

University of South Wales



2060371

Bound by

Abbey Bookbinding



Unit 3 Gabalfa Workshops
Clos Menter
Excelsior Ind. Est.
Cardiff CF14 3AY

T: +44 (0) 29 2062 3290
F: +44 (0) 29 2062 5420
E: info@abbeybookbinding.co.uk
W: www.abbeybookbinding.co.uk

**EVOLVED NEURAL NETWORK APPROXIMATION OF
DISCONTINUOUS VECTOR FIELDS IN UNIT QUATERNION
SPACE (S^3) FOR ANATOMICAL JOINT CONSTRAINT**

GLENN LLEWELLYN JENKINS

School of Computing,
Faculty of Advanced Technology,
University of Glamorgan

A submission presented in partial fulfilment of the requirements of the University of Glamorgan / Prifysgol Morgannwg for the degree of Doctor of Philosophy.

August 2007

Abstract

The creation of anatomically correct three-dimensional joints for the simulation of humans is a complex process, a key difficulty being the correction of invalid joint configurations to the nearest valid alternative. Personalised models based on individual joint mobility are in demand in both animation and medicine [1]. Medical models need to be highly accurate animated models less so, however if either are to be used in a real time environment they must have a low temporal cost (high performance). This work briefly explores Support Vector Machine neural networks as joint configuration classifiers that group joint configurations into invalid and valid. A far more detailed investigation is carried out into the use of topologically evolved feed forward neural networks for the generation of appropriately proportioned corrective components which when applied to an invalid joint configuration result in a valid configuration and the same configuration if the original configuration was valid. Discontinuous vector fields were used to represent constraints of varying size, dimensionality and complexity. This culminated in the creation corrective quaternion constraints represented by discontinuous vector fields, learned by topologically evolved neural networks and trained via the resilient back propagation algorithm. Quaternion constraints are difficult to implement and although alternative methods exist [2-6] the method presented here is superior in many respects. This method of joint constraint forms the basis of the contribution to knowledge along with the discovery of relationships between the continuity and distribution of samples in quaternion space and neural network performance. The results of the experiments for constraints on the rotation of limb with regular boundaries show that $3.7 \times 10^{-4}\%$ of patterns resulted in errors greater than 2% of the maximum possible error while for irregular boundaries 0.032% of patterns resulted in errors greater than 7.5%.


Declaration

The work presented throughout this thesis, except that which has been explicitly referenced, is solely the work of the author, Glenn Llewellyn Jenkins, and has not been submitted in part or in whole for any other academic award or to any other academic institution.

The copyright of this work is vested in the author.

Signed

Dated

 (Author)

24 /08/2007

 (Director of Studies)

24 /08/2007

Acknowledgements

I would like to extend my sincere thanks to my project supervisors at the University of Glamorgan, Dr. Paul Angel and Mr. Colin Morris, for their support and guidance over the last four years. I would also like to thank Mr. August Mayer and Dr. Helmut Mayer for providing and supporting the NetJEN system used in my work. I would like to give special thanks to my parents who have encouraged me and believed in me all my life, and to Kirstie for her support, encouragement and tolerance. I would like to thank my close friends and both Kirstie's family and my own who have helped me keep a sense of perspective during the most stressful periods of my PhD. Finally I would like to thank my colleagues at Swansea Metropolitan University for providing me with support, friendship and sufficient time to complete my PhD.

Contents

1. Introduction	8
2. Literature Review	10
2.1 Joint Modelling	10
2.1.1 Anatomically Based Joint Modelling	10
2.1.2 Phenomenological Joint Modelling	12
2.1.3 Rotational Representation	14
2.1.3.1 Euler Angles	15
2.1.3.2 Rotation Matrices	16
2.1.3.3 Exponential Map or Versor	17
2.1.3.4 Quaternions (or Euler Parameters)	18
2.1.3.5 Swing and Twist	21
2.2 Neural Networks	22
2.2.1 Feed-Forward Neural Networks	23
2.3 Support Vector Machines (SVMs)	28
2.4 Genetic Algorithms	31
2.5 Evolved Artificial Neural Networks	32
2.5.1 Topology Evolution	33
2.5.2 Activation Function Evolution	34
2.6 Local and Global Learning Characteristics	35
2.7 Neural Network Approximation of Vector Fields	37
2.8 Principle Component Analysis	42
2.9 Conclusion	47
3. Simple Corrective Constraints	52
3.1 Methodology	53
3.1.1 Dataset Generation	54
3.1.2 NetJEN	57
3.1.3 Evolution and Training	60
3.2 Results	63
3.3 Discussion	71
3.4 Conclusion	74
4.1 Methodology	75
4.1.1 Dataset Generation	75
4.1.2 Evolution and Training	79
4.2 Results	81
4.2.1 Regular Boundaries	81
4.2.2 Irregular Boundaries	94
4.3 Discussion	100
4.3.1 Regular Boundaries	100
4.3.2 Irregular Boundary	102
4.4 Conclusion	105

5.1 Methodology	106
5.1.1 Number of Hidden Layers and Nodes	106
5.1.2 Training Epochs	107
5.1.3 Number of Patterns	108
5.1.4 Pattern Order	108
5.1.5 Pattern Distribution	108
5.1.6 Generations	109
5.1.7 Population Size	110
5.1.8 Activation Function	110
5.1.9 Dataset Creation	112
5.1.10 Activation Function Evolution using NetJEN	112
5.1.11 Training and Evolution	114
5.2 Results	116
5.2.1 Discontinuous Vector Fields Representing Three Dimensional Constraints	117
5.2.1.1 Neural Network Size	117
5.2.1.2 Training Epochs	118
5.2.1.3 Number of Training Patterns	120
5.2.1.4 Generations	121
5.2.1.5 Population Size	123
5.2.2 Discontinuous Vector Fields Representing Regular and Irregular Quaternion Boundaries	125
5.2.2.1 Number of Hidden Nodes	125
5.2.2.2 Training Epochs	126
5.2.2.4 Training Patterns	128
5.2.2.5 Pattern Order	129
5.2.2.6 Pattern Distribution	130
5.2.2.7 Activation Function Evolution	130
5.3 Discussion	136
5.3.1 Discontinuous Vector Fields Representing Three-Dimensional Constraints	136
5.3.2 Discontinuous Vector Fields Describing Regular and Irregular Boundary Quaternion Constraints	138
5.4 Conclusions	142
6. Binary Constraints in S^3 Space	144
6.1 Methodology	144
6.1.1 Dataset Generation	145
6.1.2 SVMLight	145
6.1.3 Training Configuration	146
6.2 Results	146
6.3 Discussion	150
6.4 Conclusions	152
7. Discussion	153
7.1 Binary Constraints	153
7.2 Corrective Constraints	154
8. Conclusions	164
9. Future Work	168

9.1 Introduction	168
9.2 Development of the Current Work	168
9.2.1 Performance of Spline Based Neural Networks	168
9.2.2 Performance Metrics for Joint Constraint Vector Fields	169
9.2.3 The Constraint of Rotation around the Limb	169
9.2.4 Reduced Coordinate Encoding	172
9.2.5 Multiple Dependent Joints	173
9.2.6 Training from Sampled Data	174
9.3 Application of the Techniques Developed	174
9.3.2 Kinematic Modelling	174
9.3.3 Biomechanical Modelling	175
9.3.4 Dynamics Modelling	175
9.3.5 Pose Constraints	176
9.3.6 Camera Constraint	176
9.4 Conclusion	177
Appendix A.	190
Published Papers	190

1. Introduction

Joint models are important constituents of anatomical models, they are used in simulation to retain anatomically correct movement and ensure limbs do not separate or intersect. Anatomical models are used in both medicine and animation to create model humans as characters, teaching aids or to evaluate the benefits of surgical or prosthetic intervention [7-9]. It has been acknowledged that the joint models used in animation are particularly underdeveloped despite advances in other areas of humanoid modelling [1].

Many current techniques are limited by their underlying representation or their abstraction of the joint function and there is increasing demand for anatomically correct joints for both animation and medicine [1, 10, 11]. However in current applications, increasing accuracy leads to increasing complexity which requires additional computation [8, 12, 13]. No single technique has been presented suitable for accurately modelling all classifications of anatomical joint [1].

The long term aim of this work is to create an anatomically correct joint model based on person specific data (from non-invasive [14] or invasive [15] sources). Each model will provide an accurate representation of an individual's mobility.

The accurate representation of joint constraints by Artificial Neural Networks (ANN) has advantages over methods that use coarse approximations and computationally expensive iterative techniques. In combination with a quaternion based angular representation this presents an opportunity for systems with uniform constraint and angular representations.

In this thesis a number of simple cases based on contrived data are examined these provide the foundation for further research towards an eventual goal of patient or character specific joint constraints systems.

A joint constraint system must be capable of a decision regarding the validity of the current orientation and where required the appropriate correction should be assigned. Where a constraint system only describes the constraint as valid (within its constraint

limits) or invalid (outside its constraint limits) the term binary constraint is used. Where the constraint system responds with a correction for invalid configurations and a zero correction for valid corrections the term corrective constraint is used.

This work focuses on corrective constraints modelled as vector fields and investigates the application of evolved ANN techniques to model a joint constraint system, for corrective constraints. The vector fields considered are discontinuous in nature, which increases the difficulty of their approximation. Using evolutionary techniques based on genetic algorithms, the topology of the network is configured dynamically to approximate the piece-wise linear properties inherent in discontinuous functions [16]. The application of Support Vector Machines to the problem of binary constraints is also investigated. In both cases less complex constraints are considered as a precursor to those of the complexity required to model anatomical rotational constraints.

In Chapter 2 current approaches to joint constraint, rotational representation, neural networks and their evolution by genetic algorithms are reviewed. Chapter 3 introduces initial experiments exploring the capabilities of topologically evolved neural networks applied to vector fields representing corrective constraints of increasing dimensionality. This is followed in Chapter 4 by the application of these techniques to vector fields representing quaternion based constraints, with both regular and irregular boundaries. This work is concluded in Chapter 5 where training and evolution constraints imposed to minimise temporal cost in earlier experiments are removed to ascertain the capabilities of the neural networks. The construction and training of binary constraints of varying dimensionality is considered in Chapter 6. Chapters 7 and 8 contain a discussion and conclusions relevant to the thesis as a whole, finally Chapter 9 details future work.

2. Literature Review

In order to apply neural networks to the problem of joint constraint existing approaches to anatomical joint constraint and their limitations are reviewed. Current approaches can be classified as either 'anatomically based' or 'phenomenological', of which the latter are more relevant to this work [17]. The research presented here focuses on the development of phenomenological joints, which mimic the behavior of the subject joint but not its physical structure. In describing the rotational behavior of these joints a selection of rotational parameterizations utilized in previous joint modeling solutions are considered.

Joint constraints are separated into 'binary constraints' and 'corrective constraints' the distinction being the response of the constraint system. In the binary case valid and invalid rotations invoke true and false responses respectively while in the corrective case a valid input rotation invokes a zero corrective response while an invalid rotation results in the required correction to the closest valid rotation being given. Machine learning techniques are studied with focus on their properties regarding classification (for binary constraints) and vector field approximation (for corrective constraints).

2.1 Joint Modelling

The problem of constructing anatomical joints has been approached in several ways. Engin and Tumer [17] classifies these approaches as 'anatomically based' and 'phenomenological'. Anatomically based joints represent the joint through the interaction of geometrical models that represent the physical components of the joint while phenomenological joints use mathematical models to describe the behavior of the joint without reference to its constituent parts.

2.1.1 Anatomically Based Joint Modelling

Anatomically based approaches emulate the physiological properties of joint constituents in order to simulate their behaviour, as these physiological properties are responsible for both movement and constraint the desired constraint is implemented. Anatomical joints are typically made up of several constituents; bones, ligaments, tendons and muscles, each of which contributes to the constraint [17].

Gait simulations model the patterns of movement observed during a walking cycle. In many gait simulations the extremes of movement are ignored, as such limits are never reached during the gait cycle. Groups of muscles acting together prevent the limb reaching the limits of the joint [18, 19]. However for motions other than gait (e.g. jumping, stretching or a fall) joint limits may be encountered and so to create more versatile models more complex joint constraints are required.

An anatomical joint is always a connection between one or more bones, though this is often simplified to a mechanical linkage, some approaches attempt to model the interaction of the bones themselves. Bone dynamics are typically based on a physical simulation of the contact forces of the bones in question [18, 20]. These are often used in models in conjunction with other constituents of passive constraint i.e. ligaments, tendons and muscles [17, 21].

The simulation of ligaments has generated a great deal of research as ligaments provide much of the constraint in anatomical joints. Ligaments are mechanically heterogeneous complex structures in that they transfer loads non-uniformly and simultaneously in three dimensions [22]. Ligaments are responsible for the connection of articular extremities; pliant and flexible they provide maximum freedom of movement while being strong and inextensible so as not to yield under extreme force. Some ligaments are composed of yellow fibres (as apposed to the more common white, silvery variety) and have more elastic properties, it has been observed that they form a substitute for muscular power [23].

Ligaments have been simulated in several ways, the most prevalent being spring model variants. Ligaments behave much like springs at their optimal loadings though above this they are unpredictable [24]. Primitive spring models with single attachment points have been used to simulate ligaments [17, 25, 26], however ligaments have distributed

attachment points. To improve the accuracy of these models more complex approaches have been developed.

Manal *et al* [9] used a sliding attachment point that “floats” along the edge of the bone to which it is attached to simulate the active force of a group of ligaments using a single spring. Other approaches provide a more accurate representation of the large attachment area of ligaments by using elastic bundles – a collection of spring models used to simulate ligament behaviour [22, 27]. Mommersteeg *et al* acknowledge that the elements which make up the bundle cannot interact and suggests three dimensional polygons as a way forward [22]. Ligament constraint systems have also been described using mathematical models and utilizing rotational matrices as their description of the constraint [21].

Kinetic approaches have also been used to describe the forces exerted by a tendon [28]. Models have also been constructed in order to ascertain the effect of smaller tendons whose contribution to the constraint is difficult to measure [22].

2.1.2 Phenomenological Joint Modelling

Phenomenological joint models model the behaviour of the joint but not its physical structure. Primitive joint constraints have been parameterised using Euler angles [29-32]. Euler angles are one of the most established and popular parameterisations of orientation. They model the rotation about each of the principle axes (x , y and z). Euler angles suffer from the problem of “Gimbal lock”. Here a singularity occurs when 90° rotation is present around the second axis of rotation. This results in axis alignment and the loss of a degree of freedom [8, 33].

Inter-dimensional dependencies cannot be easily represented using Euler angles [34], and singularities or “Gimbal Lock” are encountered. Feikes *et al* [11] and Wilson *et al* [21] used special orthogonal matrices, a rotational parameterisation not susceptible to “Gimbal Lock”, to overcome these limitations. Inter-dimensional dependencies between Euler angles can be expressed as equations [35] though this increases computational cost.

N-dimensional boundary representations preserve the relationships between degrees of rotational freedom and are often used to supplement Euler angles. Conceptually, a number of points along the constraint boundary are obtained through measurement, and then approximated to an n -dimensional polygon. Pioneered by Korein [36] whose three dimensional spherical polygons constrained the movement of robotic arms. This technique has been employed to constrain the ‘swing’ component in a *swing-twist* parameterisation specifically for ball and socket joints [36-38]. Isaccs and Cohen [39] used an arc based approach similar to that used by Korein. In a related approach Gyi *et al* [40] projected a spherical polygon composed of arcs on to plane.

Cone based polygons using one [41] or more [10] cones have been suggested for the complex shoulder joint. In the more complex case using multiple cones, the cones themselves are planar polygons (composed of lines) much like the arcs used by Korein and others [36, 40, 42].

A number of robotics and biomechanics based joint models have been included in a single model by Shao and Ng-Thow-Hing [1]. Having reviewed the available models they concluded that no single method could adequately simulate all the joints of a human model and so a number of specialised constraint models were required to simulate the individual characteristics of anatomical joints. In their approach conical constraints and axial rotation constraints with changing centres of rotation are implemented along with dependencies between rotational constraints [1].

The use of quaternions preserves the relationship between the degrees of freedom and avoids the singularities encountered in other representations. Binary quaternion based constraints were implemented by Lee [6]. Lee decomposes a single quaternion into two quaternions each representing rotation in a single plane (effectively swing and twist for conic and axial constraints). In each case the centre of the constraint is known, a quaternion describing the swing of the joint can be created based on the angle between the centre and its image rotated by the subject quaternion and the axis calculated from the cross product of the constraint centre and its rotated image. The second quaternion representing the rotation around the axis can then be calculated by calculating the twist alone, (removing the swing component) the axis and angle of this quaternion can then be calculated. Conic, axial and revolute constraints are defined and can be used to model basic constraints, more complex constraints can be defined with a union of these

basic types. Interrogation of these shapes (to ascertain the validity of a joint configuration,) is presented, but no method of calculating a correction to the nearest valid orientation is defined.

Liu and Prakash [3] build on Lee's work. Using a sampled boundary they create a function to constrain the decomposed quaternion that can be used for both constraint validation and clamping to the boundary.

An approach by Johnson utilises logarithmic and exponential mappings between unit quaternions in S^3 and a tangent space in \mathbb{R}^3 . For this to be successfully achieved all quaternion must be moved to one side of the unit quaternion hyper-sphere as antipodal unit quaternions represent the same rotation. [2]. In Johnson's work statistical techniques are used to create both joint constraints and pose constraints. A set of valid rotations expressing joint and pose constraints on the unit quaternion hyper sphere are generated and their mean used as the centre point of the tangent space. A Gaussian probability density function is used to describe these points and boundaries can be implemented based on a maximum deviation from the mean of the sample data provided. Corrections are implemented by recursively moving an invalid point closer to the mean until the constraint is met.

In the quaternion iso-surface approach of Herda *et al* [4, 5] a subject's arm movements were recorded and represented in quaternion space. This quaternion-based representation was simplified by ensuring all scalar components were positive and omitting them, leaving the three-dimensional vector of imaginary components. A boundary (iso-surface) between valid and invalid rotations of the arm was then defined on the irregular boundary surrounding the valid region in three-dimensional space. Iterative approaches were employed to identify the closest valid joint configuration, its scalar component can be recovered from the other components, (as the quaternion is unit length) and the correction to this orientation calculated.

2.1.3 Rotational Representation

An object's orientation in three-dimensional space relative to some reference can be parameterised in a number of ways. Popular parameterisations include Euler angles,

axis angle, quaternion, the swing-twist representation, exponential map and orthogonal matrices [4, 8, 11, 37].

The parameterisation of rotation is difficult as rotations are non-Euclidean and periodic in nature, that is travelling infinitely far in any direction will return you to the starting point an infinite number of times. Any attempt to parameterise a non-Euclidean set (such as the set of rotations for a joint with three degrees of freedom) by an open subset of Euclidean space will result in ‘Gimbal lock’, the loss of degrees of freedom due to singularities [38].

The choice of rotational representation is often a trade off between the advantages and limitations of the available approaches. In some cases rotations are converted between representations for specific applications, such as the conversion of axis-angle representations to quaternion for interpolation of rotations. These conversions consume processing time and may introduce numerical errors into the system, where possible a uniform representation is preferred [2].

2.1.3.1 Euler Angles

Euler angles are one of the most established and popular parameterisations of orientation. A general rotation is described around three mutually orthogonal coordinate axis in fixed space. These three dimensional axis are reasonably familiar to most, and rotation around any one is described as a roll. (The axes are x , y and z and the corresponding rolls x -roll, y -roll and z -roll.) Euler angles ignore the interaction between the rolls around separate axis it is this failing which causes the ‘Gimbal lock’ problem [43].

Euler angles have been used by a number of authors for the parameterisation and enforcement of constraints [29, 31, 32, 37]. However constraints on a single axis may change in relation to the rotation of another axis and these relationships cannot be preserved by Euler angles alone [37].

