Article

Not peer-reviewed version

Algorithm for Propeller Optimization Based on Differential Evolution

Andry Sedelnikov * , Evgenii Kurkin , Jose Gabriel Quijada-Pioquinto * , Oleg Lukyanov , Dmitriy Nazarov , Vladislava Chertykovtseva , Ekaterina Kurkina , Van Hung Hoang

Posted Date: 10 November 2023

doi: 10.20944/preprints202311.0706.v1

Keywords: differential evolution; penalty function; SHADE algorithm; CAPR; lightweight pipelining; isolated sections method

Article

Algorithm for Propeller Optimization Based on Differential Evolution

Andry Sedelnikov, Evgenii Kurkin, Jose Gabriel Quijada-Pioquinto *, Oleg Lukyanov, Dmitriy Nazarov, Vladislava Chertykovtseva, Ekaterina Kurkina and Van Hung Hoang

Institute of Aerospace Engineering, Samara National Research University, 34 Moskovskoe Shosse, Samara 443086, Russia; axe_backdraft@inbox.ru, eugene.kurkin@mail.ru, hosekihada@yandex.ru, lukyanov.oe@ssau.ru, dvn69@mail.ru, vladislaava.s@yandex.ru, ekaterina.kurkina@mail.ru, hunghoang2508@gmail.com

* Correspondence: hosekihada@yandex.ru

Abstract: This article describes the choice of an optimization algorithm for solving problems and the optimal design of an unmanned aerial vehicle propeller. To solve the problem using evolutionary algorithms, it was transformed into an unconstrained optimization problem using a penalty function. The airfoil contours were constructed using a Bezier curve. Design variables were divided into two types: those that describe in general terms the propeller geometry or operation, such as propeller diameter, number of blades, and rev/min; and those that vary with propeller radius, such as chord length, effective airfoil angle, and airfoil geometry. The objective function is the calculation of the propeller power required to achieve a given thrust. The differential evolution algorithm was used to solve this problem.

Keywords: differential evolution; penalty function; SHADE algorithm; CAPR, lightweight Pipelining; isolated sections method

---

1. Introduction

The air propeller is one of the most important elements of an aircraft that provides lift and propulsion in the air. The relevance of using the propeller as a propulsion system has increased with the current growth in the production of unmanned aerial vehicles for various purposes around the world. An optimized air propeller can significantly reduce emissions, improve acoustic response, and enhance performance [1]. The design of air propellers to minimize propulsion system energy costs began with Zhukovsky, Betz, and Goldstein [2,3] and was then refined by a few scientists over the course of a century [4]. In recent years, computational fluid dynamics using CAE systems has become an important method for propeller design and computation [5,6,7]. Propeller optimization is a complex problem, often multi-objective optimization, requiring consideration of many factors and constraints, including not only aerodynamic issues but also strength, acoustics, etc. [8,9,10,11]. Therefore, the use of the finite volume method becomes irrational because it requires large computational power. The most common goals of propeller optimization from an aerodynamic perspective are the requirements of maximum thrust, maximum efficiency, and a balanced propeller based on Paretto optimality [12]. The paper [13] presents a methodology for inverse propeller blade design. The isolated blade section method is used to directly calculate propeller aerodynamic characteristics during the optimization process. This involved the calculation of the aerodynamic characteristics of the profiles in the blade sections using the xflr code. The obtained characteristics are verified with CFD calculations. The authors of [14] solved a multi-objective optimization problem subject to a set of constraints using a direct search algorithm. The propeller characteristics were calculated from the profiles in the blade sections using the discrete vortex method described in the Xfoil code. The paper [15] describes the experimental verification and application of a method of multi-criteria optimization using genetic algorithms for the design of a propeller for a high-altitude aircraft. The propeller characteristics were calculated from the profiles in the blade sections using vortex methods. The optimization results showed that the desirable trade-off between propeller efficiency and weight is related to power consumption, structural strength, and even manufacturing and installation conditions. The authors [16] developed a procedure to design and fabricate propellers for small unmanned aerial vehicles. The method of sections implemented in QProp is the basis for the calculation of the direct problem of aerodynamics, and the optimization process is managed by the Bearcontrol 9 written in MATLAB, which allows to create a Visual Basic code for automatic construction of 3D model in SolidWorks.

The propeller has a complex three-dimensional geometry, and its optimization problem involves a parametric approach to define the geometric model. Bezier curves and surfaces are one of the most universal and robust approaches for specifying complex shapes [17] and can be used to determine the optimal shape of propellers [18]. This paper is devoted to the development of an efficient algorithm for the parametric optimization of an unmanned aerial vehicle propeller. This paper describes step-by-step the selection of methods of parametrization of geometric model, design calculation of propeller aerodynamics, and the use of metaheuristic algorithms of parametric optimization.

2. Materials and Methods

2.1. Mathematical Model Of Propeller Optimization

The objective of optimization is to determine the geometry and appropriate operating conditions of a propeller, which requires the least operating power, depending on the operating speed of the aircraft and the diameter of the propeller, also subject to a certain thrust. Mathematically, the optimization problem can be expressed as a constrained optimization problem:

$$\begin{aligned} \min \quad & W(x), \\ \text{such that} \quad & T_{min} - T(x) \leq 0, \\ & x \in X \end{aligned}$$

where W(x) is the required power of the propeller; T(x) is the thrust provided by the propeller; T_{min} is the minimum desired thrust; x is the vector of design parameters belonging to the feasible set of solutions X.

In order for this optimization problem to be solved by evolutionary algorithms, the problem had to be converted into an unconstrained optimization problem, this was achieved by using a penalty function, which will be used as a fitness function [16]. The penalty function is expressed as:

$$L(x) = \begin{cases} W(x) & \text{if } \psi(x) = 0 \\ R\psi(x) + U* & \text{if } \psi(x) > 0 \wedge W(x) \leq U* \\ R\psi(x) + W(x) & \text{if } \psi(x) > 0 \wedge W(x) > U* \end{cases} \quad (1)$$

where

$$\psi(x) = \max\{0, T_{min} - T(x)\} \quad (2)$$

where R is a penalty parameter; U* is an upper bound on the constrained global minimum value, which is provided initially by the user. $\psi(x)>0$ only if x is infeasible. Therefore, the optimization problem remains as:

$$\begin{aligned} \min \quad & L(x), \\ \text{such that} \quad & x \in X \end{aligned}$$

2.2. Selection Of Design Variables

Two types of design variables were proposed, those that describe in a general way the geometry of the propeller or the operation, such as the diameter of the propeller (d, in m), number of blades (B) and number of revolutions per minute (n_m, rev/min); and others that change depending on the radius of the propeller, such as the chord length, the effective angle of the airfoil and the geometry of the airfoil. It was considered that this type of variables has a continuous and continuous variation along the blade of the propeller. This variation was achieved by using two quadratic Bezier curves [19], (see Figure 1). A quadratic Bezier curve is the path drawn by the following function:

$$B(t) = (1 - t^2)P_0 + 2(1 - t)tP_1 + t^2P_2, \quad (3)$$

where P_0, P_1 and P_2 are control points, and t is a parameter that always varies from 0 to 1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 November 2023 doi:10.20944/preprints202311.0706.v1

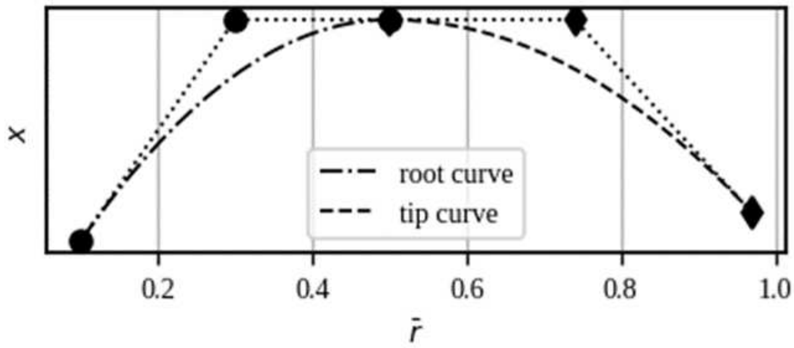


Figure 1. The Bezier curves determine the variation of the design variables.

