

Controlling a Black-Box Simulation of a Spacecraft

Claude Sammut and Donald Michie

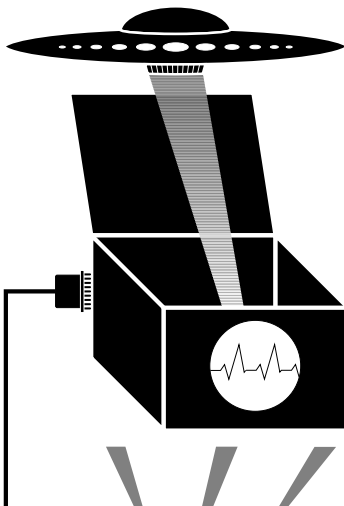
When building a controller for a physical process, traditional control theory requires a mathematical model to predict the behavior of the process so that appropriate control decisions can be made. Unfortunately, either many real-world processes are too complicated to accurately model, or insufficient information is available about the process environment. In addition, optimal control strategies can themselves exhibit undesirable complexity. However, a human controller can often acquire relatively simple strategies to effect near-optimal control from operational experience. There has been growing interest in computer simulation used in conjunction with AI methods to approach both difficulties, that is, both to simplify the construction of

This article reports on experiments performed using a black-box simulation of a spacecraft. The goal of this research is to learn to control the attitude of an orbiting satellite. The spacecraft must be able to operate with minimal human supervision. To this end, we are investigating the possibility of using adaptive controllers for such tasks. Laboratory tests have suggested that rule-based methods can be more robust than systems developed using traditional control theory. The BOXES learning system, which has already met with success in simulated laboratory tasks, is an effective design framework for this new exercise.

simulators and to test design ideas for rule-based and other AI-type controllers. The latter theme, addressed by Glicksman (1986) in the context of a planned autonomous land vehicle, was one among some two dozen contributions to a recent conference on intelligent

simulation environments (Luker and Adelsberger 1986) that was striking evidence of the recent growth of interest in this area. The specific topic of the real-time control of unstable systems, however, has not been investigated to a great degree. In particular, little information is available on the effects of substituting AI-type rule sets in industrial-strength dynamic control tasks.

In spacecraft attitude control, inherent complexity is aggravated by unpredictable changes in the task environment. It was the subject of an early study of the applicability of machine learning techniques (Barron, Davies, Schalkowsky, and Snyder 1964, 1965). If the attitude of a satellite in low earth orbit is to be kept stable by means of thrusters, the control system must contend with many unknown factors. For example, although thin, the earth's atmosphere can extend many hundreds of kilometers into space. At different times, the solar wind can cause the atmosphere's density to change, thus altering the drag and aerodynamic torques on the spacecraft. Earth-bound designers cannot predict these factors, and even after three decades of spaceflight, attitude control is still a major problem.



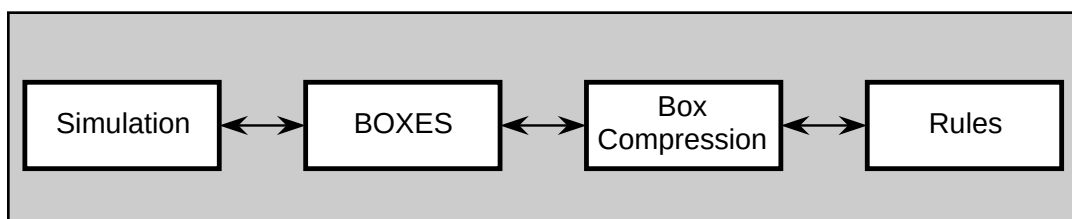


Figure 1. Learning Scheme.

...traditional control theory requires a mathematical model to predict the behavior of the process...

We were recently invited by a commercial client to test adaptive rule-based control using a computer simulation of an orbiting spacecraft under black-box conditions. Thus, knowledge of the simulation's structure and parameters was to be unavailable to the controller. Constraints and assumptions built into the client's simulation included minimal human supervision, and to this end, only one ground station would be used to control the vehicle. The ground crew only has a 16-minute window in each 90-minute orbit during which it can communicate with the spacecraft. For the rest of the orbit, the vehicle has no communication links with the control center.

