University of Twente

faculty of
electrical engineering

Recognition of Structures in Numerical Data

International Master Student in Mechatronics:

Dimitrios Iakovou

MSc. Thesis

Supervisors: **Prof. Dr. Ariën J. van der Wal**
Prof. Dr.Ir. Job van Amerongen
Ir. Mark Verwoerd

Report No: 003CE2003
March 2003

Control Engineering Laboratory
Electrical Engineering Department
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Acknowledgements

Working on this project was similar to trying to navigate a ship on an open sea. There are so many directions to take and places to explore. Only with the support of your fellows and your personal work its possible to finally reach your destination. Now, after having completed this project's work, I can look back and see that with all its difficulties, it was a very educating and enjoyable "journey".

Acknowledging the people who help me get this far, I want to thank my parents and brothers, for their 'multilevel' support and for their love. I would also like to thank Prof. Dr. Ariën J. van der Wal for finding this project for me, for being an inspiring supervisor and instructor (though sometimes too confident about my abilities). I also want to thank Prof. Dr. Ir. Job van Amerongen, as if not for him, I might not have the chance to do my Master studies in the University of Twente. But mostly, I want to thank Prof. van Amerongen for being there for all of the students, and for providing a warm, family like atmosphere together with the rest of the members of the Control Group. Finally, I would like to thank all of my friends for being there for me when I needed them, and for helping me to develop a taste for beer and frikandel.

Abstract

In control engineering, system identification is of the utmost importance for determining the correct control parameters for a plant. In the present research we investigate the structures present in numerical plant data and derive symbolic expressions that correspond to these structures. Therefore, this research can also be applied to the inverse problem. For the development of such a recognition system, we will make use of the morphogenetic neuron. Like classical neuron networks, the Morphogenetic neurons are capable in recognizing, but in a higher level of abstraction. The Morphogenetic neurons are able to encode abstract, symbolic expressions that characterize the relations between the inputs and outputs of a system.

Generally, in most measured data, there are (hidden) underlying relations (correlations, invariants, rules). Our approach is to automatically recognize these underlying structures by observing the dimensionality of the basis that is required for representation of the data. The basis is approximated by a morphogenetic neuron. The basic problem is the selection of the number and the type of the variables that will participate in the basis functions. Additionally, it is desired not to make any a-priori assumptions on the properties of the numerical data and let the Morphogenetic neuron derive to a solution unaided.

Within this project, the research has been limited to 2-dimensional space for computational reasons, but this technique is generally applicable to n-dimensional space.

Contents

1 Introduction

The first step towards designing a controller for a process or a system is the development of an efficient system model. There are times when these models can be constructed by having knowledge of the dynamics of the elements that exist on the real system. But what happens when this knowledge is unknown and we need to treat the system as a black box? At these cases system identification techniques are the ones that help us develop an approximate model of the system, using input and output sampled numerical data.

There are numerous methods to perform system identification on a given "black box" (Parameterization, Model reduction, etc). This project sets the basis for the development of such an identification technique using the "Morphogenetic Neuron" of G. Resconi and A. J. van der Wal [1]. At this first step, the morphogenetic neuron is used to recognize structures hidden in numerical data and derive a symbolic (multinomial) expression of those structures (Fig.1.1).

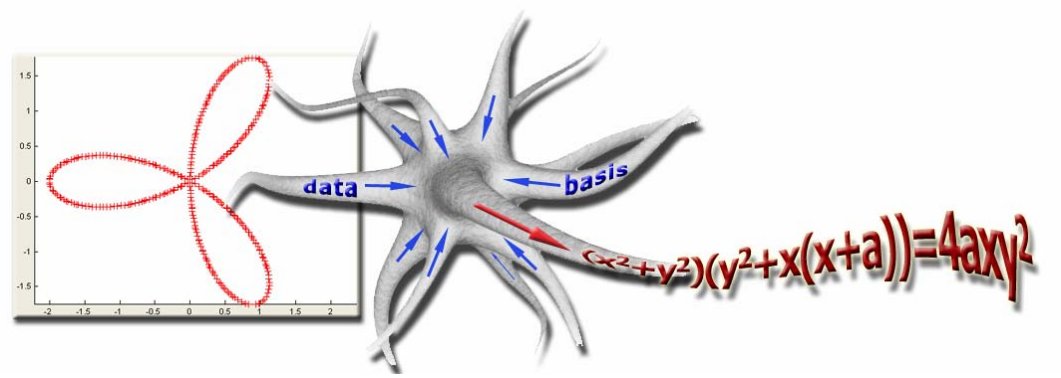


Fig.1.1: Functionality of the Morphogenetic Neuron

Every plant, system, process can be simply thought of as an equation that relates the numerical values of the input(s) to the ones of the output(s). This relating information exists hidden in the input/output numerical data. The use of the morphogenetic neuron aims to the automatic recognition of those underlying (hidden) relations in the numerical data. The efficiency of the recognition of the morphogenetic neuron depends on the number of participating variables, the maximum multinomial power, the number and type of the participating basis functions.

To evaluate the performance of the morphogenetic neuron and the quality of the recognition algorithm several tests were performed. In this first approach the test were performed on data belonging to known multinomial curves (circle, ellipse, difolium, trifolium, etc). These tests were constricted in two dimensions, for computational reasons, but this technique is without restriction applicable to n-dimensional space. Further tests were performed on the same curves but with the addition of noise into the numerical data values, to evaluate the influence of noise in the recognition process.

An autonomous evaluator based on statistic and soft computing means was also implemented to estimate the goodness of the fit to the data and choose the best multinomial fitting curve. This evaluator also performs some optimization operations and is able to manipulate and control the parameters of the algorithm using the Morphogenetic neuron.

Finally, a platform was developed to provide the user with the ability to manually control and set the recognition parameters and observe the whole process. All of the parts of this project were implemented and developed using MatLab development tool and its toolboxes.

2
Theoretical Background

2.1) Introduction

Within this chapter a more extensive description of the morphogenetic neuron as well as its structural elements will be given. Additionally, a reference on the mathematics background of the morphogenetic neuron, its dependencies and the parameters that govern its behavior will be made. Finally, a description of the Fuzzy Logic inference mechanism and background will be given.

2.2) The Morphogenetic Neuron

The morphogenetic neuron that was developed by A.J. van der Wal and G. Resconi [1] is in principle a neural network model, since the output is basically the weighted sum of the inputs. Still, there are fundamental differences between the morphogenetic neuron and the classical neuron of McCulloch and Pitts. The learning process, unlike the classic neuron, does not happen iteratively but it is a one step process leading to the final contributing input weights. The most important difference is that the morphogenetic neuron has the ability to produce abstract relations between the inputs and the outputs and exceed the "black box" capabilities of the classical neuron. The morphogenetic neuron is designed, based upon recent studies from neurobiology, to perform similar activities to the human brain for learning, recognition and identification of underlying structures in data (vision, sound, etc). From this point on, for more convenience, the "morphogenetic neuron" will be referred as "neuron" in the body of this document, unless any other distinctions are clearly made.

The neuron consists of four parts (Fig.2.1): the basis functions, the training data, the basis functions' weights and the output (recognized structure). The most important element of the neuron is the basis functions, and the correct choice of the basis functions is very crucial for achieving efficient results.

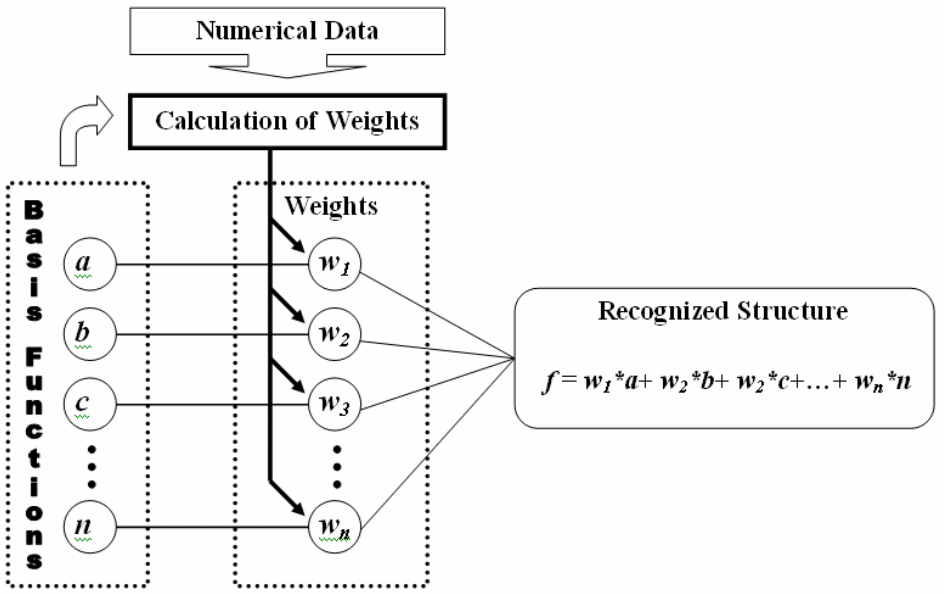


Fig.2.1: The Morphogenetic Neuron

The basis functions play two roles in the neuron. They are used by the training data to derive the basis weight values, and they are also combined with those weight values to produce the final mathematical representation of the recognized structure in the data.

Within this work, the neuron was used to recognize structures (of the multinomial form) in data existing in a two dimensional orthogonal coordinate system. Therefore the result of the neuron will always be a multinomial function, where each of the basis functions will be an element of the multinomial.

2.3) Mathematical background of the Morphogenetic Neuron

The neuron actually performs a multinomial fit on the training data. In general, every multinomial function can be described by the following equation (Eq.2.3.1) where ψ are the basic elements (variables or relation among variables, e.g. x^2, y^2z^6, etc) of the function and w are the weights (coefficients) of those basic elements. These basic elements will be referred to as "basis functions".

$$f = \sum_{j=1}^{n} w_j \psi_j \qquad \text{Eq.2.3.1}$$

If we have a number of sets of input and output numerical data samples of the same system, then for each of the input/output numerical data group, a multinomial equation of the type of Eq.2.3.1 can be formed. Since the data sample sets belong to the same system, they must all satisfy the same multinomial equation, when the correct basis function weights are applied. Still, the number and form of the basis functions that take part in the multinomial is unknown. Therefore an initial estimate of the possible elements ψ and their numerical value according the data sample sets must be made. In matrix form, Eq.2.3.1 can be rewritten in the form of Eq.2.3.2, where 'm' is the number of the basis functions and 'n' is the number of the numerical data. Each column of the

DataMatrix contains the numerical value of the basis function for the corresponding numerical data group.

$$f_{nx1} = DataMatrix_{nxm} * Weights_{mx1} \qquad \text{Eq.2.3.2}$$

The rest or the approach is based on solving Eq.2.3.2 towards finding the vector *Weights*. There is need to isolate the weights vector in order to be able to calculate its value. For this reason both parts of the equation are multiplied with the transposed *DataMatrix* (Eq.2.3.3).

$$DataMatrix^T * f_{nx1} = DataMatrix^T * DataMatrix_{nxm} * Weights_{mx1} \qquad \text{Eq.2.3.3}$$

This will result to the *G* matrix, which contains all the information of the data and is rectangular. It is also identical to the least squares matrix as it contains all the squared elements across it main diagonal. Also the result of the transposed *DataMatrix* and the output vector is replaced by the *b* matrix (Eq.2.3.4).

$$b\_Matrix_{mx1} = G\_Matrix_{mxm} * Weights_{mx1} \qquad \text{Eq.2.3.4}$$

Since the *G* matrix is rectangular, its inverse matrix can be calculated as long as it can be defined (determinant is nonzero). If the *G* matrix is invertible then both sides can be multiplied with it resulting with the values for the weights of the basis functions (Eq.2.3.5).

$$Weights_{mx1} = G\_Matrix^{-1} * b\_Matrix_{mx1} \qquad \text{Eq.2.3.5}$$

Having obtained the basis functions weights, the multinomial can be reconstructed. The resulting multinomial is the best possible fit according the freedom that is given to the neuron. It is obvious that the performance of the neuron depends directly to the selection and number of the basis functions. Therefore an evaluation mechanism is required to estimate the quality of the fitting and the possible need of change of the neuron's parameters.

2.4) Fuzzy Logic

As it was mentioned in paragraph 2.3, the efficiency of the neuron in recognizing a structure depends on the correct selection of the participation basis functions. Therefore it is important that the generated multinomial is tested and evaluated for its ability to fit to the numerical data.

When the numerical data are indefinitely accurate, there are several mathematical procedures, such as Chi-square, that can help identify the best fit. In reality, noise or errors are always present in numerical data, making the use of some kind of selection mechanism a necessity. For this reason Fuzzy Logic was selected to be used to aid the evaluation mechanism in identifying the best fit.

The task of recognizing a good multinomial fit is very complex. Even among people, what seems to be a good result for some is not for others according their requirements and experience. Fuzzy controllers make use of experience models. The description of the decision rules is not an explicit formula, but it is expressed in linguistic form. The basic problem in the design of a fuzzy logic evaluator is the representation of such an experience model in a concise and computational treatable way.

In general, there must be a mechanism that is able to translate the several parameters of the recognized multinomial to fuzzy concepts (fuzzification). There must also be a mechanism to make the fuzzy decisions according the controlling knowledge as stored in linguistic rules (Inference). Finally, there must be a mechanism that translates the fuzzy output commands (decisions) into values which will determine the quality of the multinomial fit (defuzzification).

Fuzzy variables are characterized as "Membership Functions" (MFs). They may partially overlap with each other and associate numerical values with linguistic values via a weight. The same procedure applies to the fuzzy outputs. The inference mechanism contains a Rule Base that makes decisions (activates output Membership Functions) according to the fuzzy inputs (enabled input Membership Functions). They are of the following form:

If *InputX is MF1* ***and*** *InputY is MF2* ***then*** *OutputZ is MF3*

The first step is to determine which fuzzy rules have been enabled according the participating input membership functions. The activation is represented in Eq.2.4.1, where "Rj" is the rule number "j" and "σj" is the function that determines whether the rule has been activated according to the "InputX" and "InputY" (or any other inputs that may exist).

$$R^j : \sigma_j\left(InputX, InputY\right) \qquad \text{Eq.2.4.1}$$

Then we have to calculate the weight for the output membership function(s) for each of the activated rules (for this example the weight of MF3 of OutputZ). This can be implemented with Eq.2.4.2.

$$R^j : \sigma_j\left(InputX, InputY\right) = \min\left(\mu_{MF1}\left(InputX\right), \mu_{MF2}\left(InputY\right)\right) \qquad \text{Eq.2.4.2}$$

The final weight that will be applied to the activated membership function of the fuzzy output is determined by Eq.2.4.3. There, the maximum "ξ_{MF3}" of all the output weights calculated with Eq.2.4.2, which correspond to the specific output membership function is determined.

$$\xi_{MF3} = \max\left(\sigma_{ALL}\right) \qquad \text{Eq.2.4.3}$$

The final step is to defuzzify the output function to produce a numerical value. There are several methods to implement these steps; the most common way though is to determine the final value with a simple calculation of the center of gravity of the surface below the final output membership function, Eq.2.4.4.

$$\widehat{y}_{COA} = \frac{\sum_{v=1}^{No\#\cdot of\cdot MFs} \xi_v \mu_v (InputX, InputY)}{\sum_{v=1}^{No\#\cdot of\cdot MFs} \mu_v (InputX, InputY)} \qquad \text{Eq.2.4.4}$$

3
Design of Evaluator

3.1) Introduction

The morphogenetic neuron's performance depends on the freedom (parameters) that it is given when fitting a multinomial to the numerical data. Thus, the neuron performs the best possible fit regardless if the final outcome is actually over fitting or under fitting. Therefore an evaluation mechanism was necessary in order to determine the quality of the outcome of the recognition procedure and alter the neuron's parameters if necessary (Fig.3.1).

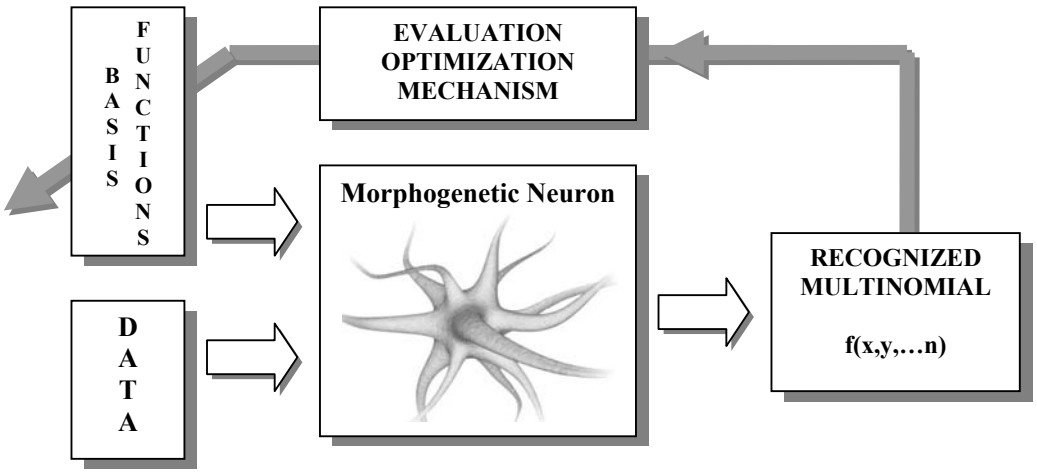
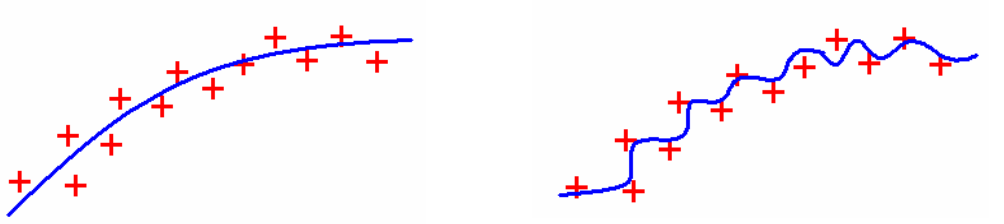


Fig.3.1: Evaluating Mechanism

In this chapter the implementation of such an evaluating operator will be presented. The Evaluator is based on soft-computing and statistic methods.

3.2) Evaluator Definition

The aim of the evaluator is to determine the quality of the fit. Still, even though the concept of evaluating a multinomial fit sounds trivial, in reality, the task of determining the quality of a data fit is complex and laborious. Several decisions have to be made in order to specify the requirements of a good fit. A simple example is displayed in the following figures (Fig.3.2.a & Fig.3.2.b)



a) Low Order Curve Fit *b) High Order Curve Fit*

Fig.3.2: Multinomial Fitting Examples

As it is shown in Fig.3.2.a a least square low order curve is fitted to the data, whereas at Fig.3.2.b a higher power multinomial is used. According to one's needs, the concept of "good fitting" finds new interpretations. Therefore, within this work a definition of the required "good fit" had to be given before the design of the evaluator can take place. Within this work, a good fit was defined as the multinomial result of the morphogenetic neuron that meets the following criteria (prioritized):

i) has the lowest multinomial power
ii) makes use of the fewest basis functions
iii) has minimum mean value of the distances of the data points to the recognized curve

Still there is one exception to these rules, in case the fit of a higher power uses significantly less basis functions than the one of a lower power. Only in this case the fitting of a higher power will be characterized as a better fit.