B(t) can also be decomposed as ($B_{\bar{r}}(t)$, $B_x(t)$), where $\bar{r}$ indicates the relative radius of the propeller and x the design parameter.

$$B_{\bar{r}}(t) = (1-t)^2\bar{r}_0 + 2(1-t)t\bar{r}_1 + t^2\bar{r}_2, \quad (4)$$

$$B_x(t) = (1-t)^2x_0 + 2(1-t)tx_1 + t^2x_2, \quad (5)$$

The control points that determine the Bézier curves for each design variable as a function of the relative radius of the propeller are:

- Root curve

$$\begin{cases} \bar{r}_0^r = 0.1 \\ \bar{r}_1^r = 0.5\bar{r}_{xm} + 0.05 \\ \bar{r}_2^r = \bar{r}_{xm} \end{cases} \qquad \begin{cases} x_0^r = x_r \\ x_1^r = x_m \\ x_2^r = x_m \end{cases} \quad (6)$$

- Tip curve

$$\begin{cases} \bar{r}_0^t = \bar{r}_{xm} \\ \bar{r}_1^t = 0.5\bar{r}_{xm} + 0.485 \\ \bar{r}_2^t = 0.97 \end{cases} \qquad \begin{cases} x_0^t = x_m \\ x_1^t = x_m \\ x_2^t = x_t \end{cases} \quad (7)$$

In Equations 4 to 7, x may be the chord relative to the propeller diameter (c/d), the effective angle (α), maximum thickness (y_t), position of the maximum thickness relative to the chord (x_t), the maximum camber (y_c), and the position of the maximum camber relative to the chord (x_c). Each of these variables is with respect to the profile of each section of the propeller blade. The variable $\bar{r}_{xm}$ refers to the union position of the two Bezier curves. The subscripts r, m, t refers to the values of the design variable with respect to 10% of the blade radius, the union point of the Bezier curves, and 97% of the blade radius respectively. Therefore, each vector of design variables is formed as x_i = [$(c/d)_r$, $(c/d)_m$, $(c/d)_t$, $\bar{r}_{cm}$, α_r, α_m, α_t, $\bar{r}_{\alpha m}$, y_{tr}, y_{tm}, y_{tt}, $\bar{r}_{ytm}$, x_{tr}, x_{tm}, x_{tt}, $\bar{r}_{xtm}$, y_{cr}, y_{cm}, y_{ct}, $\bar{r}_{ycm}$, x_{cr}, x_{cm}, x_{ct}, $\bar{r}_{xcm}$, n_m, B, d$]_i$.

2.3. Model For The Resolution Of The Objective Function

The objective function of the optimization process is the calculation of the required propeller power (W, in W) to achieve a thrust (T, in N). A modified version of the Isolated Sections Method (ISM) [20] was used to obtain these values. The variation between the original and our modified version of the ISM is to replace the geometric twisting of the sections (φ, in deg) by the effective angle of attack of the sections (α, in deg) as the input value.

The input variables required by the method are the following: B, V_∞, d, n(n, rev/s), and physical characteristics of the air (density (ρ, in kg/m^3), kinematic viscosity (ν, [m^2/s]), sound speed (a, [m/s])). ISM requires sectioning one of the propeller blades into an infinite number of sections (NS). In each section it is necessary to know the following values: chord length (c, in m), $\bar{r}$, and the geometry of the airfoil ((X_U, Y_U), (X_L, Y_L), each coordinate is in m).

To obtain the coordinates of the profile in each section, an airfoil parameterization method based on the Bezier-PARSEC technique was proposed [21]. Each airfoil is constructed with 4 cubic Bezier curves, two curves for determining the thickness of the airfoil and two curves for determining the camber of the airfoil (see Figure 2).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 November 2023 doi:10.20944/preprints202311.0706.v1

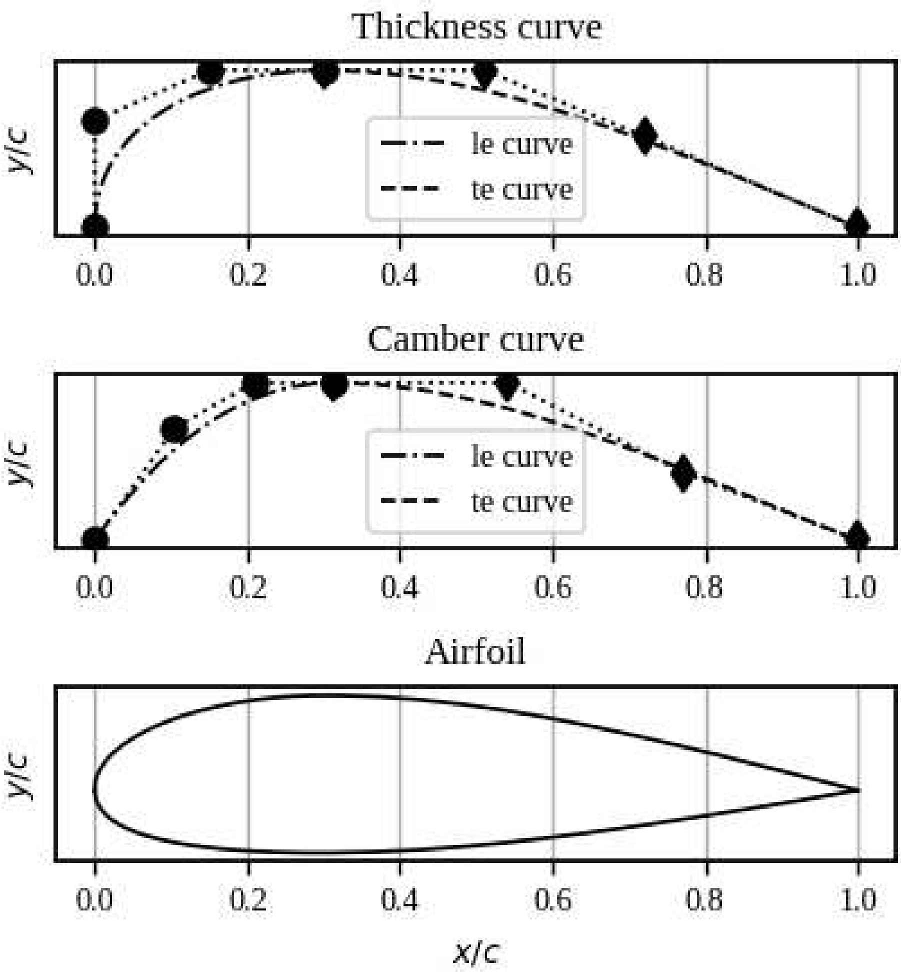


Figure 2. Airfoil construction using cubic Bezier curves.

