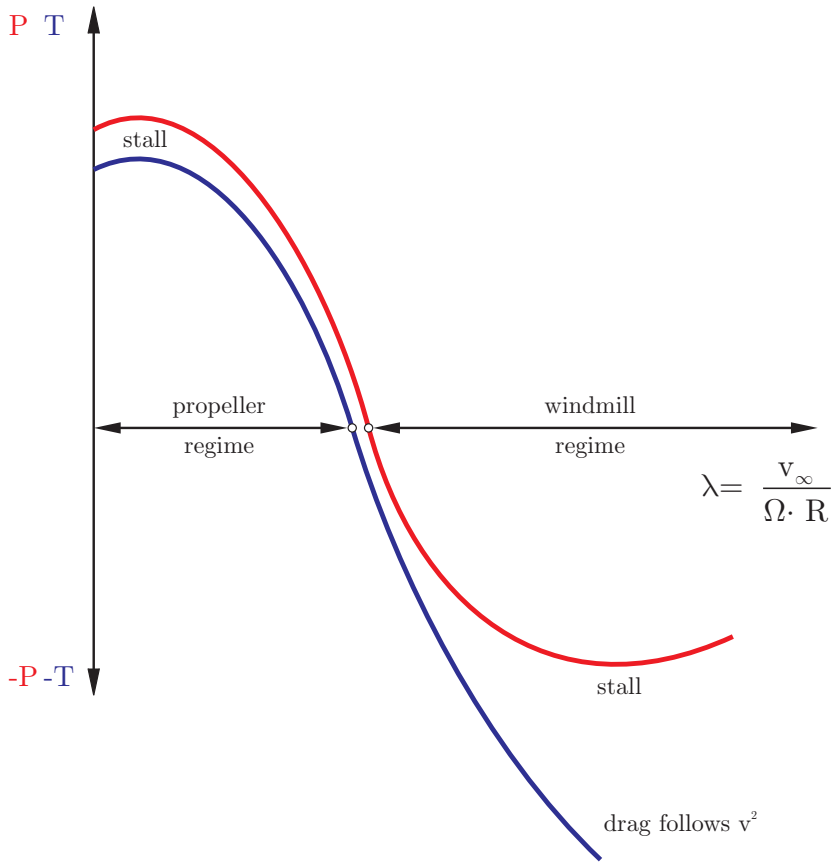


Figure 21: General operating characteristics of propellers and wind turbines, plotted versus advance ratio.

For the aerodynamic design and analysis of wind turbines the methods used for propellers can be applied. The analysis routines are the same, while the optimum design method is different, because the figure of merit of a wind turbine is different from the efficiency of a propeller.

For a propeller the figure of merit is how much thrust can be generated for a given input power. The efficiency of a wind turbine can be expressed in how much energy is extracted from the mass of air passing through the rotor disc in relation to the amount of energy contained in this stream of air. The drag (negative thrust) acting on the tower is of no primary interest, only the amount of power extracted.
Because the performance characteristics of a wind turbine start where the operating range of the propeller ends, this windmilling regime does not start at a wind speed of zero. There is a required minimum wind speed at which the wind turbine can start to turn. This has some implications on the design parameters, as a design for a too low advance ratio $v/(n \cdot D)$ would not work.
The design method implemented in JAVAPROP is based on the work of Prandtl, Betz and Glauert in the 1930s. However, it takes friction forces into account, as described in [1], which produces more realistic designs. Nevertheless, the windmill design is rather sensitive to design parameters and a subsequent Multi-Analysis may produce poor results. In this case you should move the design advance ratio to a more reasonable value, e.g. by changing the design speed of rotation n. Figure 22 gives an overview of typical operating parameters depending on the wind turbine size.

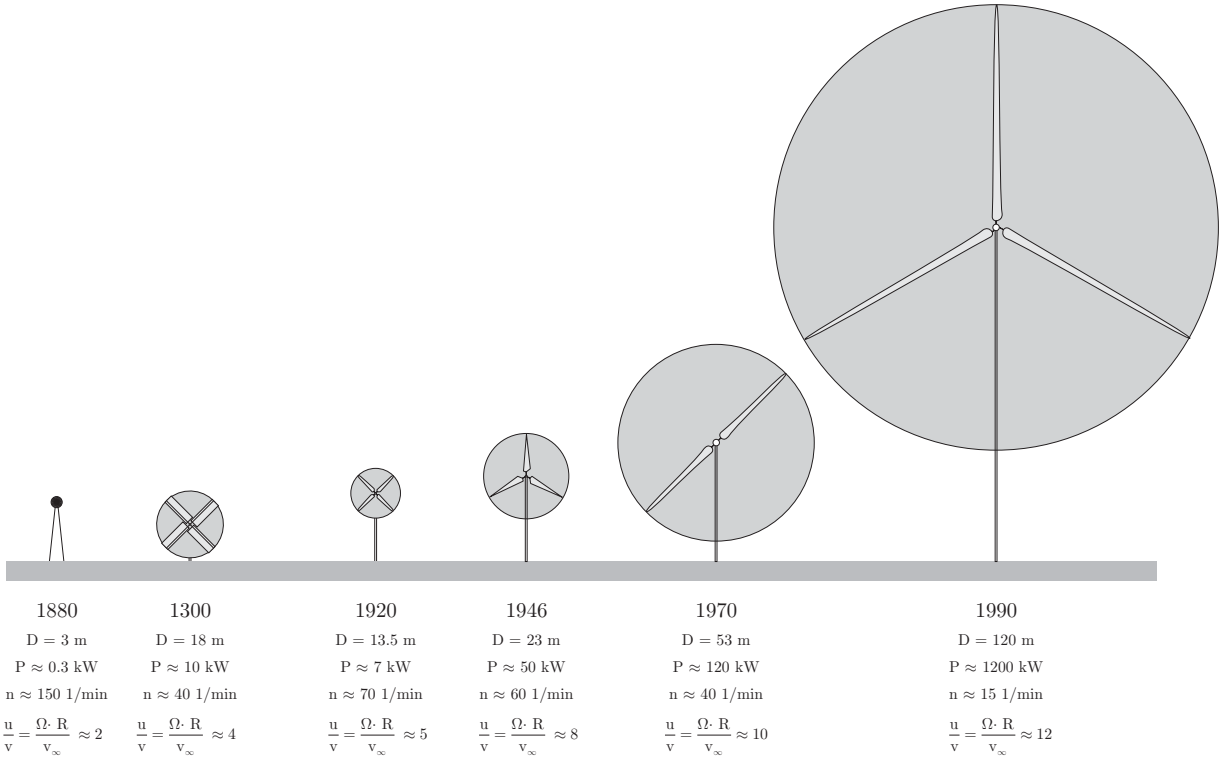


Figure 22: Typical operating parameters of different wind turbine types and sizes.