3.3) Evaluator Design

The evaluating mechanism performs a number or operations on the outcome of the morphogenetic neuron. The evaluation and the optimization of the neuron's performance are implemented in a number of iterative steps and are separated to "Local Level" and "Global Level" operations (Fig.3.3).

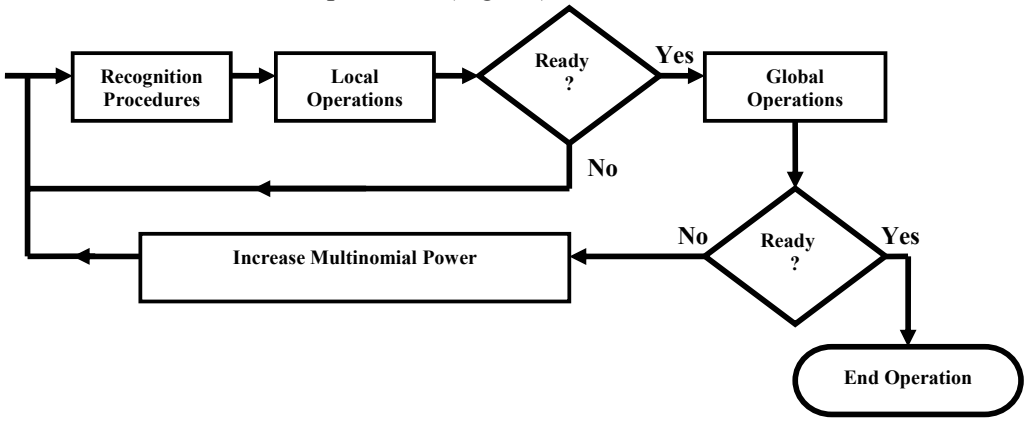


Fig.3.3: Evaluator Flow Diagram

As "Local Level" operations are defined the optimization operations that take place on the same multinomial power recognition iterations. The "Global Level" operations are

the ones that aim to distinguish the best multinomial fit among the resulting multinomials of different powers.

a) Local Level operations

At the local level, two are the main operations for evaluation and optimization. These steps are the elimination of the basis functions that contribute insignificantly to the recognized structure, and the calculation of the mean value of the data point-to-curve distances (Fig.3.4).

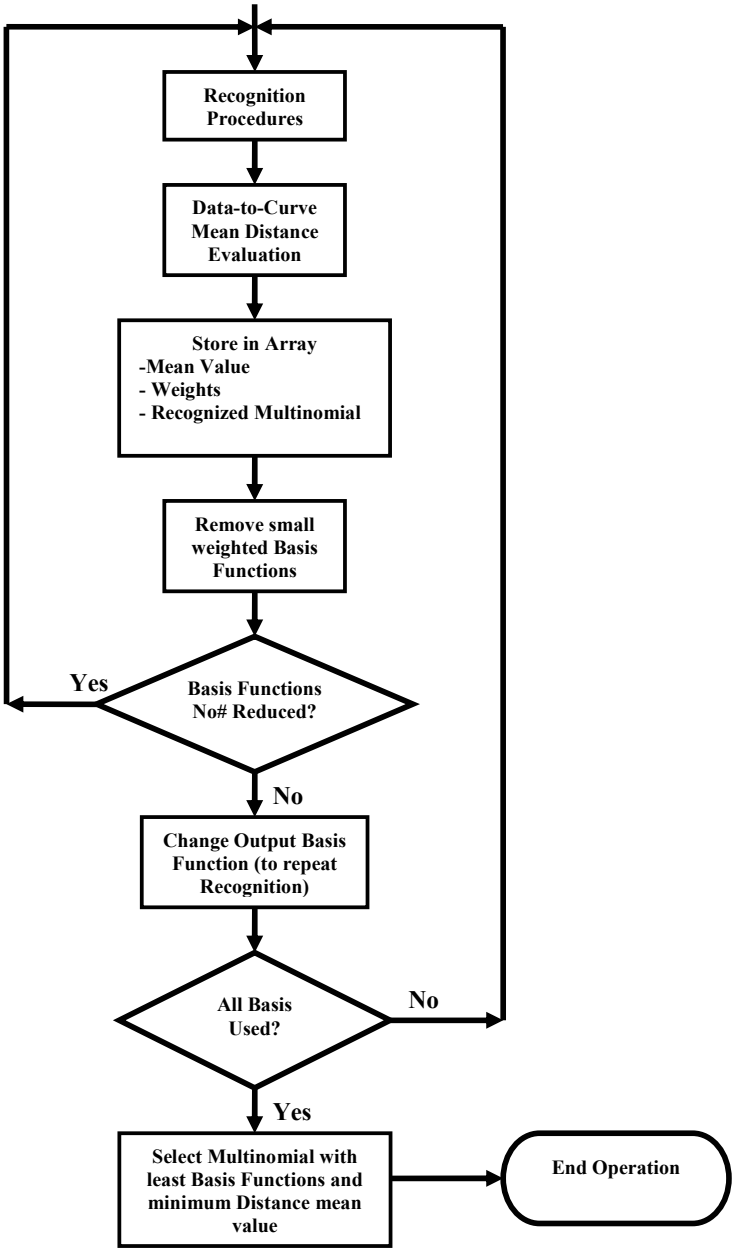
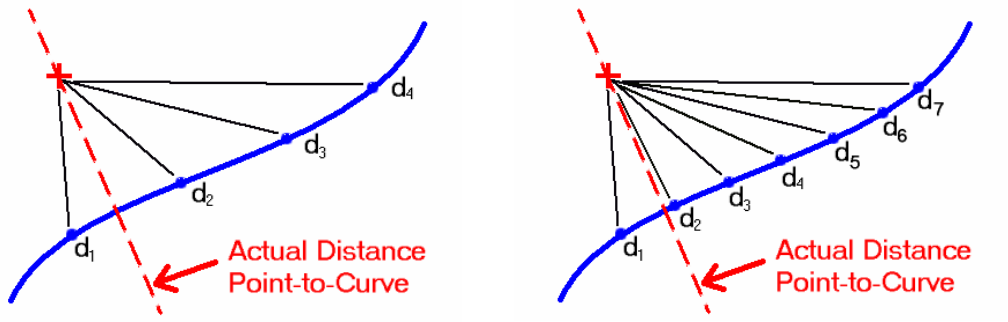


Fig.3.4: Flow Diagram of Optimization on Local Level

As it can be seen in the flow diagram of Fig.3.4 there is a "buffer" array where all the recognized multinomial are stored until the final selection, on local level is made. There is a loop where the insignificant basis functions are eliminated and forces new recognition process with the updated (fewer) basis functions.

The elimination of less contributing basis function occurs in every iteration step in local level. The elimination is based on the value of the corresponding weights of each of the basis function. It was determined experimentally that the basis functions with weights less than the 1/1000 of the maximum weight value can be deleted from the recognized multinomial, improving its performance. It must be taken into consideration that this value was derived from experiments with multinomial curves with relatively small coefficients (up to value of 20). For multinomials with higher coefficients a smaller elimination threshold should be applied (less than 1/1000).

Another operation on the local level is the brute evaluation of the fit. The first step is to determine the minimum distance from each of the data points from the recognized curve. For this reason the coordinates of points belonging to the recognized curve are required. These points must be at least of the same number as the data points that were used for the recognition, and they should be equidistantly distributed on the recognized curve. Then the distance between each of the data points and the points on the curve, is calculated ($Distance_{DATA\text{-}CURVE} = sqrt[(X_{DATA} - X_{CURVE})^2 + (Y_{DATA} - Y_{CURVE})^2]$). For each of the data points, the minimum calculated value is the one that will be used to describe the distance of the point to the curve (Fig.3.5).



a) *b)*

Fig.3.5: Distance Data point to Curve. In 'b' has more points on the recognized curve than 'a', and therefore the estimations of the minimum distances are more precise.

It can be easily understood from Fig.3.5 (a, b), that the selection of the points on the recognized curve plays a significant role for the more precise evaluations of the minimum distances (point "d_2" in Fig.3.5.b is close to the actual value of the minimum distance). After determining all the minimum distances, a mean value operator is used to all the calculated minimum distances to evaluate the mean deviation of the data points from the recognized curve.

A secondary operation that takes place at local level is the experimentation on output basis functions. For the recognition procedure, there is need to define a basis function as an output. The selection of the output function is initially arbitrary. Still, after all optimization steps have been made and the best mean distance value has been

determined, all remaining basis functions (one by one) are used as output functions to investigate possible further improvement of the neuron's results.

When all possible optimizations have occurred and all possible structures have been recognized, the best one needs to be determined among them. All the recognized structures are stored in the "buffer" array. At this point, first the multinomials with the fewer participating basis functions are selected. From this final selection, the multinomial with the smallest point-to-curve distance mean is the one that will be selected to be passed on to the Global level (best local fit).

b) Global Level operations

This is the level where the final decision making and evaluation takes place. This level is also responsible for increasing the allowed multinomial power of the basis functions. Another array is used in global level to house the best multinomial fits of each of the local level results.

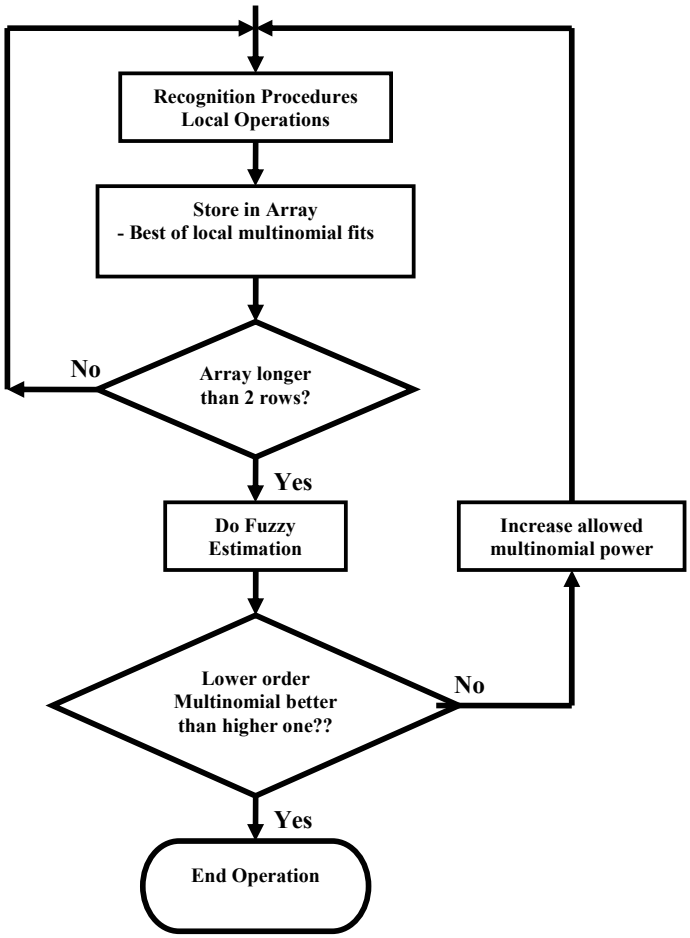


Fig.3.6: Flow Diagram of Multinomial Selection in Global Level

The fuzzy estimator compares the results of two subsequent results, one of a smaller power and one of a higher. Therefore the global array has to have at least two entries in order for the fuzzy estimator to be operative (Fig.3.6). On each global loop where the

fuzzy estimator is active, only the most recent entries are used for the evaluation. If the estimator finds the higher power multinomial as the better fit, then the allowed multinomial power is increased and further recognition is allowed. If the estimator's result is that the lower power fit is better, then the process stops, and the lower power multinomial is the final result of the platform.

3.4) Fuzzy Logic Estimator Design

The estimation rules for the determination of the goodness of a fit depend completely on the process to be recognized and the requirements and understanding of the user. As was presented in Fig.3.2, different multinomial fits can each be selected as the best fit, under certain circumstances. The Fuzzy estimator was used to accommodate two competing effects and the subjectivity present in the definition of a good data fit. For the development of the estimator, the rules that were follower are the ones are also displayed in paragraph 3.2. Therefore, the best fit:

a) has the lowest multinomial power
b) makes use of the fewest basis functions
c) has the minimum mean value of the distances of the data points to the recognized curve

As it was mentioned in paragraph 3.3, the estimator must compare the outcomes of two successive multinomial power recognitions and decide which satisfies better the above rules. The first step for the design of the fuzzy estimator is to identify the number and type of inputs and outputs that are required for the process.

To comply with the desired rules, there must be an input responsible for the number of the basis functions, and an input for the values of the data-to-curve distance means. Also the information about the power of the multinomial must also be imported in the estimator. The comparative nature of the estimator allows the incorporation of knowledge of rule 'a', for the definition of the inputs necessary to describe rules 'b' and 'c'. The comparison is implemented between two successive multinomials, which requires the existence of a lower and a higher power multinomial. This information can be used to define comparison values for the fuzzy estimator. For example the following equation:

$$BasisDifference = Basis\_Function\_No_{High} - Basis\_Function\_No_{Low} \qquad \text{Eq.3.4.1}$$

The numerical value of 'BasisDifference' carries the value of the difference of the number of basis functions that are used for the recognition between two successive multinomials. Since, the order of the subtraction is known, the sign (-, +) of the 'Result' value carries also information on which basis function no is bigger ('-' = Low power multinomial number bigger, '+' = High power multinomial number bigger). A similar operator can be used to define the comparison between the data-to-curve mean values.

The first input of the controller is the "MeanRatio", which represents the ratio of the mean values of the data-to-curve distances of two successive recognized multinomials. The mean ratio is given by the following equation

$$MeanRatio = \frac{Higher\_Power\_Polynomial\_Mean\_Value}{Lower\_Power\_Polynomial\_Mean\_Value} \qquad \text{Eq.3.4.2}$$

It can also be seen in Eq.3.4.2 that the information about the order of the recognized multinomials can also be derived by the final numerical value of "MeanRatio' ('MeanRatio<1' = Lower Power Multinomial Mean Value is bigger, 'MeanRatio>1' = Lower Power Multinomial Mean Value is smaller).

The input "MeanRatio" consists of the following five membership functions as displayed in Fig.3.7:

- ***VeryGood****:* When the higher power recognized multinomial has a significantly smaller point-to-curve distance mean value than the one of the lower power recognized multinomial [0 0.34].
- ***Good****:* When the higher power recognized multinomial has smaller point-to-curve distance mean value than the one of the lower power recognized multinomial [0.2 1].
- ***Equal****:* When the higher power recognized multinomial and the lower power recognized multinomial have approximately the same point-to-curve distance mean value [0.85 1.15].
- ***Bad****:* When the higher power recognized multinomial has a larger point-to-curve distance mean value than the one of the lower power recognized multinomial [1 3].
- ***VeryBad****:* When the higher power recognized multinomial has a significantly larger point-to-curve distance mean value than the one of the lower power recognized multinomial [2 +inf].

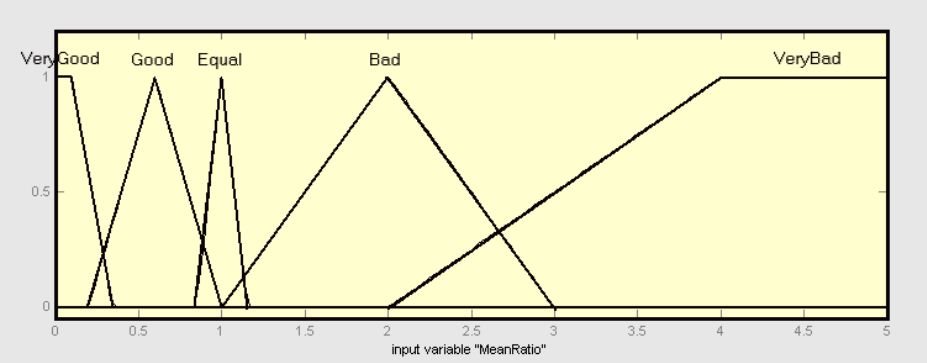


Fig.3.7: 'MeanRatio' Input of Fuzzy Controller Estimator

Then second estimator input is the "BasisDifference", which represents the difference of the number of basis functions that are used in the two successive multinomials (Eq.3.4.1.

The input "BasisDifference" also consists of five membership functions as displayed in Fig.3.8:

- ***LowBasisLotBigger****:* When the number of basis functions used in the lower power recognized multinomial is a lot larger than the number of the basis functions used in the higher one [-inf –4].

- ***LowBasisBigger**:* When the number of basis functions used in the lower power recognized multinomial is larger than the number of the basis functions used in the higher one [-6 -0].
- ***Zero**:* When the number of basis functions used in the lower power recognized multinomial is the same as the basis function number used in the higher one [-0.5 0.5].
- ***LowBasisSmaller**:* When the number of basis functions used in the lower power recognized multinomial is smaller than the number of the basis functions used in the higher one [0 6].
- ***LowBasisLotSmaller**:* When the number of basis functions used in the lower power recognized multinomial is a lot smaller than the number of the basis functions used in the higher one [4 +inf].

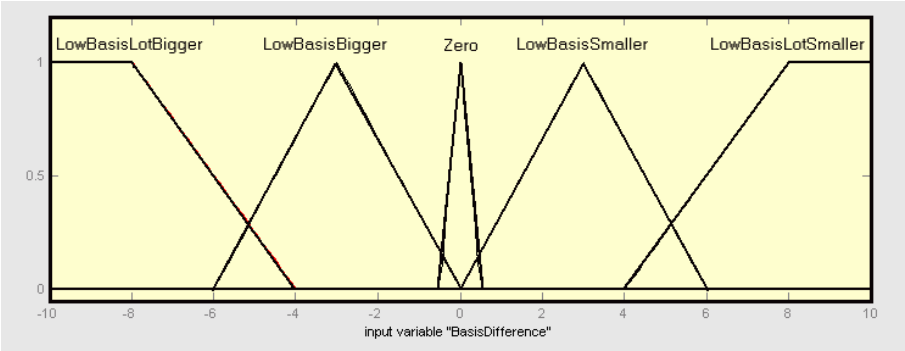


Fig.3.8: 'BasisDifference' Input of Fuzzy Controller Estimator

The estimator is called to make a decision on which of the multinomials is better fitting the numerical data. As it was also mentioned in paragraph 3.3, if the lower power recognized multinomial was found fittest then the process is terminated. It was decided to combine the two requirements. Therefore, the "Evaluation" output of the consists of three membership functions as displayed in Fig.3.9:

- ***Stop:*** When the lower power recognized multinomial is found more fit [-1 0].
- ***Undecided**:* When both recognized multinomials are found equal fitting [-0.2 0.2].
- ***Continue**:* When the higher power recognized multinomial is found more fit [0 1].

