Proceedings of the 3rd International Conference on Control, Dynamic Systems, and Robotics (CDSR'16)
Ottawa, Canada - May 9 - 10, 2016
Paper No. 123
DOI: 10.11159/cdsr16.123

Automated Model Tuning Using A Genetic Algorithm

Suzanne Swaine, Robert Langlois
Department of Mechanical and Aerospace Engineering, Carleton University
1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6
suzanneswaine@gmail.com; robert.langlois@carleton.ca

Abstract - A simple but reliable model tuning method was developed in order to tune a flight model for a high-fidelity type-specific small aircraft simulator. A genetic algorithm (GA) was used as a parameter estimation method. GAs are robust parallel heuristic search methods that often use least squares curve fitting methods to solve complex problems. They belong to the class of evolutionary computing algorithms that mimic natural processes, in this case evolution, to predict behaviour and solve optimization problems. A population of possible solution sets is selected at random and the known math model is then used to determine the behaviour of each of these possible solutions. The behaviour of each is then compared to the desired behaviour of the model, i.e., the reference data set, and the error is calculated. Those with the highest error are culled from the population while those with the lowest error are deemed to be "parents". These parent solution sets are then paired together, to create "children" by finding a weighted average of the parents. To ensure the solution space is fully explored, "mutations" are also created by replacing a single part of select solution sets with a randomly-generated value. Two mutation mechanisms were used in the algorithm described in this paper to ensure that the solution space was explored fully while avoiding convergence on a local, rather than global, minimum. The second generation solution set is 50% comprised of parents, 25% comprised of children, and 25% comprised of mutations. The process repeats until the convergence criteria is met. This algorithm was successfully tested with multiple dynamic systems, including simulated flight test data created using X-Plane, a flight simulator software. The algorithm proved to be a capable and adaptive parameter estimation method applicable to a wide variety of dynamic models, including flight models.

Keywords: Flight model, model tuning, genetic algorithm

1. Introduction

Flight model tuning is an important step in the development of a high-fidelity type-specific aircraft simulator. In an effort to develop a low-cost high-fidelity Diamond Aircraft DA20-A1 Katana flight simulator, an automated method for extracting stability derivatives from flight data was developed using a simple, but powerful, genetic algorithm (GA). As will be shown, not only is the developed algorithm capable of tuning a flight model, but it is readily adaptable to potentially tune any system for which the governing equations of motion and the desired behaviour are known. This paper provides an overview of flight model tuning, describes the operation of genetic algorithms, details the development of the flight model tuning algorithm, presents sample results, and provides relevant discussion.

2. The KatanaSim

The KatanaSim project was an investigation into the feasibility of developing a cost-effective high-fidelity type-specific small aircraft simulator for use as part of an ab-initio flight training program. The development of the KatanaSim, a Diamond DA20-A1 Katana flight simulator built with commercial off-the-shelf components and original aircraft parts wherever possible, proved the feasibility of the endeavour. Particularly successful was the high level of physical fidelity achieved by the use of original aircraft parts, most notably a Katana fuselage. The development cost of the KatanaSim was significantly less than the six and seven figure costs usually associated with high-fidelity flight simulators.

In order to acquire the data necessary for flight model evaluation and tuning, a minimally-intrusive flight testing methodology for small aircraft was developed. A compact instrumentation package was designed and tested, and a flight permit from Transport Canada was obtained. Two flight tests in a DA20-A1 Katana were completed and several hours worth of flight testing data was acquired.

X-Plane, by Laminar Research, was chosen as the core simulator software and Plane Maker, a part of the same package, was used to develop the flight model. Aircraft model parameters were obtained from published data sources, empirical measurements, and observations. The resulting flight model required additional tuning, discussed subsequently, to meet the desired performance specifications.

3. Flight Modelling

There are two main approaches to flight modelling: model prediction and parameter estimation. Model prediction does not require the equations of motion of the system to be known. Model prediction methods use complex algorithms, like neural networks, to predict model behaviour based on given sets of input and output data. Rather than optimizing the defining parameters of the system, then entering these parameters into the mathematical model, model prediction replaces the mathematical model completely.

In contrast, parameter estimation usually requires the equations of motion of the system to be known and the parameters contributing to the dynamics of the system to be identified. Flight test data or the desired performance data is used to optimize the flight parameters in order to produce the desired system output. Some examples of parameter estimation techniques include the output error method, Kalman filters, and genetic algorithms.

Of these two methods, parameter estimation was favoured by the authors due to the proven reliability of industry-standard aircraft mathematical flight models.

