JOURNAL OF AIRCRAFT
Vol. 46, No. 6, November–December 2009

Design and Optimization Method for Inertial Particle Separator Systems

Eyas F. Al-Faris*
King Abdulaziz City for Science and Technology, Riyadh 11442, Saudi Arabia
and
Farooq Saeed†
King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

DOI: 10.2514/1.40921

The paper presents a method for the design and optimization of a multi-element airfoil-based inertial particle separator. To facilitate design in an interactive manner, a MATLAB graphical user interface was developed with the following capabilities: 1) design of individual airfoil elements; 2) orientation and arrangement of the airfoils to form a multi-element airfoil-based inertial particle separator system by employing translational, scaling, and rotational functions; 3) analysis of the inertial particle separator system; and 4) optimum design of the inertial particle separator system using an evolutionary algorithm-based direct-search optimization technique. The design of individual airfoils was achieved through the use of the PROFOIL code, a state-of-the-art multipoint inverse airfoil design program. Multiple airfoils were arranged to form a multi-element airfoil-based inertial particle separator system subject to geometric constraints. A two-dimensional analysis tool for the multi-element airfoil was linked to the synthesis component of the graphical user interface to carry out flow and particle trajectory analysis component of the program. For the optimization component of the program, a computational objective function was implemented and coupled with a pattern search optimizer of the Direct Search Toolbox in MATLAB. Design and optimization was achieved using multivariable optimization. The paper presents two design examples to demonstrate the design method.

Nomenclature

C_d	=	particle drag coefficient
c	=	airfoil chord length
D_{eq}	=	equivolumetric or mean volumetric diameter
$\boldsymbol{F}_a$	=	aerodynamic force
$\boldsymbol{F}_g$	=	gravitational force
g	=	gravitational acceleration constant
$\boldsymbol{i}, \boldsymbol{k}$	=	unit direction vectors in the wind reference frame
$\boldsymbol{i}_p, \boldsymbol{k}_p$	=	unit direction vectors in the body reference frame
m_p	=	particle mass, $\rho_p V_p$
$\boldsymbol{n}$	=	surface normal vector
p	=	ambient pressure
Re	=	Reynolds number based on particle diameter, $\rho_a D_{\mathrm{eq}} U/\mu_a$
$\boldsymbol{r}_p$	=	particle position
S	=	particle surface projection on the $\boldsymbol{U}$ perpendicular plane
S_p	=	particle surface area
t	=	time
U	=	magnitude of particle relative velocity in the body reference frame, $\lvert\boldsymbol{U}\rvert$
$\boldsymbol{U}$	=	particle relative velocity in the body reference frame, $\boldsymbol{V}_a - \boldsymbol{V}_p$
$\boldsymbol{V}_a$	=	freestream velocity in the body reference frame, $u_a\boldsymbol{i} + w_a\boldsymbol{k}$
V_p	=	particle volume
$\boldsymbol{V}_p$	=	particle velocity in the body reference frame, $\mathrm{d}\boldsymbol{r}_p/\mathrm{d}t$
x, z	=	axes in the wind reference frame
x_p, z_p	=	axes in the body reference frame
z	=	pressure head
θ	=	angle between the z_p axis and z axis
μ_a	=	ambient air viscosity
ρ_a	=	ambient air density
ϱ_p	=	particle mass density
$\bar{\tau}$	=	shear stress

I. Introduction

SAUDI Arabia is known for its harsh desertlike environment. The air is usually filled with sand and dust particles. Aircraft operating under such conditions for long periods are vulnerable to internal engine damage. The operation life of a helicopter engine operating in sandy environments can be as short as 50 h [1,2]. Engine damage ranges from simple erosion in the engine blades to a completely inoperative engine with as little as one-half pound of sand [3]. In addition, the degrading of the performance and efficiency of the aircraft engine leads to increased fuel consumption and operational cost [1]. It therefore becomes imperative that some form of protection device must be used at the engine inlet to prevent sand from entering the engine. Such a device is an inertial particle separator (IPS) system.

IPS systems were developed initially for helicopters. Pictured in Fig. 1 [4] is a Boeing CH-47D helicopter with the IPS system installed (shown inside the dashed circle) on the engine inlets. The IPS is in the form of an axisymmetric bifurcated duct, as in Fig. 2a. The IPS systems available today are similar in design. Air filled with sand enters the device through the inlet and flows around sharp bend B. The end is designed in such a way that the inertia of the entering particles is ample to prevent them from flowing with the air around the bend. Thus, sand particles pass into scavenge passage A and the sand-free clean air flows into the engine.

It was proposed in [3] that a multi-element airfoil configuration IPS (Fig. 2b) be used instead of the current design (Fig. 2a). In this approach, it is assumed that the cross-sectional profile of the engine

Presented as Paper 6066 at the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, Canada, 10–12 September 2008; received 12 September 2008; revision received 16 April 2009; accepted for publication 14 June 2009.

*Research Engineer, National Aeronautics Technologies Program. Member AIAA.

†Assistant Professor, Aerospace Engineering Department. Lifetime Member AIAA.

Fig. 1 Boeing CH-47D helicopter with an IPS system installed [4].

inlet with an integral IPS system can be treated as a multi-element airfoil configuration, as illustrated in Fig. 2b. A careful examination of Fig. 2a suggests that the path of sand particles can be directed by an appropriate design as well as positioning of airfoil elements 1–5.

The objective of this study is to develop and propose a design and optimization method for a multi-element airfoil-based IPS system (Fig. 2b). The main design requirements of such a system are the separation of particles (such as sand, dust, and water droplets) from the air entering the engine, while keeping the mass flow rate entering at the inlet unchanged. Sand ingestion inside the engine is avoided by a careful design of airfoils that define the internal flow path within the engine. Large particles entering these flow paths get separated from the air solely due to their inertia. Because such an IPS system does not require any form of mechanical or electrical power, it is not a strain on the engine and keeps the engine performance unaffected. The design of the IPS system is achieved by developing a program that can

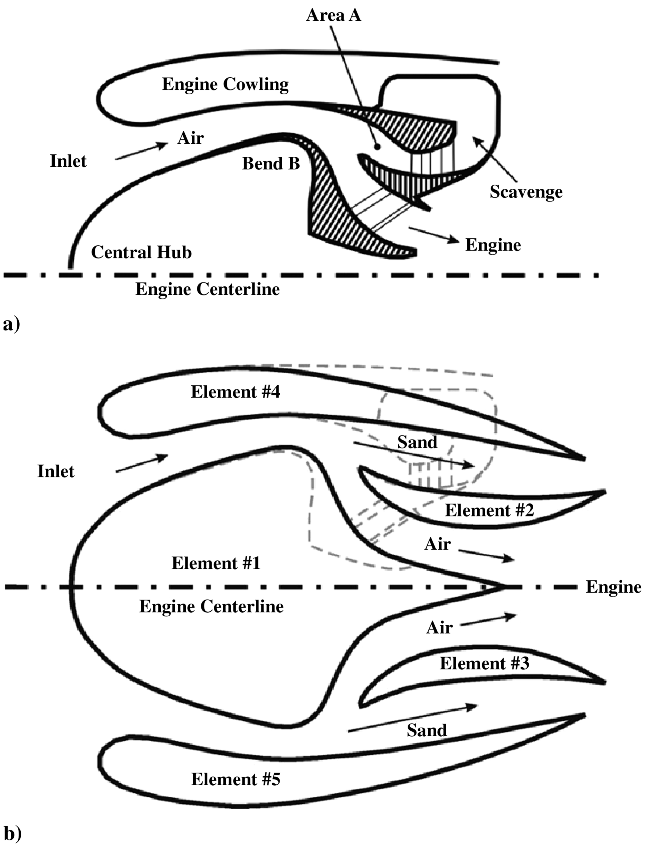


Fig. 2 Illustrations of a) typical axisymmetric helicopter engine particle separator (adapted from [4]) and b) proposed five-element airfoil configuration model for an IPS system [3].

synthesize, analyze, and perform design optimization for a given set of engine operational requirements. The final design will therefore prevent particles from entering the aircraft engine, resulting in better performance and longer operational life. The strength of such a design and optimization method lies in the fact that it is fast and less computationally intensive compared with the existing IPS design and optimization methods that take advantage of advanced analytical and computational fluid dynamics techniques for throughflow and particle trajectory analysis [5–7].

II. Design Methodology

As stated earlier, the intent is to develop a system that designs and optimizes the shape of an IPS system. To help define an initial IPS system configuration and then carry out design in an interactive manner, a MATLAB graphical user interface (GUI) was developed. The GUI performs three major tasks that are accomplished by its three main program components, as detailed next.

