Active Guidance for a Finless Rocket Using Neuroevolution

Faustino J. Gomez and Risto Miikkulainen

Department of Computer Sciences
University of Texas
Austin, TX, 78712 USA

Abstract. Finless rockets are more efficient than finned designs, but are too unstable to fly unassisted. These rockets require an active guidance system to control their orientation during flight and maintain stability. Because rocket dynamics are highly non-linear, developing such a guidance system can be prohibitively costly, especially for relatively small-scale rockets such as sounding rockets. In this paper, we propose a method for evolving a neural network guidance system using the Enforced SubPopulations (ESP) algorithm. Based on a detailed simulation model, a controller is evolved for a finless version of the Interorbital Systems RSX-2 sounding rocket. The resulting performance is compared to that of an unguided standard full-finned version. Our results show that the evolved active guidance controller can greatly increase the final altitude of the rocket, and that ESP can be an effective method for solving real-world, non-linear control tasks.

1 Introduction

Sounding rockets carry a payload for making scientific measurements to the Earth's upper atmosphere, and then return the payload to the ground by parachute. These rockets serve an invaluable role in many areas of scientific research including high-G-force testing, meteorology, radio-astronomy, environmental sampling, and micro-gravity experimentation [1,2]. They have been used for more than 40 years; they were instrumental e.g., in discovering the first evidence of X-ray sources outside the solar system in 1962 [3]. Today, sounding rockets are much in demand as the most cost-effective platform for conducting experiments in the upper atmosphere.

Like most rockets, sounding rockets are usually equipped with fins to keep them on a relatively straight path and maintain stability. While fins are an effective "passive" guidance system, they increase both mass and drag on the rocket which lowers the final altitude or *apogee* that can be reached with a given amount of fuel. A rocket with smaller fins or no fins at all can potentially fly much higher than a full-finned version. Unfortunately, such a design is unstable, requiring some kind of *active* attitude control or guidance to keep the rocket from tumbling.

Finless rockets have been used for decades in expensive, large-scale launch vehicles such as the USAF Titan family, the Russian Proton, and the Japanese H-IIA. The guidance systems on these rockets are based on classical feedback control such as Proportional-Integral-Differential (PID) methods to adjust the thrust angle (i.e. thrust vectoring) of the engines. Because rocket flight dynamics are highly non-linear, engineers must make simplifying assumptions in order to apply these linear methods, and take great care to ensure that these assumptions are not violated during operation. Such an undertaking requires detailed knowledge of the rocket's dynamics that can be very costly to acquire. Recently, non-linear approaches such as neural networks have been explored primarily for use in guided missiles (see [4] for an overview of neural network control architectures). Neural networks can implement arbitrary non-linear mappings that can make control greatly more accurate and robust, but, unfortunately, still require significant domain knowledge to train.

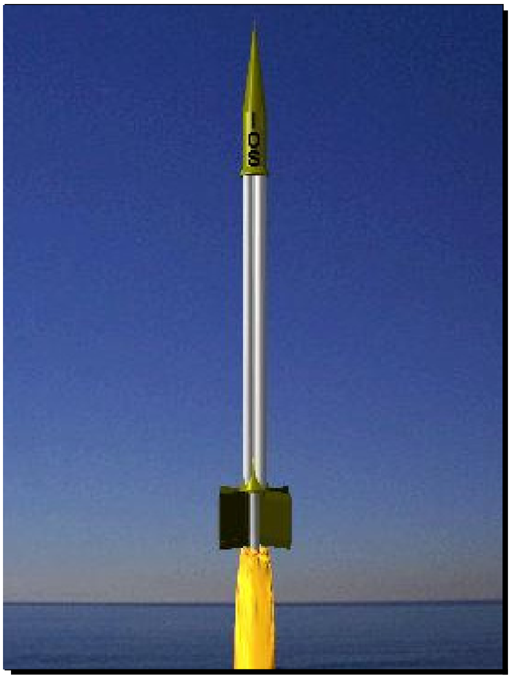


Fig. 1. The Interorbital Systems RSX-2 Rocket. The RSX-2 is capable of lifting a 5 pound payload into the upper atmosphere using a cluster of four liquid-fueled thrusters. It is currently the only liquid-fueled sounding rocket in production. Such rockets are desirable because of their low acceleration rates and non-corrosive exhaust products.

