

Acknowledgement

I am grateful to a number of people of the department of Mathematical and Information Sciences at Coventry University; this project would not have been possible without the aid and inspiration of Colin Reeves, Olivier Haas, Keith Burnham and Joe Mahtani.

Summary

This project has investigated the application of an Artificial Immune System (AIS) to system identification. The problem domain investigated is that of a system which may enter different regimes; that is, a system for which the parameters may suddenly change. The goal of using an AIS, is to store newly-encountered regimes, and to retrieve those regimes upon subsequent encounters.

The RLS algorithm was investigated to obtain a method of determining that a change of regime has taken place, and as a method to determine the system parameters. The drop in the value of the forgetting factor was found to be a good indication of the change of regime, and was used throughout.

The value of the drop in forgetting factor was initially used to successfully test the concept of the AIS. However, this had limitations, such as the timing of the regime change being tied to the system reference. The solution was to use a quick estimate of the system parameters, and to pass that to the AIS to obtain a better solution. This was shown to work well, and was also shown to work in the presence of noise.

Table of Contents

1	Introduction.....	5
1.1	Background to the project and project objectives.....	5
1.2	Outline of approach	5
2	Artificial Immune Systems	7
2.1	Biological Immune Systems.....	7
2.2	Artificial Immune Systems.....	7
2.2.1	The CLONALG algorithm	9
2.3	AIS Applied to Control	11
3	Testing the AIS	12
3.1	Notes on the algorithm.....	12
3.2	Results.....	14
3.3	Conclusion.....	14
4	System Identification.....	15
4.1	The Recursive Least Squares (RLS) algorithm	15
4.2	Using the RLS algorithm	17
4.3	Identifying the change of system.....	19
4.4	Results.....	22
4.4.1	Analysing the Results.....	24
4.5	Problems with Using Lambda to Identify the New System Regime	25
4.6	Investigating the Value of Lambda for Three Systems	27
4.7	Conclusion.....	30
5	Use of an Artificial Immune System	31
5.1	Use of the AIS to encode lambda	31
5.1.1	Results	34
5.1.2	Conclusions	40
5.2	Use of the AIS to encode lambda for three systems	40
5.2.1	Results	41
5.3	Genovalue Based Affinity for Three Systems	48
5.4	Use of the AIS to Encode the System Parameters	49
5.4.1	Encoding the parameters	49
5.4.2	Results of Testing the Algorithm.....	51
5.4.3	Results of Using the AIS and Encoding the System Parameters	56
5.5	Noise on the Results	61
6	Conclusion	67
7	References and Bibliography	69
7.1	References.....	69
7.2	Bibliography.....	69
8	Appendix	70
8.1	Matrix class	70
8.2	The System class	71
8.3	RLS and RLSSys classes	72
8.4	Code for AIS to recognise binary strings	72
8.5	Code for the AIS to Recognise Systems based on Lambda	73
8.6	System Recognition based on System Parameters.....	76
8.6.1	Affinity Based on the System Parameters	76

List of Figures

Figure 1. Representation of the AIS.....	10
Figure 2. The CLONALG algorithm (de Castro and von Zuben, 2002, page 244).	11
Figure 3. The system reference signal.....	18
Figure 4. The system response to the reference signal for switching between system 1 and system 2.	18
Figure 5. The estimated parameters without covariance matrix resetting.	19
Figure 6. The estimated parameters with covariance matrix resetting.	19
Figure 7. The predicted error (VFF with covariance matrix resetting) as a result of switching from system 1 to system 2 then back to 1.	20
Figure 8. Lambda (VFF with covariance matrix resetting) as a result of switching from system 1 to system 2 then back to 1.....	20
Figure 9. The predicted error (VFF with covariance matrix resetting) as a result of switching from system 1 to system 2 then back to 1.	20
Figure 10. Lambda (VFF with covariance matrix resetting) as a result of switching from system 1 to system 2 then back to 1.....	20
Figure 11. Switching 25 iterations later.	26
Figure 12. Switching 13 iterations later.	26
Figure 13. Switching 13 iterations later, with PRBS reference.	26
Figure 14. The system output for 3 systems.	27
Figure 15. Lambda for changing between three systems, matrix reset trigger of 0.0.	28
Figure 16. Estimated parameters for changing between three system, matrix trigger of 0.0.....	28
Figure 17. Lambda for changing between three systems, matrix reset trigger of 0.8.	29
Figure 18. Estimated parameters for changing between three system, matrix trigger of 0.8.....	29
Figure 19. Illustrating the storage of systems in the AIS.	31
Figure 20. The Effect of Noise on the Parameter Estimation.	61
Figure 21. UML diagram for using lambda to classify the system.	74
Figure 22. The Antigen class.....	75
Figure 23. The AIS class.	75
Figure 24. The Population class.....	75

List of Tables

Table 1. Illustration of Crossover	13
Table 2 Results of testing the AIS algorithm for recognition of binary strings.	14
Table 3. System Switching Permutations	27
Table 4. Values of minimum lambda for a matrix reset trigger of 0.8.	28
Table 5. Values of minimum lambda for a matrix reset trigger of 0.8.	29
Table 6. Phenovalues corresponding to minimum lambda for a matrix reset trigger of 0.8.	40
Table 7. Expected results of running the program.	41
Table 8. Table examining convergence of antibodies towards antigens.	48
Table 9 The coefficients for the three systems.	50
Table 10 The difference between the systems.	51
Table 11. Expected Results of Using an AIS to Identify Systems Based on their Parameters.	56
Table 12. The operations supported on the matrix class.	71

List of Abbreviations

AIS	Artificial Immune System
FFF	Fixed Forgetting Factor
GA	Genetic Algorithm
PRBS	Pseudo Random Binary Sequence
RLS	Recursive Least Squares
VFF	Variable Forgetting Factor

1 Introduction

1.1 Background to the project and project objectives

A plant may operate in many regimes, each of which may be modelled by a different system, for example, a steel rolling mill may operate in different regimes for producing products of a different thickness, or when operating on different alloys, or for alloys at different temperatures.

It is required that a change in system is recognised, and that the new system is identified. An Artificial Immune System (AIS) allows a library of systems to be maintained, some of the systems contained within it are allowed to breed following the rules of Genetic Algorithms (GAs).

The objective of the project is to recognise when a change of system takes place, then to identify the change in system and recognise that system from an implemented artificial immune system.

1.2 Outline of approach

Chapter 2 discusses artificial immune systems, and describes an algorithm for implementing an artificial immune system for pattern recognition. Issues involved in system identification, including recognising that the regime has changed, and identifying the systems, are discussed in Chapter 4.

The use of an artificial immune system in system identification is discussed in Chapter 5. This chapter makes use of the work in Chapter 4 to recognise systems to add and retrieve from the AIS. The initial part of this chapter uses the magnitude of the drop in forgetting factor as an indicator of the system change. This approach has limitations, but it is simple to implement and so gives a quick indication of whether the approach works.

The latter part of chapter 4 investigates using a quick estimate of the system parameters as an identification of the system; better estimates of the system are retrieved and stored in the AIS. This allows the system identification to operate independent of the reference signal, and with noise on the system output.

2 Artificial Immune Systems

This chapter discusses the motivation behind using an artificial immune system. Following this, an algorithm for implementing an artificial immune system for system recognition is discussed.

2.1 Biological Immune Systems

Immune systems in nature defend the host organism, e.g. the human body, against harmful pathogens (viruses or bacteria) originating from outside the host. The pathogens are known as antigens, and the body defends itself by a collection of antibodies. These antibodies are distributed throughout the body in the blood stream, and recognise the antigens as not being part of the body, or as being non-self. As described in Dasgupta, 1999 (page 7), there is much interest in taking inspiration from the natural immune system:

"The natural immune system is a subject of great research interest because of its powerful information processing capabilities. In particular, it performs many complex computations in a highly parallel and distributed fashion."

The information contained in the immune system is distributed throughout the body, in the actual antibodies contained within the blood stream.

2.2 Artificial Immune Systems

Artificial Immune Systems make use of and extend upon ideas from Genetic Algorithms (GA). In a genetic algorithm, a set of chromosomes, which are easily, but not necessarily, represented as binary strings, are allowed to breed (crossover) and mutate, with the probability of a chromosome being selected for crossover being proportional to its fitness. Usually, however, the GA is only used to search for one optimum solution.

Forrest, et al (1992) describes an algorithm for an artificial immune system. In their algorithm a form of implicit fitness sharing is employed to "allow the GA to discover a set of pattern matching antibodies that effectively match a set of antigen patterns". Their algorithm allows the GA to maintain a diverse population; this is achieved by setting each member's

fitness to be interdependent of the fitness of each other member of the population. Their method has advantages over using explicit fitness sharing, which has the following limitations:

- It required a comparison of every population member to every other population member in each generation (N^2 comparisons).
- Setting the critical parameter.

Their work gives details of the implicit fitness sharing, which is achieved by selecting an antigen and a number of antibodies from the antibody population, and assigning a fitness to each antibody according to its match score. The match score is a function that makes some comparison of how well an antibody and an antigen match (this measure may be the number of similar bits).

Fitness scoring is as follows:

1. A single antigen is randomly selected from the antigen population.
2. From the population of N antibodies a randomly selected sample of size σ is taken without replacement.
3. For each antibody in the sample, match it against the selected antigen, determine the number of bits that match, and assign it a match score.
4. The antibody in the sample group with the highest match score is determined.
5. The match score of the winning antibody is added to its fitness.
6. This process is repeated for C cycles.

An artificial immune system allows many solutions to be found simultaneously, and can be used to solve optimisation or pattern recognition problems (as described in Dasgupta, 2002). This report investigates system identification, which makes use of pattern recognition in the AIS. The task of the AIS is only to identify the antigen; the analogy with the biological case ends here, as there is no destruction of the antigen as would happen in a biological

immune system. The antibodies have an affinity with each of the antigens - the affinity is the measure of how well an antigen and an antibody match with each other.

The AIS algorithm works by having a population of antibodies, which follow GA breeding routines to match the population of antigens. The algorithm makes use of cloning, as in genetic algorithm schemes, to search the search space (the n-dimensional space that the variables exist over) for a better affinity match for each of the antibodies. Also, a new set of antibodies is generated on each breeding cycle, and this replaces a certain number of repository antibodies.

2.2.1 The CLONALG algorithm

The CLONALG algorithm (de Castro and Von Zuben, 2002) was used. In this algorithm, the antibody population is made up of two sets:

- a memory set of antibodies, each of which is a match to one the antigens to be recognised.
- a repository of antibodies, these may be, but are not necessarily, a match to any of the antigens.

The CLONALG algorithm is now described (Antibodies are represented as Ab, antigens are represented as Ag). The following sets of antibodies and antigens are used:

- Ag: M antigens to be recognised.
- Ab: N antibodies, split into
 - Ab_m: m memory antibodies. $m \leq N$
 - Ab_r: remaining antibodies $r = N - m$
- Ab^j: n Abs from total Ab population with highest affinities to Ag^j. ($n \leq N$).
- Ab_d: Set of d new antibodies that will replace the lowest d affinity Ab's from Ab_r.

The CLONALG algorithm is:

```

FOR loopvar1 = 1 TO ngen
  Select a random antigen,  $Ag^j$ , from the antigen population.
  FOR loopvar2 = 1 TO  $N$ 
    Present  $Ag^j$  to each antibody,  $Ab_{\{m\}}$  and calculate affinity  $f^j$ .
  END FOR
  Create a new population  $Ab_{\{n\}}^j$  and select and store the  $n$  antibodies from  $Ab$  with
  the highest affinity to  $Ag^j$ 

  Create a population  $C^j$  of  $N_c$  clones
  FOR loopvar2 = 1 TO  $N_c$ 
    Select two parents for cloning from  $Ab_{\{n\}}^j$ 
    Perform crossover. Store result in  $C^j$ 

    Mutate the clones, to create  $C^{j*}$ . Mutate with a high mutation rate for low values of
     $f^j$ , and vice-versa

    Present  $Ag^j$  to each  $C^{j*}$  to calculate affinity  $f^{j*}$ .

  END FOR

  Select clone with highest affinity to  $Ag^j$ ,  $f^{j*}$ . If this clone has a higher affinity to
   $Ag^j$  than the respective antibody in the memory set, replace the clone in the memory
  set.

  Create a set of antibodies,  $Ab_{\{d\}}$ ,  $d \leq r$ . Replace lowest  $d$  antibodies in  $Ab_{\{r\}}$  with
   $Ab_{\{d\}}$ .

END FOR
  
```

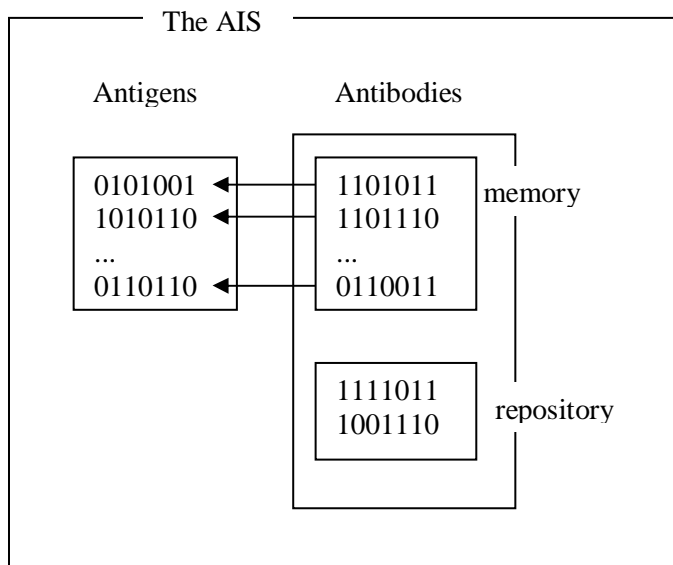


Figure 1. Representation of the AIS.

The figure above represents the AIS; each of the memory antibodies is associated with an antigen. The repository antibodies represent a set of antibodies, not directly associated with an antigen, which are gradually replaced with new information, as is described in the algorithm above, and Figure 2.

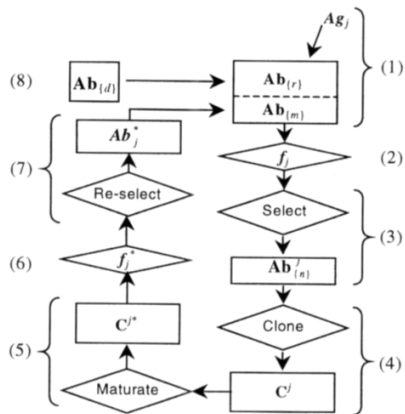


Figure 2. The CLONALG algorithm (de Castro and von Zuben, 2002, page 244).

2.3 AIS Applied to Control

The task of the AIS is to take some property of the antigen, encode this property, and store it in the AIS. Along with this property, the AIS needs to store a model of the system to be retrieved when it is recognised that the system has moved into this regime. The method of identifying the systems will be discussed in Chapter 4.

3 Testing the AIS

The work in this chapter investigates the use of the CLONALG algorithm to recognise a randomly generated set of binary strings. This illustrates the breeding of the AIS, and understanding how this works for this simple case will be of help later on. The algorithm is as appeared in section 2.2, with the following parameters and initialisation:

- Set string length of antibodies and antigens to 11
- Generate 5 random antigens
- Generate 5 random memory antibodies
- Generate 100 random repository antibodies

The Artificial Immune System is bred for 50 generations, as in the algorithm in section 2.2. The parameters used for breeding are: 50 of the *nbest* antibodies selected for cloning, with 100 clones being generated. At each step 30 new antibodies are generated to replace the 30 worst repository antibodies.

3.1 Notes on the algorithm

The affinity is calculated as the number of similar bits (bits of the same value in the same position) of the antibody and antigen binary strings.

The selection of the two parents for cloning makes use of tournament selection. Two antibodies are randomly chosen as candidates for the first parent, the one with the highest affinity is selected. The second parent is chosen randomly.

Crossover is performed by uniform crossover, where a random binary string of the same length as each chromosome (the antibody) is created, and each bit is set randomly to be '1' or '0'. For each bit in the string, if the bit is a '1' then the child bit takes its value from parent 1, else it takes its value from parent 2. This is shown in Table 1, where the bit chosen to enter the child from the crossover mask is emboldened in the selected parent, for clarity.

Description	String
Parent 1	1011001101110
Parent 2	0101110110100
Crossover Mask	1011010010110
Child	1111100100110

Table 1. Illustration of Crossover

Afterwards, the children are subjected to an affinity maturation process as follows: A mutation mask is generated for the child; for each bit in the child's string, if the mutation mask has a value of '1' then the child's bit is inverted, else it is left the same.

The mask is generated by selecting a random number having a value $0 \leq random \leq 1$. If this value is less than the mutation rate, then the mask is set to '1'. The mutation rate is higher for high values of antibody affinity, and lower for lower affinities. The mutation rate is calculated as follows:

$$mutationRate = maxMutationRate * (stringLength - affinity) / stringLength \quad 1$$

with a *maxMutationRate* of 0.5. With this, a mutation rate of 0.5 is calculated for no bits matching in the string, and a mutation rate of 0 is given if all of the bits match.