Because of the unpredictable nature of the environment and the possibility of a failure when out of communication with the ground crew, there is some interest in using an adaptive attitude controller. The experiments reported here show how a control strategy was developed on a laboratory problem, learning to balance a pole, and then transplanted into a simulation of the spacecraft. To test the robustness of the method, the internal workings of the simulation were not disclosed to the experimenters; that is, a black-box simulation, in the form of Fortran object code, was used. The client withheld the source code to safeguard the black-box condition. Even after receiving our experimental results, nothing further was revealed beyond the comment that our rule-based model marginally outperformed the client's own control-theoretic solution. The description of the black box in a later section contains the sum total of our knowledge about it. The only qualification we can make is that we saw an artist's drawing of the satellite; however, we are unaware of any way that we could have used this information about the vehicle's overall geometry.

In the following sections of this article, we briefly describe the pole-balancing experiments and their results, the black-box simulation,

and the control strategies used and report on the results of the experiments.

Pole-Balancing Experiments

A number of researchers, including Michie and Chambers (1968), Selfridge, Sutton, and Barto (1985), Anderson (1987), and Connell and Utgoff (1987), have investigated the problem of learning rules to balance a pole by trial and error. A rigid pole is hinged to a cart that is free to move along a track of finite length. The learning system attempts to keep the pole balanced and the cart within the limits of the track by applying a force of fixed magnitude to the cart, either to the left or the right at each successive moment.

It was decided to use the pole and cart as a test bed for learning strategies that might be used in satellite attitude control. As we see later, the two problems are not as dissimilar as they might, at first, appear. We only briefly describe the experiments because they were reported elsewhere (Sammur 1988).

Figure 1 illustrates the basic scheme used. A reinforcement learning algorithm called BOXES (Michie and Chambers 1968) creates a control strategy for the pole-and-cart system. This control strategy is used in an experiment that attempts to keep the system from failing for as long as possible. After each failure, the learning algorithm alters the control strategy and conducts a new experiment until it is successful in keeping the pole balanced for an indefinite period.

The pole-and-cart system is characterized by four state variables that make up a four-dimensional space. By dividing each dimension into intervals, the state space is filled by four-dimensional boxes. In the BOXES algorithm, each box contains a setting that indicates that for any point within the given box, the cart should be pushed either to the left or the right in this state.

It is usually the case that after one successful attempt to learn to balance the pole, the

...to use the pole and act as a test bed for learning strategies that might be used in satellite attitude control...

controller becomes expert in balancing in only a limited part of the state space. That is, the system is maneuvered into a region of the space about which the controller has acquired considerable experience. However, if the pole-and-cart system is placed in another part of the space about which the controller knows less, then an attempt to balance the pole could rapidly fail. Therefore, it has proved necessary to gather control strategies from many learning sequences and pool their knowledge to construct a general control strategy.

We found that when a general strategy is constructed, the box structure can be simplified to the point where human-readable rules can be extracted. Below, we show a general form of the control rules finally constructed by this process:

```

if the angular velocity of the pole is less
than a given threshold then push left
  else if the angular velocity is greater than a
given threshold then push right
    else if the angle of the pole is less than a
given threshold then push left
      else if the angle of the pole is greater than a
given threshold then push right
        else if the velocity of the cart is less than a
given threshold then push right
          else if the velocity is greater than a given
threshold then push left
            else if the position of the cart is less than a
given threshold then push right
              else if the position of the cart is greater
than a given threshold then push left

```

The logical structure of this rule is identical to one derived from the equations of motion of the pole and cart (Makarovic 1987). Note, however, that the learning system did not have any knowledge of the system it was controlling other than its response to a limited set of actions.

In the following section, we describe the satellite control problem and then show how the experience gained from the pole and cart made it possible to quickly build a controller for the satellite.

The Black Box

Here, we define the satellite's black-box model.

The Problem Domain: The model has a three-axis, rigid body with attitude control and three nonlinear, coupled, second-order, ordinary differential equations.

The Task: The task is to drive the system from its initial state (which is near its final state) to the specified final state and maintain this state.

The Black-Box Model: The black-box model includes vehicle parameters, disturbances, and differential equations. Included in the black box is a fourth-order Runge-Kutta numeric integration algorithm that integrates the dynamics of the equations of motion. The time step has a fixed value of 10 seconds. The black box tracks time. The black-box module was supplied as Fortran object code.

