Propeller Generating User-defined Primitive (UDP) in Engineering Sketch Pad

Daniel Oluwalana

Syracuse University, Syracuse, NY, 13244, USA

Engineering Sketch Pad (ESP) is a web-based system used to create and manipulate geometry for the aim of designing and analyzing aerospace vehicles. There are User-defined Primitives that are pre-packaged with ESP; however, the system also allows users to create their own single body primitives written in C, C++ or FORTRAN and have them coupled with ESP and compiled in real time. The purpose of this paper is to detail the process used in creating a User Defined Primitive (UDP) within Engineering Sketch Pad that generates a propeller using a well-established design process. Prior to this propeller scheme implementation in the software, a user would generate a propeller by manually arranging a series of airfoils at certain angles and applying a covering or skin over them, an inefficient method as users would have to permute the airfoil arrangements to achieve the design shape and power. UDP Propeller is the first power/thrust auto-derived propeller primitive to be implemented in any CAD software as it creates optimum propeller blades for an aircraft’s engine based on specifications from a user such as power coefficient and advance ratio.

I. Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>angle of attack</td>
</tr>
<tr>
<td>a</td>
<td>axial interference factor</td>
</tr>
<tr>
<td>B</td>
<td>number of propeller blades</td>
</tr>
<tr>
<td>β</td>
<td>angle of twist</td>
</tr>
<tr>
<td>C_l</td>
<td>lift coefficient at the airfoil section</td>
</tr>
<tr>
<td>C_d</td>
<td>drag coefficient</td>
</tr>
<tr>
<td>c</td>
<td>chord length</td>
</tr>
<tr>
<td>ε</td>
<td>drag-to-lift ratio</td>
</tr>
<tr>
<td>ζ</td>
<td>displacement velocity ratio</td>
</tr>
<tr>
<td>F</td>
<td>Prandtl momentum loss factor</td>
</tr>
<tr>
<td>G</td>
<td>circulation function,</td>
</tr>
<tr>
<td>θ</td>
<td>Blade rotation angle</td>
</tr>
<tr>
<td>ξ</td>
<td>nondimensional radius</td>
</tr>
<tr>
<td>ξ_0</td>
<td>nondimensional hub radius</td>
</tr>
<tr>
<td>ρ</td>
<td>density of air,</td>
</tr>
<tr>
<td>P_c</td>
<td>power coefficient,</td>
</tr>
<tr>
<td>R</td>
<td>airfoil radial coordinate</td>
</tr>
<tr>
<td>R</td>
<td>propeller tip radius,</td>
</tr>
<tr>
<td>Re</td>
<td>local Reynolds number</td>
</tr>
<tr>
<td>Re_g</td>
<td>global Reynolds number</td>
</tr>
<tr>
<td>λ</td>
<td>speed ratio (V/ΩR)</td>
</tr>
<tr>
<td>ν</td>
<td>kinematic viscosity of air</td>
</tr>
<tr>
<td>ν’</td>
<td>vortex axial velocity</td>
</tr>
<tr>
<td>V</td>
<td>freestream velocity of air</td>
</tr>
<tr>
<td>W</td>
<td>local total velocity,</td>
</tr>
<tr>
<td>w_n</td>
<td>velocity normal to the vortex sheet</td>
</tr>
<tr>
<td>w_t</td>
<td>tangential velocity to the vortex sheet</td>
</tr>
<tr>
<td>φ</td>
<td>flow angle</td>
</tr>
<tr>
<td>φ_h</td>
<td>flow angle at the tip of the airfoil.</td>
</tr>
<tr>
<td>x</td>
<td>nondimensional distance,</td>
</tr>
<tr>
<td>Ω</td>
<td>propeller’s angular velocity</td>
</tr>
</tbody>
</table>

II. Introduction

A. Momentum Theory and Blade Element Geometry

A propeller blade is formed by blending a series of airfoils arranged along radial positions from the hub to the tip. The momentum theory applied to the blade’s geometry is utilized in sizing and twisting each of those airfoils. The momentum theory, in conjunction with Betz condition express the idea that at each radial position, there are helicoidal vortex sheets as shown in Figure 1 below which must have the same regular screw-like surface in order to minimize energy loss [1].