3.2 Results

The results in Table 2 show the convergence of the antibodies towards the value of the antigens. The convergence takes place in under 30 iterations. Each breeding cycle takes 0.11 seconds there is no replacement of a memory antibody, and takes 0.17 seconds where replacement of a memory antibody takes place.

Iteration	String 1	String 2	String 3	String 4	String 5
Antigen	11000110010	01111001010	00111011001	01111101000	00101110000
1	10001110111	10101001100	10110100110	11010011110	11110111100
2	10001110111	10101001100	10110100110	01111100000	11110111100
3	10001110111	10101001100	00011011001	01111100000	11110111100
4	10001110111	10101001100	00011011001	01111101000	11110111100
5	10001110111	10101001100	00011011001	01111101000	11110111100
6	10001110111	10101001100	00011011001	01111101000	11110111100
7	11000110011	10101001100	00011011001	01111101000	11110111100
8	11000110011	10101001100	00011011001	01111101000	11110111100
9	11000110011	10101001100	00011011001	01111101000	11110111100
10	11000110011	10101001100	00011011001	01111101000	11110111100
11	11000110011	011111001010	00011011001	01111101000	11110111100
12	11000110011	011111001010	00011011001	01111101000	11110111100
13	11000110011	011111001010	00011011001	01111101000	11110111100
14	11000110011	011111001010	00111011001	01111101000	11110111100
15	11000110011	011111001010	00111011001	01111101000	00101110000
16	11000110011	011111001010	00111011001	01111101000	00101110000
17	11000110011	011111001010	00111011001	01111101000	00101110000
18	11000110011	011111001010	00111011001	01111101000	00101110000
19	11000110011	011111001010	00111011001	01111101000	00101110000
20	11000110011	011111001010	00111011001	01111101000	00101110000
21	11000110011	011111001010	00111011001	01111101000	00101110000
22	11000110011	011111001010	00111011001	01111101000	00101110000
23	11000110011	011111001010	00111011001	01111101000	00101110000
24	11000110011	011111001010	00111011001	01111101000	00101110000
25	11000110011	011111001010	00111011001	01111101000	00101110000
26	11000110011	011111001010	00111011001	01111101000	00101110000
27	11000110011	011111001010	00111011001	01111101000	00101110000
28	11000110010	011111001010	00111011001	01111101000	00101110000
Antigens	11000110010	011111001010	00111011001	01111101000	00101110000

Table 2 Results of testing the AIS algorithm for recognition of binary strings.

3.3 Conclusion

This chapter has shown that the CLONALG algorithm allows an AIS to be implemented, which can then be used for pattern recognition. This is the algorithm that will be used throughout the project.

4 System Identification

This chapter investigates a method of determining that a system change has taken place, and also investigates a method of determining the system regime being entered. The task of system identification is to identify the parameters of the system from the observed outputs and inputs. This may be done using the Recursive Least Squares (RLS) algorithm.

4.1 The Recursive Least Squares (RLS) algorithm

As mentioned, the RLS parameter may be used to estimate the model of a system. However, the estimation routine fails in cases where the system parameters are varying, as information on the previous state of the system is maintained in the algorithm. The algorithm may be adjusted to make use of forgetting factors, which are used to give more weighting to more recent observations; this allows the parameters of time-varying systems to be investigated. This technique fails with sudden changes in system parameters, but use can be made of the covariance matrix resetting technique to solve this problem.

The Recursive Least Squares algorithm, with Fixed Forgetting Factors (FFF) factors, is given in Harris and Billings, 1985, as:

$$\hat{\underline{\theta}}(t) = \hat{\underline{\theta}}(t-1) + \underline{K}(t) [\underline{\phi}(t) - \underline{x}'(t) \hat{\underline{\theta}}(t-1)] \quad 2$$

$\underline{x}(t)$ is a vector of the known data (the observation matrix)

$\hat{\underline{\theta}}(t)$ is an n-vector of the estimated parameters using all data up to time t.

$\underline{\phi}(t)$ is an 'output' observation

$\underline{K}(t)$ is an n-vector (the 'Kalman gain')

The estimation update for each parameter is of the feedback form:

$$[\text{new estimate}] = [\text{old estimate}] + \text{gain} * (\text{prediction error of old model})$$

Now, the FFF versions of $\underline{K}(t)$ and $\underline{P}(t)$ are

$$\underline{K}(t) = \frac{\underline{P}(t-1)\underline{x}(t)}{\lambda + \underline{x}'(t)\underline{P}(t-1)\underline{x}(t)} \quad 3$$

$$\underline{P}(t) = \left(I - \underline{K}(t)\underline{x}'(t) \right) \underline{P}(t-1) / \lambda \quad 4$$

$\underline{P}(t)$ is the covariance matrix.

Wellstead and Zarrop describe the use of a variable forgetting factor: With fixed forgetting factors, a value of λ of 0.95 is used for fast variations, and a value of 0.99 is used for slow variations. Variable forgetting factors are used to vary the weighting given to the observations dependent upon how fast the system parameters are changing. A value of λ of unity is used when the system is found to have constant parameters. The change of parameters can be identified using the prediction error; as the error grows it may mean that the identified model is incorrect and needs adjustment.

The RLS algorithm with variable forgetting factors may be implemented in the following algorithm:

```

define number of parameters
create matrix x, order n-by-1, and set to zero
create matrix theta, order n-by-1, and set to zero
create matrix P, order n-by-n, as an identity matrix
forgetting factor, lambda = 0.93
memory length, Mo = 1/(1-lambda)
set up previous observations (e.g. yt_1 = 0, yt_2 = 0, ut_1 = 0 etc.)
FOR i = 1 to numberOfObservations
    perror = yt(i) - x'*theta
    K = P*x/(1+x'*P*x)
    theta = theta + K*pererror
    lambda = 1 - (1 - x*K)*(pererror*pererror)/Mo
    P = (eye(n) - K*x')*P/lambda
    Time shift the variables (e.g. yt_2 = yt_1, ut_1 = ut etc.)
    Update the observation vector (e.g. x(1) = yt_2, x(2)=yt_1, x(3) = ut_1)
END FOR
    
```

4.2 Using the RLS algorithm

The RLS algorithm is implemented in `RLSSys.h`, as described in the appendix, section 8.3.

The algorithm was then investigated to verify the results of switching between two systems.

$$A(q^{-1})y(t) = q^{-k}B(q^{-1})u(t) + C(q^{-1})e(t) \quad 5$$

where

$$A(q^{-1}) = a_0 + a_1q^{-1} + a_2q^{-2} + \dots + a_{n_a}q^{-n_a}, \quad a_0 = 1 \quad 6$$

$$B(q^{-1}) = b_0 + b_1q^{-1} + b_2q^{-2} + \dots + b_{n_b}q^{-n_b} \quad 7$$

$u(t)$ is the system input, there is no control action, so $u(t)$ is equal to the system reference, $r(t)$. Equation 5 is arranged to give the system transfer function:

$$y(t) = \frac{(b_0 + b_1q^{-1} + b_2q^{-2} + \dots + b_{n_b}q^{-n_b})u(t)}{(a_1q^{-1} + a_2q^{-2} + \dots + a_{n_a}q^{-n_a})} \quad 8$$

q^{-1} is the time delay operator, so arranging 8 gives the system output in terms of previous inputs and outputs as:

$$y(t) = b_0 + b_1u(t-1) + b_2u(t-2) + \dots + b_{n_b}u(t-n_b) - a_1y(t-1) - a_2y(t-2) - \dots - a_{n_a}y(t-n_a) \quad 9$$

For a system to be stable, the poles of the transfer function should lie in the unit circle of the complex plane (i.e. the roots of the denominator should all be of magnitude < 1).

The transfer functions for the two systems being investigated are:

System 1:

$$y(t) = \frac{(0.5q^{-1} + 0.5q^{-2})u(t)}{(-0.5q^{-1} + 0.4q^{-2} + 0.0q^{-3})}$$

System 2:

$$y(t) = \frac{(1.5q^{-1} + 10q^{-2})u(t)}{(0.9q^{-1} - 0.7q^{-2} - 0.8q^{-3})}$$

The reference signal for both systems takes the form of a series of step inputs of period 50:

$$\begin{aligned} r=1 & \quad i \leq 0 < 25 + 50n \\ r=-1 & \quad i \leq 25 < 50 + 50n \end{aligned} \quad 10$$

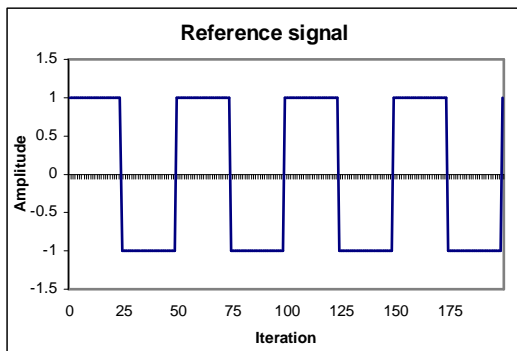


Figure 3. The system reference signal.

The system was changed to alternate between system 1 and system 2 every 200 iterations.

The system response to the reference signal for the two systems is shown in Figure 4:

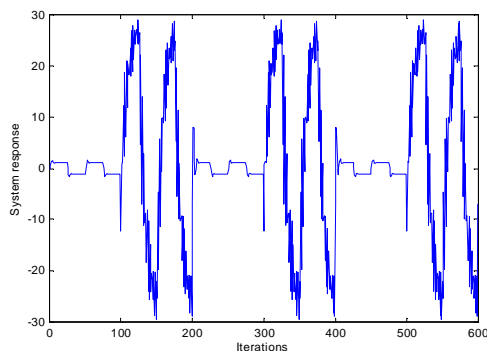


Figure 4. The system response to the reference signal for switching between system 1 and system 2.

The result of parameter estimation using the RLS algorithm, with forgetting factors but no covariance matrix reset, is shown in Figure 5. The results of parameter estimation using RLS with variable forgetting factors and covariance matrix resetting is shown in Figure 6. The parameters are incorrectly identified where there is no covariance matrix reset, information for the old system is retained in the algorithm.

Figure 6 also shows the effect of information on the RLS algorithm. At iteration 250, the reference changes, and the new information introduced into the algorithm gives a closer estimate of the system parameters.

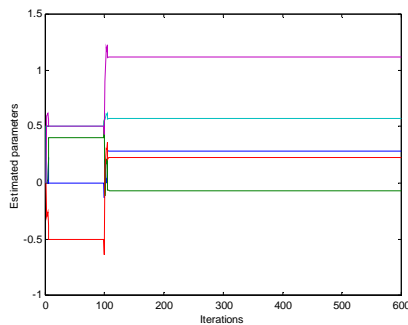


Figure 5. The estimated parameters without covariance matrix resetting.

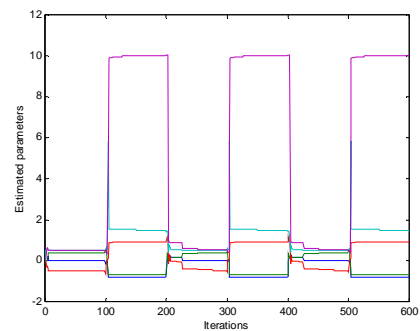


Figure 6. The estimated parameters with covariance matrix resetting.

4.3 Identifying the change of system

As can be seen in Figure 7 and Figure 9, the minimum value of lambda when switching from system 1 to system 2 (-7.46565), and the value when switching from system 2 to system 1 (-16.0319), is the same no matter if the initial system is system 2 or system 1. These results indicate that the value of lambda can be used as an identification of the new system regime being entered. These figures also show that an indication that a change of system has occurred is when the value of lambda falls below zero.

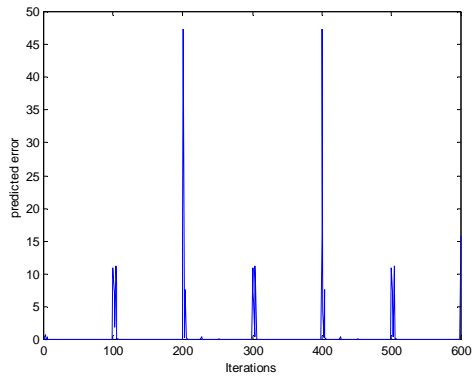


Figure 7. The predicted error (VFF with covariance matrix resetting) as a result of switching from system 1 to system 2 then back to 1.

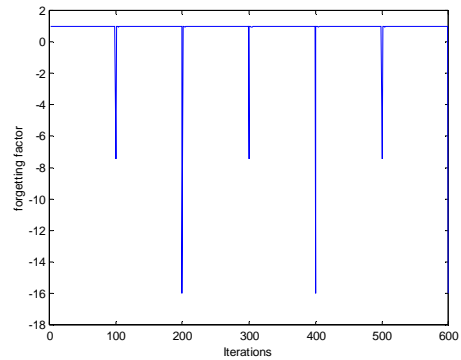


Figure 8. Lambda (VFF with covariance matrix resetting) as a result of switching from system 1 to system 2 then back to 1.

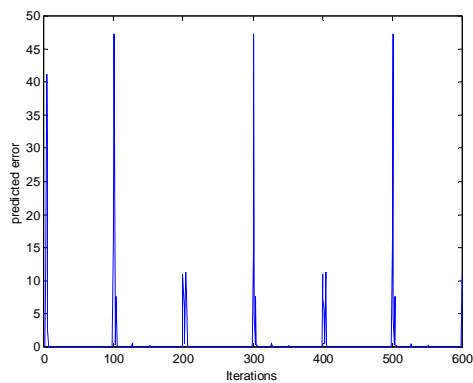


Figure 9. The predicted error (VFF with covariance matrix resetting) as a result of switching from system 1 to system 2 then back to 1.

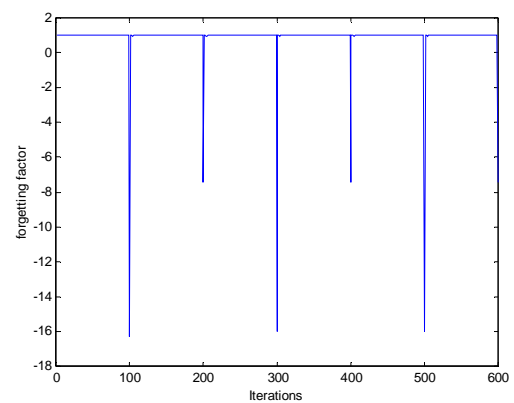


Figure 10. Lambda (VFF with covariance matrix resetting) as a result of switching from system 1 to system 2 then back to 1.

This indicates that the value of lambda can be stored along with the system that it identifies. The initial drop of lambda identifies that the change of system has taken place. However, the value of lambda at this iteration is not a good identifier of the system, the minimum value of lambda is required. When the minimum value of lambda is obtained then that value can be presented to the library of antigens to allow the relevant stored system to be retrieved.

A system cannot be added to the library of antigens and antibodies until both the minimum value of lambda, which identifies the system, and also the system model itself have been obtained. The algorithm to perform this is as follows:

```
systemChangeDetected = False
lambdaMin = 0
FOR i=1 TO number of iterations
  Perform RLS algorithm
  IF lambda < 0      (indicates that change of lambda has been detected)
    IF systemChangeDetected = False
      systemToStore = systemRLS
      (if previous system had no system retrieved for that value of lambda)
      IF(systemAIS is empty)
        store systemAIS in the AIS
      END IF
      systemChangeDetected = True
    END IF
  END IF

  (check for covariance matrix reset)
  IF lambda < 0.0
    reset the covariance matrix
  END IF

  IF systemChangeDetected = True
    IF lambdaMin < lambda
      lambdaMin = lambda
    ELSE
      lambdaAIS = lambdaMin      (lambda to add to AIS)
      systemAIS = retrieved system from AIS with corresponding lambdaAIS

      (reset variables for detection of next change of system)
      lambdaMin = 0
      systemChangeDetected = False
    END IF
  END IF
  Update system parameters (systemRLS)
END FOR
```

4.4 Results

The previous algorithm may be put into practice without using an Artificial Immune System, but by storing and retrieving the system parameters and corresponding value of lambda in a library of antigens. There are no antibodies so no breeding or mutation takes place. The results of this are shown below. The system response is generated with system 1 as the initial system; the reference is as in equation 10, and the change of system takes place at every 100th iteration, as in Figure 3. This allows lambda to be used as an indication of the new system.

