A Genetic Approach To Modelling Flight Dynamic Characteristics

P J Gage [*] R A Stuckey [†] J S Drobik [†]

Abstract

There are several parameter identification techniques that are commonly used for the determination of the dynamic characteristics of flight vehicles. Standard genetic algorithms have been used previously for parameter identification, and successfully identified globally optimal parameter sets. In this paper, we examine the use of genetic programming for system identification, and explore the automatic discovery of equation structures rather than simply determining coefficient values in models of specified form. The trajectory of a ballistic weapon is successfully modelled, but work remains to generate equations that are more physically relevant, and to improve efficiency of search.

Abbreviations

AMRL	Aeronautical and Maritime Research Laboratory
AOD	Air Operations Division
BCU	Ballistic Computer Unit
DSTO	Defence Science and Technology Organisation
FCL	Flight Control Law
MMLE3	Maximum Likelihood Parameter Estimation Program, Number 3
NASA	National Aeronautics and Space Administration
OFP	Operational Flight Program
PC	Personal Computer
RAAF	Royal Australian Air Force

[*]Australian Defence Force Academy, Member AIAA
[†]AOD, AMRL, Member AIAA
0

1 Introduction

In order to identify the stability and control characteristics of aircraft from flight data, known inputs must be applied and the responses measured. From this data, a set of equations can be developed and various parameters estimated to model the relationship between input and output. For standard configurations operating in linear flight regimes, the form of these equations is well known, and generally only the coefficients of each term need be identified. However, for unconventional designs such as the X-29 [1], or extreme flight conditions, the appropriate form of the model must be established before the coefficients can be determined.

Several parameter identification methods are available for determining coefficient values in the standard equations, including regression, maximum likelihood, filter error, and neural networks. The features of these methods and their range of application are discussed in several survey papers [2] [3] [4]. Despite the widespread success of these methods, there is a continuing search for improved techniques. Anderson et al. [6] cite the excessive data requirements of regression analysis as a factor that motivates them to investigate the performance of standard genetic algorithms in parameter identification tasks. The search space for an error minimization task is typically multimodal, so local search algorithms such as the maximum likelihood method may not find the global minimum, whereas genetic optimization works well in such domains [5].

When the appropriate form of the flight dynamics equations is unknown, the standard tools, and even standard genetic algorithms, are less useful. Stepwise regression can be used to produce a parsimonious model, by adding or removing terms according to their correlation [7], but it works only with terms of pre-determined form. Neural networks generate black box models that are of novel form, but no physical significance can be attributed to the terms (or combinations of weights and links) generated in this way [2]. The genetic programming paradigm falls between these extremes, because it permits the user to choose a set of physically significant basis functions that are automatically recombined to produce novel equations which include nonlinear terms. This method has been implemented successfully in simple function identification applications [8], where physically meaningful equations have been automatically generated. In this paper, genetic programming is used to model the behaviour of a simple flight vehicle: a ballistic weapon.

In the next section, a general description of genetic optimization is provided, with discussion of the features of genetic programming that make it particularly well-suited to this task. This is followed by a brief review of applications of genetic optimization for system modelling. Section 4 describes the use of genetic programming for a simple identification task, and explores the importance of various parameter settings on optimization performance. The method is then used to generate equations describing a weapon trajectory, to establish a basis of comparison with standard genetic algorithms. The paper concludes with a discussion of genetic programming performance, and recommendations for further work.

2 Genetic Optimization

Genetic algorithms are global search methods which use operators modelled on biological reproductive mechanisms observed in the natural world [9]. A wide variety of exotic genetic operators have been devised, but a few fundamental features are common to most genetic algorithms: population, evaluation, selection, encoding, crossover and mutation.

2.1 Essential Features

The genetic search procedure is initialized by randomly generating a population of candidate designs. These candidates should provide a statistically meaningful sample of the global search space. The population size should therefore be sufficiently large to avoid significant error.