3.1. Neural Networks

Neural networks (NNs) belong to the class of evolutionary computing algorithms and are a mathematical metaphor for a biological brain. The type used most frequently in modelling and simulation is the feed-forward network with back-propagation learning [1]. The network is given input-output pair examples and the back-propagation algorithm works to reduce the error of the program. Neural networks have a broad range of uses but have been applied specifically to parameter estimation problems like aerodynamic coefficient prediction for wind tunnel data [2] and flight vehicle parameter estimations in which the aerodynamic coefficients of a flight vehicle were calculated given the initial vehicle orientation, velocities, accelerations, and control inputs.

One of the most unique aspects of NNs is that they do not require actual mathematical modelling of a system in order to predict the response of that system, making them ideal for predicting the behaviour of complex systems. In some cases where the design of a flight simulator includes the development of the flight model engine (rather than purchasing COTS software) NNs are used as a method of predicting aircraft behaviour [3].

3.2. Output Error Method

The output error method is a maximum likelihood estimation technique based on minimizing the quadratic error between simulated and actual system outputs. Nonlinear programming algorithms are used to solve a nonlinear least-squares problem in order to converge on the optimal system parameters [4].

In Lee and Yoon [5], the output error method proved its value in flight modelling but the method does have weaknesses. The output error method will converge well on a local maximum but global convergence does not occur. More accurate results can be produced using prediction error methods or other algorithms which are better suited for global convergence [6].

3.3. Kalman Filters

Extended Kalman filters (EKFs) and unscented Kalman filters (UKFs) have both been used for recursive flight parameter estimation. Using filters for parameter estimation requires modelling the problem as a state estimation problem and then artificially defining the unknown parameters as state variables before solving. EKFs work on the principle of applying instantaneous linearizations to each time step in order to approximate the nonlinearities. Strong nonlinearities in a system can cause difficulties with the EKF, leading to incorrect results or divergence. Despite this, EKFs are one of the most popular filtering techniques in the aerospace industry and have been implemented successfully in a variety of parameter estimation problems. UKFs overcome the inability of EKFs to handle nonlinear systems by avoiding the approximations introduced by the linearizations in the EKFs but also require more processing power to do so [7].

3.4. Genetic Algorithms

Genetic algorithms (GAs) belong to the class of evolutionary computing algorithms that attempt to mimic natural processes to predict behaviour and solve optimization problems ranging far beyond flight modelling. The widely adaptable nature of these algorithms allows them to solve problems in fields ranging from economics to medicine [1]. GAs solve these problems using curve fitting, a process by which a closed-form function is approximated in order to provide the line of best fit to a given data set. There are many ways to approximate a function; however, given a complex function with a nonlinear form or without information regarding derivatives, there are few available methods that work well. Genetic algorithms are robust parallel heuristic search methods that often use least squares curve fitting methods to solve complex problems. Perhaps one of the greatest strengths of a GA is the ability it has to avoid convergence on local optima by completing a thorough search of the possible solution space. With even the weakest correlations between proposed solutions and function values, a GA will almost always be able to produce a nearly-optimal solution [8].

Prior use of genetic algorithms for flight model parameter identification has been almost exclusively in tandem with other parameter estimation techniques. The GAs have been used to optimize other algorithms and are popular in their ability

to improve neural network performance; however, the GAs have not previously been applied, to the authors's knowledge, as a flight model estimation method on their own.

This investigation into the use of GAs was motivated by the simplicity and the proven curve-fitting ability of GAs as well as the novelty of use for flight model parameter estimation. Additionally, GAs, unlike the output error method, are adept at converging on the global minimum and require less computational power than Kalman filters. A GA was chosen over a neural network because the design and implementation of a GA was deemed simpler and more rigorous than a neural network.

4. Development of a Genetic Algorithm

GAs mimic the role of genotypes and phenotypes in natural selection. The genotype of an organism is the information stored in the genetic code of that organism. The genotype holds the blueprints for that organism's appearance and behaviour. The actual appearance and behaviour themselves are called the phenotype [1]. Given an initial population and an environmental pressure which favours a particular phenotype, the genetic code of the population will trend towards those genotypes which can produce the most suitable phenotypes for the environment. New genotypes are produced with each generation due to the sharing of chromosomes through reproduction and the introduction of new chromosomes through random mutations. As the less-suitable genotypes do not reproduce, beneficial chromosomes and mutations are passed on to future generations through a process known as natural selection.

Mimicking this process in modelling, the coefficients in an equation set are assembled into a bit string that is analogous to the genotype of an organism. The behaviour of the system with those coefficients is the phenotype. A population is created with a number of different genotypes. An environmental pressure is then applied by culling those genotypes which do not closely adhere to the desired behaviour of the system.

