487
13p

GENETIC ALGORITHM APPROACHES FOR ACTUATOR PLACEMENT

Final Summary of Research
conducted under NASA Grant NAG 1-2119

Submitted by:
William A. Crossley, Assistant Professor
Andrea M. Cook, Graduate Research Assistant
School of Aeronautics and Astronautics, Purdue University
1282 Grissom Hall, West Lafayette, IN 47907-1282

ABSTRACT

This research investigated genetic algorithm approaches for smart actuator placement to provide aircraft maneuverability without requiring hinged flaps or other control surfaces. The effort supported goals of the Multidisciplinary Design Optimization focus efforts in NASA's Aircraft Morphing program. This work helped to properly identify various aspects of the genetic algorithm operators and parameters that allow for placement of discrete control actuators / effectors. An improved problem definition, including better definition of the objective function and constraints, resulted from this research effort. The work conducted for this research used a geometrically simple wing model; however, an increasing number of potential actuator placement locations were incorporated to illustrate the ability of the GA to determine promising actuator placement arrangements. This effort's major result is a useful genetic algorithm-based approach to assist in the discrete actuator / effector placement problem.

INTRODUCTION

The Aircraft Morphing program is a NASA-sponsored effort to develop smart devices for aircraft applications making use of active component technologies.[1] One of the technology areas for this program is Multidisciplinary Design Optimization (MDO), and a focus effort in MDO is to investigate and subsequently develop methods for optimal placement of sensors and actuators for aeroelastic control, flow control and acoustic control. A thrust of this effort is to evaluate novel control effectors for aircraft flight control as an alternative to conventional aircraft control surfaces.[2] These effectors are smart devices that produce a quasi-static shape change in an aircraft's wing to produce desired control moments.[3] Proposed concepts for this type of device include various smart materials and adaptive structures such as flexible/inflatable skins, piezoelectric actuators in composite laminates, shape memory alloys, and synthetic zero-mass jets driven by vibrating membranes. Proper placement of these devices is critical to their success.

Smart devices that can produce a quasi-static shape change in an aircraft wing may be able to provide three-axis flight control; this could eliminate the need for conventional control surfaces. Surfaces like ailerons, flaps, etc., have gaps between the wing and the surfaces that contribute to leakage and protuberance drag and can be a source of aerodynamic noise; these can also impact the radar cross-section of military aircraft. This report describes research of Genetic Algorithm (GA) approaches to place discrete actuators on a wing to provide three-axis flight control to a "seamless" aircraft.

The GA, a global optimization technique well suited to discrete optimization, is a promising approach for actuator placement. The GA is a computational representation of natural selection and reproduction observed in biological populations.[4] The mimicry of nature in a GA includes representing points in a design space as if they were individual organisms. Design variables are generally mapped into binary strings that provide the genes of a given design. These strings are then concatenated to form a chromosome representing the traits of an individual design point. The GA works with these binary chromosomes, which allows for discrete optimization. In this application, the binary "1" will represent an "on" actuator, while the "0" will represent an "off" actuator.

When attempting to incorporate actuator devices on an aircraft, there are several goals. One is to reduce the number of actuators required, and a second is to provide uncoupled control moments in all three axes (pitch, roll and yaw). Research for this effort investigated appropriate genetic algorithm approaches to provide fast runtimes and

high quality placement solutions. Variations of the crossover operator and the use of symmetry were addressed as part of the investigation.

RESEARCH RESULTS

AERODYNAMIC ANALYSIS AND MODELING

In order to address the actuator placement problem, 16 discrete actuator locations were defined on a geometrically simple wing. Figure 1 presents these discrete actuator locations on a wing planform that has been "unwrapped" about the leading edge. The actuator locations were chosen to mimic the effect of flaps along the trailing edge and leading edge of the wing.

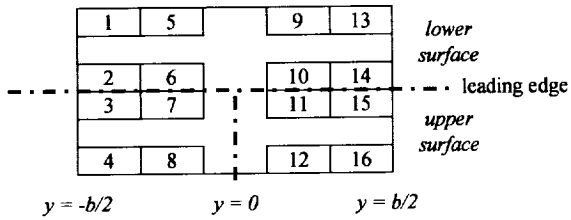

Figure 1. Discrete actuator locations on "unwrapped" wing model.

This wing model is an unswept, untapered wing with a NACA 0015 airfoil section. This wing has an aspect ratio of 8, a wing span of 96 ft, a chord of 12 ft, and a total area of 1,152 ft^2. The intent is to provide a wing dimensionally similar to a "real" airplane wing. These dimensions are comparable to the wing of a commercial transport aircraft. The 3-D panel code, PMARC,[5] predicted aerodynamic force and moment coefficients for the various actuator placements.

To begin the investigations, a panel model of the wing was constructed using the geometry described above. A common approach for paneling lifting surfaces uses a cosine spacing scheme to have a higher density of panels near the leading and trailing edges. However, to evaluate the effect of rectangular actuator patches, a uniform distribution of panels was desired to represent equally sized actuator patches. The results of a convergence study provided the total number of evenly spaced chordwise panels for the model. For this work, the wing model is split into 2400 panels, 20 panels in the spanwise direction by 60 panels in the chordwise direction. In addition, a rigid wake is attached to the trailing edge of the wing in the PMARC representation. This wake contains 240 panels with cosine spacing. The smallest panels are located closest to the wing to ensure the most accurate calculations as possible.