It is difficult to interpolate Euler angles due to the relationships that exist between the degrees of freedom. In Cartesian coordinates it is trivial to interpolate (using linear

interpolation) between positions, however applying the same technique to Euler angles the interpolation between one orientation and another is not unique [43].

Kuffner [44] details other problems regarding the creation of distance metrics between rotations when using Euler angles. This is an important consideration in the creation of joint constraints, especially trained via neural networks, as mechanisms are required to assess the accuracy of the neural network response.

The direct constraint of Euler angles is not trivial due to a number of factors. While constraints can be expressed on each of the components individually (one dimensional constraints) it is difficult to describe valid and invalid regions. Several approaches have utilized Euler angle based constraints such approaches are severely limited as both constraint and motion are divided into separate planes and considered independently. These approaches are limited to robotics applications [32] and crude planar simplifications of the human skeletal system [29, 31]. Euler angles can be used as a rotational parameterisation where other methods are employed to impose constraint, in the work of Korein [36] for example.

2.1.3.2 Rotation Matrices

The set of all possible rotations (proper and improper,) can be considered using a 3×3 matrix representation. A subset of this group of each with a unit determinant and mutually orthogonal columns of unit length describes the proper or binary rotations only. This group of matrices is known as the special orthogonal group or $SO(3)$ [38, 45]. Though a total of nine numbers are used to represent the matrix there are also six constraints, three to maintain the unit length of the columns and three maintaining the pair wise constraints which keep the columns orthogonal [44]. Rotation matrices are a non-Euclidean parameterisation and do not contain singularities [46].

Though rotation matrices seem convenient they have several properties that make them difficult to apply to anatomical joint simulation. Floating point precision and space inefficiency are problems mentioned in the literature [44]. Floating-point errors also occur when two rotations are combined via multiplication often the resulting matrix is not orthogonal and must be re-orthonormalized this increases computational cost. More

relevant to this work is the difficulty in defining a simple metric for the differences between two matrices, thus error must be indirectly calculated [44]. Interpolation of matrices is also non-trivial the constraints must be maintained if the matrix is to remain valid [46].

Feikes *et al* [11] and Wilson *et al* [21] used special orthogonal matrices to describe the rotation of the knee joint.

2.1.3.3 Exponential Map or Versor

In the exponential map the axis and angle are combined together into a single vector the direction of the vector represents the axis and the magnitude the rotation about that axis. [8, 38]. In addition to the inevitable problems with singularities the exponential map has no convenient method for combining rotations (they must be converted to another format e.g. quaternions) [38]. Exponential maps are used as rotational representations by a number of authors [38, 47].

Axis angle or angular displacement orientation is a very similar rotational parameterisation defined as a displacement around a single axis, much like a one-dimensional Euler representation. However, in this case the axis does not correspond to a three-dimensional plane but is itself relative to planes in three-dimensional space and rotation is described around this axis. Unlike the exponential map a unit length vector component represents the axis about which the rotation described by a fourth component takes place. Baerlocher and Boulic [37] indicate that the axis angle approach is remains susceptible to singularities but to a lesser extent than Euler angles.

Grassia [38] defined constraints for axis angle parameterisations suitable for describing ball and socket joints. The approach decomposed the motion into swing and twist components. The constraint here concerned only the swing component and used line segments created from an ellipsoid template. This was later described as “swing twist” parameterisation and possible swing and twist constraints were explored [37]. A number of approaches are suitable for the individual constraint of both swing and twist once decomposition has taken place, though equations are required to express any relationship between these constraints.

2.1.3.4 Quaternions (or Euler Parameters)

Quaternions form a group whose underlying set is the four dimensional vector space \mathbb{R}^4 , a subset of which the set of unit quaternions (S^3) form a hyper-sphere embedded in \mathbb{R}^4 [38]. Using unit quaternions as a parameterisation of rotation gives a non-Euclidean parameterisation that is free from singularities. However constraints must be imposed to ensure that the quaternion remains on the surface of the quaternion hyper-sphere (S^3) [38].

Quaternions were the creation of Sir William Rowan Hamilton who became interested in extending algebra to higher dimensions. Complex numbers have the form:

$$a + bi \tag{1}$$

In equation 1 the ' i ' is a symbol denoting the square root of minus one. The scalar b allows any negative square root to be represented as a multiple of minus one, as demonstrated in equation 2.

$$2i = 2\sqrt{-1} = \sqrt{-2} \tag{2}$$

This part of the complex number (bi in equation 1,) is called the imaginary part, the other (a in equation 1) is the real part. Imaginary numbers are so named as there is no square root of a negative number as any number squared is positive. This is difficult to picture, as were negative numbers before the creation of the number line. In 1833 Hamilton noted that the sign only connected the two components and they could in fact be written with notation similar to that used for Cartesian coordinates. Examples of this are shown in equations 3 and 4.

$$a + bi = (a, b) \tag{3}$$

$$a - bi = (a, -b) \tag{4}$$

This combined with the earlier ideas of Gauss (1831) and Wallis (1685) led to the creation of the complex plane, also known as Argand Diagram after J. R. Argand who published a graphical representation of complex numbers in 1806 [48]. An example of such a diagram is shown in Fig. 1 this also shows the alternative angle length $\langle \theta, r \rangle$ representation, if the length of r is fixed then the variation of θ describes a circle this sparked Hamilton's interest in complex numbers for rotational parameterisation.

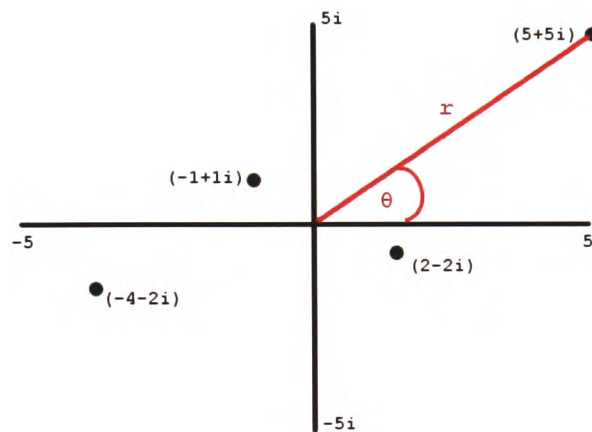


Fig. 1 - An Argand Diagram the x-axis is the familiar number line with negative and positive numbers while the y-axis depicts the imaginary component. The alternative representation is shown in red.

Hamilton tried unsuccessfully for several years to use two imaginary and one real component to describe rotation. Hamilton's epiphany came while walking past Broome Bridge in Dublin in 1843 en route to a meeting of the Royal Irish Academy. He realized that three imaginary components were required with the following properties (equations 5-7).

$$i^2 = j^2 = k^2 = -1 \quad (5)$$

$$ij = k \quad (6)$$

$$ji = -k \quad (7)$$

The components also display the cyclic permutation $i \rightarrow j \rightarrow k \rightarrow i$, (if a constant is added to the first component it generates the next and so forth finally it generates itself.) The quaternion itself takes the form:

$$q = a + bi + cj + dk \quad (8)$$

This is often condensed into the notation (s, v) where s is a scalar and v a vector. The following quaternion operations were derived: multiplication (equation 9), conjugate (equation 10) and magnitude (equation 11).

Multiplication:

$$q_1 q_2 = (s_1 s_2 - v_1 \bullet v_2, s_1 v_2 + s_2 v_1 + v_1 \times v_2) \quad (9)$$

Conjugate:

$$q = (s, v) \text{ becomes } \bar{q} = (s, -v) \quad (10)$$

Magnitude:

$$q \bar{q} = s^2 + |v|^2 = |q|^2 \quad (11)$$

In mathematics a group is a set of numbers with a rule representing their multiplication, such that the result is a member of group. A subset of the quaternion group is closely related to the group of rotation matrices. These are the unit length quaternion, their magnitude is always one and this constraint has to be ensured for the quaternion to map to a valid rotation [43].

Distance metrics in quaternion space can be defined in a number of ways arcs, angles and linear distances can be used [44]. The angle between quaternions in quaternion space and four-dimensional Pythagorean distance can be used as distance metrics. Indirect measurement based on the resulting three-dimensional difference between

rotated vectors can be used in some cases use of such techniques is limited due to the nature of the quaternion hyper-sphere.

A number of authors have implemented quaternion constraints, a set of simple quaternion based constraints were implemented by Lee [6], these simple constraints could be combined into more complex ones. Interrogation of these shapes (to ascertain the validity of a configuration,) is presented but no method of calculating a correction to the nearest valid constraint [6]. Lee's method relies on decomposing the quaternion into two quaternions representing planar rotation, based on this approach constraint systems capable of correction have been developed. Liu and Prakash [3] used a sampling approach to create boundaries in the tangent space and clamp orientations to these boundaries. Johnson [2] used a statistical approach to create both joint constraints and pose constraints. A corrective component was implemented by recursively moving an invalid point toward the mean of the sampled valid configurations. Johnson's approach again relies on projecting unit quaternions in to a tangent space. Herda *et al* [4, 5] used a three dimensional iso-surface reducing the dimensionality of the quaternion and implemented an iterative joint correction process.

2.1.3.5 Swing and Twist

The swing twist representation has been used extensively in the description of ball and socket joints, common in anatomy and robotics. This is not a parameterisation of rotation like the above but has been used in the representation of joints. The rotation of the limb is considered its swing, while rotation around the limb is considered the twist.

Specifying the swing component using axis angle rather than Euler angles reduces the effect of singularities on this parameterisation [37]. Further problems are caused by induced twist where successive swing rotations result in a change in the twist of the joint that would not have been present in a direct rotation. Additional computational expense is incurred to remove the effects of this phenomenon [37].

The twist component can be simply constrained using Euler constraints which may be a function of the swing component [37]. The swing component can be constrained using techniques such as spherical polygons [36, 37].

2.2 Neural Networks

Artificial Neural Networks (ANNs) are inspired by the structure of the human brain. Like biological neural networks they are composed of neurons linked together to form complex networks. However, they are significantly different in terms of their complexity and their method of communication. Neural networks are typically initialised to a random position in the search space from this position they attempt to reduce the error present in the network moving towards a minima.

There are many types of network architecture, from auto-associative memories such as the Hopfield network to unsupervised networks such as Kohonen's SOM (Self-Organising Map)[49].

Deciding whether a joint configuration is valid or invalid (the validity of the constraint,) can be considered as a classification problem where joint configurations are classified into two groups. Coit *et al* [50] applied a classifying neural network to decide based on a number of inputs if soldering should take place in a industrial system. In later sections Support Vector Machines (SVMs) are considered this complex machine learning technique has been shown to be superior to both neural networks and statistical classifiers on a number of classification problems [51, 52].

In the case of corrective constraints the neural network attempts to approximate a function relating the current configuration with the amount of correction required. For corrective constraints in multiple dimensions it is clear that the neural network must approximate a discontinuous vector field. A number of approaches have successfully used neural networks to approximate vector fields [16, 53-58]. In later sections feed forward neural networks, their topological evolution by genetic algorithms and finally their application to vector field approximation are considered.

2.2.1 Feed-Forward Neural Networks

Feed-forward network architectures such as that of the Multi-layer Perception (MLP) have been popular since the mid eighties when advances in their construction made them applicable to a new range of problems [59]. These are trained to give certain outputs in response to given inputs by repeatedly adjusting the strengths of the interconnections between neurons within the network (a number of training methods have been developed [60-62]).

The Multi-layer Perceptron is one of the simplest neural network architectures consisting of a number of nodes with weighted interconnections. Each node receives inputs along its connections, which are scaled from their source according to a weighting. On receiving these inputs it calculates their sum and transforms this input via an activation function to an output value [59]. The term Multi-layer Perceptron is often used to describe a feed forward neural network trained via back-propagation though there is little similarity between the Multi-layer Perceptron and its limited predecessor the Perceptron [59].

Many aspects of the networks structure and the structure of its neurons can influence the networks performance. The effect of the activation functions of neurons within the network is discussed in detail later in this chapter. The topology of the network (the way neurons are connected) determines the way computation proceeds and impacts on performance [49]. Biological neural networks are mostly feed forward, however some interconnections between nodes of the same layer exist as well as feed back connections and inhibitory nodes inspiring a plethora of network topologies [49].

Fully connected neural networks are the most general kind of architecture, where each node in the network is connected to every other node including itself. Despite their generality the use of such networks is rare due to the large number of parameters (weights) requiring training and the biological implausibility of its structure [49].

There are a number of feed-forward neural network topologies, each consisting of neurons in layers labeled either numerically or alphabetically with the input layers labeled 0 or i respectively.

Layered network – each node in the lower layers are connected to each node in all higher layers and to neighboring nodes in their own layer [49].

Acyclic network – a subclass of the layered network here no connections exist between nodes in the same layer [49].

Feed forward network – these are amongst the most common neural networks in use so much so that the term neural network is often used to describe this topology alone [49]. These networks have connections from each node in a lower layer to each node in the next layer.

This work uses generalized multi-layer Perceptrons (GMLPs) as used by Mayer and Schwaiger [63, 64] also described as fully connected feed forward neural networks by Yao and Liu [65]. These are much like layered networks with connections between each node in a lower layer with all nodes in all higher layers. Unlike the layered network there are no connections between nodes of the same layer. A single bias node is used which is connected as an input node, i.e. with connections to all hidden and output nodes.

Artificial neural networks are made up of artificial neurons, these typically have one or more input and output connections depending on the layer in which they are found. A weighted sum of the nodes inputs is modified via a transfer or activation function (sigmoid in the above example) and this is passed as the output to the next layer. The following example is based on that presented by Mehrotra, Mohan and Ranka [59].

The sum of the weighted neuron inputs (*net*) is defined as (equation 12).

$$net_l = \sum_{i=1}^n w_{l,i} x_{p,i} \quad (12)$$

Here $x_{p,i}$ is the input and $w_{p,i}$ the input weight for layer l pattern p . In this case n is used to describe the number of inputs for summation. Where the activation function is sigmoidal, the output of the neuron can be defined as (equation 13);

$$o_{p,l} = S(net) = \frac{1}{1 + e^{-net}} \quad (13)$$

In the above equation (13) $o_{p,l}$ is the output, e is the exponential function. In the following example a simple neural network with three layers i, j and k is presented. To ‘fire’ the neural network, that is to get an output for a given input the input nodes are set to the values of the input pattern. In this case there is no transformation and the outputs are weighted to form the input of the next layer, this process continues until the outputs of the final layer have been calculated, the process for a single node is shown in Fig. 2.

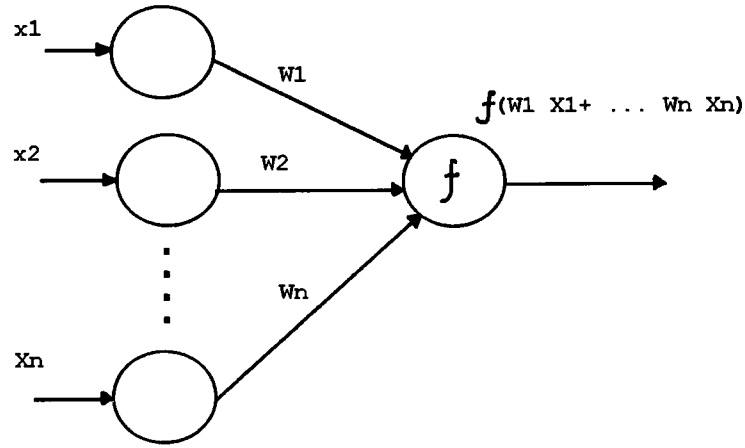


Fig. 2 - Weighted input summation [59]

The interaction of inputs, weights and functions to give the output can be described using equations. Nodes in the input layer (layer i) are a special case here the inputs are passed on without applying an activation function. This is shown in equation 14, here the subscript p refers to the pattern number, i and j represent the layer and x is the input to the given layer.

$$x_{p,j} = x_{p,i} \quad (14)$$

The equation for the hidden layer (layer j , as shown in equation (15),) shows some additional components. S is the sigmoid activation function applied to the sum of the

weighted inputs, c is a count of the number of weighted inputs to the layer. The weight from the input layer (i) and the hidden layer (j) is represented by w_{ji} .

$$x_{p,k} = S(\sum_{c=1}^n w_{ji,c} x_{p,i,c}) \quad (15)$$

For nodes in the output layer (layer k) the equation shown in equation 16 includes the output of this layer. This will be one of the network outputs and is denoted by an o , the weights between the hidden layer (j) and the output layer (k) are represented by w_{kj} .

$$o_{p,k} = S(\sum_{c=1}^n w_{kj,c} x_{p,k,c}) \quad (16)$$

Feed forward neural networks are trained using algorithms such as the back propagation algorithm. The following brief description of the back propagation algorithm, based on the example neural network above by Mehrotra, Mohan and Ranka [59].

Once the neural network has fired error for each of the output nodes can be calculated. In this example MSE an error measurement based on the norm of the difference vector between the desired neural network output (d_p) and the actual output (o_p) is used. There is however more than one vector, there is one for each of the K outputs and for each of the P patterns. These are combined using a sum of the squared error values, this provides an error function which can be differentiated (unlike the absolute error) this is essential for weight update via gradient decent [59]. The equations for MSE and SSE are shown as equations 17 and 18 respectively.

$$SSE = \sum_{p=1}^P \sum_{j=1}^K (|o_{p,j} - d_{p,j}|)^2 \quad (17)$$

$$MSE = \frac{1}{P} \sum_{p=1}^P \sum_{j=1}^K (|o_{p,j} - d_{p,j}|)^2 \quad (18)$$

The output of the neural network is a function of all the weights (w) present therefore the network error (E) is also a function of these weights. Differentiation of E with

respect to w equation 19 gives an error gradient. This gradient relates error and weight change, the weights are changed in the direction coinciding with decreasing error.

$$-\partial E / \partial w \quad (19)$$

Rather than calculate the update for all the weights (Δw) required for w this calculation is performed for each connection from the output to the hidden layer and from the hidden layer to the input layer. The corrections obtained are used to update the respective weights. This is known as the *generalised delta rule* [59].

The formulation of the rules for updating the weights relies on calculating a number of partial derivatives and evaluating them using the chain rule. To differentiate between E and w it is noted that E is dependent on the network output (o), which is itself dependent on w these partial differentiation links are evaluated using the chain rule. Mehrotra, Mohan and Ranka [59] cover the formulation of these equations. The equations derived for the update of weights connecting in the hidden and output layers are as follows (equations 20-23).

$$\Delta w_k = \eta * \delta_k * x_j \quad (20)$$

$$\Delta w_j = \eta * \mu_j * x_i \quad (21)$$

where

$$\delta_k = (d_k - o_k) S'(net_k) \quad (22)$$

and

$$\mu_j = (\sum_k \delta_k w_{kj}) S'(net_j) \quad (23)$$

In the equations above (equations 20-23), the subscript p has been omitted to maintain clarity. Here η is a user-controlled variable that scales weight updates known as the learning rate.

The equations for calculating the weight updates between the output and hidden layer (Δw_{kj}) and hidden layer and input layer (Δw_{ji}) are very similar as shown in equations 20 and 21. Both are the product of the input to the layer (x_i or x_j) the learning rate (η) and a generalized error term (δ_k or μ_j).

The generalized error term for the nodes of the output layer δ_k is proportional to the amount of error multiplied by the derivative of the output node with respect to the input node as shown in equation 22. The generalized error term for the hidden nodes μ_j is proportional to the amount of weighted generalized error for the output nodes multiplied by the derivative of the output node with respect to the input node as show in equation 23.

2.3 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) were introduced by Vladimir Vapnik and rely on the principle of structural risk minimisation (SRM) [66]. Their key advantage is in their training technique, which aims to minimise both error and network complexity and hence maintain its ability to generalise [67]. The SVM attempts to identify a mathematical function that produces the minimum error based on a cost function. Unlike traditional neural network training which attempts to solve a non-convex unconstrained minimisation problem [67, 68] the SVM minimises both the current error and learning machine complexity by solution of a quadratic programming problem with linear constraints.