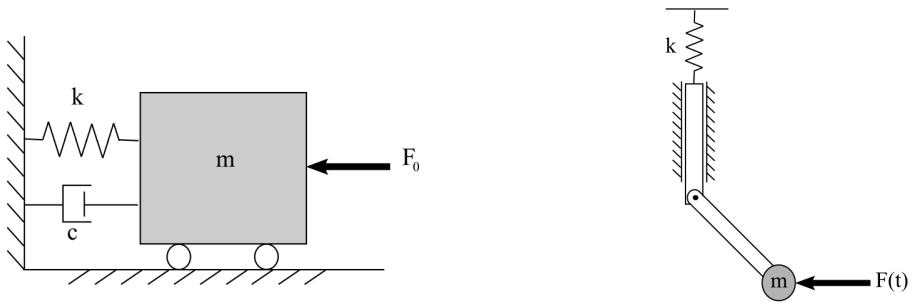
4.1. Development Example 1: Mass Spring Damper System

In order to develop a curve fitting GA which could be used for flight model tuning, a series of simple, classical problems was explored first to refine the selection, breeding, and mutation mechanisms.

The first problem was based on a basic cart, spring, and damper system as shown in Figure 1a. The GA was designed to determine the mass of the cart m, linear spring constant k, and viscous damping coefficient c, given the initial conditions of the system and the behaviour of the cart. A reference data set was created by assigning values for the mass of the cart, the spring constant, and the damping coefficient. The dynamics of the cart were then calculated, given an initial force, cart position, and cart velocity. The position of the cart was recorded at intervals of 0.05 seconds for a total of 30 seconds. This array became the reference data (phenotype) against which future possible solution sets (genotypes) would be compared and ranked.

4.1.1. Survival of the fittest

This example problem was used to extensively test the performance of the genetic algorithm with a variety of population sizes and child and mutation creation methods. Due to the simplicity of the problem, a population size of forty was chosen. The initial coefficient sets were randomly generated given certain assumptions on the upper and lower limits of the possible values of m, c, and k. The response of the system was calculated for each combination of m, c, and k and then compared to the reference behaviour array. A fitness function, the sum of squares error (SSE) was then used to compare each set. This fitness value determined whether or not that particular solution would be culled from the population, or whether it would be



(a) System consisting of a cart with spring and damper

(b) Sliding pendulum system

Fig. 1: Dynamic systems used for the creation and development of a simple curve-fitting GA.

part of the next generation. The solutions were ranked from lowest to highest SSE, and those in the lowest half were marked as the survivors and carried over to form 50% of the next generation. The better half of the surviving population members were deemed as suitable "parents" for the next generation.

4.1.2. Children

After the least fit population members were culled, it was desirable to repopulate and keep the population constant from one generation to the next. Twenty-five percent of the population was filled by "breeding" children from the best-ranked parents. Parents 1 and 2 were paired together to create two children, Parents 3 and 4 were paired together to create two children, and so on, so that the ten designated parent genotypes produced ten new genotypes to test with the system. The i^{th} coefficients, or "chromosomes" of the children, were found using an arithmetic crossover of the i^{th} chromosomes of the parents [1].

$$\text{Child1}_i = \alpha * \text{Parent1}_i + (1-\alpha) * \text{Parent2}_i$$
$$\text{Child2}_i = \alpha * \text{Parent2}_i + (1-\alpha) * \text{Parent1}_i$$

There are infinite possibilities for breeding children by varying parameter α, the number of chromosomes affected, how many children are produced from each parent set, and the selection of parents for breeding. Applying the above formula to each chromosome was found to result in the best performance of the algorithm when α was set equal to 0.35.

4.1.3. Mutations

After the children were bred, the remaining 25% of the population was filled with mutations. Mutations are a way of increasing the number of paths being explored in the solution space and decreasing the likelihood of convergence on a local optimum [1].

Mutations are created by selecting one or more chromosomes in a given genotype to be summed with a randomly generated value. A higher frequency of chromosome mutation will result in greater exploration of the possible solution space, but too high a frequency could slow solution convergence.

The mutation driver in this GA was applied to the parents. One coefficient from each solution was randomly chosen for mutation, and a value Z was added to it. Originally, the value Z was defined as a random number between -2 and 2 multiplied by the range of the values of that chromosome in the population. This mutation method was taken from Gulsen [8]. Using this mutation method alone resulted in occasional convergence at a local, rather than global, minimum. To avoid this effect, a second mutation method was established. Once convergence of the SSE had reached a certain threshold, the mutation value Z would be defined as a random number between -2 and 2 multiplied by 10% of the value of that chromosome.