ACTUATOR MODELING

Each actuator location encompasses a group of panels. In the 16-actuator wing model, the each actuator location illustrated in Figure 1 has dimensions of 4 panels in the spanwise direction by 20 panels deep (chordwise). At the beginning of this research, it was not clear how the actuator should be modeled. Initially, imposing a positive normal velocity (blowing) on each panel in the actuator location provided a representation of an active actuator. While this appeared a reasonable way to represent the effect of an actuator, this model was adding mass to the flow, which is not appropriate for most envisioned actuator devices.

To avoid adding mass to the flow, combinations using equal numbers of panels with positive normal velocity (blowing, modeled using a potential source) and with negative normal velocity (suction, modeled using a potential sink) in the area defined by the given actuator were investigated. The first modeling approach used panels with alternating positive and negative surface normal velocities, with velocity magnitudes of ± 1.0. Figure 2 depicts this type of actuator model.

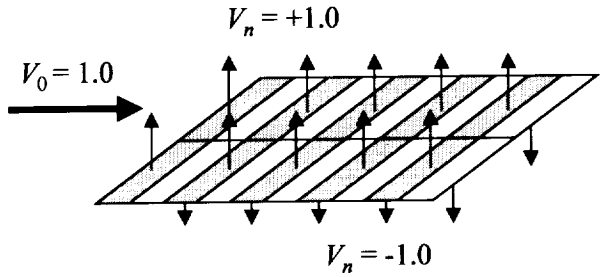

Figure 2. PMARC model of "on" actuator with alternating positive and negative normal velocities.

A wing with one "on" actuator on the trailing edge lower surface of the wing was evaluated in PMARC, which computed pressure coefficient distributions over the wing. A plot of the pressure coefficients on the surface of the wing at a chordwise location passing through the "on" actuator shows that this modeling strategy is not the best choice. Figure 3 displays this pressure distribution.

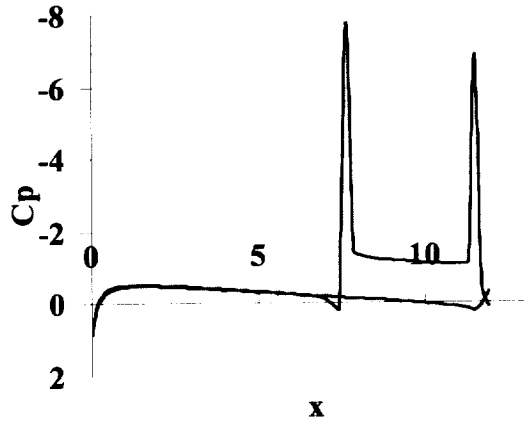

Figure 3. Pressure coefficient distribution on upper surface of airfoil with one active alternating source/sink actuator model.

Because the normal velocities are equal to the freestream velocity, the magnitude of the pressure coefficient near the active actuator is very large. Also, the alternating source / sink model causes some unusual spikes in the C_p distribution at the front and back of the active actuator.

To remedy the obvious problems in the previous actuator models, active actuators were modeled using surface blowing (again, using potential sources) on the upstream aerodynamic panels in the actuator area followed by an equal number of panels with suction (potential sinks). This arrangement corrected the shape of the pressure distribution over the wing with an "on" actuator. In addition, the magnitude of the normal velocities is decreased to ±0.26 to remove the large magnitude pressure regions of the previous models. By maintaining an equal number of source and sink panels, this representation does not add any mass to the flow. Figure 4 shows this active actuator model.

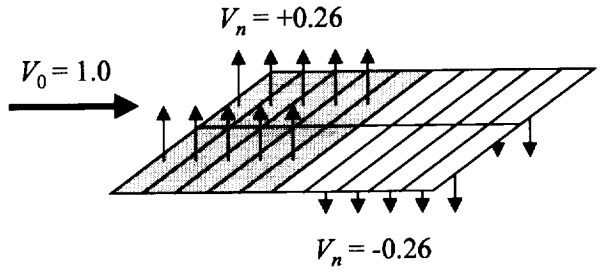

Figure 4. PMARC model of "on" actuator with separate groups of positive and negative normal velocities.

Comparing the effect of the active actuators to the effect of a conventional control surface and plotting the pressure coefficients allowed determination of the normal velocity magnitude. To provide the pressure distribution of the conventional control surface, the two-dimensional panel code, XFOIL,[6] computed C_p values for a NACA 0015 airfoil with a deflected plain flap. The magnitude of the surface normal velocities in the PMARC actuator model were varied until the peak Cp in the PMARC model closely matched the peak Cp in the XFOIL model. This study determined that an actuator with normal velocity magnitudes of ±0.26 causes the same peak in the pressure coefficient as a deployed flap. Figure 5 presents the C_p curve when an actuator is activated on the trailing edge of the PMARC wing model and the C_p curve predicted when a plain flap is deflected using the XFOIL model.

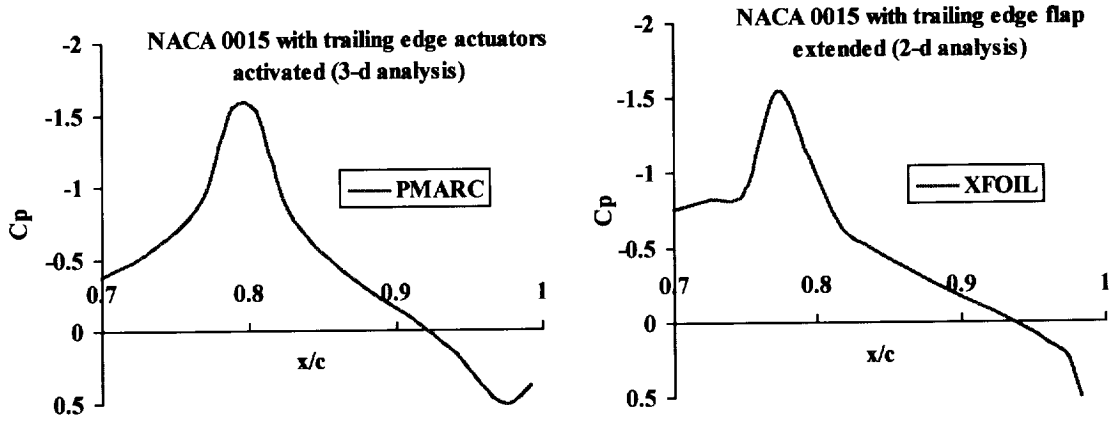

Figure 5. Pressure coefficients for active actuators (left) and conventional flap extended at 15 degrees (right).