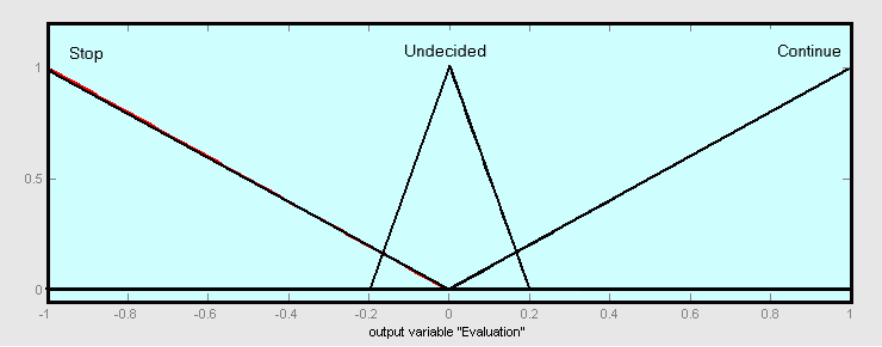


Fig.3.9: 'Evaluation' Output of Fuzzy Controller Estimator

The rules of inference for the estimation of the best fit depend on the rules initially set from the recognition requirements (rules 'a', 'b' and 'c'). In general, if the lower recognized multinomial has lower point-to-curve distance mean value and makes use of

the same or less number of basis functions than the higher one, then the lower recognized multinomial is the best candidate and vise versa. Taking this rule as a base, the following set of fuzzy rules was determined (Table 3.1).

TABLE 3.1 (Fuzzy Rules)

	Mean Ratio				
Basis Difference	**Very Good**	**Good**	**Equal**	**Bad**	**Very Bad**
LowBasLotBigger	Continue	Continue	Continue	Continue	Undecided
LowBasBigger	Continue	Continue	Continue	Undecided	Stop
Zero	Continue	Continue	Undecided	Stop	Stop
LowBasSmaller	Continue	Undecided	Stop	Stop	Stop
LowBasLotSmaller	Undecided	Stop	Stop	Stop	Stop

In the above table the relation between he output and the two inputs is displayed. The corresponding surface of the above table is displayed in the following Fig.3.10. It can easily be understood from the surface, that the behavior is non-linear.

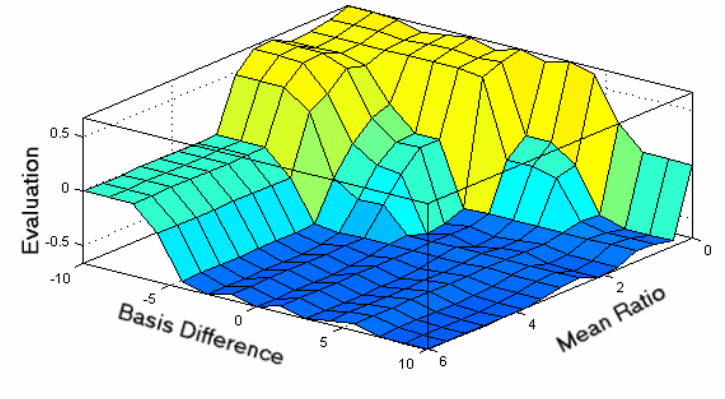


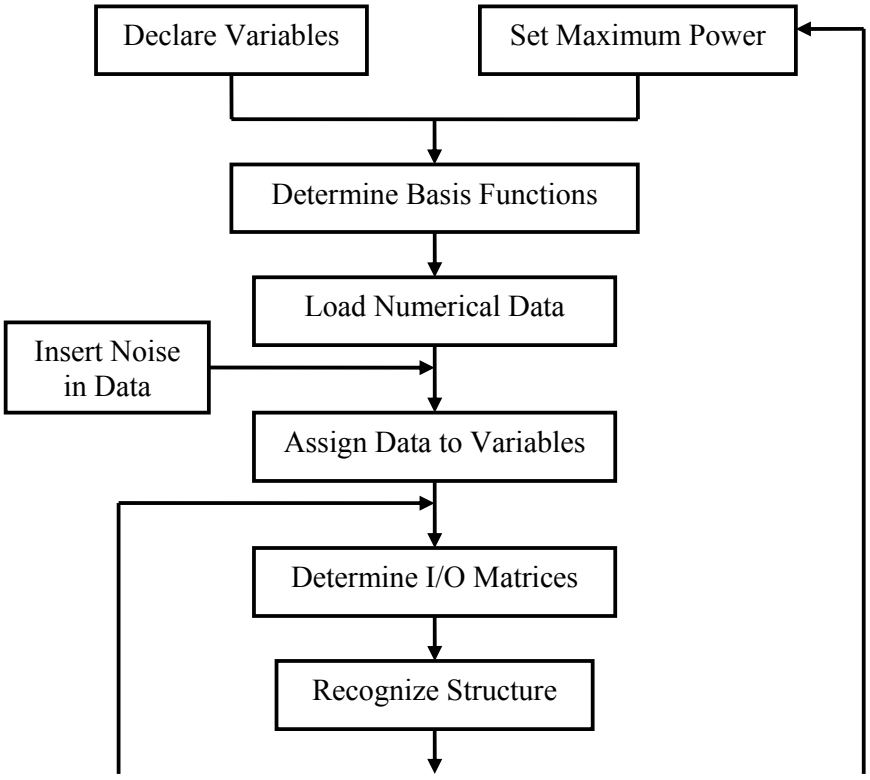
Fig.3.10: Fuzzy estimator's decision surface

4
Simulation Environment

4.1) Introduction

For deriving the output weights (Eq.2.3.5) with the use of the neuron, it is necessary to calculate the 'b' and 'G' matrices. Additionally, a display and evaluation of the neurons results must take place in order to achieve further optimization. Therefore, a platform (additional supporting algorithms) was necessary to provide the ability to produce these 'b' and 'G' matrices, and control and manipulate their parameters before they are entered in the neuron.

This platform as well as the neuron's algorithm was developed in MatLab using its GUI toolbox. At this chapter, the functionality and the order of the steps required to use the developed platform will be presented. More information about the GUI's functionality and controls is presented in Appendix B. The flow diagram of the functionality of the platform is presented in the following Fig.4.1.



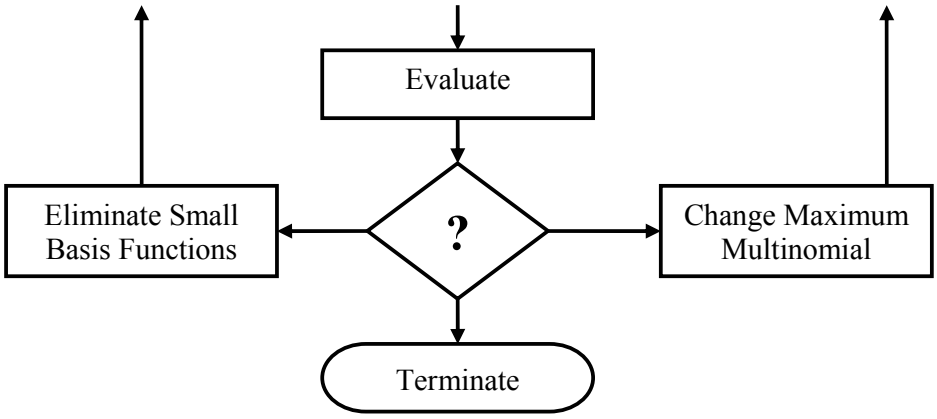


Fig.4.1: Flow Diagram of Morphogenetic Neuron's Platform

The functionality of the elements (or groups of elements) of the flow diagram of Fig.4.1 is explained in the following paragraphs.

4.2) Simulation Platform

The simulation platform consists of a number of supporting algorithms of the neuron's main algorithm. These algorithms ensure the correct implementation of the recognition process, evaluation and resetting of the initial parameters of the procedure. Most of the support algorithms exist to provide the neuron with its basic elements (b, G matrices), whereas the rest exist for optimization purposes.

a) Selection of variables.

This is the initial part of the platform where declaration of the names and number of the participating variables takes place. These are the variables that will be used to construct the recognized multinomial. The number of the variables used depends on the number of numerical data sets that are used for the recognition of the structure. Within in this project the majority of structures to be recognized are two-dimensional, and therefore only two variables are used. The name of the variables can be anything, but for easiness the "x" and "y" symbols are used in the tests of chapter 5.

b) Multinomial Power –Basis Functions Construction

The power of the multinomial is the maximum power that the basis functions of the recognized multinomial are allowed to have. Therefore, the value of the maximum power is also responsible for the form and number of the basis functions. For example, assume it is allowed to have a maximum power of "3" and there are two variables (x, y) participating in the construction of the basis functions. Then the resulting basis functions to meet this criteria would be: 1, x, y, xy, x^2, y^2, xy^2,x^2y, x^3, y^3. From the example it can be easily be understood that the higher the allowed power is, the more the initial participating basis functions that can be constructed.

The algorithm first computes all the possible combinations among the number of the variables up to the maximum allowed power and stores them in a matrix. The formula that computes the maximum number of combinations is displayed in Eq.4.2.1.

$$Number\_Of\_Combinations = Power * (Power + 1)^{No\_Of\_Variables-1} \qquad \text{Eq.4.2.1}$$

For instance, for the example of the '2' variables with a maximum allowed power of '3', the algorithm would produce twelve combinations (Fig.4.2.a), from which only nine are valid and selected (Fig.4.2.b). Each column of Fig.4.2 corresponds to one of the variables and each of the rows is the possible combinations of the variables with respect to the maximum allowed power (Fig.4.2.c).

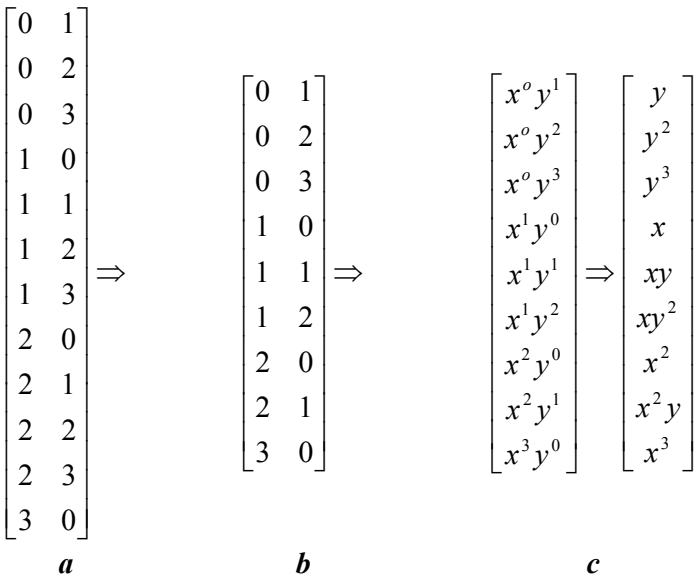


Fig.4.2: Power combinations for Basis Function construction

As it can be seen in the final constructed basis functions there is no constant number basis function present. Therefore special care must be taken in order to add a constant number to participate in the recognition of the structure. This step is necessary to compensate for any possible need for a constant (variable less) element in the recognized multinomial. The constant value that is added by the platform is the number '1', but it can be any other constant value.

c) Data Importing

This part is where the numerical data are inserted into the neuron. The data are stored into columns, one for each of the participating variables. These columns must exist in a text file, separating the columns with any separating character (space, tab). Special care must be taken about the order that the participating variables are declared, as their order will determine the column with numerical data that they will be linked to. This means that the first declared variable will be linked with the numerical data set of the first column, the second variable to the second column and so on.

As soon as the data has been imported, they are plotted on coordinate system on the display. Since within this project the research is based on two-dimensional structures the plot is restricted to two dimensions. It is easily understood that the number of numerical data must be higher or equal to the number of selected basis functions to avoid over fitting situations.

d) Construction of Initial I/O Matrix-Vector

At this phase, the construction of the training data matrix as well as the output vector selection is performed. The first step in this procedure is to calculate the numerical value of each of the basis functions. It has already been mentioned that various combinations of the participating variables construct the basis functions. Since the numerical value of those variables is already available form the "Data Importing" part, it is easy to calculate the exact value of the basis function for each set of imported data. Therefore, the training data matrix is constructed, which has a column containing the numerical values of each of the basis functions Fig.4.3. The order that the basis functions are used to construct the data matrix is not important, as long as the same order is kept when reconstruction the recognized multinomial.

$$DataMatrix_{nxm} = \begin{matrix} 1 & x & y & x\,y & \cdots & x^f y^g \end{matrix} \begin{bmatrix} 1 & a_1 & b_1 & a_1 b_1 & \cdots & a_1^f b_1^g \\ 1 & a_2 & b_2 & a_2 b_2 & \cdots & a_2^f b_2^g \\ 1 & a_3 & b_3 & a_3 b_3 & \cdots & a_3^f b_3^g \\ 1 & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_n & b_n & a_n b_n & \cdots & a_n^f b_n^g \end{bmatrix}$$

Fig.4.3: Implementation of the data matrix

For the neuron's algorithm to perform, an output vector is required. To select this output vector, the optimum situation would be to have knowledge and select the basis function that is most likely to participate on the final recognized structure. Unfortunately, from the numerical data, no safe assumption can be made on which of the basis functions will be participating in the final recognized structure. For this reason, an arbitrary initial basis function selection is made (usually the first basis function of the combinations list). This means that the final *DataMatrix* that is used in the neuron is similar to the one of Fig.4.3, but without the column of the basis function that was chosen as an output.

The arbitrary choice of the output function also makes necessary the further refinement of the recognized multinomial. The refinement of the multinomial is required in order to eliminate any undesired basis functions that were forced to participate in the initial multinomial, due to their use as an output.

e) Training and Structure Recognition

This is the step where the mathematical computations take place, as they are described in paragraph 2.3. The resulting weights are coupled with their corresponding basis

functions (Fig.4.4) and used to construct the symbolic multinomial representation of the recognized structure.

$$Output\_bf = \begin{bmatrix} bf_1 & bf_2 & bf_3 & \cdots & bf_{m-1} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{m-1} \end{bmatrix}$$

Fig.4.4: Coupling of the calculated weights with their corresponding basis functions. Where "Output_bf" is the symbolic representation of the selected output basis function, "bf" is the symbolic representation of the DataMatrix basis functions and "w" are the numerical values of the calculated basis functions' weights.

As it was mentioned at paragraph 4.2.d, for the correct construction of the resulting multinomial, the order that the symbolic basis functions in Fig.4.4 must be the same as the order that these basis functions were imported in the *DataMatrix*.

f) Fitting Optimization

In the initial recognition step of structures in the numerical data, the neuron is trying to fit all the available basis functions. This means that all basis functions are assigned a weight, even when this should not be the case (e.g. fitting a circle with a 3rd order multinomial). These basis functions are usually easy to identify, as they are usually being assigned respectively small weights and contribute slightly to the general behavior of the recognized multinomial. An example of fitting optimization due to elimination of undesired basis functions is displayed in Fig.4.5.

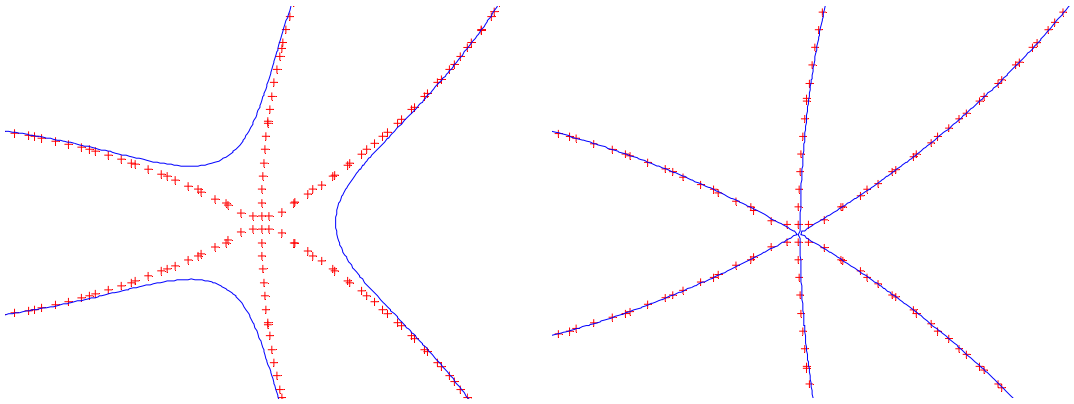


Fig.4.5: Improvement of recognition by basis function elimination

All of the basis functions are forced to be fitted in the initial recognition step, because there are always errors introduced in the data, either from noise or rounding operations.

Another, optimization that could be applied on the recognized multinomial, is the elimination of common variables. This is only applicable to multinomials that do not contain a constant basis function, and can be rewritten in the form of:

$$Multinomial = Variable * Root\_Of\_Multinomial = 0$$

At these cases, when the recognized curve passes through the beginning of the coordinate system, the neuron is incapable to distinguish the existence of additional (zero point curves) in the recognized multinomial. Though the performance of the recognized curve does not change, it is wise to extinguish any extra non-contributing terms.

Another parameter that influences the performance of the neuron and has already been mentioned several times is the selection of the output basis function. Even though after a number of iterations the unused basis function will be eliminated, a selection of a basis function as an output that is likely to take part in the final multinomial will lead to the desired result faster. An example of how the selection of the output function influences the initial result is displayed in the following Fig.4.6.

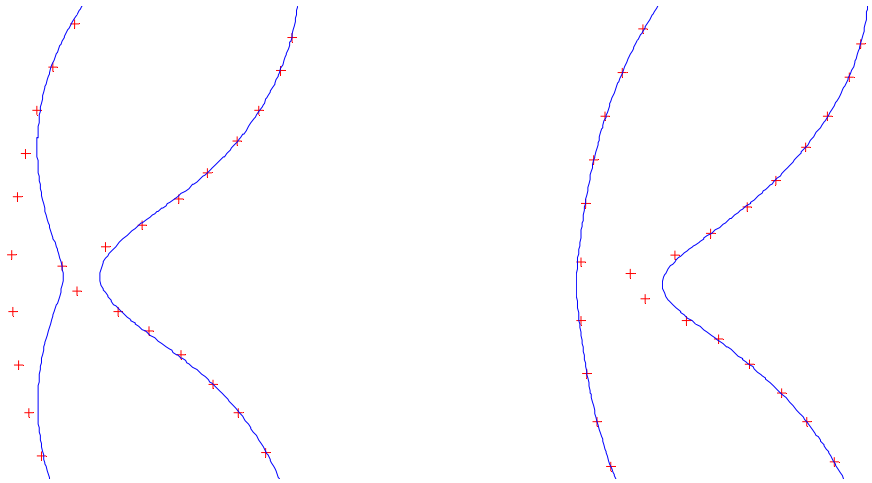


Fig.4.6: Improvement of recognition by output basis function selection. The crosses are the numerical data points; the line is the detected multinomial.

In Fig.4.6, points that belong to the difolium curve are inserted in the neuron for recognition. The difolium curve has no constant factor. Therefore if the neuron is forced to use the constant factor by selecting it as an output basis function, then the result will be the one on the left of Fig.4.6. But if a non constant basis function is selected as output, then the resulting multinomial resembles the one presented on the right of Fig.4.6.

g) Fitting Evaluation

Since no a-priori knowledge is available about the power that would efficiently fit all the numerical data, a mechanism must be developed to develop and evaluate several multinomials of different powers and choose the best one.

The mechanism is starts the fitting from the smallest possible power. The lower multinomial power curves that can be recognized are of the 2^{nd} order (no reason to use this method to fit a line into data). An evaluation mechanism is present and with the aid of a fuzzy logic estimator, a decision is make on whether the recognition procedure

should continue or terminated. If the procedure is continued, the allowed order of multinomial to be recognized is incremented by one and the recognition procedure is repeated. If the procedure is terminated, then the best multinomial fit has occurred in the previous lower order, and that recognized multinomial curve is the final result of the recognition. The evaluation and estimation takes place according the procedures that were discussed in chapter 3.

e) Importing Noise

In any process, the "clarity" of the data samples is of the play a significant role for the correct estimation and control. Still, in real life situations, noise is always present in the data samples. Therefore, it is important to evaluate the performance of the neuron with data that contain noise. For this reason noise generators were developed to introduce noise in the numerical data, according the demands of the user. The platform has the ability to introduce Uniform noise to the numerical data, with respect to the desired noise level.