4.1.4. Iterations and convergence

Once the ten child genotypes and the ten mutation genotypes had been created, they were added to the original twenty randomly-generated genotypes chosen as parents, to complete the population set. The behaviour of the genotypes (the dynamics of the cart system) were then evaluated and the entire population was reordered from smallest to largest SSE. The twenty genotypes with the highest SSEs were culled and the next generation of children and mutations was created. This continued until the lowest SSE was less than a chosen threshold. The algorithm was used to solve for the unknown cart-spring-damper system coefficients 550 times to establish the repeatability of the genetic algorithm. The average error between an actual coefficient and a coefficient tuned by the algorithm was 0.33% and the largest error was 0.92%.

After confirmation that this GA worked well in determining the mass, spring constant, and damping coefficient of the system, the size of the population was increased to 1000. This increase in size greatly decreased the number of algorithm iterations, or generations, required to reach convergence.

4.2. Development Example 2: Sliding Pendulum

In the second phase of development, the GA was used to find the unknown mass and spring constant in a sliding pendulum system, shown in Figure 1b, given a time-dependent forcing function. While the GA was used to only find two, rather than three unknown coefficients, solving this system confirmed the algorithm could perform equally well when tuning systems with multiple coupled degrees of freedom.

4.3. Development Example 3: Planes and Polynomials

In the third phase of development, the GA was used to solve for the coefficients of a variety of polynomials describing curves and planes. The objective of this phase was to evaluate the ability of the algorithm to handle a larger number of

unknown coefficients than had previously been tested. Populations of up to 5000 were tested with polynomials with as many as 15 unknown coefficients. The polynomial equations were chosen to represent a wide variety of curve shapes, sizes, and complexities. No unforeseen problems with the algorithm were encountered and model convergence was achieved for all cases.

4.3.1. Sequential Evolution Mechanisms

One particular consideration for some curve-fit-optimization problems is how the contribution of each element to the overall fitness of a genotype can vary significantly, i.e., the higher-order coefficients in a polynomial will have a greater effect on the phenotype than the lower-order coefficients. If breeding and mutations affect each chromosome equally, the evolution of the genotype will focus on the more influential chromosomes at the expense of the less influential chromosomes.

One solution for this phenomenon was the implementation of sequential evolution mechanisms (SEMs) as developed by Gulsen, Smith, and Tate [8]. In this procedure, chromosomes are grouped by the magnitude of their effect on the phenotype, and only one group at a time will be activated during the breeding and mutation sequences used to create the next generation. In the most-extreme form of SEM, each chromosome could evolve separately. In most SEMs, each chromosome group is activated sequentially so that each group is given an equal number of evolutions before convergence [8].

5. Flight Model Tuning Using Stability Derivatives

The final step in successfully applying the GA to flight model tuning was to use linearized small-disturbance rigid body equations of motion to determine the stability derivatives of an aircraft, based on recorded flight dynamics data.

Flight simulator software generally uses either stability derivatives or blade element theory to drive the simulator flight model. The required flight model parameters in each system differ greatly, with stability-derivative-based software requiring performance parameters (either aerodynamic coefficients or the stability derivatives themselves), and blade-element-theory-based systems requiring physical parameters (geometry and engine power specifications). Had the flight dynamics model for X-Plane been readily available, the geometric parameters of the X-Plane aircraft would have been tuned to produce the most accurate flight model possible for the KatanaSim. As this was not the case, a different math model was selected in order to aid in the development of an automated flight model tuning method, with the understanding that in a future phase of development, the compatibility issue between the model tuning algorithm and the KatanaSim software would be resolved.

5.1. Tuning of Two-dimensional Stability Derivatives

The longitudinal equations of motion for an aircraft were derived, encompassing the forward and vertical motion as well as pitch changes of the aircraft. A two-dimensional analysis was chosen, as the resulting equations would be adequate for initial testing of the genetic algorithm before implementing the full set of equations required to define the system in three dimensions. Figure 2 defines the relevant angular and linear velocities and accelerations.

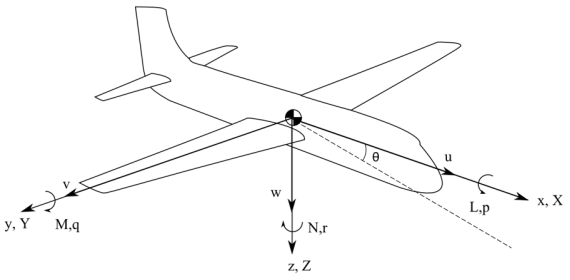


Fig. 2: Definitions of angular and linear axes, forces, moments, and velocities.

