Ashby's Homeostat in Simulation

Alice Eldridge
Adaptive Systems Project
ace21@cogs.susx.ac.uk

April 22, 2002

Abstract
Understanding adaptation in the nervous system respresents an ongoing concern of many areas of science and philosophy. In the 1950's, the cybernetician W.Ross Ashby built the 'homeostat', an electro-mechanical device, to illustrate his (logical) solution to the problem: 'Adaptation through internal ultrastability'.

The homeostat is implemented in simulation, and the relationship between stability and connectivity as a function of system size is examined. In general, stability is found to decrease as an inverse function of the number of interconnected units. For small numbers of units, the relationship is curve-linear, whilst for larger numbers, a step function is approximated.

It is noted that Ashby's legacy has received surprisingly little subsequent treatment in disciplines to which it has seemingly strong relevance. Possible reasons for this are discussed, and the place of Ashbian homeostasis in current adaptive systems research considered.

1 Introduction

The problem of how the brain produces adaptive is an ongoing preoccupation of a great many schools of intellectual inquiry. Rather than focusing the problem with contrasting approaches, treatment of the issue by philosophers, psychologists, physiologists and more recently computer scientists, converges to create a fundmental impasse: how can physiological determinism be reconciled with seemingly purposeful adaptivity? *Design for a Brain* [2](*DfaB*) is Ashby's answer to this problem.[1]

Ashby equates adaptive behaviour with 'stable' behaviour, ie behaviour that persists in the face of (minor) environmental variation. More specifically an agent's of behaviour is considered adaptive if it maintains it's essential variables within physiological limits.

Care is taken to differentiate between two forms of adaptation that occur on different time scales and are subserved by two very different processes. One is an evolutionary adaptation: the (phylogenetic) development of certain mechanisms - such as the development of shivering as an innate response. This is assumed to be mediated by natural selection. The other, occurring on an ontogenetic time scale, is the realisation of the properties of this mechanism induced by a change in variable: the onset of shivering with a reduction in temperature. It is this processes with which Ashby is principally concerned.

The problem of how the brain produces adaptive behaviour is decomposed into two subproblems:

1. The identification of the nature of change which is shown in learning.

2. The investigation of why such changes should

[1] References to DfaB are given, following Ashby, using section refs. eg (3/2) refers to chapter 3, section 2

tend to cause better adaptation.

These are familiar problems, as they are central to the design of artificial nervous systems, as well as the epistemologically purer pursuits of cognitive science and philosophy.

Despite this similarity between current concerns and Ashby's preoccupation, seemingly little attention has been paid to his solution. His ideas remain largely unexplored. Here, his solution is briefly reviewed, and the generality of his proposed mechanism of ultrastablity is investigated by implementing a simulation of the homeostat which he built as an illustrative example. Conclusions from the experiments carried out are drawn, and possible reasons for the lack of consideration of his work are discussed.

1.1 Ultrastability and the Homeostat

Ashby proposes that adaptive behaviour can be explained through the concept of ultrastability. Essentially, adaptation is seen to imply some form of internal homeostasis. In accordance with Cybernetic thought, the adaptive system consists of the interaction between the organism and the environment. Ultrastability relies upon an double feed back between environment and 'reacting part' of organism: the first is a direct feedback between reacting part and environment meditated by the sensory and motor channels of the organism. The second feedback to the environment exists through certain (essential) continuous variables which affect *step mechanisms* when they are beyond some physiologically relevant limit. The change in step mechanism then affects the response of the reacting part to subsequent environmetal stimuli. Essential variables are those "closely related to survival". The physiological correlate of the step mechanism is not established. All that is necessary is some mechanism which can be affected by the environment, and can affect the reacting part, determining its' future reaction to the environment. The homeostat was built to demonstrate the properties of this basic form, thereby obviating the need for a 'tedious and unconvincing' description.

Ashby's Homeostat The homeostat consisted of four units, each topped by a pivoted magnet. The main variables are provided by the angular deviations of these magnets, which also approximates the output. These main variables represent both the reacting part and the environment and the interaction provides the primary feedback. The inputs to each unit are the outputs from all connected units, in addition to a recurrent connection. The magnitude and sign of these signals were mediated by commutators and potentiometers, the values of which represent the system parameters. When the angle of deviation exceeds a certain value, a contact (representing the essential variable) closed, inducing randomisation of the parameter values.

2 The Simulation

Ashby's description of the homeostat served as an illustration to avoid the tedium of a verbose description of the mechanism of ultrastability. As such the details were far from the level required for an implementation - physical or otherwise. Because it was not immediately clear which aspects of the physics of the homeostat would be crucial and which irrelevant, an initially cautious approach was adopted and eg constants included wherever they may be necessary. The necessity of variables and constants and their values were established through repeated experimentation with a simple two unit homeostat.

2.1 Algorithm and Assumptions

The basic features of the homeostat were trivial to program: each unit was represented as a structure. The commutators and potentiometers were modelled as weights and their signs. The input to each unit is the summed output from of all

units (including a recurrent connection), so was assigned the same memory address to speed the simulation (LINE 286). The main output algorithm used was as follows:

$$O_i = \text{Target}_i - \left[\sum \text{input}_{ij} \text{ x weights}_{ij} \text{ x On}_{i,j}\right]$$

where O_i = Output of the i_{th} unit at time $_{t+1}$
and input_{ij} = is the output from the other $(_{j-1})$ units plus output_i, at time $_t$

At the start of each run, all weights and outputs were initialised. Weights were selected randomly from the range -1:1, outputs were initially set within the limits of the essential variables to allow the weights to stabilise. At each iteration, the output (ie deviation from target value) of each unit was checked: if outside the critical range, weights were randomised to all connected units. The recurrent connection weight remained constant thoughout each run : Ashby's homeostat did not have a uniselector on the recurrent connection. In the original Homeostat, the frequency of uniselector action (ie weights change) could be varied, and was held at three seconds. The size of this interval did not seem to critically affect the performance of the simulated homeostat and was held at 1, allowing weight change at every step, for all experiments.

