Flight Control System Design Optimisation via Genetic Programming

Anna Bourmistrova and Sergey Khantsis
Royal Melbourne Institute of Technology
Australia

1. Introduction

This chapter presents a methodology which is developed to design a controller that satisfies the objectives of shipboard recovery of a fixed-wing UAV. The methodology itself is comprehensive and should be readily applicable for different types of UAVs and various task objectives. With appropriate modification of control law representation, the methodology can be applied to a broad range of control problems. Development of the recovery controller for the UAV Ariel is a design example to support the methodology.
This chapter focuses on adaptation of Evolutionary Algorithms for aircraft control problems. It starts from analysis of typical control laws and control design techniques. Then, the structure of the UAV controller and the representation of the control laws suitable for evolutionary design are developed. This is followed by the development of the general evolutionary design algorithm, which is then applied to the UAV recovery problem. Finally the presented results demonstrate robust properties of the developed control system.

2. Aircraft flight control

2.1 Overview of types of feedback control

Not unlike the generic control approach, aircraft flight control is built around a feedback concept. Its basic scheme is shown in Fig. 1. The controller is fed by the difference between the commanded reference signal *r* and the system output *y*. It generates the system control inputs *u* according to one or another algorithm.

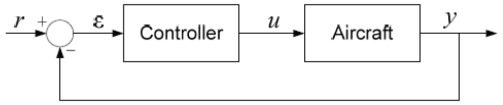

Figure 1. Feedback system (one degree of freedom)

One of the main tasks of flight control as an engineering discipline is design of the controllers which enable a given aircraft to complete a defined mission in the most optimal manner, where optimality is based on mission objective. A number of techniques of producing a required controller have been developed over the last hundred years since the first altitude hold autopilot was introduced by Sperry Gyroscope Company in 1912. Most of the techniques make certain assumptions about the controlled system (i.e. the aircraft), most notably linearity of its dynamics and rigidity of the airframe, to simplify synthesis of the controller. A brief overview of types of feedback control is presented below.

2.1.1 On-Off control

The simplest feedback control is the *on-off control*, also referred to among engineers as *bang-bang control*. This control law can be expressed as follows:

$$u = u_0 + \begin{cases} k, z < a \\ 0, z > a \end{cases} \tag{1}$$

where u_0 and k are arbitrary offset and gain suitable for the given system, and a is the threshold (set point). It can be extended to a multiple choice situation.

This law is not particularly suitable for direct flight control in normal situations. Indeed, with only two (or several) discrete control input levels, the system output will tend to oscillate about the set point, no matter how well damped the system is, because the control signal will not switch until the set point is already passed. Moreover, if the dynamics of the system is fast or a significant amount of noise in the system output is present, the controller will be switching also fast ('hunting'), possibly causing extensive wear to the control actuators.

To prevent such behaviour, a special 'dead zone' (or 'deadband') is established around the set point where no switching occurs. Obviously, this reduces the accuracy of control. However, the on-off control comes into play when event handling is required. A classic example of application of such control for aircraft is stall recovery. When angle of attack exceeds a specified value, a nose-down elevator command is issued. A similar logic can be implemented for overload protection or ground collision avoidance.

Another important area in the aerospace field where on-off rules can be successfully applied is internal switching between the controllers (or controller parameters). This approach is known as Gain scheduling (Rugh, 2000). The technique takes advantage of a set of relatively simple controllers optimised for different points of the flight envelope (or other conditions). However, it was found that rapid switching may cause stability and robustness problems (Shamma & Athans, 1992). One of the popular simple solutions is to 'blend' (interpolate) the output of two or more controllers, which effectively turns simple on-off switching into a complicated control method.

2.1.2 PID control

The proportional-integral-derivative (PID) control is probably the most widely used type of control, thanks to the simplicity of its formulation and in most cases, predictable characteristics. In a closed-loop system like that in Fig. 1, its control law is

$$u = K_P \varepsilon + K_I \int \varepsilon dt + K_D \dot{\varepsilon} \tag{2}$$

where the parameters K_P, K_I and K_D are coefficients for proportional, integral and derivative components of the input error signal respectively. By adjusting these three parameters, the desired closed-loop dynamics can be obtained.

Probably the most problematic issue with the PID control is due to the differential term. While being important for good response time and high-speed dynamics, the differential component keenly suffers from both instrumental noise and calculation errors. Indeed, even a small amount of noise can greatly affect the slope of the input signal. At the same time, numerical calculation of the derivative (for a digital controller in particular) must be done with a fairly small time step to obtain correct slope at a given point. Although lowpass filtering applied to the input signal smoothens the signal, it severely compromises the usefulness of the derivative term because the low-pass filter and derivative control effectively cancel each other out.

In contrast, the integral term averages its input, which tends to eliminate noise. However, a common problem associated with the integral control owes exactly to its 'memory'. When a large input value persists over a significant amount of time, the integral term becomes also large and remains large even after the input diminishes. This causes a significant overshoot to the opposite values and the process continues.

In general, integral control has a negative impact on stability and care must be taken when adjusting the integral coefficient. Limiting the integrator state is a common aid for this problem. A more elegant approach involves 'washing out' the integrator state by incorporating a simple gain feedback, effectively implementing a low-pass filter to the input signal. PID control found wide application in the aerospace field, especially where near-linear behaviour takes place, for example, in various hold and tracking autopilots such as attitude hold and flight path following. The techniques of selecting the optimal PID coefficients are well established and widely accepted. Typically, they use the frequency domain as a design region and utilise phase margins and gain margins to illustrate robustness. However, design of a PID controller for nonlinear or multi-input multi-output (MIMO) systems where significant coupling between the different system inputs and outputs exists is complicated. As the aircraft control objectives evolved, new design and control techniques were developing. Although many of them essentially represent an elaborated version of PID control, they are outlined in the following sections.

