2003-0298

Comparison of Evolutionary (Genetic) Algorithm and Adjoint Methods for Multi-Objective Viscous Airfoil Optimizations

T. H. Pulliam, M. Nemec, T. Holst, and D. W. Zingg

41st Aerospace Sciences Meeting
January 6-10, 2003/Reno, NV

Comparison of Evolutionary (Genetic) Algorithm and Adjoint Methods for Multi-Objective Viscous Airfoil Optimizations

T. H. Pulliam*, M. Nemec†, T. Holst‡, and D. W. Zingg§

A comparison between an Evolutionary Algorithm (EA) and an Adjoint-Gradient (AG) Method applied to a two-dimensional Navier-Stokes code for airfoil design is presented. Both approaches use a common function evaluation code, the steady-state explicit part of the code,ARC2D. The parameterization of the design space is a common B-spline approach for an airfoil surface, which together with a common griding approach, restricts the AG and EA to the same design space. Results are presented for a class of viscous transonic airfoils in which the optimization tradeoff between drag minimization as one objective and lift maximazation as another, produces the multi-objective design space. Comparisons are made for efficiency, accuracy and design consistency.

Introduction

THE main focus of this paper is a comparison of multi-objective optimization between an Evolutionary Algorithm (**EA**) and an Adjoint-Gradient (**AG**) method applied to a two-dimensional Navier-Stokes code for airfoil design. The **EA** used here is a genetic algorithm approach[1] coupled with two multi-objective optimization methods: a Weighted-Objective-Function (**WOF**) Pareto optimal set technique and a Dominance-Pareto-Front (**DPF**) technique. The **AG** approach is described in detail by Nemec[3] and is coupled with the **WOF** technique for multi-objective optimization. Both approaches use a common function evaluation code, the steady-state explicit operator from ARC2D,[8] which employs second-order finite-differences, artificial dissipation, and the Spalart-Almaras turbulence model. The parameterization of the design space is taken from the B-spline approach[2] for an airfoil surface, which together with a common griding approach, restrict the **AG** and **EA** to the same design space.

A characterization of the two distinct approaches to multi-objective optimization would be:

1. The **EA** [6][7][1] method, which using a genetic algorithm to produce a subtle interaction of explorations and exploitations of a design space.
2. The **AG** method, employing a discrete-adjoint and flow-sensitivity approach[2],[3] which attempts to estimate the slope of the landscape of a design space and follow the terrain to find local minima or maxima.

Both approaches have been extensively applied to single-objective optimization problems. The focus here will be on multi-objective optimization, which has an obvious application to multi-disciplinary problems [1]. The comparison for the two approaches will be made for the computation of a Pareto optimal set ("Pareto front"), of a two-objective optimization (one a function of lift, the other a function of drag) within the framework of a two-dimensional viscous transonic airfoil computation. In the context of Pareto optimal sets; the "Pareto front" is the set of solutions found in the design space which are *non-inferior* or *non-dominate* in both objective measures, i.e., there is no feasible solution which would decrease one objective without causing a simultaneous decrease of the other. The set of minimal non-dominated solutions constitute the "Pareto front".

The two approaches employed to compute the Pareto front are:

WOF: The method of Weighted Objective Functions,[5] sometimes called Aggregating Functions. In this method, the two objectives are combined with weights (summing to 1.0) to form a new objective function. The aggregated objective is then optimized as a single objective-optimization. This method has been demonstrated to work well for non-convex design spaces, but suffers for convex spaces and is subject to difficulties in the choice of weights, especially if the objectives are not properly scaled. Fortunately, for our application, the design space is non-convex and well scaled.

DPF: A Dominance Pareto Front technique,[9],[6][7] typically applied in conjunction with an **EA** algorithm, employs non-dominated sorting and selection coupled with a genetic algorithm to move a population toward

*NASA Ames Research Center, USA, tpulliam@mail.arc.nasa.gov, AIAA Associate Fellow

†U. of Toronto Institute of Aerospace Studies, Canada, Member AIAA

‡NASA Ames Research Center, USA, AIAA Fellow

§U. of Toronto Institute of Aerospace Studies, Canada, Member AIAA

[1] The number of references on Evolutionary Algorithms as applied to multi-objective optimization are too numerous to present here, see http://www.lania.mx/~ccoello/EMOO/EMOObib.html problems for a large repository of papers on the subject.

the Pareto front. This method does not suffer from convexity or scaling problems.

Design Process

OPTIMIZATION of a two-dimensional viscous transonic airfoil is studied where the objective is to develop a class of airfoils which have trade offs between minimization of drag and a maximization of lift. The flow conditions and modeling characteristics (e.g. turbulence model, numerical algorithm) are described below.

Design Space

A B-spline representation of the airfoil geometry is employed, details can be found in Nemec.[3] Briefly, the airfoil surface is defined by a number of control points connected by a B-spline function representation based on a default surface and grid system. Typically 10 control points are distributed over the upper and lower surface, labeled D_1 through D_{10} (starting at the lower surface trailing edge and going clockwise to the upper surface trailing edge). The control points along with angle of attack (α) constitute the design space parameters for the optimization. As the optimization progresses, perturbed control points produce a new geometry (the default grid is locally reclustered in the surface normal direction) combined with a new α defining a candidate design, which is driven by the various optimization processes defined below. The parameterization of the design space together with a common volume grid procedure, restricts all the optimization approaches to the same design space.

Objective Space

A common function evaluation code is used for all the methods, the explicit steady-state operator from ARC2D[8] (a second-order finite-difference, artificial dissipation code which uses the Spalart-Almaras turbulence model). The code produce values of lift, C_l, and drag, C_d, which are in the fitness functions defined below.

In the case of **EA** optimization, the viscous Navier-Stokes code ARC2D is used to compute a flowfield, in particular the lift and drag, and performs the function evaluation. For the **AG** optimization approach the Newton-Krylov approach of Pueyo,[4] which employs the same explicit steady-state operator from ARC2D, is used for the flow field integrator. Consistency between the two methods was verified and identical flow solver parameters were used. In fact, both approaches have identical fixed point solution spaces when provided identical parameterizations and input data.