In this paper, we propose a method for making the development of finless sounding rockets more economical by using Enforced SubPopulations (ESP; [5,6]) to evolve a neural network guidance system. As a test case, we will focus on a finless version of the Interorbital Systems RSX-2 rocket (figure 1). The RSX-2 is a liquid-fueled sounding rocket that uses the differential thrust of its four engines to control attitude. By evolving a neural network controller that maps the state of the rocket to thrust commands, the guidance problem can be solved without the need for analytical modeling of the rocket's dynamics or prior knowledge of the appropriate kind of control strategy to employ. Using ESP, all that is required is a sufficiently accurate simulator and a quantitative measure of the guidance system's performance, i.e. a fitness function.

In the next three sections we describe the Enforced SubPopulations method used to evolve the guidance system, the controller evolution simulations, and experimental results on controlling the RSX-2 rocket.

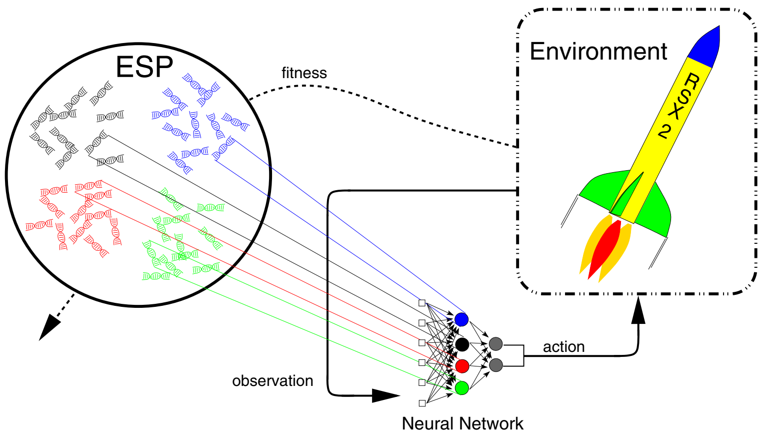


Fig. 2. The Enforced Subpopulations Method (ESP; color figure). The population of neurons is segregated into subpopulations shown here in different colors. Networks are formed by randomly selecting one neuron from each subpopulation. A neuron accumulates a fitness score by adding the fitness of each network in which it participated. This score is then normalized and the best neurons within each subpopulation are mated to form new neurons. By evolving neurons in separate subpopulations, the specialized sub-functions needed for good networks are evolved more efficiently.

2 Enforced Subpopulations (ESP)

Enforced Subpopulations[1] [5,6] is a neuroevolution method that extends the Symbiotic, Adaptive Neuroevolution algorithm (SANE; [7]). ESP and SANE differ from other NE methods in that they evolve partial solutions or *neurons* instead of complete networks, and a subset of these neurons are put together to form a complete network. In SANE, neurons are selected from a single population to form networks. In contrast, ESP makes use of explicit subtasks; a separate subpopulation is allocated for each of the h units in the network, and a neuron can only be recombined with members of its own subpopulation (figure 1). This way the neurons in each subpopulation can evolve independently and specialize rapidly into good network sub-functions.

Evolution in ESP proceeds as follows:

1. Initialization. The number of hidden units h in the networks that will be formed is specified and a subpopulation of neuron chromosomes is created. Each chromosome encodes the input and output connection weights of a neuron with a random string of real numbers.
2. Evaluation. A set of h neurons is selected randomly, one neuron from each subpopulation, to form the hidden layer of a complete network. The network is submitted to a *trial* in which it is evaluated on the task and awarded a fitness score. The score is added to the *cumulative fitness* of each neuron

[1] The ESP software package is available at:
`http://www.cs.utexas.edu/users/nn/pages/software/abstracts.html#esp-cpp`

that participated in the network. This process is repeated until each neuron has participated in an average of e.g. 10 trials.

3. Recombination. The average fitness of each neuron is calculated by dividing its cumulative fitness by the number of trials in which it participated. Neurons are then ranked by average fitness within each subpopulation. Each neuron in the top quartile is recombined with a higher-ranking neuron using 1-point crossover and mutation at low levels to create the offspring to replace the lowest-ranking half of the subpopulation.
4. The Evaluation–Recombination cycle is repeated until a network that performs sufficiently well in the task is found.

