Evolution of Two Symbol Signals by Simulated Organisms

A Thesis Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Joseph J. Crumpton

December 1994

Abstract

This research used synthetic ethology to investigate the evolution of simulated organisms that communicated information about their environment. Synthetic ethology involves creating a complete environment and simulated organisms which evolve in that environment. Since these simulated organisms exist as data structures in a computer program, their evolution can be studied more closely than the evolution of organisms in the natural world. The main goal of this study was to demonstrate the evolution of simulated organisms that used signals consisting of two symbols to communicate.

After creating a new environment for the simulated organisms, several factors affecting the evolution, such as population size and the learning rule used by each simorg, were investigated. It was concluded that the simulated organisms did successfully evolve the use of two symbol signals to denote situations within their environment.

Table of Contents

Chapter **Page**

1 Introduction..1

Motivation..1

Objective..2

2 Literature Review..3

3 Methods..6

Communication..6

Environment and Organisms..7

Cycles..12

Measurements..12

Research Areas..14

4 Results and Discussion..17

New Environment..17

Population Size..22

Internal States..26

Different Learning Rule..29

New Simorg Placement..29

Competitive Environment..29

5 Conclusion..34

Measurement Limitations..34

Further Study..34

Bibliography..36

Chapter **Page**

Appendices.. 38

Appendix A: Reproducing Single Symbol Communication.......39

Appendix B: Listing of Program....................................... 43

List of Figures

Figure **Page**

1 Environment of the Simulated Organisms 8

2 Sample Partial Phenotype of a Simorg 10

3 Illustration of Two Point Crossover 11

4 Denotation Matrix From MacLennan's Original Research 15

5 Average Fitness Smoothed Over 50 Major Cycles 18

6 Entropy Smoothed Over 50 Major Cycles 19

7 Sample Denotation Matrix from Base Version Used in this Study 20

8 Comparisons of MacLennan's Original Environment and the Environment of the Base Version Used in this Study 21

9 Comparisons of Different Population Sizes 23

10 Pair-wise Comparison of Different Population Sizes Using Student's t...24

11 Correlation between Lowest Entropy and Highest Fitness......... 25

12 Comparisons of Different Numbers of Internal States.............. 27

13 Pair-wise Comparisons of Different Numbers of Internal States using Student's t..28

14 Comparisons of Different Learning Rules............................30

15 Comparisons of Different Placement of New Simorgs............. 31

16 Comparisons of Base Environment and Competitive Environment..32

A-1 Communication Suppressed Denotation Matrix From MacLennan's Work.. 40

Figure		Page
A-2	Communication Suppressed Denotation Matrix From New Program	40
A-3	Comparison of Distribution of Actions in the Genotypes of the Population	41
A-4	Correlation Between Situations in Successful Cooperations and Actions in the Genotypes of the Population	42

Chapter 1

Introduction

Motivation

The study of communication has taken many forms. Mathematical modeling, laboratory studies, and ethological studies are but a few of the methods used. Ethology, the observation of organisms and their behavior in their natural environments, produces the most valid results. One disadvantage of ethological studies is the complexity of the organisms that are being studied. Any animal that exhibits the use of communication will possess a brain far too complex to understand through current neuroscience. Therefore our experiments will use synthetic ethology, the study of simulated organisms, to by-pass this problem.

The goal of synthetic ethology is to create the simplest organism and complete environment that produces the desired behavior and then to study the organism. In these experiments the environment and organisms will exist as data structures of a program. It is important to note that since the simorgs exist and evolve in this new environment, this environment is their "natural environment." A major point of ethology, both normal and synthetic, is to study animals within their natural environment.

A large difference between normal ethology and synthetic ethology is the amount of control the scientist has over the experiment. Synthetic ethology gives a scientist unparalleled control of the experiment while also not interfering in the environment in which the study is being done. The experiment can be stopped at any point and questions about why organisms behaved in a certain manner can be answered. Initial populations can be duplicated exactly and the results of changing the parameters of the experiment can be monitored closely. By keeping the environment of the simulated organism as simple as possible it is possible to examine what is essential to the process of communication.

Objective

The objective of this study was to create simple simulated organisms that demonstrate the evolution of communication with signals consisting of two symbols. Previously MacLennan had developed simulated organisms (referred to as simorgs) that used one symbol signals to communicate information about their environment to other simulated organisms (1990, 1992). Hopefully more complicated simorgs can be created that utilize more than one symbol in a signal. Eventually as the simorgs gain the ability to use more symbols in a signal, language features such as grammar and sentence structure may emerge. Once developed these simorgs can be studied to further our knowledge of the process of communication. The simorgs that can use two symbol signals as opposed to single symbol signals are the next step along that path.

Chapter 2

Literature Review

The study of communication and signaling has long been of interest to ethologists. Volumes such as E. S. Morton and J. Page's (1992) *Animal Talk* and T. A. Seboek's (1968) *Animal Communication* have been devoted to the techniques and results of studying communication. Many species of animals have developed signals to attract mates, warn of danger, guide others to food, and maintain a territory. Even though the fact that animals communicate in some fashion is not usually argued, the study of a particular animal or species and explanations of their communication system is very difficult. Not only must the costs and benefits of signal emitting and receiving be calculated, but the interaction with illegitimate receivers and the use of deceptive signals must be understood to fully explain any communication system (Alcock, 1993). The time scale involved in evolution leads to yet more problems in studying communication. One method to reconstruct the evolution of a behavior is to compare closely related living species that exhibit forms of the same behavior. This is how Martin Lindauer (1961) reconstructed the evolutionary history of the honeybee waggle dances. Even though this method provides strong circumstantial evidence for its conclusions, there is no way to verify the results. Issues such as these illustrate the complexity of explaining the behavior of existing animals.

Synthetic ethology is a relatively new field. In addition to MacLennan's work (1990, 1992) previous studies utilizing synthetic ethology include Gregory Werner and

Michael Dyer's work (1992). Werner and Dyer's experiments involved making communication essential to reproduction in a population of artificial organisms by creating an environment where females had to guide "blind" males through the use of signals to the female's location. Both of these previous experiments proved that it is possible to create organisms that are able to evolve to the point of using symbols effectively to solve some problem in their environment.

One main difference between this work and Werner and Dyer's work is the complexity of the environment and simulated organisms. Werner and Dyer create a world where organisms are solving the problem of finding mates. Therefore the male organisms must be able to move about in their environment and the females must be able to sense males within a certain range of their location. While these complexities make the organisms seem more life-like, they add nothing to the problem at hand, studying communication. In fact, Werner and Dyer point out that MacLennan's work "is simpler to analyze, and his use of abstract environmental states may make it easier to evolve complex protocols such as those requiring syntax" (1992, p. 683).

While synthetic ethology might be perceived as related to computer simulations, there are several important differences. Simulations and modeling attempt to reproduce the elements of a complex system that are perceived as affecting the behavior of that system. Simulations are designed and programmed to produce a specific behavior. A synthetic ethology study uses an entirely new environment and very simple organisms which are not "programmed" to behave in a certain manner. The organisms are given the capability to perform actions but the choice of how to behave is left up to their genes and evolution. In Werner and Dyer's study, communication was the main goal of the experiment; however, they pointed out "there should not be direct pressure on the animals to communicate" (1992, p. 660). Synthetic ethology is based upon Valentino Braitenberg's (1984) synthetic psychology. His "law of uphill analysis and downhill invention" (p. 20) means that designing a new system that produces a certain behavior is easier than analyzing an existing system that exhibits that same behavior. For example, the job of simulating all of the factors of a frog population that affect the frogs' communication is very difficult if not impossible. The job of creating a population of new organisms that communicate and then studying those new organisms is much easier, especially when the organisms exist as data structures in a computer program.

Synthetic ethology is more closely related to the field of artificial life. It uses several methods such as genetic programming and the creation of artificial environments that were developed by scientists investigating artificial life. Artificial life programs typically exhibit complex behavior such as self reproduction and learning. The main difference between artificial life and synthetic ethology is that artificial life researchers are interested in identifying the processes that separate living beings from non-living beings while synthetic ethologists focus on studying one particular behavior of a being.

Chapter 3

Methods

Communication

Since the main goal of this research is to develop simulated organisms that demonstrate the evolution of communication, we must first define what is to be considered communication. Deciding what actions can be called communication has been a controversial topic. For the purposes of this experiment Burghardt's (1970) definition of communication shall be used:

> Communication is the phenomenon of one organism producing a signal that, when responded to by another organism, confers some advantage (or the statistical probability of it) to the signaler or his group. (p. 16)

The physical act of communicating, the emission and reception of a signal, must take place and the response should give some advantage to the organism that emitted the signal to count the action as communication. In the new environment we are creating, communication is demonstrated when two simorgs cooperate.

A cooperation is defined as when a simorg takes an action that matches the local environment of the last simorg to emit a symbol. For example, if simorg ***A*** with local environment 3 emits 1,4 and simorg ***B*** responds with the action 3, a successful cooperation has taken place. After each cooperation the fitness level of the emitter as well as the fitness level of the responder is incremented. The likelihood of an organism reproducing is directly related to the number of successful cooperations the organism is involved in. Therefore a cooperation event satisfies the definition of communication given above. A signal is emitted and responded to, and that response confers an advantage, an increased chance at reproduction, to the emitter.