The 'ON' array was used to implement different levels of connectivity thus the *ith* element was always set to 1 for the *ith* unit and the remaining connections were randomised.

Critical Constants In calculating the output I initially included two constants as Ashby writes "the torque on the magnet is *approximately* equal to the algebraic sum of the currents in A, B, C".(8/2) , similarly, the output was described as "approximately proportional to (the needles') deviation from zero". In the physical homeostat, these would have been determined by properties of the materials, and although constant, their values unknown. In this simulation however, experimentation proved that once all other variables were set to suitable values, these two constants were redundant and thus set to zero [2].

Viscosity The viscosity of the liquid in the real homeostat would have had a dampening effect on the deviation of the magnet. Initially then the viscosity was represented by a constant below zero multiplying the main output calculated. More convincing behaviour resulted however, from the use of a more abstract 'maximum change' variable that limited the amount by which anyone units' output could change. This is in keeping with the fact that in simulation the activity of the homeostat is discrete rather than continuous. The effect of changing the value of this variable proved similar to the assumed effect of varying viscosity of the liquid: low values (representing high viscosity) reduced the influence of the each unit on the others, high values produced chaotic behaviour as each unit did not have time to achieve stable parameter settings before runaway occurred.

3 Results

3.1 Initial Replication: The homeostat as adapter

In order to establish suitable values for the variables in the simulation, a simple fully connected two unit homeostat was examined. The values of the parameters (ie weights) are reported to crucially affect the stability of the functioning homeostat (8/2) - producing stable or runaway behavior, although Ashby gives no indication of the expected proportion of each type of behaviour. It was therefore necessary to run the simulation many times to ascertain whether the behaviour was due to the parameter settings peculiar to that trial, or the values of the variables which determined the structure of the machine.

[2] Critical parameter values used can be found in appendix 2

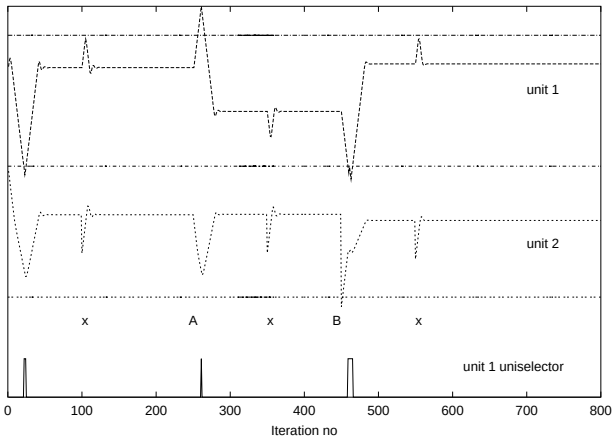

Figure 1: Main values for units 1 and 2 demonstrating adaptation in a simple homeostat

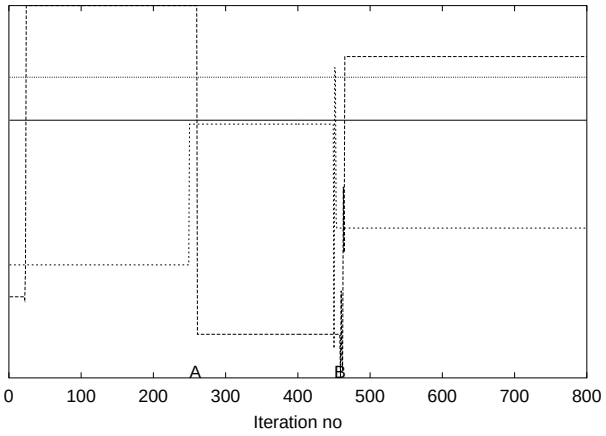

Figure 2: Parameter values for units 1 and 2 demonstrating adaptation in a simple homeostat

The basic features of adaptive behaviour in the homeostat are: resistance to minor perturbations (ie changes in essential variables within critical boundaries), and a change in step mechanism as a result of either weight change, or major perturbation (outside the critical boundaries). This change in step mechanism will then invert the response of one unit to a minor perturbation in a connected unit.

Figure 1 shows the main values for a homeostat consisting of two fully interconnected units. Figure 2 shows the weight changes for the same trial. The behaviour of the homeostat is represented as in *DfaB*: the main variables of each unit are presented divided by a line signifying the critical range. Following an initial period of runaway behaviour, unit 1's uniselector is activated (shown on the bottom line), and the two units reach a stable state. The corresponding weight change can be seen in figure 2. At each point marked 'x', the main variable of unit 2 has been perturbed downwards (manually). The initial downstroke of unit 2 causes an opposite reaction in unit 1. At point A, the sign on the weight of the connection 2$\rightarrow$ 1 is reversed. This causes the main variable of unit 1 to exceed its critical limit. The corresponding weight changes can be observed in figure 2.

In this example, the homeostat regains stability after a single change of weights (signified by the upstroke in the 'uniselector' line). For other runs - according to the weight values at initialisation and uniselector randomisation - stability was either more difficult (requiring several weight changes), or impossible to achieve. Similarly, for certain parameter settings, the weight change had no effect: this occurred when the weights were near zero, and so both the influence of unit 2 on unit 1, and the magnitude of the weight change were small enough to be resisted.

