Evolutionary Robotics in High Altitude Wind Energy Applications

Allister David John Furey

January 2011

Centre for Computational Neuroscience and Robotics

Department for Informatics

University of Sussex

Falmer BN1 9QG

A thesis submitted for the degree of Doctor of Philosophy of the University of Sussex

UNIVERSITY OF SUSSEX

Evolutionary Robotics in High Altitude Wind Energy Applications

Allister Furey

Summary

Recent years have seen the development of wind energy conversion systems that can exploit the superior wind resource that exists at altitudes above current wind turbine technology. One class of these systems incorporates a flying wing tethered to the ground which drives a winch at ground level. The wings often resemble sports kites, being composed of a combination of fabric and stiffening elements. Such wings are subject to load dependent deformation which makes them particularly difficult to model and control.

Here we apply the techniques of evolutionary robotics i.e. evolution of neural network controllers using genetic algorithms, to the task of controlling a steerable kite. We introduce a multibody kite simulation that is used in an evolutionary process in which the kite is subject to deformation. We demonstrate how discrete time recurrent neural networks that are evolved to maximise line tension fly the kite in repeated looping trajectories similar to those seen using other methods. We show that these controllers are robust to limited environmental variation but show poor generalisation and occasional failure even after extended evolution. We show that continuous time recurrent neural networks (CTRNNs) can be evolved that are capable of flying appropriate repeated trajectories even when the length of the flying lines are changing. We also show that CTRNNs can be evolved that stabilise kites with a wide range of physical attributes at a given position in the sky, and systematically add noise to the simulated task in order to maximise the transferability of the behaviour to a real world system. We demonstrate how the difficulty of the task must be increased during the evolutionary process to deal with this extreme variability in small increments. We describe the development of a real world testing platform on which the evolved neurocontrollers can be tested.

Submitted for the degree of Doctor of Philosophy

University of Sussex, UK

Submitted: January, 2011

Publications

The following publications have arisen from work described in this thesis:

Furey, A., & Harvey, I. (2007). Evolution of neural networks for active control of tethered airfoils. In F. A. e Costa, L. M. Rocha, E. Costa, Inman Harvey, & A. Coutinho (Eds.), Advances in Artificial Life, 9th European Conference, ECAL 2007 (pp. 746–755). Berlin, Heidelberg: Springer.

Furey, A., & Harvey, I. (2008). Robust adaptive control for kite wind energy using evolutionary robotics. Proc. Biological Approaches for Engineering, 17-19 March 2008 (pp. 117-120). Southampton

Furey, A., & Harvey, I. (2008). Adaptive Behavioural Modulation and Hysteresis in an Analogue of a Kite Control Task. In M. Asada, J. C. T. Hallam, J.-A. Meyer, & J. Tani (Eds.), From Animals to Animats 10 (Vol. 5040, pp. 509-518). Berlin, Heidelberg, Springer

Acknowledgements

I have been so utterly dependent on the goodwill and support of so many people, this work would have been truly impossible without them. I'll try and go some way to acknowledging them here.

First thanks must go to Inman Harvey for his support and advice and for agreeing, presumably in an uncharacteristic moment of poor judgement, to let me do this work at all. Thanks also to Sam Metcalf who started this whole thing off with lunchtime kite flying sessions in Aylestone park and in speculating with me what might be possible, those few hours set off a trajectory of over four years and hopefully more. Thanks to Bill Bigge for building the first kite robot and introducing me to the wonders of the robot lab. Thanks to everyone in the CCNR who submitted to coming down to the field and/or the downs to be dragged around and spend time relaunching kites, that's all of Jose Fernandes Leon, Lucas Wilkins, Nick Tomko, Tom Parslow. Tom you are also a legend for resurrecting the Gantry robot. Andy Philipedes for his no nonsense advice –'I think you're being somewhat naïve' which definitely kept my feet on the ground. I must thank Martin Ling, Alex and DK Arvind from Specknet at the University of Edinburgh for allowing me to fly their expensive IMUs and talk with the group there. Thanks must go to Bill Hampton and John Prewer from Kitetech who provided the push needed to get the robots working, nothing like a test being scheduled to provide the motivation to fix the robot! Along the same lines, I have to thank Colm O' Gairbhith and Rob Creighton at Windlift the vision system was born on those windless days in Salvo NC. Thanks to Dr Olly Woodford for his orientation code, Naser Sayma and Sanjeev Sanghi for helping out with the CFD. Thanks to Henry Rebbeck for being interested from word go, and for going out of his way to help out. Thanks to Roland Schmehl and the crew at TU Delft for letting me come over and test on their amazing kite system, seeing the team in action was an inspiration. Thanks to my family for their support and for not questioning why I didn't get a proper job and Clare and Nicky for putting me up in the last stretch.

Thanks most of all to Camilla Bowkett, who has been there the whole time, spent more time behind a kite than everyone else put together and listened to endless hours of kite relate boredom. This thesis is dedicated to you.

Table of Contents

List of figures

Consider that we ought not to act and speak as if we were asleep, for even in sleep we seem to act and speak; and that we ought not, like children who learn from their parents, simply act and speak as we have been taught.

Marcus Aurelius, Meditations, 180AD

1 Evolutionary robotics in high altitude wind energy applications

This thesis focuses on the use of evolutionary robotics (ER) techniques to address a single problem; that of controlling an unstable but steerable tethered wing or kite for the purposes of wind energy conversion.

This document seeks to first provide a comprehensive introduction into the use of ER through a review of relevant literature, especially where ER is applied to challenging control applications. It then goes on to assess both the common and the differentiating factors between this and other problems in the literature. The second portion of the document then describes the approach taken for evolving neurocontrollers for simulated kite control tasks of varying complexity and the value of this approach in generating highly robust flight controllers, before describing the hardware platform developed for real world flight control tests and the resulting outcome.

In this chapter, we introduce the motivations behind tackling this particular problem, introduce some issues that make it challenging and interesting from the point of ER, and finally summarise the novel contributions that this study brings to the field.

1.1 Motivation

Here we will briefly delineate why tethered wing control is both interesting from an academic perspective and worthwhile due to the value of going some way to addressing an important issue for a promising new class of renewable energy technology.

The problem of autonomous kite control is noteworthy due to the novelty of the task and the challenges imposed by poor information, underactuation and poor control authority of the wing, and variation in the task's dynamics. These variations stem from changes in the environment and, due to the flexibility of the wing, changes to the wing geometry during flight. The structural warping to which the kite is subject, coupled with the unconventional geometry of the kite and the fluid medium in which the kite is situated make conventional modelling and control techniques difficult to apply. These challenges have meant there are no successful implementations of a control algorithm on a real world test system in the academic literature, despite nearly a decade of work. The promise of using bio-inspired methods, specifically evolutionary robotics, to generate controllers that may be able to maintain the performance of qualitatively consistent flight control behaviours whilst the wing undergoes structural changes in a changing environment is the primary motivation behind this work.

Further motivation from the academic point of view is provided by the opportunity to test the scalability of the evolutionary robotics approach to particularly variable problems operating on very fast timescales in domains with complex physics and much nonlinearity, aspects which currently present insurmountable barriers to the creation of analytical models.

ER, with its history of successfully using numerical and/or minimal models to generate adaptive neural control systems appears comparatively suitable, when compared to conventional techniques such as Model Predictive Control (MPC), in which a precise model of the system being controlled must be available (Garcia, Prett, & Morari, 1989). Demonstration of the capabilities of ER to control, in the real world, entirely non-rigid bodies operating in environments dominated by fluid/solid interactions is not yet available, so we seek to work toward that end here. Further to this aim, this specific application allows us to probe the level of fidelity of system modelling that is appropriate or desirable in such complex systems, where a given level of modelling may be essential to capture the dynamics of the system, but where a 'full' description would entail such a high level of computational demand that it is rendered implausible at the current computer power:cost ratio.

As a standalone application there are compelling reasons that make airborne wind energy highly attractive. There is a great demand for technology that will assist in abating emissions of greenhouse gasses such as carbon dioxide and methane, due to pressing concerns about anthropogenic climate change. Renewable energy technologies are one such avenue and approaches that promise to tap large reserves of renewable energy at a viable cost are highly attractive.

The recent interest in using tethered wings to capture wind energy derives from the distribution of wind with altitude. As one moves up in altitude through the atmospheric boundary layer, further away from the friction caused by the underlying terrain, the average speed of the wind increases, an effect known as vertical shear (Wallace & Hobbs, 2006),(Kaimal & Finnigan, 1994). It should be noted that the aerodynamic forces available to be harnessed increase exponentially with wind speed, meaning that even slightly increased average wind speeds can yield significant improvements in power generation. Accessing this superior energy resource means that it should be possible to reduce the cost of wind energy given an equivalent hardware cost, (Loyd, 1980). Because these systems operate without resorting to tall and expensive towers necessitating deep foundations, it is plausible that the installed system costs will be similar if not superior, so it is likely that these economic advantages can be realised (Canale, Fagiano, Ippolito, & Milanese, 2006).

Loyd's seminal paper (Loyd, 1980) on this subject detailed two possible modes of operation of airborne wind energy systems; the *drag mode*, whereby impellers are placed upon a tethered wing, generating electricity on-board and passing the electricity down the tether to ground level, and the *lift mode* whereby force exerted on the tether by the kite is allowed to perform useful work at ground level by driving a winch or hydraulic accumulator or similar. It should be clarified that in this latter case, two reciprocating phases must occur. In the first ‘generation’ phase the kite is controlled such as to maximise line tension is and the force utilised to rotate a winch coupled generator or equivalent. In an alternating ‘retraction’ phase the system must be reset by retracting the line that was paid out in the generation phase. In the next chapter we argue that lift mode systems are likely to be more commercially viable than drag mode systems as they can utilise lighter, cheaper wings, with less inertial loss, reduced capital costs, greater safety, and superior upgradability. However these systems are underactuated compared to conventional flight platforms and the wings are much more flexible, rendering analytical models much more difficult to develop. The alternating phases of operation and the required transitions between flight modes also add complexity to the control requirements in lift mode systems.

Regardless of commercial or operational advantages and disadvantages, here we are compelled to tackle the lift mode approach, due to the reduced cost of testing in the real world and the ability of fabric kites to withstand impact with the ground during the development process, eliminating costly rebuilds. However, by meeting the challenges of poor control authority and dynamic wing shape head-on, we also can go some way to assisting the progress of development of lift mode airborne wind energy systems.

In summary, there is a double motivation for approaching control problems in airborne wind energy with ER techniques. Firstly it is an interesting and challenging problem from an academic perspective, a summary of the task characteristics is given below in section 1.2. Secondly, airborne wind appears to be a promising avenue for bringing down the cost of renewable energy, and due to the specific characteristics of the physical system and the control task ER approaches may make some useful contribution to the development of this technology as a whole.

1.2 Specific considerations of this task

Although this is a merely an introductory chapter, it is appropriate to give a flavour of the challenges specific to this task, to frame the detailed analysis given in later chapters. Here we introduce these key challenges, the important dynamics in the system, the availability of information, and the flight behaviours which we require our neurocontrollers to be able to perform.

At the core of the task is the need to control a tethered wing, steerable and yet flying without propulsion purely via aerodynamic lift from naturally present wind. The wing has three rotational degrees of freedom, however it is only steerable in roll by default, in some circumstances its pitch might also be under control, but there is at no time explicit control of yaw angle.

By way of orientation, Figure 1.1 shows a steerable kite attached to a control device by the default configuration of 4 four lines.

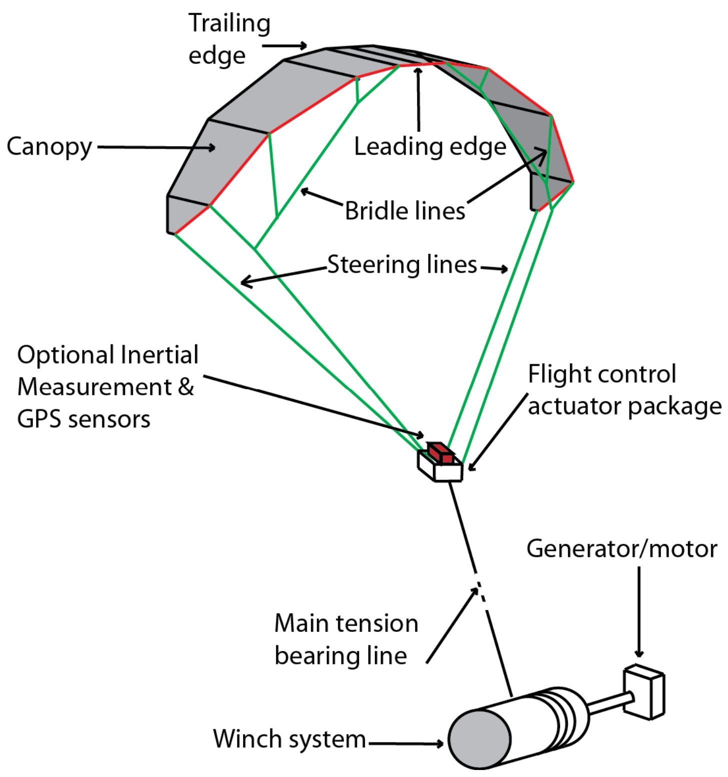

Trailing edge
Canopy
Leading edge
Bridle lines
Steering lines
Optional Inertial Measurement & GPS sensors
Flight control actuator package
Generator/motor
Main tension bearing line
Winch system

Figure 1.1 Diagram of a steerable kite attached to a winch at ground level. The actuator package alters the relative lengths of the steering lines in order to steer the kite, primarily by roll i.e. rotation of the kite around the axis running from leading to trailing edge.

There are two main sources of change or dynamics in the task; environmental changes and changes in the kite system itself. The kinds of environmental changes to which any flight control system must be robust are largely meteorological; wind speed, wind direction, air density and precipitation, each of which may be remarkably dynamic over different temporal and spatial scales. The variation of the kite system is of more varied subtlety, properties such as the altering of mass and mass distribution of the kite as the fabric is wet and dried, or the asymmetric stretching of control lines are relatively simple. However the change of the kite shape and orientation with regards to the wind is a complex interplay between the control input, tension in the line, kite position with respect to the tether and the wind itself. The feedback between the kite shape, line tension, the wind and controller is what makes this problem both interesting and difficult.

Because this is an applied problem, we have to base any flight control behaviour on the requirements of the application at hand. As introduced in section 1.1, for systems operating in the lift mode there are two alternating phases of operation; generation and retraction, which constitute a single cycle. These phases necessitate different flight behaviours, with the common constraint of keeping the kite in flight. The first behaviour is to maximise the force that is exerted along the kite tether which is used to perform some useful work at ground level, for example to drive a winch. Rotating a winch or pulling a piston will inevitably result in the extension of the flying lines. As we shall see, dynamic flight manoeuvres are necessary to achieve the highest line tensions as they raise the flight speed of the kite, and the resulting aerodynamic lift acts close to alignment with the tether. Naturally these dynamic flight manoeuvres also increase the risk of contact the ground by lowering the kite's elevation and bringing it closer to the ground. The second key behaviour is to minimise the line tension, preferably whilst keeping the kite at high elevations, as close to the zenith position as possible. This allows for retraction of the kite line with the minimum expenditure of energy resulting in the highest net gain of energy over a generation and retraction cycle. Any valid kite controller must be able to switch between these modes as directed by a human operator, any software heuristic or algorithm, or an integrated or separate evolved neural network controller.

To recapitulate briefly, we need to steer an unpowered wing being flown as a kite using one or two roll angles. The controller must be robust to changes within the environment and the unknown warping of the wing under load. There are two key flight control behaviours if a lift mode system is being used to generate power, the first involving the maximising of line tension through dynamic flight manoeuvres and the second involving minimising line tension by stabilising the kite in an overhead holding position.

1.3 Evolutionary robotics as a potential solution

The approach that we take here in order to generate these flight control behaviours is to implement an evolutionary robotics methodology. ER is potentially appropriate as it has shown to be capable of successfully generating controllers for control tasks when the exact details of required behaviour are not known, using simulations that are in some respects markedly simpler than the real systems in which the control behaviour must be performed.

This approach necessitates the creation of a simulation environment which replicates some key aspects of the system to be controlled, its environment and their interaction at some achievable level. Next we create a population of neural network based controllers, in some cases from a random seed, and in some cases from hand coded parameter sets. We then test these controllers in the simulation environment and score their performance based on some user generated metrics encapsulated in a fitness function. In a selection process analogous to natural evolution, the better scoring controllers are more likely to pass their genes into the next generation. Repeated iterations of this process, in which an element of mutation drives an exploration through parameter space, should lead to the evolution of controllers that are progressively more capable of meeting the requirements of the task.

A particularly challenging requirement of this task for an evolutionary robotics implementation is the transition from operating in the simulated environment to the real world. It is necessary for the simulated environment to capture sufficient details that are available in the real world without allowing evolved controllers to rely on artefacts present in the simulation that are not in fact present in the real world. A common approach to preventing the use of such artefacts is a severe application of noise to all of the aspects of the simulation except those that are guaranteed to be present in reality. In this case, we tread a tightrope between the high computational demand of accurate simulation, the requirements of the controller to be exposed to dynamics resembling those that they will encounter in the real world, and the fact that extreme levels of noise can render the evolution of controllers unacceptably slow or indeed impossible. To this end, particular attention is paid to the level of simulation that is most appropriate for a task of such complexity and the manner in which task variation and noise is deployed throughout the evolutionary process.

In summary, ER offers a potentially productive approach to generating controllers for this problem, due to the possibility of creating valid neurocontrollers using relatively simple simulations. However there are challenges relating to transferring these behaviours to the real world system that will have to be identified and addressed.

1.4 Novel contributions

This study makes several novel contributions to the field as detailed below:

- The primary contribution is a description of the use of evolutionary robotics in controlling the flight of what is essentially a tethered wing, where the attitude of the lifting body with respect to the air flow is only controllable in one or two axes and where the wing lacks propulsion.
    - We also demonstrate the first use of evolutionary robotics to control a fully compliant flying structure, where there are no fully rigid elements and there is heterogeneous distribution of compliance across the structure.
    - We show that evolve neurocontrollers can be generated that will stabilise such wings at a given point even in highly variable wind conditions.
    - Additionally we demonstrate that evolutionary robotics is capable of generating appropriate flight trajectories for a given task that at least qualitatively match those generated by other modelling and optimisation techniques.
    - Ultimately we demonstrate that evolved neurocontrollers are indeed a suitable substrate for the control of flexible tethered wings for multiple flight

    behaviours in simulation including taking over control from a human pilot during dynamic flight manoeuvres.

- We describe a number of novel results relating to the evolutionary process when evolving neurocontrollers in simulation for a real world flight control task
    - We show how a very fine grained incremental, shaping scheme, approach is necessary in order to evolve the required flight control behaviours under the amount of noise necessary to achieve successful transfer of behaviour simulation to reality
    - We show that providing evolution with a hand coded network as a seed improved the yield of evolutionary runs but not the peak performance, whilst the provision to the network of input parameter velocities increased both performance and yield.
    - We show that an evolutionary approach can be used as a simple tool to dramatically improve the fidelity of a multibody wing model in predicting deformation to a given directional load, when compared to random or hand-coded models.
- We describe for the first time in the academic literature implementations of both ground-based and kite suspended control robots, capable of steering kites up to 8 m^2 by manipulating their roll angle and modulating aerodynamic coefficients by changing the attack angle.
- We demonstrate a novel and simple multibody kite physics model that is able to replicate some of the common deformations that flexible kites undergo, and produce several dynamic flight behaviours.
- We show that in a simplified version of the kite control problem, evolved neurocontrollers are able to control a dynamic agent operating in a 2-D space such that it alternates between two modes of operation in an manner that adapts to environmental changes operating at multiple timescales.

We hope that the work described in this thesis is read as a specific exemplar of a wider class of problems, and that these methods can be applied not just to kite control, but for other complex real-time control problems. These techniques may be especially appropriate for problems sharing some of the characteristics of this task, namely; where difficulty exists in producing an accurate model, where there is a high degree of non-linearity in the physical system and where information available to the control system is incomplete or noisy.

2

We shape clay into a pot, but it is the emptiness inside that holds whatever we want.

Lao Tzu

2 Literature review

Here we will cover the main areas that are necessary to give sufficient context for the use of evolutionary robotics for kite control. The first section considers evolutionary robotics itself, including the motivation for use and its methodology. We then consider some specific implementations both in software and in hardware, including the use of evolved neural networks for the control of flight. The idea behind this section of the review is to gain an appreciation of what aspects of the diverse set of ER implementations in the literature would most be appropriate when we consider the application of ER specifically to kite control in Chapter 3. The second section introduces airborne wind energy, the motivation behind its development, its mode of operation including the operational environment and the very basics of the physics of kite flight. We conclude with some consideration of existing approaches to airborne wind energy generation in terms of system configuration and hardware design.

2.1 Introduction to evolutionary robotics:

Typically an evolutionary robotics approach involves the use of a neo-Darwinian evolutionary paradigm that is used to incrementally evolve artificial neural networks (ANNs) to produce a given behaviour. Here we will initially briefly introduce the use of artificial neural networks and consider the limitations of traditional approaches. This will provide the context to explain why the evolutionary method can be an appropriate one for autonomous robotics applications.

2.1.1 Traditional approach to artificial neural networks

Neural networks are abstracted parallel processing systems inspired by biological nervous systems. They have been developed over a number of decades and their use for such problems as classification and prediction is well established (Bishop, 1995) . The archetypal artificial neural network is the multilayer perceptron (MLP) (Minsky & Papert, 1969), (Rumelhart & McClelland, 1987)where each layer is composed of simple neuron inspired processing units known as perceptrons (Rosenblatt, 1958). The structure includes an input layer, an output layer, and at least one intervening hidden layer whose neurons do not receive any external input or generate direct outputs (see Figure 2.1). Each individual unit receives one or multiple signals from either external inputs or other network neurons from the immediately preceding layer and performs a weighted sum of these inputs to derive its own 'internal' activation value.

Eq. 2-1 shows how the activation *a* of given neuron *i* is derived given a connection weight *w,* output value *y* from neuron *j*, subject to a static bias *b* where there are *n* neurons in the preceding layer.

Eq. 2-1

$$a_i = \sum_{j=1}^{n} w_{ji} y_j - b_i$$

The output value of a given neuron is this activation value mapped via a transfer function which may be a linear mapping, but is normally a non-linear mapping such as a step function or a smooth differentiable function such as the sigmoid or logistic function (see Eq. 2-2).

Eq. 2-2

$$O_i = \frac{1}{1 + e^{-a_i}}$$

The architecture of a multilayer perceptron is described as feedforward, because the connectivity of the network is restricted such that information, in the form of neuron activations and their subsequent outputs, is only passed from higher to lower levels i.e. from the input through the hidden layers to the output layer.

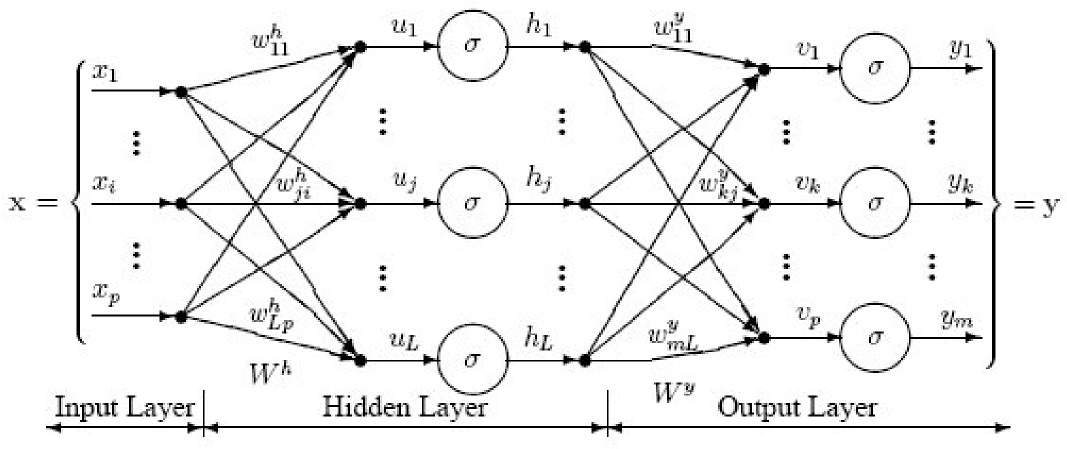


Figure 2.1 Diagram of a feedforward neural network, neurons or nodes denoted as black dots, transfer functions are marked with σ with inputs x, weights w, hidden layer activations u, hidden layer outputs h, output layer activations v and outputs y, image source (Sherrod, n.d.).

This restriction on the architecture of the network essentially restricts the network to performing a direct instantaneous mapping function from inputs to outputs and consequently the network has no internal state. By this we mean that prior inputs will have no effect on the output of the network at any given moment. Only the input that is currently presented is available to the network, although some extensions to the MLP architecture have been implemented that may use a simple storage layer that allows association of temporally contiguous data,(Elman, 1990),(Stagge & Sendhoff, 1997). It can be demonstrated that multilayer perceptrons are able to approximate complex non-linear mappings (Hornik, Stinchcombe, & White, 1989) and one can appreciate how such networks are well suited to classification tasks in which an input is presented to the network and the value of the output or outputs from the network are treated as a classification signal. Naturally, the network must undergo a training process to adjust the network weights such that the desired mapping is achieved. The breakthrough that allowed this training process to be implemented was called the back propagation rule (Rumelhart, Hinton, & Williams, 1986), whereby using an existing set of data in which the correct classification is known, the weights in the network could be adjusted when incorrect classifications were made such as to reduce the future classification error.

2.1.2 Limitations of these approaches

While these and other techniques such as radial basis functions and support vector machines are powerful and have been applied successfully within their intended domains (Bishop, 1995), they also have important limitations in scope. Foremost is the requirement within these techniques to know in advance what is considered an appropriate or successful output at a given time or for a given set of inputs. In the context of a robot existing within a relatively complex environment, it is most often not a trivial task and potentially impossible to determine what the 'correct' course of action is at a particular moment. Without knowing exactly which motor commands should be given to the robots actuators when faced with a given input for example, we have no obvious teacher with which the actual output of the network can be compared in order to gauge performance of the network. Bound in with this first limitation is the fact that a robot must necessarily exist within the real world within real time. This is an important distinction compared to simply mapping a single input to a single output which is the case with a multi-layered perceptron. The second important limitation therefore is that an MLP will always produce the same outputs when faced with a given input. In many real world situations it would be advantageous for a robot to take into account the past history of its interaction with the environment.

As an example, we can consider the task of autonomously flying a small aircraft to a given target or waypoint. We may consider a good solution to be one where the target is approached with the minimum cost in fuel and in time, thereby suggesting that a good solution may constitute one in which target is approached in the most direct possible manner. However it may well be that the conditions with which the aircraft is faced will require that a modified control output will be required. A sidewind (a wind , parallel to the ground and orthogonal to the direction of motion) would require that a more or less severe steering output should be made to adjust the course to maintain the heading of the plane towards the waypoint. If the only information available to the network is the current position and the target position, then a MLP will never be able to adjust the gain of the steering signal appropriately. Even if a wind direction sensor such as a multi port pitot tube were available, we would still desire a solution that would be robust to the failure of such a sensor. This lack of perfect knowledge about an environment is a common feature of real world control situations in autonomous robotics. However, a network that has some form of internal state or memory available to it may be able to encode its past experience of interaction with the environment. Ideally this encoding can be accomplished in such a way that even if the extent of the crosswind is not explicitly known, the appropriate steering signal can be produced, because the change in actual position compared to that which might be expected for a given steering output can be taken into account when producing steering output in subsequent times. Under such a scheme, the emphasis is shifted from an information-centric perspective to an interaction-centric one.

This problem exemplifies a difficult issue within autonomous robot control, where we have some task or goal in mind that may take some nontrivial amount of time to achieve; unless we are in a tightly controlled environment such as a automotive production line, or have perfect information as to the state of the environment at a given time, it is rarely possible to specify the exact outputs that we wish the network to produce. One technique that avoids some of these limitations is reinforcement learning, of which q learning (Watkins, 1989) is a notable example. This and related techniques provide a scalar signal which is delayed from the performance of a given action, rather than the specific and immediate feedback of the 'teacher' as per back propagation and other supervised methods. However, one crucial aspect of reinforcement techniques is that of trial and error (Sutton & Barto, 1998), i.e. the agent is required to explore the space of possible actions with no prior knowledge. Whilst this might be appropriate for behavioural simulations, optimisation of logistics problems, or even longer term adaptivity in robotics, this characteristic renders the approach inappropriate for the creation of neural controllers that will function 'well enough' immediately on introduction to an environment.

Hopefully by this point some kind of impression has been conveyed as to the need for two key elements that will enable the use of neurocontrollers in autonomous robotics problems with non-trivial dynamics that are subject to change. The first is a requirement for a neural network implementation to have an internal state or memory. In this way, neurocontrollers at least have the potential to be adaptive in the sense that a behavioural strategy may be tuned or modified in a way that is appropriate to the nature of the coupling between the agent and the environment as they now stand, in a world in which it is possible for either the environment or the agent itself to be subject to significant change. The second requirement is for a paradigm in which these networks can be built or developed to perform a specific task whereby the means of achieving the end goal are left up to the internal workings of the process itself, primarily because it is difficult or impossible to specify what is the appropriate action at any given time due to imperfect knowledge of changes within the environment or the reality of the robot's physical embodiment.

2.1.3 Achieving adaptive autonomy in robots

As intimated above, traditional ANNs in the form of an MLP are highly abstracted models of biological neural systems and we need not look far into the neurobiology for a potential solution to their limitations. There are two key ways in which the lack of state within the network may be addressed, the first and most obvious is the changing of strength of synaptic connections within the network in a way that somehow depends on experience. The second is to have neural elements that are themselves dynamic and therefore some information may be available to the network in the form of the state of individual neurons, groups of neurons, or the network as a whole that allows the robot's past experience of interaction with the environment to be taken into account at a given moment. It may be that both of these elements are required to achieve rich and adaptive robotics behaviours over long timescales. Here however we focus on the latter approach i.e. richly dynamic networks, principally because the task of kite control does not have 'soft edges' where there is a lot of information available in gentle failure such as might exist in a wheeled robot exploration task. In fact the edges are 'hard' in that catastrophic failure in the form of a crash of the kite may be unavoidable after mere seconds of an inappropriate control signal. To this end we need a neurocontroller that will enter an appropriate behavioural regime immediately upon initiation into the task environment, and in which adaptation to change within the agent, the environment or some aspects of their interaction happens rapidly.

2.1.4 Internally dynamic neurocontrollers as a route to adaptive robotics

Operating on the assumption that Hebbian learning schemes, in which the strength of connections between neurons are modified according to the relative activity of the pre and post synaptic neurons, are unable to converge within a second or so to an appropriate behaviour, at least in situations in which their weights are not pre-initialised, we must address the second possible solution, that of seeking networks rich in dynamics and state as a route to adaptivity. It should be noted that networks without Hebbian learning mechanisms do not preclude 'learning' in a formal sense (Izquierdo & Harvey, 2006) nor is it impossible to deploy Hebbian or related mechanisms in a scheme composed of dynamical neural units.

Fortunately, there exists classes of artificial neural network that are closer in some respects to real biological neural systems, that may have some of the qualities that we are seeking for autonomous robotics applications. Specific examples include spiking neural networks and continuous time recurrent neural networks (CTRNNs). The former of these two classes of network is the closer to the biological reality, in that neurons have an internal state corresponding to the voltage difference across their cell membrane which is disturbed away from an actively maintained norm by inputs arriving from other neurons across synaptic connections. As in real neurons, the output of a given neuron is a single event known as the action potential which constitutes a tightly temporally restricted binary signal that is referred to as the 'firing' of the neuron or a 'spike'. This action potential event is initiated when the membrane potential exceeds a threshold value and a rapid cascade of voltage dependent ion channel openings is initiated. Because the membrane potential takes some time to decay back to the default resting state after each synaptic input event, there is a nontrivial relationship between the current state of the membrane of a neuron, and thereby its likelihood of firing at a given subsequent time, and the recent history of inputs to that neuron, whether from external sensors or from the rest of the network. This is but one example of the state with which biological nervous systems are richly endowed. Other examples must of course include strength of synaptic connections which are subject to change according to the relative timings of pre and post synaptic action potential (Bi & Poo, 1998). More short term state changes can include firing phase relationships of comparatively large groups of neurons or participation of subpopulations of neurons in complex stereotypical time-locked firing patterns first identified as synfire chains, (Abeles, 1982) and described in more recent computational studies as polychronous groups (Izhikevich, 2006). On a medium time scale, dynamic state can include the participation of non-neural populations of glial cells in modulating synaptic efficacy (Volterra & Meldolesi, 2005), (Newman & Volterra, 2004) and development (Pfrieger, 2002). Being closer to the biological reality, spiking networks can employ learning mechanisms which modify synaptic weights according to known biological processes, even to the level of allowing secondary modulation of STDP in simulated embodied robot control tasks (Chorley & Seth, 2008).

CTRNNs constitute a more abstract model that take an intermediate space between traditional ANNs and networks with realistic spiking dynamics, operating in a continuous time in contrast to the discrete time of the traditional feedforward network. Each neuron outputs a continuous real valued signal rather than the binary on/off output of a spiking network. CTRNNs generally employ no such explicit learning mechanism, but the internal state of the network might be subject to alteration in such a way that the network outputs at a given time take into account the dynamics of agent/environment interaction at previous times. Indeed evolved CTRNNs have been shown to be capable of sequential decision making and learning (Yamauchi & Beer, 1994), properties that had hitherto been assumed to be dependent on permanent internal state modifications mediated by Hebbian or similar learning rules. CTRNNs do not attempt to replicate the multiple mechanisms that contribute to the internal state dynamics of biological nervous systems at different timescales. However, because each constituent node in the CTRRN is allowed to integrate its input at timescales varying over several orders of magnitude, CTRNNs could therefore be considered to be a generic abstract nervous system model, insofar as the state of the network is rich and heterogeneous in its reaction to, and reflection of, both internal network activity and external input.

In the work presented here, we focus on the use of CTRNNs largely because of the large volume of prior work within the literature describing the implementation of the evolutionary process and the dynamics of the resulting neurocontrollers. As already intimated, in a task that is highly performance sensitive such as that of flight control, we require robot neurocontrollers to achieve a certain level of performance ‘off the bat’ i.e. to be effective immediately on introduction to a task environment. To achieve this level of performance, artificial evolution is a promising candidate as it can produce neural network controllers that are at least in the correct region at the start of active interaction between the agent and it’s environment, producing non-deleterious motor control signals immediately without an explicit learning mechanism that might operate on the scale of minutes or hours of real world time.

2.1.5 Genetic algorithms as a method of artificial evolution

Here we will provide a concise general introduction into the use of genetic methods as a problem solving technique in order to provide the context for the specific cases that we review in Section 2.1.6.

There are a number of approaches to artificial evolution which appear to have been developed in some detail in near independent scientific groupings with relatively little crossover between approaches. For example Evolutionary Strategies (ES) and Genetic Programming (GP), for which see (Beyer, 2002) and (Koza, 1992) respectively for a review. However in common with most of the evolutionary robotics literature we here employ genetic algorithms (GAs). GAs were first popularised by Holland, (Holland, 1975) and are an incremental evolution technique for binary or real valued strings of parameters. This means that GAs are inherently only suited to problems in which solutions can be expressed as a string of digits. In the case of a neural network, the parameter string will usually encode the synaptic weights, neuron threshold values and other properties of the network such as input gains, and dynamical properties of the individual neurons. Naturally, other problems also have solutions which can be expressed in a similar manner, for example the geometry of an aerofoil, the geometry and constituent materials of architectural structures, or the rules of an automated financial trading strategy. The method is conceptually simple and is summarised in Figure 2.2.

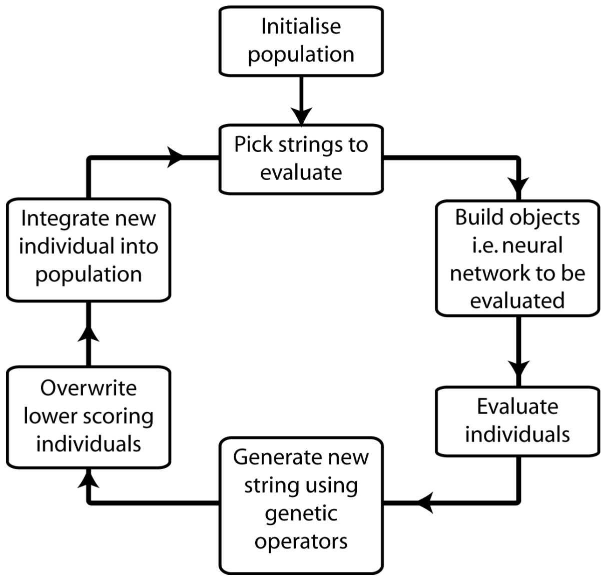

Initialise population
Pick strings to evaluate
Build objects i.e. neural network to be evaluated
Evaluate individuals
Generate new string using genetic operators
Overwrite lower scoring individuals
Integrate new individual into population

Figure 2.2 A graphical summary of the operation of a genetic algorithm

The first step in the use of a genetic algorithm is to generate a population of potential solutions, these are most often generated at random, but could also encode a known or hand designed solution as a starting point (Julstrom, 1994), (Vaughan, 2007). The method by which the population is initialised is worthy of some consideration, the random initialisation of every parameter within the network could be considered a purer implementation of the genetic algorithm as the operator's own preconceptions of how a solution to the problem might be achieved is not imposed upon the evolutionary process. It is not uncommon for evolved solutions to solve the problem in some unexpected way and it indeed might be this type of novel solution that the user of the GA is in fact aiming for. However, in problems where the encoded solution requires a large number of parameters, or if there is some kind of noise in the evaluation process it may be difficult to evolve even mediocre solutions using randomly initialised populations, in this case, it may be appropriate to seed the population with a hand coded solution that places a search process in a region of parameter space whereby even the new parameter sets produce behaviours of some validity that can subsequently be tuned by the evolutionary process.

Once the population is initialised an iterative cycle begins (see Figure 2.3), the first stage of which involves selecting some or all of the individual solutions to evaluate. Each individual string is read in order to build the structure, for example the neural network that will be used in the evaluation process. This is passed into an evaluation function which will score the fitness of each of these solutions according to predetermined criteria which are typically encoded in a fitness function. For example in the example of evolution of aerofoil design that was suggested earlier, the parameters may encode the physical parameters of the aerofoil such as its exact geometry and construction material parameters. The evaluation process will then give a score to the solution based on some predetermined criteria such as aerodynamic performance, build cost, weight or some combination of the three.

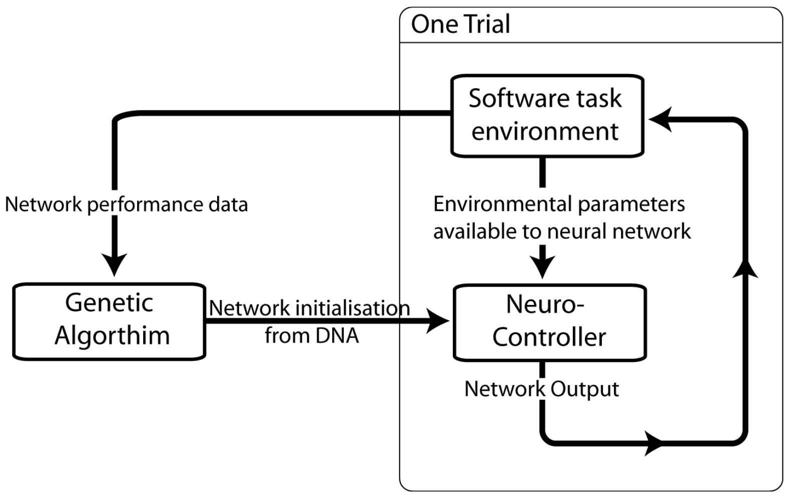

One Trial
Software task environment
Network performance data
Environmental parameters available to neural network
Genetic Algorthim
Network initialisation from DNA
Neuro-Controller
Network Output

Figure 2.3 A graphical summary of the trial structure when evolving neural networks for control tasks

Once the scores for the individuals to be tested have been derived, the status of each tested individual as a winner or loser can be assigned. This assignment can be deterministic, where the highest score is always determined to be the winner, or probabilistic where the probability of winning is proportional to the score. The loser's parameter string is then replaced with a new string that is generated using some combination of genetic operators. These operators must include mutation where either all, a subset, or a single parameter is subject to some random variation, usually a small additive value change. The mutation operator plays a crucial role in the generation of true novelty, allowing the evolutionary process to explore the space of possible solutions, beyond shuffling parameter values already extant within the population. Crossover, the other most common operator, is inspired by the recombination process of mammalian chromosomes. Using the crossover operator usually results in the swapping of contiguous sections of 'DNA' in the form of subsections of the parameter string.

The source of the DNA which is subject to the crossover process usually always includes some DNA from one or more tournament 'winner' or higher scoring individual parameter sets. In cases where this is exclusively the case and no DNA from the loser is retained the process could be thought to more closely resemble a mammalian evolutionary process. In these 'vertical' processes, successful parents i.e. the winners pass their DNA onto a subsequent generation. This need not necessarily be the case, and inspired by the 'horizontal' transfer of genes by plasmid loops in bacteria it is also possible to replace only a subset of the loser's DNA with novel DNA from a higher-scoring individual (Harvey, 2011),(Harvey & Tomko, 2010). In either case, the newly created string replaces the original DNA of the loser. The test, score and replace DNA cycle is allowed to iterate, typically for several hundred to several thousand cycles. As solutions with higher scoring traits are preferentially selected to persist within the population, these well scoring solutions will come to dominate the population. As the cycles proceed one therefore expects the fitness both of the best performing individuals and of the average performance across the population to increase over time.

This synopsis of genetic algorithms has introduced the main concepts behind the practical use of GAs; that a population of solutions is created, that each individual in the population is tested against some criteria that judges its fitness, and that high scoring individuals receive preference in passing their genes on to the next generation, in a repeated cycle of testing and mating/replacement. Below we go on to review how this technique is used to generate controllers for robots in simulation or in reality.

2.1.6 Applications of the evolutionary robotics paradigm

In this section we will review examples of evolutionary robotics implementations within the literature specifically where the focus is that of evolving active controllers, most often neural networks, for tasks where an agent is required to operate within spatial and temporal constraints. An excellent introduction to the field is available (Nolfi & Floreano, 2000), detailing the core concepts and typical methodologies of ER with a number of examples.

We will initially focus on implementations where the controller is only required to operate in a software environment. In these software-only applications, we can illustrate the diversity of motivations that drive the use of evolutionary robotics as a tool. To appreciate this breadth, one need only compare the exploration of the behaviour of neural networks in dynamical systems terms (Izquierdo & Buhrmann, 2008) which we will address first, with the highly directed and technically complex but philosophically straightforward implementation of evolved control of bipedal robot walking (Vaughan, 2007), and again to the more scientifically oriented replication of behavioural data originating from psychological (Rohde & Di Paolo, 2007) or developmental (Wood, 2007) studies.

It should be noted that ER is often used as a tool to explore a different, albeit often associated, domain. It is in this context that we shall first explore the use of ER, not for a real world robot control task, but for exploring the dynamical properties of continuous time recurrent neural networks. We will spend some time looking at an example of this kind of work in order to illustrate a typical use of evolutionary robotics as a scientific tool. Whilst worthwhile in its own right, this work also has some crucial insight to give us if we are to use evolutionary robotics for a real world control task, in which the consideration of real sensors and actuators and more complex physics can distract us from the fact that we are in fact generating dynamical systems. Knowing what kinds of dynamical properties these systems might exhibit and how we should seek to initialise the process in such a way as to create the least hindrance for the emergence of the type the dynamics that we believe to be appropriate is of no small value. From this high point of deploying ER as an assumption-free or at least assumption-light tool for exploring the mathematical properties of neural control systems of embodied robots we will move progressively towards studies that are more concerned with the actual deployment of evolutionary robotics as a tool.

There is a bewildering array of implementation specifics in evolutionary robotics not just in terms of parameter choices, but also in the way the GA is set up and managed in order to achieve a given end. This diversity is marred by a disappointing paucity of rigorous comparative studies that explore the advantages and different disadvantages of these implementation strategies on even one problem, let alone a reference suite of problems. However we can seek to gain some kind of consensus view on how best to implement the use of evolutionary robotics for a dynamical control problem with reference to the approaches that have been taken in order to build systems capable of complex behaviours and multiple behaviours. We will also sample the spread of opinion regarding whether networks should be deliberately modularised, or whether evolution should be given a completely free hand to create its own architecture in a monolithic network.

Continuing the trend from abstract towards applied uses of ER, we subsequently consider the techniques that have been utilised within the literature in order to bridge the gap between operating in software and operating in a real world environment. Finally we will look at evolved controllers specifically for the control of flight. This review will set into context the work presented within this thesis in terms of the state of the current knowledge within the field, and how the task resembles or is different to others in the literature, thus framing questions that are yet to be resolved.

2.1.6.1 ER in simulated tasks – the dynamical systems perspective

Regardless of the aim of the application of evolutionary robotics, a typical implementation will use a simulated task environment in which to test the relative fitness of the individual parameter sets that constitute the population. In this section we will look at a variety of simulated tasks in order to glean the common elements required in an implementation. We will start by reviewing some of the more theoretical applications of evolutionary robotics that at first it may appear to be far removed from practical robotics applications, but in fact provide useful information as to how to configure networks and initialise their parameters which are to be evolved and also gives some insight into how evolved controllers may solve tasks.

Many of the tasks that feature most frequently within the ER literature are ostensibly simple, often involving two wheeled robots operating in relatively static environments, or even more simple problems that are never intended to be implemented in the real world such as one legged walkers operating in a 2-D environment or agents that are capable only of moving along a one dimensional line. Such seemingly trivial tasks have however added greatly to the body of knowledge. One such example is that of Beer et al (R. D. Beer, 1995) where the focus is directed towards the understanding of the underlying dynamics of neural networks that constitute central pattern generators (CPGs). Beer subsequently used this model as a reference from which to determine how best to initialise the parameter sets within populations when evolving CTRNNs (Mathayomchan & Beer, 2002). CPGs are neural pattern generators that are used in many species to generate rhythmic motions or actions, well-known examples including that of locomotion in the lamprey (Ijspeert & Kodjabachian, 1999) and feeding in snail species such as *Aplysia* (Kandel, Schwartz, & Jessell, 2000). It is noted that these biological central pattern generators are notoriously diverse with few apparent evolutionary restrictions on how a repeated rhythm may be generated. Beer et al used a highly simplified task in which a single legged agent is capable of changing the relative angle of a single leg to its body by controlling the relative torque applied to two antagonistic actuators, and by raising and lowering the leg with a third actuator (R. D. Beer, Chiel, & Gallagher, 1999),(Chiel, Beer, & Gallagher, 1999). If the agent sweeps the leg from a forward position relative to its body to a backwards position relative to its body whilst its foot is in contact with the ground, then it will be propelled forwards, if it then lifts the leg and returns it to the starting position a cycle can be considered to have been completed.

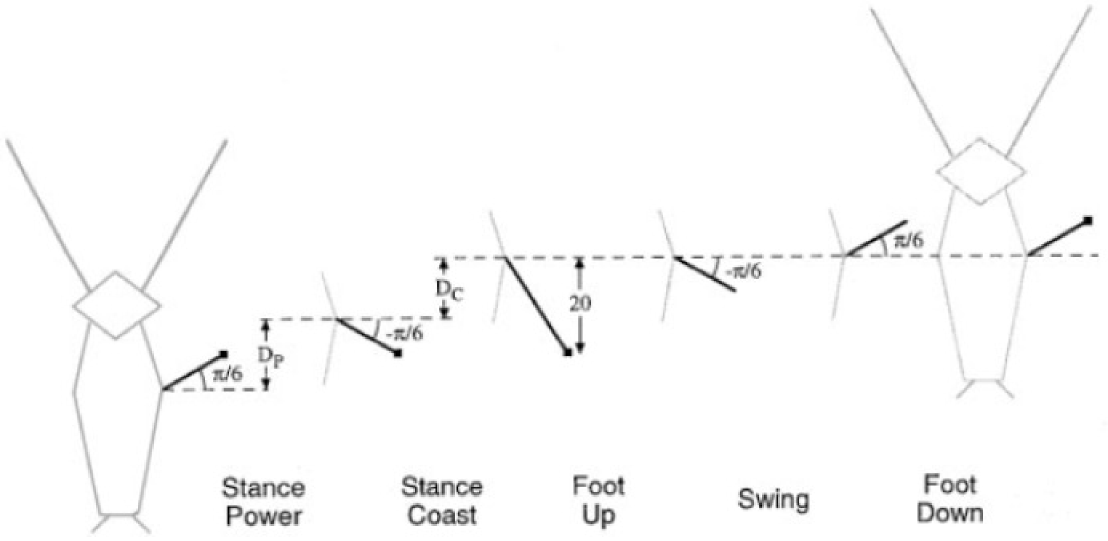


Figure 2.4 An optimal walking cycle in Beer et al's CPG task (R. D. Beer et al., 1999).

Repeating this cycle in a rhythmic manner will cause the agent to progress forwards in the space of the simulated environment thereby rendering it a problem highly amenable to the use of CPGs as a solution. One compelling argument for the use of such an abstract task which omits so much detail from the real world is that the computational demands are exceptionally low. This means that evolved solutions can be generated very fast and a large number of evolutionary runs can be completed allowing a statistical analysis of the properties of the resulting neurocontrollers to be performed. Another argument for the use of such models is that it is relatively trivial to derive an optimal behaviour with which to compare the controllers against, such an optimal cycle is shown in Figure 2.4. This is possible not only because of the simplified biomechanics of the agent but also because both the environment and the agent itself are subject to zero variation between trials or indeed within the trial itself. The neural networks are small, fully interconnected CTRNNS of three, four or five neurons that receive no inputs, either proprioceptive or from their environment. Three of the neurons are taken up controlling the effectors as described above and the larger networks have either one or two interneurons respectively.

Eq. 2-3

$$\tau_i y_i = -y_i + \sum_{j=1}^{N} w_{ji}\sigma(y_j + \theta_j) \qquad i = 1, \dots, N$$

The dynamics of the CTRNNs are governed by Eq. 2-3, where τ_i is the time constant of the ith neuron, y_i is the internal activity of the ith neuron, w_{ji} is the weight or connection strength from the *j*th to the *i*th neuron, Θ_j is the bias or threshold term of the *j*th neuron and σ is the logistic output or sigmoid function as per Eq. 2-4.

Eq. 2-4

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Networks were evolved using a public domain genetic algorithm (GA) package called GAuscd. Fitness of the agents was simply calculated as the square of the difference between the maximum distance that it was possible to travel in the given time and the distant achieved by the agent during the trial.

It was found that three, four, and five neuron networks were all capable of evolving satisfactory walking behaviours, however performance was seen to increase with the number of neurons, although this increase was not significant between four and five neuron networks. Because of the simplicity of the task, the evolved controllers can be compared very precisely with the optimal control strategy and the lower performance of the smaller networks can be explained in terms of the dynamical properties of the constituent neurons. The authors introduce the concept of a *dynamical module* which are collections of neurons that perform a transition from one quasi stable output state or attractor to another while the other neurons are effectively locked into an active or inactive state and therefore are not currently dynamic themselves. They go on to link the sequential switching of the state of the network and hence phases of motion within the pattern to two, different possible configurations of these dynamical modules.

Whilst it seems to be relatively trivial to identify these dynamical modules in the smaller networks, the number of potential dynamical modules increases exponentially along with the network size. Where only 21 dynamical modules are achievable in the three neuron network, 1082 are possible in the four neural network and 72,950 are possible in the five neuron network. These dynamical modules are themselves capable of being dynamic in that their modular structure is itself subject to alteration over the pattern cycle and the proportion of these dynamical dynamic modules with respect to the total number of the possible dynamic modules also increases with network size. These attributes coupled with a biomechanical degeneracy which allows very different internal neuron activations to achieve the same motor output due to limits of the agents body, allows a remarkable degree of freedom or variation in the parameters of the network that are capable of producing valid controllers.

While this might appear to lead us back to the biological starting point where we are faced with enormous diversity, this type of work has some concrete outputs. Firstly the network size is important in determining performance and the dynamical structure of the solution. Secondly, it is possible to analyse at least smaller networks in terms of the switching of stable states of components within the network that can be thought of as modules. However the limitations of this approach for more complex problems as opposed to an investigation of small network dynamics are apparent: a highly simplified biomechanics were necessary in order to make the required analysis, the neurocontrollers had no external input and existed within a static environment. Additional complexity might have precluded this in-depth analysis and finally, it is only possible to achieve meaningful analysis of small networks. Still this approach does have a clear utility within its intended domains and in related work (Mathayomchan & Beer, 2002) using the same simulated task the authors are able to demonstrate how seeding the initial population of an evolutionary search with particularly dynamic CTRNNs is able to increase both the average and best performance of a given evolutionary run. This is achieved by first generating a set of network weights, and then subsequently calculating the neuron biases in the initial population by Eq. 2-5.

Eq. 2-5

$$\theta_i = -\sum_{j=1}^{N} w_{ji} / 2$$

Assuming that all inputs across the set of weights have the same typical ranges, setting the threshold in this way avoids saturating the activity of each neuron at its lower or upper extremes. This can occur due to the shape of the sigmoid function where further increase or decrease respectively of the neuron input results in little or no change of the output of the neuron (see Figure 2.5).

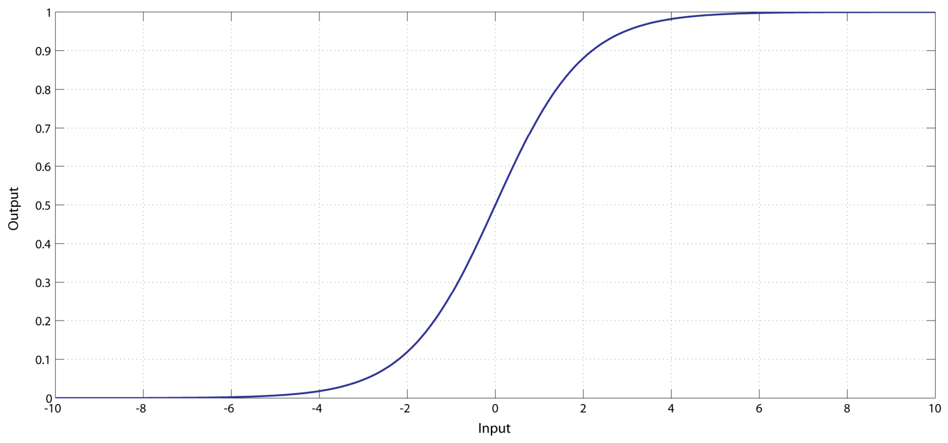


Figure 2.5 A plot of the input/output relationship for the sigmoid activation function.

By matching the threshold to the strengths of inputs weights the neuron's internal activity can be maintained such that it falls within the dynamic portion of the sigmoid curve. It should be noted that this approach may have limitations where it is not possible to scale external inputs between zero and one because the exact range of the external input may not be known in advance. However it does illustrate the power of the dynamical systems approach in producing well grounded assertions regarding the use of otherwise difficult to analyse networks.

At the most abstract end of the spectrum, autonomous CTRNN's with no dynamic input can be generated randomly without applying them to any task in order to investigate the dynamical properties of small neural networks in isolation. Such a study (R. D. Beer, 1995) into the precise dynamical properties of very small networks consisting of one and two neurons with static external inputs I_i that are independent of network output has demonstrated that stable and unstable equilibrium points within the dynamical space can be described and a surprising range of dynamical regimes may be produced. These regimes may tend towards stable equilibrium points, contain unstable equilibrium points such as saddle nodes, perturbation from which will lead to rapidly diverging patterns of activity, or may contain a limit cycle in which rhythmical activation patterns are witnessed. Depending on the configuration of weights and biases, it is possible for a number of dynamical features to coexist within even a small two neuron network.

Moving back from purely theoretical studies towards integrating the analytical dynamical systems approach into task centred agent/environment interactions, Beer proposes a set of 'minimally cognitive' tasks that are deliberately designed to be the simplest possible type of agent environment interactions that could be described as being cognitively interesting (R. Beer, 1996). This implies that unlike the above studies, there needs to be an active participation of the agent with its environment. Three tasks are proposed all of which involve a visually guided robot (see Figure 2.1), and in two of which, which we will briefly review here, the agent is able to move in two directions along a 1d line.

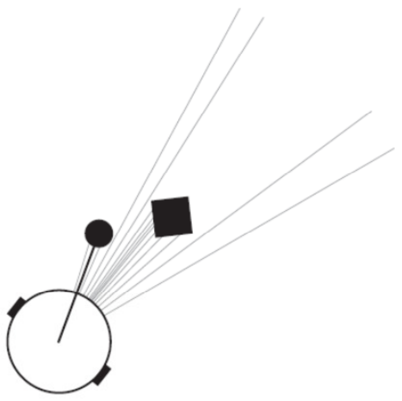


Figure 2.6 The visually guided agent presented in (R. Beer, 1996) **and the basis for a number of other studies e.g.** (Fine & Di Paolo, 2007), (Di Paolo, 2000).

The agent is a circular Khepera style robot with two actuators which is capable of exerting either rotational torque or exerting a force in line with the path from the robot centre to the actuator, depending on the experiment. It is also equipped with either five or seven 'visual' sensors that are in fact more akin to infrared distance sensors. These sensors send out virtual rays in a fan shape, and generate inputs to the neural network that are inversely proportional to the distance to the first object that is intersected by the ray. In some instantiations, there is additionally an arm with a manipulator which may be opaque or transparent to the rays emitted by the virtual sensors. Three tasks are described, in the first the agent is required to move along a one dimensional line in order to align itself with a circular object which is travelling through the task arena at a given vertical and horizontal speed which is static within a given object presentation but varies between presentations. It is found that a feedforward networks without recurrency can be evolved to align themselves with the falling object. However they generalise poorly from a training set to an evaluation set, specifically when an object is lost from their field of view, they are unable to continue acting in such a way as to regain the object. This is to be expected because as the networks have no state, they are purely reactive hence when there is no input signal to react to, they are in effect paralysed. When self connections and recurrent connections are permitted, the evolved networks are capable of continuing a behaviour when the object is lost from the field of view, in most cases regaining objects that are lost from the visual field resulting in far superior generalisation performance (R. Beer, 1996).

In a second task in the same paper (R. Beer, 1996), the agent is required to align itself with circular objects that are presented in a similar manner as to the first task, however they must avoid diamond shaped objects with the same width as the diameter of the circle in one case and avoid a line with the same length as the diameter of the circle in a second case. Here bilateral symmetry is forced upon the networks in the evolutionary process and it is found in this task it is much more difficult to evolve successful agents in terms of the number of generations required, even though there is no variation in the parameters of the objects or the parameters of the agent itself between trials. A typical strategy that emerges for agents evolved to discriminate between the circle and the diamond is for the agents to foveate the object by aligning itself with it, after foveation the agent initiates an active scanning whereby it moves left and right as the object approaches. After this active scanning process the agent appears to make a decision and either retains alignment with the agent in case of the circle or moves away from it in case of the diamond. The quality or nature of the inputs that the agent receives appears to be important because in the case where the agent is evolved to discriminate between the circle and the line, an alternative strategy is employed whereby the agent antifoveates the object that is falling i.e. it aligns itself relative to the object such that the object barely meets the periphery of the agents field of view. A similar active scanning process is initiated before a decision is made. Presumably this difference in strategy is determined by the shape of the objects and reflects where the maximum information is to be obtained that differentiates the two shapes.

Interestingly, the author refrains from the in-depth dynamical analysis performed in the papers mentioned previously, although whether this is because of space restrictions or because of inherent difficulty in the analysis of the behaviour is unclear, in-depth dynamical analyses of the discrimination task are in fact available elsewhere (Izquierdo, 2008). Nevertheless there are some key lessons to be learned from this work that do apply to more complex tasks, the first is that it is important to ensure that the network neurons are operating within their dynamic range. Secondly one might expect active strategies to be employed by the network whereby an agent will act in such a way as to maximise the information available to it in a given environment in a way that is not necessarily directly linked to the scoring scheme within its fitness evaluation. Finally it is clear that feedforward networks as suggested in the discussion above have extreme limitations as they are unable to autonomously generate behaviours without external inputs.

We have sought to illustrate in this section how evolutionary robotics techniques can be used for the exploration of the dynamical properties of evolved continuous time recurrent neural networks. We have reviewed how the functionality of evolved neural networks can be explained in terms of the dynamical landscape that is created by the agent's interactions with its body and potentially its environment. We have made it clear that this analytical approach works well in small networks yet may not scale to larger networks where the dimensionality of the dynamical space is very high. However, we have sought to illustrate that there are important insights to be gained from looking at this approach if we are to initialise the evolutionary search process in such a way as to maximise the performance of the evolved agents, specifically that the network architecture should allow for internal dynamics and that the size of the network will affect the both the performance and potential complexity of the evolved solutions.

2.1.6.2 ER in simulated tasks – ER centred perspective

In this section, we will seek to introduce a number of the considerations that a practical implementation of evolutionary robotics must consider, by looking at a number of examples within the literature. Such considerations will include how the network is encoded, how much the architecture of the network is constrained, the design of fitness functions to achieve a given behaviour and the use of shaping schemes in which agents are presented with a sequence of tasks or fitness functions.

2.1.6.2.1 Genome encoding techniques in ER

Encoding of the genome is one critical aspect of any evolutionary robotics implementation. In prior examples, the weights and the time constants of the network are encoded directly as values in the genome. However this is not necessarily the most appropriate solution. If we consider a robot with multiple repeated elements, such as segments in a snake or legs on a hexapod robot, it would place an unnecessary burden on the evolutionary process to encode the entire network from scratch when the user of the GA has the prior knowledge that there are repeated structures. In this case it would be advantageous to simply evolve the connections of a common repeated subunit of a network and then any inter-unit nodes or connections that can then co-ordinate the activity of the subunits in a manner appropriate to the task. In real biological systems, there is a complex developmental cascade involving both volume and cell surface signalling mechanisms between progenitor cells which become identified as progressively more and more specialised cells as they divide and move in the developing organism (Sanes, 2000). Whilst biologically realistic models of these processes have been built e.g. (Kerszberg & Changeux, 1998), in the ER literature the tendency is to use highly abstract models that generate networks according to a set of rules which then become the objects of evolution.

An early example of the application of a meta-level of rules or a genetic encoding that might be described as a developmental grammar was applied to the evolution of neural networks to the benchmark N-M-N encoder/decoder problem (Kitano, 1990). In this problem the goal is to reproduce a binary input vector of length N passed to an N node input layer of a 3 layer neural network at its output layer. Kitano used a modification of a Lindenmayer system - an iterative pattern generation grammar and demonstrated that for 16 and 32 node networks (4-8-4 and 8-16-8) evolution of the grammar rules outperformed a basic strategy where connectivity of each node was encoded separately within the genome. In other early work Mjolsness et al described the evolution of recursive rules sets for generating connectivity matrices, which, though further removed from analogy to biological growth were able to produce continuous real valued genomes(Mjolsness, Sharp, & Alpert, 1989).. They also reported superior performance when the network size and task complexity of a static benchmark problem were scaled up. In work evolving neurocontrollers for control of a Khepra robot the genotype encoded axon growth and branching rules which was used to encode the network architecture, through a morphogenesis inspired genotype to phenotype mapping, whilst neuron biases were encoded as more conventional direct mappings (Nolfi, Miglino, & Parisi, 1994).

On another situated robotics problem, that of evolving CTRNNs for the control of a simulated 6-legged walking agent (Gruau, 1995), the author described a graph grammar technique which iterated around the neuronal population and built up links according to evolved rules. Although performance was comparable to hand-coded networks and networks evolved by individual connection, the author pointed out that the resulting networks' modularity rendered them more comprehensible and would potentially offer benefits when evolving more complex sensory motor mappings.

One piece of work in which a similar approach is taken is that of Ijspeert (Ijspeert & Kodjabachian, 1999), in which ER is used to evolve neural CPGs for the control of swimming of an artificial lamprey, a primitive type of fish that uses rhythmic undulations of its body to propel itself through water. The encoding scheme used is that proposed by Kodjabachian and Mayer (Kodjabachian & Meyer, 1998),(Meyer, 1998) in which a geometry based encoding scheme called SGOCE is described. Under this scheme, a developmental program is encoded which starts from a predetermined minimal architecture, from which a network is built using commands for the addition and deletion of neurons and their connection. In the initial work a neurocontroller for a 6 legged walker is encoded (Kodjabachian & Meyer, 1998),(Meyer, 1998) but in the Lamprey study, 18 repetitions of the subnetwork are used. The authors show that evolution is able to produce networks that are capable of straight ahead and turning motion, which can be controlled by inputs that specify a desired rate of motion and strength and direction of turning. A similar approach to the encoding of the network was taken in an earlier study by Jakobi and Quinn in which networks are also generated spatially from a set of rules (Jakobi & Quinn, 1998) for the control of a two-wheeled Khepera robot which although lacking in repeated segments is bilaterally symmetric. They argue that a spatial approach allows the use of crossover as a genetic operator whereas GA implementations where the encoding string varies in length are inherently unsuited to crossover. This is because new parameters added into expanded networks will cause an offset mismatch if inserted into a genome that does not have the new values already in the string. This might cause a bias value from one genome to be overwritten with a weight value from another genome for example. The solution proposed in (Jakobi & Quinn, 1998) is to use a scheme which encodes a spatial position for each node and a set of node 'links'. The links specify axonal projections from each neuron via a distance and direction in a 2D network space, a connection is then made to the nearest neuron to the point defined by these coordinates. This is reminiscent of the work metioned earlier which encoded axonal branching angles and neuronal coordinates (Nolfi et al., 1994).

Yet another approach is taken in the work of Vaughan (Vaughan, 2007), in which ER is deployed for the control of bipedal robots whose joint structure and proportions are deliberately designed to mimic human anatomy. In this work, a parametric L-system is used to encode both the network architecture and parameters and the parameters of the robot body. This approach is similar to the previous work, in that the network is laid out spatially and a sequence of neuron replications and axon growths are performed. In this work the stated motivation behind the use of this tool is to increase the number of neutral networks within the evolutionary parameter space, although this assertion is never explored in a systematic manner within the study. Another study (Taylor & Massey, 2001) in which both the morphology of the agent and the control system are encoded and evolved together uses a similar approach. In that case the genetic description is in the form of a nested directed graph and the stated motivation is that such a description is more likely to lead to the repetition of body parts.

The aim of this brief review of encoding mechanisms was to illustrate the diversity of active schemes of encoding and the relationship between the necessity for the creation of repeated structures or units and the choice of encoding scheme. The examples above have illustrated the demand to consider how best to implement an encoding if the network size is subject to change over the course of evolution. Secondly, we have shown how it is common in the literature to employ a spatial or higher level structural description to be subject to evolution rather than a set of parameters in which the structure is purely implicit. It appears that in problems where there is a clear symmetry of body parts, sensors and actuators it might be the most parsimonious solution to use a high-level structural descriptions similar to those described above. Additionally these techniques also allow for open ended evolution in terms of growth and deletion of various subcomponents within the neural network. It is possible that these techniques might canalise or constrain the evolutionary process in a way that may or may not be desirable, although the impact on the evolutionary process of the use of such methods has not been investigated in depth.

The lack of a body of work within the literature comparing encoding schemes on robot control tasks makes an absolute recommendation for a particular technique impossible and we therefore rely on the examples within the literature as guidelines.

2.1.6.2.2 Imposition or emergence of modularity within ER

Some tasks seem to naturally decompose themselves into subtasks when analysed from the point of view of a human designer, yet other tasks rely on separate stages or behaviours which appear to be clearly delineated. There is therefore a question of whether an evolved neural network should be restricted specifically to a modular structure in order to best perform either multiple behaviours or constituent aspects of a single behaviour, or whether evolution is entirely left to select the type of network which performs best. We review the relevant positions here to allow an informed decision, as this issue could have some bearing in an applied task of some difficulty such as we are addressing.

There are a number of reports within the literature reporting improved performance when modularity is enforced i.e.(De Nardi, Togelius, Holland, & Lucas, 2006), however these are conflicted by others reporting excellent performance by monolithic networks in tasks where intuition might suggest that modularity would prove advantageous. One example (Izquierdo & Buhrmann, 2008) of the latter is work related to the dynamical analysis of evolved neural networks that we considered earlier. In this study a neural network is evolved in which a single circuit performs to qualitatively different behaviours, that of rhythmic control of locomotion in the same single-legged walking task that was discussed earlier (R. D. Beer, 1995). For the second behaviour, a two wheeled Khepera-like robotic agent is required to perform a chemotaxis task. It is shown in this study that even a small CTRNN is capable of producing both of these behaviours without an external 'switching' signal, purely by changes in the dynamical interaction of the neural controller and the two different robot morphologies. The importance of the physical instantiation or embodied nature of the robot is a topic we will return to later. Another study, in which the relationship between a spatial encoding of the network and modularisation is investigated, found that although a spatial encoding was liable to construct more spatially modular networks, these networks were more difficult to evolve to a given level of performance than their unconstrained counterparts (Fine, Di Paolo, & Philippides, 2006).