SRM states that the current error (after exposure to some training patterns) and the complexity of the network contribute to the generalisation error (error after infinite training patterns) [69]. Neural networks focus on reducing the current error ignoring the network complexity, an increase in which leads to over-fitting and therefore and

increase in generalisation error. SVM training reduces both the current error and the complexity of the network [69].

An optimal linear hyper-plane (O in Fig. 3) is created to divide linearly separable data this is placed between two hyper-planes (H1 and H2 in Fig. 3), which delineate the boundaries of the individual datasets [70]. The patterns on these hyper-planes (without which the solution would change) are identified as support vectors [68]. The training process aims to maximise the margin between H1 and H2, SVMs are usually trained by minimising a quadratic problem under constraints [66, 70]. Increasing the size of the margin theoretically reduces the complexity of the machine, as well as reducing the current error (number of misclassified patterns) a reduction of both terms leads to a reduction of the generalisation error [69].

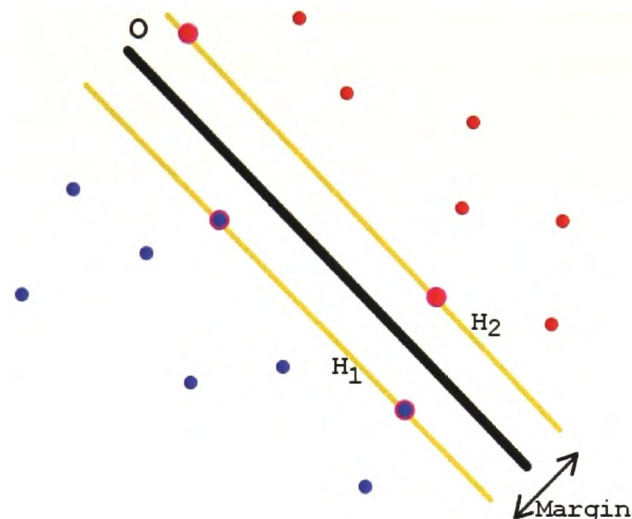


Fig. 3 - The figure shows the hyper-plane O separating the two sets of data shown as a thick black line. It also shows the margins H1 and H2 depicted as yellow lines upon which the support vectors (shown with a pink hi-light,) lie.

This technique is only suitable for linearly separable cases it is by no means general. Both non-separable and non-linear cases are dealt with using simple additions to this technique. In the non-separable case slack variables are introduced, these relax the constraints governing the distance of the hyper-planes from the support vectors with the penalty of further cost [68].

SVMs capitalise on the only reference to the training data in the in the optimisation of the hyper-plane being through a dot product in creating boundaries in the linearly inseparable case. A mapping is used which maps the data to some other possibly infinite Euclidean space this mapping is known as a kernel function.

In summary, SVMs attempt to separate sets of data with the maximum distance from points on either side. SVMs utilize kernel functions, these move the points into a higher dimensional space this has the effect of spreading the points reducing the complexity of separation.

A version of SVM for regression was proposed by Vapnik, Golowich and Smola called Support Vector Regression, (or SVR) [71] and considered the application of support vector methods to function approximation. The classification method shown above only depends on a subset of the data as the cost function ignores points that lie beyond the margin. The regression method also depends on a subset of the data but ignores points that are close to the boundary (within the threshold) [71]. To date much work has been done improving on the simple SVM shown above for both classification [72-74] and regression [75].

SVMs suffer from several limitations. One of the key limitations is the choice of kernel function, trial and error (or prior knowledge) is often required to identify the best kernel function for a dataset [68]. The computational cost of training and testing is high, though successful attempts have been made to reduce both the testing and training time [68, 72, 76]. The quadratic programming problem (quadratic optimization) is usually quite complex and therefore suspect to stability problems [76]. Attempts have been made to reformulate the quadratic optimization to improve stability and reduce computational complexity [76]. There are also occasions where SVMs select sub-optimal support vectors for categories within the training set. A multi-pass system that separates the identification of the best candidates for support vectors prior to SVM training has been developed by Masuyama Nakagawa [77].

2.4 Genetic Algorithms

Genetic Algorithms (GA) are search algorithms that utilize the mechanics of natural selection first developed by John Holland and his students at the University of Michigan. Each generation contains a number of blueprints for an individual called the genotype. The performance of these individuals is measured against some metric. New genotypes are created by retaining information from the strongest (reproduction) and swapping genetic information between pairs of individuals (crossover). The occasional new genetic feature is introduced and this is called mutation [78].

This method has advantages over traditional optimisation and search methods. Calculus based methods are local in scope and search for the local optimum only. They are also dependent on continuity and derivative existence in the search domain, making them suitable only for a limited problem domain. Enumerative and random searches are inefficient, though there is a random component to genetic algorithms [78].

Genetic Algorithms encode the parameter set rather than using the parameters themselves, hence a genotype (blueprint for the individual) is created from the phenotype (their characteristics). Each individual may evaluate a different part of the search space rather than a single point as is the case in other approaches, reducing the risk of becoming trapped in local minima. Other approaches rely on using deterministic rules, often derivatives, to evaluate the current solution, genetic algorithms use an objective function. The use of an objective function allows the comparison of local minima in a multi-modal search space. Genetic algorithms move towards a solution using probabilistic transition rules, though the direction is not decided at random [78].

Goldberg [79] shows by means of similarity metrics the workings of genetic algorithms. These metrics are called schemata (similarity templates), schemata are similar to masks placed over the genome they highlight commonality between genomes. For example the binary genome 0110110 and 0100001 are both associated with the 01***** schema, where the * represents information which is not part of the schema. Schemas have an order and defining length. The distance between first and last values exposed in the

mask is termed the length, while the order of the schema refers to the explicitly of the schema that is the number of values exposed in the mask.

The fundamental theory of genetic algorithms states that “high-performance, short-defining-length, low-order schemata receive at least exponentially increasing numbers of trials in successive generations” this is known as the building blocks hypothesis [79]. This is due to several factors;

1. Reproduction allocates more copies to the best schemata.
2. Crossover does not frequently disturb short chains where as the cross over point may fall in the middle of large ones and split them in two.
3. Mutation is infrequent and has little effect.

In essence the small high-performance (low error) schema become partial solutions to the problem (or building blocks) which the genetic algorithm then discovers new solutions by speculating on how these can be best recombined [79].

2.5 Evolved Artificial Neural Networks

The human brain, which inspired the creation of artificial neural networks, has a complex and bespoke structure that has evolved over many thousands of years. Evolved neural networks represent the application of genetic algorithmic techniques to neural network creation to enhance the specificity of the neural network to a problem or environment [80].

Early Evolutionary Artificial Neural Networks (or EANNs) approaches considered the evolution of neural components of the artificial neural network such as its structure, interconnecting weights, nodes and learning rules [80]. Inspiration from natural (human) evolution has lead to approaches where training patterns, learning scheme and other factors, such systems are described as Artificial Neural Systems (ANS) [80].

This following sections focus on the evolution of the structure and activation function of the neural network. Activation function evolution is especially interesting as it can improve the learning of local features (such as the discontinuities of the vector field)

within feed forward networks. This is followed by an introduction to local and global learning, with particular attention to the creation of neural networks that display both. Finally existing application of neural networks to vector field approximation are reviewed.

2.5.1 Topology Evolution

Huber, Mayer and Schwaiger [81] state that despite the successful application of Multi-layer Perceptron ANNs, no analytical rule has been discovered governing the optimal topology of the network. They also observe that improvement in approximation often results in a loss of generalization capabilities and that smaller ANNs with low connectivity show better generalization capabilities than more complex networks.

A number of authors have attempted to solve to this problem by means of evolutionary techniques to evolve a topology suited to the problem at hand. There are two approaches identified by Yao and Liu [53], the evolution of “pure” architectures where weights are evolved separately and the evolution of weights and architectures together. In both cases information regarding the topology of the network is encoded as the connections made by the nodes of the network [53].

Huber, Mayer and Schwaiger [81] use genetic algorithms which searched for a problem-adapted neural network topology. A hybrid system is employed using genetic algorithms to evolve the topology with the *Resilient Back-propagation* [60] learning algorithm to train the network weights. This is an example of “pure” architecture evolution with direct encoding. There are some issues with such approaches as identified by Yao and Liu [53], who observe that when training the training method may find different minima in a multi-modal error surface from the same initialised weights.

The problems encountered in “pure” architecture evolution can be alleviated by evolving both the weights and the architectures simultaneously, using a one-to-one mapping from genotype to phenotype (where the phenotype is the evolved network[53]). Difficulties here arise in the encoding of networks, as in some cases networks can have different genotypes but produce the same phenotype making evolution inefficient [53].

2.5.2 Activation Function Evolution

Neural network performance is greatly affected by the choice of activation function present in the neurons that comprise the network. In biological neural networks, specialisation of neurons takes place [82] simulating this with mixed [53, 65, 83] and adaptive activation functions [63, 64, 84] has provided improvements over classical architectures with fixed sigmoidal neurons.

A number of researchers have successfully improved on the results of classical sigmoid neurons using mixed activation functions. Successful combinations include; Gaussian and Sigmoid activation functions, both fixed in separate layers [85] and evolved in a single layer [65, 86]. Sigmoid and Sigmoid based jump approximation functions were used by Selmic and Lewis [58], these non-smooth activation functions produced good results in learning one dimensional discontinuous functions. However the nature of the activation functions made learning difficult and prior knowledge was required regarding the position of the discontinuities.

Yao and Liu [53] evolved neurons with sigmoid and Gaussian activation functions in the hidden layer using evolutionary programming techniques. More recently Mayer, Strapetz and Fuchs [83] produced a version of the NetGen system capable of selecting between multiple candidate activation functions, these included logistic, hyperbolic tangent and linear.

There is a wealth of research regarding adaptive activation functions. Several researchers have used adaptive sigmoid neural networks, where the parameters of the sigmoid function are modified during training giving the neurons limited specialisation capabilities [84, 87].

More recently research has focused on spline based activation functions. A spline is a function constructed from low order polynomial pieces connected at breakpoints (called knots) with certain smoothness conditions [82]. It is these knots that are modified by training or evolution to create specialised nodes and increase performance.

A number of different types of spline exist and several have been used as adaptable activation functions for neural networks. Some novel though limited approaches have been suggested using B-Spline neural networks [82, 88]. More recently catmull-rom cubic spline neural networks using algorithmic adjustment of the spline during learning (spline training) have been developed [84, 89, 90].

Cubic splines have been used by a number of authors, both trained [91, 92] and evolved [64]. Multi-dimensional cubic splines have also been used. Here there are as many dimensions to the spline surface as there are inputs, these are combined into a single input passed to the next layer [93-95].

Mayer [63] utilised a template based approach to cubic spline activation function evolution. This brings together pure activation function evolution and spline based activation function evolution. A number of spline based template functions are evolved as candidates for neuron activation functions in the network. This reduces the complexity of the genetic algorithm, as there are potentially fewer free parameters requiring optimisation.

2.6 Local and Global Learning Characteristics

Sample data or training data displays both local and global characteristics. Approaches that make use of these characteristics are described as local learning and global learning respectively. Global learning has a long and distinguished history, scientist have used global learning techniques to uncover the underlying mathematics that govern complex phenomena [96]. However, global learning methodologies often struggle to find the appropriate model and parameters to represent the observed data.

This has led to increasing interest in local models. Here the focus is on useful local information from the observed data [96]. It has been demonstrated that local learning is superior to global learning in many classification domains [96]. However local learning methods do not grasp the structure of the data which may be critical for generalization performance [96].

The illustration shown in Fig. 4 is similar to that shown by Huang *et al* [96]. The figure shows a decision line identified by a local learning technique. In Fig. 4(a) the spread of the data is fairly even, and there are a number of points used to make the decision regarding positioning of the decision line. However in Fig. 4(b) there are only two points identified, as being important to the boundary and much of the global information about the pattern distribution is lost. This is equally true in the case of function approximation, it may be more important to accurately describe the global data than to base the mapping on a few points considered important.

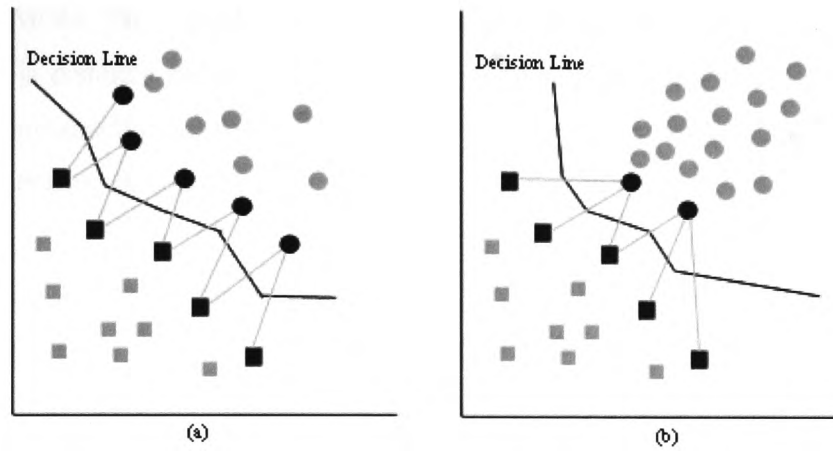


Fig. 4 - (a) An illustration showing local learning, where the decision boundary depends on a few selected points. (b) In this case local learning cannot grasp the trend present in the data. Darker markers indicate the points used in local learning.

Among the machine learning techniques which exhibit local learning are SVMs. Key to the power of this approach is the local nature of its learning, the updating of support vectors in one region does not disturb the learning in other regions [85]. However poor performance on global trends, often referred to as ‘over fitting’, has been recorded by a number of researchers and attempts have been made to minimise its effects [74, 77].

Feed forward neural networks such as the MLP and GMLP display global learning and often struggle to represent local features [85]. Due to the global nature of their learning the training of one part of the function may change the weights related to another part [85].

A combination of these two learning styles is required, a neural network where both global and local learning takes place leading to accuracy on local features while retaining generalization with regards to the global structure of the data. A number of approaches attempt to introduce components of global learning in neural networks which demonstrated good local learning, for example, the addition of a second layer of sigmoidal nodes to a Gaussian functioned neural network (RBF) was implemented by Shibata and Ito [85].

More recently, Support Vector Machines have been combined with the Minimax Probability Machine and Linear Discriminant Analysis to form the Maxi-Min Margin Machine (M^4). Their model tries to maximize the margin defined as the minimal Mahalanobis distance for all training samples while maintaining correct classification [96]. The introduction of this global distribution measurement improves the networks choice of decision boundary.

Alternatively aspects of local learning can be introduced into neural networks that display good global learning. Spline activation function neural networks display both global and local learning, here inter-nodal connections partition off areas of the dataset and the activation function becomes specific to local data [84, 95, 97].

2.7 Neural Network Approximation of Vector Fields

A vector field is defined as a mapping that assigns each input to an output via some vector function. Vector fields can be uniquely specified by giving its divergence and curl within a region, this is known as Helmholtz's theorem [98]. In mathematics vector fields typically involve a Euclidean position in two dimensions being mapped to a vector with direction and magnitude. They are used extensively to describe forces at a given point in two-dimensional space. They are however extensible to any number of dimensions in that a vector field can form a map between two vectors of equal dimensionality or of unequal dimensions, (a projection). For example the use of three dimensional vector fields for the exploration of complex problems is explored by Crawfis *et al* [99].

Dynamic behaviour (such as joint constraint) can be described as a change in state that is determined by a function dependent on the current state. There is clear similarity between the mappings required for vector fields and those involved in the description of dynamic behaviours such as joint constraint [54].

A one-dimensional vector field would simply consist of a function (as shown in equation 24.) A joint constrained in one dimension using an Euler angle can be described as a function (or one-dimensional vector field). The function is a discontinuous or piecewise linear function as it has points where there is no gradient, the discontinuities.

$$x \rightarrow f(x) \quad (24)$$

There are a number of practical applications for which the approximation of these functions by neural network has been attempted. Selmic and Lewis show that the inclusion of non-smooth neural network activation functions (sigmoid based jump functions) produce good results. In their work a feed forward network is trained via back-propagation to model friction compensation in industrial machinery. The weights connecting the nodes with non-smooth activation functions were fixed and their thresholds adjusted to correspond to the discontinuity based on prior knowledge [16, 58]. Radial Basis Function neural networks have also been used to overcome problems with friction [100].

A similar problem involving backlash compensations has been solved using a recurrent neural network using reinforcement learning [88, 101, 102]. Anderson [103] demonstrates the superiority of a modular neural network approach over reinforcement learning. In the modular approach the piecewise linear function is broken down into its linear components, (separated at the discontinuities.) Expert networks are trained for each linear part and a gate function or network used to decide which of the experts should be used.

Moving to two-dimensional vector fields, which could be used as a crude representation of a constraint, like the projected spherical polygons [40] discussed earlier. A two-

dimensional vector field used for joint constraint applications still has a discontinuous quality, in that a number of points exist (at the boundary,) where there is no gradient. Continuous vector fields in two dimensions have been trained using neural networks in the field of robot control. Here a vector fields representing path following for simple and more complex paths were trained using one and two hidden layer neural networks [54, 55]. A MLP neural network with input nodes, representing the robots position and two outputs, representing the directional change it needed to undertake to return to the path was used. The hidden layer of the neural network was composed of nodes with hyperbolic tangent (or bi-polar sigmoid) activation functions, and the output layer of nodes with linear activation functions, the neural network was trained using backpropagation.

Kuroe *et al* [104] suggested an alternative approach where an Adjoint neural network was used to learn continuous vector fields. This approach utilises the basis field simplification technique of Mussa-Ivaldi and Griszter [105]. The neural network is trained via a customised training algorithm that relies on aspects of vector field theory. Any continuous vector field can be shown to be composed of irrotational and solenoidal vector fields [106]. In the approach of Kuroe *et al* these are in turn expressed in terms a common multi-dimensional scaling function (another vector field) and two additional scalars. The scaling function and scalars are learned as part of the learning algorithm and can be recombined into the original vector field [104].

The techniques developed by Kuroe *et al* [104] were applied to flow field measurement from image data, a technique called Particle Imaging Velocimetry (PIV). An Adjoint neural network was used to approximate regions flow within artificially generated two dimensional smoke images [107].

Kulchin and Panov trained neural networks to learn two dimensional scalar fields for reconstructing data from fibre-optic measuring systems [57]. Again the hidden layer was composed of nodes with hyperbolic tangent (or bi-polar sigmoid) activation functions, and the output layer of nodes with linear activation functions. The neural network was trained using an enhanced backpropagation algorithm with simulated annealing to reduce the effects of local minima.

Evolutionary programming a technique very similar to genetic algorithms was used by Kim *et al* [108] to identify an appropriate path between an initial destination and an ideal direction in two dimensions. This technique utilised vector fields to describe attractive forces for the destination position and direction and repulsive forces for obstacles.

Mussa-Ivaldi and Griszter [105] found that the limb pre-motor control in the reptilian spine was arranged in discrete modules describing an equilibrium point for a limb using groups of antagonistic muscles. The stimulation of multiple groups leads to the superimposing of these individual modules suggesting that all combinations of posture for the limb are generated in this way. The authors make use of basis fields to describe fields of motion and imitate these discrete modules. Basis fields are the vectorial equivalent of local basis vectors, just as any vector in a vector field can be represented as a linear combination of its basis vectors a vector field can be represented as the linear combination of its basis fields. This technique can be used to simplify complex vector field representations.

Neural networks have been utilised for physics based animation by Grzeszczuk, Terzopoulos and Hinton [56]. In their approach complex forward dynamics equations required for physics based animation were replaced with neural networks, predicting the complex vector mapping (Φ) from the current state (s_t) to a future state ($s_{t+\delta t}$) based on the current state, the applied force (u_t) and external forces (f_t), (as shown in equation 25.)

$$s_{t+\delta} = \phi[s_t, u_t, f_t] \quad (25)$$

A key advantage of this approach is that the trained forward dynamics neural network mappings can be reversed by applying the chain rule of differentiation to obtain the inputs to the network given a resultant state. This is further exploited to move a limb towards a desired position utilising a gradient decent [56].