A. Synthesis Engine

The synthesis component defines an initial configuration of a multi-element airfoil-based IPS system. This is achieved by generating multiple airfoil geometries and positioning them subject to a set of geometric constraints such as size of the engine and inlet in terms of maximum diameter and length of each airfoil element.

B. Analysis Engine

The analysis component performs flow and particle trajectory analysis of the multi-element airfoil configuration. The output from the analysis engine is used to simulate particle trajectories through the IPS system. The particle trajectories are initiated sufficiently ahead of the IPS system and terminate either at the exit if they do not strike any surface or at the location at which they impinge upon any of the airfoil surface. The trajectories are displayed in the GUI after every run through the analysis engine.

C. Optimization Engine

This optimization component analyzes the results of the analysis engine and shape-optimizes the geometry of the IPS system, based on a given objective function and geometric constraints. This is achieved by employing an evolutionary optimization technique to search for an optimal design within the specified constraints.

The following sections describe the preceding program components in greater detail.

III. Synthesis Engine

As mentioned earlier, the synthesis component defines an initial configuration of a multi-element airfoil-based IPS system. The first step in this process is to design individual airfoil geometries that will represent the central hub (element 1 in Fig. 2b), the engine housing (elements 2 and 3 in Fig. 2b), and the outer engine cowling (elements 4 and 5 in Fig. 2b). Taking advantage of symmetry about the engine centerline, only elements 1, 2, and 4 need to be designed, because elements 2 and 4 are mirror images of elements 3 and 5, respectively. The design of the individual airfoils is accomplished with the help of PROFOIL, a multipoint inverse airfoil design code [8–10].

A. PROFOIL Code

PROFOIL is a low-speed airfoil design code. It employs the inverse design approach as opposed to the direct design approach. In the direct design approach (Fig. 3), the airfoil shape is specified first and then analyzed to determine a desired analysis property such as velocity distribution, lift, or drag. Here, the airfoil shape is adjusted until the desired analysis property is obtained. A drawback of the direct approach is that the designer spends a great deal of time going through a hit-and-trial process of defining different airfoil shapes to achieve the desired analysis property.

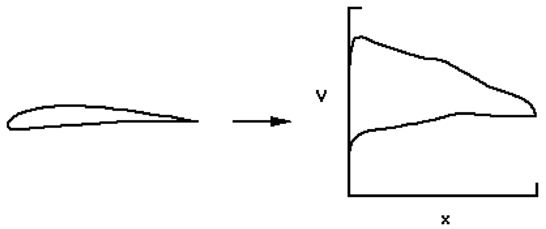


Fig. 3 Direct approach to airfoil design [8].

An inverse airfoil design technique is one in which the airfoil geometry is obtained from a specification of desired velocity distribution(s) (Fig. 4) subject to certain constraints. The method is based on conformal mapping of flow around a circle (known) to that around the airfoil (desired) through conformal transformation. The strength of the method lies in its two main features:

1) The derivative of the transformation rather than the transformation is of importance and is easily related to the desired velocity distribution.

2) Multipoint capability is achieved by dividing the circle into a number of arc segments that relate to an equal number of segments on the airfoil for which the desired velocity distribution(s) are specified (Fig. 5).

Typically, good performance is required over a range of angles of attack. For example, high-lift (high-angle-of-attack) performance may be required as well as low-lift (low angle of attack). Thus, for instance, upper-surface velocity distribution can be prescribed for a high angle of attack, and lower-surface velocity distribution can be simultaneously prescribed for a low angle of attack. Because the velocity distribution corresponds to the lift coefficient (which, in turn, depends on the angle of attack), the velocity distribution along different airfoil segments can be related to different angle-of-attack conditions (i.e., α^* or C_l^*). A good approximation is given by $C_l^* = 2\pi(1 + 0.78t/c)\sin\alpha^*$, where t/c is the airfoil thickness-to-chord ratio and α^* corresponds to the zero-lift line. The resulting airfoil will therefore exhibit the design characteristics (velocity distribution and C_l^*) when operated at the corresponding α^*.

To achieve practical airfoils, the derivative of transformation must be continuous at the junction of two segments (continuity constraints) and the airfoil trailing edge must be closed (closure constraint); in addition to the condition that the far-field flow remain unaltered, the satisfaction of these constraints leads to a system of $(N + 3)$ equations, where N is the number of segments. As mentioned earlier, the specification of the velocity distributions is not completely arbitrary. It must contain an equal number of unknowns $(N + 3)$ to obtain a solution of the problem. Typically, these unknowns are the velocity levels on $(N - 1)$ segments, and the remaining four variables define the form of the recovery and closure functions for flow at the trailing edge along the upper and lower surfaces. Additional constraints (dependent variables) such as pitching moment, maximum thickness, camber, etc., can also be imposed and satisfied through an iterative procedure by varying some independent variables in the design. Simultaneous solution of the constraints requires a multidimensional Newton iteration scheme and is accomplished within 10–15 iterations. Details of the mathematical formulation and various applications of the method are described in greater detail in [8–10].

In the present study, the PROFOIL code is used as an airfoil design tool. The design of the IPS system is achieved by varying the camber, thickness, and pitching moment coefficient of the individual airfoil elements. In turn, the individual airfoil element shape depends upon the specified camber, thickness, and pitching moment and is obtained by a systematic variation of the upper and lower surfaces' α^* or C_l^*. Thus, a suitable choice of camber, thickness, and pitching moment coefficient is used to obtain a practical IPS design. A limitation of the code is that only incompressible airfoil designs can be generated, which is sufficient for the current study.

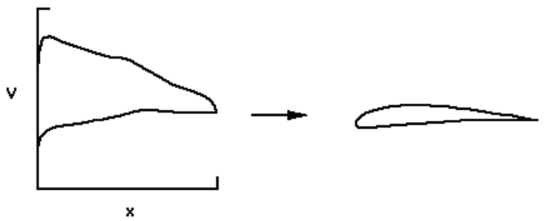


Fig. 4 Inverse approach to airfoil design [8].

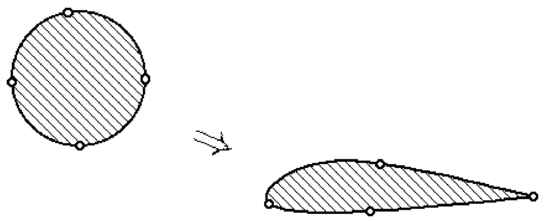


Fig. 5 Simple four-segment airfoil [8].

B. MATLAB GUI and the Synthesis Engine

After choosing the airfoil design code, it was linked via a GUI (Fig. 6) to a synthesis engine that can load these airfoils into the design space, each airfoil element at a time, by altering the PROFOIL input script to generate the IPS airfoil elements. Afterward, the designer has the ability to position each element into its appropriate space and generate the multi-element IPS configuration. This positioning ability is accomplished by implementing translational, rotational, and scaling functions. To give the designer more precision while designing, the amount by which the element is positioned can be specified to the level of the first decimal point. The synthesis engine deals with a 2-D design space (x, y). To move an airfoil element, spatial increments dx and dy are added to the original coordinates (x, y) of the airfoil to obtain the new coordinates (x', y') as follows:

$$x' = x + \mathrm{d}x, \qquad y' = y + \mathrm{d}y \tag{1}$$

To rotate an airfoil element, the transformation equations for rotation about a specified rotation position (x_r, y_r) are given by

$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta \tag{2}$$

$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta \tag{3}$$

where θ is the rotation angle.

To rescale an airfoil element, the transformation equations for scaling about a fixed reference point (x_f, y_f) are given by

