41st Aerospace Sciences Meeting and Exhibit
6-9 January 2003, Reno, Nevada

AIAA 2003–0043

AIRFOIL DESIGN USING A GENETIC ALGORITHM AND AN INVERSE METHOD

B. Allen Gardner and Michael S. Selig
Department of Aeronautical and Astronautical Engineering
University of Illinois at Urbana–Champaign
Urbana, Illinois 61801

41st Aerospace Sciences Meeting and Exhibit
6–9 January 2003
Reno, Nevada

Airfoil Design Using a Genetic Algorithm and an Inverse Method

B. Allen Gardner* and Michael S. Selig†
Department of Aeronautical and Astronautical Engineering
University of Illinois at Urbana–Champaign
Urbana, Illinois 61801

Abstract

In this paper, optimal airfoil shapes are found through manipulation of the velocity distribution by a genetic algorithm. The airfoil geometries are generated by an inverse method from velocity distribution parameters, and a viscous-flow analysis code is used to determine proper fitness values for candidate airfoils based on preset performance criteria. The method is compared with the more traditional approach of direct geometry manipulation for a simple single-objective aerodynamic optimization problem for a symmetric airfoil. The inverse and direct approaches are compared using a simple genetic algorithm and a hybrid genetic algorithm, where the hybrid method is formed by combining a simple genetic algorithm and a specialized local search method. Finally, the method is used to design a cambered airfoil that outperforms the existing state-of-the-art. Results indicate that using the design variables defining the velocity distribution in the inverse method has great potential for increasing the efficiency of airfoil shape optimization using genetic algorithms.

Introduction

In the past, several researchers have developed optimization methods that directly adjust airfoil shapes by way of spline supports, orthogonal shape functions, linear combinations of known airfoils, or geometry perturbations of an airfoil that is known to be close to an optimum. These direct-design approaches have been used within a wide range of optimization algorithms including classical methods[1,2] and genetic algorithms (GAs).[3–6]

In their GA-based optimization method, Holst and Pulliam[3] used the PARSEC[7] method to parameterize airfoil geometries using 10 control variables that represent typical geometry characteristics. Using this method, broad geometry constraints can be met by simply fixing a single control variable. By employing only 10 variables, however, the method searches a small design space compared with most other direct-design approaches. Thus, these 10 geometry controls are not sufficient to describe all possible airfoil shapes including unconventional designs.

Fanjoy and Crossley[4] developed a method to optimize airfoil shapes by using 21 design variables representing the control points of a B-spline. They found that this method could be used to represent nearly any arbitrary shape. The B-spline, however, produced airfoils with small surface "waves" between control points. Unfortunately, this "waviness" is reflected in the velocity distribution. Viccini and Quagliarella[5] also investigated a GA-based airfoil optimization scheme using B-splines, and concluded that unlike a gradient-based approach increasing the number of control variables used in their GA did not impose a proportional computational cost. They also found that a classical, conjugate gradient method was able to solve their airfoil optimization problem with about 1/5 of number of flow-solver solutions required by their best GA so long as a suitable starting point was chosen.

A problem with these direct-design approaches is that the relationship between geometry and performance is highly nonlinear. That is, a small change in the surface shape of an airfoil can cause ripples in the velocity distribution, greatly degrading performance. Because of this phenomenon, there has been interest in perturbing the geometry in such away as to produce reasonable velocity distributions with each perturbation. A way to bypass this is by using an inverse method to parameterize the airfoil. An inverse method allows the velocity distribution to be directly controlled rather than anticipated from geometry perturbations. Concerns about using the inverse method have been expressed in the past because of the risk of defining a velocity distribution that cannot be physically achievable.[8] This situation, however, can be partially avoided by using iteration schemes in the inverse method.

The inherent robustness of GAs has made them increasingly popular in engineering applications. GAs are capable of searching for optimal solutions within a

*Graduate Research Assistant, 306 Talbot Laboratory. Student Member AIAA. bagardne@uiuc.edu

†Associate Professor, 306 Talbot Laboratory. Senior Member AIAA. m-selig@uiuc.edu

highly nonlinear and discontinuous design space since they do not require sensitivity computations (derivatives). They rely only on information concerning payoff and do not breakdown when other information is not available. Because GAs work from a population rather than a single point, they are less likely to become focused on a local optimum. Considering these characteristics, GAs are much more robust than traditional optimization techniques. Of course, GAs do have drawbacks. There is no guarantee that they will converge on the best possible design. Instead, the purpose of a GA is to improve on an existing design.[9] Also, for relatively smooth design spaces, GAs may require many more iterations than traditional optimization techniques and are therefore more computationally expensive. However, with increasing computation speeds and more efficient design and analysis algorithms, coupling a GA with an inverse method is an attractive approach that should be explored.

This paper presents an airfoil shape optimization method that uses an inverse method for shape generation and a genetic algorithm for optimization. The GA operates on design variables that define the airfoil in the inverse method implemented in the code PROFOIL.[11,12] Collectively, the new optimization code is named ProfoilGA. A Newton iteration scheme is built into the inverse method to impose geometry constraints that aid in reducing the design space and thereby improves the efficiency of the optimization method. Furthermore, the Newton iteration scheme reduces the possibility that the GA will specify values for velocity distributions that generate nonphysical geometries, a common but solvable problem for traditional inverse design.

In the remainder of this paper, the primary tools used in ProfoilGA are explained, and the general flow is described. The inverse-design/GA method is compared with a direct-design/GA method named here BezierGA. The best versions of the two methods are compared for different GA and hybridization parameters. These comparisons are based on the optimization of a 10% thick symmetric airfoil for minimum drag at a lift coefficient of 0.92 and a Reynolds number of 300,000. Following the comparison, results are shown for an optimized cambered airfoil with the same constraints. The performance of the optimized cambered airfoil is compared to an existing airfoil that was designed for the same conditions.

ProfoilGA

The ProfoilGA method incorporates three primary tools: a genetic algorithm, an inverse-method airfoil design code PROFOIL,[11,12] and a coupled inviscid/viscous flow solver XFOIL.[13] The general interaction between these tools can be seen in the flowchart in Fig. 1. A more detailed description of the tools follows.

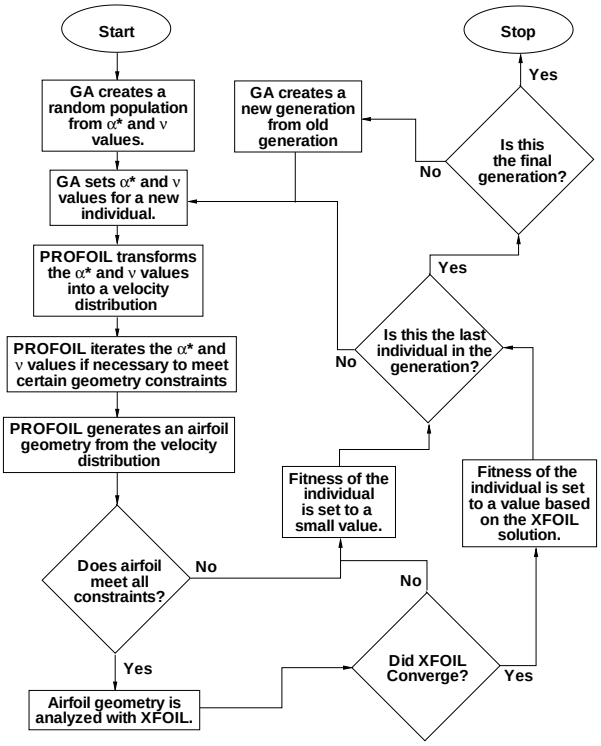

Fig. 1 ProfoilGA flowchart.

