Stochastic Design of a Non-Linear Controller for Autonomous UAV Recovery

Khantsis, S. and Bourmistrova, A.

School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University, Melbourne
E-Mail: anna.bourmistrov@rmit.edu.au

Keywords: *UAV recovery; flight control; evolutionary algorithms; stochastic optimisation*

EXTENDED ABSTRACT

An Unmanned Aerial Vehicle (UAV), like any other aircraft, represents a complex non-linear system with, generally, high degree of coupling. It is anticipated that the most effective control on such a system can be gained with an appropriate non-linear controller. However, design of a non-linear flight controller is a demanding task which usually requires deep engineering knowledge of intrinsic aircraft behaviour. A traditional approach is to linearise the aircraft model (and all linked models such as wind disturbances model) about the average 'trim' conditions and to apply one of various well developed linear design techniques. Although in many cases this approach is reasonable, it may not deliver enough robustness when the aircraft model has significant uncertainties and/or when substantial non-linear effects are expected.

This paper presents a successful implementation of a design approach which does not require linearisation nor decomposition of any sort of the aircraft model. A combination of evolutionary methods is used to evolve a non-linear controller which fulfils the predefined task.

The engineering problem being addressed is recovery (landing) of a small fixed-wing UAV on a frigate ship deck. The designed controller is expected to guide the aircraft from a point in vicinity of the glidepath to a narrow area near the deck where successful recovery is possible. The task is complicated by various types of disturbances: uncertainty of the starting point, turbulence, sensor noise, ship motion.

The whole system is implemented in MATLAB/Simulink environment. The 'driving' part consists of a set of programs which compose and encode controllers and employ Evolutionary Strategy algorithms geared with Genetic Programming approach to evolve a capable and robust controller. The evolution starts from the control laws as simple as $y = const$ and require no a priori knowledge about the controlled system. Such evolutionary design is performed step by step, gradually complicating the task and thus allowing the controller to gain experience effectively. In the process of complication need for full convergence is initially relaxed.

The 'simulation' part represents a group of non-linear Simulink models used to estimate the capabilities of the controllers. They are employed both in evolutionary search to estimate the fitness of a particular controller, and later to verify and validate the designed controllers through extensive simulation in various conditions.

It is demonstrated that effective design is possible using essentially only the simulation data from a comprehensive fully coupled non-linear model. This allows treating the target system as a 'black box,' without applying sophisticated system identification techniques. However, it is noted that tailoring the controllers representation and evolutionary algorithms utilising the basic flight control knowledge significantly improves efficiency of the design process.

1. INTRODUCTION

Over the hundred years of aviation history, various linear control methods were successfully used in aerospace area due to their simplicity and validity which can be mathematically determined. Despite their natural limitations, linear control techniques still remain as one of the most accepted design practices. However, growing demands on the performance of aircraft and, on the other hand, remarkable increase of available computation power over the last years have led to significant growth of the popularity of non-linear control techniques.

A principal difficulty of many non-linear control methods, which potentially could deliver better performance, is impossibility or extreme difficulty to theoretically predict the behaviour of a system under all possible circumstances. In fact, even design envelope of a controller often remains largely uncertain. Therefore, it becomes a challenging task to verify and validate the designed controller under all possible flight conditions. A practical solution to this problem is extensive testing of the system (or rather of its mathematical model), which is a computationally and time demanding engineering task. The design process itself is often built around the continuous testing of the controller in loop with a real system or its mathematical model rather than relies on theoretical analysis only. It becomes clear that there is a need to develop a non-linear design methodology, especially in application to such a complex system as a UAV controller.

Evolutionary Algorithms (EA) is a group of stochastic design and optimisation methods which combine such important characteristics as robustness, versatility and simplicity. They are inspired by the power of natural evolution and, indeed, proved the success in many applications, such as neural network optimisation (Sendhoff and Kreuz 1999), finance and time series analysis (Mahfoud and Mani 1996), aerodynamics and aerodynamic shape optimisation (McFarland and Duisenberg 1995; Olhofer et al. 2001), automatic evolution of computer software (Koza 1992) and, of course, control (Chipperfield and Flemming 1996). However, the majority of applications of EAs is focused on optimisation task. In contrast, this work proposes a design methodology in which EAs are used as a core of the design process. A set of different EAs is used to *develop* and optimise the control laws of the UAV recovery controller.