Grzeszczuk, Terzopoulos and Hinton [56] provide a detailed account of their network configuration and raise a number of issues regarding the capabilities of neural networks as vector field approximators. As the range of the inputs and outputs are large in

comparison to the range of the sigmoidal activation function normalisation of this range was impractical. Mapping from the current state to the difference between current and future states is more practical and by adding the approximated difference and current state the future state can be calculated. The range of the inputs to the neural network can deviate greatly adversely affecting neural network output, these were normalised and adjusted to have unit variance and zero mean. It is reported that neural networks attempting to train vector fields with high dimensionality (10+) required large numbers of hidden nodes (50+) and long training times (several CPU hours). The researchers suggest a natural sub-division to reduce the number of free parameters in each case. The neural network used for the forward dynamics has a single logistic sigmoid hidden layer and is trained via back propagation enhanced with a conjugate gradient algorithm.

The term Quaternion Vector Field is attributed in much of the literature to the visualization approach developed by A. J. Hanson, which reduces the four-dimensional quaternion to three dimensions for visualisation. Herda et al [4, 5] have implemented joint constraints based on this approach. In this thesis vector fields in quaternion space and indeed quaternion vector fields are considered as mappings of an input quaternion and an output quaternion via some function.

Research has also been undertaken towards specialized neural network architectures for solving Constraint Satisfaction Problems (CSP's) [109, 110]. These neural networks attempt to provide a general network for the solution of any CSP and consist of a number of node clusters, one for each input that have inhibitory links between them. They have been shown to be faster than conventional methods (sequential heuristic search) and have execution times of tens to hundreds of nano seconds compared to more than 20 hours for the more conventional approach [109]. These networks deal with a high number of binary inputs, outputs and constraints, successful preliminary work is also shown for non-binary problems [109].

Few of the existing approaches have been applied to discontinuous vector fields. A possible reason for this is an inherent weakness in many neural networks. The learning of individual patterns has an effect on the patterns already learned due to the update of weights shared between neurons [111]. This is known as the "stability-plasticity" problem, the neural network needs to be sensitive to but not seriously disrupted by new patterns [111]. Some interference is acceptable and has little effect on the training,

however in extreme cases “catastrophic interference” occurs, here the learning of a new group of patterns damages the patterns previously learned by the neural network [111].

French [111] states that catastrophic interference is largely a consequence of the overlap of internal distributed representations. Hidden neurons are responsible for this internal representation and catastrophic interference arises when they attempt to differentiate between overlapping input.

2.8 Principle Component Analysis

Principle Component Analysis or PCA is a statistical technique used in a number of domains, like many other multivariate statistical analysis techniques it can be used to analyse the relationships between the variables of large multivariate data sets. PCA provides an analysis of the multi-variant structure of the data giving an indication of the relationship between variables and the components contributions to these relationships [112].

At a high level PCA gives two important products, firstly a series of vectors known as the characteristic vectors or eigenvectors. These are orthogonal vectors that identify the directions in which variance takes place within the dataset. PCA also gives a set of values associated with each eigenvector known as characteristic roots, latent roots or eigenvalues, these values give the variance attributed to the associated vector [112].

A number of univariate techniques are introduced as a precursor to multivariate techniques and PCA. The mean (or \bar{x}) is defined as the summation of the elements of the data set x where x_i is the i th element of the dataset divided by the number of items in the dataset n as shown in equation 26 [113].

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad (26)$$

There are two other statistical measures of variation for univariate data sets that are of interest the first is the Standard Deviation denoted by the symbol s . This is the average

distance from the mean of the dataset to a point this is calculated by taking the square root of the summation of the squared differences between the mean and each point as shown in equation 27 [113]. The average of the squared differences is calculated using one less than the number of numbers n , as this provides a more accurate estimate for samples of data representing larger sets [113].

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{(n-1)}} \quad (27)$$

The second univariate statistical measure of interest is the variance of the dataset, which describes the spread of the dataset. It is in fact the sum of the squared distances between the mean and the individual data points. Its formula (shown in equation 28) is very similar to that of the standard deviation [114].

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{(n-1)} \quad (28)$$

Mean, variance and standard deviation are univariate and are not suitable for use in the analysis of multivariate data. A related measure the covariance can be used to describe the variance of one dimension with respect to another. The formula for the covariance of two datasets is given in equation 29 [115]. Note that here the product of the difference between the i th data points and their respective means has replaced the square of the difference between the i th data points and the respective means of two different sets of data. Variance is a measure of that variation of a dataset with respect to itself and covariance the variance of two datasets with respect to each other.

$$\text{cov}(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(n-1)} \quad (29)$$

It is important to note at this point that $\text{cov}(x,y)$ gives the same result as $\text{cov}(y,x)$ as only the order of the multiplication changes and multiplication is commutative. Covariance only gives measure of the variance between two dimensions this can be extended to more than two dimensions using the covariance matrix. For a dataset of n dimensions the covariance matrix (an $n \times n$ matrix) is shown in equation 30. The format of the equation is based on that given by Jackson [112] though this has been modified to aid clarity.

$$\text{cov}(x, y, \dots, n) = \begin{bmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \cdots & \text{cov}(x, n) \\ \text{cov}(y, x) & \text{cov}(y, y) & & \text{cov}(y, n) \\ \vdots & \vdots & & \vdots \\ \text{cov}(n, x) & \text{cov}(n, y) & \cdots & \text{cov}(n, n) \end{bmatrix} \quad (30)$$

The covariance matrix is a symmetric, non-singular square matrix it has both eigenvectors and eigenvalues. Eigenvectors when multiplied with a matrix are scaled rather than being rotating or translated. The resulting eigenvectors are scaled versions of the original the scale of each eigenvector is termed its eigenvalue. The eigenvectors and eigenvalues of a matrix can be identified by a number of methods [112, 116, 117].

The following is a brief description of eigenvectors and eigenvalues and the steps required in their identification. A matrix A is multiplied by a vector x their product is the vector B . However on closer examination B is a scaled version of x . The matrix A contains the eigenvectors while the eigenvalue (λ) is the scaling factor x has undertaken, this can be expressed as shown in equation 31 [118].

$$Ax = \lambda x \quad (31)$$

Eigenvalues and Eigenvectors can only be found for square matrices, there exist at least one eigenvalues and at most n eigenvalues for an $n \times n$ matrix where x is non-zero. Any solution for λ where x is non-zero is called an eigenvalue or characteristic value of the matrix, the corresponding solutions of x for given values of λ are called the characteristic vectors [118].

The determination of Eigenvectors and Eigenvalues is illustrated by a simple example based on that given by Kreyszig [118]. The first step is to introduce the example matrix A and express equation 31 in these terms as shown in equations 32 and 33.

$$A = \begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \quad (32)$$

$$Ax = \begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (33)$$

Equation 33 can be expressed as a set of simultaneous equations as shown in equations 34 and 35.

$$-5x_1 + 2x_2 = \lambda x_1 \quad (34)$$

$$2x_1 - 2x_2 = \lambda x_2 \quad (35)$$

Rearranging the terms of equations 34 and 35 gives equations 36 and 37.

$$(-5 - \lambda)x_1 + 2x_2 = 0 \quad (36)$$

$$2x_1 + (-2 - \lambda)x_2 = 0 \quad (37)$$

This can be expressed as a matrix (equation 38) the system has now been expressed as shown in equation 39.

$$\begin{bmatrix} (-5 - \lambda)x_1 & 2x_2 \\ 2x_1 & (-2 - \lambda)x_2 \end{bmatrix} = 0 \quad (38)$$

$$(A - \lambda I)x = 0 \quad (39)$$

Where I is the identity matrix. This is a homogeneous linear system, by Cramer's theorem it has a non-trivial solution, $x \neq 0$ if its coefficient determinant is zero.

$$D(\lambda) = \det(A - \lambda I) = \begin{vmatrix} -5 - \lambda & 2 \\ 2 & -2 - \lambda \end{vmatrix} \quad (40)$$

$$= (-5 - \lambda)(-2 - \lambda) - 4 = \lambda^2 + 7\lambda + 6 = 0$$

$D(\lambda)$ is the characteristic determinant or if expanded the characteristic polynomial. The solutions of this quadratic equation and hence the values of λ are -1 or -6 . These are the eigenvalues of A .

Substituting -1 into equation 41 values can be identified for x_1 and x_2 .

$$\begin{aligned} -5x_1 + 2x_2 &= -1x_1 \\ 4x_1 &= 2x_2 \\ x_2 &= 2x_1 \end{aligned} \quad (41)$$

Choosing a value for x_1 of 1 the resulting eigenvector is shown in equation 42.

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad (42)$$

Substituting -6 into equation 43 values can be identified for x_1 and x_2 .

$$\begin{aligned} -5x_1 + 2x_2 &= -6x_1 \\ 1x_1 + 2x_2 &= 0 \\ x_2 &= -x_1 / 2 \end{aligned} \quad (43)$$

Choosing a value x_1 for of 2 the resulting eigenvector is shown in equation 44.

$$x = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \quad (44)$$

Direct calculation of the eigenvectors and eigenvalues becomes cumbersome in cases where there are more than three dimensions a number of alternative methods have been suggested to speed up this process [118, 119].

2.9 Conclusion

The wealth of joint models uncovered by the literature review leads to the conclusion that although the reproduction of anatomical joints is a modelling problem, there are a number of fields where models of the human anatomy are required, such as animation, simulation and medicine.

While proximal constraints (those holding the joint together) are not often problematic, rotational constraints such as those required modelling the flexion and extension of limbs are often more difficult to implement. It has been reported that joint constraints in animation are particularly underdeveloped and in the absence of a single model capable of modeling all joint constraints a number of specialized joint constraint approaches (for joint structures) have been combined to produce full body systems [1].

Several approaches, Korein [36], Engin *et al* [41] and Manurel *et al* [10] use three-dimensional polygons to represent the boundary between valid and invalid rotations. The three-dimensional polygons are not exact representations of the data, but are best fitted to the data points from observation. Huang *et al* [120] stored data in a database rather than a geometrically described boundary / region. Points which are not in the database cannot be interpolated, unlike other approaches [10, 36, 41] which use geometrically defined boundaries between valid and invalid points.

Despite by their nature being simplifications of the constraints boundaries identified, these approaches can produce reasonable approximations of joint function. However the rotational representations used often contain singularities, or have other limiting

factors. Quaternions are a much more useful representation though it is difficult to define quaternion based constraints. Acknowledging the singularity free nature of the quaternion parameterisation several researchers have attempted to implement quaternion-based constraints.

Herda *et al* [4, 5] reduced the dimensionality of the quaternion data and fitted a boundary to a cloud of valid points. This approach encountered problems in fitting the boundary to the points due to gaps in the sampled data. More importantly the correction of points to the boundary described is a non-trivial problem. An iterative approach is suggested this however is inefficient and may not actually identify the closest valid rotation.

Lee [6] implements several simple constraints in using decomposed quaternions only binary constraints are provided and no method of ascertaining the appropriate correction is suggested. Liu and Prakash [3] build on this approach allowing more complex constraints. Johnson [2] used a boundary based on the maximum diversion from the mean in the quaternion tangent space. Correction to this boundary was defined based on iteratively moving the incorrect point closer to the mean of the valid points. As with Herda *et al*'s approach this approach is inefficient and may not identify the closest point.

The mapping of a quaternion to the tangent space requires the pre-processing and conversion of the quaternion prior to its constraint. Converting the quaternion to another format for constraint resolution purposes is inefficient. The tangent space is a local “linearization” (approximation) of the unit quaternion group, as with any parameterization of a non-Euclidean group by a subset of Euclidean space it contains singularities which must be avoided [3, 46].

There is a recognized requirement to minimise the conversion between rotational parameterisations within a system this penalises the use of many complex joint models in a single system [2]. Neural network based constraints present the opportunity to use a single constraint system and quaternion rotational parameterisation regardless of the joint structure. None of the joint constraint approaches discussed have utilised ANNs to provide an accurate model of individual joint constraint.

ANNs are powerful analytical tools they offer significant performance advantages over traditional methods where complex calculations must be carried out to calculate the correction required to bring an invalid configuration to a valid one. Typically firing a neural network involves a succession of multiplications and additions this allows a vector based or hardware-based implementation. In addition to their execution speed they offer similar benefits to the quaternion approach of Herda [5] and the three dimensional polygon approaches [9, 17, 36] in that they can extrapolate and interpolate from measured data.

ANNs can be thought of as a store for data, the training patterns form clusters of data in multi-dimensional feature space. The feature space is divided to best accommodate the training data. Once trained, the ANN can then extrapolate in areas of sparse or absent data in response to patterns not present in the training set. This may give an advantage over Herda *et al* [5] who's work suffered due to sparse area's of data, and the three dimensional boundary approaches [10, 36-38, 41] which require an even sampling of points to accurately best fit a boundary.

Two types of constraint are identified for implementation, constraints which indicate the validity of a given configuration termed *binary constraints* and constraints which give a correction to the nearest valid configuration termed *corrective constraints*. Binary constraints can be considered a classification problem where configurations are classified as valid or invalid. Corrective constraints can be considered as discontinuous vector fields several researchers have demonstrated the capabilities of neural networks in learning vector fields [16, 54, 100-103].

The SVM neural network was selected to classify valid and invalid rotations. The SVM approach aims to minimise both the error on the training set and the complexity of the SVM thus minimising generalisation error. An implementation of the SVM architecture with the performance improvements as indicated by Joachims is available [72].

Corrective constraints are a discontinuous vector field approximation problem and though both the local and global characteristics of the data should be approximated it must first be established that the global mapping between valid and invalid patterns can be trained. Once this is established improving the results with the inclusion of local learning may be considered.

Feed forward Multi-layer Perceptrons have a number of qualities that make them well suited as a starting point for this research. They have been extensively studied and their capabilities are well documented. The process of firing a neural network is simple and can be implemented in hardware [121, 122], it can also be easily distributed giving potential speed increases over traditional methods in the use phase [123]. The nature of the quaternion vector field required for joint constraint is unknown MLPs are capable of learning mappings without prior knowledge of the functions which relate data [124].

Selmic and Lewis successfully approximate discontinuous functions using backpropagation trained MLP neural networks with sigmoid and sigmoid jump activation functions [16, 58]. Researchers have also utilised feed forward MLP neural networks to approximate continuous vector fields in a number of dimensions. It has been shown that neural networks can learn continuous two dimensional vector fields, in this approach bi-polar sigmoid activation functions were used and networks were trained via backpropagation [54, 55]. Similarly two dimensional scalar fields have been approximated by Kulchin and Panova [57] again using bipolar sigmoid activation functions and an enhanced backpropagation algorithm. Grzesczuk, Terzopoulos and Hinton [56] successfully approximated complex multi-dimensional vector fields in their approach sigmoid activation functions were used along with an enhanced version of the backpropagation algorithm.

Topological evolution attempts to maximise performance by minimising both network error through weight adjustment and generalisation error by reduction of the neural network complexity. An implementation of these techniques (called NetJEN,) is available based on published research [63, 64, 80, 81, 83, 125-128]. NetJEN also provides activation function evolution from a number of candidate activation functions based on Mayer, Strapetz and Fuchs [83] and template based spline activation function evolution following Mayer and Schwaiger [63].

The discontinuous nature of the mapping between input and output may result in internal representations that overlap increasing the difficulty associated with the learning of such vector fields. Seipone and Bullinaria [129] suggested that the use of Artificial Neural Systems (evolved neural networks) reduces effect of interference in addition to improving performance.

Grzeszczuk, Terzopoulos and Hinton [56] have demonstrated that neural networks are best able to model vector field approximations when the magnitudes of the input and output vectors are similar and of unit variance and zero mean. Also that modelling the relationship between a state and a state change provided more comparable magnitudes with regards to input and output value than state-to-state mappings. The use of a quaternion representation to model a discontinuous vector field describing a mapping between the current rotation and the required correction neatly avoids a number of these complications hence the inputs and outputs will require no pre-processing.

3. Simple Corrective Constraints

A corrective constraint returns an appropriate correction to a given orientation. In the case of valid orientations the corrective rotation is zero while for invalid orientations the corrective rotation rotates the invalid orientation to its nearest valid counterpart. The task of mapping a current rotation to the relevant corrective rotation is complex due to the discontinuous or piecewise linear nature of the mapping.

In the previous chapter the limitations of the approaches used to implement joint constraints and those inherent in common angular representations were highlighted. As angular representations quaternions are ideal for many purposes (e.g. interpolation,) though the definition of corrective constraints is problematic due to the increased dimensionality and the difficulty of visualisation. In this approach corrective quaternion constraints described using discontinuous vector fields in quaternion space (four dimensions).

In order to gain an understanding of the performance of neural networks in learning discontinuous vector fields less complex discontinuous vector fields were studied. Studying vector fields, representing constraints in one, two and three dimensions provides an opportunity to test the abilities of the neural network before investigating more complex four-dimensional (quaternion) cases.

Having implemented these simple vector based constraints a unit sphere with a circular rotational constraint on its boundary is introduced. In initial investigations an input vector is corrected to a circular boundary on the sphere surface by a correction vector. This is then extended to the quaternion based rotational correction of similar vectors to a circular boundary.

3.1 Methodology

A Generalised Multi-Layer Perceptron (GMLP) is trained to model discontinuous vector fields in *one*, *two* and *three* dimensions. These vector fields represent simple joint constraints with two distinct regions – a valid region with zero correction per vector and an invalid region with each vector pointing to an implicit joint constraint boundary.

Initially rotations in a single dimension are considered. Here the relationship between inputs and outputs, including its discontinuous nature, are similar to those of motor dead zones and frictional forces for which compensation models have been created in the robotics field [16, 58, 101]. Fig. 10 (a) shows an example one-dimensional mapping the constrained region is the flat region at the centre of the graph this diagram clearly shows the discontinuous nature of the mapping.

This was extended into two dimensions and two-dimensional vectors and their correction to a circular constraint boundary were considered, (as illustrated in Fig. 10 (b).) The constraint region here is at the centre of the circle and the vectors shown represent the mapping from invalid positions to the circular constraint boundary. The colour lightens from the original vector to the corrected vector and points are placed at the start of each vector allowing the visualisation of vectors with zero correction. This was then extended to three dimensions with the vector field being trained to map to the surface of a sphere in \mathbb{R}^3 . A visualisation depicting this mapping is shown in Fig. 10 (c). This result is significant because quaternion based constraints were reduced to three-dimensional mappings in the approaches of both Herda *et al* [4, 5] and Johnson [2].

The results of these experiments showed that ANNs could be trained to learn vector field models. This encouraged further investigation into mappings better suited to joint modelling. A circular boundary was modelled on the surface of a unit sphere, similar to the projected spherical polygons used to model joint constraints in previous work [40]. Initial investigations considered linear correction vectors representing the vector required to translate the input vector to a valid position at the edge of the constrained region, (as shown in Fig. 5(a).) Here the input is unit a vector that describes the current limb and the output a corrective vector that maps the vector to the boundary. These

experiments were successful and extended to consider a mapping between a unit vector and the quaternion rotation from an invalid configuration to a valid configuration, (as shown in Fig. 5 (b)).