At the second 'x', the downstroke in unit 1 induces an upward movement in unit 2, confirming that the effect of one unit on the other has been reversed. At point B, unit 2s' main value is forced outside of its' critical range. This again causes the main variable of unit 1 to exceeded its' limit, invoking the uniselector action. This time two weight changes are need to regain stability (see figure 2 and double uniselector stroke in figure 1). Following the step mechanism change, unit 1s response is once more opposite to the movement

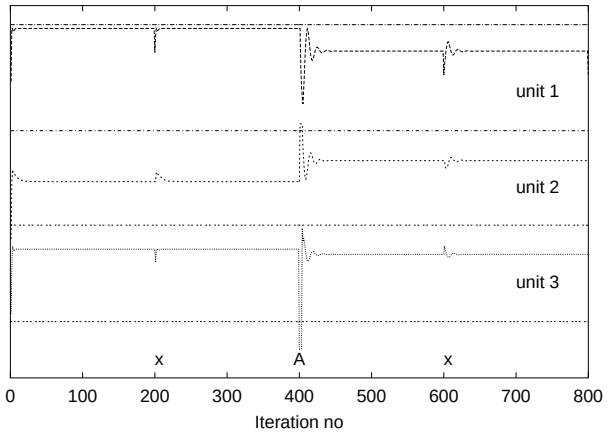

Figure 3: Main values for units 1 and 2 demonstrating training in a simple homeostat

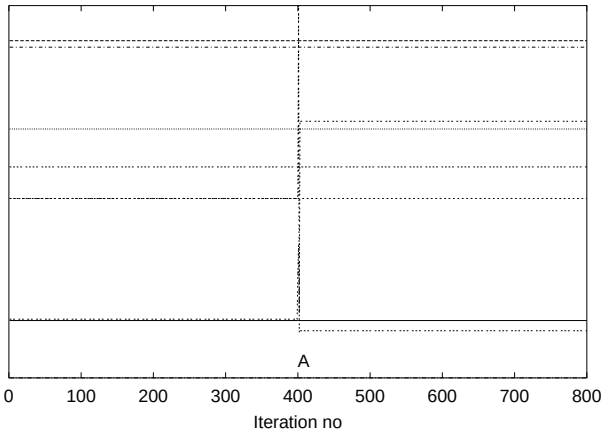

Figure 4: Parameter values for units 1 and 2 demonstrating training in a simple homeostat

induced in unit two. (figure 1 iteration 550). This demonstrates the basic self-reorganisation powers of the homeostat.

3.2 Training the Homeostat

Ashby identifies the behaviour of an animal in 'training' with that of an ultrastable system adapting to another system of fixed characteristics. (8/9). This experiment is a replication of the 'training' demonstrated on the original homeostat. Three units, were unidirectional connected 1→2, 2→3 and 3→1. The desired response of the homeostat is to follow a forced (trainer-induced) movement in unit 1, by a similar response in unit 2. If the 'incorrect response' is given (ie unit 2 moves in the opposite direction), the homeostat is 'punished' by unit 3's main variable is forced outside of the critical range. An example of behaviour under these conditions is shown in figures 2 and 3.

Figure 3 shows the main variables during the experiment, figure 4 the corresponding weight changes. The induced downstrokes of unit 1 are marked 'x'. At the test, unit 2 moves in the opposite direction as unit 1. This is an undesired response, so at 'A', unit 3 main variable is forced outside of the critical range. This causes runaway behaviour in unit 2, inducing a change in weights. The change of units 2 and 3 weights can be seen in figure 4. Subsequently, unit 2 responds, as required, in the opposite direction to the induced downstrokes of unit 1.

3.3 Increasing the Number of Units

Although the behaviour of the small homeostats investigated above is interesting, in order to examine the possibility that the mechanisms of ultrastability can explain adaptation in complex systems such as the brain, the effects of increasing the number of units must be considered. The above experiments were run using different numbers of units. (this demanded an alteration of the 'maxchange' variable inversely proportional to the number of units so that behaviour could be examined). For fully connected homeostats, percentage of times stability was achieved initially, or regained after major perturbation, decreased as the number of units was increased. Similarly, the time taken to achieve or regain stability increased. The effect of unit number on stability to

perturbation was similar to that on initial adaptation time. These results are summarised in figures 3 and 4.

In order to gain a more detailed understanding of the effects, of size on behaviour, stability as well as convergence was recorded [3]. The system was considered to have converged when the main variables did not fluctuate beyond a predefined minimal range. Stability refers to cessation of 'uniselector' action, ie the point at which the homeostat stops looking for 'new weights'.

Figure 3 shows the percentage of 50 trials for the which the system converged and stabalised. [4] it is evident that both measures of behaviour decrease with an increase in the number of units. The increase in relative difference between stability and convergence with an increase in size, represents the fact that as the number of units increases, the behaviour of the system becomes less stable in a more general sense: even when the main variables are within the critical limits, there is an increased tendency to oscillation and random fluctuation.

Figure 4 shows the average stability and convergence time for trial in which these measures were achieved in the same set of trials. As we may expect, where stability is achieved, larger systems take an increasingly long time to settle, both in terms of convergence, and stability in the Ashbian sense.

3.4 Decreasing Connectivity

The above results suggest that even a homeostat consisting of just 12 units rarely stabilised. This implies that ultrastability breaks down as an adaptive mechanism for systems many orders of magnitude smaller than any natural complex system. However, few natural system are fully

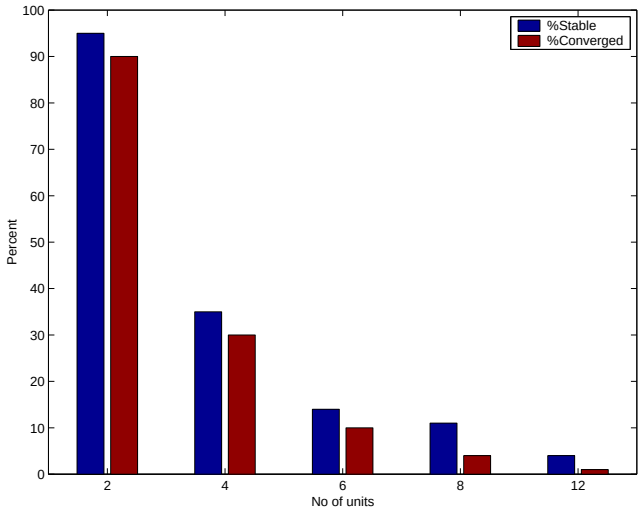

Figure 5: Variation in percentage stability and convergence time with number of units