Genetic Algorithm

The GA is the mechanism in ProfoilGA that defines candidate airfoils and searches for the best. It works on binary strings that represent values for the design variables used in PROFOIL. These binary strings are converted to real number values for the design variables that PROFOIL then uses to generate the actual geometries. Each geometry is assigned a fitness value by XFOIL based on performance. The GA uses the fitness value of each airfoil geometry (which it knows as binary strings) to create the next generation of candidate airfoils (binary strings).

The GA implemented in this research was adapted from the GA driver originally created by Carroll.[10] Tournament selection with a shuffling technique for choosing random pairs for mating is used as the selection scheme. Tournament selection was chosen, in part, for its ability to handle constraints. Each design variable is specified by a minimum value, maximum value, and number of possible values and is coded in binary using a linear interpolation between the minimum and maximum value. The method includes options for niching, jump mutation, creep mutation and elitism. The number of offspring per pair of parents can be specified and a micro-GA operation is available. The nuances and details pertaining to genetic algorithms and their implementation go beyond the scope of this paper and can be found in the literature, e.g. see Ref 9.

PROFOIL

The role of PROFOIL[11,12] in ProfoilGA is to generate candidate airfoils based on values specified by the GA. As mentioned, PROFOIL is an inverse airfoil design method that allows one to prescribed the velocity distribution from which the airfoil shape is determined. Fundamentally, the method is based in part on the work of Eppler.[14,15] Figure 2 shows the basic conceptual process by which an airfoil shape can be generated in PROFOIL. The velocity distribution (Fig. 2b) is directly controlled through a piecewise curve of the function $\alpha^*(\phi)$. Usually the values of ϕ are mapped into ν where ν ranges from 0 to 60 points instead of 0 to 360 deg as is the case for ϕ. Figure 2a shows an example $\alpha^*(\nu)$ curve. The circles in Fig. 2a represent possible design variables that can be used to define the conformal mapping curve. The geometry of an airfoil (Fig. 2c) can be quickly calculated once the velocity distribution has been set by these design variables. Through a Newton iteration on the velocity distribution and its related parameters, PROFOIL also permits specification of laminar and turbulent boundary layer developments as well as geometric constraints.

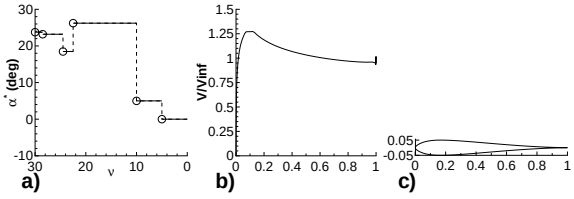

Fig. 2 Airfoil design process using PROFOIL: (a) conformal mapping function, (b) resultant velocity distribution, and (c) corresponding geometry.

The α^* function can affect the airfoil performance greatly hence it can be used to control airfoil performance as well as physical geometry. For these reasons, the basic purpose of ProfoilGA can be simply stated as follows. To investigate many different α^* curves until an optimum is found. The ProfoilGA method is quite general in that design variables used in the optimization process can be assigned for iteration either by the Newton scheme in PROFOIL or by the GA. For instance, the lower surface of an airfoil can be prescribed to have a certain boundary layer development that is achieved by the inverse method, while the upper surface is left free for iteration by the GA to achieve low drag at a given design point.

XFOIL

XFOIL[13] is a design and analysis method for single-element subcritical airfoils. A linear-vorticity second-order accurate panel method is used in the inviscid flow. This method is coupled with an integral boundary-layer method and an e^n-type transition amplification formulation using a global Newton method. An average of less than 2 sec of computation time on a 1.7-GHz pentirum 4 processor is required to calculate the performance coefficients of an airfoil with 140 coordinates at a single design condition.

XFOIL was chosen for use in ProfoilGA based on its speed and accuracy. It is very fast compared with Reynolds-averaged Navier-Stokes (RANS) methods, and it has been proven to be well suited for the analysis of subcritical airfoils even at low Reynolds numbers where laminar separation bubbles are significant. Once an airfoil is generated by PROFOIL, ProfoilGA calls XFOIL to predict the performance of the airfoil and assign it a fitness value.

BezierGA

The BezierGA method is similar to ProfoilGA only it uses Bezier curves to parameterize airfoils instead of PROFOIL. Thus instead of manipulating a conformal mapping curve, the GA directly manipulates the airfoil shape. Two Bezier curves are used to parameterize one surface of a symmetric airfoil as shown in Fig. 3. The front portion of the airfoil up to maximum thickness is parameterized by a 7^{th}-order Bezier curve. The aft part of the airfoil is specified by a 6^{th}-order Bezier curve. The curves are defined by control points (which dictate the order), seen as circles in Fig. 3. These control points can be moved vertically and horizontally to define practically any shape. First- and second-order continuity are enforced at all curve junctions except the trailing edge. With these requirements in place, only smooth, realizable airfoils are produced. Patching the two curves at the location of maximum thickness allows the designer to easily constrain the location and magnitude of maximum thickness. A total of 10 variables are used to manipulate the two curves, very comparable to similar approaches that can be found in the literature.

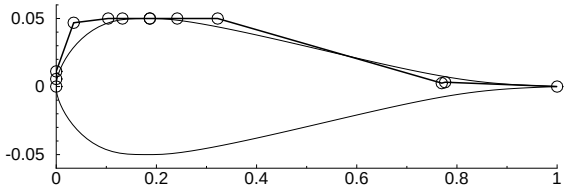

Fig. 3 Airfoil parameterization (expanded view) using Bezier curves.

Tuning The Methods

One of the primary objectives of this research, the first study, was to determine if ProfoilGA could find optimum airfoils in a shorter amount of time than direct-design methods like BezierGA. In order to make such a determination, both methods must be tuned for optimal performance before they can be compared. Many tests were performed in order to tune the methods. First, the GA parameters associated with the

Table 1 ProfoilGA Parameterization Approaches

Case	No. ν–α^* Combinations	ν Parameterization	α^* Parameterization (deg)
fixed-1	4	fixed: 22.5 to 30	range: 0 to 30
fixed-2	4	fixed: 22.5 to 30	seq: 0 to 30
fixed-3	10	fixed: 8.5 to 30	range: 0 to 30
seq-1	4	seq: 12.5 to 30	seq: 0 to 30
seq-2	4	seq: 8.5 to 30	seq: 0 to 30
seq-3	6	seq: 8.5 to 30	seq: 0 to 30

simple genetic algorithm (SGA), described previously, were varied. Second, the SGA was enhanced with a local search algorithm to form a hybrid-GA. This new hybrid-GA introduced new variables that also required tuning.