Any search for improvement requires a metric for assigning relative merit, or performance, to alternative designs. Genetic algorithms require the calculation of fitness for each member in a population. For flight vehicle system identification,it is common to minimize either equation or output error, and either value can be used as the basis of a fitness metric for genetic optimization. The optimal solution minimizes error, whereas fitness should be maximized, so the fitness function should be inversely related to error. Equation 1 assures fitness values in the range [0,1], with the ideal (zero error) case producing a fitness value of 1.

$$Fitness = \frac{1}{1 + Error} \tag{1}$$

Selection rewards high-fitness designs in a population of candidates, by assigning a relatively high probability that they will be chosen for reproduction. The selection scheme should be chosen to balance the competing desires to exploit promising features contained in the existing population and to explore the design space for new possibilities. If selection pressure is too great, diversity can be quickly lost, and the population will converge to a sub-optimal design. If selection pressure is too weak, the algorithm is reduced to random search.

When selection is the only genetic operator, the optimizer simply promotes “survival of the fittest”. If there is no mechanism for modifying existing designs, the best members of the initial population will dominate in later generations, but there will be no improvement beyond the best of the randomly generated initial sample. Additional operators are needed to introduce new features while retaining important features from existing designs. These operators do not act directly on the design itself, but on a “genetic” string, which is typically formed by concatenating values for a set of parameters that describe the search space. Algorithm efficiency depends directly on the recombination of low-order building blocks into higher-order assemblies, so encodings that promote the recognition of promising building blocks should be carefully chosen for each application [10][11]. For the parameter identification task, binary strings are appropriate.

Crossover operators generate new designs com-

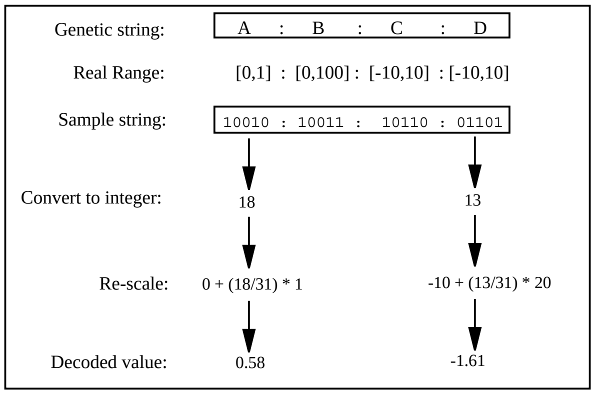


Figure 1: Parameter identification for generalized drag function. Decoding the genetic algorithm string.

posed from elements of two earlier designs, thus exploiting features already present in the population. The position of the crossover point along the string is chosen at random. Genetic algorithms generally include a pointwise mutation operation, which modifies individual bits of the string at random, and produces corresponding changes to a design variable. This operation can introduce features not present in either parent, so it helps to maintain diversity in the population.

2.2 Genetic Programming

Genetic programming shares all the essential characteristics of standard genetic algorithms, except that the encoding scheme includes functional information rather than simply concatenating parameter values. The user specifies a set of terminals (parameters and constants) and functions (arithmetical, mathematical, logical, domain-specific), and random combinations are chosen to compose programs (or complex functions). Typically, the encodings are of varying length, and the associated programs are of varying complexity. Smith [12] has demonstrated that promising building blocks are appropriately retained in variable-length strings, and Koza [8] cites the empirical evidence of successful applications in diverse disciplines as proof that genetic adaptation of variable-length strings is a valid search mechanism.

Consider the 4-parameter equation for weapon drag (Equation 2), developed by Anderson and McCurdy [15] (and discussed more fully later in this paper).

$$C_D = A + \frac{(1 - e^{-M^B})C}{M^D} \tag{2}$$

For a genetic algorithm, only the four parameters are included in the genetic string, as shown in Figure 1. In genetic programming, a set of basis functions and a set of terminals must be prescribed. For this example, a suitable basis set might include:

Basis Functions $\{\times,\div,+,\text{-},\wedge,\exp^{-},\ln\}$

Terminals {Mach Number (M), Real constants (RC)}

The genetic string encodes a tree constructed by combining functions and terminals, as indicated in Figure 2.The genetic string lists functions, followed by their arguments. Arguments may be terminals, or other functions, so that complex equations are constructed as chains of basis functions. Crossover recombines sub-chains from different parents, and mutation is used to replace one subchain with a randomly generated new chain.

Figures 3 and 4 indicate that the key distinction between genetic algorithms and genetic programming lies with the encoding of the problem into a string, which is manipulated by operators that are common to both methods.