The result of encoding lambda in the antigens and making use of the artificial immune system is discussed in section 5.1. The program listing used to produce this output is found in Appendix 8.5, `sysID16lambda.cpp`.

```
-----  
initial change in system detected, i = 100, systemRLS:  
0.5*q^-1 + 0.500003*q^-2  
-----  
1 + -0.499994*q^-1 + 0.399994*q^-2 + 3.13804e-006*q^-3  
-----  
  
previous system was empty, adding identified system to AIS  
lambdaAIS = 0  
details of the identified systemToStore:  
0.5*q^-1 + 0.500003*q^-2  
-----  
-0.499994*q^-1 + 0.399994*q^-2 + 3.13804e-006*q^-3  
-----  
  
i=100 lambdaMin=0 lambda=-7.45656  
  
Change in system, min value of lambda detected, i=101. lambdaMin=-7.45656  
  
Attempting to obtain system from AIS (for lambdaMin=-7.45656):  
empty: no system identified for new regime.  
i=101 resetting values  
  
-----  
initial change in system detected, i = 200, systemRLS:  
1.50558*q^-1 + 9.99039*q^-2  
-----  
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3  
-----  
  
previous system was empty, adding identified system to AIS  
lambdaAIS = -7.45656  
details of the identified systemToStore:  
1.50558*q^-1 + 9.99039*q^-2  
-----  
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3
```

Gary Evans
System Identification using an Artificial Immune System

i=200 lambdaMin=0 lambda=-16.0319

Change in system, min value of lambda detected, i=201. lambdaMin=-16.0319

Attempting to obtain system from AIS (for lambdaMin=-16.0319):

empty: no system identified for new regime.

i=201 resetting values

initial change in system detected, i = 300, systemRLS:

$0.494626*q^{-1} + 0.537544*q^{-2}$

 $-0.467043*q^{-1} + 0.384235*q^{-2} + 0.0121983*q^{-3}$

previous system was empty, adding identified system to AIS

lambdaAIS = -16.0319

details of the identified systemToStore:

$0.494626*q^{-1} + 0.537544*q^{-2}$

 $-0.467043*q^{-1} + 0.384235*q^{-2} + 0.0121983*q^{-3}$

i=300 lambdaMin=0 lambda=-7.46274

Change in system, min value of lambda detected, i=301. lambdaMin=-7.46274

Attempting to obtain system from AIS (for lambdaMin=-7.46274):

system retrieved, systemAIS details:

$1.50538*q^{-1} + 9.99022*q^{-2}$

 $0.879765*q^{-1} + -0.723363*q^{-2} + -0.801564*q^{-3}$

i=301 resetting values

initial change in system detected, i = 400, systemRLS:

$1.50553*q^{-1} + 9.99045*q^{-2}$

 $0.899522*q^{-1} + -0.700091*q^{-2} + -0.799593*q^{-3}$

i=400 lambdaMin=0 lambda=-16.0319

Change in system, min value of lambda detected, i=401. lambdaMin=-16.0319

Attempting to obtain system from AIS (for lambdaMin=-16.0319):

system retrieved, systemAIS details:

$0.488759*q^{-1} + 0.527859*q^{-2}$

 $-0.488759*q^{-1} + 0.371457*q^{-2} + -0.0195503*q^{-3}$

i=401 resetting values

initial change in system detected, i = 500, systemRLS:

$0.494626*q^{-1} + 0.537544*q^{-2}$

 $-0.467043*q^{-1} + 0.384235*q^{-2} + 0.0121983*q^{-3}$

i=500 lambdaMin=0 lambda=-7.46274

Change in system, min value of lambda detected, i=501. lambdaMin=-7.46274

Gary Evans
System Identification using an Artificial Immune System

```
Attempting to obtain system from AIS (for lambdaMin=-7.46274):
system retrieved, systemAIS details:
1.50538*q^-1 + 9.99022*q^-2
-----
0.879765*q^-1 + -0.723363*q^-2 + -0.801564*q^-3

i=501 resetting values

-----
initial change in system detected, i = 600, systemRLS:
1.50553*q^-1 + 9.99045*q^-2
-----
0.899522*q^-1 + -0.700091*q^-2 + -0.799593*q^-3

-----

i=600 lambdaMin=0 lambda=-16.0319
```

4.4.1 Analysing the Results

The results show the algorithm in operation. At the 100th iteration, the value of lambda drops below zero correctly identifying that a change in system has taken place. There was no system previously obtained from the AIS so the system is added to the RLS with the value of *minLambda*. As this is the first system change, *minLambda* has the value of zero (in effect the value from changing from no system to system 1 at the first iteration). The system to store has been correctly identified to 5 decimal places.

At the 101st iteration the minimum value of lambda (-7.45656) is detected, so an attempt can be made to retrieve the system from the AIS. There is no corresponding system in the AIS at this time.

A change of system is detected at the 200th iteration, the previous system obtained from the RLS was empty so the regime that is now being moved out of (system 2, the regime that has just been identified) is added to the AIS with the previously found value of *minLambda* which characterises this regime (the change of system 1 to system 2). The system parameters are not so accurately identified (only to 2 decimal points), this may be due to information remaining in the system.

At the 201st iteration the minimum value of lambda is found (-16.0319) so an attempt is made to retrieve a system from the AIS. An empty system is returned, as a system for this value of lambda is not present in the AIS.

At the 300th iteration, the identified system (system 1) is added to the AIS with the value of lambda associated with switching from system 2 to system 1 (-16.0319). Again, the system parameters are less well identified than previously.

At the 301st iteration, the minimum value of lambda is found to be -7.46274. This is the value associated with switching from system 1 to system 2. System 2 is retrieved from the AIS.

At the 400th iteration the previous systemAIS value is not empty, signifying that a system was found in the AIS for the previous system change, so no system needs to be added to the AIS. At the 401st iteration System 1 is retrieved from the AIS.

Similarly, no system is added to the AIS at the 500th iteration, and the correct system is retrieved from the AIS.

The estimation of the parameters at each time interval get less and less accurate - this may be due to information remaining in the RLS algorithm, the covariance matrix does not remove all information from the algorithm.

4.5 Problems with Using Lambda to Identify the New System Regime

As seen in the previous section, the value of lambda is a good identification of the system, but this is dependent on when the change of system takes place. The previous section only investigated changing system at the iteration where the reference changed from -1 to 1.

Figure 11 shows the result of changing system 25 iterations later than in the previous section, i.e. when the reference switches from 1 to -1 rather than -1 to 1. The minimum value of lambda associated with switching from system 1 to system 2, and the value for switching from system 2 to system 1 are the same as before. Thus, lambda is still a good indication of the new system in this situation. However, Figure 12 shows the result of changing systems 13 iterations later than in the previous section, i.e. when the reference is unchanging. It can be seen that the value of lambda is different from the previous situations, and is not constant even in this case. Figure 13 confirms this; it shows the result of changing input where the reference is a Pseudo Random Binary Sequence (PRBS). This is the most realistic result, as in reality, the change in system will not necessarily be linked to the reference value.

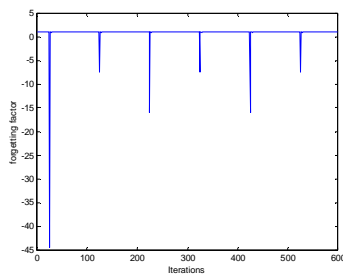


Figure 11. Switching 25 iterations later.

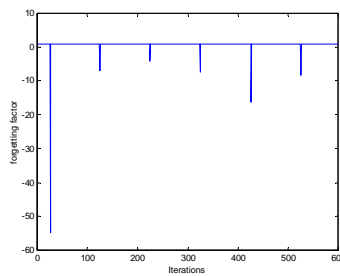


Figure 12. Switching 13 iterations later.

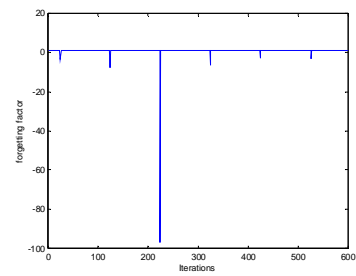


Figure 13. Switching 13 iterations later, with PRBS reference.

These results show that the value of lambda cannot be used to indicate the new regime being entered. However, the change in lambda (where the value of lambda drops below zero) remains a good indicator that the system regime has changed.

4.6 Investigating the Value of Lambda for Three Systems

With 3 systems, there are many more permutations of system changes; i.e. system 1 may switch to either system 2 or system 3. The possible system changes are listed below.

From	To
system 1	system 2
system 1	system 3
system 2	system 1
system 2	system 3
system 3	system 1
system 3	system 2

Table 3. System Switching Permutations

This may be realised by switching systems in the following sequence:

1 -> 2 -> 3 -> 1 -> 3 -> 2 -> 1 -> 2

The parameters for system 1 and system 2 were as previously, the parameters for system 3 are:

System 3:

$$y(t) = \frac{(0.7q^{-1} + 2.0q^{-2})u(t)}{(-0.2q^{-1} + 0.2q^{-2} + -0.7q^{-3})}$$

The output of switching between these three systems is shown in Figure 14:

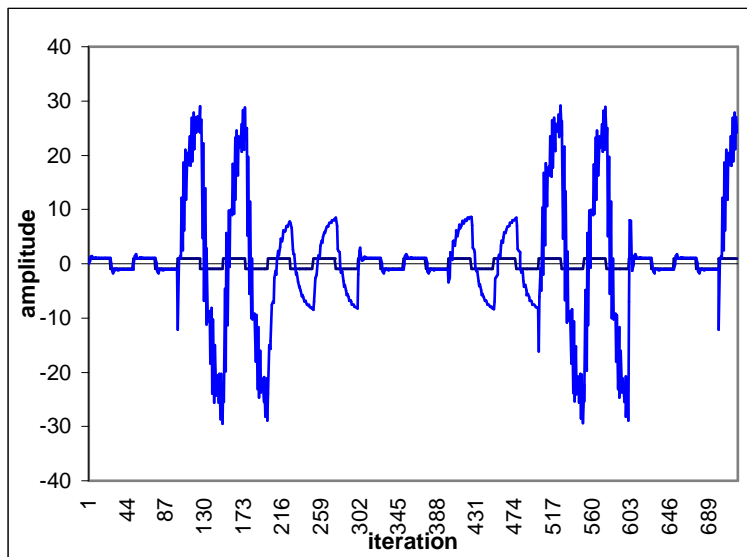


Figure 14. The system output for 3 systems.

The value of lambda for these systems is shown in Figure 15. There is no well-defined spike for lambda at the 400th iteration, this is because the covariance matrix is reset if the value of lambda falls below 0.0, and this is not sufficient in this case as the minimum value of lambda is higher than 0.0. This can be seen in the estimation of the parameters, in Figure 16. The parameters for system 3, which is between iterations 400 and 500, are incorrectly identified due to the information for system 1 still being present in the RLS algorithm. The minimum values of lambda associated with the system changes are:

From	To	Minimum lambda
system 1	system 2	-7.45656
system 1	system 3	0.61227
system 2	system 1	-15.9825
system 2	system 3	-3.33954
system 3	system 1	-1.88064
system 3	system 2	-3.07386

Table 4. Values of minimum lambda for a matrix reset trigger of 0.8.

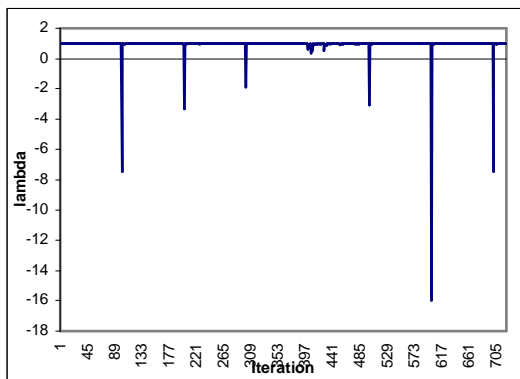


Figure 15. Lambda for changing between three systems, matrix reset trigger of 0.0.

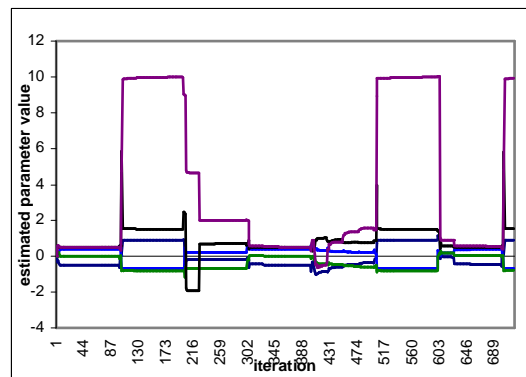


Figure 16. Estimated parameters for changing between three system, matrix trigger of 0.0.

Now, the covariance matrix reset is triggered if the value of lambda falls lower than 0.8, the results of this are shown in Figure 15 and Figure 16. The system parameters are now correctly identified. The minimum values of lambda are shown in Table 5.

From	To	Minimum lambda
system 1	system 2	-7.45656
system 1	system 3	0.61227
system 2	system 1	-15.9825
system 2	system 3	-3.33954
system 3	system 1	-1.88064
system 3	system 2	-3.08578

Table 5. Values of minimum lambda for a matrix reset trigger of 0.8.

The minimum values of lambda for triggering a covariance matrix reset when lambda falls below 0.8, are identical to the values for a reset when lambda falls below 0.0, except for the switch from system 3 to system 2. The different value of lambda may be due to the fact that the parameters for system 3 are incorrectly identified.

The results show that lambda may still be used to recognise that a change of system has taken place, and also to identify the new system. However, the values associated with changing from system 2 to system 3 (-3.33954) and system 3 to system 2 (-3.08578) are similar. This may give problems later on, and it's feasible that other systems may produce values of lambda that are even closer. The effect that similar values of lambda has on system identification is investigated in section 5.2.

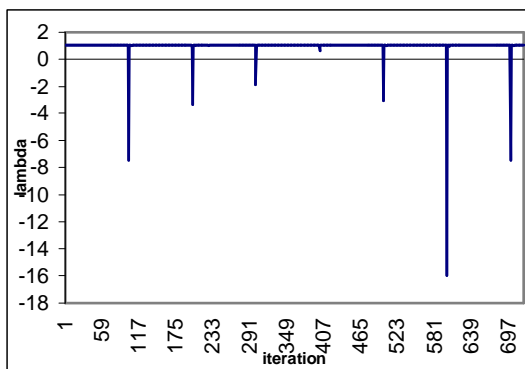


Figure 17. Lambda for changing between three systems, matrix reset trigger of 0.8.

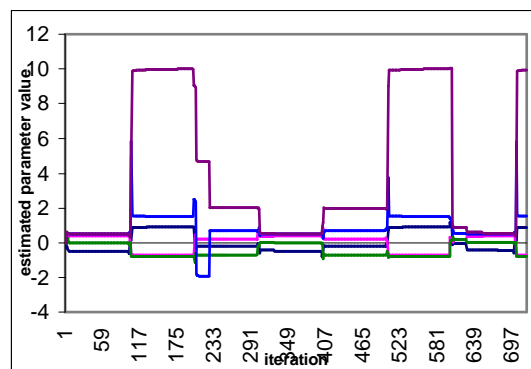


Figure 18. Estimated parameters for changing between three system, matrix trigger of 0.8.

4.7 Conclusion

This chapter has shown that a drop in forgetting factor indicates that a change in system has taken place. This drop was investigated for changing between two and three systems. The drop was also shown to be a trigger that allows the new regime being entered to be recognised, with the limitation that the timing of the change in system is dependent on the reference signal.

5 Use of an Artificial Immune System

This chapter investigates the use of an artificial immune system for system identification. The chapter starts by investigating the use of lambda to recognise the systems being encountered, for both two and three systems. The number of bits used to encode lambda in a binary string, and the associated dependence on the breeding cycle is investigated.

Following this, using the system parameters as an identification of the system change is investigated. Again, the encoding of the system parameters, including the number of bits used and the amount of breeding to take place are investigated.

5.1 Use of the AIS to encode lambda

The use of an AIS to encode lambda is now investigated. Initially lambda is presented to the AIS, and stored as an antigen along with the corresponding system parameters. Antibodies are created and mutated to match the library of antigens encountered by the system identification algorithm. To retrieve a system from the AIS, a value of lambda is presented and the memory antibodies searched through until a good match is found, for which the corresponding system is returned. Following this, work is done to investigate the identification of a system based on its parameters.

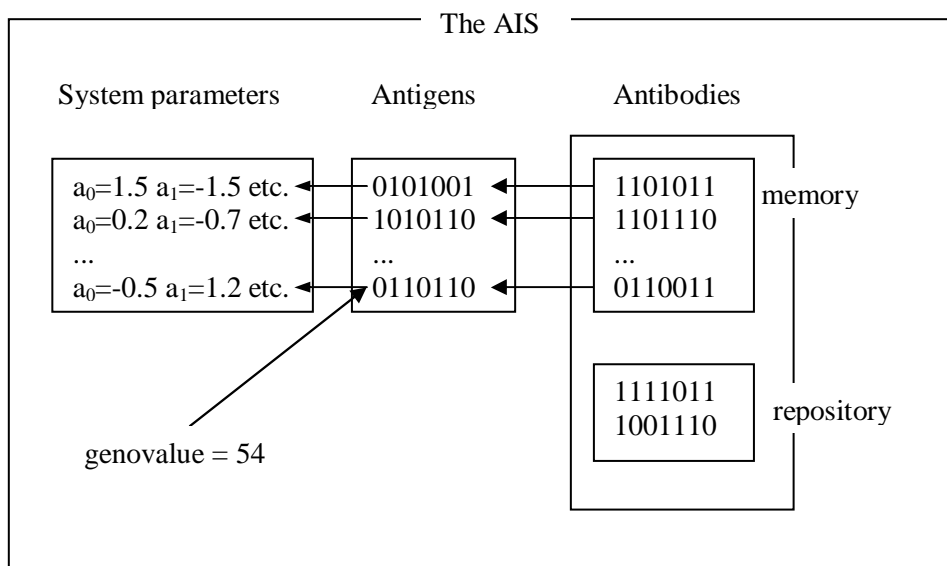


Figure 19. Illustrating the storage of systems in the AIS.