Other results point in the opposite direction, Yamauchi and Beer found it necessary to evolve separate neural networks that would function as separate neural modules to perform a task involving a sequence of actions where it was found that without such decomposition, a viable solution could not be evolved (Yamauchi & Beer, 1994). Togelius in particular has argued for the modularisation of neural networks, inspired partially by the subsumption architecture of Brooks (Brooks, 1986),(Brooks, 1991). Togelius describes a method of sequential evolution of a number of neural network layers which are specifically designed to accomplish sub elements of a complex control task as decomposed according to perceived functional separation by the designer (Togelius, 2004). One supporting argument he provides for this approach is simply related to the number of network parameters in fully connected monolithic networks scaling exponentially as the network size increases. Such connectivity thus requires the evolution of many more parameters than a series of modular networks with only limited interconnection, in the latter case the number of weights to evolve are fewer without reducing the number of neurons that are available to contribute to the performance of the task. He presents results for both a conditional phototaxis and an obstacle avoidance task in a Khepera like robot (Togelius, 2004), and with De Nardi (De Nardi et al., 2006) in a waypoint following control task in a simulated helicopter control scenario, in which it is claimed that functional modularisation allows faster evolution and in some cases is required for evolution of valid solutions. Notably the networks used in both these studies are all feedforward state-free neural networks, negating the possibility for modules to be effectively formed by the networks internal dynamics.

It should be noted that apparent spatial modularity may not correspond directly to a functional decomposition of task performance between spatial modules, nor does the lack of an imposed modularity negate the possibility of the network functioning in a modular fashion. Beer’s work emphasises the importance of dynamical modules, which may be temporally transient in nature, and Watson et al warned against neglecting the importance of neural dynamics in assessing network behaviour (Watson, Reil, & Pollack, 2000). Certainly, Izhikevich has demonstrated the formation of persistent structures in the form of highly conserved patterns of relative spike timings in groups of non-spatially related neurons (Izhikevich, 2006), which emerge from the dynamical properties of large groups of spiking neurons. Conversely, it is clear that there is modular task specialisation in animal brains, well-characterised in such structures as the mammalian visual cortex in which hierarchical connectivity and functional separation is clearly apparent (Kandel et al., 2000).

In assessing the diverse perspectives on the benefits or otherwise of modularisation within the literature, it seems that there is not currently a clear consensus about the best practice for applied ER. This being the case, the prudent view to take in a task in which performance is the most important criteria, is an open-minded willingness to adopt the approach that best suits the task in hand. Starting with the simplest approach and implementing supplementary schemes only when necessary might be an appropriate way to proceed.

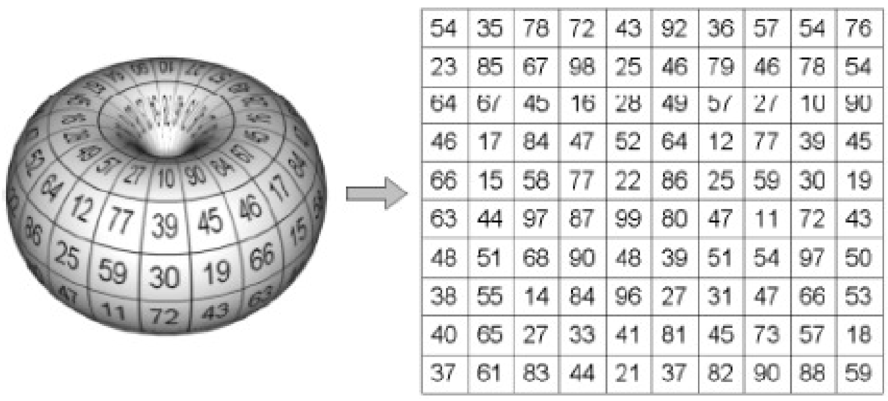
2.1.6.2.3 Which flavour of GA is appropriate for a given problem?

A further set of considerations concern the operation of the genetic algorithm itself. Once again, there are few examples in the literature comparing exact details of how the GA is operated and the resulting performance of evolved solutions on a number of tasks representing different classes of problem. In fact, the opposite seems to be true and various flavours of genetic algorithm are deployed often without justification, with recourse either to some vague notion of population dynamics or simply aesthetic appeal. While it seems that a rigorous comparison of these approaches in the context of ER is long overdue, the scale of that goal is not inconsiderable and certainly beyond the scope of this work. Motivated by performance considerations, we will later perform some basic comparatory studies and therefore we will briefly touch upon a few of the options here.

One common addition to the basic genetic algorithm as described above is the addition of some concept of space in the population. Early work explored the use of island subpopulations in which evolution proceeded separately in parallel, albeit with gradual or periodic leakage between populations, (Cantú-Paz, 1998) provides a useful summary of such approaches.

Alternatively a single population may be distributed over some notional space, in which competition and mating within the GA occurs preferentially between comparative neighbours. Being preferable in terms of modelling of natural evolution, and appealing in offering the promise of more diverse populations local mating schemes have been proposed and implemented for many years. These were soon demonstrated to produce superior performance in terms of the speed of evolution and quality of solutions on at least some classes of optimisation in addition to preserving diversity, at least where multiple solutions of equivalent fitness existed (Collins & Jefferson, 1991).

The simplest approach is to assign each member of the population a position on a 1D line or torus (Spector, 2005). Upon selecting individuals for comparison within tournament based selection, individuals are then selected within a limited geographical region or neighbourhood relative to one another. The motivation behind this is to maintain some kind of diversity within the genetic population. Other work extends the 1D distribution of population onto a 2-D plane (see Figure 2.7), the concept of neighbourhood thereby denoting a radius around the first selected individual from which potential mates or competitors are selected, and this approach is taken often within the evolutionary robotics literature e.g. (Vaughan, 2007),(Shim & Kim, 2006) (Shim, Kim, & Kim, 2004).



54	35	78	72	43	92	36	57	54	76
23	85	67	98	25	46	79	46	78	54
64	67	45	16	28	49	57	27	10	90
46	17	84	47	52	64	12	77	39	45
66	15	58	77	22	86	25	59	30	19
63	44	97	87	99	80	47	11	72	43
48	51	68	90	48	39	51	54	97	50
38	55	14	84	96	27	31	47	66	53
40	65	27	33	41	81	45	73	57	18
37	61	83	44	21	37	82	90	88	59

Figure 2.7 Image from (Shim & Kim, 2006) **illustrating the 2D spatial distribution of population members. The plane is wrapped into a torus to prevent edge effects.**

2.1.6.2.4 Parameter choice and managing the evolutionary process in complex tasks or multiple behaviours

It is not unreasonable to suppose that in demanding applications in real-world robotics, such as multiple behaviours, complex sequential tasks or tasks with a high degree of variability, that careful attention must be paid to how the evolutionary robotics process is implemented, specifically in terms of variables that are to be subject to evolution, choices of network type and parameter ranges and in how the evolutionary process is managed. In this section we will review some work that should provide some insight into these decisions.

Some studies such as (Husbands, Smith, Jakobi, & O'Shea, 1998), and (T. Smith, Husbands, & O'Shea, 2003) in the latter of which the main thrust is an investigation into fitness landscapes when evolving neural networks, use gaseous neuromodulation to modulate the slope of a transfer function. A useful comparative investigation (Pratley, 2005) inspired by this approach allows a parameter for the transfer function slope to be evolved for each neural network being used on a simulated legged locomotion task. Networks evolved with this extra parameter were shown to be sometimes faster to evolve and achieve higher performance than networks where the slope is fixed, but this did depend on the activation function used. Further compared in this study is the use of tanh as a transfer function versus the standard logistic sigmoid function that is most commonly used within the literature, the conclusion being that networks using tanh are generally faster to evolve and converge to higher levels of performance. It is possible that this improvement in performance is simply due to the fact that the tanh function scales from -1 to 1 whereas the logistic sigmoid scales from 0 to 1. This means that at the most dynamic point in the range of the transfer function, the tanh neurons pass from outputting a negative signal to a positive one, this flip in the sign of the output is likely to contribute towards oscillatory dynamics in the network, a characteristic that might be useful in many problems.

Regardless of the selection of parameters which are to be evolved, consideration must be given to the way in which the evolutionary process is managed. Often, in a demanding task evolution is performed in stages with the task being decomposed into two or three key elements. Under such a scheme, performance at the lower stages of difficulty must reach a predetermined threshold for additional elements to be implemented within the fitness scoring or in the task simulation. Such meta- management of the evolutionary process is often termed incremental evolution or a shaping scheme. The concept can be seen in subsumption architecture (Brooks, 1991) in which the intention is to train progressively more complex behaviours in a behavioural hierarchy in such a way that higher level coordination behaviours rely on previously trained primitive behaviours such as object avoidance. Researchers investigating or inspired by postnatal biological developmental processes have also long been familiar with the value and potential necessity of shaping or scaffolding an agent's interaction with its environment (Rutkowska, 1994),(Lee, Meng, & Chao, 2007). A shaping scheme approach was employed in an autonomous robotics context in the training of robot neurocontrollers using reinforcement learning (Dorigo & Columbetti, 1994). In this case shaping referred to the changes in the manner in which the critic was providing reward or penalty to the agent during the learning process, such as to sequentially train multiple behaviours. Closer to the manner in which shaping schemes are used in this work was a study in which evolution progressed on real miniature robotics platform in a desktop arena (Urzelai, Floreano, Dorigo, & Colombetti, 1998). Here the neurocontrollers were initially evolved to navigate an arena whilst avoiding its walls, after which a second task stage was initiated in which the agent was required to use a gripper to lift objects and transport them outside the arena.

A clear recent example in which shaping schemes were deployed is a task whereby a two wheeled agent is required to adapt to the shifting of position of its light sensor from the front of the agent to the back without any extra signal being given to the robot during a phototaxis task (Fine & Di Paolo, 2007). In this study agents are initially evolved to perform simple phototaxis before the task is modified to allow the movement of the light sensor, and although no comparisons made between direct evolution at the most difficult level and evolving incrementally, the authors suggest that evolution would struggle to produce valid solutions were the evolution not performed incrementally. Some performance difference was noted by Cliff et al (Cliff, Harvey, & Husbands, 1992) who noted that real robots incrementally evolved to orient to progressively smaller targets had a better generalisation performance than neurocontrollers evolved directly on the smallest target. Similar approaches have been applied to single behaviours with some success, typically the norm is for only one stage of difficulty to be passed through, often where the fitness function is modified in order to emphasise a further demand of the task designer (Reil & Husbands, 2002). Bongard presents a systematic approach to incremental evolution which he terms 'behaviour chaining' (Bongard, 2008) and applies it to the evolution of a robotic agent in a simulated environment which is required to walk towards an object and then lift it on to its back. Interestingly, as well as successive increments through the four subcomponents of the task if a performance threshold is exceeded, Bongard also allows the complexity of the task to be reduced if evolution fails to proceed through the levels of task decomposition, thereby reducing the risk of ‘dead ends’ of evolutionary stagnation.

From this brief review, it appears that the management of the evolutionary process must be considered carefully according to the demands of the task particularly in complex or challenging control tasks. It may well be that a more finely grained approach is necessary for the most demanding of tasks.

Another issue arising from the literature is at what point in parameter space to start the evolutionary process. Typically in evolutionary robotics the ranges in which parameters may be set are given some consideration, but the initial population is generally fully randomised at the beginning of a run. Vaughan notes that in a highly complex control task, that of controlling the gait of a passive dynamic walker, evolution should be provided with what he terms a 'seed attractor' for the majority of runs to produce viable solutions (Vaughan, 2007). This is a hand coded approximate solution that in itself does not produce the behaviour at the level of performance that is required, however it does start evolutionary process in a region of parameter space that is at least likely to converge to a good solution and not be trapped in a local optimum. Indeed, it is found that evolutionary runs initialised in this way produced viable walking behaviour in 100% of cases. An intermediate solution is to for a human operator to select a genome with which to seed the initial population by picking a winner demonstrating some interesting task relevant behavioural characteristics after watching the performance of an individual from a population after a few generations of evolution, and this approach has been shown to be viable (Harvey, Cliff, & Husbands, 1994).

Hopefully it is clear from this broad review of evolutionary robotics applications to simulated tasks, that thought must be given to the way in which an evolved neural controller is going to solve a particular task. This will allow appropriate parameter value ranges to be selected, an appropriate choice of which parameters to evolve to be made, and for it to be decided whether to allow evolution a free hand, or to constrain the evolutionary process by using a seed or by enforcing the network to conform to various types of modularity. We have also highlighted the role of structuring the task such that evolution is capable of solving complex or demanding behaviours. We have shown that whilst many such issues have been investigated, formal or comprehensive relationships between task, simulation, network and evolution remain absent from the literature. In the application of these techniques to a given task, it is therefore necessary to bear in mind the solutions that have proved successful in overcoming common problems, if we are to achieve acceptable performance in particularly demanding real-world control applications.

2.1.6.3 Evolutionary robotics in real world tasks

Whilst the above consideration of evolutionary robotics implementations in simulated tasks is certainly valuable, in our target application we have the additional requirement to perform a task on a real robot in the real world. In this section we will review the key ideas in evolutionary robotics literature that deal with this subject. Key considerations will be issues of matching simulated tasks to their real-world counterparts, whether to implement explicit learning algorithms and the importance of body dynamics.

In an ideal world, we would simply evolve the behaviour to be performed directly on the robot, in reality. That would spare us the burden of creating a simulation of the robot and its interaction with the environment, truly using the world as its own best model. Unfortunately, practical considerations render this approach rarely applicable. Real robots need power supplies, their servos wear out, failure to correctly complete a task may lead to damage or even destruction of the robot, or the evolutionary process may simply require too much time to converge to a useful solution. Therefore the majority of work in the field performs all or a significant amount of evolution *in silico*, i.e. in a simulated test environment. Providing that the simulation runs faster than real-time, the evolutionary process can potentially be speeded up, in the best cases allowing years of task time to be completed in a matter of hours or days. In silico evolution has the additional advantage that we have complete control over the task environment in a way that is impossible in the real world, for example in a flight control task we can control the strength and variability of the wind, or in a phototaxis task we can reset the position of the agent and any target lighting instantaneously without the need for any physical intervention. However this speed and flexibility comes at a cost, controllers that are evolved in simulation are not guaranteed to transfer their behaviours successfully into the real world. This potentially problematic transfer from simulation to reality is often referred to as “crossing the reality gap”. This phrase seems to imply that there is something missing from the simulation that is present in the real world, although it is also possible that it is something that is present in the simulation but absent from the real world that is causing the problem.

Before addressing the issues of crossing the reality gap in detail we will initially consider briefly some work in which evolution is performed on the robots themselves, which usefully highlights another issue in evolutionary robotics for hardware control, which is whether to implement specific explicit learning mechanisms that modify synaptic weights according to pre-and post-synaptic neuron activity using a Hebbian inspired weight update rule.

2.1.6.3.1 Real world evolution and perspective on learning in novel or changing environments

Floreano et al evolved neurocontrollers for a two wheeled Khepera robot in a simple maze task (D Floreano & Mondada, 1996). Evaluation of each individual was performed on the real robot system in real-time. Each neurocontroller was a small feedforward network consisting of three neurons. Interestingly no attempt was made to encode the synaptic weights, in fact the weights were randomised at the beginning of each trial. The parameters that were evolved were the rate of learning at each synapse, the type of learning rule employed from a selection of four variants of Hebbian style weight change rules and a parameter which encoded whether the synapse could drive a directional change in the output motor or merely modulate the amplitude of the signal. It was found that such an approach could evolve agents that would follow a smooth path around the trial area, the path becoming less subject to adjustments as the trial progressed in time. By restricting the exposure of the agent to its environment it was demonstrated that sensory feedback from the environment deriving from interaction between the agent and its environment was essential in the formation of the behaviours necessary to complete the task. As might be expected, the agents displayed notable variation in performance when initiated in the environment indicating that small parameter changes could result in large performance deviations. Clearly this approach would be problematic in tasks where failure in the real world is not acceptable due to safety concerns or potential damage being incurred to the robot. However, related work (Urzelai & Floreano, 2001),(D Floreano & Urzelai, 2001) has shown that adaptive synapses in which evolution encodes and selects between the same four learning rules as in the initial work demonstrated higher performance both in simulation and on real robots compared to agents where static synaptic weights were evolved. This implies that incorporating some learning mechanism is advantageous, although adaptation is not guaranteed to be a rapid process. This aspect was well illustrated by Di Paulo, who showed that in an experiment where Khepera-like agents were evolved in simulation to adapt to radical sensory disruptions in the form of visual inversion (Di Paolo, 2000), agents were in fact able to adapt using the same four variations on Hebbian synaptic learning as presented in the prior works, however adaptation was only possible after a long period of maladaptation.

Whilst such explicit learning mechanisms are attractive due to their link to neurobiological evidence, it has been demonstrated repeatedly that explicit synaptic learning mechanisms are not the only way that learning can be implemented in an evolved neurocontroller (Phattanasri, Chiel, & Beer, 2007),(Tuci, Quinn, & Harvey, 2002). We have already seen in the dual behaviour task described by Izquierdo that an agent using an evolved CTRNN is able to exploit its own internal dynamics to switch between two quite different control tasks, without any external signal. It is notable that in this case the switch is driven by the change in dynamics as the agent is transferred between bodies. These bodies have very different physical configurations and hence interactions with the environment, providing different patterns of input through different sensor configurations. From this extreme case, it might be assumed that it is necessary for large deviations in the agent/environment interaction to exist to drive the switching of behavioural modes. There is therefore an understandable concern that this adaptivity would be subject to a limitation requiring a well-defined attractor for each behavioural mode. However in other work (Izquierdo & Harvey, 2006), Izquierdo et al showed that adaptation could take place to a continuum of environmental change, which whilst not rendering the Hebbian approach invalid, at least offers the potential for evolved continuous time recurrent neural networks without explicit weight change mechanisms to allow adaptive changes to a behavioural strategy in order to meet the demands of a non-static environment.

This brief consideration of approaches to learning and adaptation has highlighted two separate approaches. The first approach, that of Hebbian synaptic learning, has been demonstrated on real robots but required lengthy adaptation periods, and formation of the behaviour as a consequence of interaction with a given environment. The second, less conventional approach has only been demonstrated in simulation but has shown remarkable potential for adaptation without synaptic strength modification, and achieving an immediate competence on introduction to the task environment.

2.1.6.3.2 Crossing the reality gap

If we are to take the latter of the two approaches mentioned above and use evolved CTRNN controllers without Hebbian learning, we still need to contend with the issue of how best to evolve neurocontrollers in simulation when we wish to transfer their behaviour qualitatively intact to their real-world robot counterparts.

Perhaps the best-known systematic approach for designing evolutionary experiments to increase the likelihood of successful behaviour transfer from simulation to reality was proposed by Jakobi as the radical envelope of noise hypothesis (REON) (Jakobi, 1997). Jakobi highlights the potential for evolved agents to depend on some aspect of the simulation that was in some way an artefact not present in reality as a key problem hindering the successful crossing the reality gap. The artefact may not necessarily be extra information or dynamics, but can be a consistency or predictability that renders the task easier. Evolution will often prioritise exploitation of these aspects over what the designer would consider meaningful engagement with the task.

The creator of the simulated task environment is often not even aware of the presence of these artefacts, however evolution will ruthlessly exploit any aspect of the simulation that leads to higher fitness score evaluations regardless of whether they are intentional or not. An example might simply be the way that the agent is oriented at the beginning of the task, but could also be a more subtle aspect of the physics of the environment, or the way that the agent environment interaction has been modelled. A particularly memorable example of a limited transferability comes from the related field of evolutionary electronics, in which solutions evolved to perform a particular task on a particular piece of FPGA hardware would not successfully transfer their behaviour even when operating on the same piece of hardware but in a different location, or operating in the same location on a different, but ostensibly identical piece of hardware (Thompson, 1998a),(Thompson, 1998b). The evolved solutions were relying on aspects of the environment and physical instantiation of the hardware that the experimenter was not even aware of.

The REON approach is a systematic methodology that is designed to force evolved solutions to rely on aspects of the task that are both known to be available in the real environment and capable of being presented effectively in simulation. Having been proposed over a decade ago when computational demands of task simulation were an even greater constraint in ER than at present, a secondary goal of Jakobi's methodology was to create 'minimal simulations' that did not attempt to replicate any more of the agent environment interaction than was absolutely necessary in order to accomplish a successful behaviour. These minimal simulations will confer the dual benefits of firstly making the simulation easier to construct and secondly speeding up the running of the simulation through reduced computational demand and therefore reduced real time required for successful evolution. Jakobi's insight was that for a robot behaviour to successfully cross the reality gap, only those aspects of the agent/environment interaction that the agent uses and relies on are necessary to be implemented in the simulation. Jakobi terms these essential aspects the “base set” of robot environment interactions. However even the base set aspects may be subject to change within the real world and it is likely that modelling them exactly will be problematic. Therefore a sensible amount of noise or variation should be applied to these base set aspects in between evaluation trials in order to render the controllers base set robust. This definition renders all other aspects of the simulation that do not fall into the base set as at best unnecessary and at worst the very type of simulation artefact that we are trying to prevent the agent exploiting. These aspects are termed the “implementation set” and the methodology aims to render the use of any of these aspects by the evolved neurocontrollers impossible by varying them so much in between trials, that any evolved strategy is unable to use them to perform the required behaviour. In various work, Jakobi demonstrates the successful application of this minimal simulation with REON approach, to octopod locomotion (Jakobi, 1998), to a Khepera robot in a t-maze task and in a gantry robot performing a visual orientation task (Jakobi, 1997).

Some very recent work proposes a kind of hybrid evaluation scheme specifically designed in order to maximise the transferability of evolved neurocontroller behaviour from simulation to reality (Koos, 2010). The authors propose that as one constituent of the fitness evaluation, a specific characteristic which they term simulation to reality (STR) disparity should be minimised. This measure is quantified by transferring neurocontrollers from simulation to reality before the completion of evolution, and comparing the predicted score from the simulation to the actual system. The other tool used is a measure of 'behavioural distance' which is a square of absolute differences of a number of behavioural features. If the newest individuals within the population are sufficiently far in behavioural distance from those already tested on the real robot, then the robot is again used in order to update the reference for STR disparity calculation. As might be expected, the authors report conflicts between the maximisation of performance scores in simulation and the transferability of the controller to reality, however they were able to evolve transferable controllers using this approach using relatively few (25-45) real-world tests during evolution.

Here we have highlighted two approaches to the transfer of valid neurocontrollers from simulation to reality. Due to practical constraints only the former of the two is likely to be deployable in our application, and how appropriate that approach is will be discussed in some detail in the next chapter. However a key element in both strategies is an acknowledgement that however well intentioned, a simulation will never perfectly replicate reality. Because evolution will exploit whatever it is presented with, measures must be taken to direct the evolutionary process to use only those aspects of the simulation that are guaranteed to be present in the real world system, to the exclusion of all other aspects.

2.1.6.4 Evolutionary robotics techniques in the control of flight

In this section we will review the use of ER techniques specifically for the control of flight, whether in simulation or on real flying test platforms. The focus will be on how the flying system is modelled in the simulation environment, how issues of simulation to reality transfer are addressed if applicable, and how the task is often constrained in such a way as to make evolution of successful agents more likely or even feasible at all. We will start at the most explorative end of the spectrum by reviewing work addressing the control of flapping fliers in simulation for the insight they provide into how modelling complex wing geometry may be accomplished. We then move towards work on control of blimps, indoor planes and helicopters.

Shim et al (Shim et al., 2004) performed evolution of controllers for flapping flyers in two stages, in the first the wing morphology was coevolved alongside low-level non-neural controllers to direct the beating of the wings. The morphology of the bird was encoded with parameters designating the position of the wing root and the number and relative angles and sizes of various wing segments. Figure 2.8 shows one wing skeleton, the number of segments is subject to change to allow for more or less complex wing geometries and bilateral symmetry is enforced as might be expected.

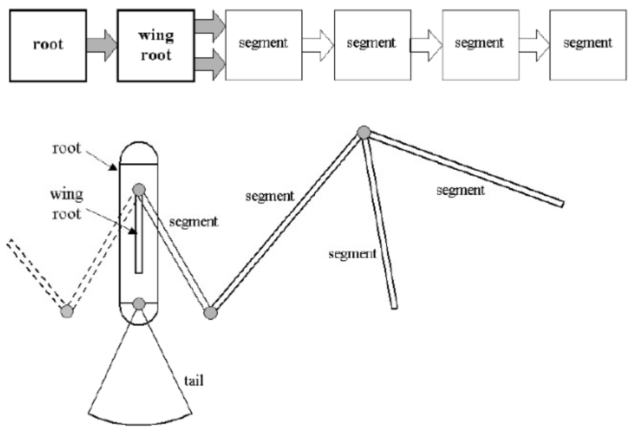


Figure 2.8 The encoding of wing morphology used by Shim et al (Shim et al., 2004)

In the second stage a feedforward artificial neural network was designed in order to perform a waypoint following task. The open source physics engine, Open Dynamics Engine (ODE) was used to perform the physics simulation, the aerodynamic forces being calculated for triangular patches of the evolved wings, and the wings were allowed to deform proportionally to the relative forces exerted on each segment. The process was able to generate plausible wing shapes and flying behaviours including trajectory following, and further work demonstrated remarkably lifelike differences in flapping flight patterns by evolved birds of varying mass, albeit without neural control (Shim & Kim, 2006).

Mouret et al used CTRNN's as central pattern generators for simulated flapping flyers (Mouret, Doncieux, Muratet, Druot, & Meyer, 2004)(Mouret, Doncieux, & Meyer, 2006). Unfortunately few details of their aerodynamic model are presented, however they do specify that they perform separate aerodynamic calculations for small rigid panels that constitute the bird wings. Using a modular neural network encoding scheme called Modnet (Doncieux & Meyer, 2004) they are able to evolve neural network controllers that generate valid flapping motions for forward flight in a birdlike system, whose controller is entirely autonomous, having neither sensory input from the environment nor proprioceptive input. Naturally, because there was no intention to transfer the controllers to reality, there was no requirement to make the controllers adaptive.

One of the most relevant examples of a conventional evolutionary robotics approach i.e. that of evolving neural network controllers to be applied to a real world flight control problem is that of Zuffery (Zufferey, 2005). In a study that constituted part of this work, discrete time recurrent neural networks with static weights are evolved for the control of a lenticular airship or blimp (see Figure 2.9) in an indoor environment (Zufferey, Guanella, Beyeler, & Floreano, 2006). The blimp platform is a rational choice to make when exploring flight control. Being a neutrally buoyant device using a lighter than air gas, simply doing nothing is not grounds for catastrophic failure through crashing into the ground. Also, whilst the platform can move in three dimensions the weight distribution of the system passively stabilises rolling and pitching movements whilst allowing full control of the yaw angle. As can be seen in Figure 2.9, this weight distribution arises from having all the sensors and actuators mounted on an undercarriage slung under the large, buoyant helium enclosure.

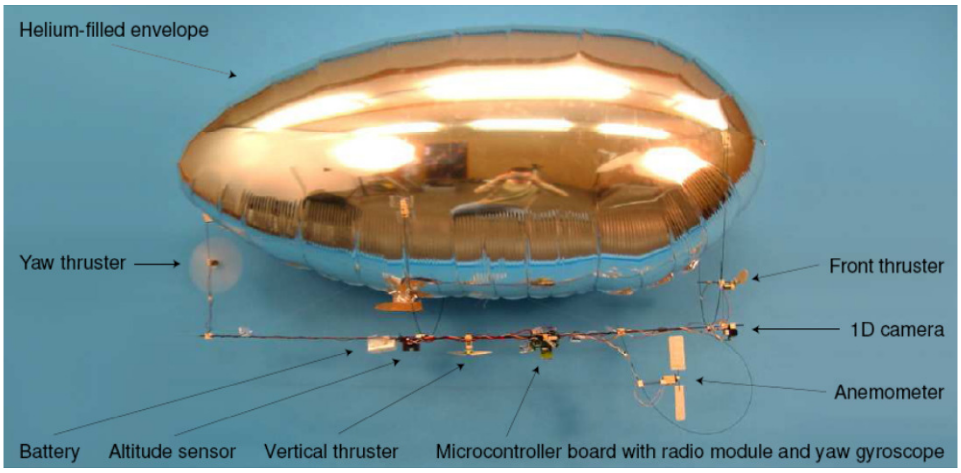


Figure 2.9 The blimp platform used in (Zufferey et al., 2006) **with sensors and actuators labelled.**

Interestingly, because of weight requirements, this work is subject to similar sensor limitations as the work in our study as presented later. This precludes the ability to attain a full state estimation vector i.e. position, orientation and a full set of velocity values for these parameters. Specifically, the system is equipped with a low resolution one-dimensional camera, altitude sensor, a yaw rate gyroscope and anemometer. Following on from work on wheeled Khepera robots and on the blimp using real world evolution (Zufferey, 2005), a simulator was implemented in order to accelerate the evolutionary process, the evolution itself being handled by the software *goevo* (EPFL, n.d.). Using an inertial i.e. world centred reference frame and a body centred reference frame a Newton-Euler approach was used to link the forces exerted on the airship with the acceleration of its mass. In a relatively small number of generations, efficient course stabilisation and wall avoidance behaviour were evolved, including the ability to extract the blimp from situations where the trial was initiated with the vehicle already jammed against a wall. Neurocontrollers transferred well from simulation to reality, displaying similar performance scores and qualitative behaviour, although as is common any quantitative measure of behavioural similarity was elusive. Both motor signals and sensor values varied somewhat from simulation to reality yet the behaviour transferred robustly despite this disparity. Because no variation in the frequency of the stripes on the wall surface was allowed in evolution, evolved controllers were fragile to the variation of this parameter. One of the most interesting results arising from this work was that controllers evolved using a simplified version of the physics simulation that neglected added mass Coriolis effects performed well in simulation but poorly in reality. Following this result the author argues for the necessity of accurate physical models and highlights the difficulty in evolution of behaviours involving relatively fast flying wings due to the problems of modelling the unsteady dynamics. Another study of autonomous blimp control (Bermudez i Badia, Pyk, & Verschure, 2007) uses visual motor control based on the fly visual system and an obstacle avoidance model based on the lobular giant movement detector neuron of the Locust. By a human designer integrating known principles of insect neurobiology, it was possible to build adaptive reactive controllers capable of autonomous flight and robust to severe perturbations such as the removal of one propeller.

Other work uses evolutionary techniques in the form of a (10+23) evolutionary strategy with mutation but no recombination, to evolve controllers in a simulation environment for autonomous helicopters whose dynamical properties are not known precisely (De Nardi et al., 2006), (De Nardi & Houska, 2007). In this work, feedforward neural networks in the form of multi layer perceptrons were evolved incrementally, after it was demonstrated that monolithic MLPs evolved from scratch were incapable of generating appropriate control behaviours for the simulated helicopter. However, it was shown that a modular MLP could be evolved for the control task if an existing PID controller was employed as a scaffold. In the PID controller, the control task is decomposed into a number of functional modules, for example control or yaw, roll etc, and the methodology was to sequentially replace each functional module of the PID controller with an individual multilayer perceptron. A similar approach in which the task was decomposed without the use of the PID scaffold also generated good results. It was shown that both evolved feedforward modular networks and PID controllers with evolved gains were capable of good performance in a waypoint following task although the authors noted that the evolved neural networks were more robust when faced with variation in the parameters of the task, vehicle or environment. It is noteworthy that in this work, a full state vector of the helicopter's position, rotation and velocities of these values was available due to the assumed presence of a valid and sensor suite and sensor fusion algorithm.

This short tour of approaches to biologically inspired approaches to flight has served to highlight the recurring themes introduced earlier in the chapter. These include how slavishly to mimic the biological system, and how to model the physics of the target system, whether the controller should be partly or wholly hand designed or purely evolved, whether modularity should be enforced on an evolved controller, and how well evolved controllers might transfer to real world test platforms. The diversity of approaches, each having some success in their intended domain serves to reinforce the impression given in reviewing the earlier work, that there is no one-size-fits-all solution, that each case should be assessed independently in terms of both the system to be controlled and the goals for the research itself.

2.2 Introduction to airborne wind energy

This section will introduce the context behind the interest in airborne wind energy and the rationale of why these flying systems offer advantages over conventional wind energy conversion device architectures. We then introduce the two practical approaches that are dominating work in the sector, focusing specifically on the operation of, and considerations regarding the ‘lift mode’ which is the subject of the work in this study, in which kites are used to drive generators on the ground.

2.2.1 Background and motivation for airborne wind energy

The current scientific consensus is that the observed climatic warming trend is at least partly attributable to anthropogenic emissions of greenhouse gases such as carbon dioxide and methane (Pachauri, Reisinger, & Eds., 2007). This has led to an increased interest in renewable energy technologies, such as wind and wave and solar energy which are not direct emitters of these gases. In the UK, wind energy is both the largest contributor to energy demands out of the current crop of renewable energy technologies and has the lowest cost per unit of electricity (UK Department of Trade, 2007). The British government has committed to increase the proportion of renewable energy to 50% by 2050, of which wind is expected to take up a large proportion (UK Commitee on Climate Change, 2010). However, current wind energy technology is not able to economically access the bulk of the wind resource which is above the height at which conventional utility scale horizontal axis wind turbines (HAWTs) currently operate (see Figure 2.10). It has been long known that this near ground wind resource is of lower quality as it is slowed down by the drag imposed by the underlying terrain. Wind at high altitudes is unencumbered by such obstacles and therefore flows more strongly and more consistently than wind at lower levels (see Figure 2.10) (Kaimal & Finnigan, 1994). The layer of wind affected by the presence of the ground is known as the atmospheric boundary layer (Wallace & Hobbs, 2006).

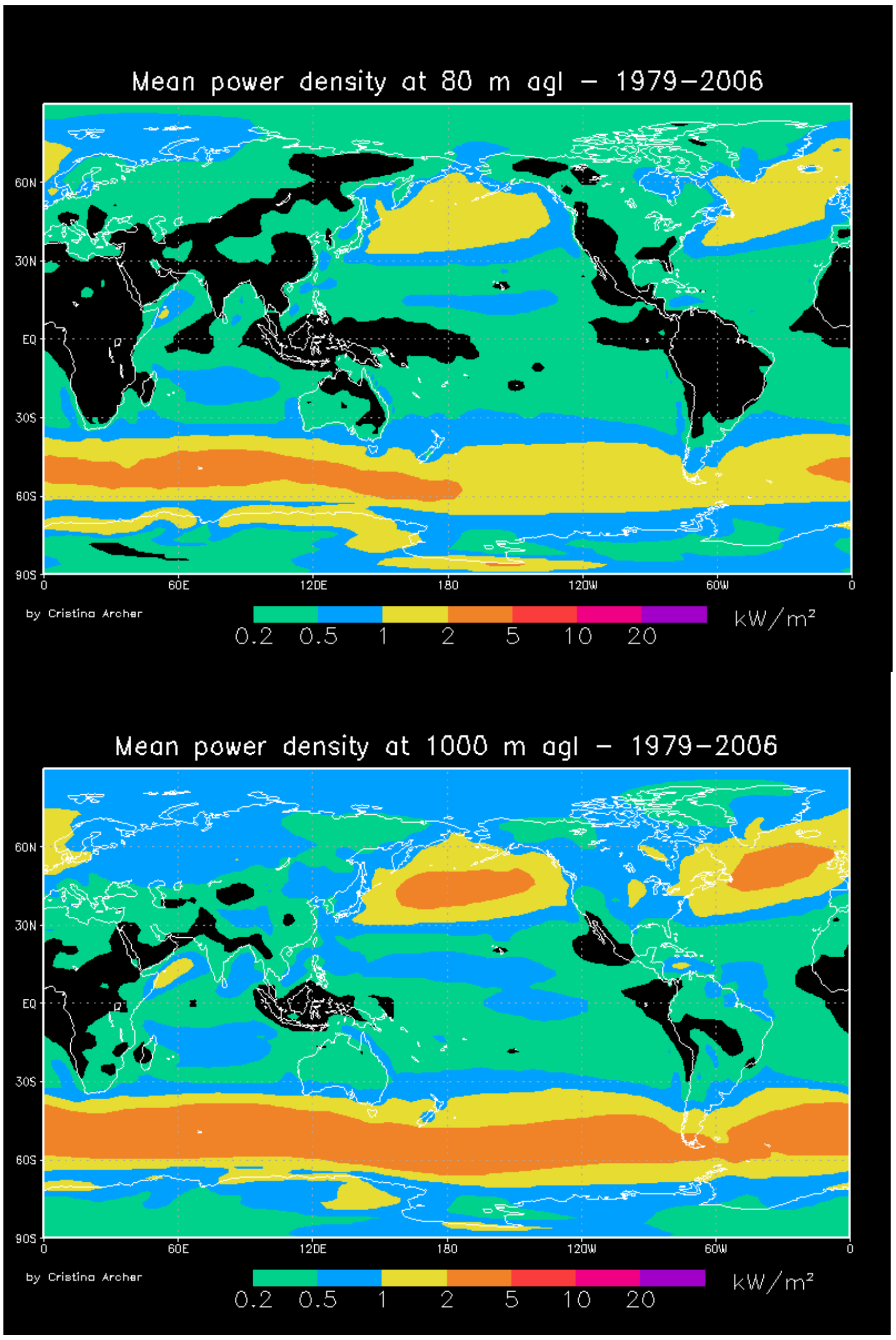

Mean power density at 80 m agl – 1979–2006
60N
30N
EQ
30S
60S
90S
0
60E
120E
180
120W
60W
by Cristina Archer
0.2
0.5
1
2
5
10
20
kW/m²
Mean power density at 1000 m agl – 1979–2006

Figure 2.10 Plots of the time averaged geographic mean power density of wind energy at 80m (upper) the typical height of a wind turbine, and 1000m (lower) the approximate proposed height of operation of many airborne wind energy systems. (Archer & Caldeira, 2009)

Airborne wind energy solutions have been long proposed to access this superior wind resource, concepts including raising an impeller based turbine using a blimp or aerostat or by using a kite to heights at which a tower would be uneconomical (Kushto, 1978),(Fry, 1978). The first mature, worked through, peer reviewed publication on airborne wind energy systems was published in 1980 by Miles Loyd (Loyd, 1980). Loyd made the crucial observation that any airborne wind energy system would need to fly near-perpendicular to the direction of the wind or 'crosswind' in order to maximise their efficiency. In some respects this resembles the fast crosswind motion of the blades of a conventional horizontal axis wind turbine (HAWT), where the tips of the blades are moving perpendicular to the wind at several times the velocity of the environmental windspeed. As the aerodynamic forces developed on the blade surface are proportional to the square of this 'apparent' windspeed, which is effectively a vector sum of the ambient windspeed and the surface’s own motion vector, higher forces are developed by these components moving at fast crosswind speeds.

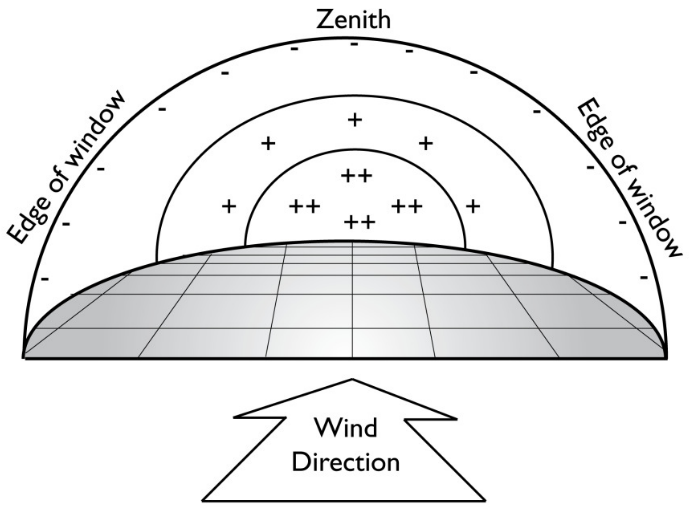


Figure 2.11 A diagram of the 'wind window' the quarter sphere surface on which a kite constrained by a given length of line is able to fly. Because of the angle of attack and resulting flight speeds, the tension in the line is high in the central regions of the window (++, and +) and low in towards the edge of the window (-)

A typical kite used in this application will have 4 control lines, two at either side of the leading edge and two at either side of the trailing edge. Changing the relative lengths of front and back lines will change the pitch or angle of attack (see Section 2.2.2) and changing the relative lengths of the trailing edge pair from left to right causes the kite to roll. The effect of these inputs is that the kite is steerable and thus it’s position and therefore speed is under control, and the lift and drag forces can be raised or lowered within bounds by controlling the attack angle.

Loyd described two possible configurations for airborne wind energy systems being flown actively in crosswind paths. In the first which he termed the *lift mode*, a tethered wing would fly crosswind paths exerting considerable pull on its tether which would be harnessed to perform useful work by allowing the tether line to be pulled out from a reel which is coupled to a generator (see phase ‘A’ in Figure 2.12).

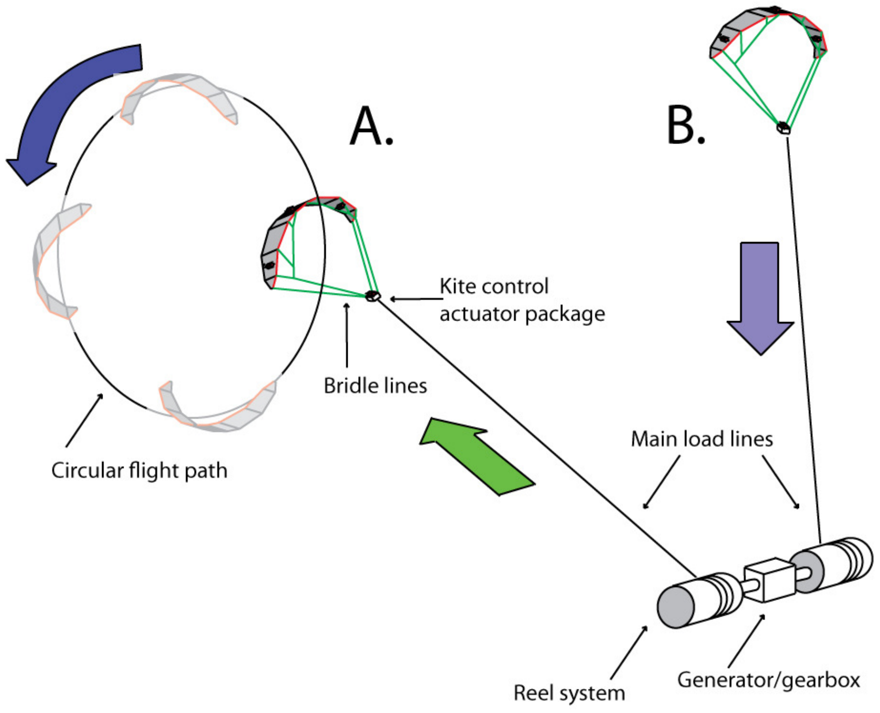


Figure 2.12 An airborne wind energy system in a yo-yo configuration operating in Loyd's 'lift mode'. Kite A is in *Generation* phase. Kite B is in *Retraction* phase. Note in this design concept the two modes are operating side by side on one system but a unit could also consist of a single kite alternating between these modes in isolation.

Because the aerodynamic force upon the wing and the resulting tension developed in the tether line depends upon the apparent wind velocity and therefore the flight speed, it is possible to minimise the tension on the line by maintaining the wing at a constant or near constant position. As mentioned above, the lift and the drag forces on the wing will also depend upon the angle of attack (AoA) which is the angle of the kite relative to the apparent wind. Therefore by reducing the attack angle and by ceasing to actively fly the wing in a crosswind flight path, the tension on the line can be minimised. This allows the retraction of the kite with a lower expenditure of energy than is generated in the reel out phase. This alternating cycle of generation and retraction has led to Loyd's lift mode being termed the yo-yo configuration. A second option for the generation of energy is described by Loyd as the 'drag mode' in which case impellers much like conventional wind turbine plant are mounted on a wing and the resulting generated power is transferred via an electrical connection through the tether line. Lloyd notes that for a given weight of wing, the theoretical maximum energy generation is roughly equivalent for the lift and drag modes (see Figure 2.13).

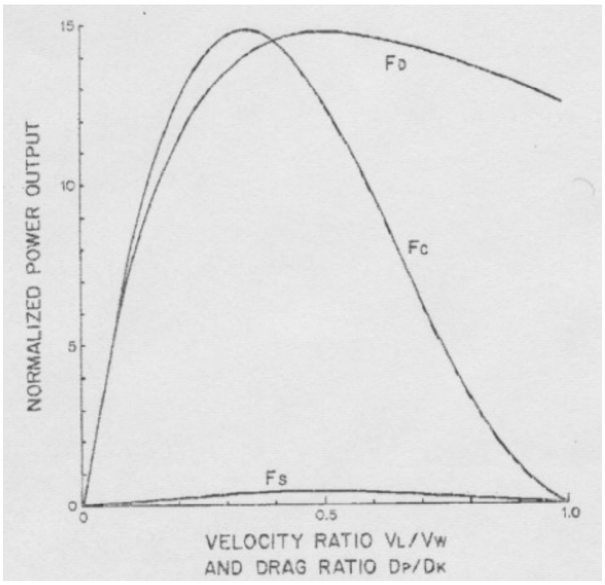


Figure 2.13 Taken from (Loyd, 1980), this plot shows the normalised power output of airborne wind generators with a glide ratio of 10 being flown in either the lift (F_C) or drag (F_D) or non-crosswind i.e. static (F_S) modes. The plot shows the relationship between power output and reel out speed (V_L) : wind speed (V_W) ratio for the lift mode and turbine drag (D_P) : Kite drag (D_K) ratio for the drag mode.

Furthermore, his analysis indicates that for wing of a given mass flown in either mode, the key determinant of system output in a given wind regime is the speed at which the kite is flown through the air, a ceiling on which is provided by the aerodynamic efficiency of the wing, known as the lift to drag ratio or glide ratio. Loyd neglects the drag of the tether in his analysis, which will be an important source of drag in any real system, especially at the end nearest the kite where the cable is travelling at the highest speed through the air.

Whilst an in-depth discussion as to the relative benefits of the lift and drag modes is beyond the scope of this document, it is clear that the approaches have respective advantages and disadvantages, the most salient of which are worth noting. Most relevant to this work is the fact that the type of wing used for lift mode systems can be either a light fabric structure closely related to a paraglider or kitesurfing kite or could be a composite rigid wing. Due to the weight of the on-board generators, systems operating in drag mode must be a rigid structure, normally carbon fibre, to accommodate the additional load on the structure (see Figure 2.14).

Each class of system has varied economic, performance, and operational pros and cons. Although rigid wing systems would be far more durable, they would involve a larger upfront investment and resulting capital costs than a system in which a cheap flexible wing was replaced at intervals. The up-front cost penalty of a drag mode system will be high, given that a winching system will still be needed at ground level to manage line tension and flight altitude, whereas a lift mode system will use that system in order to generate power. However operational costs of lift mode systems will be affected by the need to replace the tether, the repeated bending of which around the drum causes significant wear and reduction in operating life. The tether might however be expected to be cheaper and lighter than a tether for a drag mode system which needs to have sufficient electrical conductive capacity to carry the entire load of the system. The respective weight of the airborne portion of each system is very important because there will be a significant loss to inertia as the wing is flown in the repeated looping trajectory. Rigid wings will be particularly exposed to increase in weight penalty upon scaling up because their volume and hence weight will increase exponentially with surface area, whilst the mass of fabric structures will scale with far closer to a linear relationship. However current fabric structures need extensive bridle systems (see Figure 2.14, panel A) in order to prevent the wing buckling and losing shape under the high loadings expected. These bridles are a significant source of aerodynamic drag in addition to that imposed by the tether. Recent work on kites partly constructed of cable reinforced airbeams indicate that increasing the stiffness of fabric kites, and thereby reducing the amount of bridling is likely to be possible (Breuer, Ockels, & Luchsinger, 2007).

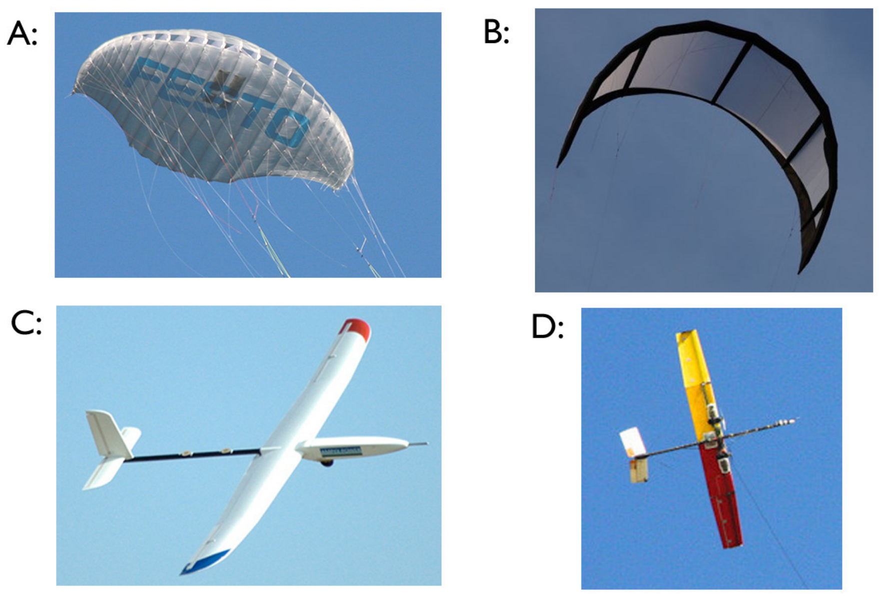


Figure 2.14 Four types of wing design for airborne wind energy systems, A) a ram-air parafoil (source: Aeroix Gmbh), B) A bridled leading edge inflatable (LEI) kite (source: Airplay Kitesailing BV), C) A fibreglass glider to be flown in lift mode (source: Ampyx Power BV), D) A carbon fibre wing with on-board electric motor/generators for drag mode power generation (source: Makani Power Inc.)

One of the key differentiating factors behind these approaches as currently proposed relates to the ease with which the respective system concepts can be modelled and controlled. Drag mode systems benefit from the control authority afforded by conventional control surfaces and multiple, electronically controlled wing mounted turbines, which can be powered to launch the system and used as differential brakes to assist in steering when the system is generating. These rigid, powered drag mode systems are far more amenable to control using techniques familiar to the autonomous airborne vehicle community. Soft and light fabric kites are particularly ill suited to conventional control techniques partly because the control actuation is limited to relative line lengths of only four lines, with only 2 degrees of freedom, i.e. pitch and roll with no direct control over yaw. Additionally the shape of the fabric kites is prone to severe warping or deformation as the wing loading changes during flight, this makes substantial contributions to the flight dynamics of these wings, rendering computational modelling of such wings exceptionally difficult, with analytical models and conventional Newton-Euler or Lagrange formulations of the equations of motion at best inaccurate, given the necessity to make simplifying assumptions.

Whilst not an exhaustive analysis, this section has sought to provide a concise introduction to airborne wind energy and has explained that the distribution of the wind with altitude is the key motivating factor driving this work. It has also established the two key ways in which energy is generated by the systems currently in development, and highlighted some advantages and disadvantages of each approach. It should be clear that the evolutionary approach described previously has the most potential to add some value to lift mode systems using flexible wings in which there are significant challenges in deploying more conventional control techniques. By designing the evolutionary process in such a way that the unknown aspects of the system are acknowledged, it might be possible to create a robust controller for this class of system.

2.2.2 Basics of kite flight

Here we seek to give the reader the basic understanding needed to appreciate what is being required in the control task. As mentioned above, the kites we are using are controlled by four lines. The relative length of the left and right trailing edge pair can be altered to roll the kite and hence steer it to any position on the quarter sphere of the wind window that is pictured in Figure 2.11. The relative lengths of the leading and trailing edge pairs can be altered to change the pitch or attack angle (α) of the kite with respect to the wind (see Figure 2.16). Being tethered wings, kites are subject to aerodynamic lift perpendicular to the relative motion of the wind and aerodynamic drag in the direction of the relative motion of the wind. The lift and drag forces on a given wing are linearly proportional to the lift and drag coefficients, Cl and Cd respectively. These coefficients change in a non-linear fashion with the angle of attack which is the angle of inclination of the wing with respect to the effective wind see Figure 2.15.

The Cl and Cd for a given wing can be measured directly in a wind tunnel or estimated computationally by numerically solving the Navier-Stokes equations of fluid flow at a large number of discrete positions on the surface of the wing and in the surrounding volume. This computational fluid dynamics techniques is exceptionally computational demanding especially for 3D structures and unsteady flows, simpler but less accurate panel based techniques such as the Vortex Lattice method are also available.

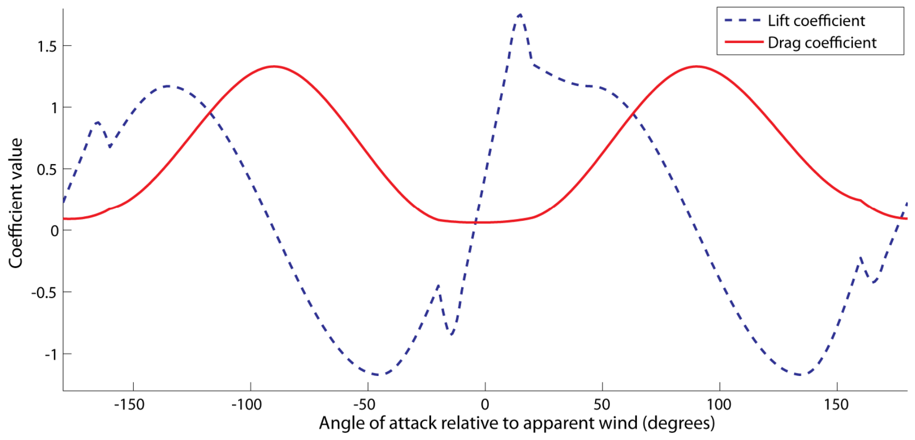


Figure 2.15 A plot of the lift and drag coefficients for a modified NACA 2412 Airfoil. Note the dramatic drop in Cl after the stall point at approximately 22 degrees AoA

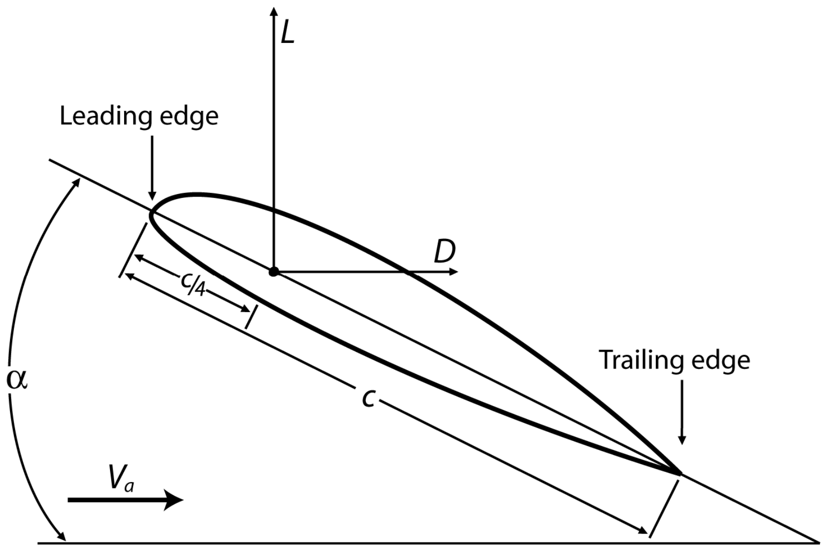


Figure 2.16 Cross section of an aerofoil showing lift L, drag D and angle of attack α relative to wind vector Va.

Knowing the Cl and Cd profile for a given wing, as per Figure 2.15 for example, it is then possible to calculate an estimation of the aerodynamic forces $\vec{F}_w^{aer}$ on a wing for a given environmental wind speed provided that the motion of the wing itself is well known. Ideally, given the spanwise asymmetry of most wings and the drag induced by turbulence at the wingtips, it is preferable to know the Cl and Cd values at all pitch angles for several points along the wing. It is worth noting that it is likely that different parts of the wing are moving at different speeds and relative angles relative to the environmental wind vector. For example, in a wing flying a looping trajectory, the side of the wing that is in the centre of the circle will necessarily be travelling more slowly than the outside wingtip.

Eq. 2-6

$$\vec{F}_w^{aer} = F_D + F_L$$

Eq. 2-7

$$F_D = -\frac{1}{2} C_D A \rho {V_a}^2$$

Eq. 2-8

$$F_L = -\frac{1}{2} C_L A \rho {V_a}^2$$

Eq. 2-7 and Eq. 2-8 show the calculations of the drag and lift forces, Fd and Fl respectively, for a wing with area A, flying in air with density ρ, given apparent wind velocity V_a. Note how the lift and drag scale with the square of the apparent wind speed. Any calculation of the force on a real wing will necessarily only be an estimate because it is not possible to have perfect knowledge of the characteristics of the wind field in which the kite is flying. If we are measuring the wind speed at or near ground level the wind shear, i.e. the tendency of the wind speed to increase with altitude, should ideally be taken into account. One method used in the literature (Lansdorp, Ruiterkamp, Williams, & Ockels, 2008) is to use measured data for a given site, otherwise the approximation described by Eq. 2-9 to calculate the wind w at height h can be used as per (Houska & Diehl, 2007), where h_r is the roughness length which is terrain dependent, and v_o is the wind speed at the reference altitude h_0.

Eq. 2-9

$$w(h) = \frac{\ln\left(\frac{h}{h_r}\right)}{\ln\left(\frac{h_0}{h_r}\right)} v_o$$

It should be acknowledged that the shear equation given above only describes a statistical correlation between wind speed at two heights that is not guaranteed to give an accurate moment to moment measure of wind speed at a distance. Additionally there is no direction component available, and the wind direction is liable to rotate with altitude due to a combination of Coriolis forces and friction with the ground. The resulting theoretical distribution, known as an Ekman spiral is shown in Figure 2.17.

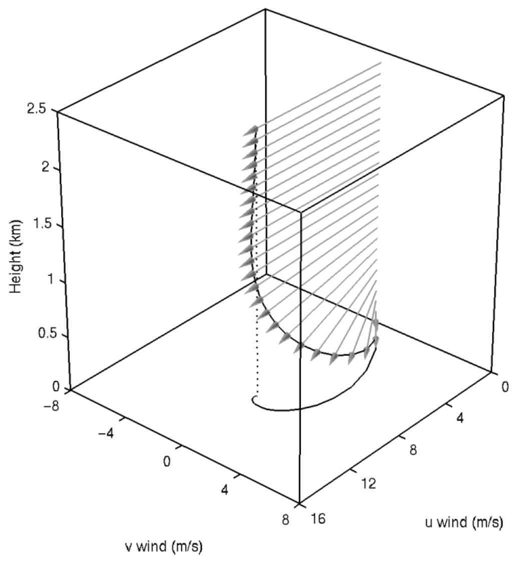


Figure 2.17 Plot showing wind vectors in an Ekman spiral in which both the speed and direction of the wind changes with altitude.

It is also possible take into account the effect of altitude on air density at a given height h as per Eq. 2-10 where ρ_0 is the density of air at ground level and h_1 is the known physical constant 8330m.

Eq. 2-10

$$\rho(h) = \rho_0 e^{-\frac{h}{h1}}$$

Tension in the cable will simply be equal to the component of $\vec{F}_w^{aer}$ that is in line with the cable, which can be calculated as the dot product of the unit vector of the aerodynamic force with the unit vector from the tether point to the kite. Naturally there will also be drag on the tether that will be proportional to its speed and diameter. The tether will also exhibit some elasticity and will tend to sag under its own weight, factors that will certainly have some bearing on the kite dynamics and the resulting control requirements.

Here we have briefly introduced the forces on a kite and some factors affecting the wind resource in which the kite flies, we will move on to approaches to modelling and controlling these systems in the literature in the following section.

2.2.3 Modelling, control and hardware for kite systems

Various approaches have been taken for modelling and control of kites, which we will briefly review here purely to gauge the common assumptions made in the literature. It is notable that although publications regarding the control of kites and descriptions of robotic control system concepts have been available since 2006 (Canale, Fagiano, Ippolito, et al., 2006), at the time of writing there is currently no published work in which controllers are deployed on kite systems in the real world. It might well be that the assumptions that are necessary to model the kite system in such a way as to render it amenable to modern non-linear control techniques, cause sufficient aspects of the dynamics of the kite system to be discarded as to impair the performance of the controllers developed from these models in the real world.

2.2.3.1 Modelling, control and trajectory optimisation

The simplest model is to consider the kite be constituted by a point mass, which is subject to aerodynamic forces that relate to wing-like characteristics associated with the point mass, i.e. area, roll angle etc. This is the approach taken in (Houska & Diehl, 2007), (Houska, 2007), where Lagrange formulations of the equations of motion are derived and then used for the optimisation of flight trajectories that are at least locally optimal in terms of power generated at a ground level generator using a multiple shooting method. An example of one of these flight trajectories is shown below in Figure 2.18.

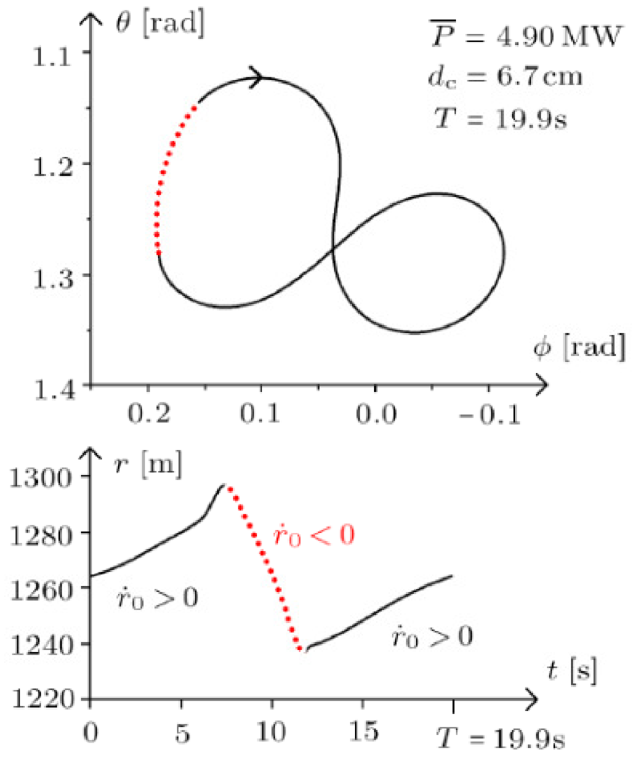


Figure 2.18 A local optimum trajectory from (Houska & Diehl, 2007) **in which the kite is retracted within a loop, shown by the red dotted section. Lower plot shows the line length**

In these studies it is assumed that the lift coefficient Cl is under direct control, rather than emerging from the interaction of the line lengths and distribution of force across the kite. This is illustrated well by Figure 2.19 where the lift coefficient is modified near instantaneously in a very rapid linear transition.

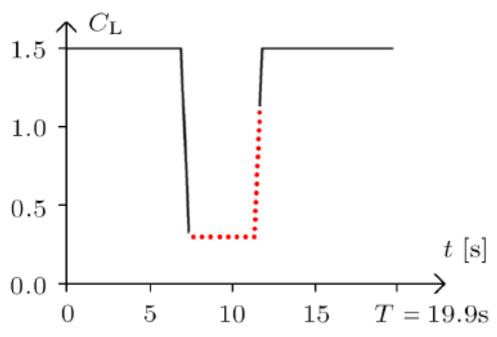


Figure 2.19 A plot showing the assumed level of control over the lift coefficient. Note the transition from very high to very low lift modes in less than a second, this would almost certainly be destabilising for a kite in the real world.

The roll angle is also assumed to be under direct control in a similar manner, in common with studies by other groups (Canale, Fagiano, Ippolito, et al., 2006),(Canale, Fagiano, Milanese, & Ippolito, 2007),(Canale et al., 2009),(P. Williams, Lansdorp, & Ockels, 2007a) in which 'fast' Model Predictive Control (MPC) is demonstrated to control the model in simulation. It is shown using this approach that the simulated kite can be flown in repeated predetermined looping trajectories even when there are perturbations in the wind speed and direction (see Figure 2.20).

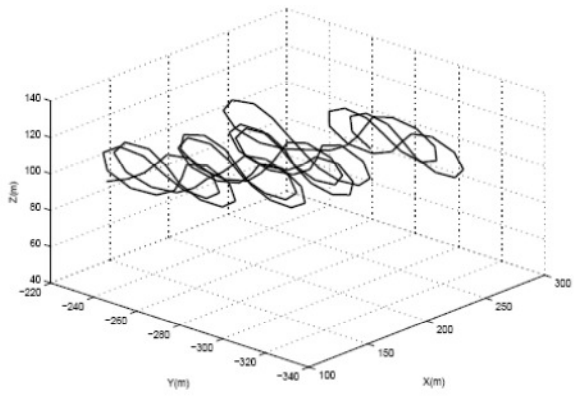


Figure 2.20 This plot shows some looping generation phase orbits from (Canale et al., 2007).

However, by considering the kite as a point mass, there can be no incorporation of a diversity of forces across its surface or a treatment of the coupling of deformation and aerodynamic load. This treatment of the kite as a point mass and assumption of roll angle as a direct and immediate control parameter persists even in very recent work (Fagiano, Milanese, Member, & Piga, 2010). Other unrealistic simplifying assumptions are commonly made, for example considering Cl and Cd to be static values (Canale, Fagiano, Ippolito, et al., 2006) although this was remedied in later work such as (Fagiano et al., 2010).

Work in the below section was not available whilst the model used in this work was being developed, however we include it here for completeness. In one study concerned the modelling and control for a 2 line kite, which was considered to consist of 2 plates as shown in Figure 2.21, (P. Williams, Lansdorp, & Ockels, 2007b).

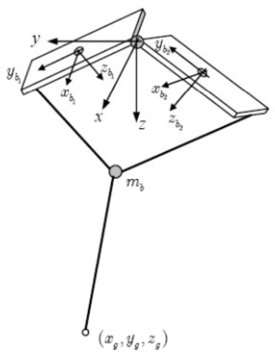


Figure 2.21 A diagram of the model used in (P. Williams, Lansdorp, & Ockels, 2007b) **showing the two flat plates, each with an independent pitch angle and joined via a hinge at the leading edge**

Also briefly tested was an extension with four plates, coupled in the latter case with torsion springs. In this study the mechanism of control, a sliding attachment point along the edge of the kite was modelled explicitly rather than assuming direct control of roll and attack angle. Roll angles are therefore controlled indirectly as a product of the actuator input and the aerodynamic forces being exerted on each plate. Symmetry however is enforced such that the angle of attack on each plate is identical. An aerodynamic moment damping term is added for stability due to the light mass of the kite in comparison with that of the air in the modelled kite system. One interesting study compares two mathematical models of a steerable kite used for ship towing, one with roll and one with yaw control (Podgaets & Ockels, 2007). It notes a remarkable lack of agreement between the results of the models, noting how optimisation techniques will produce very different 'optimal' control trajectories depending on the model employed. It concludes by arguing for the use of multibody models, without direct i.e. instantaneous perfect control of steering.

This summary of approaches to kite modelling has sought to define the key issue in the field, that if the deformation and variable loading of the wing is to be incorporated, point mass models are unlikely to be sufficient. Additionally there are a number of simplifying assumptions made in prior studies which omit key aspects of the system to be controlled, complexities of aerodynamics or any consideration of actuator dynamics. Due to the lack of studies presenting the performance of controllers developed using such models, it is difficult to gauge the impact of these simplifications on controller competence.

2.2.3.2 Hardware

Because in this work we are seeking to build an appropriate test platform on which to test evolved neurocontrollers, here we review prior work in the area. Unfortunately there is little information available in the literature regarding methods of physically steering kites using means amenable to computer control, and only one comparative study (Lansdorp, Ruiterkamp, & Ockels, 2007). Whilst a distinction must be made between test systems and commercial devices, the attributes that we are seeking in a test device i.e. control of both pitch and roll, low cost, light weight and portability are also likely to be desirable in larger scale commercial systems.

The earliest study in the literature presenting a real hardware device is a lift mode kite generator system in which steering and power generation is combined in a ground level based system termed a kite steering unit (KSU) that uses two lines and two separate winches (Canale et al., 2007) which are coupled to motor/generators (see Figure 2.22). This system uses the control of line length difference in the two lines to steer the kite primarily by roll.

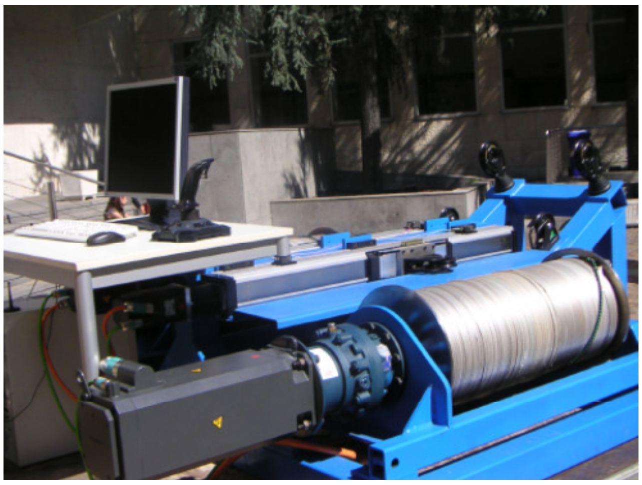


Figure 2.22 A kite steering and power generation unit rated at 40 KW

Whilst having an impressive maximum power output capacity for a test platform, such a system is clearly limited by an inability to control the attack angle of the wing and thus modify the force exerted on the line by means other than kite speed and position. To avoid duplication of costly parts such as gearboxes, generators, brakes, flying lines etc., it would be preferable to have a single drum and flying line. One necessary constraint if the kite is to be flown on a single winch is that the controller must be situated some way up the line near the kite. Lansdorp et al developed a single line test winch and then compared the performance of three control devices including a pair controllable drag flaps located at each tip of the kite's wing and a pair of servo controlled mechanisms attached to the leading edge designed to change the AoA of a desired tip (Lansdorp et al., 2007). They found the device with the greatest performance to be a system attached to each wingtip that could dynamically alter the attachment point of the kite. The device (pictured in Figure 2.23) is a motorised cart that moves along the toothed rail to which it is coupled, in order to translate the attachment point between the leading and trailing edge. By moving the carts on each wingtip in the same direction the kite's attack angle can be modified, although if a certain average AoA is to be maintained then the steering will tend to saturate, limiting the steering control at the extremes of power/depower as noted in (P. Williams, Lansdorp, & Ockels, 2007b), (P. Williams, Lansdorp, & Ockels, 2007c).