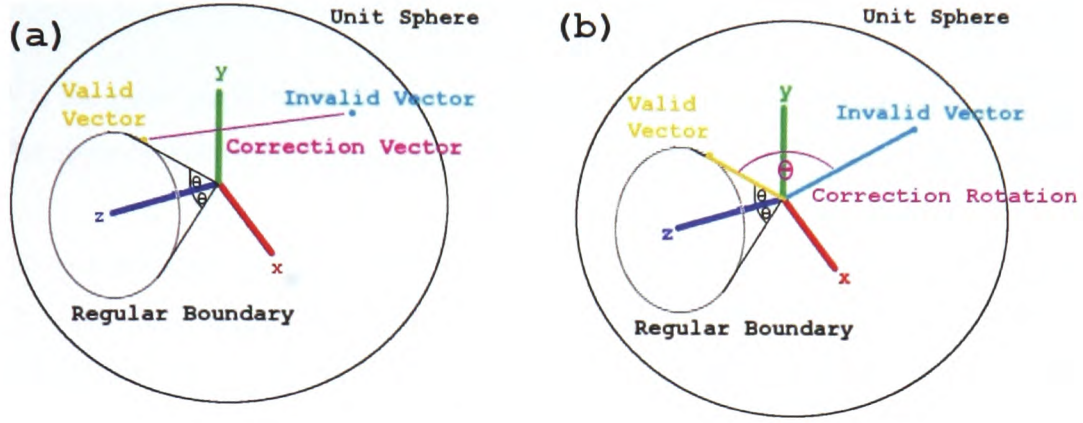


Fig. 5 – An image showing a two dimensional constraint on the surface of a three-dimensional sphere, the initial vector to correction vector case (a) and the vector to correction quaternion case (b).

3.1.1 Dataset Generation

For *one*, *two* and *three*-dimensional constraints a random vector was generated each component limited to between -1 and $+1$. The distance of this vector from a central point measured. If the vector was within a threshold then no correction was assigned (a zero vector.) If not, the relevant correction is calculated and assigned.

In the Euler angle case a simple inequality is used to differentiate between valid and invalid regions (shown in equation 45). Identifying the closer of the two boundaries and calculating the difference produces the required correction for invalid cases (shown in equations 46 and 47).

$$P \text{ is valid if } (P > \text{lower AND } P < \text{upper}) \quad (45)$$

$$C = \text{lower} - C \text{ if } (P < \text{lower}) \quad (46)$$

OR

$$C = P - upper \text{ if } (P > upper) \quad (47)$$

Where,

P is the Euler angle input; $lower$ is the lower boundary of the constrained region, $upper$ the upper boundary of the constrained region and C the corrective component.

To ascertain the validity of points in the two and three-dimensional cases the length of the vector between the origin and the point is considered as in relation to a specified radius (as shown in equation 48). To calculate the two and three-dimensional corrections for invalid cases where the correction is not zero the length of the vector from the origin to the point is calculated and compared it to the radius of the circle or sphere. The ideal is calculated by scaling the vector to the radius, the difference between the original vector and the ideal gives the correction as outlined by equation 49.

$$V \text{ is invalid if } |V| > R \quad (48)$$

$$C = -V(1 - R/|V|) \quad (49)$$

Where,

R is the radius of the constrained region; V is the input vector, $|V|$ the length of the input vector and C the required correction.

In the case of regular two-dimensional boundaries on the surface of a unit sphere, the angle between the input vector and the x-axis was used to delineate between valid and invalid inputs. The method used to calculate the correction for invalid inputs was the same in both cases the rotational correction is calculated first and used to generate the vector correction the latter is calculated as follows. A random unit vector is generated and the angle of the vector relevant to the centre of the circular constraint (the x-axis,) is calculated (see equation 50,) and compared to the constraint radius. If smaller then the

resultant correction is set to the zero, if not correction rotation is calculated as a quaternion as outlined in equations 50 and 52 to 56. This is used to create a vector corrected to the boundary the correction is the difference between these two rotated vectors. In the case of the quaternion based correction the output of equation 55 is used as the correction.

$$\theta = \text{ACOS}(V \cdot C) \quad (50)$$

$$V \text{ is valid if } (\theta > \theta_{max}) \quad (51)$$

$$\Delta\theta = \theta - \theta_{max} \quad (52)$$

$$Aa = \Delta\theta \quad (53)$$

$$Av = V \times C \quad (54)$$

$$Qo = [\text{COS}(Aa/2), Av \cdot \text{SIN}(Aa/2)] \quad (55)$$

$$O = V - (V \text{ rotated by } Qo) \quad (56)$$

Where,

V is the randomly generated vector. C is a unit vector aligned with the centre of the spherical boundary on the surface of the sphere. A is an axis angle representing the rotation of the randomly generated vector its axis part is described as Av and the angle part by Aa . Qi is the quaternion equivalent of this axis angle rotation and θ (theta) is the angle between vectors V and C . $\Delta\theta$ is the difference between the current angle and θ_{max} , the radius of the constrained region. Qo is the angular correction as a quaternion. x is a vector aligned with the x-axis and O is the vector correction.

Three datasets were prepared for each of the experiments; a training set, used to train each generation of ANN, a validation set, used to assess the fitness of the ANN for genetic selection and a test set which provided an unseen set of data on which to test the ANN. In creating the datasets patterns were clustered in the region before and after the boundary representing the discontinuity between the valid and invalid joint configurations.

3.1.2 NetJEN

Initial experiments were carried out using Multi-Layer Perceptron with a single hidden layer, implemented in C++ by the author. Preliminary experiments (Fig. 8) confirmed the superiority of evolved neural networks and hence further experiments were carried out using NetJEN.

NetJEN is a Java based implementation of NetGEN [63, 64, 81] developed by researchers at the University of Salzburg. NetJEN [125] boasts several impressive features and provides an intuitive user interface in addition to reporting tools and other useful functionality. A brief outline of the system they developed follows based on published work [53, 63-65, 81, 83].

Before GA techniques can be applied to ANN topology evolution their underlying structures, the phenotype, must be considered as a genotype (a blue print for the construction of the network.) This must be encoded such that GA techniques can be applied. There are two common approaches; *Indirect Encoding* encodes a set of constraints that govern the construction of individual neural networks within the population. The constraints are evolved indirectly impacting on the neural networks generated. In NetJEN *Direct Encoding* is used, a network topology is created and encoded minimising the decoding effort to map between the genotype and corresponding phenotype. The encoding scheme used is called the *Modified Miller Matrix*, an extension of the *Miller Matrix* [130].

The genome structure is shown in Fig. 6 (a) and comprises *Learn Parameters* (Fig. 6 (b)) which describe the values required to train the neural network, the *Activation Function Template Parameters* (Fig. 6 (c)), used to describe one or more activation

functions present in the network, the *Neuron Parameters* (Fig. 6 (d)) indicate the type of neuron and the *Structure Parameters* explicitly specify each connection within the network [63]. Markers (binary inhibitors) are used to regulate the expression of wild-type genes, for example hidden neurons, while other problem dependent genes such as output neurons are fixed [63]. As a result the bit string includes some non-coding regions (*Introns*), these have been shown to reduce the effects of crossover and are common in biological systems [126-128].

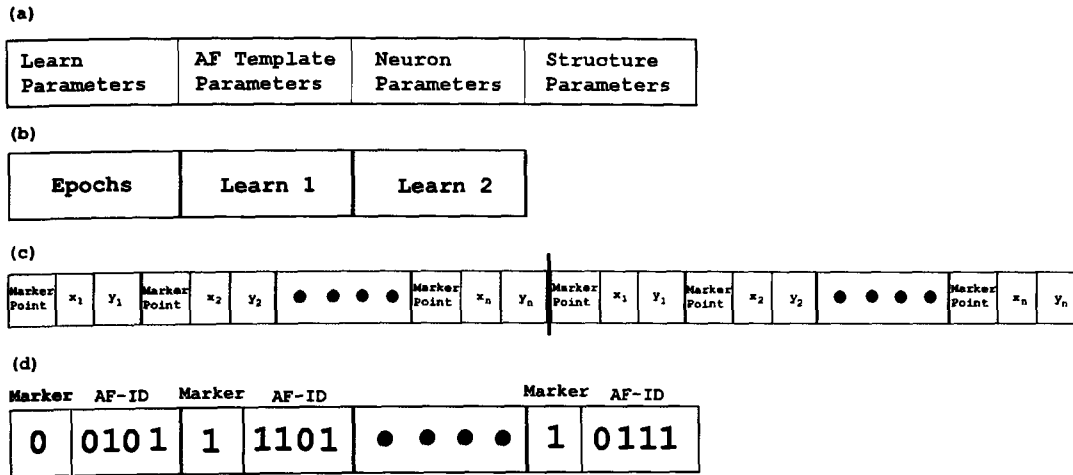


Fig. 6 - The organisation of the genotype: (a) The general structure of the genotype [63]. (b) The structure of the learn parameters segment [63]. (c) The structure of the AF Template Parameters [63]. (d) The structure of the Neuron Parameters [83].

The structure and neuron parameters are represented by a linearized binary adjacency matrix [63] shown in Fig. 7. As the network architectures are feed-forward the triangle above the main diagonal must be zero, the main diagonal is used to represent the activation function index (zero if not expressed) [63]. The maximum size of the network is set in advance and so the size of the structures does not change during evolution. The activation function template parameters and activation functions were not evolved during the following experiments but are included in descriptions of the genome for completeness. In Chapter 5 experiments are undertaken evolving both activation function type and template based spline activation functions.

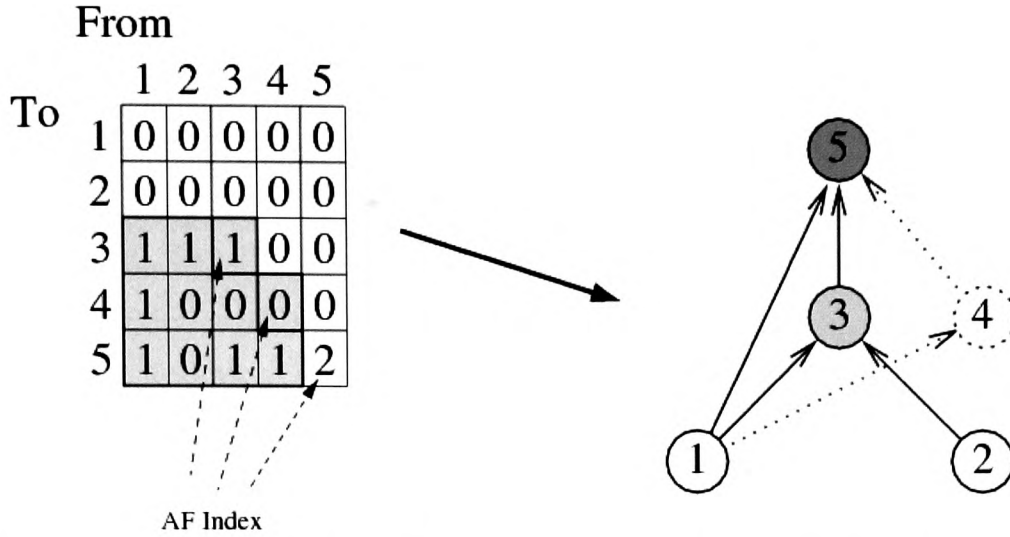


Fig. 7 - The Genotype/Phenotype mapping, here the presence of a one in indicates a forward connection from the node identified by the row number to the node identified by the column number. As there are no links from a node to itself the main diagonal represents the Activation Function (AF) Index is the index of the activation function of a given neuron. The above figure shows all the notes of the system node 4 however despite having an activation function is not part of the generated phenotype as it has no input connections. The image is similar to that given by Mayer and Schwaiger [63].

The system comprises of a Simple Genetic Algorithm (or SGA), the Genotype Phenotype Mapping and the Neural Network Manager. The Neural Network Manager (NNM) in NetGEN was the *Stuttgart Neural Network Simulator (SNNS)* [131], and the SGA from Smith *et al* [132] an implementation of previous work by Goldberg [133]. In the Java implementation the NNM used is BOONE, also developed by researchers at the University of Salzburg.

The SGA generates blueprints for a random population of ANNs which are validated and passed to the NNM where they are constructed and trained using *Resilient Back-propagation* [60]. The SGA then assigns fitness values to each network using a fitness function. This *Composite Fitness Function* comprises a measurement of the networks performance (the *Model Fitness*) and a complexity regularization term (the *Complexity Fitness*,) as expressed in equation 57.

$$F = \alpha_1 \frac{1}{1 + \epsilon m} + \alpha_2 \frac{1}{1 + \epsilon c} \quad (57)$$

Where,

$$\alpha_1 + \alpha_2 = 1.0 \quad (58)$$

In equation 57, F is the fitness of the neural network ϵ_m is the *Model Error* (Sum Squared Error or SSE) and ϵ_c the complexity regularization term. $\epsilon_c = |C_{\text{total}}|$ with C_{total} being the total set of neural network connections. The regularization weight (α_2) has been shown to be most effective in the range 0.001 to 0.01 to guide the evolution towards networks of low complexity. The error weight (α_1) is derived from regularization weight (α_2).

The SGA uses *Binary Tournament Selection* to select the best networks of the population to breed, n individuals (typically two) are selected and the individual with the highest fitness is placed in the breeding pool. The selection itself is weighted, the higher an individuals fitness the more likely it is to be chosen [79]. Binary Tournament Selection has been found to be superior to Proportional Selection methods [126].

An entirely new generation of individuals is created through crossover and mutation of the fittest individuals selected from the last generation. This completes the evolutionary cycle that runs for a specified number of generations. It should be noted that the fittest individuals of the last generation will appear in the breeding pool more than once and breed with themselves generating identical offspring in the new generation [79]. This ensures that the best genetic patterns are passed on to the next generation. Crossover and mutation are implemented on a linearized *Modified Miller Matrix* allowing standard two-point crossover, this has a more global effect on the bit string than the exchange of rows and columns used in the original *Miller Matrix* approach [126-128].

3.1.3 Evolution and Training

In each experiment the network was configured as follows. The input layer represents the current joint vector, while the output layer represents the correction vector/rotation. A population of neural networks is created these have the maximum number of hidden nodes, the appropriate region of the *Modified Miller Matrix* (below the main diagonal as shown in Fig. 7) is randomly populated creating the connections between the nodes.

This leads to a number of hidden nodes not being connected these nodes are evolved but are not present in the phenotype. Binary markers present on both the links and nodes indicate their contribution to the phenotype, if these bits change during cross over or mutation a node may be deactivated. In which case neither the node any associated links are represented in the phenotype. The validation process marks any networks with no connections between input and output nodes with a low fitness [81].

The validated neural networks were then trained by resilient back-propagation identified as being superior to back-propagation by experimentation (as shown in Fig. 8). Where necessary, the inputs and outputs were mapped to the range -1 to +1, the evolution and training parameters were configured as shown in TABLE I. The number of generations and training epochs were restricted to reduce training times. Each experiment was repeated five times to creating five neural networks with independent results to ensure consistency.

TABLE 1
EVOLUTION AND TRAINING SETTINGS

Parameter	Description	Value
Regularization function	Secondary fitness function.	Number of links
Hidden Nodes	Maximum no. of hidden nodes.	20
Number of Generations	No. of generations over which the ANN were evolved.	50
Population Size	Size of the populations evolved.	20
Fitness Function	Primary fitness function.	Inverse SSE
Regularization Weight	Regularization weight (α_2) this term controls the effect network size on the fitness function.	0.01
Evolve number of Links	Networks are pruned down from fully connected networks.	On
Evolve number of Hidden Nodes	Evolves the no. of hidden nodes.	On
Evolve number of epochs	Evolves the no. of training epochs	On
Learning Rate	Learning rate used when training the ANN.	0.1
Stopping Error	MSE at which the ANN are stopped.	0.001
Training Function	Training function used to train the weights of the ANN.	Resilient back-propagation
Max Epochs	Maximum number of training epochs	500

Through experimentation, it was determined neural networks with sigmoid activation functions in the hidden layer and linear activation functions in the output layer produced good results, (activation functions are examined in more detail in Chapter 5.) This distribution of activation functions was used throughout these experiments a similar distribution were employed for vector field approximation by Grzeszczuk *et al* [56],

linear output layers have also been used with bi-polar sigmoid hidden layers [54, 55]. Each experiment was repeated five times to ensure the consistency of the results.

The regularization weight was chosen based on publications by the authors [63], as was the learning rate [126], the stopping MSE for the networks was identified through experimentation. The size of the population, number of generations and initial limits for the number of training patterns were suggested by a co-author of the NetJEN system Dr. Helmut Mayer in private correspondence.

3.2 Results

Initial results confirm the superiority of the evolved GMLP neural network over the feed forward neural network and resilient back-propagation over back-propagation training. These results are included for completeness and shown in Fig. 8, experiments were also carried out with sigmoidal activation functions in both the hidden and output layers, though performance for this architecture is low. All subsequent experiments in this section were carried out using the evolved GMLP neural network.

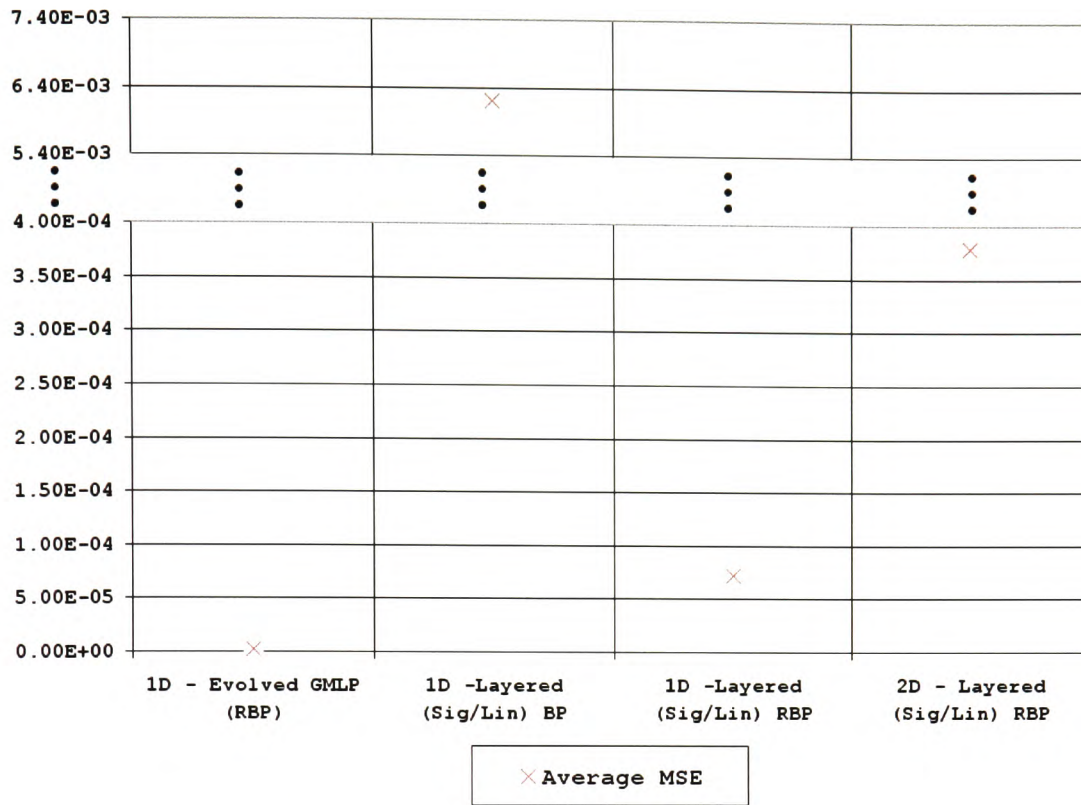


Fig. 8 - Graph comparing the performance of the Evolved GMLP and Layered neural networks with back-propagation (BP) and resilient back-propagation (RBP) training on the one and two dimensional datasets.

The results of the *one*, *two* and *three* dimensional vector field experiments show that though each of the networks trained successfully and the Mean Squared Error (MSE) of the network is low in each case. The performance of the network decreased as the number of dimensions increased, demonstrated by the increase in the MSE as shown in Fig. 9. In each case the vector field that describes the constraint has both continuous and discontinuous regions. These are of comparable size hence the decrease in accuracy is proportional to both the number of degrees of freedom being modelled and the dimensionality of the boundary between the continuous and discontinuous regions.

