Equation discovery techniques employ genetic algorithms to automatically distil data into analytical laws. The overall strategy for the project is to generate comprehensive training sets from the agent-based models and to use equation discovery to derive differential equations from the data.

We will cover the strengths and weaknesses of each of the systems and their suitability to deriving. As we intend to use equation discovery within the project, an understanding of the different available systems will be crucial in aiding us to select the right one for our application.

Processes for generating, collecting and then storing data are becoming increasingly powerful and autonomous. However, the methods for taking this data and deriving analytical rules from it are nowhere near as sophisticated. There is an increasing demand for a complimentary field to automated data generation which takes this data and turns it into useful formulae or equations. The proposed answer to this demand is equation discovery. The underlying principle of equation discovery is to employ genetic algorithms to search through the data generated from a system, and to use this data to identify useful analytical relations related to the dynamics of the investigated system.

This section will focus on a review of different equation discovery systems. The first is BACON [27, 29, 30] (Section 5.1.1), one of the original equation discovery systems. The next two, LAGRANGE [11, 12] and LAGRAMGE [60, 62] (Section 5.1.2) are related; with LAGRAMGE being a development of LAGRANGE, both having been developed post-BACON. The final equation discovery system is Nutonian Eureqa [56] (Section 5.1.3) and, by far, is the most advanced of the four. Hence, the coverage of Eureqa will be considerably longer than the discussion of BACON, LAGRANGE and LAGRAMGE.

Other equation discovery systems were considered for review, but were not included in the final few for various reasons. COPER [24], FAHRENHEIT [23], Equation Finder (EF) [28, 65], ABACUS [13], IDS [40], ARC [38] and E\* [55] were all developed in the era between BACON and LAGRANGE, but do not extend to our aim of deriving differential equations. The reason BACON was chosen for the review is as it is the original system for equation discovery, and so is monumental in that respect. LAGRANGE has been identified as the turning point for such systems as it extended the scope of equation discovery to differential equations. It was also intuitive to include LAGRAMGE as it addressed some of the weaknesses of LAGRANGE.

The GOLDHORN [26] and SDS [63] systems of equation discovery were also considered for the review, and were produced in the period between LAGRANGE and LAGRAMGE. GOLDHORN was an extension of LAGRANGE designed to deal with noisy data. SDS, on the other hand, attempts to address the same problem as LAGRAMGE; reducing the space of possible equations in the discovery, but uses a different strategy. SDS employs information on scale types of the dimension units of the system variables in order to achieve the problem space reduction, in comparison to LAGRAMGE which uses a context-free grammar. The reason that these two systems were omitted from the review is that, for this project, they do not mark an important juncture in the development of equation discovery systems. In addition, the problems that they address are better, and more reliably, solved using the Eureqa system.

5.1 Equation Discovery Systems

The following section of the research review will be used to discuss each of the equation discovery systems we have selected. Starting with BACON we will cover the origins of equation discovery. The next part will then move to LAGRANGE and LAGRAMGE in order to discuss how these systems developed, before covering Eureqa, the most developed of the modern day equation discovery implementations.

5.1.1 BACON

Developed from 1977 to around 1987, BACON [27, 29, 30] was one of the first systems of its kind and provides the basis of a number of the modern day methods of equation discovery [11, 12, 56, 60, 62]. Throughout its development it had six iterations, each building on the iteration before.

The first iteration, BACON.1, used a heuristic-driven approach to identify trends and constants in order to discover simple numeric laws between two variables. BACON.2 had the same goal, but used a differencing technique. The third iteration, BACON.3, combined the approach taken in BACON.1 with methods for recursing to higher levels of description. This extended the discovery from two variables. BACON.4 extended BACON.3 in order to note common divisors in cases where symbolic terms were involved, and to postulate intrinsic properties of the system. Following on from this was BACON.5 which incorporated a more general differencing technique than BACON.2, but still aimed to find simple numeric relations. In addition, this iteration reduced the search through the space of possible laws by using expectation-based methods. The final BACON system, BACON.6, replaced the differencing technique with a hill-climbing method in order to better handle noisy data. The progress of BACON was used to address a number of flaws in the system. Initially it could not construct conditional laws, in which the conditions were restrictions on the values of related variables.

Secondly, the system knew before the equation discovery was implemented, that certain dimensions were useful to generalise over, which removed some of the generality of the system.

Since BACON stopped its development, a number of more sophisticated systems have been implemented. It is these systems we will examine next.

5.1.2 LAGRANGE & LAGRAMGE

LAGRANGE [11, 12] was developed in 1993 and is a system for discovering differential and ordinary algebraic equations involving more than two variables. It is both open source, and written in C. The system functions by keeping all but two variables constant, and then running experiments that vary these values. By cycling through all of the possible variable pairs one can attempt to identify the relationships between the variables in the system.

The equation discovery system uses this to look for relationships within the data by minimising an error metric between a function that uses the specified grammar, and the provided data. The process terminates after a certain time limit, or after the error has reached a certain threshold. The main flaw with LAGRAMGE links to the use of a grammar to limit the search space of functions. This provides two obstacles. Firstly it requires specialist knowledge in the domain of the application in order to define a sensible grammar. Secondly, it is very limiting. If the solution does not fall in the defined grammar, then it will not be found.