A simple optimization problem was created in order to measure the success of the tuning. As will be discussed later, another primary goal of this research, the second study, was to see if ProfoilGA could find an airfoil that outperforms the current state-of-the art. The state-of-the art being a 10% thick cambered airfoil with a maximum L/D of 104 operating at a lift coefficient of 0.92 and a Reynolds number of 300,000. It was assumed that the tuning of the GA parameters could potentially be problem specific; that is, the optimal settings for one case could be poor settings for another. The optimization problem used for tuning the methods was chosen based on its similarity to the cambered-airfoil optimization problem in order to reduce the amount of tuning needed for ProfoilGA. The only difference between the test case and the cambered-airfoil case was that the former was limited to being a symmetric instead of a cambered airfoil in order to reduced the amount of time required to perform the tests. The constraints and operating points remained the same. Thus, the objective used in the tuning tests was: find a 10% thick symmetric airfoil with minimum drag at a lift coefficient of 0.92 and a Reynolds number of 300,000.

In order to gauge the success of the algorithms, the performance of the respective optimized airfoils were compared to an existing symmetric airfoil. This comparison helps to assure that the methods are not simply tuned with respect to each other, but are optimally tuned to produce high performance airfoils. Three candidate sections were considered for use as a comparison in this first study: The NACA-0010, the NACA-63010a, and the Eppler 168 (Fig. 4). The E168 has a thickness greater than 12%, so it is expected to operate better at high lift-coefficients compared with thinner symmetric airfoils. For example, the NACA-0010 (10% thickness) airfoil has a $C_d = 0.02571$ and the NACA-63010a (10% thickness) has a $C_d = 0.02275$. Both C_d values are significantly higher than the E168 ($C_d = 0.01634$). Thus, the E168 is used as the primary basis of comparison because it is a more challenging competitor. Although this is a single-optimization problem with relatively few constraints, it is very difficult owing to the fact that few symmetric airfoils are capable of operating efficiently at a lift coefficient of 0.92 at the specified Reynolds number. Worth noting is the fact that the E168 was, most likely, not designed for these conditions. It is only used here as a comparison.

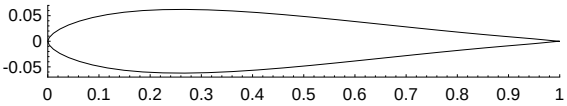

Fig. 4 Geometry of the E168 airfoil.

The best fully-tuned optimization algorithm can be determined in two ways. Either it requires less time than the other algorithm to find an airfoil that is better than the E168, or it finds the very best airfoil in a given amount of time, compared with other algorithms. Since the "bottleneck" to the process is the time required to analyze an airfoil using XFOIL, the number of calls to XFOIL is representative of the process time.

Simple Genetic Algorithm Comparisons

The parameters of the SGA used in both the ProfoilGA and BezierGA methods were tuned for the specific search spaces. The parameters investigated included population size, number of variables, range of variables, niching, creep mutation, and elitism. A micro-GA operater was also examined. For each case, the number of generations was limited to 300 while other GA parameters were varied. Over 200 cases were examined, representing over 1200 hours of processor time on a 1.7-GHz PC running Linux.

Since ProfoilGA uses a unique method to parameterize airfoils, the trade studies additionally involved determining the best way to manipulate the conformal mapping curve of Fig. 2a. After much experimentation, Table 1 lists the characteristics of six different parameterization cases that were most successful. The ν and α^* parameters were either fixed at constant values or varied within a sequence or a range. For the sequence cases, the parameters were required to be continuously increasing. For example, if the GA were

to choose a value of 10 for the first α^* parameter, and the entire sequence was limited to a maximum of 30, then the second α^* value would be automatically constrained to a range of 10 to 30. In the range cases, each parameter could take any value within the specified range. The range cases, therefore were capable of creating very distorted $\alpha^*(\nu)$ curves; whereas the sequence cases required the $\alpha^*(\nu)$ curve to always decrease monotonically from the leading edge to the trailing edge. Most airfoils do not require a $\alpha^*(\nu)$ curve that is distorted, so it was expected that the sequence approach would be the most effective (case seq-1). However, studies showed that the range cases worked best.

Table 2 lists some of the optimal GA parameters that resulted from tuning the ProfoilGA and BezierGA methods. When fully tuned, ProfoilGA used only five design variables. This tuning involves having the GA manipulate the values of each of four α^* parameters from 0 to 30 while keeping the ν parameters constant at values of 22.5, 24.5, 28.5, and 30.0. The fifth variable specifies geometry information about the cusp of the trailing edge. A Newton iteration is used in PROFOIL to iterate the α^* values until the thickness constraint of 10% is reached.

Table 2 Summary of ProfoilGA and BezierGA Optimal Settings

Parameter	ProfoilGA	BezierGA
No. Variables	5	10
Possible	1024	512
Design Space	1.1×10^{15}	1.2×10^{27}
Population Size	50	50
Crossover	Uniform	Uniform
Elitism	Yes	Yes
Micro-GA	Yes	Yes
Niching	No	No

An interesting result of the studies was that the only major difference between the optimal settings was the number of control variables required. ProfoilGA required half as many control variables as BezierGA. This resulted in the design space of ProfoilGA being substantially smaller than that of BezierGA, as shown in Table 2. This reduction in design space represents the primary advantage of ProfoilGA over direct-design codes like BezierGA. This advantage, however, is only realized since the constant values of ν were determined by a designer familiar with inverse design techniques. If a user is not familiar with inverse design, then the ProfoilGA parameterization approaches fixed-3, seq-1, seq-2 and seq-3 would be more practical. In these cases, the number of design variables increases by a minimum of four and ProfoilGA may no longer outperform BezierGA. In fact, the performance of BezierGA was very comparable to the ProfoilGA cases seq-1 and seq-2. Moreover, BezierGA was found to be generally more effective than the ProfoilGA cases fixed-3 and seq-3.

The GA driver used for both ProfoilGA and BezierGA was designed to find maximum values. Thus, in order to minimize drag, the GA was simply required to maximize negative drag. For the remainder of the paper, fitness will refer to $-C_d$ at the conditions specified earlier.

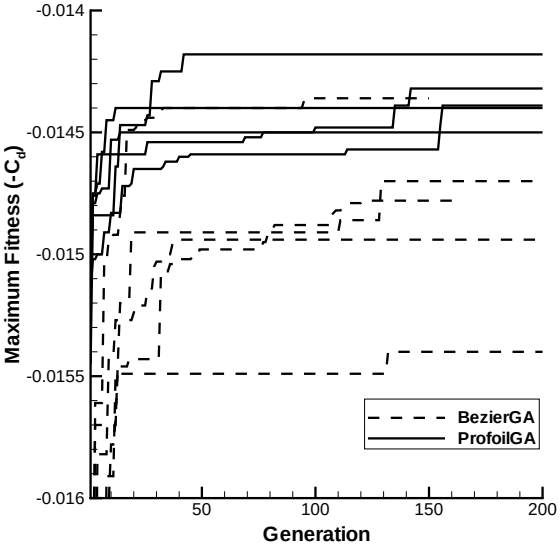

Fig. 5 Comparison of ProfoilGA case fixed-1 and BezierGA for different random seed populations.