The parametric functions for determining each cubic Bezier curve are:

$$B_x(t) = (1-t)^3 x_0 + 3(1-t)^2 t x_1 + 3(1-t) t^2 x_2 + t^3 x_3, \quad (8)$$

$$B_y(t) = (1-t)^3 y_0 + 3(1-t)^2 t y_1 + 3(1-t) t^2 y_2 + t^3 y_3, \quad (9)$$

The construction of the profile is carried out using the following equations:

$$X_t = B_{xt}^{le}(t) + B_{xt}^{te}(t) \quad (10)$$

$$Y_t = B_{yt}^{le}(t) + B_{yt}^{te}(t) \quad (11)$$

$$X_c = B_{xc}^{le}(t) + B_{xc}^{te}(t) \quad (12)$$

$$Y_c = B_{yc}^{le}(t) + B_{yc}^{te}(t) \quad (13)$$

$$\theta = \tan^{-1}\left({}^{dY_c}/_{dX_c}\right) \quad (14)$$

The control points for the leading edge thickness curve are defined by:

$$\begin{cases} x_0^{le} = 0 \\ x_1^{le} = 0 \\ x_2^{le} = 0.5x_t \\ x_3^{le} = x_t \end{cases} \qquad \begin{cases} y_0^{le} = 0 \\ y_1^{le} = 0.34y_t \\ y_2^{le} = 0.5y_t \\ y_3^{le} = 0.5y_t \end{cases} \quad (15)$$

The control points for the trailing edge thickness curve are defined by:

$$\begin{cases} x_0^{te} = x_t \\ x_1^{te} = 0.3 + 0.7x_t \\ x_2^{te} = 0.6 + 0.4x_t \\ x_3^{te} = 1 \end{cases} \qquad \begin{cases} y_0^{te} = 0.5y_t \\ y_1^{te} = 0.5y_t \\ y_2^{te} = 0.29y_t \\ y_3^{te} = 0 \end{cases} \tag{16}$$

The control points for the leading edge camber curve are defined by:

$$\begin{cases} x_0^{le} = 0 \\ x_1^{le} = x_c/3 \\ x_2^{le} = 2x_c/3 \\ x_3^{le} = x_c \end{cases} \qquad \begin{cases} y_0^{le} = 0 \\ y_1^{le} = 0.71y_c \\ y_2^{le} = y_c \\ y_3^{le} = y_c \end{cases} \tag{17}$$

And the control points for the trailing edge camber curve are defined by:

$$\begin{cases} x_0^{te} = x_c \\ x_1^{te} = (1 + 2x_c)/3 \\ x_2^{te} = (2 + x_c)/3 \\ x_3^{te} = 1 \end{cases} \qquad \begin{cases} y_0^{te} = y_c \\ y_1^{te} = y_c \\ y_2^{te} = 0.43y_c \\ y_3^{te} = 0 \end{cases} \tag{18}$$

The points for determining the upper curve of the airfoil are determined by:

$$X_U = X_c - Y_t \sin\theta \tag{19}$$

$$Y_U = Y_c + Y_t \cos\theta \tag{20}$$

And the points for determining the lower curve of the airfoil are determined by:

$$X_L = X_c + Y_t \sin\theta \tag{21}$$

$$Y_L = Y_c - Y_t \cos\theta \tag{22}$$

All the distribution curves of the design variables are generated by Algorithm 1.

Algorithm 1: Subroutine for creation of the Bezier curves and airfoils

Inputs: x_i, d, $\bar{r}$, NS;

Outputs: $(c/d)_i$, α_i, x_{ti}, y_{ti}, x_{ci}, y_{ci}, X_U, Y_U, X_L, Y_L;

Create the distribution curve for $(c/d)_i$ as a function of the $\bar{r}_i$ of the blade with (4), (5), (6), (7);

Create the distribution curve for α_i as a function of the $\bar{r}_i$ of the blade with (4), (5), (6), (7);

Create the distribution curve for x_{ti} as a function of the $\bar{r}_i$of the blade with (4), (5), (6), (7);

Create the distribution curve for y_{ti} as a function of the $\bar{r}_i$ of the blade with (4), (5), (6), (7);

Create the distribution curve for x_{ci} as a function of the $\bar{r}_i$ of the blade with (4), (5), (6), (7);

Create the distribution curve for y_{ci} as a function of the $\bar{r}_i$ of the blade with (4), (5), (6), (7);

// Get the airfoil in each section of the blade

for s = 1 **to** NS**do**

  Get $x_t(\bar{r}_{s,i})$, $y_t(\bar{r}_{s,i})$, $x_c(\bar{r}_{s,i})$ and $y_c(\bar{r}_{s,i})$;

  Get $X_{ts,i}$ and $Y_{ts,i}$ with (8), (9), (10), (11), (15), (16);

  Get $X_{cs,i}$ and $Y_{cs,i}$ with (8), (9), (12), (13), (17), (18);

  Get $\theta_{s,i}$ with (14);

  Create and save the points of the sth-airfoil X_U, Y_U, X_L, Y_L with (19), (20), (21), (22);

The iterative method to obtain the thrust provided by the propeller and the required power is shown in [20]. This method was modified to use the effective angle of attack (α) of the profile in each section as a constant and the geometric twist (φ) of the section as a variable that is updated in each iteration.

To obtain the aerodynamic coefficients of each section of the blade, it was obtained by using the Xfoil program, which shows a good performance for the evaluation of profiles (see Figure 3).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 November 2023 doi:10.20944/preprints202311.0706.v1

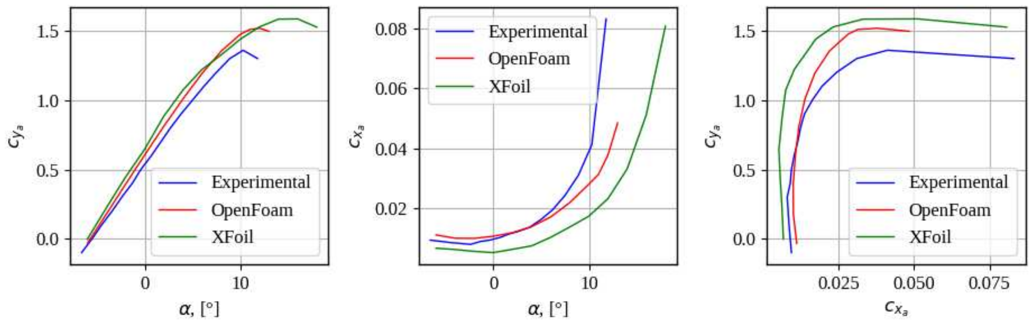


Figure 3. Aerodynamic coefficients of the CLARK Y profile by different methods.

In Algorithm 2 the modified ISM is described in pseudo code. To simplify the ISM calculations, it is necessary to obtain the radius (R, [m]), the angular velocity (ω, [rad/s]) and the relative velocity ($\bar{v}$, dimensionless) of the propeller.

$$\omega = 2\pi n \tag{23}$$

$$\bar{v} = \frac{V_\infty}{\omega R} \tag{24}$$

The sections are defined by the $\bar{r}$ array. For example, in the cases that we evaluate, the sections begin to be numbered from 10% of the R of the propeller and end at 97% of R ($\bar{r}$ = [0.1, ..., 0.97]). In our modified ISM, having the α of each section as input, it is necessary to calculate in the first instance the aerodynamic coefficients of each section delimited by $\bar{r}$. For this it is necessary to obtain the local Reynolds (Re) and Mach (M) numbers.

To determine the aerodynamic characteristics of the propeller to reduce the number of calculations, interpolation of the profile data was performed over a smaller number of sections, in each of which the characteristics of the angle of attack were calculated considering the local values of the Reynolds and Mach numbers (see Figure 4).

A study was conducted out on the required number of sections, which showed that the use of 15 sections to calculate the aerodynamics of the profile and 75 sections to integrate thrust and moment allows one to quickly obtain accurate values of the propeller characteristics.

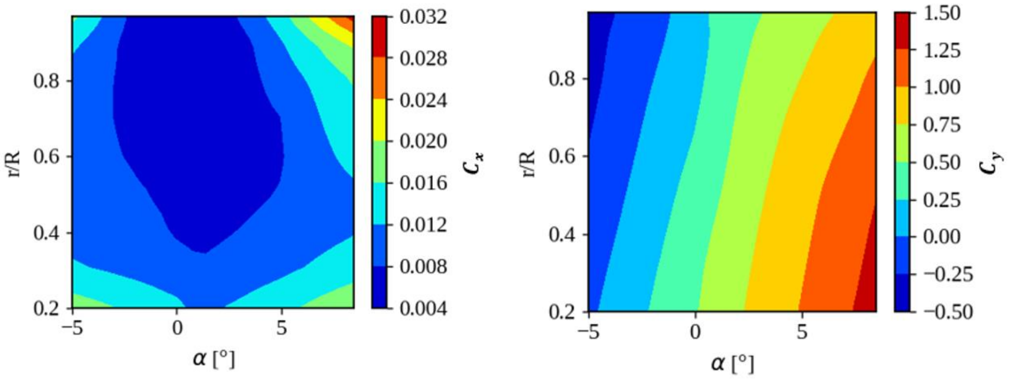


Figure 4. Calculation and interpolation of aerodynamic characteristics of the profile by propeller span.