Obviously, the most successful cooperators will use communication. If an organism can communicate its local environment by using a unique set of symbols that the other organisms can interpret, it will be more likely that the responding organisms will take the correct action. Since the genes of an organism govern the “language” that the organism initially uses, reproduction is a powerful tool that an organism uses to increase the number of organisms using its own language.

Environment and Organisms

The system studied consists of a group of simulated organisms arranged in a circle. Each organism can sense three different types of information: (1)Every organism in the circle can sense the two global symbols. (2)A simorg can sense its own internal state. (3)Each simorg can also sense what is in its local environment. The global symbols can be seen by all of the simorgs; however, the other two types of information are private to each simorg. Please refer to Figure 1. Every simorg can sense the two global symbols but only simorg ***B*** can see the local environment ***b*** and only simorg ***B*** knows its own internal state which is β. Hopefully the simorgs will evolve to the point of using symbols placed in the global area to communicate. It is expected that the internal states will be used as memory so a simorg can remember what information it has seen before. The symbols placed in local environments are what the simorgs are to communicate to each other. None of these behaviors are explicitly programmed into the simorgs; however, evolution is expected to guide the simorgs into behaving this way.

Each simorg is given two turns to respond to the information the simorg can sense. The simorgs have three options during each of the two turns: emitting a new symbol to be placed in the global environment, taking an action denoted by an integer, or doing nothing. In the first turn, the first global symbol is used in its decisions and if a symbol is emitted

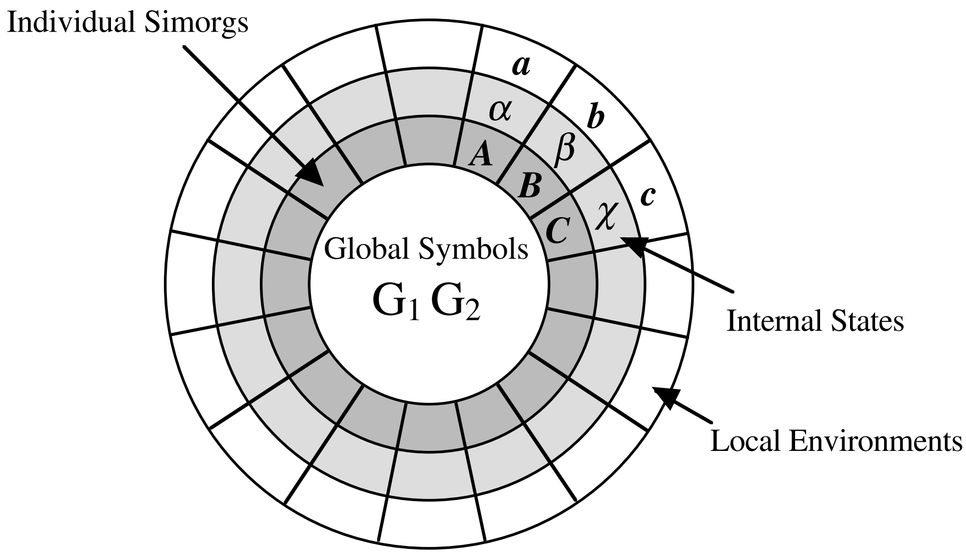


Figure 1 Environment of the Simulated Organisms

the symbol will replace the first global symbol. During the second turn, the second global symbol is used to look up which option to take and the second global symbol will be replaced if an emission occurs. After the simorg's two turns, control is passed to the next simorg.

Each simorg is a finite state machine. The phenotype of the simulated organism serves as a transition table for the organism. The phenotype has entries for every combination of global symbols, local symbols, and internal states. When it is an organism's turn to respond, the organism is governed by the entries in its own table. For example if Figure 2 is a simorg's phenotype and the simorg's internal state is 1, the global symbol being used for this turn is 2, and the simorg's local environment contains a 1, then the simorg will act with an action suitable for a local environment represented by the integer 5. The phenotype of each simorg is originally a copy of the simorg's genotype.

The genotype of a simorg is a table which has the same structure as the simorg's phenotype. The genotypes for the initial population are randomly assigned. For later generations, a new simorg's genotype is a combination of its parents' genotypes using two point crossover. In a two point crossover, two points within the parents' genotypes are randomly chosen. The genes for the new simorg between the two points are taken from one parent and the remaining genes are taken from the other parent. This is illustrated in Figure 3.

If learning is allowed, only the phenotype will be changed as knowledge is obtained. Therefore the simorg's offspring will not benefit from knowledge gained by the parent since only the genotype is passed on to the child. However, only one pair of simorgs is chosen to mate in each major cycle, so the child will be able to quickly learn from the other simorgs.

The use of the words genotype and phenotype within this study correspond to their definitions in biological terms. The genotype of an organism is the information inherited from its parents and passed on to its offspring. The phenotype is the physical interpretation of the genotype. If a person has the gene for black hair, their phenotype, in this case the specific portion of their phenotype called hair, will reflect that by being a dark color. For the simorgs the genotype is the information inherited from its parents. The phenotype is the

Internal State	Global Symbol	Local Environment	Act or Emit	Code
1	1	1	Nothing	3
1	1	2	Act	1
1	1	3	Emit	6
1	2	1	Act	5
•				
•				
•				
3	1	1	Emit	2
3	1	2	Nothing	8
3	1	3	Act	3
3	2	1	Emit	5
3	2	2	Nothing	4
3	2	3	Act	4
•				
•				
•				

Figure 2 Sample Partial Phenotype of a Simorg

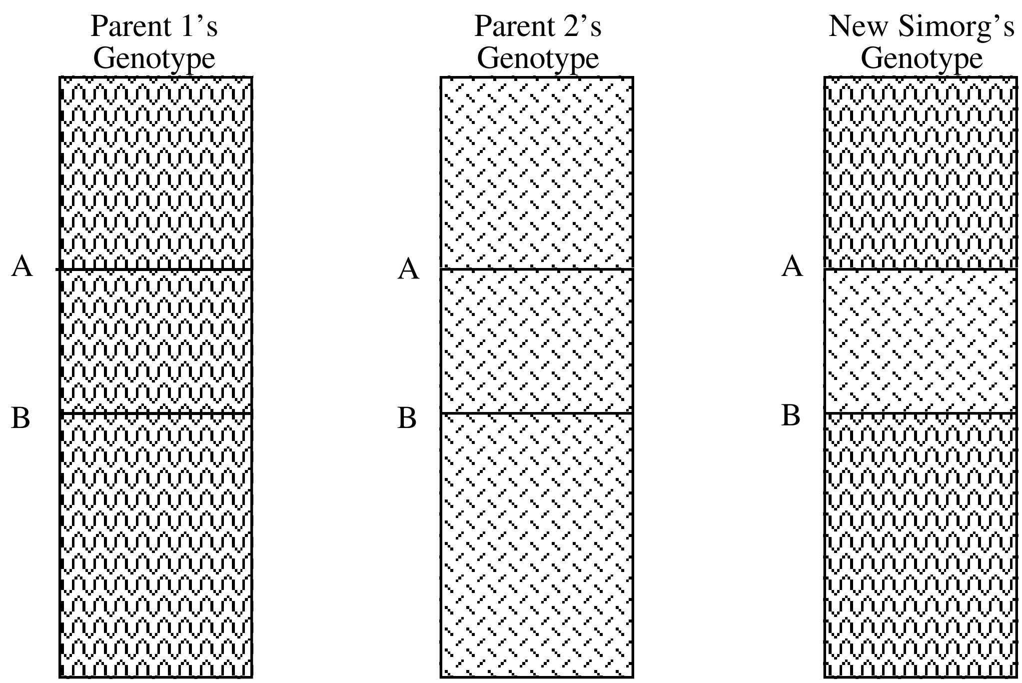


Figure 3 Illustration of Two Point Crossover

table that guides their actions and emissions. Since the simulated organisms used in these experiments are so simple it easy to blur the distinctions between the genotypes and phenotypes.

Cycles

Each experiment consists of at least 60,000 “major cycles”. At the beginning of each major cycle the fitness points for each simorg are zeroed. At the end of each major cycle two simorgs are chosen to reproduce and one simorg is chosen to be replaced. The simorgs that reproduce are chosen with a probability proportional to their fitness. Therefore, it is possible for the worst simorg to be chosen to reproduce, but this is very unlikely. It also means that the best simorg is not always chosen as one of the two to reproduce. The simorg picked to die is chosen according to a monotonically decreasing function of the simorgs’ fitness. The population stays constant since one new simorg is produced and takes the place of the simorg chosen to be replaced.

Each major cycle consists of five minor cycles. At the beginning of each minor cycle the local environments are randomized. During each minor cycle there are ten delay cycles. A delay cycle consists of every simorg being given a chance to respond. Each simorg can emit a new global symbol, it can perform an action, or it can do nothing. If the action performed matches the local environment of the last emitter, a cooperation is said to have taken place.

Measurements

Several different types of measurements are used to gauge the performance of a population of simulated organisms. The average fitness of the population smoothed over fifty major cycles is recorded. Since each point of fitness results from a successful cooperation, the average fitness gives how many times the average organism cooperated successfully in the last major cycle.

Most of the measurements deal with the denotation matrix of a population. The denotation matrix is a table with entries for each local environment and combination of global symbols. Each cooperation increments the entry in the denotation matrix corresponding to the local environment of the emitter and the global symbols the organism used to denote that local environment. At the end of the evolution the denotation matrix can be viewed as a dictionary of what combination of symbols are used to represent which local environment. The denotation matrices shown in this paper show only the cooperations during the last fifty major cycles. This allows one to see the language used at the end of the evolution without the clutter of earlier attempts to cooperate.