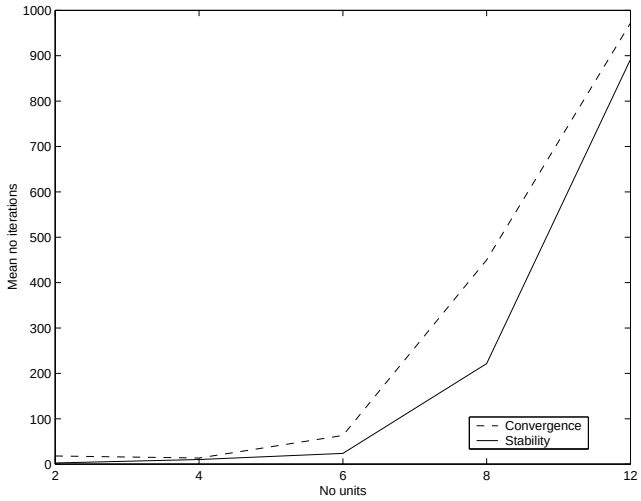

Figure 6: Variation in mean stability and convergence time with number of units

[3] Here,'maxchange' was of course held constant to enable comparison.

[4] The simulation was run for a maximum of 1000 runs as in initial trials, when stable parameter settings were discovered, this was always achieved within this time.

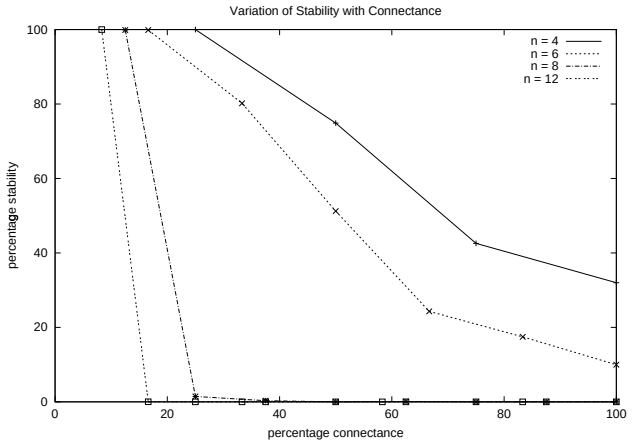

Figure 7: Variation of stability with connectance

connected. Figure 5 shows the percentage of runs achieving stability varies with percentage connectance for different number of units.[5]

From the results shown in figure 5, two things are evident. Firstly, as the connectance decreases, so the probability of stability increases. Secondly, and of greater interest, the nature of the relationship between connectance and stability seems to change as the number of units increases. For small configurations, up to about $n = 6$, the probability of stability decreases steadily with an increase in connectance. However for larger values, this relationship approaches a step function. These results suggest for larger systems, there is a critical value (here somewhere between 8% and 16% for a 12 units homeostat), either side of which stability is assured or impossible.

[5] percentage connectance is simply the number of inputs to each units divided by the number of units. thus for $n = 4$, there are only 4 values 25%, 50% etc

4 Discussion

4.1 Conclusions from experiments

The simulated homeostat demonstrates that a mechanistic, fully state determined system can produce adaptive behaviour. Experiment 1 shows that the system can maintain behaviour, resisting minor environmental perturbation, and self-organise following radical change. Experiment 2 shows a simple form of 'training', where individual units can be incited to produce a response of choice. This demonstrates the ability of the system to display apparent purposeful behaviour mediated by 'purposeless' (random) parameter change.

Investigation of the effect of increasing the number of units on stability and convergence time, showed that even for trivially small systems of 12 units, the probability of stability was slight. This could be seen to call into doubt the applicability of ultrastability as an adaptive mechanism for complex natural systems such as the brain, the explanatory object of its original formulation.

An examination of the effect of reducing connectivity, however, shows that for systems large enough to be unstable when fully connected, a high probability of stability can be virtually ensured if connectivity is sufficiently sparse. Of greater interest is the qualitative change in the relationship between percentage connectance and stability with an increase in the number of units. As reported, the curve-linear relationsip for small systems changes to approximate a step function for systems of more than about 10 units. These results are similar to those reported by Ashby in [3], and suggest that for complex systems, there may be a critical connectance level, either side of which stability is ensured or impossible.

Ashby's concept of dispersion, and this critical level of connectivity is reminiscent of Kauffman's frozen yet percolating clusters that give 'order for free'[6]. The phenomena provokes consideration of analogy with neural organisation, but the anal-

ogy is far from complete.

The relationship between units is linear in this simulation, as in Ashby's original homeostat. If we wish to pursue the analogy, the first step must be to examine the behaviour of a non-linear homeostat - for example where units interact according to a sigmoid function.

4.2 Lack of Ashbian Influence

A simulation of Ashby's homeostat exhibits the main characteristics of the original device, lending support to his proposal that adaptive, and seemingly purposeful behaviour can be generated from by a fully state-determined system.

Psychologists and philosopher of mind have long sought an answer to this very conundrum. In the past few decades, approaches such as dynamical systems theory have been explored, and ideas from 'Chaos Theory' hastily embraced, yet the literature shows few signs of reference to Ashby's solution. [6] Similarly, the specific questions that Ashby addressed (i.e. the nature of change which is shown in learning, and why such changes should tend to cause better adaptation), are a core focus of a-life research in simulation of artificial nervous systems. Although traces of the idea that adaptation is based on some sort of internal homeostasis is discernable in the field, few explicit references are made to the concept of ultrastablity: his legacy has not been systematically pursued or embodied in this field. Here, the exception being Di Paolos [4] work on simulating adaptation to visual inversion, which represents a rare attempt to rejuvinate Ashby's ideas in the context of current evolutionary robotics.