Time-Varying Vehicle Parameters: The model has time-varying vehicle parameters, including slow variations in vehicle inertia, that are unknown to the control system, such as might happen during propellant expenditure, payload redistribution, solar array articulation, gravity-gradient boom extension-retraction, or robot motion.

Time-Varying Disturbances: The model also includes time-varying disturbances resulting from external influences, such as slow variations in atmospheric density affecting aerodynamic torque, which are unknown to the control system, in addition to aerodynamic torques that change with attitude.

State Variables: The state variables in the model are attitudes: yaw, roll, and pitch; body rates: ω_x , ω_y , and ω_z (body rates might or might not equal the rate of change of attitude); and the propellant consumed to date.

Initial Conditions: The initial values for the state variables are

Yaw	=	10 degrees
Roll	=	10 degrees
Pitch	=	10 degrees

$$\begin{aligned}\omega_x &= 0.025 \text{ degree/second} \\ \omega_y &= 0.025 \text{ degree/second} \\ \omega_z &= 0.025 \text{ degree/second}\end{aligned}$$

The Desired State: The desired values for the state variables should be within the following bounds:

$$\begin{aligned}\text{Yaw} &= 0 \pm 3 \text{ degrees} \\ \text{Roll} &= 0 \pm 3 \text{ degrees} \\ \text{Pitch} &= 0 \pm 3 \text{ degrees} \\ \omega_x &= 0 \pm 0.005 \text{ degree/second} \\ \omega_y &= 0 \pm 0.005 \text{ degree/second} \\ \omega_z &= 0 \pm 0.005 \text{ degree/second}\end{aligned}$$

In addition, the propellant consumed to achieve these values should be minimized.

Failure Condition: If any of the state variables go beyond the following bounds, then the system is said to have failed:

$$\begin{aligned}\text{Yaw} &= \pm 30 \text{ degrees} \\ \text{Roll} &= \pm 30 \text{ degrees} \\ \text{Pitch} &= \pm 30 \text{ degrees} \\ \omega_x &= \pm 0.05 \text{ degree/second} \\ \omega_y &= \pm 0.05 \text{ degree/second} \\ \omega_z &= \pm 0.05 \text{ degree/second}\end{aligned}$$

A flag in the black box is turned on if these bounds are exceeded.

Available Control Input: Torques are T_x , T_y , and T_z . In this model, torque is applied by firing thrusters that are aligned to the body axes. The following bounds show the minimum and maximum torques that can be applied by the thrusters:

$$\begin{aligned}T_x(\text{Min}) &= 0 \text{ foot-pound} \\ T_y(\text{Min}) &= 0 \text{ foot-pound} \\ T_z(\text{Min}) &= 0 \text{ foot-pound} \\ T_x(\text{Max}) &= \pm 0.5 \text{ foot-pound} \\ T_y(\text{Max}) &= \pm 1.5 \text{ foot-pound} \\ T_z(\text{Max}) &= \pm 1.5 \text{ foot-pound}\end{aligned}$$

A flag in the black box is turned on if the torque command exceeds these bounds.

Developing the Black-Box Controller

Following the experience gained with the pole and cart, we hypothesized that a rule similar to the one in Pole-Balancing Experiments might be able to control the satellite. That is, first check that the velocity in one dimension does not exceed certain bounds. If it does, then take action to oppose this velocity. If it does not, then check the position in the same dimension. If it exceeds given bounds, then apply a force that pushes the position within the desired bounds. If it does not, then proceed to the next dimension, and so on.

In the case of the pole and cart, there was a necessary priority to the order in which the dimensions were checked. It was critical that the angular velocity and the angle of the pole