1 4th Year Undergraduate, Aerospace Engineering, AIAA Student Member
With reference to Fig 1 above and trigonometry, the tangential velocity ($w_t$) is shown to be a function of the velocity normal to the vortex sheet ($w_n$) and the flow angle ($\phi$). Likewise, parallel to the axis of the machine, the axial velocity of the vortex ($v'$) can be expressed a function of $w_n$ and $\phi$. Interestingly, Fig 1 experiences a similar phenomenon to barber pole stripes where they seem to be in translational motion while only possessing a rotational velocity in effect\[1\]. Thus, $v'$ which can also be referred to the vortex displacement velocity must be constant to keep the screw-like surface of the vortex hence and minimize energy loss. For good practice, a non-dimensional parameter can be generated from the ratio of $v'$ to the freestream velocity of air ($V$).

Mathematically,

$$\zeta = \frac{v'}{V}$$

(1)

where $\zeta$ is the displacement velocity ratio.

Due to the different vortices being formed, there is a need to express the circulation as a function of the momentum loss, flow angle and radial position. Thus,

$$G = F \cos(\phi) \sin(\phi)$$

(2)

where $G$ is the circulation function, $F$ is the Prandtl momentum loss factor, and $x$ is the nondimensional distance. Furthermore,

$$x = \frac{\Omega r}{V}$$

(3)

where, $\Omega$ is the propeller’s angular velocity, and $r$ is the radial coordinate of the airfoil.

As air moves upstream towards the propeller actuator disc, it arrives with a velocity greater than its original freestream velocity, $V(1+a)$, where $a$ is the axial interference factor\[1\]. Extending the parameters in equations 1–3, this factor can be expressed as

$$a = \left(\frac{\zeta}{2}\right) \cos^2 \phi \left(1 - \epsilon \tan \phi\right)$$

(4)

where $\epsilon$ is the drag-to-lift ratio. Typically, when designing a propeller, the power supplied to it is given and can be used to solve for the constant displacement velocity ratios at each radial position. Adkins and Liebeck supply a series of equations that provide the mathematical rigor to obtain a solution for $\zeta$ by integration.

$$P_c = \frac{2P}{\rho V^3 n R^2}$$

(5)

$$P'_c = J'_1 \zeta + J'_2 \zeta^2$$

(6)

$$J'_1 = 4 \xi G \left(1 + \frac{\epsilon}{\tan \phi}\right)$$

(7a)

$$J'_2 = \left(\frac{J'_1}{2}\right) (1 - \epsilon \tan \phi) \cos^2 \phi$$

(7b)
$$\zeta = -\left(\frac{J_1}{2J_2}\right) + \left(\frac{J_1}{J_2}\right)^2 + \frac{1}{J_2}$$

(8)

where R is the propeller tip radius, \( \rho \) is the density of air, \( P_c \) is the power coefficient, and \( \xi \) is the nondimensional radius \((r/R)\). It is also worth noting that the integration performed on equations 7a and 7b are bounded by the limits \( \xi = \xi_0 \) (nondimensional hub radius) to \( \xi \) [1]. For this project, a trapezoidal integration was numerically carried out from the hub to the tip radius.

Integrating the momentum theory with blade element geometry gives a direct relationship between the displacement velocity ratio and the airfoil’s chord length\((c)\). Isolating each airfoil and observing the circulation resulting in lift,

\[
W_c = \frac{4\pi \lambda GV \xi}{C_l B}
\]

(9)

\[
W = \frac{V(1+\alpha)}{\sin \phi}
\]

(10)

where \( B \) is the number of propeller blades specified, \( W \) is the local total velocity, \( C_l \) is the lift coefficient at the airfoil section, and \( \lambda \) is speed ratio \((V/\Omega R)\). An important fact to note is that the local Reynolds number \((R_e)\) can be found by dividing equation 9 by the kinematic viscosity of air.

With the local Reynolds number provided, and depending on the airfoil, an angle of attack \((\alpha)\) is picked that matches the lift coefficient. This UDP scheme uses one \( C_l \) to design each airfoil that makes up the propeller blade [1]. Finally, the angle of twist \((\beta)\) of each airfoil section will be:

\[
\beta = \alpha + \phi
\]

(11)

B. Iterative Design Procedure

The first design procedure used lift coefficient, number of blades, freestream velocity, hub radius, propeller tip radius, and power delivered to the propeller as input parameters for the scheme. The mathematics was then modified to accept nondimensional inputs like power coefficient, speed ratio, global Reynolds number across the propeller, and drag coefficient.

The following steps describe the initial procedure used in the UDP scheme:

1. Initiate the design phase by guessing a value of \( \zeta \) (zero usually works)
2. At each radial coordinate, find the Prandtl momentum loss factor \((F)\) and flow angle\((\phi)\) by using the equations below

\[
F = \frac{2}{\pi} \arccos e^{-f}
\]

(12)

\[
f = \left(\frac{\rho}{2}\right) (1 - \xi)/\sin \phi_t
\]

(13)

\[
\tan \phi_t = \lambda \left(1 + \frac{\xi}{2}\right)
\]

(14)

\[
\tan \phi = \frac{\tan \phi_t}{\xi}
\]

(15)

where \( \phi_t \) is the flow angle at the tip of the airfoil.