3 Applications in System Modelling

Several researchers have used genetic algorithms for system modelling. Hensley [13] refined the aerodynamic model for a flight simulator, including basic and aeroelastic effects, and demonstrated that the genetic algorithm was insensitive to the quality of the initial guesses for parameter values. Tan et al. [14] have used a genetic algorithm, with an embedded simulated annealing algorithm in place of a random mutation operator, to optimize parameter values in nonlinear system models. Anderson and McCurdy [15] identified optimal parameter values for a generalized drag function (which had been developed by trial and error) to match time-space-position-information (TSPI)

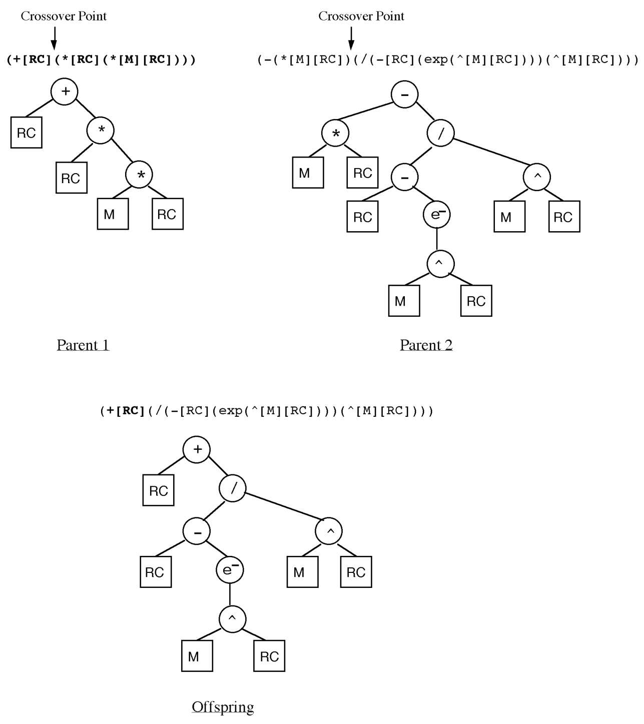


Figure 2: System identification for generalized drag function. Decoding the genetic programming string, and example of crossover.

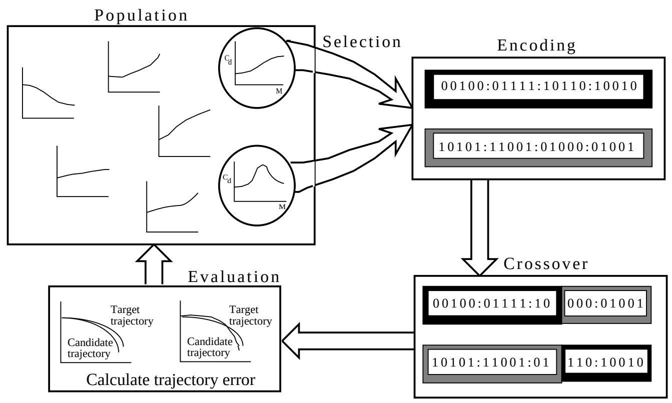


Figure 3: Parameter identification for a generalized drag function using a standard genetic algorithm

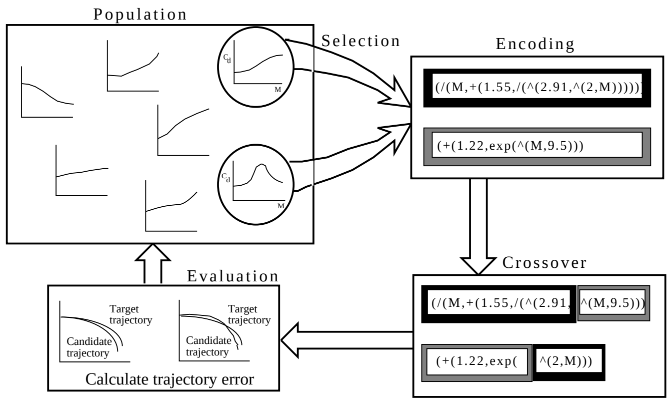


Figure 4: System identification for a generalized drag function using genetic programming

from ballistic trajectories of weapons. Their approach, which is representative of such efforts, is later presented in detail, and compared with the new genetic programming approach.