By modeling the actuator with upstream blowing and downstream suction panels, the PMARC model becomes very generic so nearly any practical smart actuator system can be represented by matching the pressure coefficient distributions of the actual actuator and the normal velocity model in PMARC. Using the 2-D XFOIL results here is not completely realistic, but it demonstrates the flexibility of the actuator model. Experimental data, especially pressure distributions past an activated smart actuator device, would provide an inverse design approach to determine normal velocities for the PMARC models. Unfortunately, experimental data was not available for this research.

PMARC SOLVER MODIFICATIONS

Representing actuators with surface normal velocities provides an additional advantage. Using a vortex-panel aerodynamic analysis tool like PMARC requires a geometric description of the body to be modeled. This description allows the calculation of influence coefficients, which become the elements of a large matrix, **C**, in a system of linear equations, $\mathbf{C}\mu = \mathbf{b}$. If the surface geometry is to be changed, new coordinates need to be entered into the program, and, in turn, a new set of influence coefficients must be calculated and assembled into a new **C** matrix. Velocity and wake influence terms appear in the right-hand side vector, **b**, of the linear system of equations.

The vector of source doublet strength values, μ, provide values used to predict aerodynamic properties of the body. By representing the active actuator (which may provide an actual geometry change) through the blowing/suction model, the large influence coefficient can remain unchanged, and only a new right-hand side vector needs to be computed. This saves a great deal of computational effort. Unfortunately for this application, PMARC, like most panel codes, originated 15 or more years ago when computer memory and disk storage were limited in comparison to today's computational resources.

Immediately prior to this research effort described in this report, PMARC was modified for use as a subroutine that can be called numerous times in succession. While this modification shortened the run time of the genetic algorithm significantly by requiring only one assembly of $\mathbf{C}$, that can be reused throughout the run, using PMARC still requires solution of $\mathbf{C}\mu = \mathbf{b}$ for each new actuator configuration.

The original matrix solver for PMARC solves for the doublet strength vector using an iterative technique that partitions the system of equations into smaller sub-problems that require significantly less memory.[7] Using this solver, each call to PMARC requires about one minute and 30 seconds of runtime to compute the μ vector for the wing model. Because of the computational resources available, and because the $\mathbf{C}$ matrix remains unchanged regardless of the actuator configuration being evaluated, this iterative solution was not necessary for every call to PMARC. In an effort speed up the run time for the GA, which can require thousands of PMARC analyses, several different numerical methods were investigated to improve the PMARC solver.

First, a sparse matrix solver from the IMSL library[8] was implemented to replace the original iterative solver. While the geometry-based influence coefficient matrix is very large and contains many zero (or nearly zero) elements, it was not "sparse enough" for the IMSL routine to solve. Then, the IMSL version of LU decomposition was attempted. This routine did not work either; given the run-time error messages provided, it appears that the influence coefficient matrix was too large for the IMSL routine to handle. Additional investigations of LU solvers may provide an alternative approach.

Finally, an approach to store the inverse of the influence coefficient matrix, $\mathbf{C}^{-1}$, was developed using routines from both IMSL and Numerical Recipes.[9] This inverse matrix is assembled and stored during the initial call to PMARC and is then used to solve the forward multiplication matrix equation, $\mu = \mathbf{C}^{-1}\mathbf{b}$, on every subsequent call. Computing an inverse matrix is a comparatively expensive endeavor in computational terms, so the initial call to PMARC will take much longer. However, the intent is to recover this initial investment of computational effort in the much faster forward multiplication.

The Numerical Recipes version of the new PMARC solver uses the "LUDCMP" and "LUBKSB" routines. The first subroutine performs L-U decomposition on the influence coefficient matrix. This separates the matrix into upper and lower triangular matrices. The next subroutine uses back substitution to convert these two triangular matrices into the inverse of the original $\mathbf{C}$ matrix. The inverted matrix ($\mathbf{C}^{-1}$) is then stored in a scratch file for use in subsequent calls to PMARC, which then perform the, $\mu = \mathbf{C}^{-1}\mathbf{b}$ calculation. Using the Numerical Recipes routines for matrix inversion then subsequent forward multiplication requires approximately 144 calls to PMARC before the runtime is equal to the runtime required using PMARC with its original solver. Because the GA calls PMARC 50 times or more during each generation, the reduction in overall run time of the genetic algorithm is obvious after just a few generations.

In the second version of the inverse-matrix solver approach, the Fortran 77 IMSL subroutine "LINRG" directly inverts the influence coefficient matrix. This matrix is then stored in a scratch file and is used in the forward multiplication of subsequent PMARC calls. The IMSL solver requires only about 28 calls to PMARC before its runtime is equal to the runtime of the original solver. Thus, the reduction in runtime using the IMSL routines to calculate the $\mathbf{C}^{-1}$ matrix is obvious within the first generation of the GA.