3. Using equation 9, solve for \( W_c \) and obtain the Reynolds number by dividing \( W_c \) by kinematic viscosity of air.
4. As an initial test, the scheme was used to obtain values of airfoil chord lengths and angles of twist that matched those in the Adkins and Liebeck paper [1] which are shown in the Results section of this paper. After running step 3 at various radial points for the input parameters in the paper, the \( R_e \) was at or above 500,000 at each position. This UDP uses NACA 4415 airfoils to build the propeller blades because that airfoil is the most used for propellers. Therefore, the drag coefficient \((C_d)\) versus \( C_l \) as well as the \( C_l \) versus alpha data points for the NACA 4415 airfoil were obtained from airfoiltools.com [2] at a
Reynolds number of 500,000. The graphs of \(R_e > 500,000\) all seemed to overlap with those of 500,000, so the data points for both plots at \(RN = 500,000\) were downloaded off the website, and interpolated to generate formulas and corresponding R squared values for \(C_d\) and \(\alpha\) as functions of \(C_l\) as shown below.

\[
\alpha = 16.633C_l^4 - 40.930C_l^3 + 30.829C_l^2 + 2.632C_l - 3.698
\]  

R-squared value = 0.988

\[
C_d = -7.464C_l^{11} + 69.134C_l^{10} - 274.868C_l^9 + 616.215C_l^8 - 859.560C_l^7 + 776.03C_l^6 - 457.018C_l^5 + 172.750C_l^4 - 40.224C_l^3 + 5.35000C_l^2 - 0.35500C_l + 0.017
\]

R-squared value = 0.901

Fig 2a: \(C_d\) vs \(C_l\) for NACA4415  
Fig 2b: Regression Analysis for \(C_d\) vs \(C_l\)

Fig 3a: \(\alpha\) vs \(C_l\) for NACA4415  
Fig 3b: Regression Analysis for \(\alpha\) vs \(C_l\)
Figures 2a, 2b, 3a and 3b show the airfoil data plotted alongside the interpolated formulas as well as a regression analysis done on each formula. Equations 16 and 17 were used to compute the corresponding \( \alpha \) and \( C_d \) values based on the \( C_l \) value inputted.

5. Use the newly found \( \epsilon \) with equations 4 to find axial interference factor followed by \( W \) from equation 10.
6. Obtain the chord width by dividing equation 9 by equation 10, and the angle of twist from equation 11.
7. Repeat steps 2-6 for each airfoil
8. Integrate equations 7a and 7b from hub to tip radius, and use the solution to find the value of \( \zeta \).
9. If this value of \( \zeta \) is not within 0.1\% of the previous value, iterate though steps 2-8 using the new value of \( \zeta \) until error criteria is met.

There were three major problems that arose while using this procedure. Firstly, The \( C_d \) and \( \alpha \) functions are only applicable at local Reynolds numbers between 500,000 – 1,000,000 making the UDP inaccurate for slower aircrafts with local Reynolds numbers under 500,000 and much faster aircrafts with Reynolds numbers greater than 1,000,000. Also, the initial procedure restricted a user’s ability to observe the propeller change as the angle of attack \( \alpha \) and \( C_d \) are varied independently of the \( C_l \). Finally, the polynomial \( C_d \) function expressed in equation 17 has an absurd order of 11 that only gets an R-squared value of 0.901.

To mitigate against these problems, certain changes were implemented in the design procedure. The first was to remove step 4 and create four new input parameters: \( C_d \), \( \alpha \), \( P_c \) and global Reynolds number \( (Re_g) \). \( Re_g \) is mathematically defined as

\[
Re_g = \frac{VR}{\nu} \tag{18}
\]

Furthermore, step 5 was changed to solve for \( \frac{WC}{V} \) and \( \frac{W}{V} \) from modifying equations 9 and 10 respectively.

III. Generating Propeller Geometry Using EGADS

The Engineering Geometry Aircraft Design System or EGADS, embedded in ESP, enables the creation of UDPs. For this project, the programming was done entirely in C. EGADS allows for a “bottom up” geometry generation or vice-versa or in-between (combination of “bottom up” or “top down”) \(^3\)\(^4\).