Ryan [16] has searched for combinations of control inputs that cause departure in high-performance aircraft. The genetic algorithm effectively located departures caused by inertial coupling and aerodynamic asymmetries, even for aircraft operating at high angles of attack, where the dynamics are particularly complex. Ryan notes that the user is not required to guess at possible initial flight conditions that may be prone to departure, and that this technique locates departure conditions overlooked by other prediction methods. This application uses the genetic algorithm for flight envelop validation, but it could readily be modified for model validation, where the algorithm would search for points of maximum discrepancy between model and system.

Genetic programming has been applied by Koza [8] to the derivation of Kepler's law of planetary motion ($\frac{D^3}{T^2} = c$). The algorithm produces functions expressing the period as a function of orbit diameter. The generated functions are evolved to minimize the discrepancy between the period predicted by the candidate equations and the true (measured) period, for each of nine planets. Menon et al. [17] have used genetic programming for the generation of simple flight control laws, and suggest that it will be a useful tool provided that the search effort required for more complex equations is not prohibitive.

4 Parameter Settings for Genetic Programming

Several genetic programming software packages are available: *lil-gp* [18] is used in this study. This package is written in ANSI C, so execution speeds are much higher than Lisp implementations of the genetic programming paradigm. Source code is freely available, and documentation is comprehensive, so it is possible to modify the algorithm quite readily.

It is well recognized that values of several input parameters strongly influence the behaviour of genetic optimizers. Here, we use a simple symbolic regression example to tune the parameter settings to be used in the system identification task described in the next section. A quartic polynomial is fitted to Anderson and McCurdy's equation for drag coefficient (Equation 2):

$$\begin{aligned} y &= -5.0318x^4 + 16.6426x^3 - 19.085x^2 \\ &\quad +9.1170x - 1.4240 \end{aligned} \tag{3}$$

The user must select the set of basis functions available to the genetic programming algorithm, and the set of terminals to be used as arguments in those functions. The following sets were used:

Basis Functions {×,÷,+,-,sin,cos,exp,ln}

Terminals

$$\left\{ \begin{array}{c} \text{Mach Number} \\ \text{Real Constants}[-20, 20] \end{array} \right\}$$

Koza derived Kepler's Law consistently within 50 generations, using a population of 500 candidate functions [8]. These should be considered as lower limits for the investigations included in this paper, which require the development of more complex functions.

The length of genetic strings in the initial population must be specified in genetic programming. Experience with variable-length strings in parameter optimization [19] suggested that maximum length strings are unlikely to be helpful in the early stages, but strings should be long enough for beneficial structures to be recognized and retained for future recombination. The initial population is seeded with random structures with tree depths betwwen 2 and 8, while a maximum depth of 17 is permitted in later generations. There is no restriction on the number of nodes, and no requirement for different branches of each tree to have consistent depth.

fitness	tournament		
-proportionate	size=2	size=5	size=7
0.7024	0.7299	0.8291	0.8757

Table 1: Influence of selection method on optimizer performance.

Fitness-proportionate selection and tournament selection (with a range of tournament sizes) were tested. The fitness values quoted in the following table are averages of the best adjusted fitness for each of 10 independent populations, because the genetic method is probabilistic and conclusions drawn from individual tests are unreliable. Larger tournament sizes clearly improved optimizer performance, so tournamnet selection, with a size of seven, was chosen for subsequent tests. Note that this contrasts with standard genetic algorithms, where smaller tournaments are generally effective. This may be due to a larger proportion of candidate solutions being non-viable in genetic programming, so the less promising candidates should be culled more aggressively.

Crossover rates between 0.7 and 0.9 are commonly quoted in the genetic optimization literature, and these values were quite appropriate for this application. There was not a significant influence on algorithm performance for different rates within this range. A mutation rate of 0.2 (meaning 20% of new individuals had a branch replaced by a randomly generated new branch) was also effective.

In standard parameter estimation, only the leaf nodes of a tree with specified structure are modified. This suggests that a new mutation operator, which concentrates on modifying constants in the tree without modifying tree structure, might be able to perform parameter identification within the genetic programming structure. This new operator, termed *perturbation* is a small step towards inner loop parameter optimization, which has been highly successful in earlier work on structural topology optimization [19]. Preliminary investigations indicated that there may be some benefit from such an operator, but the current implementation has only a small impact on algorithm behaviour.