Figure 6 shows a plot of the user time versus the number of subsequent PMARC calls for the three solver approaches (original, Numerical Recipes inversion, and IMSL inversion). To create this plot, PMARC ran for one, two, five and ten consecutive evaluations and the user was measured. A linear trend was fit to this data; these trends project that for large numbers of PMARC calls, the expense of the matrix inversion is rapidly recovered. Runs with the new inverse-based solver scheme demonstrate this behavior.

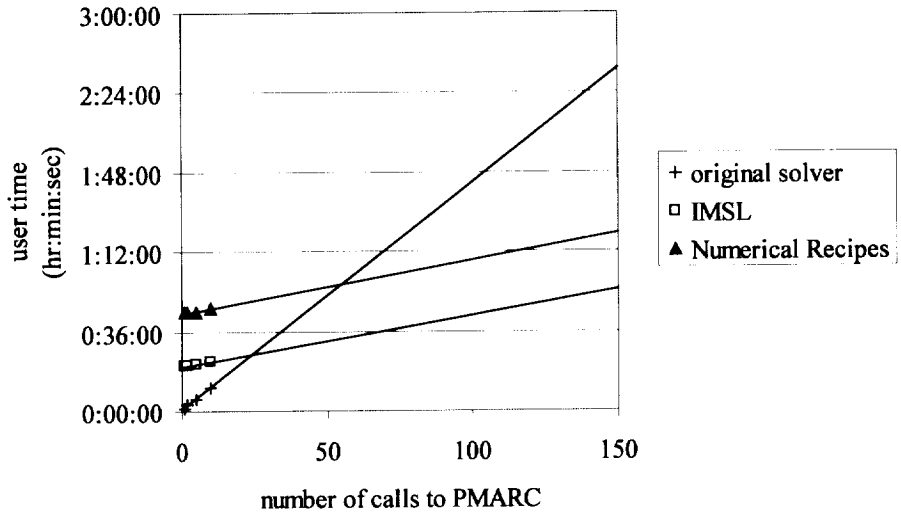

Figure 6. Number of calls to PMARC versus user time for original solver and new solver.

PROBLEM STATEMENT

To form chromosomes for the genetic algorithm, a 16-bit string of "0"s and "1"s were used to represent whether the corresponding actuator location was "on" or "off". As an example, the chromosome, "0110110110010010", would represent an actuator configuration in which the "1" locations were turned on. This example is presented in Figure 7, where the shaded regions indicate the "on" actuator.

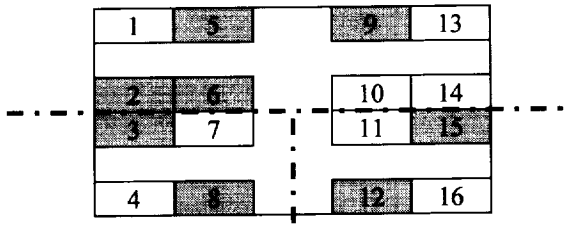

Figure 7. Example wing with "on" actuators shaded.

To begin, an uncoupled pitch control problem was developed. In this uncoupled pitch case, the objective was to minimize the objective function, ϕ,

$$\text{minimize } \phi = \sum_{i=1}^{\ell} a_i \tag{1}$$

where a_i is the i^{th} bit in the chromosome string corresponding to the i^{th} actuator location and ℓ is the total chromosome length (here, this is 16). Because each "1" bit represents an "on" actuator, this minimizes the number of "on" actuators.

Constraints are imposed on the pitching, rolling, and yawing coefficients in order to obtain uncoupled moments about one axis. For the problem describing a pitching maneuver, the rolling and yawing moments are constrained to a small values ($|c_n| \le 0.001$ and $|c_l| \le 0.001$, respectively). A limit value for the minimum pitching moment was determined by conducting a PMARC analysis of a wing model with actuator locations 4, 8, 12, and 16 turned "on" to represent a condition like deflecting trailing edge flaps upward to provide an uncoupled pitching moment. The value of c_m from this analysis is reduced by five percent to obtain the limiting value used in the constraint. For the pitching moment case, with 16 possible actuator locations, these constraints are

$$g_1 = 1 + \frac{c_m}{0.057} \tag{2}$$

$$g_2 = \frac{|c_n|}{0.001} - 1 \tag{3}$$

$$g_3 = \frac{|c_l|}{0.001} - 1 \tag{4}$$

where c_m is the pitching moment coefficient, c_n is the yawing moment coefficient, and c_l is the rolling moment coefficient. In this scaled form, the constraint function values, g_i, are positive when the constraint is violated. These constraints will ensure that the feasible designs are capable of performing a pitching maneuver, without any rolling or yawing.

Similar PMARC analyses were conducted to generate the constraint limits imposed to provide uncoupled rolling and yawing maneuvers. For the rolling moment, actuators 1, 5, 12, and 16 were turned "on" to represent aileron deflection in a rolling maneuver. For yaw, actuators 13 and 16 were turned "on" to represent a split-flap yawing maneuver. Illustrates the models used to compute the limiting moment coefficient values for the constraints.

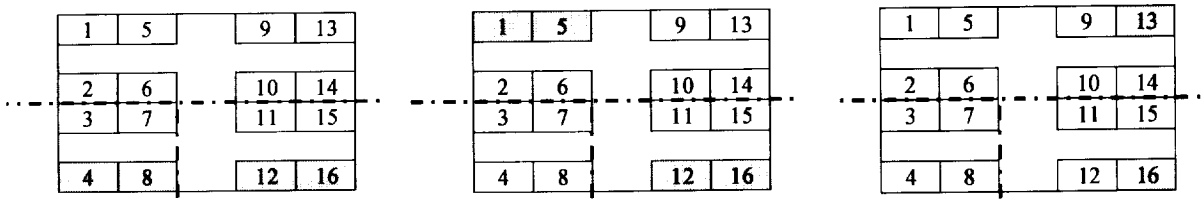

Figure 8. Wing models used to calculate constraints for pitch (left), roll (center), and yaw (right) for single moment problem.