In order to allow the evolutionary methods to estimate the fitness of a controller, they need a mathematical model of the target system. The fitness of the controller is evaluated through a simulation run (or several runs) of the model with the controller in the loop. The controller is assigned with a fitness measure taking into account both mission fulfilment and quality of control. Therefore, the mathematical model is used in a 'direct' way, without linearisation, decomposition of any kind, obtaining derivatives, etc. This is a very convenient feature, because it allows using third-party models in out-of-the-box fashion.

However, EAs require a large number of fitness evaluations (i.e. simulation runs). For some comprehensive models, this may be computationally too expensive. There are several techniques that deal with this problem, for example, metamodel strategies (Emmerich et al. 2002). In this work, a classic approach is used: during evolutionary search (design stage), the target system is simulated with lesser precision and several modules are excluded. When a set of candidate solutions is found, they are thoroughly tested in 'full' mode.

Another issue is optimisation of the evolutionary methods themselves. Unfortunately, EA theory is still in its early stages of development and has little predictive power. Most of the findings in this area are based on empirical data and engineering intuition. There are three areas in which the EAs are tailored in this work: adaptive encoding of the control laws, which allows them to change the size and structure during evolution; decoupled evolution of the numeric coefficients and the structure of the laws; and multi-stage evolution, aimed at gradual 'learning' of the controllers.

This paper demonstrates how EAs can be used for controller design. The design process consists of iterative application of the EAs to the UAV model linked with the controller, and the following testing of the best controllers.

2. UAV RECOVERY PROBLEM

The problem being addressed is recovery (landing) of a small fixed-wing UAV (Unmanned Aerial Vehicle) on a confined space such as frigate-size ship deck. The proposed recovery method involves capture of a damped arresting wire, stretched over the deck between two poles or in a similar manner, by an onboard flexible trailing line with a self-locking hook attached at the end. The method is somewhat similar to that used in conventional aircraft landings on a carrier, but the UAV is slowed down in the air without using valuable deck space as a runway (Crump et al. 2003).

Only final approach is considered, i.e. the last 8–20 seconds of the flight. The mission controller is supposed to pilot the aircraft to the point approximately 300 metres before the 'touchdown' point on a perpendicular to the arresting wire, with 0–

20 m elevation relative to the wire and with normal flight airspeed and attitude.

The controller being designed guides the UAV from this point to the recovery point, set 1.5 m above the middle of the arresting wire. There are several challenges that make the approach not as trivial as the flight along a straight glidepath line. First, the requirements to the precision are quite high: 2.5–3 m for elevation and ±2.5 m for sideways displacement. Meanwhile, the number of available sensors is very limited; for example, no accurate altitude sensor is available onboard. Second, the whole system is affected by different types of disturbances. Apart from the random starting point, the aircraft is subjected to wind turbulence during the flight, including ship airwake; sensor noise; ship motion.

To allow the EA to work, the models of both the UAV and the environment must be implemented. The models are build so that they can be connected or disconnected at any time, allowing testing of the UAV in various conditions.

3. THE UAV AND ENVIRONMENT MODELS

3.1. The Ariel UAV

The aircraft chosen as a prototype for the research is the UAV *Ariel* (Newman and Wong 1993), which is an unmanned aircraft developed by the UAV group in the Department of Aeronautical Engineering at the University of Sydney. The Ariel is a relatively small aircraft (35 kg maximum take-off weight and 3 m wingspan), manufactured predominantly from fibreglass and foam components. The UAV has three main control surfaces (elevator, ailerons and rudder), plus flaps and throttle control. Flaps have only fixed settings and are set to the specified landing position. The remaining four controls are under full authority of the landing controller.