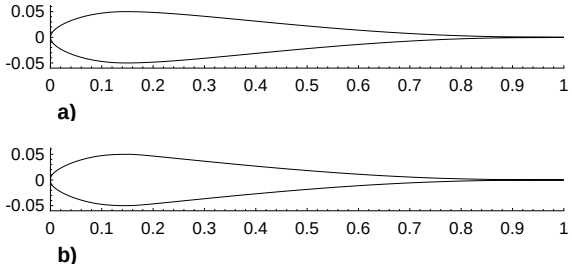

Fig. 6 Optimum airfoils found by (a) ProfoilGA and (b) BezierGA.

Figure 5 shows results for ProfoilGA (case fixed-1) and BezierGA using the optimal settings of Table 2 and different randomly generated initial seed populations. As can be seen in the figure, ProfoilGA outperforms BezierGA in nearly all cases. Also, note that the performance of BezierGA was more sensitive to the seed population than ProfoilGA. From these results, it is evident that ProfoilGA is generally more efficient and reliable than BezierGA. The best airfoils found by ProfoilGA and BezierGA, respectively, are shown in Fig. 6. It is interesting to note that the location of maximum thickness for both airfoils is nearly

Table 3 Summary of hybrid-GA operators

Parameter	Description
khybrid	Number representing frequency of local search (i.e., 2 = every other generation)
phybrid	Probability that a single individual will be locally searched
phybridparam	Probability that a particular variable within an individual will be locally searched
kmax	Maximum number of iterations within the local search
mhybrid	Represents which individuals in a pop. are chosen for local search (1=random, 2 = best)
iLamarckian	Flag to backsubstitute LSA genotype (1 = Lamarckian, 0 = Baldwinian)

the same. In addition, the shape of these airfoils are nearly identical up to the point of maximum thickness.

Hybrid-GA Comparisons

Most modern genetic algorithms are combined with a specialized search method.[16] To explore this potential, the SGA used in ProfoilGA and BezierGA was enhanced with a local search algorithm to form a hybrid-GA. A GA combined with a conjugate gradient method has been used in airfoil design and shown to reduce convergence time to a third.[6] The ability of a specialized search method to enhance the performance of a GA depends entirely on the structure of the design space.

A more smooth and linear design space lends itself to rapid search by most classical optimization methods. The design spaces searched by ProfoilGA and BezierGA are both nonlinear and discontinuous. A smooth variation of a PROFOIL parameter can be used to get a family of many very similar airfoils, but when the XFOIL drag of these airfoils is plotted vs the smoothly varying parameter, the curve is nonlinear (jagged). The design space of BezierGA is expected to be even more nonlinear due to the nature of direct-design methods, as discussed in the introduction. Discontinuities arise primarily because XFOIL may not converge. In addition, since ProfoilGA manipulates the velocity distribution rather than the airfoil surface, there are cases when a non-realizable airfoil (e.q., an airfoil with a crossed trailing edge) is produced causing a discontinuity in the design space. Thus, it is expected that ProfoilGA must search a space that has many more discontinuities than the space searched by BezierGA. In contrast, it is expected that the design space that BezierGA searches is more highly nonlinear than the design space of ProfoilGA, at least where it is continuous. Since the effectiveness of a hybrid-GA depends on the design space it is required to search, determining the relative performance of ProfoilGA and BezierGA using a hybrid-GA driver is worth investigating.

A hybrid-GA takes advantage of the global search capabilities of a GA and the efficient local search capabilities of a traditional "hill-climber" algorithm. Often these local search algorithms (LSA) are classical gradient-based methods. A hybrid-GA follows the same basic process as an SGA; the only difference being the LSA is used within the GA driver as an additional GA operator, similar to crossover and mutation. The process by which the LSA is implemented in a hybrid-GA is as follows. After all of the individuals within a population have been assigned fitness values, and before the crossover and mutation operators are implemented, certain individuals within the population are chosen for enhancement through local search. These individuals are used as starting points in the LSA. The LSA is then allowed to "climb the nearest hill," and is stopped when it either converges or reaches a specified maximum number of iterations. After the LSA is completed, the individual used as the starting point is given a new fitness value corresponding to the best solution found by the LSA.

Many factors must be considered when constructing a hybrid-GA. There is an optimal ratio relating the time spent performing global search with the GA to the time spent in local search.[19,20] This ratio can be controlled in four ways: 1) the number of individuals within a population chosen for enhancement through the LSA can be limited, 2) instead of using the LSA every generation, the LSA can be limited to every n generations, 3) a subset rather than all of the design variables for a particular individual can be locally searched, and 4) a maximum number of iterations can be specified for the LSA. Four hybrid-GA parameters associated with the four methods to control the global-local time ratio were introduced in this study: khbyrid, phybrid, phybridparam, and kmax. Their descriptions are given in Table 3.

Another concern for a hybrid-GA is determining which individuals to choose for local search. Two methods were used in this study: 1) individuals are chosen at random, and 2) the best fit individual within a population is chosen. In order to facilitate these methods, an additional hybrid-GA parameter was created for this study: mhybrid. Its description is also found in Table 3. Note that when only the best individual in a generation is chosen for local search, i.e. mybrid = 2, the parameter phybrid is not used. The value of mhybrid reflects the level of selection pressure. For a value of mhybrid = 2, the best individual in a

population is chosen for local search, thus representing an increase in selection pressure. For a value of mhybrid = 1, individuals are chosen randomly within the population for local search, the hope being that diversity is preserved in each population by relaxation of selection pressure.

Also of importance in a hybrid-GA is whether or not to replace the individual chosen for local search with the best solution of the LSA. It was stated earlier that the fitness value of the individual is replaced with the LSA solution. This does not mean that the values of the design variables associated with such a solution replace those of the chosen individual. Backsubstituting the genotype (binary representation of the design variables) of the LSA solution into the population by replacing the chosen individual is known as Lamarckian learning. Simply changing the fitness value of the chosen individual to that of the LSA solution is known as Baldwinian learning. The distinction between Baldwinian and Lamarckian learning must be considered in any hybrid-GA.[19] Thus, an additional hybrid-GA parameter was introduced for this study: iLamarckian. A description of the variable is given in Table 3.

Before these new hybrid-GA parameters could be tuned, a local search algorithm had to be chosen. In order to find a good specialized search method for hybrid-GA use, seven algorithms were examined: five gradient-based algorithms and two non-gradient algorithms. All gradients were calculated using forward difference. None of the gradient-based algorithms were able to effectively find local optima. The most likely reason being that the gradients could not be accurately determined due to the nonlinearities and discontinuities in the design spaces of ProfoilGA and BezierGA. The two non-gradient methods, Nelder-Mead[17] and Powell-Brent,[18] were able to successfully locate local optima. The results of these two methods for different starting positions within the ProfoilGA design space are shown in Fig. 7. For all cases, the Nelder-Mead algorithm was best and, therefore, was chosen as the specialized search algorithm for the hybrid-GA drivers of ProfoilGA and BezierGA.

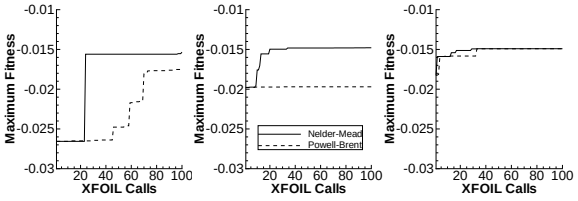

Fig. 7 Performance of local-search algorithms in the ProfoilGA design space.