Because the genetic algorithm uses only a single fitness function value to guide its search, the fitness function must reflect both the objective and the constraints. The approach used in this work was to apply a step-linear external penalty function. With this formulation, the fitness function, f, was expressed as:

$$f = \phi + \sum_{i=1}^{n_{con}} \left(r_0 \zeta_i + r_i \max[0, g_i] \right)$$
$$\zeta_i = \begin{cases} 0 & g_i \le 0 \\ 1 & g_i > 0 \end{cases} \tag{5}$$

The values of the penalty multipliers, r_i, were chosen so that the penalty terms have a similar order of magnitude to the objective function when the constraints are violated. The operator, ζ_i, has a value of 1 when the corresponding constraint is violated. In this manner, any constraint violation is assessed a penalty at least equal to the threshold value, r_0.

SYMMETRY STUDIES

Some natural symmetry exists in producing uncoupled aerodynamic moments that may be exploited to make this problem easier to address. Because the binary code lends itself very well to the "on / off" representation of actuators, it is natural to code a chromosome that represents all possible actuator locations. The 16-bit chromosomes used in the previous studies followed this approach. However, this can present some problems for the crossover operator if symmetry is not incorporated. For a design to be selected as a "parent" in the GA, it must have a good

fitness value. To provide an uncoupled pitch maneuver, the actuators need to be symmetrically placed about the $y = 0$ line on the wing model. Asymmetry about this line will lead to rolling and / or yawing moments. Therefore, parent designs (which are selected as good designs) are likely to have symmetric actuator placements. An example of how both uniform and single point crossover can create asymmetric (and infeasible) children from two feasible parents is presented in Figure 9. The actuator arrangements and corresponding chromosome representations are shown.

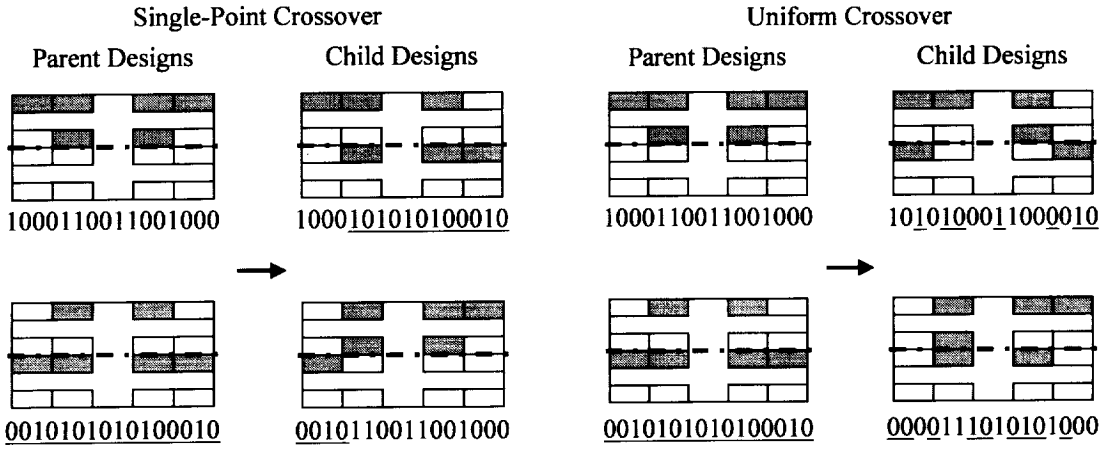

Figure 9. Asymmetric children resulting from two symmetric parents.

As mentioned previously, to provide an uncoupled pitching moment, the "on" actuators need to have left-right symmetry so that yawing or rolling moments are not produced. The rolling moment has similar properties, with the top left and right bottom of the wing requiring symmetric actuators. Uncoupled yaw can utilize top-bottom symmetry. The axes of symmetry for each of the maneuver conditions are shown in Figure 10. By employing these symmetry conditions, the length of the chromosome can be reduced from 16 bits to eight bits.

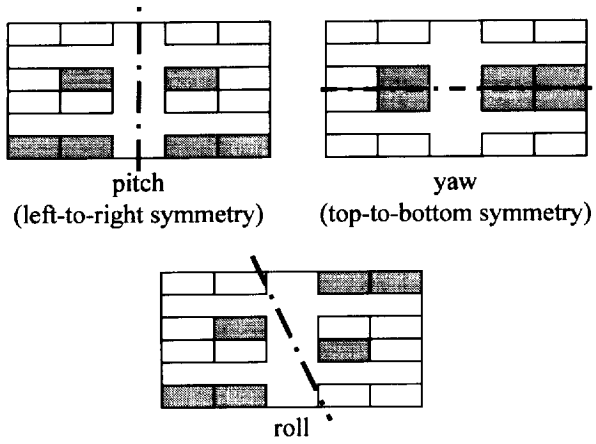

Figure 10. Axes of symmetry.