Lambda is encoded as a binary string to allow it to be easily represented in the antigens or antibodies (known as chromosomes in the Genetic Algorithm).

The Genetic Algorithm representation takes ideas from biology, the real-world parameter value (lambda) is called the phenotype, and this is mapped into the value encoded into the binary string, known as the genotype.

The genotype-phenotype mapping is as follows.

$$\text{genoValue} = \text{minGeno} + (\text{pheno} - \text{minPheno}) / (\text{maxPheno} - \text{minPheno}) * (\text{maxGeno} - \text{minGeno}) \quad 11$$

and so

$$\text{phenoValue} = \text{minPheno} + (\text{geno} - \text{minGeno}) / (\text{maxGeno} - \text{minGeno}) * (\text{maxPheno} - \text{minPheno}); \quad 12$$

where the minimum value of the genotype is zero, and the maximum value of the genotype is the maximum value of the binary string, which is $2^N - 1$, where N is the length of the binary string.

The phenotype values are obviously chosen so that they can hold the range that the parameters are to take.

There is a limit to the resolution of the phenotype due to encoding the value as a binary string, this gives an error in the value. The resolution of the genotype is given as $1/(\text{maxGeno})$, and the resolution of the phenotype is the range that the pheno value takes multiplied by the resolution of the geno value:

$$(maxPheno-minPheno)/maxGeno$$

Obviously, higher bit rates lessen the error in the phenotype. The real-world value is always rounded down to nearest pheno value giving a value less than or equal to the real-world value. This means that the possible error in the pheno value has a maximum value equal to the resolution.

The same algorithm is used as in section 4.3. However, instead of only keeping an antigen population, which is searched for a good match, a collection of memory antibodies is kept alongside the set of antigens. These antibodies are allowed to breed to converge on the antigens. The antibodies are searched, and if one of them is a good match then lambda and the system are retrieved from the antigen that it corresponds to.

The two values of lambda to be stored/recognised for system 2 and system 1 are (from section 4.3), respectively, -7.45656 and -16.0319. However, a value of lambda of 0 has to be stored to represent switching from no system (the initial condition) to system 1.

The number of bits used to encode lambda gives the precision for lambda, or the number of possible values between the minimum and maximum values of the pheno value, For example, with two bits to encode lambda, the possible values of the binary string (the geno values) are 0, 1, 2 and 3, with corresponding pheno values of 0.0, -5.666, -11.333 and -17.0. As the geno value is always rounded down, the pheno value for a value of lambda - 16.0319 will be -11.3333; this is not to a sufficient accuracy. Using 3 bits gives geno values of 0, 1, 2, ... , 7. These correspond to pheno values of 0.0, -2.42857, -4.85714, -7.28571, -9.71429, -12.14286, -14.57143 and -17.0.

The phenovalues are separated by a value of 2.42857. To allow the AIS to return a result for the next nearest phenovalue, but not the one after, the threshold for a match of 2.5 was used. i.e if the magnitude of difference between the value of lambda associated with the system to be recognised, and the value of lambda stored in the antigen, is within 2.5 then a positive match occurs and the system stored with the associated antigen is returned from the AIS.

5.1.1 Results

The program has been modified so that the value of lambda stored in each antibody is displayed each time that an attempt is made to retrieve an antibody from the artificial immune system. The result of running the program (sysID16lambda.cpp, in appendix 8.5) with the following AIS parameters and a string length of 3 is shown below.

AIS parameters:

1 repository antibody

3 clones

nbest = 2

d=1

3 breeding cycles

```
-----  
initial change in system detected, i = 100, systemRLS:  
0.5*q^-1 + 0.500003*q^-2  
-----  
-0.499994*q^-1 + 0.399994*q^-2 + 3.13804e-006*q^-3  
-----  
  
previous system was empty, adding identified system to AIS  
lambdaAIS = 0  
details of the identified systemToStore:  
0.5*q^-1 + 0.500003*q^-2  
-----  
-0.499994*q^-1 + 0.399994*q^-2 + 3.13804e-006*q^-3  
  
lambda=0 genovalue=0  
  
-----  
i=100 lambdaMin=0 lambda=-7.45656  
  
Change in system, min value of lambda detected, i=101. lambdaMin=-7.45656  
  
lambdaAb for antibody[0]=0  
Attempting to obtain system from AIS (for lambdaMin=-7.45656):  
empty: no system identified for new regime.  
i=101 resetting values  
  
-----  
initial change in system detected, i = 200, systemRLS:  
1.50558*q^-1 + 9.99039*q^-2  
-----  
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3  
-----
```

Gary Evans

System Identification using an Artificial Immune System

```
previous system was empty, adding identified system to AIS
lambdaAIS = -7.45656
details of the identified systemToStore:
1.50558*q^-1 + 9.99039*q^-2
-----
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3

lambda=-7.45656 genoalue=3

-----
i=200 lambdaMin=0 lambda=-16.0319

Change in system, min value of lambda detected, i=201. lambdaMin=-16.0319

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-14.5714
system obtained, i=1
Attempting to obtain system from AIS (for lambdaMin=-16.0319):
system retrieved, systemAIS details:
1.50558*q^-1 + 9.99039*q^-2
-----
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3

i=201 resetting values

-----
initial change in system detected, i = 300, systemRLS:
0.494626*q^-1 + 0.537544*q^-2
-----
-0.467043*q^-1 + 0.384235*q^-2 + 0.0121983*q^-3

-----
i=300 lambdaMin=0 lambda=-7.46274

Change in system, min value of lambda detected, i=301. lambdaMin=-7.46274

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-14.5714
Attempting to obtain system from AIS (for lambdaMin=-7.46274):
empty: no system identified for new regime.
i=301 resetting values

-----
initial change in system detected, i = 400, systemRLS:
1.50553*q^-1 + 9.99045*q^-2
-----
0.899522*q^-1 + -0.700091*q^-2 + -0.799593*q^-3

-----

previous system was empty, adding identified system to AIS
lambdaAIS = -7.46274
details of the identified systemToStore:
1.50553*q^-1 + 9.99045*q^-2
-----
0.899522*q^-1 + -0.700091*q^-2 + -0.799593*q^-3

lambda=-7.46274 genoalue=3

-----
i=400 lambdaMin=0 lambda=-16.0319

Change in system, min value of lambda detected, i=401. lambdaMin=-16.0319

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-14.5714
system obtained, i=1
lambdaAb for antibody[2]=-4.85714
Attempting to obtain system from AIS (for lambdaMin=-16.0319):
system retrieved, systemAIS details:
1.50558*q^-1 + 9.99039*q^-2
-----
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3

i=401 resetting values
```

Gary Evans
System Identification using an Artificial Immune System

```
-----  
initial change in system detected, i = 500, systemRLS:  
0.494626*q^-1 + 0.537544*q^-2  
-----  
-0.467043*q^-1 + 0.384235*q^-2 + 0.0121983*q^-3  
  
-----  
i=500 lambdaMin=0 lambda=-7.46274  
  
Change in system, min value of lambda detected, i=501. lambdaMin=-7.46274  
  
lambdaAb for antibody[0]=0  
lambdaAb for antibody[1]=-14.5714  
lambdaAb for antibody[2]=-4.85714  
system obtained, i=2  
Attempting to obtain system from AIS (for lambdaMin=-7.46274):  
system retrieved, systemAIS details:  
1.50553*q^-1 + 9.99045*q^-2  
-----  
0.899522*q^-1 + -0.700091*q^-2 + -0.799593*q^-3  
  
i=501 resetting values  
  
-----  
initial change in system detected, i = 600, systemRLS:  
1.50553*q^-1 + 9.99045*q^-2  
-----  
0.899522*q^-1 + -0.700091*q^-2 + -0.799593*q^-3  
  
-----  
i=600 lambdaMin=0 lambda=-16.0319  
  
initial clock = 0  
final clock = 0.34
```

At the 100th iteration, system 1 is stored in the antigen along with a value of lambda of 0, the value of lambda associated with switching from no system (the initial condition) to system 1. The AIS is bred for 3 cycles after which the value of the antibody matches the antigen. At the 101st iteration an attempt is made to retrieve a system from the AIS for a value of lambda of -7.4565, no system has been saved in the AIS so no system can be retrieved, as expected.

At the 200th iteration, system 2 is stored along with the value of lambda associated with switching from system 1 to system 2 (-7.45656, obtained at iteration 101). The value stored in the antigen (the pheno value) is -7.28571 (a geno value of 3, or a string of '011'). At iteration 201, an attempt is made to retrieve a system from the AIS for a value of lambda of -16.0319, the value associated from switching from system 2 to system 1. This switch has not yet been encountered, so this should result in an empty system being returned from the AIS. However, as the antibody has not yet converged to value of lambda previously stored (-

7.45656), the AIS wrongly identifies that the switch has been encountered, and returns system 2, where the system encountered is system 1.

At the 300th iteration, further problems are encountered. The value of lambda associated with switching from system 2 to system 1 should be stored (-16.0319) along with system 1. However, as a system was wrongly retrieved from the AIS in iteration 201 the algorithm assumes that this condition has already been stored in the AIS, so no attempt is made now to do so. At iteration 301, an attempt is made to retrieve a system from the AIS for the switch from system 1 to system 2. This switch has been encountered at iteration 100, however, the value stored in the antibody still has not converged on the antigen, meaning that no system is retrieved from the AIS.

This causes more problems at the next switch, iteration 400, as no system was obtained from the AIS at the previous switch, system 2 is stored, along with the value of lambda associated with switching from system 1 to system 2 (-7.45656). However, this system was stored at iteration 200, so this is now stored twice in the AIS. The newly added one has a value of lambda in the antibody of -4.85714.

The problem is that there was not enough breeding of the strings to allow the antibodies to converge on the antigens. Also, the value of the repository and the number of clones was too small. With few clones, not much of the search space is explored in each iteration, giving a low probability of a better solution being found, which leads to poor convergence on the desired result.

Running the program again, with a larger number of clones and more breeding cycles (listed below) gives the expected results: the same results as were obtained in section 4.4, as can be seen below.

AIS parameters:

Gary Evans
System Identification using an Artificial Immune System

3 repository antibody
6 clones
nbest = 5
d=2
5 breeding cycles

initial change in system detected, i = 100, systemRLS:
 $0.5*q^{-1} + 0.500003*q^{-2}$

 $-0.499994*q^{-1} + 0.399994*q^{-2} + 3.13804e-006*q^{-3}$

previous system was empty, adding identified system to AIS
lambdaAIS = 0

details of the identified systemToStore:
 $0.5*q^{-1} + 0.500003*q^{-2}$

 $-0.499994*q^{-1} + 0.399994*q^{-2} + 3.13804e-006*q^{-3}$

lambda=0 genovalue=0

i=100 lambdaMin=0 lambda=-7.45656

Change in system, min value of lambda detected, i=101. lambdaMin=-7.45656

lambdaAb for antibody[0]=0

Attempting to obtain system from AIS (for lambdaMin=-7.45656):
empty: no system identified for new regime.

i=101 resetting values

initial change in system detected, i = 200, systemRLS:
 $1.50558*q^{-1} + 9.99039*q^{-2}$

 $0.899519*q^{-1} + -0.700091*q^{-2} + -0.799591*q^{-3}$

previous system was empty, adding identified system to AIS
lambdaAIS = -7.45656

details of the identified systemToStore:
 $1.50558*q^{-1} + 9.99039*q^{-2}$

 $0.899519*q^{-1} + -0.700091*q^{-2} + -0.799591*q^{-3}$

lambda=-7.45656 genovalue=3

i=200 lambdaMin=0 lambda=-16.0319

Change in system, min value of lambda detected, i=201. lambdaMin=-16.0319

lambdaAb for antibody[0]=0

lambdaAb for antibody[1]=-7.28571

Attempting to obtain system from AIS (for lambdaMin=-16.0319):
empty: no system identified for new regime.

i=201 resetting values

initial change in system detected, i = 300, systemRLS:
 $0.494626*q^{-1} + 0.537544*q^{-2}$

 $-0.467043*q^{-1} + 0.384235*q^{-2} + 0.0121983*q^{-3}$

previous system was empty, adding identified system to AIS
lambdaAIS = -16.0319

details of the identified systemToStore:
 $0.494626*q^{-1} + 0.537544*q^{-2}$

 $-0.467043*q^{-1} + 0.384235*q^{-2} + 0.0121983*q^{-3}$

Gary Evans

System Identification using an Artificial Immune System

```
lambda=-16.0319 genovalue=6

-----
i=300 lambdaMin=0 lambda=-7.46274

Change in system, min value of lambda detected, i=301. lambdaMin=-7.46274

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-7.28571
system obtained, i=1
lambdaAb for antibody[2]=-14.5714
Attempting to obtain system from AIS (for lambdaMin=-7.46274):
system retrieved, systemAIS details:
1.50558*q^-1 + 9.99039*q^-2
-----
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3

i=301 resetting values

-----
initial change in system detected, i = 400, systemRLS:
1.50553*q^-1 + 9.99045*q^-2
-----
0.899522*q^-1 + -0.700091*q^-2 + -0.799593*q^-3

-----
i=400 lambdaMin=0 lambda=-16.0319

Change in system, min value of lambda detected, i=401. lambdaMin=-16.0319

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-7.28571
lambdaAb for antibody[2]=-14.5714
system obtained, i=2
Attempting to obtain system from AIS (for lambdaMin=-16.0319):
system retrieved, systemAIS details:
0.494626*q^-1 + 0.537544*q^-2
-----
-0.467043*q^-1 + 0.384235*q^-2 + 0.0121983*q^-3

i=401 resetting values

-----
initial change in system detected, i = 500, systemRLS:
0.494626*q^-1 + 0.537544*q^-2
-----
-0.467043*q^-1 + 0.384235*q^-2 + 0.0121983*q^-3

-----
i=500 lambdaMin=0 lambda=-7.46274

Change in system, min value of lambda detected, i=501. lambdaMin=-7.46274

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-7.28571
system obtained, i=1
lambdaAb for antibody[2]=-14.5714
Attempting to obtain system from AIS (for lambdaMin=-7.46274):
system retrieved, systemAIS details:
1.50558*q^-1 + 9.99039*q^-2
-----
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3

i=501 resetting values

-----
initial change in system detected, i = 600, systemRLS:
1.50553*q^-1 + 9.99045*q^-2
-----
0.899522*q^-1 + -0.700091*q^-2 + -0.799593*q^-3

-----
i=600 lambdaMin=0 lambda=-16.0319

initial clock = 0
final clock = 0.35
```

It can be seen that the antibody values have converged on the antigen values when added to the AIS, so the results are as expected.

5.1.2 Conclusions

The AIS has been shown to work in encoding lambda with few bits, as long as, when adding a system to the AIS, it is bred for sufficient iterations with a large enough pool of repository and clone antibodies.

5.2 Use of the AIS to encode lambda for three systems

Now an AIS is used to identify three systems. From the table of minimum values of lambda for a covariance matrix trigger of 0.8, it can be seen that the range of values is approximately 17.0. The nearest values of lambda to be recognised are -3.33954 and -3.08578; this is a range of 0.25376. The resolution required is half this number, to obtain a resolution of 0.125 the maximum genovalue is (from equation 13) $17/0.125 = 136$, so 8 bits are required (8 bits gives a resolution of $17/255 = 0.066$). Now, the threshold for recognition of the correct value of lambda is taken to be 0.1.

From	To	Minimum lambda	Genovalue	Phenovalue	Binary String
system 1	system 2	-7.45656	126	-7.4000000	01111110
system 1	system 3	0.61227	5	0.666667	00000101
system 2	system 1	-15.9825	254	-15.933333	11111110
system 2	system 3	-3.33954	65	-3.3333333	01000001
system 3	system 1	-1.88064	43	-1.8666666	00101011
system 3	system 2	-3.08578	61	-3.0666666	00111101

Table 6. Phenovalues corresponding to minimum lambda for a matrix reset trigger of 0.8.

The systems presented to the AIS are as in section 4.6. That is, the order is

1 -> 2 -> 3 -> 1 -> 3 -> 2 -> 1 -> 2

These systems are presented to the AIS twice, this allows the AIS to first learn the information for each possible switch, and then it tests whether the AIS can recognise the new system.