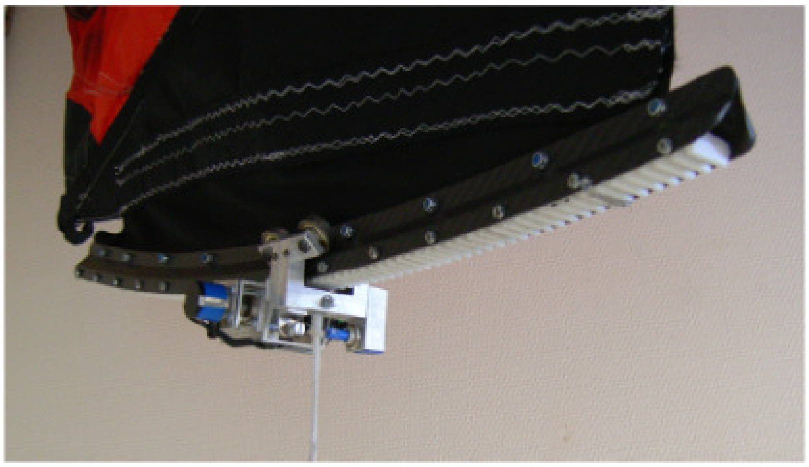


Figure 2.23 A close up of TUDelft's cart and rail steering mechanism design to steer a kite by moving the tow point chordwise, i.e. between the leading and trailing edge.

Whilst reasonably effective, these cart and rail systems were rather heavy and are not well suited to the control of parafoil kites or newer and higher performance surf kites that are equipped with an inflatable skeleton and a bridle. Recent development on combining tension and compressive elements with the air beams of inflatable skeleton kites have been highly promising (Breuer et al., 2007) and it would be preferable if any control hardware did not rule out the use of such concepts. Another approach might be to control the wing from an actuator unit that hangs under the kite, such devices have been developed commercially for controlling kites that are used to tow cargo ships, see Figure 2.24. Unfortunately there is no published information to assess the design or performance of these devices. One potential bonus of such systems is that as a significant mass under the wing, they may act to stabilise the rotation of the kite. However if they are too heavy they might start to affect the flight of the kite in an undesirable manner, and they will certainly be a source of additional aerodynamic drag.

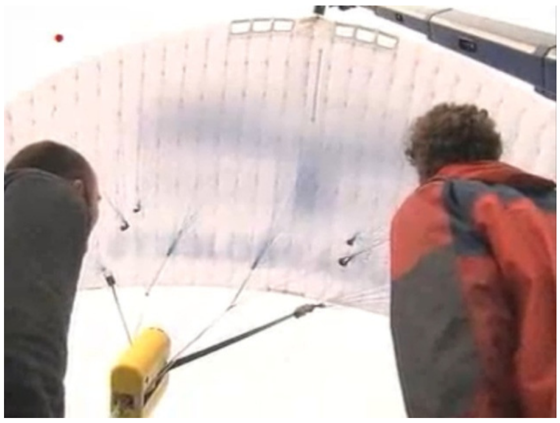


Figure 2.24 A kite suspended control actuator unit, also known as a pod or gondola attached to an 80m^2 ship towing kite. (Source: Skysails gmbh)

This section has sought to introduce some approaches to controlling kites by electromechanical means. Hopefully the impression has been given that there are several possible approaches, each with their own advantages and disadvantages. Any given design will therefore be a compromise that will need to be judged against the specific demands of the task.

2.3 Summary

In the first section of this chapter we attempted to provide an appreciation of the motivation behind the application of ER to certain classes of problem. We have reviewed the literature relating to the use of ER for simulated tasks, and highlighted issues that are potentially relevant to the evolution of neurocontrollers for control of tethered wings. We have also reviewed ER implementations that have been used to control robots in the real world in order to emphasize the choices that need to be made in order to implement the controller successfully in reality. We have touched upon the important issue of crossing the reality gap and replicating the performance achieved in simulation. We also reviewed the literature where evolutionary techniques have been applied to the control of flight both in simulation and reality on a number of free flying platforms.

In the second section, we introduced how environmental and economic considerations are driving the interest in airborne wind energy, which permits exploitation of the superior wind resource at altitude. We explained the basic concepts of flight and constraints specific to this application that form the context for work in airborne wind energy systems. Finally, we briefly reviewed the modelling techniques used in the literature and possible designs for control hardware for kites. This information from this second section will allow us to make a considered evaluation of how best an ER approach may be implemented specifically for the control of tethered wings, which we will consider in the following chapter.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing, Computing Machinery and Intelligence

3 Considering kite control as a task for ER

This chapter follows on from our review of the literature in chapter 2 and assesses specific considerations that might be applicable when deploying an evolutionary robotics approach to kite control, as opposed to previous problems in the literature. Here we will consider how kite control might differ from other problems considered in the literature and how best to approach these differences. We also consider the strengths and limitations of different approaches to kite modelling and robotic control of kites and assess the considerations that need to be made in order to overcome some of the problems inherent with the approaches and design concepts utilised in the past.

3.1 Kite control as a problem for evolutionary robotics

In this section we will attempt to characterise similarities and differences between evolutionary robotics tasks in the literature, and the kite control problem as presented here. As we have seen in the last chapter, there is no one-size-fits-all approach to the implementation of the evolutionary robotics paradigm, nor is there a large body of work that seeks to characterise in a systematic way what approaches may be useful for which problems. The aim of this section therefore, is simply to set down the problem that we face, and note any particular characteristics or combination of characteristics unique to this problem. This will allow us to appraise the suitability of implementation approaches from the literature in order to best consider a potential route to successful implementation.

3.1.1 Embodiment and simulation issues within the kite control problem

One critical aspect of the kite control problem is the extent to which it is deeply embodied and situated. The nature of the kite's physical embodiment is crucial, and the assumptions made in many of the models of kite physics within the literature will seem foreign to many accomplished human kite flyers. Human experts are well aware of the dynamic nature of the kite itself, the environment in which it is flying and the manifestly non-trivial interaction between the way the kite's state is modified by the environment, and the way the kites state affects how it responds to its environment.

The implication then is that it will be beneficial to capture as many of the key aspects of the flight dynamics of the kite as is practical. Additionally, if dynamics exist which we cannot model, they should be acknowledged and controllers developed that should not be dependent on them operating in a certain manner. These considerations take particular importance given that there is very limited direct control over the 'body' of the kite itself in the manner that might be typical in a robot with many actuators. Being limited to control of only two degrees of freedom corresponding to rotations in roll and pitch the system is already underactuated. There is no redundancy such as in a conventional aircraft which upon suffering a yaw rudder failure, can rotate 90 degrees in roll and use it's ailerons to achieve yaw rotation.

The sole method of control, i.e. the changing of line lengths is itself indirect, operating through and dependent on interaction with the environment. Figure 3.1 attempts to summarise some key aspects of manner in which the kite system is embodied and the coupling between constituents and the environment.

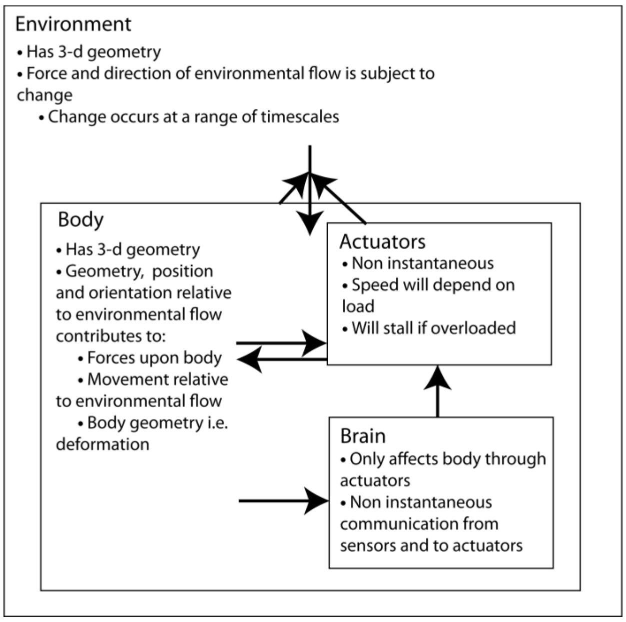

Environment
• Has 3-d geometry
• Force and direction of environmental flow is subject to change
• Change occurs at a range of timescales
Body
• Has 3-d geometry
• Geometry, position and orientation relative to environmental flow contributes to:
• Forces upon body
• Movement relative to environmental flow
• Body geometry i.e. deformation
Actuators
• Non instantaneous
• Speed will depend on load
• Will stall if overloaded
Brain
• Only affects body through actuators
• Non instantaneous communication from sensors and to actuators

Figure 3.1 Schematic noting key characteristics of the kite control task. Arrows indicate where one component affects another. Arrows pointing to another arrow indicates modulation of a coupling.

Specifically, if the wind is weak, steering commands will typically take longer to take effect, partly because there will be less tension in the cable so the first part of the steering motion will be adding sufficient tension in the cable to achieve some movement of the wing. Secondly because there is less force on the wing, it will both deform and roll in a qualitatively different manner than if the wind were strong. At the other extreme, very strong winds will cause high tension in the steering lines that will slow the servos down, or even cause them to stall or be backdriven. An analogous situation might be that of a robot arm where the position of the joint in response to a control command to a servo motor would be dependent in some nontrivially dynamic manner on some aspect of the environment, the light level in the room for example. Whilst it might not be essential to capture precisely the relationship between the light level and the servo modulation, we would expect it to be necessary that some dynamical relationship would need to be implemented in the simulated task environment if we were to expect successful transition of controller behaviour from the simulation to reality.

There is certainly a tension between the attraction of minimal models which are computationally less demanding and more complex models incorporating more prior

knowledge. The noise required to develop successful behaviours that transfer from simulation to reality can create an additional burden on the evolutionary process, a point acknowledged by the proponents of the approach themselves (Jakobi, 1997). It is then for the designer of the process to assess what trade off between simulation complexity and systematic noise application is appropriate for the task at hand. We highlighted in the above discussion the importance of physical dynamics in the kite control problem. Contrary to the most extreme minimal modelling approaches (Jakobi, 1998), in problems similarly rich in physical dynamics such as the control of passive dynamic walkers (Vaughan, 2007) some considerable effort was made in order to render the simulation environment at least relatively accurate. In that case the best results are obtained when the most prior knowledge possible is incorporated, it seems sensible here to follow suit. Where possible, the ideal kite simulation is then one that captures as many of the salient dynamics as possible, ideally including some coupling between the deformation of the kite and the aerodynamic forces to which it is subject. As it is unlikely that we will be able to reproduce the exact coupling between wing loading and deformation, we can vary this property across a wide range to ensure that controllers are robust to the presence of a variety of deformation characteristics. Contrary to the majority of the work in the field, we cannot make simplifying assumptions such as treating the kite as a point mass. Instead we must incorporate knowledge of the physical system where it is possible under the computational constraints to which we are subject.
We here present the key characteristics of the kite that are desirable targets for replication in a simulation environment, in the form of a list for the sake of brevity. This list will also inform our later discussion of how successful transfer behaviour can be accomplished using a radical envelope of noise approach:

- The kite is an arc shaped wing.
- The wing will be subject to different aerodynamic lift and drag depending on its angle of attack.
- Under turning, different parts of the wing will be subject to different apparent wind speeds.
- Under turning, different parts of the wing will be operating at different angles of attack.
- The wing will deform in response to load.
- The wing is steered through constraining its roll angle and through deformation in response to the changing the lengths of four control lines.
- Because of limitations in the relative difference of the line lengths, the wing is constrained to flying at certain attack angles in different parts of the wind window.

Due to the physical complexity of the system, we need to acknowledge that a truly accurate model of the system is not currently feasible in the context of evolutionary robotics. However, in chapter 4, we will describe a simulation that is intended to incorporate as many of these key aspects as possible whilst being as computationally efficient as possible.

3.1.2 Concerns about generalisation

Although human kite flyers typically perform very poorly when first attempting to control a steerable kite with controllable pitch and hence considered 'depowerable', intermediate pilots are easily capable of controlling different kites that may have very different flying characteristics in a wide variety of conditions with only a short period of adjustment. This generalisation ability is not only highly desirable, but may be a pre-requisite for a successfully operating control system given the lack of control over the environmental variables that we have during the testing process, even whilst using an identical kite in each test. It is our contention here that attaining the goal of a *general* kite flyer will require neurocontrollers to be exposed to large and systematic variations in the simulated task during evolution. It will be a simple enough task to assess the generalisation ability of controllers between different instantiations of a simulated task environment to confirm or refute this assertion. The task variation will have to be implemented to such an extent that a scheme that will allow the development of such a general controller will naturally share some of the characteristics proposed by Jakobi in his REON process for successful transfer of behaviour from simulation to reality. How the demands of generality and transferability conflict or complement each other under this process will be considered in some detail in section 3.3.

3.2 Differences and similarities to other ER tasks

Here we will take the identified key characteristics of the kite control problem and assess the similarities and differences between this problem and other evolutionary robotics implementations in the literature. We can then at least be aware of issues that may arise, and if possible refine appropriate strategies to cope with any particularly problematic characteristics.

3.2.1 Unstable tasks, input selection and initialisation

One important characteristic of the kite control problem is its instability, i.e. that given no control input, the vast majority of steerable kites will wander from any starting position and hit the ground, usually within a matter of seconds, but certainly in under a minute. Being a flexible object on a tether with some element of elasticity, subject to complex non-linear aerodynamic forces this is perhaps to be expected. Whilst there are many tasks in the evolutionary robotics literature where simply doing nothing will result in failure, here we have the additional implication that doing nothing will in fact result in some change in the physical status of the wing. The position, velocity, rotation or deformation all could change, which will all alter the impact of any subsequent steering action. This could be summarised as active control being required simply for things to stay the same. It seems that there are relatively few tasks in the evolutionary robotics literature where this could be said to be the case. A particularly classic example where the problem is in fact unstable, although implemented only in simulation in its original formulation, is the pole balancing problem (see Figure 3.2) in which a putative cart must balance a pole, or even a jointed pole, as near to upright as possible (Wieland, 1991).

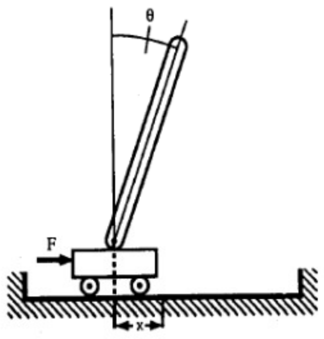


Figure 3.2 Figure from (Wieland, 1991) **demonstrating the pole balancing problem, where F is the corrective force to be applied to the cart, Θ is the angle of the pole away from perpendicular to the cart, and x is the position of the cart relative to a starting point.**

Because the pole is unstable even when completely perpendicular to the cart, the evolved solution is for the cart to perform oscillatory movements left and right, such that the pole to is oscillating around a near vertical position. It will be interesting to observe whether a kite controller approaches a stabilisation task with a similar strategy. In this study, provision to the neural network of instantaneous values for the angle of the pole and position the cart is sufficient for the successful completion of the task, given a moderately sized network of 10 neurons. However, when provided with derivatives of these values in addition to the instantaneous values, it was possible to evolve controllers with only a single neuron. Interestingly, the author used fully interconnected recurrent neural networks, albeit with instantaneous dynamics, to solve the control task. Unfortunately the author does not state whether it was possible to generate competent controllers through the use of feedforward neural networks. In another example of an unstable system, one with even richer dynamics, i.e. those of a passive dynamic walker, velocity values are also provided to the network by default (Vaughan, 2007).

In common with these unstable problems but at variance with the majority of control tasks within the ER literature, task timescales in the kite control problem are rapid. Either the lack of a appropriate control signal or the actuation of an inappropriate deleterious control signal can commit the neurocontroller to catastrophic failure in the form of a crash within perhaps one or two seconds. I refer to this characteristic as the task having 'hard edges', meaning that the agent must react more or less immediately in a given situation by steering somewhere near the appropriate direction. Any kind of an exploratory behaviour will be punished in a way that is not the case for tasks such as one legged walking in a near physics free environment, phototaxis or chemotaxis, object discrimination or even multi-legged walking tasks, where the multiplicity of actuators or the nature of embodiment creates leeway or redundancy in the control of the robot body.

3.2.2 Task decomposability

The examples cited above and that of evolved helicopter control (De Nardi et al., 2006) which could also be considered to be a rapid task, either have simple dynamics or when more complex dynamics do exist, significant amounts of prior knowledge are specifically incorporated in the structure of the network. Both walking and helicopter control can be decomposed into elements, relating to the coupling of joint or system rotational degrees of freedom to specific actuators. This allows the specification of what control action is appropriate in a given situation, and this prior analysis can be incorporated into the network and utilised effectively given the availability of a direct measurement or good estimation of the state of each DoF. At a gross level, it is also possible to achieve this with control of kites, however when linking specific input ranges or combinations of inputs, the ability to classify what is appropriate at a given moment frequently breaks down. This is especially true if the environment or kite is subject to significant variation which cannot be measured or accurately estimated. For example, if we wish to fly a kite in a 'lying eight' trajectory (see Figure 3.3), we might generate or attempt to encode a heuristic such as: if the kite is in the left half of the wind window, turn right, and vice versa if the kite is in the right of the wind window.

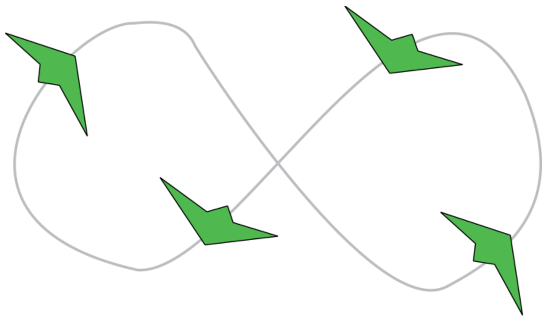


Figure 3.3 The simplest lying 8 trajectory, down loop at centre, uploop at edge of window

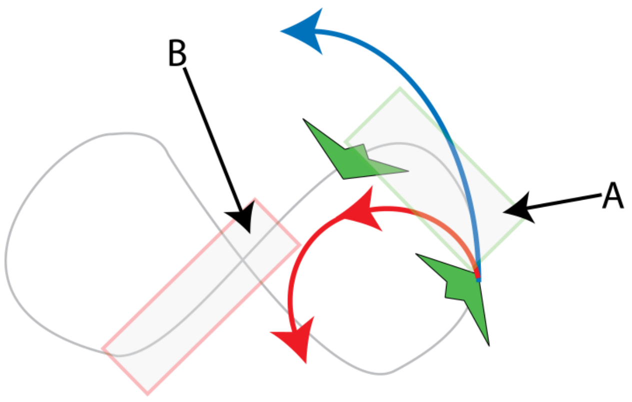


Figure 3.4 The two shaded areas mark 2 of the areas where steering transitions are required. In region A steering must be changed from a left turn to straight ahead and in region B from straight ahead to a right turn. The blue and red arrows show the consequence of mistiming or inappropriate amplitude steering during A: oversteer (red), or understeer (blue).

In reality however, the point at which the steering command is initiated and the extent of an 'ideal' steering command both in terms of amplitude and duration will vary greatly depending on the characteristics of the kite and the wind in which it is being flown. Figure 3.4 shows the consequences of over or under steer at a transitional phase in the loop, which could arise from mistiming the onset of corrective steering after the uploop, steering with an inappropriate amplitude, or persisting with the corrective steering for too long. Defining a rule for which direction of steering is appropriate at a particular point in the cycle is highly dependent on kite and wind characteristics. Larger, slower kites or kite operating in lighter wind will need to initiate the corrective steering far earlier in the cycle in order to bring the kite round of the uploop successfully, often paradoxically requiring the kite to be steered right when it is still in the right half of the wind window. With a fast kite or with strong wind however, the rightward steering needs to be delayed until the very last moment.

Here we are beginning to approach the nub of the problem, environmental variation is severe, unmeasureable and severely affects the dynamics required for control. Additionally variation in the embodied state of the robot i.e. the physical attributes relevant to the agent/environment coupling is greater than in most evolutionary robotics tasks. Given the instability of the system to be controlled and the speed at which the appropriate control signal must be settled upon, the natural approach would be to decompose the problem using as much prior knowledge as possible. However this very variation hinders both the decomposition of the task into subtasks which might be reflected in modularisation of the network and the description of an appropriate control system through a set of heuristics that might be implemented in a fitness function.

In addition we have the added burden of developing a controller for a task that is impossible to model accurately with current computer technology, full volumetric CFD models being several orders of magnitude too slow for an evolutionary process. A full description of aeroelasticity i.e. load dependent deformation of the wing would demand a further computational burden. We are forced therefore to use a minimal model, bearing in mind that we expect that there are some essential aspects of the physical embodiment of the kite that a competent controller must incorporate some implicit knowledge of, without ourselves having prior knowledge of which aspects are most important or mandatory. Therefore our best bet appears to be, to try and build a parsimonious model that incorporates as many of the salient aspects of kite flight as we can afford. Acknowledging the incompleteness and inaccuracy of this model, we must then expose controllers during the evolutionary process to even more intra-trial variation i.e. kite deformation and change in wind conditions, than we expect to be present in the real world.

In summary, we can see that individual aspects of the kite control problem are present in other tasks in the ER literature, in particular instability and rapid time to potential failure. We can seek to implement solutions that have proved to be useful in dealing with these problems in the literature, notably providing the network with velocity signals, and incorporating prior knowledge wherever possible. However there are notable difficulties in the kite control problem, specifically the results of a specific control action, being highly dependent of the existing body state, position within the environment and independent environmental conditions. These difficulties hinder decomposition of the task into stages that might render the evolutionary process less onerous. Additionally, the complexity of the physics coupled with the underlying requirement to minimise computational demand in any evolutionary process creates difficulties in knowing which aspects of the agent/environment interaction should be incorporated. It might be that inadequately modelling a physical process creates an artefact within the simulation that controllers may learn to rely on, a situation to be avoided at all costs. Below, we discuss how this might be avoided.

3.3 Concerns regarding crossing the reality gap

In this section, we discuss how the issues raised in the previous sections impinge on the transfer of neurocontrollers from simulation to reality, with specific reference to implementing a systematic scheme such as Jacobi's radical envelope of noise.

3.3.1 Key characteristics of concern

As described above, because of the instability of the problem and the importance of physical dynamics, we expect a true minimal model to be unavailable to us, in terms of disregarding much of the real world instantiation and its physics. Additionally, we strongly suspect that there are some aspects that comprise what Jakobi would term the base set, at least a subset of which must be replicated in the simulation, whose interactions are non-trivial, and the implementation of which will constitute a level of simulation beyond that are normally considered minimal. The most minimal model then that meets the requirements of real world-like dynamics, and the particular constraints of this problem is likely to be a test environment based on real world physics. In this case, the model will resemble the target system more closely than in Jakobi's ER implementations.

At the same time, we know that any model will be inadequate, or at the least inaccurate, in terms of replicating the fine dynamics of kite flight in simulation if it is to be computationally feasible. Still, there is a range of compositional complexity open to us although the level of accuracy or detail which we should implement and model is not necessarily obvious. The known inadequacy of the model, coupled with potential interactions of the putative essential elements, creates the risk of creating artefacts present in the simulation that are not present in reality.

Being aware of the limitations and requirements, once again the conclusion is simply to create the most parsimonious model replicating what we consider to be most likely to be important within reality but seeking to pre-empt likely artefacts or implementation aspects wherever possible. Knowing these characteristics, we can now assess the suitability of the radical envelope of noise approach based on a description of its operation.

3.3.2 Is this type of problem compatible with REON?

To answer how these characteristics impinge on a systematic use of minimal modelling and radical envelope of noise, we first delineate the approach as described by the original proponent. Usefully, in one of his studies (Jakobi, 1998), Jakobi summarises the methodology in seven steps which are replicated in the list below:

1. Precisely define the behaviour desired
2. Identify the real world base set
3. Build a model of the way in which the members of the base set interact with each other and react to controller input
4. Build a model of (enough of) the way in which the members of the base set affect controller input.
5. Design a suitable fitness test.
6. Ensure that evolving controllers are base set exclusive
7. Ensure that evolving controllers are base set robust

Whilst Jakobi's approach is certainly exceptionally elegant as a strategy for overcoming the challenges of crossing the reality gap, and is highly applicable for the tasks presented within his work, it is not clear whether the approach as spelt out in the seven methodological steps above is applicable to all tasks. For example, even the first step is not as straightforward as it might appear. It may not be a trivial matter to define the behaviour desired, let alone to any level of precision. In the T-maze task for example (Jakobi, 1997) it is clear exactly which behavioural steps right down to the level of motor control signals of each wheel are appropriate at any given stage within the task sequence, rendering the design of fitness function and the removal of much of the real world detail relatively easy. However if we are not sure how the robot is interacting with the environment, if this robot/environment interaction is subject to change or if the cheapest valid description is that of a complex physical model with complex non-linear dynamics, we are already somewhat stuck.

For the second step, it is usually possible to define some features of the task that are relevant to a controller trying to achieve a behaviour, and that can be replicated at least approximately in simulation. However it is no means a certainty that all of the relevant features will be able to be identified. By forcing the controller to rely on a subset of the available features, we may inadvertently render it unnecessarily difficult for evolution to build a valid solution. Simply by underestimating the importance or relevance of some aspect to the task solution we might end up withholding aspects of the environment or agent environment/interaction that might simplify the task or facilitate a good control strategy. In the strapline to this step within his text, Jakobi encourages us to 'identify the way in which the members of the base set interact with each other and react to motor signals during the performance of the behaviour'. From the perspective of indoor robotics, on flat surfaces, with agents with comparatively slow dynamics, and whose physical configuration is easy to relate to the goals of the task, this does not seem an unreasonable request. However in the work presented here, it is by no means a trivial feat. In fact it is probable that if the way in which the base set interacted and responded to the robot state and output could be fully defined, it would be possible to create a full system model and a hand designed or analytically derived controller without resorting to evolutionary robotics. Given the difficulties that are present in our case, a weaker version of the REON methodology may have to be implemented, especially given that the two next steps are to build a model of these interactions. For the fifth step, the requirement for a fitness test is common to all evolutionary robotics implementations, and this will be considered independently. The sixth step is subject to similar problems as the second in that identification of implementation set aspects is not necessarily a clear-cut procedure. Additionally extreme variation of the parameters that are identified as difficult to replicate or potentially absent in reality, may render our model unstable or unrealistic, especially if parameters are interdependent thus causing difficulties in predicting combinatorial effects. The final step is certainly very important and noise on the base set parameters is used in every real world evolutionary robotics implementation, Jakobi does however give a salutary warning as to the difficulties posed to the evolutionary process of too much noise in the base set.

In summary, it appears that the most fundamental version of Jacobi's methodology, in terms of predefinition of specific behaviour and the stripping away of much of what might normally be considered necessary in a model may be inappropriate or simply not implementable for this task. However, the systematic identification of a base set and implementation set of properties in the simulated task environment does seem likely to successfully increase the probability of crossing the reality gap. Given the lack of a clear definition of the system or the desired behaviour in our case, there may be a considerable cost in terms of added difficulty to the evolutionary process if we are to follow this methodology. The section below considers some of these more practical aspects of the implementation.

3.3.3 Considerations for crossing the reality gap with a poorly defined problem, rich agent/environment dynamics and difficulties in modelling

Here we delineate the considerations that must be taken to increase the chance of successful transfer behaviour from simulation to reality in this specific problem. Below we list the factors that might impinge on the dynamical relationship between the kite and its environment, given active control with a robot. Once identified, we can assess the means of integration of these factors into the evolutionary process.

There may be a need to incorporate aspects of the environment, embodiment of the physical system and the sensorimotor loop in order to expect reasonable transference of behaviours. These are:

- Environmental aspects
  - Wind speed
  - Wind change with altitude, i.e. wind shear
  - Wind direction
  - Air density
- Aspects regarding physical instantiation of a kite
  - Rigidity of kite
  - Weight of kite
    - Mass distribution of kite
  - Aerodynamic properties of kite
  - Line length
  - Line positioning at ground level
- Robotics hardware implementation aspects
  - Sensor delay
  - Sensor noise
  - Sensor sampling rates
  - Sensor or ADC resolution
  - Processing and communications delays
  - Motor update sampling rates
  - Motor performance characteristics

The environmental and kite related aspects are unambiguously going to be present in the real world and contribute to the dynamics of the kite in response to control output. Most of, if not all of these parameters will be subject to change in a real world implementation. Robotics hardware implementation aspects will certainly exist in the real world, but should be more stable between and within trials. Jakobi exhorts the evolutionary robotics practitioner to essentially disregard these aspects (Jakobi, 1998) but there may be justification for doing otherwise in this case. For example, if we are to use inputs that are responding to some available characteristic of the system, but that parameter is sampled discreetly at some rate that is not identical to the update time step of the neural network, then the input to the neural network will be necessarily jagged. This is especially pronounced for velocity inputs, unless some state estimation algorithm or smoothing strategy is employed.

The lower the rate of sensor update relative to the speed of environmental change, the more material the difference in input is. In the case of sensor delay, it is difficult to justify the assumption of instantaneous magic sensors if we know that some complex pre-processing occurs on the raw data. Such processing, for example a computer vision algorithm for tracking the kite, will mean that data regarding the kite's characteristics is necessarily always out of date and possibly out of sync with other sources of data. For a task in which an appropriate control output is tightly coupled to the state of the kite at a given moment, it does seem natural to incorporate sensor delay within our simulated task environment if it is known that one exists in the real world. The same rationale applies to the modelling of motor characteristics, we know that the motor cannot move instantaneously to a given position, in fact it may take over a second to reach its target position, especially when under load. Whilst a precise model of a given servo motor may be superfluous, incorporating prior knowledge of non-instantaneous motor speed does seem to be a sensible path of action.

This assessment tells us that there are a number of aspects of the system that will need to be varied during the evolutionary process such that any controller is not limited inadvertently to operating only in certain subsets of the potential conditions. Being 'real' aspects, these might be considered as base set variations, although the scope of the simulation is greater than a typical minimal model. Equally, we know that any simulation that we create of a deformable wing operating in real world air flows will be inaccurate. The implementation of the simulation then will need to be mistrusted, necessitating severe noise on what are true implementation set parameters .

This requirement for such a wide range of both 'base set' parameter variation and imposed noise will have two consequences, the first is that in such an unstable problem if a controller is exposed during the evolutionary process to the full extent of both these types of variation, it is likely that there will be too little information within a sensible number of trials for any neurocontroller to evolve at all. The second is that in the process of imposing noise on the physics simulation, we might end up creating physical systems in the simulation environment that are not kite-like. This will unnecessarily render the problem more difficult than it needs to be. In the worst case, the task parameter changes may render the problem generally qualitatively different in dynamics to that of real kite flight. In that case a selection pressure would be created tailoring neurocontrollers for the wrong task, thereby falling into the very trap the application of noise and variation is trying to avoid.

The first problem, that of the initial task requirement being simply too difficult should be addressable by shaping carefully managing or shaping the task, in order to gradually release difficulty. Although there are many examples of this in the literature of evolutionary (Fine & Di Paolo, 2007),(Bongard, 2008), and developmental (Lee et al., 2007) robotics, the manner in which this is done may need to be finely divided in a more systematic or fine grained manner than simply changing a fitness function once or twice per evolutionary run. Because the character of the task may change in unforeseen ways during this process, an unwanted side effect might be the convergence of the population on locally optimal solutions relying on strategies that are rendered completely ineffective with the requirements of the task. It might be necessary in order to allow the escape of evolved solutions from such local optima by allowing structural changes to the neural network above simple weight mutation, for instance the addition and deletion of synaptic connections and entire neurons. An alternative approach would be to allow the option of stepping back the increment in difficulty, and incrementing a different subset of parameters, (Bongard, 2008).

To address the second problem, that of rendering the task either unstable or completely unrealistic when adding parameter noise, two options present themselves, of which the simplest is manually initialising the task environment at the extremes of selected parameter ranges and interacting with the kite system as a human operator to check if the performance is remotely plausible. Naturally not every single combination of parameter ranges can be manually verified in this way thereby leaving the possibility of implausible task scenarios being presented, whilst averting the possibility of a systematic warping of the dynamics of the task. The second option would be to have some kind of parameter estimation scheme in which some aspects of the physics model are calibrated against some known measurements. This would provide a reference point around which variation and noise could be centred, at the risk of leading the operator into believing that the task scenario is more accurate than it really is.

To briefly recapitulate this section, we have argued that the nature of flight dynamics demands some key aspects of the system to be incorporated if we are to expect the evolution of appropriate behaviours. In this respect, the model will be considerable less minimal than Jakobi argued was adequate. Second we also argued that elements of the hardware that affected sensing or actuation should be explicitly considered due to the speed and instability of the system dynamics. If these task and hardware aspects truly are important however, we need to ensure that controllers are valid across wide ranges of the parameters concerned, necessitating their systematic variation. Finally the difficulty that this variation engenders will have to be mitigated using considered task management during evolution, including shaping schemes and the use of multiple trials per evaluation.

3.4 General implementation related considerations

More prosaic practical limitations might include the limited suite of sensors available to us primarily due to financial constraints. Specifically we have no on-board measurement of orientation that might be provided by gyroscopic sensors, accelerometers and magnetometers in a typical inertial motion unit. Naturally this means that attaining a state estimation vector that is typical to traditional unmanned aerial vehicle methods is impossible. Recent work (Leven, Zufferey, & Floreano, 2009) however has described a minimal reactive control approach without a sensor fusion algorithm and full estimated state vector for control of a free flying wing.

3.5 Chapter summary

There is a definite requirement to be able to transfer neurocontrollers from simulation to reality whilst retaining the same qualitative behaviour of flight control of a kite. However, we have noted some key characteristics of the kite control task; instability of the system to be controlled, the fast dynamics and highly coupled nature of the wing and its environment in determining both uncontrolled dynamics and response to a given control input, and the fact that an adequate model is almost by definition out of reach of the computational budget of a typical evolutionary robotics implementation. We have assessed characteristics as being only partially compatible with the radical envelope of noise approach of Jakobi, and proposed a pragmatic approach to dealing with some of these issues, some of which will be explored in the experimental process.

4

There is no greater anomaly in nature than a bird that cannot fly, yet there are several in this state.

Charles Darwin, paraphrasing Prof Richard Owen

4 Evolving neurocontrollers for a simulated kite control task

In this chapter we introduce the core of the particular evolutionary robotics methodology that will be common to the majority of the work within this thesis, and apply it to a kite control task in simulation. In this early work, the neurocontroller must control a kite in the generation phase of the generation/retraction cycle, where the task parameters described in Chapter 3 are subject to a restricted range of variation. This serves to assess the suitability of the approach for producing worthwhile neurocontrollers and may highlight the need for incorporating some of the possible solutions to issues particular to this type of problem that were identified in the previous chapter. The brief for this initial foray is to apply 'off the shelf' ER techniques to the problem of controlling a simulated kite in order to maximise aerodynamic forces that act upon the tether. In the first instance, the aim is to see if stable repeated trajectories are evolved and to assess the qualitative similarity to flight trajectories in the literature. We then go on to probe the ability of the evolved controllers to generalise to variation in the length of the flying lines and deviations in the wind direction.

To this end, we present here a simple aerodynamic simulation of a steerable four-line kite, we then employ an evolutionary robotics approach in order to maximise aerodynamic forces acting along the same vector as the lines. This metric selects solely for productivity, without imposing any preconceived notions of what is an appropriate flight trajectory. Initially naïve, randomly configured neural networks are evolved using a microbial genetic algorithm (Harvey, 1992) through selection and mutation of the controllers that produce the greatest mean aerodynamic forces over a given test period. Resulting controllers should steer the kite in consistent looping trajectories, which prior work has demonstrated are optimal paths for maximising energy recoverable from the wind (Houska & Diehl, 2007), albeit for a kite simplified point mass kite model. The controllers should also be robust, being able to maintain stable trajectories even with significant changes in the wind velocity.

4.1 Methodology

The software implementation consists of three core components, the kite physics simulation, the genetic algorithm and the neural network, their interrelationship is summarised in Figure 4.1. All three components were implemented from scratch using the MATLAB technical programming environment, which is quick to develop but has poor performance in terms of speed of program execution, at least where code involves repetitive iteration that cannot be vectorised. All evolutionary experiments in this chapter were run on the 20 x 3GHz Xeon CPU Linux cluster operating at Sussex University, one core per evolutionary run, each run taking several days of CPU time.

The kite itself is simulated as a curved airfoil, which viewed from the front forms a semi-circular arc (see Figure 4.2). The kite is tethered to ground level by 4 lines, and controlled from the ground by adjusting the relative lengths of the rear 2 lines. The kite is allowed to flex and following the common Leading Edge Inflatable (LEI) kite configuration, in which the leading edge is an inflatable air beam, the leading edge of the kite that normally faces into the wind is more rigid than the trailing edge opposite. Line tension and angle data is fed to the neurocontroller, which feeds back line length actuation to the kite model as per Figure 4.1.

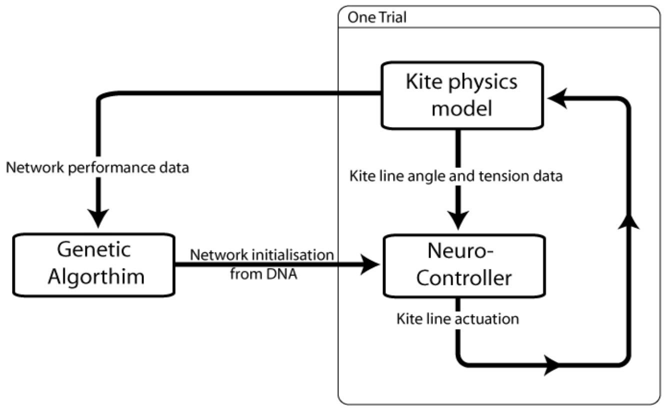


Figure 4.1 A simplified schematic of the system in which neurocontrollers are evolved.

4.1.1 Kite physics simulation

In contrast to all the studies that existed in the literature prior to this work in which the kite was treated as single static point mass (Loyd, 1980),(Houska & Diehl, 2007),(Canale, Fagiano, & Ippolito, 2006), the method of choice for this work was to use a multi-body or particle based simulation. As discussed in chapter 3. the motivation for this decision was to provide a framework which allows explicit consideration of variation in the kite configuration in terms of kite shape, bridle setup if any, and physical properties of the kite such as relative rigidity and mass of kite components. This allows room for specific anticipated defects to be introduced to the system at a later stage and the adequacy of the controllers' reaction assessed. The kite is initialised as repeated rows of equidistant particles in a semicircular arc as shown in Figure 4.2, which illustrates the default setup of two rows of 5 particles.

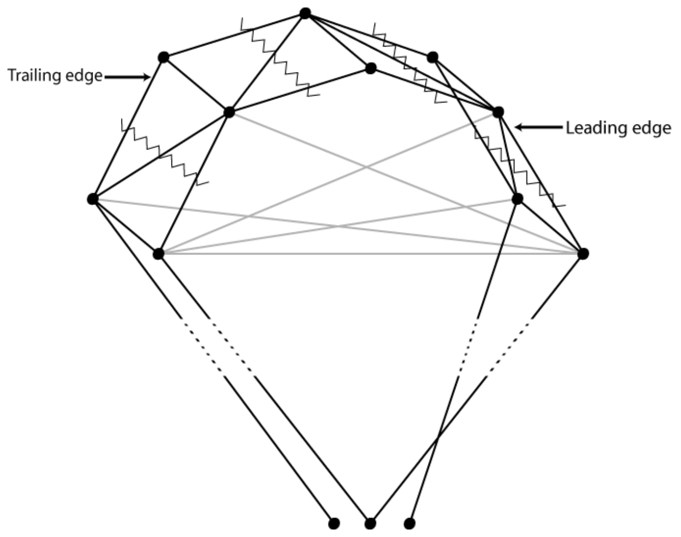


Figure 4.2 The initial configuration of particles (circles) and constraints (straight lines). The light grey constraints reinforce the arc shape of the kite and prevent 'jellyfish' type flapping motion, effectively performing the same role as the inflatable ribs that maintain the shape of LEI kites. The three lowest particles are the tether points. Zigzag lines indicate the positions at which the canopy is sliced for aerodynamic calculations.

For simulation purposes we make a discrete update of the state of the system at small increments of time in order to approximate the continuous system. Unless otherwise stated, the value of the timestep dt used in these experiments is 0.004 seconds. The kite canopy is split into five sections or slices as demarcated by the zigzag lines running from the leading to trailing edge in Figure 4.2. The aerodynamic forces at each timestep are then calculated for each slice independently. By slicing the kite up in this way, the force on a section of canopy depends on its particular angle of attack and the apparent wind velocity to which it is subjected. Treating the canopy as independent sections also allows the possibility of using different sets of aerodynamic lift and drag coefficients at different portions across the wing span although in these initial experiments the same coefficient values are used on all of the wing slices. The aerodynamic and gravitational forces are assumed to be distributable equally amongst the particles pairs that constitute each slice, although as described below this is a simplifying assumption that ignores any pitching moments acting on the aerofoil.

4.1.2 Multibody or constrained particle system

The acceleration on each particle is simply determined by Newton's second law of motion. In addition to the acceleration due to aerodynamic forces, gravity is incorporated by accelerating each particle -9.81m/s^2 in the Z axis. This is described in Eq. 4-1 in which the acceleration a on a particle p, is the sum of the aerodynamic and gravitational forces, F_s^{Aer} and F^{Grav} respectively, acting on that slice s, divided by the mass of the slice M_s and scaled by the number of constituent particles of that slice P_s.

Eq. 4-1

$$a_p = \frac{F_s^{Aer} + F^{Grav}}{M_s} \frac{1}{P_s}$$

Integration is performed according to the velocity free Verlet method (Press, 1993) as per Eq. 4-2; this method is used due to its relative stability and speed of execution. x_p here simply represents the position of particle p, as mentioned previously the time step Δt is kept small at 0.004s to avoid numerical instability.

Eq. 4-2

$$x_p^{t+1} = 2x_p - x_p^{t-1} + a_p \Delta t^2$$

The very diversity in forces across the canopy that is being encouraged by the slice system will quickly cause the particles to scatter, given that the dominant lift force is generally normal to the leading edge vector of the slice. It is therefore necessary to constrain the particles to maintain the coherence of the kite structure. The constraints linking the particles are treated as infinitely stiff springs, and their positions are iterated according to the Gauss-Seidel relaxation method (H. Jeffreys & Jeffreys, 1988). The system can be described in three steps for a given constrained pair of particles:

1. Calculate the distance between the particles
2. Calculate the mismatch between current distance and constrained distance
3. Move the particles along their current relative vector such as to restore the constrained distance

This simply moves particles along the vector that links them in order to satisfy the constraint, i.e. two particles 2 units apart but with a constraint distance of 4 will each be moved 1 unit away from their original positions along the vector between them. By iterating around the set of constraints a number of times the system can be forced to converge to its exact initial configuration in terms of relative distances of particles, and hence return the wing to the exact geometry in which it was initialised. However, real kites have some flexibility and the warping of the shape of the wing clearly contributes to the subsequent aerodynamic force imposed upon the wing. Therefore we are ideally not seeking complete rigidity in the wing, but for the wing to deform in response to a variable load imposed on a semi compliant structure. For this reason, the constraints are only iterated once per timestep. One further measure that is implemented here is for the system to be set to respect the relative masses of the particles. This means that the total length 'correction' for a given constraint enforcement is the same, but is split between the particle pair dependent on their mass. This is in fact necessary to prevent what might be unrealistic accelerations of heavy ground based elements for example.

Due to the arbitrary nature of this system, it is left to the user to adjust the way the constraints are configured in order to generate the most kite-like warping behaviour and hence flight. Real aerofoils develop most of their aerodynamic force towards the leading edge, and as the profile of the kite is particularly important here for aerodynamic reasons, the leading edge in real kites is often stiffened or reinforced. In the case of LEI kites, the whole leading edge consists of an air beam, and parafoils are typically heavily bridled towards the leading edge in order to distribute the load more effectively. Whilst it would possible to model an air beam as a sequence of torque springs, for simplicity and to minimise the computational demand, here we try and approximate the effect by adding additional constraints preferentially at the leading edge. These are shown in Figure 4.2 as grey lines and clearly have no physical counterpart on a real kite. Applying this span-wise stiffening heterogeneously at the leading and trailing edges, allows the trailing edge of the canopy that is less constrained to flex more easily than the front, which will itself flex slightly at the upper range of aerodynamic forces.

The adoption of this particle method has some possible drawbacks worth brief consideration due to their potential to destabilise the dynamics or introduce unwanted artefacts. A particular concern is that the effect of implementing one constraint pair may be altered by the implementation of the second constraint pair if one of the members of the pairs is shared. In fact this is required in the Gauss Seidel relaxation method because calculations are sequential and not independent, i.e. the results of the first constraint enforcement are carried through to the second and so on. Because of this, it is possible and in fact likely, that the effect of constraint enforcements might differ depending on the order in which they are enforced. Therefore, to ensure stability of the constraint system and to remove any bias that might systematically bias the change in shape of the kite on one side of the wing span versus the other, consideration must be given to the order in which the constraints are enforced. Repeating the same order of constraint enforcement over and over will tend to accumulate any effect caused by the contradiction of constraints and therefore should be avoided. In calibration of this process, some methods of constraint ordering were attempted. It was found that neither repeating the same order, nor randomising the order of constraint pair enforcement was conducive to stability. However by simply generating an order of constraints and then reversing that order in the next timestep, stability could be obtained. It did not seem to matter whether this order was generated by a random process or if the constraint satisfaction order was simply inherited from the process of their generation.

It is worth noting that it is also possible and probably desirable to allocate particles to represent the lines and therefore include the effects of their drag and momentum upon the kite, and additionally allow sagging of the lines when under low tension. This was avoided in this study due to the additional computational overhead. The single constraints that constitute each line are one-way, only being enforced when the lines exceed their initial length. If the current line length is less than the reference constraint length, then it means that there is effectively some slack in the line and the constraint is therefore not enforced.

4.1.3 Aerodynamics model

The forces upon each slice are determined according to a simplified aerodynamics model. This model is simplified in that no moment coefficient is used and that the aerodynamic and gravitational forces are applied equally amongst the slice's constituent particles. The value of the lift force for each slice is derived through Eq. 4-3 and the lift force always acts in a direction perpendicular to the apparent wind vector as per Eq. 4-4. In the model, the direction of the lift force vector F_L is given by the cross product of the apparent wind vector and the vector that describes the slice's leading edge, $\bar{a}$ and $\bar{e}$ respectively in Eq. 4-4. The drag force F_D is always in line with the apparent wind vector; therefore a unit vector of the apparent wind directs the drag force upon the particles as per Eq. 4-5. A is the slice's area, V_a the apparent wind velocity, ρ the air density, C_l and C_d are the lift and drag coefficients at the current angle of attack.

Eq. 4-3

$$F_L = \bar{L}\frac{1}{2}C_l \rho V_a^2 A$$

Where

Eq. 4-4

$$\bar{L} = \frac{\bar{a}}{\|\bar{a}\|} \times \bar{e}$$

Eq. 4-5

$$F_D = \frac{\bar{a}}{\|\bar{a}\|}\frac{1}{2}C_d \rho V_a^2 A$$

The lift and drag coefficients are read from a lookup table according to the slice angle of attack, values are plotted over all angles in Fig.4. Values were generated using the X-plane® Airfoil Maker version 860 software, using typical traction kite characteristics of moderate camber and thickness and relatively high drag and lift, giving a relatively poor glide ratio peaking around 5.

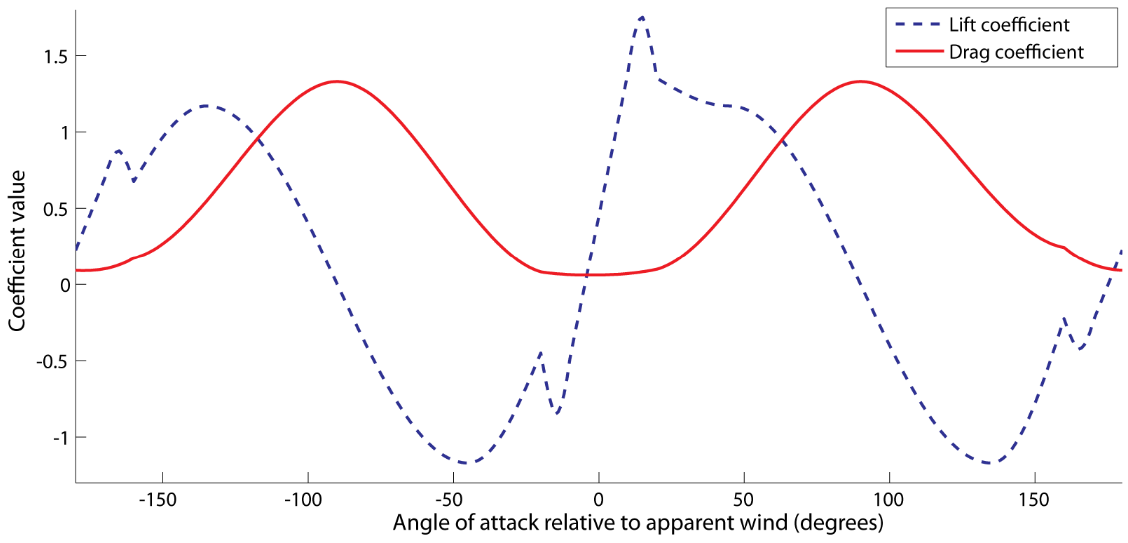


Figure 4.3 A plot of the lift and drag coefficients used, over a full range of 360 degrees.

We do not allow angle of attack α to be actively changed by the neurocontroller, although this could be achieved by changing the relative lengths of the front and back lines. Following (Houska, 2007) and as discussed in chapter 2, we use a simple model of wind shear to modify the wind speed with altitude giving the wind speed w at height h, where h_r is the roughness length which is terrain dependent, and v_o is the wind speed at the reference altitude h_0.

4-6

$$w(h) = \frac{\ln\left(\frac{h}{h_r}\right)}{\ln\left(\frac{h_0}{h_r}\right)} v_o$$

For this initial investigation, a simple arbitrary gust generation model is used that generates deviations around a base windspeed of 8m/s. Each timestep has a small probability that a gust or lull is initiated, if so its maximum deviation, duration, attack and decay values are set randomly. Gust amplitude can be +/- 25% of the winds speed at the beginning of the trial. Decay and attack for each gust event are set independently as being in one of two modes; linear or exponential. This means that for a given gust the onset could be fast but with quick decay, or the onset might be slow but exponentially increasing with linear decay for example. Attack and decay can be so high that the gust or lull is effectively instantaneous, but will be set such that an individual gust would reach it's predetermined maximum or minimum value within the length of the trial. At each subsequent timestep the windspeed is altered by a small portion of the difference between the current windspeed and the predetermined maximum gust/lull value using these attack and decay characteristics. In this version, a new gust or lull can interrupt one that is already initiated, but the two cannot run simultaneously. A typical set of wind traces generated with this method are shown in Figure 4.4, These are not intended to be a realistic reproduction of the way windspeed varies in the real world, however they are intended to expose the neurocontrollers to a wide variety of variation during the evolutionary process. In the latter set of experiments in this chapter, we also allow probabilistic deviations in the wind direction of up to 20º using the same set of heuristics.

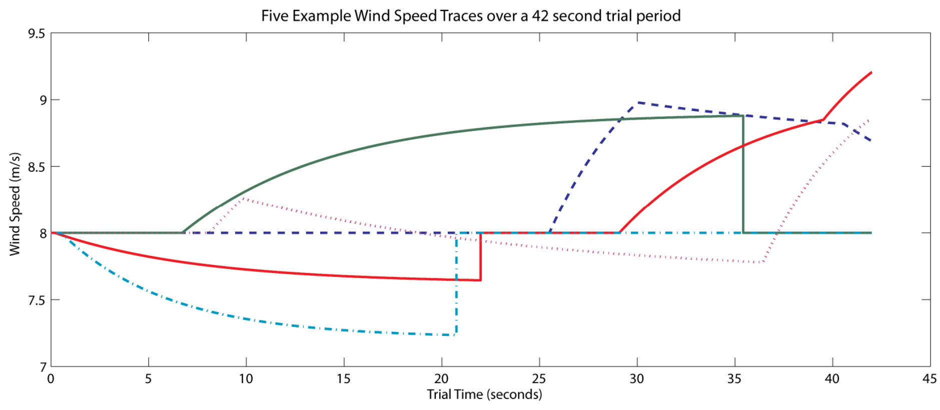


Figure 4.4 Five examples of windspeed variation for a 42 second trial period, generated using the simple heuristic described

4.1.4 Neurocontrollers

Two classes of neurocontroller were assessed, both are small recurrent neural networks of 5 input neurons and 7 fully connected interneurons, both inhibitory and excitatory connections are permissible. In these experiments only data measurable with putative line angle and tension sensors at ground level is made available to the network, as described in Table 1. These were chosen according to what we assessed might be available after a period of hardware development subsequent to this work. For the tension inputs we make the assumption that the line tension T developed will be the forces on the kite that are in line with the lines, as given by Eq. 4-7 where s is the index of the S slices being used for the calculation and $\bar{x}$ is the average line vector of either all the kites lines or the lines to the half of the kite that is being calculated. All the sensor neuron values here immediately assume the latest value from the simulation with no delay and have no internal dynamics. The connection weights from the input layer to the recurrent layer therefore act as evolvable gains with which to scale the input values.

Eq. 4-7

$$T = \left(\left(\sum_{s=1}^{S} F_s^{Aer} \right) + F^{Grav} \right) \times \bar{x}$$

Input No.	Input Data
1	Total line tension
2	Tension difference between left and right line sets
3	Average line azimuth
4	Average line elevation
5	Difference in elevation between left and right steering lines

Table 1 List of inputs to the neural network

The simplest neurocontroller class was a discrete time recurrent network whose nodes' activation value at a given timestep t is given by Eq. 4-8. Note that there is an implicit delay of exactly one timestep in the recurrent connections because in a given timestep, the inputs from other neurons in the fully interconnected hidden layer necessarily come from the activations of the last timestep. Although this approach is taken in other studies in the literature e.g. (Wieland, 1991),(De Nardi et al., 2006), it is not really desirable for a real world robotic implementation, because the evolved network is not likely to be robust to changes in the update timestep which might be harder to guarantee in the real world. For this reason, the second neurocontroller class tested was a continuous time recurrent network (CTRNN), in which a single neuron derives its dynamics from Eq. 4-10: CTRNN neurons are integrated using Euler integration at the same small timestep as the physics simulation.

Eq. 4-8

$$a_j^t = \sigma\left(\sum w_{ij} a_i^{t-1}\right) + \theta_j$$

Where

Eq. 4-9

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Eq. 4-10

$$\tau_i y_i = -y_i + \sum_{j=1}^{N} w_{ji} \sigma\left(y_j + \theta_j\right) \qquad i = 1, \dots, N$$

In both cases θ is the bias term, w the weight and σ is the logistic sigmoid function which scales values to a range between 0 and 1 as per Eq. 4-9. In the CTRNN equation, τ_i is the neuron's time constant, and y_i the old activation value from the previous timestep. One of the neurons from the output layer is designated as the output neuron. For the purpose of assigning a motor output value, 0.5 is subtracted from the sigmoided output of this output neuron and this value is then doubled to bring it into the range [-1 1]. The motor position as a proportion of its range is then modified by 1% of the difference between the network output and its current position each millisecond. This process has three consequences; the motor output is shielded from the majority of noise in the network, the motors are prevented from moving at speeds that are unrealistically fast and finally the extremes of motor output that correspond to a 1m difference in line lengths in these trials, usually destabilising for the kite, are only rarely achieved. This method of translating network output into motor position was modified in later work.

4.1.5 Genetic algorithm

The evolutionary algorithm is a simple tournament based microbial GA (Harvey, 2011), with DNA strings composed of real values determining weights between all neurons and sensory inputs, thresholds, neuron bias values and in the CTRNN case, time constants for each neuron in the network. The slope of the activation function is not subject to evolution in these experiments. The GA uses the gust generation heuristic described in section 4.1.3 to generate a wind trace the length of the trial, to be used to evaluate both individuals. Two individuals are then selected at random from the population and their fitness, the average line tension given by the cross product of the aerodynamic force vector with the average vector of the four lines, is evaluated for both individuals using identical conditions.

The evaluations in these experiments always begin with the kite initialised at the zenith position, i.e. directly above the tether points. The task proceeds until the end of the trial or until a termination condition is met. In the real world, continued flight might be possible if the kite brushes the ground, but determining this is beyond the capabilities of the simulation. Therefore the test is terminated prematurely as soon as any component of the kite meets ground level, regardless of whether the simulated kite is capable of proceeding in flight or not, to bias the evolved solutions against nearing the ground at all. The individual out of the evaluated pair with the poorest fitness has its DNA string copied over by the winner's DNA with a small random mutation in the range [-0.01 0.01] applied individually to every value in the string. Here we employ no crossover operators or any other operators specifically designed to modify the structure of the network.

4.2 Results

Here we present the results of evolution in three stages, in the first two stages the line length is kept constant at 25m to match typical commercial kite lines. Initially we assess performance after a limited period of evolution and subsequently after an extended period of evolution. We demonstrate the robustness of the evolved flight control to various types of perturbation, and also compare the performance of different classes of neurocontroller, in terms of evolvability of fit solutions within a given number of generations. In the final section we extend the scope of evolution to allow multiple line lengths in order to assess the generalisation ability of the neural networks.

4.2.1 Requirement for fitness function modification

In the preliminary tests, the fitness function employed was:

Eq. 4-11

$$F_A = \sum_{i=1}^{S} T_i \Big/ t_{term}$$

In F_A is the fitness score of neurocontroller *A*, *S* is the maximum number of timesteps, T_i is the line tension in timestep *i* and t_{term} is the number of the timestep in which the trial is terminated. Essentially this fitness function scores a neurocontrollers based on the average line tension over the length of the trial.

The whole population in every evolutionary run would rapidly converge on a clearly suboptimal strategy of steering the kite directly into the ground. Every evolutionary run became trapped in this clearly suboptimal strategy. Investigating this result, it became apparent that it was the operation of the scoring that was causing this effect. Specifically the score was generated by dividing the sum of line tension values over the trial by the number of available values. This effectively rewarded controllers that steered the kite directly at the ground as the kite speed and hence line tension would be high on average *over that period*. More conservative neurocontrollers that kept the kite aloft but relatively motionless would have been penalised under this scheme. As a result the fitness function for all the subsequent experiments was modified to divide the sum of line tension values by the maximum possible number of values were the trial to take its full 42 second duration as per equation 4-12 below, where tmax is the maximum trial duration.

Eq. 4-12

$$F_A = \sum_{i=1}^{S} T_i \Big/ t_{max}$$

4.2.2 Performance after 200 generations, 25m line length

In this section we show the evolved flight trajectories and demonstrate how neurocontrollers responded to gusts of wind 50% greater than allowed during the evolutionary process. These results, first presented in (Furey & Harvey, 2007) are following 200 generations of evolution with a population size of 30 and only one flight test being used per fitness evaluation. Variation in the wind speed is implemented as described in section 4.1.3 but no lateral deviations in wind speed are employed. In all the plots in this document showing the kite's flight trajectory when controlled by the evolved networks, the convention followed is for the position of both the left and right tip of the leading edge of the kite to be plotted, the left tip from the perspective of a pilot standing at the tether point is always shown in red, the right tip is shown in blue.

Figure 4.5 shows that the highest scoring neurocontroller from 10 concurrently executed evolutionary runs steers the kite in a repeated looping pattern. This could be described as an approximately figure eight shaped trajectory that is skewed at 45 degrees and offset away from a directly downwind position, over to one side of the wind window. It maintains this trajectory closely throughout the trial. The response of this controller to changes in the wind speed is shown in Figure 4.6 which shows the path in which the kite is flown before, during and after a single gust of an amplitude of 50% more than permitted during the evolutionary process. As shown in part B, during the gust the trajectory becomes disturbed and interestingly, the flight path then repeats a slightly changed trajectory four times with some precision. As soon as the wind is reverted to the normal regime, the flight path of the kite reverts to the original trajectory as shown in part C. Whilst the controller is robust in that it does not crash the kite, there does not seem to be any adaptive mechanism that acts to restore the evolved path of the kite to a looping path that is orientated more directly downwind and hence better performing according to the chosen fitness function.

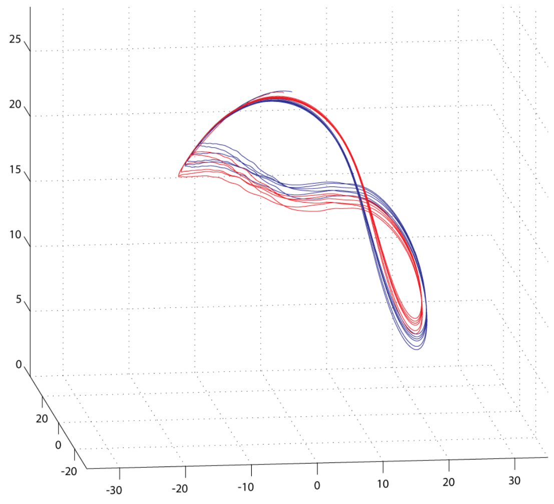


Figure 4.5 The flight trajectory over a 42 second trial of the best individual, from the best evolutionary run after 200 generations

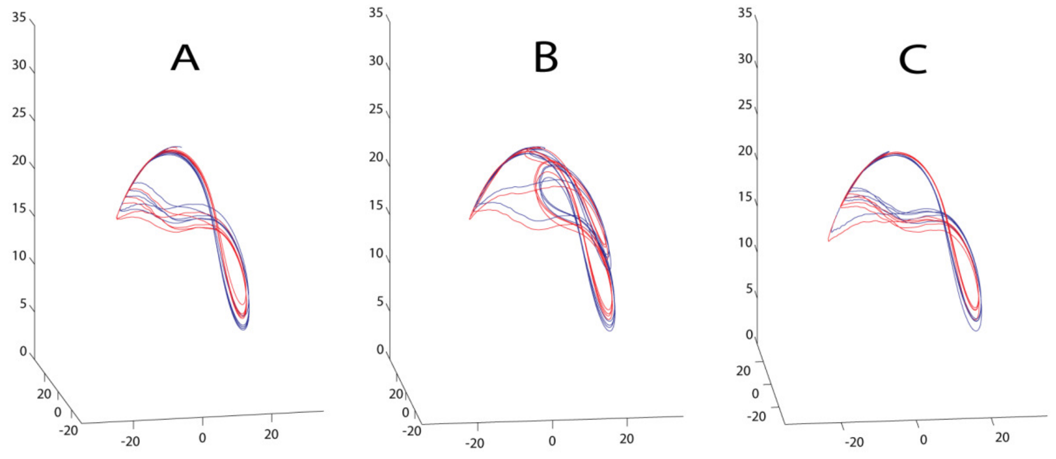


Figure 4.6 The flight trajectories before (A), during (B), and after (C) a large single gust in wind.

4.2.3 Performance after 1000 generations, 25m line length

The best performing neurocontroller whose performance was shown in the previous section was evolved for a further 800 generations. This section demonstrates the change in performance over the entire combined evolutionary period of 1000 generations and the trajectory in which the kite is steered by the best neurocontroller from the population at the end of this process.

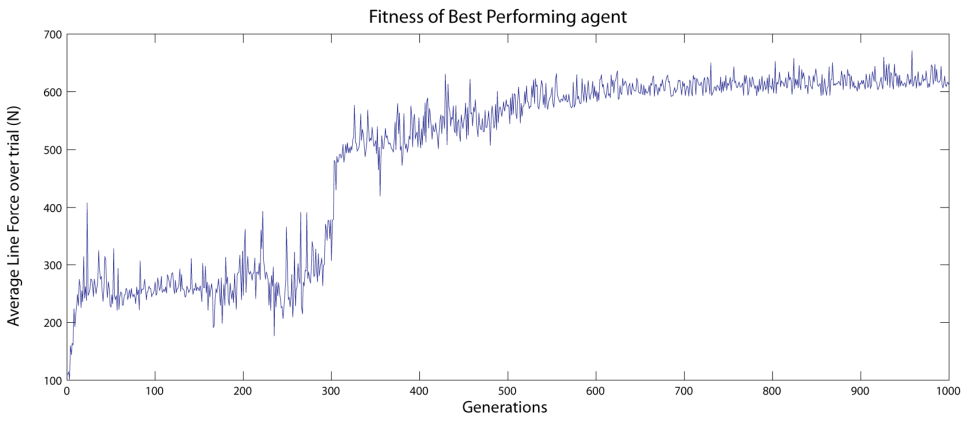


Figure 4.7 A plot of the fitness of the best performing neurocontroller from each generation over 1000 generations of evolution. Note the rapid increase in fitness from 0 generations and just after 300 generations leading to two fitness plateaus.

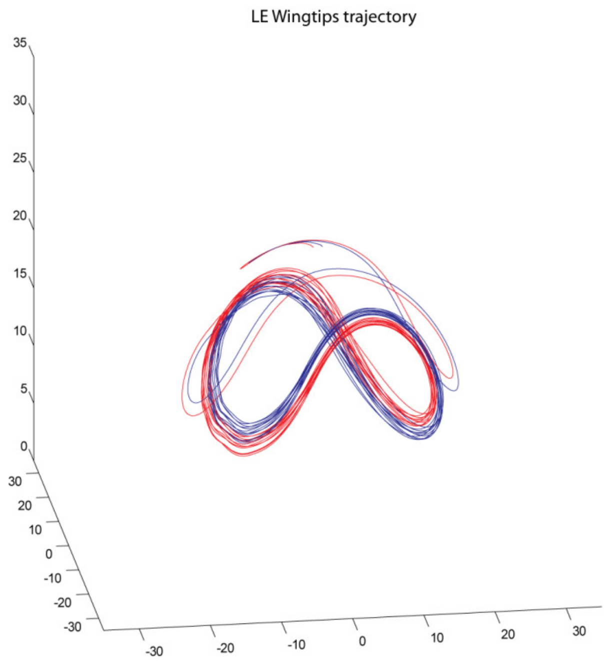


Figure 4.8 A plot of the flight trajectory of the best performing neurocontroller derived from 4.2.2 after 800 further generations of evolution, during a trial with constant wind velocity. The evolved trajectory is an asymmetric lying figure eight.

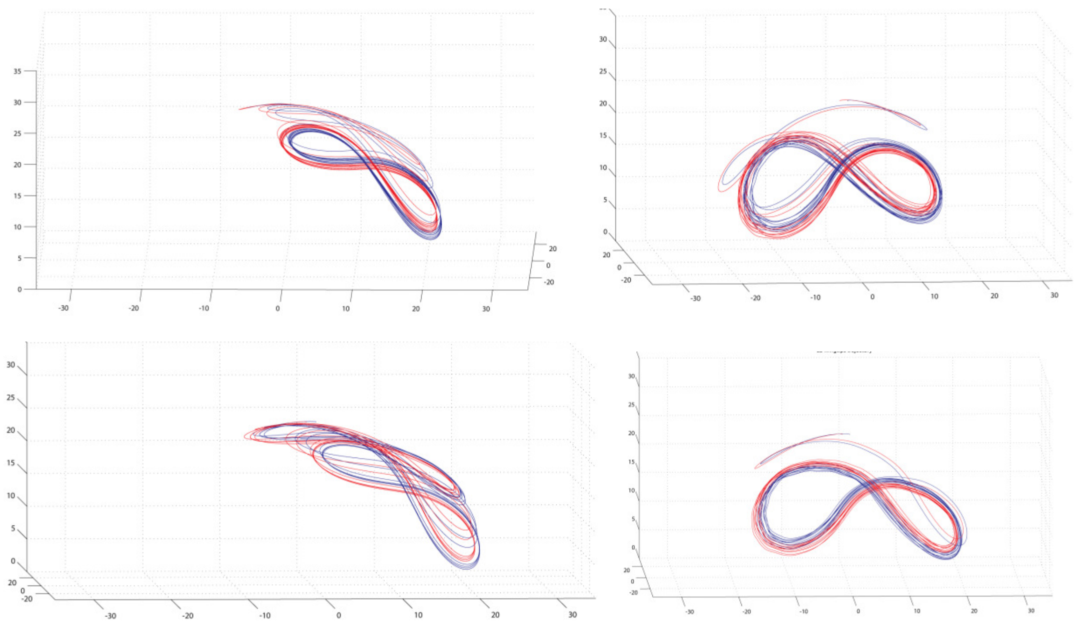


Figure 4.9 The final evolved trajectory of four other 1000 generation evolutionary runs, started from random valued genomes

Figure 4.7 shows how the fitness of the best performing individual per generation changes as the evolutionary process progresses, having allowed the best performing controller from the initial set of experiments to progress for another 800 generations. In the first 25 generations, the fitness rapidly increases, presumably as the population converges on the most appropriate strategy from the random initial population. This is followed by 150 or so generations of little performance variation. After 100 generations where the fitness seems to be subject to far greater variation, a new strategy emerges at generation 300 which the population rapidly converges on and then refines to reach a new plateau of fitness. The resulting trajectory is a 'lying eight' trajectory qualitatively similar to those obtained in other studies (Houska & Diehl, 2007).