The rigid body equations governing aircraft motion in response to the X and Z aerodynamic forces, and the aerodynamic moment M, are written as:

$$X - mg\sin\theta = m(\dot{u} + qw) \quad (1)$$

$$Z + mg\cos\theta = m(\dot{w} - qu) \quad (2)$$

$$M = I_y\dot{q} \quad (3)$$

The equations above are nonlinear, coupled, and lend no direct insight as to the handling characteristics of the aircraft [9]. The linearized approximations of these equations are much more useful in analysing the stability and control characteristics of the aircraft. These approximations can be found using the small-disturbance theory in which the motion of the aircraft is described as small perturbations about a reference condition [10]. Because of these approximations, the resulting equations of motion cannot be used to produce accurate results for manoeuvres where large-amplitude perturbations occur, as is the case for spins and stalls. Note that these approximations are not found in blade element theory models, thereby providing another advantage of using the X-Plane software for the KatanaSim model.

The final version of the linearized small-disturbance longitudinal rigid body equations of motion were written in state space form as follows:

$$\begin{Bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{Bmatrix} = \begin{bmatrix} X_u & X_w & 0 & -g\cos\theta_0 \\ Z_u & Z_w & u_0 & -g\sin\theta_0 \\ M_u + M_{\dot{w}} & M_w + M_{\dot{w}} Z_w & M_q + M_{\dot{w}} u_0 & -M_{\dot{w}} g\sin\theta_0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} u \\ w \\ q \\ \theta \end{Bmatrix} + \begin{bmatrix} X_{\delta_e} & X_{\delta_T} \\ Z_{\delta_e} & Z_{\delta_T} \\ M_{\delta_e} + M_{\dot{w}} Z_{\delta_e} & M_{\delta_T} + M_{\dot{w}} Z_{\delta_T} \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \delta_e \\ \delta_T \end{Bmatrix}$$

where the vector containing u, w, q, and θ is the state variable vector and the vector containing δ_e and δ_T is the control vector. Combined, these two vectors comprise the required inputs to the algorithm. The constants in the matrices (other than the acceleration due to gravity g and the nominal pitch angle θ_0) are the unknown stability derivatives. The full derivation and definition of these equations is available [11].

The inputs could have been acquired either through flight testing or through flight simulation. Due to test schedule constraints, flight simulator output files were selected for testing. Simulated flight data from a Cessna 172 model was collected using X-Plane. The X-Plane output file contained the roll and pitch angles, angular rates, control surface deflection angles, angle of attack, angle of sideslip, and thrust of the aircraft. Once the inputs and reference performance curves had been defined, the tuning parameters were set as the 14 stability derivatives used in defining the longitudinal motion of the aircraft. The pseudo-random number generator used to create the initial population was set to generate random numbers between -1.0 and 1.0 but no limitations were put on how high or low the mutations could push the coefficients.

The algorithm was initiated using the X-Plane output as the reference input file for the program. As the flight model was significantly more complex than the previously-tested models, the convergence criterion was redefined. Rather than having a fixed SSE goal, the final version of the genetic algorithm defined convergence to be when the mutations and children were no longer improving the solutions. Figure 3a shows the convergence plot where the SSE began with a value over 96000 and converged with an SSE of 79.2 for the 1500 performance points being matched. Figures 3b, 3c, and 3d, show the agreement between the reference curves and the curves produced using the tuned stability derivatives.

5.2. Tuning of Three-dimensional Stability Derivatives

Once tuning the longitudinal stability derivatives using the GA had been proven possible using the two-dimensional formulation, it was a natural next step to expand the algorithm to perform a three-dimensional analysis which would include the lateral and directional stability derivatives as well. As was done with the longitudinal equations of motion, the lateral and directional equations were derived and written in state space form as

$$\begin{Bmatrix} \dot{v} \\ \dot{p} \\ \dot{r} \\ \dot{\Phi} \end{Bmatrix} = \begin{bmatrix} Y_v & Y_p & -(u_0 - Y_r) & g\cos\theta_0 \\ L_v & L_p & L_r & 0 \\ N_v & N_p & N_r & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{Bmatrix} v \\ p \\ r \\ \Phi \end{Bmatrix} + \begin{bmatrix} 0 & Y_{\delta_r} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \delta_a \\ \delta_r \end{Bmatrix}$$

where the structure is similar to the previously-described longitudinal equation [11]. The longitudinal stability derivatives remained the same despite the fact that the derivations of the two-dimensional and three-dimensional equations of motion differ slightly. The additional 14 stability derivatives were combined with the previous tuning parameters resulting in a total of 28.