The expected sequence of events is:

Iteration	Event
100	Add system 1 (following system 0) to the AIS, with a value of lambda of 0.0.
101	Unable to retrieve system from the AIS, for lambda = -7.45656
200	Add system 2 (following system 1) to the AIS, value of lambda of -7.45656
201	Unable to retrieve system from the AIS, for lambda = -3.33954
300	Add system 3 (following system 2) to the AIS, value of lambda of -3.33954
301	Unable to retrieve system from the AIS, for lambda = -1.88064
400	Add system 1 (following system 3) to the AIS, value of lambda of -1.88064
401	Unable to retrieve system from the AIS, for lambda = 0.612275
500	Add system 3 (following system 1) to the AIS, value of lambda of 0.612275
501	Unable to retrieve system from the AIS, for lambda = -3.08578
600	Add system 2 (following system 3) to the AIS, value of lambda of -3.08578
601	Unable to retrieve system from the AIS, for lambda = -15.9825
700	Add system 1 (following system 2) to the AIS, value of lambda of -15.9825
701	Retrieve system 2 (following system 1) from the AIS, lambda = -7.45656
800	Add no system
801	Retrieve system 3 (following system 2) from the AIS, lambda = -3.33954
900	Add no system
901	Retrieve system 1 (following system 3) from the AIS, lambda = -1.88064
1000	Add no system
1001	Retrieve system 3 (following system 1) from the AIS, lambda = 0.6667
1100	Add no system
1101	Retrieve system 2 (following system 3) from the AIS, lambda = -3.08578
1200	Add no system
1201	Retrieve system 1 (following system 2) from the AIS, lambda = -15.9825

Table 7. Expected results of running the program.

5.2.1 Results

The AIS was run with 8 bits for each binary string, with a *minPheno* of 1.0 and a *maxPheno* of -17.0, with a *threshold* of 0.1 and a *trigger* of 0.8 with the following AIS parameters:

3 repository antibody
 6 clones
nbest = 5
d=2
 5 breeding cycles

```

-----
initial change in system detected, i = 100, systemRLS:
0.5*q^-1 + 0.500003*q^-2
-----
-0.499994*q^-1 + 0.399994*q^-2 + 3.13804e-006*q^-3
-----

```

Gary Evans

System Identification using an Artificial Immune System

```
previous system was empty, adding identified system to AIS
lambdaAIS = 0
details of the identified systemToStore:

0.5*q^-1 + 0.500003*q^-2
-----
-0.499994*q^-1 + 0.399994*q^-2 + 3.13804e-006*q^-3

lambda=0 genovalue=15

-----
i=100 lambdaMin=1 lambda=-7.45656

Change in system, min value of lambda detected, i=101. lambdaMin=-7.45656

lambdaAb for antibody[0]=0
Attempting to obtain system from AIS (for lambdaMin=-7.45656):
empty: no system identified for new regime.
i=101 resetting values

-----
initial change in system detected, i = 201, systemRLS:

1.51415*q^-1 + 9.98637*q^-2
-----
0.899276*q^-1 + -0.699749*q^-2 + -0.800495*q^-3

-----
previous system was empty, adding identified system to AIS
lambdaAIS = -7.45656
details of the identified systemToStore:

1.51415*q^-1 + 9.98637*q^-2
-----
0.899276*q^-1 + -0.699749*q^-2 + -0.800495*q^-3

lambda=-7.45656 genovalue=126

-----
i=201 lambdaMin=1 lambda=-3.33954

Change in system, min value of lambda detected, i=202. lambdaMin=-3.33954

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-3.13333
Attempting to obtain system from AIS (for lambdaMin=-3.33954):
empty: no system identified for new regime.
i=202 resetting values

-----
initial change in system detected, i = 300, systemRLS:

0.695683*q^-1 + 2.00716*q^-2
-----
-0.198839*q^-1 + 0.199896*q^-2 + -0.700713*q^-3

-----
previous system was empty, adding identified system to AIS
lambdaAIS = -3.33954
details of the identified systemToStore:

0.695683*q^-1 + 2.00716*q^-2
-----
-0.198839*q^-1 + 0.199896*q^-2 + -0.700713*q^-3

lambda=-3.33954 genovalue=65

-----
i=300 lambdaMin=1 lambda=-1.88064

Change in system, min value of lambda detected, i=301. lambdaMin=-1.88064

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-3.13333
lambdaAb for antibody[2]=-3.33333
Attempting to obtain system from AIS (for lambdaMin=-1.88064):
empty: no system identified for new regime.
```

Gary Evans

System Identification using an Artificial Immune System

i=301 resetting values

initial change in system detected, i = 400, systemRLS:

0.499339*q⁻¹ + 0.505971*q⁻²

-0.494341*q⁻¹ + 0.397398*q⁻² + 0.0017605*q⁻³

previous system was empty, adding identified system to AIS

lambdaAIS = -1.88064

details of the identified systemToStore:

0.499339*q⁻¹ + 0.505971*q⁻²

-0.494341*q⁻¹ + 0.397398*q⁻² + 0.0017605*q⁻³

lambda=-1.88064 genovalue=43

i=400 lambdaMin=1 lambda=0.61227

Change in system, min value of lambda detected, i=401. lambdaMin=0.61227

lambdaAb for antibody[0]=0

lambdaAb for antibody[1]=-3.13333

lambdaAb for antibody[2]=-3.33333

lambdaAb for antibody[3]=-2.13333

Attempting to obtain system from AIS (for lambdaMin=0.61227):

empty: no system identified for new regime.

i=401 resetting values

initial change in system detected, i = 500, systemRLS:

0.700897*q⁻¹ + 1.99814*q⁻²

-0.20044*q⁻¹ + 0.200103*q⁻² + -0.699781*q⁻³

previous system was empty, adding identified system to AIS

lambdaAIS = 0.61227

details of the identified systemToStore:

0.700897*q⁻¹ + 1.99814*q⁻²

-0.20044*q⁻¹ + 0.200103*q⁻² + -0.699781*q⁻³

lambda=0.61227 genovalue=5

i=500 lambdaMin=1 lambda=-3.08578

Change in system, min value of lambda detected, i=501. lambdaMin=-3.08578

lambdaAb for antibody[0]=0

lambdaAb for antibody[1]=-3.13333

system obtained, i=1

lambdaAb for antibody[2]=-3.33333

lambdaAb for antibody[3]=-2.13333

lambdaAb for antibody[4]=-3.73333

Attempting to obtain system from AIS (for lambdaMin=-3.08578):

system retrieved, systemAIS details:

1.51415*q⁻¹ + 9.98637*q⁻²

0.899276*q⁻¹ + -0.699749*q⁻² + -0.800495*q⁻³

i=501 resetting values

initial change in system detected, i = 600, systemRLS:

1.50481*q⁻¹ + 9.99181*q⁻²

0.899599*q⁻¹ + -0.70008*q⁻² + -0.799657*q⁻³

Gary Evans
System Identification using an Artificial Immune System

```
-----  
i=600 lambdaMin=1 lambda=-15.9825  
  
Change in system, min value of lambda detected, i=601. lambdaMin=-15.9825  
  
lambdaAb for antibody[0]=0  
lambdaAb for antibody[1]=-3.13333  
lambdaAb for antibody[2]=-3.33333  
lambdaAb for antibody[3]=-2.13333  
lambdaAb for antibody[4]=-3.73333  
Attempting to obtain system from AIS (for lambdaMin=-15.9825):  
empty: no system identified for new regime.  
i=601 resetting values  
  
-----  
initial change in system detected, i = 700, systemRLS:  
  
0.494614*q^-1 + 0.537568*q^-2  
-----  
-0.467031*q^-1 + 0.384228*q^-2 + 0.0122032*q^-3  
  
-----  
previous system was empty, adding identified system to AIS  
lambdaAIS = -15.9825  
details of the identified systemToStore:  
  
0.494614*q^-1 + 0.537568*q^-2  
-----  
-0.467031*q^-1 + 0.384228*q^-2 + 0.0122032*q^-3  
  
lambda=-15.9825 genovalue=254  
  
-----  
i=700 lambdaMin=1 lambda=-7.46274  
  
Change in system, min value of lambda detected, i=701. lambdaMin=-7.46274  
  
lambdaAb for antibody[0]=0  
lambdaAb for antibody[1]=-3.13333  
lambdaAb for antibody[2]=-3.33333  
lambdaAb for antibody[3]=-2.13333  
lambdaAb for antibody[4]=0.666667  
lambdaAb for antibody[5]=-2.06667  
Attempting to obtain system from AIS (for lambdaMin=-7.46274):  
empty: no system identified for new regime.  
  
i=701 resetting values  
  
-----  
initial change in system detected, i = 801, systemRLS:  
1.51411*q^-1 + 9.98643*q^-2  
-----  
0.899278*q^-1 + -0.699748*q^-2 + -0.800497*q^-3  
  
-----  
previous system was empty, adding identified system to AIS  
lambdaAIS = -7.46274  
details of the identified systemToStore:  
  
1.51411*q^-1 + 9.98643*q^-2  
-----  
0.899278*q^-1 + -0.699748*q^-2 + -0.800497*q^-3  
  
lambda=-7.46274 genovalue=126  
  
-----  
i=801 lambdaMin=1 lambda=-3.33983  
  
Change in system, min value of lambda detected, i=802. lambdaMin=-3.33983  
  
lambdaAb for antibody[0]=0  
lambdaAb for antibody[1]=-3.13333  
lambdaAb for antibody[2]=-3.33333  
system obtained, i=2  
lambdaAb for antibody[3]=-2.13333  
lambdaAb for antibody[4]=0.666667
```

Gary Evans

System Identification using an Artificial Immune System

```
lambdaAb for antibody[5]=-2.06667
lambdaAb for antibody[6]=-7.33333
Attempting to obtain system from AIS (for lambdaMin=-3.33983):
system retrieved, systemAIS details:

0.695683*q^-1 + 2.00716*q^-2
-----
-0.198839*q^-1 + 0.199896*q^-2 + -0.700713*q^-3

i=802 resetting values

-----
initial change in system detected, i = 900, systemRLS:

0.695683*q^-1 + 2.00717*q^-2
-----
-0.198839*q^-1 + 0.199896*q^-2 + -0.700713*q^-3

-----
i=900 lambdaMin=1 lambda=-1.88064

Change in system, min value of lambda detected, i=901. lambdaMin=-1.88064

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-3.13333
lambdaAb for antibody[2]=-3.33333
lambdaAb for antibody[3]=-2.13333
lambdaAb for antibody[4]=0.666667
lambdaAb for antibody[5]=-2.06667
lambdaAb for antibody[6]=-7.33333
Attempting to obtain system from AIS (for lambdaMin=-1.88064):
empty: no system identified for new regime.
i=901 resetting values

-----
initial change in system detected, i = 1000, systemRLS:

0.499339*q^-1 + 0.505971*q^-2
-----
-0.494341*q^-1 + 0.397398*q^-2 + 0.0017605*q^-3

-----
previous system was empty, adding identified system to AIS
lambdaAIS = -1.88064
details of the identified systemToStore:

0.499339*q^-1 + 0.505971*q^-2
-----
-0.494341*q^-1 + 0.397398*q^-2 + 0.0017605*q^-3

lambda=-1.88064 genoalue=43

-----
i=1000 lambdaMin=1 lambda=0.61227

Change in system, min value of lambda detected, i=1001. lambdaMin=0.61227

lambdaAb for antibody[0]=0
lambdaAb for antibody[1]=-3.13333
lambdaAb for antibody[2]=-3.33333
lambdaAb for antibody[3]=-2.13333
lambdaAb for antibody[4]=0.666667
system obtained, i=4
lambdaAb for antibody[5]=-2.06667
lambdaAb for antibody[6]=-7.4
lambdaAb for antibody[7]=-4.33333
Attempting to obtain system from AIS (for lambdaMin=0.61227):
system retrieved, systemAIS details:
0.700897*q^-1 + 1.99814*q^-2
-----
-0.20044*q^-1 + 0.200103*q^-2 + -0.699781*q^-3

i=1001 resetting values

-----
initial change in system detected, i = 1100, systemRLS:
0.700897*q^-1 + 1.99814*q^-2
```

Gary Evans
System Identification using an Artificial Immune System

```
-----  
-0.20044*q^-1 + 0.200103*q^-2 + -0.699781*q^-3  
-----  
i=1100 lambdaMin=1 lambda=-3.08578  
  
Change in system, min value of lambda detected, i=1101. lambdaMin=-3.08578  
  
lambdaAb for antibody[0]=0  
lambdaAb for antibody[1]=-3.13333  
system obtained, i=1  
lambdaAb for antibody[2]=-3.33333  
lambdaAb for antibody[3]=-2.13333  
lambdaAb for antibody[4]=0.666667  
lambdaAb for antibody[5]=-2.06667  
lambdaAb for antibody[6]=-7.4  
lambdaAb for antibody[7]=-4.33333  
Attempting to obtain system from AIS (for lambdaMin=-3.08578):  
system retrieved, systemAIS details:  
  
1.51415*q^-1 + 9.98637*q^-2  
-----  
0.899276*q^-1 + -0.699749*q^-2 + -0.800495*q^-3  
  
i=1101 resetting values  
  
-----  
initial change in system detected, i = 1200, systemRLS:  
  
1.50481*q^-1 + 9.99181*q^-2  
-----  
0.899599*q^-1 + -0.70008*q^-2 + -0.799657*q^-3  
  
-----  
i=1200 lambdaMin=1 lambda=-15.9825  
  
Change in system, min value of lambda detected, i=1201. lambdaMin=-15.9825  
  
lambdaAb for antibody[0]=0  
lambdaAb for antibody[1]=-3.13333  
lambdaAb for antibody[2]=-3.33333  
lambdaAb for antibody[3]=-2.13333  
lambdaAb for antibody[4]=0.666667  
lambdaAb for antibody[5]=-2.06667  
lambdaAb for antibody[6]=-7.4  
lambdaAb for antibody[7]=-4.33333  
Attempting to obtain system from AIS (for lambdaMin=-15.9825):  
empty: no system identified for new regime.  
i=1201 resetting values  
  
-----  
initial change in system detected, i = 1300, systemRLS:  
  
0.494614*q^-1 + 0.537568*q^-2  
-----  
-0.467031*q^-1 + 0.384228*q^-2 + 0.0122032*q^-3  
  
-----  
previous system was empty, adding identified system to AIS  
lambdaAIS = -15.9825  
details of the identified systemToStore:  
  
0.494614*q^-1 + 0.537568*q^-2  
-----  
-0.467031*q^-1 + 0.384228*q^-2 + 0.0122032*q^-3  
  
lambda=-15.9825 genovalue=254  
  
-----  
i=1300 lambdaMin=1 lambda=-7.46274  
  
Change in system, min value of lambda detected, i=1301. lambdaMin=-7.46274  
  
lambdaAb for antibody[0]=0  
lambdaAb for antibody[1]=-7.4  
system obtained, i=1  
lambdaAb for antibody[2]=-3.33333
```

Gary Evans
System Identification using an Artificial Immune System

```
lambdaAb for antibody[3]=-2.13333
lambdaAb for antibody[4]=0.666667
lambdaAb for antibody[5]=-2.06667
lambdaAb for antibody[6]=-7.4
system obtained, i=6
lambdaAb for antibody[7]=-2.13333
lambdaAb for antibody[8]=-15.9333
Attempting to obtain system from AIS (for lambdaMin=-7.46274):
system retrieved, systemAIS details:
1.51411*q^-1 + 9.98643*q^-2
-----
0.899278*q^-1 + -0.699748*q^-2 + -0.800497*q^-3

i=1301 resetting values
```

The results are as expected (from Table 7) for the first 500 iterations. However, the value of lambda added to the AIS for system 2, at iteration 200, is -3.133 rather than -7.4, and the correct value has not been converged on by iteration 500, meaning that system 2 is incorrectly retrieved instead of system 3. This leads to further errors; as a system was incorrectly retrieved, no system is added to the AIS at the 600th iteration. Iterations 601 and 700 were correct. However, at iteration 701, lambda for system 2 still hasn't converged (this caused an incorrect system to be retrieved at iteration 500). Now, this means that system 2 cannot be retrieved. This means that system 3 (following 2) is added to the AIS again at iteration 800.

At iteration 801, system 3 (following 2) is correctly retrieved. At iteration 901, system 1 should be retrieved from the AIS. However, the pheno value in the antibody has not yet converged - the value of lambda is -2.13333 where it should be -1.86666. This means that system 1 (following 3) is again added to the AIS. The correct system is retrieved at iteration 1001.

This work shows that the AIS can be used to recognise three systems, but that there are problems with the antibodies not converging on the antigen. This is as expected, the same AIS parameters were used as with 2 systems, when more should be used - with more antibodies there is less of a chance of each one being selected for improvement in each breeding cycle.

5.3 Genovalue Based Affinity for Three Systems

Apart from the AIS parameters having an effect on the convergence of the antibodies, there is another factor; the antigen values for system 2 following on from system 1, and system 1 following on from system 3, didn't converge on their solution throughout, but remained on the same value. This is examined in Table 8. The phenovalues (and genovalues) converged upon are not near to the desired value. However, the binary strings for the genovalues converged upon are only 1 bit different from the binary strings for the required genovalues. The only possible improvement in the value of the antibody is for the correct string to be found in the AIS's breeding search of results, there is no nearer string which will replace the antibody other than the correct one. One bit difference gives the same affinity, but the genovalue and phenovalue discrepancy is dependent on that bit's position in the binary string.

Description	System	Phenovalue	GenoValue	String
Desired (correct) value	2 (from 1)	-7.4	126	01111110
Actual value (converged to)	""	-3.13333	62	00111110
Desired (correct) value	1 (from 3)	-1.86666	43	00101011
Actual value (converged to)	""	-2.13333	47	00101111

Table 8. Table examining convergence of antibodies towards antigens.