Fitness Function

Fitness functions (one for lift F_l and one for drag F_d) are

$$F_l = (1 - C_l/C_l^*)^2 + \Im \quad F_d = (1 - C_d/C_d^*)^2 + \Im$$

where the targets are $C_l^* = 0.55, C_d^* = 0.0095$ and $\Im$ is a thickness constraint designed to increase when a minimum thickness distribution is violated. The thickness constraint limits are chosen so that the design can achieve the target C_l^*, but not C_d^*. Further details for these objective functions can be found in Nemec.[3]

Adjoint Gradient Method

THE **AG** method is described in detail in Nemec,et.al.[2] and Nemec.[3] The Newton–Krylov algorithm consists of four modules: 1) design variables and grid perturbation, 2) flow solver, 3) gradient computation, and 4) optimizer. The design variables are based on the B-spline parameterization of the airfoil.

The discretized Navier–Stokes and turbulence model equations are differentiated by hand, and the adjoint method is used to compute the objective function gradient. The preconditioned GMRES method is applied to solve not only the flow equations, where it is used in conjunction with an inexact-Newton method, but also the adjoint equation. The accuracy of the gradient is verified by comparison with gradients based on the finite-difference and flow-sensitivity methods. The optimization problem is cast as an unconstrained problem by using quadratic penalty functions. A BFGS quasi-Newton optimizer is used to solve the unconstrained problem. A detailed evaluation of the algorithm was performed[3] with emphasis on the accuracy and efficiency of the gradient computation and the efficiency of the flow solver. The resulting algorithm provides a highly efficient approach for aerodynamic design problems governed by the Navier–Stokes equations. The preconditioning strategy, in particular, has been optimized for both the flow solution and the gradient evaluation. Nemec[3] provides a detailed development and validation of the design process for single and multi-point airfoil optimization.

Genetic Algorithm

A generational genetic algorithm[1] is used to drive the **EA**. It combines a number of ranking and selection techniques, mutations and perturbations performed on the exploited "chromosomes" producing the exploration set for the next generation. In the case of the **DPF** results, a chromosome archival strategy is also used, where new points found on the Pareto front are stored in an accumulation file producing a well defined front. The archival file can be used in the ranking process and also as part of the selection pool. The flow code ARC2D[8] is used to evaluate a chromosome for it's fitness to be used in the next generation's ranking. Efficiency is obtained on a parallel computing system by parallel evaluation of each chromosomes fitness. Most of the results shown for the **EA** cases were performed on 16, 32 or 64 processors of a workstation cluster, where the number of processors chosen was equal to the population size of the optimization.

Briefly, a set of "chromosomes" ("population") consisting of the design parameters "genes" is first processed by the objective function, ARC2D, producing "fitness" values (F_l, F_d). The "chromosomes" are then ranked by their "fitness," a new intermediate generation is selected from the current "population," using a number of selection algorithms (e.g. tournament selection, where 3 randomly chosen "parents" compete based on ranking, the winners surviving to form the intermediate generation). The intermediate generation is then processed by cross-over and mutation strategies to produce the the new generation of "chromosomes". The genetic algorithm process is repeated until convergence. Convergence is hard to define for algorithms of this type. Typically, convergence of a genetic algorithm is assumed when the objective function ceases to improve, although this is not always a good measure, since the genetic algorithm can wander around for a rather long time, until new region of convergence is found. There are a number of genetic algorithm parameters involved, such as, number of "chromosomes" in a generation, probability of selection, mutation, cross-over, and convergence criteria. The genetic algorithm parameters were chosen to optimized the convergence of the **EA** , see Holst and Pulliam.[1] Also, in the context of the objective function ("fitness"), a flow solver, such as ARC2D, involves a large number of input variables controlling accuracy, convergence of the flow computation, physical modeling etc. In the results presented below, we freeze the flow solver (ARC2D) parameters, except where noted, so that the design space has consistency over the various optimizations

Optimization Processes

THE two Pareto front calculation techniques are combined with the optimization algorithms to produce three Pareto optimization methods: **AG-WOF** , **EA-WOF** and **EA-DPF** .

AG-WOF : The **AG** is coupled with the **WOF** to produce a Pareto front. The two objective functions F_l, F_d are combined to form an aggregate objective function $F = W_l * F_l + W_d * F_d$ with a set of weighting coefficients:

$$\{(W_l, W_d) : (0.99, 0.01), (0.90, 0.10), (0.80, 0.20), (0.70, 0.30), (0.60, 0.40), (0.55, 0.45), (0.50, 0.50), (0.40, 0.60), (0.45, 0.55), (0.30, 0.70), (0.20, 0.80), (0.15, 0.85), (0.10, 0.90), (0.05, 0.95), (0.01, 0.99)\}$$

The **AG** optimization method is then applied to the aggregated objective to find optimal results for discrete points on the Pareto front.

EA-WOF : The **EA** is coupled with the **WOF** using the same weighting procedure as **AG-WOF** .

EA-DPF : The **EA** is applied with the **DPF** to produce the Pareto front directly. The Pareto dominance approach of Goldberg[9] is employed to establish ranking for the **EA** . In this case, the Pareto front will be defined as a set of discrete points which in some sense represent an equivalence to **WOF** set of results. This is only an appropriate analogy for non-convex simple fronts.

Results

RESULTS are presented for a viscous airfoil optimization of lift and drag. The flow conditions are a Mach number, $M_\infty = 0.7$, and a Reynolds number, $Re = 9 \times 10^6$. The angle of attack, α, is one of the design variables. A C-grid consisting of 201 by 45 points is used.

Pareto Front Comparison

Figure 1 shows the computed Pareto fronts from the three approaches. Results from **AG-WOF** may be considered the "true Pareto front," since the **AG** results are driven to a zero gradient at convergence. The large symbols refer to the specific results for the various weighting ratios as given above. The **EA-WOF** results compare well with the "true Pareto front," except in the region of large W_l weighting where the objective function F_l goes to zero. This region is sensitive, since without the drag constraint, the maximized lift problem ($F_l = 0$) is ill-defined (an infinite number of airfoils can satisfy the optimization). In general, the **EA-WOF** results show excellent agreement with the **AG-WOF** results. Results from **EA-DPF** are also a good approximation to the "true Pareto front" and produce a well populated set of optimizations along the Pareto front. In fact, one of the strengths of the **EA-DPF** method is the automatic development of a large class of optimization results, which can be evaluated by the designer. On the other hand, when the gradient can be formed (a difficult task in general, but possible as demonstrated in Nemec[3]), the **AG-WOF** method can more efficiently obtain specific designs, i.e., in the case of non-convex Pareto fronts.