Tests were performed for the hybrid-GA parameters listed in Table 3 in order to tune the hybrid-GA. Over 300 cases were examined representing approximately 1800 hours of processor time on a 1.7-GHz machine. All cases used elitism, micro-GA, and a population size of 50 individuals through a maximum of 200 generations. Each case began with the same randomly generated seed population. Since the first four hybrid-GA parameters in Table 3 control the ratio between the global search using the GA and the local search using the Nelder-Mead algorithm, every case resulted in a different average number of XFOIL calls per generation. Thus, the relative performance of each test case can not be measured with respect to the number of generations. Instead, the actual number of calls to XFOIL is used as the measure to gauge performance.

The two operators mhybrid and iLamarckian were found to have the largest influence on the effectiveness of the hybrid-GAs. The results of the hybrid-GA tuning tests, separated into graphs representing constant mhybrid and iLamarckian values, are shown in Figs. 8 and 9 for ProfoilGA and BezierGA, respectively. These results are displayed in comparison to the SGA versions of ProfoilGA and BezierGA. It should be noted that, in order to avoid displaying graphs with bewildering amount of data, the y axis in each of these figures does not simply show the maximum fitness associated with a particular XFOIL call, it shows the maximum fitness value found up to that point, i.e., the best fitness so far.

The results of the hybrid-GA tuning tests show that the effectiveness of a hybrid-GA depends greatly on the choice of operators. Local search improved BezierGA in all cases where mhybrid = 1 and iLamarckian = 0. These values mean that individuals in the population are chosen for local search based on a random selection, and the genotype of the LSA solution does not replace the chosen individuals. The opposite was true for ProfoilGA. The tuning tests indicated that optimal hybrid operators for ProfoilGA are mhybrid = 2 and iLamarckian = 1. These values mean that only the best individual within a population be chosen for local search, and the genotype of the LSA solution replace the best individual of each population.

Overall Comparison

All of the 400+ cases examined for both ProfoilGA and BezierGA produced airfoils that had lower C_d values than the comparison E168 airfoil at a C_l of 0.92 and a Reynolds number of 300,000. Figure 10 shows the overall performance comparison of ProfoilGA and BezierGA for both the SGA and hybrid-GA drivers. The hybrid cases displayed in the figure represent the best solutions from the hybrid-GA tuning tests. These results show that ProfoilGA is more efficient than BezierGA in both the original and hybrid forms. In fact, ProfoilGA beats BezierGA according to both criteria mentioned previously: it finds an airfoil that is better than the E168 in minimum time and it finds the best airfoil in a given amount of time. The maximum fitness values for each algorithm and a comparison of

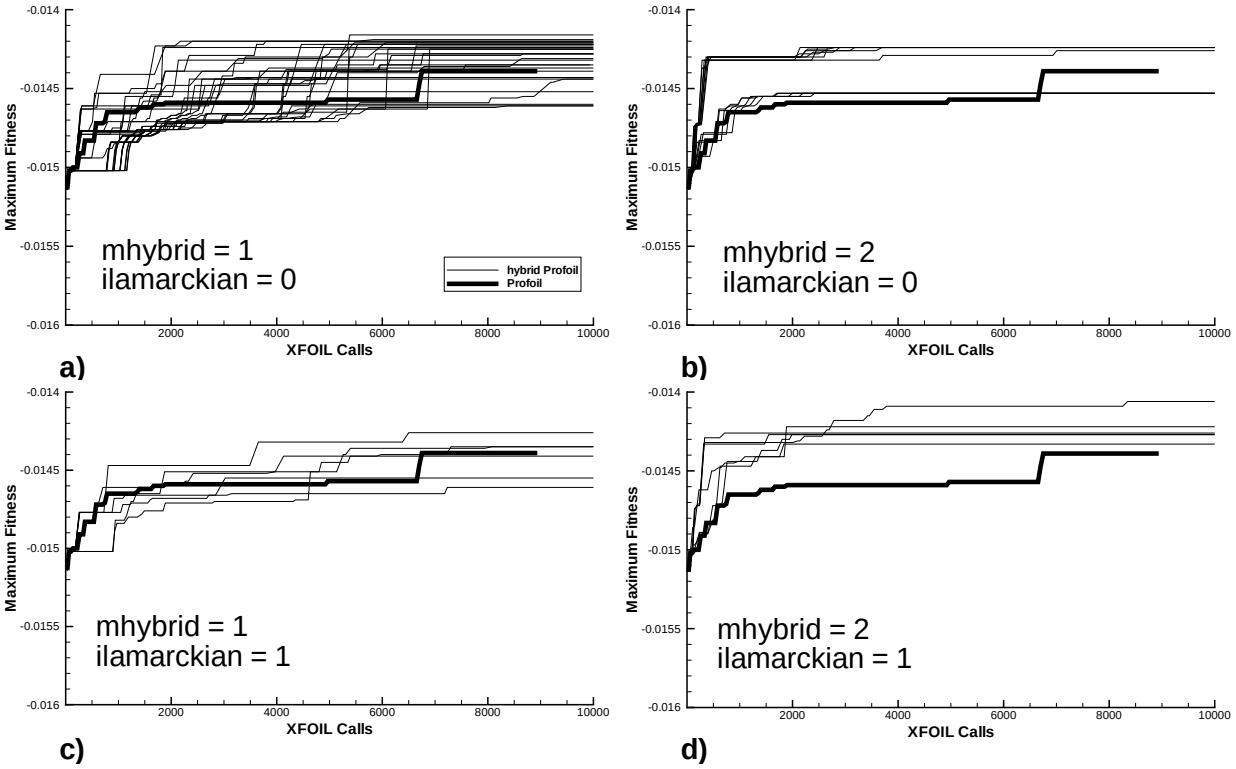

Fig. 8 Hybridization trade studies using ProfoilGA.