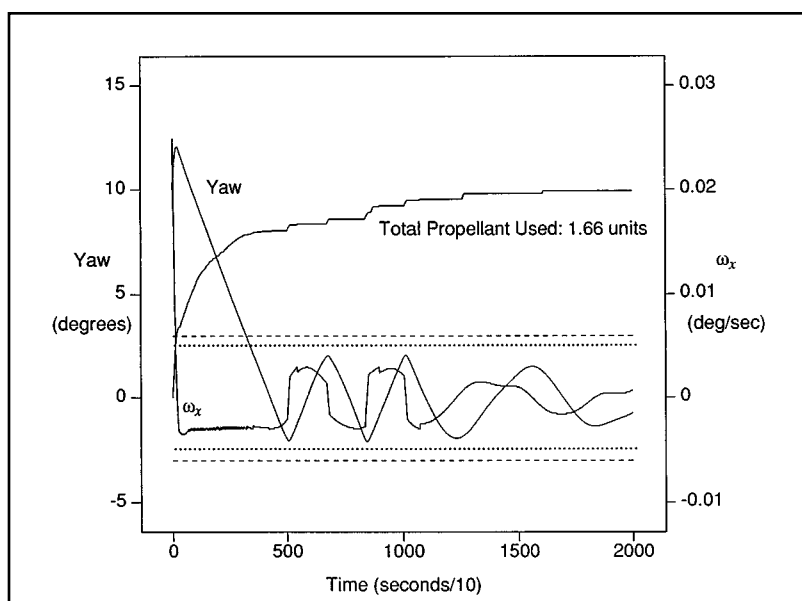


Figure 2. X Axis (yaw) Behavior of Satellite.

be considered before the velocity and position of the cart because failing to control the pole's behavior leads to more rapid overall failure than failing to keep the cart away from the ends of the track.

If this were also true of the spacecraft, then it would be necessary to determine in which of these dimensions the state variables changed most rapidly when no torques were applied. This dimension was the pitch dimension, followed by roll, yielding an initial rule:

if $\omega_z < -0.002$	then apply a torque of 1.5
else if $\omega_z > 0.002$	then apply a torque of -1.5
else if pitch < -2	then apply a torque of 1.5
else if pitch > 2	then apply a torque of -1.5
else if $\omega_y < -0.002$	then apply a torque of 1.5
else if $\omega_y > 0.002$	then apply a torque of -1.5
else if roll < -2	then apply a torque of 1.5
else if roll > 2	then apply a torque of -1.5
else if $\omega_x < -0.002$	then apply a torque of 0.5
else if $\omega_x > 0.002$	then apply a torque of -0.5
else if yaw < -2	then apply a torque of 0.5
else if yaw > 2	then apply a torque of -0.5

Note that we are using bang-bang control;

Yaw positive	$\frac{T_x}{4}$	0	$-\frac{T_x}{4}$	$-\frac{T_x}{2}$	$-T_x$
Yaw ok	$\frac{T_x}{2}$	0	0	0	$-\frac{T_x}{2}$
Yaw negative	T_x	$\frac{T_x}{2}$	$\frac{T_x}{4}$	0	$-\frac{T_x}{4}$
	Yaw rate very negative	Yaw rate negative	Yaw rate ok	Yaw rate positive	Yaw rate very positive

Table 1. Yaw Control Rule.

that is, the torquers are either fully on or fully off, just as in the pole-balancing experiments. The thresholds for the variables were determined by choosing an arbitrary value slightly within the bounds given for the desired values of the variables.

This control strategy proved to be successful but slow, requiring 8700 seconds to bring the vehicle under control, that is, all the state variables within the desired bounds. The strategy also consumed 11.2 units of propellant. The question arose about whether the control of each dimension could be decoupled. The previous rule only allows one thruster to be fired at any one time. If each axis of the spacecraft were separately considered, then all three thrusters could be fired simultaneously. This consideration led to three rules:

if $\omega_z < -0.002$ then apply a torque of 1.5
 else if $\omega_z > 0.002$ then apply a torque of 1.5
 else if pitch < -2 then apply a torque of 1.5
 else if pitch > 2 then apply a torque of 1.5
 if $\omega_y < -0.002$ then apply a torque of 1.5
 else if $\omega_y > 0.002$ then apply a torque of -1.5
 else if roll < -2 then apply torque of 1.5
 else if roll > 2 then apply a torque of 1.5
 if $\omega_x < -0.002$ then apply a torque of 0.5
 else if $\omega_x > 0.002$ then apply a torque of -0.5

else if yaw < -2 then apply a torque of 0.5
 else if yaw > 2 then apply a torque of -0.5

These rules not only bring the spacecraft under control, they do so rapidly, requiring only 4090 seconds.

Another important factor in controlling a spacecraft's attitude is the amount of propellant used. Once the propellant is exhausted, the craft is effectively dead because the solar arrays cannot be aligned properly; communication becomes difficult because the antennae cannot be pointed in the correct direction; and if the spacecraft is tumbling, the microgravity experiments can be destroyed. The second control strategy rapidly consumed propellant, using 7.68 units before the spacecraft became stable. Therefore, it became necessary to reexamine the bang-bang strategy with a view to replacing it with finer thruster control.