Gray coding may provide one solution for this. Another solution is for the fitness used in the tournament selection of parents in the AIS's breeding cycle to take the fittest solution as the one with the closest genovalue, rather than the one with the largest number of matching bits.

The AIS was run with the affinity based on the genovalue, rather than the number of matching bits (as just described). The breeding parameters were as in the previous section, apart from the number of generations that breeding took place for (breeding taking place each time that an antibody is added to the AIS). The number of generations was increased from 10 by intervals of size 10. The required results, as shown in Table 7 are achieved with

a value of *ngen* of 50. These results were achieved by running the program with a seed value of 1031758903.

5.4 Use of the AIS to Encode the System Parameters

Section 5.1 investigates the use of an AIS to retrieve systems, identifying them based on the value of λ . However, as seen in section 4.5, λ is only suitable in certain conditions, specifically the value of λ is dependent upon changing the system at the same iteration as the reference changes, when the reference changes from -1 to +1. To allow the system to be changed at any point in time, the actual system parameters may be encoded and stored in the antibody strings.

5.4.1 Encoding the parameters

The system parameters may be encoded as a binary string, this allows them to be easily integrated into the artificial immune system. The string is created by calculating the string for each individual parameter, as in section 5.1, and concatenating these strings together.

After a change of system is detected, the system parameters are found for a small number of time instances, and that newly identified system is used to try and retrieve a more accurately identified system from the AIS. This gives the advantage that a system can be accurately identified with many observations, and that system can be quickly retrieved using few observations. Now, rather than waiting for the minimum value of λ to be encountered to identify the system, as in section 5.2, an attempt is made to retrieve a system from the antibody after a given number of observations, *numobs*. The algorithm is as follows:

```

systemChangeDetected = False
systemChangeIteration = 0
FOR i=1 TO number of iterations
  Perform RLS algorithm
  IF lambda < 0    (indicates that change of lambda has been detected)
    IF systemChangedDetected = False
      systemToStore = systemRLS
      (if previous system had no system retrieved for that value of lambda)
      IF(systemAIS is empty)
        store systemAIS in the AIS
      END IF
      systemChangeDetected = True
      systemChangedIteration = i
    END IF
  END IF

  (check for covariance matrix reset)
  IF lambda < 0.0
    reset the covariance matrix
  END IF

  IF systemChangeDetected = True
    IF ((i-systemChangedIteration) = numObsIterations)

      systemAIS = retrieved system from AIS with corresponding lambdaAIS

      (reset variables for detection of next change of system)
      systemChangeDetected = False
    END IF
  END IF
  Update system parameters (systemRLS)
END FOR

```

The coefficients of the three systems are:

System	Coefficient				
	a ₁	a ₂	a ₃	b ₁	b ₂
1	-0.5	0.4	0.0	0.5	0.5
2	0.9	-0.7	-0.8	1.5	10.0
3	-0.2	0.2	0.7	0.7	2.0

Table 9 The coefficients for the three systems.

The denominator coefficients range between -0.8 and 0.9, so values of *minDenom* and *maxDenom* of -1.0 and 1.0 were chosen. The values of the numerators range between -0.8

and 10.0, so values of *minNumer* and *maxNumer* of -1.0 and 11.0 were chosen. The difference between the nearest values for the denominator coefficients is 0.2 (values of 0.2 and 0.4 for a_2), a resolution of 0.1 was chosen requiring 20 ($2.0/0.1$) graduations in the phenovalue, this is achievable with 5 bits per denominator coefficient. The difference between the nearest values of the numerator coefficients is 0.2 (0.5 and 0.7 for b_1); again, a resolution of 0.1 was chosen requiring 120 ($12/0.1$) graduations in the phenovalue. This is achieved with 7 bits per numerator coefficient.

As seen in the previous section, a good affinity match shouldn't rely on the number of bits in the string being in the same position, as 1 bit difference can give an error as large as 2^{N-1} (where N is the string length). Some other measure needs to be made of how well the antibody and antigen match, i.e. how close the system parameters are to each other. Using the sum of the absolute differences of each parameter for a measure of the match, as shown in Table 9, gives the measure of how near to each system is to each other, as given in Table 10.

Systems to compare		Value of difference
system 1	system 2	13.8
system 1	system 3	2.9
system 2	system 3	12.3

Table 10 The difference between the systems.

5.4.2 Results of Testing the Algorithm

The above algorithm was investigated by adding systems to the AIS, and searching through the antigens for a good match. Doing this, the antibodies are ignored, and the AIS acts as a simple library of antigens, as in Chapter 3. This is just to test the feasibility of the algorithm without worrying about breeding the AIS. Also, the algorithm was only tested for system 1 and system 2, with the same reference as in Chapter 3. A *threshold* of 2.5 and a *trigger* of 0.8 were chosen. A value of *numobs* of 50 was chosen to allow a good estimate to be made of the system being entered. Running the program (`sysID14sysParams.cpp`, documented in appendix 8.6) with a seed value of 1032362196 gives the following results:

Gary Evans
System Identification using an Artificial Immune System

```
-----  
initial change in system detected, i = 100, systemRLS:  
0.5*q^-1 + 0.500003*q^-2  
-----  
-0.499994*q^-1 + 0.399994*q^-2 + 3.13804e-006*q^-3  
-----  
  
previous system was empty, adding identified system to AIS  
lambdaAIS = 0  
details of the identified systemToStore:  
  
0.5*q^-1 + 0.500003*q^-2  
-----  
-0.499994*q^-1 + 0.399994*q^-2 + 3.13804e-006*q^-3  
  
Change in system after number of observations, i=150. lambdaMin=0  
toIdentify:  
  
1.51113*q^-1 + 9.98051*q^-2  
-----  
0.899022*q^-1 + -0.700147*q^-2 + -0.799209*q^-3  
  
numAntigens=1  
toIdentify:  
00110101110100001110000100000  
currentMatch[0]=13.8618  
currentMatch string:  
00011110001111000010010100011  
no match found  
Attempting to obtain system from AIS (for lambdaMin=0):  
empty: no system identified for new regime.  
i=150 resetting values  
  
-----  
initial change in system detected, i = 200, systemRLS:  
1.50558*q^-1 + 9.99039*q^-2  
-----  
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3  
-----  
  
previous system was empty, adding identified system to AIS  
lambdaAIS = 0  
details of the identified systemToStore:  
  
1.50558*q^-1 + 9.99039*q^-2  
-----  
0.899519*q^-1 + -0.700091*q^-2 + -0.799591*q^-3  
  
Change in system after number of observations, i=250. lambdaMin=1  
toIdentify:  
  
0.488075*q^-1 + 0.593114*q^-2  
-----  
-0.417431*q^-1 + 0.360665*q^-2 + 0.0305621*q^-3  
  
numAntigens=2  
toIdentify:  
00011110010000000100010100011  
currentMatch[0]=0.89776  
currentMatch string:  
00011110001111000010010100011  
currentMatch[1]=13.6932  
currentMatch string:  
00110101110100001110000100000  
stringLength =29  
best match index=0 best match =0.89776  
to return:  
Attempting to obtain system from AIS (for lambdaMin=1):  
system retrieved, systemAIS details:  
0.417323*q^-1 + 0.417323*q^-2  
-----  
-0.741935*q^-1 + 0.290323*q^-2 + -0.225806*q^-3  
  
i=250 resetting values
```

Gary Evans
System Identification using an Artificial Immune System

```
-----  
initial change in system detected, i = 300, systemRLS:  
0.494626*q^-1 + 0.537544*q^-2  
-----  
-0.467043*q^-1 + 0.384235*q^-2 + 0.0121983*q^-3  
  
Change in system after number of observations, i=350. lambdaMin=1  
toIdentify:  
  
1.51105*q^-1 + 9.98063*q^-2  
-----  
0.899027*q^-1 + -0.700146*q^-2 + -0.799212*q^-3  
  
numAntigens=2  
toIdentify:  
00110101110100001110000100000  
currentMatch[0]=13.8619  
currentMatch string:  
00011110001111000010010100011  
currentMatch[1]=0.409511  
currentMatch string:  
00110101110100001110000100000  
stringLength =29  
best match index=1 best match =0.409511  
to return:  
Attempting to obtain system from AIS (for lambdaMin=1):  
system retrieved, systemAIS details:  
  
1.45669*q^-1 + 9.96063*q^-2  
-----  
0.806452*q^-1 + -0.741935*q^-2 + -1*q^-3  
  
i=350 resetting values  
  
-----  
initial change in system detected, i = 400, systemRLS:  
  
1.50553*q^-1 + 9.99045*q^-2  
-----  
0.899522*q^-1 + -0.700091*q^-2 + -0.799593*q^-3  
  
Change in system after number of observations, i=450. lambdaMin=1  
toIdentify:  
  
0.488075*q^-1 + 0.593114*q^-2  
-----  
-0.41743*q^-1 + 0.360665*q^-2 + 0.0305623*q^-3  
  
numAntigens=2  
toIdentify:  
00011110010000000100010100011  
currentMatch[0]=0.89776  
currentMatch string:  
00011110001111000010010100011  
currentMatch[1]=13.6932  
currentMatch string:  
00110101110100001110000100000  
stringLength =29  
best match index=0 best match =0.89776  
to return:  
Attempting to obtain system from AIS (for lambdaMin=1):  
system retrieved, systemAIS details:  
  
0.417323*q^-1 + 0.417323*q^-2  
-----  
-0.741935*q^-1 + 0.290323*q^-2 + -0.225806*q^-3  
  
i=450 resetting values  
  
-----  
initial change in system detected, i = 500, systemRLS:  
  
0.494626*q^-1 + 0.537544*q^-2  
-----  
-0.467043*q^-1 + 0.384235*q^-2 + 0.0121983*q^-3
```

Gary Evans System Identification using an Artificial Immune System

Change in system after number of observations, $i=550$. $\lambda_{\min}=1$
toIdentify:

$$\frac{1.51105q^{-1} + 9.98063q^{-2}}{\text{-----}} \\ 0.899027q^{-1} + -0.700146q^{-2} + -0.799212q^{-3}$$

```
numAntigens=2
toIdentify:
00110101110100001110000100000
currentMatch[0]=13.8619
currentMatch string:
00011110001111000010010100011
currentMatch[1]=0.409511
currentMatch string:
00110101110100001110000100000
stringLength =29
best match index=1 best match =0.409511
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
system retrieved, systemAIS details:
```

$$\frac{1.45669q^{-1} + 9.96063q^{-2}}{\text{-----}} \\ 0.806452q^{-1} + -0.741935q^{-2} + -1q^{-3}$$

$i=550$ resetting values

initial change in system detected, $i = 600$, systemRLS:

$$\frac{1.50553q^{-1} + 9.99045q^{-2}}{\text{-----}} \\ 0.899522q^{-1} + -0.700091q^{-2} + -0.799593q^{-3}$$

initial time =1032370996

At iteration 100 the change in system is detected, and as the previous system was empty, system 1 is added to the AIS. The system was correctly identified to 5 decimal place. After 50 iterations (*numobs*) of identifying the new regime being entered (System 2), an attempt is made to retrieve a system from the AIS. No system is present for system 2 so the attempt fails.

A change of system is identified at iteration 200. The previous systemAIS was empty, so system 2 is added to the library of antigens. After a further 50 iterations, an attempt is made to retrieve a system from the AIS. A match is found, system 1 is retrieved from the AIS. System 1 was added to the AIS accurate to 5 decimal places in iteration 100. However, the error due to encoding the system gives the returned system as:

$$y(t) = \frac{(0.417323^{-1} + 0.417323q^{-2})u(t)}{-0.417431q^{-1} + 0.360665q^{-2} - 0.0305621q^{-3}}$$

Also, after 50 iterations, system 1 has only been accurately identified to 1 decimal place. There are two sources of error, which may cause problems when the systems are nearer (which is the case when system 3 is also being identified). The first source of error is due to the encoding of the systems; using more bits to encode the strings will alleviate this error. The second problem is the inaccuracy of identifying the system. Providing more information to the system (with a greater change of the reference by using a PRBS, or with noise on the input), as seen in section 4.4) may improve this error. A total reset of the RLS algorithm (resetting the parameter vectors etc.) may also aid this.

At iteration 300, the next change in system regime is detected. A system was retrieved at iteration 250, so no system is added to the AIS. At iteration 350, system 2 is identified and retrieved from the algorithm. The rest of the system changes proceed as expected.

This has shown that encoding the system parameters can be used to add and retrieve systems from a library of antigens. This should also be possible for a library of antibodies, which have been bred to match these antigens, as was investigated in the previous section.

Now use is made of an AIS, breeding a set of antibodies and checking them against the antigen, as before. The expected results of using an AIS are shown in Table 11. Whereas before, a system was stored in the system multiple times, for each value of lambda associated with it, now it is only necessary for a system to be stored once. Previously, the AIS only underwent breeding when adding a system, but now, as systems are only added to the AIS three times, breeding also takes place when attempting to retrieve a system.

Iteration	Description
100	Add system 1 to the AIS
125	Attempt to retrieve system 2 from the AIS, with no success
200	Add system 2 to the AIS
225	Attempt to retrieve system 3 from the AIS, with no success
300	Add system 3 to the AIS
325	Retrieve system 1 from the AIS
400	No system added to the AIS
425	Retrieve system 3 from the AIS
500	No system added to the AIS
525	Retrieve system 2 from the AIS
600	No system added to the AIS
625	Retrieve system 1 from the AIS

Table 11. Expected Results of Using an AIS to Identify Systems Based on their Parameters.

5.4.3 Results of Using the AIS and Encoding the System Parameters

The algorithm was then investigated with the three systems, presented as earlier. 9 bits were used per numerator, and 7 bits were used per denominator. The reference signal was a PRBS, *numobs* was given the value of 25, and a threshold of 2.0 was used. The AIS parameters were

AIS parameters:
 5 repository antibody
 20 clones
 $nbest = 5$
 $d=2$

The program was run with a seed value of 1032362196 with *ngen* being incremented from 0 in steps of 200. The antibodies were not converging on the antigens by the 1600th iteration, this is due to the number of bits and the number of systems used. The same problem occurs as was discussed in Section 5.3. An affinity measure of the number of matching bits is not sufficient for the AIS to breed, as was previously discussed. Some measure of the affinity needs to be made depending on how well the antibody and antigen system parameters match.

Gary Evans
System Identification using an Artificial Immune System

This change was made, and the AIS parameters were as previously. The antibodies were found to converge on the antigens with a value of *ngen* of 50, giving the expected results.