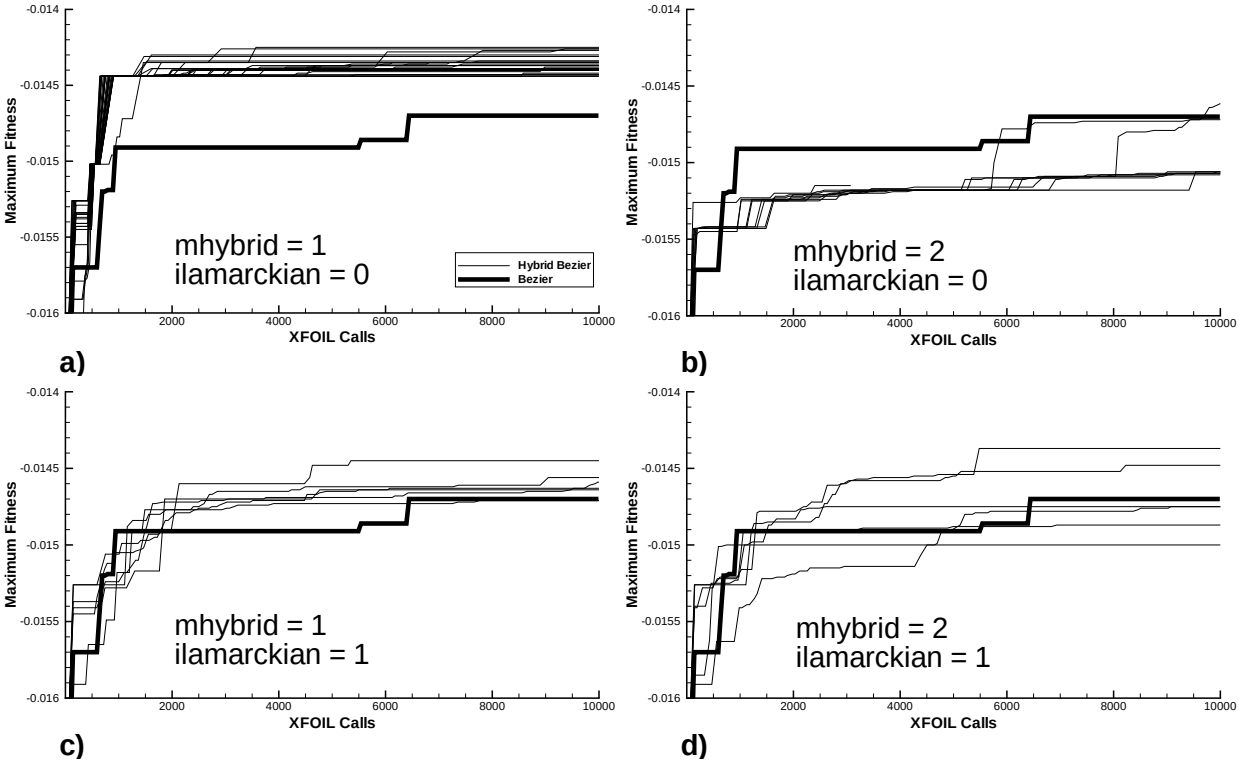

Fig. 9 Hybridization trades studies using BezierGA.

their results to the E168 airfoil are shown in Table 4. The performance of the Nelder-Mead algorithm as it searched the ProfoilGA design space is also shown in Figure 10 and Table 4 in order to compare the effectiveness of the SGA and hybrid-GA search methods to a traditional optimization method.

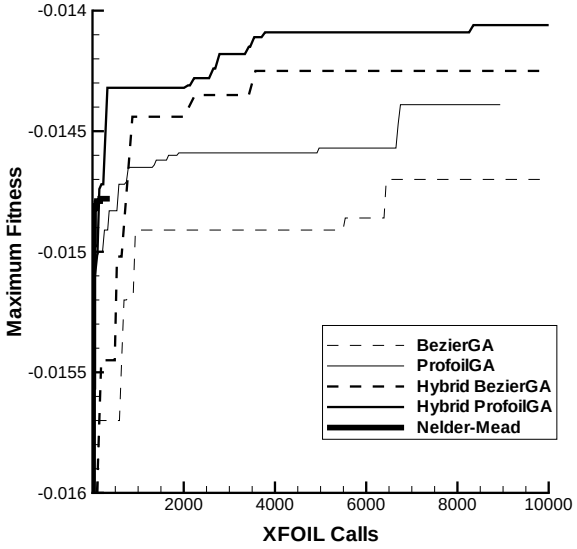

Fig. 10 Performance of the original and hybrid versions of ProfoilGA and BezierGA.

Table 4 Performance Summary

Algorithm	C_d	Drag reduction w.r.t. the E168
ProfoilGA	0.01439	11.9%
Hybrid ProfoilGA	0.01406	14.0%
BezierGA	0.0147	10.0%
Hybrid BezierGA	0.01425	12.8%
Nelder-Mead	0.01478	9.5%

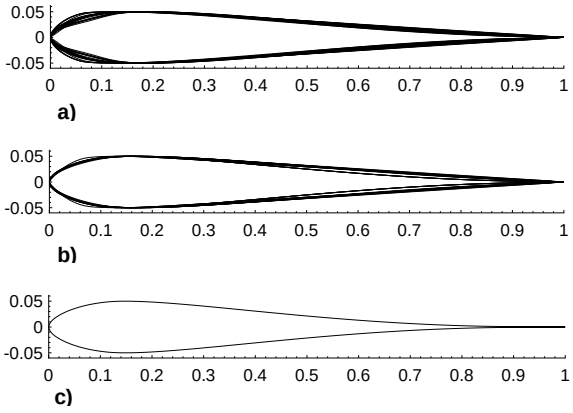

Fig. 11 Airfoils generated by Hybrid-ProfoilGA at different stages: (a) airfoils in the first generation, (b) best airfoils for each generation, and (c) final optimal airfoil.

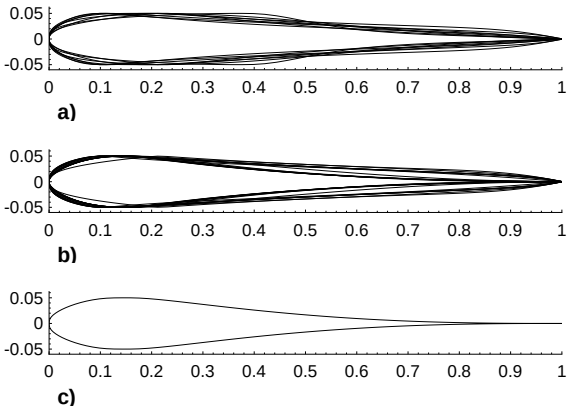

Fig. 12 Airfoils generated by Hybrid-BezierGA at different stages: (a) airfoils in the first generation, (b) best airfoils for each generation, (c) final optimal airfoil.

The actual airfoils generated by the hybrid version of ProfoilGA are shown in Figs. 11a–c. Figure 11a shows all of the airfoils generated in the first generation. Figure 11b shows the best airfoils for each generation. Figure 11c shows the final optimal airfoil. Similar graphs are shown for the BezierGA in Figs. 12a–c. Note that the aft portions of the optimal airfoils generated for both methods are very thin. While this type of geometry is not usually desirable, there are application where it can be useful, namely, the design of partially-double-surfaced sails. If airfoils with thicker aft regions are desired, geometry constraints can easily be implemented into both methods.

Recall that the airfoils shown in Figs. 11a and 12a where randomly generated. The fact that all of the airfoils in Fig. 11a have maximum thickness locations close to the optimal value is a result of the values chosen for the ν parameters. As mentioned previously, values of 22.5, 24.5, 28.5 and 30.0 were chosen based on experience. Hence, an advantage of using the ProfoilGA method is that a user familiar with inverse design techniques can significantly reduce the design space by constraining the design variables. However, if nothing is known a priori about the optimal velocity distribution, then the performance of BezierGA is generally equivalent to ProfoilGA. The XFOIL-predicted drag polars corresponding to the airfoils shown in Figs. 11c and 12c are shown in Fig. 13 compared with the E168.

Figures 14a–c display the results for the best case examined—the case representing the hybrid version of ProfoilGA shown in Fig. 10. Figure 14a shows the dispersion of individuals at each generation. Instant increases in diversity within the population are visible at generation 51 and generation 93. This phenomenon is caused by the use of the micro-GA operator. The

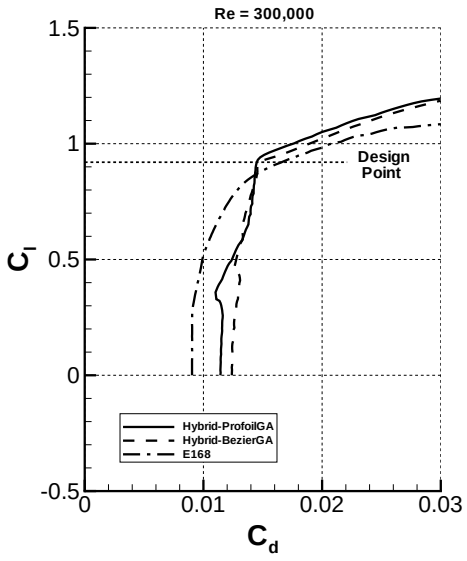

Fig. 13 Drag polars (XFOIL) for the respective optimized airfoils for ProfoilGA and BezierGA.