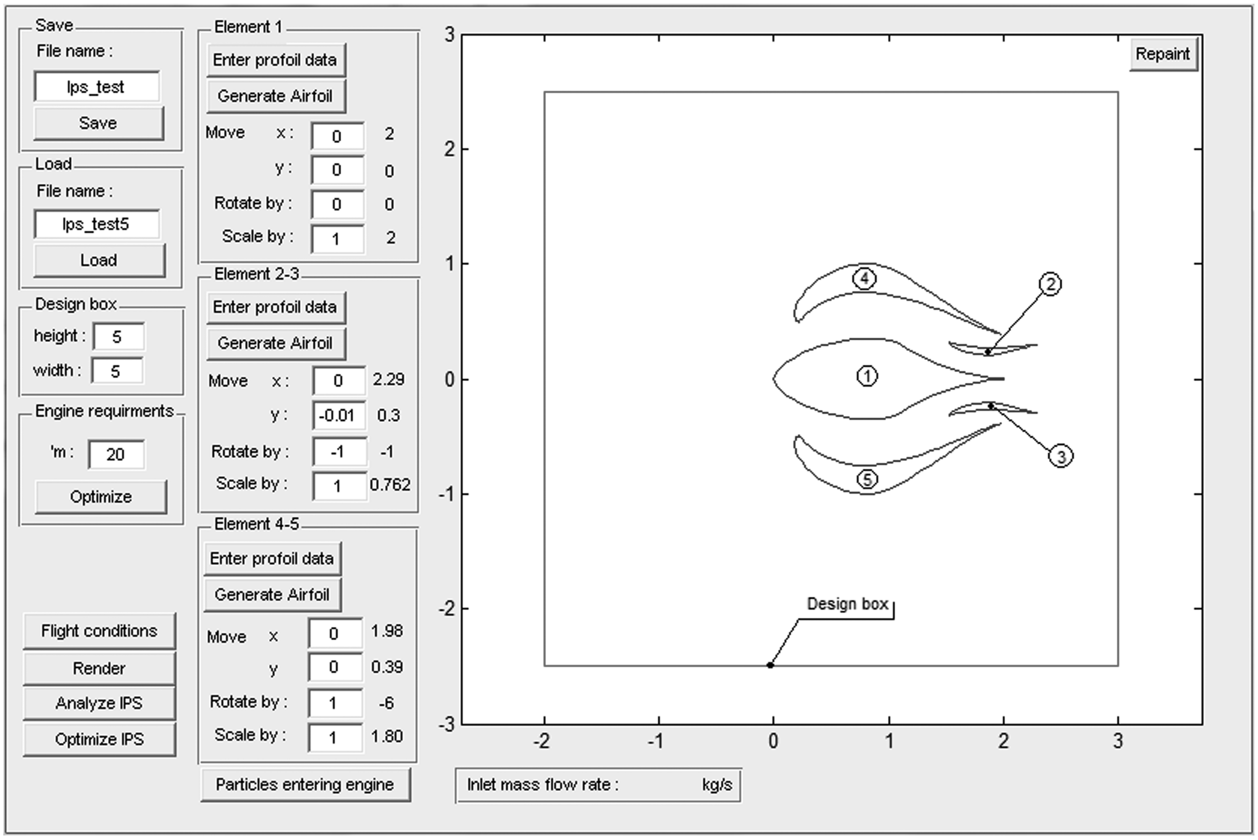
$$x' = x_f + (x - x_f)s_x \tag{4}$$

$$y' = y_f + (y - y_f)s_y \tag{5}$$

where s_x and s_y are the scale factors in the x and y directions, respectively.

In this study, (x_r, y_r) and (x_f, y_f) are taken to be the trailing-edge coordinates of the airfoil element. Moreover, s_x and s_y are taken to be equal, to preserve the original airfoil shape.

As mentioned earlier, symmetry about the engine centerline requires that only elements 1, 2, and 4 need to be designed, because elements 2 and 4 are mirror images of elements 3 and 5, respectively. Thus, once a design of elements 1, 2, and 4 is obtained, the airfoils are arranged into an IPS system for which the geometry is saved in a MATLAB data file for later use. A design box was implemented into the synthesis engine to constrain the element positioning inside the box, and the width and height of the box can be specified by the designer. The synthesis engine also has the capability of rendering the IPS system in 3-D (Fig. 7) by revolving each element about the engine centerline axis (x axis) using the following parametric equations:



Save
File name : Ips_test
Save

Load
File name : Ips_test5
Load

Design box
height : 5
width : 5

Engine requirments
'm : 20
Optimize

Flight conditions
Render
Analyze IPS
Optimize IPS

Element 1
Enter profoil data
Generate Airfoil
Move x : 0 2
y : 0 0
Rotate by : 0 0
Scale by : 1 2

Element 2-3
Enter profoil data
Generate Airfoil
Move x : 0 2.29
y : -0.01 0.3
Rotate by : -1 -1
Scale by : 1 0.762

Element 4-5
Enter profoil data
Generate Airfoil
Move x 0 1.98
y 0 0.39
Rotate by : 1 -6
Scale by : 1 1.80

Particles entering engine

Inlet mass flow rate : kg/s

Fig. 6 MATLAB GUI for IPS system design.

$$0 \leq t \leq 2\pi \tag{6}$$

$$x^R = x \tag{7}$$

$$y^R = y \sin(t) \tag{8}$$

$$z^R = y \cos(t) \tag{9}$$

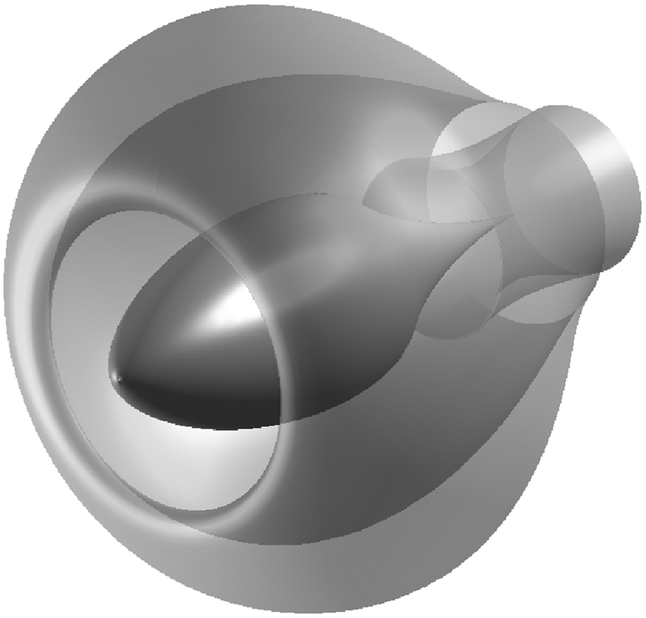


Fig. 7 A 3-D rendering of the IPS system.

where t is the surface parameter used to revolve the points, and x^R, y^R, and z^R are the coordinates needed to generate the revolved surface. The revolved surfaces are rendered transparent to reveal the inner elements of the IPS system, and element 1 (the middle element) was rendered as a solid surface (Fig. 8). MATLAB 3-D OpenGL shading and lighting capabilities were used to implement the IPS 3-D rendering function.

IV. Analysis Engine

A. Flow and Trajectory Analysis

A flow and trajectory analysis code [3] for the multi-element configuration-based IPS system analysis was used in the current program for sand-particle trajectory and impingement analysis. The IPS system analyzer is able to perform impingement analysis using spherical/nonspherical solid particles as well as water droplets for a range of Reynolds numbers ($10^{-4} \leq Re \leq 5 \times 10^5$).

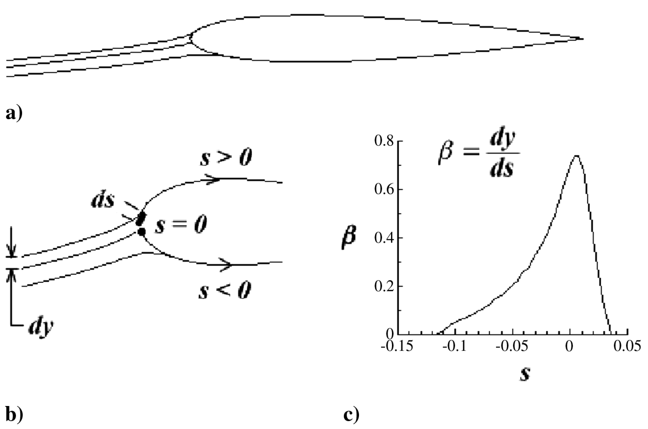


Fig. 8 Illustrations of a) particle trajectories, b) close-up view and nomenclature, and c) typical local impingement efficiency *b*-curve [12].

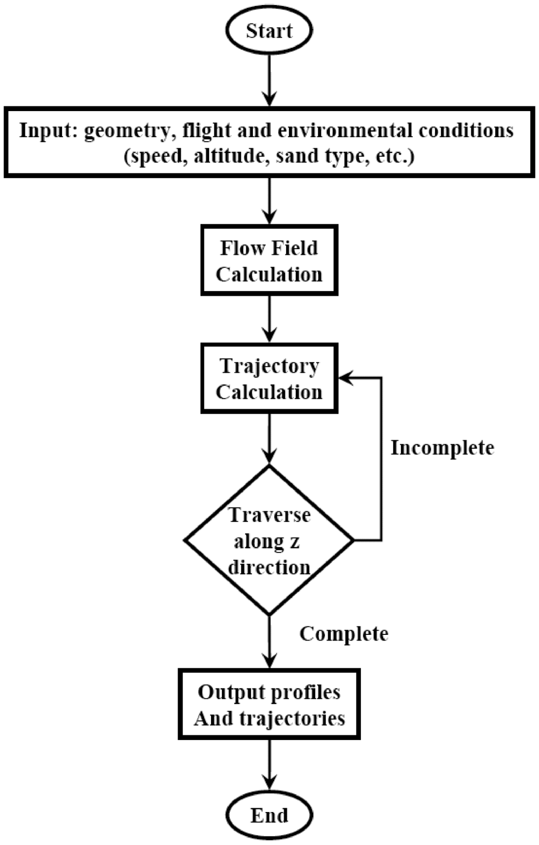

Start
Input: geometry, flight and environmental conditions (speed, altitude, sand type, etc.)
Flow Field Calculation
Trajectory Calculation
Incomplete
Traverse along z direction
Complete
Output profiles And trajectories
End

Fig. 9 Flowchart of trajectory calculation numerical procedure.