Figure 4.9 shows the flight trajectory of the best performing neurocontroller from four separate populations after 1000 generations of evolution. All of these are asymmetric to some extent although the top right plot is relatively symmetric and aligned almost directly downwind. These were the best 4 of 20 concurrently executed evolutionary runs, and were initialised with random genomes and not prepopulated with the results of the initial 200 generations as per the experiments whose results are shown in Figure 4.8.

Figure 4.10 shows a plot of the value of the steering neuron output over the trial shown in Figure 4.8 in which the oscillating steering signal driving the looping flight can be clearly observed.

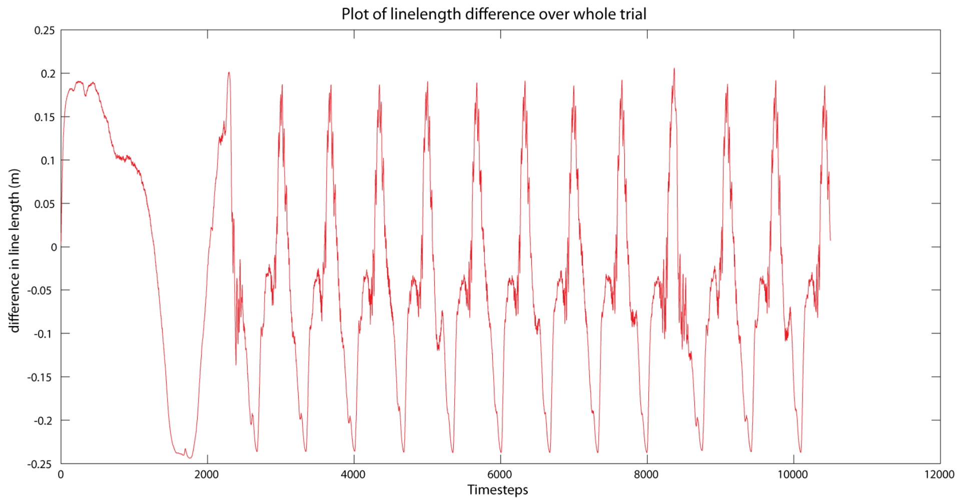


Figure 4.10 A plot of the control sequence in the same trial as shown in the previous figure, Note the kink in the line near the middle of the steering range.

One notable aspect of the neurocontroller performance in this task is that although the evolutionary process finds an appropriate trajectory, the controller is still prone to failure, even within the range of wind amplitude variation that the controllers are exposed to during evolution. Testing the controller used to generate Figure 4.8 and Figure 4.10 with 100 different wind sequences resulted in a mean performance score of 442.9 with a s.d. of 187.8. Whilst the average score is acceptable, albeit considerably less than the 582.3 achieved with constant wind speed, of greater concern is the high variability, which included catastrophic failure with 32% of trials resulting in crashes.

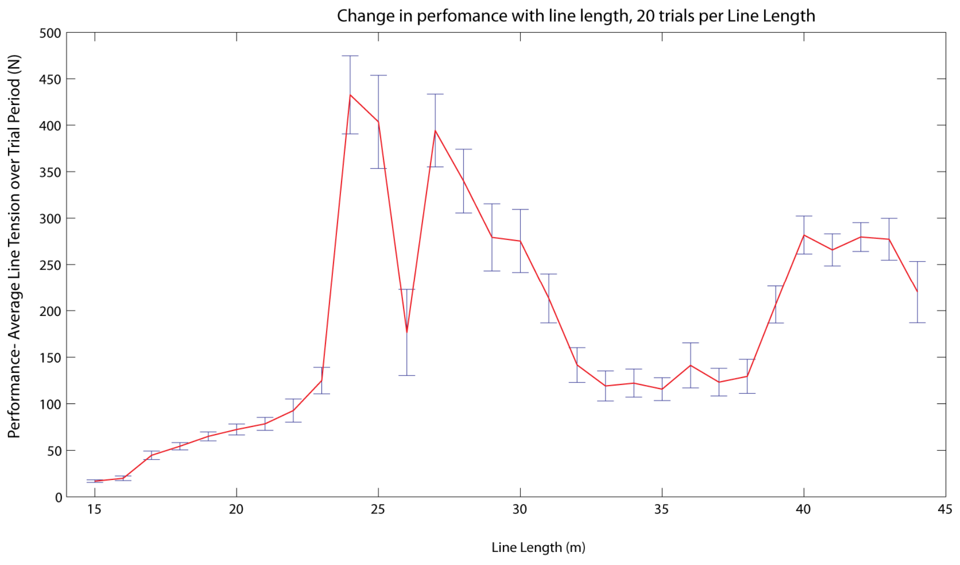


Figure 4.11 A plot of the mean performance (red line) of the same neurocontroller over 30 different line lengths, 20 trials per line length. Bars of standard error demonstrate variation in performance.

Figure 4.11 shows the results of testing the same neurocontroller repeatedly, with different wind conditions and at different line lengths. We can see that the generalisation to different line lengths is quite poor with a pronounced peak at the 25m length used during evolution.

4.2.4 Extended evolution with lateral wind deviations

The neurocontroller developed in the previous section was then evolved for a further 600 generations with the probabilistic heuristic for lateral changes in wind directions applied as well as the deviations in strength. Figure 4.12 demonstrates an evolved neurocontroller successfully coping with a transient change in the wind direction as shown in the lower half of the figure. It is notable how much less consistent the alignment of the repeated flight trajectory is in this case, it appears to wander rather than follow the wind. This wandering appears to occur even when the wind direction is unperturbed, at the beginning and the end of the trial for example where the flight trajectory is also not consistent. As might be expected, the reliability of the controllers in dealing with these lateral gusts or changes in wind direction was poorer than their ability to deal with changes in the strength of the wind alone, with over 50% of trials resulting in crashes.

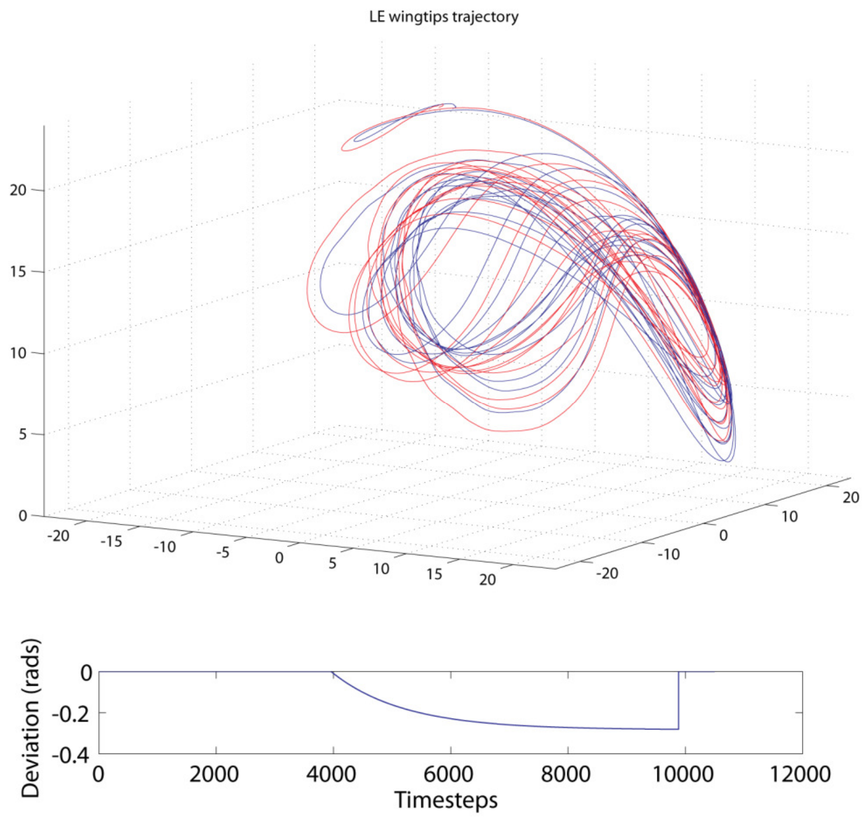


Figure 4.12 A plot of the flight trajectory generated by the best evolved neurocontroller, taken from the previous experiment and evolved for a further 400 generations with the lateral deviations in windspeed. The lateral deviation in wind direction used for this trial is shown in the lower half of the figure.

4.2.5 Extended evolution, 20 to 400m line lengths

We demonstrated in the previous section in which the flight control ability of even the best neurocontrollers exhibited poor generalisation to changes in the lengths of the lines. Therefore, we performed further evolution of 800 generations using the same seed as the lateral wind deviation experiments, i.e. the population from the 1000 generations of evolution. In these experiments, in each evaluation pair, the kite was flown with line lengths randomly initialised between 20 and 1000 meters. The sequence of graphs from Figure 4.13 to Figure 4.17 shows the trajectories resulting from the best performing neurocontroller from one evolutionary run flying the kite at 25, 50, 100, 200 and 400 meters respectively. What is clearly apparent is that the controller does not fly the kite in the same path when the length of the lines changes. The path in which the kite is flown at 25m is clearly suboptimal, and very different to that with which the population was initialised. The path taken at 50m is completely different with a tight circular trajectory being taken, immediately downwind of the tether point, interestingly this trajectory is another of the local optima described in (Houska & Diehl, 2007). At 100m and higher, the neurocontrollers generate yet another different flight path, comprising of a long downward swoop oriented immediately downwind of the tether with a loop either side of this downstroke, joined by a crosswind sweep about half of the length of the downstroke. At above 350 metres, the trajectory again changes, to a repeated looping in one direction, spiralling from the zenith towards lower elevations.

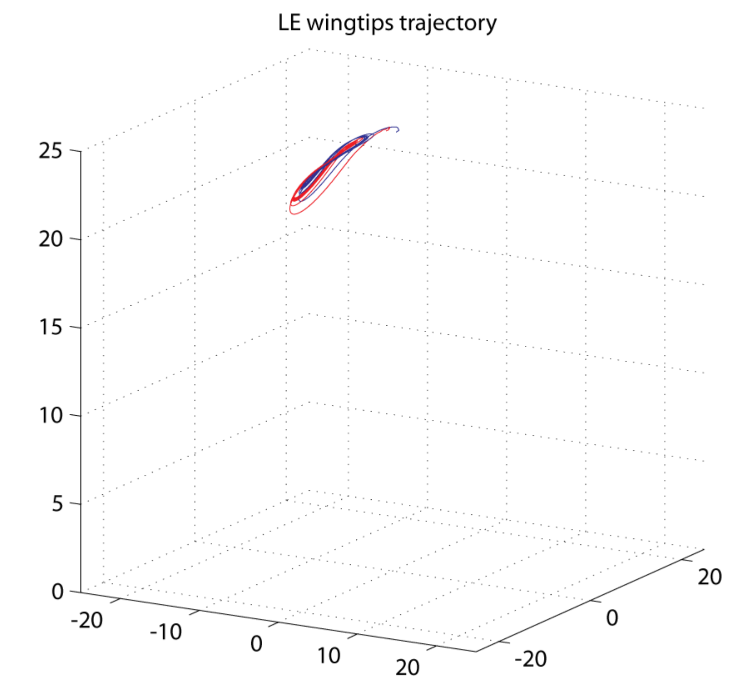


Figure 4.13 A plot of the flight trajectory generated over 42 seconds by performing neurocontroller evolved at multiple line lengths, with 25m lines in this case.

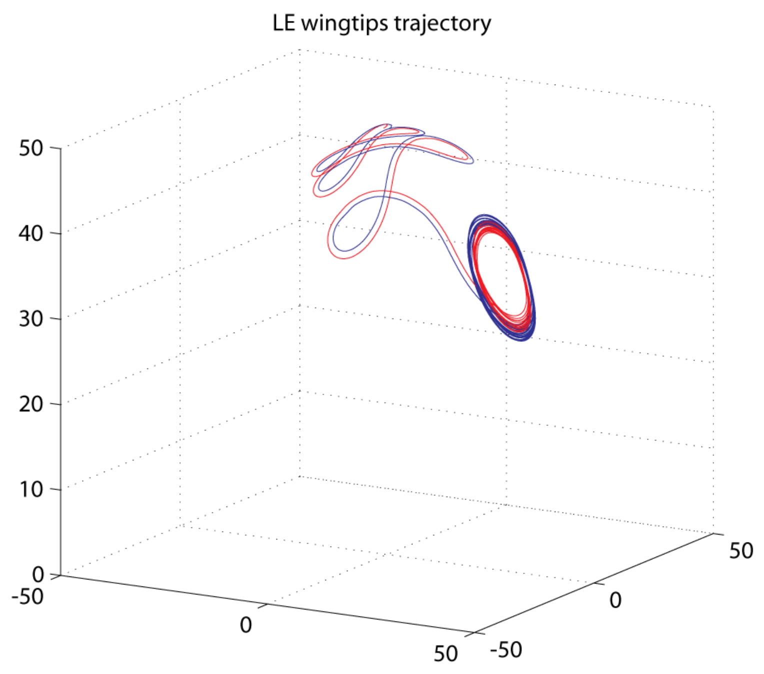


Figure 4.14 The same neurocontroller flying the same kite on 50m lines.

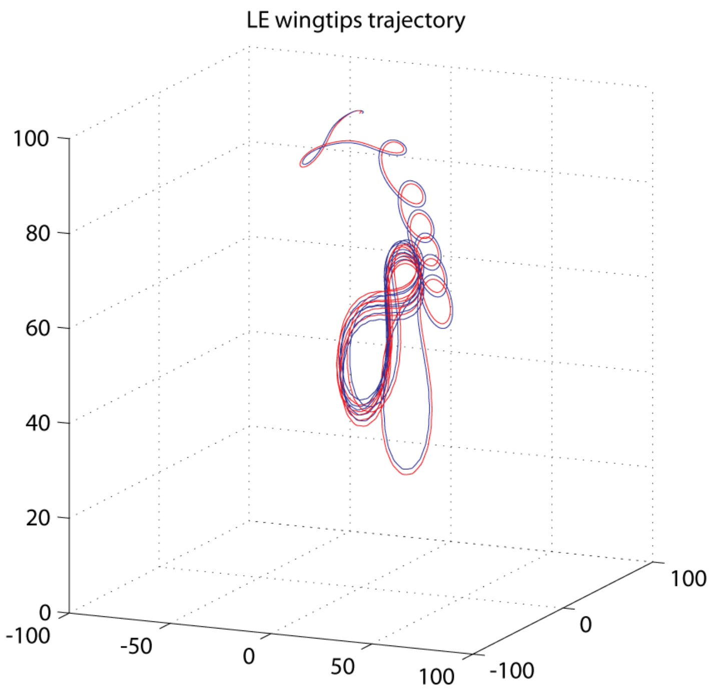


Figure 4.15 The same neurocontroller flying the same kite on 100m lines.

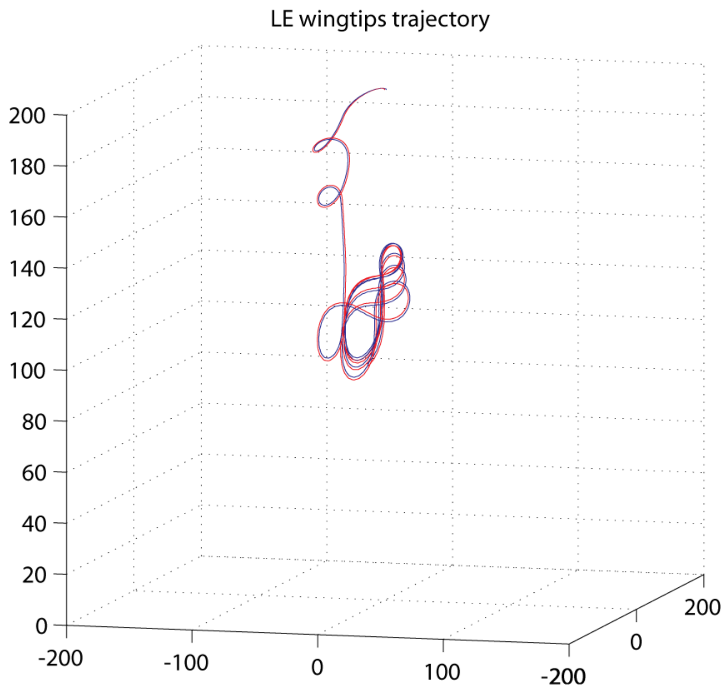


Figure 4.16 The same neurocontroller flying the same kite on 200m lines.

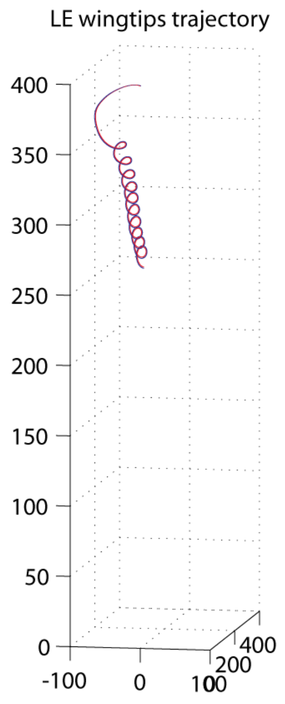


Figure 4.17 The same neurocontroller flying the same kite on 400m lines.

4.2.6 Evolving CTRNN controllers for the same task

At this stage, some attempts were made to evolve CTRNN's of the same network size for the same task. Due to constraints on computational resources, it was not possible to run many evolutionary runs. Whilst the evolution of discrete time networks converged in 200 generations to a flight control trajectory that might be considered reasonable in four out of six runs, it proved much harder to evolve the CTRNNs to the same level of performance. It was apparent that contrary to the discrete time networks, oscillatory steering output behaviour was not being evolved. The evolutionary runs either failed, or occasionally settled on a point in the steering that would exploit an artefact in the physics simulation that would allow the kite to make small oscillations around a single point at the side of the wind window before stabilising. The fact that relatively low scores were achieved in this manner suggests that it was the initialisation of the networks that was making the emergence of oscillating output the result of a rare combination of parameter values.

4.3 Discussion

The key result derived from this initial investigation is that an evolutionary robotics approach is able to generate 7 interneuron recurrent neural networks that control a simulated kite in an appropriate flight path and maintain that path in the face of a limited but non-trivial amount of environmental variation. Some robustness to perturbation was apparent after only 200 generations of evolution with a population size of 20. With a trial time of 42 seconds, this corresponds to less than 47 hours of real world flight time. However, even after a further 800 generations, corresponding to over two weeks of simulated flight, limitations were clearly apparent in the robustness to perturbation. Why this fragility existed is not clear, however, it is likely that the method of evaluation made some contribution. It is normally considered good practice in ER to include multiple trials in a single evaluation score. However in this case only a single trial was used because of the extreme computational demand of the process. This being the case, the only way in which two competing individuals are differentiated is their score on a single 42 second wind sequence. By testing the individuals on 2 or preferably more separate sequences, individuals that produced good scores in a range of conditions would be selected over individuals that, through good fortune or an aggressive strategy, scored well in a given trial but poorly in others. The arbitrary nature of the wind speed variation may also have been a factor, with the simulated kites being occasionally exposed to unrealistic severe and instantaneous changes of windspeed that would affect even a competent controller.

When the length of the flying lines was varied during evolution, the neurocontrollers were capable of flying the kites successfully without generalisation being an issue. However networks could not be generated that would maintain a relatively consistent trajectory regardless of the length of the flying lines. The management of the evolutionary process was likely to have contributed to this diversity of flight trajectories when neurocontrollers were evolved at multiple line lengths. Because the dynamics and response to flight controls changes most when the lines are relatively short, it might have been a good choice to bias the line lengths used towards the shorter line lengths in the range. In keeping with the point made about generalisation to different wind conditions, using multiple evaluations with enforced selection of short, medium and long lines might have improved the performance at 25m line length. Additionally the choice of inputs may have made some contribution, using the difference in wingtip angles as a measure of the kite's direction would mean that this input parameter would effectively have had different ranges depending on the length of the lines. It is also possible however that the limited dynamical resources of the discrete-time neural network mean that task dynamics changes drive the network response in a maladaptive manner and that the multiple trajectories seen were merely an evolutionary compromise. In short, it is potentially impossible for control strategies for the different dynamics at multiple line lengths to coexist in the fixed weight discrete time neural network.

It is noteworthy that in contrast to other work (Houska, 2007), the evolved trajectories tend to use the whole wind window and not a small arc of less 0.3 radians directly downwind of the controller. The much shorter line length in this model (25m here vs. 100m in (Houska, 2007)) is likely to be the main factor contributing to this difference, and loops with smaller angular arcs were seen at longer line lengths in the later experiments. Naturally, the turning radius of the simulated kite will also be a contributory factor. Additionally, the trajectories evolved often were, as per Figure 4.5, not centred directly downwind of the tether point. This may have been due to the restriction of actuation speed imposed by the motor output mapping process but is more likely to be an artefact of evolution or the test regime. Early successful controllers from an initial population are more likely to stay in flight if they swoop down to one side, where the kite naturally slows, than swoop down to the centre where the kite continually accelerates and is very sensitive to input. In many of the cases presented, evolution seemed to act to align the looping trajectory in a more downwind position. This effect will be addressed later in chapters 5 and 7, potentially by initialising the kite in a different position at the start of each trial.

The discrete-time neural networks found the best solutions in this limited evolution period. The reason for their outperformance of the CTRNN networks is unclear, but is potentially due to the additional time required for the selection of satisfactory time constants, or the fact that the CTRNNs were not initialised under centre crossing conditions (Mathayomchan & Beer, 2002) that might have improved signal propagation (H. Williams & Noble, 2005) and promoted the kind of oscillatory dynamics seen in Figure 4.10 by initialising the population of neurons with their biases set so as to avoid saturation of neuron activation. It could simply be that the yield of the evolutionary process is lower for CTRNNS, which have an additional set of parameters to evolve. Because of the computational demands of the evolutionary process, only 12 complete evolutionary runs were attempted with CTRNNs. However, the prevalence of non-oscillatory dynamics in the results of CTRNN evolution suggests that it was inappropriate initialisation of the networks in this case that was the issue. This concern is addressed in later chapters.

4.4 Summary

The work in this chapter demonstrated that the application of ER techniques to kite control produces controllers that fly the kite in stable figure eight trajectories, shown previously to be an optimal path for recovery of energy from the wind. Evolved neurocontrollers robustly maintain these trajectories during significant deviations of wind speed. Some adaptation to lateral deviations in the wind direction were possible, however the greater fragility of these systems to this type of perturbation was noteworthy.

There were some clear lessons learned in this initial exploration further to the promise shown. The foremost lesson was that the evolutionary process needs to be very carefully considered, if a given trait is desirable then it must be ensured that the structure of the evaluation process is set up such that it is selected for. Additionally, careful consideration needs to be given to the information that is available to the neurocontroller such that generalisation is not rendered unnecessarily difficult. The initialised values and parameter ranges of the network must also be set such that the type of dynamics necessary to solve the task result from common parameter combinations.

In general these results suggest that ER can be an appropriate approach for kite control systems, however considerable development will be needed to achieve the necessary level of both robustness and performance, especially if CTRNNs are to be used.

Seamen have often felt chagrined when they have viewed the light clouds scudding swiftly aloft, while not a single breath would stoop even to their sky-sails.

George Pocock

5 Adaptive behavioural modulation and switching in an analogue of a kite control task

In the previous chapter, we demonstrated how ER could produce neurocontrollers which could steer a kite in an appropriate path in order to maximise the tension in the lines. We made it clear that this only corresponded to the generation phase of the generation/retraction cycle, flight control in the retraction phase will be considered later in chapter 7. We were interested in the possibility of using an evolved controller to determine when to switch between these operating phases in a scenario where constraints exist on switching, and where adaptation is necessary in order to cope with highly variable environmental conditions.

To this end we define a simplified analogue of a kite control task that requires, in its simplest form, a situated artificial agent to switch between two mutually exclusive behaviours. In more complex versions of the task, the agent is required to adapt to changes within its environment that occur on different temporal scales. We describe the failure to evolve successful agents when a decision threshold is defined artificially and conversely the evolution of successful agents when they themselves are allowed to determine their own threshold through interaction with the environment. Agents are demonstrated to be capable of adapting both their switching behaviour and spatial domain according to environmental changes on three temporal scales, on the fastest of which, the agents appear to behave in an opportunistic manner.

5.1 Introduction

In airborne wind energy systems, the kites require active control that ideally modulates the flight behaviour according to environmental conditions, often to maintain the operation of the system within certain hard limits of viability. Additionally, the lift mode class of currently posited system configurations requires alternation between two discrete modes of operation. The point of switching between these two, mutually exclusive actions or ‘behaviours’ could be determined by a simple but rigid threshold or heuristic that is predetermined. However, it is likely to be beneficial to have an adaptive strategy that modulates behavioural hysteresis through some mechanism that takes into account the current state of the agent and its environment.

Here we present a simplified analogue or ‘toy version’ of the kite control task where an environmental resource with spatially heterogeneous distribution is exploited by an evolved agent. The task is designed so as to mirror key aspects of the kite control problem at a ‘higher’ level i.e. that of behavioural modulation and/or hysteresis without the complication and heavy computational burden of simulating the dynamics of kite flight, treated in the previous chapter.

In fact, the slow speed at which the kite physics can currently be simulated rules out exploring longer term adaptive control of switching modes over time periods longer than a minute. The task is designed to be simple enough to allow evolution to proceed rapidly whilst still retaining a dynamical embodied interaction between the agent and its environment. Assessing the adaptive abilities of the evolved agents may indicate the potential of evolutionary robotics for generating long term kite control behaviours. Additionally, determining the best way in which to evolve adaptive behaviour in this simple analogue task may inform future work in generating similar behaviours in real kite control.

5.2 Operational considerations in a two phase task

Most current concepts for kite energy systems operate with two functional phases as per Figure 5.1. In the 'generation' phase (A), the kite is steered actively through the airspace in order to increase the apparent wind speed across the kite. The kite therefore pulls its lines out from a reel at ground level which is coupled to a dynamo, thus generating electricity. We have demonstrated previously in simulation [2,3] that an evolutionary robotics approach can be used to produce neural network controllers that both find an optimal flight trajectory for phase A and maintain it in the face of significant environmental perturbations.. In the 'retraction' phase (B), the lines are retracted with the kite controlled so as to reduce resistance, the process is therefore cyclical and sustainable indefinitely whilst the wind allows. As long as A produces more energy than consumed in B then there is a net gain of energy. The kite control problem has some further key characteristics, firstly, the environment is highly dynamic, both the wind speed and direction can change rapidly. Secondly there are some hard constraints within which the agent must operate in order to keep the kite aloft and not overpower the hardware, as discussed below. Given these characteristics a high quality controller will have to perform a) in-behaviour adaptive modification in order to maximise productivity whilst b) meeting the viability limits imposed by the operational constraints.

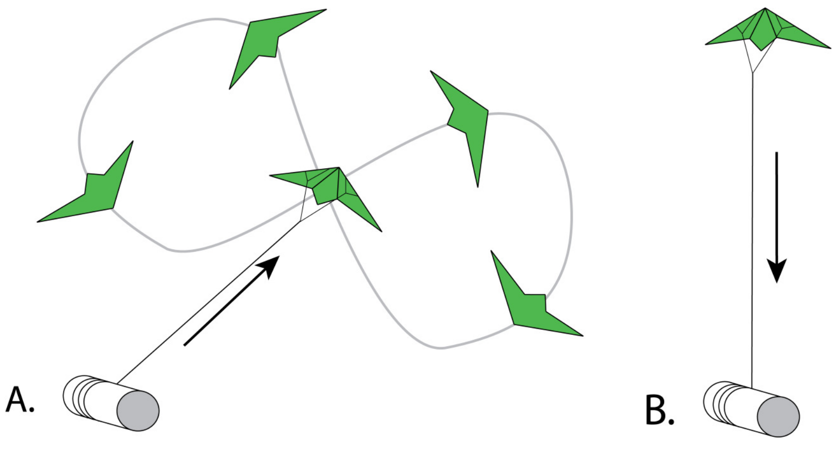


Figure 5.1 Generation (reel out) phase A and retraction (reel in) phase B.

Our exercise here of building a task analogous to the kite control problem is intended primarily to be informative as to the validity of an evolutionary robotics (ER) approach to tackling its behavioural aspects and to gain insight into how it might best be implemented. With this analogous task, without necessarily expecting direct cross-applicability, we aim to address two components of the kite control problem that may benefit from an adaptive controller. The first issue is one of switching between the two separate behaviours, that of generation and retraction. Given a finite length of line on the reel, there is a need to retract the kite after a generation 'stroke' pulls some or all of the line off the reel. The goal will always be to maximise the time spent generating and the difference in energy gained via the outstroke vs the energy spent on the instroke. There is a hard constraint which is that the line tension must never exceed its breaking load, one way of reducing the line tension is to allow the kite to be reeled out, so this is an argument for always leaving a reserve of line on the reel,
There is some time or performance cost to the switching from generation to retraction, given that the kite should be ideally be steered to the zenith or the side of the wind window in order to minimise both the force and the time required to retract the lines. Differences between switching strategies are exemplified by two examples existing in the literature, one of which is a heuristic simply involving switching between generation and retraction at predetermined line lengths of some sensible intermediate values (Canale, Fagiano, & Ippolito, 2006). At the other end of the spectrum is the possibility of much more frequent alternation, whereby retraction occurs within a given portion of every figure eight trajectory as posited by Houska (Houska, 2007). The latter strategy avoids some of the problems of the former, namely a reduction in yield at lower altitudes due to wind shear, and that of potentially reaching a hard line length limit and resulting wasted generation time or hardware damage. However, by not placing the kite at zenith or windows edge, greater energy needs to be expended to retract the kite and the momentum of the kite is reduced.
We intend to approach this trade-off with our toy problem using standard ER techniques, and the as yet unaddressed question of whether to modify the switching points according to environmental conditions. Being able to opportunistically switch between modes depending on the current characteristics of the environment might be a useful attribute. If for example, gusts in the wind strength of a considerable duration and amplitude were occurring, then it would make sense to generate power during those gusts and not retract given that the retraction cost and the generation reward would both be increased. The opposite would also clearly be true for protracted lulls in wind strength. Neither of the aforementioned studies consider the possibility that a strategy will need modification when the wind is at different strengths or subject to relatively rapid change.
Further to the frequency of phase switching, the second issue is specifically related to the effective spatial distribution of the wind resource in relation to the kite tether point. As mentioned previously, in normal wind conditions, the kite is steered in a looping pattern

directly downwind of the tether point (the ++ and + area in Figure 5.2, left), in order to maximise the aerodynamic forces away from the reel and therefore the energy that is generated. In this area the kite's angle of attack is limited to high inclinations due to the bridle and wing configuration, and the flight is most perpendicular to the wind direction. However, in strong wind conditions, flying in this region will generate forces that may damage the generating hardware, the kite, or its lines.

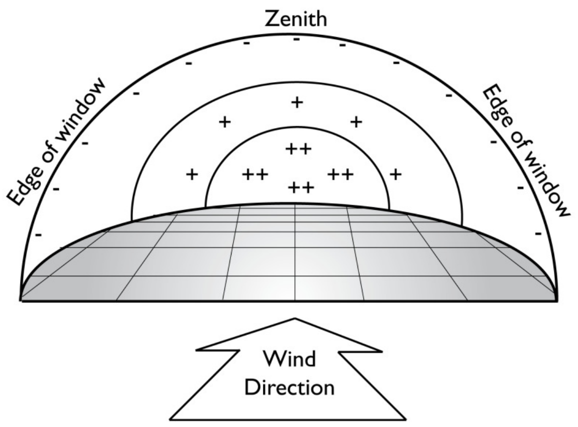


Figure 5.2 The wind window concept. The most 'powerful' portion of the window is signified by ++, less powerful by + and the least powerful by -.

This overpowered situation can be prevented by simply avoiding the most powerful region of flight space. As the wind speed increases, the extent of the avoided region should increase. Conversely with a weak wind resource the agent must confine itself to the most powerful region of flight space where high apparent wind speeds can be sustained. These changes constitute short term, reactive modifications to the kite's spatial domain of activity and are the second level of adaptation that we shall address in the model. As we detail below, we intend to explore how changes in the intensity of the energy source on different temporal scales affect the agent and its behaviour.

5.3 Methodology

5.3.1 Agent and task design

We have designed a task that captures some of the key properties of the kite generation strategy problem without the computational burden of modelling the aerodynamics of the kite. Our task is based around a simple 2 wheeled, 2 sensor agent (see Figure 5.3) situated within an unbounded 2d area. The analogy for the flying lines is a battery, charged by an orientation independent 'solar panel' on the agent. As with the tether length on a real kite generator system, there are hard upper and lower limits to the battery level. As success is judged by quantity of energy passed to the battery and the battery charging time is a small fraction of the trial time, the agent can score more highly by repetitively switching between charging and discharging the battery, corresponding to the repetition of the generation/retraction cycle of a pumping kite generator system.

The agent's sensors are mounted 90 degrees apart on its perimeter; each has a 180 degree range and provides a signal linearly proportional to the local light intensity. Within the simulated environment there is a light point source which provides an energy resource which the agent must exploit, the equivalent of the wind in the kite control problem. Because the intensity of the light drops away exponentially with distance from the source, there is a light intensity gradient which mirrors the wind window of the kite problem in that the energy resource has a heterogeneous spatial distribution. This radial distribution regionalises the arena into 3 key areas as per Figure 5.4.

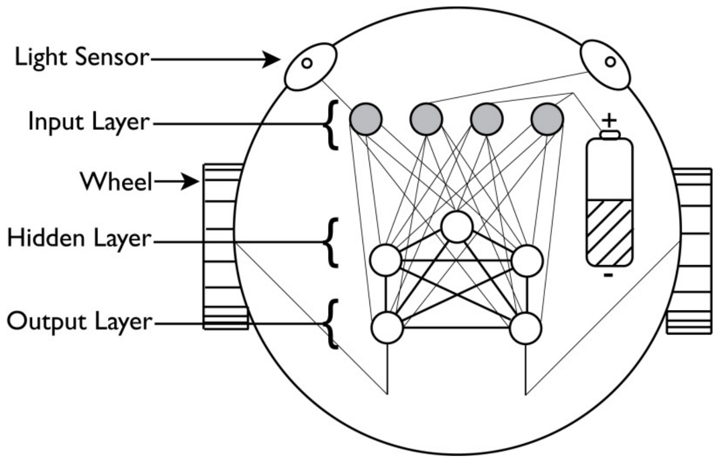


Figure 5.3 A diagram of the agent and its neurocontroller architecture, light grey neural connections are feed forward, bold black neural connections are reciprocal with asymmetric weights. The battery connects to two input neurons, one connection carrying a battery level signal and the other corresponding to rate of change of battery level

As would be expected in a kite hardware implementation, and in common with conventional wind turbines, we implement both a plateau of the generation/charge rate at some moderate to high light level (line B in Figure 5.4), and a 'cut out' at a higher level still, where the environmental power source, if it were coupled to the generator, would overload and damage it. To select against this overpowered scenario, when a threshold point is reached corresponding to entering the area inside line C in Figure 5.4, the battery continues to increase but the agent does not accumulate fitness until it leaves the region. The fitness function used is therefore described in Equation 5-1 and 5-2 where the fitness *F* of agent *A* is the sum of the power *P* stored in the battery in each timestep *i*, P_{Peak} is the maximum charge rate, P_{Cut} is the light intensity above which zero fitness is accumulated and *S* is the total number of timesteps,

Eq. 5-1

$$F_A = \sum_{i=1}^{S} \min\,(f(P_i), P_{Peak})$$

Eq. 5-2

$$F_{(x)} = \begin{cases} 0, & x > P_{Cut} \\ x, & x < P_{Cut} \end{cases}$$

This corresponds to the lines being allowed to reel out without being coupled to the generator, a situation that can only persist indefinitely whilst there is still line available on the reel. Therefore, if the threshold is exceeded when the battery is full, the agent is considered to be broken and the trial is ended prematurely with implications for the agent's fitness as detailed below. Similarly, if the agent's battery reaches zero, the agent has failed to collect sufficient energy from the environment and the trial is also ended.

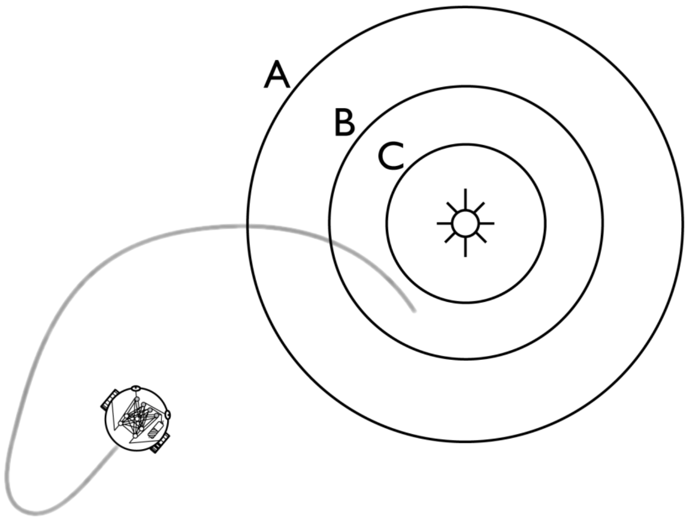


Figure 5.4 Diagram of a portion of the experimental area, light source marked by the sun icon. Between line C and the light source is the overpowered region, between B and C, the region where the charging rate plateaus. A marks the line at which the switching threshold is met

Naturally some consideration needs to be given to how the agent's interaction with the environment leads to either charging or discharging behaviours and at what point the behaviour can be said to have been 'switched'. Therefore we consider two potential options, we first simplify the problem of making some trade off between time cost of switching and energy required to switch by defining a lower force threshold at which switching automatically occurs. Once the agent crosses line A, at which point the battery charge rate input to the neural network will become negative, it will have effectively switched behaviour although the 'decision' to do so may have been made at some earlier point. As the boundary is determined by an environmental threshold completely independent of the agent's activity, we term this regime 'threshold switching'. In the second scenario, we assess a variation in which the rules are unchanged, except that the agent has some control over its own switching point. This is implemented implicitly by introducing an energy cost to movement that will drain the battery. If the agent stays still, it will charge until full at any point in the environment at a rate determined by distance to the light, it is movement at a speed which consumes more energy than is recouped from the environment that will drive down the battery. The agent in this scenario is forced to make a more complex, but more realistic trade off in that moving far from the high-quality resource will speed up discharging, but mean that for portions of the return journey, time is spent near energy neutral and at suboptimal charging speeds. We term this regime 'enactive switching'. Crucially, in both these scenarios, the agent has to commit in advance to moving in a given direction in order to switch between the two modes at a point in the future.

5.3.2 Neural network and agent dynamics

The neural network that drives the agent's behaviour is a small continuous time recurrent neural network (CTRNN) of 4 input neurons, 3 hidden layer neurons and 2 output layer neurons, with both inhibitory and excitatory connections permissible. Each of the four input layer neurons receives input constituting one of the following four data values; left light sensor value, right light sensor value, proportion of battery capacity remaining and the final value, rate of change of battery charge rate which is the first time that we introduced a velocity signal as a separate input. Each input neuron connects to every hidden and output layer neuron, but receives no other input (see Figure 5.3). The hidden and output layer neurons are fully interconnected, connection weights are asymmetric. A single neuron derives its dynamics from the CTRNN equation given in the previous chapter. However in this case neuron activation values are integrated using Euler integration with a larger timestep of 0.1. The sigmoided values of the two output neurons form the motor activation values for the left and right wheels respectively. There is no rescaling of these values outside of the range [0,1], therefore the agent can turn but not reverse. Average wheel speed, which is an instantaneous reflection of the output neurons' values, determines the agent's velocity. Difference in wheel speed determines any change in the agent's orientation angle, dependent on its diameter. The agent's position is also determined by Euler integration with a step size of 0.1. As the motivation of this work is to answer the questions whether ER can as tool generate the kind of switching behaviour required for kite control devices and how best to achieve and appropriate strategy, we do not intend to analyse the network dynamics that result from the evolved weights.

5.3.3 Genetic algorithm and trial configuration

We use the same tournament based, steady state microbial genetic algorithm (Harvey, 2011), as the work in the preceding chapter. The population of genotypes or artificial 'DNA', is randomly initialised, specifying possible parameter values for the neurocontroller. Pairs of neurocontroller parameter sets taken from the population of 20 have their performance evaluated. The winner overwrites the loser with small additive random mutations in the range [-0.05, 0.05] added to a random 50% subset of its parameter values. This constituted a departure from the work in the previous chapter in which a small parameter was applied to every value in the genome. Although not based on a comparative analysis, it was felt that later in evolution changing all the parameters simultaneously would be disruptive by preventing the conservation of particularly useful or well adapted portions of the genome. This process continues for several hundred iterations, each consisting of 10 competitive evaluations. For all tasks, fitness is judged by the amount that the battery is charged, over the time course of the trial. As the battery cannot be further charged once a hard upper limit is reached, and given that the trial length is set between 30 and 300 seconds such that the battery could be charged and discharged fully at least twice, the agent is effectively penalised if it does not play the game that we intend it to and switch between charging and discharging behaviours.

Contrary to the previous chapter, to ensure that the agents are robust across a range of the trial parameters that they will encounter, the evaluation of one agent consists of five separate trials. The common elements to these trials are as follows; the agent is reinitialised at a new random orientation, in a random location within a square of side 30 units. Its battery level is initialise at a random value between a minimum of 20 units and its maximum value of 60 units. The evaluated score is the mean of the five trials and naturally the competing pairs share the same random initialisations for fair comparison. Prior to the main experiments, we evolved a number of populations capable of successfully performing phototaxis, regardless of initial orientation or location in the arena. A mixed seed population for later trials was created by selecting 4 random agents from five populations of these phototactic agents.

The main experimental phase is split into three portions, in the first set of experiments, we evolve the agent for 1000 generations to maximise charging when the light intensity value of 5 remains virtually constant between experiments (+/- noise of 0.2). In the second we use a successful population from the first experiment as a seed for evolution (500 generations) during which we vary the intensity of the light from evaluation to evaluation within a range of 1 to 10. In the third, (500 generations) the intensity of the light is varied within each experiment, with either continual, linear slow increases/decreases that run the whole length of the trial, constituting a gradual trend, or immediate discontinuous jumps between the original value and some higher or lower intensity values, constituting 'gusts' and 'lulls'. In this final experiment, we only judge the agent on its fitness in the second half of the experiment, the agent is therefore given an opportunity to have adapted to the changes in the environment after a period of unassessed interaction.

5.4 Results

The first result was that only the populations that were pre-initialised with successful phototactic agents were able to produce a successful alternating charging/discharging behaviour. All the results described below are therefore for populations that had first gone through a successful phototactic stage. Because the initial phototactic population was an amalgam of the product of entirely separate evolutionary runs that are likely to have evolved different methods and dynamics with which to solve the task, it is unlikely that these predisposed one or other of the threshold schemes to success or failure.

The most salient result was the poor performance of the agents evolved under the threshold switching regime. Although some dozens of evolutionary runs were completed, most resulted in the agent approaching the light and either stopping in that position or moving in a small stereotypic trajectory near the light. This was true whether the population was seeded with successful phototactic agents or not. In doing so they successfully completed one behaviour but were unable to switch to another. This incomplete strategy consisted of approaching the light and then successfully stopping in the plateau zone of the resource, but never moving from that position. Due to the failure on this stage of the task where the lack of resource amplitude variation should have rendered the task relatively simple, threshold switching was not carried on to either of the other two task stages.

In contrast, the controllers evolved with 'enactive' switching evolved a variety of switching strategies all of which involved repetitively moving closer and further from the light. Interestingly, even though these agents were not evolved under the rules of threshold selection, they performed equally as well and with the same apparent strategy when operating under those rules. They seemed to be capable of generalizing from the smooth gradient of switching that they were evolved under to a sharp threshold, and are therefore surprisingly independent of the need to control their rate of discharge, once the behaviour is fully evolved. All the subsequent plots are of the 'enactive switching' evolved agents, but operating under threshold selection rules for clarity of interpretation. With the exception of one of several alternative suboptimal solutions shown in Figure 5.5, all plots are of the best performing agent from the best evolutionary run. After evolution, populations tended to be highly converged, with each agent from a given population pursuing near-identical strategies that were highly robust to changes in agent starting location or orientation and light dynamics, at least within the ranges experienced during the evolutionary process.

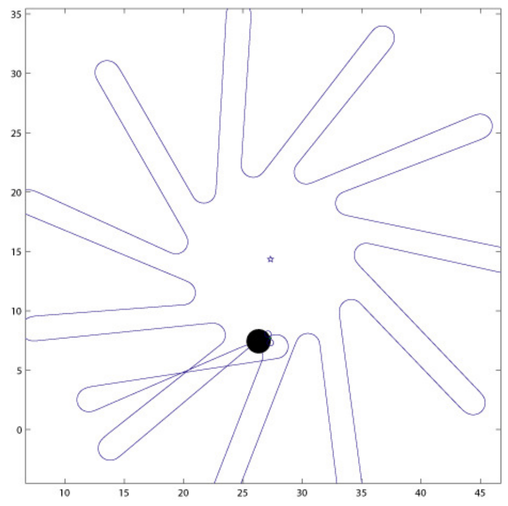


Figure 5.5 An interesting but sub-optimal strategy is shown.

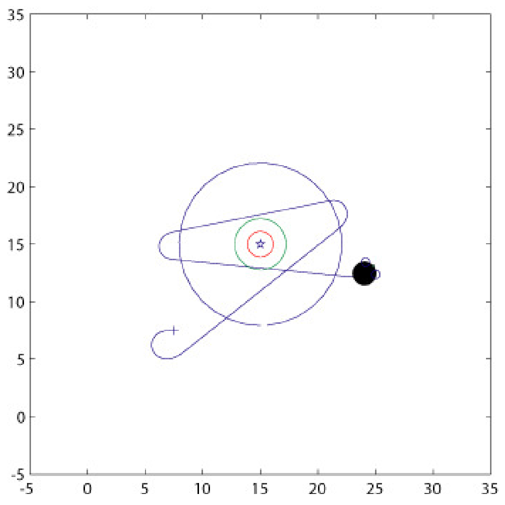


Figure 5.6 First few passes of the best performing agent with light source intensity of 5.

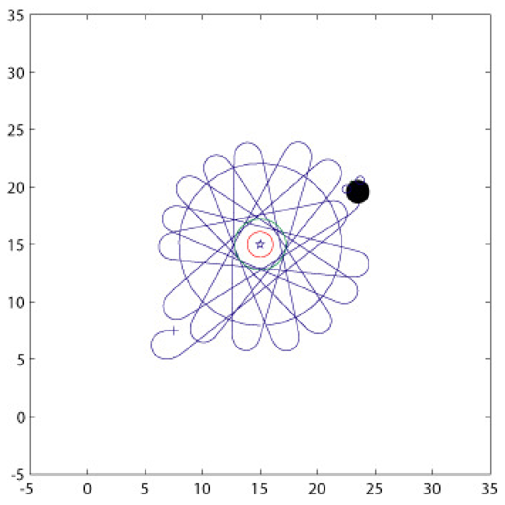


Figure 5.7 The trajectory at the end of the experiment.

As Figure 5.6 and Figure 5.7 demonstrate, the agents skim the edge or just enter the middle ring that marks the region where charging rate plateaus. They then continue the same trajectory until they make an abrupt turn once in the discharging portion of the arena.

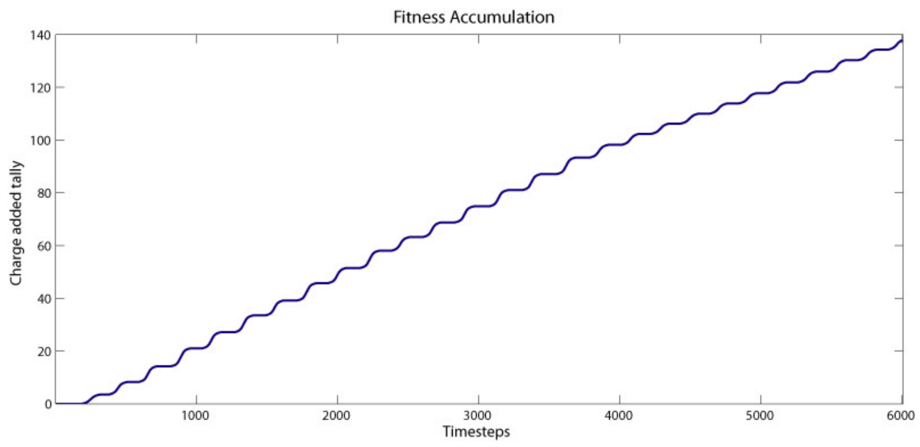


Figure 5.8 The accumulation of fitness over the same experiment shown in Figure 5.7

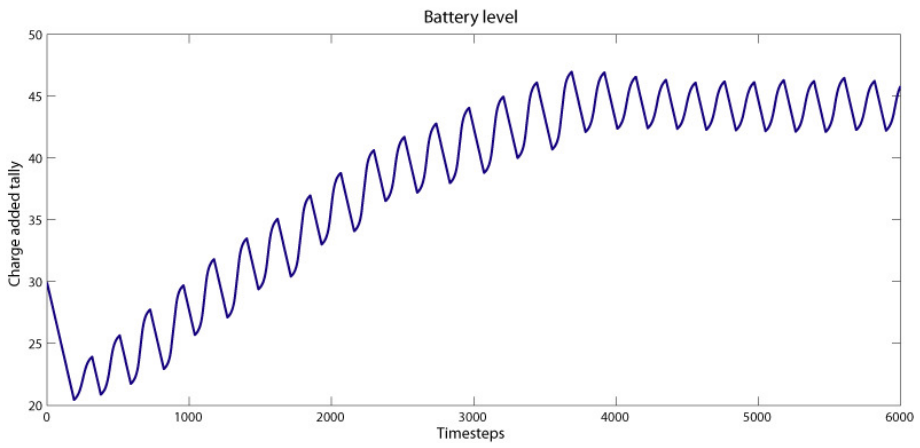


Figure 5.9 The cycling of battery level in the same experiment.

Although it is not immediately apparent from Figure 5.8, it is clear from Figure 5.9 shows how this cycle initially is skewed in favour of charging, resulting in a gradual accumulation of battery which then settles out and oscillates round some intermediate capacity value (c. 75% of capacity). After a further period of evolution with the light intensity varying between experiments by up to a factor of 10, the controllers were able to adapt well to changes in the light intensity. This seemed to be achieved by modulation of the turning angle on exit from the charging area, and as Figure 5.10 shows, the agent enters the charging zone at virtually the same angle. Changing the discharge rate had a similar effect with wide turns keeping the agent in the discharging area longer when the discharge rate was lowered (Figure 5.11) and tight turns occurring when the discharge rate is elevated (Figure 5.12).

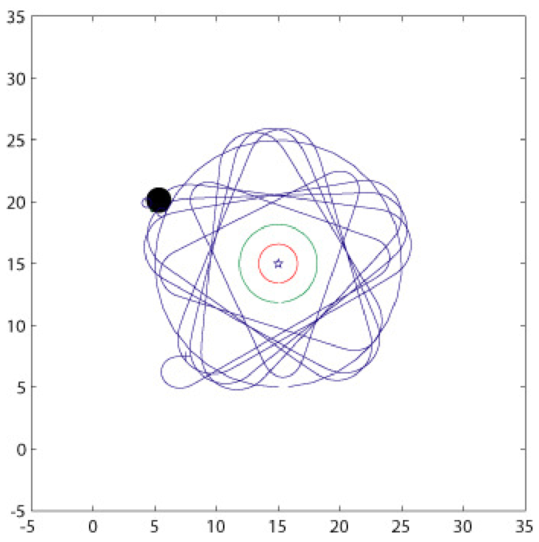


Figure 5.10 Adaptation to an increased light intensity of 10

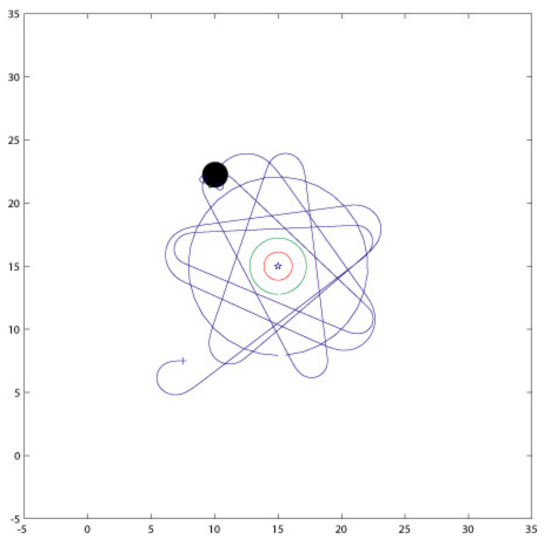


Figure 5.11 Adaptation to lowered discharge rate.

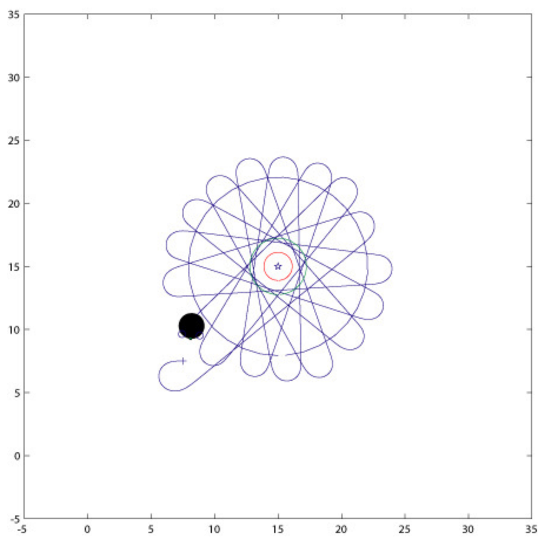


Figure 5.12 Adaptation to an elevated discharge rate.

Finally, we compared the performance of the intensity-adaptive agents from the prior experiment with that of agents subject to further evolution in which the light intensity changed within the experiment on one of two temporal scales. When slow linear changes of intensity occurred, the performance of both sets of agents was essentially equal, all agents continuously modulating their turning angle as per Figure 5.13. Interestingly, the path between turns always appears to remain straight.

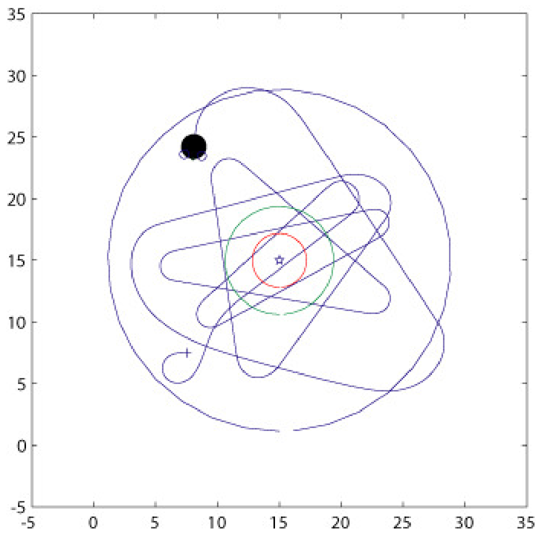


Figure 5.13 Trajectory modulation with continuous linear increase in light intensity (1-20).

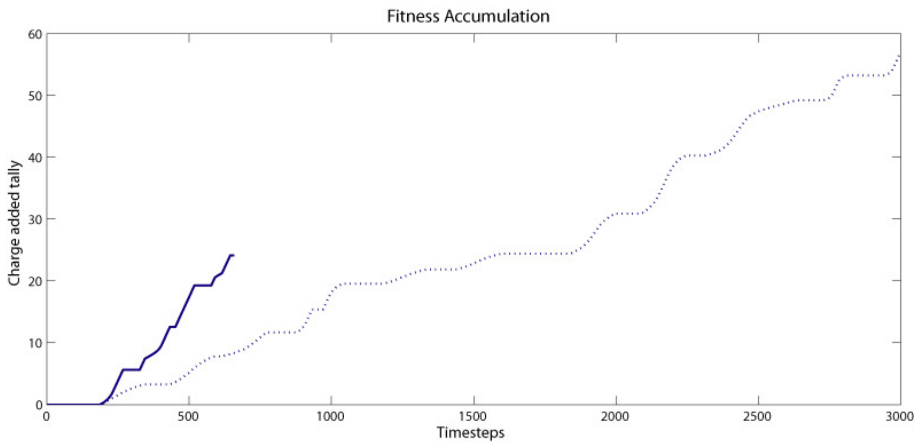


Figure 5.14 The right plot compares the fitness of the agents with (dotted line) and without (solid line) additional evolution with dynamic light strength when exposed to rapid perturbations in light intensity.

However when the intensity changes were rapid, gust-like perturbations, there was a marked improvement in the population that was further evolved (see Figure 5.14), which accumulates fitness more slowly than the agents naïve to intraexperimental changes in light intensity, but this trade-off allows it to survive indefinitely in the highly dynamic environment. Figure 5.15 illustrates that the strategy of the non-naïve agents in response to gust-like perturbations in light intensity is an opportunistic one, in which during a gust the agent is always charging and immediately after a gust, always discharging. Additionally Figure 5.15 seems to imply that the agent takes the strategy of charging at an appropriate rate for its battery level, presumably by regulating its turns to lead it closer or further from the source according to its battery state as it approaches the turn.

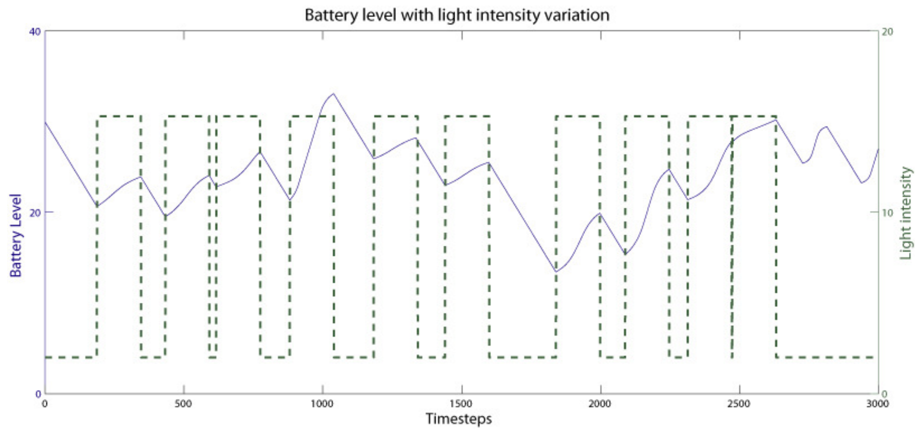


Figure 5.15 A dual plot showing the change in battery level over time (solid line) concurrently with the sudden changes in the light intensity (dotted line)

At the start, when its battery is relatively high, it takes relatively small quantities of the available increased resource within gusts, then after much charge is lost during the lull that begins at c.1600 timesteps, much more energy is extracted from the subsequent three gusts. When the light intensity variation stops near the end of the trial, the agent reverts to its default behaviour. The fact that neither the charge speed nor the reversion to oscillatory behaviour seems consistent is likely due to the agents natural in/out oscillations and the delay inherent in turning where the agent is slowed due to the forward operation of only one wheel.

5.5 Discussion

From the perspective of getting neurocontrollers to perform a desired task there was a clear message which was that breaking down complex problems into subcomponents can be essential. Without evolving the phototaxis behaviour first, it was impossible to evolve the alternating charging behaviour. Breaking down the subsequent task into three progressively more difficult stages also was successful, and we demonstrated that agents evolved to the second stage could not accomplish the third.

Perhaps the most surprising lesson of this exercise for future applications of evolutionary robotics for actual adaptive kite control was the relative success of evolution using the threshold and enactive rules sets for switching between charging and discharging behaviours. Imposing a sharp, externally-defined boundary between two behaviours prevented evolution of successful agents, contrary to their enactive peers which, through their own interactions with the environment determined their own behavioural boundary or lack of it. Given this experience, it would certainly seem prudent to assess a similar scoring mechanism when evolving reel-in and reel-out behaviours for kite control, especially given that a sharp boundary could be subsequently imposed on the enactive switching agents in the toy case with little impact on performance, which may be a necessity due to hardware constraints in kite energy systems.

The strategy of the most successful neurocontrollers was an elegant one that defied our prior expectations of how the problem would be solved. Instead of heading directly for the energy source and then doubling back for discharging, the supposedly discrete behaviours were subsumed into one action constituting a single pass. Whilst the agent approaches the region around line B at an oblique angle that appears to be suboptimal, it actually saves time in reversing direction where one wheel has to effectively stop. Additionally, the 'decision' is then reduced to when, and at what angle to turn in order to start the next charge/discharge pair whose duration and outcome, in a static environment at least, is essentially predetermined. The ability of the evolved neurocontrollers to adapt to changes in light intensity and indeed discharging rates (Figure 5.10) both between and within (Figure 5.13) experiments dispelled our early concerns as to the fragility of this strategy. It was notable how the neurocontrollers were able to adapt to large changes of the agent's discharging rate (Figure 5.11, Figure 5.12) even when these had not been subject to change during evolution. The indication seems to be that robustness to change is not necessarily limited to the parameter subject to change. In this case supposed problems such as a trade off between persistence and dithering almost appear to be resolved almost by the environment itself. This type of behaviour seems best described as an ongoing interaction between the agent and its environment and less well described as one of discrete decisions at the top of a hierarchy being passed down to subordinate effector systems. This may be a product of the relative simplicity of the task and the lack of constraints on the network connectivity. Unfortunately, the relatively large number of interconnected neurons would render analysis of the network dynamics difficult. The imposition of symmetry or other neuroanatomically inspired constraints could potentially generate more transparent internal dynamics.

The final experiment in which neurocontrollers were further evolved with exposure to dynamical environments produced the unexpected result of two co-existing strategies. The standard light passing behaviour switched seamlessly into one in which the environment was permitted to almost completely dominate whether the agent charged or discharged. This apparently submissive behaviour was much more robust than that of the naïve agents, with the agent exerting a more subtle control by increasing or decreasing its exposure to resource variations in order to regulate its state to a level that was notably more conservative than in the prior experiments (<50% vs 75%). In this scenario there was a clear performance cost to reduction of the risk of becoming overpowered. It would be interesting to see whether neurocontrollers could be evolved to match their conservatism to the level of risk in the environment. A somewhat inelegant workaround might be to use different neurocontrollers and deploy them in different operational scenarios. In any case a thorough analysis of this final behaviour in response to a range of gust intensities and durations and indeed more realistic variations in intensity would be enlightening.

In summary, this work has succeeded in its brief of informing future work in the control of kite energy systems at a behavioural level, specifically by suggesting that highly adaptive spatially embedded, behaviour-switching agents can be evolved if decision boundaries are determined by the agents themselves. The opportunistic behaviour seen in the most complex task bodes well for the application of ER to the development of kite control behaviours.

6

If we can really understand the problem, the answer will come out of it, because the answer is not separate from the problem.

Jiddu Krishnamurti

Extended evolution and performance comparisons

Chapters 4 and 5 demonstrated that evolutionary robotics techniques have some value for generating flight control behaviours, appropriate flight trajectories and for investigating strategies of operating in kite generators over longer timescales. However, whilst the flight trajectories evolved in chapter 4 appeared highly suitable for power generation, the robustness of the evolved neurocontrollers in terms of both maintaining an appropriate flight path and preventing catastrophic failure in the task by crashing the kite was far below the level of performance required for a real world implementation operating on costly hardware. The suggestion was made that continuous time recurrent neural networks might be a more suitable substrate, partly because at least the potential exists for some aspects of the ongoing interaction between the agent and its environment to be captured in such a way as to allow some adaptivity to agent or environment change. There were also some concerns about the applicability of recurrent neural networks with a one timestep lag in the recurrent connections, in a real world environment where the period of a single timestep could not be guaranteed.

Unfortunately, as we saw in chapter 4 the continuous time recurrent neural networks struggled to generate flight trajectories that scored as highly as the simpler discrete time recurrent neural networks. As mentioned before, there is little work in the literature dealing specifically with how best to evolve neurocontrollers for difficult to evolve problems. Where shaping schemes and other tools or rules of thumb are applied, it is rarely in the form of a comparative study, though we have already presented a task in chapter 5 in which it was essential to deploy a simple shaping scheme in order to produce valid neurocontrollers. Therefore, this chapter details some experiments in which the configurations of the evolutionary runs are varied in a systematic manner in order to ascertain how best to both initialise the neural network and to manage the evolutionary process. Where some increase in performance is apparent, the configuration is usually adopted, when the effect of a change is neutral and selection between methods is required, it is usually the simplest or most elegant method that is chosen, although the decision is occasionally arbitrary. We acknowledge that changes to neural network and genetic algorithm implementations may interact to create unforeseen changes to the fitness landscape that are not desirable. However there are so many options that assessing these possible interactions is not feasible. The sole motivation here is to simply determine which approach can produce the best performance on this specific task, and not to analyse in depth the cause for any variation in performance using different network or evolutionary configurations.

The work delineated in Chapter 4 did not address the transferability of the neurocontrollers from a simulated task environment to the real world. This chapter takes the first steps in this direction and in the latter half of the experiments a simple method of enhancing the probability of crossing the reality gap through systematic variation of the simulated task parameters is implemented.

6.1 Extended methodology

These and all subsequent evolutionary runs were performed on Matlab running on a Sun GridEngine managed Linux cluster of 20 quad-core Xeon CPU's giving 80 compute cores. In order to prevent any instability in the physics, the timestep was reduced by a factor of 4 from 0.004 to 0.001 seconds. The most frequently repeated calculations in the physics simulation, for example computing lift vectors, were reprogrammed in C and compiled as .mex dlls to reduce the computation time of the Matlab simulation. With the experiments conducted in this chapter, each set of results is internally comparable, for example if network size is being assessed, only network size will be changed and every other trial variable will be kept constant. However, because these experimental results derive from a period of continual development, in which aspects of the physics model or aspects of management of the evolutionary process may be subject to change, the performance scoring between groups are only mutually comparable if explicitly stated. All of the core methodology from Chapter 4 is replicated unless otherwise specified.

6.1.1 Neural network size and dynamic neural network size

It is not necessarily obvious what size of neural network is appropriate in terms of the number of neurons although we might reasonably expect networks that are at extremes of size to have poor performance. Very small networks will not be able to integrate all of the necessary information available in the inputs, and very large networks will have an enormous parameter space that will be difficult to evolve. This is especially true of fully connected recurrent networks in which the number of weight parameters scales exponentially as the square of the number of neurons in the network. If the number of neurons in the network is to be predetermined, it makes sense to test various network sizes in order to gauge the appropriate number of neurons rather than to simply pick an arbitrary value based on intuition and past experience. In the experiments presented below in which the network size is varied, only the number of intermediate neurons is changed, the input layer remains identical.

6.1.2 Neural network initialisation

As well as network size, the range to which the weights in the neural network are scaled needs to be selected. The values in the genome are initialised in the range [-1 1], however it is common in the literature to scale these values to a wider range, with scaling factors of 4,6,10 and even 15 being used. We will compare two moderate scaling factors of 4 and 10 to assess what is most appropriate in this task.

Usually a single activation function is employed and chosen with reference to the properties of the task, however here it is not immediately apparent which activation function is most appropriate or what bearing this choice may have on performance. Here therefore, we assess the comparative performance of neural networks using the logistic sigmoid function and the hyperbolic tangent function respectively. Whilst they are both S shaped functions with an approximately linear central range, there are subtle differences. Perhaps the most salient is that the sigmoid function output is in the range [0 1] whereas the hyperbolic tangent output is in the range [-1 1]. This means that a neuron equipped with the latter function is capable of both self inhibition and self excitation depending on its own internal activation, and its effect on other neurons is similarly more flexible. In this scenario, depending on the time constants, a single neuron is capable of producing an oscillatory output with no external input. However, these characteristics are not necessarily advantageous in every evolutionary robotics task.

A further factor to be assessed is the value of allowing the slope of the transfer function to be evolvable, as some work has reported this to be worthwhile (Pratley, 2005). Figure 6.1 shows the effect of changing the slope parameter k on the shape of the hyperbolic tangent transfer function, which is to effectively tighten or widen the linear portion of the transfer function, where the response to changes in the values of the neuron activation values is most dynamic. Because of the relatively high computational expense use of repeated calls to the hyperbolic tangent function, the values are precomputed to form a lookup table at an input resolution of 0.002 which the neuron update calculations then reference. Because of this pre-computation, only six K values are permitted, which are [0.0125, 0.25, 0.5, 1, 2, 4] which generate the slopes shown in Figure 6.1. The mapping from the genome to the appropriate k values is implemented using a simple binning scheme to map the continuous genome values from the range [-1,1] to the six K values.