The results were as follows:

```
-----
initial change in system detected, i = 100, systemRLS:
0.5*q^-1 + 0.500001*q^-2
-----
-0.499999*q^-1 + 0.399999*q^-2 + 7.55492e-007*q^-3
-----

previous system was empty, adding identified system to AIS
lambdaAIS = 0
details of the identified systemToStore:

0.5*q^-1 + 0.500001*q^-2
-----
-0.499999*q^-1 + 0.399999*q^-2 + 7.55492e-007*q^-3

newAntigenPop[0]=
000111111000111111000011100101100001111
in AIS::addSystem(), stringLength=39

Change in system after number of observations, i=125. lambdaMin=0
toIdentify:

1.50799*q^-1 + 9.98253*q^-2
-----
0.900006*q^-1 + -0.700231*q^-2 + -0.800218*q^-3

numAntigens=1
toIdentify:
001101010111010011001111000001000000011
currentMatch[0]=14.3642
currentMatch string:
0001111110001110000000011100101000010111
0.479452*q^-1 + 0.315068*q^-2
-----
-0.559055*q^-1 + 0.259843*q^-2 + 0.448819*q^-3

no match found
Attempting to obtain system from AIS (for lambdaMin=0):
empty: no system identified for new regime.
i=125 resetting values

-----
initial change in system detected, i = 200, systemRLS:
1.49989*q^-1 + 9.99662*q^-2
-----
0.899974*q^-1 + -0.700059*q^-2 + -0.800035*q^-3
-----

previous system was empty, adding identified system to AIS
lambdaAIS = 0
details of the identified systemToStore:

1.49989*q^-1 + 9.99662*q^-2
-----
0.899974*q^-1 + -0.700059*q^-2 + -0.800035*q^-3

newAntigenPop[1]=
001101010111010100001111000001000000011
in AIS::addSystem(), stringLength=39

Change in system after number of observations, i=225. lambdaMin=1
toIdentify:

0.701938*q^-1 + 2.00916*q^-2
-----
```

Gary Evans
System Identification using an Artificial Immune System

```

-0.199449*q^-1 + 0.200329*q^-2 + -0.699588*q^-3

numAntigens=2
toIdentify:
001001000010000000000110000100110000100
currentMatch[0]=3.07521
currentMatch string:
000111101000111110000010100101010001111
0.432485*q^-1 + 0.455969*q^-2
-----
-0.685039*q^-1 + 0.322835*q^-2 + -0.0551181*q^-3

currentMatch[1]=10.7202
currentMatch string:
0011100011101111100100001010000111
1.65362*q^-1 + 9.47358*q^-2
-----
1.07874*q^-1 + -0.685039*q^-2 + -0.559055*q^-3

no match found
Attempting to obtain system from AIS (for lambdaMin=1):
empty: no system identified for new regime.
i=225 resetting values

-----
initial change in system detected, i = 300, systemRLS:

0.700424*q^-1 + 2.00249*q^-2
-----
-0.199859*q^-1 + 0.200078*q^-2 + -0.6999*q^-3

-----

previous system was empty, adding identified system to AIS
lambdaAIS = 1
details of the identified systemToStore:

0.700424*q^-1 + 2.00249*q^-2
-----
-0.199859*q^-1 + 0.200078*q^-2 + -0.6999*q^-3

newAntigenPop[2]=
001001000001111111000110000100110000100
in AIS::addSystem(), stringLength=39

Change in system after number of observations, i=325. lambdaMin=1
toIdentify:

0.501367*q^-1 + 0.512131*q^-2
-----
-0.485024*q^-1 + 0.387998*q^-2 + 0.00824623*q^-3

numAntigens=3
toIdentify:
00011111001000000000100000101100010000
currentMatch[0]=0.453587
currentMatch string:
000111101000111110000010100101010001111
0.432485*q^-1 + 0.455969*q^-2
-----
-0.685039*q^-1 + 0.322835*q^-2 + -0.0551181*q^-3

currentMatch[1]=13.3178
currentMatch string:
001110001110111110010000100001010000111
1.65362*q^-1 + 9.47358*q^-2
-----
1.07874*q^-1 + -0.685039*q^-2 + -0.559055*q^-3

currentMatch[2]=2.91173
currentMatch string:
000111101010010110000010100101010000111
0.432485*q^-1 + 2.5225*q^-2
-----
-0.685039*q^-1 + 0.322835*q^-2 + -0.559055*q^-3

stringLength =39

```

Gary Evans

System Identification using an Artificial Immune System

```
best match index=0 best match =0.453587
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
system retrieved, systemAIS details:

0.479452*q^-1 + 0.479452*q^-2
-----
-0.559055*q^-1 + 0.385827*q^-2 + -0.0551181*q^-3

i=325 resetting values

-----
initial change in system detected, i = 401, systemRLS:

0.520753*q^-1 + 0.470822*q^-2
-----
-0.510632*q^-1 + 0.377596*q^-2 + -0.00430873*q^-3

Change in system after number of observations, i=426. lambdaMin=1
toIdentify:

0.700712*q^-1 + 1.99607*q^-2
-----
-0.200675*q^-1 + 0.200529*q^-2 + -0.70028*q^-3

numAntigens=3
toIdentify:
001001000001111111000110000100110000100
currentMatch[0]=3.06016
currentMatch string:
0001111010001111110000010100101010001111
0.432485*q^-1 + 0.455969*q^-2
-----
-0.685039*q^-1 + 0.322835*q^-2 + -0.0551181*q^-3

currentMatch[1]=10.7366
currentMatch string:
001110001110111110010000100001010000111
1.65362*q^-1 + 9.47358*q^-2
-----
1.07874*q^-1 + -0.685039*q^-2 + -0.559055*q^-3

currentMatch[2]=0.865206
currentMatch string:
000111101010010010000110100100110000111
0.432485*q^-1 + 2.42857*q^-2
-----
-0.181102*q^-1 + 0.19685*q^-2 + -0.559055*q^-3

stringLength =39
best match index=2 best match =0.865206
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
system retrieved, systemAIS details:

0.690802*q^-1 + 1.98239*q^-2
-----
-0.244094*q^-1 + 0.19685*q^-2 + -0.748031*q^-3

i=426 resetting values

-----
initial change in system detected, i = 500, systemRLS:

0.699941*q^-1 + 1.99922*q^-2
-----
-0.200114*q^-1 + 0.200049*q^-2 + -0.700025*q^-3

Change in system after number of observations, i=525. lambdaMin=1
toIdentify:

1.50663*q^-1 + 9.98208*q^-2
-----
0.899484*q^-1 + -0.700396*q^-2 + -0.799526*q^-3

numAntigens=3
toIdentify:
```

Gary Evans
System Identification using an Artificial Immune System

```
001101010111010011001111000001000000011
currentMatch[0]=13.9524
currentMatch string:
000111101000111110000010100101010001111
0.432485*q^-1 + 0.455969*q^-2
-----
-0.685039*q^-1 + 0.322835*q^-2 + -0.0551181*q^-3

currentMatch[1]=1.09057
currentMatch string:
001110001110111110010000100001010000111
1.65362*q^-1 + 9.47358*q^-2
-----
1.07874*q^-1 + -0.685039*q^-2 + -0.559055*q^-3

currentMatch[2]=10.846
currentMatch string:
000111101010010010000110100100110000111
0.432485*q^-1 + 2.42857*q^-2
-----
-0.181102*q^-1 + 0.19685*q^-2 + -0.559055*q^-3

stringLength =39
best match index=1 best match =1.09057
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
system retrieved, systemAIS details:

1.48924*q^-1 + 9.99022*q^-2
-----
0.889764*q^-1 + -0.748031*q^-2 + -0.811024*q^-3

i=525 resetting values

-----
initial change in system detected, i = 600, systemRLS:
1.50009*q^-1 + 9.99649*q^-2
-----
0.899897*q^-1 + -0.700067*q^-2 + -0.799945*q^-3

Change in system after number of observations, i=625. lambdaMin=1
toIdentify:

0.509989*q^-1 + 0.53776*q^-2
-----
-0.452681*q^-1 + 0.37585*q^-2 + 0.0188175*q^-3

numAntigens=3
toIdentify:
001000000001000001000100000101010010000
currentMatch[0]=0.518605
currentMatch string:
000111101000111110000010100101010001111
0.432485*q^-1 + 0.455969*q^-2
-----
-0.685039*q^-1 + 0.322835*q^-2 + -0.0551181*q^-3

currentMatch[1]=14.0727
currentMatch string:
001110010111011000010000100001110000010
1.6771*q^-1 + 10.0841*q^-2
-----
1.07874*q^-1 + -0.559055*q^-2 + -0.874016*q^-3

currentMatch[2]=2.55804
currentMatch string:
000111101010000010000110100101010000110
0.432485*q^-1 + 2.05284*q^-2
-----
-0.181102*q^-1 + 0.322835*q^-2 + -0.622047*q^-3

stringLength =39
best match index=0 best match =0.518605
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
```

```
system retrieved, systemAIS details:  
0.479452*q^-1 + 0.479452*q^-2  
-----  
-0.559055*q^-1 + 0.385827*q^-2 + -0.0551181*q^-3  
  
i=625 resetting values
```

The systems saved and retrieved in the AIS were found to be accurate to 1 decimal place. This may be improved by using more bits, but this would require more breeding of the AIS. However, this has shown that the AIS can identify a system based on its system parameters, and that this leads to less redundancy in the AIS.

5.5 Noise on the Results

This section investigates the results of adding noise onto the output. With white noise (noise with no bias, i.e. noise with a zero mean value), the RLS algorithm should function as before. Figure 20 shows the effect of noise on the parameter estimation. The estimation of the parameters improves with the number of observations.

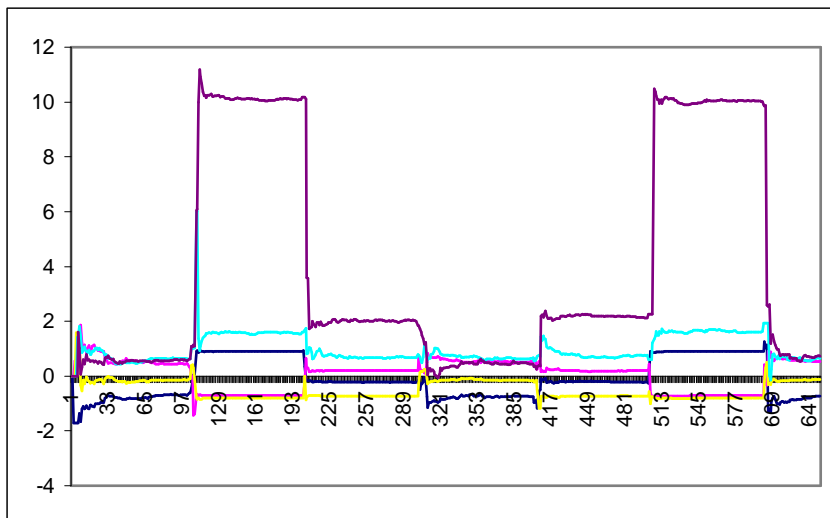


Figure 20. The Effect of Noise on the Parameter Estimation.

This is investigated by running the program with 10 bits per numerator coefficient, and 8 bits per denominator coefficient. The following AIS parameters were used

AIS parameters:
10 repository antibody
20 clones
 $n_{best} = 8$
 $d=3$

Gary Evans
System Identification using an Artificial Immune System

Running the program with a seed value of 1032362196, with a value of *numobs* of 25, a *ngen* of 50 and a *threshold* of 2.5 gives the following results:

```
-----
initial change in system detected, i = 104, systemRLS:
0.625056*q^-1 + 0.616193*q^-2
-----
-0.61659*q^-1 + 0.341225*q^-2 + -0.111445*q^-3
-----

previous system was empty, adding identified system to AIS
lambdaAIS = 0
details of the identified systemToStore:

0.625056*q^-1 + 0.616193*q^-2
-----
-0.61659*q^-1 + 0.341225*q^-2 + -0.111445*q^-3

newAntigenPop[0]=
001000101000100010001000011000010101000011100
in AIS::addSystem(), stringLength=44

Change in system after number of observations, i=129. lambdaMin=0
toIdentify:

1.5306*q^-1 + 10.245*q^-2
-----
0.883213*q^-1 + -0.711359*q^-2 + -0.799346*q^-3

numAntigens=1
toIdentify:
00110101111110111110001111000000100100000110
currentMatch[0]=13.5273
currentMatch string:
00011010110010010100001000010010111100011101
0.255132*q^-1 + 0.73607*q^-2
-----
0.0352941*q^-1 + 0.47451*q^-2 + -0.0901961*q^-3

no match found
Attempting to obtain system from AIS (for lambdaMin=0):
empty: no system identified for new regime.
i=129 resetting values

-----
initial change in system detected, i = 200, systemRLS:
1.53415*q^-1 + 10.085*q^-2
-----
0.887706*q^-1 + -0.714181*q^-2 + -0.803922*q^-3
-----

previous system was empty, adding identified system to AIS
lambdaAIS = 0
details of the identified systemToStore:
1.53415*q^-1 + 10.085*q^-2
-----
0.887706*q^-1 + -0.714181*q^-2 + -0.803922*q^-3

newAntigenPop[1]=
00110110001110110000001111000000100100000110
in AIS::addSystem(), stringLength=44

Change in system after number of observations, i=225. lambdaMin=1
toIdentify:

0.792132*q^-1 + 1.93645*q^-2
-----
-0.208267*q^-1 + 0.18244*q^-2 + -0.717061*q^-3

numAntigens=2
toIdentify:
00100110000011111010000110010010010100001001
currentMatch[0]=2.89987
currentMatch string:
```

Gary Evans

System Identification using an Artificial Immune System

```
00011010110010010100001000010010111100011101
0.255132*q^-1 + 0.73607*q^-2
-----
0.0352941*q^-1 + 0.47451*q^-2 + -0.0901961*q^-3

currentMatch[1]=11.2659
currentMatch string:
01000001111110000000010000100000101000000001
2.08504*q^-1 + 9.51026*q^-2
-----
1.07059*q^-1 + -0.686275*q^-2 + -0.968627*q^-3

no match found
Attempting to obtain system from AIS (for lambdaMin=1):
empty: no system identified for new regime.
i=225 resetting values

-----
initial change in system detected, i = 300, systemRLS:

0.663124*q^-1 + 2.00618*q^-2
-----
-0.22791*q^-1 + 0.183383*q^-2 + -0.728673*q^-3

-----
previous system was empty, adding identified system to AIS
lambdaAIS = 1
details of the identified systemToStore:

0.663124*q^-1 + 2.00618*q^-2
-----
-0.22791*q^-1 + 0.183383*q^-2 + -0.728673*q^-3

newAntigenPop[2]=
0010001101010000000000110000010010100001000
in AIS::addSystem(), stringLength=44

Change in system after number of observations, i=325. lambdaMin=1
toIdentify:

0.759042*q^-1 + 0.283714*q^-2
-----
-0.838568*q^-1 + 0.611674*q^-2 + -0.149177*q^-3

numAntigens=3
toIdentify:
00100101010001101101000001010011001100011011
currentMatch[0]=0.882587
currentMatch string:
00100011010010000100000010010010101000011000
0.653959*q^-1 + 0.548387*q^-2
-----
-0.717647*q^-1 + 0.317647*q^-2 + -0.247059*q^-3

currentMatch[1]=14.5791
currentMatch string:
01000001111110000000010000100000101000000001
2.08504*q^-1 + 9.51026*q^-2
-----
1.07059*q^-1 + -0.686275*q^-2 + -0.968627*q^-3

currentMatch[2]=3.13699
currentMatch string:
00100011010100000100000010010010001000001000
0.653959*q^-1 + 2.04985*q^-2
-----
-0.717647*q^-1 + 0.0666667*q^-2 + -0.74902*q^-3

stringLength =44
best match index=0 best match =0.882587
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
system retrieved, systemAIS details:

0.618768*q^-1 + 0.607038*q^-2
```

Gary Evans

System Identification using an Artificial Immune System

```
-----
-0.623529*q^-1 + 0.317647*q^-2 + -0.121569*q^-3

i=325 resetting values

-----
initial change in system detected, i = 403, systemRLS:
0.552707*q^-1 + 0.366693*q^-2
-----
-0.956852*q^-1 + 0.610756*q^-2 + -0.214689*q^-3

Change in system after number of observations, i=428. lambdaMin=1
toIdentify:

0.806047*q^-1 + 2.20008*q^-2
-----
-0.203594*q^-1 + 0.200285*q^-2 + -0.752877*q^-3

numAntigens=3
toIdentify:
00100110010100010000000110010010011000000111
currentMatch[0]=2.94102
currentMatch string:
00100011010010000100000010010010101000011000
0.653959*q^-1 + 0.548387*q^-2
-----
-0.717647*q^-1 + 0.317647*q^-2 + -0.247059*q^-3

currentMatch[1]=11.7085
currentMatch string:
01000001111111000010010000000000101000000000
2.08504*q^-1 + 10.2845*q^-2
-----
1.00784*q^-1 + -0.686275*q^-2 + -1*q^-3

currentMatch[2]=0.953844
currentMatch string:
00100011010100000100000010010010001000001000
0.653959*q^-1 + 2.04985*q^-2
-----
-0.717647*q^-1 + 0.0666667*q^-2 + -0.74902*q^-3

stringLength =44
best match index=2 best match =0.953844
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
system retrieved, systemAIS details:

0.653959*q^-1 + 2.00293*q^-2
-----
-0.247059*q^-1 + 0.160784*q^-2 + -0.74902*q^-3

i=428 resetting values

-----
initial change in system detected, i = 500, systemRLS:

0.75055*q^-1 + 2.11262*q^-2
-----
-0.225703*q^-1 + 0.185728*q^-2 + -0.723512*q^-3

Change in system after number of observations, i=525. lambdaMin=1
toIdentify:

1.61842*q^-1 + 10.0669*q^-2
-----
0.891612*q^-1 + -0.719466*q^-2 + -0.817802*q^-3

numAntigens=3
toIdentify:
0011011111110101111001111000000100000000101
currentMatch[0]=13.7001
currentMatch string:
00100011010010000100000010010010101000011000
0.653959*q^-1 + 0.548387*q^-2
-----
-0.717647*q^-1 + 0.317647*q^-2 + -0.247059*q^-3
```

Gary Evans
System Identification using an Artificial Immune System

```
currentMatch[1]=1.01577
currentMatch string:
01000001111111000010010000000000101000000000
2.08504*q^-1 + 10.2845*q^-2
-----
1.00784*q^-1 + -0.686275*q^-2 + -1*q^-3

currentMatch[2]=11.4457
currentMatch string:
00100011010100000100000010010010001000001000
0.653959*q^-1 + 2.04985*q^-2
-----
-0.717647*q^-1 + 0.0666667*q^-2 + -0.74902*q^-3

stringLength =44
best match index=1 best match =1.01577
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
system retrieved, systemAIS details:
1.53372*q^-1 + 10.0733*q^-2
-----
0.882353*q^-1 + -0.717647*q^-2 + -0.811765*q^-3

i=525 resetting values

-----
initial change in system detected, i = 600, systemRLS:

1.62244*q^-1 + 10.0337*q^-2
-----
0.89543*q^-1 + -0.713533*q^-2 + -0.808567*q^-3

Change in system after number of observations, i=625. lambdaMin=1
toIdentify:

0.670328*q^-1 + 0.648104*q^-2
-----
-0.872624*q^-1 + 0.580938*q^-2 + -0.153065*q^-3

numAntigens=3
toIdentify:
00100011100010001100000001000011001000011010
currentMatch[0]=0.628347
currentMatch string:
0010001101010000100000010010010101000011000
0.653959*q^-1 + 0.548387*q^-2
-----
-0.717647*q^-1 + 0.317647*q^-2 + -0.247059*q^-3

currentMatch[1]=15.0457
currentMatch string:
01000001111111000010010000000000101000000000
2.08504*q^-1 + 10.2845*q^-2
-----
1.00784*q^-1 + -0.686275*q^-2 + -1*q^-3

currentMatch[2]=2.68332
currentMatch string:
00100011010100000100000010010010001000001000
0.653959*q^-1 + 2.04985*q^-2
-----
-0.717647*q^-1 + 0.0666667*q^-2 + -0.74902*q^-3

stringLength =44
best match index=0 best match =0.628347
to return:
Attempting to obtain system from AIS (for lambdaMin=1):
system retrieved, systemAIS details:
0.618768*q^-1 + 0.607038*q^-2
-----
-0.623529*q^-1 + 0.317647*q^-2 + -0.121569*q^-3

i=625 resetting values
```

The results are as expected from Table 11. The AIS was bred for the same number of generations as in the previous example, and the correct systems were retrieved from the AIS in spite of the presence of noise. The system parameters were not correctly identified, but the system retrieved from the AIS on the subsequent encounters of the system are better than the initial estimate of the system.