To introduce uniform noise to the numerical data, an equal set of normally distributed random numbers are generated. This set has 'zero' mean value, a variance of 'one' and deviation of 'one'.

Then this set is scaled to the desired noise value that can be imported by the user through the platform. The formula for the generation of uniform noise is given by Eq.4.2.2.

$$Noise(n) = \begin{cases} \dfrac{1}{2\sigma} & ,|n| < \sigma \\ 0 & ,Else \end{cases} \qquad \text{Eq.4.2.2}$$

4.3) Graphic User Interface

The different elements of the platform are used to develop the Graphic User Interface (GUI) that is displayed in Fig.4.7. The GUI, like the rest of the algorithm/platform, is constructed in MatLab using the GUI Development Toolbox. Each element of the algorithm has been separated in the GUI by a frame and contains its own control objects.

The GUI was developed to provide the ability for the user to interact with the whole recognition process and alter its parameters, as well as to display the final recognition products. The GUI and the recognition process have two modes, the Autonomous and the Manual. Within both the Manual and Autonomous mode the GUI the user can:

a) Select and load the text file containing the Numerical Data of the structures that is to be recognized. The list of files containing numerical data is displayed in a list box.
b) Define the number and the name of the variables that will be used for the construction of the basis functions.

c) Select the option of autonomous of manual recognition. In "Autonomous" mode the fuzzy logic estimator is the one that alters and manipulates the parameters of the morphogenetic neuron and makes decision for the best multinomial fit. In "Manual" mode, the user is responsible for the control of the recognition process.
d) Insert uniform noise in the numerical data. The user can apply noise on either or both directions of the coordinate system. The user can also select the boundaries of the generated noise values.
e) Inspect the position of the numerical values on plot, as well as the recognized multinomial.
f) Zoom in and out in the plotted structures for inspection. Reset the plot and export the plot image to a file.

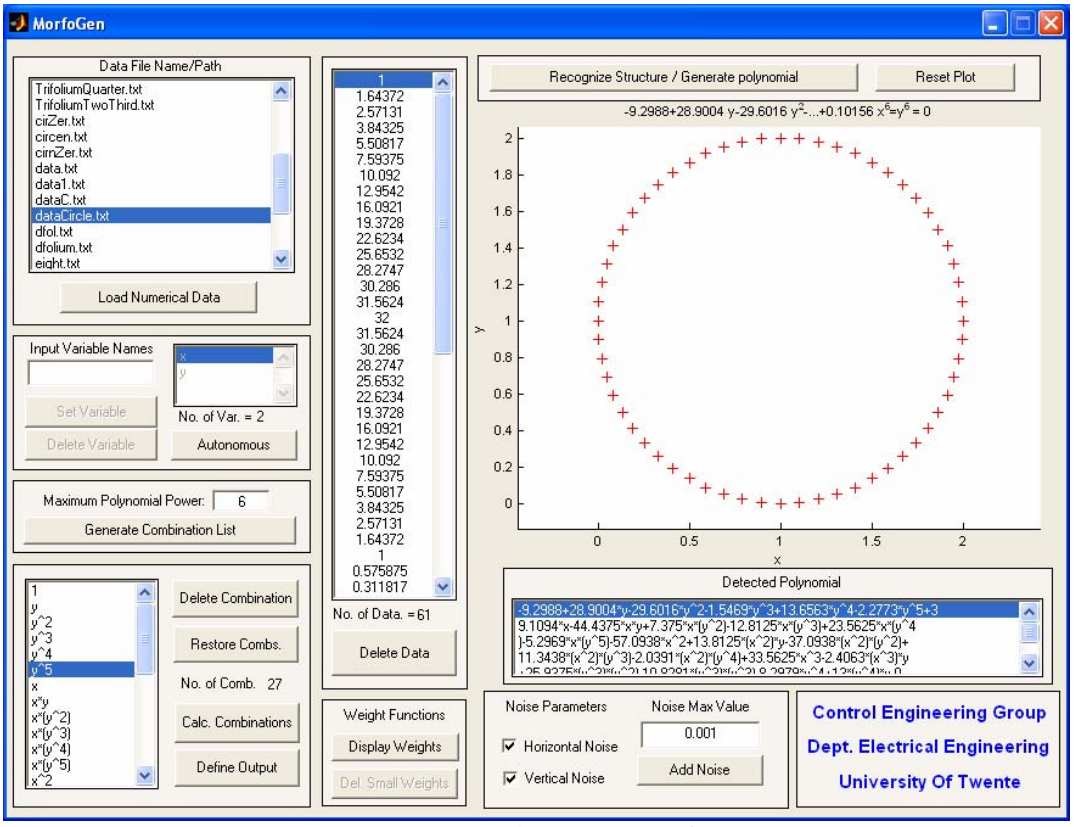


Fig.4.7: Graphic User Interface

Within only the Manual mode the GUI the user can:

i) Select the maximum allowed multinomial power for the construction of the basis functions.
ii) Manually, delete any undesired basis functions and define the basis function that is to be used as an output.
iii) Display the weights of the used basis functions and decide which ones need to be eliminated.
iv) At all times display the numerical values of the basis functions and of the variables.

All intermediate and final results are displayed graphically in the axes (plot) object of the GUI in a graphic form. The same results are presented in their symbolic form in the "Detected Multinomial" frame. More extensive information on the parts and the use of the GUI can be found in Appendix B.

5
Tests & Results

5.1) Introduction

To estimate the performance of the neuron for the discovery of underlying structures in numerical data, a number of tests were conducted. The numerical data that were used in these tests are groups of points belonging to known curves (circle, ellipse, difolium, trifolium, etc). The quality of the neuron's results will be estimated according to its curve fitting ability, even when there is an amount of noise (Uniform, Gaussian) in the numerical data.

In the following paragraphs, a number of tests will be displayed as well as a small numerical example to demonstrate the functionality of the neuron's algorithm. For each of the tests that are conducted, plots of the numerical data and the fitted curve are displayed as well as the initial and recognized multinomial function. Additional tests, and curve fittings are displayed in Appendix A.

5.2) Simple Numerical Example of Circle Recognition

For this simple numerical example a circle is used with a curve function of "$x^2+y^2=1$". A number of points belonging on this circle are selected to be use as the input training data of the morphogenetic neuron. These data are displayed in Table 5.1 and presented plotted in Fig.5.1. Let's assume that we have the following data on an x-y orthogonal coordinate system.

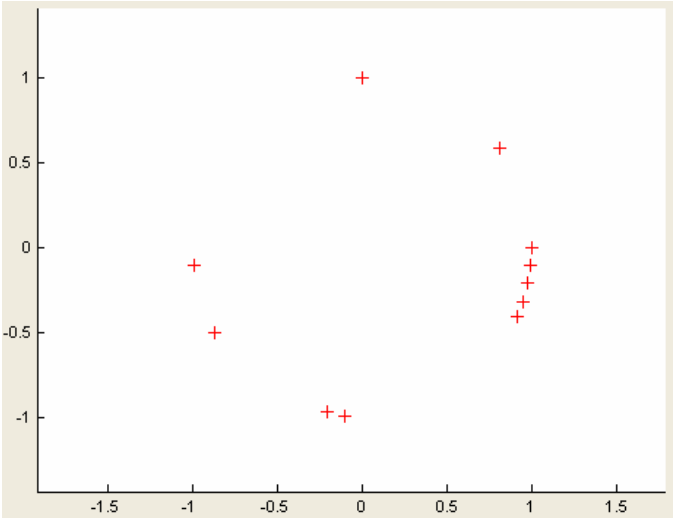


Fig.5.1: Numerical Data Plot

TABLE 5.1 (Numerical Data)

x	0.9511	0.8090	0.0	-0.9945	-0.8760	-0.2097	-0.1045	0.9135	0.9781	0.9945	1.000
y	-0.3190	0.5878	1.0	-0.1045	-0.5000	-0.9701	-0.9945	-0.4077	-0.2079	-0.1045	0.000

Since the numerical data have been obtained, there is need to define the maximum power of the multinomial that is to be fitted to the data. It is already known that the circle is of 2^{nd} power multinomial, and to help make the example easier to understand it was selected directly. The responsibility of multinomial power selection falls on the performance estimator which is not used in this example. Since the selected power is '2' the resulted basis functions for the two input variables are: 1, x, y, x*y, x^2 and y^2. The next step it to calculate the numerical values of those basis functions for every pair of input (x, y) values. This action will result to the following initial DataMatrix that is displayed in Table 5.2.

TABLE 5.2 (Data Matrix)

1	**x**	**y**	**x*y**	**x^2**	**y^2**
1	0.9511	-0.3190	-0.3034	0.9046	0.1018
1	0.8090	0.5878	0.4755	0.6545	0.3455
1	0	1	0	0	1
1	-0.9945	-0.1045	0.1039	0.9890	0.0109
1	-0.8760	-0.5000	0.438	0.7674	0.25
1	-0.2097	-0.9701	0.2034	0.0440	0.9411
1	-0.1045	-0.9945	0.1039	0.0109	0.9890
1	0.9135	-0.4077	-0.3724	0.8345	0.1662
1	0.9781	-0.2079	-0.2033	0.9567	0.0432
1	0.9945	-0.1045	-0.1039	0.9890	0.0109
1	1	0	0	1	0

One of the basis functions has to be selected as the output of the multinomial function. For this example the basis function of '1' was selected, and therefore the "f" vector of Eq.2.2 is equal to the first column of the matrix of Table 5.2. The actual "DataMatrix" that is going to be used in the algorithm is the rest of the columns of Table 5.2.

$$DataMatrix = \begin{bmatrix} 0.9511 & -0.3190 & -0.3034 & 0.9046 & 0.1018 \\ 0.8090 & 0.5878 & 0.4755 & 0.6545 & 0.3455 \\ 0 & 1 & 0 & 0 & 1 \\ -0.9945 & -0.1045 & 0.1039 & 0.9890 & 0.0109 \\ -0.8760 & -0.5000 & 0.438 & 0.7674 & 0.25 \\ -0.2097 & -0.9701 & 0.2034 & 0.0440 & 0.9411 \\ -0.1045 & -0.9945 & 0.1039 & 0.0109 & 0.9890 \\ 0.9135 & -0.4077 & -0.3724 & 0.8345 & 0.1662 \\ 0.9781 & -0.2079 & -0.2033 & 0.9567 & 0.0432 \\ 0.9945 & -0.1045 & -0.1039 & 0.9890 & 0.0109 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad f = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Since these matrices have been determined, then the rest of the procedure is simple application of the equations of paragraph 2.3. Therefore the calculation of the "G" matrix results to the following:

$$G = DataMatrix^T * DataMatrix = \begin{bmatrix} 7.1506 & 0.3417 & -1.0868 & 3.4053 & 0.0507 \\ 0.3417 & 3.8587 & 0.0507 & -1.0869 & -0.9299 \\ -1.0868 & 0.0507 & 0.7638 & -01223 & 0.4664 \\ 3.4053 & -1.0869 & -01223 & 6.4055 & 0.7638 \\ 0.0507 & -0.9299 & 0.4664 & 0.7638 & 3.0858 \end{bmatrix}$$

Then, if "G" matrix is invertible (in this case it is), the calculation of the "G^{-1}" is required and results to:

$$G^{-1} = \begin{bmatrix} 0.2765 & -0.0846 & 0.4094 & -0.1469 & -0.0556 \\ -0.0846 & 0.3180 & -0.1930 & 0.0826 & 0.1059 \\ 0.4094 & -0.1930 & 2.0836 & -0.1704 & -0.3377 \\ -0.1469 & 0.0826 & -0.1704 & 0.2459 & -0.0078 \\ -0.0556 & 0.1059 & -0.3377 & -0.0078 & 0.4099 \end{bmatrix}$$

From Eq.2.3 the "b" matrix was defined as the following, and results to:

$$b = DataMatrix^T * f = \begin{bmatrix} 3.4615 \\ -2.0204 \\ 0.3417 \\ 7.1506 \\ 3.8586 \end{bmatrix}$$

Finally, the weights that need to be applied to the basis functions to give "birth" to the recognized multinomial; can be calculated from Eq.2.5 and result to:

$$Weights = G^{-1} * b = \begin{bmatrix} 0.0031 \\ -0.0017 \\ -0.0019 \\ 0.9946 \\ 1.0040 \end{bmatrix}$$

The resulting weights are applied to the all the basis functions except the one that was selected for output. Therefore the resulting recognized multinomial is the following:

$$\mathbf{0.0031*x-0.0017*y-0.0019*x*y+0.9946*x^2+1.0040*y^2=1}$$

It can easily be seen that the recognized multinomial is very similar to the original ($x^2+y^2=1$) from which the numerical data were derived. The plotted multinomial is

displayed in Fig.5.2 and is represented by the line, whereas the crosses represent the used numerical data.

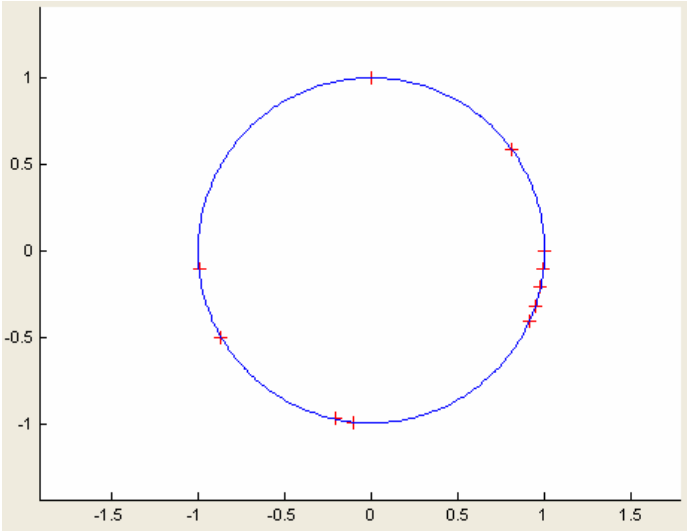


Fig.5.2: Recognized structure

5.3) Structure Recognition Tests

Several tests were performed in order to evaluate the performance of the neuron and its supporting algorithms. In this paragraph, only a few of the tests will be presented in order to demonstrate the performance of the system under certain common/special situations. More recognition examples are displayed in Appendix A.

In the first example (Eight) an extensive description of the several intermediate possible steps until the final recognized multinomial will be given. At the rest of the tests, as well as the ones that are presented in Appendix A, only the important information will be presented.

a) Eight Curve

The eight curve has the Cartesian equation of $x^4=a^2(x^2-y^2)$, and it is represented in Fig.5.3. For this test the value of 'a' was set equal to '2'. Data points that belong on this curve and are displayed as red crosses, were fed in the neuron for recognition.

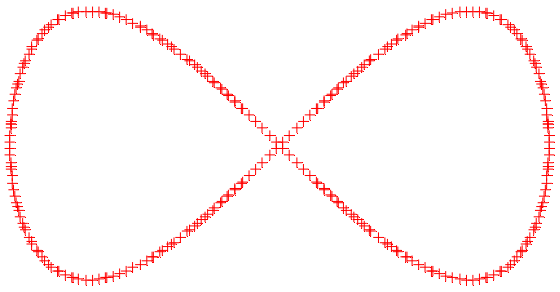


Fig.5.3: Eight Curve [$x^4=a^2(x^2-y^2)$]

From the equation of eight curve, it can be seen that the curve passes from the center of the coordinate system. As it was mentioned in previous paragraphs the system begins its recognition with the smallest possible allowed power. Therefore the power is set to '1', and the allowed basis functions are:

1st Power Basis Functions: 1, x, y

Selecting the "1" basis function as an output, the following recognized multinomial equation (Eq.5.3.1) is resulted, which is a line almost identical with the 'x' axis.

$$0.0022953y + 6.1138e^{-17}x = 1 \qquad \text{Eq.5.3.1}$$

Then the maximum allowed power is raised to '2' and the allowed basis functions are:

2nd Power Basis Functions: 1, x, x^2, xy, y, y^2

Again, selecting the "1" basis function as an output, the following recognized multinomial equation (Eq.5.3.2) is resulted.

$$0.9279y^2 + 0.1908x^2 = 1 \qquad \text{Eq.5.3.2}$$

Plotting Eq.5.3.2 on the same coordinate system as the numerical data, produce the image of Fig.5.4. The blue line represents the recognized multinomial, which represents an ellipse.

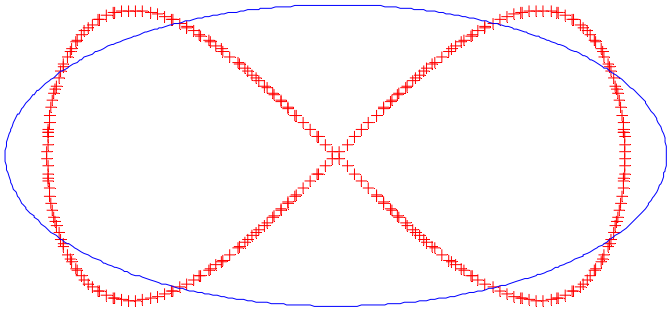


Fig.5.4: Recognized 2nd order curve (blue line)

It is obvious by the display of Fig.5.4, that there is an under fit, as the ellipse doe not pass from all the data points of the eight curve. The maximum allowed power is once more raised to '3' and the allowed basis functions are:

3rd Power Basis Functions: 1, x, x^2, x^3, y, xy, xy^2, y^2, x^2y, y^3

Again, selecting the "1" basis function as an output, the following recognized multinomial equation (Eq.5.3.3) is resulted.

$$\begin{aligned} &0.0024y + 0.93y^2 - 0.0055y^3 + 1e^{-15}x - 1e^{-17}xy - 1e^{-15}xy^2 + \ldots \\ &\ldots + 0.19x^2 + 0.001x^2y - 4e^{-16}x^3 = 1 \end{aligned} \qquad \text{Eq.5.3.3}$$

Eq.5.3.3 after optimization resulted to Eq.5.3.4.

$$0.9279y^2 + 0.1908x^2 = 1 \qquad \text{Eq.5.3.4}$$

Plotting Eq.5.3.4 on the same coordinate system as the numerical data, produce the image of Fig.5.5. The blue line represents the recognized multinomial, which represents an ellipse.