Parameters and coefficients

Note that JAVAPROP sticks to its propeller roots in maintaining the usual propeller coefficients and plots. This leads to wind turbines having negative values for torque and power as well as thrust. This sign change indicates that power and torque are delivered, not consumed, and that the thrust is actually a drag force, acting on the tower. Also the graphs of coefficients versus advance ratio are different from the common graphing of coefficients versus the tip speed ratio X, which is the reciprocal of the advance ratio λ, i.e. $X = 1/\lambda = \Omega \cdot R / v_\infty$.
Keeping the propeller conventions is not too inconvenient though, as the graphs versus $v_\infty/(n \cdot D)$ still display the behavior versus wind speed for a constant speed of rotation. Note especially, theta the power coefficient C_P as commonly used for wind turbines is not

identical to the power coefficient C_P of propellers, but to the coefficient named P_C according to propeller terminology. Also, many parameters, like the axial and circumferential induction factors of a wind turbine are output as a negative value because it delivers power instead of the propeller which absorbs power.
For comparison with wind turbine codes you should compare the coefficient P_C versus the reciprocal of the advance ratio λ, which equals the tip speed ratio X as used for wind turbines.

Figure 23 shows how the starting advance ratio of a wind turbine depends on the airfoil performance. An ideal wind turbine would start at almost zero wind speed, but due to finite lift over drag ratios a realistic wind turbine requires a minimum advance ratio to start. The required minimum wind speed v_∞ for a given generator speed Ω and average airfoil performance can be read from the figure.

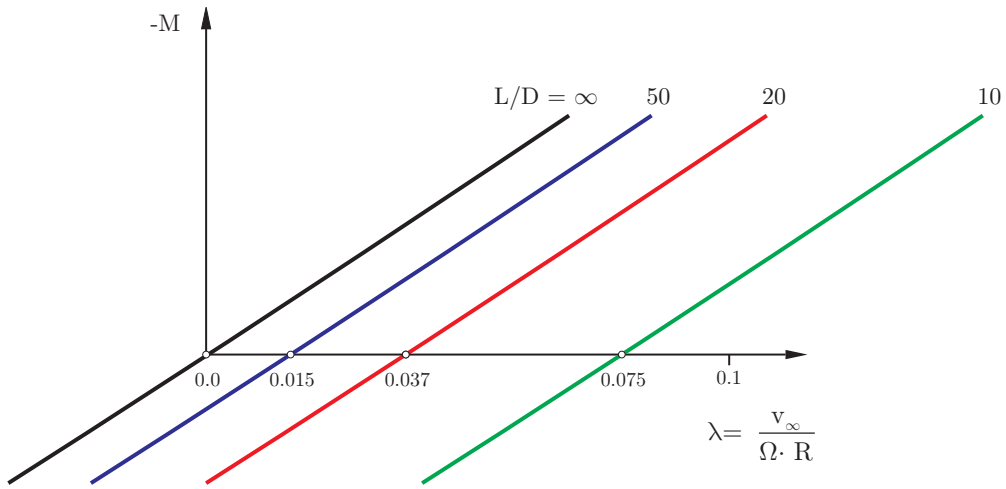


Figure 23: Effect of airfoil L/D ratio on the starting speed of a wind turbine.

Airfoil orientation

One main difference between the geometry of a wind turbine and a propeller is the orientation of the airfoil sections. JAVAPROP automatically turns the airfoils upside down if you specify a negative value for power, thrust or torque in the combo box on the design card. This indicates that a wind turbine is designed or analyzed. JAVAPROP maintains these inverted airfoils for the analysis as long as the negative value is maintained on the Design card. In terms of geometry you will note that for example in the output of the Geometry card all airfoils are arranged with an "upside down" orientation.
The inversion of the airfoils is indicated in the polar graph on the Airfoils card by an additional subtitle "(upside down)". When the state of JavaFoil is restored from a saved `.jpdata` configuration file, any negative value for power, thrust or torque will switch the airfoils to wind turbine orientation.
This upside down orientation also means that you cannot simply make an efficient wind turbine from an existing propeller. While a propeller can act as a wind turbine, its efficiency will be rather low.

How to design a wind turbine

JAVAPROP can design an optimum wind turbine for a given wind speed, speed of rotation and diameter. In order to design a wind turbine, you have to specify a negative value for the power on the Design card. The value itself is not used, only its sign is checked. Everything else is identical to the propeller design, the data on the Airfoils and the Options cards are used for the design.
The design follows the method of Glauert and therefore takes swirl losses into account but neglects friction losses. JAVAPROP determines the efficiency of a wind turbine as the ratio of the power coefficient P_C for wind turbines) to the power coefficient P_C^* which represents the maximum power which could be extracted from the stream tube passing through the rotor. Swirl losses become very large when the tip speed ratio X is considerably lower than 1.0, i.e. the wind turbine is turning too slow. At high tip speed ratios the power coefficient approaches the limit derived by Betz for zero swirl, i.e. $P_C^* = 16/27$.

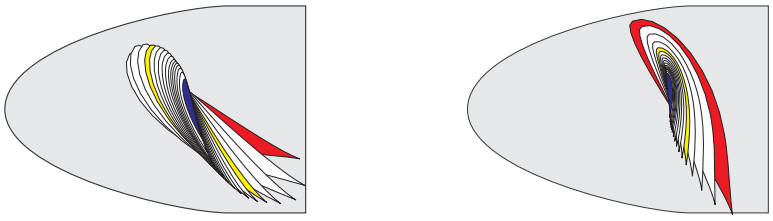


Figure 24: Orientation of airfoil sections on a propeller and on a wind turbine.