There are several ways of measuring the organization of a denotation matrix. The entropy of a two dimensional discrete probability distribution $p_{\gamma,\lambda}$ is defined as:

$$H = -\sum_{\gamma,\lambda} p_{\gamma,\lambda} \lg p_{\gamma,\lambda}$$

For all calculations $\lg x = \log_2 x$. The upper limit of entropy, representing a totally random use of symbols, can be found by substituting $1/G^2L$ for $p_{\gamma,\lambda}$ where G equals the number of values one of the two global symbols can be assigned and L equals the number of values a local environment can be assigned. Unless otherwise noted, G = 4 and L = 8 for all of the experiments presented in this thesis. Therefore by substituting 1/128 for $p_{\gamma,\lambda}$ the upper limit is $H = 7$. For an ideal matrix, where eight combinations of two symbols uniquely denote the eight possible local environments, the entropy equals 3. If the denotation matrix has only one non-zero entry then the entropy equals 0. This condition is referred to as the denotation matrix being over structured. See MacLennan (1990) for further analysis of these measures.

Finally, two measurements are taken from each denotation matrix which indicate the "richness" of the language developed by the simorgs. The first is how many different pairs of symbols are the most used symbols in a column of a denotation matrix. This measure was used because the entropy values only indicate how many pairs of symbols are used, not which pairs were used. A perfect entropy value of 3 could only indicate that the population used 1,1 to denote all eight of the situations. The second measure is, of the different pairs mentioned above, how may are non-repeating pairs. For example 1,1 is a repeating pair but 1,4 is not a repeating pair. This measure was used because the use of non-repeating pairs indicates the true use of two symbol signals to uniquely denote a situation.

Research Areas

MacLennan's initial studies (1990, 1992) resulted in denotation matrices such as Figure 4. The denotation matrix shows that there were not many successful cooperations. Also, the fact that successful uses of symbols are distributed in groups of four indicates that the simorgs are using the second symbol and ignoring the first. My study was initiated to see what changes had to be made to encourage the simorgs to evolve a more successful language consisting of signals made up of two symbols.

The most important change made in this study was to the environment and the rules being used to govern the action of simorgs. In the original work (MacLennan 1990, 1992), there was only one global symbol, not two as used in this study. Therefore each simorg was only given one chance to respond during its turn. Also, the simorgs did not have the option of doing nothing. During each simorg's turn, the simorg had to either emit or act.

After making the discussed modifications to the environment and rules, several more smaller changes were investigated. The first was varying the population size. A larger population can be used to increase genetic diversity at the beginning of the evolution. The mutation rate was low, there was only a 1 in a 100 chance of a single gene being mutated in a new simorg. While mutation will in theory explore the entire genetic space if given infinite time, these experiments were planned to be completed in a shorter amount of time. Therefore it was important to have a broad base of genes in the population at the beginning of the program's run.

Another parameter that was explored is the number of internal states. We expected that evolution would guide the simorgs to use the internal states to remember the previously seen symbol. Since there are four different values that a global symbol can take, one would expect that four internal states would be adequate. Both smaller and larger values were used to determine what number of internal states was optimal.

Symbols	Situation							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0/0	746	**5260**	**2871**	1282	0	0	923	**1544**
1/0	299	359	669	507	0	0	248	424
2/0	45	916	117	652	0	0	256	36
3/0	37	140	252	658	0	0	244	447
0/1	435	94	138	204	237	0	49	431
1/1	**1855**	39	85	1394	**3795**	0	122	541
2/1	279	102	6	46	190	0	34	77
3/1	227	14	34	25	164	0	52	631
0/2	37	577	0	644	0	426	144	0
1/2	56	307	0	22	0	454	20	0
2/2	100	1252	0	323	0	**9220**	20	0
3/2	9	28	0	202	0	1180	42	0
0/3	12	0	99	586	0	5	494	603
1/3	27	0	14	192	0	5	135	594
2/3	0	0	5	216	0	126	120	57
3/3	0	0	341	**1481**	0	130	**1679**	684

$H =$ 5.1 Different Pairs= 4 Non-Repeating Pairs= 0

Figure 4 Denotation Matrix From MacLennan's Original Research

Another area to be examined was the use of different learning rules. The learning rule used originally in MacLennan (1990, 1992) is a single case learning rule. If a simorg performed a wrong action, the phenotype of the simorg was changed so that the simorg would have produced the correct behavior under precisely the same circumstances. While this learning rule was adequate for communication using of one symbol, it did not lead to communication using two symbols. One possibility for the failure to learn more complex behavior is that the single case rule is too volatile. The new learning rule used in this study was a real-valued neuromorphic acceptor tree (referred to as a NAT) of depth 2 (Hand, 1994). A NAT is a simplified representation of a neural net. The version used in this study was tested to verify its reliability and was found to learn linearly non-separable functions very quickly.

In the original experiments the new simorg created after each major cycle was placed in the position occupied by the simorg that was chosen to die. This could lead to instability if the new simorg was not placed in the spatially defined dialect group of its parents, especially if the single case learning rule is used. Another change considered was placing a new simorg next to one of its parents.

The final change studied was to create a competitive environment for the simorgs. Preliminary investigations showed that the simorgs were not evolving “optimum” languages which had a unique pair of symbols for each situation. A population of “predators” was added to push the simorgs to evolve further. If a predator could output the same action as a successful simorg, neither the emitting simorg or the responding simorg was given any fitness points. The predator’s fitness level was increased for every cooperation that the predator “blocked.” The predator only had one internal state, corresponding to no memory, while the simorgs still had four internal states. It was hoped that this change would force the simorgs to use non-repeating symbols.

Chapter 4

Results and Discussion

New Environment

A graph of average fitness during a typical evolution is shown in Figure 5. The average fitness values begin around 30 and quickly rise to over 60. A graph of the entropy of the denotation matrix during the evolution is shown in Figure 6. The entropy values are initially close to 7, the highest possible entropy, and fall off to around 5 before leveling off. For both of these graphs the values are smoothed over fifty major cycles.

Figure 7 is a typical denotation matrix from the base version of the program used in these experiments. Figure 8 show the results of comparing the two different environments over ten different evolutions each. **F** is the value obtained from the analysis of variance for each of the performance measures. The value **p** is the probability of a larger **F** by chance. The values for **F** and **p** show that there are significant differences between the values for the two different environments on two of the performance measures.

The new environment promotes a much higher average fitness and a larger use of repeating pairs of symbols. The average simorg in the new environment learned to cooperate much more successfully than the simorgs in MacLennan's earlier work (1990, 1992) concerning two symbol signals. The use of more non-repeating pairs shows that the simorgs are using two symbols to denote a situation. This finding is supported by the fact that the values in the denotation matrices are not broken into groups of four. This indicates