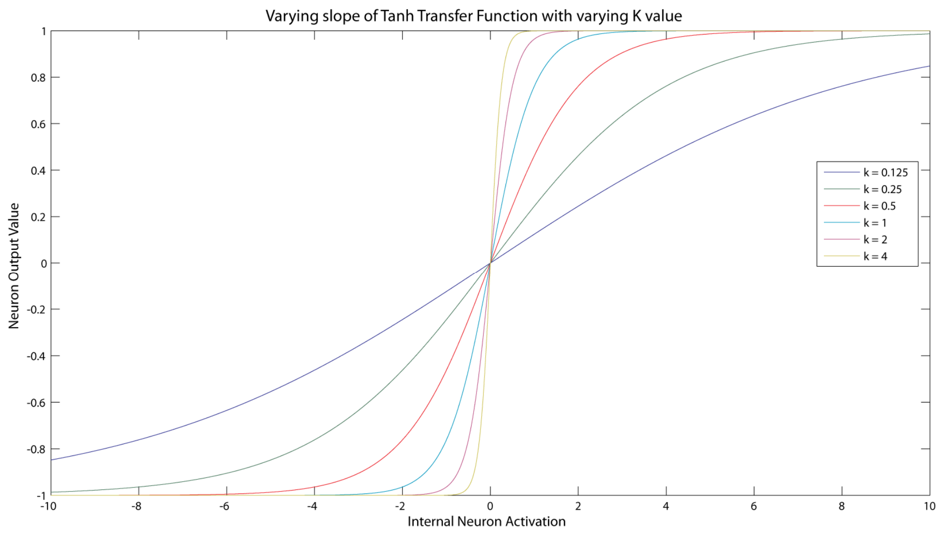


Figure 6.1 Plot showing the change in slope of the transfer function using the 6 evolvable values of the K parameter.

Another factor relating to the implementation of the neural network is whether internal neuron noise is implemented. There is some evidence that surprisingly high levels of noise may in fact be beneficial if the goal is to create neurocontrollers that are robust, i.e. that exhibit limited graceful degradation of performance in response to perturbations (Fernandez-Leon & Di Paolo, 2008),(Fernandez-Leon & Di Paolo, 2007). Here we implement this internal network noise simply by changing the bias of each neuron at each timestep by a normally distributed Gaussian value of mean 0 and s.d. 0.1. Noise at the same level is also applied to network inputs and outputs.

It is often the case in evolutionary robotics implementations that networks evolved with some kind of noise can become dependent on that noise in order to operate successfully. Although here we can guarantee the presence of noise in a way that is not possible if the noise is meant to mimic some aspect of noise on a real world sensor, we still here make the noise unreliable by making its presence state random from trial to trial with a 50% probability.

6.1.3 Evolutionary initialisation and management

Our default configuration in all the experiments within this chapter is to allow genetic crossover in contrast to work in chapter 4 where solely mutation operators were permitted. Crossover is implemented probabilistically at either 0 (i.e. no crossover), 1 or 2 loci within the genome, such that in the latter case one portion of the offspring's DNA derive from one parent and two portions of the offspring's DNA derive from the remaining parent.

We implement the minimal 1D geography described by Spektor (Spector, 2005), to evaluate its effect on performance. In this method, the population is distributed on a toroidal grid as shown in Figure 6.2. The first competing individual in a pair is selected randomly as usual, however the individual with which this initial solution is compared against must be selected from a geographical neighbourhood known as a deme. This method can potentially allow a greater diversity amongst the population as a more fit solution will take longer to dominate the population. However it is possible that this longer time for a solution to propagate through the population, might delay the convergence of the population to favourable solutions. This would reduce the number of evaluations that take place in the volumes of parameter space nearest to the current fittest solutions. Because our primary concern is performance, we will assess evolution with and without this spatial heuristic on performance grounds alone and not in terms of population dynamics or speciation.

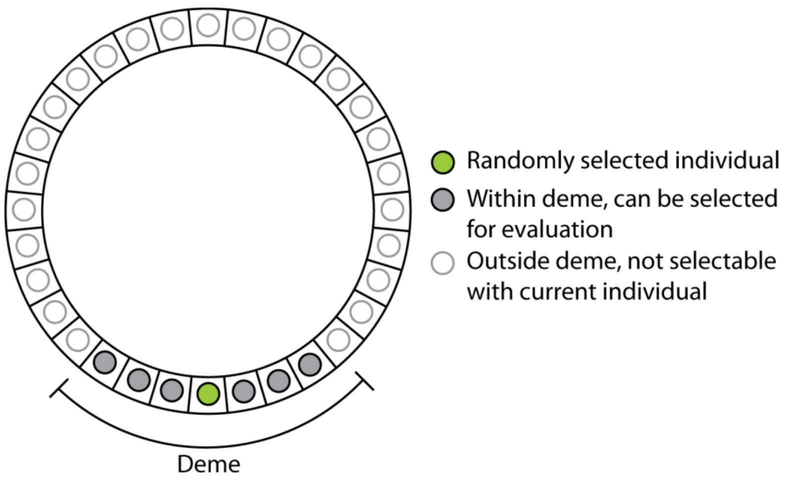

Randomly selected individual
Within deme, can be selected for evaluation
Outside deme, not selectable with current individual
Deme

Figure 6.2 Minimal geography implementation, competing pairs are selected from a geographically restricted region marked as the deme

Conscious of the need for the evolved neurocontrollers to be transferable from the simulated task environment to the real world, in the latter half of the chapter systematic variation is implemented on both implementation aspects of the simulation and the physical attributes of the simulated kite. Each generation, a new set of these values is generated and all of the trials within this generation will use this set of values. All the experiments from this chapter were based on two evaluations of each neurocontroller to generate a single score value, varying the wind conditions between the each evaluation.

The implementation aspects that are varied are essentially parameter values that might have a real world analogue but have been generated arbitrarily here, for example the drag profile and the lift profile, as it changes with attack angle. The drag coefficient values used in the aerodynamic simulation have a single random number in the range [-0.01 0.01] added to every data point, and the lift coefficient values have a different random number of the same range applied in the same manner. We also introduce here a ratio that scales the drag coefficients of the panels at each tips of the wing furthest from the centre of the kite, in order to some way approximate the greater drag that occurs at the wing tips due to downwash arising from spanwise flow. This ratio is set to a random value in the range [1.1 2] corresponding greater drag at the wing tips of between 10 and 100% compared to the centre portions of the wing. The most clear-cut implementation aspect is the damping factor which has no direct analogue but simply serves to prevent unrealistic oscillations building up. This value is varied in the range [0.09 0.15].

A second set of parameters to be varied have direct analogues in the real world, the first of these is the line length which is varied in the range [25 30], the second is the relative lengths of the front and rear lines which are varied in the range [0.05 0.05]m, the third is to be mass per square metre of the kite which is varied in the range [0.2 0.4] kg/m^2, the fourth is the span of the kite which is varied in the range [1 3]m and the last parameter to be varied is the aspect ratio of the kite corresponding to the ratio of span to cord which is varied in the range [4 6].

6.2 Results

Because of the inherent noisiness of the evaluation due to random changes in the resource intensity and the imposed parameter variation, it can be difficult to see the trends behind the generation to generation variation. Therefore in many of the plots demonstrating fitness progression over an evolutionary run, a running average is used, typically averaging over a moving window of 5% of the total number of generations. The window size is available in the title of each plot. Due to the computationally intensive nature of running these evolutionary runs, generally only between 6 and 10 runs could be run for each configuration which is intended to give some indication of the relative strengths of each approach and indicate areas that might be particularly useful. To achieve a high level of statistical significance many more evaluations would be necessary, bars of standard error are generally shown to give an indication of the variability of the data.

6.2.1 Comparative performance with neural network size

In the first experiment, the effect on performance of the number of into neurons in the neural network was investigated. 7 evolutionary runs with each of four, five, six and 10 into neurons were initiated. In the plots of performance over the evolutionary period we can see that the average population fitness (see and the best score per generation appear similar for networks with four and five neurons. However, the evolutionary runs with six neurons did appear to achieve higher scores and the networks with 10 achieved higher performance still in these tests.

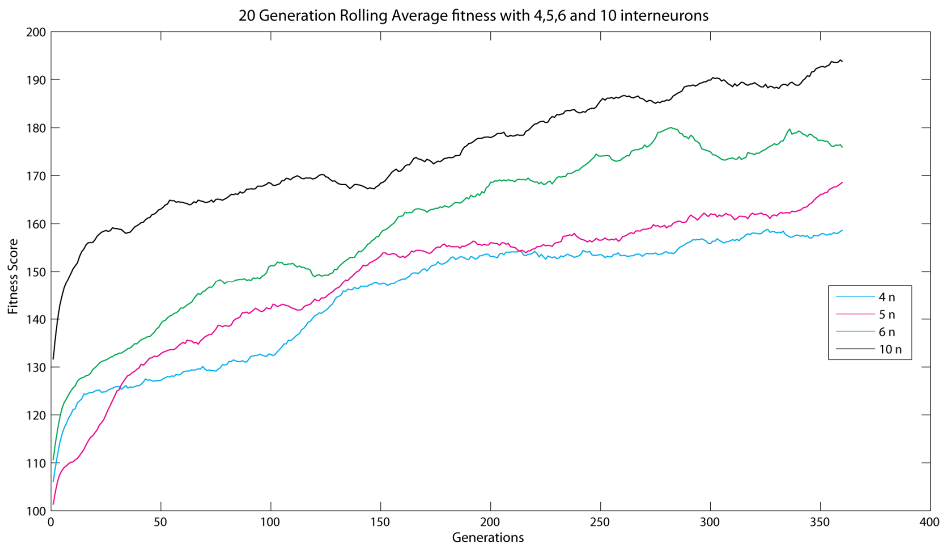


Figure 6.3 A rolling average plot of the average population performance of 7 runs with each of 4, 5, 6 and 10 interneurons over 400 generations of evolution, standard error bars omitted for clarity

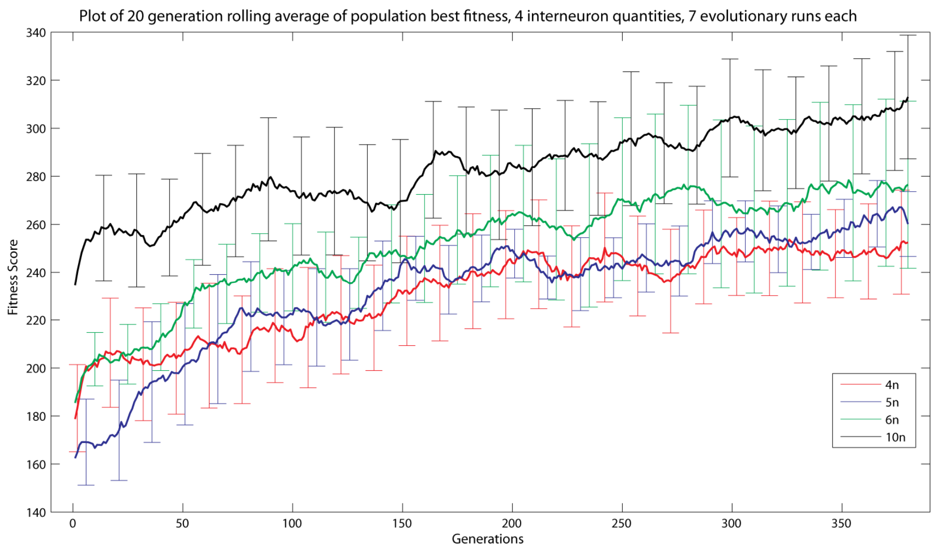


Figure 6.4 A rolling average plot of the best population performance of 7 runs with each of 4, 5, 6 and 10 interneurons over 400 generations of evolution, bars show standard error.

6.2.2 Comparative performance with weight scaling factor

The below plots show performance data detailing the average (Figure 6.5) and best (Figure 6.6) fitness per generation of 10 interneuron networks with a scaling factor of 4 and 10 to bring the weights from the genetic data range of [-1 1] to [-4 4] and [-10 10] respectively. 7 evolutionary runs of 400 generations were performed for each scaling factor. It is apparent that both the population average performance and the best performance achieved are nearly identical between the two groups. It is noteworthy however that the runs with the scaling factor of 4 had much higher variability than the runs with the scaling factor of 10.

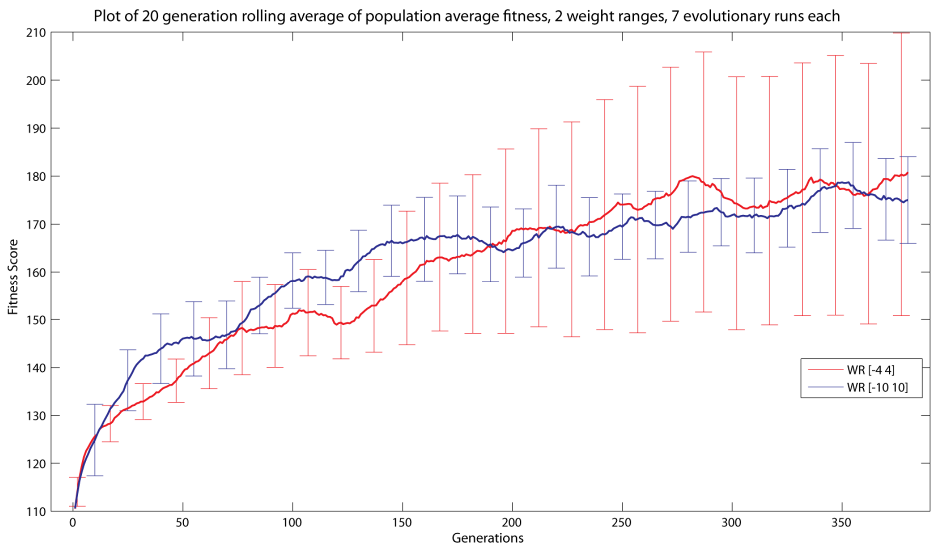


Figure 6.5 Plot showing the average score of the population over 400 generations of evolution in two groups of 7 evolutionary runs. The red line corresponds to weights scaled to the range [-4 4] and the blue line scaled between [-10 10]

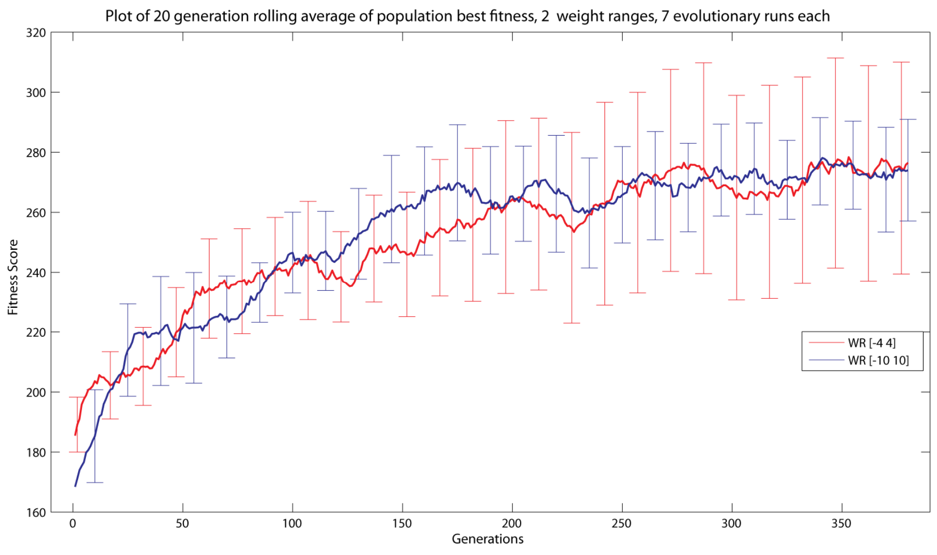


Figure 6.6 Plot showing the best score of the population over 400 generations of evolution in two groups of 7 evolutionary runs. Weights in range [-4 4] (red) and [-10 10](blue) .

6.2.3 Effect of minimal geography on evolution performance.

The below plots show performance data detailing the average (Figure 6.7) and best (Figure 6.8) fitness per generation of 10 interneuron networks with and without minimal geography operating with a deme size of 20% of the population size. 10 evolutionary runs of 400 generations were performed for each case. The average and best performance of the runs in which the population is spatially distributed appears to be lower than those runs without the minimal geography operating, however the error bars indicate an overlap in performance and need for more runs to be performed.

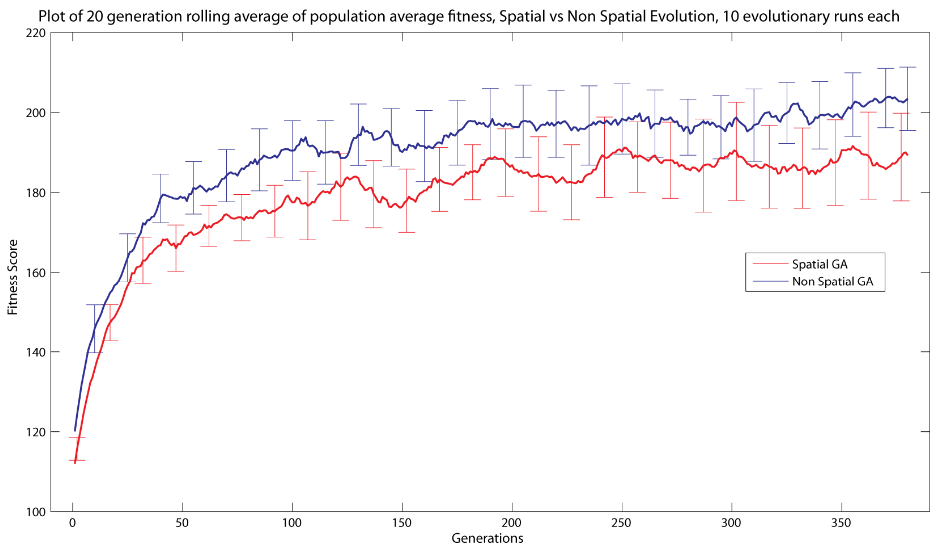


Figure 6.7 Plot showing the average score of the population over 400 generations of evolution in two groups of 10 evolutionary runs. The red line corresponds GA operating with individuals in the population distributed and selected according to a minimal geography, the blue line corresponds to a GA operating without any geographical heuristics.

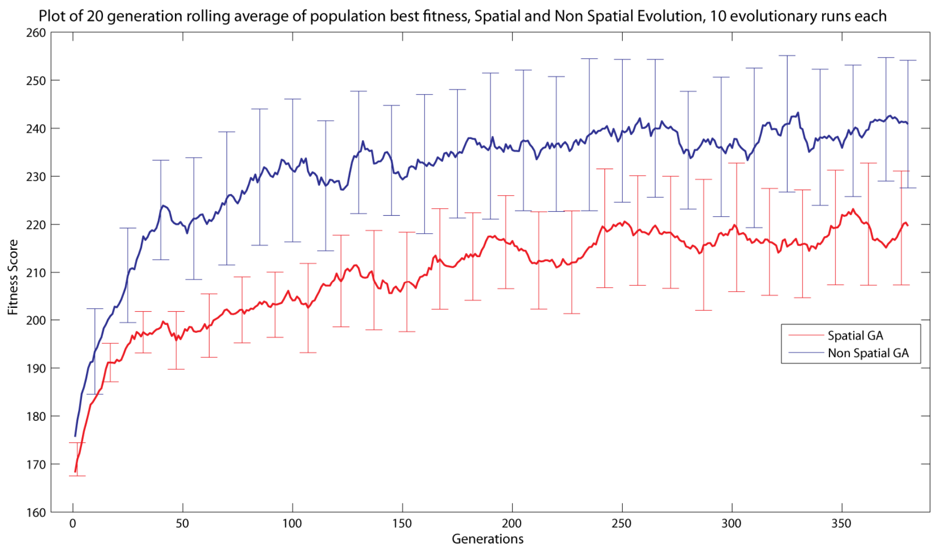


Figure 6.8 Plot showing the best score of the population over 400 generations of evolution in two groups of 10 evolutionary runs. The red line corresponds GA operating with individuals in the population distributed and selected according to a minimal geography, the blue line corresponds to a GA operating without any geographical heuristics.

6.2.4 Effect of changing the activation function

In these and all subsequent experiments, the systematic variation of parameters described earlier is deployed. The below plots show performance data detailing the average (Figure 6.9) and best (Figure 6.10) fitness per generation of 10 interneuron networks evolved using sigmoid and tanh activation functions. 8 evolutionary runs of 800 generations were performed for each case. Both the average and best performance of the runs in which the tanh activation function is used is initially much lower than that of the networks using the sigmoid activation function, however over the period of evolution, the performance of the tanh networks steadily improves whereas the sigmoid networks appear to rapidly reach a plateau beyond which fitness never improves. It is notable that these experiments in which the systematic parameter noise is enabled produce average results that are much reduced compared to the previous plots, and best fitness results that are also somewhat reduced.

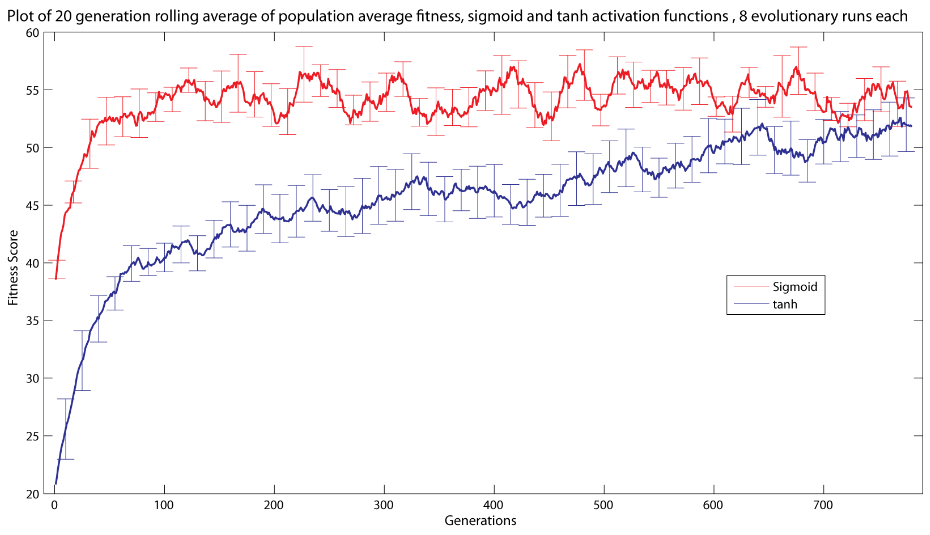


Figure 6.9 Plot of 20 generation rolling average of population average fitness the data in red corresponding to the sigmoid activation functions and the data in blue corresponding to tanh, 8 evolutionary runs each

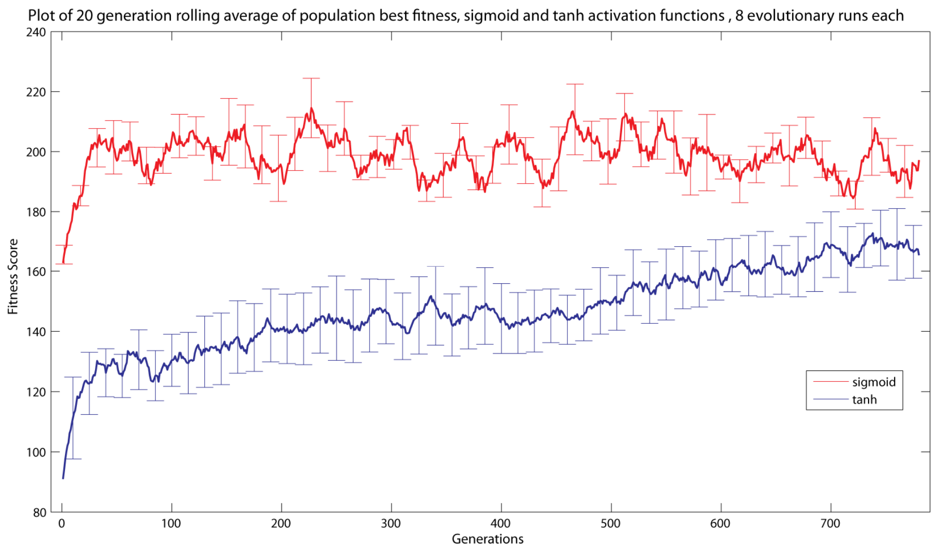


Figure 6.10 Plot of 20 generation rolling average of population best fitness, the data in red corresponding to the sigmoid activation functions and the data in blue corresponding to tanh, 8 evolutionary runs each

6.2.5 Effect of allowing the slope of the activation function to be evolvable

The below plots show performance data detailing the average (Figure 6.11) and best (Figure 6.12) fitness per generation of 10 interneuron networks evolved using with and without an evolvable k parameter with which the slope of the activation function can be modified. Eight evolutionary runs of 800 generations were performed for each case. Interestingly, even though in the case of evolvable k parameters there is an additional burden to the evolutionary process of 10 parameters, both the average and best performance of the runs in which k may be evolved are marginally higher than the runs in which k is fixed at a moderate value of 1. The gap in performance is more sustained than the previous case but the level of difference is relatively slight. A greater quantity of evolutionary runs, ideally operating for more generations would be necessary to make a more concrete conclusion.

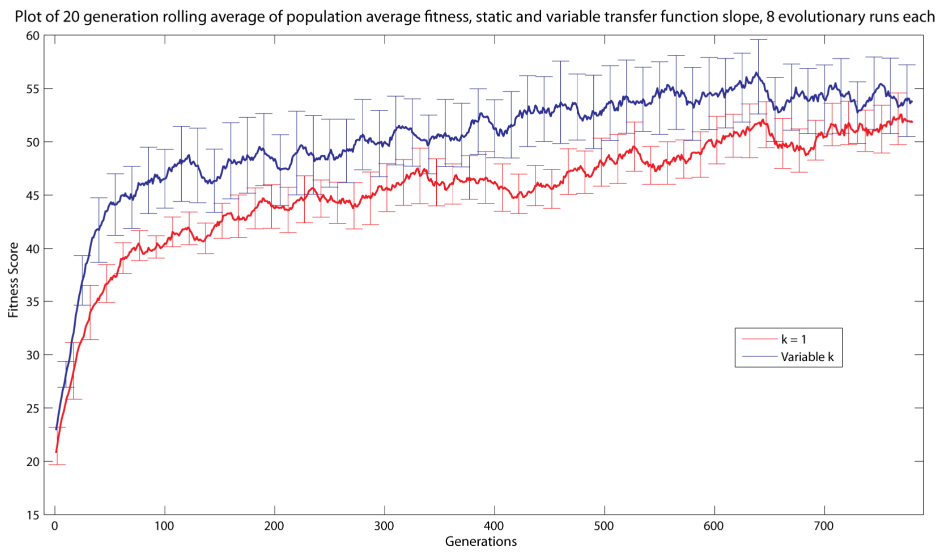


Figure 6.11 Plot of 20 generation rolling average of population average fitness, the data in red corresponding to using a single transfer function slope with k=1 and the data in blue corresponding to allowing evolution to select between 6 transfer function slopes per neuron, 8 evolutionary runs each

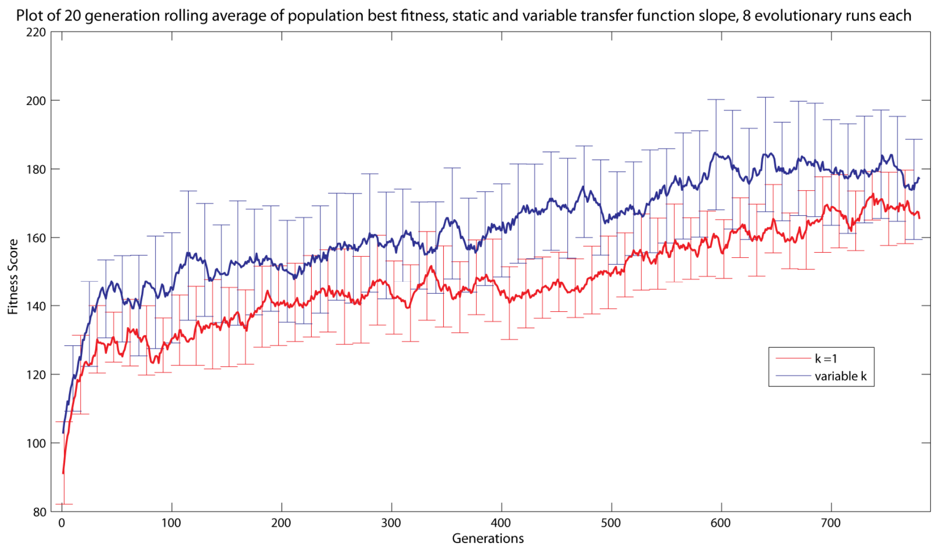


Figure 6.12 Plot of 20 generation rolling average of population best fitness, with static and evolvable k, 8 evolutionary runs each

Figure 6.13 and Figure 6.14 show the population average and best individual performance of the best evolutionary run. Interestingly, whilst the fixed k, tanh activation function whose best run showed the poorest performance of the three configurations, it was the tanh activation function with evolvable k that had the best single performing run. In this instance the use of an evolvable transfer function slope has overcome the apparent deficit of using tanh as a transfer function as opposed to the sigmoid function.

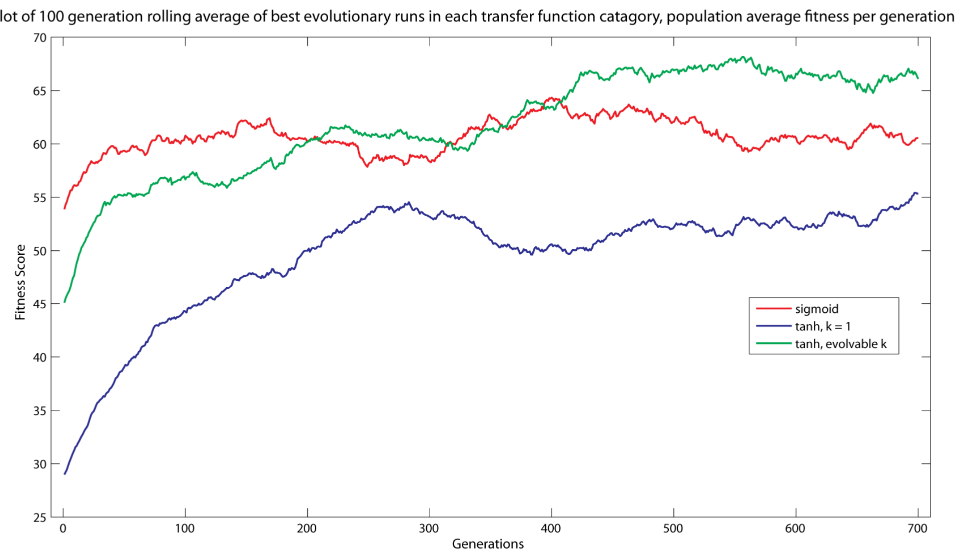


Figure 6.13 Plot of 100 generation rolling average of per generation average fitness of the whole population of the best single evolutionary run with sigmoid activation function(red), tanh activation function with a static k value of one (blue) and tanh with evolvable k values (green)

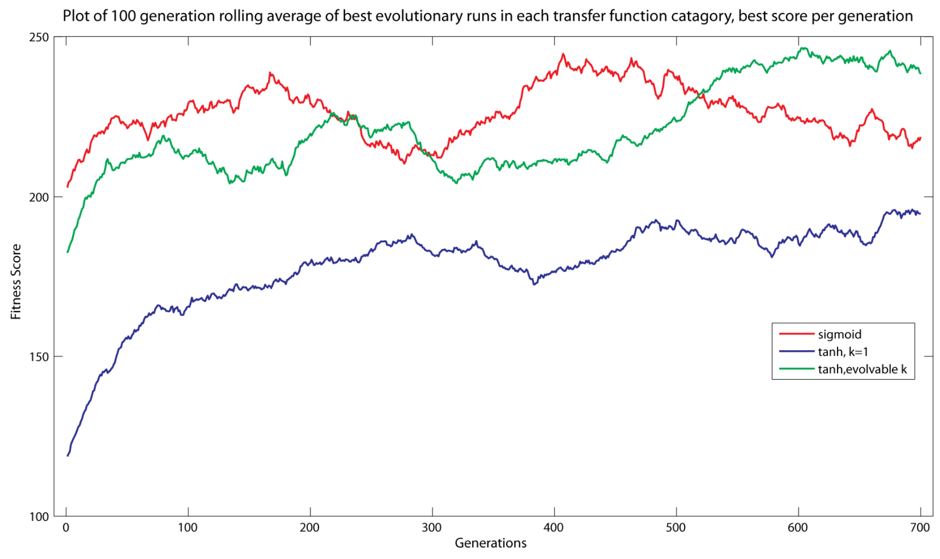


Figure 6.14 Plot of 100 generation rolling average of per generation average fitness of the whole population of the best single evolutionary run with sigmoid activation function(red), tanh activation function with a static k value of one (blue) and tanh with evolvable k values (green)

6.2.6 Effect of neuronal noise on performance

The below plots show performance data detailing the average (Figure 6.15) and best (Figure 6.16) fitness per generation of 10 interneuron networks evolved with and without noise applied to the bias of each neuron. 10 evolutionary runs of 2000 generations were performed for each case. Both the average and best performance of the runs in which the internal neuron noise is applied is higher than that of the networks without internal noise. In contrast to some of the other results, this performance gap seems to persist throughout the whole of the period of evolution.

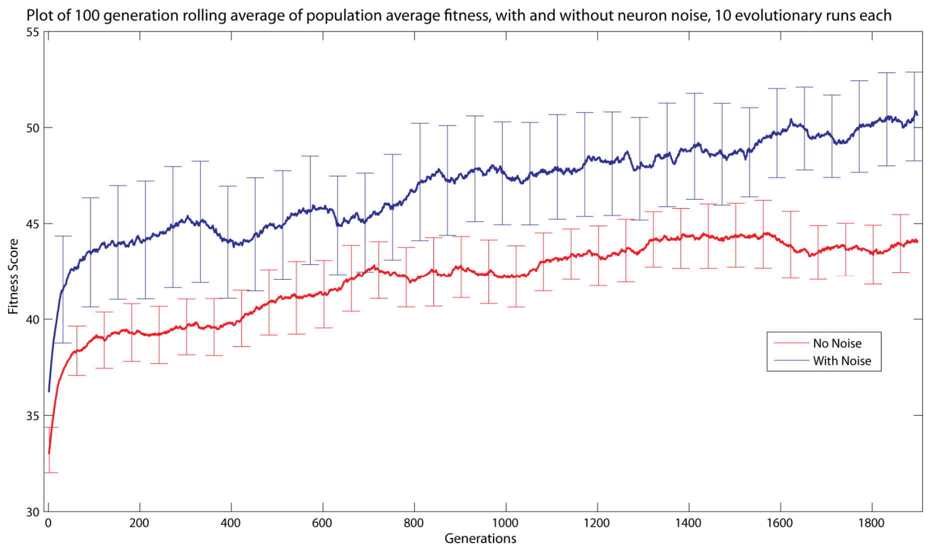


Figure 6.15 Plot of 100 generation rolling average of per generation average fitness of all runs both with and without internal neuronal noise. 10 runs per configuration were evolved for 2000 generation each.

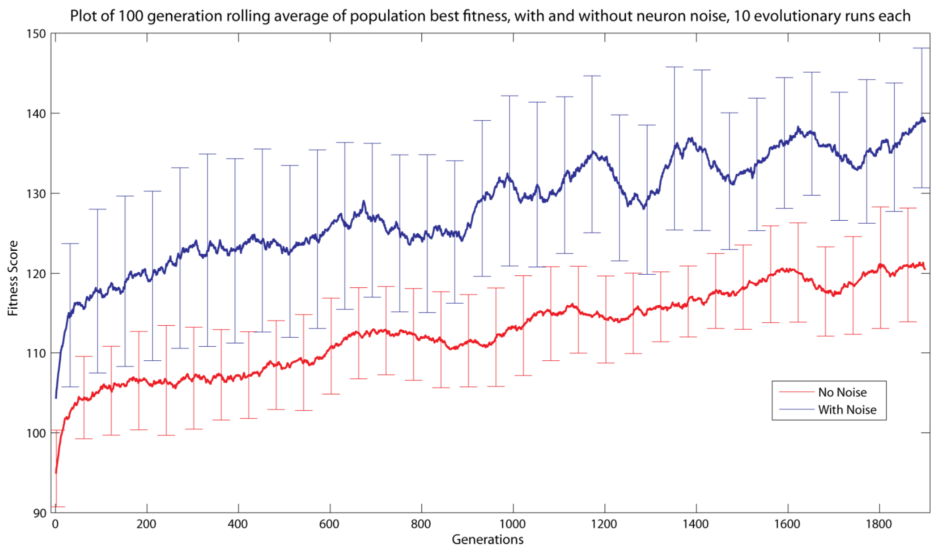


Figure 6.16 Plot of 100 generation rolling average of per generation average fitness of all runs both with and without internal neuronal noise. 10 runs per configuration were evolved for 2000 generation each

6.3 Discussion

Generally it appears that changing the configuration of the initialisation of the neural network, the details of its operation and the way in which the genetic algorithm is operated does indeed have an effect on the performance of the resulting neurocontrollers. This performance difference is of a magnitude that makes it worthwhile to consider these issues. As stated above, the principal reason for undertaking these explorations was simply to ensure that an appropriate implementation of the evolutionary approach was used for this kite control problem, and not to perform detailed analysis on the dynamical underpinnings behind these changes and how those might relate to the task. However as we briefly review the results in this section, commentary suggesting potential reasons for any performance differences and suggestions for future investigation in that direction will be included.

In section 6.2.1, we started by investigating the size of the neural network because it is a fundamental property of the network that impinges on with the operation of the genetic algorithm because of the direct effect on the number of parameters to be evolved. Specifically, in a fully interconnected network, the number of weights will increase exponentially along with the number of neurons. In the evolutionary robotics literature there are examples where surprisingly complex behaviours can be produced by remarkably small continuous time recurrent neural networks. The expectation prior to running these experiments was that a whilst very small neural networks of two or three interneurons might not perform well, the addition of even one extra neuron would markedly increase performance whilst networks above maybe half a dozen neurons might actually be detrimental in an evolutionary process purely because of the additional number of parameters to be evolved. Counter intuitively, the results demonstrated that increasing the neural network size appeared to have little effect between four and five neurons. A single extra neuron, bringing the total to 6 however did seem to provide markedly better performance. Rather than producing no advantage or even a reduction in performance, the 10 interneuron networks seemed to achieve far superior performance. This is notably different to the work of Beer considered in Chapter 2 (R. D. Beer et al., 1999),(H J Chiel et al., 1999) in which a large difference was seen as small networks were incremented in size but performance soon plateaued. Interestingly, although one might expect that our larger networks might suffer a performance penalty at the beginning of the evolutionary run, superior performance was apparent right from the first generations. The reasons for the apparent requirements of this task for comparatively large CTRNNs are not immediately clear, it might be that the dynamic requirements of the task simply require more complex internal dynamics. In Beer's task it could be shown that the medium size networks were already performing very close to a theoretically perfect solution so it is not surprising that performance increases with neuron addition plateaued. The fact that in our task typically some of the neurons within the network are saturated at extremes of the activation function even when centre crossing is used, might mean that the smaller networks simply do not have the resources to match the dynamics of the task environment. In the case of networks with multiple inputs it might be necessary to supplement the centre crossing condition to include expected inputs ranges in order to maintain neurons in the dynamic portion of their input range, whereas Beer's fully autonomous, input free networks do not have this additional consideration. Any further speculation would probably not be productive, however there is certainly scope for an in-depth investigation of what size network is appropriate for different types of task. It would also be worthwhile to assess at what network size the potential performance advantage is outweighed by the penalising effect of having to evolve a very large set of parameters.

In the next section 6.2.2, we investigated the effect of changing the weight scaling factor responsible for scaling the genome parameters from their range of [-1 1] to more appropriate values. Whilst less clear-cut than the results from the prior section, there was nevertheless an interesting pattern to be observed. Specifically, although the average performance of the evolutionary runs using the scaling factors of four and 10 respectively was a near identical, the variation in performance between the evolutionary runs in the case of the lower scaling factor of four was far greater than that of the runs in which the higher scaling factor was used. Worth noting is the fact that this variability seemed to increase over the evolutionary period in the case of the lower scaling factor but not noticeably for the higher scaling factor. Also, for the scaling factor of four there was more variation in the case of the average performance per generation than the best performance per generation, where conversely for the scaling factor of 10 there was more variation in the best performance case. This might indicate that for the lower scaling factor, whilst the best networks were operating around some local performance ceiling, the whole population was more subject to change from the mutation produced within a single generation. This apparently greater sensitivity is likely to derive from the S shaped transfer function of a single neuron, where weights are low, the internal activation value of each neuron before it is passed through the transfer function is likely to be nearer zero thus placing the neuron at the most dynamic part of the S curve i.e. it is linear central section. A change in weights is therefore reasonably likely to yield a significant behavioural effect. The neurons with the higher weight scaling factor are more likely to be in the saturated portions of their transfer function, however due to the potential for higher input values, they are potentially subject to more powerful input driving changes in their internal neuron activation value. That these possible dynamical differences actually produce average performances that are near identical is remarkable in itself, further investigation in this area might show directly what leads to these different performance characteristics, and suggest how best the positive aspects of each of these weight scaling ranges might be harnessed. Also warranting further investigation is whether a given individual using each of these scaling ranges is more or less susceptible to environmental or parameter variation. This would help tease out whether these changes relate more to evolutionary parameter space exploration or network dynamics.

In section 6.2.3, we compared the performance of multiple evolutionary runs both with and without spatial distribution of individuals within the population affecting the selection of pairs of individuals to compare. Interestingly, the runs with the minimal geography operating generally scored lower than those runs with no geography across the whole period of evolution. The gap appeared to be larger for the best performance per generation than the average performance per generation. These results are in direct contrast to those from Spector et al (Spector, 2005) which demonstrated dramatic improvements in performance when using this minimal geography heuristic. It is not clear why there is such a large difference in the results, however the authors of that work were using a evolutionary programming technique and not a genetic algorithm, and applying it to not to a dynamical control task but to a series of linear regression problems. The minimal geography technique is posited to increase diversity across the population, partly by limiting the speed at which the population converges to the current best solution. It appears that when a using a genetic algorithm at least for this task, this is not a desirable characteristic. Perhaps the fact that the GA is operating on a large number of parameters, in this case nearly 200, means that local optima are less likely to be a problem and neutrality within the solution space allows continual exploration even from a relatively good current solution. These results suggest that a full exploration using different deme sizes and assessing whether 2D and 3D geography schemes are more or less beneficial for different types of evolutionary robotics problems would seem to be necessary. Perhaps the lesson that can be immediately drawn from these results is that applying techniques developed for one technique or task will not necessarily generate similar beneficial effects on different evolutionary techniques and classes of problem.

In the following section 6.2.4, we compared the performance of networks of the same size using different neuron transfer functions, specifically the logistic sigmoid function and the hyperbolic tangent. Having intuitively expected to see superior performance from the networks using tanh, on the assumption that oscillatory behaviour would be easier to generate by neurons capable of both positive and negative outputs, it was somewhat surprising to note that there was a sizeable gap in performance with the sigmoid networks outperforming the tanh networks in both the average performance of the population per generation and the best performance per generation. However, over the course of the 800 generations of evolution, this gap narrowed more than any of the other comparative experiments in this chapter, the average population fitness was negligibly different and the best performance only slightly different. Interestingly, in common with all of the prior results in this chapter the fitness of networks utilising the sigmoid transfer function plateaued very early in the evolutionary process, with no further gradual improvement and with no runs displaying any sudden performance improvements during the evolutionary runs such as was seen in chapter 4. Although in these experiments were terminated before it could be determined if the tanh networks would surpass the performance of the sigmoid networks, the results of section 6.2.6 show sustained improvement in performance over 1000 more generations of evolution. If the sigmoid networks continued their plateaued performance level then the tanh networks would ultimately demonstrate better performance. This extrapolation would clearly demand further experiments in order to be substantiated. The reasons for this interesting performance difference are not immediately obvious, however if the tanh networks are indeed more prone to oscillatory behaviour that the sigmoid networks, it might be that the tanh networks are steering the kite more aggressively, leading to more failures in terms of trials ending in a crash.

In contrast the sigmoid networks might converge on control approaches that are adequate in terms of keeping the kite aloft but somewhat limited in terms of producing looping paths, which might be an interesting path for future investigation, ideally using some quantitative metrics with which to compare flight control behaviours, other than just an average tension score.

In the related section 6.2.5, we compared the performance of networks using the evolvable k parameter which modifies the slope of the tanh activation function, with the simpler case where the value k is set to 1 by default. One study seemed to suggest that in some cases this might be advantageous ((Pratley, 2005)) yet the additional burden of having to evolve extra parameters seemed to justify a comparative test of performance. Interestingly in the first tens of generations the performance was almost indistinguishable, however after about 50 or so generations the networks using evolvable k term, displayed both superior average population per generation and best performance per generation to the networks where k was set to 1. The difference however was not enormous and there were some signs that the average performance per generation of the two groups was converging by the end of the 800 generation evolutionary period. It would certainly be worthwhile to run large groups of evolutionary runs for a further period if the computational resources were available. However the preliminary indication is that the additional flexibility of widening or shrinking the dynamic range of the transfer function of individual neurons justifies the cost of evolving additional parameters. The last plots in this section (Figure 6.13 and Figure 6.14) show the performance of the best evolutionary runs from each of the three groups sigmoid, tanh with fixed k and tanh with the evolvable k. Whilst the performance of the sigmoid network barely seems to improve after the initial hundred generations, both of the tanh-utilising networks showed improvements over long periods that often appeared to be punctuated by moderate periods of neutral evolution where no performance increase occurred. Taken as a set, the implication seems to be that for very short runs sigmoid transfer function networks might be the most appropriate to achieve an adequate result, however for evolution over longer timescales it might be more appropriate to use networks with the tanh transfer function, ideally with an evolvable transfer function slope.

In the final section 6.2.6, we assessed the effect of the addition of an internal neural noise term on performance. These experiments were implemented solely out of interest following recent work (Fernandez-Leon & Di Paolo, 2008),(Fernandez-Leon & Di Paolo, 2007)) which presented demonstrable improvement in robustness of neurocontroller behaviours to perturbation when using internal neural noise, albeit applied in a different manner. Our results were perhaps the most striking out of the whole set, the runs in which the internal neuron noise term was implemented unambiguously outperformed their counterparts that did not have any noise applied. This difference in performance was present from the beginning the evolutionary run and did not appear to diminish in any way over a relatively long period of

2000 generations. Whilst the natural intuition is that the presence of noise would impair the ability of the neural network to integrate information relevant to its environment in order to generate valid behaviours, it appears that either this 'impairment' actually conveys some advantage, or that this effect is neutral and another mechanism is responsible for the increase in performance. It might be that by introducing some element of unreliability into the performance of the neural network and therefore presumably its behaviour, networks that exploit some strategy that is fragile or that happen to achieve inappropriately good scores by relying on the exploitation of some possibly unrealistic aspects of the simulation, are less likely to be successful thereby focusing evolution on valid strategies that are more likely to produce good scores over a number of generations. It is worth revisiting here the fact that this noise is probabilistic and not applied in every single evaluation, therefore the evolved neurocontrollers are not dependent on the noise being present. Whatever the cause of the increase in performance when using this internal neural noise, it does seem to be a useful and easy to implement mechanism to achieve superior performance on evolved CTRNN tasks. It would be interesting for some further work to assess exactly what categories of task this benefits extends to, and what is the best level of noise to apply, for a given neural network class or class of problem.

6.4 Summary

In summary, the results presented in this chapter demonstrate that it is worthwhile to investigate the way in which both the neural network is implemented and the way that the genetic algorithm is operated for a target problem by using comparative trials. Not every change to the evolutionary robotics implementation that would appear to be an improvement actually resulted in superior performance. For this task, the implementation of minimal geography implemented here was actually detrimental to performance. Equally some implementation details might have different effects depending on the length of evolution, for example the use of the sigmoid transfer function appears to allow early convergence on an adequate solution yet seemed to engender a stagnation of the evolutionary process where further improvement in performance was hindered. Of particular note was the use of an internal neuron noise term, which seemed to have a beneficial effect on performance out of all proportion to the difficulty of its implementation. However, it is by no means certain whether this performance advantage would transfer to different tasks, neuron models or modes of evolution.

This is perhaps the most salient message of this chapter, specifically that if performance is important, the effects of parameter or implementation changes should not be taken for granted unless prior results using a very similar methodology on a very similar task are available. It seems that there would be great value in assessing a suite of these less glamorous implementation details on a suite of tests problems that might enable researchers to pick the implementation best suited to their task. This could even be extended to assessing how these factors might interact. The final chapter dealing with experiments performed in simulation will hopefully benefit from the application of the lessons learned from these initial investigations.

7

If we have learned one thing from the history of invention and discovery, it is that, in the long run - and often in the short one - the most daring prophecies seem laughably conservative.

Arthur C. Clarke,

A full noise envelope and a transferable flight stabilisation behaviour through multistage shaping

This is the final chapter that is concerned with the evolution of neural networks for a simulated flight control task. It seeks to integrate the basic approach elucidated in chapter 4 and the lessons learned regarding implementation in chapter 6, as well as taking account of likely hardware configurations and operational parameters. The focus is on developing and implementing a methodology that is capable of producing neurocontrollers that will generate a valid flight behaviour that is suitable for implementation on our real-world hardware platform. To this end, we begin with an application of the existing technique with an expanded range of variation of the simulated task parameters on a modified version of the original kite control task. We then introduce a flight stabilisation control task that requires less heavy duty equipment to be available at the ground level. We also introduce the use of hand coded neural network seeds in some cases. We describe and deploy a systematic method of progressively releasing difficulty in the simulated task in order to render evolution possible with the high levels of noise and parameter variation that are required. We also introduce the concept of levels of fidelity of simulation, describing a simpler kite physics model and two more complex models, one of which requires a model tuning or parameter identification process performed by evolving model parameters in order to reproduce measurements of deformation under load of a real world wing.

7.1 New flight control tasks

We begin with an application of the existing technique with an expanded range of variability imposed upon the simulated task parameters, at a level that might feasibly allow the transfer of behaviours from the simulated task environment to the real world. In these evolutionary runs we require the neurocontroller to be able to perform an appropriate flight manoeuvre that acts to maximise the tension in the lines whilst pulling its line from a reel at a range of speeds. This is a similar task to that used in chapters 4 and 6, however the change in the line length renders the task far more difficult, because the angular speeds will become far slower, as the kite goes higher, rendering a direct ‘mapping’ from input to output even less appropriate.

The second control task is essentially the counterpart to the aggressively looping flight behaviour developed previously, namely that of stabilising the kite at the zenith position, i.e. as close to directly above the tether point as the glide ratio will permit. This will result in the minimisation of tension in the tether thereby allowing the kite to be retracted with the minimum of energy expenditure. Whilst this task may appear trivial in comparison to the acrobatic flight control task, it does entail considerable difficulty. Specifically, whereas in the looping flight task the controller effectively determines the speed with which air flows over the kite and thereby the resulting forces, in the stabilisation task the static kite is subject to the full effect of natural wind speed variation. As well as maintaining the kite's position at zenith when starting at that position, we will also expect these controllers to be able to navigate the kite to zenith from any static position at the edge of the wind window. To ensure compatibility with controllers for looping flight, we also expect a controller to be able to steer the kite to the zenith position from situations where the kite might be moving in any direction at any position in the wind window. The implementation of the task initialisation and 'handover' from dynamic flight manoeuvres is described in section 7.3.5.

7.2 Model precision concerns

As well as trying to develop a methodology that is simply concerned with creating neurocontrollers that are most likely to function adequately in the real world, this chapter also revisits some of the issues raised in chapter 3. There it was posited that to develop neurocontrollers to be used to control a poorly understood physical system with fast and unstable dynamics, the simplest valid 'minimal simulation' might be a physics simulation that, whilst simple, might go beyond the bounds of what was envisaged in the original radical envelope of noise methodology. This would carry the risk of bringing our assumptions into the evolutionary process and allowing evolved neurocontrollers to exploit aspects of a specific model that might not be reproduced in the real world. Here we introduce three extra kite models, one simpler model and two more complex models that might be considered closer to the real world system. By evolving neurocontrollers using multiple models, we are more likely to create model independent controllers, and by creating a more accurate and more computationally demanding model, we can validate controllers evolved on the simpler models. The creation of the most complex model necessitates a model validation process in which parameters of the model are evolved using real-world measurements. By creating this model hierarchy we can assess the generalisation properties of the neurocontrollers evolved using the simpler model and hope to draw some conclusions about how much precision is desirable in this type of process.

7.3 Extended methodology

7.3.1 Neural Network Methodology

As per the previous chapter, we will follow the existing methodology as previously described unless otherwise stated. Following from the results in chapter 6, we use a fully interconnected CTRNN with neurons that employ a tanh transfer function, evolvable transfer function slopes via the k parameter, Gaussian distributed noise on each neuron bias centred on zero with a standard deviation of 0.01. However we also make several changes to the implementation as described below.

7.3.1.1 Modified input parameters

Because this work was performed concurrently with the development of test hardware, a better idea of what input parameters would be available in a real world implementation was available. Therefore the input parameters available to the network were changed to those listed in Table 2 below.

Input No.	Input Data
1	Load cell tension estimate
2	Azimuth angle estimate
3	Elevation angle estimate
4	Orientation angle estimate (yaw in kite coordinate system)
5	Line Length

Table 2 List of modified input parameters available to the network

7.3.1.2 Dynamic neural network size and architecture

In chapter 6, we performed some experiments trying to determine what network size was appropriate to the task at hand. An alternative approach is to allow evolution to pick the size of neural network that achieves the best performance, thus avoiding the requirement for comparative analysis for every problem used. However, as pointed out by others in the literature (Jakobi & Quinn, 1998) this does introduce complications in the encoding of the network parameters. Specifically, in a fully interconnected CTRNN the number of parameters to be evolved will be $n^2 + 4n$, where n is the number of neurons in the network, if we are evolving weights, the time constants of each neuron, neuron biases and the slope of the transfer function each neuron. If the size of the network changes, the number of parameters will also necessarily change, and if crossover is to be used as an evolutionary operator, then a section of genome from one agent is likely to mismatch the genome from other agent. Not only will a set of values evolved as biases for one agent are unlikely to serve as useful weights if spliced into another genome during the crossover process, but all values after the mismatch would be offset, thereby rendering the solution almost random. One potential solution to this problem involves an alternative encoding scheme which defines the network spatially, however here in the experiments where the network size is allowed to change, a somewhat novel approach is taken.

We simply define a maximum network size and generate the population with every individual neurocontroller encoded at this maximum network size. However, we also add a string of extra parameters to the network which serve as switches that control whether a neuron is on or off. If the value of this parameter zero or less, then the neuron is considered to be off and when the network is generated, all of the output weights of this neuron are set to 0. If the value is above zero, then the values of the output weights encoded in the genome are allowed to be used in the initialisation of the network. Because the DNA still exists, although it is not allowed to be expressed within the neurocontroller, it acts as a kind of 'junk' or non-coding DNA in that it is allowed to accumulate mutations and might well be inserted through the process of crossover into the genome of another neurocontroller in which that neuron is turned on. After some initial investigation in which the newly added neurons tended to disturb the network dynamics in a way that was nearly always deleterious, if a neuron state is changed from off to on, all of the weights are multiplied by 0.000001 in order to prevent this network destabilisation, although the neuron bias, time constant and transfer function slope are retained.

We also here allow a dynamic architecture of the network at a lower level, that of a single weights. In contrast to the previous experiments, the network is initialised with only 70% connectivity, we add an additional operator in the genetic algorithm which prunes or grows single connections in the network. Using this operator there is a 10% probability that for each new DNA string that is created a single weight is selected and its status is changed i.e. if the current weight value is zero then its value is modified to a random value in the range [-1 1], and if a nonzero weight already exists then its value is set to 0. This pruning/growth operator will also apply even if the neuron activation switch is turned off, this means if crossover inserts this gene into another population member whose neuron activation switch for that neuron is turned on, then these accumulated architectural mutations will be embodied in the encoded network.

7.3.1.3 Velocity inputs

Another aspect of the neural network implementation is whether inputs corresponding to the rate of change or velocity of the standard input set are provided to the neural network. In chapter 5, we used a velocity input describing rate of change of battery level in addition to an absolute current value input. Here we will compare the performance with and without these velocity inputs based on the final selection of inputs listed in Table 2. Because here we use a more realistic discrete sampling of the input at a slower rate than the update of the network, the values of these velocity inputs are not calculated as a simple timestep to timestep changes as that would result in 24 timesteps of zero velocity and a sudden leap of velocity when the new value is available. Simply using the most recent nonzero value to calculate velocity value would render this input prone to error from the noise added in the task simulation and the putative noise in the environment. Because of these issues we take the approach of calculating the velocity based on a half second moving window of passed input values.

7.3.1.4 Handcoded stabilisation network

The work prior to this point deliberately avoided the use of hand coded network seeds in order to avoid prejudicing evolved solutions. Here however, we introduce a very simple seed that is capable of approximating the desired behaviour of the stabilisation task albeit on a relatively small range of parameter values, and a reduced range of starting positions. The seed is simply a network with all weights set to zero except a direct connection from the azimuth value to the output neuron such that a kite in the left half of the window will be steered right and vice versa. This will essentially steer the kite away from any location at the side of the window towards a point directly downwind in the window. There is also a negative self connection on the steering neuron such that it damps its own output to a certain extent. Even though it seems to be possible to evolve networks without using a hand coded seed, contrary to some other applications ((Vaughan, 2007)), it was deemed worthwhile to investigate this avenue to assess whether a greater percentage of evolutionary runs were successful using this method and to compare the results of successful seeded and unseeded evolutionary runs.

7.3.2 Full parameter variation

In these experiments, the application of noise is widened with the addition of 10 extra parameters that are systematically altered between trials. Here we allow this noise to assume different levels, as described in Table 3. Alongside each parameter the default value is shown which is considered to be the most likely value for that parameter. The minimum amount of variation which this parameter is subject between experiments is shown alongside the maximal variation. These maximum values are selected either from prior knowledge of the hardware or because further variation would render the simulation unstable or cause it to exhibit unrealistic dynamics. These levels were validated through manual control of the simulation as the values were modified. All of the parameters marked with an asterisk were modified in the previous chapter and described there. All of the additional parameters that are varied are described below.

Parameter	Default Value	Min Variation	Max Variation
Tip Drag ratio*	1.5	0.005	0.5
Damping Factor*	0.008	0.002	0.04
Front / Rear Line difference(m)*	0.02	0.0005	0.01
Drag Coefficient Modifier*	0.05	0.005	0.05
Lift Coefficient Modifier*	0	0.005	0.05
Mass Per Kite area (kg/m^2)*	0.25	0.002	0.1
Kite Span(m)*	4.3	0.2	2.5
Aspect Ratio*	4.8	0.2	1

Depower Range(m)	0.2	0.01	0.03
Depower Max Speed(m/s)	0.2	0.8	0.03
Reel Gearing	0	0.000000001	0.000001
Constraint Damping	0.8	0.005	0.2
Actuator Range Time (s)	1.2	0.05	0.4
Max Line Adjust (m)	0.3	0.002	0.05
Sensor sample interval (s)	0.05	0.0025	0.015
Motor update interval (s)	0.05	0.0025	0.015
Sensor Delay (s)	0.05	0.005	0.025
Motor Delay (s)	0.05	0.005	0.025
Model Version	10 Particle	2 possibilities	2 possibilities
Line Length*	Variable	Variable	
Aerodynamic profile version	A	2 possibilities	2 possibilities

Table 3 The parameters that are systematically varied between trials, along with their default values and the minimum and maximum variation to which they are subject

Depower Range: In some experiments a heuristic mechanism was allowed to modify the attack angle of the kite in order to prevent spikes in line tension. This parameter changes the range of depower available to this heuristic.

Depower Max Speed: This is the second parameter associated with the depowering mechanism heuristic, it controls the speed with which a depower signal can be actuated.

Actuator Range Time: The actuator range time effectively governs the speed of the actuators such that a given signal from the neural network cannot be implemented immediately in a 'magic' manner. The actuators in the simulation will take in this number of seconds to travel between extremes of their range. The default value is taken from the RX-64 servo specification.

Reel Gearing: In some experiments, the line length is allowed to increase to represent the line being drawn out from a reel mechanism. Because in some instantiations the speed at which a reel would be turned is dependent on the tension in the line, this parameter controls the gearing relating the current line tension to the speed of increase of the line length.

Constraint Damping: This constraint damping parameter was found to increase stability of the simulation at extremes of steering input, wind speed etc. It is applied by scaling the constraint enforcement by a value less than one, i.e. not all of the mismatched distance is reset in one timestep.

As a true implementation set parameter, this value is scaled to a maximum variation point which stops just before the simulation is rendered completely unstable.

Max Line Adjust: The maximum line adjustment value limits the range of the steering to this length per actuator, therefore the maximum difference in left and right line length is two times this value.

Sensor Update Interval: The value is relatively self-explanatory denoting the time between sensor samples, which in reality is limited by the hardware analogue digital converters and the communication speed.

Motor Update Interval: The motor sample rate denotes the time between updates of the motor torque signal, which is likewise limited by communication considerations.

Sensor Delay: The sensor delay is an estimated delay corresponding to the lag from the real world quantity to its availability as a numeric value for the neural network. In simulation this delay is implemented via a software buffer, but in reality it will be caused by time for analogue digital conversion and communications.

Motor Delay: The motor delay is the time delay between when the neural network is output becomes available and that signal becoming available to the servo as a torque command.

Model Fidelity: The model fidelity switches the multibody model used for the kite simulation and can take one of two values corresponding to a six particle kite model or a 10 particle kite model.

Aerodynamic profile: This can take one of two values, which switch the Lift and Drag coefficient values between an arbitrary lift and drag profile and the results of a two-dimensional computational fluid dynamics simulation on a real leading edge inflatable kite profile.

7.3.3 Task management: Angular termination

In the work described in previous chapters, trials were only terminated prematurely when a part of the kite reached ground level. However here we wish to bias against any flight near the ground. We therefore introduced an early termination of the trial if the elevation angle dropped below a certain value, even in tasks where the looping flight was encouraged in order to maximise line tension. In the case of evolving zenith stabilisation behaviours, the elevation angle at which the trial was terminated prematurely was allowed to become higher during the course of evolution according to the scheme described in section 7.3.4.

7.3.4 Task management: Fine grained task difficulty shaping

In some evolutionary runs, there appeared to be some difficulty in achieving the required performance, especially with the burden of the extra variation of the task parameters. Therefore here we implemented a systematic releasing of the difficulty of the task that could be compared to the unfreezing of degrees of freedom during development or to some shaping schemes in which for multiple behaviours aspects of the task are chained in a systematic method (Bongard, 2008). We showed in chapter 5 that such a shaping scheme was necessary even to generate valid behaviours in a relatively simple task. The concept underlying this methodology is that for a task where there is a large amount of variation between evaluations due to changes within a simulated task environment or due to parameter 'noise' imposed by the operator to enhance the transferability of the neurocontrollers across the reality gap, the number of evaluations needed in order to achieve a balanced measure of the performance of the neurocontroller is being compared may be so high as to render the use of computationally expensive physics simulations impossible.

There may be so little in common between successive evaluations that a strategy or behaviour that is successful for one set of parameters or task initialisation may be completely inappropriate when the task is initialised with a different set of parameters in the next generation. This problem may be ameliorated but not eliminated by utilising multiple trials per evaluation. If all of the evaluations in one generation shared the same set of task initialisation parameters then it may result in the deletion from the network of what actually are reasonably valid general strategies simply because the task initialisation parameters are towards the extremes of their possible values. Equally, if the task parameters are only shared between the pairs to be evaluated, then it is conceivable that this may favour the development of very conservative networks that simply do nothing rather than develop a general solution to the task. The proposed solution is to begin evolution with a restricted range of variation such as to favour the rapid development of a coherent strategy without hindering the evolutionary process. Naturally we would expect that this solution would be fragile in some respects and would not be likely to transfer across the reality gap. A sequence of widening the variation of task parameters is therefore implemented as described below. This approach of provision to the evolutionary process of a series of small hurdles can result in as many as 50 discrete stages of task difficulty. The selection of which aspects of the task initialisation and parameterisation are subject to variation at successive stages might well be crucial. It is possible that this unwrapping process might backfire if careful thought is not given to the sequence in which the various aspects to be varied are unfrozen.

7.3.4.1 Task difficulty staging management algorithm

The mechanism by which difficulty staging is managed is directly inspired by the mechanism used by Bongard for behavioural shaping in a task with multiple sub behaviours (Bongard, 2008). There are two simple rules; a forward step and a backwards step, that relate to the performance characteristics of the evolutionary run as follows. On initiating the process we expect the performance to improve as the evolutionary run progresses, once the performance reaches a predetermined threshold, measured by the average performance of the whole population over a number of generations, the next stage of difficulty is initiated. This might also be a sub stage which is a difficulty increment within a stage, as described below. There is also provision made for steps that might be too difficult by allowing a backwards step, triggered by a performance threshold, or the failure of the population to proceed to the next level within a certain number of generations. In Bongard's work the task simply reverts to the previous trial step, presumably to allow some possibly neutral exploration of the parameter space. However, we here implement two modifications to that strategy, firstly we modify another aspect of the task, namely the elevation angle termination severity. The idea behind this is that the behaviour may be still coherently being performed but at a reduced level of accuracy that will cause the position of the kite to wander beyond the thresholds previously considered unacceptable. If this was occurring almost universally across the population, then a trap will have been set from which the evolutionary process will not be able to extricate itself, because every single trial will result in early termination and near identical scoring. By widening the acceptable margins of behaviour, without allowing crashes, the neurocontrollers' scores will once again be differentiated allowing selection of fitter solutions. The next time the performance threshold is met, it is the elevation angle severity parameter that is incremented before the normal difficulty staging process is resumed. This backward step is only permitted from stages six and higher, because it is only in stage five that the elevation angle severity is increased above its own minimum level. The second modification occurs only from Stage six whose retrograde step which is described in the section 7.3.4.2 below.

7.3.4.2 Task difficulty staging list

The evolutionary runs progress through the stages on this list and their constituent substages as fitness thresholds are met. The threshold for progression between stages used will depend on which error or fitness function is used, as described in section 7.3.5.3.

Stage one: Every evolutionary run starts at this point unless otherwise noted. This stage effectively consists of the same task set up as the early experiment as described in chapter 4. There is no systematic variation of any task parameter between evaluations or between generations other than that of the wind speed, which is varied according to the heuristics earlier described, and in the case of the zenith stabilisation task, the starting position of the kite.

Stage two: In the next stage, systematic variation of the parameters listed in section 7.3.2 is enabled at the lowest level as listed in Table 3.

Stage three: At the next stage, variation of the line length is enabled between the full range of minimum and maximum values specified for that evolutionary run. Typically this will be between 17m and 40m which was found to be sufficient to allow generalisation to longer line lengths.

Stage four: At this stage, changes in the number of particles used to simulate the kite as described in section 7.3.6 are enabled, either six or 10 particles are used in the experiments in this chapter.

Stage five: At this stage, the termination elevation angle severity described in section 7.3.3 is incrementally stepped up, i.e. the kite elevation angle at which the trial is terminated becomes progressively higher. This stage typically consists of four sub-stages.

Stage six: This is the longest and most important stage in which the deviation ranges of the trial parameters to be varied as listed in section 7.3.2, are incrementally stepped up towards their maximum levels. Rather than incrementing all 20 parameter variation ranges simultaneously, a list of all of the parameters not operating at their maximal level of variation is generated. Four of these are picked at random and incremented by a quarter of their full range. If the average of the current ranges exceeds 80% of the maximal range of each parameter's variability then all of the parameters are incremented to their maximum variation range simultaneously. The four-parameter incrementation is usually repeated approximately 15 to 25 times before this point is reached. The backward step implementation in stage six involves reversing the four most recent parameter variation range changes, and selecting another four at random.

Stage seven: At this stage, the gust generation heuristic is modified to a more difficult version but instead of always using a base windspeed of 8 m/s around which to generate variation, its base windspeed is allowed to vary between five and 15 m/s.

Stage eight: At this stage, deviations are introduced into the length of the steering lines that essentially cause a bias for the kite to steer in one or other direction, even when the output from the neural network is zero. This is designed to create tolerance for either human error in setting up the kite system or asymmetric stretching of the flying lines.

Stage nine: Here the baton-passing task initialisation described in section 7.3.5.5 is enabled. This is split into eight stages, in the first four the probability of having to face a baton passing task is stepped up in increments of 0.125, in the second four stages the task difficulty in terms of the maximum steering input and maximum duration of the pre-handover control sequence is increased by 25% of the possible range. Initial and maximum values for the duration and steering extent are [0.25 1.5] and [0.1 1] respectively whether duration is measured in seconds and the extent are measured in proportion of the entire steering range.

Stage ten: Trials are generally terminated before this point however this stage involves switching the polarity of the steering outputs, effectively reversing the steering. This is the most difficult stage which even experienced human pilots are generally incapable of adjusting to, at least in common kite systems with where the short lines do not allow time for adaptation before inappropriate steering actions effectively lock-in a crash.

Stage eleven: The final stage involves incrementally increasing the task duration, the state is open-ended and will carry on until a predetermined number of generations is reached.

7.3.5 Other implementation changes

A number of other changes were made to the implementation of the experiments in these chapters, as not all the experiments shared each of these below modifications, it will be made clear in the results section which configuration is used for each evolutionary run.