Because JAVAFOIL turns the airfoils automatically upside down when a wind turbine design is performed, the design can be performed as usual with a positive design lift coefficient at each radial station. This lift coefficient is typically close to the maximum L/D of each airfoil section.
Note that the wind turbine design procedure is rather sensitive and may produce shapes, which will not lead to realistic results in subsequent analysis. This is usually the case when the advance ratio is too far off. Compared to propeller design, where any magnitude of power can be made available, the useful range of advance ratios for wind turbines is smaller. The wind turbine draws its power only from the incoming wind and must overcome airfoil drag and momentum losses.
Therefore some experimentation may be required to achieve a reasonable design. Typical advance ratios for wind turbines are in the order of $v/(n \cdot D) = 0.5$ to $v/(n \cdot D) = 1.5$. It is also recommended to use more sophisticated airfoils than for example the flat plate.
Furthermore a spinner of a not too small diameter to cover the hub region is useful to avoid too large angles of attack towards the center of the rotor.
The design method can be used to produce a first geometry, which is later refined and modified to suit additional requirements, like a desired maximum chord length.

Large Wind Turbine		
diameter D	120	m
spinner $D_{spinner}$	4	m
speed of rotation n	12	1/min
velocity v	10	m/s
number of blades B	3	
airfoils		

Small Wind Turbine		
diameter D	0.35	m
spinner $D_{spinner}$	0.05	m
speed of rotation n	1600	1/min
velocity v	7	m/s
number of blades B	2	
airfoils		

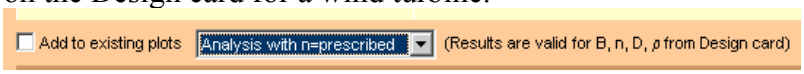
r/R=0.0	MH 126	α=14°		r/R=0.0	E 193	α=8°
r/R=0.333	MH 112	α=9°		r/R=0.333	E 193	α=7°
r/R=0.667	MH 116	α=6°		r/R=0.667	E 193	α=6°
r/R=1.0	MH 116	α=5°		r/R=1.0	E 193	α=3°

Table 2: Example cases: "Large Wind Turbine" and "Small Wind Turbine".

How to analyze a wind turbine

The single and multiple operating point analyses of JAVAPROP are performed exactly in the same way as the propeller analysis. At times, the analysis of wind turbines can be a bit more sensitive – it is recommended to use a spinner to blank out the innermost sections and to use reasonable airfoil sections, not the flat plate.
If you want to analyze an imported geometry, you should make sure that the airfoil polars are switched upside down by specifying a negative value in the power/thrust/torque combo box on the Design card.

Note that the analysis of rotors in JAVAPROP can be performed for constant values of n or V as well as constant power, thrust or torque –the last three options (constant P, T, Q) require negative values on the Design card for a wind turbine.



A wind turbine should always produce the maximum possible power output at any wind speed. Therefore it must operate at its design tip speed ratio X_{design} which links wind speed and speed of rotation by $\Omega = v_\infty \cdot X_{design} / R$. Then the power coefficient would always be the maximum of $P_{C\,design}$ and the resulting power increases with the 3rd power of the wind speed: $P = P_{C\,design} \cdot \rho / 2 \cdot v_\infty^3 \cdot \pi \cdot R^2$.

At a certain wind speed the maximum power of the generator is reached and the machine must be relieved by reducing the power coefficient. This must be done by moving the tip speed ratio off the optimum point, either by actively braking to reduce Ω or by reducing the generator load and letting the machine spin up. The latter is probably only possible for small machines where high speeds of rotation are not dangerous.

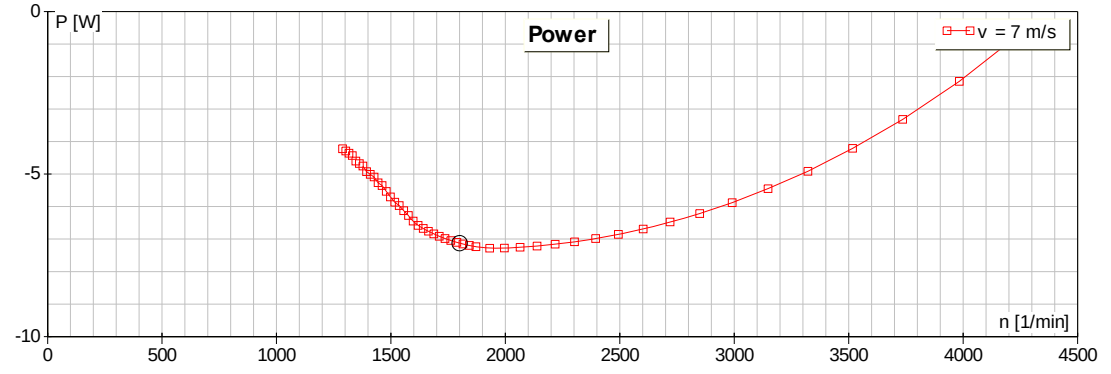


Figure 25: Variation of the power generation of the small wind turbine example with speed of rotation for a constant wind speed $v_\infty = 7\ m/s$. The circle indicates the design point close to maximum power coefficient. Power can be reduced by increasing or by decreasing n.

5. Validation of JAVAPROP

To compare the results of JAVAPROP with experimental data, a set of test results for a propeller according to design 5868-9 of the Navy Bureau of Aeronautics was selected. The propeller geometry as well as the test data can be found in NACA Report 594. The propeller used airfoils of the Clark Y type and had 3 blades. Data shown in Figure 27 are for the configuration "Nose 6, Propeller C". The blade geometry according to the NACA report has been imported into JAVAPROP via the geometry card. The blade angle at 75% of the radius was adjusted to match the angles given in the report and the results produced by the Multi-Analysis card were collected in an Excel spreadsheet. No further tweaking was performed. The comparison shows that JAVAPROP predicts the general performance characteristics in the typical "linear" operating range quite well. For this example, thrust and power are somewhat under-predicted, indicating that possibly the zero lift angle of the Clark Y airfoil in JAVAPROP might be too low. A more likely explanation however, is that the blade angle of the NACA tests refers to the flat underside of the blade while JAVAPROP uses the x-axis of the airfoil section for reference. Unfortunately the NACA reports do not give a clear indication, how exactly the blade angle was measured. In case of a Clark Y airfoil having 12% thickness, the difference amounts to about 2°. Note that the angle difference depends on the airfoil thickness if the lower surface is maintained as a straight line.