The issue of symmetry was addressed to determine its effect on the overall speed and efficiency of the GA. A general case was run using a 16-bit string length with a population size of 64 individuals and a mutation rate of 0.0083. Then, a symmetrical case using an eight-bit string length with a population size of 32 individuals and a mutation rate of 0.0176 was run. As before, the population size and probability of mutation were determined using published guidelines.[10] For both the symmetric and general cases, uniform crossover was used, and the pitching moment case provided the problem statement. The stopping criterion halted the GA when the minimum fitness was the same for five successive generations or when the maximum fitness equaled the minimum fitness. Figure 11

shows the convergence history for a general (nonsymmetrical) and a symmetrical case, both using the same random seed to start the runs and both using uniform crossover.

The nonsymmetrical case ran for 41 generations before meeting the stopping criterion. The best feasible design for this run contained six actuators, found at generation 9. The symmetrical case required 4 generations to meet the stopping criterion. The best feasible design contained 4 actuators and was located immediately in generation 0. By using symmetry in this problem, a large part of the infeasible design space is automatically eliminated. Using a nonsymmetrical chromosome representation, the GA wastes most of its computational effort evaluating nonsymmetrical, infeasible designs. Thus, the symmetric chromosome representation is more computationally efficient, and the GA is capable of finding good designs quickly. As shown in Figure 11, the maximum, average, and minimum fitness values were substantially lower throughout the duration of the symmetrical run.

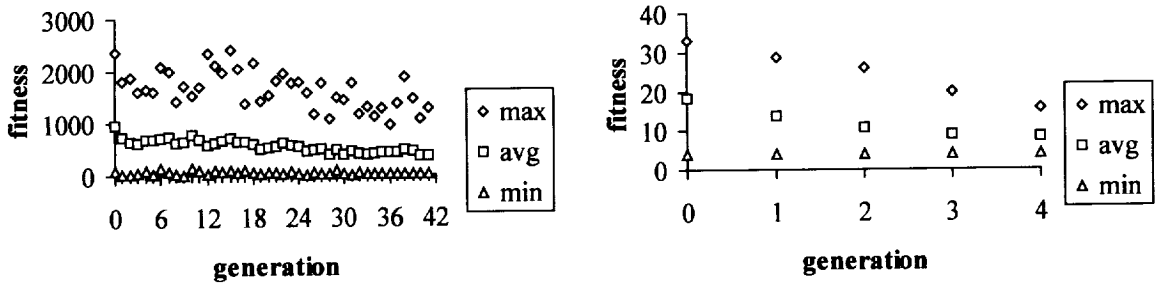

Figure 11. Fitness history for GA runs using general (top) and symmetric (bottom) chromosome representations.
(Note different scales for each plot.)

After running the GA with and without symmetry in the problem, it is obvious that by exploiting this characteristic of the problem, the computational time of the program can be drastically reduced. After five runs, the average runtime for a symmetrical run was approximately 9.3 hours real time using a Sun Ultra-450 computer. The nonsymmetrical case, however, took around 250 hours (10 days) real time to reach the convergence criteria.

The use of one crossover routine over another seems to have little effect when symmetry is also used in this problem. Five cases incorporating symmetry were run for the pitch case using each type of crossover routine (10 total cases). On average, a GA run with uniform crossover required about 10.5 CPU hours to reach the stopping criterion, while a GA run with single-point crossover required about 10 CPU hours. In all cases, the best feasible design contained 4 actuators. There is no significant difference in the convergence behavior or the fitness values of these routines. The crossover routine may have a more profound effect on the nonsymmetrical problem. Because single-point crossover is more capable of retaining patterns than uniform crossover, it may converge more quickly and find better designs.

Because of the obvious benefits of exploiting the symmetry in this problem, the GA was run to minimize the number of actuators in order to have uncoupled control in each of the three axes. Figure 12 shows the results from a typical pitch case. The problem statement for the pitch case was to minimize the total number of actuators subject to a minimum pitching moment of 0.0570 and absolute values of rolling and yawing moments less than 0.001. As in the previous cases, the constraint values were determined using a PMARC analysis with the appropriate actuators turned "on" to mimic a conventional control surface maneuver. The case shown in the figure required 7 generations to reach the stopping criterion and produced a scheme of four "on" actuators that produce a pitching moment of 0.0578. This solution is first encountered in the initial generation (generation 0), and is re-encountered by generation 3.

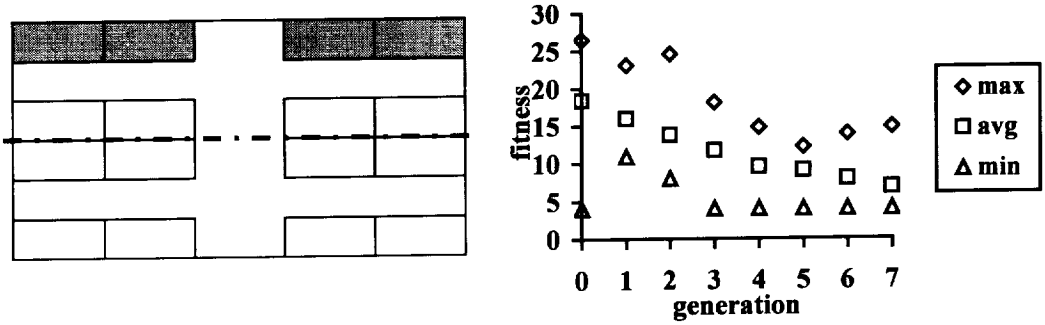

Figure 12. Actuator locations for pitching moment (top) and fitness history (bottom)

In addition to the "best" solution of four actuators, the GA finds several other feasible solutions, which can be used as alternate solutions should the "best" solution not meet all of the designer's manufacturing, safety, or other requirements. One of these six actuator schemes consists of actuators 1, 3, 5, 9, 13, and 15 and produces a pitching moment of 0.0652. The other alternate solution requires actuators 1, 5, 7, 9, 11, and 13 and produces a pitching moment of 0.0655. Because the GA's population-based search offers several design solutions from one run, it is very useful in complex, computationally expensive problems.