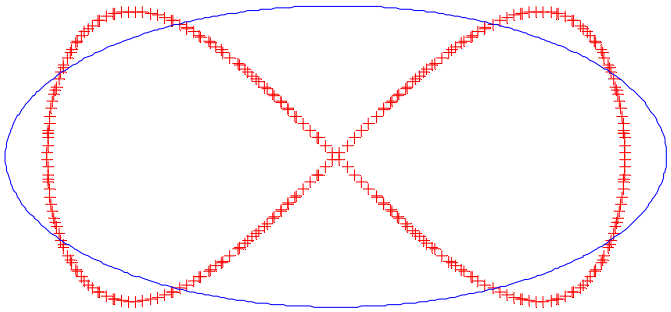


Fig.5.5: Recognized curve with maximum allowed multinomial power '3' (blue line)

Even though the maximum allowed power of the multinomial was '3', the neuron has recognized an ellipse as the underlying structure in the data. Still, there is an obvious under fitting of the curve towards the numerical data. The maximum allowed power is increased to '4' and the resulting basis functions are:

4^{th} Power Basis Functions: 1, x, x^2, x^3, x^4, y, xy, x^2y, x^3y, y^2, xy^2, x^2y^2, y^3, xy^3, y^4

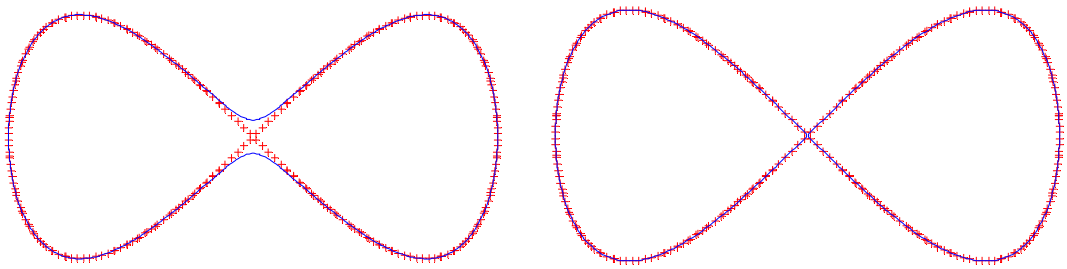
Again, selecting the "1" basis function as an output, the following recognized multinomial equation (Eq.5.3.5) is resulted.

$$\begin{aligned} &0.0005y + 53.31y^2 - 0.0013y^3 - 2.2022y^4 + 5e^{-14}x - 2.6e^{-14}xy + \ldots \\ &\ldots + 1.2e^{-14}xy^2 + 7.6e^{-15}xy^3 - 49.6x^2 + 0.0002x^2y - 0.44x^2y^2 + \ldots \\ &\ldots + 1.4e^{-14}x^3 + 7.5e^{-15}x^3y + 12.5x^4 = 1 \end{aligned} \qquad \text{Eq.5.3.5}$$

The multinomial of Eq.5.3.5 is displayed in the following Fig.5.6.a. As it can be seen that the curve fitting is becoming better but still the resulted multinomial does not resemble the equation of the eight curve. At this point the recognized multinomial curve can be locally optimized by eliminating the basis function that are not contributing significantly to the resulted outcome (small weighted basis functions). After optimization, the constant basis function was eliminated as well as a number of other basis functions and the resulted multinomial has the form of Eq.5.3.6.

$$0.2501x^4 + 1.0002y^2 = x^2 \qquad \text{Eq.5.3.6}$$

Plotting Eq.5.3.6 on the same coordinate system as the numerical data, produce the image of Fig.5.6.b.



(a) ***(b)***

Fig.5.6: Recognized curve with maximum allowed multinomial power '4' (blue line)

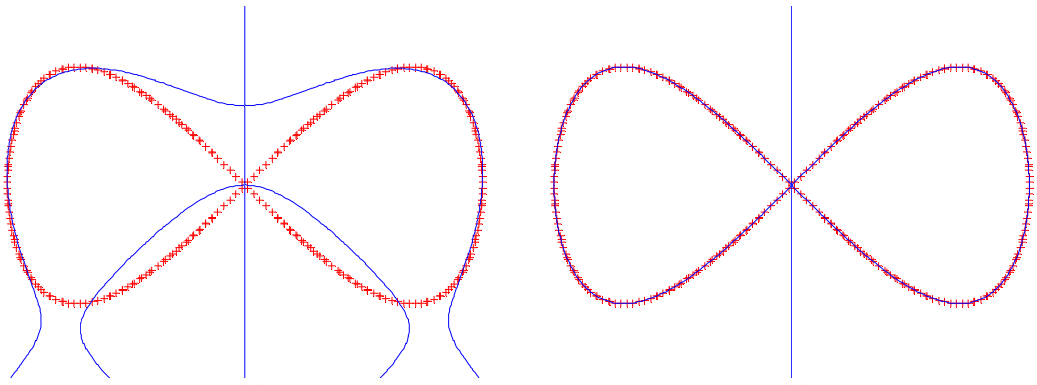
It can be seen in Fig.5.6.b that the fitting is quite efficient, and from Eq.5.3.6 we can derive the equation of the eight curve $x^4=2^2(x^2-y^2)$. Even though the multinomial fit of the 4rt order multinomial seems to be sufficient enough, we continue to investigate for higher multinomial power fitting. This action is required, as there is no a-priori knowledge about the optimum power value, and therefore is has no way to know whether a higher will not produce a better result. Therefore, the maximum allowed power is raised to '5' and the resulting basis functions are:

5^{th} Power Basis Functions: 1, x, x^2, x^3, x^4, x^5, y, xy, x^2y, x^3y, x^4y, y^2, xy^2, x^2y^2, x^3y^2, y^3, xy^3, x^2y^3, y^4, xy^4, y^5

Again, selecting the "1" basis function as an output, the recognized multinomial equation that was resulted is displayed in Fig.5.7.a. As it can be seen that the quality of the curve fitting has not improved and it seems to be over fitting on the data. After performing local optimization of the multinomial, Eq.5.3.7 is obtained.

$$0.2501x^5 + 1.0002y^2x = x^3 \qquad \text{Eq.5.3.7}$$

Plotting Eq.5.3.7 on the same coordinate system as the numerical data, produce the image of Fig.5.7.b.



(a) ***(b)***

Fig.5.7: Recognized curve with maximum allowed multinomial power '5' (blue line)

It can be seen in Fig.5.7.b that the fitting is quite efficient, and it is obvious that there is an over fit on the numerical data. From Eq.5.3.7 we can derive the equation of the eight curve $x^4=2^2(x^2-y^2)$ and a line x=0 which is actually what is presented in Fig.5.7.b. This is evidence of over fitting and it is expected since the neuron has more freedom in

fitting multinomials. When allowing higher order multinomials to be fitted to data belonging to curves of lower order, it is expected to have extra structures that are there to compensate for the excess of freedom. Observing the final result of the 5^{th} order fit it is easy to understand that the result is a 4^{th} order curve (Eight curve) and a 1^{st} order line summing up to 5^{th} order. If we allowed a 6^{th} order fit the result would again be a 4^{th} order curve and a 2^{nd} or two 1^{st} order curves (depending on the symmetry) in order for the final result to be a 6^{th} order curve.

b) Eight Curve with Noise

At this recognition test the population of the numerical data of the eight curve is reduced to half. Additionally, uniform noise has also been added to the data. For different values of noise in the numerical data the following results have been obtained.

For noise magnitude of '0.05' the best fit is obtained with a fourth order multinomial and the recognized structure is presented in the following Fig.5.8.

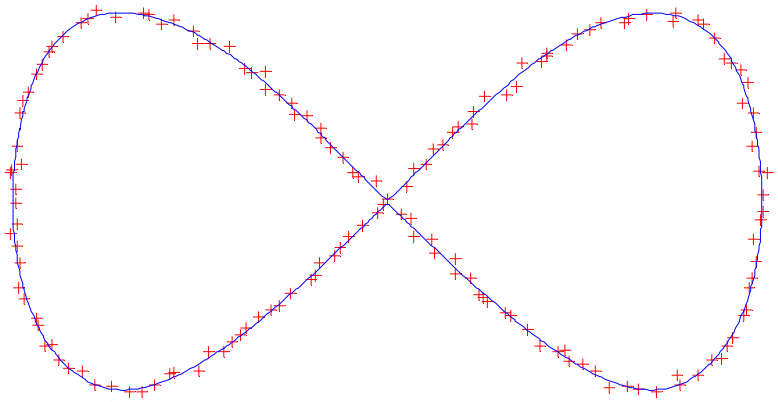


Fig.5.8: 4^{th} order recognized curve for 0.05 noise in data (blue line)

The resulted multinomial of the recognition is the following, where the original basis functions of the eight curve appear in bold.

$$0.027y\mathbf{-3.9y^2}-0.1y^3-0.126y^4-0.007xy^2-0.014xy^3\mathbf{+3.98x^2}+0.0179x^2y+0.055x^2y^2=\mathbf{x^4}$$

It can be seen for the above recognized multinomial, that the dominant (higher weighted) basis functions (y^2, x^2 and x^4) are the ones actually belong to the initial eight curve ($x^4=2^2(x^2-y^2) = 4x^2-4y^2$) and carry approximately the same weights as the ones of the initial curve ($3.9y^2 \approx 4y^2$, $3.98x^2 \approx 4x^2$ and $x^4=x^4$).

The noise in the numerical data of the eight curve is increased by '0.05'. For noise magnitude of '0.1' the best fit is obtained with a fourth order multinomial and the recognized structure is presented in the following Fig.5.9.

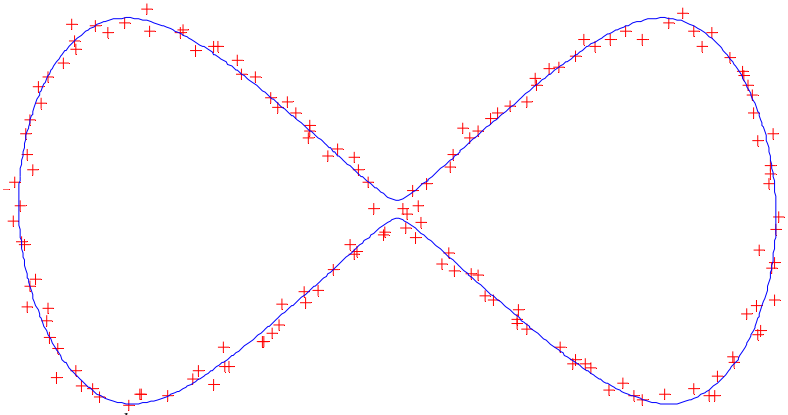


Fig.5.9: 4th order recognized curve for 0.1 noise in data (blue line)

The resulted multinomial of the recognition is the following, where the original basis functions of the eight curve appear in bold.

$$0.015x^3y + \mathbf{0.248\ x^4} - 0.003 - 0.015y + \mathbf{1.18\ y^2} + 0.016\ y^3 - 0.212y^4 - 0.021xy - 0.009xy^3 = \mathbf{x^2}$$

If we multiply each part of the recognized multinomial we have the following:

$$0.06x^3y + \mathbf{0.92\ x^4} - 0.012 - 0.06y + \mathbf{4.72\ y^2} + 0.064\ y^3 - 0.848y^4 - 0.084xy - 0.036xy^3 = \mathbf{4x^2}$$

Again, it can be seen for the above recognized multinomial, that the dominant (higher weighted) basis functions (y^2, x^2 and x^4) are the ones actually belong to the initial eight curve and carry approximately the same weights as the ones of the initial curve ($4.72y^2$, $4x^2$ and $0.92x^4$). The only differences are the fact that the basis function of y^4 also started to have a significant role to the final outcome, as well as the introduction of a constant value basis function that does not allow the curve to pass through the center of the coordinate system.

The noise in the numerical data of the eight curve is again increased by '0.05'. For noise magnitude of '0.15' the best fit is obtained again with a fourth order multinomial and the recognized structure is presented in the following Fig.5.10.

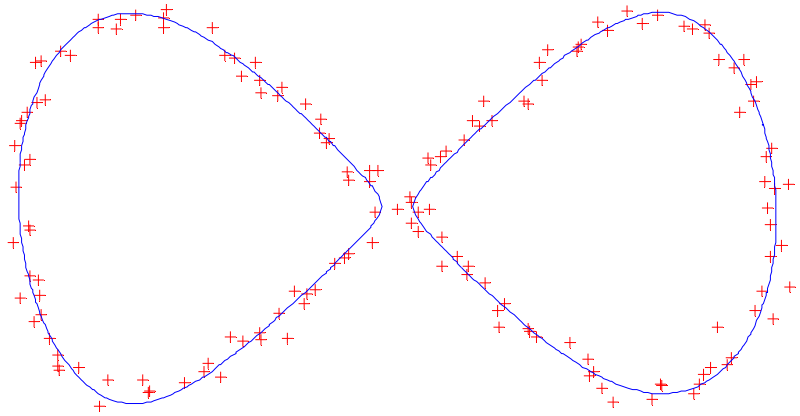


Fig.5.10: 4th order recognized curve for 0.15 noise in data (blue line)

The resulted multinomial of the recognition is the following, where the original basis functions of the eight curve appear in bold.

$$-0.03-0.1y-\mathbf{4.204y^2}+0.05y^3+0.5y^4-0.14xy-0.05xy^2+0.27xy^3+\mathbf{4.06x^2}-0.11y^2x^2-0.04x^3y=\mathbf{x^4}$$

Again, it can be seen for the above recognized multinomial, that the dominant (higher weighted) basis functions (y^2, x^2 and x^4) are the ones actually belong to the initial eight curve and carry approximately the same weights as the ones of the initial curve ($4.204y^2$, $4.06x^2$ and x^4). Still, now a lot more basis functions are starting to contribute to the final outcome, and the constant value begins to play a more significant role.

For noise magnitude of '0.2' the best fit is obtained again with a fourth order multinomial and the recognized structure is presented in the following Fig.5.1.

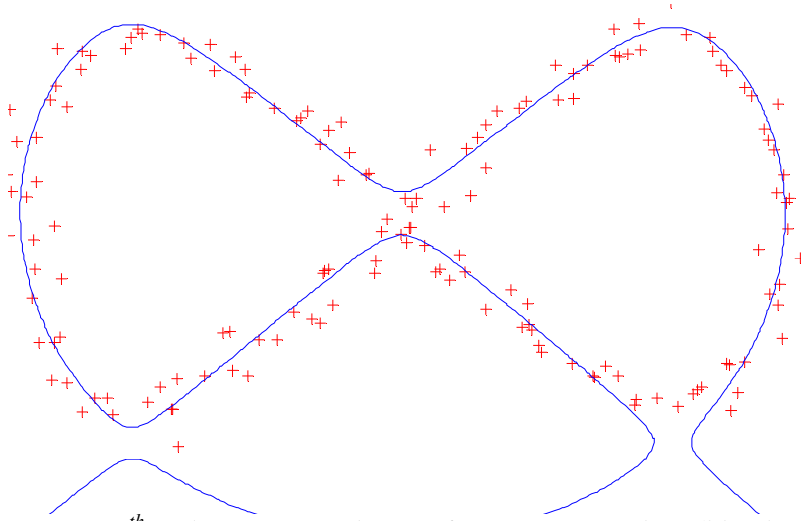


Fig.5.11: 4^{th} order recognized curve for 0.2 noise in data (blue line)

As it can be observed from Fig.5.11, the recognized multinomial begins to escape the "closed curve" shape that the initial eight curve possesses. The resulted multinomial of the recognition is the following, where the original basis functions of the eight curve appear in bold.

$$0.068+0.49y-\mathbf{5.86y^2}-0.55y^3+\mathbf{1.82y^4}+0.01x-0.038xy+\mathbf{4.11x^2}+0.013x^2y^2=\mathbf{x^4}$$

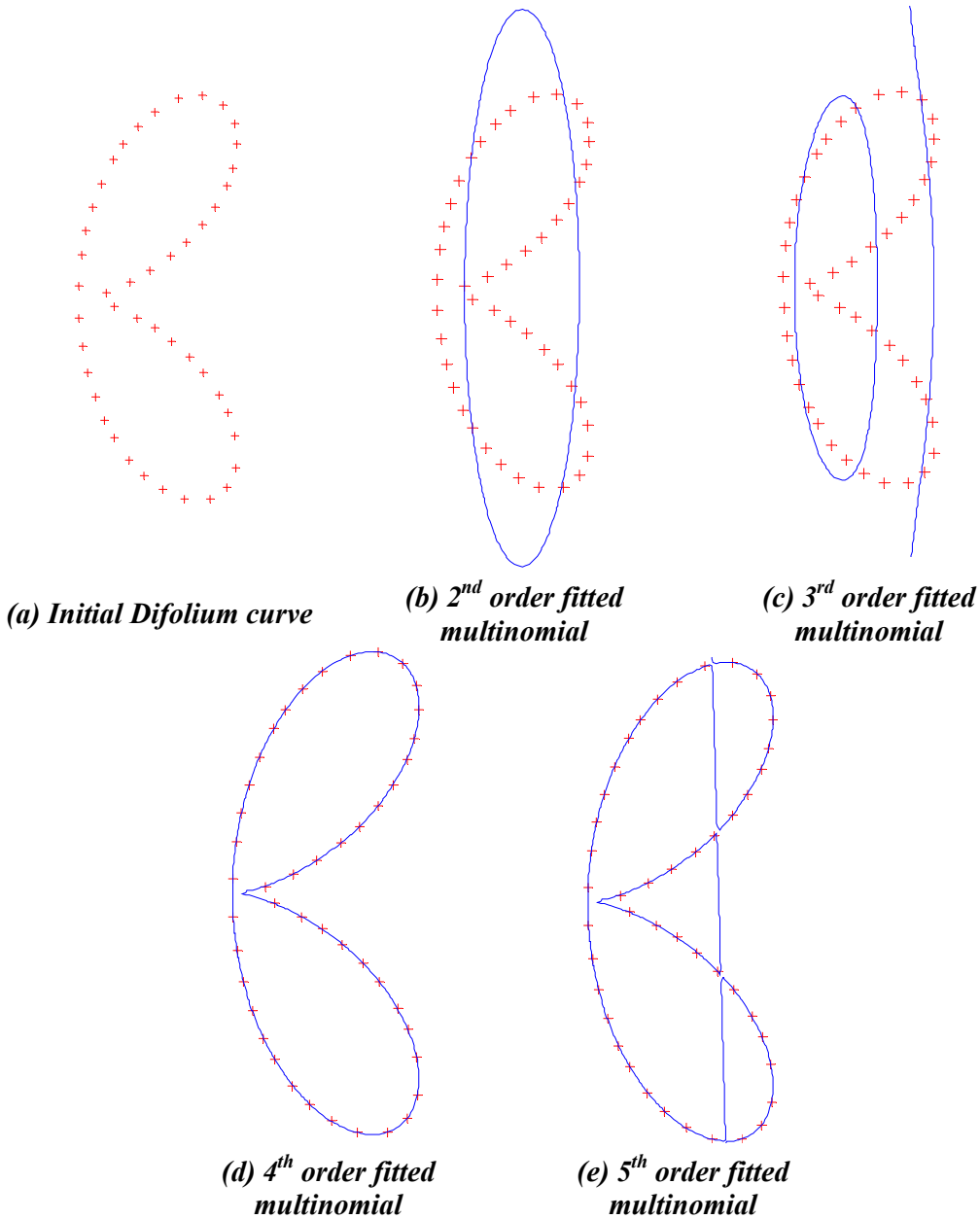
Again, it can be seen for the above recognized multinomial, that the dominant basis functions have now exceeded the ones that describe the initial eight curve. The 'y^2' bases function as and the constant values are now obtaining more important roles.

From the above recognition results it can be seen how noise in the data can actually influence the quality of the multinomial fit, at least at the level of resemblance with the initial (expected) eight curve.

c) Difolium Curve Autonomous Procedure

The difolium curve has the Cartesian equation of $4axy^4=(x^2+y^2)^2$ (where 'a' was set equal to '1') as it is displayed in Fig.5.12.a. For this test the morphogenetic neuron will autonomously decide for the best multinomial fit, both for the initial curve numerical data, but also for data containing noise of '0.1'.

The first recognition process is implemented on numerical data that do not contain any noise. The morphogenetic neuron, initially attempts to fit a 2^{nd} order multinomial and the result is displayed in Fig.5.12.b.