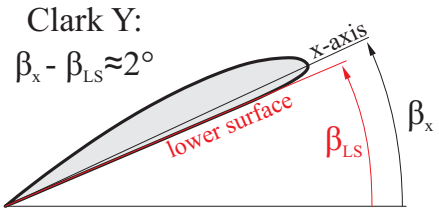


Figure 26: Possible reference lines for blade angle measurement.

Similar levels of the efficiency indicate that the lift to drag ratio of the Clark Y airfoil model in JAVAFOIL corresponds well to the tests.
Large deviations occur in the regions towards the left, where the propeller stalls. Here the flow is largely separated, three dimensional, unsteady and also depending on the external flow field (e.g. crosswind, wind tunnel interference). Such flow regimes are beyond the assumptions of the underling theory so that no good match can be expected here. It should be noted, that the experimental data show considerable scatter and irregular behavior in this regime too.

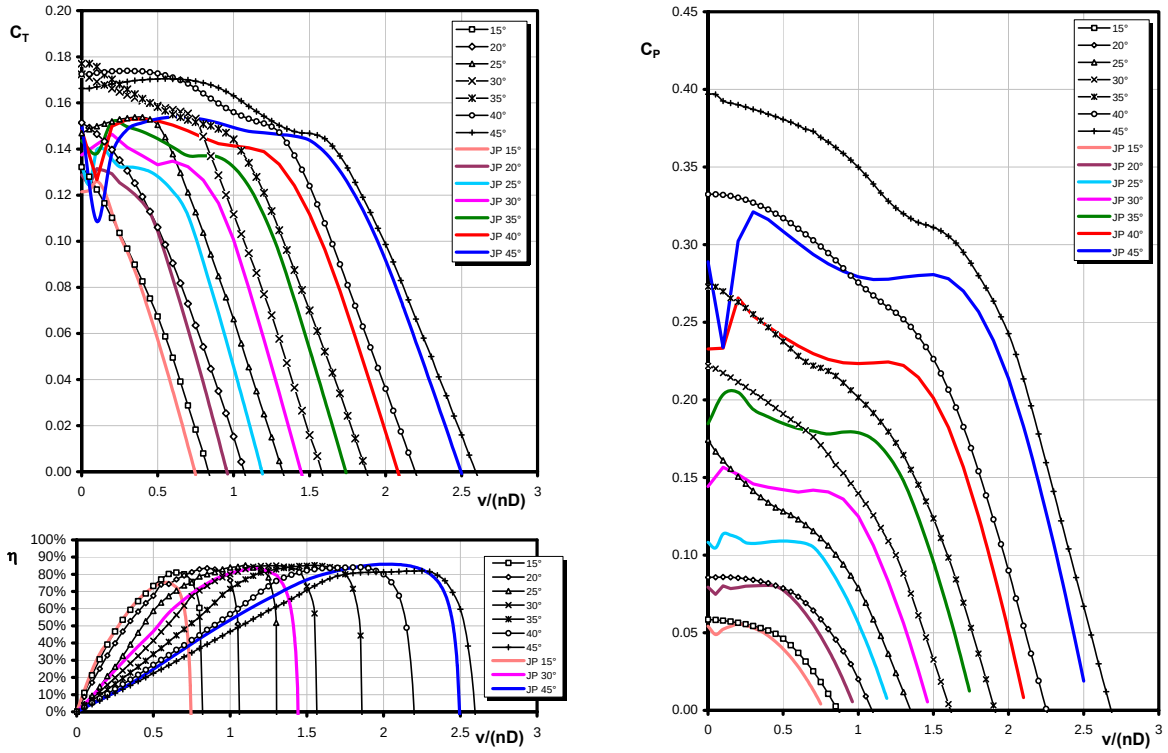


Figure 27: Prediction by JAVAPROP and experimental data from NACA R-594 (symbols).

Another comparison was performed for the propeller described in NACA Report 350. The results compare slightly worse, but still acceptable. The power coefficient predicted by JAVAPROP drops steeper with increasing advance ratio than the experiments indicate. This may be due to differences in the airfoil polars towards lower and negative lift coefficients.