Having the local characteristics of the flow, in addition to the chord, geometry and angle of attack of the airfoil, coefficients of lift (c_l) and drag (c_d) coefficients in each section are calculated. The next step is to get $\bar{U}_1$, $\bar{V}_1$ and $\bar{\Gamma}_r$ by an iterative process, which are necessary to get the coefficients ct and mk. The equations involved in this process are shown below:

$$\bar{v}_1 = -\frac{\bar{v}}{2} + \sqrt{\frac{\bar{v}^2}{4} + \bar{u}_1(\bar{r} - \bar{u}_1) + 2\int_{\bar{r}}^{1} \frac{\bar{u}_1^2}{\bar{r}} d\bar{r}} \tag{25}$$

$$\bar{\mathrm{U}}_1 = \bar{\mathrm{r}} - \bar{\mathrm{u}}_1 \tag{26}$$

$$\bar{\mathrm{V}}_1 = \bar{\mathrm{v}} + \bar{\mathrm{v}}_1 \tag{27}$$

$$\bar{\mathrm{W}}_1 = \sqrt{\bar{\mathrm{V}}_1^2 + \bar{\mathrm{U}}_1^2} \tag{28}$$

$$\beta_1 = \tan^{-1}\left(\frac{\bar{\mathrm{V}}_1}{\bar{\mathrm{U}}_1}\right) \tag{29}$$

$$\sigma = \frac{\mathrm{Bc}}{\mathrm{R}\pi} \tag{30}$$

$$\bar{\Gamma}_\mathrm{r} = \frac{1}{8}\sigma c_l \bar{\mathrm{W}}_1 \tag{31}$$

$$\mathrm{f_r} = \frac{2}{\pi}\cos^{-1}\left(e^{-\frac{0.5\mathrm{B}(1-\bar{\mathrm{r}})}{\bar{\mathrm{r}}\sin(\beta_1)}}\right) \tag{32}$$

previously before initializing the iterative process, it is important to initialize the n values of $\bar{\mathrm{u}}_1$ and $\int_{\bar{r}}^{1}\frac{\bar{u}_1^2}{\bar{r}}d\bar{r}$equal to zero. At the end of each iteration $\bar{\mathrm{u}}_1$ has to be updated making use by:

$$\bar{\mathrm{u}}_1 = \frac{\bar{\Gamma}_\mathrm{r}}{\mathrm{f_r}\bar{\mathrm{r}}} \tag{33}$$

while the values of $\int_{\bar{r}}^{1}\frac{\bar{u}_1^2}{\bar{r}}d\bar{r}$ are updated as shown in lines 24, 25 and 26 of Algorithm 2, where trapz(Y, X) integrates along the given axis using the compound trapezoidal rule, Y is the input array to integrate and X is the sample points corresponding to the Y values.

It has been proven that the loop described between lines 10 and 26 of Algorithm 2 only needs 10 cycles to obtain good results.

To obtain the coefficients c_t and m_k, it is first necessary to calculate their differentials in each section, and then integrate with respect to $\bar{r}$, making use of trapz().

$$\mathrm{dc_t} = 8\bar{\Gamma}_\mathrm{r}(\bar{\mathrm{U}}_1 - \mathrm{K}^{-1}\bar{\mathrm{V}}_1) \tag{34}$$

$$\mathrm{dm_k} = 8\bar{\Gamma}_\mathrm{r}(\bar{\mathrm{V}}_1 + \mathrm{K}^{-1}\bar{\mathrm{U}}_1)\bar{\mathrm{r}} \tag{35}$$

By obtaining the coefficients c_t and m_k, the thrust provided by the propeller (T) and the required power of the propeller (W) can be calculated.

$$\mathrm{T} = 0.5\mathrm{c_t}\rho(\omega\mathrm{R})^2\pi\mathrm{R}^2 \tag{36}$$

$$\mathrm{W} = 0.5\mathrm{m_k}\rho(\omega\mathrm{R})^3\pi\mathrm{R}^2 \tag{37}$$

Other propeller performance metrics that can be obtained with this method are the dynamic efficiency η_d and the static efficiency η_s of the propeller.

$$\alpha_\mathrm{p} = \frac{\mathrm{T}}{\rho \mathrm{n}^2\mathrm{d}^4} \tag{38}$$

$$\beta_\mathrm{p} = \frac{\mathrm{W}}{\rho \mathrm{n}^3\mathrm{d}^5} \tag{39}$$

$$\lambda_\mathrm{p} = \frac{\mathrm{V}_\infty}{\mathrm{nd}} \tag{40}$$

$$\eta_\mathrm{d} = \frac{\alpha_\mathrm{p}\lambda_\mathrm{p}}{\beta_\mathrm{p}} \tag{41}$$

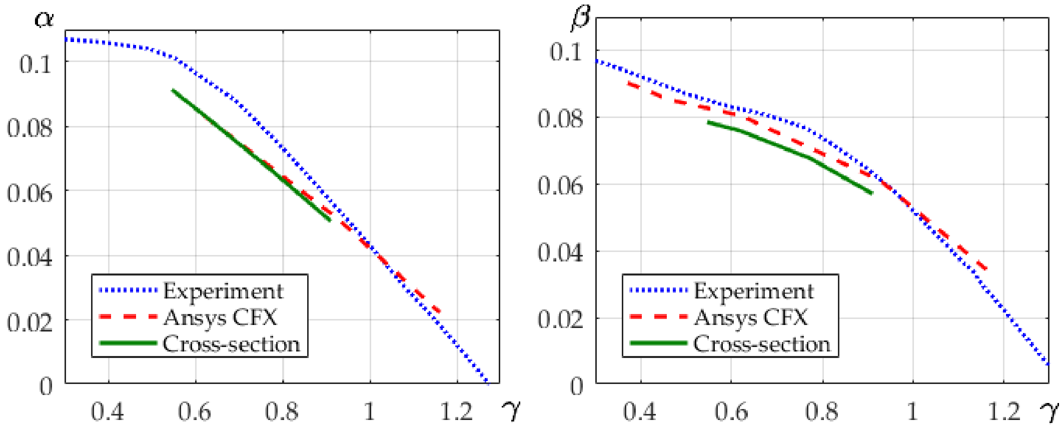
$$\eta_\mathrm{s} = \frac{\mathrm{c_t}^{3/2}}{2\mathrm{m_k}} \tag{42}$$

Algorithm 2: Modified ISM

Input $n, B, V_\infty, d, \rho, \nu, a, c, \bar{r}$, [$X_U$, Y_U, X_L, Y_L], NS;
$R = d/2$;
Get ω with (33);