It is desirable that solutions produced by genetic programming should have relatively small tree size, so that they can be readily interpreted and related to physical behaviour. Parsimony pressure may be used to adjust the fitness of a tree according to its size, so that large trees are penalized [20]. In this application, however, increased parsimony pressure reduced the complexity of the best expression found by the genetic program, but fitness also decreased. For example, a parsimony pressure of 1.0 reduced the solution tree size, from 140 nodes with a depth of 17, to 20 nodes with a depth of 7, but the average adjusted fitness also decreased from 0.9283 to 0.8298. An adaptive penalty method may be used in the future, but for the remainder of this paper parsimony is not considered.

5 A Generalized Drag Function for Ballistic Weapons

We choose to investigate the drag characteristics of a ballistic weapon, because we can directly compare genetic programming with the genetic method used by Anderson and McCurdy [15]. They point out that learning methods capable of producing smooth, continuously differentiable aerodynamic models would be particularly useful for guidance and control applications. Success in this relatively simple application will justify further investigation of more complex flying vehicle dynamics.

Anderson and McCurdy note that a generalized drag function, relating drag coefficient to Mach number, should capture the following basic features:

- relatively invariant for low Mach numbers
- moderate increase at onset of compressibility effects
- steep rise as compressibility effects dominate
- mild decrease as flow becomes fully supersonic

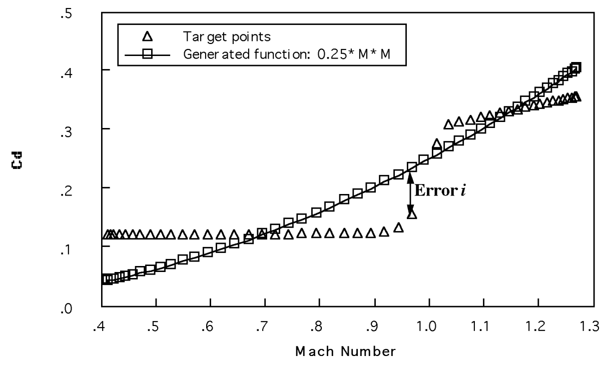


Figure 5: Objective is to minimize the error in Cd curve.

They made several attempts to develop a suitable function using high order polynomials and power series, but eventually produced their equation (Equation 2), repeated below for convenience) only after realizing that the exponential function would provide reasonable decay when coupled with a suitable power function. The insight required to combine basis functions in this manner was critical to their success in generating the simple equation. Our chief goal is to determine whether similar equations can be discovered automatically.

$$C_D = A + \frac{(1 - e^{-M^B})C}{M^D}$$

The four parameters A,B,C,D are each strongly associated with a single feature listed above, and variation of their values permits the general equation to be tailored to particular weapons.

5.1 Matching the Drag Curve

The equation developed by Anderson and McCurdy was used to generate simulated data for drag and Mach Number for the 2000-lb Mk-84 conventional bomb. We seek to generate an equation to model the relationship, using genetic programming.

5.1.1 Simulating a Genetic Algorithm

The first test contrives to limit the search power of genetic programming, so that it can only perform the parameter identification that is available with

Parameter	Known Value	Gen Prog	Gen Alg Ref [15]
A	0.12	.1196	.1181
B	45.0	50.9	50.1
C	0.18	.1852	.1978
D	-1.1	-1.091	-0.6227

Table 2: Genetic programming successfully performs parameter identification.

a standard genetic algorithm. This is achieved by defining a single basis function, with the exact form of the generating equation. This basis function takes four arguments, all of which are random constants, so the terminal set also has only one element. The range of permissible values for each argument of the basis function are chosen to match those used by Anderson and McCurdy in their study.

Basis Function

$$\{[0,1] + \frac{(1 - e^{-M^{[0,100]}})[-10,10]}{M^{[-10,10]}}\}$$

Terminals {Real constants}

For 10 randomly generated populations, each with 1000 individuals, the average fitness was 0.771 after 32 generations when evolution was terminated because a member was found to completely satisfy the user-specified criteria for an acceptable function match. The function compares favourably with that reported by Anderson and McCurdy for a similar problem (see Table 2), although it should be noted that an order of magnitude more function evaluations were used here. This demonstrates that the genetic programming tool is able to match the genetic algorithm in a problem of restricted scope. It remains to investigate genetic programming performance in the wider search for functional form of the drag equation.

