JOURNAL OF THEORETICAL AND APPLIED MECHANICS
57, 3, pp. 697-711, Warsaw 2019
DOI: 10.15632/jtam-pl/110130

INVERSE AND DIRECT OPTIMIZATION SHAPE OF AIRFOIL USING HYBRID ALGORITHM BIG BANG-BIG CRUNCH AND PARTICLE SWARM OPTIMIZATION

Heidar Masoumi
Engineering Department, Khoy, Iran
e-mail: hm.masoumi65@gmail.com

Farhad Jalili
Azad University, Ajabshir, Iran

In this paper, Big Bang-Big Crunch and Particle Swarm Optimization algorithms are combined and used for the first time to optimize airfoil geometry as a aerodynamic cross section. The optimization process is carried out both in reverse and direct directions. In the reverse approach, the object function is the difference between pressure coefficients of the optimized and target airfoils, which must be minimized. In the direct approach, three objective functions are introduced, the first of which is the drag to lift (D/L) ratio. It is minimized considering four different initial geometries, ultimately, all four geometries converge to the same final geometry. In other cases, maximizing lift the coefficient with the fixed drag coefficient constraint and minimizing the drag coefficient while the lift coefficient is fixed are defined as purposes. The results show that by changing the design parameters of the initial airfoil geometry, the proposed hybrid optimization algorithm as a powerful method satisfies the needs with proper accuracy and finally reaches the desired geometry.

Keywords: hybrid optimization algorithm, airfoil, inverse and direct optimization approaches, Euler's equations

1. Introduction

Airfoils are one of the important aerodynamic cross-sections that have always attracted researchers' attention. An airfoil is a body in which a difference is created between the pressure of its lower and upper parts as the air flows around it. Airfoil features are very important not only in airplane wings but also in ships, helicopters, compressors, turbines, fans, pumps, and other equipment. The first efforts for developing airfoil cross-sections were made in late 1800s. In 1939, Jacobs who was a scientist in NASA designed and tested the first layered-flow airfoil cross-sections. Most of these works were done using the trial and error technique and were time--consuming and expensive. The calculation burden of initial optimization approaches increased with the adding in number of variables, therefore, they were not cost-effective at all. With the advent of Computation Fluid Dynamics (CFD) as a strong tool for performing fluid dynamics calculations, a revolution took place in designing airfoils and wings. Using CFD, the wing of interest can be fed as an input to a viscous or inviscid solver, and its drag or lift can be calculated with lower error as compared to conventional approaches. This approach not only speeds up the operation but also is more accurate and less expensive (Zhang and Lum, 2006).

Optimization approaches can be divided into two categories: gradient-based and heuristic approaches (non-gradient). In the first category, gradients of the objective function with respect to design variables play a key role in the optimization process. The conjugate gradient and Levenberg-Marquart approaches belong to this category. In the second category, no information

about the gradients of the objective function with respect to design variables is needed during the optimization process, and the search for the optimum point is performed by comparing the values of the objective function at different points. Random search algorithms such as genetic algorithms and particle swarm otimization belong to this category.

Gradient-based approaches are faster than heuristic methods provided that a proper initial starting point is selected. However, these approaches are highly dependent on starting point. In these methods, the objective function is a continuous and differentiable function of design variables and they are costly when the objective function is complicated or the number of design variables is high. But heuristic approaches only need the objective function and are suitable for complicated functions (there is no need for differentiating), are not dependent on the starting point and can be applied to discontinuous functions as well. These approaches have no problems with a high number of design variables and do not use complicated mathematical equations. However, these approaches have a disadvantage that there is no guarantee for global convergence (Zhang and Lum, 2006; Lane and Marshall, 2009).

1.1. A review on previous works

Researches have been conducted on optimization of aerodynamic cross-sections for two decades continuously. Wauquiez (2000) investigated optimization of low-speed airfoils geometry by using MATLAB software and an automatic differentiating method. Giannakoglou (2002) performed optimal designing of aerodynamic geometries using stochastic approaches and computational intelligence. Khurana (2008) used a combination of Particle Swarm Optimization and neural network to design long wings. Hájek (2009) dealt with optimization of aerodynamic airfoils and blades in two and three dimensions. Jahangirian and Shahrokhi (2009) dealt with airfoil optimization through inverse optimization. In this research, a genetic algorithm had been used as the optimization algorithm. Wickramasinghe *et al.* (2010) addressed the problem of designing airfoils by using reference points based on a multi-objective particle swarm algorithm in the subsonic mode. Lane and Marshall (2010) studied airfoils by using an inverse optimization approach. Endo (20111) used the bee colony optimization algorithm to investigate airfoils of wind turbines. Xu and Xia (2016) dealt with direct and inverse optimization of airfoils using a continuous adjoint method and Euler equations. Alves *et al.* (2016) conducted airfoil shape optimization using Parces and Particle Swarm Optimization (PSO) methods. In 2017, by doing some modifications and by using a genetic algorithm and Parces approach, Ebrahimi and Jahangirian (2017) investigated airfoil geometry optimization.

One of the other heuristic optimization approaches is the Big Bang-Big Crunch algorithm introduced by Erol and Eksin (2006), which is inspired by the creation of Universe with of a massive explosion. An investigation of previous works shows that Big Bang-Big Crunch algorithm has never been used in the areodynamic field and it has never been combined with the PSO algorithm. Therefore, optimization of airfoils has not been performed by using the hybrid algorithm presented in this paper. In this research, two algorithms are combined with each other for the first time to achieve the optimum airfoil geometry in both inverse and direct aproaches.

Any act of fluid optimization needs geometry, flow solver and an optimization algorithm by merging which the designers try to reach the optimum objective of interest. In the process of solving a problem, the first these three factors are introduced one by one, then by incorporating them an effort is made to achieve the purpose of interest.

2. Airfoil shape parametrization

There are different methods for airfoil geometry parameterization with different numbers of parameters like: bezier curves, parces approach and NACA four-digit formula. The geometry

usually becomes more flexible as the number of design parameters increases. The approach used in this paper is the NACA four-digit formula. Three parameters used for airfoil geometry parameterization are the maximum camber m, maximum camber location, and finally the maximum thickness of the airfoil τ (Wauquiez, 2000). These parameters are shown in Fig. 1.