Get $\bar{v}$ with (34);
for s = 1 **to** NS **do**
  $r_s = \bar{r}_s R$;
  Get Re_s and M_s in each section;
  c_{ls}, c_{ds} = runXFOIL(α_s, $[X_U, Y_U, X_L, Y_L]_s$, Re_s, M_s);
  $0 \rightarrow \bar{u}_{1_s}$, $0 \rightarrow \left(\int_{\bar{r}}^{1} \frac{\bar{u}_1^2}{\bar{r}} d\bar{r}\right)_s$
for l = 1 **to** 10 **do**
  $I_u = \varnothing$;;
  for s = 1 **to** NS **do**
    Get $\bar{v}_{1_s}$ with (25);
    Get $\bar{U}_{1_s}$ with (26);
    Get $\bar{V}_{1_s}$ with (27);
    Get $\bar{W}_{1_s}$ with (28);
    Get β1s with (29);
    $\Phi_s = \alpha_s + \beta_{1s}$;
    $K_s = c_{ls}/c_{ds}$;
    Get σ_s with (30);
    Get $\bar{\Gamma}_{r_s}$ with (31);
    Get f_{rs} with (32);
    Update $\bar{u}_{1_s}$ with (33);
    $\left(\frac{\bar{u}_{1_s}^2}{\bar{r}_s}\right)_s \rightarrow I_{u_s}$;
  fors = 1 **to** NS **do**
    Get $\left(\int_{\bar{r}}^{1} \frac{\bar{u}_1^2}{\bar{r}} d\bar{r}\right)_s$ with $trapz(I_u$[s:**end**], $\bar{r}$[s:**end**]);
fors = 1 **to** NS **do**
  Get dc_{t_s} with (34);
  Get dm_{k_s} with (35);
$c_t = trapz(dc_t, \bar{r})$;
$m_k = trapz(dm_k, \bar{r})$;
Get α_p, β_p and λ_p with (38), (39) and (40) respectively;
Get outputs T, W, η_d, and η_s with (36), (37), (41) and (42) respectively;

Verification of the proposed method of propeller calculation has been carried out by comparing it with the experimental data of the NACA 5868-9 propeller, (experimental results, geometric and kinematic characteristics of the propeller are given in [22]), as well as by comparison with the results of numerical mathematical modeling by solving the Navier-Stokes equations in CFX software. The results of comparison in terms of thrust coefficient and power are shown in Figure 5.



(a) (b)

Figure 5. Comparison by thrust coefficient (a) and power coefficient (b).

2.4. Optimization Algorithm

The optimization algorithm used to solve the task is based on the differential evolution (DE) algorithm. The proposed algorithm contemplates the use of self-adaptive schemes of the evolutionary operators, methods of population size reduction (PSR), sampling techniques for selection of individuals from the initial population and stopping conditions that adapt to the needs of the optimization process. In addition, the developed algorithm contemplates the use of parallel computing strategies to accelerate the calculation of the values of the objective function.

DE is a stochastic, population-based algorithm developed for real-valued function optimization problems. It operates by having a population of individuals (x vector) that move around in the search space by recombining through crossover and mutation with other existing individuals in the population. Through a selection process, a newly generated individual is accepted as part of the population if the new individual x is an improvement; otherwise, it is discarded. This iteration process is repeated to find a vector x that optimizes a function f(x) [23]. As with other evolutionary algorithms, the search performance of DE algorithms depends on control parameter settings. A standard DE has three main control parameters, which are the population size, scaling factor F, and crossover factor CR. However, it is well-known that the optimal settings of these parameters are problem dependent. Therefore, when applying DE to a real-world problem, it is often necessary to tune the control parameters to obtain the desired results. In practical cases, many researchers suggest the use of self-adaptive schemes to adjust the online control parameters during the search process. One of the variants of DE that apply this type of schemes is the success-history based adaptation for DE (SHADE) [24].

SHADE uses a historical memory MCR, MF which stores a set of CR, F values that have performed well in the past, and generates new CR, F pairs by directly sampling the parameter space close to one of these stored pairs.

Algorithm 3: Memory update algorithm in SHADE

If $S_{CR} \neq \varnothing$ **and** $S_F \neq \varnothing$ **then**

  If $M_{CR,k,g} = -1$ **or** $max(S_{CR}) = 0$ **then**

    $M_{CR,k,g+1} = -1$;

  else

    $M_{CR,k,g+1} = mean_{WL}(S_{CR})$;

  $M_{F,k,g+1} = mean_{WL}(S_F)$;

  k++;

  If $k > H$ **then**, $k = 1$;

else

$M_{CR,k,g+1} = M_{CR,k,g}$;

$M_{F,k,g+1} = M_{F,k,g}$;

In Algorithm 3, index k (1 ≤ k ≤ H) determines the position in the memory to update. In generation g, the k-th element in the memory is updated. At the beginning of the search k is initialized to 1. k is incremented whenever a new element is inserted into the history. If k > H, k is set to 1. In the update algorithm 1, note that when all individuals in generation g fail to generate a trial vector which is better than the parent, i.e., SCR = SF = ∅, the memory is not updated. The weighted Lehmer mean meanWL(S) is computed using the formula below:

$$\mathrm{mean}_{WL}(S) = \frac{\sum_{m=1}^{|S|} w_m S_m^2}{\sum_{m=1}^{|S|} w_m S_m} \tag{43}$$

$$w_m = \frac{\Delta f_m}{\sum_{l=1}^{|S|} \Delta f_l} \tag{44}$$

$$\Delta f_m = \left| f(u_{m,g}) - f(x_{m,g}) \right| \tag{45}$$

The amount of fitness improvement Δfmis used in order to influence the parameter adaptation (S refers to either SCR or SF). As MCR is updated, if MCR,k,g = -1 or max(SCR) = 0 (i.e., all elements of SCR are 0), MCR,k,g+1 is set to -1. Thus, if MCR is assigned the terminal value -1, then MCR will remain fixed at -1 until the end of the search. This has the effect of locking CRi to 0 until the end of the search, causing the algorithm to enforce a "change-one-parameter-at-a-time" policy, which tends to slow down convergence, and is effective on multimodal problems.

The SHADE algorithm has been shown to work well in conjunction with PSR methods. For the development of the present optimization algorithm, it has been decided to incorporate the continuous adaptive population reduction (CAPR) method. The CAPR method gradually reduces the population size according to the change of gradient of the fitness value [25].

$$NP_{g+1} = \begin{cases} \sqrt[\gamma]{\Delta_g/\Delta_{g-1}} & 0 < \Delta_g/\Delta_{g-1} < 1 \\ NP_g & \text{otherwise} \end{cases} \tag{46}$$

$$NP_{g+1} = \begin{cases} NP_{g+1} & NP_{g+1} > NP_{min} \\ NP_{min} & NP_{g+1} \le NP_{min} \end{cases} \tag{47}$$

where

$$\Delta_g = \frac{f_{avg}(x_g) - f_{avg}(x_{g-1})}{f_{avg}(x_g)}, \Delta_{g-1} = \frac{f_{avg}(x_{g-1}) - f_{avg}(x_{g-2})}{f_{avg}(x_{g-1})} \tag{48}$$

In the third generation and in subsequent generations, the evaluated function values of all vectors in the population are averaged to be $f_{avg}(x_g)$. This value, together with that form the previous evaluation generation, is used to calculate the normalized gradient value Δ_g. Δ_{g-1} is calculated in a similar fashion using the previous average function evaluation value, $f_{avg}(x_{g-1})$, and the one before the previous$f_{avg}(x_{g-2})$. If the ratio Δ_g/Δ_{g-1} is within the range of [0, 1], then NP is reduced by a fraction equal to the γ-th root of the ratio Δ_g/Δ_{g-1}. The reason for taking root of the ratio is to slow down the population size reduction rate.

Another criterion to consider when wanting to increase the performance of algorithms based on Differential Evolution is the way to generate the initial population. It has been shown that a population, whose individuals are best distributed throughout the entire design space, has a greater chance of finding a global optimum, in addition to reducing the search time. The LHS design is a statistical method for generating a quasi-random sampling distribution. It is among the most popular sampling techniques in computer experiments thanks to its simplicity and projection properties with high-dimensional problems. LHS is built as follows: each dimensional space, representing a variable, is cut into n sections where n is the number of sampling points, and only one point is placed in each section [26].

For real-case optimization processes, it is common to make use of two types of stopping criteria. The following two types of stopping criteria were considered for this algorithm [27]:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 November 2023 doi:10.20944/preprints202311.0706.v1

- Exhaustion-based criteria: Due to limited computational resources optimization run might be terminated after a certain generation, number of objective function evaluations or CPU time. Commonly, a maximum number of generations or number of objective function evaluations is used in combination with every stopping criterion to prevent the algorithm from running forever if a criterion is not able to stop the run.
- Distribution-based criteria: For DE algorithms all individuals converge to the optimum eventually. Therefore, it can be concluded that convergence is reached when the individuals are close to each other. Because is assumed that the optimum is not known as for the reference criterion, the distances between the population members are examined. This type of criterion can be applied in the design space or in the objective space.