The control strategy that was devised is best understood using a decision array. For example, the yaw control rule can be visualized as in table 1. Each of the 15 boxes corresponds to one control rule. Thus, the box in the top left-hand corner states that if the yaw is positive (that is, above the bounds on the desirable yaw) and if the yaw rate, ω_x , is well below the bounds of desirability, then apply a quarter of the full torque in the positive direction. In fact, the thresholds were set slightly within the bounds of desirability: For angles, they were $\pm 2^\circ$ and $\pm 30^\circ$, and for angular velocities, they were ± 0.002 , ± 0.003 , and ± 0.05 degree/second. Similar strategies were

Roll positive	$\frac{T_y}{4}$	0	$-\frac{T_y}{4}$	$-\frac{T_y}{2}$	$-T_y$
Roll ok	$\frac{T_y}{2}$	0	0	0	$-\frac{T_y}{2}$
Roll negative	T_y	$\frac{T_y}{2}$	$\frac{T_y}{4}$	0	$-\frac{T_y}{4}$
	Roll rate very negative	Roll rate negative	Roll rate ok	Roll rate positive	Roll rate very positive

Table 2. Roll Control Rule.

Pitch positive	$\frac{T_z}{4}$	0	$-\frac{T_z}{4}$	$-\frac{T_z}{2}$	$-T_z$
Pitch ok	$\frac{T_z}{2}$	0	0	0	$-\frac{T_z}{2}$
Pitch negative	T_z	$\frac{T_z}{2}$	$\frac{T_z}{4}$	0	$-\frac{T_z}{4}$
	Pitch rate very negative	Pitch rate negative	Pitch rate ok	Pitch rate positive	Pitch rate very positive

Table 3. Pitch Control Rule.

used for roll (table 2) and pitch.

During the formulation of these rules, it was found that the satellite had greater inertia in the z axis than in the other two. Therefore, it was decided to thrust somewhat harder with the z torquer when the pitch requires correction, but the pitch rate is ok, as in the decision array shown in table 3. This control strategy brought the spacecraft under control in 5290 seconds, requiring only 1.66 units of propellant, a substantial savings. The previous sequence constitutes all the experiments that were done. The next section describes the results of the culminating experiment that incorporated the final z torquer adjustment.

The experiments were conducted on a UNIX workstation and allowed the simulator to interact with a modifiable control procedure. As previously mentioned, the simulator was provided as Fortran object code. This simulator was linked to a c module that implemented the decision arrays. The two modules

communicated through a single procedure call from the c code to the Fortran code. All data passed from one module to the other through parameters to the procedure call.

Results

The graph in figure 2 shows the x axis behavior of the final control strategy when applied to the black-box model. The dashed horizontal line indicates the bounds of desirability for the yaw, and the dotted line indicates the bounds for ω_x . Note that in the early stages of the simulation, propellant is rapidly being consumed because the spacecraft is well out of bounds. As both the position and velocity decrease, so, too, does propellant consumption. In fact, the amount of propellant used to bring the system under control compared favorably with the propellant use of a controller developed using traditional means and having full knowledge of the differential

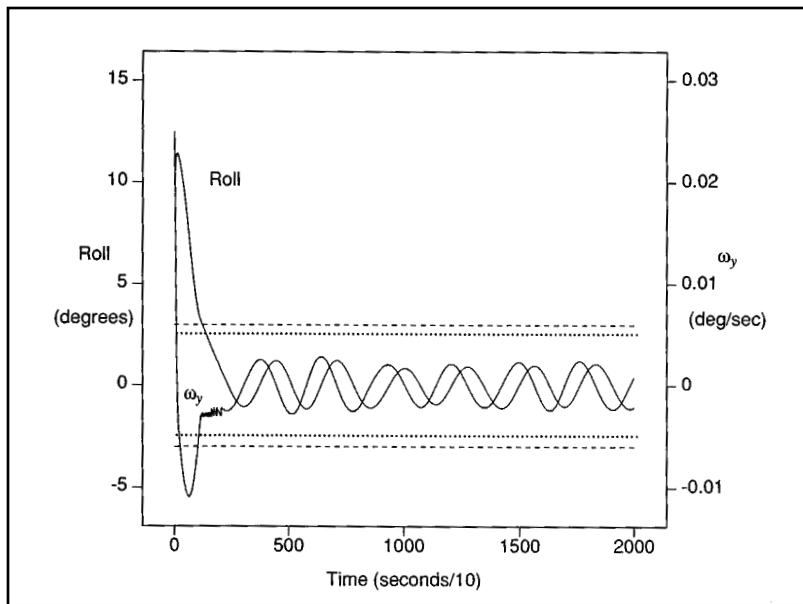


Figure 3. Y Axis (roll) Behavior of Satellite.