2.1.3 Linear optimal control

The concept of optimality in mathematics refers to minimisation of a certain problem dependent cost functional:

$$J = \int_0^T g(t, x, u)\,dt + h(x(T)) \tag{3}$$

where T is the final time, u is the control inputs and x is the state of the controlled system. The last term represents the final cost that depends on the state in which the system ends up. Optimal control is, therefore, finding the control law $u(t)$ that minimises J. In general, for an arbitrary system and cost functional, only numerical search can find the optimal (or near optimal) solution.

There is a number of types of linear optimal control, such as the *Linear Quadratic Regulator (LQR),* the *Linear Quadratic Gaussian with Loop Transfer Recovery (LQG/LTR)* and the extended Kalman filter (EKF). The theory and application of these control techniques and Kalman filtering is detailed in many common control and signal processing textbooks, such as for example (Brown & Hwang, 1992; Bryson & Ho, 1975; Kalman, 1960; Maciejowski, 1989).

The Kalman filter relies on the model of the system (in its predicting part), and the guaranteed performance of the controller can easily be lost when unmodelled dynamics, disturbances and measurement noise are introduced. Indeed, robustness is a challenging issue of modern control designs. Also, the well known in classic linear design phase and gain margin concepts cannot be easily applied to the multivariable systems that modern control is so suited for. These problems led to the introduction of robust modern control theory.

2.1.4 Robust modern control

Unlike traditional optimal control, robust optimal control minimises the influence of various types of uncertainties in addition to (or even instead of) performance and control energy optimisation. Generally, this implies design of a possibly low gain controller with reduced sensitivity to input changes. As a consequence, robust controllers often tend to be conservative and slow. On the other hand, they may be thought as the stabilising controllers for the whole set of plants (which include a range of uncertainties) and not only for the modelled system, which is a more demanding task.

The majority of modern robust control techniques have origins in the classical frequency domain methods. The key modification of the classic methods is shifting from eigenvalues to singular values (of the transfer function that describes the system), the singular value Bode plot being the major indicator of multivariable feedback system performance (Doyle & Stein, 1981).

Probably the most popular modern robust control design techniques (particularly in the aerospace field) are H_2 and H_∞ control, also known as the frequency-weighted LQG synthesis and the small gain problem respectively. These techniques and the underlying theories are thoroughly described in several works, notably (Doyle at al., 1989; Kwakernaak, 1993; Zhou & Doyle, 1998).

The H_∞ control design found extensive use for aircraft flight control. One of the first such applications was the development of controllers for the longitudinal control of a Harrier jump jet (Hyde, 1991). This work has been subsequently extended in (Postlethwaite & Bates, 1999) to fully integrated longitudinal, lateral and propulsive control. Other works include (Kaminer et al., 1990), where a lateral autopilot for a large civil aircraft is designed, and (Khammash & Zou, 1999), where a robust longitudinal controller subject to aircraft weight and c.g. uncertainty is demonstrated.

A mixed H_2 / H_∞ approach is applied in (Shue & Agarwal, 1999) to design an autoland controller for a large commercial aircraft. The method employed here utilises the H_2 controller for slow trajectory tracking and the H_∞ controller for fast dynamic robustness and disturbance rejection.

Several H_∞ controllers have been tried to accomplish the UAV shipboard launch task (Crump, 2002). It has been found that these controllers perform quite well in nominal situations. However, in the presence of large disturbances which place the aircraft well beyond its linear design operating point, the controllers performed poorly (sometimes extremely). At the same time, inability to include even simple static nonlinearities such as time delays and saturations made it difficult to synthesise a practical controller for this task within linear approach. Another deficiency found is common to all frequency domain techniques: the frequency domain performance specifications cannot be rigidly translated into time and spatial domain specifications. Meanwhile, time and especially spatial (trajectory) constrains are crucial for both launch and recovery tasks.

The time domain performance can be accounted for directly in the time domain l_1 design (Blanchini & Sznaier, 1994; Dahleh & Pearson, 1987).

It is often extended to include the frequency domain objectives, resulting in a mixed norm approach. However, l_1 design is plagued by the excessive order of the generated controllers. This is usually solved by reducing the problem to suboptimal control, imposing several restrictions on the system and performance specifications (Sznaier & Holmes, 1996). The application of l_1 approach to flight control is discussed in (Skogestad & Postlewaite, 1997), with the conclusion that controllers with excessive order will generally be produced when using practical constraints.
There have been attempts to solve the H_∞ optimal control problems for nonlinear systems. However, these methods usually rely on very limiting assumptions about the model, uncertainties and disturbance structure. The mathematical development of nonlinear H_∞ control can be found in (Dalsamo & Egeland, 1995). An example of nonlinear control of an agile missile is given in (Wise, 1996).
Despite a very complicated solution algorithm, this work is limited by a linear assumption on the vehicle aerodynamics, reducing the benefits gained from the use of nonlinear control.