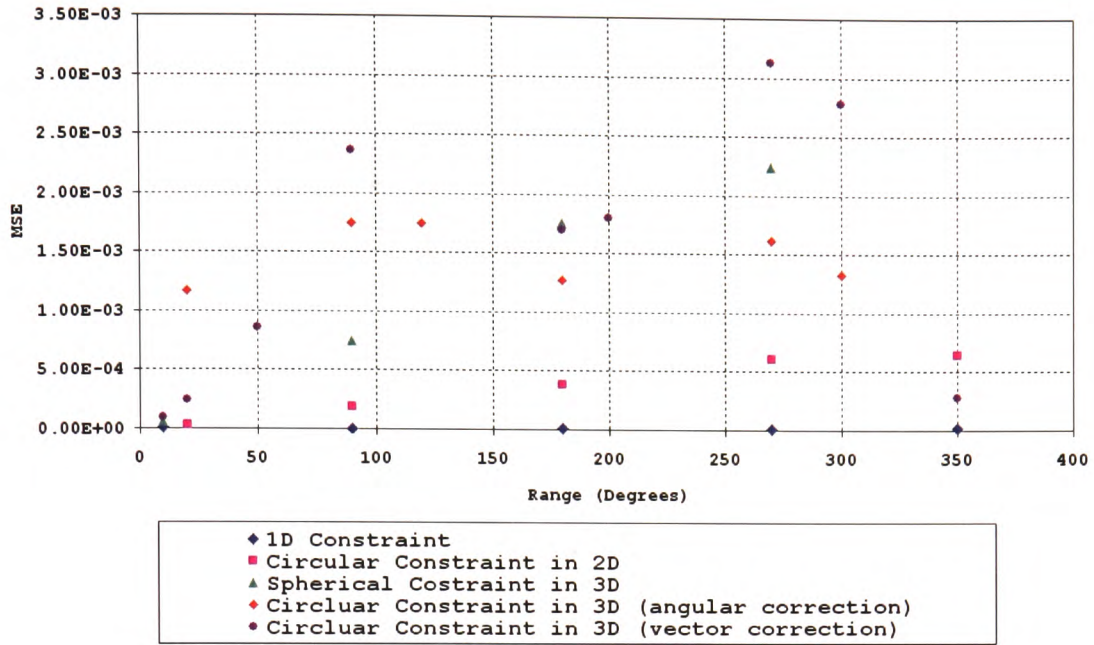


Fig. 9 -Average MSE on a test set (unseen patterns) for constraints of increasing size. In the two and three-dimensional cases the range refers to the diameter of the constraint.

The number of hidden nodes and the number of inter-connections, which are to a certain extent linked, increased as the number of dimensions increased, though only between the one and two-dimensional constraints. This suggests that the number of inter-connections and nodes required to approximate a constraint in two dimensions was sufficient also to approximate a constraint in three dimensions. However, later experiments investigating the improvement of these results (detailed in Chapter 5,) provide evidence that the restrictions placed on the size of the neural network combined with the regularization function limits the performance of the neural network. Similar performance was observed for both test and training sets, indicating the network performed well on unseen patterns.

With regards to the size of the discontinuous (constraint) region, the techniques performed well. It was noted that in the two and three-dimensional case the network performance decreased as the size of the constrained region increased, (Fig. 9.) For each of the ranges tested the size of the evolved networks varied little. Visualisation of these results along with a plot of the test data (or ideal data) demonstrates the accuracy of the approximation, as shown in Fig. 10.

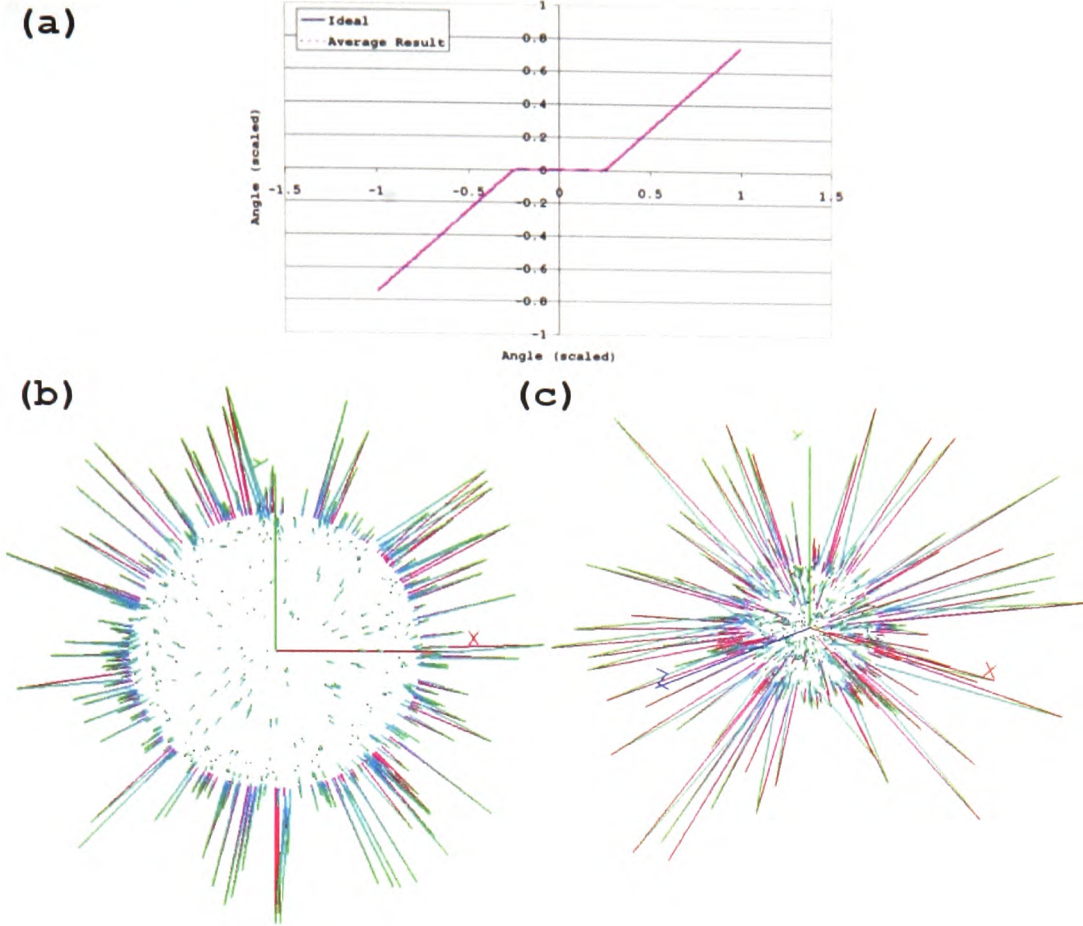


Fig. 10 – (a) A visualisation of the 1D approximation showing the ideal (test data) in solid blue and the neural network approximation in dashed pink. (b) A visualisation of the two dimensional approximation, the ideal is shown in solid pink while the neural network output is shown in solid green. (c) A visualisation of the three dimensional approximation the ideal is again shown in pink and the neural network output in green.

Our attention now turns to more practical two-dimensional boundary constraints on a three-dimensional surface. These were first trained as mappings between vectors and vector based corrections. The results show a significantly different distribution of error despite similar numbers of hidden neurons, links and training epochs being evolved (Fig. 9). High error is observed at 90 and 270 degrees. At this point the two-dimensional boundaries on the surface of the sphere are of equal circumference. Constraints with both larger and smaller circumferences demonstrate lower error indicating that the error does not have a linear relationship with the circumference of the constraint.

Experiments attempting to map between a vector and a corrective orientation using an axis angle representation (calculated in equations 53 and 54) failed to evolve networks with comparable performance to the two and three-dimensional cases. Attempts were made to amend this by scaling the angle part of the axis angle representation to a $-1.0/+1.0$ range without success.

A similar experiment using a corrective quaternion rather than an axis angle produced MSE results that were generally lower than those of the earlier vector correction mapping. Peaks are once more observed at 90 and 270 degrees, suggesting that these features are the results of the vector-based description of the problem rather than the output, which is different in each case.

The vector based correction and quaternion based correction results cannot be compared directly, so the difference between two quaternions was considered indirectly as a positional error in three-dimensional space. This is achieved by comparing the input vector rotated by the ideal correction quaternion (training data) with the same input vector rotated by the correction quaternion produced by the neural network. A plot of the length of the vector between the input vector and the corrected input vector in each case is shown in Fig. 11.

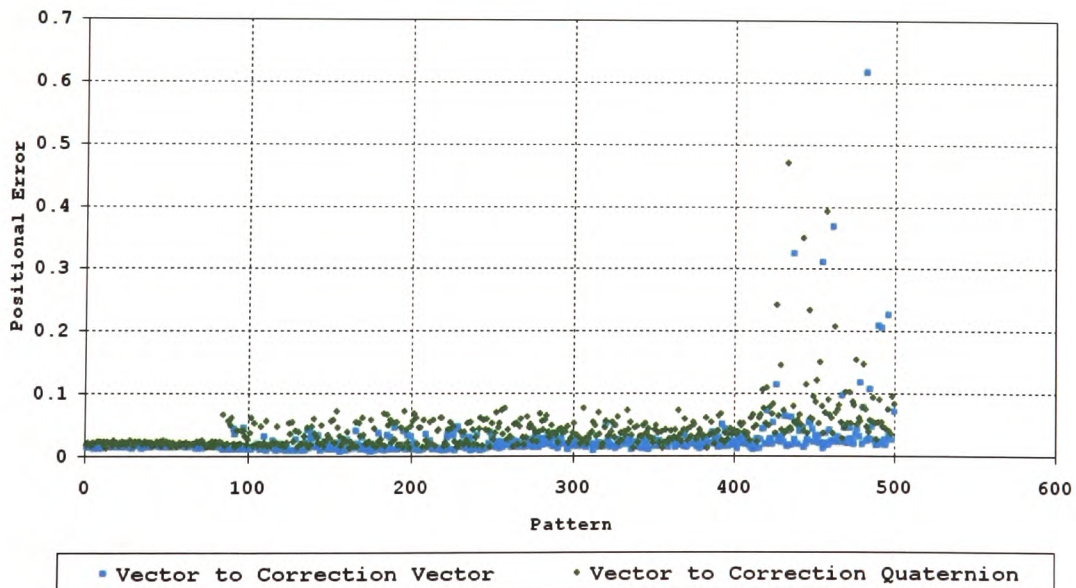


Fig. 11 - Comparison of Positional Errors per Pattern

The results show that the positional error is below 0.1 for more than 97% of the dataset in both cases, (the maximum possible error is 2.0). The vectors of the training, validation and test data are generated at random in sequential groups starting inside the constraint and working outwards. This arrangement of patterns provided good results (as detailed in Chapter 5,) and allows some context to be attributed to the results shown in Fig. 11. The central region in the graph from around 90 to over 400 contains two groups with their division at around 250 patterns, this represents the two groups either side of the boundary. The group on the far left of the graph (Fig. 11,) represents the valid region while the one on the far right represents the region diametrically opposite to the valid region. The region furthest from the boundary (far right of Fig. 11,) contains those points with the highest error, though less than 3% of the points have errors greater than 0.1 in either case.

Comparing the three-dimensional error shows that the increase in error with respect to the radii of the constraint can be attributed to the contribution of patterns in the region furthest from the boundary as shown in Fig. 12.

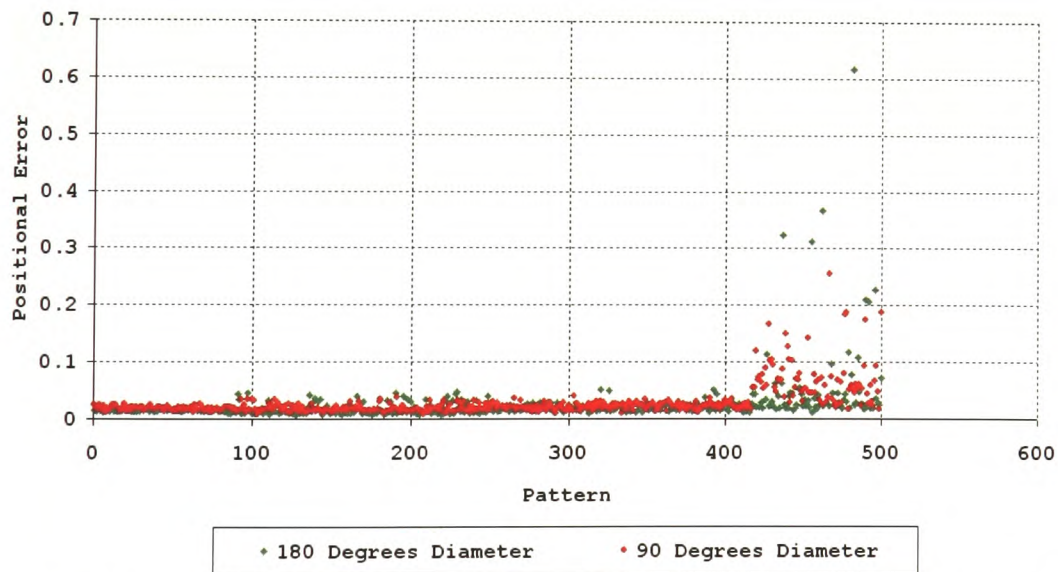


Fig. 12 - Graph showing the increase in error with range being concentrated in the invalid region.

Though generating and plotting the points in groups is useful, it is difficult to identify relationships between patterns with high error. Visualizing the output helps put the results in context and demonstrates the accuracy of the neural network in the region of discontinuity, (at the boundary.) In this case the neural network input represents a unit

vector that is rotated by the output quaternion to lie within spherical boundary on the surface of a unit sphere.

Plotting the vector difference between the input vector and the quaternion corrected input vector gives a visual representation of each of the corrections. In addition to these lines points are rendered at their start, allowing valid points that are not corrected to be displayed. The input data set can be displayed along side the neural network results, (shown in Fig. 13.) Alone the neural network results are difficult to interpret though the boundary region is evident as demonstrated by Fig. 14.

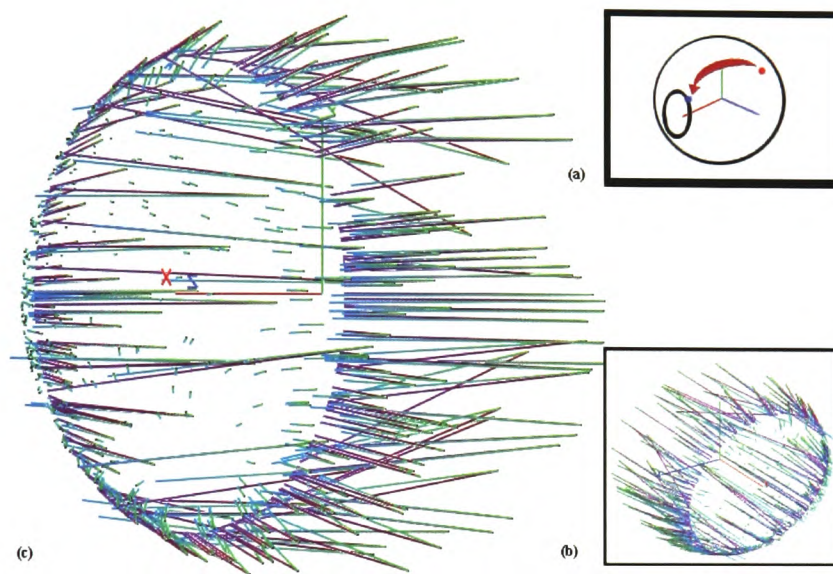


Fig. 13 - A screenshot of the visualisation program showing both ideal and neural network corrections (c). The training data is shown as the dark lines while the light lines show the output of a single neural network. Both lines are graded dark to light to show their direction. To clarify the direction of corrections a simplified sketch of their path (a) has been included and to reinforce the lost 3D element of the subject a second view (b) is included.

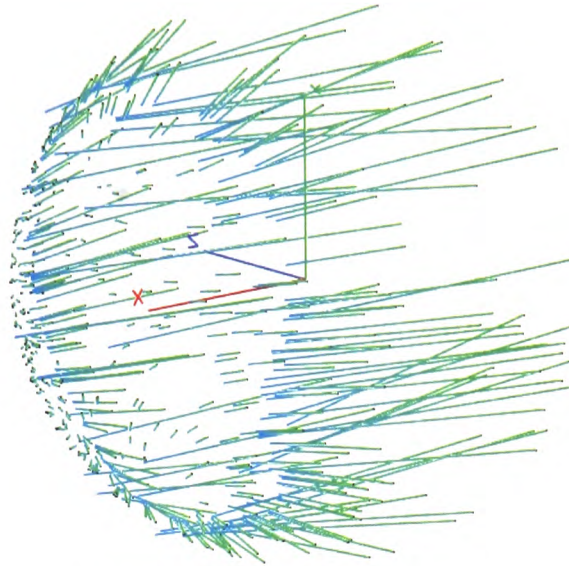


Fig. 14 - A screenshot of the visualisation program showing only the neural network output.

Visualising the results helps to demonstrate the capability of the network in terms of its learning of the discontinuity at the boundary, (shown in Fig. 13, and Fig. 14.) The visualisation also highlights the causes of the high neural network error for certain patterns. This is illustrated in Fig. 15 where a threshold is imposed such that only the results with three-dimensional error greater than this threshold are plotted.



Fig. 15 -A screenshot of the visualisation program showing training, validation and test dataset ideals (shown in red.) The neural network results with Pythagorean error greater than a threshold of 0.1 (shown in green). Five corrections are shown per input orientation representing the five neural networks trained on the input set.

Visualising the results with the threshold shows that for each of the five different datasets the points with the highest error were in a similar location.

3.3 Discussion

In the *one*, *two* and *three* dimensional discontinuous vector field experiments the MSE increased with the dimensionality of the constraint and the problem space. With each increase in dimensionality of the constraint more relationships are included and must be learned, however neither the learning capabilities of the neural network nor the number of patterns representing each relationship are increased. Consequently the performance decreases.

The evolutionary aspects of the experiments indicate that as the constraints increased in complexity more complex networks were required to maintain accuracy. This increase is less pronounced between the two and three dimensional vector fields, due to the constraints placed on the size of the hidden layer to limit the temporal cost of the experiments.

The increase in the MSE in relation to the size of the constrained (discontinuous) region for *one*, *two* and *three*-dimensional boundaries can be attributed to a lack of exposure to the complex patterns in the invalid region where the neural network attempts to learn a continuous non-zero mapping.

Two-dimensional polygons have been used for joint constraint by projecting spherical polygons onto a surface [40], there is evidence to suggest our approach is capable of learning such constraints. In published work quaternion based corrective constraints systems have reduced the dimensionality of the quaternion representation [2-6], in such cases the resulting constraint boundary is a three dimensional surface in three dimensional space. It may be that evolved topology neural networks can be used to implicitly model such boundaries.

In relation to the earlier *one*, *two* and *three*-dimensional constraints, the MSE for neural networks learning vector fields representing circular constraints on the surface of a unit sphere is comparable with two and three-dimensional constraints. The results (in Fig. 9) show an increase in the MSE between the two-dimensional constraint in two-dimensional space and the two-dimensional constraint in three-dimensional space. This indicates that the dimensionality of the problem space has caused an increase in the MSE as dimensionality of the constraint has not increased.

Increases in MSE are observed around 90 and 270 degrees radius, these variations do not seem to relate to the radius of the constraint but occur for constraints with identical circumference. Their occurrence in both the vector-to-vector correction and vector-to-quaternion correction indicate the cause of this error to be the vector field encoding of the problem domain.

In both the vector-to-vector and vector-to-quaternion correction experiments, isolated patterns of high error are encountered. Through visualisation these errors are attributed to a lack of test data in the region of an additional discontinuity. This discontinuity is diametrically opposite the boundary where points are of equal proximity to diametrically opposite sides of the spherical boundary this is termed the *correctional discontinuity*.