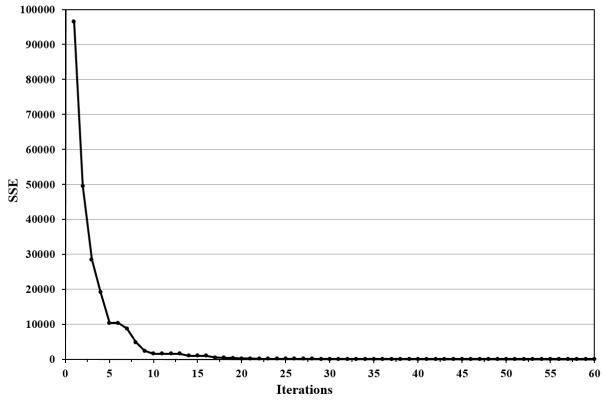
A new input file was created by flying a flight in X-Plane which included pitch, roll, yaw, and throttle inputs. The GA converged with an SSE of 74.5 for the 2280 performance points being matched. The reference accelerations and those produced by the 28 tuned stability derivatives were plotted and the results are shown in Figures 4a through 4f.

6. Discussion

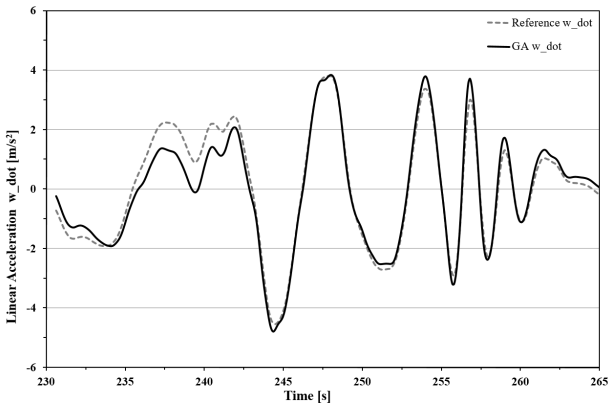
The ability of the GA to tune the stability derivatives defining the equations of motion of an aircraft was tested using simulated flight test data from X-Plane. The GA was able to optimize the coefficients to produce performance curves that closely match the reference curves. Some variations in the curves were seen but these were not unexpected. Because the

reference curves originate from the blade-element-theory-based math model in X-Plane, while the algorithm calculates the performance curves using a stability-derivative-based math model, differences in the data were inevitable due to the assumptions and omissions that are inherent in the stability derivative derivations.

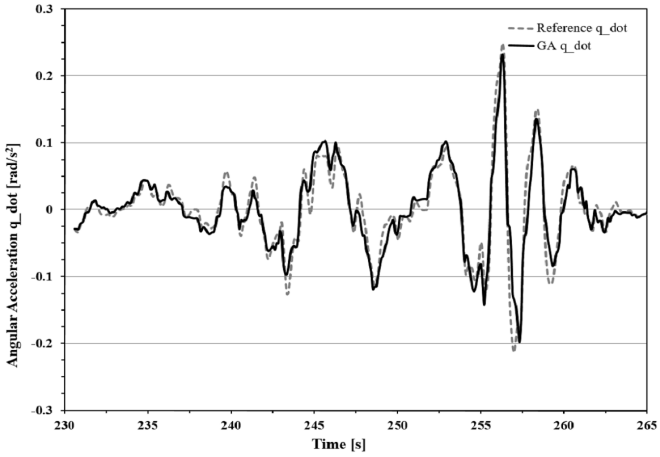
The genetic algorithm that was developed proved to be a very reliable, capable, and adaptable parameter estimation method. The simplicity and robustness of the algorithm make it well suited for adaptation to a variety of dynamic models, flight models, and similar applications.



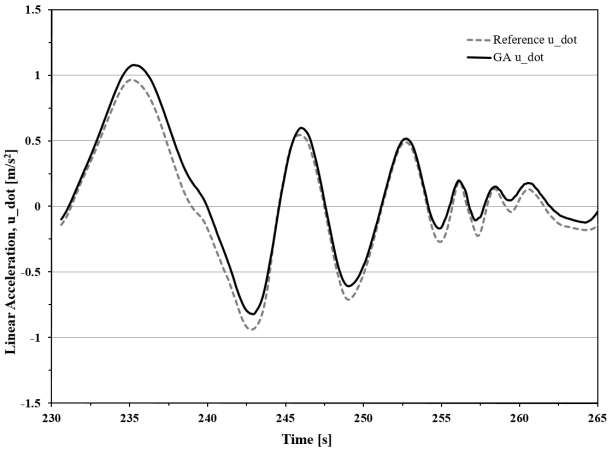
(a) Convergence of SSE by iterations.