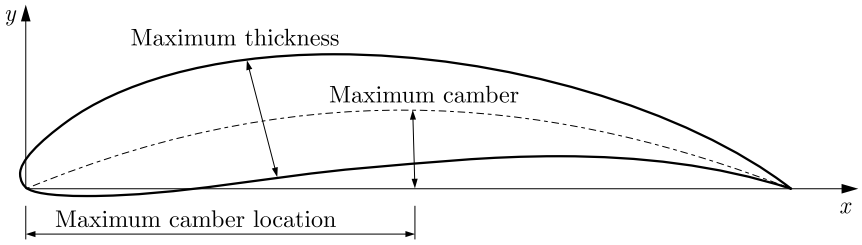

Fig. 1. Parameters used for airfoil geometry parameterization in NACA four-digit formula

However, as it is clear from its name, in this approach the airfoil surface is introduced by means of certain equations. For the upper and lower surfaces, these equations are as follows

$$\begin{aligned} X_{up} &= x - y_{th}\sin\theta \qquad & Y_{up} &= y_c + y_{th}\cos\theta \\ X_{lo} &= x + y_{th}\sin\theta \qquad & Y_{lo} &= y_c - y_{th}\cos\theta \end{aligned} \tag{2.1}$$

where y_{th} is the thickness distribution and y_c is the camber line, which are represented by the following eqution

$$y_{th} = 5\tau c\Big[0.2969\sqrt{\frac{x}{c}} - 0.126\frac{x}{c} - 0.3537\Big(\frac{x}{c}\Big)^2 + 0.2843\Big(\frac{x}{c}\Big)^3 - 0.1015\Big(\frac{x}{c}\Big)^4\Big]$$

$$y_c = \begin{cases} \frac{m}{p^2}\Big[2p\frac{x}{c} - \Big(\frac{x}{c}\Big)^2\Big] & \text{for} \quad \frac{x}{c} \leqslant p \\ \frac{m}{(1-p)^2}\Big[1 - 2p + 2p\frac{x}{c} - \Big(\frac{x}{c}\Big)^2\Big] & \text{for} \quad \frac{x}{c} \geqslant p \end{cases} \tag{2.2}$$

According to Fig. 2, θ depends on the slope of the camber at each point, and it can be calculated by taking the derivative of Eq. $(2.2)_2$. This parameter is represented by the following equation

$$\theta(x) = \tan^{-1}\frac{dy_c(x)}{dx} \tag{2.3}$$

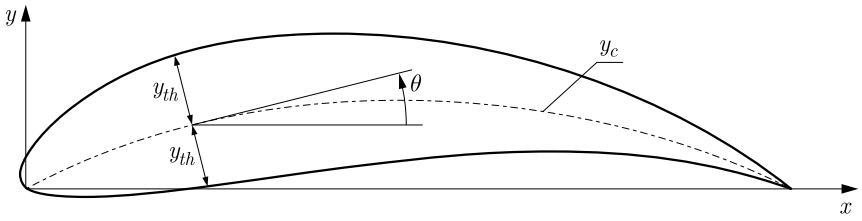

Fig. 2. The camber and its slope

3. Airflow around the airfoil

Once the airfoil geometry is formed, a circular space is formed around it, then the area between the airfoil and this circular area is meshed. Meshing is asymmetric and the number of points creating the geometry is high at the beginning and end of the airfoil due to the higher pressure gradient compared to the middle points. The flow solver must solve the airflow around the airfoil and present the pressure distribution around it. To solve this flow, after defining the relationships governing the flow, these equations must be discretized and solved.

4. Governing equations

Equations governing the inviscid flow are compressible time-dependent Euler's equations. The conservative forms of these equations in two-dimensional coordinates are as follows

$$\frac{\partial \mathbf{W}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x} + \frac{\partial \mathbf{G}_i}{\partial y} = \mathbf{0} \tag{4.1}$$

where $\mathbf{W}$ is the vector of flow quantities

$$\mathbf{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix} \tag{4.2}$$

Vectors $\mathbf{F}_i$ and $\mathbf{G}_i$ are inviscid flux vectors defined as follows

$$\mathbf{F}_i = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E+p)u \end{bmatrix} \qquad \mathbf{G}_i = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E+p)v \end{bmatrix} \tag{4.3}$$

In the above equations, u and v are velocity components in the x and y directions, respectively, p is the static pressure, and E is the inner energy of the fluid. The governing equations are partial differential equations, which are solved by using the discretized finite volume approach through the four-step explicit Runge-Kutta method introduced by Jameson *et al.* (1981).

5. Optimization algorithm

In this paper, a hybrid optimization algorithm consisting of the Big Bang-Big Crunch (BB-BC) and Particle Swarm Optimization (PSO) algorithms is used for optimization of airfoil aerodynamic cross section. First, both the algorithms are explained in details. Then, the way they are combined to obtain a new algorithm with better performance is presented.

5.1. The Big Bang-Big Crunch algorithm

This algorithm is one the meta-heuristic algorithms inspired by nature, which was introduced for the first time by Erol and Eksin in the University of Istanbul in 2005. The basis of this algorithm is the theory of initial Big Bang-Big Crunch of the Universe. This theory states that due to a massive initial explosion, a large amount of energy has been released and then this energy has been concentrated, and the current Universe has been formed. This algorithm consists of two phases, the first phase is the big bang and the second one is the big crunch. In the first phase, individuals of the initial population (energy particles) are distributed randomly in the search space, then in the second phase, these particles are attracted towards a new certain particle. In fact, this optimization algorithm creates some points in an irregular way in the big bang phase then, in the big crunch phase, these points are moved towards a single point known as the center of mass. After several steps of random distribution of particles in the search space, this space becomes smaller and smaller until it reaches the calculated mean point in the big crunch stage, and these steps are repeated until the problem converges to the solution of interest. By carrying out operations which include several input stages (BB stage) and a single output stage

(BC stage), the big bang big crunch algorithm converges to the solution that is the center of mass. The center of mass is represented by $x_i^c(k)$ and is calculated using the following equation

$$x_i^{c(k)} = \frac{\sum_{j=1}^{n} \frac{x_i^{(k,j)}}{f_j}}{\sum_{j=1}^{n} \frac{1}{f_j}} \tag{5.1}$$

where $x_i^{(k,j)}$ is the i-th component of j-th solution generated at k-th iteration, and n is the population size in big bang phase and f_i is the value of the objective function. After the big crunch phase, the algorithm creates new solutions which are used in the next phase of the big bang stage. This is done by using the previous information which is the center of mass. In fact, this is done by covering a new spring around the center of mass through natural distribution at each direction until the standard deviation from this natural distribution function decreases as the number of iterations of this algorithm increases. $x_i^{c(k)}$ created in the big crunch stage is used for generating new particles in the next iteration. The new particles around the center of mass are calculated using the following equation

$$x_i^{(k+1,j)} = x_i^{c(k)} + \frac{r_j \alpha_1 (x_{max,i} - x_{min,i})}{k+1} \tag{5.2}$$

where r_j is a random number with standard normal distribution that varies for each candidate solution and α_1 is a parameter for limiting the size of search space which is usually within $[0, 1]$. but it can be selected within $[-1, 1]$ in order to place new solution around the center of mass (Erol and Eksin, 2006).