Next, the GA was used to minimize the number of actuators subject to a minimum rolling moment of 0.0650 and absolute values of pitching and yawing moments less than 0.001. The case shown in Figure 13 required 19 generations to reach the stopping criterion. As in the pitch case, the solution utilizes four actuators. The rolling moment produced by these four actuators is 0.0667 and is first encountered in generation 6. Alternate solutions found by this GA run include one four-actuator scheme and several six actuator schemes. The four actuator alternate scheme consists of actuators 1, 5, 12, and 16 to produce a slightly larger rolling moment of 0.0672. The six-actuator schemes all provide larger rolling moments and can be chosen at the designer's discretion.

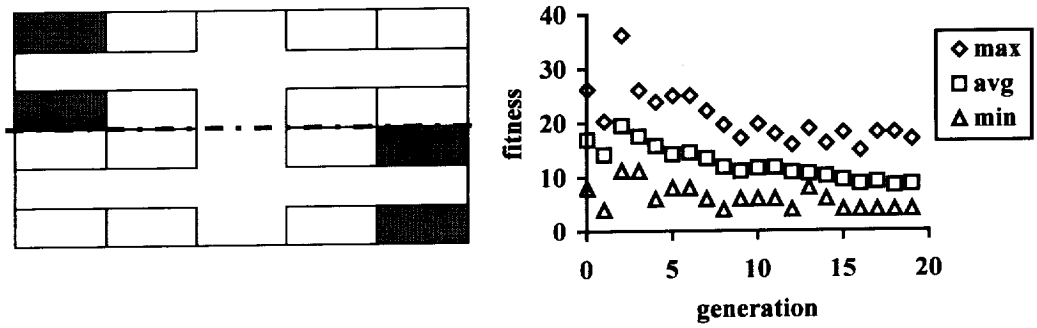

Figure 13. Actuator locations for rolling moment (top) and fitness history (bottom).

Finally, the number of actuators were minimized in order to achieve a minimum yawing moment of 0.0042 and absolute values of pitching and rolling moments less than 0.001. Figure 14 shows the solution and the convergence plot. This case required 6 generations to meet the stopping criterion. The solution uses only 2 actuators, positions 1 and 4, to produce a yawing moment of 0.0043. The first occurrence of this solution is found in generation 0. An alternate two-actuator solution uses positions 14 and 15, producing a yawing moment of 0.0083. There were also several four actuator schemes found, which produced larger yawing moments.

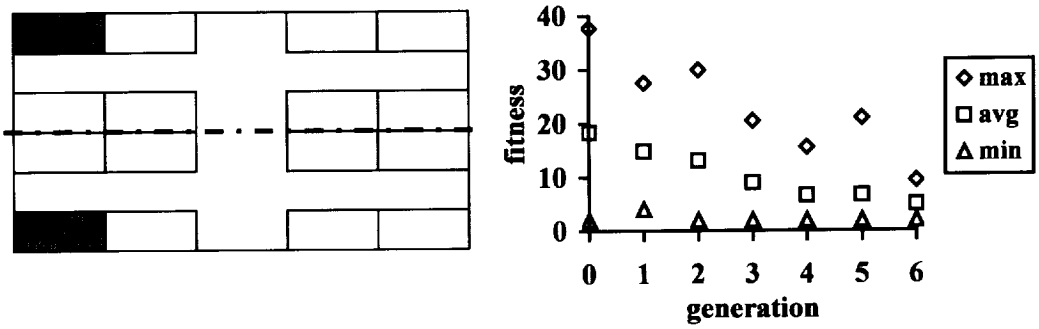

Figure 14. Actuator locations for yawing moment (top) and fitness history (bottom).

For all three uncoupled moment cases with the symmetric chromosome representation shown above, the resulting best actuator placements have the expected number of actuators and the number of generations to obtain these results is much lower than the previous studies. These results indicate that the GA approach for actuator placement is successful. However, using only eight bits to represent the available actuator locations, there exist only 2^8 (or 256) possible combinations of "on" and "off" actuators. This means that an exhaustive search of the design space may take comparatively fewer function evaluations than the GA search. In the examples shown above, the GA required 256 function evaluations to generate results for the pitch moment case, 640 for the rolling moment case, and 256 for the yawing moment case.

PROBLEM SIZE STUDIES

To determine the effect of increasing the size and complexity of the problem, the number of actuators was doubled, to 32. Each actuator in the original problem was divided in half chordwise. It was expected that each GA run would require approximately twice as many generations to reach convergence; however, the increased complexity may cause convergence problems in PMARC, which would drastically slow down the process. The results for a typical pitch case are shown in Figure 15. The results show that the more complex problem did take about twice as many generations to reach the stopping criteria. Also, the fitness value is exactly twice as large as the simpler problem, requiring 8 actuators instead of 4 to attain a pitching moment of 0.0380. This problem required about 3.5 hours of CPU time, about twice as much as the 16-actuator problem.