"It may be a beautifully replica of something, but heaven only knows what" *Mathematician Julian Bigelow at the Macy meeting 1952.*

[6] the obvious exception is the work of Taylor on adaptation to visual inversion [9], surprisingly, given strength of experimental results, this did not renew interest in Ashby's legacy - although this may be due to Taylors interpretations

Given the relevance of Ashby's work to current problems in the cognitive sciences, it is perhaps surprising that there is so little evidence of his influence. His relative anonymity in these fields may be attributable partly to historical reasons, or be represent more fundamental problems of the account in terms of generality and .

4.3 Historical Reasons

AI, in contrast to cybernetics that had close links with the life sciences, identified 'intelligence' (adaptivity) strongly with logic. From the outset, the approach focused on what was essentially a subproblem of Cybernetics - finding mechanical ways for solving 'cognitive feats'. Led by Newell and Simon, there was a strong insistence on the processing of symbols by rules eg[8]. The system of interest was therefore purely 'psychological' - a subsystem of organism-environment system that was central in cybernetics. The notion of adaptivity itself was divided and the principle concern was with 'intelligent' behaviours.

Within this approach there is little room for notions of feedback between different systems of continuous variables. The fervor and promise surrounding AI largely eclipsed the insights of Cyberneticians such as Ashby. Although the importance of feedback as a homeostatic mechanism has been well established in biology, the reinforcement of Cartesianism split and enthusiasm for computational metaphors of mind left these mechanism largely ignored.

If early AI contributed toward the demise of cybernetic thought 'nouvelle AI' and the development of artificial evolutionary approaches to understanding adaptation did little to resurect Ashby's work. The simultaneous development of the field of evolutionary biology and increases in computational power fuelled the conception of the evolutionary algorithm. The work of Holland [5], elected the concepts of mutation and recombination as the principle effectors of change in natural organisms: consideration of mechanisms of onto-

genetic adaptivity was eclipsed by investigation of the phylogentic, evolutionary change. This was reinforced by a strong group of NeoDarwinians in theoretical biology, demonstrating the power of natural selection in shaping organismic adaptations on an evolutionary time scale. Mechanisms of ontogenetic change were comparatively neglected.

4.4 Problems with the Account

What does essential mean ? Ashby is deliberately elusive as to the exact physiological correlate of 'essential variables'. Whilst in some examples he describes physiological factors such as body temperature (3/14) in other cases they are not physiological (17/4). I believe that some scepticism over his formulation of adaptation comes from misinterpretation of his intended level of analogy. For example, Arbib discredits the concept of homeostatic adaptation, as he believes that "the overall state of the brain is constantly developing. There is no state to which it constantly returns". [1].

If 'essential variable' refers to a particular brain state, then this argument may have some weight. However, the concept of essential variable is perhaps more properly treated in a more abstract sense. Indeed in the simulation, as in the real homeostat, as Di Paolo points out, the variables are not essential in themselves, rather they exist as tokens or proxies for essential conditions. So the essential variables may not necessarily represent a physiological mechanism, but be usefully considered as an ecological invariance. Under this interpretation, there is no contradiction between homeostasis as a mechanisms of adaptation, and Arbib insistence that the brain has no 'goal state'.

Accepting this looser definition of essential variable, as one which is "closely related to survival" (3/14), rather then determining survival, the variables are perhaps improperly described as *essential*. However, in using the term more loosely, Ashby's definition of adaptation, which conflates other notions of adaption and viability [7]pp xx-xxi - escapes some of the problems from which they suffer. Principally, it leaves room for the prospect of the existence maladaption in a living organism where these other accounts the only alternative is death.

Whilst the relationship between each essential variable and agents' survival is flexible, the relationship between each of them is prescribed: "They are closely linked dynamically, so that marked changes in any one sooner or later leads to marked changes in the others." (3/14). This allows the possibility for systems where changes in a non-terminal essential variable, beyond the critical range, can induce step-mechanism changes so as to reduce the probability of the truly essential, life-defining essential variables going out of limit.

In biological agents, such a configuration is exemplified by pain mechanisms. Up to a certain limit, pain increases in proportion to the harmfull effects of the environment, which will similarly push truly essential variables such as eg blood loss toward their critical boundary. However, pain only increases to a certain limit, beyond which opiate release has takes over, acting to pacify the organism, enforcing a relaxed state to enable recovery. [7] In this way the essential variables themselves interact to produce another homeostatic mechanism to maintain adaptation. Such a mechanism could have pragmatic use for autonomous adaptive agents design in enginering applications.

Whilst the concept of ultrastability provides a stimulating analogy for understanding adaptation in living organisms, the analogy is by no means direct nor complete. The obvious immediate requirements are explanations of the mechanism in terms of evolutionary history as well as examination its' plausibility in terms of physiological substrates.

[7] arguably the mechanism switches when there is 'no hope', and thus simply makes death less painful, but we shall ignore this interpretation!

In terms of scope, the connection between ultrastability and adaptation is perhaps hypothetical [4], and can only account for a sub-set of all possible instances of adaptation. That patterns of agent/environment interaction persist in the face of minor environmental change, implies the maintenance of some ecological invariant: internal stability provides a satisfactory but not exclusive explanation: the stability could be due to plastic changes outside of the organism, for example in another organism or the external environment.

4.5 Conclusion

The concept of ultrastability exists as a persuasive deduction, electro-mechanical demonstration and functional algorithm. It demonstrates a mechanism by which seemingly purposeful behaviour can result from behaviourally purposeless physiological mechanisms. As such the homeostat exists as a proof of concept of a problem that dogs psychologists and cognitive scientists to date.

Demonstration of the powers of homeostatsis in adaptation should similarly warm the cockles of the hearts of proponents of strong Alife, as it suggests

"that we must give up our naive conviction that the outstanding behavioural properties of adaptation indicate some unique cerebral mechanism, or that they will provide the unique explanation of the features of living brain. Many of these features cannot be realted uniquely to the processes of adaptation, for theses processes can go on in systems that lack those neurological features, systems very different form the living brain, such as the modern computer".(10/13)

Even if the concept of ultrastability is not a universal panacea for our epistemological needs, Di Paolo's pioneering work has demonstrated that an Ashbian concept of homeostasis can be usefull applied in the context of current adaptive systems research. To date evolutionary robotics has predominantly focused on phylogenetic adaptation, the next development must be investigation of life time learning, and Ashby's notion of homeostatic adaptation, even if it is not a universal panacea as it stands, provides fertile ground for further investigation.