2.1.5 Nonlinear control

Linear control design techniques have been used for flight control problems for many years. One of the reasons why aircraft can be controlled quite well by linear controllers is that they behave almost linearly through most of their flight envelope. However, when the aircraft is required to pass through a highly nonlinear dynamic region or when other complicated control objectives are set, it has been found by several researchers that it is difficult to obtain practical controllers based on linear design techniques. The UAV shipboard launch and recovery tasks are substantially nonlinear problems. The sources of nonlinearities are the aerodynamic forces generated at low airspeeds and high angles of attack (especially when wind disturbances are present); trajectory constraints imposed due to proximity of ground (water) and ship installations; kinematic nonlinearities when active manoeuvring is required; actuator saturations and some more.
In contrast to the linear systems, the characteristics of the nonlinear systems are not simply classified and there are no general methods comparable in power to those of linear analysis. Nonlinear techniques are quite often designed for individual cases, regularly with no mathematical justification and no clear idea of re-applicability of the methods.
Some of the more popular nonlinear control techniques are covered in textbooks (Atherton, 1975; Graham & McRuler, 1971).
The most basic nonlinear control laws and the On-off control and Gain scheduling, noting that these controllers often lack robustness when the controllers are scheduled rapidly.
Another modern control technique remotely related to the on-off control is *variable structure* (also known as *sliding mode*) control (DeCarlo et al., 1988; Utkin, 1978). In this approach, a hypersurface (in state space) called *sliding surface* or *switching surface* is selected so that the system trajectory exhibits desirable behaviour when confined to this hypersurface. Depending on whether the current state is above or below the sliding surface, a different control gain is applied. Unlike gain scheduling, the method involves high speed switching to keep the system on the sliding surface.
A completely different approach is to enable applicability of the well known linear control methods to control nonlinear systems. This can be achieved using *nonlinear dynamic inversion*. This process, also known as *feedback linearisation*, involves online approximate linearisation of a nonlinear plant via feedback. Dynamic inversion gained particular

attention in aviation industry in the late 1980s and 90s, aiming to control high performance fighters during high angle of attack manoeuvres (known as *supermanoeuvres*). One of the early applications is NASA High Angle of Attack Vehicle (HARV) (Bugajski & Enns, 1992). In this work, quite good simulation performance results are obtained; however, with the inversion based on the same simulation model, any possible discrepancies are transferred into the controller, leading to questionable results in physical implementation with respect to incorrectly modelled dynamics.

2.1.6 Intelligent control

Intelligent control is a general and somewhat bold term that describes a diverse collection of relatively novel and non-traditional control techniques based on the so called *soft computing* approach. These include neural networks, fuzzy logic, adaptive control, genetic algorithms and several others. Often they are combined with each other as well as with more traditional methods; for example, fuzzy logic controller parameters being optimised using genetic algorithms or a neural network driving a traditional linear controller.

Neural network (NN), very basically, is a network of simple nonlinear processing elements (neurons) which can exhibit complex global behaviour determined by element parameters and the connections between the processing elements. The use of artificial neural networks for control problems receives an increased attention over the last two decades. It has been shown that a certain class of NN can approximate any continuous nonlinear function with any desired accuracy (Spooner, 2002). This property allows to employ NN for *system identification* purposes, which can be performed both offline and online. This approach is used in (Coley, 1998; Kim & Calise, 1997) for flight control of a tilt-rotor aircraft and the F-18 fighter.

Another area of intensive application of NN is fault tolerant control, made possible due to online adaptation capability of NN. In the recent work (Pashilkar et al., 2006), an 'add-on' neural controller is developed to increase auto-landing capabilities of a damaged aircraft with one of two stuck control surfaces. A significant increase in successful landings rate is shown, especially when the control surfaces are stuck at large deflections.

Fuzzy logic control also gained some popularity among flight control engineers. This type of control relies on approximate reasoning based on a set of rules where intermediate positions between 'false' and 'truth' are possible. Fuzzy logic control may be especially useful when a decision must be made between several controversial conditions. For example, in (Fernandez-Montesinos, 1999) fuzzy logic is used for windshear recovery in order to decide whether energy should be given to altitude or airspeed, based upon the current situation.

Adaptive control term covers a set of various control techniques that are capable of online adaptation. A good survey of adaptive control methods is given in (Astrom & Wittenmark, 1995). The applications of adaptive control is generally biased towards control for large time scales so that the controller has sufficient time to learn how to behave. This makes the relatively short-time recovery process unsuitable for online adaptation.

Evolutionary and *genetic algorithms* (EAs, GAs) are global optimisation techniques applicable to a broad area of engineering problems. They can be used to optimise the parameters of various control systems, from simple PID controllers (Zein-Sabatto & Zheng, 1997) to fuzzy logic and neural network driven controllers (Bourmistrova, 2001; Kaise & Fujimoto, 1999). Another common design approach is evolutionary optimisation of trajectories, accompanied by a suitable tracking controller (e.g. (Wang & Zalzala, 1996)). An elaborated study of applications of EAs to control and system identification problems can be found in (Uzrem, 2003).

Unlike the majority of other techniques, Genetic Algorithms (in the form of *Genetic Programming*) are able to evolve not only the parameters, but also the *structure* of the controller.
In general, EAs require substantial computational power and thus are more suitable for offline optimisation. However, online evolutionary-based controllers have also been successfully designed and used. The *model predictive control* is typically employed for this purpose, where the controller constantly evolves (or refines) control laws using an integrated simulation model of the controlled system. A comprehensive description of this approach is given in (Onnen et al., 1997).

2.2 Flight control for the UAV recovery task

Aircraft control at recovery stage of flight can be conventionally separated into two closely related, but distinctive tasks: guidance and flight control. Guidance is the high-level ('outer loop') control intended to accomplish a defined mission. This may be path following, target tracking or various navigation tasks. Flight control is aimed at providing the most suitable conditions for guidance by maintaining a range of flight parameters at their optimal levels and delivering the best possible handling characteristics.