A selection of the weighing coefficients is indicated in Figure 1, with black lines pointing to cases where **AG-WOF** and **EA-WOF** produce similar results, red **AG-WOF** and blue **EA-WOF** arrows where they differ, and a single blue arrow at $\{W_l, W_d\} = \{0.02, 0.98\}$ a point added to the **EA-WOF** case to fill in the results in that region. The results from **EA-DPF** consist of about 500 points along the Pareto front, fairly well distributed, and demonstrate a strength of the dominance approach, i.e., no weighting choices have to be made to produce a reasonable front.

Figure 2 shows a comparison of $-C_l$, (we plot negative lift to emphasize the minimization aspect of the optimization) against C_d for the three methods. Specific values of C_l, C_D, α, F_l, and F_d are shown in Tables

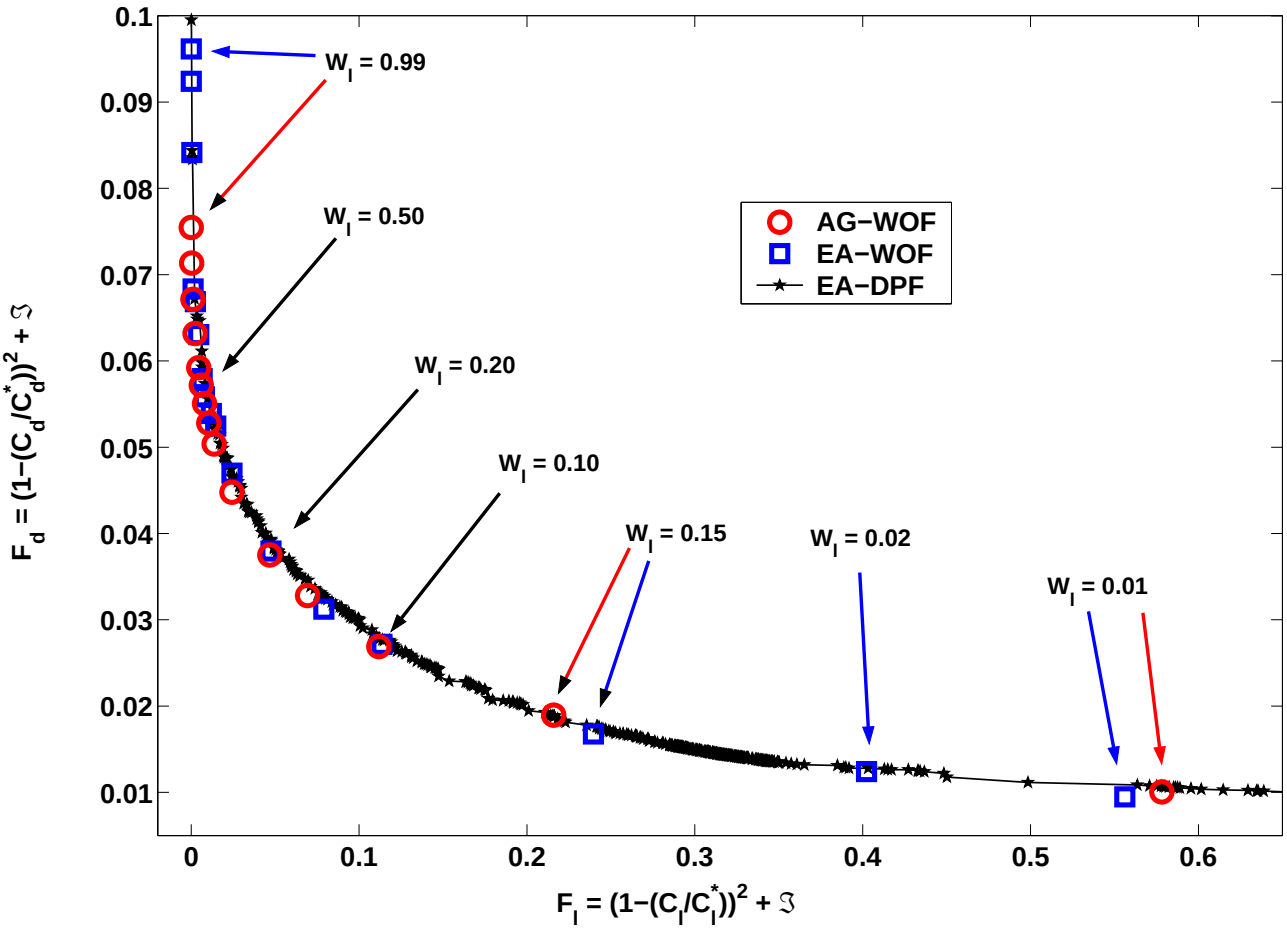


Fig. 1 Pareto Front Calculation

1 and 2 for the **AG-WOF** and **EA-WOF** methods. The results from **EA-DPF** are too numerous to show in a table format and will be discussed in more detail below. For the two similar approaches **AG-WOF** and **EA-WOF** the comparison are excellent for C_l and C_d. The objective functions F_l and F_d are also is good agreement. Somewhat interesting is the disparity in terms of α and by inference the details of the designed airfoil shapes. A further discussion of these results is given below.

Work Estimates

Before proceeding with further comparisons, we will discuss timing or work estimates. The three design optimization methods (**AG-WOF** , **EA-WOF** and **EA-DPF**) are composed of similar components (they all use the same explicit function evaluation of ARC2D, design space representations and grid generation), but they have enough dissimilar components (e.g., adjoint gradient in contrast to genetic algorithm process) to make detailed CPU timings unreliable and misleading. Results for the **AG-WOF** cases will be presented in terms of **design cycles** , which include, the design space variations, flow solves, gradient (adjoint) evaluations and optimizer steps. Work estimates from the **EA-WOF** and **EA-DPF** cases are

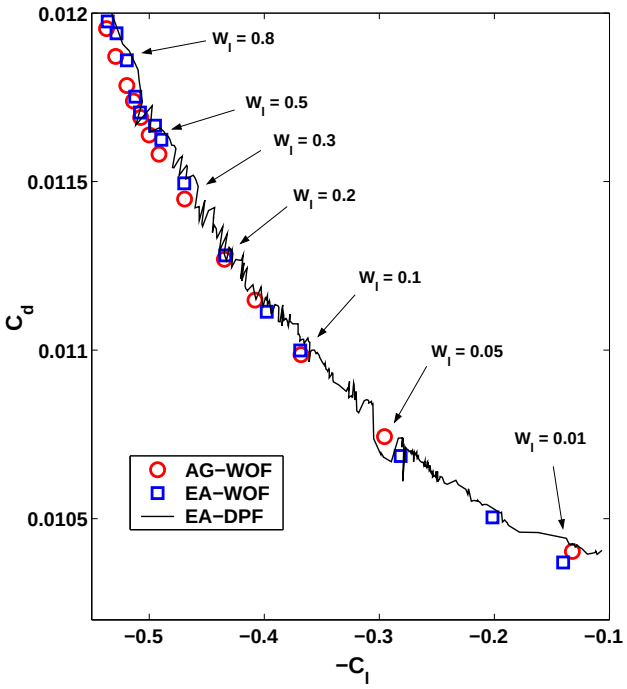


Fig. 2 Lift-Drag comparison for the three methods.