Typically, the bottom-up topological entities go as follows:

Node \rightarrow Edge \rightarrow Loop \rightarrow Face \rightarrow Shell \rightarrow Body \rightarrow Model

A Node is a point in 3D space. An Edge is a connection between two nodes. Loops are ordered collections of Edges. A Face is collection of one or more closed Loop, and a Shell is a set of properly oriented Faces. A Body is a collection of grouped topological entities and can be split into four types: WireBody, FaceBody, SheetBody, SolidBody. A WireBody contains a single open or closed Loop. When extruded the WireBody can become a SheetBody. A FaceBody has a single Face and can be revolved, extruded, or blended with other Facebody(s) to form a SolidBody. A SheetBody has one or more open or closed Shells while a SolidBody only consists of one or more closed shells. Models can contain any number of Bodies.\(^4\)

To build the propeller SolidBody, the first thing created is an airfoil. The propeller UDP starts off by generating three nodes which are used to make a singular airfoil as shown in Fig 4a and b.
Three edges are made to represent the lower, upper airfoil edges and trailing edge thickness from the three nodes.

![Fig 5: The three NACA 4415 airfoil edges](image)

An airfoil is then built up to a face and duplicated 12 times. The thirteen airfoils are then scaled, rotated, and translated along the $z$-axis to their equally spaced radial coordinates. They were then stacked along their quarter chord at their radial stations.

![Fig 6: An airfoil face](image)

A 3-D transformation matrix is used to perform the scaling and rotation operations however, a fourth column is added to incorporate the translation operation. For the scaling and rotation about the $z$ axis, the chord length and twist angle results at the radial coordinates $(c(r), \beta(r))$ from the iterative design procedure are used.

$$
\begin{bmatrix}
  c(r) \cos(90 - \beta(r)) & -c(r) \sin(90 - \beta(r)) & 0 & -\frac{c(r) \cos(90 - \beta(r))}{4} \\
  c(r) \sin(90 - \beta(r)) & c(r) \cos(90 - \beta(r)) & 0 & -\frac{c(r) \sin(90 - \beta(r))}{4} \\
  0 & 0 & c(r) & r
\end{bmatrix}
$$

This 3x4 homogenous matrix above performs a rotation $(90 - \beta)$ along the $z$-axis, scales the airfoil by the chord width, and translates each airfoil to its designated radial coordinate along the $z$ axis. The homogenous nature allows each airfoil to retain its stipulated size as it is translated along the computer screen.

![Fig 7: Series of airfoils stacked at their radial stations and along their quarter-chord](image)

Including a circular protrusion to the blade was necessary in ensuring it intersected properly with other blades and with the propeller hub. The circular protrusion is made up of two circles, one at the origin and the other halfway between it and the first airfoil. The EG_blend function then blends a surface from the first circle to the last airfoil which creates the blade with the cylindrical protrusion as shown in figure 8 below.
Fig 8: One propeller blade with the airfoils and circle blended.

The next part of the geometry generation is the replicating the number of blades depending on the B value. To achieve this, a copy of a finished blade is made and spaced $\theta$ degrees about the x-axis with a transformation matrix $^5[6]$. Mathematically,

$$\theta = \frac{360(B-1)}{B}$$

(19)

Lastly, the propeller hub body which includes a spinner and shaft is made in a similar topological process as the propeller blades based on the idea that a spinner is a revolved parabola, and a shaft is a cylinder. Normally, the goal of a UDP is to generate a singular body, but the fuse function which is used to merge the blades and hub bodies together creates a model. Since the result should be a body therefore, the “EG_getTopology” function is used to step the model down to a body $^{[4]}$.

ESP also incorporates the Open Source Constructive Solid Modeler (OpenCSM) to provide a pathway of describing the CAD system’s modelling operations. Such operations include geometric creations, calculations, calling and manipulating a UDP.$^{[3][4]}$ OpenCSM provides ESP a significant advantage over most CAD systems with its simple and readable ASCII description (.csm) file which can be easily created and modified by a user.$^{[3][4]}$ A sample csm file of calling the propeller UDP is shown in a case discussed in the Results section of the paper.

IV. Results

A. Adkins and Leibeck Propeller Case

The first part of this project was to obtain values of airfoil chord lengths and angles of twist that matched those in the Adkins and Liebeck paper. The same input parameters in Adkins and Liebeck’s paper $^{[1]}$: Power = 70 horsepower, B = 2, hub diameter = 1ft, tip diameter = 5.75ft, $C_I = 0.7$, $V = 110$ mph, rpm = 2400 and airfoil = NACA4415, were used in the propeller UDP scheme to test the accuracy of the chord lengths and twist angles.