2.2.1 Traditional landing

In a typical landing procedure, the aircraft follows a defined glide path. The current position of the aircraft with respect to the glidepath is measured in a variety of ways, ranging from pilot's eyesight to automatic radio equipment such as the Instrument Landing System (ILS). Basically, the objective of the pilot (or autopilot) is to keep the aircraft on the glidepath, removing any positioning error caused by disturbances and aircraft's dynamics. This stage of landing is known as *approach*. Approach may be divided into *initial approach*, in which the aircraft makes contact with ('fixes') the approach navigation system (or just makes visual contact with the runway) and aligns with the runway; and *final approach*, when the aircraft descends along a (usually) straight line. In conventional landing, final approach is followed by a flare manoeuvre or nose-up input to soften the touchdown; however, flare is typically not performed for shipboard landing due to random ship motion and severe constraints on the landing space.
The guidance task during final approach involves trajectory tracking with both horizontal and vertical errors (and their rates) readily available. It is important to note that the errors being physically measured are *angular* deviations of the aircraft position (Fig.2) as seen from the designated touchdown point (or more precisely, from where the radars or antennas or other guidance systems are located). They can be converted to linear errors Δh if the *distance* L to the aircraft is known.

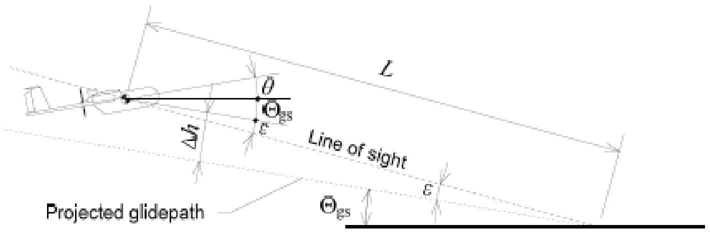

Figure 2. Angular and linear glidepath errors

However, in many applications precise distance measurement is unavailable. Nevertheless, successful landing can be carried out even without continuous distance information. This comes from the fact that exactly the angular errors are relevant to precise landing. Indeed, the goal is to bring the aircraft to a specified point on the runway (or on the deck) in a certain state. The choice of the landing trajectory only serves this purpose, considering also possible limitations and secondary objectives such as avoiding terrain obstacles, minimising noise level and fuel consumption and so on. The angular errors provide an adequate measurement of the current aircraft position with respect to the glidepath.
However, if the landing guidance system takes no account of distance and is built around the angular error only, it may cause stability problems at close distances, because the increasing sensitivity of the angular errors to any linear displacement effectively amplifies the system gain.

2.2.2 UAV control for shipboard recovery

As it was seen from the discussion in the previous section, landing of an aircraft is a well established procedure which involves following a predefined flight path. More often than not, this is a rectilinear trajectory on which the aircraft can be stabilised, and the control interventions are needed only to compensate disturbances and other sources of errors.
The position errors with respect to the ideal glidepath can be measured relatively easily. Shipboard landing on air carriers is principally similar; the main differences are much tighter error tolerances and absence of flare manoeuvres before touchdown. The periodic ship motion does have an effect on touchdown; however, it does not affect significantly the glidepath, which is projected assuming the average deck position. The choice of the landing deck size, glideslope, aircraft sink rate and other parameters is made to account for any actual deck position at the moment of touchdown. For example, a steeper glideslope (typically 4°) is used to provide a safe altitude clearance at the deck ramp for its worst possible position (i.e. ship pitched nose down and heaved up). This makes unnecessary to correct the ideal reference trajectory on the fly.
For the UAV shipboard recovery, ship oscillations in high sea cause periodic displacement of the recovery window (the area where capture can be done) several times greater than the size of the window itself. This fact (and also the assumption that the final recovery window position cannot be predicted for a sufficient time ahead) makes it impossible to project an optimal flight path when the final approach starts. Instead, the UAV must constantly track the actual position of the window and approach so that the final miss is minimised. Therefore, it turns out that the UAV recovery problem resembles that of homing guidance rather than typical landing. While stabilisation on a known steady flight path can be done relatively easy with a PID controller, homing guidance to a moving target often requires a more sophisticated control.
Not surprisingly, homing guidance found particularly wide application in ballistic missiles development, hence the accepted terminology owes to this engineering area. In the context of UAV recovery, the UAV moves almost straight towards the ‘target’ from the beginning (compared to typical missile intercept scenarios) and thus the velocity vector and the line of sight almost coincide.
However, there are two major difficulties that can compromise the effectiveness of Proportional Navigation (PN) for UAV recovery. First, PN laws are known as generating excessive acceleration demands near the target. For a UAV with limited manoeuvrability,

such demands may be prohibitive, especially at low approach airspeeds. On the other hand, the PN guidance strategy does not change during the flight. Several alternative guidance strategies with more favourable acceleration demands exist, e.g. augmented proportional navigation. However, it is unlikely they can sufficiently improve the guidance to an oscillating target such as ship's deck.

2.3 UAV controller structure

The objective is therefore to synthesise such guidance strategy that enables reliable UAV recovery, and to produce a controller that implements the target tracking guidance strategy.
The evolutionary design (ED) method applied for this task allows to evolve automatically both the structure and the parameters of the control laws, thus potentially enabling to generate a 'full' controller, which links available measurements directly with the aircraft control inputs (throttle, ailerons, rudder and elevator) and implements both the guidance strategy and flight control (Fig. 3):

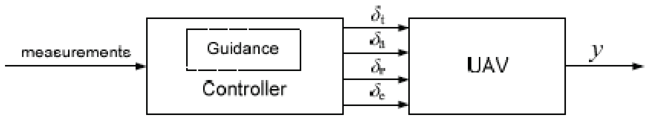

Figure 3. Full controller with embedded guidance strategy

However, this approach, even though appealing at first and requiring minimum initial knowledge, proves to be impractical as the computational demands of the evolutionary algorithms (EAs) soar exponentially with the dimensionality of the problem. It is therefore desirable to reduce complexity of the problem by reducing the number of inputs/outputs and limiting, if appropriate, possible structures of the controllers.
Another difficulty is the evaluation of the controller's performance. In ED, performance or fitness is multiobjective. Therefore, it is highly desirable to decompose this complex task into several simpler problems and to solve them separately.
A natural way of such decomposition is separating the trajectory control (guidance) and flight control. The guidance controller issues commands u_g to the flight controller, which executes these commands by manipulating the control surfaces of the UAV (Fig. 4). These two controllers can be synthesised separately using appropriate fitness evaluation for each case.