7.3.5.1 Simple trial management changes

Because it seemed that in some experiments, aspects of the timing of the onset of scoring and the total length of the trial were having some effect on the evolutionary process, in all of the experiments within this chapter, the stated trial length is a baseline around which the trial length for each set of comparative evaluations is varied by a factor of 50%, i.e. a nominal 60 second trial could result in a trial of between 30 and 90 seconds. Additionally, because of the higher level of noise used in these experiments, each evaluation of the fitness of a single agent consists of at least two separate trials using different task parameters, and where noted, four trials were used in a single fitness evaluation. Although taking the lowest score was considered as a mechanism to minimise the frequency of poor or catastrophic performances, we simply took an average here.

7.3.5.2 Trial starting position and rotation changes

In the stabilisation task described in section 7.1, the early experiments follow the previous work in which the kite position is always initialised at zenith. In the results presented here however, all of the evolutionary runs were performed with the starting position at the beginning of each trial randomised, although identical between pairs of individuals from the population that are being compared. Some thought needed to be given to the way in which this was implemented, the kite must be rotated appropriately and positioned in such a way that it is controllable at the start of the trial. To appropriately orient the wing when starting the kite at a non-zenith position, the whole set of particles is rotated around the Y axis to determine the elevation angle, and subsequently rotated a second time around the Z axis, to determine the azimuth angle.

The azimuth angle initialisation at non-zenith elevation is less trivial than in the normal case. In the case of initialisation at the zenith, the kite is simply initialised in a position directly above the tether point. In fact this is not the true zenith position at which the kite will settle, which is in fact further downwind at a point directly determined by the lift to drag ratio. The fact that the kite is initialised further forward than the natural flying area is not a problem in the case of initialisation above the tether point, the kite simply drifts downwind to settle at its true zenith, the fact that the lift vector is acting counter to gravity ensures that the kite is very unlikely to fall and that there is sufficient tension in the lines to retain control over the kite. However, in the case of initialising the kite at the side of the window, if the kite is initialised too far forward i.e. past the point determined by its glide ratio, then it will be unstable and may roll and/or drift back into the wind in a way that is unlikely to be captured effectively by our physics simulation. If the kite is initialised too far backwards past its settling point, then it will surge forwards unpredictably, but at least remain in control, whilst not ideal in an operational context, this is a situation that it would be desirable for a controller to cope with.

For those experiments in which the starting position is moved in this way, the elevation angle is set at a random value in the range [-pi/3 pi/3] radians where negative values denote starting at the left side of the wind window and positive values, the right side of the wind window. The secondary rotation around the Z axis, Rot_z to bring the kite back into the wind window is simply scaled by the first rotation Rot_y as below.

Eq. 7-1

$$Rot_z = 0.2\left(\frac{Rot_y}{\pi/2}\right)$$

7.3.5.3 Fitness and error functions

For the force maximisation/looping flight tasks the fitness function remains the same as the previous experiments, repeated below as Eq. 7-2 for convenience.

Eq. 7-2

$$F_A = {\textstyle\sum_{i=1}^{S} T_i} \Big/ t_{max}$$

For the zenith stabilisation task however, a number of fitness functions were considered. Some of these were tested informally, or were selected at random as described in 7.3.5.4, however the default fitness function used in all the zenith stabilisation experiments unless stated otherwise is the final function given in Eq. 7-8.

To lower the computational and memory demand of using these fitness functions, a sample value was taken at a frequency of 10 times per second. In all of these experiments a lower score was considered to be a better score than a higher one so the term error function might be more appropriate. In all of these experiments the scoring began approximately four seconds after the start of the trial.

One simple error function for the stabilisation task involved minimising the average angular distance from the kites elevation position φ to the target point t_{φ}, in this case maximum elevation, over S samples taken during the trial.

Eq. 7-3

$$E_A = \left(\sum_{i=1}^{S} \varphi_i - t_{\varphi}\right) \Big/ S$$

A slightly more complex version of this error function takes the above measure and multiplies it by the total distance travelled in the task, where x is the position in Cartesian coordinates. This extra measure was put in place to minimise rapid oscillations around the target position which would tend to overheat the servos and consume the battery.

Eq. 7-4

$$E_B = E_A \ . \ \sum_{i=1}^{S} \|x_i - x_{i-1}\|$$

The simplest error function of all, was simply a measure of the time taken for a crash to occur, where the crash sample index is given by c.

Eq. 7-5

$$E_C = \begin{cases} 0, & c = S, \\ 1 - (c/S), & c < S, \end{cases}$$

In the fourth error function, both the azimuth and elevation angles have a target, corresponding to the glide ratio determined zenith point. The average deviation of these is then multiplied by the flight time maximising function used above, with a modifier of 0.1 such that a perfect temporal score does not destroy differentiation in the spatial score.

Eq. 7-6

$$E_D = (E_C + 0.1)\frac{\left(\sum_{i=1}^{S} \theta_i - t_{\theta} + \sum_{i=1}^{S} \varphi_i - t_{\varphi}\right)}{S}$$

The fifth error function is a development of the previous one, with the addition of a term designed to minimise the total angular movement.

Eq. 7-7

$$E_E = E_D\left(\sum_{i=1}^{S}\|\theta_i - \theta_{i-1}\| + \sum_{i=1}^{S}\|\varphi_i - \varphi_{i-1}\|\right)$$

The final error function neglects an azimuth target but retains the elevation target and temporal term, adding a term to minimise speed of change in the rotation angle estimate γ as estimated by a putative camera system which would correspond to yaw in a kite centred frame of reference, where t is the time interval between samples. *This is the default scoring function used.*

Eq. 7-8

$$E_F = (E_C + 0.1)\left(\sum_{i=1}^{S}\frac{\|\gamma_i - \gamma_{i-1}\|}{t}\right)$$

As described in 7.3.4.1, each of the fitness functions needs to have its own threshold to apply to progression through the multi-stage shaping process. The thresholds were determined by recording a human pilot achieving adequate performance on the zenith stabilisation task as scored by each function. The below table summarises the rationale for each error function.

Error function	Function basis	Function use
E_A	Elevation deviation from target.	Shuffled evaluation function
E_B	Elevation deviation from target multiplied by distance travelled.	Shuffled evaluation function
E_C	Time to crash	Shuffled evaluation function
E_D	Time to crash multiplied by sum of elevation and azimuth distances to target.	Shuffled evaluation function
E_E	Time to crash multiplied by sum of elevation and azimuth distances travelled	Shuffled evaluation function
E_F	Time to crash multiplied by average yaw speed	All single evaluation experiments and Shuffled evaluation function

7.3.5.4 Shuffling error functions

Each of the above error functions are designed to select for kite stabilisation behaviour using different measures, however in evolutionary robotics it is often possible for unforeseen and possibly undesirable traits to be selected for inadvertently, simply because the fitness function allows good scores in inappropriate situations that have not been envisaged by the designer of the fitness function. One way around this problem might be to use a weighted sum of multiple different fitness functions. Alternatively and in this case we assess the viability of 'shuffling' fitness functions i.e. in each generation selecting one fitness function at random from a predetermined repertoire, thereby favouring the selection of neurocontrollers that are able to score well at each fitness function on its own. The intention here is that 'cheating' neurocontrollers that exploit some aspect of a single fitness function, will be selected against because this strategy will not be applicable across the whole range of fitness functions. At the same time, the various aspects are trying to be encouraged by different fitness functions will be represented by the best evolved neurocontrollers.

7.3.5.5 Baton-passing

It was suggested earlier that kites at the edge of the window can be less stable than those flown at zenith, particularly if the kite is static and therefore exposed to the full effect of changing wind speed and direction. This was a key motivation for exposing controllers to this scenario during the evolutionary process. In a similar vein, for reasons of practicality, it would be preferable that for each real-world experiment some flexibility would be allowed in the passing over of the control of the kite from a human operator to the neurocontroller. Furthermore, it would be interesting to see if neurocontrollers were capable of rescuing the kite from completely inappropriate control actions from the point of view of stabilisation, for example if the kite has been steered such that it is heading towards the ground. This property would indeed be essential if control authority were to be alternated between the looping flight and stability control neurocontrollers in a real-world generation context. For these reasons we implemented a baton-passing heuristic. In this scenario, a control sequence is generated at random and the simulation is started with this control sequence in place for a number of seconds after which control is transferred to the neural network. The control sequences are generated through a simple heuristic scheme described by the below pseudocode:

```
for each trial that constitutes one evaluation
    Probabilistically decide whether a handover is used
    for those trials where a handover is used
        Randomly assign a maximum steering force in the
        range determined by current difficulty stage
        Randomly assign a control sequence duration from
        the range determined by difficulty stage
```

```
                    Generate oscillatory steering movement using sin
                    function with no. cycles in the range [1 4]
                end
            end
```

Examples of the resulting control sequences are shown in Figure 7.1 The graph shows the deviation in the length of one steering line in a set of possible pre-handover steering sequences using one duration and one strength value. Note that whilst all of the sequences return to a zero value corresponding to equal line lengths in the steering lines, this by no means indicates that the kite will have returned to the initial position. In fact in some cases, this steering action will have caused the kite to crash or to be heading towards the ground so fast that it will be impossible for the neurocontroller to recover. This additional task difficulty level is activated at stage 9 in the list given in section 7.3.4.2.

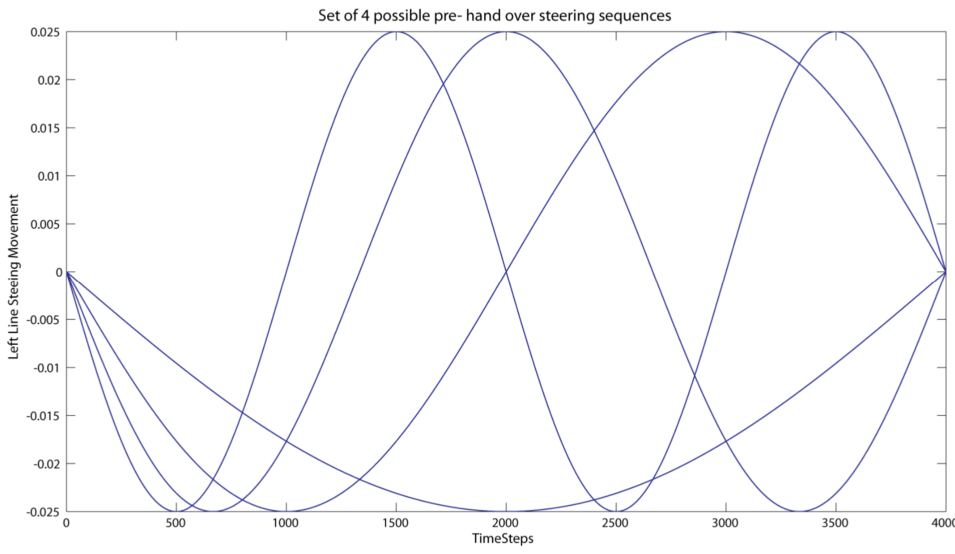


Figure 7.1 The graph shows the deviation in the length of one steering line in a set of possible pre-handover steering sequences using one duration and one strength value. The length of the other steering line mirrors these values

7.3.6 What model complexity is most appropriate?

Clearly the unusually involved shaping scheme is necessary because the process of application of systematic noise to ensure transferability from the simulation environment to the real world is going to make the evolutionary process more difficult. This is because evolved neurocontrollers need to be robust not to just changes within the task environment that will exist in the real world but also to artificially imposed variability that is at least partially motivated by the fact that the physical model is not accurate. It has already been acknowledged that the level of fidelity of the simulation of the kite is somewhat arbitrary, we were seeking to capture some key characteristics of the kite flight, but this was selected without a rigorous process of evaluation. The computational demands of the modelling process were only taken into account as far as selecting a model that was just about affordable in terms of allowing the evolution of neurocontrollers within a reasonable amount of time i.e. a few days.

The assumption is that there is a continuum of model fidelity, a perfect model would clearly require no variation other than that to be expected in the natural environment, however this model would be extraordinarily computationally intensive, and for the kite problem is currently out of reach. At the other end of the spectrum is a completely minimal model that neglects nearly all of the physics and that incorporate as much of the users knowledge of the problem is possible in terms of what output or behaviour is desirable with different input regimes or scenarios. This 'minimal' model will need to be wrapped in a deep envelope of noise if we expect transferability of the behaviour intact from the minimal model to the real world. It might be that not only are we denying the neurocontrollers evolutionary experience of physical effects that may render the problem easier, but that the amount of noise that is necessary might make the fitness landscape so unreliable that evolutionary progress stalls.

Whilst an in-depth investigation of this issue would require more time than was available here, an opportunity present itself to touch upon the key issues.

The idea is simply that if we use a hierarchy of models, we reduce the possibility that evolved neurocontrollers are exploiting some persistent artefact present in a given model. It would also be possible to check whether behaviours evolved on simpler models of the same problem transfer successfully to a more complex model(s) that is intended be more a more accurate description of the real system. Were time to permit, it would be also possible to assess whether the amount of parameter variation necessary could be lowered in the case of a neurocontroller evolved on a medium complexity model being transferred to the high complexity model relative to a neurocontroller evolved on a low complexity model being transferred to a high complexity model. The below graphs show a model that is simpler (Figure 7.2) than the one used for the prior work, a more complex model with 14 particles and 8 panels (Figure 7.3), and a final 22 particle, 12 panel model that is far more complex (Figure 7.4) and whose geometry is taken directly from the kite that is to be controlled. The first of these is to be used during the evolutionary process, the 8 panel model is used in a final validation step and the last model could be used as a supplementary validation step in order to confirm that the controller is relatively independent to the model that is being used. With the latter model, we restrict ourselves to highlighting the difficulties of using a comparatively complex wing model and present a simple evolutionary model validation method in section 7.3.6.1 in which the model parameters are tuned using real world deformation measurements.

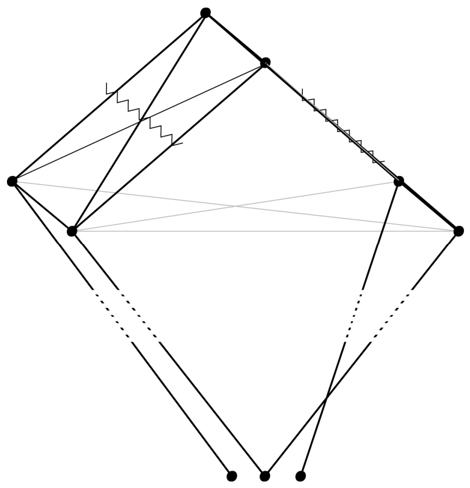


Figure 7.2 The simpler 6 particle model with only 3 slices used for the aerodynamic calculations

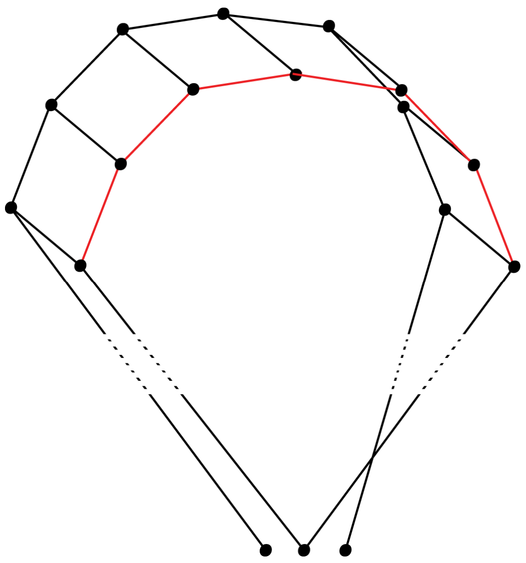


Figure 7.3 The more complex 14 particle model with 8 slices used for the aerodynamic calculations

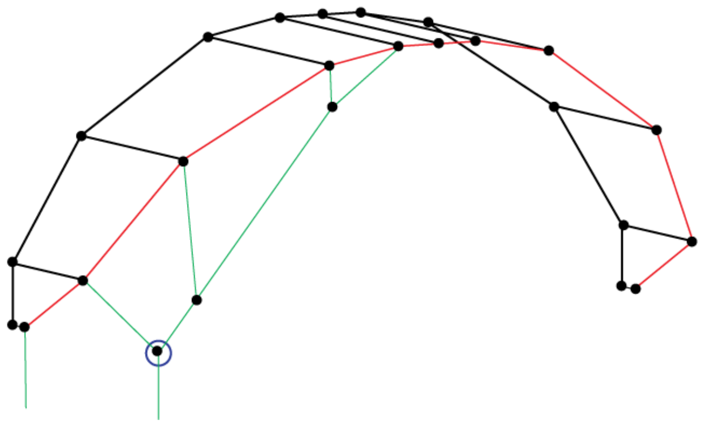


Figure 7.4 The most complex 22 particle kite canopy, Leading edge is shown in red and the trailing edge in black. One of two, 3-particle bridles is shown in green, the other is omitted for clarity. The blue circle around one of the bridle particles is the location of a block, or sheave, on the real kite.

7.3.6.1 Using automated model tuning techniques to derive the most accurate model available with limited computational resources

The creation of the simple model shown in Figure 7.2 was relatively trivial, simply using the same program that generated the normal model but with a smaller number of particles. It was easy to hand design a set of constraints that would allow the shape of the kite to be retained in flight in a manner that produced reasonably realistic flight behaviours, when the model was used in the aerodynamics simulation, although it was notable that for a given size of kite, the simple model produced a much more responsive kite than the normal one. In the case of the more complex models, the greater number of particles meant that it was not a trivial matter to design a set of constraints that would adequately maintain the shape of the kite whilst allowing a degree of deformation. In fact, using the most complex model from Figure 7.4 with a hand designed set of constraints resulted in unrealistic deformation and flight behaviour in the simulation environment. Even the slightly more complex 8 panel model shown in Figure 7.3 exhibited some resonant deformation or 'jellyfishing' when tested with manual control.

To overcome this obstacle for the most complex model, we decided to acquire some information on the real kite system with which to assess the performance of various sets of constraints. To achieve this, the kite was mounted such that half of the kite's position was completely constrained by a framework and half of the kite was free floating, supported purely by the kite's own structural integrity as shown in Figure 7.5. To acquire the dataset, a number of measurements were taken by loading the free half of the kite with a known mass at locations corresponding to the particle positions, and manually measuring the deformation. This method proved somewhat inaccurate and very time consuming however a limited set of data was acquired. We did develop an improved method which used an inertial measurement unit mounted on the kite at locations corresponding to the particle positions, and automating the process of manipulating the kite using a large gantry robot (See Figure 7.5).

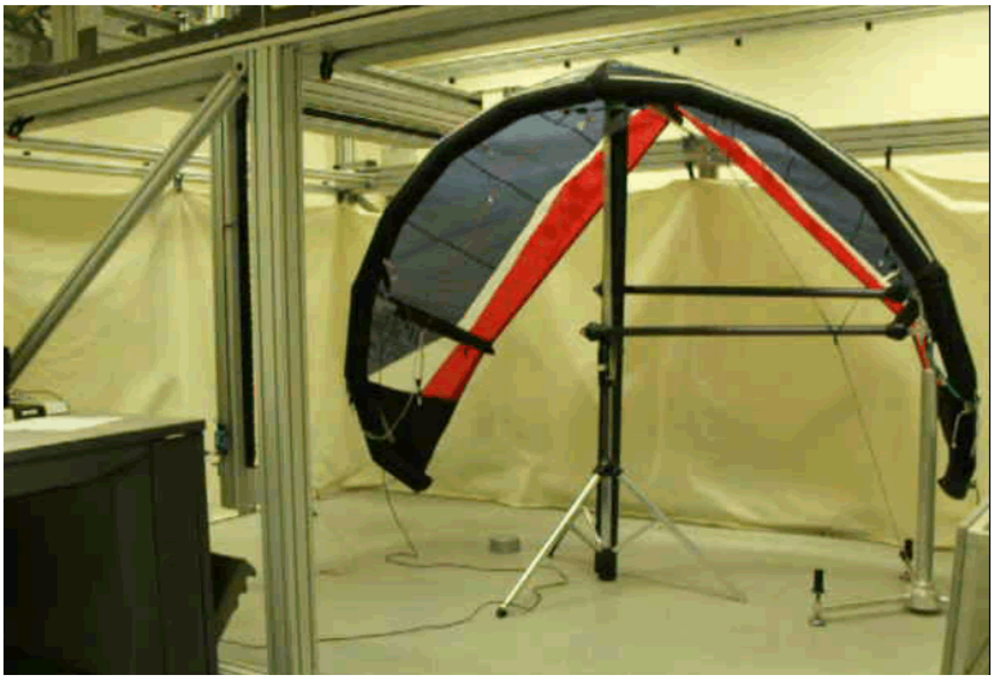


Figure 7.5 The test kite mounted in the frame in the gantry robot.

A string of known length was tied to the wing tip of the kite, and the gantry robot is used to perform a series of manipulations, pulling the kite by relocating the head of the gantry robot to certain pre-defined positions within the robot arena. The rotation of the IMU is measured at each position in the sequence. Because of the accuracy of the gantry robot movement, it was possible to repeat this sequence of movements with the IMU mounted at different points on the kite thereby building up a reasonable set of data. Unfortunately a hardware failure prevented the collection of the full data set so the hand measured data set was used for evolution instead. The same genetic algorithm used for the experiments was then used to generate a set of constraints that would match these deformations under known manipulations. In this preliminary work, the GA only had to evolve a 19 parameter binary string which controlled the activation of the constraint generation rules.

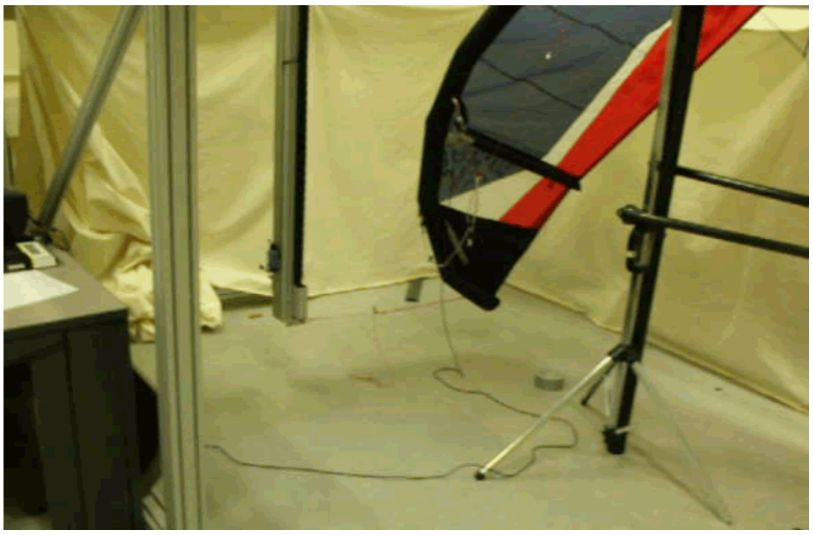


Figure 7.6 A photo showing the attachment from the gantry robot head to the wing tip

7.4 Results

7.4.1 Control of looping flight whilst reeling out

The first set of results shows the application of the standard evolutionary process without the wide variation range and difficulty stage management, but with the adaptive network size. These experiments were performed with velocity inputs as well as instantaneous values but lacking an estimate for orientation because they were performed before the visual tracking algorithm described in chapter 8 was written. The results therefore are for 8 inputs, maximum 21 neuron CTRNN, tanh transfer function, evolvable k parameter, non-spatial, with noise on the neuron biases in the range [-0.05 0.05]. In these experiments, parameter noise at a moderate range was implemented immediately at the start of evolution.

The goal was to ascertain whether a CTRNN configured using a combination of the most promising configuration details identified in the previous chapter could be evolved to control a kite to fly in appropriate flight pattern if the line length were subject to change over the course of the trial, as if the kite were pulling its line off its drum as per a real world electricity generation application.

The first notable characteristic was the poor yield of evolutionary runs. Of 20 separate runs, only three moved from a baseline of a behaviour that effectively constituted immediate crashing. Of these three, only two displayed a relatively consistent behaviour that could be considered appropriate to the requirements of the task. The results of the best performing individual from those two evolutionary runs are shown below.

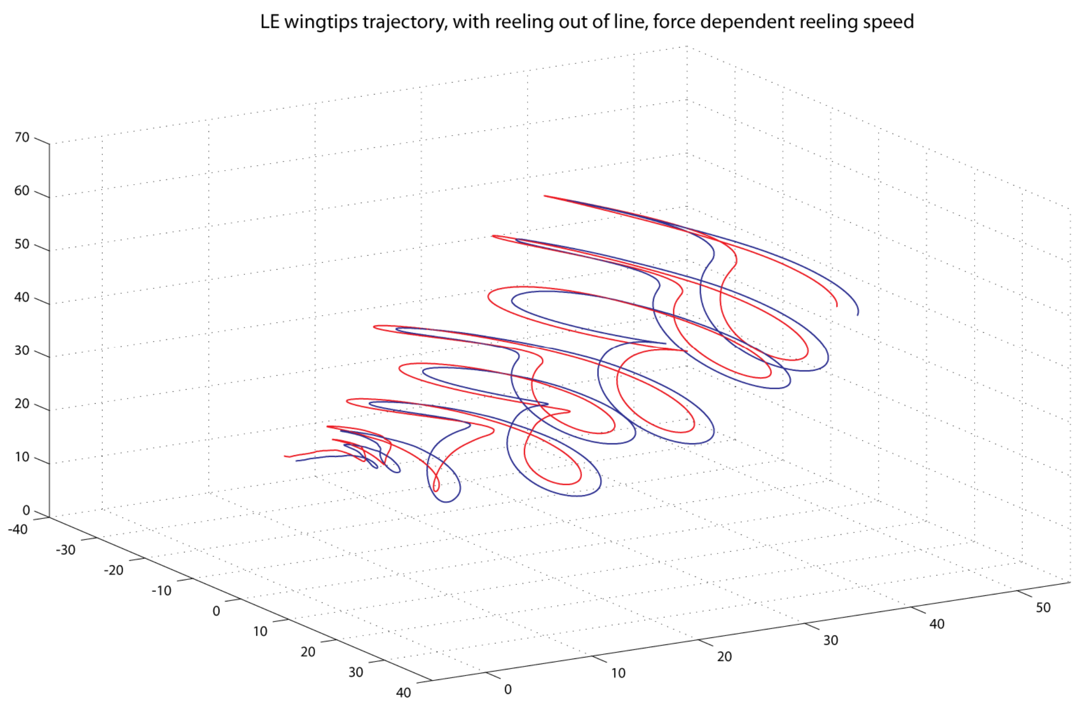


Figure 7.7 This plot is a view diagonally from the front and to the side showing the flight trajectory of a kite controlled by the best neurocontroller (Network A), from the best performing run from 20 runs over 60 seconds, when the line is being reeled out,with reelout speed proportional to the instantaneous aerodynamic force

These results were obtained after 500 generations of evolution. The best and average performance per generation over the evolutionary period for the two best evolutionary runs is shown below. These plots demonstrate that the performance improvement in the task had not plateaued after 500 generations which is probably to be expected given that this did not occur until after 800 generations in the simpler version of the task used in Chapter 4.

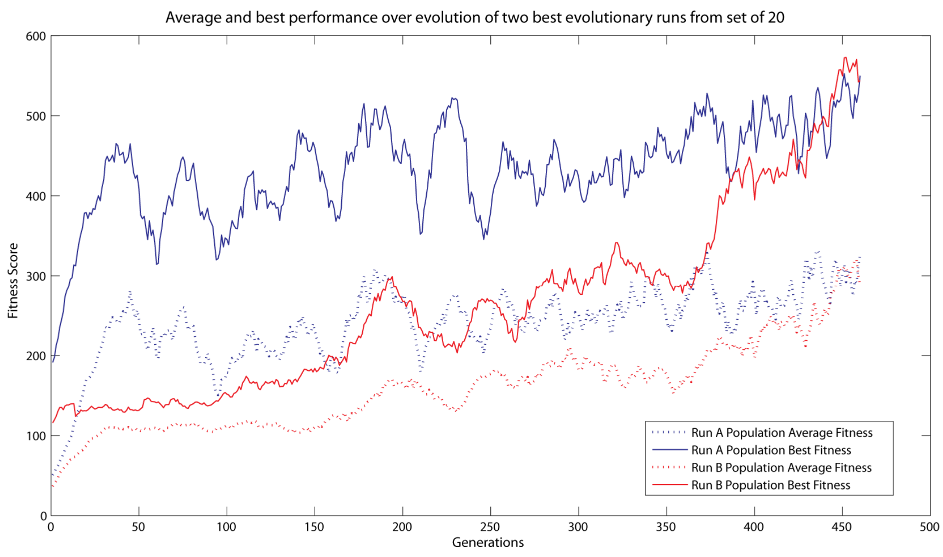


Figure 7.8 Population average and best performance over the two best evolutionary runs out of a group of 20, showing steady improvement in Run A and more dynamic, punctuated changes in Run B.

Both networks are much more capable of controlling the kite of the kite without crashing than those described in Chapter 4. We took the best performing individual from runs A and B, termed network A and B respectively and tested them on a number of randomly generated wind conditions, with a number if initial conditions to test for stability in terms of probability of crashing. Network A suffered 0 crashes during 100 simulation runs, starting at 5 line lengths equally spaced across the range [24 26] meters , with 20 different wind conditions. Network B suffered 8 crashes with the same evaluation. That an appropriate flight path could generally be maintained in variable conditions before the plateauing of evolution is encouraging, especially given that these networks did not employ the orientation estimate, which simplifies the problem by giving a direction heading for the kite.

To evaluate the performance further and assess the generalisation performance to higher altitudes, we ran network B for a 150 second trial, over double the length of the maximum trial length of 63 seconds possible in evolution. Interestingly, after about 70 seconds the figure of eight flight pattern was effectively abandoned by the neurocontroller which appeared to default to a self-extinguishing oscillation leaving the kite at zenith position.

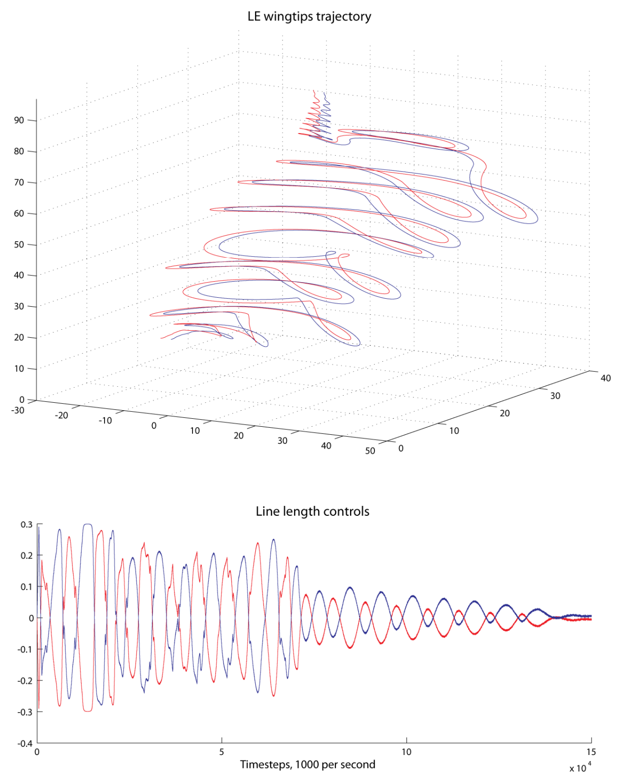


Figure 7.9 A trial using network A, but for 150 seconds than the 63 second maximum evolution period. Upper plot shows the flight trajectory, Lower plot shows the steering commands generated by the steering neuron

Network B also displayed a similar characteristic in an extended test (see Figure 7.10), at approximately 70m of altitude, the flight pattern changes somewhat and after a few more sweeping manoeuvres the oscillations once again are extinguished, albeit slightly more progressively. It was not clear at this point whether it was the task duration exceeding that used in evolution that both of these networks were failing to generalise to, or some other factor. A test was conducted therefore, using the initial Network A for a yet longer duration of 300 seconds, but with a reduced rate of reeling speed. The resulting flight trajectory shown in Figure 7.11 clearly shows that the network carries on performing the looping trajectory for the whole task, indicating that task duration was not the determining factor.

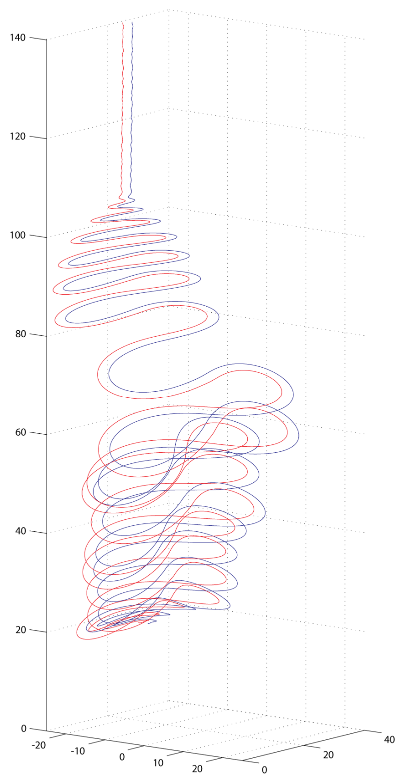


Figure 7.10 Network B flight trajectory over 300 seconds, also display a lack of dynamic flight behaviour at high altitudes

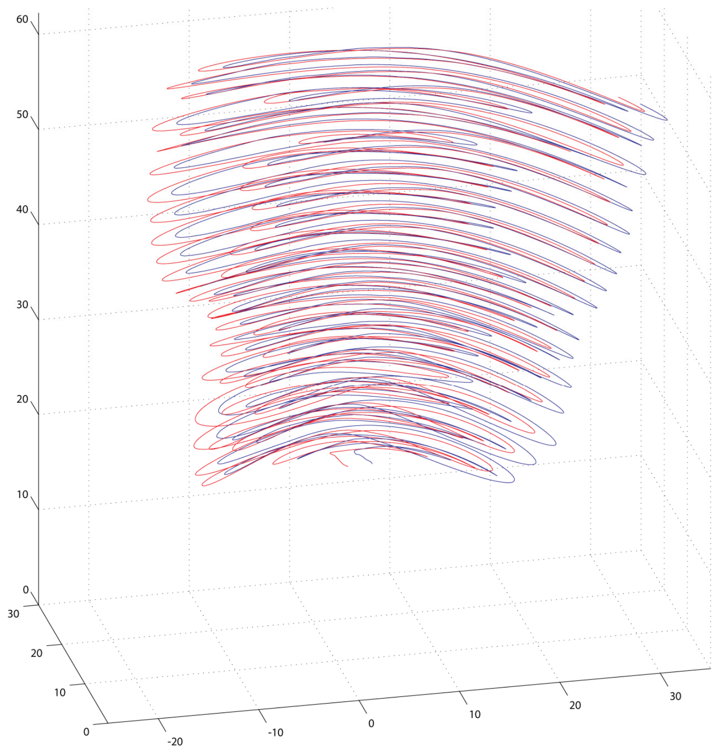
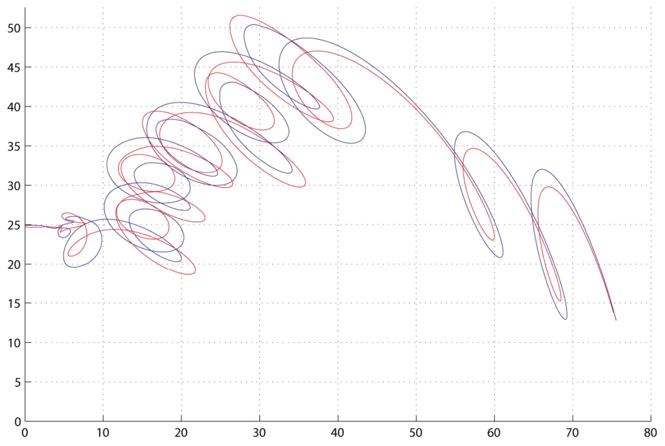


Figure 7.11 Network A flight trajectory after 5 minutes of flight height with 80% slower reel-out speed

Although it was not the intention to specifically analyse the function of the network, which is comparatively large by the standards of CTRNNs that have been subject to dynamical systems analysis, this quirk of behaviour did seem to warrant some further investigation so three further trials were conducted. After duration, the next most striking correlation was that of altitude, namely that both the normal duration trials with fast moving out and the long trial with slow moving out did not exceed 60 m of altitude. As line length is an input to the neural network it could be that simply because the networks were not exposed to input values exceeding certain quantity corresponding to 60 metres of line length during evolution, their dynamics are disturbed when this value is exceeded. In the next test we attempted to assess the importance of this input parameter to the network by freezing it such that the input corresponding to line length would not change once the line length exceeded 50 m. As Figure 7.12 shows in a side-on profile view, at around 45 m of altitude the normal pattern of the looping becomes disturbed and the kite is rapidly steered in a corkscrew looping pattern after which it crosses the angular threshold at which the task is terminated. The line length control plots in the lower portion of the figure shows that at around 5000 timesteps, corresponding to 50 seconds into the trial, where the right steering lines would normally become shorter than the left steering line, it in fact stays longer resulting in a counter clockwise spiral. This seems to confirm that the line length input is crucial to the adaptation of the network as it is reeled out.

Having apparent confirmation of the importance of the line length input, two more trials attempted to probe further. In the first the line length input value was scaled by a factor of 2.

Interestingly a similar effect as observed in the first trial can be seen in figure where the oscillatory flight transitions to a near static position however in this case it starts at a somewhat lower altitude of approximately 45 metres in contrast to the 60 m seen in the normal case, which although lower is not half the altitude as might be expected for a simple mapping. The complexity of the internal dynamics is further evidenced by the last trial in which the line length input value was scaled by a factor of 0.5. In this case, one might expect that the oscillatory behaviour might continue to a higher altitude but in fact it barely begins at all and very rapidly reverts to the static holding of the kite at the zenith as seen in the other experiments.



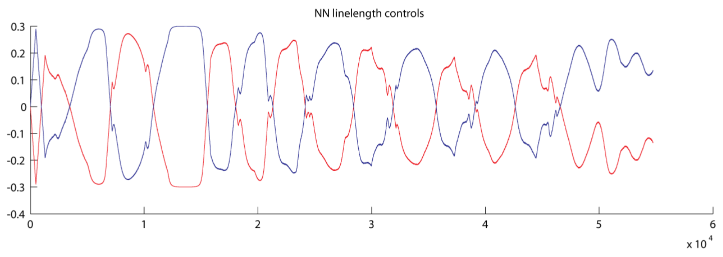


Figure 7.12 Upper plot shows the flight trajectory as seen from the side when line length input has a ceiling of 50m regardless of true value. Lower plot shows neural network steering output

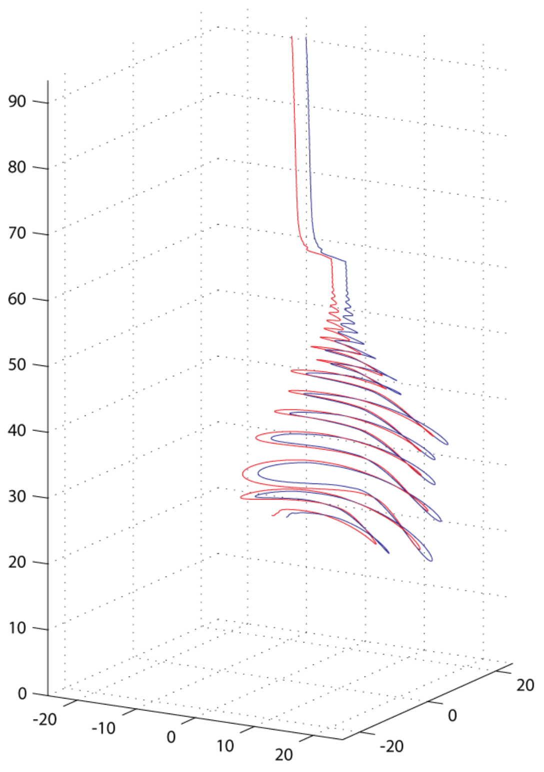


Figure 7.13 A plot of the flight trajectory generated by network A when the line length input gain is doubled

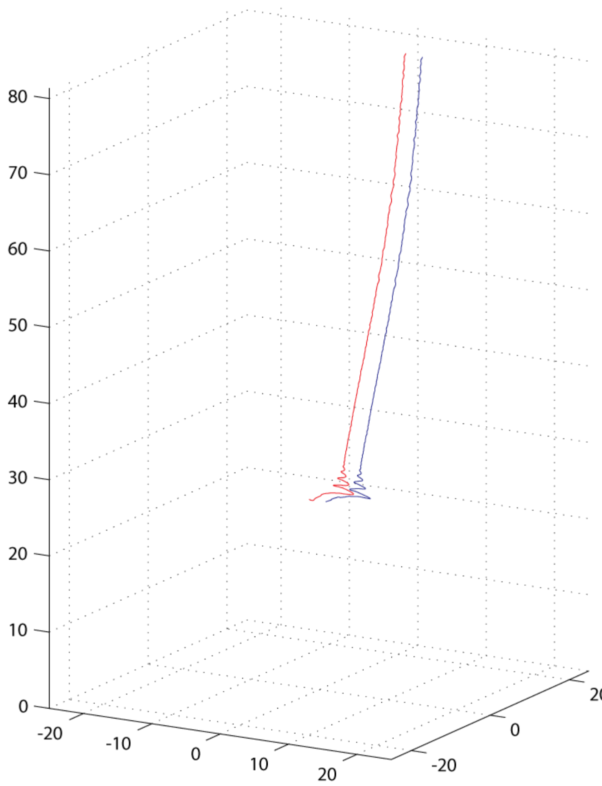


Figure 7.14 A plot of the flight trajectory generated by network A when the line length gain is divided by two

7.4.2 Stabilisation of the kite at the zenith position

The following sequence of results is concerned with the stabilisation at zenith behaviour. As described earlier, these trials start with the kite at some valid position within the wind window and the controller must subsequently navigate the kite to the zenith position and maintain it at that point. In all of the plots the kite is started at either the left or right of the wind window.

7.4.2.1 Handcoded neurocontroller performance

Figure 7.15 shows the performance of the hand coded seed at this task including the steering signal from the neurocontroller in the lower half of the plot. This shows a perfectly adequate performance of the neurocontroller with which the population of many of the subsequent trials was seeded.

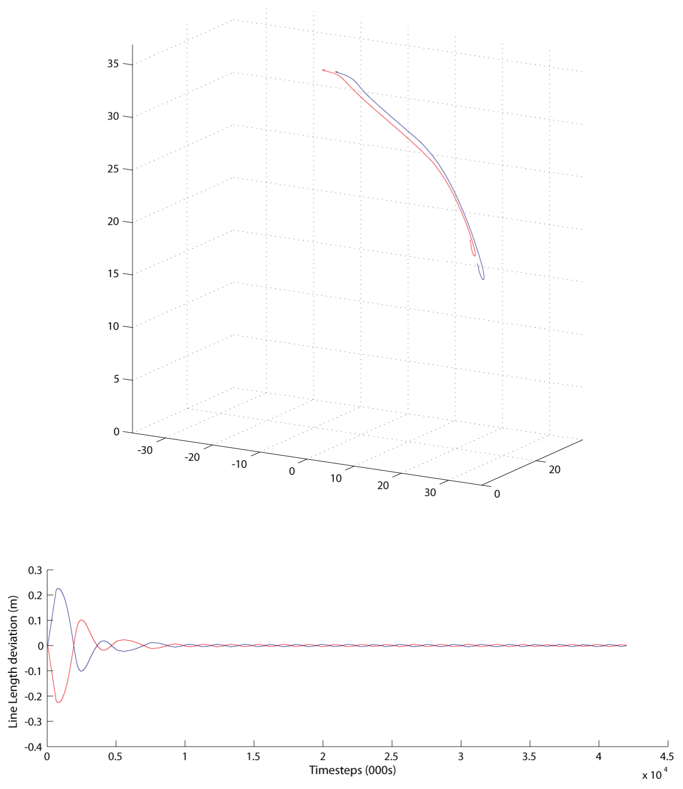


Figure 7.15 Flight trajectory (upper) and steering commands (lower) using the hand-coded seed, test with 37m static line length, starting at position rotated 60 degrees clockwise from zenith

In that case, with a medium line length of 37m, the kite is steered up from its position at the side of the window with minimal overshoot. With shorter 25m lines however, the hand coded neurocontroller fares much less well (see Figure 7.16), initially overshooting the zenith point by some margin and then oscillating at progressively reduced amplitude. These oscillations are visible in the lower half of the plot which shows that they are not quite extinguished even by the end of the trial.

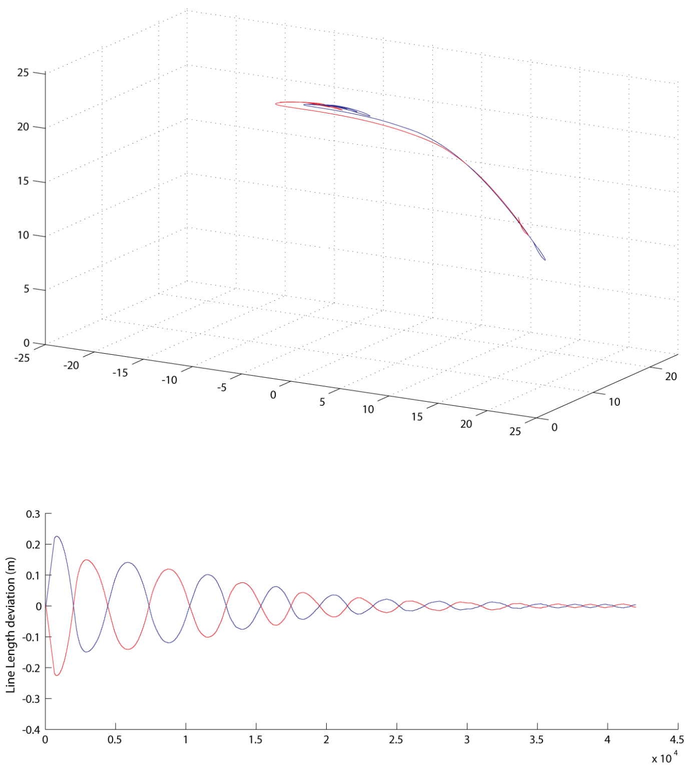


Figure 7.16 Flight trajectory (upper) and steering commands (lower) using the hand-coded seed, 25m static line length, starting at position rotated 60 degrees clockwise from zenith

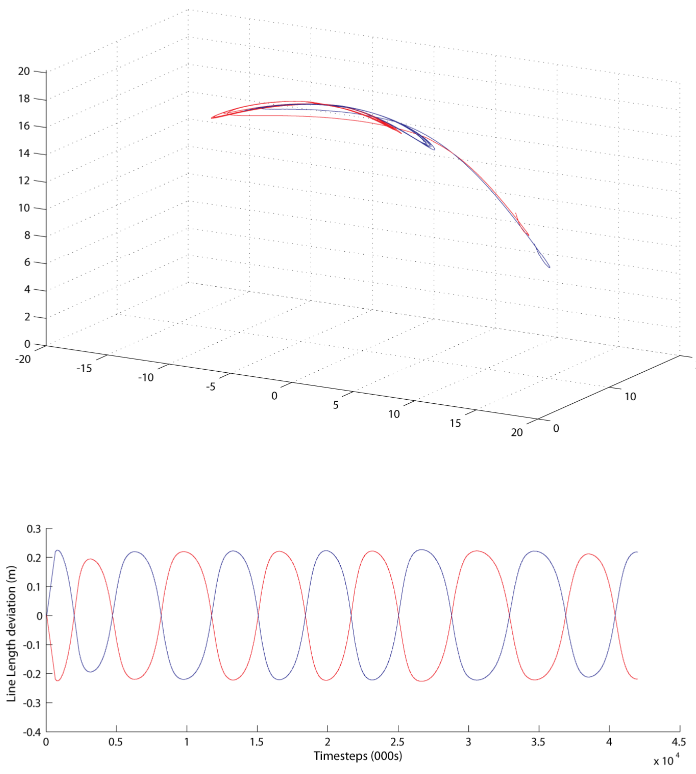


Figure 7.17 Flight trajectory (upper) and steering commands (lower) using the hand-coded seed, test with 20m static line length, starting at position rotated 60 degrees clockwise from zenith

Further reducing the line length leads to an even more severe overshoot and oscillations whose amplitude is sustained as shown in Figure 7.17.

The prior plots illustrate that the controller appears to be unable to anticipate the movement of the kite in situations where it is highly responsive to steering input. This is well illustrated in Figure 7.18 in which the wind speed is increased and the oscillations lead to a rapid failure by crashing the kite. The linear appearance of the steering output is due to the motor commands causing the servos to be operated at their maximum speed. As a group these plots demonstrate that the handcoded neurocontroller produces approximately the correct gross behaviour, i.e. if the kite is in the left half of the wind window, it steers right. However this network is not able to generalise to different environmental conditions or physical configurations of the kite system.

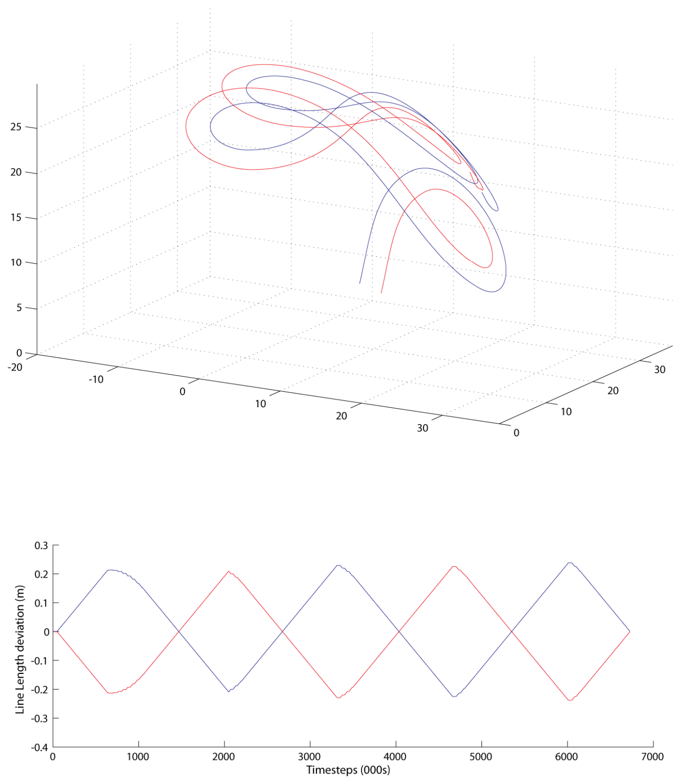


Figure 7.18 Flight trajectory (upper) and steering commands (lower) using the hand-coded seed, test with 25m static line length, starting at position rotated 60 degrees clockwise from zenith. Wind speed is 40% higher than prior trials.

7.4.2.2 Evolved neurocontroller performance on stabilisation task

In all of the experiments the full shaping scheme described in section 7.3.4 is deployed. The first four graphs show an evolved neurocontroller performing the stabilisation task in the equivalent situations as the prior four graphs. Unsurprisingly, in our default case, with a steady 8m/s wind, the evolved neurocontroller is easily able to replicate the performance seen with the handcoded seed neurocontroller. However, the controller after evolution is also capable of flying the kite to the zenith position and maintaining it there without significant oscillation, when the line length is reduced to 20 and 25 metres as shown in Figure 7.20 and Figure 7.21 respectively. Figure 7.22 shows that the controller is able to perform the task successfully with 40% higher windspeed and relatively short lines, in which case the kite becomes very sensitive to input.

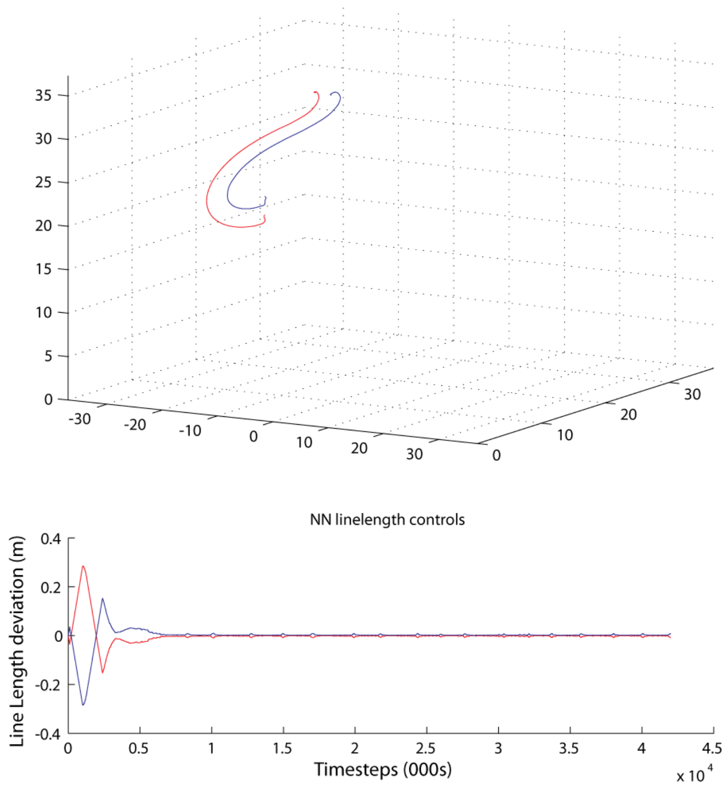


Figure 7.19 Evolved neurocontroller in stabilisation behaviour with 37m lines.

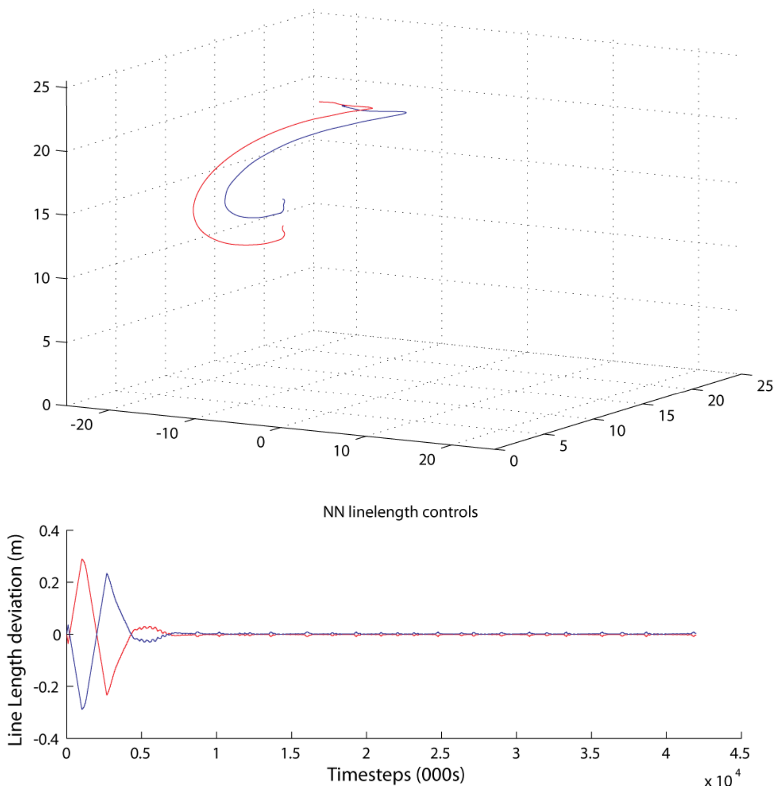


Figure 7.20 Evolved neurocontroller in stabilisation behaviour with 25m lines.

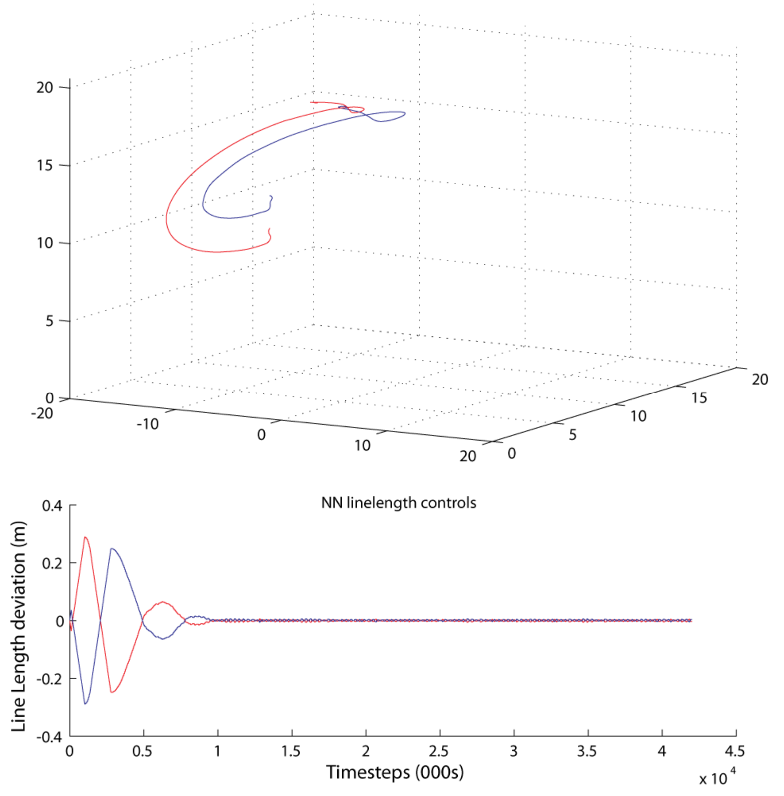


Figure 7.21 Evolved neurocontroller in stabilisation behaviour with 20m lines.

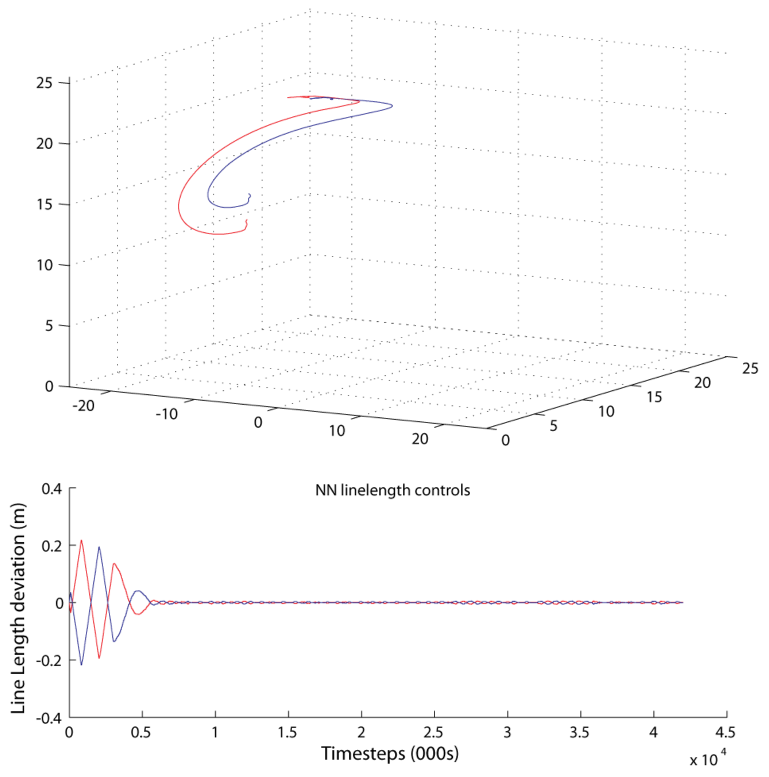


Figure 7.22 Evolved neurocontroller in stabilisation behaviour with 25m lines, and 40% increased windspeed, (11.2m/s).

7.4.2.3 Evolved neurocontroller performance on stabilisation task

As discussed at length, we exposed the neurocontrollers to a wide range of parameter variation during the evolutionary process. The intention of this variation was to render the neurocontrollers incapable of exploiting particular implementation aspects of the model. Figure 7.23 shows the neurocontroller performing the zenith stabilisation task when using 8 panel model described in 7.3.6. The controller is able to successfully complete the task even though this model was not used during the evolutionary process. This success is heartening given that the model is slightly unstable and prone to a 'jellyfishing' motion. This results from the comparatively lower number of constraints on the eighth panel kite as opposed to the six and more panel kites. The upper half of Figure 7.23 shows small oscillations in the location of the wing tips and the lower half shows the comparatively longer duration and steering signal for required because more of the actuated line length difference is being absorbed by the deformation of the kite rather than its rotation. There was also a longer period of dampening required by the controller to maintain the kite at the zenith position once it has been flown there.

In addition to making the controller as independent of the model used as possible, the variation process was also intended to make the controller robust to changes in the kite geometry, and aspects of the control hardware initialisation and performance. Figure 7.24 and Figure 7.25 show the behaviour of the neurocontroller when the control line lengths have been offset by half of the steering range, i.e. 15cm per side pulling left and right respectively. In each case the controller converges on an output value exactly offsetting this bias in order to stabilise the kite at zenith. The controller is also robust to extreme changes of the kite's aspect ratio as shown in Figure 7.26, which shows the controller performing the task when the aspect ratio is halved or doubled. The final change in kite geometry was to change the kite area between 1.8 and 8m which makes the kite much more or less responsive to a given adjustment length of steering line. This can be seen in the much greater response of the kite to a much shorter duration steering input in the left vs right pair of plots. Naturally, none of these changes are signalled explicitly to the neurocontroller, it is for the neurocontroller to adapt to the dynamics of the system.

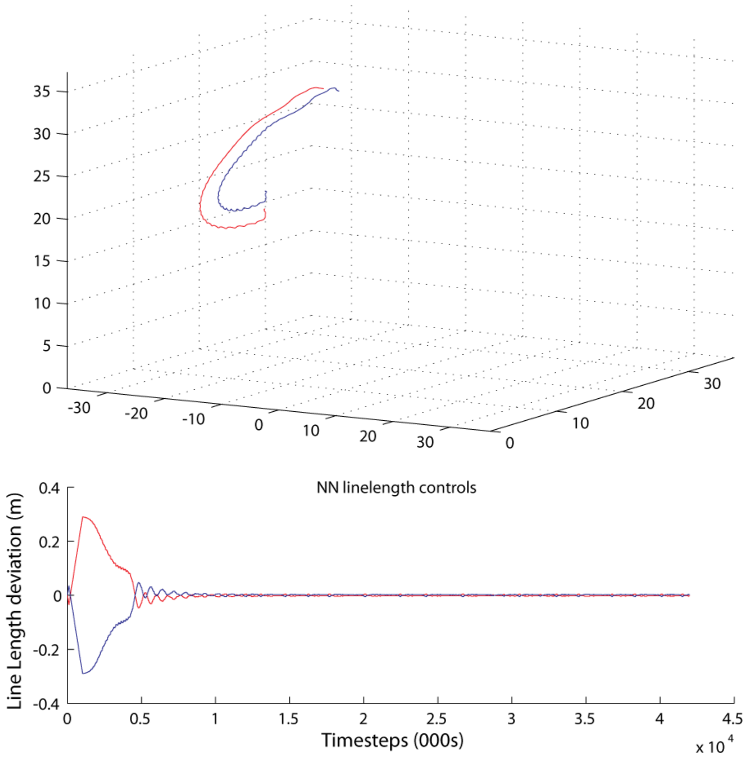


Figure 7.23 Neurocontroller performing stabilisation task with 14 particle kite model not utilised during the evolutionary process.

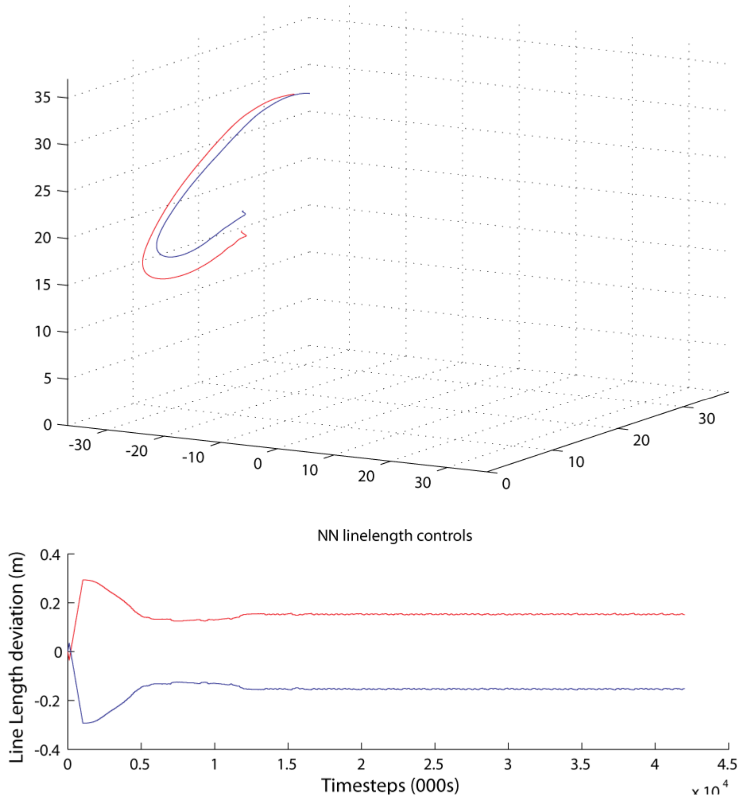


Figure 7.24 Neurocontroller performing stabilisation task with the left steering line length shortened by 15cm and the right increased by 15cm, a bias of half the steering range.

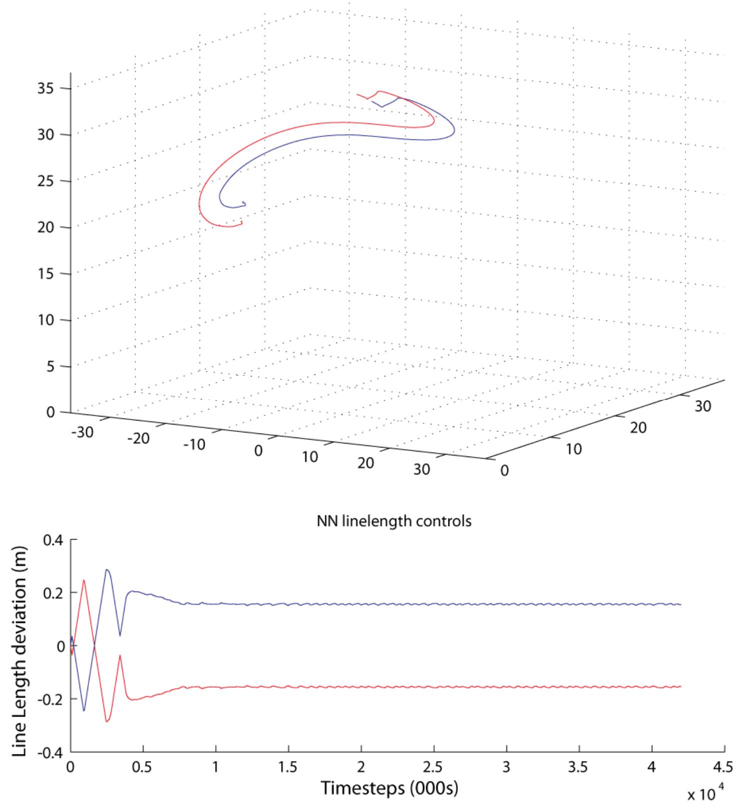


Figure 7.25 Neurocontroller performing stabilisation task with the right steering line artificially shortened by 15cm and the left lengthened by 15cm, a bias of half the steering range, in the opposite direction to the previous figure.