Table 1 Table 1: AG-WOF Results

W_l	C_l	C_d	α	F_l	F_d
0.99	0.5494	0.01211	0.286	0.0000	0.0754
0.90	0.5439	0.01203	0.264	0.0002	0.0713
0.80	0.5371	0.01195	0.243	0.0009	0.0671
0.70	0.5291	0.01187	0.221	0.0023	0.0631
0.60	0.5194	0.01178	0.193	0.0044	0.0592
0.55	0.5137	0.01174	0.180	0.0059	0.0571
0.50	0.5073	0.01169	0.166	0.0079	0.0550
0.45	0.5000	0.01164	0.152	0.0104	0.0527
0.40	0.4915	0.01158	0.135	0.0136	0.0503
0.30	0.4693	0.01145	0.090	0.0242	0.0447
0.20	0.4346	0.01127	0.023	0.0467	0.0375
0.15	0.4080	0.01115	-0.007	0.0693	0.0328
0.10	0.3680	0.01099	-0.055	0.1117	0.0268
0.05	0.2955	0.01074	-0.140	0.2159	0.0189
0.01	0.1321	0.01040	-0.261	0.5781	0.0100

Table 2 Table 2: EA-WOF Results

W_l	C_l	C_d	α	F_l	F_d
0.99	0.5491	0.01245	2.123	0.0000	0.0961
0.90	0.5437	0.01221	1.884	0.0004	0.0814
0.80	0.5361	0.01197	0.406	0.0011	0.0683
0.70	0.5284	0.01194	1.333	0.0024	0.0668
0.60	0.5194	0.01186	1.309	0.0044	0.0630
0.55	0.5120	0.01175	0.712	0.0064	0.0579
0.50	0.5079	0.01170	0.434	0.0078	0.0558
0.45	0.4949	0.01167	1.000	0.0119	0.0539
0.40	0.4894	0.01162	1.063	0.0145	0.0524
0.30	0.4695	0.01149	0.984	0.0242	0.0470
0.20	0.4338	0.01128	0.420	0.0474	0.0379
0.15	0.3978	0.01111	0.192	0.0789	0.0312
0.10	0.3687	0.01010	-0.003	0.1109	0.0272
0.05	0.2814	0.01069	-0.115	0.2395	0.0167
0.01	0.1402	0.01037	-0.275	0.5561	0.0094

given in terms of **generations**, which include, design space variations, flow solves, and genetic algorithm processing. Work estimates for **design cycles** and **generations** are given in terms of explicit ARC2D evaluations, this is the time it takes to form the steady state right-hand-side in ARC2D (**RHS-Evals**). The **AG-WOF** requires approximately 1,000 **RHS-Evals** per **design cycle**, while **EA-WOF** and **AG-WOF** require approximately 10,000 **RHS-Evals** per generation. Typically 100 **design cycles** are used for complete design using **AG-WOF** resulting in 100,000 **RHS-Evals** per design. In contrast, **EA-WOF** requires about 300 **generations** for a design, resulting in 3,000,000 **RHS-Evals**, a factor of 30 times that of an **AG-WOF** design. These are just ballpark estimates. Part of the time advantage of **AG-WOF** over **EA-WOF** comes from the efficiency of the Newton Krylov algorithm used for the flow solver and adjoint solution, compared to the approximate factorization time integration used by ARC2D. Typically, the Newton Krylov solver is an order of magnitude faster per Navier–Stokes solution compared to the default approximate factorization solver in ARC2D.

Another aspect of the work associated with each of these methods are the requirements on the flow solver. The **AG-WOF** method requires complete convergence of the flow solver at each **design cycle**, about 10 orders of magnitude drop in the flow solver residual. It also requires a 3-4 order of magnitude drop in the gradient as a design convergence criteria. Smart restart strategies, in some cases, reduce the work required for subsequent designs along the Pareto front.

For the **EA** results, shown here, the flow solver residual is converged 10 order of magnitude as above, although this is not a requirement. In fact, partial convergence of the flow solver (ARC2D) can be used (especially in the early stages of the design). A 3-4 order of magnitude drop in residual of the flow solver is sufficient for a design. This can reduce the work of the **EA** designs by as much as a factor of three.

Parallelization

One advantage of the **EA** processes is its embarrassingly parallel nature. For example, if a population size of 32 is employed, each chromosome set can be parsed out to a separate processor on a 32 processor parallel system and simultaneously evaluated. This results in a direct linear scalability of processing and can make up for the factor of 30 between **AG-WOF** and **EA-WOF** - in wall clock turn around time. The **AG-WOF** process could also be parallelized, where different weights of the **WOF** Pareto front could be solved simultaneously. Parallelization also provides a degree of efficiency for the **EA-DPF** method, where multiple design points are simultaneously computed for approximately the same work. In fact, the amount of work for the **EA-DPF** results shown in Figure 1 is equivalent to two **EA-WOF** results (i.e. two weighting choices), and produces over 500 distributed points over the Pareto front.

Comparison of AG-WOF and EA-WOF

We focus now on a detailed comparison between **AG-WOF** and **EA-WOF** for one point in the Pareto front computation. The **WOF** approach is applied in both cases for each point generated on the front, and, in general, except near $W_l = 1.0$, the results from each method compare quite well. The $W_l = 0.5$, $W_d = 0.5$ case is chosen for comparison and typifies the other results.