A six degree-of-freedom non-linear model of the UAV has been implemented in MATLAB/Simulink programming environment. This model is a further development of the model by (Crump 2002), which was used for the design of a take-off controller for the same aircraft. The model takes into account all measurable dependencies such as control surfaces deflections on moments and forces, propeller torque, etc. Most of the aerodynamic data, used by the model, are taken directly from lookup tables, which have been obtained through wind tunnel testing of a prototype aircraft. Although fairly complicated, the model is built upon directly measured data and common rigid body dynamic models, and thus simple enough to design. The model has been validated (within the normal operating envelope) against the prototype's flight test data and the reference model (Newman and Wong 1993).

The UAV model also includes linear models of the onboard sensors and actuators, which optionally allow simulate sensor noise.

The model of the onboard winch, and, in particular, of the flexible line with the arresting hook, has been implemented as a plug-in module to the main UAV model. The line can be 'deployed' at any time by enabling this model. The model uses lumped mass method and Kane's dynamic equations (Trivailo et al. 2002), with several modifications that allow realistic simulation of highly flexible slack lines.

However, as the line is not supposed to be very elastic (slowing down is provided mostly by the shipboard arresting wire), its natural longitudinal frequency is quite high as compared to the short-period UAV motion. Therefore, it requires very small time steps for accurate simulation and thus a lot of computation resources. For that reason, this model has been used, in the first place, to estimate the sag of the line in various flight conditions and, therefore, the allowed elevation of the UAV relative to the arresting wire at the moment of capture. In addition, the drag of the line has been obtained.

At the design stage, the winch model was not employed. Instead, the pre-calculated parameters (line sag and aerodynamic drag) were used in simulation and to determine if the recovery was successful. Later, the winch model was used for verification of the designed controllers: a part of the verification runs were made with the full model simulation.

3.2. Wind Models

The wind model is divided into five separate modules. The first model calculates standard atmospheric parameters (temperature, air density, air pressure) at a given altitude. *Steady wind* model calculates wind magnitude at a given level. The calculations are based on a given wind direction and mean wind magnitude at 6 m of altitude. It also takes into account statistical dependency of wind magnitude on altitude for wind shear.

The *von Kármán wind turbulence model* is build for low altitude profile. The required disturbances are obtained by passing white noise through third-order shaping filters. *Gust model* employs '1–cosine' model from the same standard. It is used only at the verification stage to estimate the UAV behaviour in presence of random gusts.

The *ship airwake model* required more customisation. With no appropriate airwake model available in the public domain, the carrier airwake model has been used as the basis. However, the latter was designed for normal carrier-based operation of piloted aircraft. It needed downscaling to a frigate-size ship and, more importantly, incorporating varying possible glidepaths, including down- and crosswind cases.

3.3. Ship Model

The last model used in the simulation environment is the model of ship motion. The motion consists of two motions: steady, or commanded, motion, which is set as an external condition before landing, and uncommanded oscillatory motion, caused by the wind and waves. As a human being is not involved in the control loop, nor a motion predictive system is installed, a simple one-mode harmonic approximation has been used to simulate wave motion. The amplitude and frequency of the oscillations for each of the six motion components – bank, yaw, pitch, surge, heave, sway – were obtained using statistical data for FFG Sydney from (Hope 1996). These parameters depend on the Sea state and the heading of the ship relative to the waves direction. The objective of the research is to provide a good recovery success rate up to Sea state 6 (Fleet Oceanographic and Acoustic Reference Manual 1999).

4. THE CONTROLLER

The controller being designed is an offline controller with a fixed structure once designed and installed. Because the final approach is very short (8 to 20 seconds), no requirements of on-the-fly (online) self-adaptation are set forth.

The controller directly links the input signals, measured by onboard sensors, with control surface deflections:

$$\delta_i = f_i(u),$$

where δ_i is the ith control surface deflection and u is the vector of input signals.

The controller can be fed by 36 input signals, of which 12 come directly from the onboard flight sensors, 8 from the short-range radio positioning system, and the rest forms particularly valuable derivatives and combinations of the former parameters, such as, for example, no-wind climb rate $V_{yg} = V \sin \theta$ (see (Khantsis et al. 2005) for details). Although these combinations may emerge naturally during evolutionary search, pre-calculating them appeared to be highly effective.