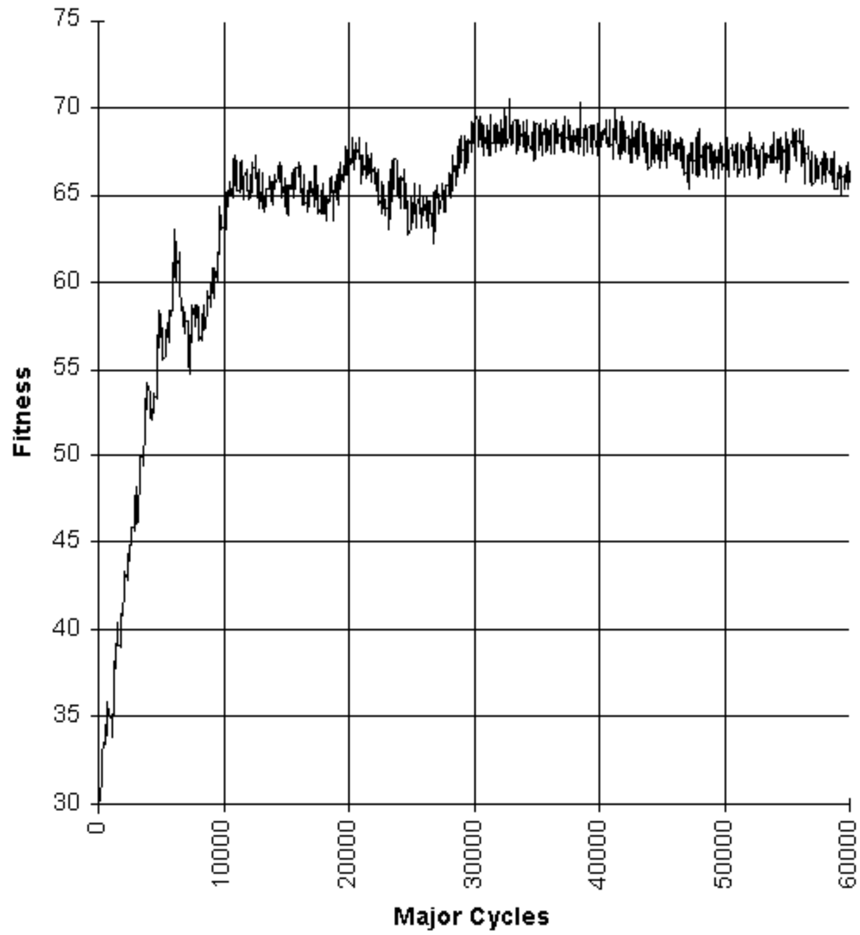


Figure 5 Average Fitness Smoothed Over 50 Major Cycles

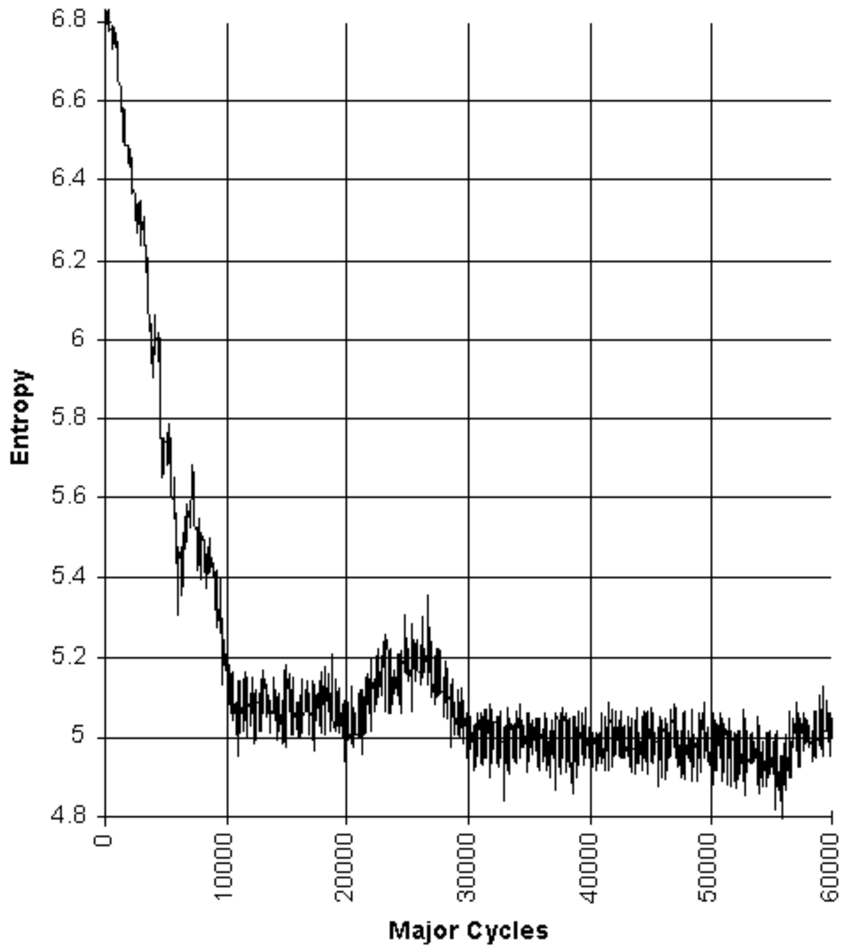


Figure 6 Entropy Smoothed Over 50 Major Cycles

Symbols	Situation							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1/1	**10886**	**8934**	0	0	450	**18148**	0	149
2/1	10	190	315	21	0	5	219	581
3/1	375	571	0	75	926	1730	0	0
4/1	1035	419	18	314	1310	525	0	0
1/2	0	119	0	17	76	0	0	420
2/2	0	33	**7458**	735	54	0	681	443
3/2	81	0	0	624	789	0	219	**1317**
4/2	135	706	2209	271	56	0	706	155
1/3	222	953	0	29	1083	2513	229	53
2/3	51	63	679	428	632	0	414	794
3/3	237	2164	0	327	**3839**	1425	**3850**	0
4/3	456	203	39	1239	1381	297	1397	0
1/4	1723	608	177	191	679	523	27	0
2/4	17	371	1423	101	354	11	351	343
3/4	293	189	559	**1643**	1419	328	2021	0
4/4	3574	508	410	782	3	809	1323	0

$H =$ 5.18 Different Pairs= 4 Non-Repeating Pairs= 1

Figure 7 Sample Denotation Matrix from Base Version Used in this Study

Performance Measures	Mean	SD	F	p
Lowest H			0.01	0.91
Original	4.94	0.40		
Base	4.97	0.65		
Highest Fitness			113.78	0.00
Original	23.20	3.43		
Base	63.50	11.44		
Different Pairs			0.50	0.49
Original	4.00	0.47		
Base	4.30	1.25		
Non-Repeating Pairs			10.37	0.00
Original	0.30	0.48		
Base	1.40	0.97		

Figure 8 Comparisons of MacLennan's Original Environment and the Environment of the Base Version Used in this Study

that the new simorgs are not ignoring the first symbol.

The values recorded for lowest *H* and the use of different pairs are not significantly different. Neither versions' matrices reached the entropy level of a perfect matrix.

Population Size

To compare different population sizes, each size population (50, 100, 200, 400, and 800) was used for ten program runs. The number of internal states was set to four. The results are summarized in Figure 9. The values for **F** and **p** show that there are significant differences between the values for different population sizes on each of the performance measures. Further analysis using comparisons for each pair of populations using a Student's t are discussed below. Figure 10 points out which pairs are significantly different to a 0.05 confidence level.

For lowest *H*, the population size 800 is significantly different from all of the other population sizes. The values obtained from the population sizes 50, 100, 200, and 400 are not significantly different from each other. This is attributed to the fact that 60,000 major cycles is enough time for most of the simorg populations of less than 800 to stabilize and settle into a set language. The populations of 800 were still not settled into a fixed language before the program ended.

The means recorded for highest fitness during a run increases with population size except for the values reported for the population size 800. Again this is attributed to the fact that the populations of 800 did not have long enough to settle on a language. In fact, there is a negative correlation between entropy values and fitness values. This is shown in Figure 11. This correlation was expected since a lower entropy indicates that the simorgs were successful in settling on what symbols represent which situations and this leads to more successful communications which lead to a higher average fitness.

For both the number of different pairs of symbols used and the number of non-repeating pairs, the population size 100 outperformed the other populations sizes. Both of the symbol usage performance measures for the population size 100 are significantly

Performance Measures		Mean	SD	F	p
	Lowest H			8.38	0.00
	50	4.66	0.70		
	100	4.97	0.65		
	200	4.65	0.88		
	400	4.65	0.91		
	800	6.18	0.30		
	Highest Fitness			10.04	0.00
	50	58.58	10.55		
	100	63.50	11.44		
	200	71.29	10.38		
	400	77.98	8.66		
	800	54.04	6.12		
	Different Pairs			3.83	0.01
	50	3.40	1.43		
	100	4.30	1.25		
	200	3.70	1.16		
	400	2.70	0.67		
	800	2.60	1.07		
	Non-Repeating Pairs			6.10	0.00
	50	0.80	0.79		
	100	1.40	0.97		
	200	1.00	0.67		
	400	0.50	0.53		
	800	0.00	0.00		

Figure 9 Comparisons of Different Population Sizes

Lowest Entropy

	50	100	200	400	800
50	*	No	No	No	Yes
100	No	*	No	No	Yes
200	No	No	*	No	Yes
400	No	No	No	*	Yes
800	Yes	Yes	Yes	Yes	*

Highest Fitness

	50	100	200	400	800
50	*	No	Yes	Yes	No
100	No	*	No	Yes	Yes
200	Yes	No	*	No	Yes
400	Yes	Yes	No	*	Yes
800	No	Yes	Yes	Yes	*

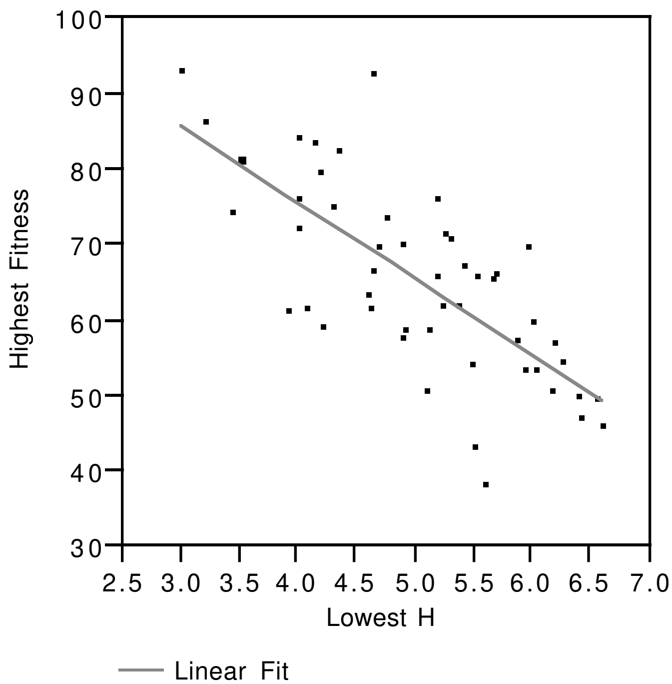
Different Pairs

	50	100	200	400	800
50	*	No	No	Yes	No
100	No	*	No	No	Yes
200	No	No	*	No	Yes
400	Yes	No	No	*	No
800	No	Yes	Yes	No	*

Non-Repeating Pairs

	50	100	200	400	800
50	*	No	No	No	Yes
100	No	*	No	Yes	Yes
200	No	No	*	No	Yes
400	No	Yes	No	*	No
800	Yes	Yes	Yes	No	*

Figure 10 Pair-wise Comparison of Different Population Sizes Using Student's t



Summary of Fit

RSquare	0.533678
RSquare Adj	0.523963
Root Mean Square Error	8.747296
Mean of Response	65.0776
Observations (or Sum Wgts)	50

Figure 11 Correlation between Lowest Entropy and Highest Fitness

different than the values recorded for population sizes 400 and 800. The larger population sizes do not show the diversity of signals that was previously assumed would result from having more genetic diversity in the starting population size. This was believed to result from the placement of new simorgs. If all of the simorgs use the same pair of symbols for every situation, it is easier for a new simorg to respond to the pair of signals no matter where the new simorg is placed.