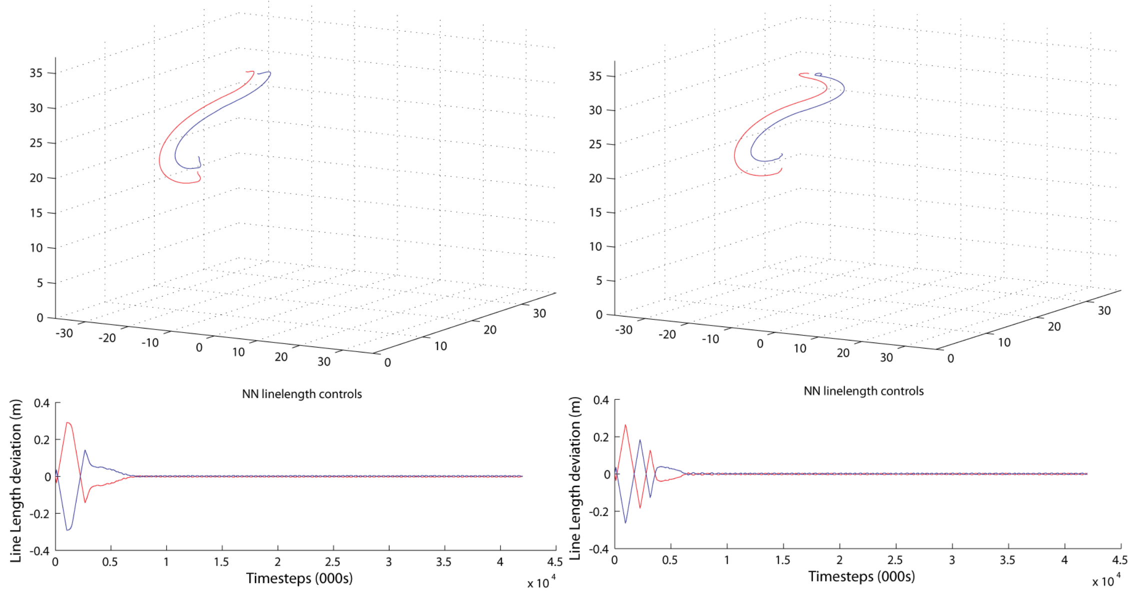


Figure 7.26 Neurocontroller performing stabilisation task with kites of aspect ratio 2.5 and 10, left and right respectively

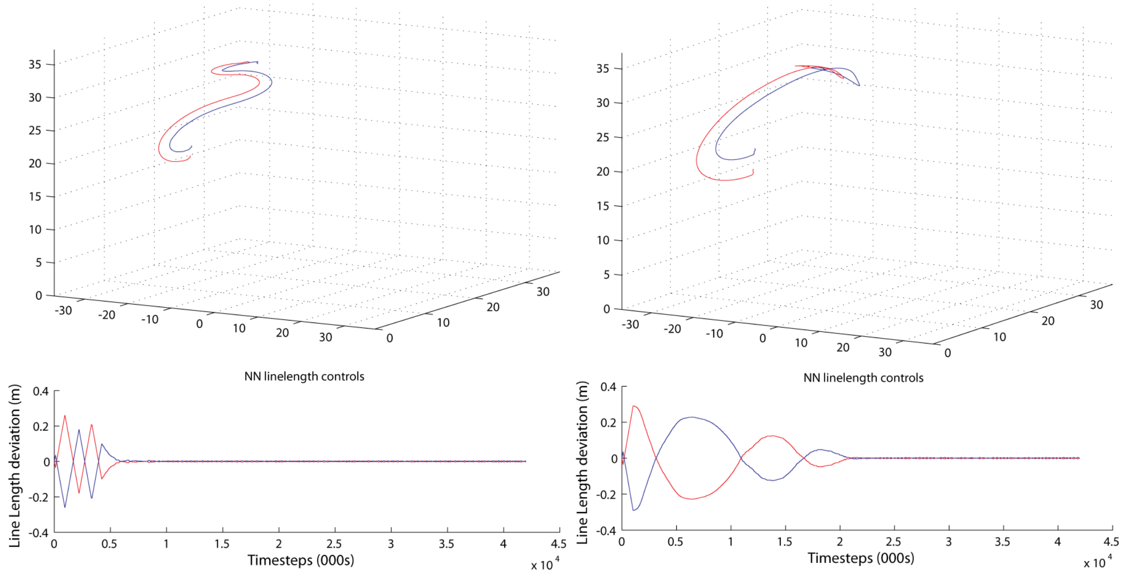


Figure 7.27 Neurocontroller performing stabilisation task with wing area of 1.8 and 8 metres, left and right respectively.

We also described a 'baton-passing' component of the task in 7.3.5.5 in which the controller was required to take control following a sequence of imposed control. The controllers were able to pass the stage in which this component was introduced. Because it was so easy to generate control sequences that would crash the kite, the sequences in evolution are limited to a maximum of 5 seconds and 0.1m deviation in line length per actuator. Below we show the network taking control in one scenario within the evolved range (Figure 7.28) and another outside that range (Figure 7.29).

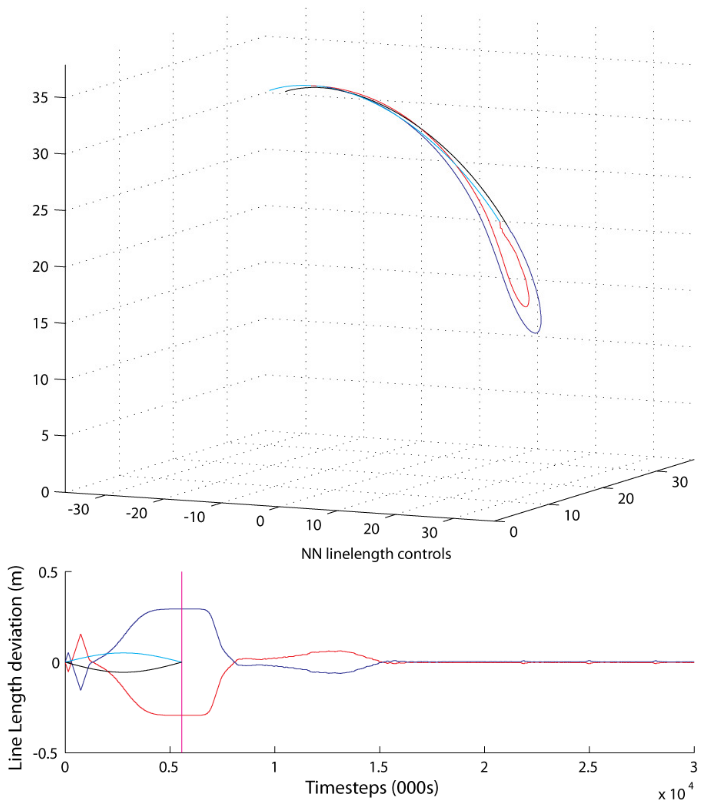


Figure 7.28 Neurocontroller performance in baton passing task. Wingtip positions pre-handover shown in black/cyan (above). Lower plot shows pre-handover control sequence in cyan and black for left and right steering lines. Vertical purple line indicates handover time.

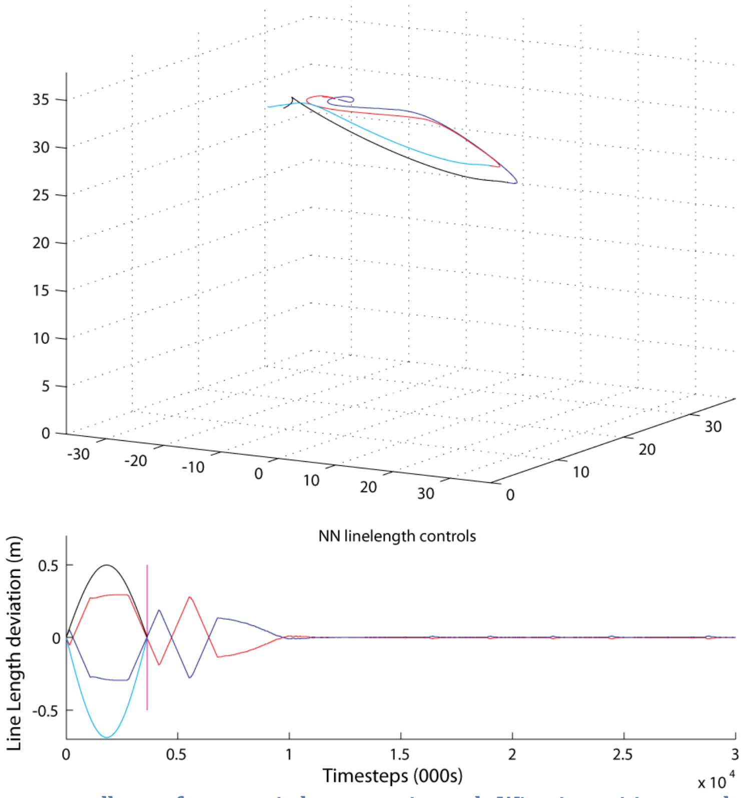


Figure 7.29 Neurocontroller performance in baton passing task. Wingtip positions pre-handover shown in adjusted colour (top). The pre-handover control sequence in cyan and black for left and right steering lines(bottom). Vertical purple line indicates handover.

Figure 7.28 shows the neurocontroller taking control after a long, low intensity steering input steered the kite over to the right hand side of the wind window. The control output for the neural network prior to handover is plotted along with the artificial control signal. Interestingly the network output, which is still receiving input whilst uncoupled from the 'actuators', oscillates briefly in the wrong direction before saturating at the maximum value acting counter to the generated sequence. After handover it remains saturated briefly until the kite rotates to head back up towards the zenith after which it quickly drops off. In Figure 7.29 the pre-handover sequence is shorter but much more aggressive, looping the kite downwind. The handover point is at the bottom of the loop when the kite is travelling from left to right horizontally across the wind window. The controller successfully completes the loop rather than steering back in the original direction, although as can be seen, its early uncoupled output would have tried to steer the kite back the way it had come. Interestingly the controller seemed to have committed to complete the loop even before it had been handed control as can be seen on the pre-handover control plot. It should be noted that the controller was not infallible in these situations beyond the evolved range and if the kite was being steered very aggressively into the ground, the controller would need a second or two to be able to retrieve the kite. Also the network seemed to have been 'wound up' in that the controllers would often oversteer the kite for a brief period after taking control.

We also confirmed that the networks were capable of performing the task across the full range of variation of kite mass, actuator speed and actuator and sensor delay.

7.4.2.4 Commentary on evolutionary progress

To the end of delivering a viable controller we deployed a number of strategies implementing and managing the evolutionary process, which have been introduced previously and of which some were novel methods which are still at a very early state of investigation. Plotting evolutionary progress by fitness scores cannot be used to compare performance of groups of evolutionary runs as per Chapter 6, because it does not capture progress through the stages of the shaping scheme. Figure 7.30 shows the progression through the high level stages in three groups of ten evolutionary runs that were limited to stage six. It is clear that the seeded runs exhibit a greater average progression with the entire set of runs achieving the final stage. The unseeded runs still performed well with a majority of runs achieving the final stage. The performance of the most successful runs was on a par with those of the seeded runs but the yield in terms of the proportion of runs achieving a good level of performance was lower. Finally, the unseeded runs without velocity inputs struggled to progress through the stages of evolution successfully, which provides confirmation that the provision of this data does indeed render this particular task simpler for the neural network to complete.

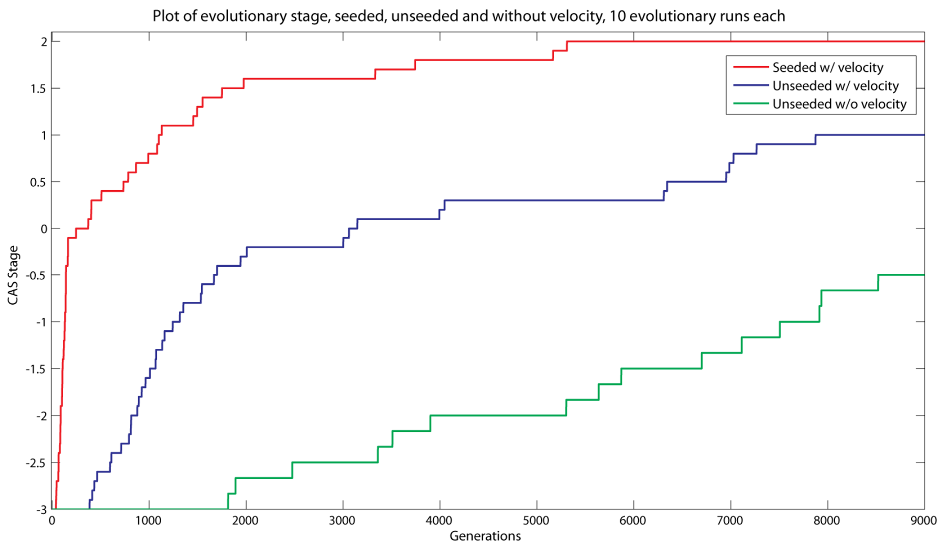


Figure 7.30 Plot average progression through the shaping scheme, limited to stage 6. Three groups of 10 evolutionary runs, seeded, unseeded and unseeded without velocity inputs.

Figure 7.31 shows the progression of a single seeded evolutionary run, with the progression through the staging process marked by vertical lines, that change colour with each major stage. It can be seen that in this run, the seed network is already performing at an adequate level and rapidly progresses through the early stages where there is limited task variability. The error can be seen to jump after a new stage or substage is initiated, an effect particularly pronounced at the start of stage six where the purple vertical lines start. It is notable that the progression through the substages of stage 6 is somewhat erratic, with some substages being incremented in quick succession, and some only being surpassed after a long period of near neutral evolution.

Figure 7.32 shows an unseeded evolutionary run in which an early random population performs far worse although the population rapidly increases in performance and progresses to Stage 6. Once there it also proceeds to increment through this apparently more difficult stage in an equally punctuated manner.

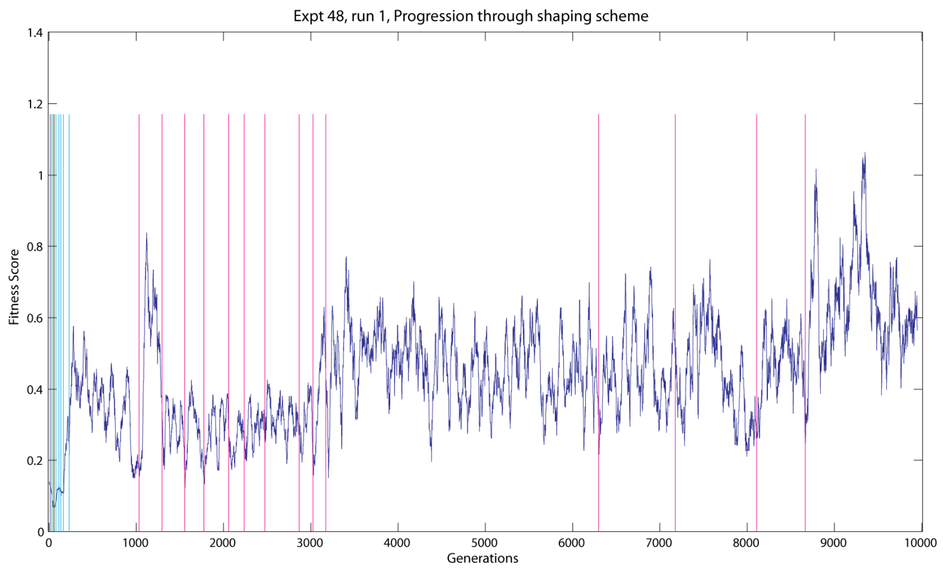


Figure 7.31 Plot of fitness (lower is better) of a single seeded evolutionary run with the progression through the shaping scheme stages marked with vertical lines

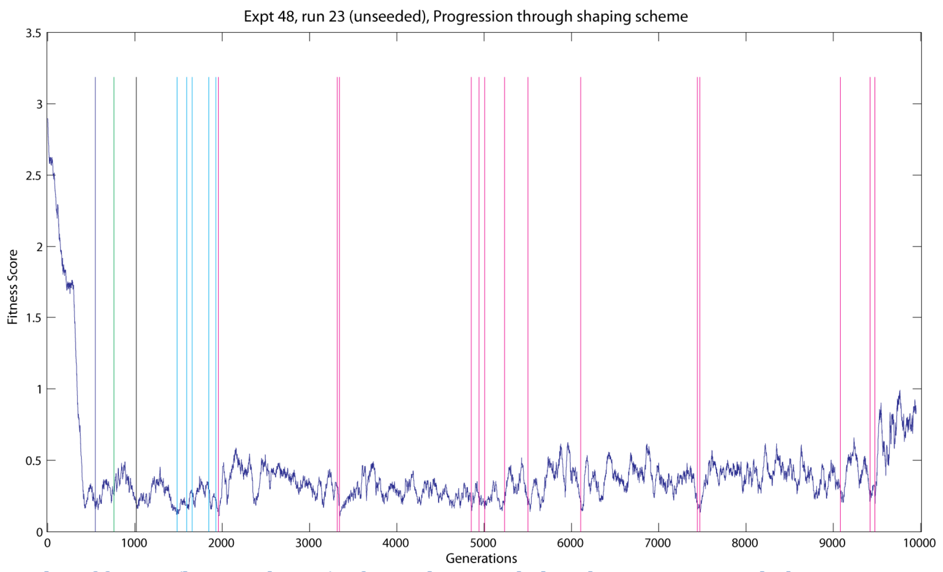


Figure 7.32 Plot of fitness (lower is better) of a single unseeded evolutionary run with the progression through the shaping scheme stages marked with vertical lines

Figure 7.33 shows the progression of a single evolutionary run where there are 6 different fitness functions which are selected probabilistically for a given generation. It is clear that this run did not lead to much progression through the staging process, however this is not comparable with the prior experiments because each fitness function had its own arbitrarily set threshold for progression. The relative performance of each fitness function is interesting however, one fitness function appears to become stuck at a high value after the last increment is made, perhaps indicating that it is not compatible with the minimisation of the other fitness measures. The other function scores are remarkably dynamic, and do not necessarily move in synchrony, indicating that they do indeed relate to different aspects of the task.

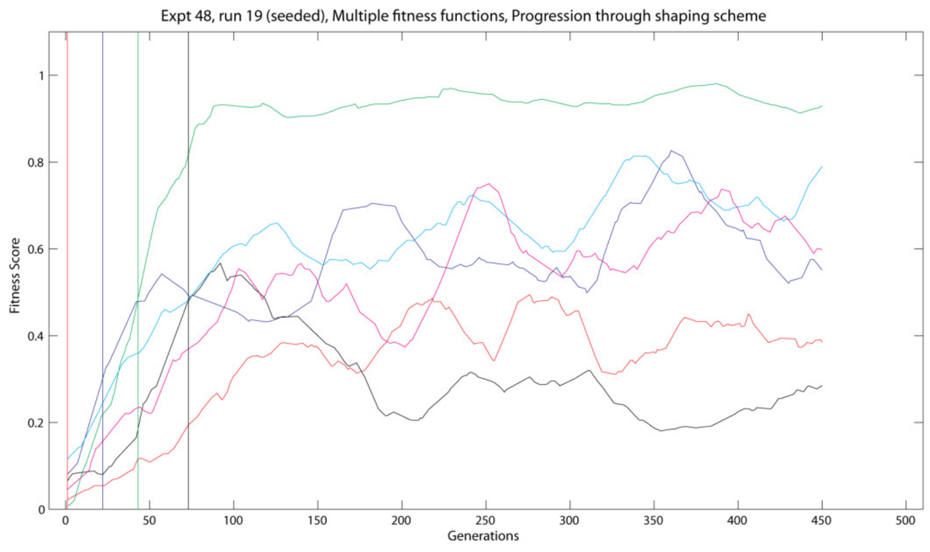


Figure 7.33 Plot of fitness (lower is better) using multiple shuffled fitness functions.

7.4.3 Evolutionary model tuning

The below plots of the kite structure when half the kite is constrained and half loaded at one or more locations demonstrate that it was indeed possible to use a GA to tune a multi-body constrained particle model to match measured deformations. Figure 7.34 shows the very poor performance of a model from the initial generation in which the kite buckles and twists in an unrealistic manner.

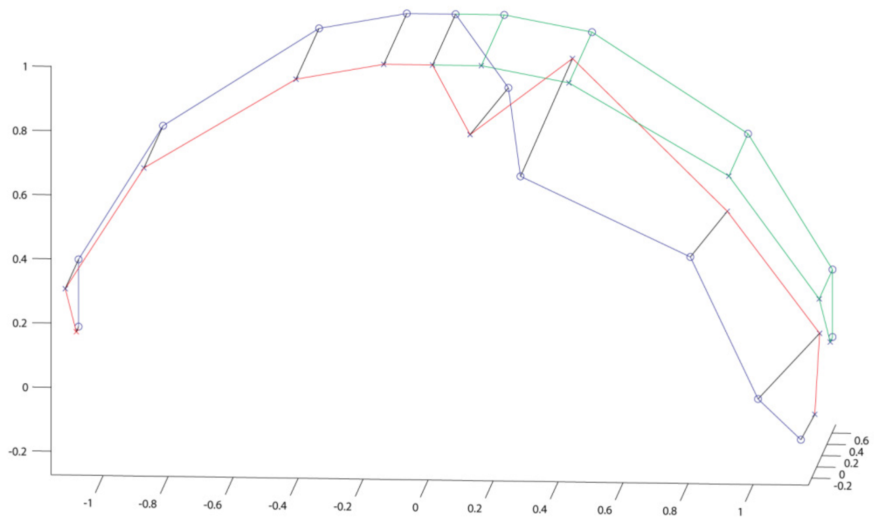


Figure 7.34 A naive kite model with random constraint generation rules, right half of kite loaded and exhibiting unrealistic deformation. Unloaded right half shown in green.

Figure 7.35 shows a model after 200 generations of evolution which is exhibiting much more realistic deformation, in terms of both amplitude which is well with the expected range, and distribution, as can be seen the rear of the kite has deformed considerably more than the front. This and other well performing constraint sets could be used to validate the evolved neurocontroller in further work, in the same manner as the 8 panel model described earlier.

Further evolution however did seem to fit the data by overconstraining the kite, indicating that some of the measurements were mutually incompatible. However this approach should work well with data more systematically collected using the gantry robot or stereo photogrammetry.

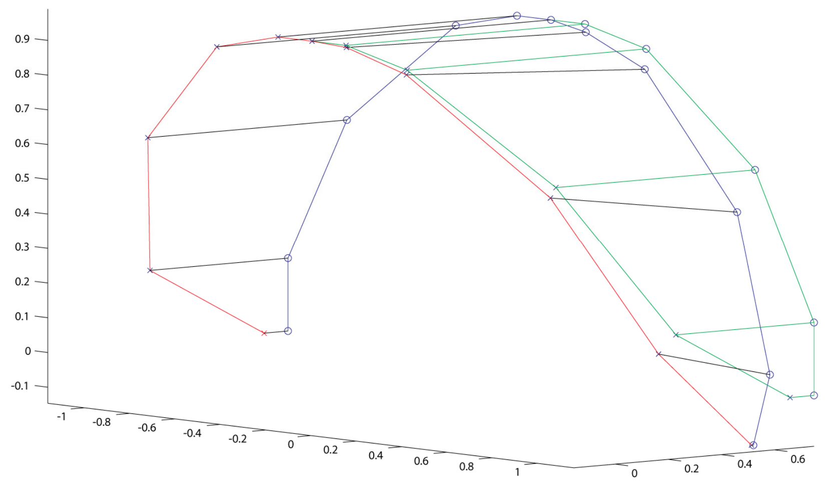


Figure 7.35 A kite model with evolved constraint generation rules, right half of kite loaded and exhibiting fairly realistic deformation. Unloaded right half shown in green.

7.5 Discussion

In the first set of results in section 7.4.1, we carried the work presented in chapter 6 to its logical conclusion and it showed that it was possible to evolve valid flight control behaviour with a repetitive looping flight trajectory when the line length was subject to change. We described how the neurocontrollers were robust to change within the evolved parameter space, yet demonstrated an interesting fragility to a key input value being out of range, namely that of the line length input. This was highlighted by running experiments for longer than was possible during the evolutionary process, resulting in the neurocontrollers simply stopping the oscillatory flight behaviour. This fragility is to be expected and is the key motivation behind the systematic incremental introduction of variation in the task and model parameters in the stabilisation task. One interesting result was that there was only a 10% yield of evolutionary runs that produced a sensible behaviour, even without the full range of parameter variation. This was without the fine grained shaping scheme being deployed and seems to suggest that it was in fact necessary in the next task.

In the subsequent section 7.4.2 we showed that these simple hand coded network was capable of performing in the zenith stabilisation task at the slower end of possible dynamics i.e. when the flying lines or longer or the wind slower. We showed how when the kite dynamics became faster with shorter lines or faster windspeed, the controller was prone to flying the kite in undesirable oscillations, and in yet faster situations the kite would be crashed. In contrast the evolved neurocontroller was capable of performing the task across this whole operational envelope. We also demonstrated that this evolved neurocontroller was able to adapt to relatively severe changes in the physical set up of the kite system including large biases on the length of the steering lines and changes to the kite aspect ratio or wing area. In some cases the controllers were prone to a slight amount of oversteer but generally they performed the task well in difficult conditions. The fact these controllers were able to generalise to such a wide range of operational conditions bodes well for transfer to a real world system. Indeed the controllers were able to steer a kite composed of a larger number of panels and subject to a greater degree of deformation due to a lower number of constraints per particle. This kite model was not used at all in the evolutionary process, so the neurocontrollers have been demonstrated to have some kind of model independence. Whilst this does not guarantee that the behaviours will transfer to the real world system, if the neurocontroller was not able to successfully control the eight panel model we could not reasonably expect it to transfer the behaviour successfully on to the hardware platform.

In this work we deployed an unusually fine grained shaping algorithm which gradually incremented the variability of the task environment. Whilst we did not systematically compare evolution with and without this shaping procedure, the few unseeded test runs that were performed without the shaping procedure could not generate any kind of valid zenith stabilisation behaviour. We expect therefore that the fine grained shaping algorithm was in fact necessary. By structuring the evolutionary process in this manner, we initially generate the gross behaviour that is desired, even though the neurocontroller is likely to be highly fragile to changes within the task dynamics or parameters. By releasing elements of variability such as changes in wind speed, changes in kite physics and changes to the implementation set of model parameters in a very gradual manner, we are able to ensure that the desired behaviour is retained within the population and yet we are able to progressively widen the conditions in which we expect the behaviour to be reproduced. The value of such an approach is clear, using such a technique we were able to perform a given task in an extremely wide range of conditions that would have inhibited the evolution of successful control strategies were this variation to have been present from the beginning of the evolutionary process.

We also confirmed the expected benefits of two aspects of the evolutionary algorithm and neural network implementation that were not systematically compared in the previous chapter. Firstly we showed that seeding the initial population with hand coded neural networks that produced a roughly correct yet fragile version of the desired behaviour did increase the success rate revolutionary runs in terms of progression through the staging process. We showed that populations that were randomly initialised in the usual fashion were capable of achieving equivalent performance levels yet the yield of the evolutionary runs was poorer with more runs failing to proceed incrementally through the stages. Finally we demonstrated that unseeded populations in which the neural networks did not have inputs corresponding to the rate of change of the primary input parameters really did struggle to generate appropriate behaviours. Those evolutionary runs where an appropriate behaviour was evolved in the velocity free input networks struggled to perform the desired behaviour when the widening range of possible task parameters rendered of the physical dynamics far more variable. Finally we showed that evolution can be used in order to select a set of constraints such as to reproduce measured deformation in an asymmetrical flexible structure. This could be not only used in a validation step but deliberately damaged or underinflated structures could be measured, their deformation dynamics replicated and the ability of neurocontroller to deal with these specific defects assessed.

7.6 Summary

In summary, this chapter has sought to demonstrate that an evolutionary robotics approach can generate valid, robust behaviours for both flying the kite in repeated looping trajectories in order to maximise line tension during the generation phase of a two-part generation/retraction pumping cycle, and in stabilising the kite at the zenith regardless of the starting position, orientation or movement of the kite at the start of the trial. As the latter behaviour was the most appropriate for our test hardware where a reeling system is not available, we aimed to make this behaviour as transferable as possible. To this end we introduced and deployed a fine grained shaping algorithm in which incrementally larger variations to the environmental characteristics, physical instantiation of the kite and model implementation set parameters were released once the average performance of the population met a predetermined threshold. We reported that networks without velocity inputs struggled to pass through the shaping stages within the evolutionary time period. We showed seeding the population with a hand coded neurocontroller increased the yield of the evolutionary runs but not necessarily the best performance attained. The best resulting neurocontrollers were capable of performing stabilisation task when faced with a wide variety of parameter changes and were able to control an eight-panel underconstrained kite model which they were not exposed to during the evolutionary process. We also described results showing how a GA could be used to configure more complex kite models in such a way as to replicate measured deformation data.

Engineering is the art of modelling materials we do not wholly understand, into shapes we cannot precisely analyse so as to withstand forces we cannot properly assess, in such a way that the public has no reason to suspect the extent of our ignorance.

Dr AR Dykes, British Institution of Structural Engineers, 1976.

8 Development of a hardware platform

In this chapter we will review the implementation of the hardware and software necessary to control the kite and acquire the information that must be provided to the neural network. We start with a description of how the position and orientation of the kite is tracked. We then go on to describe the development of kite control hardware through a series of prototypes. Unfortunately even the most powerful servos that were installed in the control equipment (Dynamixel 106+) suffered failures which rendered the testing of the neurocontrollers impossible. However, the description of the techniques and systems used may be useful.

8.1 Kite tracking

Human pilots have a lot of information regarding both the kite and the environment when they are flying steerable kites. Not only do they have very precise information regarding the load on the line and the position of the kite, but they are also aware of the kite's orientation with respect to the wind window and of the current speed of the wind and its variability. It was certainly not the intention of this study to render the problem of kite control more difficult than necessary, therefore our initial instinct was to make as much information as possible available to the neurocontroller. Here we describe efforts taken to measure the state of the kite in terms of position and orientation and the force being exerted against its lines, starting with the use of MEMS –based inertial measurement units (IMUs), followed by the use of visual tracking algorithms. Successes and failures of both of these approaches are noted and the final configuration of the kite tracking system used for real-world neurocontroller testing is presented.

8.1.1 Experiments with IMU devices

An appealing approach to the estimation of the kite's state would be to measure it directly in hardware using an inertial measurement unit. These are capable of providing an accurate estimate of orientation of the kite in three degrees of freedom (i.e. roll, yaw and pitch) in real time. Commercially available systems such as the MTI-G marketed by the Swiss company Xsens are even able to incorporate GPS data in order to generate a 3d position estimate alongside the orientation estimate, although the position estimate has a rather coarse spatial and temporal resolution ("Xsens MTI-g: GPS aided AHRS," n.d.). Unfortunately these systems are rather expensive and beyond the budget of a DPhil project.

Still desirous of an IMU solution, we did some initial tests alongside of the Specknet group at the University of Edinburgh in which we attached a wireless IMU device, the Orient 2 (Arvind & Bartosik, 2009a),(Arvind & Bartosik, 2009b) to a Flexifoil Atom08 kite which was flown manually whilst being recorded on video. The video was taken in order to verify the IMU estimate by later synchronisation of the two data sources. As a Leading Edge Inflatable (LEI) kite, this kite has a semi rigid skeleton structure consisting of airbeams (see Figure 8.1), the IMU could therefore be mounted to a relatively stable point which was chosen to be the intersection of the leading edge beam and the most central chordwise beam.

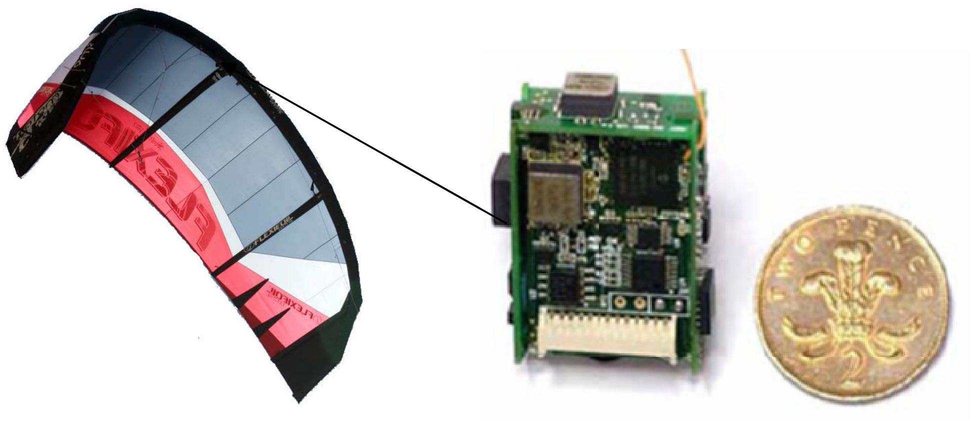


Figure 8.1 The Flexifoil Atom 08 kite used for testing (left). The airbeams running along the leading edge and also chordwise can be seen in black. The Orient 2 wireless IMU device with integrated power supply produced by the Specknet group at the University of Edinburgh is shown on the right. This was mounted on the stiffest part of the kite at the position marked by an arrow.

This device is able to compute a quaternion based orientation estimate on-board and transmit this estimate wirelessly to a laptop connected receiver at ground level using an integrated Bluetooth module. By performing some aggressive non-periodic flight manoeuvres, it was simple to recognise enough of the flight pattern to synchronise the recorded orientation estimate with the video. Unfortunately the orientation estimate was seen to drift significantly from the visual recording. Partly this was due to the accelerometers on the device having their range limited to ± 2g, whereas in flight, much more severe accelerations were encountered during aggressive manoeuvres.

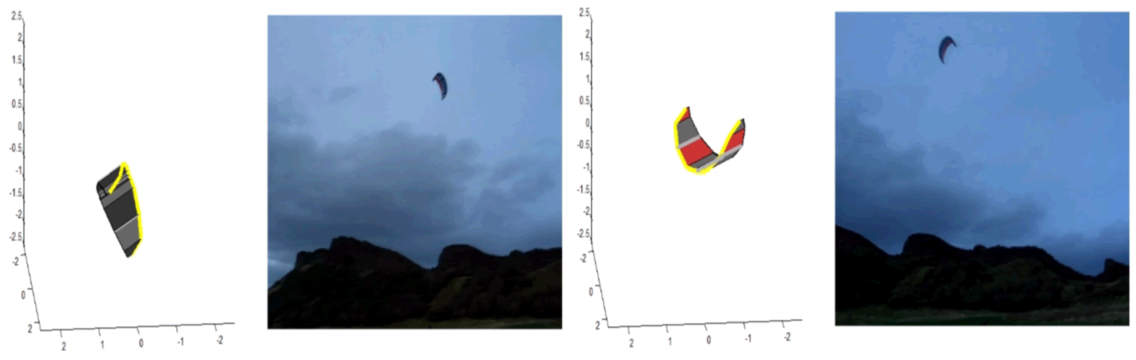


Figure 8.2 Two pairs of figures showing the kite orientation estimate from the IMU against the simultaneous frame from the video. The left pair of images shows an estimate early in the sequence which is reasonably well matched to the real kite. One minute later, the right image pair demonstrates that error has accumulated and the estimate is a poor reflection of the real system.

However the underlying reason behind the drift in the orientation estimate from reality was principally limited by the implementation of the sensor fusion algorithm onboard the device. This required regular periods of near zero acceleration in order to reset the orientation estimate using a combination of the known acceleration under gravity and the magnetometer readings. For the tracking of human body movements, which is the application for which the Orient2 fusion algorithm was intended, this is a reasonable assumption to make, however the flight of the kite is so dynamic that it could not meet this requirement of the sensor fusion algorithm. Classical sensor fusion algorithms such as the Kalman filter should be able to cope with the integration of the accelerometer, gyroscope, and magnetometer data. However the implementation of a Kalman filter or one of its more advanced and potentially more appropriate variants such as the Extended or Unscented Kalman filter is unfortunately beyond the scope of the work here.

8.1.2 Camera tracking

Having discounted the use of inertial measurement units due to cost and performance issues, the next most obvious solution is to use a camera. However there did not seem to be an off-the-shelf solution capable of tracking the position and preferably orientation of the kite at the required frequency within the required field of view. Therefore we created a solution from scratch with the aid of some excellent public domain tools that are available.

The first issue was that the kite is free to move in a full quarter sphere of space, and potentially slightly more than this if the wind moves or the kite overflies the Zenith position. As most available digital cameras and lenses have an angle of view far narrower than this, either the camera would have to be moved such that it constantly pointed near the kite or a camera with a mirror or fisheye lens would have to be used. We discounted the first option as it would potentially place restrictions on the manner by which the kite would be physically controlled and limit opportunities to test control hardware and software on other systems under development. On investigating the other options we looked at the use of mirrors of various shapes including spherical mirrors and conical mirrors. However it appeared that the use of mirrors was going to be problematic because of frequent overrepresentation of certain areas of the visual field at the expense of other areas (Haber, Zhou, & Murai, 2006) and common blackspots due to the necessity for the camera to be placed not too distant from the mirror or because of the use of support pillars. Therefore we used the Omnitech robotics 190° fisheye lens, which exceeded the necessary characteristics by capturing more than a half sphere field of view. By being compatible with a standard micro-lens mount we were able to use this lens with a standard web cam. Section 8.1.2.1 illustrates how we correct for the shape of this lens when tracking the kite and section 8.1.2.2 describes the algorithm used to provide estimates of the location of the kite and its orientation.

8.1.2.1 Pixel to world mapping with a 190⁰ fisheye lens

Because the position of an object on the sensor of the camera does not correspond directly to the position of the object within the real world, it is necessary to correct for the shape of the lens such that a mapping can be provided from any given pixel to a angular position within the real world. Fortunately there is an excellent tool available in the form of the OCamCalib toolbox, which requires only a series of images featuring a chequerboard of known geometry to appear at various positions within the field of view(Rufli, Scaramuzza, & Siegwart, 2008)(Scaramuzza, Martinelli, & Siegwart, 2006). The software performs a semi-automated process in order to extract the corners of the board with which a lens geometry model is generated, allowing the production of a reference mapping from a given pixel in the visual field to real world positions in the form of elevation and azimuth angles, or Cartesian coordinates on a unit sphere (See Figure 8.3).

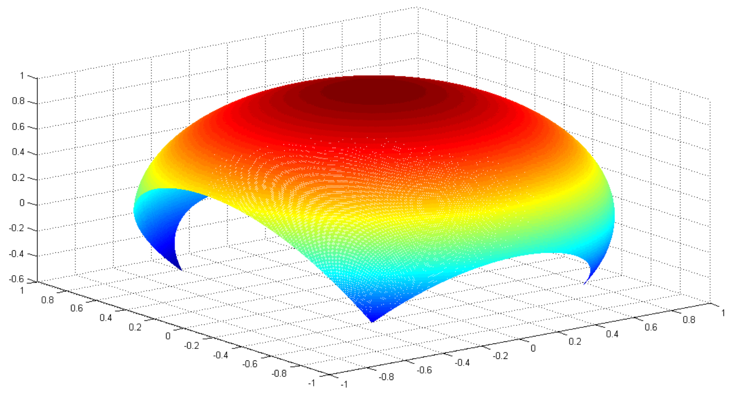


Figure 8.3 A 3d plot of the map from pixel X/Y positions to Cartesian coordinates on a unit sphere. The areas where the sphere is truncated reflect portions of the lens projection that fall off the edge of the cameras CCD detector.

8.1.2.2 Blob tracking algorithm

Being able to map from the camera to the real world, the remaining task was to write an algorithm capable of identifying the kite position and orientation and track it in real time, preferably at a rate 20 Hz or above. The kite tracking problem has some aspects which make it a relatively simple computer vision problem, specifically that all that is required is to track a high contrast object across a generally brightly lit background. However some other aspects do add considerable difficulty, namely the sudden and extreme lighting differences to which the camera is exposed when pointing directly at sky in which clouds may be alternately obscuring and revealing the sun. Because in early trials it became apparent that the only camera to which we had access had exceptionally variable colour performance across trials, and additionally because the test kites available for different wind conditions were different colours, we made the decision to use only black and white data from the camera. We implemented two very simple blob tracking algorithms, each with slightly different weaknesses when deployed in the real world so as to provide different options in the field. The first and simplest approach was simply to assume that the kite was the darkest object in the sky. By setting an appropriate threshold of intensity at the beginning of each trial, it was possible to pull out only the kite from the image in many lighting conditions. The second approach was to assume that the kite was the most changeable portion of the sky. By taking a snapshot of the sky immediately before the launch of the kite, or after moving the kite significantly, then performing a simple subtraction of the image intensities, it could be assumed that the patch of sky that just suffered the largest drop in image intensity was the kite. This background reference image is automatically captured when the tracking program is initiated, but a fresh capture can be triggered by pressing a button on the control joystick assigned for this purpose.

The first step in deploying both of these algorithms is to define the area of the image in which the algorithm searches for the kite. Because we expect flight trials to be short enough that the wind direction will change very little, we first crop the image in half and remove the upwind half of the wind window simply to reduce the computational demand. Secondly, because the camera is not designed for professional use, the lens is misaligned with the CCD sensor and so this misalignment must be quantified and corrected for. Assuming that the lens produces a circular image of a given radius, if we can estimate coordinates for the circle centre as well as its radius then the search area can be restricted purely to the area of the CCD which is being exposed to the light. The screenshot in Figure 8.4 shows the GUI which allows the user to specify the size of the image on the detector and its vertical and horizontal offset.

Once these basic characteristics are set, the areas of the CCD not exposed to light are excluded from the search area, and an area close to the horizon of approximately 15^0 of elevation is defined in which more extreme contrast rules can apply due to the likelihood of effectively thicker cloud at the edge of the image. This band of low elevation image is also excluded from the initial search area from which the kite blob is initially acquired in both algorithms. Once this is performed the user can tune the default initial contrast threshold. This is selected such that intermediately dark objects such as clouds fall below the threshold and only the kite has sufficient intensity values to exceed the threshold. The threshold is not absolute but a multiple of the mean intensity of all the non-excluded pixels. Because the kite is flying at some distance from the ground it is reasonably well lit, therefore pixels that are nearly perfectly black are also excluded from the search area,

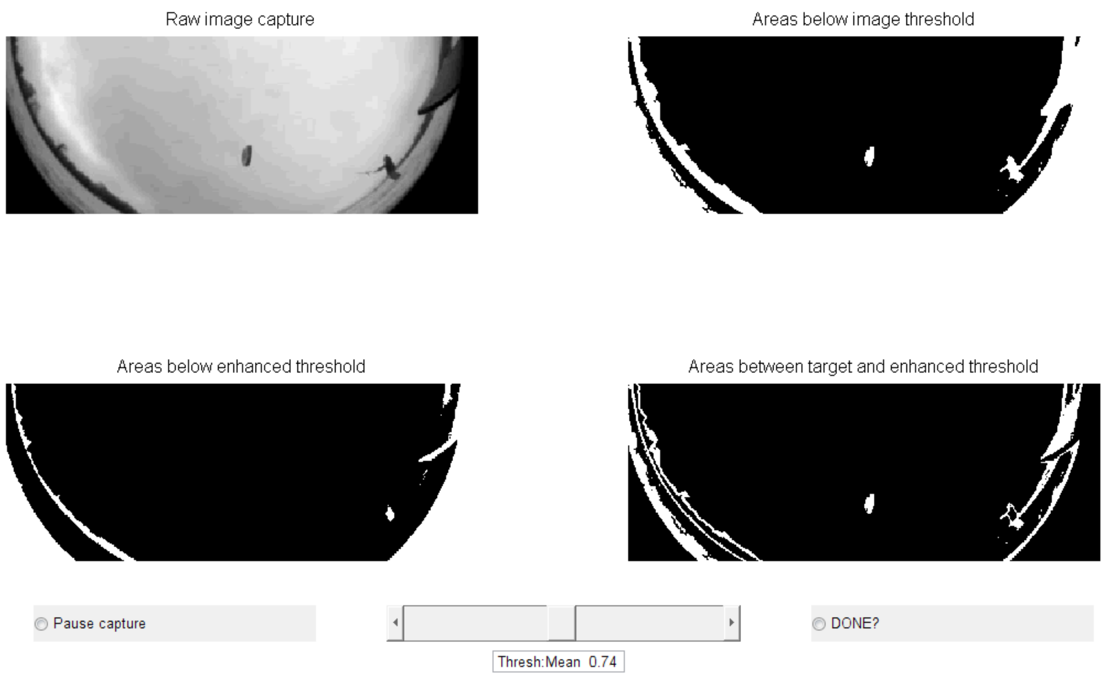

Raw image capture
Areas below image threshold
Areas below enhanced threshold
Areas between target and enhanced threshold
Pause capture
Thresh:Mean 0.74
DONE?

Figure 8.4 The GUI with which the threshold detection values are set.

For the second algorithm, the change in intensity between the reference and current image is also subject to a threshold. However in this case the process starts with a preset value that does not require a GUI. In both cases however, because we know the size of the kite we are using and the length of the lines, we also know approximately the mass of the pixels in the kite blob. If the number of pixels that differ between the reference and current images by the threshold exceeds this expectation by >20% in either direction, then the threshold is incremented or decremented by a small value.

At this point, in either algorithm, all that is necessary to acquire the position of the kite is to find the maximum density of above threshold pixels on both the X and Y axes. The last stage in the initialisation process is to row by row work inwards from either edge of the image to remove the objects on the horizon. If we expect people or other elements to be moving it is possible to repeat this horizon extraction at an appropriate frequency. Because we already know the kite position, the kite blob can be exempted from this process.

Once more than one image has been captured, the speed and direction of the centre of mass of the above threshold pixel blob can be used to predict the next position of the blob. This information coupled with descriptors of the size of the expected pixel mass of the kite can be used to define a search window for subsequent timesteps. This means that a much smaller portion of the image needs to be processed than in the initial location of the kite. It also prevents distracting objects such as birds and people walking through image from being picked up if they exceed the contrast threshold. The search window size is decremented until the kite blob takes up approximately 1/3 of the pixels in the image. This active windowing process is essential in the case of the difference algorithm due to cloud movement and lighting changes rendering the reference image out of date. Therefore at 10Hz the reference image is replaced with the latest image with the exception of the current search window where we know the kite to be.

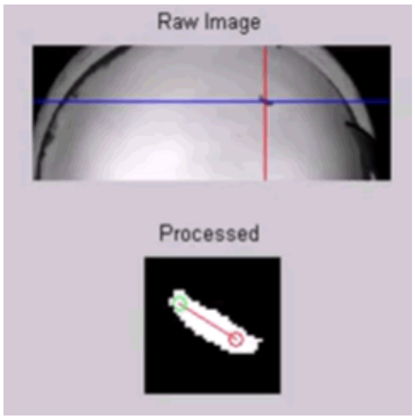


Figure 8.5 A screen shot of the live tracking software, the currently identified kite blob central pixel is plotted on the raw image as a pair of crosshairs (upper). The extracted blob in its window is shown in the lower pane, with a superimposition of the currently identified blob orientation.

It is possible that if a person or object might come in between the lens and the kite the blob will be lost and the window may fall out of alignment with the kite. If this happens, the window is rapidly enlarged such that within three samples it once again encompasses the whole visual field as in the initial timestep. In this way, providing no other high contrast objects, or new objects in the case of the difference algorithm, are present in the visual field, the kite will be reacquired as in the first timestep, allowing the windowing process to resume.

We obtain an estimate of the orientation of the kite by using the Matlab function written by Dr Oliver Woodford. We pass the function a binary matrix that is the blob within its search window, and the function finds the principle component of the directionality of the blob. Unfortunately due to the warping effect of the lens, we cannot consider this to be an estimate of yaw of the kite in the real world system, which is the value that is utilised by the neural network in the simulation environment. For example if the blob is aligned with a completely horizontal line whilst at the zenith position, the orientation will be given as zero, which corresponds to its real state. However, the same blob alignment occurring when the kite is near the ground at the left or rightmost extremes of the wind window will also produce a blob orientation of zero when in fact the kite is in fact rotated by 90^0. The orientation estimate is therefore produced by processing the blob orientation and coordinates in exactly the same manner as in the simulation. A pixel at a set distance from the blob centre is chosen according to the orientation to be the wingtip reference location. Both pixels are then converted to real world co-ordinates. These coordinates are rotated in sequence around the elevation and then azimuth angles to place the coordinate pair directly downwind and at ground level and their relative angle is calculated.

The visual processing described was found to give a good estimate of kite position and orientation in one axis at approximately 25Hz in favourable lighting conditions. It was found however that if the sun was in the image the camera would have a blind spot in the region of the sun. Also if there were brightly lit clouds with patches of blue in the sky, the camera would seek to enhance the contrast, making the blue parts of the sky far darker than they appeared in reality which would occasionally impair the acquisition of the kite. For this reason, testing was generally performed in conditions of nearly 100% cloud coverage, when both tracking algorithms performed well.

8.2 Hardware progression

In this section we review the series of test robots that were built for the purpose of testing the neurocontrollers. For each robot the design is described along with the reasons for key design choices and then its performance for real-world flight control with a human pilot is evaluated.

8.2.1 First robot

The first robotic testing platform was built by Bill Bigge, the robotics technician for the CCNR, and is shown in Figure 8.8 alongside the kite that it was used to control.

8.2.1.1 Design and mode of operation

This platform was closely inspired by human control of kites using what is known as a control bar. This is illustrated for a conventional manually controlled ‘de-powerable’ kite in Figure 8.6. All of the lines from the kite run to the control bar, the lines attached to the leading edge run through the bar to attach to the pilot and the trailing edge lines are attached to either end of the bar. The trailing edge lines (red) can have their relative lengths altered by rotating the control bar in order to steer the kite. Moving the control bar towards or away from the pilot will alter the relative lengths of the leading and trailing edge line pairs and thus alter the attack angle.

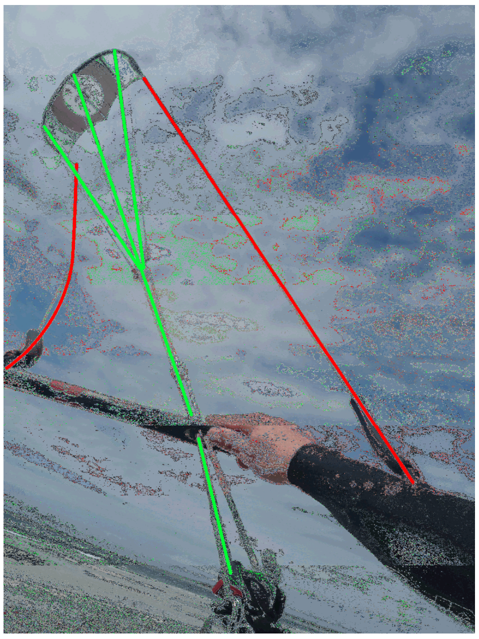


Figure 8.6 A typical 'de-powerable' kite setup. The leading edge lines (green) bear most of the aerodynamic load which is transmitted to the pilot via a harness. Note how one steering line is slack.

As shown in Figure 8.7, the control bar and associated servo is mounted at the end of an arm which is itself mounted on the top of a vertical support frame. The weight of the arm is counterbalanced by a short section which extends rearward of the mounting point, to which a battery pack is attached. This lead acid battery pack is of considerable weight (approximately 2 kg) which is sufficient to act as a counterbalance to the longer but lighter portion of the arm that holds the servo and electronics. This counterbalancing means that when there is no tension on the flying lines, the device sits in a slightly inclined position running from battery to actuator. This inclination is possible because there is a set of gimbals at the junction between the actuator arm and a vertical frame. The gimbals allow a limited degree of rotation such that the arm can be parallel to the ground or can be raised approximately 60° away from parallel position, i.e. 30° away from perpendicular to the ground. Rotation in azimuth of approximately 45^0 in either direction was also permitted as shown in Figure 8.7 (right).

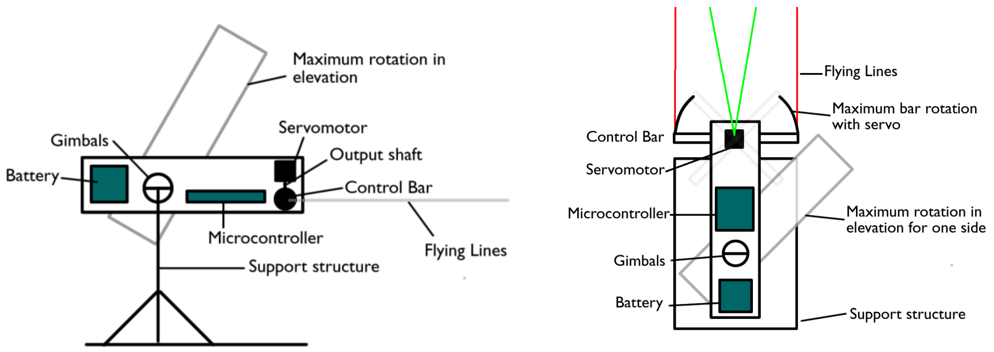


Figure 8.7 Simplified schematic views for robot 1, for the side (left) and overhead (right).

A servo motor has its output shaft running through the centre of the bar and is coupled to the bar such that rotation of the servo will cause rotation of the bar around a vertical axis. The effect of this rotation is to pull one end of the bar away from the kite and push the opposite end towards the kite, this effective difference in the length of the steering lines causes the kite to respond appropriately.

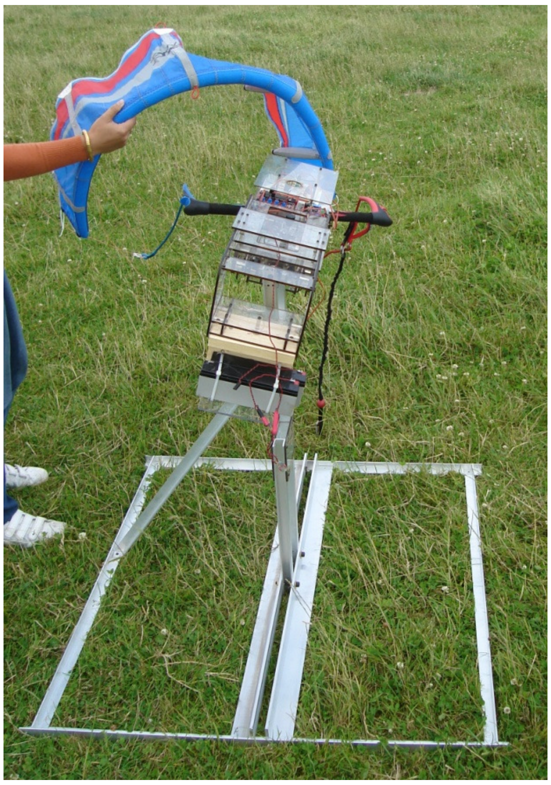


Figure 8.8 A photo of the joystick controlled robotic test platform with the small kite which it was capable of steering.

There is another degree of freedom at the junction between the vertical support frame and the actuator arm which allows a limited degree of roll as can be seen in Figure 8.8. The motivation behind giving the arm these 3 degrees of freedom was that with the kite at any point in the sky, tension in the lines would cause the arm to point almost directly towards the kite, thus ensuring that the rotation of the control bar would cause the intended difference in the length of the steering lines to be manifested. For example, if this were not the case and the kite were directly overhead of the robotic platform, rotation of the bar around a vertical axis would have zero net effect on the relative lengths of the lines from control bar to kite therefore no steering would occur. By allowing a pitching motion of the actuator arm, we allow the steering effect to be aligned with the kite thus rotation of the control bar now occurs around a horizontal axis such that its rotation causes the intended changes in relative line lengths to be manifested. The necessity for these degrees of freedom was determined in an early test flight in which the support frame and hence the potential for controlled rotation of the actuator arm was absent.

8.2.1.2 Operational performance assessment

The initial performance of the robotic platform was highly encouraging, the robot was able to be used to control the flight of the small kite with some accuracy. A video demonstrating a test flight of this machine is online at: http://www.youtube.com/watch?v=_8umgx4h2Us , some stills are shown below.

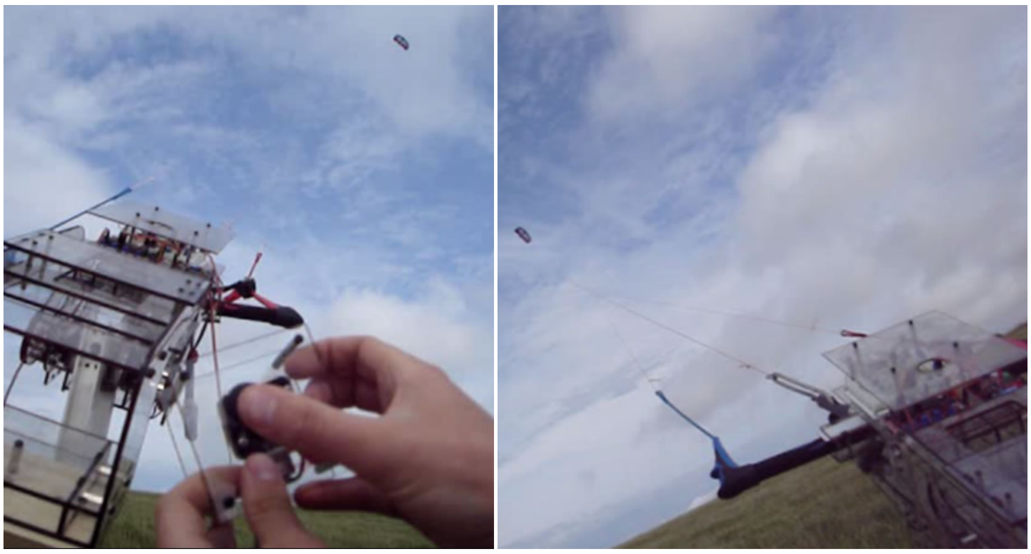


Figure 8.9 Images showing the first robot in testing with manual control

However successfully the system operated under manual control, it soon became apparent that this design concept had important limitations were the system to be used to test evolved neurocontrollers. Firstly, because of physical constraints, the arm rotations of the arm could not be provided with the angular range demanded by the potential of the kite to move within a full quarter-circle of sky. This meant that there was effectively some limitation of the steering actuation available at the periphery of the wind window. Secondly because the arm has considerable mass, it had its own dynamics that would impinge on the steering of the kite. In this incarnation, a mechanism to change the relative lengths of the front and rear lines and hence the angle of attack of the kite was not deployed. Adding this capability would have at least required another servo motor and would have added additional weight and complexity to the system. Incorporating the movement and constraints of the arm mechanism into the physics model with which the kite controllers are evolved would not have been trivial and would have added extra computational demand to the evolutionary process which was already operating at the very lower levels of desired speeds. Additionally because the kite is being steered from ground level using all four lines, this system is not amenable to integration with typical winches that would be required to test the functionality of any evolved neural controller on systems where the line length is dynamic. Finally, because the system is attached to a fixed point at ground level, there is no way to relieve excess tension in the lines such as might be the case if the line were running round a winch drum in which case extra line can be fed into the system. Because the support structure was not designed to withstand significant loading this limits the operation of the system to a very small kite which causes further problems. Principally, a small kite is very sensitive to turbulence within the air and its rotational speed and flight speed is very fast making it a rather challenging candidate for testing an evolved control system. Additionally such a small kite presents a very small visual target rendering optical-based tracking methods difficult to implement using cost effective hardware. The physics of the arm are also such that the arm is in fact not guaranteed to be pointing at the kite at any given moment and therefore the arm angle is not a reliable signal of kite position.

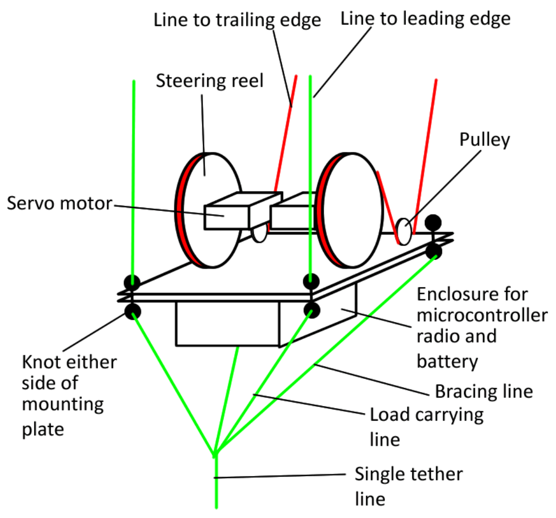
The lessons learned from this platform therefore are that a system that offers the potential to either be operable at ground level **or** mounted on/under the kite is preferable. A system whose dynamics are least likely to impact the steering of the kite in operation is also preferred, the system itself must not have designed limits that impede the motion of the kite **or** affect the efficacy of steering any more than absolutely necessary. These lessons we have tried to take on board when building subsequent control units.

8.2.2 Second robot

The second and subsequent robots were designed and built by the author after Bill Bigge or any other technical assistance became unavailable for this purpose.

8.2.2.1 Robot 2: Hardware configuration and mode of operation

Not only did we have to solve the design problems of the initial concept, but there also had to be a mechanism for computer control of the device, ideally that would allow both hardwired and wireless communication. For the second robot therefore, we moved to a pod or gondola configuration which is a small and lightweight frame that supports one or more servo motors, small reels, the necessary electronics and a power supply. The pod can be maintained at or close to ground level or conversely be hung by the flying lines between 1m and 5m under the kite, allowing a single line to attach the system to a ground attachment point or winch. A basic schematic of the initial design is shown below in Figure 8.10.



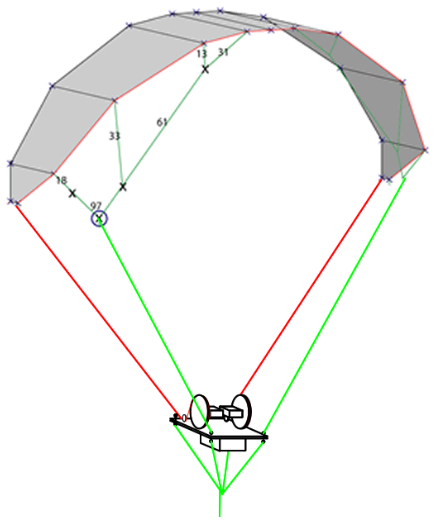


Figure 8.10 Simplified schematic of robot 2 (left) demonstrating the transition from single to multiple lines. The manner of attachment to the wing is also shown (right)

One of the innovative steps of this device was to effectively ‘float’ the pod on the power lines by running them through the device itself, locking the position of the device relative to the power lines simply by knotting the lines above and below the main plate of the unit. The only alternative to this configuration would be to attach the power lines at the bottom and top of the unit meaning that the structure would have to withstand the full load upon leading edge lines and hence suffer the gain in weight that the necessary additional strength would entail. As can be seen, the control bar has been replaced by two actuation wheels around which the steering lines are wrapped, the line is forced around a pulley as it leaves the wheel. The pulley is intended to control the angle at which the line is fed onto the main actuation wheel and prevent the line being pulled out of the actuation wheel groove. Contrarotating the wheels will introduce a difference in length of the trailing edge lines and thus result in a steering effect, and rotating wheels in the same direction will result in shortening or lengthening of the steering lines relative to the front bridle lines resulting in a change of attack angle of the kite.

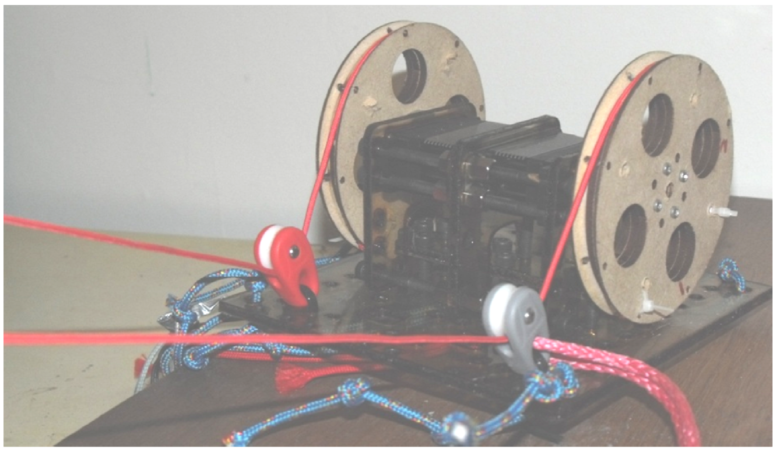


Figure 8.11 A photo of the second robot controller from behind, showing the path of the steering lines from actuator wheel through a set of blocks

The mounting plate and servo supports are made from laser-cut 4 mm polycarbonate sheet, and the wheels are made from three sheets of 3 mm laser-cut wooden chipboard. The outer two layers of the wheels have had a bevel cut into them on a desktop lathe in order to guide the lines towards the centre of the wheel. The servos that we used are HiTec HSR-5990TGs, the most powerful servos in the HiTec range with a stall torque of 30kg.cm which means that as the wheel radius is 4 cm, the servos are able to cause the wheel to turn in the desired direction until the force on the line exceeds 7.5 kg or 75 N. The range of line length difference for each wheel is 12.56cm because the maximum rotation angle of these servos is 180°. Therefore the difference in line length between extremes of steering is 25.12 cm, which is approximately half the length of a typical commercial kite control bar. These servos are powered with a 7.6 V nickel metal hydride battery and controlled via a pulse width modulation signal that is provided by an Arduino board (see Figure 8.12) powered by an Atmel Atmega 168 microcontroller.

We use the Arduino Xbee shield which allows an Xbee radio to interface with the Arduino, this provides effectively a transparent serial connection between a computer and the Arduino board allowing comparatively low latency communication at 57600 bps between the computer and board. The Arduino itself was powered with a small 5V lithium ion battery and assembled combination of Arduino, shield, radio and battery is shown in Figure 8.12. For the few tests that were done with this unit, we simply placed the electronics and power supply in a plastic enclosure with some padding material and attached it to the underside of the actuator mounting plate.

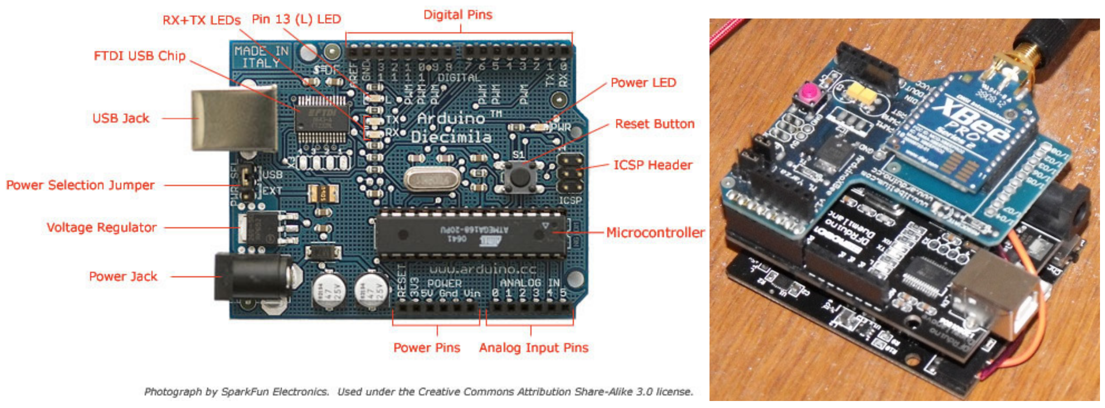


Figure 8.12 An annotated diagram of the Arduino Diecimila, four of the digital pins are capable of PWM servo control. Pins 0 and 1 are capable of TTL level RS232 serial communication. The board is programmed from a PC using the USB jack. The arduino assembled with XBee shield, XBee radio and Lithium battery is also shown (right).

8.2.2.2 Robot 2: Operational performance assessment

This device was intended to control substantially larger kites than the first device, primarily so that they would generate sufficient lift to support the device under the kite. However, due to the importance of flight stability and the need to be able to track the kite with optical methods, moving to larger kites was obligatory. Therefore we needed to assess whether the actuator range of 25 cm was sufficient for this larger class of kites. We tested this system on two Flexifoil Atom08 kites of 3m^2 and 5m^2 respectively. The device worked adequately on both kites however it was felt that the steering range provided was not quite sufficient to achieve the full range of steering required for the eventual use of a larger kite. Additionally, it was barely possible to power or depower the kite because of the effective reduction in steering that was caused. A final issue with the design of this control unit was that it was too easy for the line to come off the wheel, because there was no structure in place to effectively trap line onto the wheel. Once the line had come off the wheel, it would become tightly wrapped around the output shaft of the servo, a failure which was effectively catastrophic, resulting in the inevitable crash of the kite.

The lessons learned from this device were clear: a greater steering range would be required, stronger servos would likely be needed for yet larger kites and some structure would need to be implemented that would keep the line on the wheel. Finally, some enclosure of the wheels would need to be provided to prevent forces when crashing from being transmitted to the servo causing extensive damage. However we did show that the concept of a pod was a workable one and the weight of the pod did not seem to noticeably impinge on the flight of the kite which was encouraging.

8.2.3 Third robot

The third robot was designed in such a way as to apply the necessary changes to the previous system, whilst retaining its successful elements. This third robot incremented through two design phases which will be described separately here.

8.2.3.1 Robot 3, Phase 1: Hardware configuration and mode of operation

One key development was a change of servo motor. Because there was no motor in the HiTec range more powerful or with the greater angular operating range than the model we used on the second robot, we used the Robotis Dynamixel RX-64 which has an angular operating range of 300° and over double the stall torque at 77 kg.com, which means that the servo would still retain control authority over the steering line up to a line load of 192.5 N when using a 4 cm diameter wheel.

As might be expected its power demand is considerably higher than that of the HiTec servo, requiring between 12 and 21V and currents of up to 1.2A. These requirements necessitated the use of a lithium polymer battery in order to provide the necessary power without adding excessive weight and volume to the pod controller. An additional feature of this particular servo is that it communicates with its controller via an RS 485 communications protocol rather than the more common pulse width modulation (PWM). This allows a greatly enhanced level of communication with each servo, whereby internal settings and limits can be managed over the RS485 link via an embedded microprocessor on board the servo. For example it is possible to control the RX-64 by torque, speed or position control and to request the device to provide data on current, voltage, position, load and temperature etc. The downside to this functionality is the requirement to program the Arduino microprocessor with an entire communication protocol to incorporate details of the proprietary communications used by the device. This included the memory table addresses of each parameter to be read or written, the message format such as opening and closing signals, addressing and the error-checking scheme employed on board the servo. Because no open source library was available for control of these dynamixel servos, one had to be written by the author.

Additionally, because the Arduino microprocessor board lacks hardware with which to communicate with RS 485, it was necessary to use an integrated circuit (IC) transceiver chip, the MAX485, and create a circuit with the appropriate electronics to supply the chip with power at the correct voltage and the appropriate inputs and outputs. The schematic from the IC's datasheet is shown in Figure 8.13 for reference.