References

[1] Arbib, M., A. (1989). *The Metaphorical Brain 2. Neural Networks and Beyond.* John Wiles and Sons.

[2] Ashby, W.R. (1960). *Design for a Brain*

[3] Ashby, W.R. (1970) Connectance of large dynamic (Cybernetic) Systems: Critical Values for Stability. *Nature* 228:784.

[4] Di Paolo E.A. (1998). Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions. In J.A. Meyer, A. berthoz, D. floreano, H. Roiblat, S.W. Wilson (Eds). *From Animals to Animats 6: Proceedings of the Sixth Interntational Conference on Simuloation of Adaptive Behaviour*, Harvard, MA: MIT Press

[5] Holland, J. (1992). *Adapatation in natural and artifical systems.*

[6] Kauffman, S. (1993). *The Origins of Order* Oxford University Press

[7] Maturana, H. and Varela, F.J. (1980). *Autopoeises and cognition: The realisation of the living.* Dordrecht, Holland: D. Reidel Publishing.

[8] Newell, A. and Simon, H. (1972) *Human Problem Solving.* Prentice-Hall.

[9] Taylor, J.G. (1962). *The behavioural basis of Perception* New Haven: Yale University.

Appendix 1

Description of the Homeostat. DfaB (8/2)
"The Homeostat consists of four units, each of which carries on top a pivoted magnet. The angular deviations of the four magnets from the central position provide the four main variables.

Its construction will be described in stages. Each unit emits a D.C. output proportional to the deviation of its magnet from the central position. The output is controlled in the following way. In front of each magnet is a trough of water; electrodes at each end provide a potential gradient. The magnet carries a wire which dips into the water, picks up a potential depending on the position of the magnet, and sends it to the grid of the triode. J provides the anode potential at 150 V., while H is at 180V,; so E carries a constant current. If the grid-potential allows just this current to pass through the valve, then no current will flow through the output. But if the valve passes more or less, current than this, the output circuit will carry the difference in one direction or the other. So after E is adjusted, the output is approximately proportional to M's deviation from its central position.

Next, the units are joined together so that each sends its output to the other three. These inputs act on the unit's magnet through the coils A, B, C, so that the torque on the magnet is approximately proportional to the algebraic sum of the currents in A, B, and C. (D also affects M as self-feedback). But before each input current reaches its coil, it passes though a commutator (X) which determines the polarity of entry to the coil, and through a potentiometer (P), which determines what fraction of the input shall reach the coil.

As soon as the system is switched on, the magnets are moved by the currents from the other units, but these movements change the currents, which modify the movements, and so on. It may be shown that if there is sufficient viscosity in the troughs, the four-variable system of the magnet-positions is approximately state-determined. To this system the commutators and potentiometers act as parameters.

When these parameters are given a definite set of values, the magnets show some definite pattern of behaviour; for the parameters determine the field, and thus the lines of behaviour. If the field is stable, the four magnets move to the central position, where they actively resist any attempt to displace them. if displaced, *co-ordinated* activity brings them back to the centre. Other parameter-settings may, however, give instability; in which case, a 'runaway' occurs and the magnets diverge from the central positions with increasing velocity - till they hit the ends of the troughs."

(Appendix 2)

Critical Parameter Values Used

WEIGHT RANGE (-1,1)

NOISE (-0.005, 0.005)

DEV (-0.05, 0.05)

MAXCHANGE (0.004)

A random number uniformly distributed between WEIGHT RANGE was used to simulate both 'commutator' and 'potentiometer'. To simulate achange in comutator then, the sum of the weights were simply changed, preserving the absolute value.

Similarly, a random value for in the range NOISE was selected on each run. This was included in the output calculations for versimilitude.

DEV determined the 'critical' range for outpus: output values outside of this range induced weight change.

MAXCHANGE was the final choice for simuation of viscosity. This was held constant for all experiments reported here. However, as mentioned, the initial experiments reported (effect of weight change and reaction to minor and major perturbation) were run using increased numbers of weights. In these cases, the value of MAXCHANGE was decreased for increasing numbres of units.