Five steps can be considered for the Big Bang-Big Crunch algorithm:

- The first step: Randomly and irregularly generating the initial population (candidate solutions) within the search space (design parameters).
- The second step: Calculating the objective function for the generated initial population.
- The third step: Finding the center of mass of the scattered population by using Equation (5.1).
- The fourth step: Calculating new solutions around the center of mass using Equation (5.2).
- The fifth step: Repeating the above steps from step 2 until the stopping criterion is met (Kumbasar *et al.*, 2008).

5.2. Particle Swarm Optimization

This algorithm is inspired by the natural behavior of birds in search for food, which was introduced for the first time by Eberhart and Kennedy in USA in 1995. This algorithm was developed based on using collective intelligence for finding the shortest path to reach the objective of interest, i.e. food. In the PSO algorithm, a certain number of birds, which are called particles, are randomly distributed within the search space. For each particle, the value of the objective function is calculated for the position of that particle in the space. Then, it selects a direction for movement by combining the data of its current position with the best position that it has previously been there, as well as using the data of one or more best particles existing in the colony. All particles select a direction for movement and after moving, a stage finishes. These steps are repeated several times until it converges to the optimum solution. In fact, the particle swarm which searches for the minimum or maximum of a function acts like a flock of birds searching for food. In the beginning, a position and a velocity vector is assigned to each bird, which are randomly distributed within the search space. Then, the value of the objective function is calculated for each particle, and the initial position of each particle is assumed its best position and is represented by *lb*.

In addition, the best particle in the memory, which is represented by *gb*, is obtained so that in the next iteration these values are updated and their new and modified values are calculated. The values of the velocity and position of particles in the next iteration are calculated using the quantities calculated in the previous iteration, as shown by the following equations

$$\begin{aligned} v_j^i[k+1] &= wv_j^i[k] + c_1 r_1 \left(x_j^{i,lb}[k] - x_j^i[k]\right) + c_2 r_2 \left(x_j^{gb}[k] - x_j^i[k]\right) \\ x_j^i[k+1] &= x_j^i[k] + v_j^i[k+1] \end{aligned} \tag{5.3}$$

For the new memory of objective functions, the best position of each particle and the best particle existing in the memory are calculated so that more updated cases can be obtained by comparing them. In the equation above, w is the inertia coefficient, r_1 and r_2 are random numbers with a uniform distribution within $[0, 1]$. c_1 is the learning coefficient of personal experience of each particle and c_2 is the learning coefficient of the colony. Proper selection of these three variables w, c_1 and c_2 has a great effect on the convergence of the problem. As times goes by, the particles will tend towards those particles that have better objective function values until, finally, the best particle is introduced along with its objective function (Kennedy and Eberhart, 1995).

5.3. The hybrid algorithm

Any optimization algorithm has its own shortcomings, some of them cause delay in convergence, and some of them converge to a wrong solution. The most common drawback in optimization algorithms is entrapment in local optima, which may be encountered in some problems. In BB-BC algorithm, the initial population might be created in a way that concentration of particles is higher in one side while the right solution is located far way. Since in this approach the center of mass of distributed particles is calculated, and new solutions are calculated in a distance around it, it might not converge to the solution of interest and might get stuck in local optima. By combining this algorithm, with the PSO, its capabilities are used and the search process is performed in the entire memory space, and the probability of getting stuck in local optima decreases significantly. In fact, by mergring these two algorithms, in addition to using their advantages, their disadvantages are compensated such that by using three important features, namely the center of mass of particles, the best position that each particle has been so far there, and the best particle among all particles, not only it gets far from local optima, but also converges to solutions with higher accuracy with a smaller number of iterations compared to PSO and BB-BC algorithms. The idea of this hybrid algorithm is taken from the hybrid BB-BC algorithm introduced by Kaveh and Talatahari (2010b). Their algorithm was only a modified variant of the original BB-BC algorithm. They were inspired in generating the modified population by the PSO, such that the candidate population had three steps at each iteration, and to understand the concepts of exploration (search) and exploitation, three terms of self-adaption, cooperation and competition were introduced. In the self-adaption stage, each particle improves its performance. In the cooperation stage, by transferring data, memory members cooperate with each other, and in the competition stage, the memory members try to survive by competing with each other. In the BB-BC algorithm, although the cooperation stage is satisfied using the center of mass concept, other stages are not carried out sufficiently. Therefore, by adding the PSO algorithm, self-modification or *lb* term (each particle improves its position) and competition to survive, i.e. *gb* (the fittest particle among all) are entered into the problem so that by using the above three concepts, a new memory is formed. The new memory in the hybrid algorithm of Kaveh is calculated using the following equation (Kaveh and Talatahari, 2010a,b)

$$x_i^{(k+1,j)} = \alpha_2 x_i^{c(k)} + \frac{r_j \alpha_1 (x_{max} - x_{min})}{k+1} + (1-\alpha_2)\left(\alpha_3 x_i^{(gb,k)} + (1-\alpha_3) x_i^{(lb,k)}\right) \tag{5.4}$$

In the present research, for the first time the BB-BC and PSO algorithms are combined with each other to be used for optimization of aerodynamic cross sections. The hybrid algorithm begins the optimization process with the PSO and, as the number of iterations increases, the effect of PSO decreases and the effect of BB-BC increases. Different steps of this hybrid algorithm are as follows:

- The first step: the initial population, velocity values, the best particle and the best position of each particle (which is the same as the position of particles at the iteration zero) are calculated to be run at the next iteration of the program where the next loops start.
- The second step: velocities and positions are modified using Equations $(5.3)_1$ and $(5.3)_2$. The value of w decreases linearly as the number of iterations increases (due to starting from a global search and converging to local optima).
- The third step: the values of objective functions are calculated for the modified population, and the best position of each particle and the best particle in the population are modified.
- The fourth step: using this modified population, the center of mass of particles is obtained.
- The fifth step: using Equation (5.4), a new modified population is calculated. For this calculated population, the values of the objective function are not calculated in the initial iterations, but this population is put in Equation $(5.3)_2$ and is used in the next iteration as the values of the previous population. This cycle continues until the convergence condition is met.

To prevent the modified populations from exceeding the ranges of defined variables, the initial velocities are considered to be equal to 1/150 of the positions range, and the velocity values used in Equation $(5.3)_2$ are considered to be $4/5w$ of the modified velocity of each particle (i.e. velocities used in Equation $(5.3)_2$ are equal to $4/5w$ of the velocities calculated from Equation $(5.3)_1$) so that as the number of iterations increases, the modified velocity decreases and the effect of the population modified by Equation (5.4).