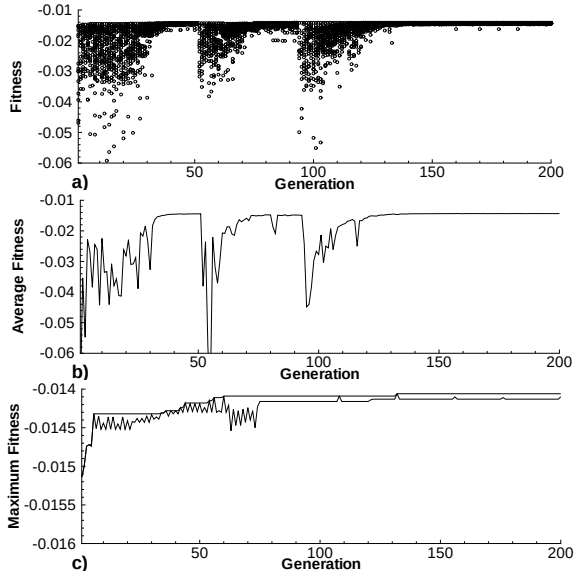

Fig. 14 Hybrid-ProfoilGA iteration history: (a) fitness values of every individual within each generation, (b) average fitness values of the individuals for each generation, and (c) the maximum fitness value of the individuals for each generation.

micro-GA operator was originally developed to generate diversity for very small population sizes.[21] A micro-GA begins with a randomly generated population and continues in normal GA fashion until convergence is reached. At this point, a new random population is created while keeping the best individual from the previous generations. As can be seen in Figs. 14a,b, the GA began to converge slightly before generation 50 and again near generation 93. For the cases discussed in this paper, micro-GA convergence was determined when less than 5% of the bits of the individuals within a population were different than the best individual. The effect of the new micro-GA population on performance can be seen in Fig 14c. There is a rapid increase in performance just after generation 50. However, there is no increase in performance at generation 93.

The history of the maximum fitness values for each generation is shown in Fig. 14c. There are two curves shown. The more jagged curve shows the maximum fitness of the individuals of each specific generation. The smoother curve shows the maximum fitness value for all individuals up to that point. The jagged curve shows somewhat large variations even though elitism was specified. The reason for this is that, for the case shown, Lamarckian learning was implemented. When an individual was enhanced using local search, the terminating value had to be backsubstituted into the population. The local search operated on real numbers, so the values of the design variables associated with the terminating solution had to be converted into binary in order to substitute the solution into the population. Since the binary conversion involves interpolation, the precision of the terminating values of the design variables can be reduced through the conversion process. Precision can be increased if the number of possible values associated with each control variable is increased, albeit at the cost of an increased design space for the GA to search.

Cambered Airfoil Design

As mentioned previously, a second objective of this research was to see if ProfoilGA could find an airfoil that outperforms the current state-of-the-art. After ProfoilGA was tuned by the test cases presented in the last section, the performance of ProfoilGA was measured against an optimized cambered airfoil with known design specification. This test is different than the previous section in that the goals of the designer for the benchmark airfoil are known. Thus, the real effectiveness of ProfoilGA could be compared to a current, modern airfoil design technique. Figure 15, from the paper by Selig, et al.,[22] shows the performance of many existing airfoils for different design lift coefficients at a Reynolds number of 300,000. As can be seen in the figure, the SG604x family of airfoils represents a bound on the current state-of-the-art. A goal of this research was to see if ProfoilGA could find airfoils that could extend this bound. The SG6042 was chosen as the benchmark airfoil for this test. The optimization problem formulated as follows: find a 10% thick cambered airfoil with minimum drag at a lift coefficient of 0.92 and a Reynolds number of 300,000. XFOIL predicts $L/D = 104$ for the SG6042 section at the specified conditions.

The lower surface of the cambered airfoils could be parameterized many ways including the approaches

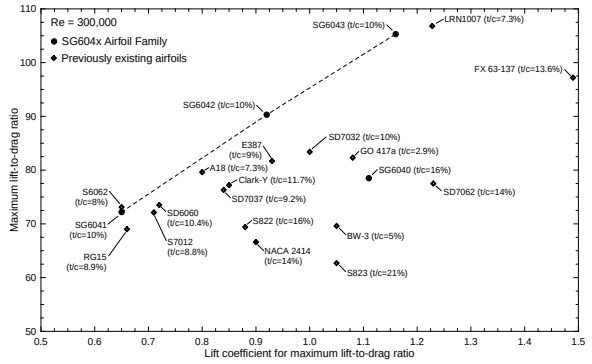

Fig. 15 Maximum lift-to-drag ratio versus the corresponding lift coefficient for the SG604x airfoil series as compared with several other airfoils.

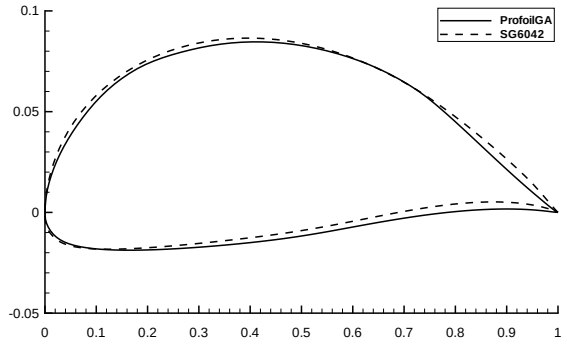

Fig. 16 Comparison of the optimal ProfoilGA airfoil with the SG6042 (expanded y-axis).

mentioned previously. For this research, the lower surface of the airfoil was determined by prescribing the boundary layer shape parameter through iteration on the lower surface velocity distribution. The approach taken is discussed in detail in Selig, et al.[22]

ProfoilGA was able to produce an airfoil that outperforms all published airfoils for the specified conditions. The optimum airfoil was found to have a drag coefficient of $C_d = 0.00875$ at a lift coefficient of $C_l = 0.92$. This corresponds to a lift-to-drag ratio of $L/D = 105$, which puts the airfoil beyond the boundary in Fig. 15. The geometry of the airfoil is compared to the SG6042 in Fig. 16 wherein the y-axis is expanded in order to show more detail. The ProfoilGA-generated airfoil has an L/D that is about 1% greater than that of the SG6042. While this performance difference may seem small, there are many applications where this margin of improvement is very significant. However, this increase in performance at the design condition is countered by a decrease in off-design performance, as can be seen in Fig. 17. This situation was expected because the SG6042 was highly optimized for the given conditions and ProfoilGA was not concerned with off-design drag behavior. However, this does not mean that ProfoilGA is a single-point optimization technique. Any quantifiable off-design characteristics can easily be implemented in the ProfoilGA code in the form of constraints.