The flow analysis code employs the panel method of Hess and Smith, as described in [11], which is an inviscid flow analysis method for multi-element configurations. This method determines the velocity potential field around multi-element airfoil configurations. Panel-method-based flow analysis code applies to two-dimensional flow, whereas the flow in the IPS system can be three-dimensional, depending on the magnitude of swirl present in the flow. In the implementation of the analysis code [3], it was assumed that the magnitude of swirl inside the IPS is small enough to be neglected and flow can be treated as quasi-two-dimensional. Thus, the flow analysis and trajectory calculations of the analysis code are carried out in a two-dimensional space.

The trajectory of each sand particle impinging on the element surface needs to be determined to identify the particle's impingement characteristics. The analysis code [3] achieves this trajectory calculation by applying a force-momentum balance on a particle moving through air. The resulting momentum equation (a differential equation) is then solved with some known initial conditions until the particle impacts a surface panel or travels past the entire multi-element configuration without an impact. Similar studies related to sand and water-droplet impingement and ice accretion on aircraft and engine inlet surfaces are available in the literature [5–7,12–21].

Particle position and velocity with respect to the body reference frame are represented by the vectors $\boldsymbol{r}_r$ and $\boldsymbol{V}_p$. The motion of the particle is governed by the particle momentum equation, which is written as

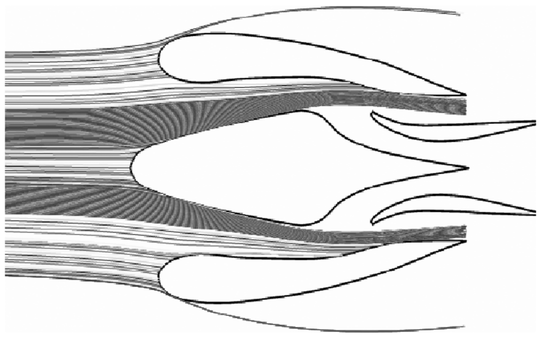


Fig. 10 Particle trajectories around a five-element airfoil configuration IPS system [3].

$$m_p \frac{\mathrm{d}^2 \boldsymbol{r}_p}{\mathrm{d}t^2} = \boldsymbol{F}_a + \boldsymbol{F}_g \tag{10}$$

where m_p is the particle mass, $\boldsymbol{F}_a$ is the aerodynamic force (pressure and shear), and $\boldsymbol{F}_g$ is the gravity force.

The gravity force is related to the weight of the particle as follows:

$$\boldsymbol{F}_g = m_p g(\sin\theta \boldsymbol{i} - \cos\theta \boldsymbol{k}) = \rho_p V_p g(\sin\theta \boldsymbol{i} - \cos\theta \boldsymbol{k}) \tag{11}$$

where g is the gravitational acceleration. The aerodynamic force is due to the pressure and shear forces acting on the particle surface. If S_p is considered as the particle surface, $\boldsymbol{n}$ is the normal vector on the particle surface, and $\boldsymbol{k}_p$ is the direction of the local vertical axis, the aerodynamic force can be expressed by the relation

$$\boldsymbol{F}_a = -\int_{S_p} (p - \rho_a g z)\boldsymbol{n}\,\mathrm{d}S + \int_{S_p} \bar{\bar{\tau}}.\boldsymbol{n}\,\mathrm{d}S \tag{12}$$

The term relating the gravity force can be rewritten as

$$\rho_a \int_{S_p} gzn\,\mathrm{d}S = \rho_a \int_{V_p} \nabla(gz)\,\mathrm{d}V = \rho_a g V_p k_p$$
$$= \rho_a g V_p(-\sin\theta i + \cos\theta k) \tag{13}$$

where V_p is the particle volume. The other terms of Eq. (12) can be divided into two components: one in the same direction as the velocity $\boldsymbol{U}$ (the flow velocity in the body reference frame), which is the drag, and the other term in the direction perpendicular to $\boldsymbol{U}$, which is the lift. Here, ρ_a is the density of air, and p is the ambient pressure. Studies have indicated that there is no lift if the particle does not have a rotational movement and keeps an axisymmetric shape along the $\boldsymbol{U}$ direction; furthermore, there is no lift if the flow is irrotational. The analysis code uses this assumption to neglect the lift force and only consider the drag force. In addition, the drag force evaluation needs to consider both pressure and shear forces, because the small size of the particles is in the range in which shear forces cannot be neglected. The analysis code uses a commonly used empirical correlation to find the drag coefficient C_D of the particle, because calculation of such terms can be very demanding.

Finally, the aerodynamic force is given by the following relation:

$$\boldsymbol{F}_a = \rho_a g V_p(-\sin\theta \boldsymbol{i} + \cos\theta \boldsymbol{k}) + \tfrac{1}{2}\rho_a S C_d U U \tag{14}$$

Substituting the preceding expressions for both aerodynamic force and gravity force in the particle momentum equation (10) yields

Table 1 Types and characteristics of loose, dry sand (density = 1422 kg/m^3) [3]

Sand types	Diameter range, mm	Diameter range, μm	Minimum volume, m^3	Mass, kg
Very fine	0.05–0.1	50–100	6.54E − 14	9.44E − 11
Fine	0.1–0.25	100–250	5.24E − 13	7.55E − 10
Medium	0.25–0.5	250–500	8.18E − 12	1.18E − 08
Coarse	0.5–1	500–1000	6.54E − 11	9.44E − 08
Very coarse	1–2	1000–2000	5.24E − 10	7.55E − 07

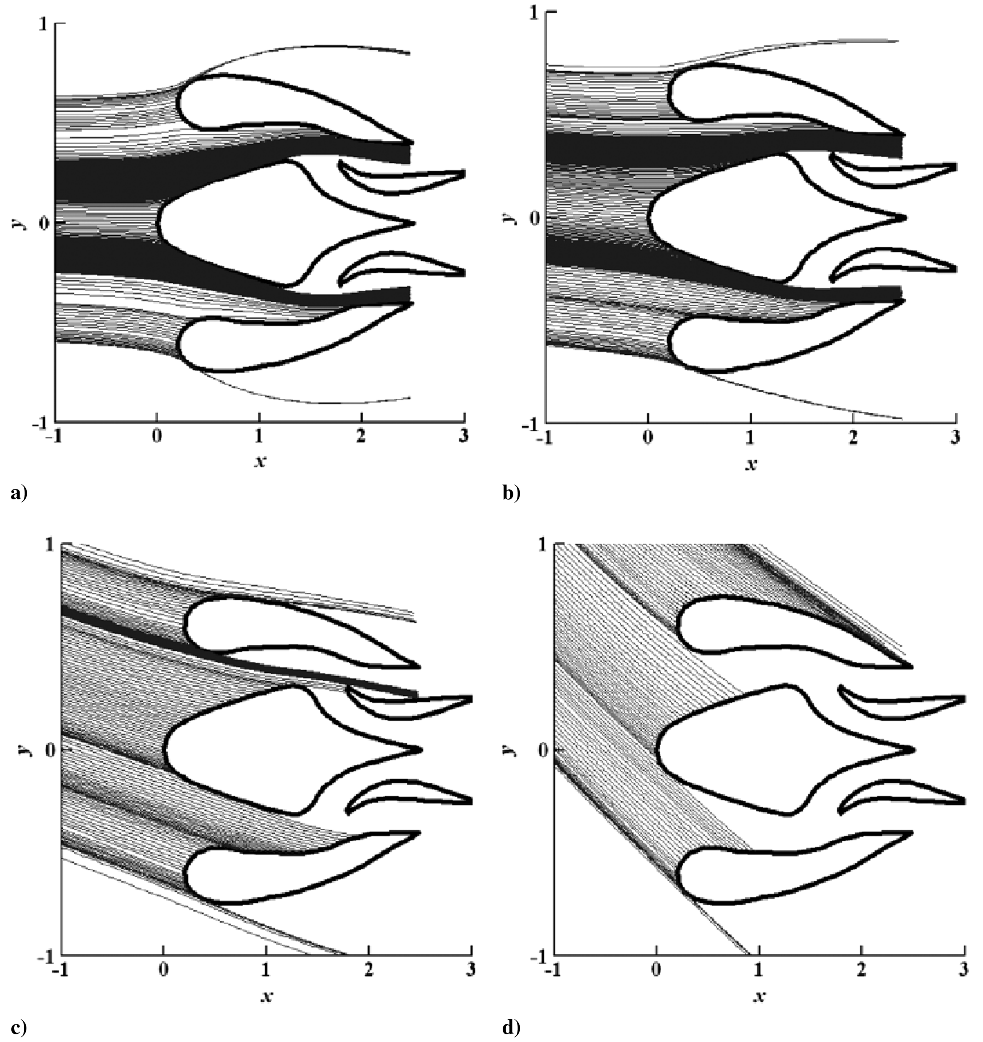


Fig. 11 Plot of sand-particle trajectories of a) very fine, b) fine, c) medium, and d) coarse sand particles [3].