In this hybrid algorithm, thw population of particles is updated twice during each iteration, once with the PSO algorithm and second time with the Kaveh Big-Bang Crunch algorithm. This method reduces the number of iterations and increases the speed of convergence. Also, for this algorithm to start from a global search and converge to desired answer, modified speeds are considered to be descending. Descending speed causes the effect of the PSO algorithm to decrease and the effect of the Big-Bang Crunch to increase over time in a way that new populations can be further modified by Equation (5.4). In the proposed hybrid algorithm, the values of parameters are as follows: $\alpha_1 = 0.001$, $\alpha_2 = 0.008$, $\alpha_3 = 0.01$, $c_1 = 0.6$, $c_2 = 0.6$ and $w = (kk - k)/kk$ (k is the number of iterations in one step and kk is the total number of iterations). Using the proposed algorithm in the present work reduces the number of iterations considerably.

6. Results

6.1. Validation of the optimization algorithm

At first, the presented optimization algorithm is used to solve the wood 4-variable function which is a famous benchmark. Then the results are compared with the Harmony Search (HS) algorithm (Lee and Greem, 2005)

$$f(x_1, x_2, x_3, x_4) = 100(x_1 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1) \tag{6.1}$$

Solving the minimization problem is aimed at minimizing the function $f(x_i)$ with the variable range of $-5 < x_i < 5$. It is clear that the minimum value of the function, which is equal to zero,

is obtained for the variable values of 1. The Harmony Search algorithm presented 20 different solutions (for 20 different initializations) and led to the best solution. The best solution in 70000 iterations converged to the following values: $x_1 = 1.0000349$, $x_2 = 1.0000702$, $x_3 = 0.9999628$, $x_4 = 0.9999260$ with $f(x_1, x_2, x_3, x_4) = 4.8515 \cdot 10^{-9}$.

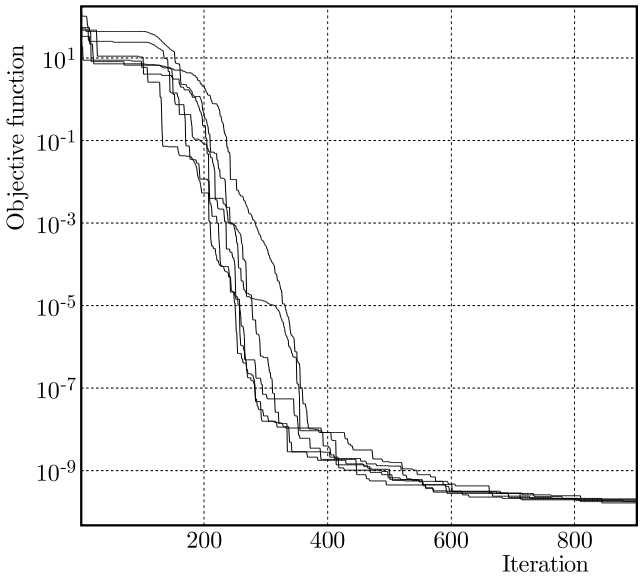

Fig. 3. Convergence behavior of the wood function resulting from the proposed algorithm for several runs

As Table 1, the shows presented hybrid algorithm convergences with 800 iterations. The number of iterations is much lower than in the Harmony Search while the results have higher accuracy, too. The higher accuracy is accompanied by a lower iteration number. These two important factors the hybrid BB-BC, PSO algorithms are satisfactorily provided.

Table 1. Comparison of results for the proposed and HS algorithms for wood function

	Analytical	BB-BC, PSO	HS
Objective function	0	$9.28 \cdot 10^{-11}$	$4.85 \cdot 10^{-9}$
Iteration	–	800	70000

6.2. Inverse airfoil design (aerodynamic verification of optimization algorithm)

In this Section, results obtained from the inverse optimization are presented. To do this, the initial and the target airfoil must be considered. Because the initial and final airfoils are specified, this method can be considered as a criterion to evaluate precision and accuracy of an optimization algorithm. It could be a reliable method for evaluation of the applied algorithm in the aerodynamic field. In the case of the airfoil inverse optimization, the objective function is the difference of pressure distribution of optimized airfoils in each step, and the final step is such that the optimization algorithm should minimize this objective function during problem solving. As a result, the curve of the optimized pressure coefficient in each step converges towards the curve of the pressure coefficient of the target airfoil. In fact, the optimization problem starts form the initial airfoil, and by modification of design parameters, optimizes towards an airfoil in which the pressure coefficient has the lowest difference in the pressure coefficient of the target airfoil

$$\text{Objective function} = \frac{1}{2}\int_0^1 (C_p - C_{p\,target})_{LS}^2\,dx + \frac{1}{2}\int_0^1 (C_p - C_{p\,target})_{US}^2\,dx \tag{6.2}$$

where C_p is the optimized pressure coefficient in each step and $C_{p\,target}$ is the pressure coefficient of the target airfoil. To solve, Airfoil NACA0012 as the initial airfoil is entered into the algorithm to converge towards the target airfoil NACA4812. Relative parameters are assumed as follows: solving network is 71×33, the angle of attack of the inviscid flow is 1.25 degrees and the Mach number is 0.8. In Table 2, the range of design parameters is specified.

Table 2. Valid range for design parameters

Parameter	m	p	τ
Valid range	0-0.01	0-0.1	0-0.3

6.2.1. Inverse transform of airfoil NACA0012 into NACA4812

Figure 4 shows the geometry of the initial, target and optimized airfoils. Dashed line airfoil geometry is NACA0012 which must be transformed to NACA4812, the target airfoil specified with bold line. Geometry with square markers is the optimized geometry achieved from the proposed algorithm and has good cosistency with the target airfoil.

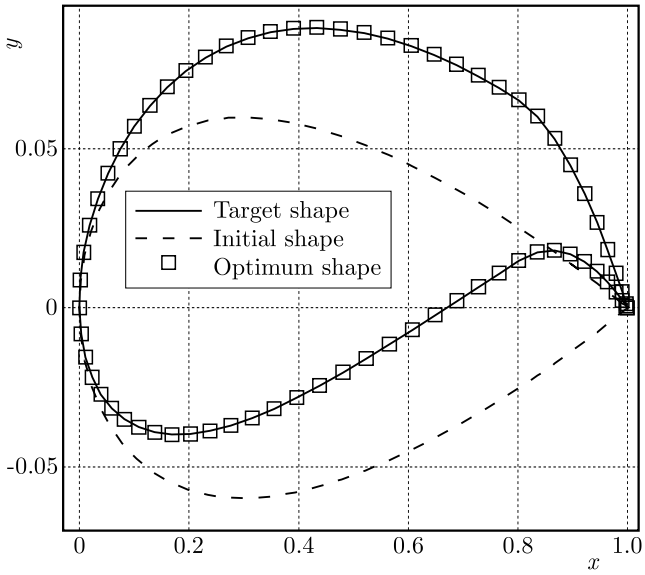

Fig. 4. Geometry of the initial, target and optimized airfoil in the inverse transform of NACA0012 into NACA4812

Figure 5a shows convergence of the objective function versus the iteration number. During solution, the objective function must be minimized and approach zero. The optimization algorithm has done this well with 40 iterations such that the value of the objective function decreased from 0.7788 to $7.249 \cdot 10^{-5}$.