<table>
<thead>
<tr>
<th>UDP Propeller Scheme</th>
<th>Adkins &amp; Liebeck’s Paper</th>
<th>% error in c</th>
<th>% error in $\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>c</td>
<td>$\beta$</td>
<td>c</td>
</tr>
<tr>
<td>0.500</td>
<td>0.3574</td>
<td>58.1319</td>
<td>0.3424</td>
</tr>
<tr>
<td>0.8958</td>
<td>0.4817</td>
<td>41.6888</td>
<td>0.4605</td>
</tr>
<tr>
<td>1.2917</td>
<td>0.4470</td>
<td>32.0738</td>
<td>0.4269</td>
</tr>
<tr>
<td>1.6875</td>
<td>0.3739</td>
<td>22.8799</td>
<td>0.3569</td>
</tr>
<tr>
<td>2.0833</td>
<td>0.2929</td>
<td>18.8719</td>
<td>0.2796</td>
</tr>
<tr>
<td>2.4792</td>
<td>0.2004</td>
<td>16.0263</td>
<td>0.1913</td>
</tr>
</tbody>
</table>

Table 1: Comparing UDP Propeller values with Adkins & Liebeck Paper

Fig 9: The 2 bladed propeller formed from parameters described in Adkins & Liebeck’s paper
B. F1D Propeller Case

An F1D is a lightweight, slow-flying rubber-powered aircraft that is built to stay fly in an indoor space \(^7\). An F1D propeller case was tested in ESP with the propeller UDP scheme, and the results were compared with the real-life propeller as a way of showing the UDP scheme’s efficacy.

The following inputs were taken plugged into the ESP using the UDP Scheme:

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Blades</td>
<td>2</td>
</tr>
<tr>
<td>R</td>
<td>0.3 meters</td>
</tr>
<tr>
<td>Hub Radius</td>
<td>0.03 meters</td>
</tr>
<tr>
<td>Power Coefficient (Pc)</td>
<td>0.068</td>
</tr>
<tr>
<td>λ</td>
<td>0.6</td>
</tr>
<tr>
<td>V</td>
<td>0.7 meters/seconds</td>
</tr>
<tr>
<td>Re</td>
<td>14500</td>
</tr>
<tr>
<td>C_l</td>
<td>0.5</td>
</tr>
<tr>
<td>C_d</td>
<td>0.062</td>
</tr>
<tr>
<td>α</td>
<td>0.5 degrees</td>
</tr>
</tbody>
</table>

Table 2: Input Parameters

The .csm script below was written to call the propeller UDP and generate the FID configuration

```
# parameters from F1D
UDPARG  Prop  nblade  2
UDPARG  Prop  cpower 0.0680
UDPARG  Prop  lambda 0.6000
UDPARG  Prop  reyr  1.45e+4
UDPARG  Prop  rtip  0.300  # m
UDPARG  Prop  rhub  0.030  # m
UDPARG  Prop  clift  0.500
UDPARG  Prop  cdrag  0.062
UDPARG  Prop  alfa  0.500  # deg

# shaft/spinner parameters
UDPARG  Prop  shdiam  -0.002
UDPARG  Prop  shxmin  -.005
UDPARG  Prop  shxmax  0.005

UDPRIM  Prop
```

Fig 10: F1D propeller built in ESP  
Fig 11: Real life F1D propeller \(^7\)
The shaft/spinner parameters can be altered in a way that results in the addition or removal of a shaft and spinner. Suppose \textbf{shdiam} is less than zero, the UDP interprets this as to skip the process of making neither shaft nor spinner will appear as seen in Figure 10. Conversely, a \textbf{shdiam} with a value greater than zero will generate a shaft.

![Fig 12: A 2-bladed propeller with a shaft and spinner](image1)

![Fig 13: A 3-bladed propeller](image2)

V. Conclusion

The propeller UDP accepts certain nondimensional input parameters from the users such as number of blades, lift coefficient, power coefficient and advance ratio, and automatically generates propellers based on the math discussed in Adkins & Liebeck’s paper[1]. The values of the chord and the angle of twist made with the UDP scheme were under 4.8% and 2.7% error (respectively) when compared to the results in the Adkins paper. Similarly, the FID propeller modelled with the UDP in ESP matched the real-life FID propeller design as shown in figures 10 and 11, respectively. The goal is to have the UDP completed, and fully prepackaged with ESP by the end of the first quarter of 2021.

Acknowledgements

I would like to express my very deep appreciation to Dr. John F. Dannenhoffer III, my research advisor, for his guidance, encouragement, and valuable critiques of this research project. His eagerness to generously lend his time has been much appreciated.

References