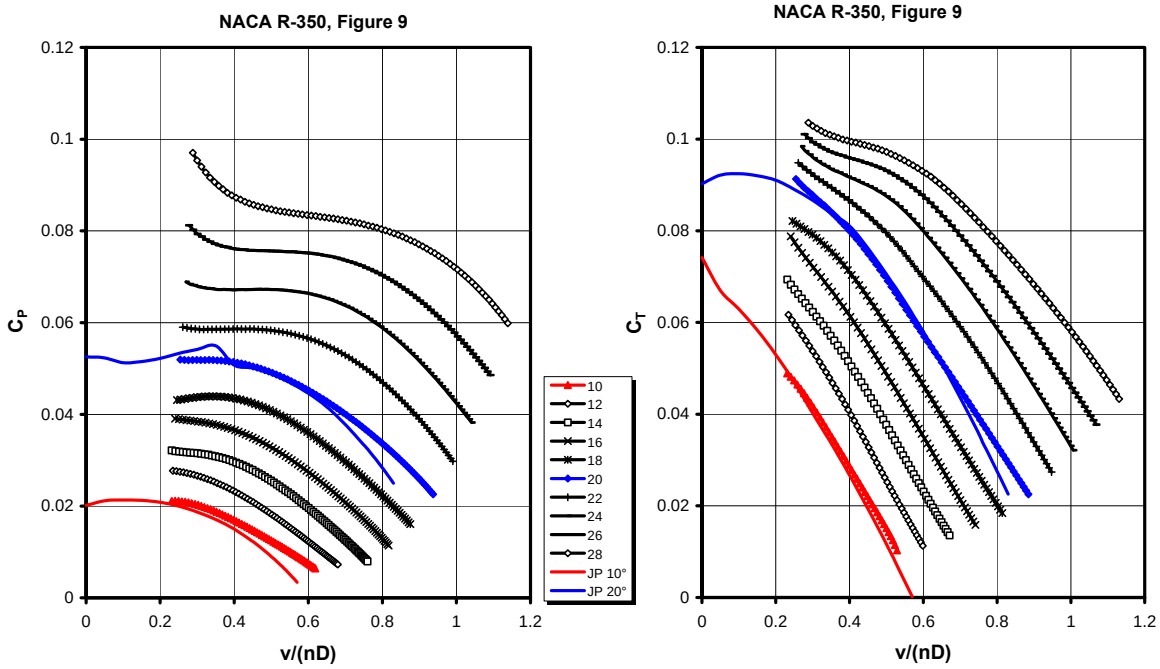


Figure 28: Data predicted by JAVAPROP and experimental data from NACA R-350 (symbols).

6. Controlling JAVAPROP from external applications

Using JavaProp with GNU OCTAVE

GNU OCTAVE [10] is very similar to Matlab and allows you to use JAVAPROP directly from an OCTAVE script. This enables you to perform more complex tasks without using the JAVAPROP GUI, for example parameter sweeps or numerical optimization. The example script below shows all major steps to design and analyze a propeller.

```
%-------------------------------------------------------------------------
%
% A function to encapsulate the JavaProp propeller code
%
%-------------------------------------------------------------------------
% Can be used with MatLab (tested with version 7.9.0)
% as well as with GNU Octave (tested with version 3.2.4).
%
% Requirements:
% For Octave the package java-1.2.7 or higher is required.
%
% Written by Martin Hepperle, December 2010.
% The initial version of this example has been developed by
% Ed Waugh, University of Southampton, 2007.
%-------------------------------------------------------------------------
%
% JavaProp is Copyright 2001-2013 Martin Hepperle
% http://www.mh-aerotools.de/airfoils/javaprop.htm
%
%-------------------------------------------------------------------------

%% DesignProp

function DesignProp ()

   % add the archives to the Java classpath (must be done only once
   % per session, but can be called multiple times)
   basepath = '..\java';
   javaaddpath ( [basepath, '\', 'JavaProp.jar'] );
   javaaddpath ( [basepath, '\', 'MHClasses.jar'] );

   % Number of blade sections, increasing this means a longer run time
   % but greater accuracy.
   % Sufficiently accurate results are obtained with 25 to 50 blade elements.
   blade_sections = 21;

   % Create a Propeller object to work on and name it
   PropDesign = javaObject ( 'MH.JavaProp.Propeller', blade_sections );
   PropDesign.Name = 'JavaProp-Test';
```

```
% environmental parameters (atmosphere at sea level)
PropDesign.Density = 1.225;                    % [kg/m^3]
PropDesign.KinematicViscosity = 0.000014607;    % [m^2/s]
PropDesign.SpeedOfSound = 340.29;              % [m/s]

% Define the airfoil distribution
% clear all values from airfoil distribution
PropDesign.removeAirfoils();

% define new values for airfoil distribution
theAirfoil = createAirfoil ( 3 );
PropDesign.setAirfoil ( 0.000, theAirfoil );

theAirfoil = createAirfoil ( 3 );
PropDesign.setAirfoil ( (1/3), theAirfoil );

theAirfoil = createAirfoil ( 4 );
PropDesign.setAirfoil ( (2/3), theAirfoil );

theAirfoil = createAirfoil ( 4 );
PropDesign.setAirfoil ( 1.000, theAirfoil );

% define the design angles of attack (degrees)
% clear all values from angle of attack distribution
PropDesign.removeAnglesOfAttack();
% define new values for angle of attack distribution
PropDesign.addAlfa ( 0.000, 3.0 );
PropDesign.addAlfa ( (1/3), 3.0 );
PropDesign.addAlfa ( (2/3), 3.0 );
PropDesign.addAlfa ( 1.000, 3.0 );

% set prop dimensions [m]
Diameter = 0.457;
Radius = Diameter / 2;

% number of blades
PropDesign.BladeCount = 2;

% set the spinner size [m]
SpinDiameter = 0.126;
PropDesign.rRSpinner = SpinDiameter / Diameter;

% Set the design conditions for the propeller
% The geometry of the resulting prop will be optimized to give the maximum
% efficiency at the point specified. Either Thrust or Power can be defined
% but not both. One must be set to zero. To use a Torque value, set both
% Power and Thrust to zero.

Airspeed = 30;                  % [m/s]
RPM = 5000;                     % [revs/min]

Frequency = (RPM / 60);         % frequency [Hz]
Omega = 2 * pi * Frequency;     % angular velocity [rad/s]

% specify one of these three, set the others to zero
Power = 0;                      % [Watt}
Thrust = 0;                     % [Newton}
Torque = 0.915;                 % [Nm}

if Power == 0                   % if power and thrust are zero use torque
   if Thrust == 0
```

```
        Power = Torque * Omega;
      end
    end

    % create a propeller
    PropDesign.performPropellerDesign ( Airspeed, Omega, …
                                        Radius, Power, Thrust );

    % perform an analysis at a lower advance ratio (just as an example)
    PropDesign.performAnalysis ( 0.5*Airspeed / (Frequency * Diameter) );

    % assign the PropDesign object to the base workspace
    assignin ( 'base','PropDesign', PropDesign );

end
%--------------------------------------------------------------------------
function [theAirfoil] = createAirfoil ( AirfoilNo )
   %
   % Generate and initialize an airfoil.
   %
   % It is also possible to set a base directory which is later used,
   % when Init(n) is called with n >= 14  to read airfoil polars
   % from file af_1.afl (or, if not found from af_1.xml) in JP directory
   %
   % theAirfoil.setBaseDir("c:/...");
   % theAirfoil.Init ( 14 );
   %

   theAirfoil = javaObject ( 'MH.AeroTools.Airfoils.Airfoil' );
   theAirfoil.Init ( AirfoilNo );

end
```

Using JavaProp with MATLAB

Like with OCTAVE is also possible to access its classes from a MATLAB [11] script. The application is identical to integration into GNU OCTAVE.