$$\rho_p V_p \frac{\mathrm{d}^2 \boldsymbol{r_p}}{\mathrm{d}t^2} = (\rho_a - \rho_p) g V_p (-\sin\theta \boldsymbol{i} + \cos\theta \boldsymbol{k}) + \frac{1}{2}\rho_a S C_d U \boldsymbol{U} \quad (15)$$

By assuming [3] that the particle surface area and volume are $S = \pi D_{\mathrm{eq}}^2/4$ and $V_p = \pi D_{\mathrm{eq}}^3/6$, respectively, and that the Reynolds number based on equivolumetric particle diameter D_{eq} is $Re = \rho_a D_{\mathrm{eq}} U/\mu_a$, the previous equation can be rewritten as

$$\frac{\mathrm{d}^2 \boldsymbol{r}_p}{\mathrm{d}t^2} = \frac{\rho_a - \rho_p}{\rho_p} g(-\sin\theta \boldsymbol{i} + \cos\theta \boldsymbol{k}) + \frac{3 C_d Re \mu}{4 \rho_p D_{\mathrm{eq}}^2} \boldsymbol{U} \quad (16)$$

Finally, introducing two parameters $K_g = (\rho_p - \rho_a) g/\rho_p$ and $K_a = \rho_p D_{\mathrm{eq}}^2/(18\mu_a)$ in the preceding equation and noting that $\boldsymbol{U} = \boldsymbol{V}_a - \boldsymbol{V}_p$ yields

$$\frac{\mathrm{d}^2 \boldsymbol{r}_p}{\mathrm{d}t^2} + \frac{C_d Re}{24 K_a} \frac{\mathrm{d}\boldsymbol{r}_p}{\mathrm{d}t} = K_g(\sin\theta \boldsymbol{i} - \cos\theta \boldsymbol{k}) + \frac{C_d Re}{24 K_a} \boldsymbol{V}_a \quad (17)$$

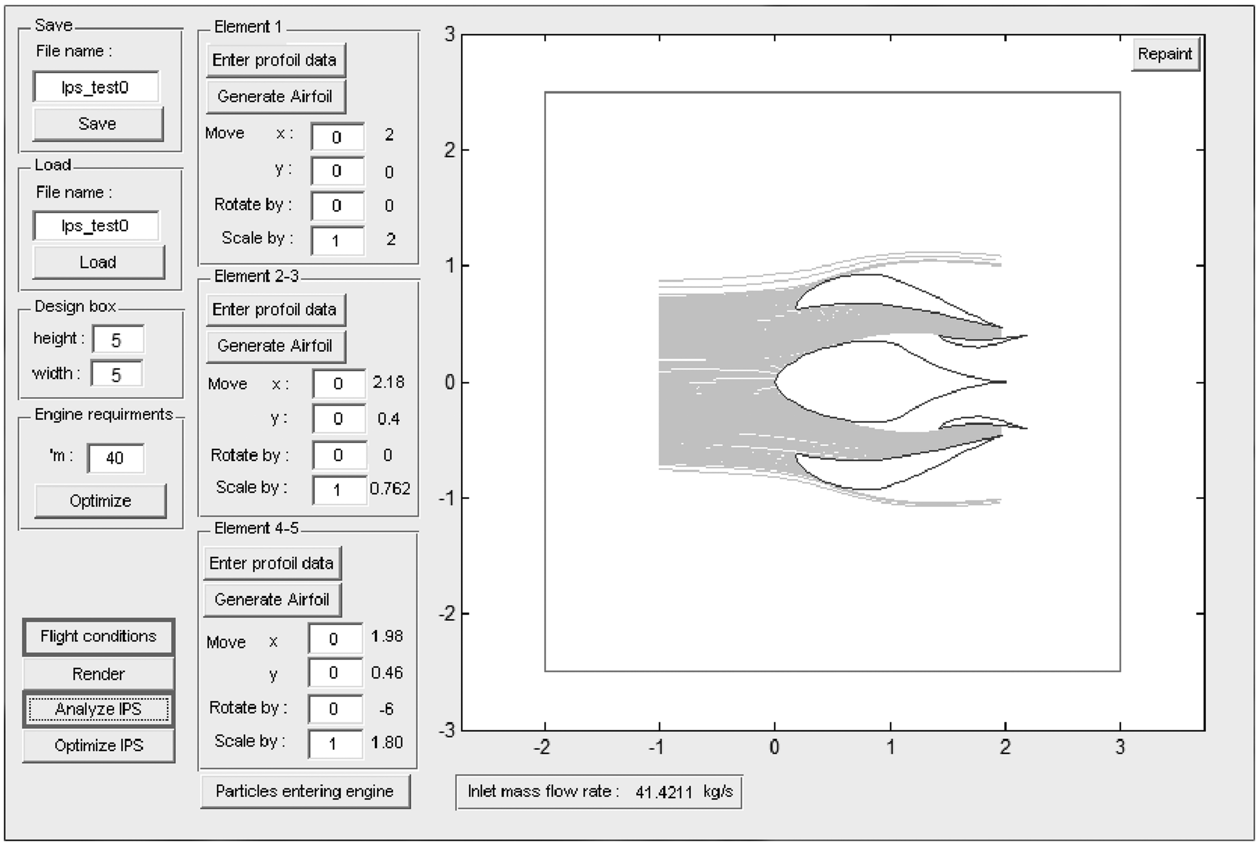
where $\boldsymbol{V}_a = u_a \boldsymbol{i} + w_a \boldsymbol{k}$. Note that this second-order differential equation is nonlinear because of the term $C_d Re/24K_a$, which depends on the particle position and the velocity. The difficulty in determining the term $C_d Re/24K_a$ suggests that a numerical technique must be employed to integrate the momentum equation (17). The fourth-order Runge–Kutta method [22] is used to integrate the preceding nonlinear equations.

The particle trajectories start at a distance of about five chord lengths (of the middle airfoil representing the engine centerline) and are calculated until they either impact any of the airfoil elements or go around them. The analysis code [3] locates the impingement point of the particle on any airfoil element surface by using a systematic search approach. When the particle is upstream of any airfoil element, no impact or impingement search is performed. Once the particle reaches the border or the bounding box around any of the elements along the x axis, a search is initiated that checks for any impingement on surface panels of all elements with the knowledge of the particle position (x, z).

This procedure is repeated for each particle released in the flow. The particle trajectories are initiated by releasing the particles from different upstream locations z_0 along the z axis (Fig. 8) while keeping a constant upstream distance of five chord lengths ($x_0 = -5c$). Thus, the impingement regions on individual elements are determined by an appropriate sweep of the z axis [3].

Figure 9 shows the flowchart of the numerical method employed in the IPS system analyzer. It can be seen from the flowchart that the sand type needs to be input. Because there is no single dust size that can represent the full range of sand and dust encountered in flight operations, the different sizes listed in Table 1 were defined as standard test dust [3]. Figure 10 shows sand-particle trajectories around an IPS system modeled as a five-element airfoil configuration.

Figure 11 [3] shows output of the analysis code obtained for an IPS system modeled as a five-element airfoil configuration, as in Fig. 2b. In this figure, the effect of the different type of sand particles on impingement trajectories can be observed. Here, only the diameter of the particle has been changed and the other parameters are kept constant. The effect of gravity on the particle trajectory becomes more apparent as the particle size is increased from very fine to coarse. Figures 11c and 11d show that for medium-to-coarse sand



Save
File name : Ips_test0

Load
File name : Ips_test0

Design box
height : 5
width : 5

Engine requirments
'm : 40

Element 1
Move x : 0 2
y : 0 0
Rotate by : 0 0
Scale by : 1 2

Element 2-3
Move x : 0 2.18
y : 0 0.4
Rotate by : 0 0
Scale by : 1 0.762

Element 4-5
Move x 0 1.98
y 0 0.46
Rotate by : 0 -6
Scale by : 1 1.80

Inlet mass flow rate : 41.4211 kg/s

Fig. 12 GUI showing IPS system and particle trajectories after an analysis run.

particles the flow circulation strength has very little effect on the particle trajectories and that the trajectories for this design are almost unaffected by inertia. This limitation can be overcome by a detailed study into the design of such arrangements. Another limitation of the analysis code is also clear from Fig. 11, in which sand-particle trajectories terminate upon hitting the surface. In reality, the sand particles bounce back from the impact and reenter the airstream [3].