Figure 5b depicts the distribution of the pressure coefficients for the initial, target and optimized airfoils in the process of going from NACA0012 to NACA4812. The dashed line is the pressure coefficient of the initial NACA0012 that the optimization algorithm should turn it into the bold line curve (pressure coefficient of the target airfoil) during solution. Square markers are related to the optimized airfoil and show good consistency with the target airfoil coefficients. This proves higher accuracy of the proposed algorithm in reaching the target airfoil.

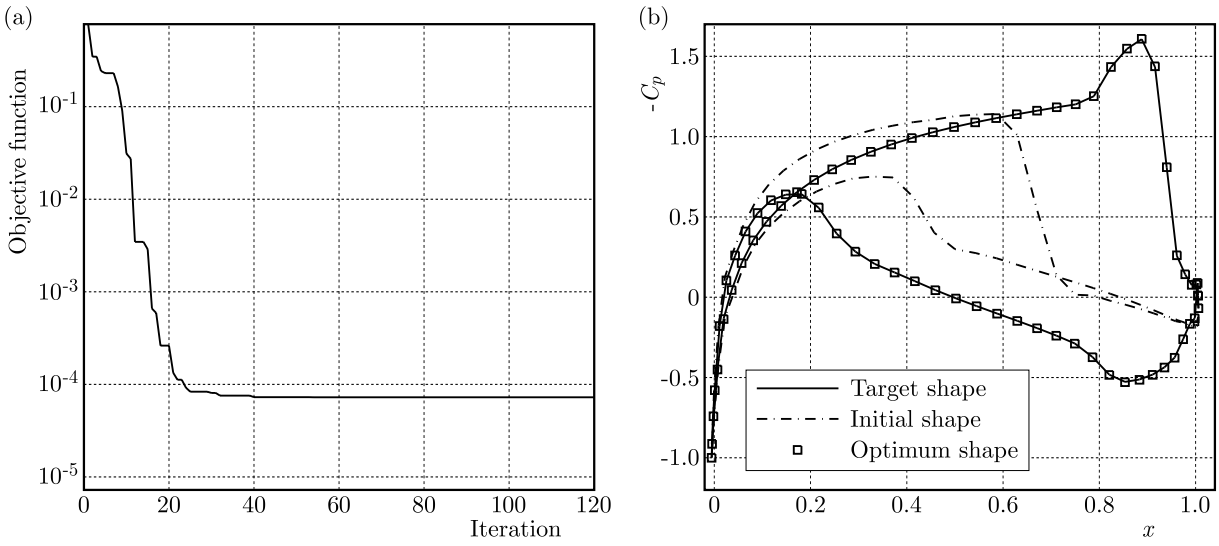

Fig. 5. Convergence history (a) and distribution of the pressure coefficient for the initial, target and optimized airfoil (b) in the inverse transform of NACA0012 into NACA4812

6.3. Direct design of airfoil

In this Section, direct optimization is used to find the optimized airfoil with different objective functions. The first objective function is minimization of the drag coefficient while the lift coefficient is fixed. In the second problem, maximization of the lift coefficient is investigated while drag coefficient is fixed. And finally, in the third case, the objective function is the ratio of the drag to lift coefficient, and the goal is to minimize it.

6.3.1. Optimization with the goal of minimizing the drag coefficient with a fixed lift coefficient constraint

In this case, the goal is to minimize the drag coefficient while satisfying a fixed lift coefficient

$$\text{Objective function or } f = C_D + 10\Big(1 - \frac{C_L}{0.6}\Big)^2 \tag{6.3}$$

The optimization algorithm should minimize the function f, but $[1 - (C_L/0.6)]^2$ is the penalty function for it. So, to minimize f, the optimization algorithm must push the lift coefficient towards 0.6 and keep it fix at this value while the drag coefficient is minimized. Whenever the goal is to minimize the drag coefficient, optimization could result in very thin airfoils while such geometries are not of the designer's interest. To prevent generating such geometries, a thickness constraint is introduced to the method. The minimum thickness considered is 0.086. Figures 6 up to 8 express convergence histories versus the iteration number.

6.3.2. Optimization with the goal of maximizing the lift coefficient with a fixed drag coefficient constraint

In this method, the objective function introduced to the algorithm must be such that it maximizes the lift coefficient while it keeps the drag coefficient fixed. Thus, the term related to the drag coefficient is considered as the penalty function. By varying this term, the optimization algorithm tries to keep the drag coefficient at a fixed value while maximizing the lift coefficient. For this case, the objective function is considered as

$$\text{Objective function or } f = C_L - 10\Big(1 - \frac{C_D}{0.06}\Big)^2 \tag{6.4}$$

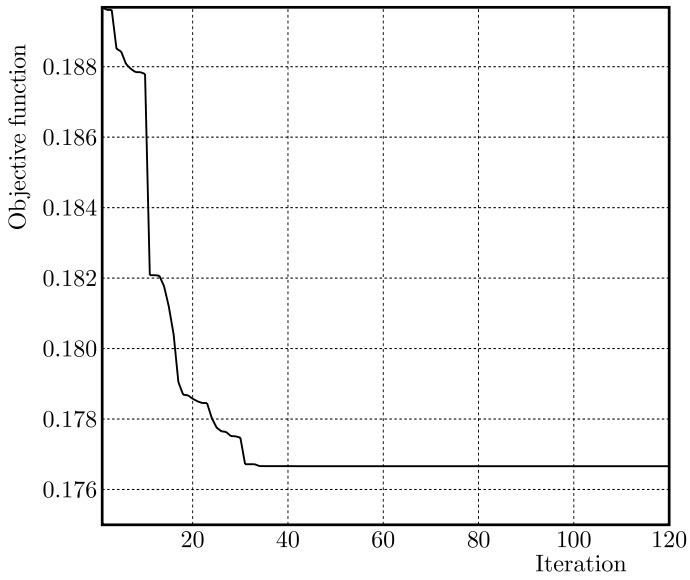

Fig. 6. Convergence history of the objective function for minimizing the drag coefficient while the lift coefficient is fixed