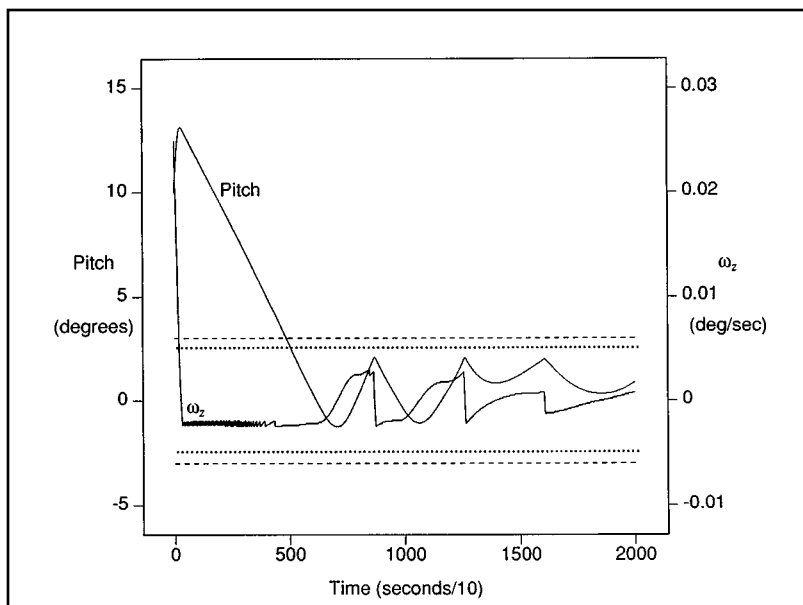


Figure 4. Z Axis (pitch) Behavior of Satellite.

equations used in the model that we did not have access to.

It might be expected that a good controller could damp out all oscillations in the system. This idea is clearly not supported in the graph in figure 2. However, in a realistic situation, it is never possible to do so. For example, the period of the oscillations is approximately equal to one orbit time, indicating that the satellite suffers from varying

amounts of atmospheric effects in different parts of the orbit. Also note that there are irregularities in the period of oscillation. These irregularities are caused by simulated disturbances in the model from both changes in the inertia of the spacecraft and external influences, as described in The Black Box. Despite these disturbances, the control strategy keeps the spacecraft stable.

The behaviors in the y and z axes were similar and are shown in figures 3 and 4. Note that the pitch decreases slower than either yaw or roll. An earlier and more extreme form of this feature led us to use more thrust in the z axis than in the other two axes.

Conclusion

These experiments show that rule-based methods can be used effectively to control complex physical systems. The form of the rule previously reported appears general and robust. However, further experimentation is required to determine the extent of the generality and robustness.

One of the most notable features of the work is that the form of the rules was generated by learning to control a different and somewhat easier task. As algorithm designers, we felt inclined to map the new and unfamiliar task onto a familiar one and, thus, make possible the reuse of existing representational tools. Analogic problem solving of this kind is recognized as a common feature in cognition. Eliot (1986) discusses it in the context of expert systems and notes both benefits and risks. In our particular case, analogy proved to be a powerful aid, starting us along the right general track. However, it also imported a false preconception, namely, that the interrogation of state variables should proceed in strict sequence. However, we already knew that the problem was one of motion in three dimensions of space, and the client treated these dimensions separately to the extent that it installed a separate sensor-effector system for each dimension (see The Black Box). This consideration, together with the rapid cut-and-try experimentation facilitated by the simulator, enabled us almost immediately to throw off the sequential bias that an otherwise helpful analogy had imposed. The rest followed naturally. Introducing the null action and softening the severity of pure bang-bang control then seemed simple, even obvious steps. The experimental runs (all of which are reported here) required less than two days of cooperative work to carry out the

successive cycles of execution, analysis, discussion, modification, and reexecution.