Evolving networks at the neuron level has proven to be a very efficient method for solving reinforcement learning tasks such as pole-balancing [6], robot arm control [8], and game playing [7]. ESP is more efficient that SANE because the subpopulation architecture makes the evaluations more consistent in two ways: first, the subpopulations that gradually form in SANE are already present by design in ESP. The species do not have to organize themselves out of a single large population, and their progressive specialization is not hindered by recombination across specializations that usually fulfill relatively orthogonal roles in the network. Second, because the networks formed by ESP always consist of a representative from each evolving specialization, a neuron is always evaluated on how well it performs its role in the context of all the other players.

The accelerated specialization in ESP makes it more efficient than SANE, but it also causes diversity decline over the course of evolution like that of a normal GA. This can be a problem because a converged population cannot easily adapt to a new task. To deal with premature convergence, ESP is combined with *burst mutation*. The idea is to search for optimal modifications of the current best solution. When performance has stagnated for a predetermined number of generations, new subpopulations are created by adding noise to each of the neurons in the best solution. Each new subpopulation contains neurons that represent differences from the best solution. Evolution then resumes, but now searching the space in a "neighborhood" around the best previous solution. Burst mutation can be applied multiple times, with successive invocations representing differences to the previous best solution. Assuming the best solution already has some competence in the task, most of its weights will not need to be changed radically. To ensure that most changes are small while allowing for larger changes to some weights, ESP uses the Cauchy distribution to generate noise:

$$f(x) = \frac{\alpha}{\pi(\alpha^2 + x^2)} \tag{1}$$

With this distribution 50% of the values will fall within the interval $\pm\alpha$ and 99.9% within the interval $\pm 318.3\alpha$. This technique of "recharging" the subpopulations keeps diversity in the population so that ESP can continue to make progress toward a solution even in prolonged evolution.

Burst mutation is similar to the Delta-Coding technique of [9] which was developed to improve the precision of genetic algorithms for numerical optimization problems. Because our goal is to maintain diversity, we do not reduce the range of the noise on successive applications of burst mutation and we use Cauchy rather that uniformly distributed noise.

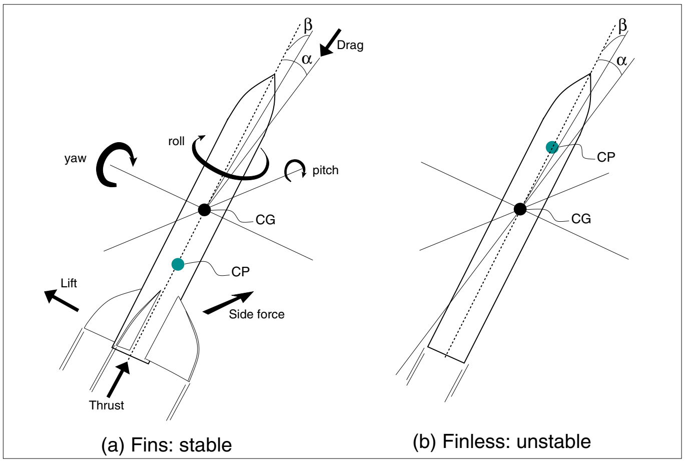


Fig. 3. Rocket Dynamics. The rocket (a) is stable because the fins increase drag in the rear of the rocket moving the center of pressure (CP) behind the center of gravity (CG). As a result, any small angles α and β are automatically corrected. In contrast, the finless rocket (b) is unstable because the CP stays well ahead of the CG. To keep α and β from increasing, i.e to keep the rocket from tumbling, active guidance is needed to counteract the destabilizing torque produced by drag.