Since the smaller population sizes performed better than larger population sizes, the size 100 was chosen to use in the rest of the experiments.

Internal States

The results of ten runs of the program for each of the values 2, 3, 4, 5, and 6 for the number of internal states were recorded. Figure 12 gives the results from varying the number of internal states. There is significant differences in all of the performance measures except for non-repeating pairs. Figure 13 shows which pairs of number of states are significantly different from each other to a 0.05 confidence level for each of the performance measures.

Once again there is a negative correlation between H and fitness as expected. The results from the runs with 2 and 3 internal states are significantly different from the larger numbers of internal states for both lowest H and highest fitness. These results are offset by the poor performance that the 2 and 3 internal state populations recorded for the number of different pairs used. This is a prime example of a small H being misleading. The populations with 2 and 3 internal states are on average using 3 different pairs of symbols to denote all 8 situations. The populations with 4 internal states are using 4 different pairs on average.

Up to this point the experiments have been run as if the population size and the number of internal states are independent variables. This is almost certainly not the case. To investigate this further, some evolutions with a population size of 400 with 2 internal states and a population size of 50 with 6 internal states were run. Neither of these combinations performed any better that a population size of 100 with 4 internal states. Therefore, the rest

Performance Measures	Mean	SD	F	p
Lowest H			15.52	0.00
2	3.10	0.88		
3	3.83	0.61		
4	4.97	0.65		
5	4.79	0.80		
6	5.25	0.61		
Highest Fitness			50.97	0.00
2	108.42	11.60		
3	83.34	10.10		
4	63.50	11.44		
5	56.66	13.59		
6	45.59	7.15		
Different Pairs			2.58	0.049
2	3.20	0.63		
3	3.20	0.79		
4	4.30	1.25		
5	3.60	0.97		
6	3.30	0.82		
Non-Repeating Pairs			1.02	0.41
2	1.10	0.99		
3	0.70	0.82		
4	1.40	0.97		
5	0.90	0.74		
6	0.80	0.79		

Figure 12 Comparisons of Different Numbers of Internal States

Lowest Entropy

	2	3	4	5	6
2	*	Yes	Yes	Yes	Yes
3	Yes	*	Yes	Yes	Yes
4	Yes	Yes	*	No	No
5	Yes	Yes	No	*	No
6	Yes	Yes	No	No	*

Highest Fitness

	2	3	4	5	6
2	*	Yes	Yes	Yes	Yes
3	Yes	*	Yes	Yes	Yes
4	Yes	Yes	*	No	Yes
5	Yes	Yes	No	*	Yes
6	Yes	Yes	Yes	Yes	*

Different Pairs

	2	3	4	5	6
2	*	No	Yes	No	No
3	No	*	Yes	No	No
4	Yes	Yes	*	No	Yes
5	No	No	No	*	No
6	No	No	Yes	No	*

Non-Repeating Pairs

	2	3	4	5	6
2	*	No	No	No	No
3	No	*	No	No	No
4	No	No	*	No	No
5	No	No	No	*	No
6	No	No	No	No	*

Figure 13 Pair-wise Comparisons of Different Numbers of Internal States using Student's t

of the experiments will use 100 as the population size and 4 as the number of internal states.

Different Learning Rule

Figure 14 shows the results of using the single case learning rule as opposed to using a neuromorphic acceptor tree. As the figure shows the NAT version did not differ significantly on the any of the measures except highest fitness. The difference between highest fitness values was large enough to say that the fitness levels achieved by the base version are significantly higher than the results recorded by the NAT version. It is interesting that the NAT version can perform as well on the other measures while not performing as well on the number of cooperations.

New Simorg Placement

The next change investigated was changing the placement of the new simorg after each major cycle. In the base version a new simorg is placed in the location of the simorg that was chosen to be replaced. In this new version the new simorg is placed next to one of its parents. Ten runs of the program for each version were run with the same seeds being used on each version. Figure 15 shows the results of a paired t test between each of the performance measures. A paired t test was used for these tests since the initial populations of each the runs were exactly the same. On the first 3 performance measures the new placement version did better than the base version. But on the number of non-repeating pairs used, the new version did worse than the base version. However, none of these differences were large enough to say that the results were significantly different to a 0.05 level of confidence.

Competitive Environment

The final change performed was to create a competitive environment for the simorgs. Figure 16 compares the results for the base program and the competitive environment version. The base version performed better on all of the performance

Performance Measures		Mean	SD	F	p
	Lowest H			3.78	0.07
	Base	4.97	0.65		
	New	4.28	0.91		
	Highest Fitness			6.99	0.02
	Base	63.50	11.44		
	New	47.73	15.01		
	Different Pairs			0.63	0.44
	Base	4.30	1.25		
	New	3.90	0.99		
	Non-Repeating Pairs			0.00	1.00
	Base	1.40	0.97		
	New	1.40	1.26		

Figure 14 Comparisons of Different Learning Rules

Performance Measures	Mean	SD	t ratio	p
Lowest H			1.60	0.14
Base	4.97	0.65		
New	4.59	0.55		
Highest Fitness			-0.93	0.38
Base	63.50	11.44		
New	68.46	10.51		
Different Pairs			2.08	0.07
Base	4.30	1.25		
New	3.40	0.97		
Non-Repeating Pairs			1.08	0.31
Base	1.40	0.97		
New	1.00	1.15		

Figure 15 Comparisons of Different Placement of New Simorgs

Performance Measures	Mean	SD	F	p
Lowest H			0.04	0.85
Base	4.97	0.65		
Competitive	5.01	0.34		
Highest Fitness			11.95	0.00
Base	63.50	11.44		
Competitive	47.96	8.42		
Different Pairs			3.09	0.10
Base	4.30	1.25		
Competitive	3.50	0.70		
Non-Repeating Pairs			5.36	0.03
Base	1.40	0.97		
Competitive	0.40	0.97		

Figure 16 Comparisons of Base Environment and Competitive Environment

measures. This was disappointing since the presence of "predators" was hoped to push the simorgs to evolve further. In fact it was hard to create an environment were the predators did not dominate the evolution. In initial trials the predators were given the ability to learn and the simorgs did not evolve to use symbols effectively. Too strong of a predator did not give the simorgs a chance to evolve and too weak of a predator did not push the simorgs enough.

Chapter 5

Conclusion

The new environment enabled the simorgs to use two symbol signals more effectively than the experiments reported by MacLennan (1990,1992). However, any further changes, such as new learning rules and different placement of new simorgs, did not show significant improvement over the base version used in this study.

Measurement Limitations

I believe that the limit has been reached on what can be learned by examining denotation matrices for the entire population. Tools which investigate the reactions of each simorg to sets of symbols might lead to discoveries about dialect groups within the population as a whole and about what effect symbol order has on an individual simorg's reactions.

Further Study

One obvious area for future study is increasing the number of symbols used in a signal to 3. The present system of using a finite state machine as a simorg may have reached its limit for producing complex behavior. Changing the basic mechanism of a simorg to a rule-based system, neural net, or NAT might encourage the development of more complex behavior.

Other new work might involve changing the topology of the simorg environment. This change could allow the environment to represent more biologically correct versions of mating and signal propagation. As more complex behavior is expected of the simorgs, the more complex their environment and the simorgs themselves will have to be.

Bibliography

Bibliography

Alcock, J. (1993). *Animal behavior: An evolutionary approach* (5th ed.). Sunderland, MA: Sinauer Associates.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. Cambridge, MA: MIT Press.

Burghardt, G. M. (1970). Defining “communication”. In J. W. Johnston Jr., D. G. Moulton, & A. Turk (Eds.), *Communication by chemical signals*. New York: Appleton-Century-Crofts.

Hand, C. (1994). *Object oriented logic* (Tech. Rep. No. CS-94-235). Knoxville, TN: Computer Science Department, University of Tennessee.

Lindauer, M. (1961). *Communication among social bees*. Cambridge, MA: Havard University Press.

MacLennan, B. J. (1990). *Evolution of communication in a population of simple machines* (Tech. Rep. No. CS-90-99). Knoxville, TN: Computer Science Department, University of Tennessee.

MacLennan, B. J. (1992). Synthetic ethology: An approach to the study of communication. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II*. Redwood City: Addison-Wesley.

MacLennan, B. J. & Burghardt, G. M. (1993). Synthetic ethology and the evolution of cooperative communication. *Adaptive behavior*, *2*, 161-188.

Morton, E. S., & Page, J. (1992). *Animal talk: Science and the voices of nature*. New York: Random House.

Sebeok, T. A. (Ed.). (1968). *Animal communication: Techniques of study and results of research.* Bloomington: Indiana University Press.

Werner, G. M. & Dyer, M. G. (1992). Evolution of communication in artificial organisms. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial life II*. Redwood City: Addison-Wesley.