Using JavaProp with MATHEMATICA

WOLFRAM MATHEMATICA [12] can also interface to Java classes. As I am not an expert in MATHEMATICA, the following notebook is probably not the most elegant way to use MATHEMATICA, but it should suffice to demonstrate how to interface JAVAPROP and MATHEMATICA. Note the comments in the example about how to use a recent Java runtime.

```
 (* This simple example shows how JavaProp can be used from Wolfram \
Mathematica 9.0 *)

(* prepare environment *)
path = "D:\\Users\\Martin Hepperle\\workspace\\JavaProp\\java";
Needs["JLink`"];
InstallJava[];

(* Adapt as needed by your local installation.
```

```
 I tested briefly with the evaluation version of Mathematica 8.0 and 9.0
  but I am no expert in Mathematica nor do I have the software available on my personal system.
 I have renamed the subdirectory
 "D:\\Program Files\\Wolfram\
Research\\Mathematica\\9.0\\SystemFiles\\Java\\Windows" to "__Windows" and also
 "Windows-x86-64" to "__Windows-x86-64"
 the so that Mathematica cannot find its own Java installation.

  It then seems to use the Java runtime it finds via the system PATH - I have
 "D:\\Java\\jdk1.8.0_05\\bin" in my PATH environment variable.

 If I do not remove the access to the Java runtime installed by Wolfram,
 Mathematica complains about class files being in unreadable format version 51.
 It is possible that later version (10.0 and up) install a newer Java runtime so that this procedure
 is not necessary anymore.
*)
AddToClassPath[path <> "\\JavaProp.jar"];
AddToClassPath[path <> "\\JavaClasses.jar"];

(* define the number of blade sections for elemental discretization *)
bladeSections = 20;

(* Create a Propeller object to work on and name it *)
PropDesign =
  JavaNew["MH.JavaProp.Propeller", bladeSections];
PropDesign@Name = "JavaProp from Mathematica";

(* define environment - note: metric units are used throughout JavaProp *)
(* density of air in kg/m^3 *)
PropDesign@Density = 1.2210;
(* kinematic viscosity of air in m^2/s *)
PropDesign@KinematicViscosity = 0.000014607;
(* speed of sound in air in m/s *)
PropDesign@SpeedOfSound = 340.29;

(* Define the airfoil distribution by airfoilSections
 Syntax: setAirfoil(N,r/R,Section)
 N - Airfoil number in list order,0...(airfoilSections-1)
 r/R - Position ratio of the foil, R=Radius, r=position
       0.0 = root, 0.5=center, 1.0=tip
 Section - A number from the following list of airfoil sections
 1-Flat plate,Re=100'000
 2-Flat plate,Re=500'000
 3-Clark Y,Re=100'000
 4-Clark Y,Re=500'000
 5-E 193,Re=100'000
 6-E 193,Re=300'000
 7-ARA D 6%,Re=50'000
 8-ARA D 6%,Re=100'000
 9-MH 126,Re=500'000
 10-MH 112 16.2%,Re=500'000
 11-MH 114 13%,Re=500'000
 12-MH 116 9.8%,Re=500'000
 13-MH 120 11.7%,Re=400'000,M=0.75
 14-Read from file af_1.afl (or,if not found from af_1.xml) in JP directory
 15-Read from file af_2.afl (or,if not found from af_2.xml) in JP directory
*)

(* Example is a Clark Y Propeller using two different airfoil polars *)
 PropDesign@removeAirfoils[];
```

```
AirfoilNo = 3;
theAirfoil = JavaNew["MH.AeroTools.Airfoils.Airfoil"];
theAirfoil@Init [AirfoilNo];
PropDesign@setAirfoil[0.0, theAirfoil];

theAirfoil = JavaNew["MH.AeroTools.Airfoils.Airfoil"];
theAirfoil@Init [AirfoilNo];
PropDesign@setAirfoil[(1.0/3.0), theAirfoil];

AirfoilNo = 4;
theAirfoil = JavaNew["MH.AeroTools.Airfoils.Airfoil"];
theAirfoil@Init [AirfoilNo];
PropDesign@setAirfoil[(2.0/3.0), theAirfoil];

theAirfoil = JavaNew["MH.AeroTools.Airfoils.Airfoil"];
theAirfoil@Init [AirfoilNo];
PropDesign@setAirfoil[1.0, theAirfoil];

(* define the design angles of attack (in degrees) *)
PropDesign@removeAnglesOfAttack[];
PropDesign@setAlfa[0.000, 4.0];
PropDesign@setAlfa[(1.0/3.0), 3.0];
PropDesign@setAlfa[(2.0/3.0), 2.0];
PropDesign@setAlfa[1.000, 1.0];

(*============================================================*)

(* Set prop dimensions (meters) *)
Diameter = 0.457;
Radius = Diameter/2;

(* Number of blades *)
PropDesign@BladeCount = 2;

(* Set the spinner size (in meters) *)
SpinDiameter = 0.126;
PropDesign@rRSpinner = SpinDiameter/Diameter;

(* Set the design conditions for the propeller
   The geometry of the resulting prop will be optimized to give the maximum
   efficiency at the point specified. Either Thrust or Power can be defined
   but not both. One must be set to zero. To use a torque value, set both
   power and thrust to zero. *)

(* flight speed meters/second *)
Airspeed = 50.0;
(* propeller shaft revolutions per minute *)
RPM = 5000;

(* angular velocity rad/s *)
Omega = 2.0*Pi*(RPM/60);
(* Watt *)
ShaftPower = 0;
(* Newton *)
Thrust = 0;
(* Nm *)
Torque = 0.915;

(* If power and thrust are zero use torque *)
if [ShaftPower == 0  ,
  if [  Thrust == 0 , ShaftPower = Torque*Omega  ] ];
```

```
(* create a propeller geometry by design *)
PropDesign@
  performPropellerDesign[Airspeed, Omega, Radius, ShaftPower, Thrust];

(* now that we have a design, we can modify is, can be omitted, only for demonstration *)
PropDesign@
 incrementBladeAngle[
  0.0];(* Increases the local blade angle Beta but constant dBeta *)
PropDesign@
 multiplyBladeAngle[
  1.0];  (* Multiplies local blade angle Beta by a constant *)
PropDesign@
 incrementChord[
  0.0];        (* Increments local chord c/R by constant c/R *)
PropDesign@
 multiplyChord[
  1.0];       (* Multiplies local chord c/R by constant c/R factor *)
\

PropDesign@
 taperChord[
  1.0];          (* Multiplies local chord c/R by a linear varying \
c/R fact or *)

(* create a nice table of results *)
resultTable1 = Text@Grid[
   {
    {PropDesign@Name, SpanFromLeft},
    {"Blades", PropDesign@BladeCount},
    {"RPM", RPM, "1/min"},
    {"v", PropDesign@V, "m/s"},
    {"Diameter", Diameter, "m"},
    {"Thrust", PropDesign@Thrust, "N"},
    {"Power", PropDesign@Power, "W"},
    {"Torque", PropDesign@getTorque[], "Nm"},
    {"v/(nD)", Airspeed/(RPM/60.0*Diameter)},
    {"Pitch", PropDesign@getBladePitch[], "m"},
    {"Blade Angle", PropDesign@getBladeAngle[], "\[Degree]"},
    {"CP", PropDesign@CP},
    {"CT", PropDesign@CT},
    {"Eta", PropDesign@Eta}
    },
   Frame -> All];

cr = Transpose[{PropDesign@rR, PropDesign@Chord}];
br = Transpose[{PropDesign@rR, 180.0*PropDesign@Beta/Pi}];

(* Perform an off-design analysis at prescribed v/(nD) *)
PropDesign@performAnalysis[0.5*Airspeed/((RPM/60)*Diameter)];

(* create a nice table with off-design results *)
resultTable2 = Text@Grid[
    {
     {"Off-Design", SpanFromLeft},
     {"Blades", PropDesign@BladeCount},
     {"RPM", RPM, "1/min"},
     {"v", PropDesign@V, "m/s"},
     {"Diameter", Diameter, "m"},
     {"Thrust", PropDesign@Thrust, "N"},
     {"Power", PropDesign@Power, "W"},
```

```
    {"Torque", PropDesign@getTorque[], "Nm"},
    {"v/(nD)", Airspeed/(RPM/60.0*Diameter)},
    {"Pitch", PropDesign@getBladePitch[], "m"},
    {"Blade Angle", PropDesign@getBladeAngle[], "\[Degree]"},
    {"CP", PropDesign@CP},
    {"CT", PropDesign@CT},
    {"Eta", PropDesign@Eta}
    },
    Frame -> All];

(* terminate any Java virtual machines and processes *)
UninstallJava[];

(* show results *)
Join[resultTable1, resultTable2]
ListLinePlot[cr, PlotLabel -> "chord length distribution",  AxesLabel -> {"r/R", "c/R"}]
ListLinePlot[br, PlotLabel -> "blade angle distribution",  AxesLabel -> {"r/R", "beta [\[Degree]]"}]
```