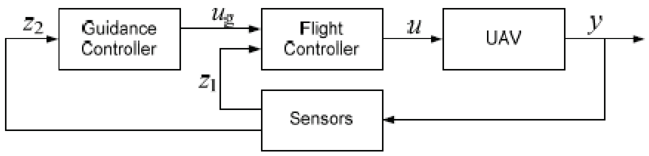

Figure 4. UAV recovery control diagram

2.3.1 Guidance controller

The internal structure of the controller is defined by the automatic evolutionary design based on predefined set of inputs and outputs. It is desirable to keep the number of inputs and outputs to minimum, but without considerably compromising potential performance.

An additional requirement to the outputs u_g is that these signals should be straightforwardly executable by the flight controller. This means that u_g should represent a group of measurable flight parameters such as body accelerations, velocities and Euler angles, which the flight controller can easily track.
The structure of the output part of the guidance controller is as shown in Fig. 5. With this scheme, the guidance laws produce general requests to change trajectory in horizontal and vertical planes. The kinematic converter then recalculates these requests to the form convenient for the flight controller. Both the bank angle and normal body load factor n_y can be relatively easily tracked, with the sensors providing direct measurements of their actual values. At the same time, this approach allows to evolve the horizontal and vertical guidance laws separately, which may be desirable due to different dynamics of the UAV's longitudinal and lateral motion and also due to computational limitations.
Input measurements to the guidance controller should be those relevant to trajectory. First of all, this is all available positioning information, pitch and yaw angles and airspeed. They do not account for steady wind, but still provide substantial information regarding the current 'shape' of trajectory.
The yaw angle fed into the controllers is corrected by the 'reference' yaw ψ_0, which is perpendicular to the arresting wire in the direction of anticipated approach. A zero yaw indicates that the UAV is pointed perpendicularly to the arresting wire (ignoring ship oscillations), which is the ideal condition in the absence of side wind and when the UAV moves along the ideal glidepath. This is similar to rotating the ground reference frame $O_g x_g y_g z_g$ by the correction yaw angle ψ_0. The rotated frame is referred as *approach ground reference frame*.

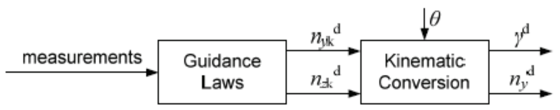

Figure 5. Guidance controller

The derived quantities from raw measurements are the vertical and lateral velocity components with respect to the approach ground reference frame.
Determination of the current UAV position and velocities with respect to the arresting wire is crucial for successful recovery. While previously discussed flight parameters may only help to improve the guidance quality, positioning carries direct responsibility for recovery.

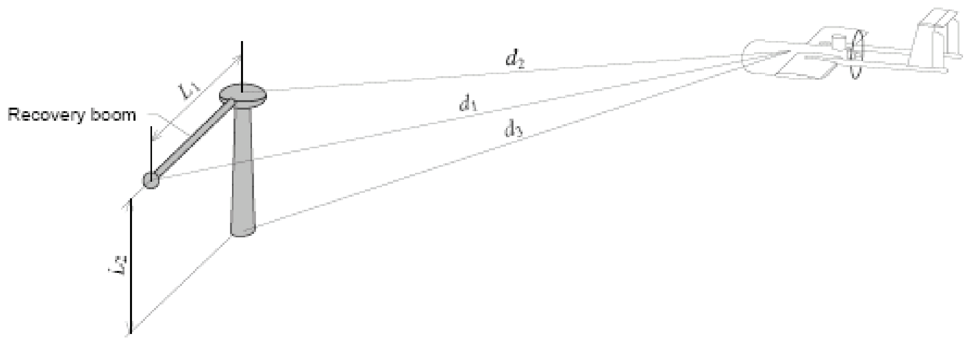

Figure 6. Positioning scheme

The system is based on radio distance metering and provides ten independent raw measurements (Fig. 6): three distances d_1, d_2 and d_3 from the UAV to the radio transmitters located at both ends of the recovery boom which supports the arresting wire and at the base of recovery mast; three rates of change of these distances; distance differences ($d_1 - d_2$) and ($d_3 - d_2$); and rates of change of the differences.

The guidance laws evolution process is potentially capable to produce the laws directly from raw measurements, automatically finding necessary relationships between the provided data and the required output.

The target spot elevation is chosen to be a constant, and the value is determined in view of the expected cable sag obtained from simulation of the cable model. For normal approach speed, the cable sag varies between 3.5 and 4.5 m. Accordingly, the target spot elevation is chosen to be $h_T = 2$ m above the arresting wire.

The recovery procedure, if successful, lasts until the cable hook captures the arresting wire. This happens when the UAV have moved about the full length of the cable *past* the recovery boom. However, position measurements may be unavailable beyond the boom threshold, and even shortly before the crossing the readings may become unreliable. For these reasons, the terminal phase of approach, from the distance about 6–10 m until the capture (or detection of a miss), should be handled separately.