(a) Initial Difolium curve

(b) 2^{nd} order fitted multinomial

(c) 3^{rd} order fitted multinomial

(d) 4^{th} order fitted multinomial

(e) 5^{th} order fitted multinomial

Fig.5.12: Automated recognition and selection on difolium curve without uniform noise in the numerical data (blue line)

The result of the fit, as a point-to-curve distance mean, carries the value of '0.1290'. The estimator is not yet active as it requires at least two entries to perform the comparison and derive to some result. Therefore the maximum allowed power is increased and a 3rd order multinomial is being attempted for fitting.

The result of the 3rd order fit is displayed in Fig.5.12.c. The point-to-curve distance mean value this time is '0.0826'. The evaluator is active and taking into account the difference in number of the used basis functions and the value of the point-to-curve distance means, it derives to the result that the second fit was better than the first one, and gives the order to continue to a higher power. Therefore the allowed multinomial power in this step is of 4th order.

The result of the 4th order fit is displayed in Fig.5.12.d. The point-to-curve distance mean value this time is '0.0050'. The evaluator is again derives to the result that the 4th order fit was better than the one of the 3rd order, and continues the recognition with a higher power. Therefore the allowed multinomial power in this step is of 5th order.

The result of the 5th order fit is displayed in Fig.5.12.e. The point-to-curve distance mean value this time is '0.0051'. The point-to-curve distance mean value at this recognition is approximately equal to the one of the 4th order multinomial fit. Still the evaluator takes into account the difference in number of the used basis functions and the fact that the 4th order multinomial is a lower order fit and arrives to the result that the 4th order fit was better than the one of the 5rd order. Then, it terminates the procedure and decides that the 4th order multinomial is the best candidate.

The resulted multinomial of the recognition is the following:

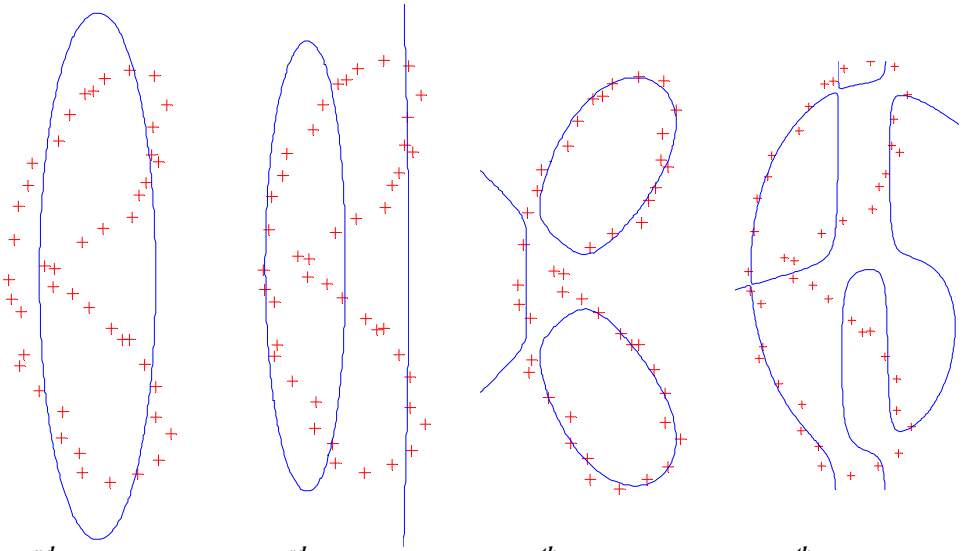
$$0.49992\, x^2y^2+0.2505\, x^4+0.25028\, y^4 = xy^4$$

It can be seen at the above recognized multinomial, that it is actually equal to the function of the initial difolium curve ($4xy^4=(x^2+y^2)^2$), when all members are multiplied by '4'.

The second autonomous recognition process is implemented on numerical data that contain uniform noise of '0.1'. The same procedure as above was followed and the results are displayed in the following Fig.5.13.

The recognition algorithm starts the process with a 2nd order multinomial, and an ellipse is recognized, which is displayed in Fig.5.13.a. The point-to-curve distance mean value that was obtained was '0.1297'. The allowed maximum power is incremented and the 3rd order multinomial fit is displayed in Fig.5.13.b. The evaluator decides that the 3rd order fit is a better candidate and allows the procedure to continue, since the distance mean is better (0.0852) than the 2nd order.

Then a 4th order multinomial is fitted in the numerical data. The recognized structure is displayed in Fig.5.13.c. Here it is obvious that the influence of the noise has altered the expected result, still the point-to-curve distance mean value has improved to '0.0371'. The recognized curve hardly resembles part of the initial dfolium curve. Still the resulting fit, is better that the one of the 3rd order and therefore the evaluator allows the process to continue.



(a) 2nd order fitted multinomial ***(b) 3rd order fitted multinomial*** ***(c) 4th order fitted multinomial*** ***(d) 5th order fitted multinomial***

Fig.5.13: Automated recognition and selection on difolium curve with uniform noise of '0.1' in the numerical data (blue line)

The outcome of the recognition of the 5th order is displayed in Fig.5.13.d. It can be seen that the actual result is not sufficient, and numerically the point-to-curve distance mean value has become worse (0.0416). Therefore the evaluator selects the multinomial expression of the 4th order as the best candidate for the fit.

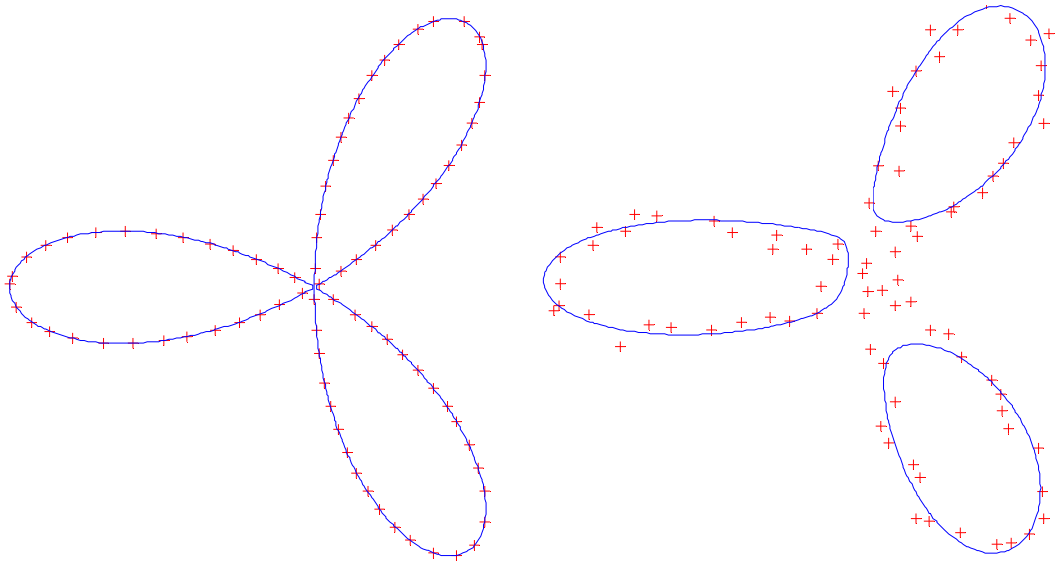
The resulting multinomial of the fourth order is the following where the original basis functions of the difolium curve appear in bold.

$$2.7x\,y^2\text{-}x\text{-}1.3xy\ \text{-}0.4x\,y^3+5.5x^2+2.9x^2y\mathbf{-x^2y^2}\text{-}9.7x^3\text{-}1.5x^3y\ +\mathbf{4.1x^4}+0.3y^2+0.3y^3=\mathbf{y^4}$$

It can be seen for the above recognized multinomial that one of the initial existing basis functions '$x\,y^4$' was eliminated during the optimization process, and does not participate at the final result. This occurs due to the high amount of noise in a relatively small population on numerical data. The influence of the number of data population to the goodness of a fit will be discussed in the following paragraph.

5.4) Noise and Data Population Influence

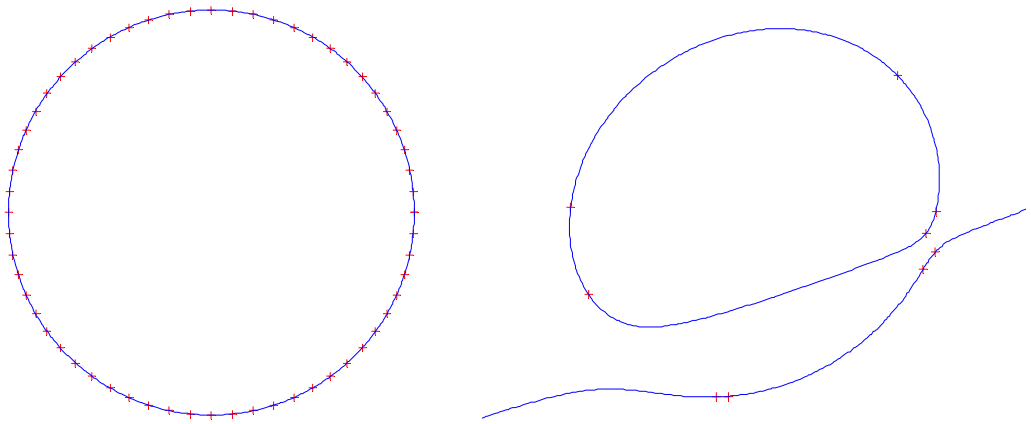
It can be easily understood that noise existing in data introduces difficulties in the "correct" recognition of structures. As it is displayed in Fig.5.14.b, the addition of noise in the numerical data results to a structure that differs from the original expected one (Fig.5.14.a). Still, this does not mean that the recognition is not satisfactory.



(a) Numerical Data without noise ***(b) Numerical Data with noise***

Fig.5.14: Influence of Data Noise in Recognition

Furthermore, the lesser the number of available numerical data is, the more the number of possible multinomial structures to be fitted. This can also be derived from Fig.5.15 where the numerical data used to recognize the structure belong to a circle.



(a) Large number of Numerical Data ***(b) Small number of Numerical Data***

Fig.5.15: Influence of Data Population in Recognition

Therefore, another question was raised on how and in what extent the addition of noise and the change of number of numerical data influence the recognition procedure and result. I order to determine that influence factor; a number of tests were executed.

A set of numerical data was subjected to structure recognition while the uniform noise was imported to it, and the number of its data population was reducing. The experiment that is presented below was performed on the numerical data of the curve “trifolium” which is displayed in Fig.5.16.

The trifolium curve is a 4^{th} order multinomial curve. The initial developed population of this curve was 334 data points and contained only rounding errors. Out of this

population, another five sets of data points were produced with less number of points (223, 167, 112, 84 and 42 data points).

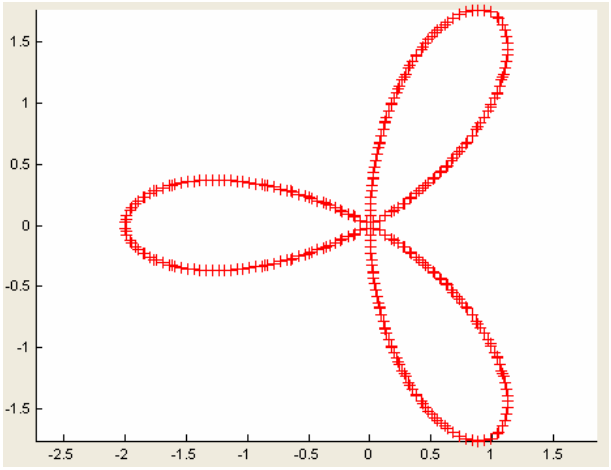
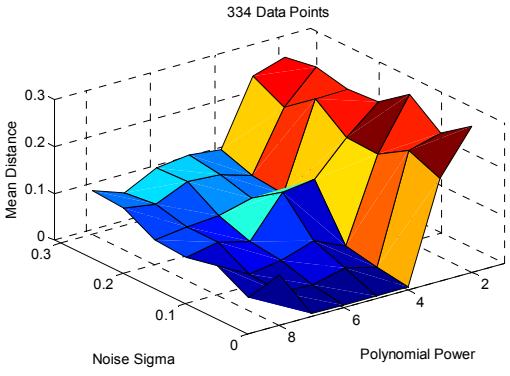
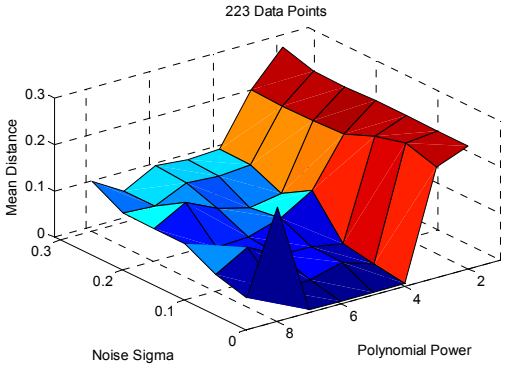


Fig.5.16: Trifolium Curve

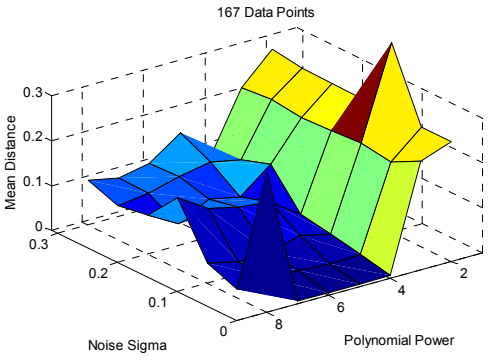
Each of this set was used for manual structure recognition starting from 2nd multinomial fit to 8th multinomial fit. For all the intermediate steps their point-to-curve mean distance was obtained. Then, different amounts of noise (from 0 to 0.3) were imported in the data, and the recognition process was again initiated from 2nd multinomial fit to 8th multinomial fit. Plotting the resulting point-to-curve mean distance values of each of the nth multinomial recognized curves of data with 'sigma' amount of noise, the surfaces of Fig.5.17 are produced.



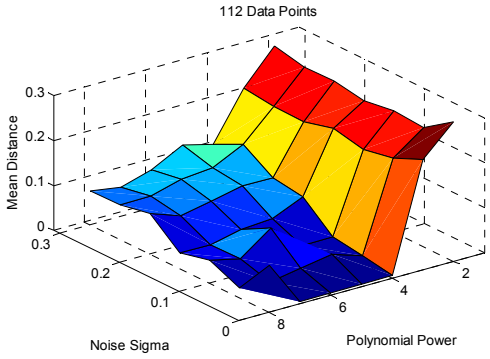
Surface A (334 points)



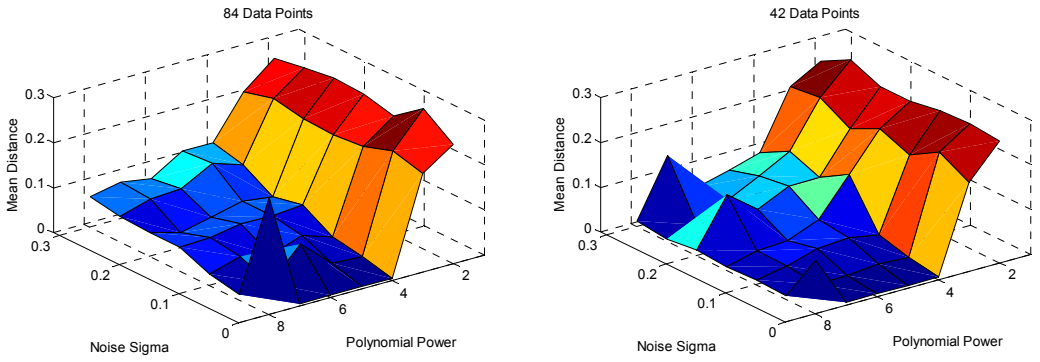
Surface B(223 points)



Surface C (167 points)



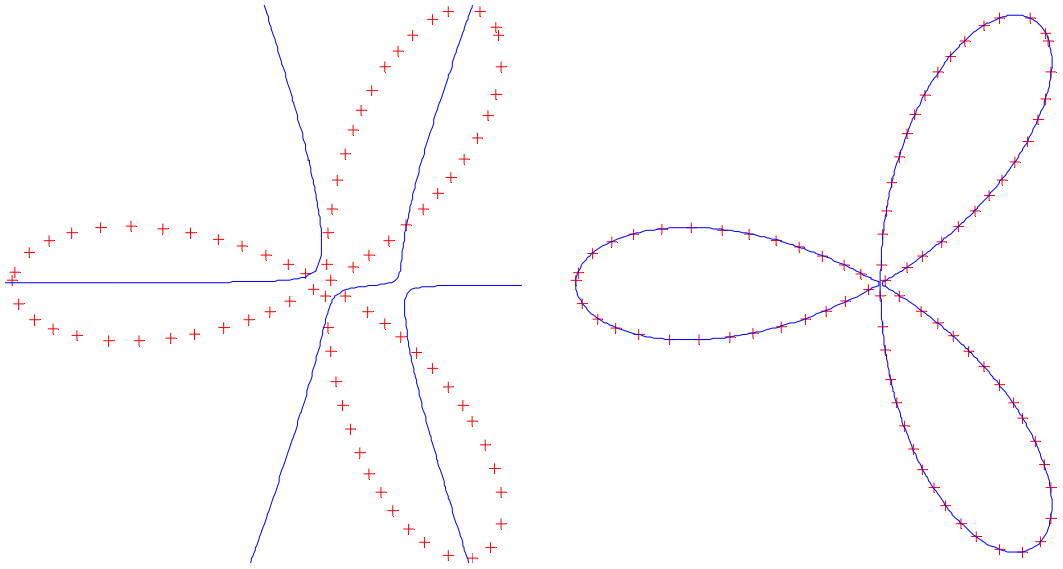
Surface D(112 points)



Surface E (84 points) ***Surface F(42 points)***

Fig.5.17: Surfaces of point-to-curve mean distance values according the noise in data and the maximum allowed multinomial fit.

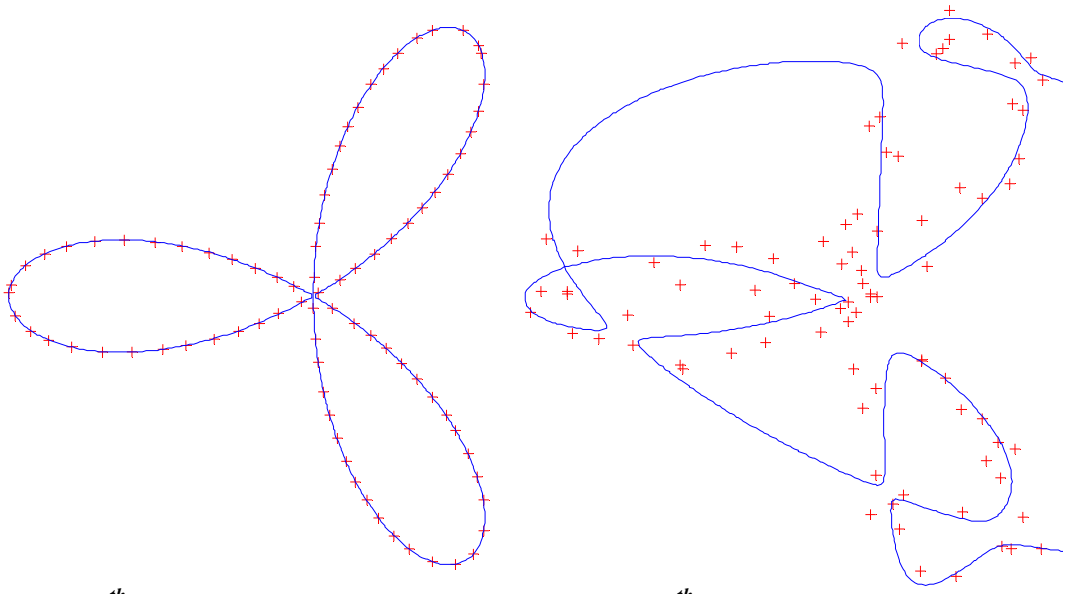
As it can be seen in all of the surfaces, there is a steep slope, when the allowed maximum multinomial fit reached the value of the order of the trifolium curve (4^{th} order). This is the first indicator that the required multinomial power might have been reached (Fig.5.18).