3 The Finless Rocket Guidance Task

In previous work, ESP was shown to outperform a wide range of reinforcement learning algorithms including Q-learning, SARSA(λ), Evolutionary Programming, and SANE on several difficult versions of the pole-balancing or inverted pendulum task [6]. The rocket guidance domain is similar to pole-balancing in that both involve stabilizing an inherently unstable system. Figure 1 gives a basic overview of rocket dynamics. The motion of a rocket is defined by the translation of its center of gravity (CG), and the rotation of the body about the CG in the pitch, yaw, and roll axes. Four forces act upon a rocket in flight: (1) the thrust of the engines which propel the rocket, (2) the drag of the atmosphere exerted at the *center of pressure* (CP) in roughly the opposite direction to the thrust, (3) the *lift force* generated by the fins along the yaw axis, and (4) the *side force* generated by the fins along the pitch axis. The angle between the direction the rocket is flying and the longitudinal axis of the rocket in the yaw-roll plane is known as the *angle of attack* or α, the corresponding angle in the pitch-roll plane is known as the *sideslip angle* or β. When either α or β is greater than 0 degrees the drag exerts a torque on the rocket that can cause the rocket to tumble if it is not stable. The arm through which this torque acts is the distance between the CP and the CG.

In figure 1a, the finned rocket is stable because the CP is behind the rocket's CG. When α (β) is non-zero, a torque is generated by the lift (side) force of the fins that counteracts the drag torque, and tends to minimize α (β). This situation corresponds to a pendulum hanging down from its hinge; the pendulum

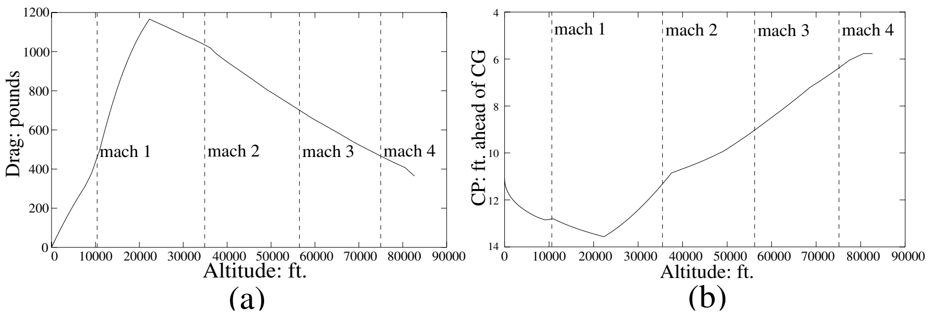
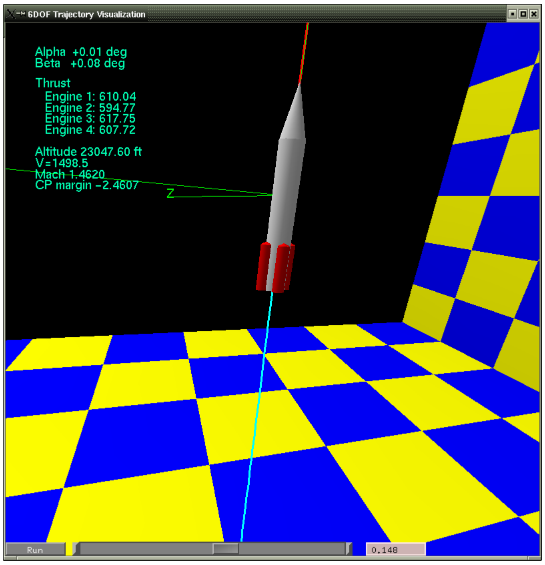


Fig. 4. The time-varying difficulty of the guidance task. Plot (a) shows the amount of drag force acting on the finless rocket as it ascends through the atmosphere. More drag means that it takes more differential thrust to control the rocket. Plot (b) shows the position of the center of pressure in terms of how many feet ahead it is of the center of gravity. From 0 to about 22,000ft the control task becomes more difficult due to the rapid increase in drag and the movement of the CG away from the nose of the rocket. At approximately 22,000ft, drag peaks and begins a decline as the air gets thinner, and the CP starts a steady migration towards the CG. As it ascends further, the rocket becomes progressively easier to control as the density of the atmosphere decreases.

will return to this stable equilibrium point if it is disturbed. When the rocket does not have fins, as in figure 1b, the CP is ahead of the CG causing the rocket to be unstable. A non-zero α or β will tend to grow causing the rocket to eventually tumble. This situation corresponds to a pendulum at its unstable upright equilibrium point where any disturbance will cause it to diverge away from this state.