It is possible to disconnect the guidance controller several metres before the recovery boom without affecting the quality of guidance. The allowed error (approximately 2 m in all directions, determined by the lengths of the arresting wire and the cable) should absorb the absence of controlled guidance in the last 0.3 to 1 second (depending on the headwind) in most situations.

2.3.2 Flight controller

Flight controller receives two inputs from the guidance controller: bank angle demand γ^d and normal body load factor demand n_y^d. It should track these inputs as precisely as possible by manipulating four aircraft controls: throttle, ailerons, rudder and elevator.

The available measurements from the onboard sensors are body angular rates ω_x, ω_y, ω_z from rate gyros, Euler angles γ, ψ, θ from strapdown INS, body accelerations n_x, n_y, n_z from the respective accelerometers, airspeed V_a, aerial angles α and β, actual deflection of the control surfaces δ_a, δ_r, δ_e, and engine rotation speed N_{rpm}.

For simplicity of design, the controller is subdivided into independent longitudinal and lateral components. In longitudinal branch, elevator tracks the input signal n_y^d, while throttle is responsible for maintaining a required airspeed. In lateral control, naturally, ailerons track γ^d, while rudder minimises sideforce by keeping n_z near zero.

3. Evolutionary design

The Evolutionary Design (ED) presented in this section, generally, takes no assumptions regarding the system and thus can be used for wide variety of problems, including nonlinear systems with unknown structure. In many parts, this is a novel technique, and the application of ED to the UAV guidance and control problems demonstrates the potential of this design method.

The core of evolutionary design is a specially tailored evolutionary algorithm (EA) which evolves both the structure and parameters of the control laws.
Since the algorithm is used for *creative* work only at the *design* stage, its performance is rather of secondary importance as long as the calculations take a sensible amount of time. The major requirements to automatic design methods are *quality* of the result and *exploration* abilities.
Although the basic framework of an EA is quite simple, there are three key elements that must be prepared before the algorithm can work. They are:

- representation of phenotype (control laws in our case) suitable for genetic operations (genome encoding);
- simulation environment, which enables to implement the control laws within the closed loop system;
- fitness evaluation function, which assesses the performance of given control laws.

These elements, as well as the whole algorithm outline, are addressed below.
Parallel evolution of both the structure and the parameters of a controller can be implemented in a variety of ways. One of the few successfully employed variants is the *block structure controller evolution* (Koza et al., 2000).
In this work the ED algorithm enables to evolve suitable control laws within a reasonable time by utilising gradual evolution with the principle of strong casualty. This means that structure alterations are performed so that the information gained so far in the structure of the control law is preserved. Addition of a new block, though being random, does not cause disruption to the structure. Instead, it adds a new dimension and new potential which may evolve later during numerical optimisation. The principle of strong casualty is often regarded as an important property for the success of continuous evolution (Sendhoff et al., 1997).
The addition of new points or blocks is carried out as a separate dedicated operation (unlike sporadic structure alterations in the sub-tree crossover), and is termed *structure mutation*. Furthermore, in this work structure mutation is performed in a way known as *neutral structure mutation*. That's when the new block should be placed initially with zero coefficient. This will not produce any immediate improvement and may even deteriorate the result slightly because more blocks are used for the same approximation. However, further numerical optimisation should fairly quickly arrive at a better solution. The usefulness of neutral mutations has been demonstrated for the evolution of digital circuits (Van Laarhoven & Aarts, 1987) and aerodynamic shapes (Olhofer et al., 2001).
As a result, the ED algorithm basically represents a numerical EA with the inclusion of structure mutations mechanism.

3.1 Representation of the control laws

Control laws are represented as a combination of static functions and input signals, which are organised as a dynamic structure of *state equations* and *output equations* in form of continuous representation.
The controller being evolved has *m* inputs, *r* outputs and *n* states. The number of inputs and outputs is fixed. The algorithm allows varying number of states; however, in this work, the number of states is also fixed during the evolution. As a result, the controller comprises of *n* state equations and *r* output equations:

$$\begin{array}{ccc} \dot{x}_1 = g_1(x,u) & & y_1 = f_1(x,u) \\ \dot{x}_2 = g_2(x,u) & \text{and} & y_2 = f_2(x,u) \\ \vdots & & \vdots \\ \dot{x}_n = g_n(x,u) & & y_n = f_n(x,u) \end{array} \tag{4}$$

where $\boldsymbol{u}$ is size m vector of input signals, $\boldsymbol{x} = [x_1,x_2,\ldots x_n]$ is size n vector of state variables, $y_{1\ldots r}$ are controller outputs. Initial value of all state variables is zero. All $n+r$ equations are built on the same principle and are evolved simultaneously. For structure mutations, a random equation is selected from this pool and mutated.

3.1.1 Representation of input signals

Input signals delivered to each particular controller are directly measured signals as well as the quantities derived from them.
Within each group, inputs are organised in the subgroups of 'compatible' parameters. Compatible parameters are those which have close relationship with each other, have the same dimensions and similarly scaled. The examples of compatible parameters are the pairs (n_x, n_{xg}), (ω_y, $\dot{\psi}$), (V_a, V_{CL}). As a rule, only one of the compatible parameters is needed in a given control law. For this reason, the probability of selection of such parameters for the structure mutation is reduced by grouping them in the subgroups. Each subgroup receives equal chances to be selected. If the selected subgroup consists of more than one input, a single input is then selected with uniform probability.
Therefore, every controller input may be represented by a unique *code* consisting of three indices: the number of group *a*, the number of subgroup *b* and the number of item in the subgroup *c*. The code is designated as u(a,b,c).