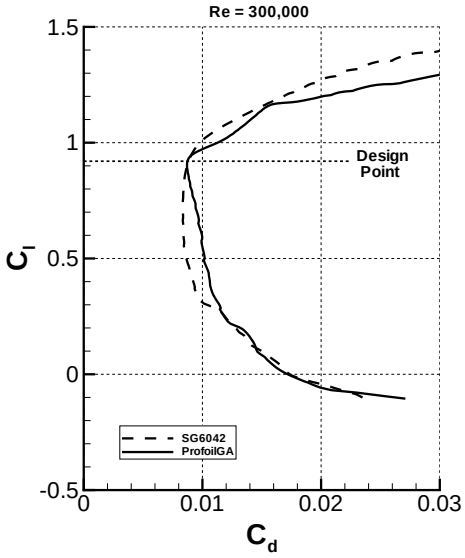

Fig. 17 Comparison of the XFOIL-predicted drag polars for the SG6042 and the optimized cambered airfoil of ProfoilGA.

Conclusions

Results from over 400 cases indicate that both ProfoilGA and BezierGA have potential to find solutions that improve upon existing airfoils. ProfoilGA was successfully used to find an airfoil with a 14% reduction in drag from the comparison E168 airfoil at $C_l = 0.92$ and $Re = 300{,}000$. Trade studies showed that ProfoilGA finds solution more reliably and in shorter time than BezierGA, so long as the user is familiar with inverse design. This result is due primarily to the smaller design space that ProfoilGA must search. Only five design variables were necessary to parameterize candidate symmetric airfoils for the conditions specified in this paper.

Both ProfoilGA and BezierGA were able to benefit from hybridization of the GA driver. Due to the nonlinearity and discontinuity of the design spaces, only non gradient-based local search schemes were effective. Care must be taken when creating a hybrid scheme for ProfoilGA or BezierGA. Proper choices of GA and hybridization parameters will cause a significant increase in performance, while poor choices will degrade performance substantially.

The design variables used by ProfoilGA allow it to directly control the velocity distribution. PROFOIL is used as the inverse method because of its Newton iteration capabilities. These schemes are used to fulfill geometry constraints by direct iteration on the velocity distribution. The schemes also aid in reducing the design space. XFOIL is used to analyze candidate airfoils

for assignment of a fitness value. Although XFOIL is very fast compared with RANS methods, it still remains the "bottleneck" of the process. One ProfoilGA case with 50 individuals and maximum of 300 generations typically requires 6 hr of processor time on a 1.7 GHz desktop PC.

The real advantages of the ProfoilGA method are realized when designing cambered airfoils. PROFOIL includes boundary-layer-development iteration schemes that can be used to significantly reduce the number of parameterization variables for the lower surface, thus substantially reducing the search space as compared with direct-design methods. This method has been used successfully to find an airfoil that surpasses, albeit slightly, the current state-of-the-art.

References

[1]Topliss, M.E., Toomer, C.A., and Hills, D.P., "Rapid Design Space Approximation for Two-Dimensional Transonic Aerofoil Design," *Journal of Aircraft*, Vol. 33, No. 6, 1996, pp. 1101–1108.

[2]Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, 1998, pp. 233–260.

[3]Holst, T.L. and Pulliam, T.H., "Aerodynamic Shape Optimization Using a Real-Number-Encoded Genetic Algorithm," AIAA 2001-2473, June 2001.

[4]Fanjoy, D.W. and Crossley, W.A., "Aerodynamic Shape Design for Rotor Airfoils via Genetic Algorithm," *Journal of the American Helicopter Society*, Vol. 43, No. 3, July 1998, pp. 263–270.

[5]Viccini, A., and Quagliarella, D., "Inverse and Direct Airfoil Design Using a Multiobjective Genetic Algorithm," *AIAA Journal*, Vol. 25, No. 9, 1997, pp. 1499–1505.

[6]Viccini, A. and Quagliarella, D., "Airfoil and Wing Design Through Hybrid Optimization Strategies," *AIAA Journal*, Vol. 37, No. 5, 1999, pp. 634–641.

[7]Sobieczky, H., "Parametric Airfoils and Wings." Recent Development of Aerodynamic Design Methodologies-Inverse Design and Optimization, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, Germany, 1999, pp. 72–74.

[8]Jones, B.R., Crossley, W.A., and Lyrintzis, A.S. "Aerodynamic and Aeroacoustic Optimization of Airfoils via a Parallel Genetic Algorithm," AIAA 98-4811, 1998.

[9]Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Massachusetts, Jan 1989.

[10]Carroll, D., "Fortran Genetic Algorithm Driver," CU Aerospace, Urbana, IL. Available online from: http://cuaerospace.com/carroll/ga.html, 1995–

[11]Selig, M.S. and Maughmer, M.D., "A Multi-Point Inverse Airfoil Design Method Based on Conformal Mapping," *AIAA Journal*, Vol. 30, No. 5, May 1992, pp. 1162–1170.

[12]Selig, M.S. and Maughmer, M.D., "Generalized Multipoint Inverse Airfoil Design," *AIAA Journal*, Vol. 30, No. 11, Nov 1992, pp. 2618–2625.

[13]Drela, M., "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils," *Low Reynolds Number Aerodynamics*, ed. by T. J. Mueller, Vol. 54 of Lecture Notes in Engineering, Springer-Verlag, New York, June 1989, pp. 1–12.

[14]Eppler, R. and Somers, D.M., "A Computer Program for the Design and Analysis of Low-Speed Airfoils," NASA TM 80210, Aug 1980.

[15]Eppler, R., *Airfoil Design and Data*, Springer-Verlag, New York, 1990.

[16]Goldberg, D.E., "Genetic and Evolutionary Algorithms Come of Age," *Communications of the ACM*, Vol. 37, No. 3, 1994, pp. 113–119.

[17]Nelder, J.A. and Mead, R., "A Simplex Method for Function Minimization," *Computer Journal*, Vol. 7, No. 4, 1965, pp. 308–313.

[18]Brent, P.B., *Algorithms for finding zeros and extrema of functions without calculating derivatives*, PhD Thesis, Stanford University, 1971.

[19]Goldberg, D.E., and Voessner, S., "Optimizing Global-Local Search Hybrids," *Proceedings of the Genetic and Evolutionary Computation Conference*, 1999, pp. 220–228.

[20]Sinha, A., and Goldberg, D.E., "Verification and Extension of the Theory of Global-Local Hybrids," *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001, pp. 592–598.

[21]Carroll, D.L., "Genetic Algorithms and Optimizing Chemical Oxygen-Iodine Lasers," *Developments in Theoretical and Applied Mechanics, Vol. XVIII*, H.B. Wilson, R.C. Batra, C.W. Bert, A.M.J. Davis, R.A. Schapery, D.S. Stewart, and F.F. Swinson (ed.), School of Engineering, The University of Alabama, 1996, pp. 411–424.

[22]Selig, M.S., Gopolarathnam, A., Giguere, P., and Lyon, C. "Systematic Airfoil Design Studies at Low Reynolds Numbers," *Fixed, Flapping, and Rotary Wing Vehicles at Very Low Reynolds*, T.J. Mueller (ed.), Progress in Astronautics and Aeronautics, Vol. 195, AIAA, Reston, VA, 2001.