Although the rocket domain is similar to the inverted pendulum, the rocket guidance problem is significantly more difficult for two reasons: the interactions between the rocket and the atmosphere are highly non-linear and complex, and the rocket's behavior continuously changes throughout the course of a flight due to system variables that are either not under control or not directly observable (e.g. air density, fuel load, drag, etc.).

Figure 3 shows how the difficulty of stabilization varies over the course of a successful flight for the finless rocket. In figure 3a, drag is plotted against altitude. From 0ft to about 22,000ft, the rocket approaches the sound barrier (Mach 1) and drag rises sharply. This drag increases the torque exerted on the rocket in the yaw and pitch axes for a given α and β, making it more difficult to control its attitude. In figure 3b, we see that also during this period the distance between the CG and CP increases because the consumption of fuel causes the CG to move back, making the rocket increasingly unstable. After 22,000ft, drag starts to decrease as the air becomes less dense, and the CP steadily migrates back towards the CG, so that the rocket becomes easier to control.



```
Alpha  +0.01 deg
Beta   +0.08 deg

Thrust
  Engine 1: 610.04
  Engine 2: 594.77
  Engine 3: 617.75
  Engine 4: 607.72

Altitude 23047.60 ft
V=1498.5
Mach 1.4620
CP margin -2.4607
```

Fig. 5. RSX-2 Rocket Simulator. The picture shows a 3D visualization of the JSBSim rocket simulation used to evolve the RSX-2 guidance controllers. The simulator provides a realistic environment for designing and verifying aircraft dynamics and guidance systems.

For ESP, this means that the fitness function automatically scales the difficulty of the task in response to the performance of the population. At the beginning of evolution the task is relatively easy. As the population improves and individuals are able to control the rocket to higher altitudes, the task becomes progressively harder. Although figure 3 indicates that above 22,000ft the task again becomes easier, progress in evolution continues to be difficult because the controller is constantly entering an unfamiliar part of the state space. A fitness function that gradually increases in difficulty is usually desirable because it allows for sufficient selective pressure at the beginning of evolution to direct the search into a favorable region of the solution space. However, the rocket control task is already too hard in the beginning—all members of the initial population perform so poorly that the evolution stalls and converges to a local maxima. In other words, direct evolution does not even get started on this very challenging task. One way to overcome this problem is to make the initial task easier and then increase the difficulty as the performance of the population improves. In the experiments below we employ such an *incremental evolution* approach [5]. Instead of trying to evolve a controller for the finless rocket directly, a more stable version of the rocket is used initially to make the ultimate task more accessible.

The following section describes the simulation environment used to evolve the controller, the details of how a guidance controller interacts with the simulator, and the experimental setup for evolving a neural network controller for this task.

4 Rocket Control Experiments

4.1 The RSX2 Rocket Simulator

As an evolution environment we used the JSBSim Flight Dynamics Model[2] adapted for the RSX-2 rocket by Eric Gullichsen of Interorbital Systems. JSB-

[2] More information about the free JSBSim software package is available at: `http://jsbsim.sourceforge.net/`

Sim is an open source, object-oriented flight dynamics simulator with the ability to specify a flight control system of any complexity. JSBSim provides a realistic simulation of the complex dynamic interaction between the airframe, propulsion system, fuel tanks, atmosphere, and flight controls. The aerodynamic forces and moments on the rocket were calculated using a detailed geometric model of the RSX-2. Four versions of the rocket with different fin configurations were used: full fins, half fins (smaller fins), quarter fins (smaller still), and no fins, i.e. the actual finless rocket. This set of rockets allowed us to observe the behavior of the RSX2 at different levels of instability, and provided a sequence of increasingly difficult tasks with which to evolve incrementally. All simulations used Adams-Bashforth 4th-order integration with a time step of 0.0025 seconds.