One of the main disadvantages of evolutionary algorithms is that they need to evaluate multiple vectors to find the global optimum, which implies calculating the values of the objective function many times. One of the ways to speed up the calculation process is by using parallel computing strategies. The strategy that was proposed to be used is Lightweight Pipelining (LP) [28]. The pipelining process helps in providing an easy approach in downloading and using the models on-demand. It helps in parallelization which means different jobs can be run parallelly also it reduces redundancy and helps to inspect and debug the data flow in the model. Some of the features that pipelines provide are on-demand computing, tracking of data and computation, inspecting the data flow, etc.

OpenVINT is the union and adaptation of each of the aforementioned algorithms and methods to achieve the objective of the optimization process mentioned at the beginning of this work. The coding of this algorithm was performed based on the Python 3 language mainly, in a GNU/LINUX environment.

Algorithm 4: OpenVINT algorithm

// Initialization phase

Input the design constants d, V_∞, ρ, ν, a, T_{min}, $\bar{r}$, NS;

Input the design variables intervals [X_{min}, X_{max}];

Input the optimization conditions G, NP, NP_{min}, ε, U^*, γ, H, p, g=1;

Initialize of metrics;

Initialize population P_g with LHS;

//Parallelized loop by joblib

for i = 1 **to** NP **do**

    Apply Algorithm 1 for $x_{i,g}$;

//Parallelized loop by joblib

for i = 1 **to** NP **do**

    Get $T(x_{i,g})$, $W(x_{i,g})$, $\eta_d(x_{i,g})$, $\eta_s(x_{i,g})$ with Algorithm 2;

Get $\psi(x_g)$ with (2) and $L(x_g)$ with (1);

Update U^*;

Save $L_{avg}(x_g)$;

Save data of generation g;

Set all values in M_{CR}, M_F to 0.5;

Archive $A = \varnothing$;

k = 1;

// Main loop

for g = 1 **to** G **do**

    $S_{CR} = \varnothing$, $S_F = \varnothing$;

    for i = 1 **to** NP **do**

```
            r_i = select from [1, H] randomly;
            Get CR_{i,g};
            Get F_{i,g};
            Get mutation vector v_{i,g};
            Get trial vector u_{i,g};
        //Parallelized loop by joblib
        for i = 1 to NP do
            Apply Algorithm 1 with u_{i,g};
        //Parallelized loop by joblib
        for i = 1 to NP do
            Get T(u_{i,g}), W(u_{i,g}), η_d(u_{i,g}), η_s(u_{i,g}) with Algorithm 2;
        Get ψ(u_g)  with (2) and L(u_g) with (1);
        Update U*;
        for i = 1 to NP do
            if L(u_{i,g}) ≤L(x_{i,g}) then
                x_{i,g+1} = u_{i,g};
                x_{i,g}→ A;
                CR_{i,g}→ S_CR, F_{i,g}→ S_F;
            else
                x_{i,g+1} = x_{i,g};
        Update memories M_CR and M_F with Algorithm 3;
        Save L_avg(x_{g+1});
        if g≥ 3 then
            Get Δ_g and Δ_{g-1} with (48);
            Get NP_{g+1} with (46);
            if NP_{g+1}<NP_min then
                Apply (47);
            (NP_g – NP_{g+1})-th worst vectors → A;
            Remove the (NP_g – NP_{g+1})-th worst vectors from P_{g+1};
        Save data of generation g+1;
        if |L_avg(x_{g+1}) – L_opt(x_{g+1})| ≤ε then
            break;
        k++;
    Print metrics plots;
    Output x_opt, L(x_opt);
    Drawing the optimal propeller in point clouds;
```

3. Study Case

To evaluate the performance of the *OpenVINT* algorithm, a test was conducted to obtain the optimal design of a propeller used for an engine of a fixed-wing aircraft.

The flow characteristics for this study are V_∞, 25 m/s; ρ, 1.225 kg/m^3; ν, 0.000014607 m^2/s; and a, 340.294 m/s. A minimum permissible thrust of 7.5 N was considered for the optimization process. The intervals of the design variables are shown in Table 1.

 doi:10.20944/preprints202311.0706.v1

Table 1. Intervals of the design variables.

Variable	Interval	Variable	Interval
$(c/d)_r$	[0.03, 0.10]	x_{tt}	[0.30, 0.45]
$(c/d)_m$	[0.03, 0.10]	$\bar{r}_{xtm}$	[0.30, 0.80]
$(c/d)_t$	[0.01, 0.02]	y_{cr}	[0.01, 0.05]
$\bar{r}_{cm}$	[0.35, 0.60]	y_{cm}	[0.01, 0.05]
α_r	[-7, 7] °	y_{ct}	[0.005, 0.05]
α_m	[-5, 7] °	$\bar{r}_{ycm}$	[0.30, 0.80]
α_t	[-5, 7] °	x_{cr}	[0.30, 0.40]
$\bar{r}_{\alpha m}$	[0.25, 0.75]	x_{cm}	[0.30, 0.45]
y_{tr}	[0.12, 0.20]	x_{ct}	[0.30, 0.45]
y_{tm}	[0.12, 0.16]	$\bar{r}_{xcm}$	[0.30, 0.80]
y_{tt}	[0.10, 0.12]	n_m	[5e^3, 10e^3] rev/min
$\bar{r}_{ytm}$	[0.30, 0.80]	B	[2, 4]
x_{tr}	[0.30, 0.40]	d	[0.3, 0.3] m
x_{tm}	[0.30, 0.45]		

For this optimization case the following parameters of the optimization algorithm will be taken:

- in real optimization problems it is considered to use at least 50 individuals in the initial population, and as a minimum population 10 individuals were considered;
- the stop conditions contemplated for this case were, a maximum number of evaluated generations of 200, an ε value of 1 W to fulfill the condition indicated in line 48 of algorithm 4;
- a γ factor of 50 was used in equation 64;
- finally, an initial U*-value of 100 W was considered.

4. Results

To find the optimal propeller geometry, the algorithm needed to evaluate 127 generations to meet one of the stopping conditions. The penalty function was evaluated 5742 times. Figure 6 shows the evolution of the optimization process.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 November 2023 doi:10.20944/preprints202311.0706.v1

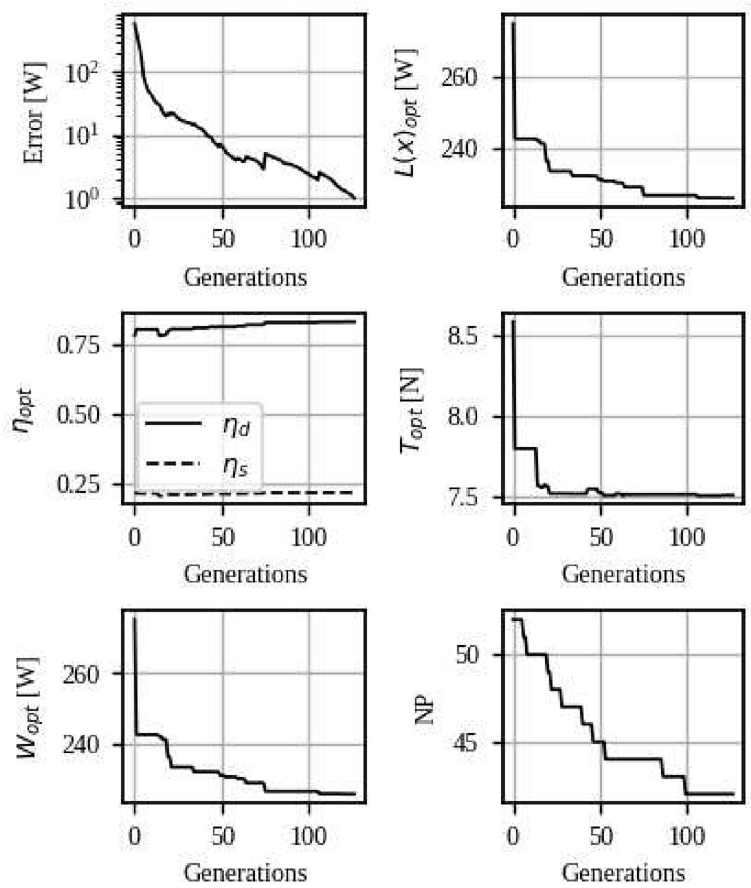


Figure 6. Metrics of the optimization process.