B. Linking the Synthesis and Analysis Engines

The analysis engine is linked to the synthesis engine via the program GUI. In the GUI, two controls (namely, flight conditions and analyze IPS) are included that, as their name implies, facilitate the task of specifying the flight and ambient conditions and performing the analysis, respectively. The analysis engine outputs particle trajectory data that are then displayed in the GUI design box (Fig. 12). Figure 13 shows a 3-D rendering of the IPS system along with particle trajectories after an analysis run.

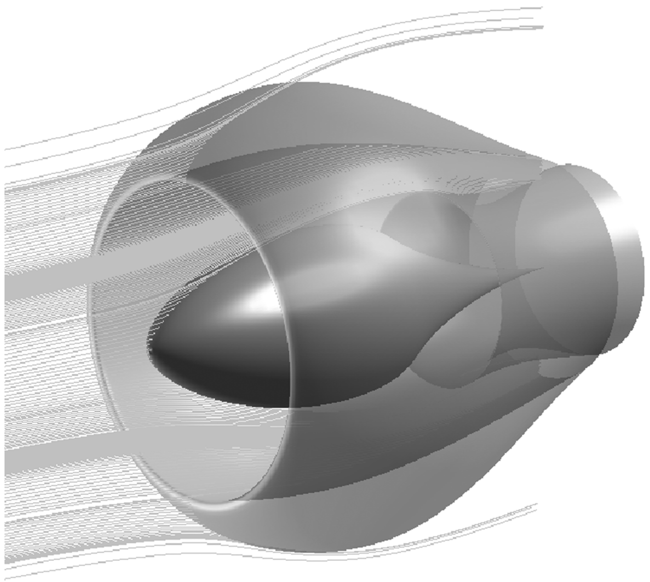


Fig. 13 A 3-D rendering of the IPS system showing particle trajectories.

V. Optimization Engine

A. Objective Function and Design Constraints

To carry out the optimization of the IPS system, an appropriate objective function must be defined, along with design constraints. The main design objective of the IPS system is to prevent sand ingestion into the engine without affecting the engine performance (inlet mass flow rate). A generated design is evaluated based on the

```
function objective = ips_ObjFun(X)
change positioning property 1 by (X(1))
..........................................
change positioning property n by (X(n))
change element shape property 1 to (X(n+1))
.........................................
change element shape property m to (X( n+m))
if geometry constraints are met
   Analyze IPS and output the mass flow rate
   if inlet mass flow rate requirement and trajectory analysis limitations are met
      objective= number of particle trajectories entering the engine inlet
   else
      objective=penalty
   end
else
   objective= penalty
end
change positioning property 1 by (-X(1))
..........................................
change positioning property n by (-X(n))
```

Fig. 14 Objective function algorithm.

Table 2 Three-variable optimization constraints and results

Element no.	Variable	Shape property	Lower bound	Upper bound	Unoptimized	Optimized
1	1	Thickness	20%	35%	25%	31.25%
2,3	2	Move in x direction	−0.2	0.5	0	−0.125
2,3	3	Move in y direction	−0.1	0.5	0	0
——	——	Mass flow rate, kg/s	27	——	26.3996	27.044

number of particle trajectories entering the engine, in which the objective is to minimize this number to zero and, at the same time, to constrain the inlet mass flow rate to a value very close to the engine operational mass flow rate without the IPS system installed. The optimization problem at hand involves a vast number of variables that all need to be linked to an objective function and simultaneously accounted for. Thus, to accomplish the design of an IPS system, a multivariable optimization scheme is adopted.

A computational objective function was implemented with function parameters (design variables) stored in a 1-D matrix $\boldsymbol{X}$. The design variables are in the form of *element positioning properties* and *element shape properties*. The element positioning properties are the design variables that affect the orientation of the elements within the IPS system. Such design variables are the translation, rotation, and scaling of any airfoil element. The element shape properties, on the other hand, are the design variables that affect the shape design of any airfoil element through PROFOIL. Such design variables are the thickness, pitching moment, and camber of any airfoil element.

The design problem has three types of constraints: geometric positioning constraints for each element, airfoil PROFOIL design constraints for each element, and the main engine inlet mass flow rate constraint.

There are two subtypes of geometric constraints. First, the IPS elements should not overlap (thickness between any two elements must be greater than zero). These constraints were implemented within the objective function. The value of the function is given a penalty (high value) if these constraints are not met. Second, each airfoil element should have a specified design box (x and y limits). These constraints were specified as inequality constraints (upper and lower bounds) within the optimizer.

Finally, the main constraint of the design problem, the operational inlet mass flow rate constraint, is a nonlinear constraint that was implemented inside the objective function. The value of the objective function is given a penalty (high value) if the inlet mass flow rate is not attained.

In each design iteration, the values of the design variables ($\boldsymbol{X}$ matrix) are chosen by the optimizer within the constraints and these values are input to the computational objective function. The function then uses these values to generate a new design. This design is then analyzed and the result of the analysis (particle trajectories) is searched for any trajectories entering the engine inlet (any trajectories flowing between the lower surface of element 2 and the upper surface of element 3). The number of these trajectories is used as the value of the objective function, which must be minimized to zero to ensure that no particles are entering the engine inlet. The optimizer uses the value of the objective function (number of entering particle trajectories) to evaluate a current design generation and make changes to the design variables accordingly. The design loop is repeated until the value of the objective function is minimized to zero. The objective function's general algorithm is given in Fig. 14.

B. Optimizer

After implementing an appropriate objective function, the next step was to choose a suitable optimization method. This method should have the ability to search the design space and minimize the given objective function. Because the objective function implemented is computational, the function is not differentiable. Methods that do not require any information about the gradient of the objective function need to be used. Moreover, because the objective function is nonlinear and noncontinuous and contains nonlinear constraints, the design space needs to be searched in a random manner so that the optimizer will not get locked in a local minimum. Such traits are found in derivative-free heuristic and constrained direct-search methods [23]. In contrast to the more traditional optimization methods that use information about the gradient or higher derivatives to search for an optimal point, direct-search methods search a set of

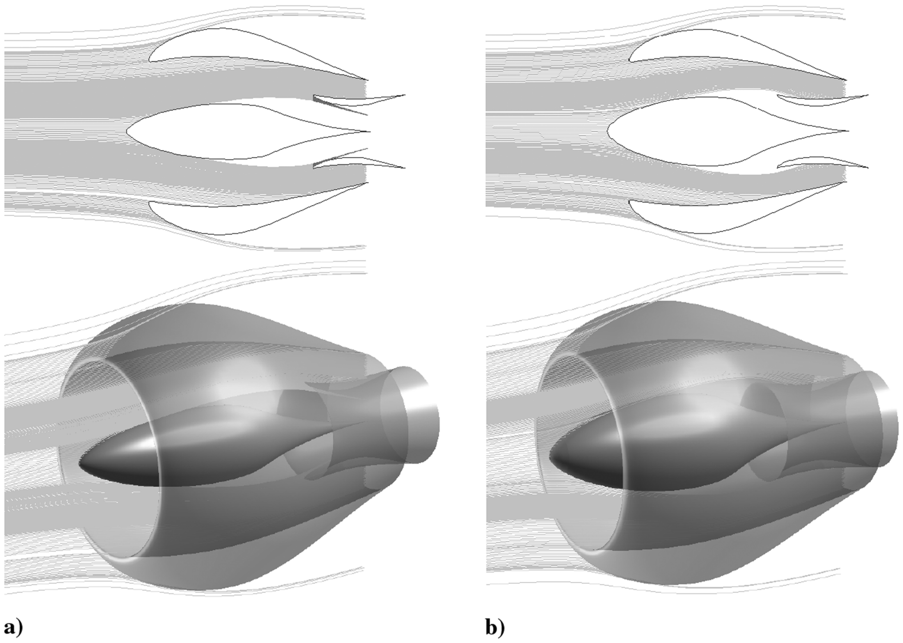

a) b)

Fig. 15 Three-variable optimization example: a) unoptimized case and b) optimized case.