5.1.2 A Richer Basis Set

The full power of genetic programming is realised only when more fundamental basis functions are made available for automatic combination by the program. Exponential and power functions are now included in the basis set. One power function permits Mach Number to be raised to any power in the range [0,50], while a second function restricts the range to lie between [0,10].

Basis Functions $\{\times, \div, +, -, \exp^{-}, \exp, M^{[0,50]}, M^{[0,10]}\}$

Terminals { M, RC }

When this basis set was used, reasonable solutions were often found, but it was possible for some populations to prematurely converge on excessively simple functions (a quadratic, for example). Basis functions are chosen at random from the available list, so power functions appeared infrequently, and useful combinations with exponential functions were difficult to locate. The situation improved when several copies of the power functions were included in the basis set, so that the proportion of power functions in the initial population increased. This is an example of the influence of basis function selection on optimizer performance. Global optimality is not guaranteed by genetic optimizers, and solution quality is strongly dependent on problem formulation.

For a basis set with 3 copies of the $M^{[0,50]}$ function, 10 independent populations with 1000 members each were generated. All populations produced functions that were reasonably flat at the ends, with a steep rise in the central portion. The two best solutions are shown in Figure 6. The dotted curve matches the target very closely for relatively high Mach Numbers, but has slightly high drag at low Mach Numbers, and a slight dip as the drag curve steepens. The second solution matches very well at the ends, but the slope of the drag rise is slightly low.

The form of the best solution is shown in Figure 7. This equation is clearly more complex than the target, but it has discovered the combination of exponential and power function, which Anderson and McCurdy considered essential. While the generated equations are not completely accurate, they do contain important combinations of basis functions. This example suggests that genetic programming could well be useful in domains where a suitable target function is not known *a priori*. In the next example, we test whether genetic programing can produce a reasonable drag curve by minimizing the error of a simulated trajectory.

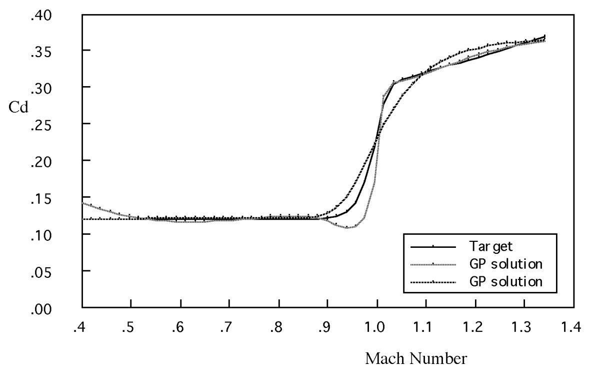


Figure 6: Best fits from 10 runs of the genetic program.

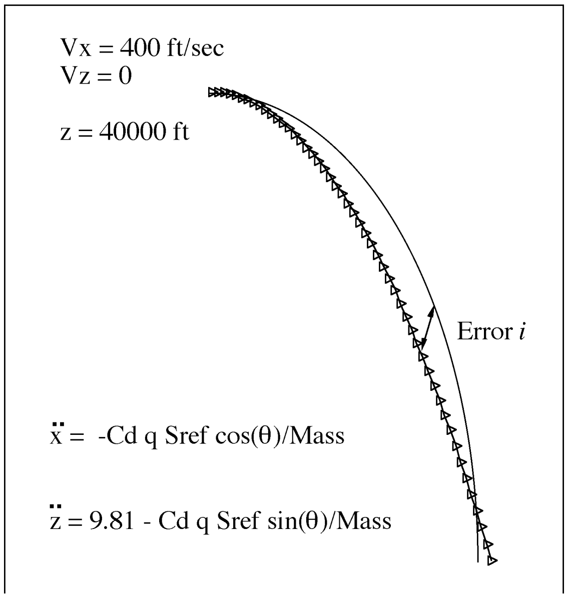
```
(exp(exp(/(exp(exp(exp M)))
          (+(+(exp(exp(exp M)))
                  (/(M^50 0.25766) 0.50563))
            (*(exp(/(M^50 0.25766) 0.61589))
              (-(-(M^10 (M^10 0.11333))
                (M^50 0.52493)) xp3))))))
```

Figure 7: Best function from genetic programming.