The optimal propeller can generate a thrust of 7.505 N, requiring a power of 226.08 W, with a dynamic efficiency of 83%. The optimal vector is shown in Table 2. The next figure shows the geometric characteristics of the propeller.

Table 2. Optimal vector.

Variable	Value	Variable	Value
$(c/d)_r$	0.0798405	x_{tt}	0.42644
$(c/d)_m$	0.0842341	$\bar{r}_{xtm}$	0.8
$(c/d)_t$	0.010028	y_{cr}	0.0439051
$\bar{r}_{cm}$	0.513672	y_{cm}	0.0152008
α_r	7 °	y_{ct}	0.005
α_m	4.71326 °	$\bar{r}_{ycm}$	0.3
α_t	5.23632 °	x_{cr}	0.385744
$\bar{r}_{\alpha m}$	0.2543	x_{cm}	0.317216
y_{tr}	0.12	x_{ct}	0.3
y_{tm}	0.1007486	$\bar{r}_{xcm}$	0.487034
y_{tt}	0.1071704	n_m	6896.51 rev/min
$\bar{r}_{ytm}$	0.52874	B	2
x_{tr}	0.338746	d	0.3 m
x_{tm}	0.3		

Geometric characteristics of the optimal propeller are shown in Figure 7.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 November 2023 doi:10.20944/preprints202311.0706.v1

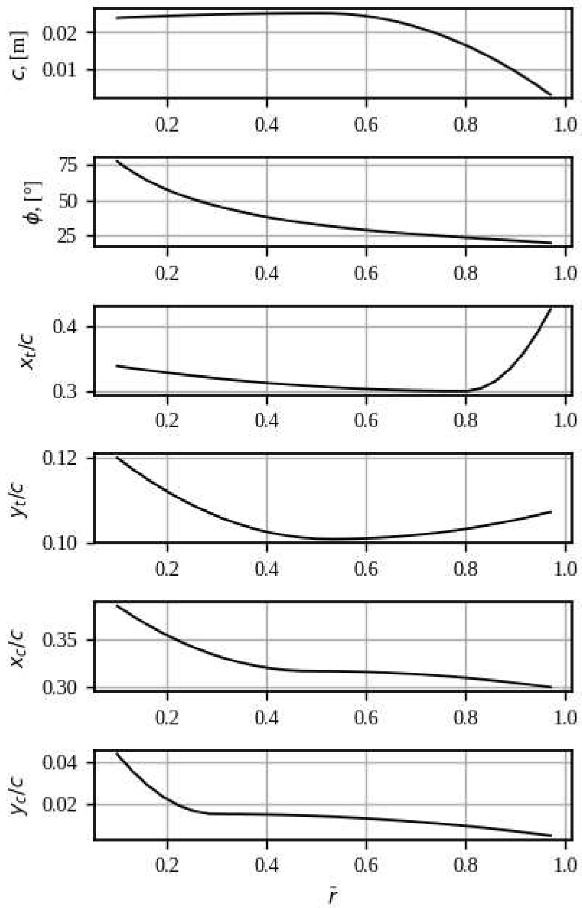


Figure 7. Geometric characteristics of the optimal propeller.

The general view of the propeller and the shape of the airfoils in each section in $\bar{r}$ of the propeller blade are shown in Figure 8.

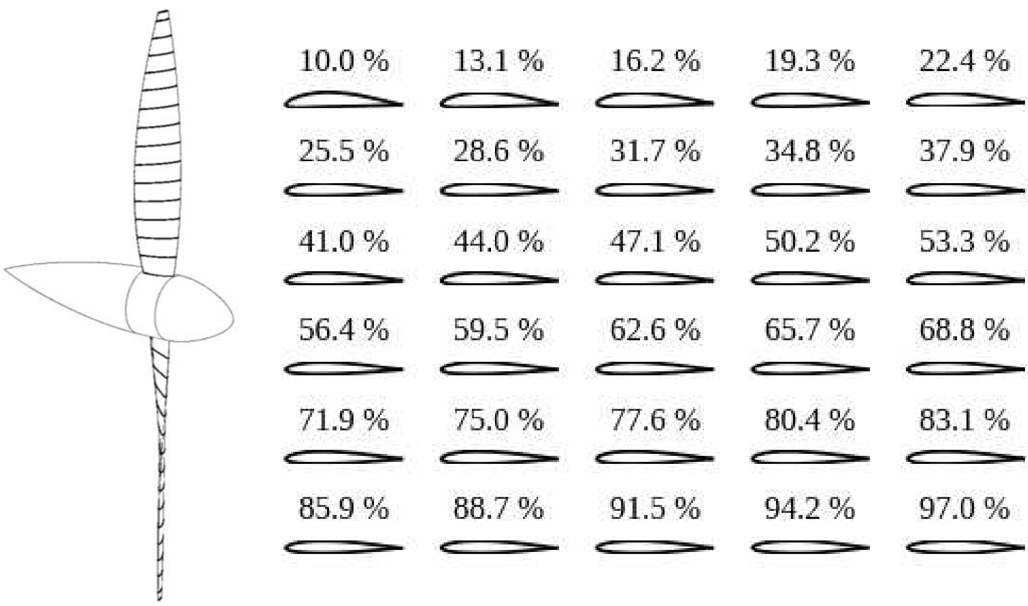


Figure 8. The general view of the propeller and the shape of the airfoils in each section in $\bar{r}$ of the propeller blade.

4. Discussion

 doi:10.20944/preprints202311.0706.v1

Good agreement was shown by comparing the calculations of the aerodynamic characteristics of airfoils in Xfoil with the results of solving the Navier-Stokes equations using the control volume method and experimental data. This allows fast discrete eddy Xfoil calculations with viscosity and compressibility corrections to be used for propeller design calculations.

The cross-sectional calculation method implemented for propeller design showed good agreement with experiment and CFD modeling in terms of thrust and torque (Figures 5a and 5b) and provided a quick time for assessing aerodynamic characteristics.

The optimization of propeller parameters was based on the differential evolution method. It is shown that 120 generations of 60 individuals are sufficient to determine 27 design variables.

The solution to the demonstration problem of designing an aircraft propeller with a required thrust of 7.505 N with a diameter of 300 mm and a free-stream speed of 25 m/s yielded a combination of design parameters that provides a propeller efficiency of 83% and requires 226 W of power at a rotation speed of 6896.5 rev/min.

Approbation of the proposed methodology in the design of real propellers of unmanned aerial vehicles has shown its effectiveness. The propellers obtained because of optimization have consistently high efficiency. This methodology has high performance and does not require large computing power.

5. Conclusions

This paper describes the developed methodology for designing air propellers. The novelty of the technique is the selection of the aerodynamic profile for each cross-section of the propeller blade by using Bezier curves for accuracy, flexible and robust setting of the geometry of the blades. It is shown that to accurately describe a propeller, 27 parameters are sufficient, of which 16 are for describing the profile, 10 are for the shape of the blade, and 1 is for specifying the kinematics (rotation speed of the propeller).

The use of discrete vortex and isolated section methods makes it possible to calculate the thrust and required propeller power in less than one second. Combined with the differential evolution method, this makes it possible to solve the optimization problem in less than 2 hours on a workstation with four Intel Core i5 cores.