4.2 Neural Guidance Control Architecture

The rocket controller is represented by a feedforward neural network with one hidden layer (figure 6). Every 0.05 seconds (i.e. the control time-step) the controller receives a vector of readings from the rocket's onboard sensors that provide information about the current orientation (pitch, yaw, roll), rate of orientation change, angle of attack α, sideslip angle β, the current throttle position of the four thrusters, altitude, and velocity in the direction of flight. This input vector is propagated through the sigmoidal hidden and output units of the network to produce a new throttle position for each engine determined by:

$$u_i = 1.0 - o_i/\delta, \; i = 1..4 \quad (2)$$

where u_i is the throttle position of thruster i, o_i is the value of network output unit i, $0 \leq u_i, o_i \leq 1$, and $\delta \geq 1.0$. A value of ω for u_i means that the controller wants thruster i to generate $\omega \times 100\%$ of maximum thrust. The parameter δ controls how far the controller is permitted to "throttle back" an engine from 100% thrust.

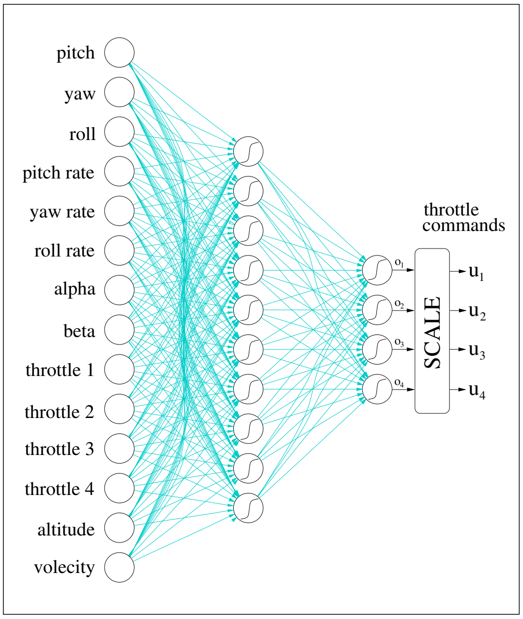


Fig. 6. Neural Network Guidance. The control network receives the state of the rocket every time step through its input layer. The input consists of the rocket's orientation, the rate of change in orientation, α, β, the current throttle position of each engine, the altitude, and the velocity of the rocket in the direction of flight. These values are propagated through the network to produce a new throttle command (the amount of thrust) for each engine.

4.3 Experimental Setup

The objective of the experiments was to determine whether ESP could evolve a controller to stabilize the finless version of the RSX-2 rocket. To do so, the neural guidance system has to control the thrust of each of the four engines in order to maintain the rocket's angle of attack α and sideslip angle β within ± 5 degrees from the time of ignition to burnout when all of the fuel has been expended. Exceeding the ± 5 degree boundary indicates that the rocket is about to tumble and is therefore considered a catastrophic failure. Each ESP network was evaluated in a single trial that consisted of the following four phases:

1. At time t_0, the rocket is attached to a launch rail that will guide it on a straight path for the first 50 feet of flight. The fuel tanks are full and the engines are ignited.
2. At time $t_1 > t_0$, the rocket begins its ascent as the engines are powered to full thrust.
3. At time $t_2 > t_1$, the rocket leaves the launch rail and the controller begins to modulate the thrust as described in section 4.2 according to equation 2.
4. While controlling the rocket one of two events occurs at time $t_f > t_2$:
   a) α or β exceeds ± 5 degrees, in which case the controller has failed.
   b) the rocket reaches burnout, in which case the controller has succeeded.

   In either case, the trial is over and the altitude of the rocket at t_f becomes the fitness score for the network.

In a real launch, the rocket would continue after burnout and coast to apogee. Since we are only concerned with the control phase, for efficiency the trials were limited to reaching burnout. This fitness measure is all that is needed to encourage evolutionary progress. However, there is a large locally maximal region in the network weight space corresponding to the policy $o_i = 1.0, \ i = 1..4$; the policy of keeping all four engines at full throttle. Since it is very easy to randomly generate networks that saturate their outputs, this policy will be present in the first generations. Such a policy clearly does not solve the task, but because the rocket is so unstable, no better policy is likely to be present in the initial population. Therefore, it will quickly dominate the population and halt progress toward a solution. To avoid this problem, all controllers that exhibited this policy were penalized by setting their fitness to zero. This procedure ensured that the controller was not rewarded for doing nothing.

The simulations used 10 subpopulations of 200 neurons (i.e. the networks were composed of 10 hidden units), and δ was set to 10 so the network could only control the thrust in the range between 90% and 100% for each engine. It was determined in early testing that this range produced sufficient differential thrust to counteract side forces, and solve the task.