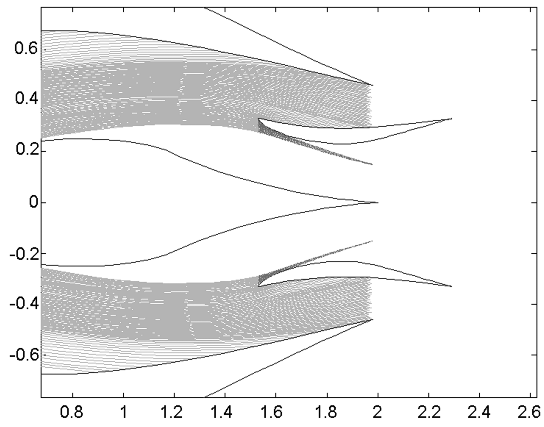


Fig. 16 Close-up view of unoptimized design.

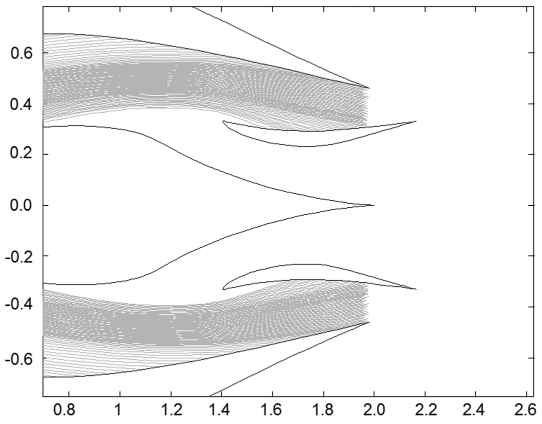


Fig. 17 Close-up view of optimized design.

points around the current point, looking for one in which the value of the objective function is lower than the value at the current point. The direct-search method can be used to solve problems for which the objective function is not differentiable, stochastic, or even discontinuous.

The pattern search algorithm from the Genetic Algorithm and Direct Search Toolbox in MATLAB was chosen as the optimizer for the IPS design [24]. Pattern search is a superior substitute to the genetic algorithm because it is usually computationally less expensive and can minimize the same types of functions.

The pattern search function works by searching a set of points called a pattern or mesh. This mesh expands or reduces in size, depending on whether any point within the pattern (mesh) has a lower objective function value than the current point. The search stops when a minimum pattern size (mesh size) is reached.

Similar to the genetic algorithm, the pattern search algorithm does not use derivatives to determine descent and works well on nondifferentiable, stochastic, and discontinuous objective functions. In addition, pattern search is very effective at finding a global minimum because of the way it searches the design space.

In the problem at hand, nonlinear constraints are present, which cannot be supplied to pattern search in the current version of the toolbox, which is why these constraints (the mass flow rate constraint in this case) were implemented inside the objective function as penalties if not satisfied.

Finally, the optimizer requires an initial guess (initial design) that is close to or inside the feasible region so that it would converge and would not get locked into a local minimum. To solve this problem, an additional program function is used that gives the designer the ability to optimize only for the inlet mass flow rate required. Use of this function has been found to yield a good initial design. The function moves elements 2 and 3 in the y direction, changing the inlet area, and analyzes the IPS for the inlet mass flow rate, resulting in a design with an inlet mass flow rate very close to that required by the engine and designer.

VI. Design Examples

The design examples that follow were performed on a computer with a 3 GHz Intel Core 2 Extreme 6800, with 2 GB 800 MHz RAM. In all of the problems, the angle of attack was kept at 0 deg and the freestream velocity was set to 40 m/s. The type of sand particles was set to very fine.

A. Example 1: A Three-Variable Optimization Problem

In this example, three variables were optimized and were given upper and lower bounds, as shown in Table 2. Figure 15a shows the unoptimized design in which sand-particle trajectories are hitting the lower surface of element 2 and the upper surface of element 3 as they enter inside the engine. A close-up view of the inlet (elements 2 and 3) of the unoptimized design (Fig. 16) shows the impinging trajectories more clearly. Only the variables given in Table 2 were optimized; all other variables and shape properties were kept constant. The mass flow rate of the unoptimized design is 26.3996 kg/s and the unoptimized design variable values are given in Table 2. The mass flow rate engine requirement was set to a value of 27 kg/s.

The computational time for the optimization run was 25 min. After optimization, the objective function was minimized to zero. Figure 15b shows the optimized IPS design. A close-up view of the engine inlet (elements 2 and 3) of the optimized design (Fig. 17) shows clearly that the particle trajectories clear the engine. The mass flow rate of the optimized design is 27.044 kg/s, which is the required mass flow rate. The optimized design variable values are given in Table 2.

B. Example 2: A Nine-Variable Optimization Problem

In this example, nine variables were optimized and were given upper and lower bounds, as shown in Table 3. Figure 18a shows the unoptimized design, in which sand-particle trajectories are hitting the lower surface of element 2 and the upper surface of element 3 as they

Table 3 Nine-variable optimization constraints and results

Element	Variable	Shape property	Lower bound	Upper bound	Unoptimized	Optimized
1	1	Thickness	20%	35%	35%	34.063%
2,3	2	Camber	5%	15%	10%	5%
2,3	3	Thickness	8%	20%	8%	14.25%
2,3	4	Pitching moment coefficient	−0.3	−0.16	−0.18	−0.18
2,3	5	Move in x direction	−0.2	0.5	0	−0.063
2,3	6	Move in y direction	−0.1	0.5	0	−0.016
4,5	7	Camber	5%	20%	10%	6.5625%
4,5	8	Thickness	10%	20%	14%	19.375%
4,5	9	Pitching moment coefficient	−0.35	−0.2	−0.25	−0.2
——	——	Mass flow rate, kg/s	37	42	41.2516	39.2255

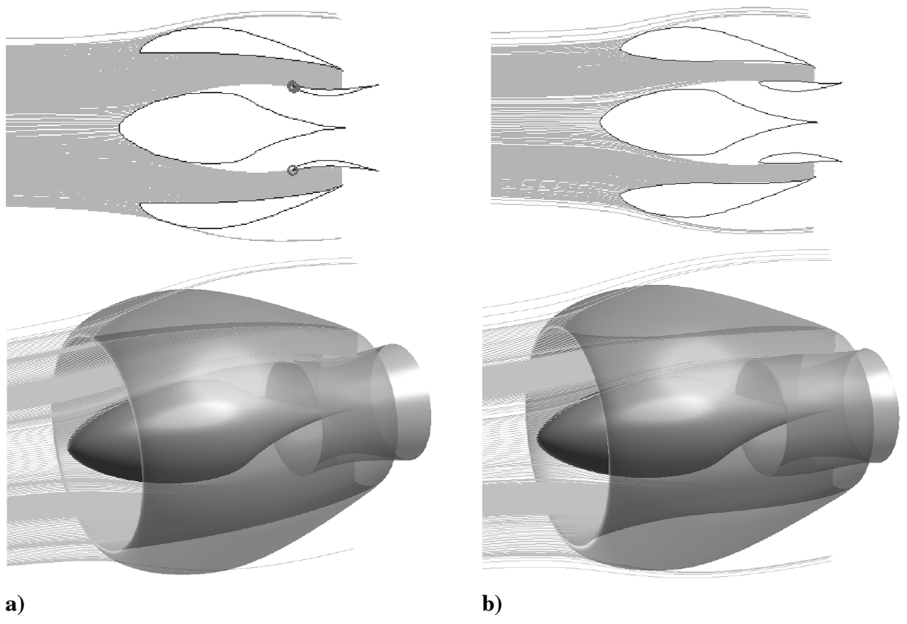

a) b)

Fig. 18 Nine-variable optimization example: a) unoptimized case and b) optimized case.

enter inside the engine. A close-up view of the inlet (elements 2 and 3) of the unoptimized design (Fig. 19) shows the impinging trajectories more clearly. Only the variables given in Table 3 were optimized; all other variables and shape properties were kept constant. The mass flow rate of the unoptimized design is 41.25 kg/s and the unoptimized design variable values are given in Table 3. The mass flow rate engine requirement was set to a range between 37–42 kg/s.

The computational time for the optimization run was 3 h and 30 min. After optimization, the objective function was minimized to zero. Figure 18b shows the optimized IPS design. A close-up view of element 2 of the optimized design (Fig. 20) shows clearly that the particle trajectories clear the engine inlet and only hit the upper surface of element 2. The mass flow rate of the optimized design is 39.2255 kg/s which is within the required mass flow rate range. The optimized design variable values are given in Table 3.

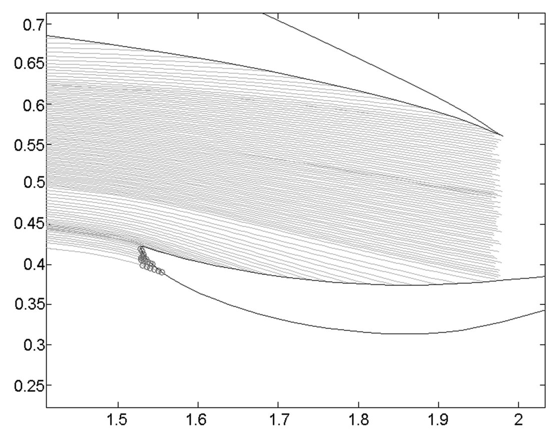


Fig. 19 Close-up view of element 2 of the unoptimized design.