The left side of Figure 3 shows the convergence for **AG-WOF** in **design cycles** for the total fitness function, F, the lift coefficient, C_l, the drag coefficient, C_d, angle of attack, α (one of the design variables), and two other design variables (the control points near

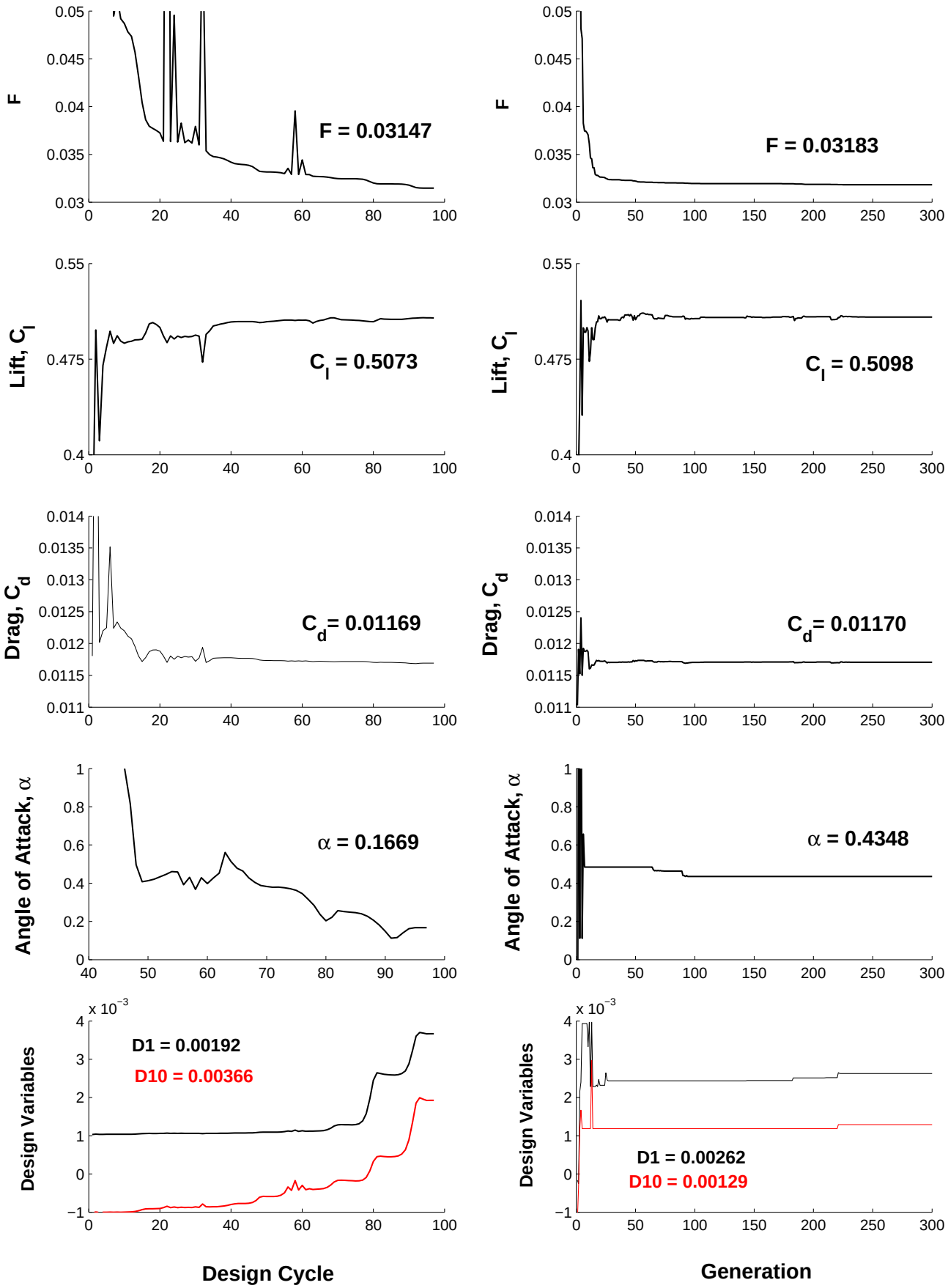


Fig. 3 Convergence for AG-WOF and EA-WOF, $W_l = 0.5$

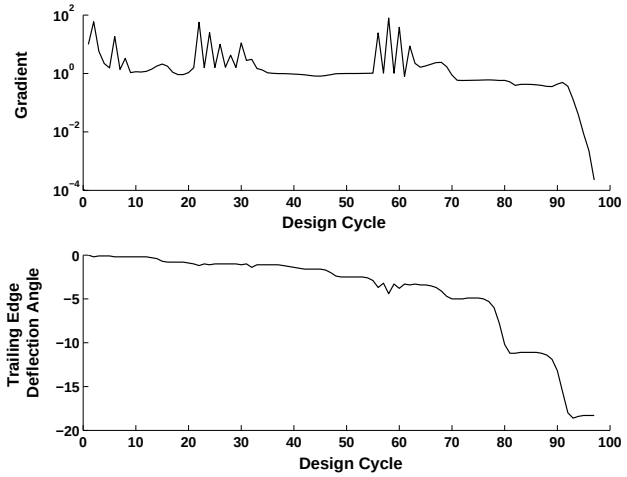


Fig. 4 Convergence of Gradient and Trailing Edge Angle for AG-WOF, $W_l = 0.5$

the trailing edge), D_1 and D_{10}. The right side of Figure 3 shows similar convergence results for **EA-WOF** in **generations**. The final values are shown in the plots for each of the results. Except in the case of α and the design variables, both method converge to consistent values.

Obviously, the convergence path of each method is different and specific comparisons would be fruitless. Instead, certain characteristics should be noted. The results for F in **AG-WOF** are somewhat noisy (due to activation of the thickness constraints, see[3]), but do show a steady convergence. In contrast, **EA-WOF** shows a rapid convergence of F (we are actually showing the F from the best chromosome in the **EA** which is guaranteed to be a monotonically decreasing function). Lift convergence is very consistent between the two methods, which is to be expected since the lift objective function is a strong forcing constraint. More interesting is the convergence of α and the two design variables, D_1 and D_{10}. As noted above, angle of attack α and the trailing edge region are the slowest to converge in both methods. Approximately, for **AG-WOF**, α steadily decreases, while for **EA-WOF**, α converges in distinct steps. Simultaneously, the two design variables, D_1 and D_{10} steadily increase for **AG-WOF** and also for **EA-WOF**.

Some insight into the convergence process can be found in Figure 4. Here we show the convergence for **AG-WOF** of the gradient and also the computed trailing edge deflection angle as a function of **design cycles**. The gradient shows slow convergence for 90 **design cycles** and then rapid convergence over the last 10 **design cycles**. The trailing edge deflection angle converges in a similar manner, although it does start it's rapid decent at about 70 **design cycles**.