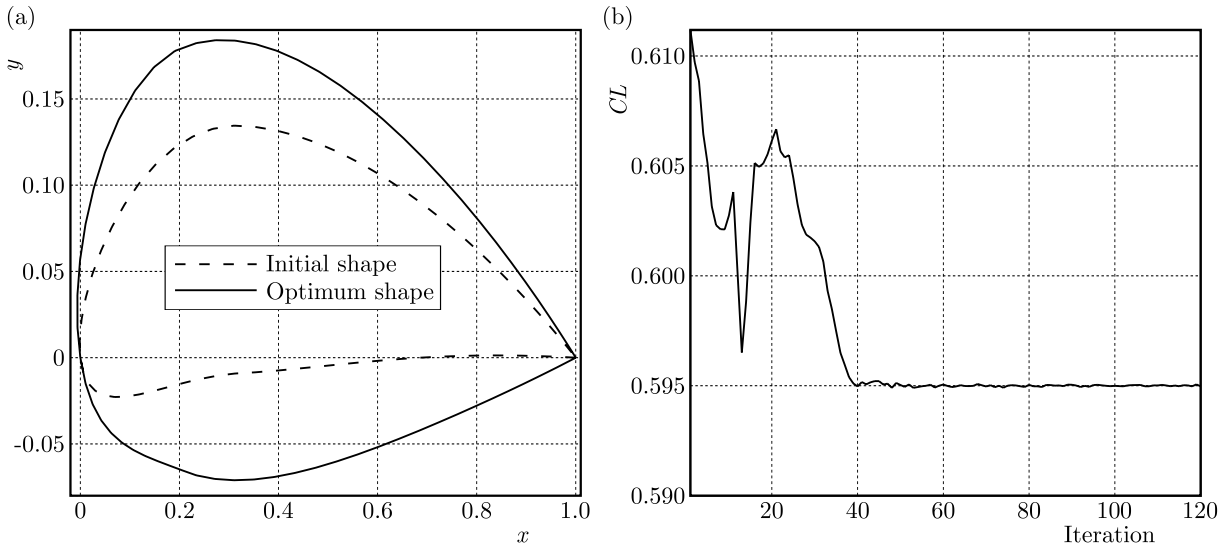

Fig. 7. (a) Initial and optimized airfoil geometries for f, (b) convergence history of the lift coefficient

Therefore, the optimization process pushes the drag coefficient towards 0.06 and tries to maximize the lift coefficient. Figures 9 up to 11 show convergence histories versus the iteration number.

6.3.3. Direct optimization with the ratio of the drag to lift coefficient objective function

The problem under investigation is minimization of the ratio of the drag to lift coefficient. The optimization starts with the initial airfoil and, by varying design parameters, is finally optimized to desired geometry. To be sure about its performance, the problem is solved 4 times with different initial airfoils. It is worth mentioning that 3 initial airfoils are chosen randomly, and in one solution NACA4812 is specified as the initial airfoil for the problem. In this work, the population of airfoils is 5 and the iterations are 120 for all optimizations. To prevent very thin airfoils, the minimum thickness τ is considered to be 0.036. Figure 12 illustrates the initial and optimized geometries for 4 different solutions. The dashed lines are random initial geometries

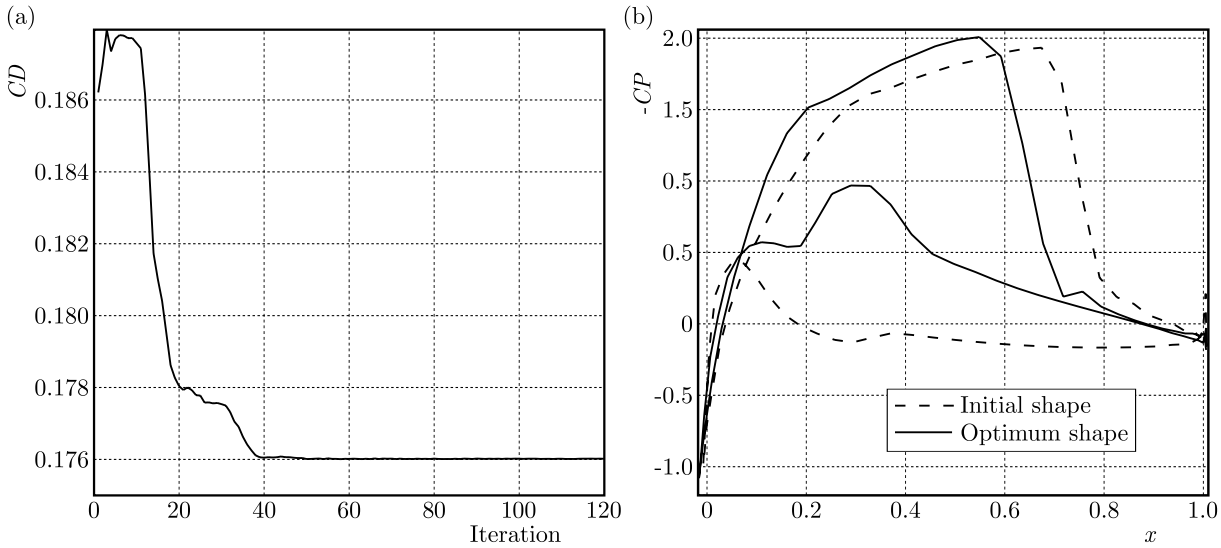

Fig. 8. (a) Convergence history of the drag coefficient to minimize, (b) distribution of the pressure coefficient around the initial and optimized airfoils