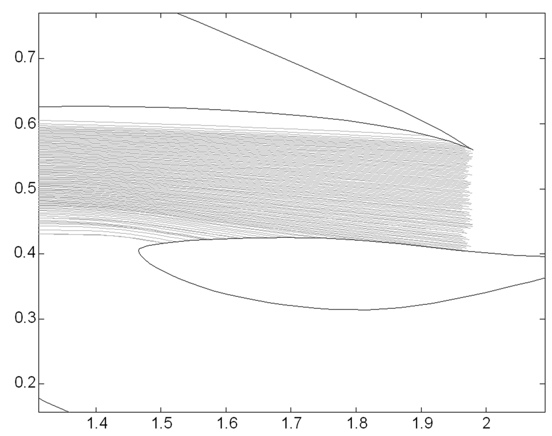


Fig. 20 Close-up view of element 2 of the optimized design.

VII. Conclusions

The paper presents a method for the successful design and optimization of a multi-element airfoil-based inertial particle separator (IPS). A MATLAB graphical user interface (GUI) was developed to facilitate design in an interactive manner. The method has the capability to 1) design individual airfoil elements; 2) orient and arrange airfoils to form a multi-element airfoil-based IPS system by employing translational, scaling, and rotational functions; 3) carry out 2-D flow and trajectory analysis of multi-element airfoil-based IPS system; and 4) perform design optimization using pattern search: a genetic algorithm-based optimization technique. The design of individual airfoils is achieved through the use of the PROFOIL code, a state-of-the-art multipoint inverse airfoil design program. Multiple airfoils are arranged to form a multi-element airfoil-based IPS system subject to geometric constraints. A 2-D analysis tool for the multi-element airfoil is linked to the synthesis component of the GUI to carry out flow and particle trajectory analysis component of the program. For the optimization component of the program, a computational objective function was successfully implemented and coupled with a pattern search optimizer located in the Genetic Algorithm and Direct Search Toolbox in MATLAB. Design and optimization were achieved using multivariable optimization. Two design examples are used to illustrate and demonstrate the design method.

Although the method presented can be used for the design of an IPS system, it has certain limitations that need to be addressed to make it very versatile and robust. These limitations are as follows:

1) The present method assumes that the flow is two-dimensional. In reality, the flow has a swirl component that can affect sand-particle trajectories and therefore result in sand ingestion. A proper treatment of the problem must consider the effect of swirl.

2) In the present method, the particle trajectory calculation is terminated if a particle strikes a surface. In reality, particles may rebound from the surface and enter the flow stream. In such cases, particles may enter the engine after rebound. Thus, it becomes imperative that the particle rebound characteristics be accounted for in the analysis and design of an IPS system.

3) The current method also neglects viscous effects, and therefore the effect of flow separation from sharp bends in the flow is absent in the analysis and design. A more accurate analysis and design method must also account for the effects of viscous boundary-layer development and flow separation.

Acknowledgments

The authors would like to acknowledge the support of King Abdulaziz City for Science and Technology (KACST) and Deanship of Scientific Research of King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, in accomplishing this study. The authors are also grateful to Michael Selig for the use of his code PROFOIL in the study.

References

[1] Tabakoff, W., and Hamed, A., "Installed Engine Performance in Dust Laden Atmosphere," AIAA Paper 1984-2488, 1984.

[2] van der Walt, J. P., and Nurick, A., "Prediction of Helicopter Engines Fitted with Dust Filters," *Journal of Aircraft*, Vol. 32, No. 1, Jan.–Feb. 1995, pp. 118–123. doi:10.2514/3.46691

[3] Saeed, F., and Al-Garni, A. Z., "Analysis Method for Inertial Particle Separator," *Journal of Aircraft*, Vol. 44, No. 4, July–Aug. 2007, pp. 1150–1158. doi:10.2514/1.20245

[4] Mann, D., "Case Studies in TRIZ: Helicopter Engine Particle Separator," *The TRIZ Journal*, Feb. 1999.

[5] Vittal, B. V. R., Tipton, D. L., and Bennett, W. A., "Development of an Advanced Vaneless Inlet Particle Separator for Helicopter Engines," *Journal of Propulsion and Power*, Vol. 2, No. 5, 1986, pp. 438–444. doi:10.2514/3.22926

[6] Breitman, D. S., Dueck, E. G., and Habashi, W. G., "Analysis of a Split-Flow Inertial Particle Separator by Finite Elements," *Journal of Aircraft*, Vol. 22, No. 2, 1985, pp. 135–140. doi:10.2514/3.45097

[7] Zedan, M., Mostafa, A., Hartman, P., and Sehra, A., "Viscous Flow Analysis of Advanced Particle Separators," *Journal of Propulsion and Power*, Vol. 8, No. 4, 1992, pp. 843–848. doi:10.2514/3.23558

[8] Selig, M., S., "PROFOIL—A Multipoint Inverse Airfoil Design Method, User's Guide," Published by the author, Urbana–Champaign, IL, Feb. 1999.

[9] Selig, M. S., and Maughmer, M., "Multipoint Inverse Airfoil Design Method Based on Conformal Mapping," *AIAA Journal*, Vol. 30, No. 5, May 1992, pp. 1162–1170. doi:10.2514/3.11046

[10] Saeed, F., and Selig, M. S., "A Multipoint Inverse Airfoil Design Method for Slot-Suction Airfoils," *Journal of Aircraft*, Vol. 33, No. 4, July–Aug. 1996, pp. 708–715. doi:10.2514/3.47005

[11] Tuncer, C., *An Engineering Approach to the Calculation of Aerodynamic Flows*, Horizons, Long Beach, CA, 1999.

[12] Saeed, F., Brette, C., Fregeau, M., Trifu, O., and Paraschivoiu, I., "A Three-Dimensional Water Droplet Trajectory and Impingement Analysis Program," AIAA Paper 2005-4838, June 2005.

[13] Ruff, G. A., and Berkowitz, B. M., "Users Manual for the NASA Lewis Ice Accretion Prediction Code (LEWICE)," NASA CR-185129, May 1990.

[14] Saeed, F., "State-of-the-Art Aircraft Icing and Anti-Icing Simulation," *ARA Journal*, Vol. 2000–2002, No. 25–27, June 2002, pp. 106–113.

[15] Langmuir, I., and Blodgett, K. B., "A Mathematical Investigation of Water Droplet Trajectories," U.S. Army Air Forces Headquarters, TR 5418, Feb. 1946; also U.S. Dept. of Commerce Publication Board, Rept. 27565, 1946.

[16] Bragg, M. B., "A Similarity Analysis of the Droplet Trajectory Equation," *AIAA Journal*, Vol. 20, No. 12, Dec. 1982, pp. 1681–1686. doi:10.2514/3.8004

[17] Wells, S. L., and Bragg, M. B., "A Computational Method for Calculating Droplet Trajectories Including the Effects of Wind Tunnel Walls," AIAA Paper 92-0642, Jan. 1992.

[18] Bergrun, Norman, R., "A Method for Numerically Calculating the Area and Distribution of Water Impingement on the Leading edge of an Airfoil in a Cloud," NACA TN 1397, Aug. 1947.

[19] Tran, P., Brahimi, M. T., Paraschivoiu, I., Pueyo, A., and Tezok, F., "Ice Accretion on Aircraft Wings with Thermodynamic Effects," *Journal of Aircraft*, Vol. 32, No. 2, March–April 1995, pp. 444–446. doi:10.2514/3.46737

[20] Tran, P., Brahimi, M. T., and Paraschivoiu, I., "Ice Accretion on Aircraft Wings," *Canadian Aeronautics and Space Journal*, Vol. 40, No. 3, Sept. 1994, pp. 91–98.

[21] Paraschivoiu, I., Tran, P., and Brahmi, M. T., "Prediction of Ice Accretion with Viscous Effects on Aircraft Wings," *Journal of Aircraft*, Vol. 31, No. 4, July–Aug. 1994, pp. 855–861. doi:10.2514/3.46571

[22] Fehlberg, E., "Classical Eighth- and Lower-Order Runge-Kutta-Nyström Formulas with a New Stepsize Control Procedure for Special Second-Order Differential Equations," NASA TR R-381, March 1972.

[23] Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M., *Engineering Optimization: Methods and Applications*, Wiley-Interscience, New York, 1983.

[24] MATLAB, Software Package, Ver. 7.0, The MathWorks, Inc., Natick, MA, 2008.