The resulting tables and graphs should look like the following figure.

Out[712]= Text[

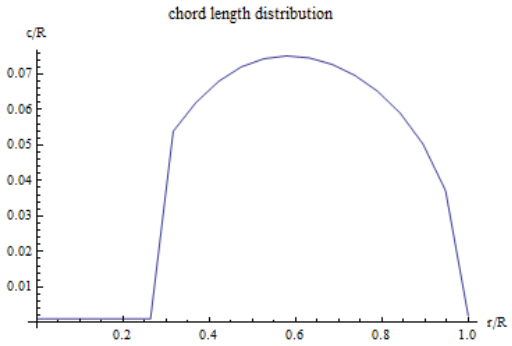
JavaProp from Mathematica		
Blades	2	
RPM	5000	1/min
v	50.	m/s
Diameter	0.457	m
Thrust	7.87687	N
Power	479.081	W
Torque	0.914977	Nm
v/(nD)	1.31291	
Pitch	0.670034	m
Blade Angle	31.8922	°
CP	0.0340139	
CT	0.0212978	
Eta	0.822082	

,

Off-Design		
Blades	2	
RPM	5000	1/min
v	25.	m/s
Diameter	0.457	m
Thrust	20.0035	N
Power	824.61	W
Torque	1.57489	Nm
v/(nD)	1.31291	
Pitch	0.670034	m
Blade Angle	31.8922	°
CP	0.0585458	
CT	0.0540865	
Eta	0.606454	

]

Out[713]=



Out[714]=

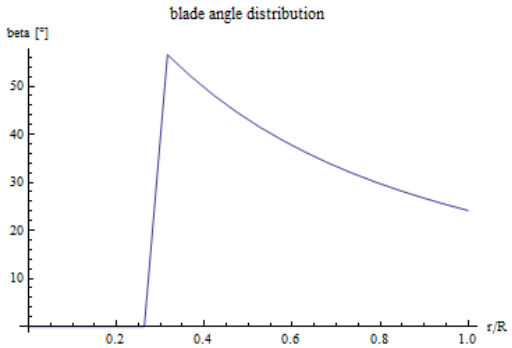


Figure 29: Results produced by calling JAVAPROP from Mathematica 9.0.

Using JavaProp with MAPLE

In order to spare you lengthy trials I must say that Maple has only a very limited and almost useless support for Java. It does not support the creation of Java objects and therefore cannot easily interact with JAVAPROP without the creation of extensive and ugly workarounds.

The Propeller Object Model

The public fields, properties and methods available in the Propeller object are described in a separate PDF JavaDoc document. These can be used from all Java enabled programming environments as demonstrated above.

7. References

[1] A. Betz: “Schraubenpropeller mit geringstem Energieverlust” (with an Addendum by L. Prandtl) in “Vier Abhandlungen zur Hydrodynamik und Aerodynamik”, reprint of the 1927 edition, Göttingen, 1944.