Appendices

Appendix A

Reproducing Single Symbol Communication

Since this study was based on MacLennan's earlier work, the first step taken was to reproduce MacLennan's results concerning one symbol signal evolution. The new version of the program was written based on descriptions of the algorithms from previously published articles. The results produced were similar to the results of MacLennan's program when communication was enabled. Denotation matrices produced by MacLennan's program, such as Figure A-1, when communication was suppressed were substantially different from the denotation matrices produced by the new program, a sample of which is shown in Figure A-2. After examining MacLennan's program a mistake was found. When communication was suppressed, the global symbol was randomized between the times when the symbol was used to decide what action to take and the time when the denotation matrix was incremented due to a successful cooperation. Therefore, the symbol used to make the decision about what action to take was not the symbol recorded in the denotation matrix after a successful action.

Since the randomly distributed values in the denotation matrix seemed to coincide with the communication suppressed case, the question of where the order in the denotation matrices produced by the corrected program came from was raised. Genetic drift was assumed to cause the order appearing in the denotation matrices. Genetic drift can occur when the distribution of actions and emissions in new simorgs' genotypes do not match the

					Situations			
Symbols	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	180	201	27	712	149	296	254	292
1	202	191	21	707	140	268	240	338
2	196	199	24	699	145	284	235	290
3	168	154	20	713	135	312	214	314
4	200	182	15	643	149	310	226	284
5	206	183	28	684	142	283	243	280
6	204	191	21	676	145	290	221	310
7	198	186	19	689	128	276	236	297

$H =$ 5.66

Figure A-1 Communication Suppressed Denotation Matrix From MacLennan's Work

					Situations			
Symbols	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	94	130	133	34	166	0	150	682
2	16	105	279	228	261	307	0	118
3	0	199	229	12	0	0	161	274
4	95	19	93	283	669	89	0	201
5	1	97	212	200	112	0	0	0
6	28	135	84	8	600	215	0	351
7	0	0	0	118	59	70	0	690
8	0	33	41	0	371	0	0	0

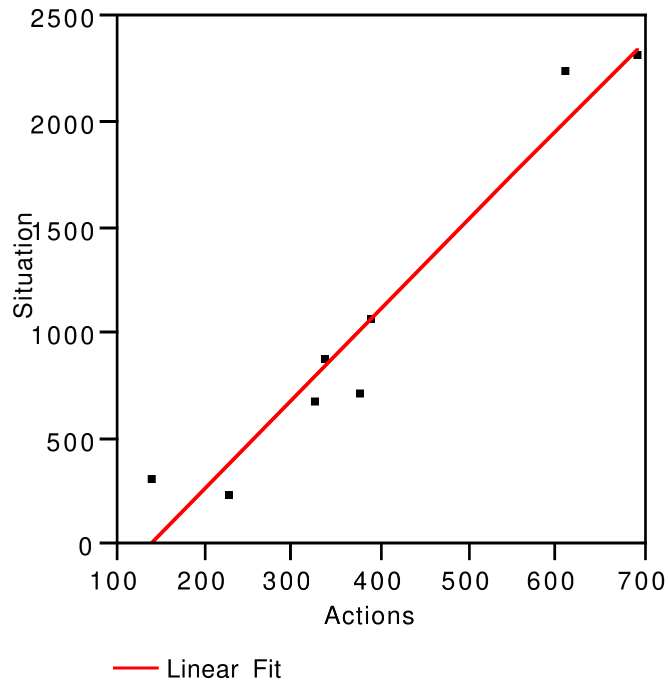
$H =$ 4.95

Figure A-2 Communication Suppressed Denotation Matrix From New Program distribution of actions and emissions in the simorgs being replaced. Obviously, this can occur when using two-point crossover with the two points being picked randomly.

To test the hypothesis that the order came from the genetic drift, the following experiment was performed. At the beginning of each evolution how many times each action appeared in the genotypes of the simorg population was recorded. At the end of the evolution the same data was recorded again. The values recorded are shown in Figure A-3. By examining the values at the beginning of the evolution it is obvious that the distribution is random. The values recorded at the end of the evolution are not evenly distributed. To test that this drift corresponds to the order of the denotation matrix shown in Figure A-2, the correlation between the distribution of situations in the final denotation matrix to the distribution of actions in the genotypes of the simorgs in the population was calculated. The results are illustrated in Figure A-4. The strong correlation should not be a surprise since a successful cooperation was defined as when a action is taken that matches the local environment (referred to as the situation) of the last emitter. This indicates that genetic drift is the reason for the order in the denotation matrices when communication is suppressed.

		Total in Genotype	
		Before	**After**
Actions	**1**	398	228
	2	400	377
	3	402	390
	4	395	337
	5	371	610
	6	406	327
	7	395	141
	8	412	691
	Mean=	397.38	387.62
	SD=	12.09	182.89

Figure A-3 Comparison of Distribution of Actions in the Genotypes of the Population



Summary of Fit	
RSquare	0.939821
RSquare Adj	0.929791
Root Mean Square Error	212.4597
Mean of Response	1056.5
Observations (or Sum Wgts)	8

Figure A-4 Correlation Between Situations in Successful Cooperations and Actions in the Genotypes of the Population