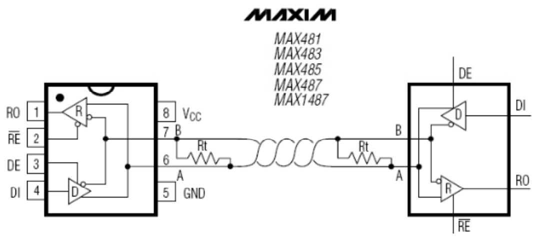


Figure 8.13 Schematic showing configuration for the Max 485 transceiver, lines B and A carry the RS 485 signal, note the requirement for a resistor between signal lines and the twisting of the cable

Because RS 485 communication between the servo and this transceiver chip is not full duplex, we must alternate between reading and writing modes by switching the transceiver chip between receive and transmit modes via a voltage signal from the Arduino board, allowing for an appropriate time delay for the chip to change mode. This fact coupled with the desire to not impede the speed of communication between the Arduino and servos and the Arduino and the controlling PC over radio link meant that we had to use the more capable Arduino Mega board. This board has multiple hardware serial ports rather than the simpler Diecimila or Duemilenove models. In this way the radio serial link between the computer and the Arduino board, and the serial link between the Arduino and the servos can be operated concurrently, thereby minimising lag times in achieving a desired servo response.

Because of the fragility of the wooden wheels in the initial robot, we opted for polycarbonate wheels in this version. We enclosed the battery, electronics and servos within an off-the-shelf ABS plastic box, the servos were mounted on the internal side of the lid with their output shaft passing through a hole laser cut into the lid to allow coupling of the servo output shaft and the actuator wheels. Because the laser cut polycarbonate is a high friction surface we mounted the steering lines simply by a friction fit loop, rather than the small cable tie that had been employed on the wooden wheels which was not a robust solution. The wheels were protected from impact with the ground by a 3 mm polycarbonate sheet raised above the lid with steel spacers, see Figure 8.14. In the first build we hoped that the protective lid coupled with the physical presence of the spacers would be enough to guide the steering line onto the wheel at all times.

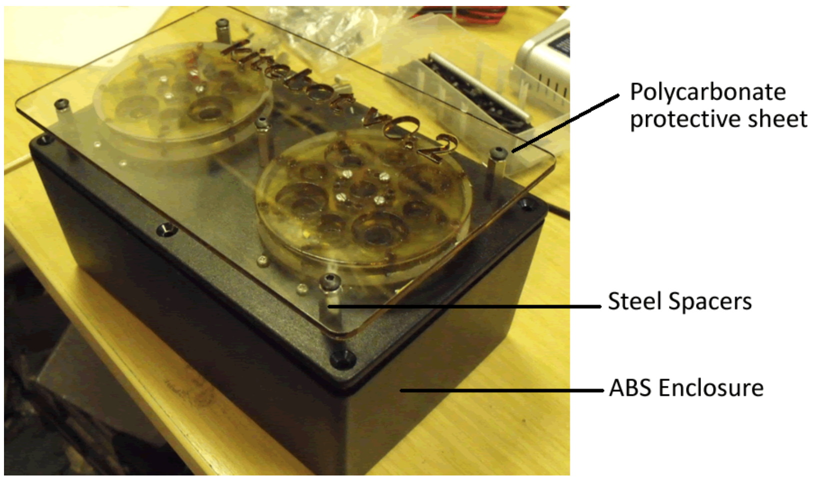


Figure 8.14 Robot 3 in first phase of development before addition of line rigging

The system was attached to the kite in a similar manner to the first robot, with the actuator wheels connected to the trailing edge lines. The single tether line split into a 'Y' bridle c.20cm under the control pod and passed through the pod, with knots above and below the exit and entrance points respectively. The line rigging can be seen in Figure 8.16.

8.2.3.2 Robot 3, Phase 1: Operational performance assessment

This robot was tested on an 8m^2 Ozone Access ram-air kite which was chosen for its stability and its availability in a red/black colour scheme that facilitated easy tracking. However during testing in this initial phase the lines frequently slipped off the reel during operation proving that our hopes that only minimal management of the lines would be required ill-founded. Once this occurred, if any further actuation signals were provided, the line would be jammed severely between the lower surface of the reel and the lid of the enclosure. In addition it was found that even the nearly doubled range of steering was only just sufficient to control the kite which has extremely sluggish handling in comparison to the Flexifoil Atoms. This meant that there was insufficient spare range with which to change the attack angle of the kite, which was severely limiting because this kite has a tendency to stall in light winds.

It was also found that the electronics could easily be damaged on opening and closing the enclosure even with protective padding in place, for this reason in the second revision of this robot we would be required to remove all of the electronics and keep them in a secondary smaller enclosure.

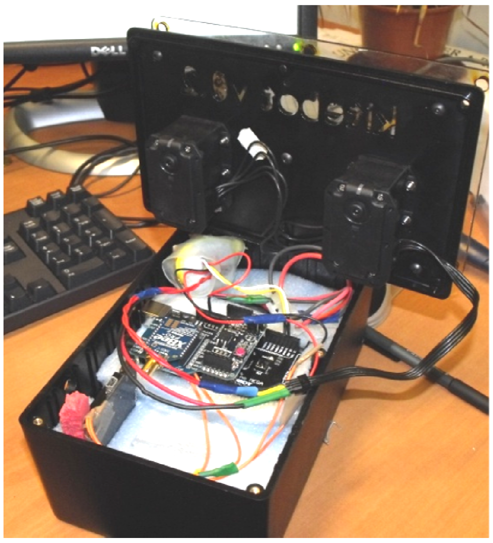


Figure 8.15 Robot 3 in first phase with servos visible mounted to box lid and electronics and battery visible before wiring

8.2.3.3 Robot 3, Phase 2: Hardware configuration and mode of operation

Figure 8.16 shows the final version of Robot 3 partly dismantled during decommissioning, visible is the separate control electronics box which also contains an independent battery for the electronics. The second aforementioned feature of this revision was the addition of four guides placed very close to each actuator wheel to prevent the steering lines from slipping off. In the first instance these were drilled manually and it was clear that the accuracy required to get the guides close to the wheel without impeding its rotation was almost impossible to achieve. A stopgap solution of adding tape to the guides in order to increase the diameter to the required dimensions proved ill advised because of the high friction of the surface. Additionally our positioning of the guides could potentially force the line to pass at a steep angle around the guide which could cause a jamming of the system by embedding the high tension line into the tape layer around the guides. Ultimately we passed lines around a supplementary guide pair with a free rotating brass bushing, an example of which is visible above the left wheel on Figure 8.16. This did ameliorate this problem however the positioning of the guides would have to be more carefully considered for the final robot version. After these tests it was clear that the ideal design would allow the line to exit the reel at a near tangential angle and not be forced to bend around any guides.

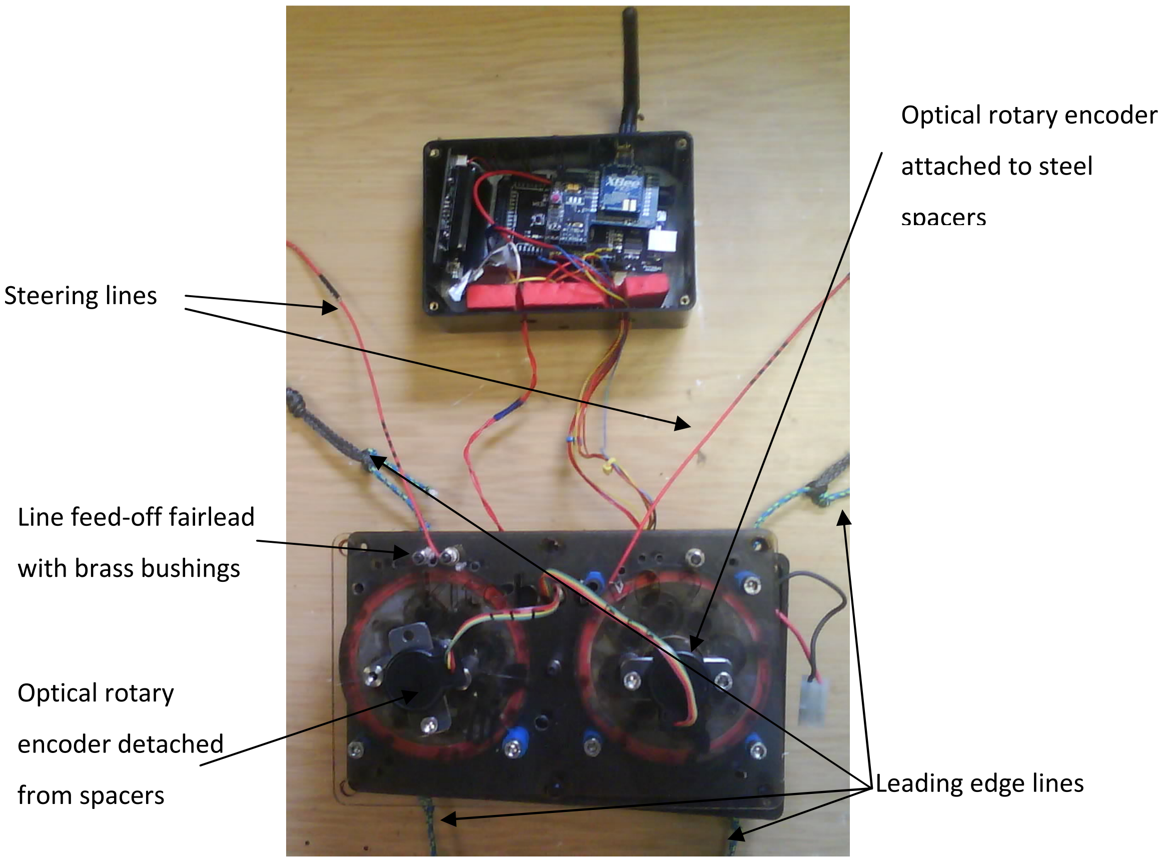

Optical rotary encoder attached to steel spacers
Steering lines
Line feed-off fairlead with brass bushings
Optical rotary encoder detached from spacers
Leading edge lines

Figure 8.16 Robot 3 in final phase of development, with additional optical encoders, line guides, bushings, rigging and separate electronics enclosure.

The other main revision we made before moving on from this development system was to allow multiple rotations of servo. This fact that multiple rotations can be configured by changing a setting on the servo microcontroller without any hardware modification is one of the advantageous features of the RX-64. However the drawback is that the position information becomes unavailable to the on-board microcontroller. This means that a separate mechanism for position control is required, we used a standard coupling in order to mate an optical quadrature encoder to each actuator wheel. The Arduino code is then interrupted every time one of the optical encoders communicates an incremental position change with a resolution of 128 ‘clicks’ per turn of each wheel, in order to keep track of the number of rotations of the wheel and hence the length of each steering line. The Arduino must provide a torque signal to the servos that will turn the wheel in the appropriate direction for the correct amount of time in order to reach the goal steering position. In this robot we accomplish this task with an open source PID algorithm (Beauregard, 2010) to which we provide the wheel positions as plant state values. We also provide the goal positions for each actuator wheel, and the PID algorithm generates an appropriate torque value which we communicate to each servo. We tuned the gains for the proportional, integral and differential portions of the algorithm manually to provide appropriate sensitivity to control signals without overshooting the goal position. One advantage of this approach is that the PID algorithm can be tuned to allow small deviations from the goal, providing some degree of compliance in the servo. The result of this compliance is that sudden large loads in steering lines caused by transient gusts in the wind will result in the reeling out of some of the steering lines, which will effectively lengthen the back lines, thus changing the attack angle such that the power developed in all lines is reduced. When the gusts stops, the slack fed into the system will be retracted. Such a situation will effectively smooth out the power developed by the kite, and limit the variability of the responsiveness of the kite to control actuation. Finally we also implemented a safety mode which would spool out all of the trailing edge lines of the reels if the radio link cut out for more than c.2 seconds or if an assigned button on the joystick were pressed.

8.2.3.4 Robot 2: Phase 2 Operational performance assessment

After the changes made to the robots in the second phase the performance was the highest level achieved until that point up to that point, the additional range provided by the ability to rotate each wheel multiple times, allowed a full range of depower enabling the attack angle of kite to be trimmed in flight by the operator. On tests with an 8 m^2 kite in winds of up to 12 miles an hour the robot was easily able to fly the kite dynamically throughout the whole wind window, generating enough force to pull the person on the ground who was acting as an anchor. Additionally the safety mode that we introduced to spool all the line off the reel to depower the wing completely functioned successfully. When triggered the kite depowered consistently enough and fast enough to effectively remove all tension from the tether line, and the kite would drift to the ground within a matter of seconds.

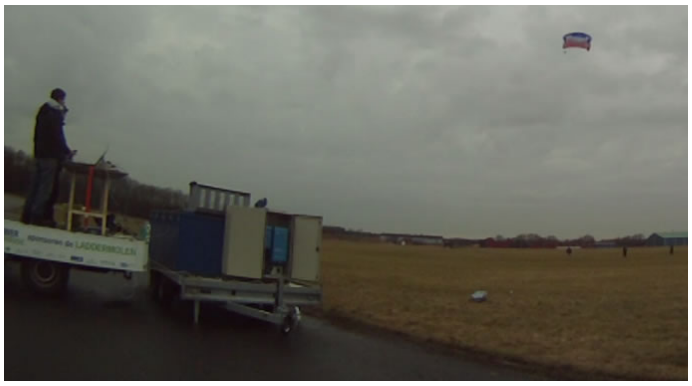


Figure 8.17 Testing of robot 3 controlling a 24m^2 Peter Lynn Synergy kite in Holland in collaboration with kite energy researchers at TUDelft

For the purposes of testing evolved neurocontrollers, the servos, line actuation method, speed and range, radio communication protocols, electronic design, pod/kite attachment configuration all performed acceptably. However it was felt that the PID implementation had effectively reduced the power of the servos by only using the maximum torque of the servo when there were very large proportional deviations between the target and current position. Some tuning of the gains was able to improve this but only at the cost of reducing the compliance effect and even introducing some jitter into the servo positioning. Additionally, some elements of the physical design of the system were sub optimal. For example the quadrature encoders were extremely exposed outside of the robot enclosure which had led to their attachment points to the protective shield covering the wheels starting to break. The line path was sub optimal in that the line was forced to bend around a steep angle, unnecessarily increasing the load on the servos. The tolerances of the line guides proximity to the actuation wheels was unacceptably poor and the battery compartment was not easily accessible, requiring the robot to be dismantled simply to change the battery pack for a freshly charged one. The design of the raised cover of the wheels was also fragile and although the unit did survive several crashes, the design was suboptimal in that if it took a significant impact at certain angles, the cover would shear off. The design goals for the final robot would be to correct these physical design problems and the electronics and line actuation mechanisms would be retained with only small revisions.

8.2.4 Fourth robot

8.2.4.1 Robot 4, Hardware configuration and mode of operation

As mentioned above, the fourth robot retained all of the electronics and the servo motors from the third robot but with a complete redesign of the physical enclosure. Because it was not possible to obtain a box of exactly the correct dimensions, we created a custom enclosure using three layers of laser cut 4 mm acrylic sheeting, see Figure 8.18.

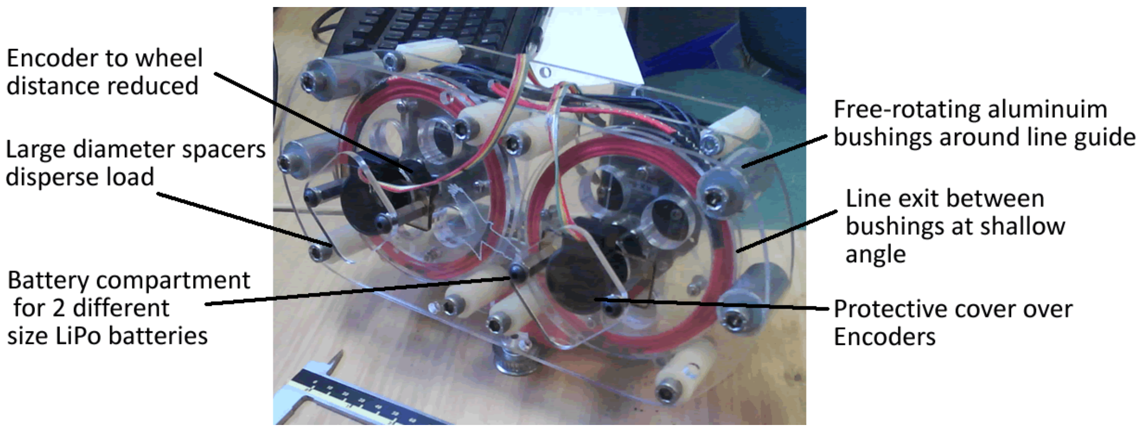

Encoder to wheel distance reduced
Large diameter spacers disperse load
Battery compartment for 2 different size LiPo batteries
Free-rotating aluminuim bushings around line guide
Line exit between bushings at shallow angle
Protective cover over Encoders

Figure 8.18 Robot 4 with the main changes highlighted

The structural elements holding the sheets together are acrylic spacers with threaded brass inserts known as transipillars, these specialist parts are unusually wide relative to the diameter of the threaded insert, meaning that they disperse the load more effectively than narrow metal spacers which cause high stresses around the bolts. The holes for the spacers are laser cut to bring the distance between the guides/bushings and the actuator wheel down to c.0.5mm, preventing the line ever leaving the wheel without impeding wheel rotation. In this design we intended the line to leave between two of these pillars such that in nearly all modes of operation the line would barely touch the guides, and would never be bent round them at a steep angle. To minimise the friction around these guides as much as possible, we added a custom cut aluminium bushing which fits precisely over the spacer and is lubricated so that it rotates freely. We also reduced the height of the coupling between the quadrature encoder and the actuation wheel by removing the lower half of the coupling and cutting inserts into the upper layer of the wheel to which the half coupling could mount directly. We also added protective shielding above the quadrature encoders. Finally we created two possible spaces for lithium polymer batteries of different dimensions which could be sealed by reusable Velcro loops allowing for rapid change of batteries. Due to the apparent limitations of the PID algorithm for servo control we re-implemented the servos' native position control algorithm on the Arduino. This can be quickly summarised in Figure 8.19 which shows how clockwise and counterclockwise torque signals are generated given a difference from a goal. The essence of the approach is that there are three regimes; a margin of error around the goal position in which zero force is exerted to move the servo toward the goal, a compliant region beyond the margin of error in which the force exerted linearly increases with distance from the goal, and finally a full power region. On the Dynamixel itself, the margin region size, the torque compliance slope and the minimum torque or punch are all redefineable, and we also allowed these parameters to be modifiable in our Arduino re-implementation.

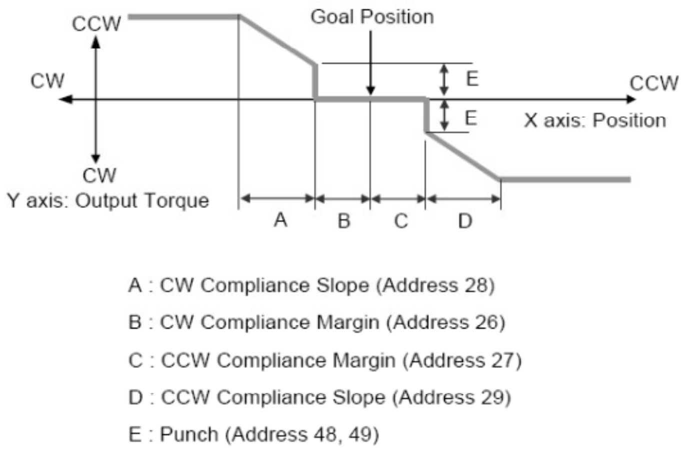

CCW
Goal Position
CW
E
CCW
E
X axis: Position
CW
Y axis: Output Torque
A
B
C
D
A : CW Compliance Slope (Address 28)
B : CW Compliance Margin (Address 26)
C : CCW Compliance Margin (Address 27)
D : CCW Compliance Slope (Address 29)
E : Punch (Address 48, 49)

Figure 8.19 Diagram from the RX-64 manual showing the Dynamixel's native servo position control algorithm

The Dynamixel servos are designed to cut all torque if a certain temperature is exceeded, this did occur in one test where there was intermittent power to the Arduino and hence the current reference position was being reset, this led to the trailing edge steering lines being constantly retracted until a knot in the line became jammed at the reel entry point. Unfortunately in this instance, the servo did not cut off in time and the motor was destroyed. To prevent this event reoccurring, we fitted this knot with a pair of magnets and the entry point to the control pod was fitted with a pair of micro reed-switches. If triggered, the Arduino would itself stop feeding torque commands to the servos in order to prevent any damage.

The last addition to this robot implementation was the deployment of a load cell to continuously measure the tension in the line, to be used either with the pod, or independently at ground level with a second Arduino to process the analog signal. We use an Omega Engineering LC220, S-type load cell, capable of providing tension load data up to 250 kg load, which is the approximate breaking strength of a single Dyneema line of the type we commonly test with. The load cell was initially calibrated on the bench in the setup shown in Figure 8.20 and was found to have a non-linear region in the first 10 kg of loading. Post-calibration this non-linearity was then corrected for in the data acquisition process prior to passing the value to the neural network.

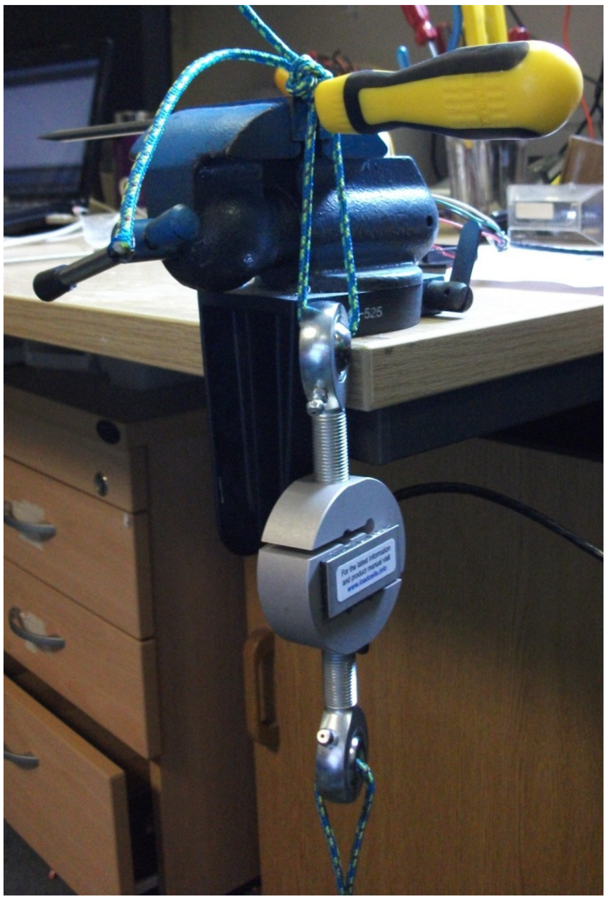


Figure 8.20 The load cell during bench calibration

To bring the signal into line with the normal 0 to 5 V sensitivity range of the Arduino's on-board analogue to digital converter we needed to implement an instrument amplifier. To this end we used an INA125 IC chip. The load cell requires a 10V regulated input voltage so we use a separate 12v NiMH battery which is down-regulated to 10v by the instrument amplifier.

8.2.4.2 Robot 4: Performance assessment

This robot was by far the most competent device yet tested. The line guide setup successfully kept the lines on the reel, the depower range of 50cm was easily sufficient to alter the attack angle and prevent the kite from stalling even in very light winds, and the trailing edge line maximum difference was easily capable of generating sufficient steering movements to cause the kite to respond as intended.

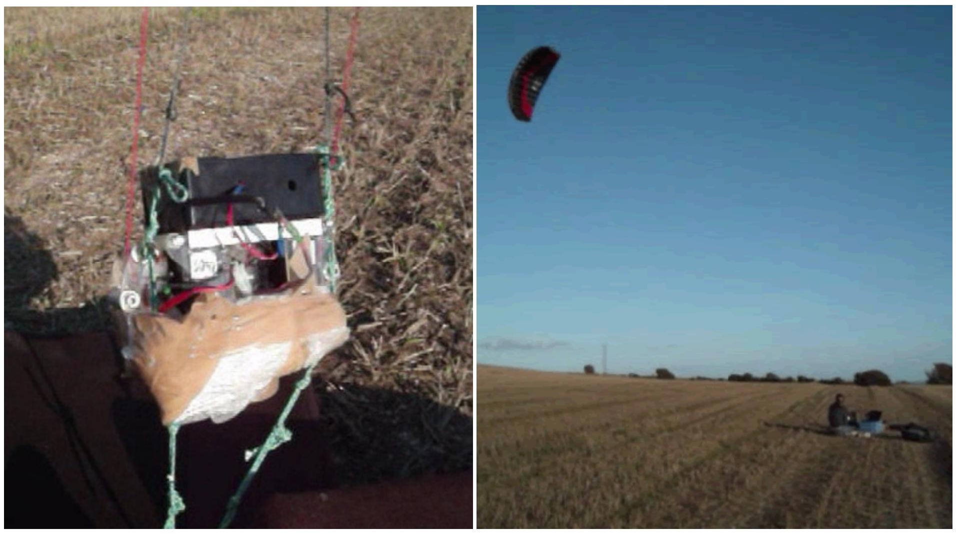


Figure 8.21 A test flight showing the pod being held in tension about 30cm above ground level (left). The pod being used to fly an 8m2 Ozone Access kite in looping trajectories with a human pilot (right).

Figure 8.21 shows the pod being used in a test controlling the Ozone Access kite in light wind of c. 5m/s. It can be clearly seen in the left half of the plot that the pod is lifted completely off the ground by the kite even in this light wind.

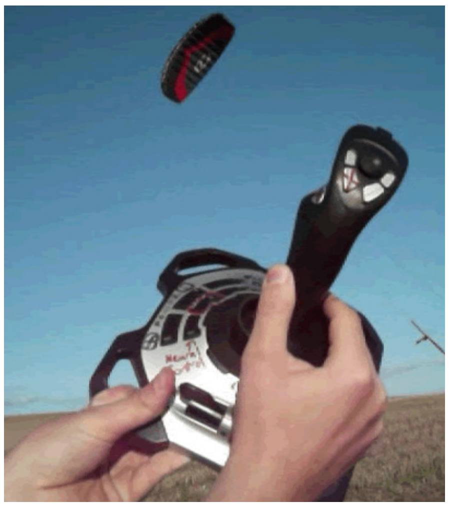


Figure 8.22 Photo showing joystick used for manual control during a flight test.

During manual control (see Figure 8.22), the joystick is moved left or right to change the relative lengths of the flying lines, the slider shown by my left thumb is used to change the AoA of the kite. The buttons are assigned for initialisation purposes and to hand over control to the neural network, or trigger complete depower for safety purposes.

Unfortunately, even with the more powerful servos used in this robot, the servos suffered fatal overheating or gearbox failure after a few minutes of flight control. These failures occurred even in light winds and rendered testing of the evolved neurocontroller impossible. Improved servos would be considerably heavier and would most likely require separate power electronics, also increasing system weight. The increased component weight would in turn require a stronger, probably metal flight control pod. The resulting system would likely be 3 -5 times heavier and require larger kites or higher wind speed. Due to the high line tensions developed as a result, it would likely be unwise to fly such a system on a fixed length tether and therefore a winch would be necessary to actively moderate the line tension. Alternatively, a pair of motors could be mounted to a structure at ground level being used to control a kite at a similar scale to the one used in the experiments described.

8.3 Summary

In this chapter, we have presented the elements of a system intended to track and provide the means to steer a kite via electromechanical means. We described our camera tracking algorithm which although basic, is effective in tracking the kite position and orientation unless the sun is present in the image. We described some preliminary experiments with IUs that suggest they could be effective in tracking at least the orientation of a kite. The remainder of the chapter then described a sequence of four robotic kite control systems. We demonstrated that systems that can be operated both at ground level and suspended from the kite in flight are achievable and offer some advantages over purely ground based systems. From the progression between the units it is clear that key to the development of a successful kite control pod was managing the flying lines properly such that their path was not impeded, yet they were always kept from leaving their reels except as intended. Secondly if the control pod is mounted on the tense flying lines without much of the force being loaded directly onto the pod, it could be designed in a very lightweight manner. Finally the strength and position control algorithm was crucial and further experimentation in compliance would be likely to prove fruitful.

Ultimately, the servos in the pod were not able to cope with the loads generated by the kite and this prevented testing of the neurocontrollers outside of the simulation environment. A system with motors an order of magnitude more powerful is in development, and the neurocontrollers will be deployed on this system in future work.

9

Reality is that which, when you stop believing in it, doesn't go away.

Philip K Dick, I Hope I Shall Arrive Soon

Discussion

In this chapter we seek to provide an overview of this document in order to furnish the reader with a summary of the work performed. This also presents an opportunity to add some brief points of discussion further to those raised in the earlier chapters themselves and to reiterate the novel contributions of this thesis with reference to the previous chapters. Finally we are able to exploit the perspective afforded to us by the successes and failures of the experiments undertaken here and address the suitability of the evolutionary robotics approach for highly applied real-world problems such as this.

9.1 Kite control as a problem for evolutionary robotics

In chapter 2 we sought to provide an overview of evolutionary robotics, starting by touching on traditional approaches to artificial neural networks. We described how common training methods could produce powerful classifiers but had limitations for the creation of robot controllers due to the need for lengthy periods of training and potentially the knowledge of exactly what constitutes a good or correct behaviour when such knowledge might be unobtainable. This was followed with a suggested explanation of why the use of evolution is particularly apt as a method of evolving neurocontrollers for real or simulated robotic agents. More specifically it was suggested that by assessing the success or failure of a neural network controlled robot's behaviour in a real or simulated environment over a longer period it is possible to create robot controllers through a process that itself determines the best manner in which a given task can be accomplished.

We then went on to describe the use of evolutionary robotics to create neurocontrollers for robot control tasks in simulation both as a tool to explore brain body environment couplings as a dynamical system and as a tool with which to create adaptive control systems. We highlighted some of the issues in an applied evolutionary robotics approach such as how best to encode parameters within the genome, whether modularity should be imposed or allowed to emerge within a monolithic network, and how parameter selection and evolutionary management can be especially important in difficult tasks with complex dynamics or requiring multiple behaviours. Subsequently we addressed the particular concerns when using evolutionary robotics techniques for real-world control tasks highlighting in particular the considerations that need to be taken when evolving neurocontrollers in simulation to be used on real world hardware systems. We continued by reviewing some of the available literature where ER was used to develop neural networks for flight control. Finally we introduced the background and motivation for the current interest in airborne wind energy systems, summarising the basics of kite flight and approaches in the literature to the modelling and control of the systems in simulation and the development of hardware tools with which to control real world kite systems.

This introduction set the context for chapter 3 in which we discussed how the specific attributes of the kite control problem fitted within the evolutionary robotics paradigm. In particular we highlighted the importance of the particular instantiation of the kite as a flexible tethered wing within a changeable environment. We highlighted the rich physical dynamics of the system due to the nontrivial coupling of the wing deformation, steering input and the environmental conditions. It was noted that the wide range of operating conditions would mean that any controller would have to be able to generalise to an unusually broad set of system dynamics. We then assessed the impact of some key attributes of the kite control problem that are lacking from many archetypal evolutionary robotics tasks. These included the instability of the physical dynamics of the system and the rapid timescale upon which the system operates including a very short time to complete failure. A crucial attribute was the difficulty in decomposing the task given poor expected quality of information regarding the state of the system at a given time and the difficulty in creating an accurate model of such a variable system in a variable environment. We described how recurrent neural networks were used successfully in one ER deployment on an unstable system and how velocity inputs could reduce the complexity of the task for a neural network. We also reiterated the necessity to evolve neurocontrollers in a simulation environment that would be successful on first deployment onto a real world system as opposed to hardware evolution or online learning methods, due to the short time to catastrophic failure and the potential for damage to hardware. With regard to the decomposability of the task, we stressed how the wide operational envelope and the low likelihood that any given simulation model would be accurate, would render an otherwise attractive modularisation of the task difficult and would demand particular emphasis on the transferability of intact robot control behaviours across the reality gap.

In the remainder of the chapter we considered how the attributes of the kite control task resembled or differed from the classic implementations of Jakobi's radical envelope of noise hypothesis. Our preliminary assessment was that the extreme minimal models proposed by Jakobi would be inappropriate for this task because of the wide range of dynamics and large environmental variability that could be expected. Because of the importance of physical dynamics in this task it seemed more appropriate to use a simplified physical model in order to expose evolving neurocontrollers to some semblance of the dynamics which they would be faced with in reality. The importance of rapid response due to the fast dynamics of the task and it's instability also seemed to demand some consideration of the actuator dynamics and delays involved in communication and sensing the environment. However we did conclude that the core of Jacobi's approach would need to be heeded and that any model would have to be mistrusted and noise systematically applied to simulation parameters. Deploying noise at the level required would add considerable difficulty to the evolutionary process and the management of this difficulty would be addressed in the experimental work in later chapters.

9.2 Could ER produce appropriate flight control strategies?

After the introduction to the problem in chapter 2 and the analysis of the key challenges to be faced in chapter 3, we moved on to question whether evolutionary robotics was capable of producing valid flight control behaviours for the generation phase of the two-phase cycle of a pumping kite generation system. We began by introducing a novel multi-body multiple panel kite simulation method that whilst clearly simplified also sought to remedy some of the key deficiencies of prior models in the literature. We described how the numerical modelling approach allowed a simple way of applying a complete range of lift and drag coefficients as the angle of attack of the wing varied. We also showed how splitting the kite into several panels allowed the local speed and angle of attack of different portions of the wing to be taken into account which is particularly crucial for the kind of acrobatic looping trajectories commonly seen in kite flight. Finally we described how by manually designing a system of constraints that preferentially reinforced the leading edge of the kite, the kite was capable of deformation under load in a manner at least analogous to leading edge inflatable sport kites.

We demonstrated for the first time in the literature that it was possible to evolve discrete time recurrent neural networks that would fly a kite in an appropriate figure 8 trajectory that is similar to that obtained using more conventional optimisation methods. This flight control was accomplished without a full state estimation vector, only having access to relatively sparse information describing the kite's angular position in the sky, and line tensions. We demonstrated that these evolved neurocontrollers exhibited some robustness to changes in wind amplitude and direction of which the controller has no explicit knowledge. However the flight control performance of these networks was certainly not without flaws. Of particular concern was the poor generalisation capability of these networks which proved very fragile to changes in lengths of the lines if these were not varied during evolution. Of greater concern was that a fair proportion of these trials ending in crashes even at the evolved line length. Interestingly when evolved over a large range of line lengths, the controllers produced different flight trajectories depending on the length of the lines. Although some of these trajectories were highly appropriate, others were clearly suboptimal. Finally we described how continuous time recurrent neural networks evolved on the same task struggles to produce valid flight control behaviours and suggested that greater attention would have to be paid to the exact implementation and initialisation of the CTRNN if adequate performance was to be obtained using this class of network.

9.3 Is it possible to use ER as a tool to investigate longer term generator control behaviours?

We made it clear in chapter 2 that in high altitude wind energy conversion systems where the generator was on the ground, the system would have to be operated in two alternating phases.

In the first the kite would be flown in such a way as to maximise the tension in the line, the aerodynamic force generated by the kite would be harnessed to perform work at ground level by turning a reel and thus extending the flying lines. These lines would need to be retracted periodically in order to reset the system with the kite being flown in such a way as to minimise the line tension and therefore maximise the net gain of energy over the entire cycle. Because of the computational demands of simulating the kite's flight for long periods, we created an analogue task involving the capture of a spatially distributed and variable amplitude environmental resource by a situated simulated agent. This agent had to alternate between capturing and expending energy available in the environment. This simple simulation allowed us to address three crucial questions that had hitherto been neglected in the literature. These were: how to capture an environmental resource which could be of sufficient strength to damage the system, how to balance the time and energy costs of changing between modes, and whether switching behaviour could be adapted in environments where the strength of the resource could change on slow, medium or rapid timescales. Furthermore we sought to investigate a potentially important question from the ER perspective which is how best to evolve switching behaviours, either switching by simply crossing a resource intensity threshold, or by introducing an element of control by incorporating a coupling between agent instigated movement and the cyclical resource gathering behaviour.

This exercise also allows us to incorporate some changes regarding the implementation of the CTRNN namely the introduction of a velocity input for one of the key dynamic properties of the task. We were also able to make some changes to the management of the evolutionary process, specifically using multiple trials with highly variable starting conditions in order to preferentially select for strategies robust across a range of initial conditions. We also evolved the neurocontrollers using a shaping scheme that split the task into four stages of incremental difficulty. In terms of answering the question posed, we showed that it certainly was possible and useful to investigate longer term pumping kite generator behaviours using a simplified analogue task.

Some results were of immediate practical value, for instance the necessity to implement a shaping scheme in which sub components of the switching behaviour required were evolved sequentially. The use of multiple trials per scoring evaluation did appear to prevent the occurrence of the fragility and poor generalisation that we saw in the experiments of the previous chapter.

There were also some results that were interesting in themselves, for instance the failure of the evolutionary process to produce successful alternating charging and discharging behaviours when the boundary between charging and discharging was determined purely by the position of the agent on a spatial resource gradient. This contrasted sharply with the experiments in which a movement cost was introduced which coupled the agent's ongoing behaviour to the performance of the task. It was notable how the agents evolved under the second regime were able to successfully complete the task when the rules were changed to the former scheme.

The successful agents demonstrated adaptivity to environmental change at a number of timescales; in between each trial, slow linear incremental change within a trial, and rapid non-linear discontinuous change within a trial. This would be an important characteristic of any successful pumping kite generation system. Notable was the ability of agents evolved under the latter most difficult environmental change regime to switch between modes driven by the environment. Also worthy of note was the fact that this apparently superior adaptive behaviour whilst allowing the agent to carry on performing the task without damage, did reduce the performance of the agent over shorter timescales. This highlighted the cost of a conservative control strategy and we suggested that it might be appropriate to switch between controllers operating at different levels of conservatism depending on environmental conditions as determined by a set of predetermined metrics.

9.4 How important is it to choose the most appropriate GA and NN parameters?

In chapter 6, partly motivated by the poor generalisation of the discrete time recurrent neural networks evolved on the flight control task in chapter four, and also by the difficulty in evolving valid continuous time recurrent neural networks in the same task, we investigated changing some key neural network parameters and elements of the genetic algorithm. The high-level conclusion was that it was indeed worthwhile to investigate the use issues not only because some of them had a significant effect on the performance of the resulting neurocontrollers but also because some of the results ran contrary to existing work in the literature. This highlighted the importance of ascertaining whether a particular approach is applicable or appropriate to the problem at hand and the techniques that is being used to solve it.

Particularly notable was the reduced performance when a spatial heuristic was implemented which ran counter to the results of Spector (Spector, 2005). However Spector's results derived from genetic programming problem not the evolution of a neurocontroller for a dynamical control task. This being so, it is not surprising that there may be some performance differences, however such geographically distributed populations are often used in genetic algorithms in the evolution of neural networks and yet our data here suggests that it might not be beneficial in all cases and might even hinder evolution or lower performance. A second case in which significant differences were evident in comparison to previous work was that of the effect of the number of neurons in the network on the performance in the task. In Beer's work increases in performance were noted as the network size increased but the gains were less pronounced as the network size could larger (R. D. Beer et al., 1999). In our case increases in the size of the smallest networks produced little effect and yet the addition of more neurons appeared to continue to confer performance advantages up to 10 neurons which was the highest number of neurons that we tested. Once again we do not claim that this somehow invalidates the prior work, rather the point is that trends or rules of thumb that are found to apply to a given task or technique might not necessarily be transferable to significantly different tasks or techniques.

Other interesting results arising from this chapter included a striking increase in performance when a minimal amount of noise was applied to the biases of each neuron. This increase in performance was retained even when the noise was removed and the fact that the noise was applied in a probabilistic manner during the evolutionary process meant that there was no dependence of the controllers on noise. Notable in another set of experiments was the wider spread of both average individual performance per generation and best performance per generation when the neuron weights scaled in the range [-4 4] as compared to [-10 10], a result which was likely to relate to the saturation or maintenance within the dynamic band of individual neurons in the latter and former case respectively. Once again it is not necessarily a generic result applicable to other work but bound in with the number and amplitude of the expected inputs to the network.

Finally and also relating to the transfer functions of each neuron, we noted that networks using the tanh function unexpectedly underperformed when compared to networks using the sigmoid function, a result which contrasted with other work. Concurring with that same study however we demonstrated that networks where the transfer function slope of each neuron was individually evolvable performed better than networks where this parameter was not available to evolution. We showed that the best performance obtained overall was from an evolutionary run in which the tanh activation function was combined with evolvable transfer function slopes. As performance was our main motivation for carrying out this work, simply obtaining varying levels of performance was a satisfactory outcome, however these results do raise interesting questions that might warrant further attention preferably in a manner that links tasks with different properties, inputs of different ranges and assesses where network neurons are operating within the dynamic range. Realistically this might have to be performed in small networks in tasks with less complex physics in order to render the networks more amenable to analysis.

9.5 Looping flight and stabilisation behaviours, GA meta-management and multi-level models

In chapter 7 we evolved the complete control behaviour for the generation phase in which a neurocontroller is required to fly the kite in repeated looping trajectories whilst the flying lines are being reeled out at a range of speeds. The neurocontrollers were capable of performing this behaviour across a moderate range of environmental variation. The controllers exhibited an interesting fragility to changes in line length, highlighting the importance of this input in this task where the dynamics of line length take particular importance.

It was clear that a more aggressive deployment of task variability was going to be necessary to maximise the probability of a neurocontroller's flight control behaviour being successfully transferred to a real world system, especially given the known inadequacy of the kite physics model. We described a fine grained task difficulty shaping algorithm in which relatively small increments in the variation of a comparatively large number of parameters were deployed. Learning from our experiences in the prior work we used multiple trials per evaluation, provided the network with velocity inputs and varied the starting conditions of each trial. We also varied the model used to describe the kite during evolution, and described two more complex models to be used for validation purposes. We showed that it was possible to evolve a zenith stabilisation flight control behaviour where the neurocontroller would fly the kite at the zenith and maintain its position there regardless of the starting location or orientation of the kite. After progressing through the difficulty stages of the task, evolved neurocontrollers were able to perform this task when faced with an exceptionally large range of parameter variation. We demonstrated how populations seeded with a hand coded neural network yielded more successful evolutionary runs, and also showed how unseeded networks produced more successful evolutionary runs than unseeded networks in which there was not a set of velocity inputs provided. Crucially, the best neurocontrollers were able to generalise this behaviour to a model that had not been used during the evolutionary process, even though that model was somewhat unstable. These results are promising and the generality and robustness of the resulting controllers indicate that the fine grained shaping technique would have wider applicability in evolutionary robotics outside of controlling kites.

Finally we demonstrated that our GA implementation was capable of generating sets of constraints such as to improve the replication of a series of measured deformations of a real kite under load. The resulting kite model could be used in a secondary validation step, for extended evolution, or the technique could be used to replicate expected failures and to assess the ability of the neurocontroller to retain control of the compromised wing.

9.6 Hardware and expectation of transference

In chapter 8, we described the process of hardware development and the visual tracking algorithm used to determine the position and orientation of the kite. Whilst the final iteration of the kite control pod is more than competent, it is clear that multiple iterations were necessary to converge on a suitable design. The resulting kite controller can be used at ground level or flown with the kite, and is exceptionally light for the force that it is capable of exerting. The visual tracking algorithm whilst adequate suffered from glare when the sun was in shot which effectively rendered it impossible to track the kite if the sun was in the wind window. In the longer term it would be advisable to have some combination of kite tracking systems that could measure the line angle directly, track the kite visually and measure the kites position and orientation directly using an on-board IMU and GPS system. If a better camera were available it would also seem advisable to use the colour of the wing and as part of the blob identification algorithm. Migrating to an open framework such as Open CV might also be advisable, being implemented in C++ this should also run considerably faster than an equivalent Matlab package.

We tried to pre-empt failure modes wherever possible and implemented a safety mode that would be activated automatically in case of communication failure. In future testing we will also have at least one specifically rated weak link in the line that will remove all tension from three of the four flying lines and hence limit the maximum force that the kite is capable of exerting through the flying lines. It is clear that operating a system with a battery powered pod would not be feasible in a commercial system. It would therefore be necessary to have some mechanism travelling up the line to deliver fresh batteries to the pod, or to transmit power up to flying lines although this would render the line far more expensive. One way of eaking out the power of a battery would be to implement a locking mechanism in the pod. With such a mechanism a hard steering movement that would cause a near circular looping flight pattern would be physically locked in, after which the servos could be powered off to reduce battery consumption. The tightness of the turning angle at which the kite is looping can then be adjusted by management of the line tension via the winch speed. Realistically such an option could only be implemented when the kite is being flown at relatively long line lengths.

Given the lengths taken in the previous chapter to ensure the successful transfer of the zenith stabilisation behaviour from simulation to reality it could be expected that in the middle of the range of the intended operating conditions the neurocontroller behaviour would in fact transfer. Presumably, the further the task parameters or dynamics moved from evolved condition, the more difficulty the evolved neurocontrollers would have. There are certainly some details known to be present in reality that were not implemented in the simulation. For example the servo speed in simulation was always the maximum value, however in reality as the servo becomes loaded the travel time between extremes of range increases dramatically. This might create an understeer situation at the top of the intended wind range. Equally, at the lower end of the wind range, kites can stall unpredictably and ram air parafoil type kites without an airbeam skeleton will fold, especially at the edge of the wind window, in a way that cannot currently be replicated in simulation. We could expect failures of the neurocontroller in these situations if methods are not put in place to avoid these eventualities.

9.7 How appropriate is ER for this application?

Hopefully we have demonstrated to the reader that a systematic and thoughtful implementation of the evolutionary robotics paradigm can be used to produce neurocontrollers for the two phases of operation of a pumping lift mode kite wind energy system and also be used to investigate the optimisation of longer term behaviours in changing environmental conditions. However it is worth asking the question; is this an appropriate application of evolutionary robotics?

If we were to argue in the affirmative, we could point to the highly adaptive nature of the evolved neurocontrollers even to extreme changes in environmental conditions and to the physical parameters of the kite system. We could point to the ability of the neurocontrollers to perform both key tasks in a manner where two separate relatively simple neurocontrollers could be alternated and take control from one another producing the desired behaviours. We could also demonstrate how the neurocontrollers easily outperform a skilled human pilot, in the simulation environment at least. We could also demonstrate that the resulting neurocontroller is an exceptionally lightweight neural network that could be run easily on very cheap embedded systems. The ability of these networks to control these flexible wings with relatively incomplete and noisy data here is quite remarkable and there may be some room for applying such systems especially when unknown failures changed the flight dynamics of the wing whilst it is still in flight.

If we were to argue that such an approach was inappropriate, we might point to the opaque nature of the operation of these networks. Being comparatively large they are quite difficult to analyse, however by enforcing modularisation on networks it might be possible to describe their operation in a manner that could satisfy any regulatory constraints that may apply to these systems in future. One aspect of the deployment of evolution robotics that is certainly true is that it is not a magic wand. Expertise is not only required in order to avoid the standard pitfalls of evolutionary robotics but domain knowledge of the problem at hand is vital in order to prevent the task from being rendered unnecessarily difficult. Finally if wings that are highly rigid, and therefore able to be described analytically far more easily than flexible wings, are deployed in these systems, it would always be the first choice to deploy more conventional control methodologies unless they show some tangible performance limitations. However the prospect of using deliberately deformable, adaptive wing shapes is one of the most exciting aspects of using tethered wings in wind energy applications, and it would be a shame to rule out this avenue of development.

Regardless of the value of the evolved neurocontrollers in commercial hardware, such work is certainly valuable as an exploratory tool and given the extraordinary dynamic environment in which these systems are expected to operate, the robust and adaptive properties of evolved neurocontrollers remain attractive.

9.8 Directions for future research

Looking forwards, as always there is much useful work that could be performed. A flight testing programme using these neurocontrollers is underway and as already stated, we expect that the neurocontrollers will be able to produce the desired behaviour. Deliberately modularising the neural networks would render them far more amenable to analysis and make hand-coding of useful more advanced behaviours such as waypoint following much simpler. It would certainly be interesting to use evolutionary robotics techniques to investigate how such systems might exploit spatially discontinuous variation in the strength of wind that are known to be possible at the operating altitude of proposed systems. Equally it is conceivable that evolved neurocontrollers could exploit irregular spatial distributions of wind amplitude in certain portions of the wind window caused by the influence of the landscape or other kite systems. Not only might such advanced behaviours render these proposed systems more efficient, but the avoidance of extremely low or high wind speeds or excessively turbulent wind flow is likely to be obligatory if airborne wind energy systems are to be deployed safely and cost effectively. The apparent robustness of the evolved flight control behaviours to changes in wing geometry might render the ER approach particularly suitable to morphing wing systems or kite systems where the dihedral angle can be adjusted in flight in order to modify the wing properties according to environmental conditions or demands of the task.

10 Bibliography

Abeles, M. (1982). *Local cortical circuits : an electrophysiological study*. Berlin: Springer.

Archer, C. L., & Caldeira, K. (2009). Global Assessment of High-Altitude Wind Power. *Energies*, *2*(2), 307-319.

Arvind, D. K., & Bartosik, M. M. (2009a). Clustering of motion data from on-body wireless sensor networks for human-imitative walking in bipedal robots. *Advanced Robotics, 2009. ICAR 2009. International Conference on* (pp. 1–6). IEEE.

Arvind, D. K., & Bartosik, M. M. (2009b). Motion capture and classification for real-time interaction with a bipedal robot using on-body, fully wireless, motion capture specknets. *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication* (pp. 1087-1092).

Beauregard, B. (2010). Arduino PID Library.

Beer, R. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. *From animals to animats*.

Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, *3*(4), 469. ISAB.

Beer, R. D., Chiel, H. J., & Gallagher, J. C. (1999). Evolution and analysis of model CPGs for walking: II. General principles and individual variability. *Journal of computational neuroscience*, *7*(2), 119-47.

Bermudez i Badia, S., Pyk, P., & Verschure, P. F. M. J. (2007). A fly-locust based neuronal control system applied to an unmanned aerial vehicle: the invertebrate neuronal principles for course stabilization, altitude control and collision avoidance. *The International Journal of Robotics Research*, *26*(7), 759-772.

Beyer, H. (2002). Evolution strategies. *Natural computing*, 3-52.

Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, *18*(24), 10464-72.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press.

Bongard, J. (2008). Behavior chaining: Incremental behavior integration for evolutionary robotics. *Artificial Life*, *11*, 64.

Breuer, J., Ockels, W. J., & Luchsinger, R. H. (2007). An inflatable wing using the principle of Tensairity. *Proceedings 48th AIAA Structures, Structural Dynamics and Materials Conference* (pp. 1-12).

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, *2*(1), 14-23.

Brooks, R. (1991). Intelligence without reason. *Artificial intelligence: critical concepts*, *3*, 107–63.

Canale, M., Fagiano, L., & Ippolito, M. (2006). Control of tethered airfoils for a new class of wind energy generator. *45 th IEEE Conference on Decision and Control, San Diego (CA), USA*, 4020-4026.

Canale, M., Fagiano, L., & Milanese, M. (2009). KiteGen: A revolution in wind energy generation. *Energy*, *34*(3), 355-361.

Canale, M., Fagiano, L., Ippolito, M., & Milanese, M. (2006). Control of tethered airfoils for a new class of wind energy generator. *45 th IEEE Conference on Decision and Control, San Diego (CA), USA*.

Canale, M., Fagiano, L., Milanese, M., & Ippolito, M. (2007). KiteGen project: control as key technology for a quantum leap in wind energy generators. *American Control Conference, 2007. ACC'07* (pp. 3522–3528). IEEE.

Cantú-Paz, E. (1998). A survey of parallel genetic algorithms. *Parallel Computation*, *10*(2), 141–171.

Chiel, H J, Beer, R. D., & Gallagher, J. C. (1999). Evolution and analysis of model CPGs for walking: I. Dynamical modules. *Journal of computational neuroscience*, *7*(2), 99-118.

Chorley, P., & Seth, A. K. (2008). Closing the Sensory-Motor Loop on Dopamine Signalled Reinforcement Learning. *From Animats to Animals 10, The Tenth International Conference on the Simulation of Adaptive Behavior* (pp. 280-290).

Cliff, D., Harvey, I., & Husbands, P. (1992). Incremental Evolution of Neural Network Architectures for Adaptive Behaviour. *Univeristy of Sussex Cognitive Science Research Paper*, (256).

Collins, R. J., & Jefferson, D. R. (1991). Selection in massively parallel genetic algorithms. In Belew, Rick, & Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 249-256). San Mateo, California: Morgan Kaufmann.

De Nardi, R., & Houska, B. (2007). Ultraswarm: A further step towards a flock of miniature helicopters. *Swarm Robotics*, 116–128. Springer.

De Nardi, R., Togelius, J., Holland, O. E., & Lucas, S. M. (2006). Evolution of neural networks for helicopter control: Why modularity matters. *Proceedings of the IEEE Congress on Evolutionary Computation*.

Di Paolo, E. (2000). Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions. In J.-A. Meyer, D. Floreano, H. Roitblat, & S. Wilson (Eds.), *From Animals to Animats6: Proceedings of the 6th Intl Conf. on Simulation of Adaptive Behavior* (pp. 440-449). Cambridge, MA: MIT Press.

Doncieux, S., & Meyer, J. A. (2004). Evolving modular neural networks to solve challenging control problems. *Proceedings of the Fourth International ICSC Symposium on Engineering of Intelligent systems*.

Dorigo, M, & Columbetti, M. (1994). Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, (70), 331-370.

EPFL. (n.d.). goevo. Laboratory of Intelligent Systems, EPFL, Lausanne & Adaptive Intelligence Lab.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*(2), 179-211.

Fagiano, L., Milanese, M., Member, S., & Piga, D. (2010). High-Altitude Wind Power Generation. *Energy*, *25*(1), 168-180.

Fernandez-Leon, J., & Di Paolo, E. (2007). Neural uncertainty and sensorimotor robustness. In F. Almeida e Costa, L. M. Rocha, E. Costa, I. Harvey, & A. Coutinho (Eds.), (Vol. 4648, pp. 786-795). Berlin, Heidelberg: Springer Berlin Heidelberg.

Fernandez-Leon, J., & Di Paolo, E. (2008). Neural Noise Induces the Evolution of Robust Behaviour by Avoiding Non-functional Bifurcations. In M. Asada, J. C. T. Hallam, J.-A. Meyer, & J. Tani (Eds.), *From Animals to Animats 10. 10th International Conference on the Simulation of Adaptive Behavior* (Vol. 5040, pp. 32-41). Berlin, Heidelberg: Springer Berlin Heidelberg.

Fine, P., & Di Paolo, E. (2007). Adapting to your body. *Proceedings of the 9th European Conference on Artificial Life* (pp. 203-212).

Fine, P., Di Paolo, E., & Philippides, A. (2006). Spatially constrained networks and the evolution of modular control systems. *From Animals to Animats 9*, 546–557. Springer.

Floreano, D, & Mondada, F. (1996). Evolution of Plastic Neurocontrollers for Situated Agents. In P. Maes, M. Matarc, J.-A. Meyer, J. Pollack, & S. Wilson (Eds.), *From Animals to Animats 4, Proceedings of the International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press.

Floreano, D, & Urzelai, J. (2001). Evolution of plastic control networks. *Autonomous Robots*, *11*(3), 311–317. Springer.

Fry, C. (1978). Patent: Wind driven, high altitude power apparatus.

Furey, A., & Harvey, I. (2007). Evolution of neural networks for active control of tethered airfoils. In F. A. e Costa, L. M. Rocha, E. Costa, I. Harvey, & A. Coutinho (Eds.), *Advances in Artificial Life, 9th European Conference, ECAL 2007* (pp. 746–755). Berlin, Heidelberg: Springer.

Garcia, C. E., Prett, D. M., & Morari, M. (1989). Model predictive control: Theory and practice–a survey. *Automatica*, *25*(3), 335–348. Elsevier.

Gruau, F. (1995). Automatic definition of modular neural networks. *Adaptive behavior*, *3*(2), 151–183. Sage Publications.

Haber, M., Zhou, L., & Murai, K. K. (2006). Cooperative astrocyte and dendritic spine dynamics at hippocampal excitatory synapses. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, *26*(35), 8881-91.

Harvey, I. (1992). Species adaptation genetic algorithms: A basis for a continuing SAGA. *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life* (pp. 346–354).

Harvey, I. (2011). The microbial genetic algorithm. In G. Kampis, I. Karsai, & E. Szathmáry (Eds.), *Advances in Artificial Life: ECAL 2009 Part II* (pp. 126-133.).

Harvey, I., & Tomko, N. (2010). Binomics : Where Metagenomics meets the Binomial World Metagenomics Learning Classifier Systems. In T. Lenaerts, M. Giacobini, H. Bersini, P. Bourgine, M. Dorigo, & R. Doursat (Eds.), *Proceedings of 11th Eur. Conf. on the Synthesis and Simulation of Living Systems* (pp. 805-812).

Harvey, I., Cliff, D., & Husbands, P. (1994). Seeing the light: Artificial evolution, real vision. In D. Cliff, P. Husbands, J. Meyer, & S. Wilson (Eds.), *3rd Intl Conf on Simulation of Adaptive Behaviour SAB94* (Vol. 1994). School of Cognitive and Computing Sciences, University of Sussex.

Holland, J. (1975). *Adaptation in natural and artificial systems*. (M. U. Press, Ed.) (Vol. Ann Arbor). Ann Arbor, MI: The University of Michigan Press, Ann Arbor.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366. Elsevier.

Houska, B. (2007). *Robustness and Stability Optimization of Open-Loop Controlled Power Generating Kites*. University of Heidelberg.

Houska, B., & Diehl, M. (2007). Optimal control for power generating kites. *Proc, European Control Conference* (pp. 1-14). Kos, Greece.

Husbands, P., Smith, T., Jakobi, N., & O'Shea, M. (1998). Better living through chemistry: Evolving GasNets for robot control. *Connection Science*, *10*(3&4), 185–210. Taylor & Francis.

Ijspeert, J., & Kodjabachian, J. (1999). Evolution and development of a central pattern generator for the swimming of a lamprey. *Artificial life*, *5*(3), 247-69.

Izhikevich, E. (2006). Polychronization: computation with spikes. *Neural computation*, *18*(2), 245-82.

Izquierdo, E. (2008). *The dynamics of learning behaviour : A situated , embodied , and dynamical systems approach.*

Izquierdo, E., & Buhrmann, T. (2008). Analysis of a Dynamical Recurrent Neural Network Evolved for Two Qualitatively Different Tasks: Walking and Chemotaxis. *Artificial Life*, 257-264.

Izquierdo, E., & Harvey, I. (2006). Learning on a continuum in evolved dynamical node networks. *Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems* (pp. 507-512).

Jakobi, N. (1997). Evolutionary Robotics and the Radical Envelope-of-Noise Hypothesis. *Adaptive Behavior*, *6*(2), 325-368.

Jakobi, N. (1998). Running Across the Reality Gap : Octopod Locomotion Evolved in a Minimal Simulation. *Proceedings of the First European Workshop on Evolutionary Robotics: EvoRobot'98* (pp. 39-58). Springer Verlag.

Jakobi, N., & Quinn, M. (1998). Some Problems (And A Few Solutions) For Open-Ended Evolutionary Robotics. In P. Husbands & J.-A. Meyer (Eds.), *Proceedings of the First European Workshop on Evolutionary Robotics* (pp. 108-122). London, UK: Springer-Verlag.

Jeffreys, H., & Jeffreys, B. . (1988). *Methods of Mathematical Physics* (3rd ed.). Cambridge: Cambridge University Press.

Julstrom, B. A. (1994). Seeding the population: improved performance in a genetic algorithm for the rectilinear Steiner problem. *Proceedings of the 1994 ACM symposium on Applied computing - SAC '94* (pp. 222-226). New York, USA: ACM Press.

Kaimal, J. C., & Finnigan, J. J. (1994). *Atmospheric boundary layer flows: their structure and measurement. Boundary-Layer Meteorology* (Vol. 72). Springer.

Kandel, E., Schwartz, J., & Jessell, T. (2000). *Principles of Neural Science* (p. 1414). McGraw-Hill Medical.

Kerszberg, M., & Changeux, J. P. (1998). A simple molecular model of neurulation. *BioEssays : news and reviews in molecular, cellular and developmental biology*, *20*(9), 758-70.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems Journal*, *4*, 461–476.

Kodjabachian, J., & Meyer, J. A. (1998). Evolution and Development of Modular Control Architectures for 1D Locomotion in Six-legged Animats. *Connection Science*, *10*(3), 211-237.

Koos, S. (2010). Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers. *GECCO*, 119-126.

Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MIT Press.

Kushto, O. J. (1978). Patent: Tethered lighter than air turbine.

Lansdorp, B., Ruiterkamp, R., & Ockels, W. J. (2007). Development and testing of a steering mechanism for Laddermill kites. *Proceedings of the Windpower Shanghai Conference 2007* (pp. 1-9).

Lansdorp, B., Ruiterkamp, R., Williams, P., & Ockels, W. (2008). Long-Term Laddermill Modeling for Site Selection. *AIAA Modeling and Simulation Conference*.

Lee, M. H., Meng, Q., & Chao, F. (2007). Staged Competence Learning in Developmental Robotics. *Adaptive Behavior*, *15*(3), 241-255.

Leven, S., Zufferey, J. C., & Floreano, D. (2009). A minimalist control strategy for small UAVs. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2873-2878).

Loyd, M. L. (1980). Crosswind kite power. *Journal of Energy*, *4*, 106–111.

Mathayomchan, B., & Beer, R. D. (2002). Center-crossing recurrent neural networks for the evolution of rhythmic behavior. *Neural computation*, *14*(9), 2043-51.

Meyer, J. A. (1998). Obstacle-Avoidance in Artificial Insects. *Evolution and Development*, *9*(5), 796-812.

Minsky, M. L., & Papert, S. A. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Mjolsness, E., Sharp, D. H., & Alpert, B. K. (1989). Scaling, machine learning, and genetic neural nets. *Advances in Applied Mathematics*, *10*(2), 137–163. Elsevier.

Mouret, J. B., Doncieux, S., & Meyer, J. A. (2006). Incremental evolution of target-following neuro-controllers for flapping-wing animats. *From Animals to Animats 9*, 606–618. Springer.

Mouret, J. B., Doncieux, S., Muratet, L., Druot, T., & Meyer, J. A. (2004). Evolution of neuro-controllers for flapping-wing animats. *Proceedings of the Journées MicroDrones, Toulouse*.

Newman, E. a, & Volterra, A. (2004). Glial control of synaptic function. *Glia*, *47*(3), 207-8.

Nolfi, S., & Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA: MIT Press/Bradford Books.

Nolfi, S., Miglino, O., & Parisi, D. (1994). Phenotypic plasticity in evolving neural networks. In D. P. Gaussier & J.-D. Nicoud (Eds.), *Proceedings of the International Conference From Perception to Action* (pp. 146-157). Los Alamitos, CA: IEEE Computer Society Press.

Pachauri, R. ., Reisinger, A., & Eds. (2007). *IPCC AR4 SYR*. Geneva, Switzerland.

Pfrieger, F. W. (2002). Role of glia in synapse development. *Current opinion in neurobiology*, *12*(5), 486-90.

Phattanasri, P., Chiel, H. J., & Beer, R. D. (2007). The Dynamics of Associative Learning in Evolved Model Circuits. *Adaptive Behavior*, *15*(4), 377-396.

Podgaets, A. R., & Ockels, W. J. (2007). Comparison of two mathematical models of the kite for Laddermill sail simulation. *Lecture Notes in Engineering and Computer Science* (Vol. 2167, pp. 24-26).

Pratley, M. (2005). *Investigating the Importance of Coupling for GasNet Design*. University of Sussex.

Press, W. (1993). *Numerical Recipies*. Cambridge University Press.

Reil, T., & Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *Evolutionary Computation, IEEE*, *6*(2), 159-168.

Rohde, M., & Di Paolo, E. (2007). Adaptation to Sensory Delays An Evolutionary Robotics Model of an Empirical Study, 193-202.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386-408.

Rufli, M., Scaramuzza, D., & Siegwart, R. (2008). Automatic detection of checkerboards on blurred and distorted images. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3121-3126). Ieee.

Rumelhart, D. E., & McClelland, J. L. (1987). *Parallel Distributed Processing - Vol. 1: Foundations*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1* (pp. 318-362). Cambridge, MA: MIT Press.

Rutkowska, J. C. (1994). Scaling Up Sensorimotor Systems: Constraints from Human Infancy. *Adaptive Behavior*, *2*, 349-373.

Sanes. (2000). *Development of the Nervous System*. Academic Press.

Scaramuzza, D., Martinelli, A., & Siegwart, R. (2006). A Toolbox for Easily Calibrating Omnidirectional Cameras. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5695-5701).

Sherrod, P. (n.d.). MLP Image. *source: http://www.dtreg.com/MLFNwithWeights.jpg*. Retrieved July 10, 2010, from http://www.dtreg.com/MLFNwithWeights.jpg

Shim, Y.-S., & Kim, C.-H. (2006). Evolving physically simulated flying creatures for efficient cruising. *Artificial life*, *12*(4), 561-91.

Shim, Y.-S., Kim, S. J., & Kim, C. H. (2004). Evolving Flying Creatures with Path Following Behaviors. *ALife IX: Proceedings of the 9th International Conference on the Simulation and Synthesis of Living Systems* (pp. 125-132). Boston, MA: MIT Press.

Smith, T., Husbands, P., & O'Shea, M. (2003). Local evolvability of statistically neutral GasNet robot controllers. *Biosystems*, *69*(2-3), 223–243. Elsevier.

Spector, L. (2005). Trivial geography in genetic programming. *Genetic programming theory and practice III*, 109–124. Springer.

Stagge, P., & Sendhoff, B. (1997). An extended Elman net for modeling time series. *Lecture Notes in Computer Science* (pp. 427–432). Springer.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning:An Introduction*. Cambridge, MA: MIT.

Taylor, T., & Massey, C. (2001). Recent developments in the evolution of morphologies and controllers for physically simulated creatures. *Artificial life* (Vol. 7, pp. 77-87).

Thompson, A. (1998a). On The Automatic Design of Robust Electronics Through Artificial Evolution. In M. Sipper & D. Mange (Eds.), *Proc. 2nd Int.Conf. on Evolvable Systems (ICES'98)* (pp. 13-24). Springer-Verlag.

Thompson, A. (1998b). *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution* (p. PhD Thesis). Distinguished Dissertation Serier, Springer-Verlag.

Togelius, J. (2004). Evolution of a subsumption architecture neurocontroller. *Journal of Intelligent and Fuzzy Systems*, *15*(1), 15–20. IOS Press.

Tuci, E., Quinn, M., & Harvey, I. (2002). An Evolutionary Ecological Approach to the Study of Learning Behavior Using a Robot-Based Model. *Adaptive Behavior*, *10*(3), 201-221.

UK Commitee on Climate Change. (2010). *Building a low-carbon economy – the UK's innovation challenge*.

UK Department of Trade. (2007). Meeting the energy challenge: a White Paper on energy, (May).

Urzelai, J., & Floreano, D. (2001). Evolution of adaptive synapses: robots with fast adaptive behavior in new environments. *Evolutionary computation*, *9*(4), 495-524.

Urzelai, J., Floreano, D., Dorigo, M., & Colombetti, M. (1998). Incremental Robot Shaping. *Connection Science*, *10*(384), 341-360.

Vaughan, E. D. (2007). The Evolution of an Omni-Directional Bipedal Robot. *DPhil Thesis*.

Volterra, A., & Meldolesi, J. (2005). Astrocytes, from brain glue to communication elements: the revolution continues. *Nature Reviews Neuroscience*, (July), 2-16.

Wallace, J. M., & Hobbs, P. V. (2006). *Atmospheric science: an introductory survey. International Geophysics Series*. Academic press.

Watkins, C. (1989). *Learning from delayed rewards*. Kings College, Cambridge.

Watson, R. A., Reil, T., & Pollack, J. B. (2000). Mutualism, Parasitism, and Evolutionary Adaptation. *Proceedings of Artificial Life VII* (pp. 170-178).

Wieland, A. (1991). Evolving neural network controllers for unstable systems. *Proc IJCNN-91-Seattle International Joint Conference on Neural Networks* (pp. 667 - 673).

Williams, H., & Noble, J. (2005). Homeostatic plasticity improves signal propagation in continuous-time recurrent neural networks. *Neural Networks*, *44*(0), 1-15.

Williams, P., Lansdorp, B., & Ockels, W. J. (2007a). Modeling and Control of a Kite on a Variable Length Flexible Inelastic Tether. *AIAA Paper 2007- 6705, Aug*.

Williams, P., Lansdorp, B., & Ockels, W. J. (2007b). Flexible Tethered Kite with Moveable Attachment Points, Part 1: Dynamics and Control. *AIAA Paper 2007-6628, Aug*.

Williams, P., Lansdorp, B., & Ockels, W. J. (2007c). Modeling and control of a kite on a variable length flexible inelastic tether. *AIAA Guidance, navigation and control conference*.

Wood, R. (2007). New models for old questions: evolutionary robotics and the "A not B" error. *Proceedings of the 9th European conference on Artificial Life, ECAL 2007* (pp. 1141-1150). Springer.

Xsens MTI-g: GPS aided AHRS. (n.d.). Retrieved from http://www.xsens.com/en/general/mti-g

Yamauchi, B. M., & Beer, R. D. (1994). Sequential Behavior and Learning in Evolved Dynamical Neural Networks. *Adaptive Behavior*, *2*(3), 219-246.

Zufferey, J. C. (2005). *Bio-inspired vision-based flying robots*. PhD Thesis. EPFL.

Zufferey, J. C., Guanella, A., Beyeler, A., & Floreano, D. (2006). Flying over the reality gap: From simulated to real indoor airships. *Autonomous Robots*, *21*(3), 243–254. Springer.