Appendix 3

Source Code

```
//------------------------------------------------------------------------------
//      SOURCE CODE FOR BASIC HOMEOSTAT
//      (extra data collection loops and amendments used to investigate effect of
//                              connectivity on stability not shown for clarity)
//------------------------------------------------------------------------------

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

typedef struct                  //structure for each unit
{
        double *weights;
        double *on;             //connex on/off
        double output;          //output at t;
        double **input;         //array of pointers, same loc as output(t)
        double noutput;         //output at t+1
        double *usel;           //simulates 'uniselector' if (weight change = 1) else(=0)
        //parameters
        int nconnex;            //no. of connections
        int nunits;
        double target;
        double noise;           //added in output calaculation
        double v;               //viscosity c
        double k1;              //const for output
        double k2;              //const for each input
        double dev;             //critical deviation
        double wrnge;           //weight range
        double maxdev;          //max deviation

} UNIT;


typedef struct                  //'structure for homeostat (entire system)'
{
        UNIT **unit;
        int  nunits;
        int nconnex;
        double target;
        double noise;
```

```
        double maxchange;
        int test;
        double v;               //viscosity
        double k1;              //const for output
        double k2;              //const for each input
        double dev;             //critical deviation
        double wrnge;           //weight range
        double maxdev;          //max deviation
        int maxruns;

} HOM;


#define NUNITS 12
#define NCONNEX 6
#define K1 1.0                  //*             *denotes constants initially included
#define K2 1.0                  //*              that experimentally proved uneccessary
#define DEV 0.05
#define MAXCHANGE 0.004
#define WRNGE 1.0
#define NRNGE 0.005             //range for noise
#define MAXDEV 10.0
#define TARGET 0.0
#define MAXRUNS 1000
#define FILENAME "hom.dat"
#define FILENAMEW "homw.dat"

#define TEST 1                  //frequency with which weight change allowed
#define V  1.0                  //*

void InitHom(HOM *hom);
void InitWeights(HOM *hom);
void RandWeights(UNIT *unit, int n);
int CalcOutput(HOM *hom);
void InitOutput(HOM *hom);
void RunHom(HOM *hom, int maxruns, FILE* fp, FILE *fpp);
void FreePtrs(HOM *hom);
//DEBUG FUNCTIONS
void PrintCnx(HOM *hom);
void PrintWeights(HOM *hom);


int main(int argc, char **argv)
{
```

```
        int i,j;
        FILE *fp;
        FILE *fpp;
        time_t t;
        HOM *hom = malloc(sizeof(HOM));
        fp = fopen(FILENAME, "w");
        fpp = fopen(FILENAMEW, "w");

        srand48(time(&t));

        InitHom(hom);
        InitOutput(hom);
        InitWeights(hom);
        RunHom(hom, MAXRUNS, fp, fpp);

        fclose(fp);
        fclose(fpp);
        exit(EXIT_SUCCESS);
}
//------------------------------------------------------------------------
//ITERATES FOR MAXRUNS SAVES OUTPUT AND WEIGHTS EACH ITERATION, AND PRINTS
//CONVERGENCE AND STABILITY TIMES

 void RunHom(HOM *hom, int maxruns, FILE *fp, FILE *fpp)
{
        register int i, j, k, cnt = 0;
        int cp, conv;
        cp = conv = 0;

        for(i=1;i<=maxruns;i++)
        {
                cp = CalcOutput(hom);

                if (cp == 0)                                    //CONVERGENCE TEST:
                        conv = i;                               //UPDATES conv WHILST NOT CONVERG


                        //EXAMPLE EXPERIMENTAL 'INTERFERENCE CONDITIONS'
                        //exp2(3 units)
                        /*
                        if (i==200) hom->unit[0]->output -=(hom->dev/4);
                        if (i==400) hom->unit[2]->output = -1*(hom->dev*6/5);
                        if (i==600) hom->unit[0]->output -=(hom->dev/4);
```

```
                        if (i==800) hom->unit[0]->output -=(hom->dev/4);
                        */


                        if((i % hom->test) == 0)                //CHECK IF MAIN VALUE IN LIMITS
                        {
                                for(j=0;j<hom->nunits;j++)
                                        if(fabs(hom->unit[j]->output)>hom->dev)
                                        {
                                                cnt = i;        //COUNTER FOR STABILITY
                                                RandWeights(hom->unit[j], j);
                                                if (j==0) hom->unit[0]->usel[i] = (hom->dev/2);
                                        }

                        }

                                                                //DATA COLLECTION

                if(fp!=NULL)
                {
                        fprintf(fp, "%d\t%f ",i,  hom->unit[0]->usel[i]-(hom->dev*4));
                        for(j=0;j<hom->nunits;j++)
                                fprintf(fp, "\t%.20f", hom->unit[j]->output-(j*(hom->dev*1.0)));
                        fprintf(fp, "\n");
                }

                if (fpp!=NULL)
                {
                        fprintf(fpp, "%d",i);
                        for(j=0;j<hom->nunits;j++)
                                for(k=0;k<hom->nunits;k++)
                                        fprintf(fpp, "\t%.20f", hom->unit[j]->weights[k]);
                        fprintf(fpp, "\n");


                }


        }
        printf("Convergence = %d\n", conv);                     //FOR ESTABLISHING EFFECT OF CONN
        printf("Stability = %d\n", cnt);                        //AND CONVERGENCE, ON STABILITY,
                                                                //RETURNED TO CALLING FUNCTION
}
//-------------------------------------------------------------------------
```

```
//INITIALISES MEMBERS OF UNIT STRUCTURE AND SET CONNECTIONS

void InitHom(HOM *hom)
{
   register int i, j;
   register int tmp;

        hom->nunits = NUNITS;
        hom->unit = calloc(hom->nunits, sizeof(UNIT *));
        hom->nconnex = NCONNEX;
        hom->test = TEST;
        hom->v = V;
        hom->k1 = K1;
        hom->k2 = K2;
        hom->dev = DEV;
        hom->wrnge = WRNGE;
        hom->maxdev = MAXDEV;
        hom->target = TARGET;
        hom->noise = NRNGE;
        hom->maxchange = MAXCHANGE;
        hom->maxruns = MAXRUNS;

        for(i=0;i<hom->nunits;i++)
        {
                hom->unit[i] = malloc(sizeof(UNIT));