As was discussed in section 3, evolving a controller directly for the finless rocket was too difficult and an incremental evolution method was used instead. We first evolved a controller for the quarter-finned rocket. Once a solution to this easier task was found, the evolution was transitioned to the more difficult finless rocket.

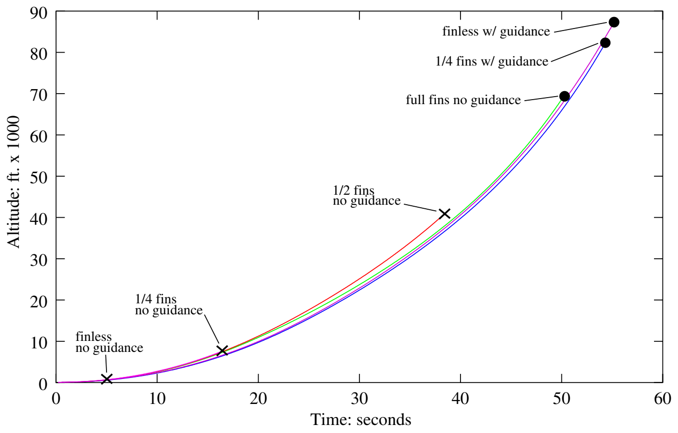


Fig. 7. Comparison of burnout altitudes for different fin-size rockets with and without guidance. The crosses indicate the altitude at which a particular rocket becomes unstable (i.e. either α or $\beta > \pm 5$ degrees). The circles indicate the altitude of a successful rocket that remained stable all the way to burnout. The guided quarter-finned and finless rockets fly significantly higher than the unguided full-finned rocket.

5 Results

ESP solved the task of controlling the quarter-finned rocket in approximately 600,000 evaluations. Another 50,000 evaluations were required to successfully transition to the finless rocket. Figure 4.3 compares the altitudes that the various rockets reach in simulation. Without guidance, the full-finned rocket reaches burnout at approximately 70,000ft, whereas the finless, quarter-finned, and half-finned rockets all fail before reaching burnout. However, with neural network guidance the quarter-finned and finless rockets do reach burnout and exceed the full-finned rocket's altitude by 10,000ft and 15,000ft, respectively. After burnout, the rocket will begin to coast at a higher velocity in a less dense part of the atmosphere; the higher burnout altitude for the finless rocket translates into an apogee that is about 20 miles higher than that of the finned rocket.

Figure 5a shows the behavior of the four engines during a guided flight for the finless rocket. The controller makes smooth changes to the thrust of the engines throughout the flight. This very fine control is required because any abrupt changes in thrust at speeds of up to Mach 4 can quickly cause failure. Figure 5b shows α and β for the various rockets with and without guidance. Without guidance, the quarter-finned and even the half-finned rocket start to tumble as soon as α or β start to diverge from 0 degrees. Using guidance, both the quarter-finned and finless rockets keep α and β at very small values up to burnout. Note that although the finless controller was produced by further evolving the quarter-finned controller, the finless controller not only solves a more difficult task, but does so with more optimal performance.

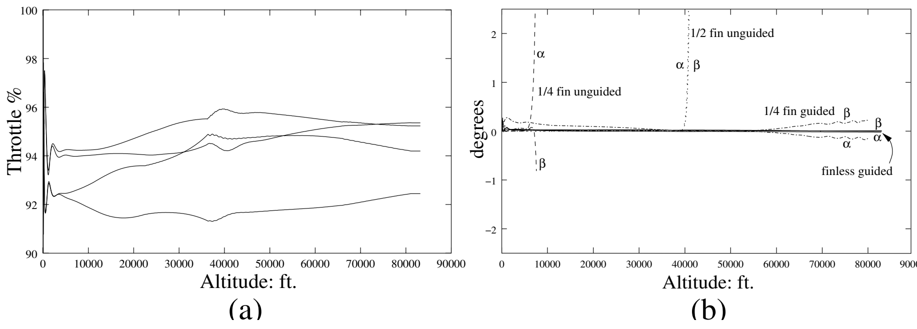


Fig. 8. Controller performance for the finless rocket. Plot (a) shows the policy implemented by the controller. Each curve corresponds to the percent thrust of one of the four rocket engines over the course of a successful flight. After some initial oscillation the control becomes very fine, changing less than 2% of total thrust for any given engine. Plot (b) compares the values α and β for various rocket configurations and illustrates how well the neural guidance system is able to minimize α and β. The unguided quarter-finned and half-finned rockets maintain low α and β for a while, but as soon as either starts to grow the rocket tumbles. In contrast, the guidance systems for the quarter-finned and finless rockets are able to contain α and β all the way up to burnout. The finless controller, although evolved from the quarter-finned controller, is more optimal.