This trade-off in the design space between the trailing edge deflection angle and angle of attack make the **EA-WOF** results slow to converge to the final α, D_1 and D_{10} of the "true Pareto front", i.e. the **AG-WOF** results. Figure 5 compares the airfoil surface obtained with the two methods. The general airfoil shape, leading edge, thickness, and camber compare well for both of the methods. The main differences occur near the trailing edge and are indicative of the slow convergence of **EA-WOF** for angle of attack and trailing edge deflection angle.

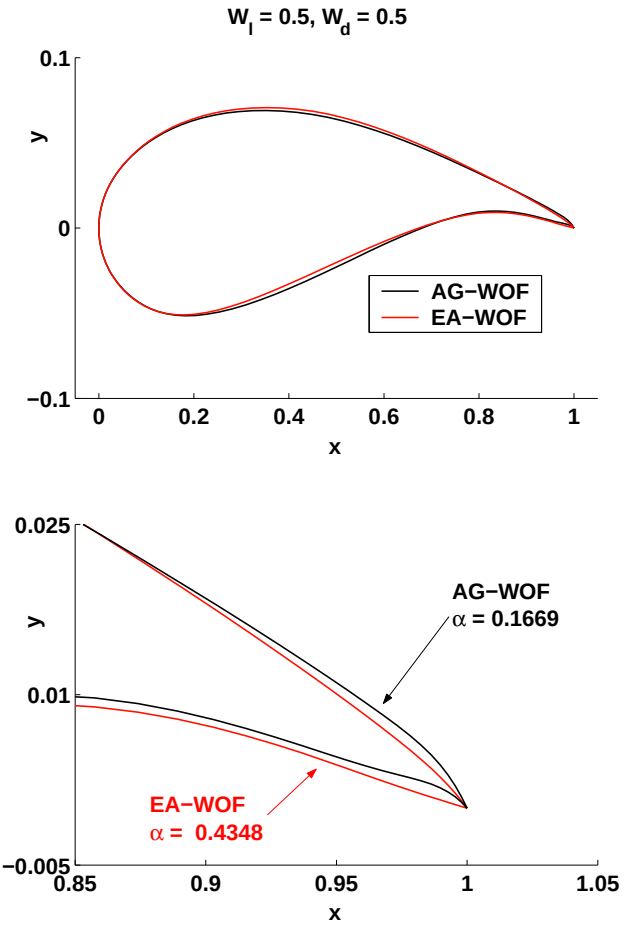


Fig. 5 Designed airfoil comparison for AG-WOF and EA-WOF, $W_l = 0.5$

The results for the other weightings are very similar to the $W_l = 0.5$ results presented above. The larger W_l weightings converge the slowest and therefore have the largest deviation from the **AG-WOF** results. Based on Figure 1, the two weighting methods (**AG-WOF** and **EA-WOF**) converge to the same Pareto front. Details in regions such as the trailing edge do not compare as well.

This slow convergence of the trailing edge region, and by inference the angle of attack, α, is typical of all the results. It may be a difficulty created by the choice of the design space parameterization or just a consequence of the Newton-like performance of the **AG-WOF** method, see Nemec.[3] It could also indicate a flat region in the design space, where the **AG** optimization slowly converges until the gradient converges. Such a region would also slow down an **EA**, which would be good at getting into the flat region, but

have difficultly in the later stages of finding an absolute extrema. The **EA-WOF** method relies on an exploration (somewhat stochastically) of the design space and suffers greatly in relatively flat design spaces. On the other hand **AG** optimization would have more difficulty than an **EA** in noisy design spaces, (since **AG** relies on smooth gradients), and multi-modal design spaces, (where **AG** could get stuck in a local but not global extrema).

Pareto Front Results From EA-DPF

The true strength of the **EA** approach for multi-objective optimization, is not in the area of a Weighted-Objective approach, rather it is in the computation of the total Pareto front in one total design integration –**EA-DPF** . Results for the **EA-DPF** are presented in Figure 6. The development of the Pareto front is shown after 5 10 and 100 **generations** . The results at 600 **generations** are declared converged by virtue of the lack of movement of the Pareto front. In fact, the Pareto front was fairly well established by about 400 **generations** , but to ensure the final development another 200 **generations** was computed. The complete search design space is shown as points. The search space is a plot of results from all the generations and shows all the locations in design space visited by the **EA** . Also shown is the "true Pareto front" from **AG-WOF** . Convergence of **EA-DPF** is rapid (approximately 100 **generations**) over most of the front. The front fills out and converges more slowly in the high F_l region. These results were obtained in approximately 6,000,000 evaluations which is a factor of 60 larger than a single **AG-WOF** design, but only a factor of 5 times more expensive than a Pareto front computation using the **AG-WOF** approach which produced approximately 15 points along the front. The final Pareto front shown for **EA-DPF** contains 500 individual designs and although some regions are more densely populated than others the entire curve is well represented. The **EA-DPF** approach does not rely upon a set of weightings and is a more hands off approach to finding a Pareto front. It still suffers from some of the poor convergence exhibited in **EA-WOF** for probably the same reasons.

The **EA-DPF** approach takes advantage of the economy of scale available because of the parallel nature of the computations. Multiple function evaluations (ARC2D runs) can be processed simultaneously across processors and thereby cover up much of the cost inherent in the **EA** as compared to **AG** optimization. The wall clock times can actually be less for a parallelized **EA** approach relative to a no-parallel **AG** approach, although parallelization of **AG-WOF** for multiple weightings could be employed also.

Figure 7 shows comparisons of results from **AG-WOF** at weightings $W_l = 0.01, 0.20, 0.50, 0.80$, respectively, with results taken from the **EA-DPF** Pareto front at points very close in fitness space to the **EA-WOF** values. In Figure 6, we have marked the locations of the comparisons, e.g. CASE $W_l = 0.20$. Also shown are the convergence characteristics of the ARC2D runs made to compute the analysis. In general, the results compare quite well in terms of basic airfoil shape, aerodynamic loads (C_l, C_d, coefficient of pressure, C_p) and designs. The main difference is again in the trailing edge region and the trade off between trailing edge deflection angle and angle of attack.