Appendix B

Listing of Program

On the following pages is a listing of the base version of the program used in these experiments.

```
c       Synthetic Ethology Code
c       This version uses two symbols in the Global Environment.
c       $Id$
c
c       NUMERIC PARAMETERS CONTROLLING THE SIMULATION
        integer Size, Lenv, Genv, IntStates
        parameter (Size=100, Lenv=8, Genv=4, IntStates=4)
        integer MajorCycle, MinorCycle, Delay
        parameter (MinorCycle=5, Delay=10)
        real Rate
        parameter (Rate=0.01)
        integer Seed
        integer WindowSize, Step
        parameter (WindowSize=50, Step=25)

c       LOGICAL PARAMETERS CONTROLLING THE SIMULATION
        logical EnableLearning, SuppressComm

c       VARIABLES USED IN THE SIMULATION
        integer Simorg (1:Size)
        integer Geno (1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)
        integer Pheno (1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)
        integer Fitness (1:Size)
        integer LocEnv (1:Size)
        integer GlobalEnv (1:2)
        integer NextGE (1:2)
        logical  Emission
        integer  Symbol
        integer LastEmitter, Goal, Which
        integer NextState, ActOrEmit, Code
        integer Mother, Father, Deceased, BestFit, StartingPoint
        real TotalFitness
        real AverFitness (1:WindowSize)
        real AverBestFit (1:WindowSize)
        integer AverageCount
        integer Denotation (1:Genv, 1:Genv, 1:Lenv)
        integer Loop1, Loop2, Loop3, Loop4, i
        character yn

c       FUNCTIONS CALLED
        integer nrand, FindParent, FindDead, Increment
        real Average
c------------------------------------------------------------------------
c       INPUT THE VARIABLES
        print *, 'Enter the number of Major Cycles:'
        read *, MajorCycle
        print *,  'Enter the random seed:'
        read *, Seed
        print *, 'Suppress Communication? (y/n)'
```

```
      read *, yn
      if ((yn .EQ. 'y') .OR. (yn .EQ. 'Y')) then
        SuppressComm = .TRUE.
      else
        SuppressComm = .FALSE.
      endif
      print *, 'Permit Learning? (y/n)'
      read *, yn
      if ((yn .EQ. 'y') .OR. (yn .EQ. 'Y')) then
        EnableLearning = .TRUE.
      else
        EnableLearning = .FALSE.
      endif

c     OPEN OUTPUT FILES
      call OpenFiles(Seed,EnableLearning,SuppressComm)
c     SEED THE RANDOM NUMBER GENERATOR
      call InitRand (Seed)
c     RANDOMIZE THE INITIAL GENOTYPES AND PHENOTYPES
      call InitGenes (Size, IntStates, Genv, Lenv, Geno, Pheno)
c     RANDOMIZE THE INITIAL INTERNAL STATES
      call RandStates (Size, Simorg, IntStates)
c     INITIALIZE LAST EMITTER AND GOAL
      LastEmitter = 0
      Goal = 0
      Emission = .FALSE.
c     RANDOMIZE THE GLOBAL ENVIRONMENT
      symbol = nrand(1,Genv)
      GlobalEnv(1) = symbol
      NextGE(1) = symbol
      symbol = nrand(1,Genv)
      GlobalEnv(2) = symbol
      NextGE(2) = symbol
c     INITIALIZE THE DENOTATION MATRIX
      call InitDeno(Genv,Lenv,Denotation)

      AverageCount=0

      do Loop1=1,MajorCycle

c        INITIALIZE FITNESS COUNTERS
         do i=1,Size
            Fitness(i) = 0
         enddo

         do Loop2=1,MinorCycle

c           RANDOMIZE THE LOCAL ENVIRONMENTS
```

```
            call RandEnv(Size, LocEnv, Lenv)

            do Loop3=1,Delay
               do Which=1,Size
                  do Loop4=1,2
                     NextState=Pheno(Which,Simorg(Which),
     &                              GlobalEnv(Loop4),LocEnv(Which),1)
                     ActOrEmit=Pheno(Which,Simorg(Which),
     &                         GlobalEnv(Loop4),LocEnv(Which),2)
                     Code=Pheno(Which,Simorg(Which),GlobalEnv(Loop4),
     &                         LocEnv(Which),3)

                     if (ActOrEmit .EQ. 1) then
                        if (Code .EQ. Goal) then
                           Fitness(LastEmitter) = Fitness(LastEmitter) + 1
                           Fitness(Which) = Fitness(Which) + 1
                           Denotation(GlobalEnv(1),GlobalEnv(2),Code)=
     &                        Denotation(GlobalEnv(1),GlobalEnv(2),Code) + 1
                        else
                           if (EnableLearning) then
                              call Learn(Size, Lenv, Genv, IntStates, Simorg,
     &                                 GlobalEnv, LocEnv, Which, Pheno,
     &                                 Goal, Loop4)
                           endif
                        endif
                     endif
                     if (ActOrEmit .EQ. 2) then
                        Emission = .TRUE.
                        Code = mod(Code,Genv)
                        if (Code .EQ. 0) then
                           Code = Genv
                        endif
                        NextGE(Loop4) = Code
                     endif
                     if (SuppressComm) then
                        symbol = nrand(1,Genv)
                        GlobalEnv(1) = symbol
                        NextGE(1) = symbol
                        symbol = nrand(1,Genv)
                        GlobalEnv(2) = symbol
                        NextGE(2) = symbol
                     endif
                     Simorg(Which) = NextState
                  enddo
                  if (Emission) then
                     GlobalEnv(1) = NextGE(1)
                     GlobalEnv(2) = NextGE(2)
                     LastEmitter = Which
```

```
                    Goal = LocEnv(Which)
                    Emission = .FALSE.
                 endif
              enddo
           enddo
        enddo

        Goal = 0
        call FindBest(Size, Fitness, TotalFitness, BestFit)
        StartingPoint = nrand(1,Size)
        Father = FindParent (Size, Fitness, TotalFitness,
     &                      StartingPoint)
        Mother = FindParent (Size, Fitness, TotalFitness, Father)
        Deceased = FindDead (Size, Fitness, TotalFitness, BestFit,
     &                      Mother)

        AverageCount = Increment(AverageCount,WindowSize)
        AverFitness(AverageCount) = TotalFitness / real(Size)
        AverBestFit(AverageCount) = Fitness(BestFit)

        if (Loop1 .EQ. MajorCycle) then
           call WriteFinalDeno(Genv,Lenv,Denotation)
        endif

        if ((Loop1 .GE. WindowSize) .AND. (mod(Loop1,Step) .EQ. 0)) then
           write (20,*) Loop1, Average(WindowSize,AverFitness)
           write (21,*) Loop1, Average(WindowSize,AverBestFit)
           call WriteEntropy(Genv,Lenv,Denotation,Loop1)
           call InitDeno(Genv,Lenv,Denotation)
        endif

        call Crossover (Size, Lenv, Genv, IntStates, Geno, Mother,
     &                 Father, Deceased)
        call Mutation (Size, Lenv, Genv, IntStates, Geno, Deceased,
     &                 Rate)
        call CopyOver (Size, Lenv, Genv, IntStates, Geno,
     &                 Pheno, Deceased)

      enddo

c     Close the files used for saving data
      close(20)
      close(21)
      close(22)
      close(25)
      close(26)
      stop
      end
```

```
c-------------------------------------------------------------------------
      subroutine OpenFiles(Seed,EnableLearning,SuppressComm)
      integer Seed
      logical EnableLearning, SuppressComm

c     LOCAL VARIABLES
      character*15 FileName

c     CALCULATE THE FILE NAMES
      call convert(Seed, FileName)

      if (EnableLearning) then
         FileName= '.L.' // FileName
      else
         FileName= '.NL.'// FileName
      endif

      if (SuppressComm) then
         FileName= '.NC' // FileName
      else
         FileName= '.C' // FileName
      endif

c     OPEN THE FILES USED FOR SAVING DATA
      open (unit=20, file='FitAver'//FileName)
      write (20,10) Seed, EnableLearning , SuppressComm
      open (unit=21, file='BestAver'//FileName)
      write (21,10) Seed, EnableLearning , SuppressComm
      open (unit=22, file='DenoMatrix'//FileName)
      write (22,10) Seed, EnableLearning , SuppressComm
      open (unit=25, file='V'//FileName)
      write (25,10) Seed, EnableLearning , SuppressComm
      open (unit=26, file='H'//FileName)
      write (26,10) Seed, EnableLearning , SuppressComm

10    format (1x,'# Seed= ',I10,'  EnableLearning= ',L,
   &              '  SuppressComm=',L)

      return
      end
c-------------------------------------------------------------------------
      subroutine InitRand (Seed)
      integer Seed

c     LOCAL VARIABLES
      integer i
      real Ignore
```

```
c       FUNCTIONS CALLED
        real rand

        do i=1,Seed
           Ignore = rand(0)
        enddo
        return
        end
c------------------------------------------------------------------------
        subroutine InitGenes (Size, IntStates, Genv, Lenv, Geno, Pheno)
        integer Size, IntStates, Genv, Lenv
        integer Geno (1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)
        integer Pheno (1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)

c       LOCAL VARIABLES
        integer I, J, K, L
        integer NextState, ActOrEmit, Code

c       FUNCTIONS CALLED
        integer nrand

        do I=1,Size
           do J=1,IntStates
              do K=1,Genv
                 do L=1,Lenv
                    NextState = nrand(1,IntStates)
                    ActOrEmit = nrand(1,3)
                    Code = nrand(1,Lenv)
                    Geno(I,J,K,L,1) = NextState
                    Geno(I,J,K,L,2) = ActOrEmit
                    Geno(I,J,K,L,3) = Code
                    Pheno(I,J,K,L,1) = NextState
                    Pheno(I,J,K,L,2) = ActOrEmit
                    Pheno(I,J,K,L,3) = Code
                 enddo
              enddo
           enddo
        enddo
        return
        end
c------------------------------------------------------------------------
        subroutine InitDeno (Genv,Lenv,Matrix)
        integer Genv, Lenv
        integer Matrix(1:Genv,1:Genv,1:Lenv)

c       LOCAL VARIABLES
        integer i,j,k
```

```
      do i=1,Genv
         do j=1,Genv
            do k=1,Lenv
               Matrix(i,j,k)=0
            enddo
         enddo
      enddo
      return
      end
c-----------------------------------------------------------------------
      subroutine RandEnv (Size, LocEnv, Lenv)
      integer Size, LocEnv(1:Size), Lenv

c     LOCAL  VARIABLES
      integer i

c     FUNCTIONS  CALLED
      integer nrand

      do i=1,Size
         LocEnv(i) = nrand(1,Lenv)
      enddo
      return
      end
c-----------------------------------------------------------------------
      subroutine RandStates (Size, Simorg, IntStates)
      integer Size, Simorg(1:Size), IntStates

c     LOCAL  VARIABLES
      integer i

c     FUNCTIONS  CALLED
      integer nrand

      do i=1,Size
         Simorg(i) = nrand(1,IntStates)
      enddo
      return
      end
c-----------------------------------------------------------------------
      subroutine Learn (Size, Lenv, Genv, IntStates, Simorg, GlobalEnv,
     &                        LocEnv, Which, Pheno, Goal, Loop4)
      integer Size, Lenv, Genv, IntStates, Which, Goal, Loop4
      integer Simorg(1:Size), GlobalEnv(1:2), LocEnv(1:Size)
      integer Pheno(1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)