On the most stages of the design, only two of the four control surfaces available are used as the main outputs for both longitudinal and lateral control. The elevator is used in longitudinal control together with a fixed throttle setting; and the ailerons are used to produce turns in conjunction with the rudder working as a simple damper. This allows reducing the dimensionality of the problem in half, as only two control laws are designed instead of four. At the later stages, the other two control laws may be included, allowing greater flexibility and higher degree of coupling.

5. EVOLUTIONARY DESIGN

The aim of the design methodology considered here is to provide the potential ability of automatic or semi-automatic *development* of the control laws. Not only numeric coefficients must be optimised, but also the whole structure of the control laws should be designed automatically, with as little a priori knowledge about the system as possible.

Evolutionary algorithms (EAs) (Holland 1975), (Bäck 1996), (Goldberg 1989) is a powerful and flexible tool that can handle many hard-to-solve problems, including those with large amount of uncertainty, noise, discontinuities, complex constraints. However, the great majority of the works in this area is focused on function optimisation problems. Nevertheless, EAs can be extended to the area of structure optimisation and program development, one of the most successful examples being Genetic Programming (GP) (Koza 1992).

Unfortunately, GP can not be applied directly to flight controller development. This area has several distinctive characteristics that make it harder to evolve a control law using a classic GP method. First, the typical control laws are continuous equations which usually require fine tuning of its numeric coefficients. This is a work for a conventional EA such as Genetic Algorithm (GA) or Evolutionary Strategies (ES), but not for GP. Second, classical GP approach with random initial population and tree-based crossover requires large population sizes to be effective. Meanwhile, each fitness evaluation requires at least one simulation run, which takes up to 1 second on a 2 GHz PC. Therefore, the number of fitness evaluations should be minimised as much as possible. In addition, large population sizes appear to be excessive for numeric coefficients optimisation.

To circumvent these problems and to make use of what is known about the aircraft control, an EA has been developed, which combine three main features:

- Automatic gradual complication of the task, set forth before the algorithm. This represents an attempt to introduce gradual ‘learning’ of the evolving controllers. First, they ‘learn to fly’ in a calm weather; when they can keep in the air for the normal duration of approach, the controllers are evaluated against a more complicated task: flying towards the target point along a straight line. On the third stage, the full set of requirements is applied. When a significant portion of the population succeeds the task, disturbances are gradually introduced. All the adjustments are made within the fitness evaluation function separately for each individual.

- Adaptive encoding of the control laws, which allows them to develop from the simplest possible law $y = const$ at start to complex non-linear equations. The encoding (see (Khantsis et al. 2005) resembles Polish notation, traditionally used in GP, but with implicit mathematical operations. This became possible because most of the physically sensible combinations of the sensor measurements were already included in the controller inputs (see Section 4), allowing significant simplification of the control law structure. In particular, each input (variable) is always accompanied by a coefficient and an added value (free term). Any variable, together with its two terms, can replace any term of any other variable, thus forming nested expressions. The genetic operator, affecting the structure (*structure mutation*) is implemented so that its effect is non-destructive, unlike classical tree-based crossover in GP. It does not have immediate effect, but rather adds a new dimension to the expression. This helps to preserve the information, gained so far by gradual learning.
- Separate evolution of the control laws structure and its numeric coefficients. As mentioned above, control laws require fine tuning of the coefficients. This is achieved by applying an ES algorithm with individual step size adaptation. Structure mutations occur at a slower rate, letting the current structures to be (roughly) numerically optimised.

The controller design process starts from configuration of the Simulink model of the UAV. Typically, disturbances and the winch model are disabled by default. The integration time step should be set to a maximum possible value that allows fairly precise simulation: calculation speed is the most important factor at this stage. At least 100,000 simulation runs should be allocated for full controller evolution. If no disturbances are involved, the time step up to 0.1 s and the 4th order Runge-Kutta integration algorithm showed good results. Turbulence models require reduction of the time step to about 0.02 s.

The evolution proceeds automatically when started. However, no termination criteria have been set apart from the number of generations. As shown in (Khantsis et al. 2005), the convergence behaviour may be significantly different even for the same initial conditions. For instance, Figure 1 illustrates the evolution of the current-best fitness across the population, 5 consecutive runs with the best found algorithm settings (no disturbances). It can be seen that the convergence time varies more than twice between the runs. More importantly, except for the final stage, it appears to be unnecessary to wait for full convergence: the next level of disturbances can be added after a reliable progress is observed.