5.2 Matching the Trajectory

To test the performance of the genetic algorithm in this application, Anderson and McCurdy generated a simulated trajectory, for particular release condition ($V_x = 400$ft/sec, $V_y = V_z = 0.0$; $x = y = 0$, $z = 40000$ ft) for a particular weapon (2000lb Mk-84 conventional bomb). Trajectories are then generated using the drag functions associated with each set of parameters in the population, and candidates are ranked according to the error between their trajectory and the baseline. The optimizer identified parameters that produce trajectories with a root mean square error (normalized by number of evaluation points) of less than 2 feet in a 55 second simulation. Here, the same example (release conditions and weapon type) is used to produce 50 target points (at 1 second intervals), and the genetic program is to minimize the total RMS position error between points on a simulated trajectory and the targets (see Figure 8).

The best drag curve produced in this investigation is presented in Figure 9. It is very accurate for



Vx = 400 ft/sec
Vz = 0
z = 40000 ft
Error *i*
$\ddot{x}$ = -Cd q Sref cos(θ)/Mass
$\ddot{z}$ = 9.81 - Cd q Sref sin(θ)/Mass

Figure 8: Objetive is to minimize total RMS position error.

the lower Mach Numbers, but loses some quality in the supersonic region. This is perhaps due to the fact that supersonic speeds are attained only late in the flight, so that errors in this part of the drag curve affect only a few points in the trajectory. It is possible that a mosified objective function, which penalized errors later in the trajectory more heavily, might improve performance. Performing the optimization for several trajectories, including some that are dominated by supersonic speeds, is also likely to improve solution quality.

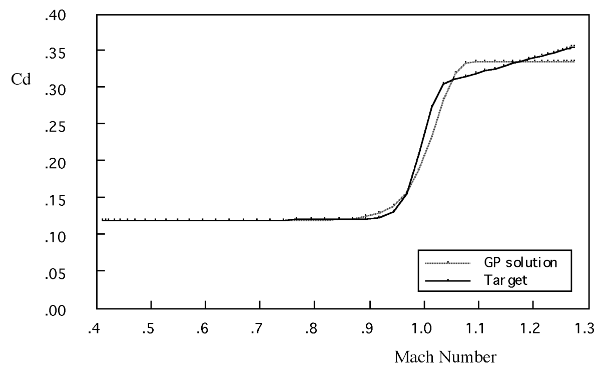


Figure 9: Best fit from 10 runs of the genetic program.

Despite the error in drag curve at supersonic

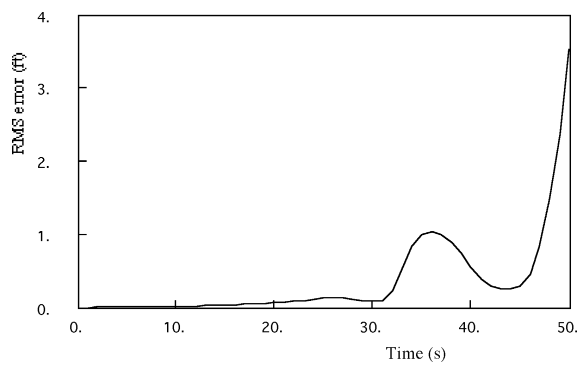


Figure 10: RMS error of trajectory produced by best C_D equation.

speeds, Figure 10 indicates that the position error is less than 4 feet for a weapon dropped from an altitude of 40000 ft. In fact, most of the error is in vertical position, and the error in range is less than 1 foot. Accuracy is much more strongly influenced by the details of separation from the aircraft than it is by the ballistic portion of the flight.

6 Conclusions and Future Work

Genetic programming can be a productive method for system identification of flight vehicles. It can discover equations of novel form, and it produced a function for drag variation with Mach Number with accuracy similar to the one developed by Anderson and McCurdy, which took painstaking effort to derive. Genetic programming is not appropriate for domains where the appropriate functions are well understood, because its broad search is inefficient.

Genetic methods do not guarantee global optimality. Solution quality is strongly dependent on the choice of basis function and terminal sets. The breadth of search should be restricted as far as possible, so that a population of moderate size can give a representative sample of the search space. Genetic programming operators must be chosen with care, with the selection operator exerting a particularly strong influence on algorithm behaviour.