c     IF THERE HAS BEEN AN EMISSION APPLY THE LEARNING RULE
```

```
          if (Goal .NE. 0) then
             Pheno(Which, Simorg(Which), GlobalEnv(Loop4),
     &                   LocEnv(Which), 3) = Goal
          endif

          return
          end
c-------------------------------------------------------------------------
          subroutine FindBest(Size, Fitness, TotalFitness, BestFit)
          integer Size, BestFit, Fitness(1:Size)
          real TotalFitness

c         LOCAL  VARIABLES
          integer i

          TotalFitness = Fitness(1)
          BestFit = 1

          do i=2,Size
             TotalFitness = TotalFitness + Fitness(i)
             if ( Fitness(i) .GT. Fitness(BestFit) ) then
                BestFit = i
             endif
          enddo
          return
          end
c-------------------------------------------------------------------------
          integer function FindParent(Size, Fitness, TotalFitness, Count)
          integer Size, Fitness(1:Size)
          real TotalFitness
          integer Count

c         LOCAL  VARIABLES
          integer Number
          real Prob, SumProb

c         FUNCTIONS  CALLED
          integer Increment
          real rand

          SumProb = 0.0
          Number = Count
          Number = Increment(Number, Size)
10        if (TotalFitness .EQ. 0.0) then
             Prob = 1.0 / real(Size)
          else
             Prob = real(Fitness(Number)) / TotalFitness
          endif
```

```
          if (rand(0) .GE. (Prob / (1.0 - SumProb))) then
             Number = Increment(Number,Size)
             SumProb = SumProb + Prob
             goto 10
          endif
          FindParent = Number
          return
          end
c-----------------------------------------------------------------------
          integer function FindDead (Size, Fitness, TotalFitness,
     &                                         BestFit, Count)
          integer Size, Fitness(1:Size)
          real TotalFitness
          integer BestFit, Count

c         LOCAL VARIABLES
          integer PBF, Number
          real Prob, SumProb

c         FUNCTIONS CALLED
          integer Increment
          real rand

          SumProb = 0.0
          Number = Count
          Number = Increment(Number, Size)
          PBF = Size * Fitness(BestFit)
10        if ( TotalFitness .EQ. PBF ) then
             Prob = 1.0 / real(Size)
          else
             Prob = real( Fitness(BestFit) - Fitness(Number) ) /
     &                       real( PBF - TotalFitness )
          endif
          if ( rand(0) .GE. (Prob / (1.0 - SumProb)) ) then
             Number = Increment(Number,Size)
             SumProb = SumProb + Prob
             goto 10
          endif
          FindDead = Number
          return
          end
c-----------------------------------------------------------------------
          integer function Increment(Count, UpperLimit)
          integer Count, UpperLimit

c         LOCAL VARIABLES
          integer Number
```

```
      Number = Count + 1
      if (Number .GT. UpperLimit) then
         Number = 1
      endif
      Increment = Number
      return
      end
c------------------------------------------------------------------------
      subroutine Crossover(Size, Lenv, Genv, IntStates, Geno, Mother,
   &                          Father, Deceased)
      integer Size, Lenv, Genv, IntStates, Mother, Father, Deceased
      integer Geno(1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)

c     LOCAL  VARIABLES
      integer I, J, K
      integer A, B, Temp, HowMany, Where

c     FUNCTIONS  CALLED
      integer nrand

      HowMany = IntStates * Genv * Lenv

      A = nrand(1,HowMany)
      B = nrand(1,HowMany)

      if (A .GT. B) then
         Temp = B
         B = A
         A = Temp
      endif

      Where = 0
      do I=1,IntStates
         do J=1,Genv
            do K=1,Lenv
               Where = Where + 1
               if (Where .LT. A) then
                  Geno(Deceased,I,J,K,1) = Geno(Father,I,J,K,1)
                  Geno(Deceased,I,J,K,2) = Geno(Father,I,J,K,2)
                  Geno(Deceased,I,J,K,3) = Geno(Father,I,J,K,3)
               else
                  if (Where .LT. B) then
                     Geno(Deceased,I,J,K,1) = Geno(Mother,I,J,K,1)
                     Geno(Deceased,I,J,K,2) = Geno(Mother,I,J,K,2)
                     Geno(Deceased,I,J,K,3) = Geno(Mother,I,J,K,3)
                  else
                     Geno(Deceased,I,J,K,1) = Geno(Father,I,J,K,1)
                     Geno(Deceased,I,J,K,2) = Geno(Father,I,J,K,2)
```

```
                    Geno(Deceased,I,J,K,3) = Geno(Father,I,J,K,3)
                 endif
              endif
           enddo
        enddo
      enddo
      return
      end
c------------------------------------------------------------------------
      subroutine Mutation(Size, Lenv, Genv, IntStates, Geno,
     &                    Deceased, Rate)
      integer Size, Lenv, Genv, IntStates, Deceased
      integer Geno(1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)
      real Rate

c     LOCAL  VARIABLES
      integer A, B, C, NextState, ActOrEmit, Code

c     FUNCTIONS  CALLED
      real rand
      integer nrand

      if (rand(0) .LT. Rate) then
         A = nrand(1,IntStates)
         B = nrand(1,Genv)
         C = nrand(1,Lenv)
         NextState = nrand(1,IntStates)
         ActOrEmit = nrand(1,3)
         Code = nrand(1,Lenv)

         Geno(Deceased,A,B,C,1) = NextState
         Geno(Deceased,A,B,C,2) = ActOrEmit
         Geno(Deceased,A,B,C,3) = Code
      endif
      return
      end
c------------------------------------------------------------------------
      subroutine CopyOver (Size, Lenv, Genv, IntStates, Geno,
     &                     Pheno, Deceased)
      integer Size, Lenv, Genv, IntStates, Deceased
      integer Geno(1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)
      integer Pheno(1:Size, 1:IntStates, 1:Genv, 1:Lenv, 1:3)

c     LOCAL  VARIABLES
      integer I, J, K

      do I=1,IntStates
         do J=1,Genv
```

```
          do K=1,Lenv
            Pheno(Deceased,I,J,K,1) = Geno(Deceased,I,J,K,1)
            Pheno(Deceased,I,J,K,2) = Geno(Deceased,I,J,K,2)
            Pheno(Deceased,I,J,K,3) = Geno(Deceased,I,J,K,3)
          enddo
        enddo
      enddo
      return
      end
c-----------------------------------------------------------------------
      real function Average(HowMany,Numbers)
      integer HowMany
      real Numbers(1:HowMany)

c     LOCAL  VARIABLES
      integer i
      real Sum

      Sum = 0.0
      do i=1,HowMany
        Sum = Sum + Numbers(i)
      enddo

      Average = Sum / real(HowMany)
      return
      end
c-----------------------------------------------------------------------
      subroutine WriteEntropy(Genv,Lenv,Matrix,Cycle)
      integer Genv, Lenv, Cycle
      integer Matrix(1:Genv,1:Genv,1:Lenv)

c     LOCAL  VARIABLES
      integer i,j,k,Sum
      real DSum, Mean, sigma, V, H, sm, n, C, Log2

      Log2 = Log10(2.0)

c     CALCULATE  STATISTICS
      Sum = 0
      do i=1,Genv
        do j=1,Genv
          do k=1,Lenv
            Sum = Sum + Matrix(i,j,k)
          enddo
        enddo
      enddo
      Mean = float(Sum) / float(Genv * Genv * Lenv)
```

```
          DSum = 0.0
          do i=1,Genv
              do j=1,Genv
                  do k=1,Lenv
                      DSum = DSum + (Matrix(i,j,k)-Mean)**2
                  enddo
              enddo
          enddo
          sigma = sqrt(Dsum / float(Genv * Genv * Lenv))
          if (Mean .EQ. 0.0) then
              V = -999999
          else
              V = sigma / Mean
          endif

          sm = 0.0
          C = 0.0
          do i=1,Genv
              do j=1,Genv
                  do k=1,Lenv
                      n = Matrix(i,j,k)
                      C = C + n
                      if (n .NE. 0) then
                          sm = sm + (n * (Log10(n) / Log2))
                      endif
                  enddo
              enddo
          enddo
          H = (Log10(float(C)) / Log2) - sm/C

c         WRITE RESULTS TO THE FILES
          write(25,*) Cycle,V
          write(26,*) Cycle,H
          return
          end
c------------------------------------------------------------------------
          subroutine WriteFinalDeno(Genv,Lenv,Matrix)
          integer Genv, Lenv
          integer Matrix(1:Genv,11:Genv,1:Lenv)

c         LOCAL VARIABLES
          integer i,j,k,Sum
          real DSum, Mean, sigma, V, H, Hmax, sm, n, C, eta, Log2
          character*15 HowMany

          Log2 = Log10(2.0)

c         CALCULATE STATISTICS
```

```
        Sum = 0
        do i=1,Genv
            do j=1,Genv
                do k=1,Lenv
                    Sum = Sum + Matrix(i,j,k)
                enddo
            enddo
        enddo
        Mean = float(Sum) / float(Genv * Genv * Lenv)

        DSum = 0.0
        do i=1,Genv
            do j=1,Genv
                do k=1,Lenv
                    DSum = DSum + (Matrix(i,j,k)-Mean)**2
                enddo
            enddo
        enddo
        sigma = sqrt(Dsum / float(Genv * Genv * Lenv))
        if (Mean .EQ. 0.0) then
            V = -999999
        else
            V = sigma / Mean
        endif

        sm = 0.0
        C = 0.0
        do i=1,Genv
            do j=1,Genv
                do k=1,Lenv
                    n = Matrix(i,j,k)
                    C = C + n
                    if (n .NE. 0) then
                        sm = sm + (n * (Log10(n) / Log2))
                    endif
                enddo
            enddo
        enddo
        H = (Log10(float(C)) / Log2) - sm/C
        Hmax = 2*(Log10(float(Genv))/Log2) + (Log10(float(Lenv))/Log2)
        eta = H / (Log10(float(Lenv))/Log2) - 1.0

c       WRITE THE DENOTATION MATRIX TO A FILE
        call convert(Lenv,HowMany)
        write(22,*)
c       write(22,'('//HowMany//'(i8))'),(k,k=1,Lenv)
        do i=1,Genv
            do j=1,Genv
```

```
c               write (22,*),i,j,' :'
                write (22,'('//HowMany//'(i8))'),(matrix(i,j,k),k=1,Lenv)
            enddo
        enddo

        write(22,*)
        write(22,10) 'V',V
        write(22,10) 'H',H
        write(22,10) 'Hmax', Hmax
        write(22,10) 'eta',eta

10      format (1X, A, ' = ',F)
        return
        end
c------------------------------------------------------------------------
        integer function nrand(i, j)
        integer i, j

c       FUNCTIONS CALLED
        real rand

c       RETURNS A NUMBER BETWEEN i AND j, INCLUSIVE
        nrand = i + int(real(j+1-i)*rand(0))
        if (nrand .EQ. (j+1)) then
            nrand = j
        endif

        return
        end
c------------------------------------------------------------------------
        subroutine convert(in,letters)
        integer in
        character*15 letters

c       LOCAL VARIABLES
        integer number, digit

        number=in
        letters=''
10      digit=mod(number,10)
        number=number/10
        letters=char(48+digit)//letters
        if (number .NE. 0) then
            goto 10
        endif
        return
        end
c--------------------------------------------------------------------
```