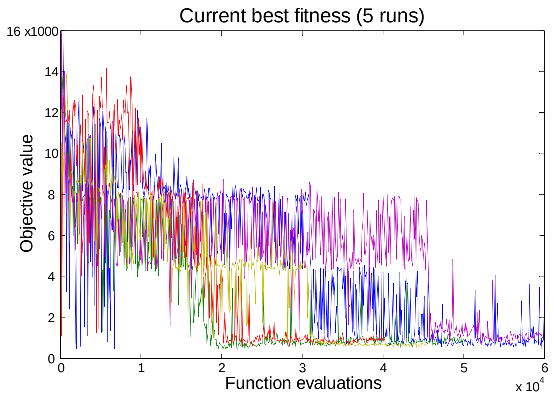

Figure 1. Convergence of the best fitness

In addition, the algorithm was considered rather as a research and engineering tool. As a result, termination and changing the environment settings was handled manually. The algorithm allows interrupt and resume the evolution at any point.

The principal set-up of the experiments, performed by the fitness evaluation function, is illustrated on Figure 2. For each evaluation, three simulations are executed from randomly chosen positions, with a bias to the left, right and down from the reference point (the 'ideal' point to which the mission controller is supposed to guide the UAV for recovery). This is done to estimate the ability of the controller to cope with different situations. The initial airspeed varies as well. The function, as noted before, takes into account both mission completion (the aircraft enters the target window with normal flight parameters) and the flight quality, and assigns a penalty score according to the current level of success of this particular individual.

When a reliable progress on the first stage of evolutionary design is achieved, disturbances are added one by one: turbulence, sensor noise, ship motion, and the process repeats. In this work, it was done manually at the discretion of the researcher, although this can be easily programmed.

The final population is then analysed. As a rule, about 10% of the population demonstrate comparable score and can be considered for verification. Although the EA discourages so called 'code bloat,' common for GP, the solutions are nearly always far from parsimony and are incomprehensible. Any attempt of manual tuning is usually worthless. Even the example of an intermediate controller (after about 65,000 objective function evaluations, 900 generations), shown on **Error! Reference source not found.**, is fairly complex. This is normal in evolutionary computation, see, for example, (Langdon and Poli 2002). Obviously, analytical analysis of such complex laws is not possible, therefore extensive testing is the only possibility to verify the controller.

Testing is performed on the same model. The integration step size should be set to obtain the optimal precision, in this research 0.01 s was used (0.002 s when the winch model was employed). Testings were done in separate series for calm weather, sea state 1, sea state 2 and so on up to the highest required level (6). The waves heights, wind speed and turbulence intensity are linked for a given series according to the UK Defence Standard 00-970 or MIL-STD-1797. In each series, statistics of successful recoveries is gathered. The controller may be considered successful, if the recovery success rate is not worse than a specified level for each sea state. Below are examples of control laws δ_e is the law for elevator, δ_a is the law for ailerons

$$\begin{aligned}\delta_a = {} & ((-2.6193\ (\psi-\psi_0) - 5.0003)\ (\psi-\psi_0) + 0.8304\ (d_1-d_2) + 1.5972\ n_z^2 \ldots \\ & + (1.534\ (1/d_2) + ((-0.90373\ n_z + 0.076235)\ \omega_x + 0.33578)\ V \ldots \\ & + 1.6151)\ \omega_y^2 + (-1.0412\ n_z + (2.8318\ n_z + 0.10118)\ (d_1-d_2) \ldots \\ & + 7.0475)\ \gamma + 3.5991\ \omega_x + 0.2673)\end{aligned}$$

$$\begin{aligned}\delta_e = {} & (-6.1752\ (1/d_2) + 1.3064\ (d_3-d_2) - 3.9956\ (1/V) + ((4.9783\ n_x \ldots \\ & + 3.6957)\ (1/d_2)^2 + 22.9016)\ \theta + 2.1812\ (1/V)^2 + ((((-0.36455\ (1/d_2) \ldots \\ & - 1.8637)\ V_{yg} + (-4.9151\ \theta + 0.092625)\ \omega_z + 3.8462)\ V \ldots \\ & + 0.24908)\ n_x - 0.79592\ (d_3-d_2) - 2.8818\ (1/d_2') + 1.4698)\ \omega_z \ldots \\ & + 9.117)\end{aligned}$$

The typical trajectories are presented on Figure 3.