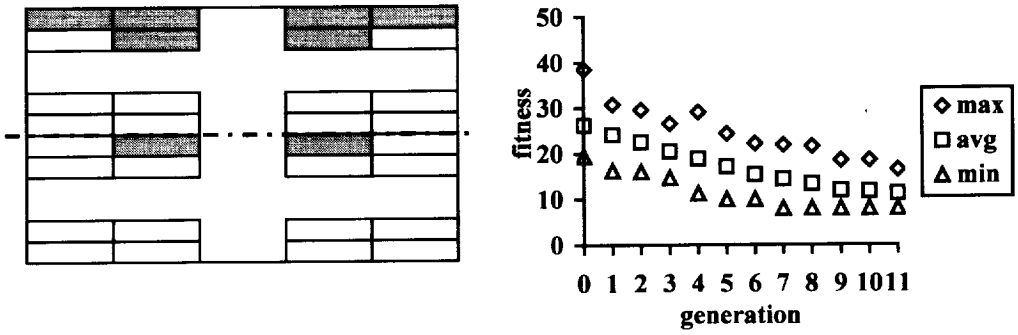

Figure 15. Actuator locations for pitching moment with 32 possible actuator locations (top) and fitness history (bottom).

Making use of symmetry, this larger problem size used a 16-bit chromosome to represent the various actuator locations, which corresponds to 2^{16} (or 65,536) possible combinations of "on" and "off" actuators. Now, the GA approach requires far fewer evaluations than an enumerative approach to find optimal actuator arrangements. The example described above required only 768 function evaluations before reaching the stopping criterion.

CONCLUSIONS

The research presented here has shown that the genetic algorithm approach is successful at minimizing the number of discrete actuators needed to provide an uncoupled moment in each of the three body axes. The GA approach is very straightforward and capable of accommodating any type of actuator with minimal program modifications. The PMARC actuator model reduces the computational effort and allows the actuator to be modeled as a function rather than an actual device. This flexibility means that the GA approach should be successful in this type of problem, regardless of the type of device being modeled.

By replacing the original linear system solver in PMARC with an approach that inverts the influence coefficient matrix on the first call to PMARC, then performs forward multiplication for subsequent calls to PMARC provides a great reduction in time needed to complete one run of the genetic algorithm.

It has also been shown that the use of symmetry in the problem provides the greatest benefits in reducing computational time. By exploiting the natural symmetry inherent in the actuator placement problem, the total runtimes can be reduced from almost 2 weeks of real time to 8 hours real time for the simple wing case. This drastic reduction in runtime will allow a more complex problem to be solved in a realistic amount of time.

The choice of crossover routine seems to have little effect on the efficiency of the GA. From the results presented here, it has been shown that uniform and single point crossover produce nearly identical results in approximately the same amount of time. For this reason, the importance of the crossover routine appears to be minimal when symmetry is used in the simple wing problem.

The results presented also show that a more complex problem is computationally feasible and that the increased complexity does not drastically slow down the total runtime. An enumerative search of the 32-actuator problem would require the evaluation of 65,536 designs, but the GA required only 768 evaluations to find a "good" design. In addition to this, the GA is able to find multiple feasible designs in each run, providing flexibility to the designer and making the GA a more efficient design tool. This illustrates the overwhelming benefits of the GA approach for determining actuator placement in more complex problems.

Future work on this problem will include applying a genetic algorithm to optimize for the minimum number of unique actuators that can achieve uncoupled moments in all three axes. This work will continue the effort to increase the number of actuator locations on the wing and eventually, a full-size aircraft will be studied. This type of large problem may not have been possible without the work discussed in this report to reduce the computational time and increase the overall efficiency of the GA for this problem.

REFERENCES

[1] Wlezien, R. W., Horner, G. C., McGowan, A. R., Padula, S. L., Scott, M. A., Silcox, R. H., and Simpson, J. O., "The Aircraft Morphing Program," AIAA 98-1927, 39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Apr. 1998.

[2] Scott, M. A., Montgomery, R. C., and Weston, R. P., "Subsonic Maneuvering Effectiveness of High Performance Aircraft Which Employ Quasi-Static Shape Change Devices," 36th SPIE 5th Annual International Symposium on Smart Structures and Materials, San Diego, CA, Mar. 1-6, 1998.

[3] Lachowicz, J. T., Yao, C. S., and Wlezien, R. W., "Scaling of an Oscillatory Flow Control Actuator," AIAA 98-0330, 36th Aerospace Sciences Meeting & Exhibit, Reno, NV, Jan.12-15, 1998.

[4] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[5] Ashby, D. L., Dudley, M. R., Iguchi, S. K., Browne, L., and Katz, J., "Potential Flow Theory and Operation Guide for the Panel Code PMARC_12," NASA Ames Research Center, Dec. 1992.

[6] Drela, M., "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils," Conference on Low Reynolds Number Airfoil Aerodynamics, University of Notre Dame, June 1989.

[7] Davidson, E. R., "The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-Symmetric Matrices," *Journal of Computational Physics*, 17, 1975, pp. 87-94.

[8] *IMSL Math/Library Online User's Guide Volume 1*, Visual Numerics Inc., Houston, TX, 1997 [http://www.vni.com/products/imsl/]

[9] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P., *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, second edition, Cambridge University Press, Cambridge, UK, 1992.

[10] Williams, E. A., and Crossley, W. A., "Empirically-Derived Population Size and Mutation Rate Guidelines for a Genetic Algorithm with Uniform Crossover," *Soft Computing in Engineering Design and Manufacturing*, P. K. Chawdhry, R. Roy and R. K. Pant (editors), Springer-Verlag, 1998, pp. 163-172.

INVENTIONS

No inventions were developed as a result of this research.

BIBLIOGRAPHY

Publications resulting from this research are listed below.

Cook, A. M., and Crossley, W. A., "Genetic Algorithm Approaches to Smart Actuator Placement for Aircraft Flight Control," AIAA Paper 2000-1582, 41st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Atlanta, GA, Apr. 3-6, 2000.