This is illustrated at iteration 325, where a quick estimate of system 1 gives:

$$y(t) = \frac{(0.759042q^{-1} + 0.283714q^{-2})u(t)}{(-0.838568q^{-1} + 0.611674q^{-2} - 0.149177q^{-3})}$$

The system retrieved from the AIS at this estimation is the system added to the AIS at iteration 100. This system has been better identified, and the parameters of the system retrieved are:

$$y(t) = \frac{(0.618768q^{-1} + 0.607038q^{-2})u(t)}{(-0.623529q^{-1} + 0.317647q^{-2} - 0.121569q^{-3})}$$

This shows how the AIS can help to quickly retrieve a known system with a quick estimate of its parameters.

6 Conclusion

This work has investigated the use of an artificial immune system for system identification. The value of lambda has been shown to be a good indicator of the change in system regime, and has been used throughout.

The value of lambda was also shown to be a good identifier of the new system regime being entered, as long as the system regime changed when the system regime changed. This allowed the use of the AIS to be investigated. It was shown that precision of lambda, based on the number of bits used to encode it, was important. It was also shown that care must be taken to ensure that the AIS undergoes breeding for an adequate amount of cycles, with enough clones and repository antibodies to allow the search space of possible solutions to be covered.

Both the precision and the convergence upon the correct solution are important in making sure that a system is retrieved from the AIS if present, and that the wrong solution is not inadvertently retrieved. The AIS was shown to perform better with an affinity measure of the actual genotype of the binary string, rather than the number of matching binary strings in the encoding of lambda. This is due to the greater value of left-hand bit values. Future work may investigate the possibility of using Gray coding.

The encoding of the system parameters was shown to be a more robust method of system identification; the system was correctly recognised using a reference of a Pseudo Random Binary Sequence (PRBS). Again, issues with the number of bits used to encode the system coefficients, and the amount of breeding of the AIS, were shown to be important. A measure of affinity based on the number of matching bits was, again, shown to be insufficient. A measure of the affinity based on the 'nearness' of the coefficients was devised, and shown to work satisfactorily. This encoding was shown to work in the presence of noise, where a

better-estimated system can be added to the AIS, and retrieved from an estimate using few estimations.

Further work might focus on getting quantitative measures of the number of bits to use, and the amount of breeding necessary, taking into the account the computational effort to get the results. The computational effort would be important were this work to be used as part of a control scheme. The relationship between the number of antigens and the number of antibodies would be a further avenue of investigation; to see whether a number of antigens may be adequately represented with a lesser number of antibodies.

It would be interesting to apply this work to real-world data. In this case, the reference may be unchanging, meaning that not enough information is present in the RLS algorithm for a good identification of the system, or the system regimes being entered may be similar.

7 References and Bibliography

7.1 References

Robert E. Smith and Stephanie Forrest. "Population diversity in an immune system model: Implications for genetic search." in *Foundations of Genetic Algorithms 2 (FOGA-92)*, L. Darrell Whitley, editor. Morgan Kaufmann. San Mateo, California. 1992.

Leandro N. de Castro and Fernando J. Von Zuben. *Learning and Optimization using the Clonal Selection Principle*. IEEE Transactions on Evolutionary Computing. Vol 6. No 3. June 2002.

D. Dasgupta. *Guest Editorial, Special Issue on Artificial Immune Systems*. IEEE Transactions on Evolutionary Computing. Vol 6. No 3. June 2002.

C. J. Harris and S. A. Billings. *Self-Tuning and Adaptive Control: Theory and Applications. Revised 2nd Edition*. Peter Peregrinus Ltd. 1985

P. E. Welstead and M. B. Zarrop. *Self-Tuning Systems. Control and Signal Processing*. Wiley. 1991.

7.2 Bibliography

D. Dasgupta. *Artificial Immune Systems and Their Applications*. Springer 1999.

David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley. 1989.

Herbert Schildt. *C++ From the Ground Up*. Osborne, 1998.

8 Appendix

This appendix documents the code used to generate the results. The code is provided on a disc; each section gives the source code file name being discussed. The appendix starts by documenting general classes which are used throughout - a class used for matrix algebra, a class used for the RLS algorithm, and a class used to describe a system.

8.1 Matrix class

The code for a simple matrix class is shown below. The class is instantiated by giving the number of rows and columns in its constructor. The matrix supports arithmetic operations for scalars and other matrices; operators are overloaded to allow operations such as `Matrix*Scalar`, and `Matrix-Matrix`. Elements are accessed and set with the `getValue()` and `setValue()` accessors - these are non-zero indexed, to be consistent with the normal method of referring to matrix entries. The data is stored internally in an array of size `rows*columns`.

The operations supported on the matrix class are shown in Table 12. For any operations that cannot be performed on arrays, such as adding two arrays of different size, an exception is thrown. The exception is thrown as a string, and can be caught with the following:

```
try
{
    cout << "matrix1 + matrix2=" << matrix1 + matrix2 << endl;
}
catch(char *str)
{
    cout << "exception caught:" << endl;
    cout << str << endl;
}
```

Operation	Description	Constraint
getTranspose()	called on matrix of order n*m, returns matrix of order m*n	none
getValue(row, column)	returns scalar value; the value stored in the row and column	row and column must be in range
setValue(row, column, value)	sets value stored in the row and column to the parameter value	row and column must be in range
setEye()	sets the values on the leading diagonal to '1', all others to '0'	none
matrix1 * matrix2	returns the result, the row order of matrix1, and the column order of matrix2	column order of matrix1 must match row order of matrix2
matrix * value	returns the result of multiplying matrix 1 by a scalar	none
value * matrix	as matrix * value	none
matrix + value	returns the result of adding a scalar to each element in the matrix	none
value + matrix	as matrix + value	none
matrix - value	returns the result of subtracting a scalar from each element in the matrix	none
matrix1 + matrix2	returns the result of adding matrix1 to matrix2	matrix orders must match
matrix1 - matrix2	returns the result of subtracting matrix2 from matrix1	matrix orders must match
operator(double)	converts a matrix to a scalar	the matrix order must be 1x1

Table 12. The operations supported on the matrix class.

The operator(double) is used to cast a matrix object to a double, as in the following:

```
double result;
result = matrix1 * matrix1.getTranspose();
```

This is only valid where the multiplication of two matrices results in a scalar, i.e. when the resulting matrix is of order 1x1.

8.2 The System class

The System class holds the numerator and denominator parameters of a system. The code for this class is given in `system2.h`. The constructor class initialises all of the parameters to zero. The denominator and numerator parameters are stored in their own arrays, and have their own accessor methods: `getNumerator(int index)` and `setNumerator(int index, double value)`, with similar methods for the denominator parameters. The accessor methods throw an exception if the index is out of bounds.

The parameters are stored as $a_1=\text{denominator}(0)$, $a_2=\text{denominator}(1)$, ... , $a_n=\text{denominator}(n)$. Similarly $b_n=\text{numerator}(n)$.

The `isEmpty()` method returns a bool of value true if each of the parameters are set to zero. This is useful for determining when returning a System from the AIS, if no system exists then an empty system is returned.

8.3 RLS and RLSSys classes

The RLS algorithm, as in section 4.1, is implemented in the RLS class. The code routine is present in `RLS.h`. The class is instantiated by calling the constructor with the number of numerator and denominator coefficients. The constructor allocates memory for the matrices and initialises the variables. The class has accessors to retrieve the value of lambda and perror.

A new observation is added to the RLS object with the `addData(double ut, double yt)` method. This is where the actual RLS algorithm takes place - the parameter vector, the covariance matrix and the observation vector are updated with the new observation.

The RLSSys class inherits from the RLS class and adds a new method; `getSystem()`. This method retrieves the system parameters by returning a System object. The code for this class is found in the file `rlssys2.h`.

8.4 Code for AIS to recognise binary strings

The code to perform the recognition of binary strings, as was performed in Chapter 3 is found in the file `clonalg1h.cpp`. The AIS holds a collection of antigens, which contain the strings to be recognised. The AIS also holds two populations, one is a memory population, which holds the memory antibodies corresponding to the antigens. The other population contains the repository antibodies. The Antigen and Antibody objects both contain binary strings which are accessed with the `getBit()` and `setBit()` methods. The Antibody class defines a method, `calcAffinity()`, which takes in an antigen object and sets the Antibody affinity to the number of similar bits in their strings. The `AIS::breed()` method performs

breeding, as described in the CLONALG algorithm, presented in 2.2.1. An antigen is selected randomly from the population, this antigen is presented to the `generateClones()` method, which generates a population of child antibodies. This method makes use of the `Population::calcAffinity()` method to calculate the affinity of each antibody in the population to the antigen. This affinity is used to select the two parents for breeding by tournament selection, after which a crossover mask is generated for uniform crossover and crossover proceeds.

The affinity of the child clones to the antigen is found, and the best clone is retrieved. If the affinity of that antibody to the antigen clone is higher than the affinity of the memory antibody, then that antibody is replaced. A new random population of antibodies is generated, to replace the worst d antibodies in the repository population.

8.5 Code for the AIS to Recognise Systems based on Lambda

The UML diagram for the program `sysID16lambda.cpp` is shown in Figure 21. The Chromosome class holds the binary string, with accessor methods to set and get the value of each individual bit. The class also has a method, `calcAffinity()` that takes in another Chromosome object, and calculates the number of similar bits in their strings. The Antigen and Antibody classes both inherit from this class.

The Antigen class holds the value of lambda associated with the system change, and a system class that the system change identifies. The binary string that the Antigen and Antibody classes inherit from the Chromosome class holds the value of lambda. Breeding in the AIS happens as previously. To retrieve the results found in section 4.4, where the AIS is used as a library of antigens, the commented out code in `AIS::getSystem()` method is used in place of the code as presented on the disc. This only searches the antigens and not the antibodies for a good match, and returns any antigen that is found to be a good match.

The Project class holds the variables used for system identification. The arrays `ut[]` and `yt[]` hold the system inputs and outputs, which are generated in the `generateSysResponse()` method.

The `Project::sysID()` method performs the task of identifying the systems, and adding and attempting to retrieve them from the AIS. The algorithm is as in section 5.1. Observations are added to the `RLSSys` class instance, and the value of `lambda` is retrieved as a trigger indicating that the system has changed. The estimation of the system parameters is retrieved from the `RLSSys` object as a `System` object.

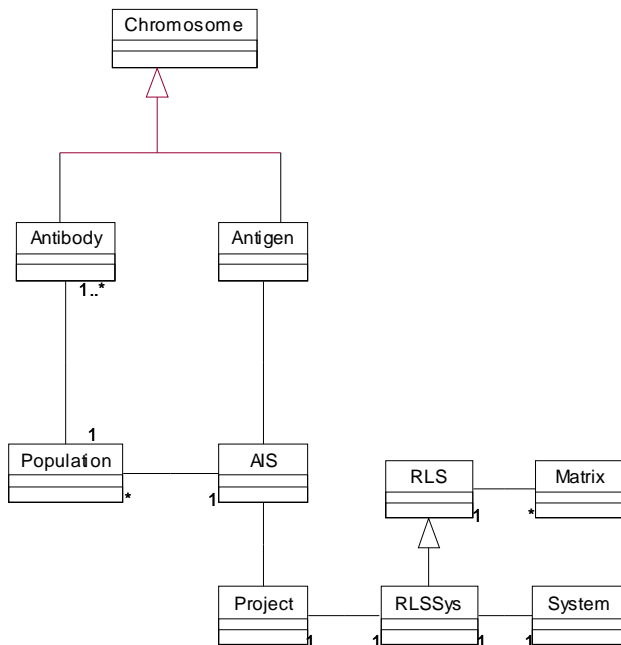


Figure 21. UML diagram for using lambda to classify the system.

Gary Evans
 System Identification using an Artificial Immune System

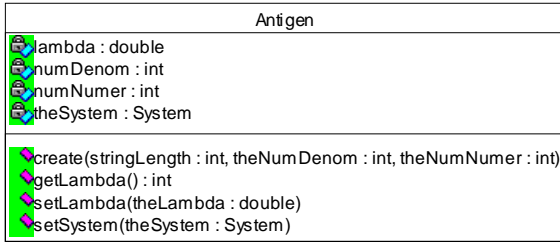


Figure 22. The Antigen class

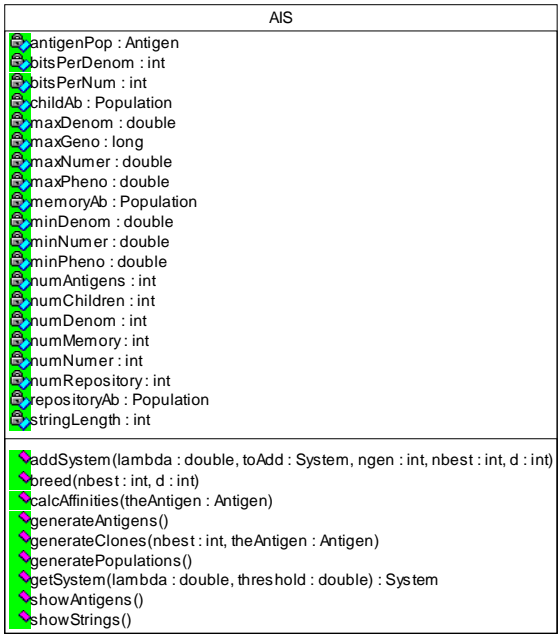


Figure 23. The AIS class.

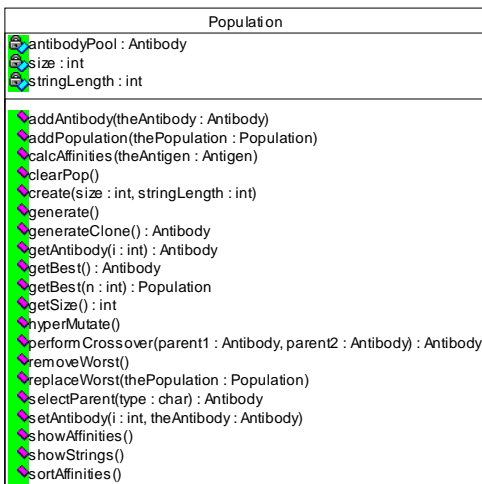


Figure 24. The Population class.

8.6 System Recognition based on System Parameters

The code to identify the system based on an estimation of the system parameters is found in `sysID14sysParams.cpp` and `sysID15sysParams.cpp`. The first file bases the affinity on the number of matching strings, the second file bases the affinity on the match between the systems that the antibody and antigens represent. The first version of the program is discussed first. The UML diagram is as in Figure 21. The Chromosome classes are as before, as are the System and RLSSys classes. Now, a SystemAIS class is inherited from both the System and Chromosome classes. This gives the System the ability to convert its parameters into a binary string, and conversely, to convert its binary string to its system parameters. This allows its parameters to be easily stored as a binary string, which is then more readily assigned to and from the antigens and antibodies.

The algorithm in the `Project::sysID()` method is as in section 5.4. The SystemAIS class is used throughout, where the System class was used in the previous program. The `getSystem()` method is overloaded: one method takes in a SystemAIS object and an int, and returns the closest match based on the affinity of the bits, as in the previous section. The overloaded method that takes a SystemAIS object, and a double, calculates a good match based on how near the system parameters are.

8.6.1 Affinity Based on the System Parameters

The program in the previous section is amended to allow the affinity to be based on the match between the systems that their strings represent. The `calcAffinity()` method is removed from the Chromosome class, and instead is present on the Antigen class. The Antigen constructor is amended with enough information to allow it to instantiate its own systemAIS object.

The maximum affinity with the minimum and maximum phenovalues given in section 5.4.3 gives a maximum difference between systems of 30. To keep a high affinity as a good match, the system difference is given as

$$\text{affinity} = 3000 - (100 * \text{difference})$$

The results are multiplied by 100 to maintain precision whilst using an integer datatype.