(b) Linear acceleration, $\dot{w}$

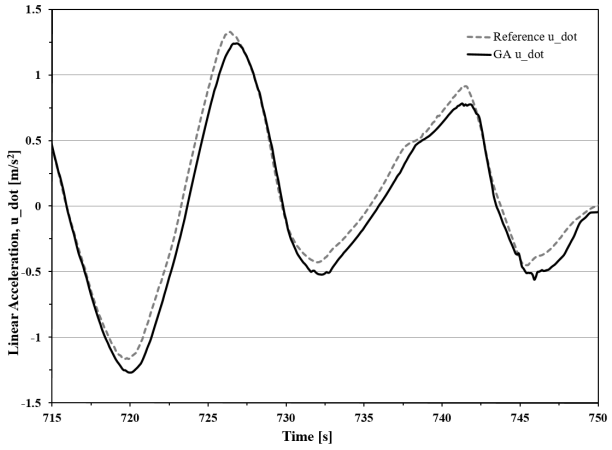


(c) Angular acceleration, $\dot{q}$

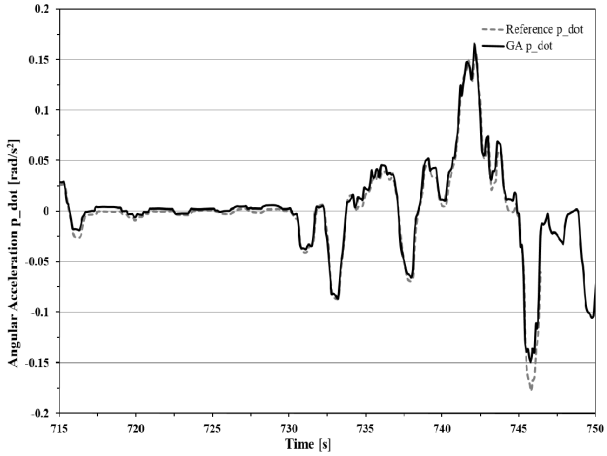


(d) Linear acceleration, $\dot{u}$

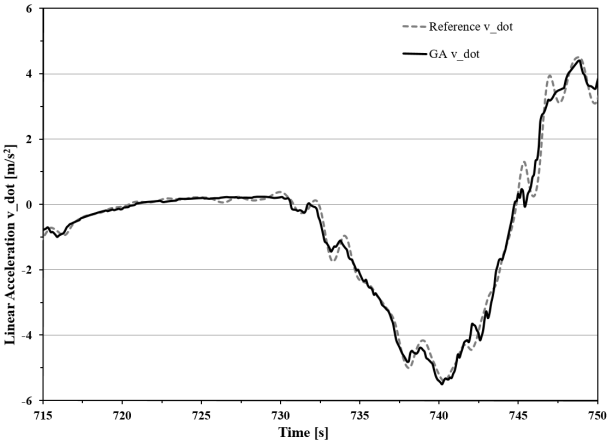
Fig. 3: Linear accelerations versus time, for calculated data (black lines) from the GA and reference data from X-Plane (dashed lines) in two dimensions, as well as the convergence plot.