(a) 3^{rd} order fit ***(b) 4^{th} order fit***

Fig.5.18: Difference of recognized Multinomials when the surface slope occurs.

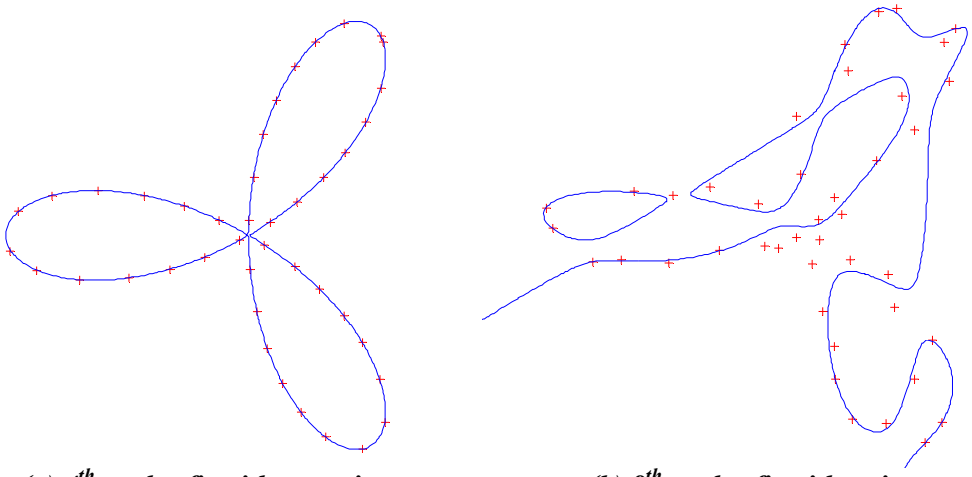
A second observation that can immediately be made from the surfaces is that the more the noise in the data, the worse the value of the point-to-curve distance mean. Still the use of higher order multinomials seems to produce an acceptable result. As it can be seen in Fig.5.19.b, even though the recognized multinomial might have reduced the resulted point-to-curve distance mean value, but the outcome (eve though satisfactory) does not resemble the initial trifolium curve.



(a) 4^{th} order fit without noise ***(b) 8^{th} order fit with noise***

Fig.5.19: Difference of recognized Multinomials when noise is present.

A third observation is that when then number of participation numerical data is reduced (Surface F); higher order multinomials can provide good results regardless the amount of noise in data.



(a) 4^{th} order fit without noise ***(b) 8^{th} order fit with noise***

Fig.5.20: Difference of recognized Multinomials when small data population.

As it is displayed in Fig.5.20, the multinomial fit again might be good numerically, but the outcome again does not resemble the initial curve.

6
Discussion & Recommendations

6.1) Introduction

During this project, many roads could be followed and investigated. Many of them were followed and many of them are still left to be explored. The morphogenetic neuron offers a generic solution to the curve fitting problem and can find application in numerous other processes. This work sets the basis towards the direction of using the morphogenetic neuron for structure recognition. Therefore priority was given to the paths that will lead to the correct recognition of structures and secondarily to the ones that lead to the estimation of the quality of that fit.

To support and evaluate the functionality of the morphogenetic neuron, a number of mechanisms were developed. The mechanisms also offer space for further investigation and improvement. These matters will be presented in the following paragraphs.

6.2) Discussion

The morphogenetic neuron was used within this project to recognize structures in two dimensional orthogonal spaces. Its capabilities though are not limited to two dimensional or orthogonal spaces. The morphogenetic neuron can make use of the properties of dot product of vectors and recognize structures in non orthogonal multidimensional spaces. The recognition process is straight forward and noticeably fast.

At this first step, the morphogenetic neuron was able to identify the underling structures in numerical data. It was shown that, up to a certain level, the morphogenetic neuron is robust and can still recognize structures similar to the ones that were expected. Still, the multinomial fit is sufficient even with high noise in data, though the structure differs a lot from the expected.

An evaluating and optimizing mechanism was developed to estimate the goodness of the fit and manipulate the parameters of the morphogenetic neuron. Still, it must be stated that this mechanism is designed according a subjective definition of an optimum fit. The chi-squared test was applied to the recognize curve to get an estimation of the goodness of the fit, but upon implementation the algorithm was not any different that the squared mean value of the point-to-curve distance mean.

6.3) Recommendations

As it was stated also in the introduction paragraph, there is a lot o space for research within this field. Still a few more steps can be taken at this point in order to enhance the performance of the currently existing platform.

a) At this point the estimator only calculates how well the data fit on the recognized multinomial, and does not take into account that there might be parts of the multinomial do not correspond to any points (paragraph 5.3.a where the eight curve of 5^{th} order is recognized with a line that should not exist). Therefore, a further optimization of the estimator or the addition of higher intelligence should be implemented.
b) A parameter that controls the estimation of the goodness of the fit is the point-to-curve distance mean. The point-to-curve distance is calculated according paragraph 3.3.a, and though it provides a sufficient approximation, a more accurate calculation of the distances would be useful.
c) The research for other statistical methods, instead of using the mean value of the calculated point-to-curve distances, could prove helpful.
d) The output value of the fuzzy logic estimator can be used to estimate a certain confidence level value of the goodness of the resulting multinomial fit.
e) A certain clustering of points before applying the recognition algorithm could help achieve better recognition results in data that contain noise.
f) When the final multinomial has been detected, an estimation of the noise existing to the data according the recognized multinomial could prove useful for better optimization.
g) Further work should be directed towards adopting the present recognition scheme for system identification.

7
Conclusions

The Morphogenetic neuron is a very powerful tool to use for recognizing structures hidden in plant numerical data. It is robust and can efficiently recognize structures even with data that contain a certain amount of noise. The recognition algorithm is straight forward and fast. Still, its performance depends on the selection of its recognition parameters (allowed multinomial order, number and type of basis functions).

The application of optimization mechanisms (elimination of insignificant basis functions, selection of the optimum output function) to the final recognized multinomial, improved the quality of the fitting.

The multinomial fit evaluating mechanism based on statistical and soft-computing means, is performing efficiently. Nevertheless, the evaluating mechanism could be further researched and improved.

The developed platform was very useful for conducting and observing the various experiments. It reduced experimentation time for the setting up of the recognition parameters for different test.

8
References

[1] G. Resconi, A.J. van der Wal , “A data model for the Morphogenetic Neuron”, Int. J. General System, Vol.29(1), pp.141-174, 2000

[2] Elisa Alghisi, “A study on Boolean functions by a neural network model”, Mathematical Department, Catholic University, Brescia, Italy, 2001

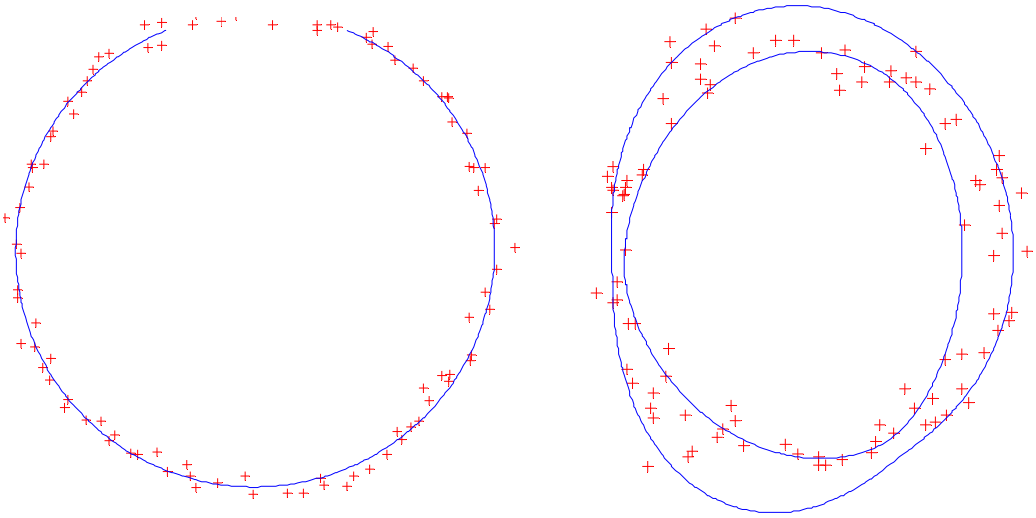
[3] D. A. Berry, B. W. Lindgren, “Statistics – Theory and Methods”, Second Edition 1996, Wadsworth Publishing, ISBN 0-534-50479-5

[4] K.M. Passino, S. Yurkovich, “Fuzzy Control” Addison Wesley, Menlo Park, CA (1998)

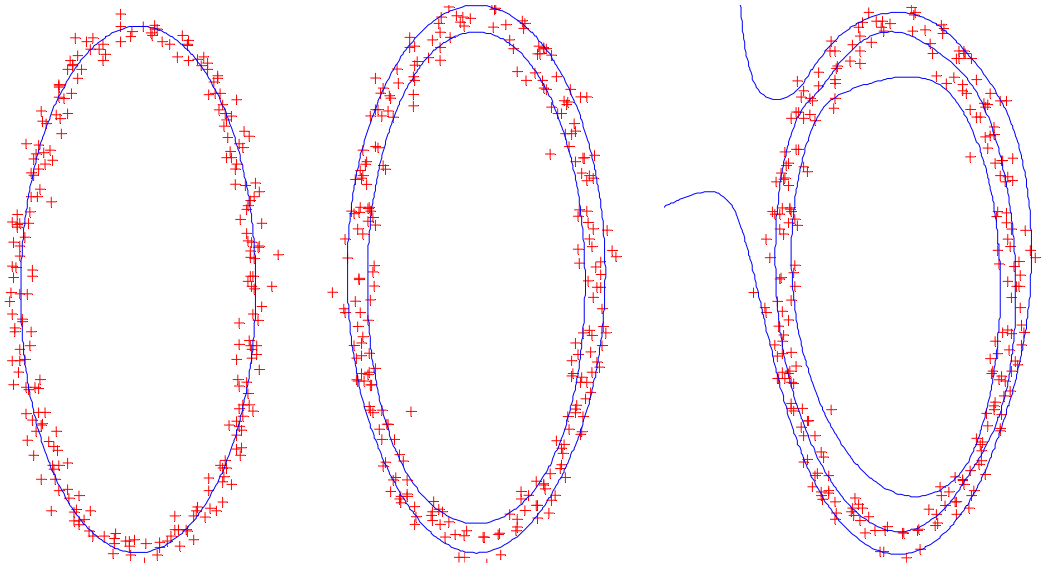
[5] G. Resconi, A.J. van der Wal , “The Morphogenetic Neuron”, not yet published

Appendix A – Various Other Recognition Tests

In this Appendix, a number of multinomial fitting examples are presented for different sets of numerical data. These recognitions have been implemented to display the performance of the Morphogenetic neuron, according to the population of the numerical data and the amount of existing noise.

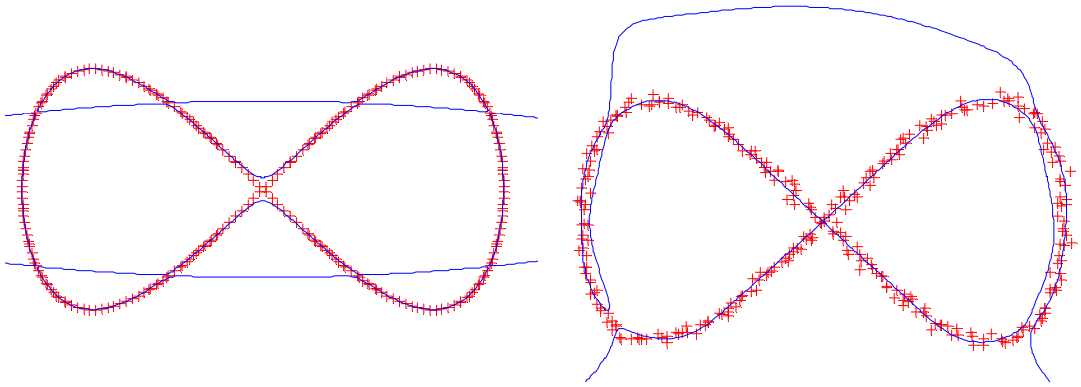


Recognition of structures in numerical data that initially belong to a circle. The data on the left figure contain less noise than the right one. For the left figure, the resulting multinomial is of the 2^{nd} order, whereas the right one is of 4^{th} order.

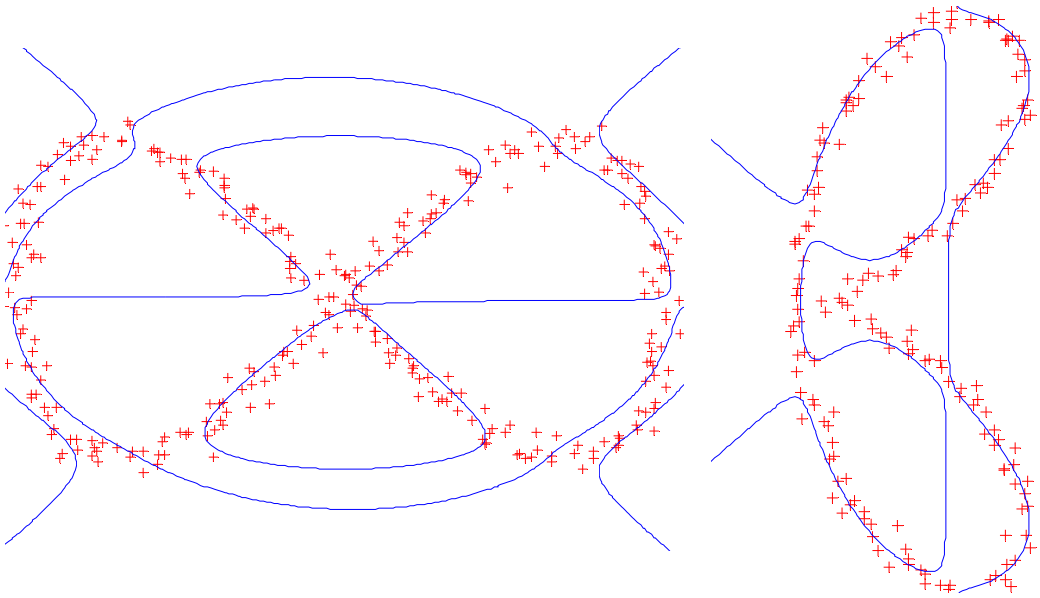


Recognition of structures in numerical data that initially belong to an ellipse. The data on the left figure contain less noise that the middle and right one. For the left figure, the

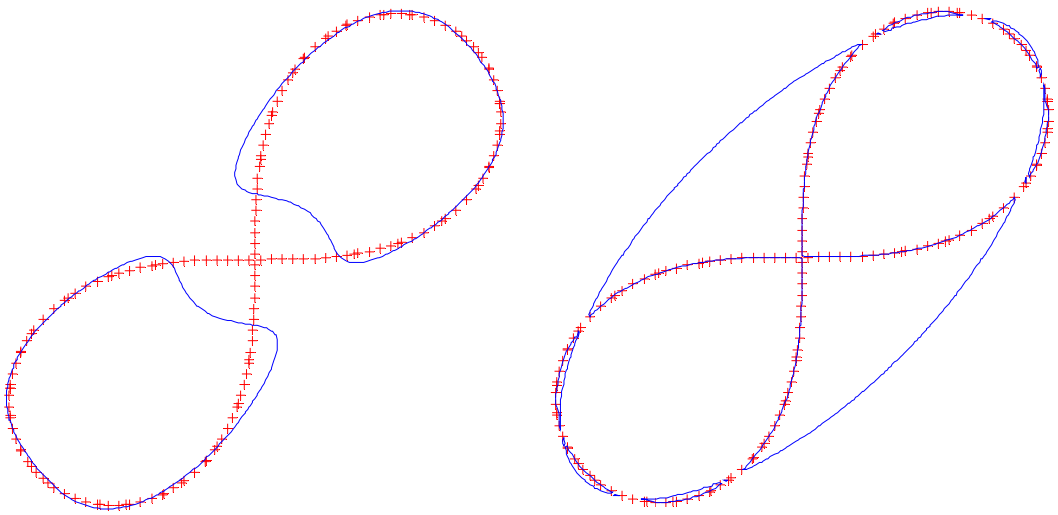
resulting multinomial is of the 2nd order, whereas the middle one is of 4th order. For the right one the morphogenetic neuron was used to fit an 8th order multinomial to the data.



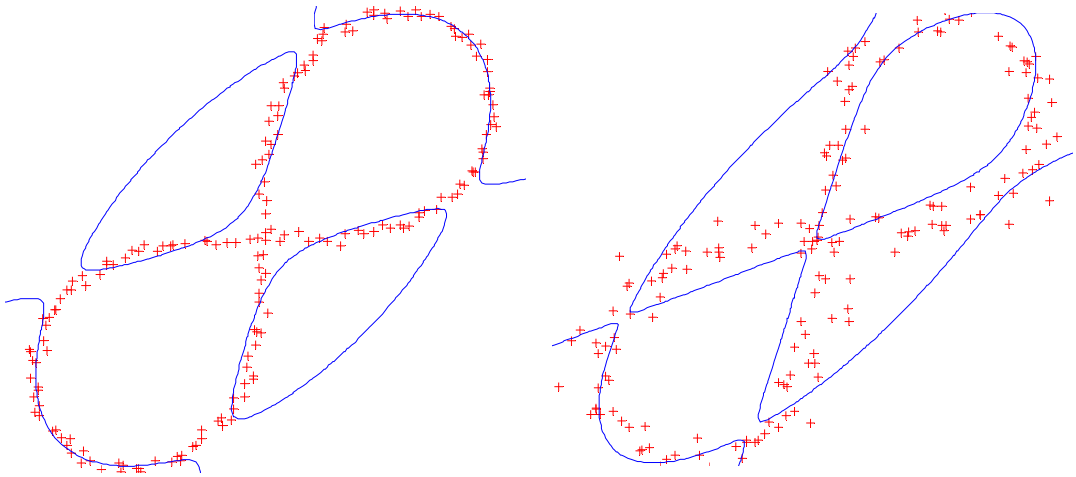
Recognition of structures in numerical data that initially belong to an eight curve. The data on the left figure contain no noise unlike the one on the right. In both cases a 6th order multinomial was allowed to be fitted in the data without performing any optimization techniques.