[2] H.Glauert: “Die Grundlagen der Tragflügel und Luftschraubentheorie”, Translation by H. Holl, Berlin 1929.

[3] H. B. Helmbold: “Über die Goldstein’sche Lösung des Problems der Luftschraube mit endlicher Flügelzahl”, Zeitschrift für Flugtechnik und Motorluftschifffahrt, 14. Heft, 1931.

[4] Larrabee, E., “Practical Design of Minimum Induced Loss Propellers”, Society of Automotive Engineers Business Aircraft Meeting and Exposition, Wichita, KS, SAE paper 790585, April 3-6, 1979.

[5] Adkins, Liebeck, N., R. H., “Design of Optimum Propellers”, Journal of Propulsion and Power, Vol. 10, No. 5, 1994, pp. 676-682.

[6] Adkins, N., Liebeck, R. H., “Design of Optimum Propellers”, 21st Aerospace Sciences Meeting, Reno, AIAA Paper 83-0190, Jan. 1983.

[7] Wald, Q. R., “The aerodynamics of propellers”, Progress in Aerospace Sciences, 42, 2006, pp. 85-128, doi:10/1016/j.paerosci.2006.04.001.

[8] R. Eppler and M. Hepperle: “A procedure for Propeller Design by Inverse Methods”, in G.S. Dulikravich: Proceedings of the “International Conference on Inverse Design Concepts in Engineering Sciences” (ICIDES), pp. 445-460, Austin, October 17-18, 1984.

[9] K. Wiederhöft, “60 Jahre nach Hütter. Aerodynamische Radialschnitttheorie für Windenergieanlagen”, DEWEK 2002.

[10] GNU Octave: see http://www.octave.org [retrieved 20 December 2010].

[11] Mathworks Matlab: see http://www.mathworks.com/ [retrieved 20 December 2010].

[12] Wolfram Mathematica: see http://www.wolfram.com/ [retrieved 20 December 2010].

8. Localization

The language resources of JavaProp have been translated by the persons listed in the table below. Over time new strings may have been added so that the translations may not be up-to-date. In this case you are kindly asked to supply the missing or incorrect strings so that I can integrate them into JAVAPROP.

Language	**Code**	**Author**	**Year**
English	en	Martin Hepperle, Braunschweig, Germany	2001
German	de	Martin Hepperle, Braunschweig, Germany	2001
French	fr	Giorgio Toso, Montreal, Canada	2002
Italian	it	Giorgio Toso, Montreal, Canada	2002
Portuguese	pt	João Alveirinho Correia, Portugal	
Dutch	nl	not yet done	
Simplified Chinese	zh_CN	J. X. Ding, Taipei, Republic of China	2011
Traditional Chinese	zh_TW	J. X. Ding, Taipei, Republic of China	2011

9. Incomplete Version History

Version	Date	Comments
1.69	1 Aug 18	Added correction to maintain design lift coefficients, added buttons “Air” and “Water” on Options card for setting default properties for medium.
1.68	1 Jun 16	Added option for hub loss factor.
1.67	26 Jan 16	Fixed import of external polar files.
1.66	19 Apr 15	Fixed determination of activity factor.
1.65	25 Jan 15	Added analysis after import of geometry on Geometry card so that Design card reflects imported geometry Momentum integrals added to analysis to obtain power in axial and swirl momentum as well as power lost. Added methods to adjust blade angle for desired CP resp. CT (not available via the GUI). Added methods to adjust propeller speed for desired power resp. thrust (not available via the GUI). Simplified addition of airfoil sections and design angle of attack distribution (requires changes to external scripting application)
1.64	05 Jan 14	Made import function on Geometry card more robust.
1.63	20 Oct 13	Added threading line and trailing edge thickness to Modify card.
1.62	22 July 13	Added v_{ax}/v and v_{tan}/v to Single Analysis table.
1.61	24 May 13	Added option to supply shapes for user defined airfoils. Changed polar naming scheme to be consistent with geometry naming scheme.
1.6	12 May 13	Added plots of wake properties to Geometry card
1.59	01 Nov 12	Smaller modifications and increased robustness.
1.58	1 Mar 12	Fixed locale dependency of floating point data in .jpdata. Added saving and restoring of last settings. Added local blade thickness to table on Geometry card.
1.57	25 Aug 11	Added Chinese language features.
		Added some additional resource strings.
		Cleaned up expiration reminder code.
		Added saving and restoring of most GUI settings on exit and startup to application preferences.
		Added styled labels to graphs for sub- and superscripts to mhclasses.jar.
1.56	07 Jan 11	Introduced sweep angle as a function of local Mach number
1.55	30 Jun 10	Fix: airfoil list was not updated when working directory was changed by WorkDir=.... command line option.
		Fix: Geometry table was not updated when airfoil was selected.
		PROPPY compatible XML export of propeller geometry added.
1.54	31 Mar 10	local thrust, power, efficiency added to single analysis card, windmill design method Wiederhöft added.

1.53	01 Jan 10	Refined Multi-Analysis card, added V=const. analysis mode.
1.52	10 Oct 09	IGES export of 3D blade section geometry added
1.51	20 Sep 09	Added shear force and bending moment graphs to single analysis card.
1.5	02 Sep 09	Added windmill design and analysis.
1.41		Additional output of shear force and bending moment coefficients on single analysis card.
1.4		Refinement of Multi-Analysis card, Q=const. added.
1.39		Refinement of Multi-Analysis card, RPM graph added.
1.38		Fix: error in save/restore of Torque setting on design card
1.37		Fix: error in DXF 3D export (spinner was neglected in
		DXF j-index).
1.36		Fix: angle of attack transfer error on Airfoil card.
		Added iteration on Design card so that analysis matches design parameters accurately.
1.35		Modified clipping of induction factor a during analysis.
1.33		Extended Single-Analysis CopyText and CopyHTML commands to include more data.
1.3		Changed XYCanvas sizing.
1.29		Fix: Error in Modify card: angle always went to 90°
1.27		Fix: Flow Field card calculation of "a" (dA).
1.26		Added Flow Field card.
1.0	2001	Initial version