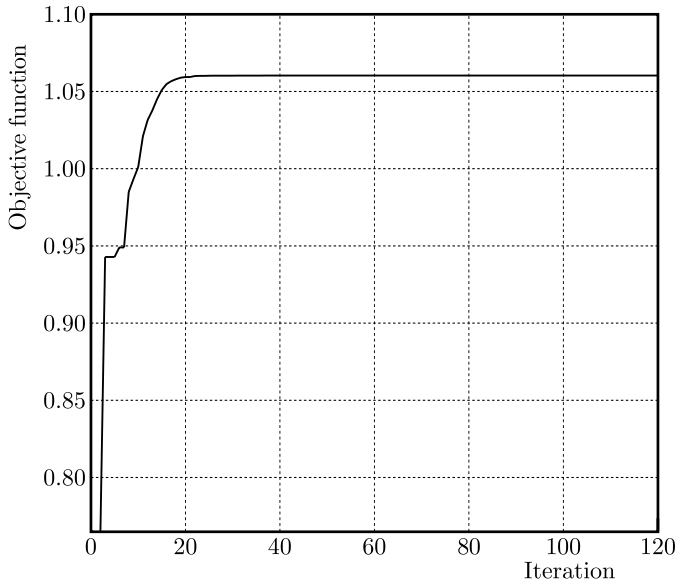

Fig. 9. Convergence history of f

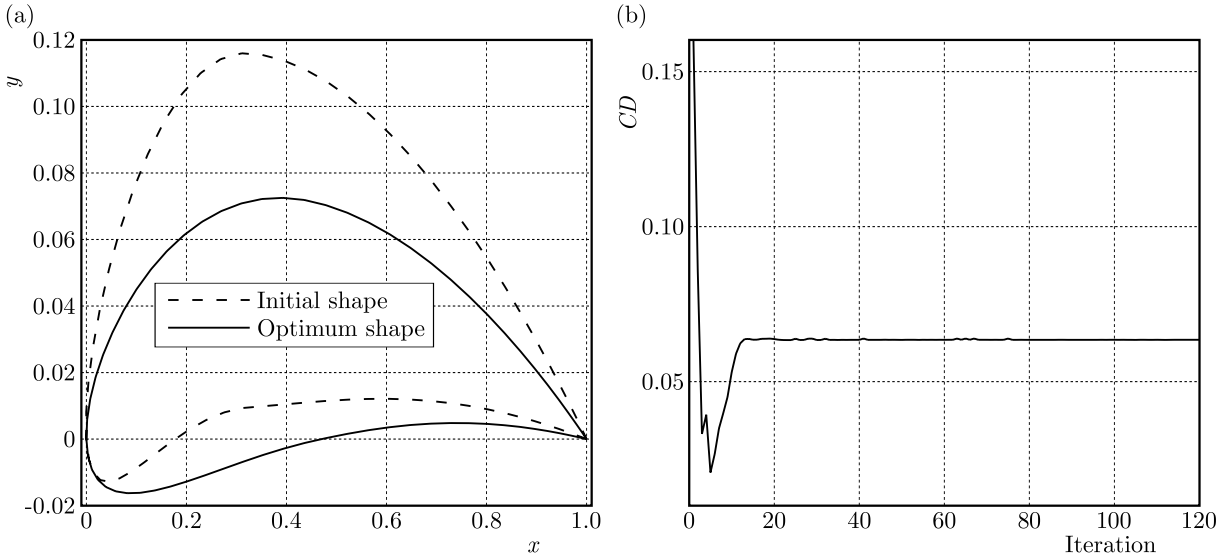

Fig. 10. (a) Geometry of the initial and optimized airfoils of f, (b) convergence history of the drag coefficient

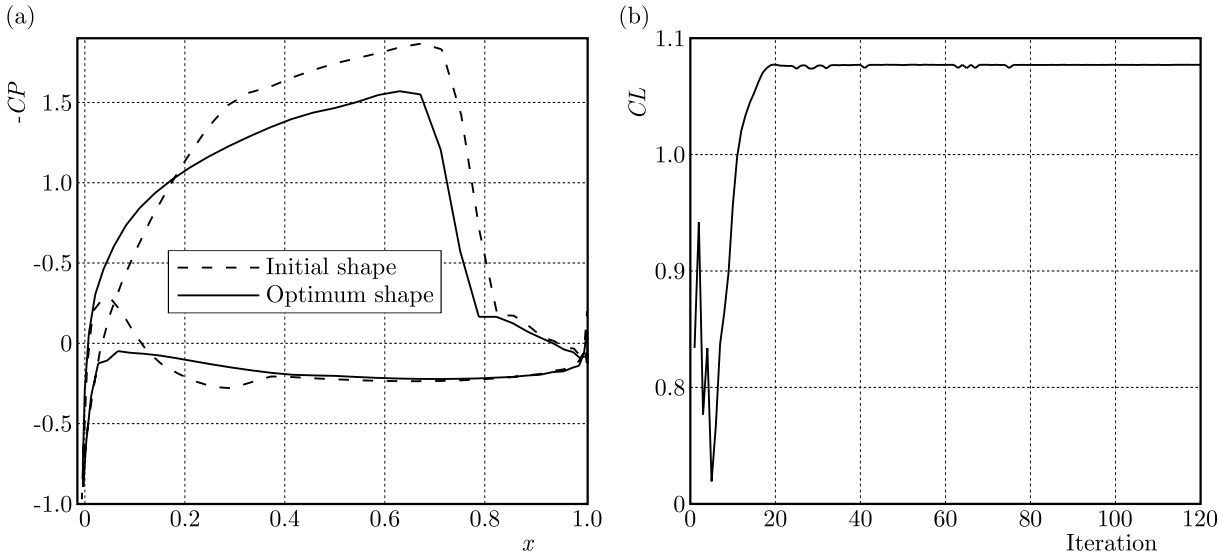

Fig. 11. (a) Convergence history of the lift coefficient to minimize, (b) distribution of the pressure coefficient around the initial and optimized airfoils

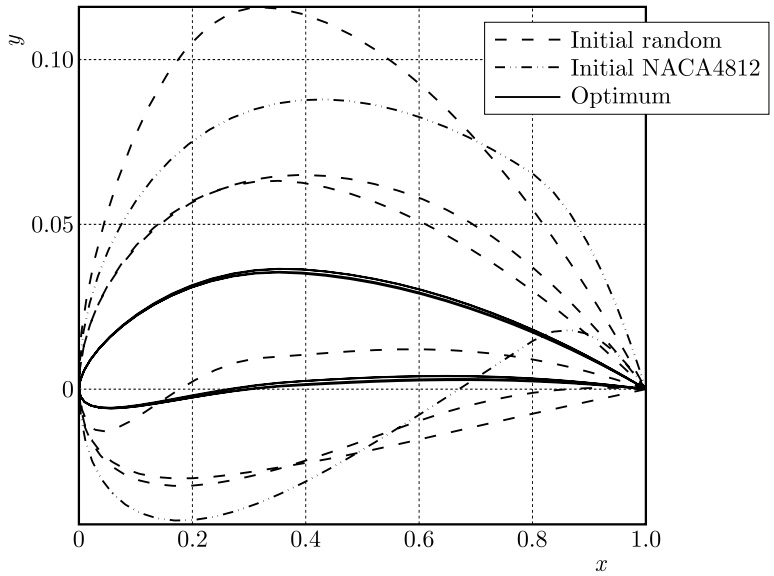

Fig. 12. Initial and optimized airfoil geometries with the objective function C_D/C_L

and the dash-dot line is the initial NACA4812. The optimization algorithm is finally converged to the same geometry in all 4 solutions which are clarified by solid lines. It proves the preciseness of the presented optimization algorithm.

Figure 13a shows the convergence history of the objective function for 4 different initial airfoils (the curve shows that in all 4 solutions, it starts with the initial value of their objective function and finally converges to the best target airfoil of the final population). The optimization algorithm is able to decrease the initial values of different objective functions of 4 airfoils to $2.38 \cdot 10^{-2}$ (the problem is solved 4 times. In 4 runs, the population of airfoils is 5 but in three runs all 5 airfoils are chosen randomly, and in one run a NACA4812 is chosen as the initial airfoil while other four airfoils are selected randomly. The optimized airfoils are the best final population for each solution).

Figure 13b shows the coefficients for 4 solutions. The bold line curve is the distribution of optimized pressure coefficients around the initial airfoil (NACA4812), and the dashed lines are

optimized pressure coefficients for other 3 initial airfoils. As it can be understood from Fig. 7b, in all 4 solutions, the pressure coefficients have good matching, which proves the exactness of the optimization algorithm.