Conclusions

DETAILED comparisons of three approaches for multi-objective airfoil optimization have been presented. In terms of the approach to computing a Pareto front, the two similar methods, (**AG-WOF** and **EA-WOF**) produce consistent fronts and designs. The third method **EA-DPF** also produces a consistent front and is able to compute a large number of design points on the Pareto front for a reasonable cost. In general, the designs obtained are consistent across the approaches and except in the trailing edge region, they represent similar optimal designs.

The main purpose of this paper was to demonstrate the applicability of an **EA** approach for optimization and Pareto front computation and to contrast its performance and results with an **AG** approach. It was not to cast one method as being superior to another. Even within the rather restricted class of airfoil design using Navier-Stokes, there will be occasions where one method will be more appropriate than another based on speed or efficiency issues. The **AG** method is obviously more efficient than an **EA** approach on a point for point basis of the **WOF** for a Pareto front calculation. The **AG** does involve significant programming to develop a solver for the adjoint and a proper gradient operator. When this can be accomplished an **AG** approach will be superior and is a very capable and useful design tool. On the other hand, if the model (flow solver, design parameterization, etc) require significant modifications, the computation of the gradient may require significant recoding and analysis. The **EA** approach requires little if any modification to an existing flow solver and the actual genetic algorithm processing of chromosomes requires trivial additional work.

There are other issues which may favor the EA, but we have not explored or addressed them here. These include multi-modal design spaces and non-smooth design spaces. Future work could concentrate on characterizing the two approaches for design problems with these characteristics (if we can find such problems!).

Finally, we may conclude that measuring convergence based on the achieved level of the objective function for **AG-WOF** is be misleading. The objective function converges rapidly while the gradient is

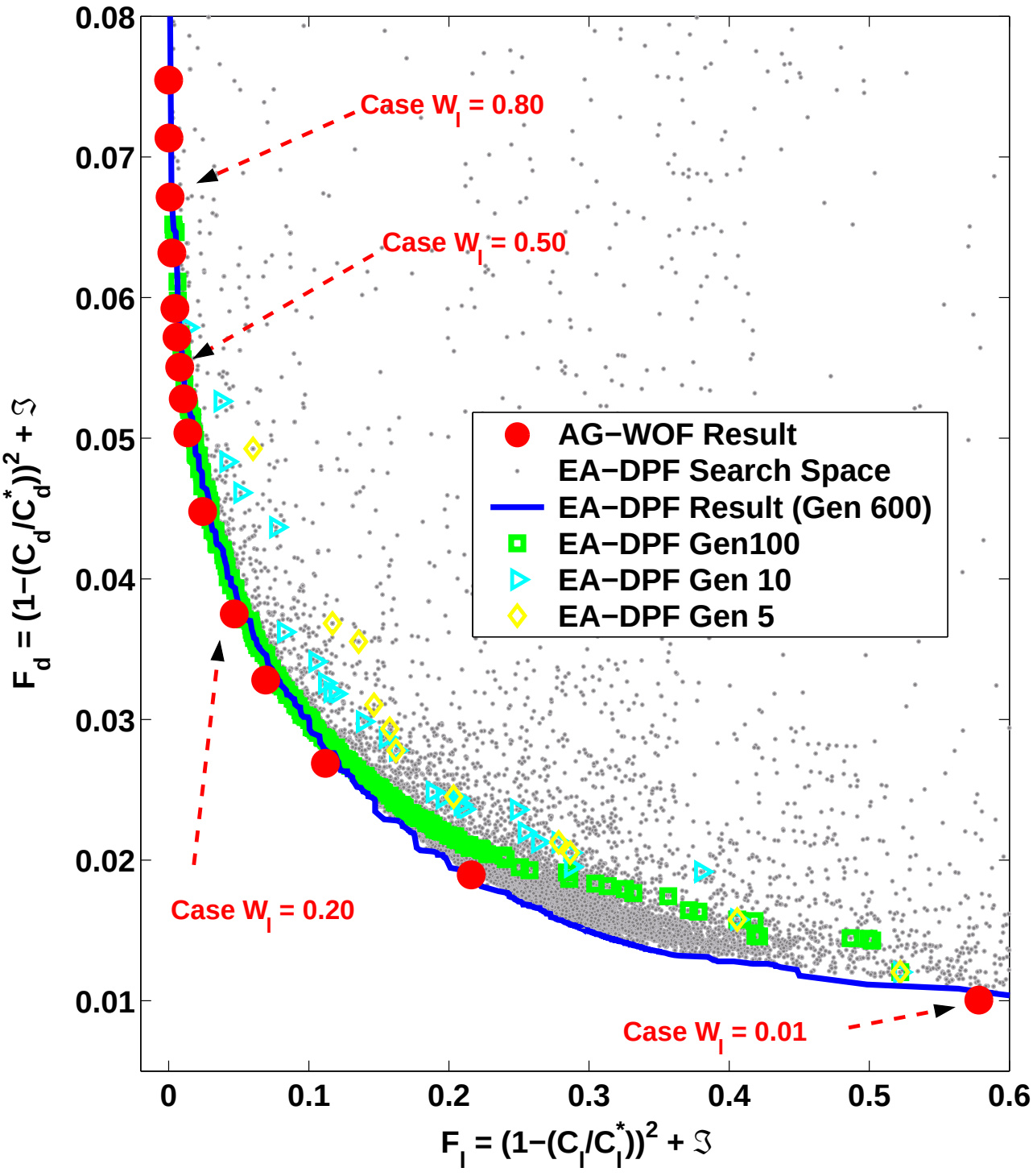
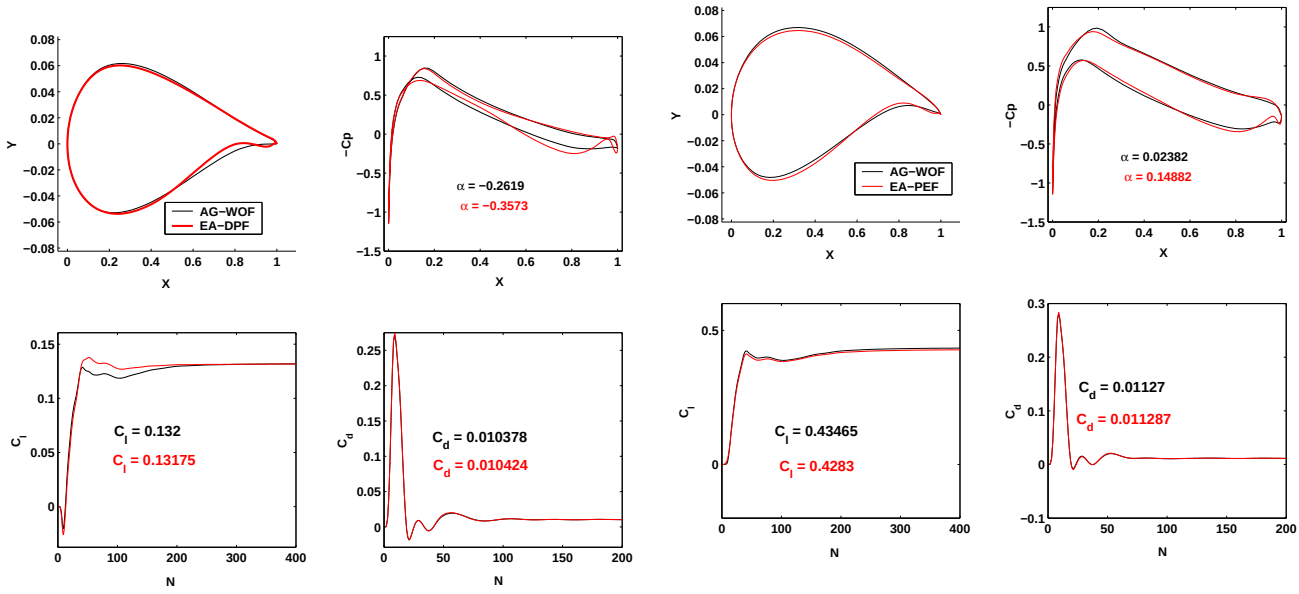
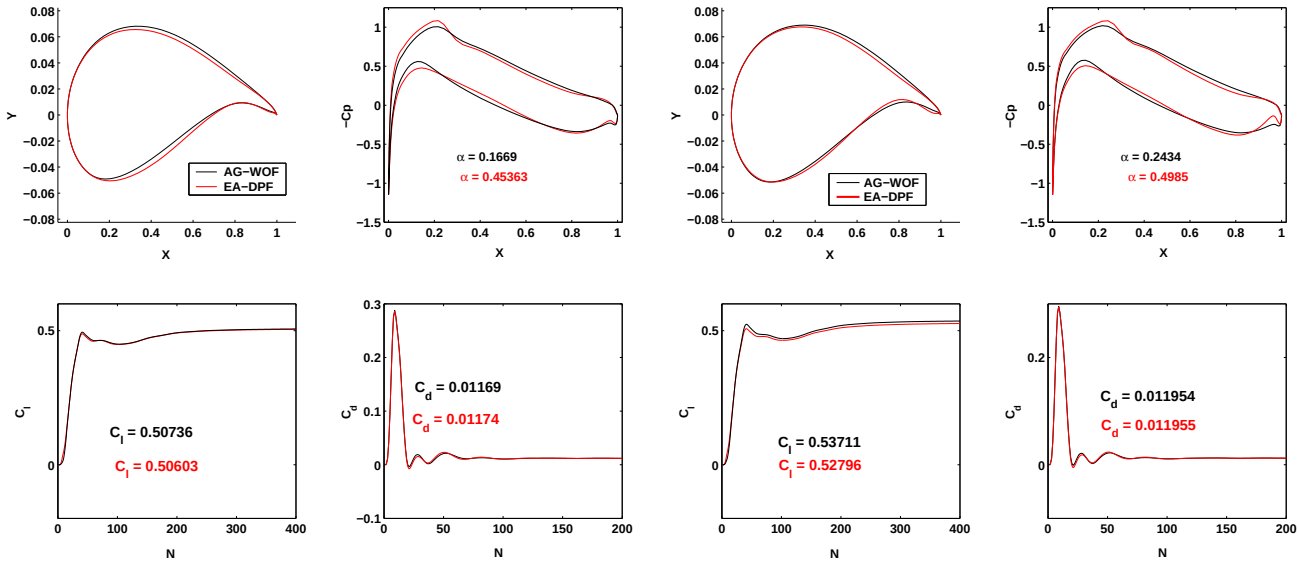