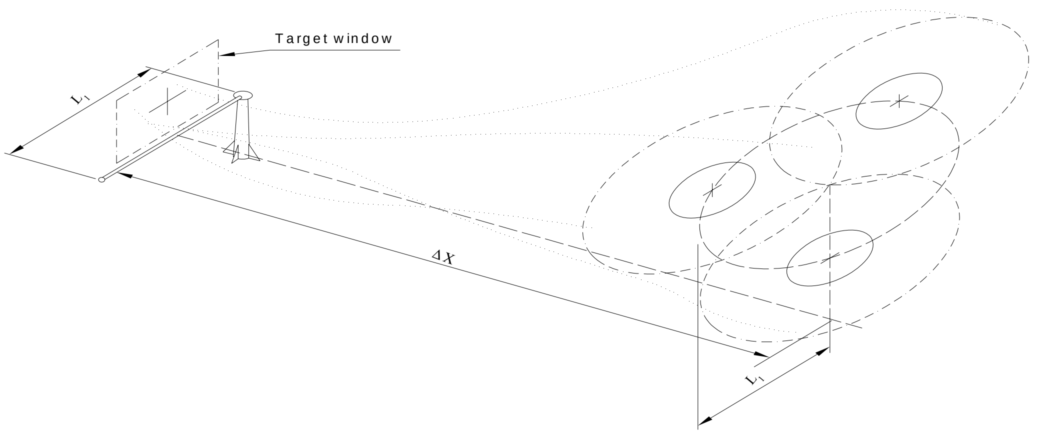

Figure 2. Experiment set-up (distance ΔX is not to scale). Three solid ellipses represent 1σ-, and three dash-dot ellipses – 3σ-lines of equal probability density of three independent starting position distributions. L_1 = 6 m. ΔX = 300 m on average and varies with σ = 20 m. Dotted lines represent possible successful trajectories.

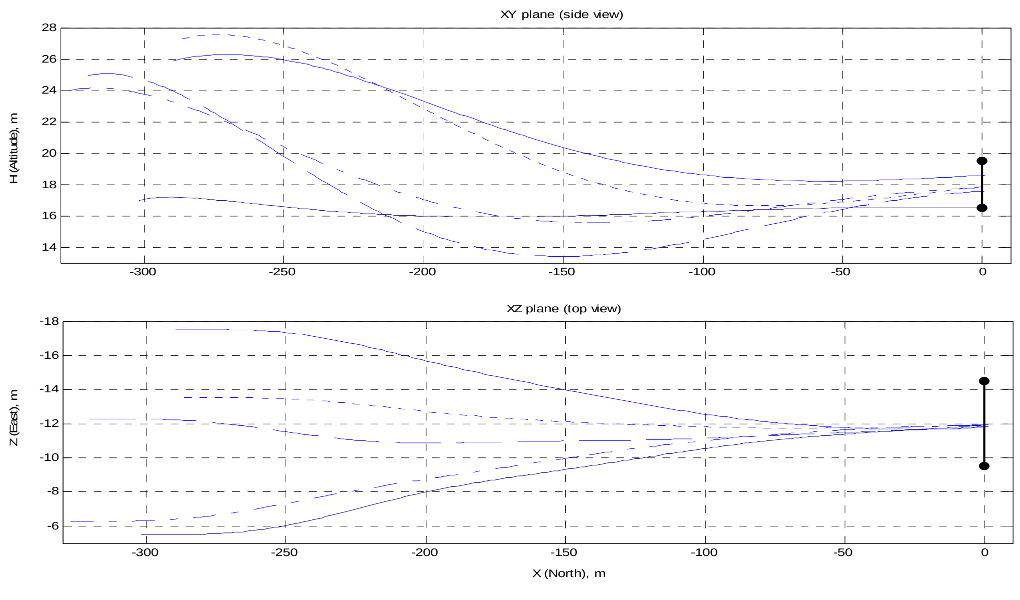

Figure 3. Flight paths (five independent runs). The bar at the finish (X = 0) illustrates the allowed error.