3.1.2 Representation of control equations and the structure mutation

Each of the control equations (4) is encoded as described above. To this end, only one single output equation of the form $y = f(\boldsymbol{u})$ will be considered in this section. State variables $\boldsymbol{x}$ are considered as special inputs and have no effect on the encoding.
This is done to speed up the search, which could otherwise be hampered by the multitude of possible operations. It proved to be more effective to include all meaningful quantities derived from source measurements as independent inputs than to implement all the functions and operations which potentially allow to emerge all necessary quantities automatically in the course of evolution.
The encoding should allow a simple way to insert a new parameter in any place of the equation without disrupting its validity and in a way that this insertion initially does not affect the result, thus allowing neutral structure mutations.
Conceptually, the equation is a sum of input signals, in which:

- every input is multiplied by a numeric coefficient or another similarly constructed expression;
- the product of the input and its coefficient (whether numeric or expression) is raised to the power assigned to the input;
- a free (absolute) term is present.

The simplest possible expression is a constant:

$$y = k_0 \tag{5}$$

A linear combination of inputs plus a free term is also a valid expression:

$$y = k_2u_2 + k_1u_1 + k_0 \tag{6}$$

Any numeric constant can be replaced with another expression. An example of a full featured equation is

$$y = ((k_4u_4 + k_3)u_3)^{-0.5} + k_2u_2 + (k_1u_1)^2 + k_0 \tag{7}$$

This algorithm can be illustrated by encoding the example (7). The respective internal representation of this expression is:

- Equation: $y = ((k_4u_4 + k_3)u_3)^{-0.5} + k_2u_2 + (k_1u_1)^2 + k_0$
- Expression: { u(3,-0.5) u(4) 1 2 *u*(2) 3 u(1,2) 4 5 }
- Object parameters: [k_4 k_3 k_2 k_1 k_0]
- Strategy parameters: [s_4 s_3 s_2 s_1 s_0]

This syntax somewhat resembles Polish notation with implicit '+' and '*' operators before each variable. The representation ensures presence of a free term in any sub-expression, such as k_3 and k_0 in the example above.

The algorithm of structure mutation is presented below.

1. Select an input (or a state variable) at random: u(a,b,c).
2. Obtain the initial values of the numeric coefficient and the strategy parameter (initial step size). The initial coefficient k is selected as described above, it can be either 0 or 10^6.
3. Append the object parameters vector with the initial coefficient, and the strategy parameters vector with the initial step size.
4. Form a sub-expression consisting of the selected variable code and the obtained index: { u(a,b,c) n }.
5. With 40% probability, set the insertion point (locus) to 1; otherwise, select a numeric value in the expression at random with equal probability among all numeric values present and set the locus to the index of this value.
6. Insert the sub-expression into the original expression at the chosen locus (*before* the item pointed).

This procedure may produce redundant expressions when the selected variable already exists at the same level. Thus an algebraic simplification procedure is implemented. It parses given expression, recursively collects the factors of each variable encountered and then rebuilds the expression.

3.2 Simulation environment

Fitness evaluation of the controllers is based on the outcome of one or more simulation runs of the models involved with the controller being assessed in the loop.

Simulation environment for this study is constructed in the MATLAB/Simulink software. All the models are implemented as Simulink library blocks (Khantsis, 2006). The models participating in the evolution include:

- The *Ariel* UAV model;
- The atmospheric model;
- The ship model;
- The static cable model.

The sample rate should be kept at minimum, ensuring, however, numerical stability. The fastest and thus the most demanding dynamics in the model is contained in

- sensors (both the lags and noise);
- control actuators;
- turbulence model;
- and potentially, controllers.

These components should be especially carefully examined when adjusting the sample rate. The experimentally determined minimum sample rate for the model is 100 Hz. The model is integrated using the 4th order Runge-Kutta method.

3.3 Fitness evaluation

Fitness evaluation is based on objectives which are individual for each controller and therefore is calculated individually. As discussed above having several specific controllers makes it easier to define their particular objectives and reduces the number of these objectives as compared to one all-purpose autopilot.

Fitness evaluation of a controller can be divided into two main stages. First is the preparation of the sample task and simulation of the model with the controller in the loop. The second stage is analysis of the results obtained from the simulation and evaluation of the fitness as such. These steps are often repeated several times, making the controller perform varying tasks in order to obtain a balanced measure of its performance via multi-task fitness evaluation.

Fitness value may reflect different levels of performance achieved by the controller, including 'hard bounds.' For example, if the controller crashes the UAV, a high penalty score may be assigned.

Other parameters of flight taken into account is control usage (or control effort): it is desirable to keep control usage at minimum. Tracking abilities, if applicable, can be estimated by the weighted sum of the tracking error.

The total fitness value is calculated as the weighted sum of all estimates:

$$F = W_c C_c + W_f C_f + W_e C_e + \ldots \tag{8}$$

The exact value of the weighting coefficients W, as well as the number of estimates taken into account, is individual for each controller. As a rule, the weighting coefficients are chosen empirically. It should be noted that the signals are dimensionalised and thus the coefficients may vary significantly. For example, aerodynamic surfaces deflection is measured in degrees and may range from –16 to 16 (for rudder and ailerons); considering 10-fold saturation limits, the peak values may reach ±160. At the same time, throttle range is 0.01 to 1, thus the cost for its usage will be about 30 times as low (for similar control dynamics).

3.4 Evolutionary design algorithm outline

The Evolutionary Design algorithm is implemented as a function with two input arguments: population size and number of generations to run. Other, more specific parameters are initialised within the program. The algorithm framework is as follows.