The *OpenVINT* program has been developed in Python, which implements the presented methodology with the possibility of parallel calculations (standardly with four threads). To evaluate the performance of the *OpenVINT* algorithm, a test was conducted to determine the optimal design of the propeller used for the engine of an unmanned aerial vehicle. The algorithm makes it possible to obtain propellers with parameters that provide an efficiency of up to 80% under given requirements, which is a good indicator.

The program that implements the technique also allows you to automatically obtain a geometric description of the obtained result in the form of point clouds, from which it is possible to quickly construct a three-dimensional geometric model in any of the existing CAD systems (see Figure 8) for the purpose of its further use in subsequent calculations in other programs or for the development of equipment for the manufacture of propellers using any available technologies.

Author Contributions: Conceptualization, D. N., Ev. K. and O.L.; methodology, D. N., JG. QP., Ev. K. and O.L.; software, JG. QP., Ev. K.; validation, D. N., JG. QP; formal analysis, V. Ch., Ek. K. and VH. H.; resources, A.S.; writing—original draft preparation, A.S., JG. QP, O.L. and V. Ch.; writing—review and editing, JG. QP., Ev. K., O.L., D. N., V. Ch., Ek. K., VH. H. and A.S.; visualization, V. Ch., Ek. K. and VH. H.; supervision, D. N.; project administration, D. N.; funding acquisition, D. N., O.L. and A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Samara National Research University Development Program for 2021–2030 as part of the Strategic Academic Leadership Program "Priority 2030",grant number PR-NU 2.1-08-2023.

Data Availability Statement: The raw data cannot be shared at this time as the data also forms part of an ongoing study.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Traub, L.W. Considerations in optimal propeller design. *Journal of Aircraft* **2021**, 58(4), 8 p., https://doi.org/10.2514/1.C036258.
2. Betz, A. Introduction to the Theory of Flow Machines. *Oxford: Pergamon Press* **1966**.
3. Betz, A. Schraubenpropeller Mit Geringstem Energieverlust (Screw Propeller with Least Energy Loss). *Akademie der Wissenschaften in Gottingen, Germany* **1919**, pp. 193-217.
4. Adkins, C.N; Liebeck, R.H. Design of optimum propellers. *J Propul Power* **1994**, *10*, pp. 676–682.
5. Fang, B R. Design of Aircraft Aerodynamic Configuration. *Beijing: Chinese Aviation Industry Press* **1997**, pp. 97–130, https://doi.org/10.3390/aerospace9020079.
6. Colozza, A. High altitude propeller design and analysis overview. NASA/CR 98-208520. **1998**.
7. Morgado, J., Abdollahzadeh, M. Silvestre MAR, et al. High altitude propeller design and analysis. *Aerosp Sci Technol* **2015**, *45*, pp. 398–407, https://doi.org/10.1016/j.ast.2015.06.011.
8. Gur, O.; Rosen, A. Optimization of Propeller Bases Propulsion Systems. *Journal of Aircraft*, **2009**, *46(1)*, pp. 95–106, https://doi.org/10.2514/1.36055.
9. Dorfling, J.; Rokhsaz, K. Constrained and Unconstrained Propeller Blade Optimization. *Journal of Aircraft*, **2015**, *52(4)*, pp. 1179–1188, https://doi.org/10.2514/1.C032859.
10. Yu, P.; Peng, J.; Bai, J.; Han, X.; Song, X. Aeroacoustic and Aerodynamic Optimization of Propeller Blades. *Chinese Journal of Aeronautics* **2020**, *33(3)*, pp. 826–839, https://doi.org/10.1016/j.cja.2019.11.005.
11. Wang, S.; Zhang, S.; Ma, S. An Energy Efficiency Optimization Method for Fixed Pitch Propeller Electric Aircraft Propulsion Systems. *IEEE Access* **2019**, *7*, pp. (99):1-1, https://doi.org/10.1109/ACCESS.2019.2950453.
12. Bekele, E.G.; Nicklow J.W. Multi-objective automatic calibration of SWAT using NSGA-II. *J Hydrology* **2007**, *341*, pp. 165–176, https://doi.org/10.1016/j.jhydrol.2007.05.014.
13. Ma, R; Zhong, B.W, Liu P.Q. Optimization design study of low-Reynolds-number high-lift airfoils for the high-efficiency propeller of low-dynamic vehicles in stratosphere. *Sci China Technol Sci* **2010**, *53*, pp. 2792–2807, https://doi.org/10.1007/s11431-010-4087-0.
14. Jiao, J.; Song B.-F.; Zhang Y.-G.; Li Y.-B. Optimal design and experiment of propellers for high altitude airship. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* **2017**, *232(10)*, pp.1887-1902, https://doi.org/ 10.1177/0954410017704217.
15. Munguia, J.; Van Treuren W. Designing small propellers for optimum efficiency. *Baylor University, Waco, TX, 76798*, 27 p.
16. Ali, M.M.; Zhu W.X. A penalty function-based differential evolution algorithm for constrained global optimization. *Comput. Optim. Appl.* **2013**, *54*, pp. 707-739, https://doi.org/10.1007/s10589-012-9498-3.
17. Pierret S.S.; Van den Braembussche R.A. Turbomachinery Blade Design Using a Navier-Stokes Solver and Artificial Neural Net-work. *Journal of Turbomachinery* **1999**, *121(2)*, pp.326-332
18. Borovkov A.I.; Voinov I.B.; Ibraev D.F. Determination of the Optimal Aerodynamic Shape for a Propeller Blade Based on Parametric Optimization. *Izv. VUZ. Aviatsionnaya Tekhnika* **2021**, *2*, 9 p.
19. Van To, T.; Ngoc Phien H. Development of Bezier-based curves. *Computers in Industry* **1992**, 20(1), pp. 109-115.
20. Zherejov, V.V.; Kusumov A.N. Aehrodinamicheskij raschet nesushchego vinta vertoleta uch posobie po kursovomu i diplomnomu proektirovaniyu (in rus). *Russia, Kazan: KGTU* **1997**, 28 p.
21. Derksen, R.W.; Rogalsky T. Bezier-PARSEC: An optimized aerofoil parameterization for design. *Advances in Engineering Software*, **2010**, *41*, pp. 923-930, https://doi.org/10.1016/j.advengsoft.2010.05.002.
22. Kravec, A.S. Kharakteristiki vozdushnykh vintov (in rus). *Uchebnoe posobie, Gos. Izd. oboronnoj promyshlennosti, Moscow* **1941**.
23. Eltaeib, T.; Mahmood A. Differential Evolution: A survey and analysis. *Appl. Sci.* **2018**, *8*, P. 1954, https://doi.org/10.3390/app8101945.
24. Tanabe, R.; Fukunaga A.S. Improving the search performance of SHADE using linear population size reduction. *IEEE Congress on Evolutionary Computation (CEC)* **2014**, pp. 1200-1207, https://doi.org/10.1109/CEC.2014.6900380.
25. Wong, I.; Liu W.; Ho C.H.; Ding X. Continuous adaptive population reduction (CAPR) for differential evolution optimization. *SLAS Technology* **2017**, 22(23), pp. 289-305, https://doi.org/10.1177/2472630317690318.
26. Iman, R.L. Encyclopedia of quantitative risk analysis and assessment. *John Wiley & Sons* **2008**, pp. 408-411.
27. Zielinksi, K.; Weitkemper P.; Laur R.; Kammeyer K.D. Examination of stopping criteria for differential evolution based on a power allocation problem. Available online. URL: https://api.semanticscholar.org/CorpusID:13358798 (accessed on 02.11.2023).
28. Sharma, H. Lightweight pipelining in Python. Using Joblib for storing the machine learning pipeline to a file. Available online. URL: https://towardsdatascience.com/lightweight-pipelining-in-python-1c7a874794f4 (accessed on 02.11.2023).