6. CONCLUSIONS

Evolutionary methods can be successfully used not only for function optimisations, but also for automatic development of such complex structures as aircraft control laws. Unlike traditional methods, evolutionary algorithms require little or no a priori knowledge about the controlled system. At the same time, they demonstrate very high robustness, handling non-linear systems in presence of various random disturbances.

This paper presents an integrated design environment which incorporates evolutionary algorithms as a main design tool. The approach is successfully applied to a practical task in the area of system control. As the considered system is complex and non-linear, the environment includes not only design part as such, but also the means to verify and validate the designed controller. It is demonstrated that such 'artificially intelligent' environment is capable for automatic or at least semi-automatic design process. At all stages of design, the system is evaluated by its actual performance. This does not require decomposition or any deep analysis of the system and thus allows applying design in out-of-the-box fashion.

7. REFERENCES

Bäck, T. (1996). *Evolutionary algorithms in theory and practice*, Oxford University Press.

Chipperfield, A., and Flemming, P. (1996). "Genetic algorithms in control systems engineering." *Journal of Computers and Control*, 24(1).

Crump, M. R. (2002). "The dynamics and control of catapult launching Unmanned Air Vehicles from moving platforms," Ph.D., RMIT.

Crump, M. R., Khantsis, S., Bil, C., and Bourmistrova, A. (2003). "Aspects of Launch and Recovery of Fixed-Wing Unmanned Air Vehicles (UAV) from Small Surface Vessels." *10th Australian International Aerospace Congress*, Brisbane.

Emmerich, M., Giotis, A., Özdemir, M., Bäck, T., and Giannakoglou, K. (2002). "Metamodel-Assisted Evolution Strategies." Parallel Problem Solving from Nature VII, J. J. Merelo Guervós and et al, eds., Springer-Verlag, Berlin Heidelberg, 361-370.

Fleet Oceanographic and Acoustic Reference Manual (1999), Naval Oceanographic Office, RP33, *Stennis SpaceCenter*.

Goldberg, D. E. (1989). *Genetic Algorithms in search, optimisation and machine learning*, Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

Hope, K. (1996). *Angulat Motions of an Australian FFG in Rough Seas SS6-7 - Statistics and Spectra*, Australian Defence Force Academy.

Khantsis, S., and Bourmistrova, A. (2005). "UAV controller design using evolutionary algorithms." *The 18th Australian Joint Conference on Artificial Intelligence*, 5-9 Dec., Sydney, Australia, to appear.

Koza, J. R. (1992). *Genetic Programming: On the programming of computers by means of natural selection*, MIT Press, Cambridge, Massachusetts.

Langdon, W. B., and Poli, R. (2002). *Foundations of genetic programming*, Springer-Verlag, Berlin.

Mahfoud, S. W., and Mani, G. (1996). "Financial forecasting using genetic algorithms." *Applied Artificial Intelligence*, 10, 543-565.

McFarland, R. E., and Duisenberg, K. (1995). "Simulation of rotor blade element turbulence." NASA.

Newman, D. M., and Wong, K. C. (1993). *Six Degree of Freedom Flight Dynamic and Performance Simulation of a Remotely-Piloted Vehicle*, The University of Sydney.

Olhofer, M., Jin, Y., and Sendhoff, B. (2001). "Adaptive encoding for aerodynamic shape optimization using Evolution Strategies." *Congress on Evolutionary Computation*, Seoul, South Korea, 576-583.

Sendhoff, B., and Kreuz, M. (1999). "Variable encoding of modular neural networks for time series prediction." *Congress on Evolutionary Computation*, 259-266.

Trivailo, P., Blanksby, C., and et al. (2002). "Development of advanced cable dynamic model based on Kane's dynamic equations." *TR ERC-DCTSG 2002-01*, RMIT University, Melbourne.