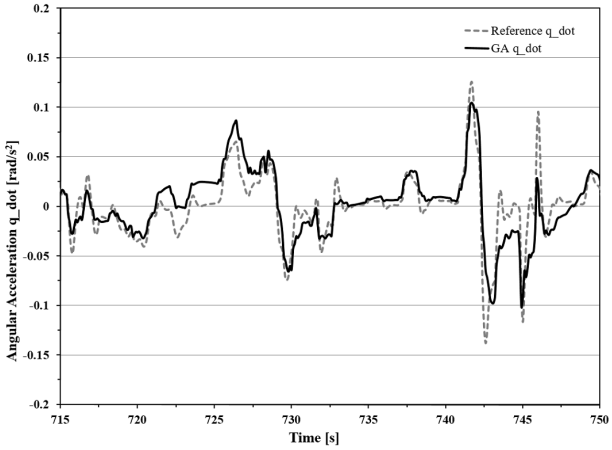
(a) Linear acceleration, $\dot{u}$



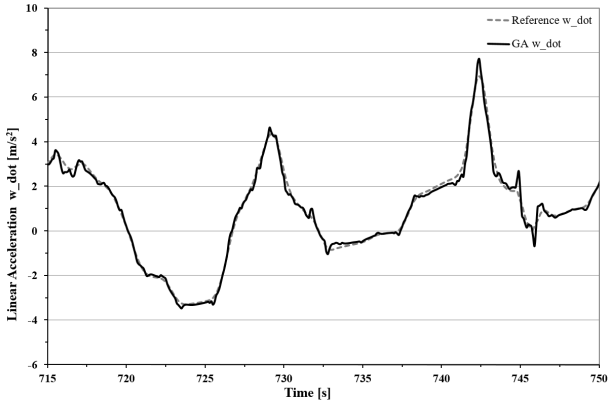
(b) Angular acceleration, $\dot{p}$



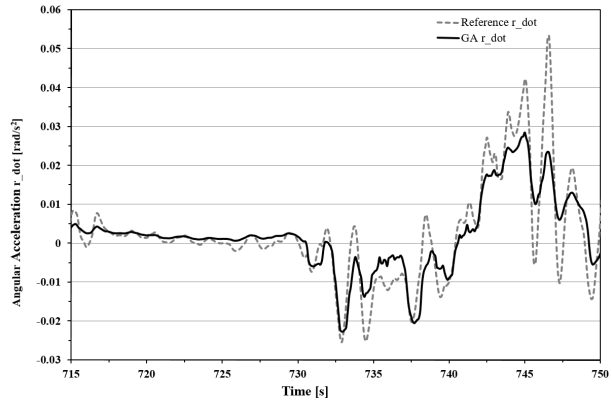
(c) Linear acceleration, $\dot{v}$



(d) Angular acceleration, $\dot{q}$



(e) Linear acceleration, $\dot{w}$



(f) Angular acceleration, $\dot{r}$

Fig. 4: Calculated data from the GA (black lines) and reference data from X-Plane (dashed lines) in three dimensions.

Acknowledgements

Financial support for this research was provided by the Ontario Centres of Excellence, Mitacs, the Natural Sciences and Engineering Research Council of Canada, Ottawa Aviation Services, and B-Con Engineering.

References

[1] R.E. Marks and H. Schnabl, "Genetic Algorithms and Neural Networks: A Comparison Based on the Repeated Prisoners Dilemma," in *Computational Techniques for Modelling Learning in Economics*, T. Brenner, Ed. Boston: Springer US, 1999, pp. 197–219.

[2] T. Rajkumar and J. Bardina. "Prediction of Aerodynamic Coefficient Using Genetic Algorithm Optimized Neural Network for Sparse Data," 2002.

[3] G. Valmorbida, W. Lu, and F. Mora-Camino, "A Neural Approach for Fast Simulation of Flight Mechanics," in *38th Annual Simulation Symposium, 2005. Proceedings.* 2005, pp. 168-172.

[4] M. Gautier and A. Janot and P. Vandanjon, "A New Closed-loop Output Error Method for Parameter Identification of Robot Dynamics," *IEEE Transactions on Control Systems Technology,* vol. 21, no. 2, pp. 428–444, 2013.

[5] J. Lee and S. Yoon, "Application of Parameter Estimation to FTD for a Light Airplane via Flight Tests," *Simulation Modelling Practice and Theory,* vol. 15, no. 6, pp. 660 – 702, 2007.

[6] T. Söderström and P. Stoica, "Some Properties of the Output Error Method," *Automatica,* vol. 18, no. 1, pp. 93-99, 1982.

[7] G. Chowdhary and R. Jategaonkar, "Aerodynamic Parameter Estimation from Flight Data Applying Extended and Unscented Kalman Filter," *Aerospace Science and Technology,* vol. 14, no. 2, pp. 106-117, 2010.

[8] M. Gulsen, A.E. Smith, D.M. Tate, "A Genetic Algorithm Approach to Curve Fitting," *Int. J. Production Research*, vol. 33, no. 7, pp. 1911–1923, 1995.

[9] D. A. Caughey. Introduction to Aircraft Stability and Control Course Notes [Online]. Available: https://courses.cit.cornell.edu/mae5070/Caughey_2011_04.pdf

[10] R.C. Nelson, *Flight Stability and Automatic Control*, New York: McGraw-Hill, 1998.

[11] S. Swaine, "Development of a Cost-effective High-fidelity Type-specific Flight Simulator with Emphasis on Flight Modelling," M.A.Sc. thesis, Dept. Mechanical and Aerospace Engineering, Carleton University, Ottawa, Canada, 2014.