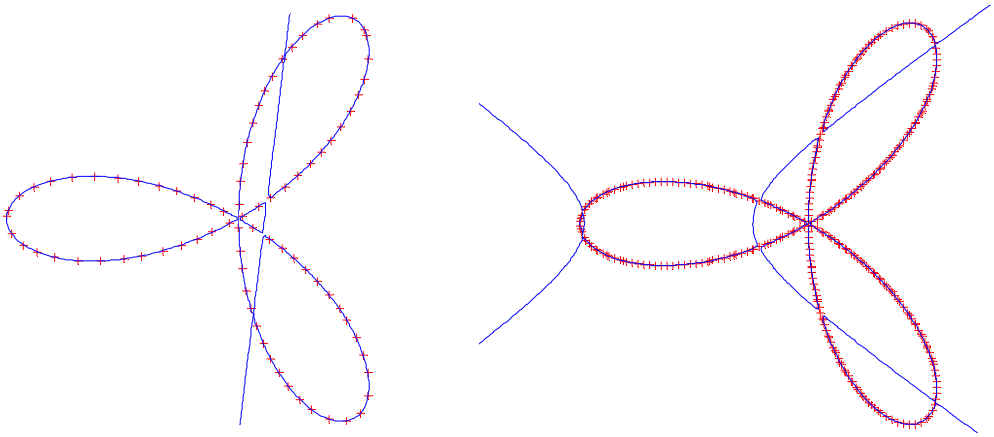
These are examples of resulting multinomial when over fitting is allowed to numerical data that contain sufficient amount of noise. The numerical data of the left figure belong to an eight curve, whereas the numerical data of the right figure belong to a difolium curve.



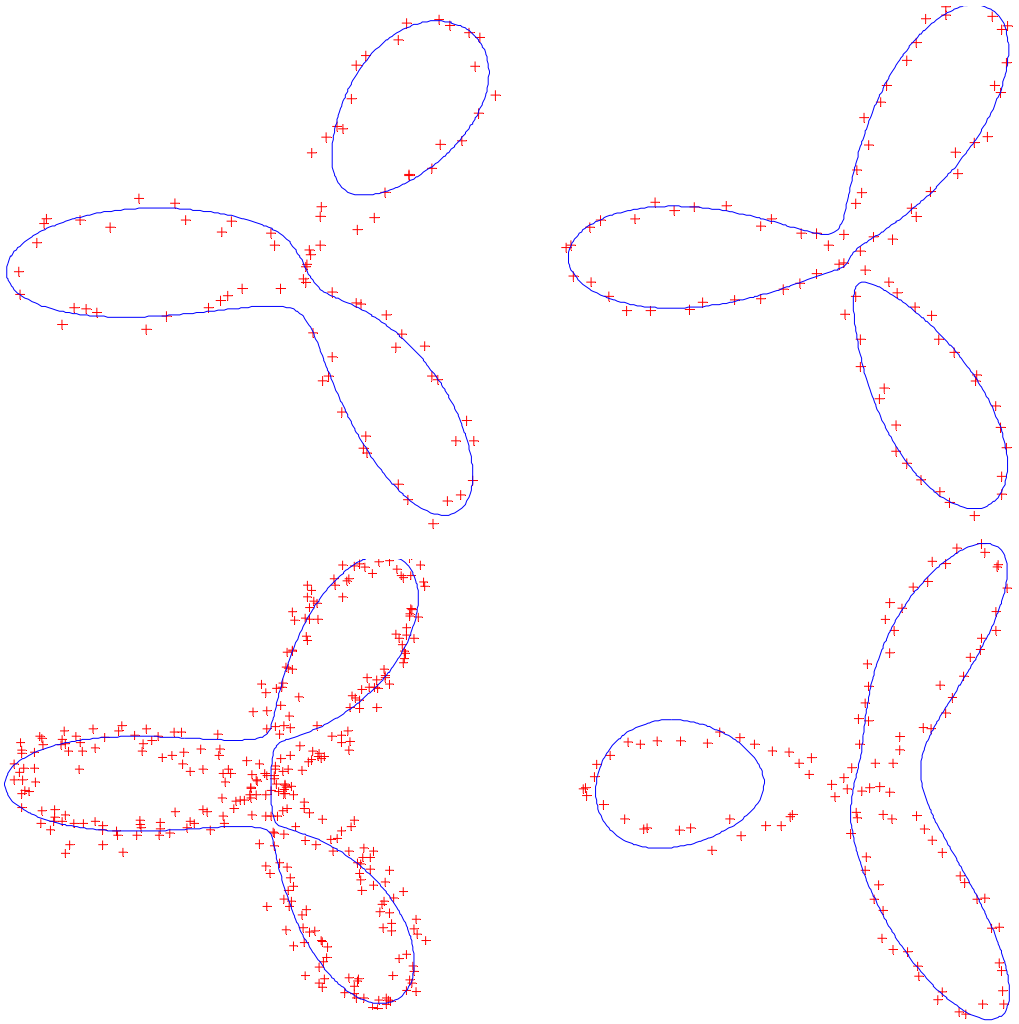
Recognition of structures in numerical data that initially belong to a rotated infinity curve. The numerical data on both figures contain no noise. At the left image no optimization techniques have been used yet. On the right figure, over fitting (6^{th} instead of 4^{th} order) is allowed to take place on "accurate" data.



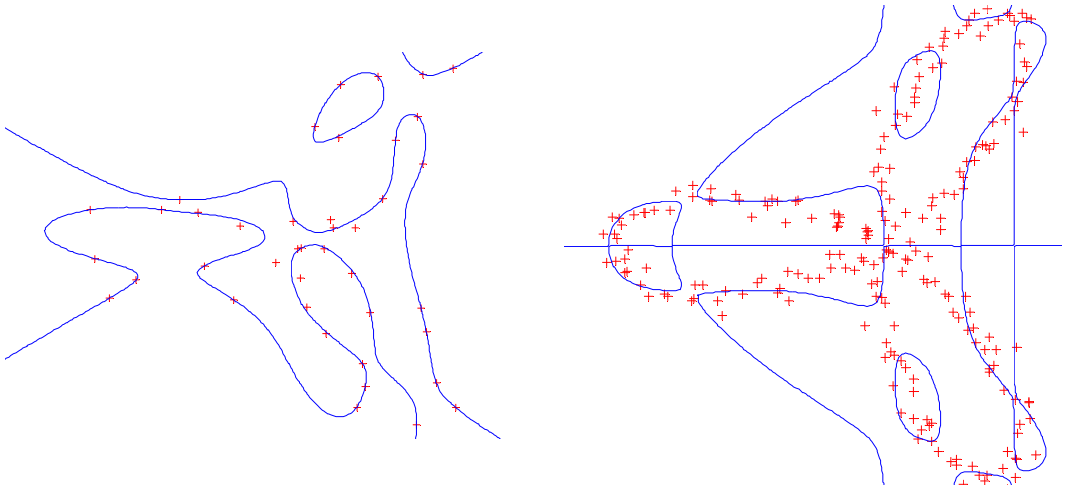
Recognition of structures in numerical data that initially belong to a rotated infinity curve. The data on the left figure contain less noise than the right one. The recognition, in both cases, has been implemented autonomously by the Morphogenetic neuron's algorithm and resulted to 4^{th} order multinomials.



Recognition of structures in numerical data that initially belong to a trifolium curve. The numerical data on both figures contain no noise. During the recognition, in both cases, the Morphogenetic neuron was allowed to over fit in "accurate" numerical data. The figure on the left contains a 5^{th} order curve, whereas the figure on the right contains a 6^{th} order curve.



Several examples of autonomous curve fitting to numerical data (trifolium) for different amounts of noise and number of data samples.



Trifolium curves with different amount noise and numerical data samples. The Morphogenetic neuron is allowed to over fit with 8^{th} order multinomials. Clearly on the left figure the curves passes through almost every point, but the final outcome differs from the original trifolium curve.

Appendix B – Graphic User Interface Manual

At this Appendix the usage of the developed platform will be explained, as well as the most important actions behind it. The Graphical Users Interface (GUI) of the morphogenetic neuron, if the one displayed in the following Fig.B.1.

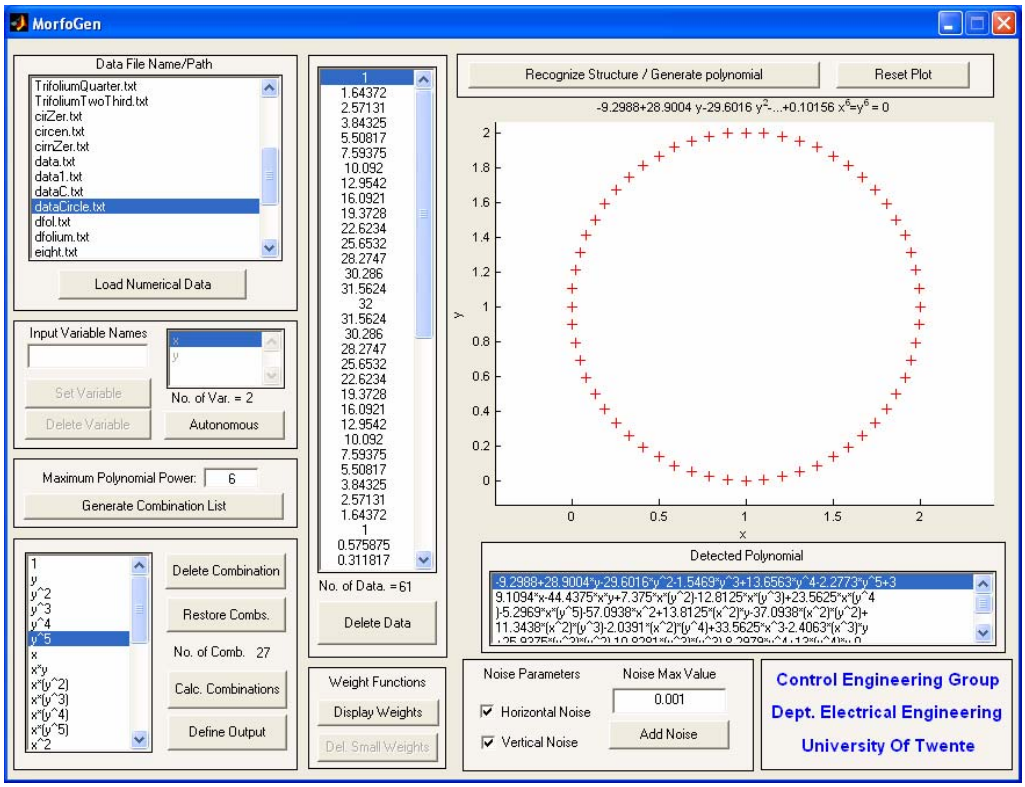


Fig.B.1: The GUI while in use

The GUI gives the possibility to the user to overlook the recognition procedures and follow each of the recognition steps. The Platform can run either autonomously or manually. At autonomously mode the user can have no input in the multinomial fit evaluation process, whereas in the manual mode, the user is the one who selects the best fit and manipulates the parameters of the morphogenetic neuron. The GUI is divided in several different parts, one for each of the required steps. These separate parts are displayed on the GUI with a frame line around them and are explained in detail in the following paragraphs.

a) Data Importing

This is the part where the data is imported in the platform. As it can be viewed in Fig.B.2, a number of text files are displayed in a list window within this part. These files are the ones that contain the numerical data of the structure that needs to be recognized.

These files must exist in the "DataSets" directory that exists in the same folder as the platform's MatLab m-files.

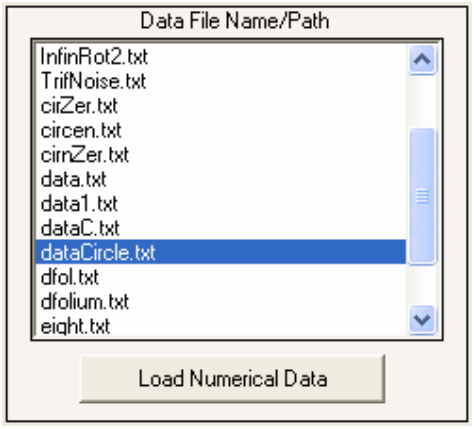


Fig.B.2: The Load Data part

The data in these files are organized in columns. Each column contains the numerical value of one variable that has been measured. For instance, one column contains all the values of the variable 'x', another all the values of a variable 'y' and so on. Each of the rows contains the numerical values of the variables for the same measurement. For example, if the measured values of 'x' and 'y' are 'a' and 'b', then 'a' and 'b' must exist on the same row.

The user can select the name of the file that carries the data, and then press the "Load Numerical Data" button. The values inside the selected file will be imported in a matrix with as many rows and columns as the number of the samples and the measured variables. Once the data are loaded, they are displayed in the axes window existing within the platform.

b) Variable Declaration

This is the part where the declaration of the variable names is implemented and the decision for autonomous or manual use of the morphogenetic neuron (Fig.B.3). To set a variable, a name must be given at the blank field above the buttons and the "Set Variable" button must be pressed. When a variable is set, it appears on the list at the right of the frame. To delete a variable, a variable form the list must be selected, and the "Delete Variable" button must be pressed. An indicator, showing the number of the total variables that have been declared exists under the list window (No. of War).

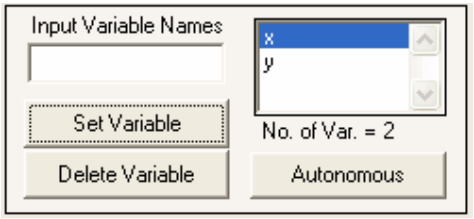


Fig.B.3: The Input Variables part

When pressing the "Autonomous" button, the platform is set in the autonomous mode, all the controls of the GUI are disabled and every action from that point on is taken by

the platform. In any other case the platform operates manually and enables the user to interact and manipulate the parameters and performance of the recognition procedure.

c) Maximum Multinomial Power

At this is the part the maximum allowed power of the basis functions is set (Fig.B.4). This selection of the maximum power has meaning only in the manual recognition mode. When setting the power and pressing the "Generate Combination List" button, the platform calculates all the possible combinations among the declared variables up to the selected maximum power, and stores them in and matrix of 'char' type.

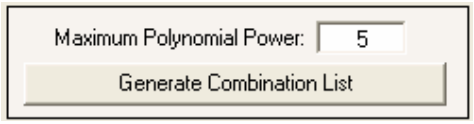


Fig.B.4: The Maximum Power part

The full description of the algorithm that calculates all the basis functions is presented in paragraph 4.2.b.

d) Basis Function Combinations

The generated basis function list of the process that was described in the previous paragraph ('c') is displayed in a list box within this part (Fig.B.5). Here the user is able to select and delete any basis functions appearing on the list. This way, the user can control the performance of the recognition algorithm, and therefore its output. An indicator of the number of the basis functions existing in the list is also present.

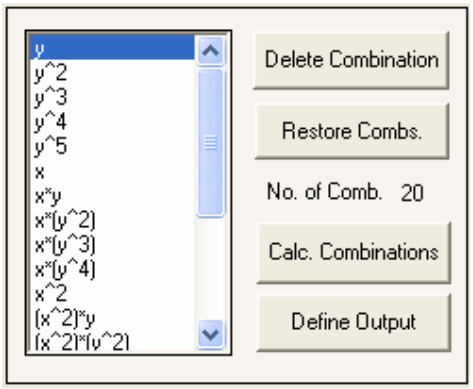


Fig.B.5: The Basis Function part

The "Delete Combination" button deletes from the combination list the one that is selected. The "Restore Combs" button restores the initial combinations in the list. When the unwanted basis functions have been deleted from the list then the "Calc. Combinations" button initiate the calculation of the *DataMatrix*. As described in paragraph 5.2. Then, the "Define Output" button is the one that used the selected basis function from the list as the system's output. The "Define Output" button, initially deletes the selected basis function from the list. Then, it cuts the numerical values for the *DataMatrix* and stores them in a vector to be used by the neuron.

e) Recognition

In this part, the actual recognition takes place. Here all the matrices that were constructed in the previous steps are imported in the morphogenetic neuron. The "Recognize Structure / Generate Multinomial" button initiates the recognition procedures and performs the mathematic operations that are described in paragraph 2.3.

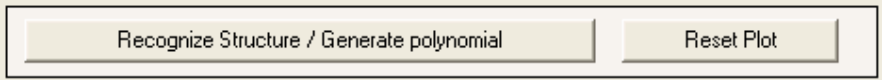


Fig.B.6: The Recognition part

The "Reset Plot" button resets the axes of the platform to the state where only the initial numerical data are plotted. After the curve is recognized, a line plotting it form on the axes will appear. This way the user has visual representation of the recognized curve plotted on top of the initial numerical values. This way an immediate estimation of the goodness of the fit can me made.

f) Initial Optimization

It is already mentioned the body of this report, that the first step that someone can take towards optimizing the outcome of the recognition, is to eliminate the basis functions that contribute very little to the final multinomial. This can be very easily determined by the values of the weights of each of the basis functions.

This part is the one that enables the user to view the resulting weight and delete the basis functions that are thought to be insignificant.

Fig.B.7: The Weight Optimization part

The "Display Weights", displays the values of the final basis function weight in the list box existing in the middle of the GUI. The weights appear in the same order as the basis functions in the combinations list box. The "Del. Small Weights" button eliminates the small weights in the same manner as it is implemented in the autonomous mode.

g) Data Noise

When the data are imported in the platform, contain only rounding errors and no actual noise. In order to test the performance of the morphogenetic neuron with noisy data, a way to impose noise in the dada had to be implemented. At this part, there are two types on noise that can be imported in the data and in two directions (Fig.B.8).

The first noise generator is the "Uniform Noise" one. When this option is selected, then the value in the edit box is used to indicate the maximum possible value of the noise. A vector with length equal to the number of the samples is formed and filled with uniform-random values existing between *'-Value of Edit box'* and *'+Value of Edit box'*.

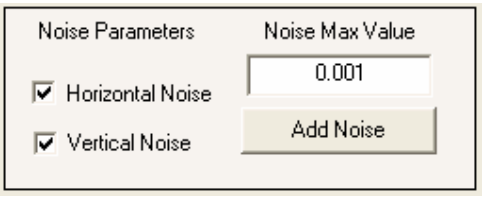


Fig.B.8: The Data Noise part

At this part, the user can also choose the direction that the noise will be applied (Horizontally, Vertically).

h) Axes Submenu

A number of controls were also developed for the axes were the plots appear (Fig.B.9). The first two commands deal with zooming in and out to the plot, for the user to have closer observation of the goodness of the fit. The "Normal View" command brings the plot to its initial condition with both the numerical data and the recognized multinomial present.

Fig.B.9: The Axes Menu part

The "Reset Plot" command is the same as the "Reset Plot" button of paragraph 'e', and resets the plot to its initial condition with only the numerical data plotted. The "Export Plot" command exports the image that appears in the axes to an Encapsulated Postscript File. The file is saves in the "Images" directory which exists in the same folder as the platform's m-files. The name is the same as the date and time that the export was instructed.

Appendix C – Running the Platform in MatLab

For the Morphogenetic Neuron platform to be executable in MatLab 6.5, the following files must exist in the "work" or the current working directory of the MatLab directory or in any of the predefined MatLab paths.

MatLab GUI & Function code m-files
Combo.m
CurvePoints.m
DevOrthWeights.m
EvalPolynomial.m
FactorFunction.m
FindFactors.m
FormulaDisplay.m
FuzzyEval.m
MorfoGen.m
Numconvert.m
Place_Powers.m
ProducePoly.m
VarCurves.m

Fuzzy Logic Controllers
FuzzyEstimatorN.fis

GUI figure
MorfoGen.fig

To start the simulations just type "morfogen" after the MatLab 6.5 window command prompt as is displayed in the following figure.