Patterns that demonstrate high error are isolated in the region of the correctional discontinuity. Individual patterns are similar to neighbouring patterns and corrected to one side of the sphere, this is in conflict with test data that states it should be corrected to the other side of the sphere. The neural network successfully corrects the vector to the boundary but as this is not the boundary indicated by in the test data, a large error is reported. These errors affect the MSE as their magnitude (but not frequency) increases as the radii of the constraint is increased Fig. 12. These errors are some distance from the boundary and in an anatomically correct joint constraint system it is unlikely that a joint would reach these configurations.

Encoding the output as a quaternion produces an improvement in the MSE of the result over a range of radii, together with a slight increase in three-dimensional error (as shown in Fig. 9.) Researchers have demonstrated that neural networks are best able to model vector field approximations when the magnitudes of the input and output vectors

are similar and of unit variance and zero mean [56]. Quaternion encoding meets the majority of these criteria. It has also been shown that modelling the relationship between a state and a state change provided more comparable magnitudes with regards to input and output value than state-to-state mappings [56].

Regular two-dimensional boundaries on the surface of a sphere provided encouraging results for constrained regions of different sizes. These techniques can effectively implement simple constraints similar to those implemented by Baerlocher [37] and Korein [36] .

3.4 Conclusion

The results show that a Generalised Multi-Layer Perceptron (GMLP) with evolved structure can model multidimensional discontinuous vector fields suitable for the accurate joint constraint simulation in *one*, *two* and *three* dimensions. There are also indications that these techniques may be applied to other problems of similar dimensionality which can be represented as vector fields, for example the dead-zone compensation systems of Selmic and Lewis [16].

It was also noted that the dimensionalities of the constraint and problem space have an effect on the complexity of the mapping and therefore the performance of the neural network.

The results confirm that evolved neural networks can learn rotational corrections for erroneous unit vectors with respect to regular two dimensional boundaries on the surface of a three dimensional sphere.

An advantage to using the Evolved GMLP method has been identified. It has been found that examination of the evolved networks and the evolutionary process can identify limiting factors. In this case the number of hidden nodes tended towards the maximum when error was highest indicating that an increase in the limit on the number of hidden nodes evolved would produce a decrease in error. Later experiments show that removing the limit on the number of hidden nodes produces an increase in performance as discussed in Chapter 5.

A quaternion-based parameterisation offers a number of advantages. The use of Quaternion based input and output parameters make this approach independent of the position of the limb or body in three-dimensional space, also in any practical application of this technique an orientation for each joint orientation need only be stored once as a quaternion. The use of a quaternion based angular representation for the correction demonstrates a low error in the majority of cases. This encourages the consideration of mappings from a quaternion representing the current orientation to a relative corrective quaternion.

4. Corrective Constraints in S^3 Space

In the previous chapter vectors representing initial joint configuration were successfully corrected to a circular boundary on the surface of a unit sphere using both positional (vector) and rotational (quaternion) corrections. In this chapter evolved neural networks are trained to correct quaternions representing initial joint orientation to a regular (circular) or irregular boundaries using a quaternion-based correction. The discontinuous vector fields learned by the neural network represent the rotation of the joint as a precursor to implementing more complex boundaries representing both rotation of and around the joint [2, 4-6].

Quaternions are used as a rotational representation in a number of applications [2-6], and have a number of properties which make them useful [2, 4-6, 8, 33]. However it is difficult to define correctional constraints in quaternion space and existing approaches are flawed in that they rely on reducing the dimensionality of the quaternion [2-5], iterative corrections [2, 4, 5] or provide no method for generating corrections [6].

4.1 Methodology

Experiments were undertaken to develop corrective quaternion-based constraints (like those of Herda *et al* [4, 5] and Liu and Prakash [3],) which describe rotational constraints for both regular and irregular boundaries in quaternion space. Once again NetJEN was used to evolve and train Generalised Multi-Layer Perceptron (GMLP) neural networks to simulate joint behaviour.

4.1.1 Dataset Generation

In these experiments the current joint orientation is described using a quaternion, as was the corrective rotational output from the ANN. The corrective output quaternion, when combined with the current rotation, rotates an invalid rotation to a valid rotation on the constraint boundary. Valid input rotations are given no correction and the network

outputs the identity quaternion. A boundary between valid and invalid joint constraints is implicitly defined. This boundary marks a discontinuity in the vector field between corrective and non-corrective (identity) quaternions.

Three datasets were prepared for each of the experiments; a training set, used to train each generation of ANNs, a validation set, used to assess the fitness of the ANNs for genetic selection and a test set which provided an unseen set of data on which to test the ANN's performance.

An even distribution of patterns is used with patterns grouped into valid and invalid and trained in the same order. These distributions were found to provide superior results as discussed later in Chapter 5. In the case of regular boundaries an automated dataset generator was used. This generated a random unit vector then calculated its orientation as a quaternion with respect to the constraint centre (aligned to the x-axis). This process is demonstrated by equations 59 to 61.

$$Aa = \text{ACOS}(V \cdot C) \quad (59)$$

$$Av = C \times V \quad (60)$$

$$Qi = [\text{COS}(Aa/2), Av \cdot \text{SIN}(Aa/2)] \quad (61)$$

Where V is the randomly generated unit vector, C is a unit vector aligned with the centre of the spherical boundary on the surface of the sphere (the x-axis). A represents an axis angle describing the rotation of the randomly generated vector, its axis part is described as Av and its angle part as Aa . Qi is the quaternion equivalent of this axis angle rotation.

The correction was generated by examining the effect of the quaternion on a unit vector placed at the centre of the constrained region. The angle between the initial vector and the effected vector was calculated (see equations 62 and 63) and compared to the constraint radius (see equation 64). If the angle was smaller then the generated vector was within the constrained region and so the corrective rotation was set to the identity

quaternion. If larger then a correction rotation was calculated as an axis angle and then converted to a quaternion as outlined in equations 65 to 68.

$$P = X \text{ rotated by } Qi \quad (62)$$

$$\theta = \text{ACOS}(P \cdot X) \quad (63)$$

$$P \text{ is valid if } (\theta < \theta_{max}) \quad (64)$$

$$\Delta\theta = \theta - \theta_{max} \quad (65)$$

$$Aa = \Delta\theta \quad (66)$$

$$Av = P \times X \quad (67)$$

$$Qo = [\text{COS}(Aa/2), Av \cdot \text{SIN}(Aa/2)] \quad (68)$$

Here P is a vector used to represent the effect of the quaternion rotation. $\Delta\theta$ is the difference between the current angle θ and θ_{max} the maximum valid rotation. Qi is in this case the input quaternion and Qo is the corrective output quaternion. X is a vector aligned with the x-axis.

A similar automated dataset generator for irregular boundaries was discounted on the grounds of complexity and predicted development time. Instead a semi-automated system was adopted. An interactive virtual arm was created in a three dimensional environment (using OpenGL) and used to record a boundary between invalid and valid rotations. The valid and invalid rotations were then recorded individually relative to this boundary. Rotations were sampled at a set interval while the virtual arm was interactively manipulated.

The quaternions generated as part of the valid rotation set (inside the constraint boundary,) were assigned a no correction output (the identity quaternion). The invalid rotations were initially corrected to the nearest point on the constraint boundary. These points were assigned the output quaternion representing the rotation from the given invalid point to the nearest valid point calculated on the constraint boundary. The closest boundary point was identified using angular and proximal comparisons in quaternion space, however visualisation of the datasets generated showed the corrections to be slightly skewed to what was expected. To overcome this problem the nearest of the boundary quaternions was calculated based on their effects on unit vectors in three-dimensions.

It is impractical to attempt to test the network with every variation of irregular boundary. In contrast to the regular boundary, the initial irregular boundary (shown in Fig. 16,) contains a single concave region. Concave regions are essential for modelling boundaries such as that of the shoulder complex and so the boundaries considered are of varying size and with one or more concave regions. A 'C' shaped boundary provides a shape with a very pronounced concave region, while a boundary with a highly irregular surface is used to assess the capability of the technique on highly irregular boundaries.

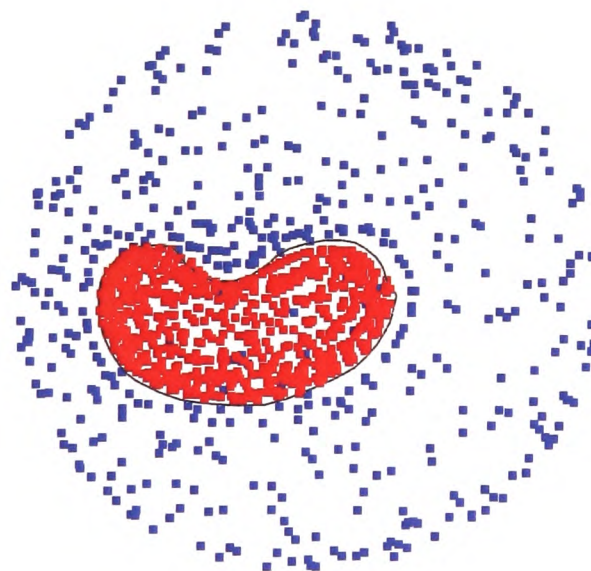


Fig. 16- The image shows an indirect visualisation of the valid (red) and invalid (blue) quaternion in the irregular boundary dataset.

4.1.2 Evolution and Training

In each experiment the NetJEN system (described in section 3.1.2) was configured as follows. The input layer represents the current joint rotation, while the output layer represents the correction rotation. The number of hidden nodes and connection topology are initially randomised and then evolved during the learning process using Genetic Algorithms. The weights of the interconnections are also initially randomised then updated using the resilient back-propagation algorithm. The evolution and training parameters were set as shown in TABLE II. The number of generations, training epochs and hidden nodes were limited to reduce training times. Each experiment was repeated five times to ensure the consistency of the results.

TABLE II
EVOLUTION AND TRAINING SETTINGS

Parameter	Description	Value
Regularization function	Secondary fitness function.	Number of links
Hidden Nodes	Maximum no. of hidden nodes.	20
Number of Generations	No. of generations over which the ANN were evolved.	50
Population Size	Size of the populations evolved.	20
Fitness Function	Primary fitness function.	Inverse SSE
Evolve number of Links	Networks are pruned down from fully connected networks.	On
Evolve number of Hidden Nodes	Evolves the no. of hidden nodes.	On
Evolve number of training epochs	Evolves the no. of training epochs	On
Learning Rate	Learning rate used when training the ANN.	0.1
Stopping Error	MSE at which the ANN are stopped.	0.001
Training Function	Training function used to train the weights of the ANN.	Resilient back-propagation
Max Epochs	Maximum number of training epochs	500

Through experimentation, it was found that good results were obtained for neural networks with a sigmoid hidden layer and linear output layer, (as discussed greater detail in Chapter 5). This distribution of activation functions was used throughout these

experiments. Each experiment was repeated five times giving five neural networks with individual results ensuring consistency

The regularization weight was chosen based on publications by the authors [63], as was the learning rate [126], the stopping MSE for the networks was identified through experimentation. The size of the population, number of generations and initial limits for the number of training patterns were suggested by a co-author of the NetJEN system Dr. Helmut Mayer in private correspondence.

4.2 Results

4.2.1 Regular Boundaries

Neural networks were successfully evolved and trained to model discontinuous vector fields representing regular (spherical) constraints. This is reflected both by the low Mean Squared Error (MSE) values and the structure of the neural networks - indicated by the number of hidden nodes as shown Fig. 17. The number of hidden nodes was limited to reduce training times (as indicated in TABLE II,) and the size of the hidden layer for each trained network was close to this maximum throughout. The number of hidden nodes appears to increase with the MSE (as shown in Fig. 17). This indicates that more complex networks were required for these constraint ranges and that the high error is contributed to by the restriction on network size.

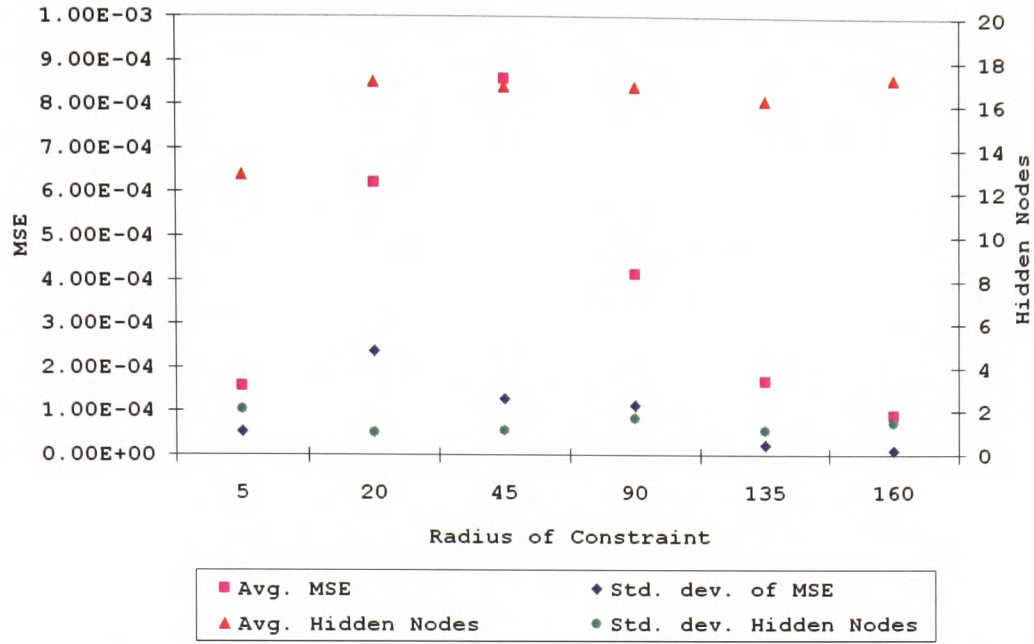


Fig. 17 – Graph showing the effect of range variation on MSE and network complexity (hidden nodes).

The increase in MSE between a 20° and 45° constraint radius cannot be related to any obvious three-dimensional factors. To understand this behaviour, further correlations are sought with respect to the distribution of training patterns in quaternion space. Principle Component Analysis (PCA) gives a set of eigenvectors that describe the orientation of the data in four dimensions while the eigenvalues describe the variance within each dimension. The eigenvectors produced were compared to the identity quaternion and the change in rotation between them noted. The fourth principle component has an eigenvalues of zero and is thus ignored.

Fig. 18 illustrates the correlation between the first principle component and MSE indicating the success of the training process is sensitive to the orientation of the data set in quaternion space.

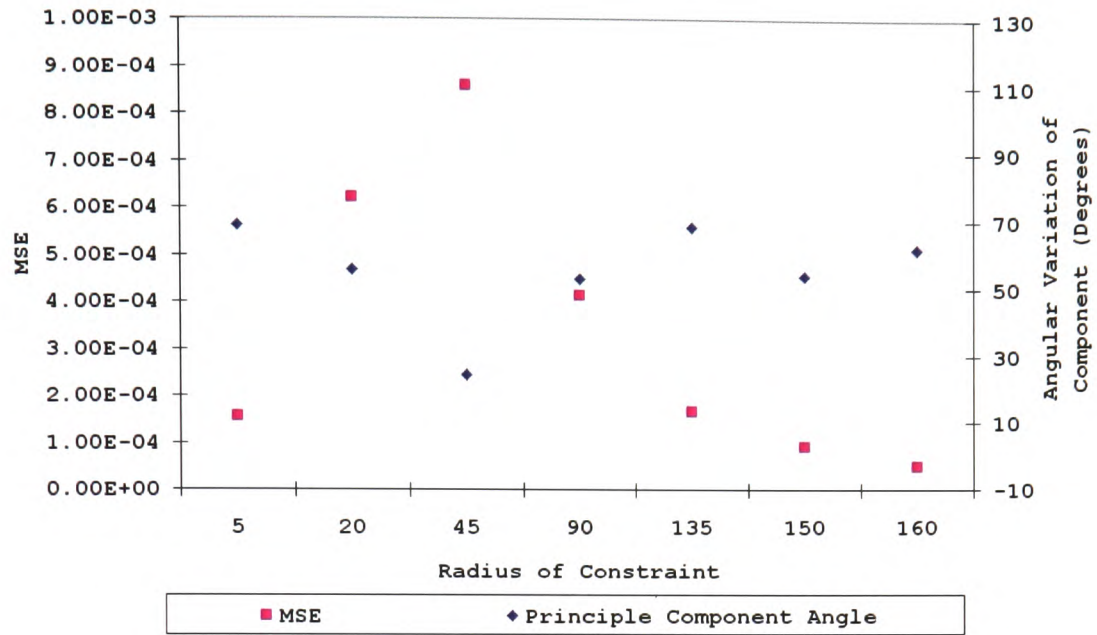


Fig. 18 – Graph showing the effect of range variation on the MSE and the change in orientation of the principle component.

MSE is a holistic measurement that masks regional variations in error over the vector field. To quantify these errors, the behaviour of a virtual joint constrained by the trained neural networks is observed. The l_2 norm (Pythagorean distance) between the ideally corrected and neural network corrected endpoint of a virtual limb is observed. This more direct comparison demonstrates that the error is highest around the boundary separating the valid and invalid regions, as shown in Fig. 19.

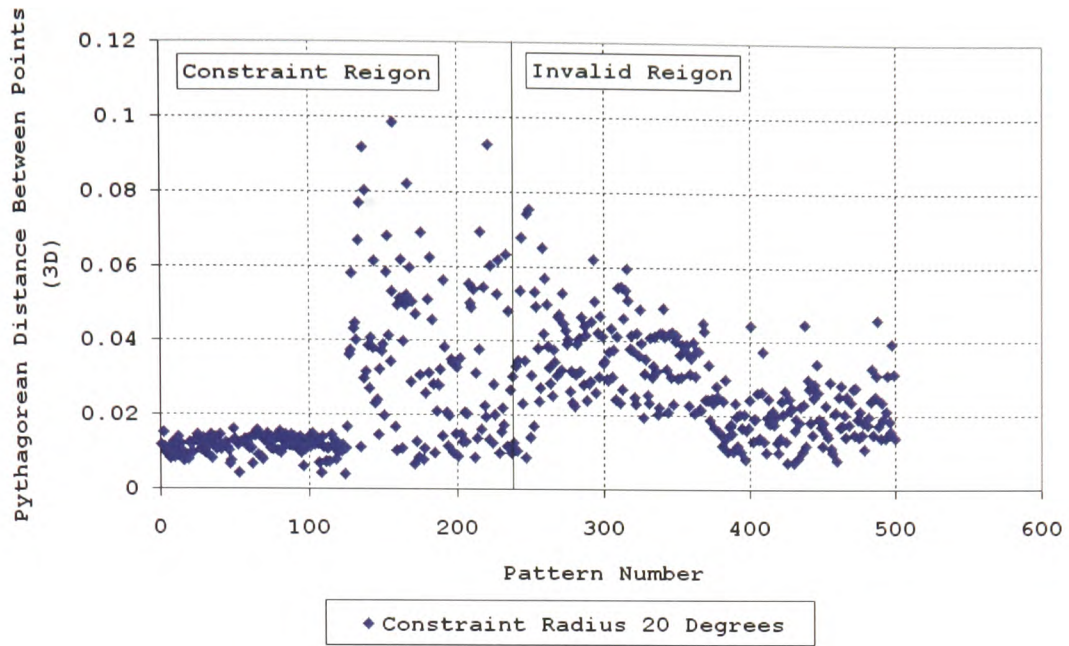


Fig. 19 – Plot showing the four-dimensional Pythagorean error between the ideal quaternion correction and the corresponding neural network output.