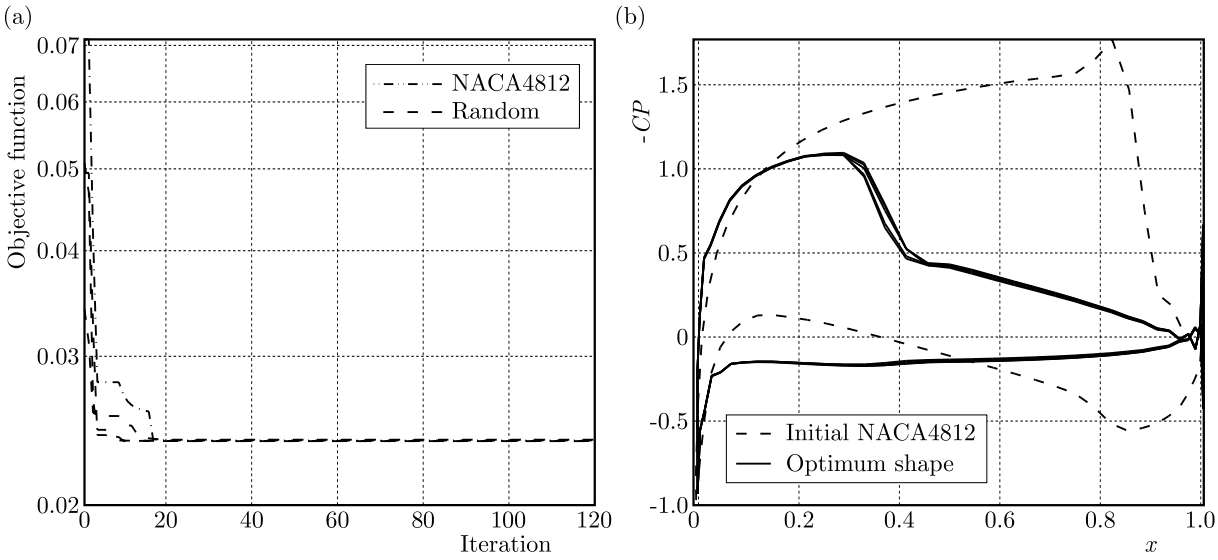

Fig. 13. (a) Convergence history for the objective function of the C_D/C_L ratio for different initial airfoils, (b) pressure coefficient around initial NACA4812 and optimized airfoils for C_D/C_L

7. Conclusions

In this work, for the first time Big-Bang Crunch and Particle Swarm optimization algorithms are combined. This hybrid algorithm has been used to find the optimum airfoil geometry. The airfoil geometry is introduced with 4-digit NACA method, and Euler's equations have been used for modeling the flow around it. Inverse and direct methods have been considered for optimization. In the inverse method, excellent consistency of pressure coefficients of optimized and target airfoils proves high accuracy of the proposed optimization algorithm. In the direct method, the problem has been solved for three different objective functions. The results are another proof of high accuracy of the proposed optimization algorithm to reach into an optimized solution with a lower number of iterations, which can be asserted as a shining advantage for it. Therefore, the hybrid algorithm presented in this paper is a strong method to reach an optimized solution.

References

1. Alves R., Goncalves L., Aguiar J., Brasil A.C.P. Jr., 2016, Parsec parameterization methodology for enhancing airfoils geometry using PSO algorithm, *CILAMCE 2016, Proceedings of the XXXVII Iberian Latin-American Congress on Computational Methods in Engineering*, Suzana Moreira Avila (Editor), ABMEC, DF, Brazil, 6-9

2. Ebrahimi M., Jahangirian A.R., 2017, Airfoil shape optimization with adaptive mutation genetic algorithm, *Journal of Agricultural Science and Technology (JAST)*, **11**, 1, 47-59

3. Endo M., 2011, Wind turbine airfoil optimization by particle swarm method, M.SC. Thesis, Case Western Reserve University

4. Erol O., Eksin I., 2006, A new optimization method: big bang-big crunch, *Advances in Engineering Software*, **37**, 106-111

5. Giannakoglou K.C., 2002, Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence, *Progress in Aerospace Sciences*, **38**, 43-76
6. Hájek J., 2009, Aerodynamic optimization of airfoils and wings using fast solvers, Ph.D. Thesis, Charles University, Prague
7. Jahangirian A., Shahrokhi A., 2009, Inverse design of transonic airfoils using genetic algorithm and a new parametric shape method, *Inverse Problems in Science and Engineering*, **17**, 5, 681-699
8. Jameson A., Schmidt W., Turkel E., 1981, Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes, *AIAA Meeting Paper*, DOI: 10.2514/6.1981-1259
9. Kaveh A., Talatahari S., 2010a, A discrete Big Bang-Big Crunch algorithm for optimal design of skeletal structures, *Asian Journal of Civil Engeneering (Building and Housing)*, **1**, 103-122
10. Kaveh A., Talatahari S., 2010b, Optimal design of Schwedler and ribbed domes via hybrid Big Bang-Big Crunch algorithm, *Journal of Constructional Steel Research*, **66**, 412-419
11. Kennedy J., Eberhart R.C., 1995, Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, **4**, 1942-1948
12. Khurana M.S., 2008, Application of an hybrid optimization approach in the design of long endurance airfoils, *26th Congress of International Council of the Aeronautical Sciences 2008, ICAS 2008*
13. Kumbasar T., Eksin I., Güzelkaya M., Yeşil E., 2008, *Big Bang Big Crunch Optimization Method Based Fuzzy Model Inversion*, Springer-Verlag Berlin Heidelberg, 732-740
14. Lane K., Marshall D., 2010, Inverse airfoil design utilizing CST parameterization, *48th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*
15. Lee K.S., Geem Z.W., 2005, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, *Computer Methods in Applied Mechanics and Engineering*, **194**, 3902-3933
16. Xu Z.H., Xia J., 2016, Aerodynamic optimization based on continuous adjoint method for a flexible wing, *International Journal of Aerospace Engineering*, Article ID 4706925
17. Wauquiez C., 2000, Shape optimization of low speed airfoils using MATLAB and automatic differentiation, Licentiate's Thesis
18. Wickramasinghe U.K., Carrese R., Li X., 2010, Designing airfoils using a reference point based evolutionary many objective Particle Swarm Optimization algorithm, *WCCI 2010, IEEE World Congress on Computational Intelligence*, Barcelona, Spain, 1857-1864
19. Zhang Z., Lum K.Y., 2006, Airfoil optimization design of drag minimization with lift constraint using adjoint equation method, *44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 9-12

Manuscript received February 18, 2019; accepted for print March 31, 2019