Directly learning on the spacecraft model would have been more time consuming than on the pole-and-cart simulation because there are more control variables and more actions. However, there has been considerable improvement (Bain 1990) in the reinforcement learning system of BOXES, so that for a more complex satellite control task than reported here, it could well be worth making use not only of the BOXES representational structure but also its adaptive learning facilities.

References

- Anderson, C. W. 1987. Strategy Learning with Multi-layer Connectionist Representations. In *Proceedings of the Fourth International Workshop on Machine Learning*, 103–114. San Mateo, Calif.: Morgan Kaufmann.
- Bain, M. 1990. Machine-Learned Rule-Based Control. In *Knowledge-Based Systems in Industrial Control*, eds. M. Grimble, J. McGhee, and P. Mowforth, 222–244. Stevenage: Peter Peregrinus.
- Barron, R. L.; Davies, J. M.; Schalkowsky, S.; and Snyder, R. F. 1965. Self-Organising Adaptive Space Vehicle Attitude Control. Presented at the AIAA/ION Guidance Control Conference, August, Minneapolis, Minn.
- Barron, R. L.; Davies, J. M.; Schalkowsky, S.; and Snyder, R. F. 1964. Self-Organising Adaptive Systems for Space Vehicle Attitude Control. Presented at the SAE-18 Committee Meeting, December, Miami Beach, Fla.
- Connell, M. E., and Utgoff, P. E. 1987. Learning to Control a Dynamic Physical System. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 456–460. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Eliot, L. 1986. Analogical Problem Solving and Expert Systems. *IEEE Expert* 1(2): 17–28.
- Glicksman, J. 1986. A Simulator Environment for an Autonomous Land Vehicle. In *Intelligent Simulation Environments*, eds. P. A. Luker and H. H. Adelsberger, 53–57. San Diego, Calif.: Simulation Councils.
- Luker, P. A., and Adelsberger, H. H. 1986. *Intelligent Simulation Environments*. San Diego, Calif.: Simulation Councils.
- Makarovic, A. 1987. Pole Balancing as a Benchmark Problem for Qualitative Modelling, Technical Report DP-4953, Joseph Stefan Institute, Ljubljana.
- Michie, D., and Chambers, R. A. 1968. BOXES: An Experiment in Adaptive Control. In *Machine Intelligence 2*, eds. E. Dale and D. Michie, 137–152. Edinburgh: Oliver and Boyd.
- Sammut, C. 1988. Experimental Results from an Evaluation of Algorithms That Learn to Control Dynamic Systems. In *Proceedings of the Fifth International Conference on Machine Learning*, 437–443. San Mateo, Calif.: Morgan Kaufmann.
- Selfridge, O. G.; Sutton, R. S.; and Barto, A. G. 1985. Training and Tracking in Robotics. In *Proceedings of the Ninth International Conference on Artificial Intelligence*, 670–672. Menlo Park, Calif.: American Association for Artificial Intelligence.



Claude Sammut is a senior lecturer in the School of Computer Science and Engineering at the University of New South Wales, where his research interests are primarily in machine learning, especially skill acquisition and inductive logic programming. He received his Ph.D. in AI from the University of New South Wales and has worked at Saint Joseph's University, Philadelphia; the University of Illinois at Urbana-Champaign; and the Turing Institute, Glasgow.



Donald Michie is a specialist in knowledge-based systems. Educated at Rugby School and Balliol College (an Open Classics Scholar at Oxford University), he served in World War II with the Foreign Office at Bletchley Park from 1942 to 1945, where he worked on the COLOSSUS code-breaking project and became acquainted with computer pioneers such as Alan Turing, M. H. A. Newman, and T. H. Flowers. After a biological career in experimental genetics (he received the Pioneer Award of the International Embryo Transfer Society), he developed the study of machine intelligence at the University of Edinburgh as director of experimental programming (1963–1966) and professor of machine intelligence (1967–1984). He is editor in chief of the series *Machine Intelligence* (Oxford University Press), the first 12 volumes of which span the period 1967–1991. Since 1986, he has been chief scientist at the Turing Institute, which he founded in Glasgow in 1984. Michie was elected a Fellow of the Royal Society of Edinburgh in 1969, a Fellow of the British Computer Society in 1971, and a Fellow of the American Association for Artificial Intelligence in 1990.