Fig. 6 Pareto Front Comparison: AG-WOF ,EA-WOF ,EA-DPF



a) Case $W_l = 0.01$

b) Case $W_l = 0.20$



c) Case $W_l = 0.50$

d) Case $W_l = 0.80$

Fig. 7 Comparison of AG-WOF and EA-DPF

slow to converge except in the last 10% for the optimization, see Figures 3 and 4. In the case of an **EA** optimization, the only measure of convergence is the objective function, and it's convergence is similar to the **AG** results. Therefore, we may also conclude that in the absence of a measure on the gradient, the best we can hope for in an **EA** optimization is to get close to the extrema. One could argue, though, that this would be sufficient since other influences, (such as, design space approximations, flow solver error, and gradient error in an **AG**), could produce larger design differences than lack of convergence of the optimization. This should be the subject of future studies, where such errors are investigated for the influence on the final design and design tolerances can be established.

References

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

[2]Nemec, M. and Zingg, D.W., Towards Efficient Aerodynamic Shape Optimization Based on the Navier-Stokes Equations, *AIAA 2001-2532*, June, 2001.

[3]Nemec, M.Optimal Shape Design of Aerodynamic Configurations: A Newton-Krylov Approach, University of Toronto, PhD Thesis, 2002.

[4]Pueyo, A., An Efficient Newton-Krylov Method for the Euler and Navier-Stokes Equations, PhD Thesis, University of Toronto, 1998.

[5]Bentley, P. J. and Wakefield, J. P., An Analysis of Multiobjective Optimization Within Genetic Algorithms, *Technical Report ENGPJB96*, University of Huddersfield, UK, 1996.

[6]Deb, K., Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Chichester, UK, 2001, ISBN 0-471-87339-X.

[7]Coello Coello, C. A. Evolutionary Multi-Objective Optimization: A Critical Review, *Evolutionary Optimization*, pp. 117–146, Kluwer Academic Publishers, New York, February 2002, ISBN 0-7923-7654-4.

[8]Pulliam, T.H., Efficient Solution Methods for The Navier-Stokes Equations, *Lecture Notes for the von Kármán Institute For Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation In Turbomachinery Bladings*, von Kármán Institute, Rhode-St-Genese, Belgium, 1985

[9]Goldberg, D. E., Genetic Algorithms in Search, Optimization and Machine Learning, *Addison-Wesley Publication Company*, Reading Mass., 1989.