                hom->unit[i]->weights = calloc(hom->nunits, sizeof(double));
                hom->unit[i]->usel = calloc(hom->maxruns, sizeof(double));
                hom->unit[i]->on = calloc(hom->nunits, sizeof(double));
                hom->unit[i]->input = calloc(hom->nunits, sizeof(double *));
                hom->unit[i]->nunits = hom->nunits;
                hom->unit[i]->nconnex = hom->nconnex;
                hom->unit[i]->target = hom->target+i;
                hom->unit[i]->v = hom->v;
                hom->unit[i]->k1 = hom->k1;
                hom->unit[i]->k2 = hom->k2;
                hom->unit[i]->dev = hom->dev;
                hom->unit[i]->wrnge = hom->wrnge;
                hom->unit[i]->maxdev = hom->maxdev;
                hom->unit[i]->on[i] = 1.0;              //SET RECURRENT CONNEX
                hom->unit[i]->noise = drand48()*hom->noise;



                j=0;
```

```
                while (j<(hom->nconnex-1))              //RANDOMISE CONNEX
                {
                        tmp = rand() % hom->nunits;
                        if (hom->unit[i]->on[tmp] != 1.0)//DO NOT RE-SET RECURRENT CONNEX
                                {
                                        hom->unit[i]->on[tmp] = 1.0;
                                        j++;
                                }
                }
        }
//conex setting for ex 2
/*
        hom->unit[0]->on[0] = 1.0;
        hom->unit[0]->on[1] = 1.0;
        hom->unit[0]->on[2] = 0.0;
        hom->unit[1]->on[0] = 0.0;
        hom->unit[1]->on[1] = 1.0;
        hom->unit[1]->on[2] = 1.0;
        hom->unit[2]->on[0] = 1.0;
        hom->unit[2]->on[1] = 0.0;
        hom->unit[2]->on[2] = 1.0;
*/
                                //AUTO CONNECT OUTPUT AT TIME T TO INPUT AT T (used at t+1)
        for(i=0;i<hom->nunits;i++)
        {
                for(j=0;j<hom->nunits;j++)
                        hom->unit[i]->input[j] = &(hom->unit[j]->output);
        }
}
//-------------------------------------------------------------------------
//INITIALISES OUTPUT AND WEIGHTS FOR INITIAL ITERATION

void InitOutput(HOM *hom)
{
        int i;

        for(i=0;i<hom->nunits;i++)
        {
                hom->unit[i]->weights[i] = ((drand48()-0.5) * 2.0)*hom->wrnge;
                hom->unit[i]->output = (drand48()- 0.5) * 2.0 * hom->dev;
        }

}
```

```
//-------------------------------------------------------------------------
//CALLS RandWeights TO INITIAISE WEIGHT VALUES

void InitWeights(HOM *hom)
{
        register int i,j;

        for(i=0;i<hom->nunits;i++)
        {
                RandWeights(hom->unit[i], i);
        }

//exp 2
/*
        hom->unit[0]->weights[1]= -1* (drand48());
        hom->unit[0]->weights[2]= 0.0;
        hom->unit[1]->weights[0]= 0.0;
        hom->unit[1]->weights[2]= drand48();
        hom->unit[2]->weights[0]= -1*(drand48());
        hom->unit[2]->weights[1]= 0.0;

        hom->unit[0]->weights[1]= (drand48());
        hom->unit[0]->weights[2]= 0.0;
        hom->unit[1]->weights[0]= 0.0;
        hom->unit[1]->weights[2]= drand48();
        hom->unit[2]->weights[0]= -1*(drand48());
        hom->unit[2]->weights[1]= 0.0;
*/

}

//-------------------------------------------------------------------------
//RANDOMISES WEIGHTS WITHIN SPECIFIED RANGE (EXCEPT RECURRENT)
//USED WHEN OUTPUT OUT OF RANGE

void RandWeights(UNIT *unit, int n)
{
        register int i;

        for(i=0;i<unit->nunits;i++)
                if(i!= n)
                unit->weights[i] = ((drand48()-0.5) * 2.0)*unit->wrnge;

}
```

```
//-------------------------------------------------------------------------
//CALCULATES 'MAIN VALUES' (OUTPUT) ESTABLISHES AND RETURNS CONVERGENCE POINT

int  CalcOutput(HOM *hom)
{
        register int i, j;
        int conv, cnt;
        conv = cnt = 0;

        for(i=0;i<hom->nunits;i++)
        {
                hom->unit[i]->noutput = 0.0;

                                                        //SUMS INPUTS, ADDS NOISE
                for(j=0;j<hom->nunits;j++)
                {
                        hom->unit[i]->noutput += ((( *(hom->unit[i]->input[j])*
                                (hom->unit[i]->weights[j]) * hom->unit[i]->on[j] ) * hom->k2)
                                + hom->unit[i]->noise);
                }
                hom->unit[i]->noutput =  ((hom->target-(hom->unit[i]->noutput)) * hom->v ) ;

                                                        //IMPOSES MAX CHANGE
                if (fabs(hom->unit[i]->noutput) > hom->maxchange)
                        hom->unit[i]->noutput =
                                (hom->maxchange*(hom->unit[i]->noutput/fabs(hom->unit[i]->noutput

                                                        //COUNTER TO ESTABLISH CONVERGENCE POINT

                if (fabs((hom->unit[i]->noutput)-hom->unit[i]->output)< hom->dev/100)
                        cnt++;
        }
                                                        //UPDATES OUTPUT
                for(i=0;i<hom->nunits;i++)
                {
                        hom->unit[i]->output += hom->unit[i]->noutput;
                }

        if (cnt == hom->nunits)                         //IE RECORDS FIRST ITERATION WHICH ALL
                conv = 1;                               //UNITS CHANGE BY LESS THAN (ARBITRARY VA

        return (conv);
```

```
}
//-------------------------------------------------------------------------
//FREES MEMORY

void FreePtrs(HOM *hom)
{
        register int i;

        for(i=0;i<hom->nunits;i++)
        {
                free(hom->unit[i]->input);
                free(hom->unit[i]->on);
                free(hom->unit[i]->usel);
                free(hom->unit[i]->weights);

        }

        free(hom->unit);
        free(hom);

}
//-------------------------------------------------------------------------
//              DEBUG FUNCTIONS
//-------------------------------------------------------------------------

 void PrintCnx(HOM *hom)
{
        int i, j;

        for (i=0;i<hom->nunits;i++)
        {
                printf("unit %d conx:", i);
                for(j=0;j<hom->nunits;j++)
                        printf("%1.1f, ", hom->unit[i]->on[j]);
                printf("\n");
        }
        printf("\n");

}
//-------------------------------------------------------------------------
 void PrintWeights(HOM *hom)
{
        int i, j;
        for (i=0;i<hom->nunits;i++)
```

```
        {
                printf("unit %d weights:", i);
                for(j=0;j<hom->nunits;j++)
                        printf("%f, ", hom->unit[i]->weights[j]);
                printf("\n");
        }
        printf("\n");


}
```