Solutions generated by the genetic program are generally more complex than those produced by humans. They are consequently more difficult to interpret, and less likely to be physically meaningful. In such situations, there is not a significant advantage over "black-box" methods such as neural networks. Further effort is required to investigate parsimony, particularly the adaptive penalty method recommended by Blickle, which was not investigated here. Consideration of new mutation operators to perform inner-loop parameter optimization should also improve solution quality in future applications.

The promising performance of genetic programming on the simulated flight data of a simple weapon justifies further application to flight vehicles. Investigations using real flight data from aircraft with unusual characteristics will be conducted in the future.

Acknowledgements

This work was funded in part by Air Operations Division, AMRL, Defence Science and Technology Organization, Australia. Their support is gratefully acknowledged.

References

[1] Klein, V.; Ratvasky, T.P. and Cobleigh, B.R., "Aerodynamic Parameters of High-Angle-of-Attack Research Vehicle (HARV) Estimated from Flight Data", NASA TM-102692, August 1990.

[2] Hamel, P.G.; Jategaonkar, R.V., "Evolution of Flight Vehicle System Identification",*Journal of Aircraft* Vol 33, 1, pp 9-28, Jan-Feb 1996.

[3] Maine, R.E.; Iliff, K.W., "User's Manual for MMLE3, a General FORTRAN Program for Maximum Likelihood Parameter Estimation", NASA TP-1563, 1980.

[4] Klein,V., "Estimation of Aircraft Aerodynamic Parameters from Flight Data", *Progress in Aerospace Sciences*, Vol 26, pp 1-77, 1989.

[5] Gage, P.; Braun, R.; Kroo, I., "Interplanetary Trajectory Optimization Using a Genetic Algorithm", *The Journal of Astronautical Sciences*, Vol 43, No 1, Jan-Mar 1995, pp 59-75.

[6] Anderson, M.B.; Lawrence, W.R., Gerbert,G.A., “Using an Elitist Pareto Genetic Algorithm for Aerodynamic Data Extraction”, AIAA 96-0514, AIAA Aerosciences Meeting, Reno, NV, Jan 1996.

[7] Stuckey, R.A., “Flight-estimated Spoiler Aerodynamics of the F-111C Aircraft”, AIAA 94-3459, AIAA Atmospheric Flight Mechanics Conference, Scottsdale, AZ, 1994.

[8] Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press, 1992.

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

[10] Liepins, G., Vose, M., “Deceptiveness and Genetic Algorithm Dynamics”, in *Foundations of Genetic Algorithms*, ed. Rawlins, G., Morgan Kaufmann, 1991.

[11] Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[12] Smith, S.F. “A Learning System Based On Genetic Adaptive Algorithms”, PhD Thesis, University of Pittsburgh,1980.

[13] Hensley, D.D., “Optimal Simulator Aerodynamic Model Definition Using a Genetic Algorithm”, AIAA-94-3403, AIAA Flight Simulation Technologies Conference, Scottsdale, AZ, 1994.

[14] Tan, K.C., Li, Y., Murray-Smith, D.J. and Sharman, K.C., “System Identification and Linearization Using Genetic Algorithms with Simulated Annealing”, Proc. First Int. Conf. on GA in Eng. Syst.: Innovations and Appl., Sheffield, UK, 1995.

[15] Anderson, M.B. and McCurdy, R.E., “Weapon Drag Coefficient Determination Using Genetic Algorithm”, AIAA 94-3468.

[16] Ryan, G.W. “A Genetic Search Technique for Identification of Aircraft Departures”, AIAA Atmospheric Flight Mechanics Conference, Baltimore, MD, 1995.

[17] Menon, P.K.; Yousefpor, M.; Lam, T.; Steinberg, M.L., “Nonlinear Flight Control Systems Using Genetic Programming”, AIAA 95-3224, 1995.

[18] Zongker, D., Punch, W. and Rand, W., “*lil-gp* 1.01 User’s Manual”, Free Software Foundation, Inc, Cambridge, MA, 1995.

[19] Gage, P.; Sobieski, I.; Kroo, I., “A Variable-Complexity Genetic Algorithm for Topological Design” *AIAA Journal*, Vol 33, No 11, Nov 1995, pp 2212-2217.

[20] Blickle, T., “Evolving Compact Solutions in Genetic Programming: A Case Study”, International Conference on Evolutionary Computation: Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV), Berlin, Sept, 1996.