6 Discussion and Future Work

The rocket control task is representative of many real world problems such as manufacturing, financial prediction, and robotics that are characterized by a complex non-linear interaction between system components. The critical advantage of using ESP over traditional engineering approaches is that it can produce a controller for these systems without requiring formal knowledge of system behavior or prior knowledge of correct control behavior. The experiments in this paper demonstrate this ability by solving a high-precision, high-dimensional, non-linear control task.

Of equal importance is the result that the differential thrust approach for the finless version of the current RSX-2 rocket is feasible. Also, having a controller that can complete the sounding rocket mission in simulation has provided valuable information about the behavior of the rocket that would not otherwise be available.

In the future, we plan on making the task more realistic in two ways: (1) the controller will no longer receive α and β values as input, and (2) instead of a generating continuous control signal, the network will output a binary vector indicating whether or not each engine should "throttle back" to a preset low throttle position. This scheme will greatly simplify the control hardware on the real rocket. Once a controller for this new task is evolved, work will focus on varying environmental parameters and incorporating noise and wind so that the controller will be forced to utilize a robust and general strategy to stabilize the rocket. This phase will be critical to our ultimate goal of transferring the controller to the RSX-2 and testing it in an actual rocket launch.

7 Conclusion

In this paper, we propose a method for learning an active guidance system for a dynamically unstable, finless rocket by using the Enforced Subpopulation neuroevolution algorithm. Our experiments show that the evolved guidance system is able to stabilize the finless rocket and greatly improve its final altitude compared to the full-finned, stable version of the rocket. The results suggest that neuroevolution is a promising approach for difficult nonlinear control tasks in the real world.

Acknowledgments. This research was supported in part by the National Science Foundation under grant IIS-0083776, and the Texas Higher Education Coordinating Board under grant ARP-003658-476-2001. The authors would like to give special thanks to Eric Gullichsen of Interorbital Systems (egullich@colo.to) for his collaboration and invaluable assistance in this project.

References

1. Corliss, W.R.: NASA sounding rockets, 1958–1968: A historical summary. Technical Report NASA SP-4401, National Aeronautics and Space Administration, Washington, D.C. (1971)
2. Seibert, G.: A world without gravity. Technical Report SP-1251, European Space Agency (2001)
3. Giacconi, R., Gursky, H., Paolini, F., Rossi, B.: Evidence for X-rays from sources outside the solar system. Physical Review Letters 9 (1962) 439–444
4. Liang Lin, C., Wen Su, H.: Intelligent control theory in guidance and control system design: an overview. Proc. Natl. Sci, Counc. ROC(A) 24 (2000) 15–30
5. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Adaptive Behavior 5 (1997) 317–342
6. Gomez, F., Miikkulainen, R.: Robust non-linear control through neuroevolution. Technical Report AI02-292, Department of Computer Sciences, The University of Texas at Austin (2002)
7. Moriarty, D.E.: Symbiotic Evolution of Neural Networks in Sequential Decision Tasks. PhD thesis, Department of Computer Sciences, The University of Texas at Austin (1997) Technical Report UT-AI97-257.
8. Moriarty, D.E., Miikkulainen, R.: Evolving obstacle avoidance behavior in a robot arm. Technical Report AI96-243, Department of Computer Sciences, The University of Texas at Austin (1996)
9. Whitley, D., Mathias, K., Fitzhorn, P.: Delta-Coding: An iterative search strategy for genetic algorithms. In Belew, R.K., Booker, L.B., eds.: Proceedings of the Fourth International Conference on Genetic Algorithms, San Francisco, CA: Morgan Kaufmann (1991) 77–84