1. Initialise all parameters. If the evolution is continued, go to step 3.
2. Create initial population.
3. Evaluate fitness of each individual.
4. Save the full state to a temporary file for emergency recovery.
5. If the required number of generations is reached, return.

6. Sort the population according to the fitness of each individual.
7. If elitism is enabled, copy the best scored individual into the new population.
8. Until the new population is filled up:
9. Select the next individual from the sorted list (starting from the elite member).
10. Every (*k*s)th generation, with probability *P*s, perform structure mutation of the selected individual.
11. Reproduce *n* offspring individuals from the selected (and possibly mutated) individual.
12. Put these *n* new individuals to the new population.
13. Continue the loop from step 8.
14. Increase the generation counter.
15. Continue from step 3.

Several steps of this algorithm need further clarification.

Initial population is created according to the specified task. By default, it is initialised with the control laws of the form $y = const$ with randomly chosen constants. Most of the controllers, however, are initialised with more meaningful control laws. For example, tracking controllers may be initialised with the laws $y = k_1\varepsilon + k_0$, where ε is the respective error signal and the coefficients *k* are sampled at random (taking into account default step sizes for the respective signal).

It can be seen that *selection* is performed deterministically. This is the most commonly used way in ES. The populations used in this study were of moderate size, usually 24 to 49 members. Selection pressure is determined by the number of the offspring *n* of each individual. The smaller *n*, the lower the selection pressure. For nearly all runs in this work $n = 2$, which means that half of the population is selected. This is a rather mild level of selection pressure.

The parameters determining *structure mutation* occurrence, k_s and P_s, both change during the evolution. As discussed previously, the number of structure mutations should decrease as the complexity of the controllers grows and more time to optimise the coefficients is required. The probability of structure mutation P_s is normally high in the beginning (0.7 to 1.0) and then decrease exponentially to moderate levels (0.4 to 0.6) with the exponent 0.97 to the generation number. Default value of k_s is set according to overall complexity of the controller being evolved. For simpler single-output controllers, as a rule, $k_s = 10$; for more complex controllers $k_s = 20$. However, in the beginning of evolution, k_s is reduced by half until the generation 20 and 100 respectively.

Reproduction is performed simultaneously with mutation, as it is typically done in ES, with the exception that this operation is performed separately for each selected member.

4. Controller synthesis and testing

The UAV control system is synthesised in several steps. First, the flight controller is produced. This requires several stages, since the flight controller is designed separately for longitudinal and lateral channels. When the flight controller is obtained, the guidance control laws are evolved.

Application of the ED algorithm to control laws evolution is fairly straightforward: 1) preparation of the sample task for the controller, 2) execution of the simulation model for the given sample task and 3) analysis of the obtained performance and evaluation of the fitness value. When both the model and fitness evaluation are prepared, the final evolution may be started. Typically, the algorithm is run for 100–200 generations (depending on complexity of the controller being evolved). The convergence and the resulting design is then analysed and the evolution, if necessary, is continued.

4.1 Step 1: PID autothrottle

Initially a simple PID variant of autothrottle is evolved - to ensure a more or less accurate airspeed hold. At the next stage, its evolution is continued in a full form together with the elevator control law. The PID structure of the controller may be ensured by appropriate initialisation of the initial population and by disabling structure mutations. Therefore, the algorithm works as a numerical optimisation procedure. The structure of the autothrottle control law is following:

$$\begin{aligned} \dot{x}_1 &= k_1 x_1 + k_2 \Delta V_a \\ \delta_t &= k_3 x_1 + k_4 \Delta V_a + k_5 \dot{V}_a + k_6 \end{aligned} \tag{9}$$

where $\Delta V_a = V^d - V_a$ is the airspeed error signal, δ_t is the throttle position command and $k_{1...6}$ are the coefficients to be optimised.

A 30-second flight is allocated for performance measurement. Such a long flight is needed because throttle response is quite slow. Since robustness of the PID controller to discrepancies in the UAV model is not topical at this stage (it will be addressed in further evolution), and because structure of the controller is fixed so that irrelevant measurements cannot be attracted, only a single simulation run for each fitness evaluation is performed.

Elevator inputs provide the main source of disturbances for training the autothrottle. A 2.5-degree nose-up step input is executed at time t = 5 s. It produces nearly steady climb without reaching stall angles of attack and saturation on the throttle (except for, possibly, dynamic transition moments).

At t = 14 s, a similar nose-down elevator step is commanded, entering the UAV into glide with small descent angle. In addition, a 3 m/s tailwind gust is imposed at t = 23 s. Generally, manoeuvres and wind disturbances are the most prominent sources of airspeed variations, therefore they are included for autothrottle assessment.

Initial airspeed is set to 23 m/s, i.e. 1 m/s above the required airspeed. This provides an additional small scale disturbance. Initial altitude is 100 m, providing enough elevation to avoid crash for any sensible throttle control.

Algorithm settings are as follows. Since the structure is fixed, large population size N is unnecessary. Population size of 25 members has been used in most runs. N = 13 showed similar results in terms of fitness evaluation demands (red line on the convergence graph in Fig. 7a; adjusted to the population size 25 for comparison). Fitness is evaluated deterministically because all random signals are repeatable from run to run. Elitism is enabled. Fitness is calculated with the following weighting coefficients:

$$F_t = 2000 C_e(V_e) + 1000 C_c(\delta_t) + 2000 C_f(\delta_t) \tag{10}$$

It should be noted that the *commanded* signal δ_t (which enters the actuator) is used for evaluation.

Convergence graphs for three independent runs of the ED algorithm are presented in Fig. 7a. They show the best fitness values for each generation. On average, 2000 to 2500