5.1.3 Nutonian Eureqa

Nutonian Eureqa [56] is the most developed of the current equation discovery systems and is regularly used in a variety of contexts; both academic and commercial [10, 37, 64]. BACON, LAGRANGE and LAGRAMGE all helped to pave the way for Eureqa, and so are important in that they offer context for the system. However, in terms of implementation, Eureqa is the most advanced to date. It addresses many of the problems of previous systems such as needing to declare a grammar or not being able to deal with noisy datasets, and is much more user friendly. Due to this, it has the highest probability of being implemented in the project, and so deserves the most focus in the research review.

Eureqa employs symbolic regression [25] to achieve its aim.

The symbolic regression algorithm functions by forming initial expressions randomly from algebraic operators (+, −×, ÷, ...), analytical functions (sin, cos, tan, ...), constants, state variables and other mathematical functions and operators. New equations are then formed by recombining the existing ones and probabilistically varying their subexpressions. Equations that minimise a specified error metric are retained, and the process continues; terminating at the point when the measurement of error has reached a certain threshold, or after a certain time.

Within the paper on which Eureqa is based (Distilling Free-Form Natural Laws from Experimental Data, Schmidt, M. and Lipson, H. (2009) [56]) several important factors relating to the algorithm are discussed. The first of these factors is how the symbolic regression deals with interpolation. Within the algorithm we have a set of data and would like to find a model that minimises the error metric between itself and the data. The easiest way to do this is by connecting the datapoints with straight lines, as the error measurement will be zero. However, this model will poorly predict the underlying relationship between the variables.

In order to prevent this occurring, the algorithm divides the data into a training set and a validation set. The training set is used to generate potential solutions, which are then tested on the validation set. If the algorithm were to interpolate within the training set, on testing it on the validation set, its poor predictive ability would be recognised and it would be rejected. The optimal solution the both minimises some error metric for the training set, and also accurately predicts the validation data. Also included within the paper is a discussion of the importance of finding meaningful and non-trivial invariants. In relation to the project, this speaks to what we have previously discussed about spatial information. The principle of equation discovery is to identify useful analytical relations associated with the dynamics of the investigated system. We can only discover these analytical relations given the necessary components to form them. Therefore, a failure to include a sense of spatial information could result in a failure to identify the relationship between that, and the population dynamics of the colony.

Conversely, we could also include too much information, and identify meaningless relationships. For example, if we were to take the function f(y) = 1 then we could identify the function f in several different forms: Each of which is equivalent to the true form, but with varying levels of complexity that could be simplified away. Eureqa deals with this by not presenting a single solution, but a small set of solutions that lie across an accuracy-parsimony Pareto front. Parsimony is defined to be the inverse of the number of terms in the expression and accuracy is defined in terms of validation data. This addresses one of the weaknesses of LAGRANGE which we saw in Section 5.1.2. Something noted within the original Nutonian paper, and that has been witnessed in experimentation with Eureqa (refer to Section 6.2 for details), is that this Pareto front contains a point where predictive ability jumps rapidly with some minimum complexity, after which accuracy only increases minimally with added complexity. When searching for solutions we will aim for them to fall on this point, as it represents the optimal balance between accuracy and complexity.

Figure 11: The accuracy-parsimony Pareto front for rediscovering one of the Planqu ́e model equations from its simulated data. On the left is a plot of the number of ants active in the old nest through time. On the right is the plot of error versus complexity for the discovered solutions. Highlighted with a red dot is a

However, this definition of complexity has its weaknesses. Choosing to use the number of terms does not necessarily reflect the complexity of a function. For example, 1 have an equal impact on complexity as just an x term.

Another useful facet of the paper on which Eureqa is based is the methodology employed when using he algorithms. Within the publication they talk at length about how they validated Eureqa using two systems, an air track oscillator, and single and double pendulums. By evaluating their methodology for testing the algorithm we hope to inspire our own, which will be covered in Section 7.2.1. Single and double pendulums can be justified as a reasonable testing choice, as within these two similar experiments it can be demonstrated that the algorithm can discover both simple and complex systems.

The harmonic motion generated by single pendulums is a well understood, simple dynamic. However on attaching a second pendulum we transcend into the realm of chaos after reaching certain energies 1 Figure 12: On the left is the single pendulum. The θ angle is used in equations of motion along with the mass, m of the ball. On moving to the double pendulum we introduce another mass and angle, creating

Their experiments underline a few key points relating to the project work. Initially, they discuss how, dependent on the form of the data, the system will discover different laws. Contextualising this in terms of the pendulums; given position and velocity, the algorithm discovered the Hamiltonian and Lagrangian energy equations. Given the acceleration data, it discovered the differential equation of motion corresponding to Newton’s second law. Finally, given only position data, it discovered the equation of a circle for the single pendulum (the circle is due to the pendulum being confined to this trajectory).

1Does God Play Dice, Stewart (2002) contains an excellent explanation of this, as well as some fascinating discussion around the subject [59].

This relates to the project in that the form of the data given to the algorithm will determine its output: therefore it will need to be picked carefully. The form of the data will be discussed in detail in Section 7.2.2 on deriving the differential equation summaries of SPACE and AH-HA, but has been noted here as a point of interest.

The final piece of information it is possible to take away from their paper are the findings in simulations when they did not give the algorithm the correct building blocks. Since the algorithm incrementally builds solutions from initial combinations of mathematical operators and functions, if the correct operators and functions are not presented to the algorithm, it will struggle to find the most accurate solution. On testing this, they note that the algorithm can develop reasonable approximations to functions. For example we can expand sin into its Taylor series: If we were to look for a solution from the algorithm which included sin but did not give it the necessary building block, then it may approximate the function using this. Being aware of approximations and equivalent forms will become useful in our own employment of the algorithm, and is well highlighted by the paper.