Comparing the results using the Pythagorean error metric it is observed that for the ranges investigated the average error for all patterns for each of the five networks is less than 0.2 (as shown in Fig. 20). Given that the maximum error for on the unit sphere is 2 (diametrically opposite) this gives an average error of less than 1% with a maximum error of 6.3%. Considered in terms of the virtual arm (which is unit length) the average error is less than 2% and the maximum error 12.6%.

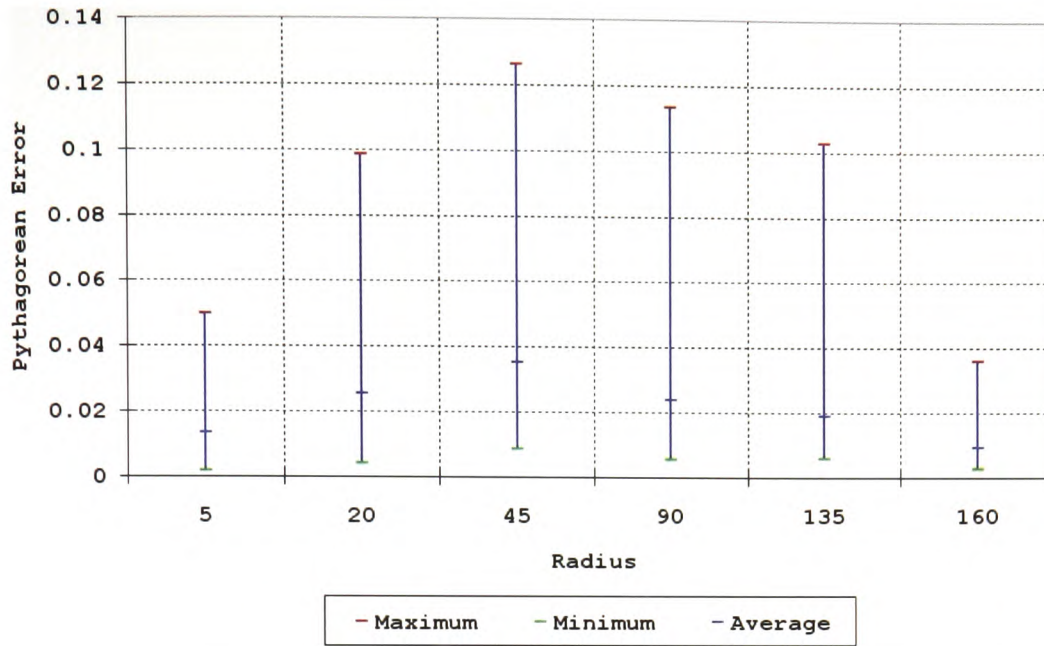


Fig. 20 – A graph showing a comparison of Pythagorean error for constraints of varying radii.

Fig. 21 illustrates the ideal and actual correction vectors for a virtual limb constrained by a regular boundary constraint. The red dashed line shows the result of the corrections from the training data while the green solid line shows the results of the neural network corrections, both lines lighten from their initial positions to their corrected positions. The boundary can be clearly identified and it is noted that all corrected rotations within a Pythagorean distance of 0.1 from the constraint boundary (as indicated by Fig. 19).

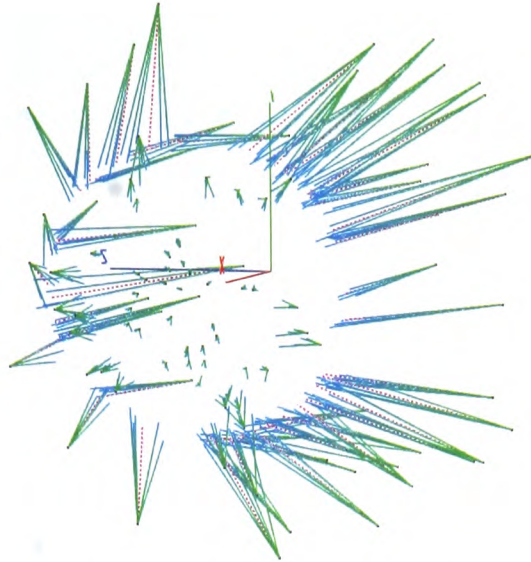


Fig. 21 – Ideal and neural network corrected rotations. Ideal corrections are shown as red dashed lines; neural network corrections (for each pattern) are shown as green solid lines.

In order to highlight the patterns with high error observed in Fig. 21 a threshold is set and only the corrections whose error exceeds this are displayed. This allows the worst results to be viewed in the context of the constraint boundary, as shown in Fig. 22. Fig. 21 and Fig. 22 demonstrate that even the points with the highest error are corrected to the boundary.

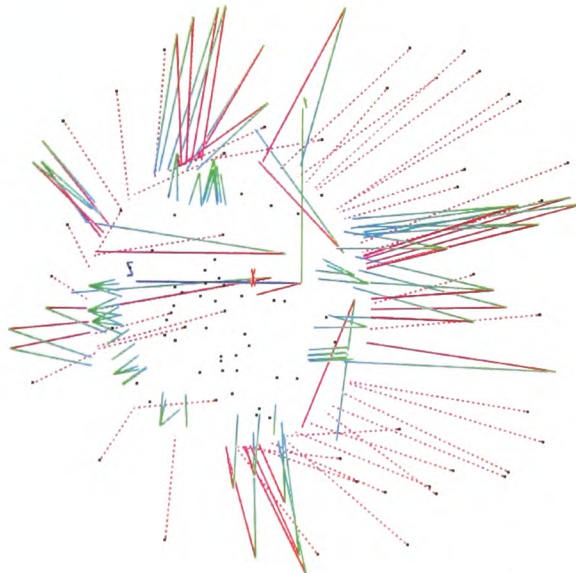


Fig. 22 –A visualisation showing patterns with error above a threshold of 0.08 (three dimensional Pythagorean error). The corrections of all the training patterns are shown as red dashed lines. The neural network outputs above the threshold and their training patterns are shown as green solid lines, with the neural network output shown as the lighter lines.

The network's behaviour can be further understood by looking at the distribution of training patterns. Fig. 23 shows the test inputs for a regular constraint with 20° radius. The input rotations of the test set are coloured according to their error, with blue points having lower error and red points having higher error. Salient regions of higher error are those around the constraint boundary and in the region opposite the constraint on the quaternion unit hyper-sphere.

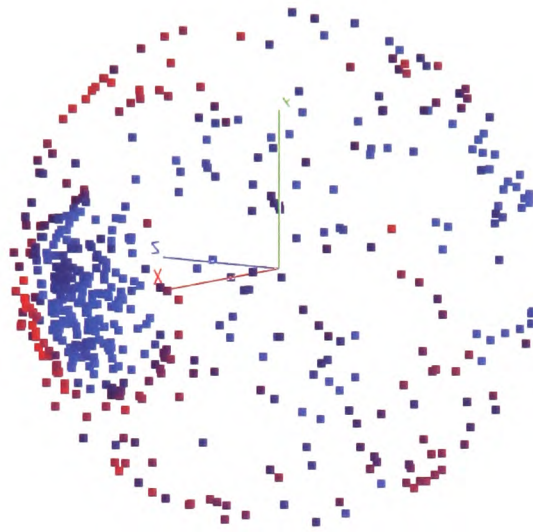


Fig. 23 - Regular test patterns coloured to represent the l_2 norm (i.e. error) of the relative neural network output. The blue patterns represent low error and the red ones high error.

Overlaying the patterns used in training ('x') and evolution ('+') of the neural network it is observed that regions of sparse training data correspond to regions of high error. In Fig. 24 the camera is placed inside the same sphere of points shown in Fig. 23. High error points are observed in an area where there are few training set points. Low error points are observed in regions well populated with training points. The validation set points ('+'), used to assess network fitness, have less effect.

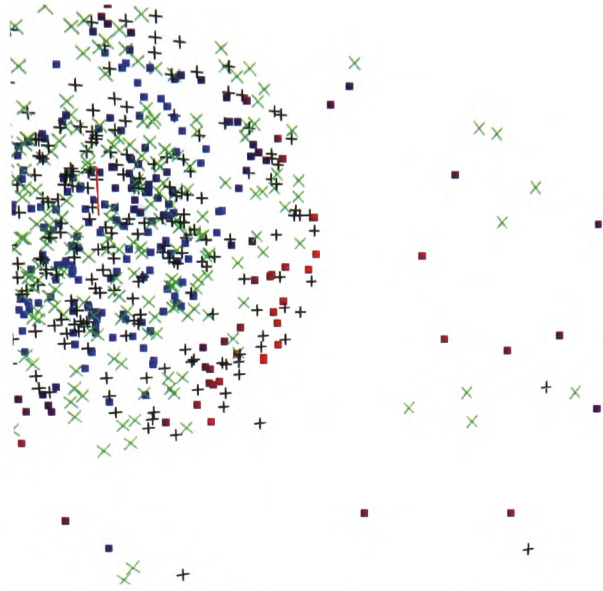


Fig. 24 - Regions displaying poor performance. Blue squares depict low error and red squares high error. Training patterns ('x') and validation patterns ('+') are overlaid.

As discussed earlier the datasets were generated with reference to circular constraint boundary centred on the x-axis. A number of experiments were carried out with regards to the orientation of the constraint centre. The experiments considered alignment with each of the principle axis and varied little in terms of their MSE. However the distribution of error changed significantly, as shown in Fig. 25. The results indicate that a constraint centred on the y-axis offers the best results.

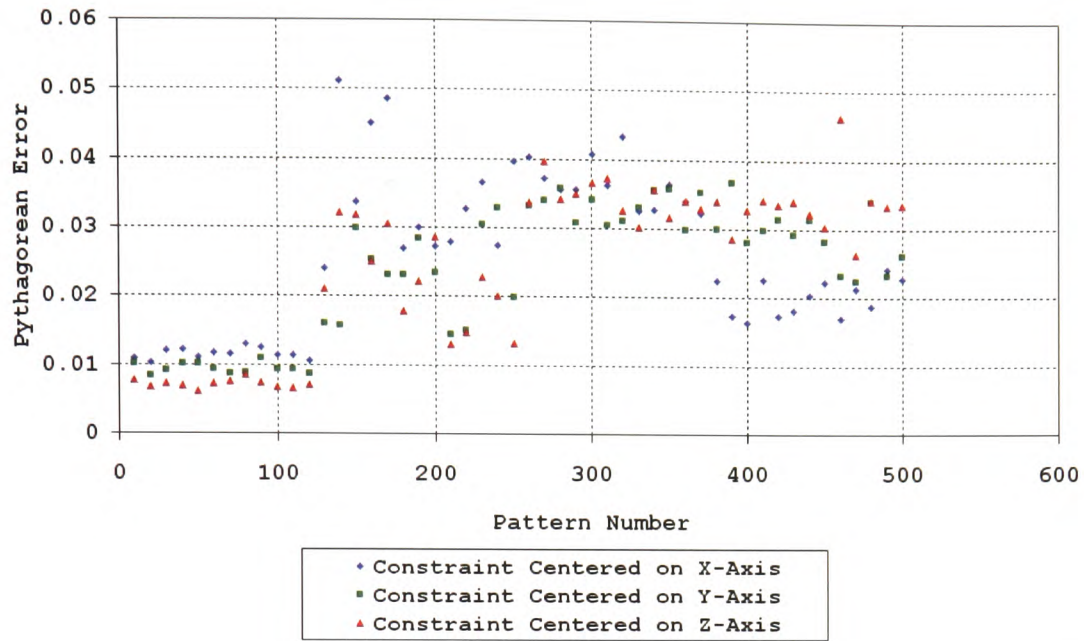


Fig. 25 – Plot showing the average Pythagorean error for limbs with differing constraint centres. The results represent an average error per 10 patterns to aid clarity.

These results can be supplemented using the MSE and PCA of the datasets involved. Here the variation in the effects of the principle components (given by their eigenvalues) is examined, smaller values indicate a less elongated dataset and if all eigenvalues were equal (and variance zero) the dataset would be hyper-spherical. A correlation between the variance of the eigenvalues and the average Pythagorean error (as shown in both TABLE III and TABLE IV) indicates that the performance of the test set is linked to the distribution of patterns in quaternion space.

TABLE III
EFFECT OF EIGENVALUE VARIANCE ON MSE

Limb Start Alignment	Variance of Eigenvector Contributions	Average MSE	Average Pythagorean Error
Y	310.99	5.61E-04	0.023
Z	311.44	4.60E-04	0.024
X	317.23	4.88E-04	0.025

The results (shown in TABLE III,) show that although an initial limb alignment with the Z-axis produced the lowest MSE, the networks performance in three-dimensional space (indicated by the average Pythagorean error for all patterns,) increases as the variation in eigenvalues increases, i.e. as the dataset becomes less elongated in quaternion space.

In the regular boundary experiments detailed above the dataset generation method generates a random unit vector in three-dimensions, then calculates a quaternion representing its current orientation and finally the correction required to return it to the boundary. Unit quaternions used to represent rotation occupy a S^3 hyper-sphere in four-dimensional space. The S^3 hyper-sphere represents 4π rotations, therefore quaternions on opposite sides of the S^3 hyper-sphere represent the same rotation [43].

This ambiguity occurs with respect to the rotations represented by quaternions on the unit quaternion hyper-sphere and not quaternion space itself. However since their representation is used to generate the dataset this potentially poses a problem. The situation arises where there are two boundaries - one on each side of the unit quaternion hyper-sphere. The neural network has to learn to correct to the appropriate boundary and this gives rise to second discontinuity in the dataset at which there will be a division between quaternions who are corrected to a boundary on either half of the hyper-sphere.

The quaternion creation method used in the above experiments avoided these problems as most of its invalid rotations, all valid rotations and more significantly the boundary to which it generated its corrections were on one side of the unit quaternion hyper-sphere. It was postulated that forcing all the points to be generated on one side of the unit quaternion hyper-sphere would simplify the vector field and improve training. However, results show this not to be the case, (Fig. 26). The original dataset with the *majority* of points on one side is described as ambiguous (or AMB) and the dataset with *all* points forced to one side as non-ambiguous (or Non-AMB).

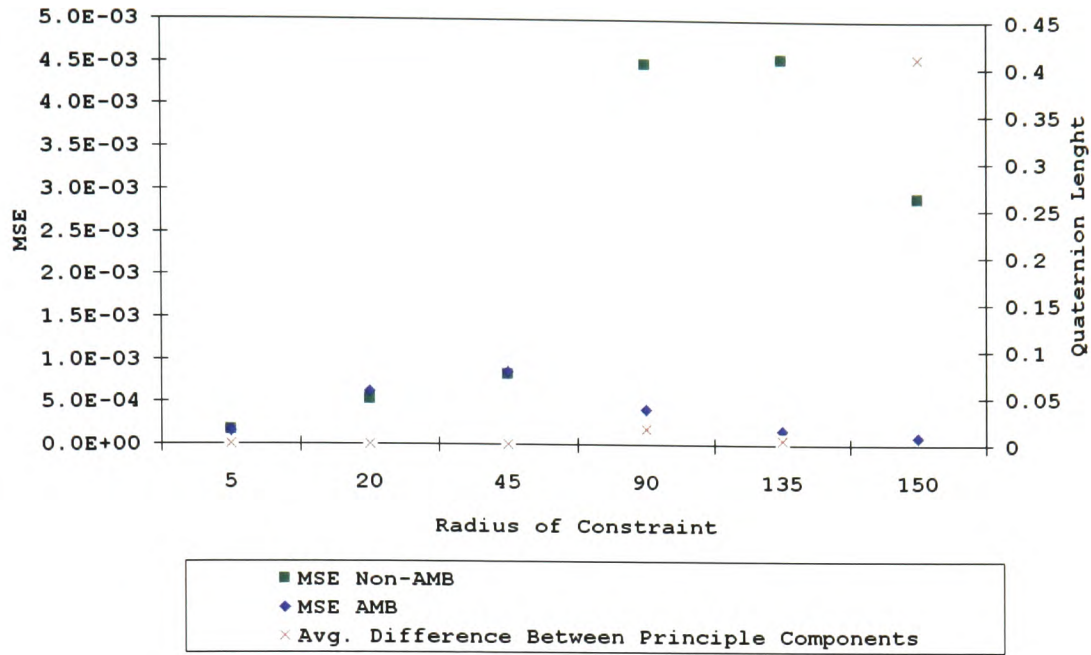


Fig. 26 -A comparison of the MSE recorded over various ranges for ambiguous (AMB) and non-ambiguous (non-AMB) results, plotted against the length of the difference quaternion between the principle components of each.

When the quaternions are forced to one side of the hyper-sphere the error on larger ranges increases. In order to explore this further PCA was performed on pairs of datasets over the same range. In each case one dataset was modified such that all the quaternions were on the same side of the quaternion hyper-sphere.

A comparison of the principle components over the range found that below 90 degrees there were no points moved and the PCA gave similar eigenvectors and eigenvalues. Above 90 degrees the number of quaternions moved increases, as does the difference between the principle components correlating with a rise in MSE.

The difference is measured by subtracting each vector in the principle component matrix (4×4) to give the difference for each as a four-dimensional vector. The average length of these vectors is used as a distance metric. In Fig. 26 a clear correlation can be observed between the increasing difference in the distribution of patterns in quaternion space represented by the length of the difference vector and the increase in error.

To further understand how the patterns in quaternion space are changing the orientation and influence of each of the principle components was investigated. The following graphs shows the MSE of the two test sets as before, the first (Fig. 27) in addition shows

the difference in orientation of the principal components (compared to a single 4D vector). The second (Fig. 28) shows the contributions of the principle components contribution, that is the percentage of the variation can be attributed to the component.

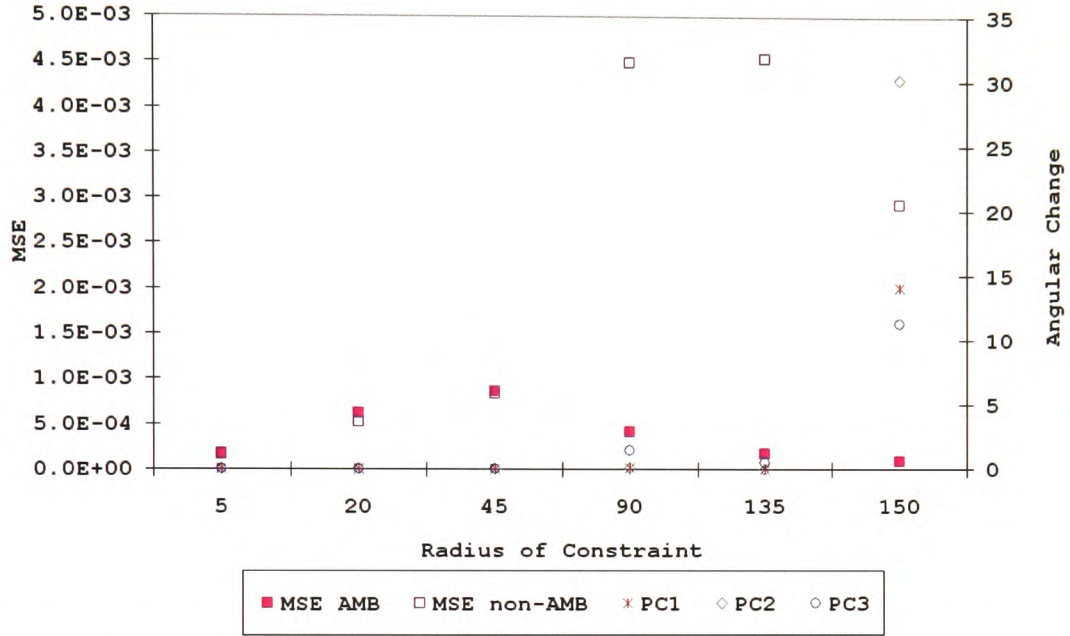


Fig. 27 -The difference in the orientation of principle components plotted against the average MSE for the ambiguous (AMB) dataset (with quaternions on both sides of the hyper-sphere,) and the non-ambiguous (non-AMB) dataset (with all quaternions on one side of the hyper-sphere.)

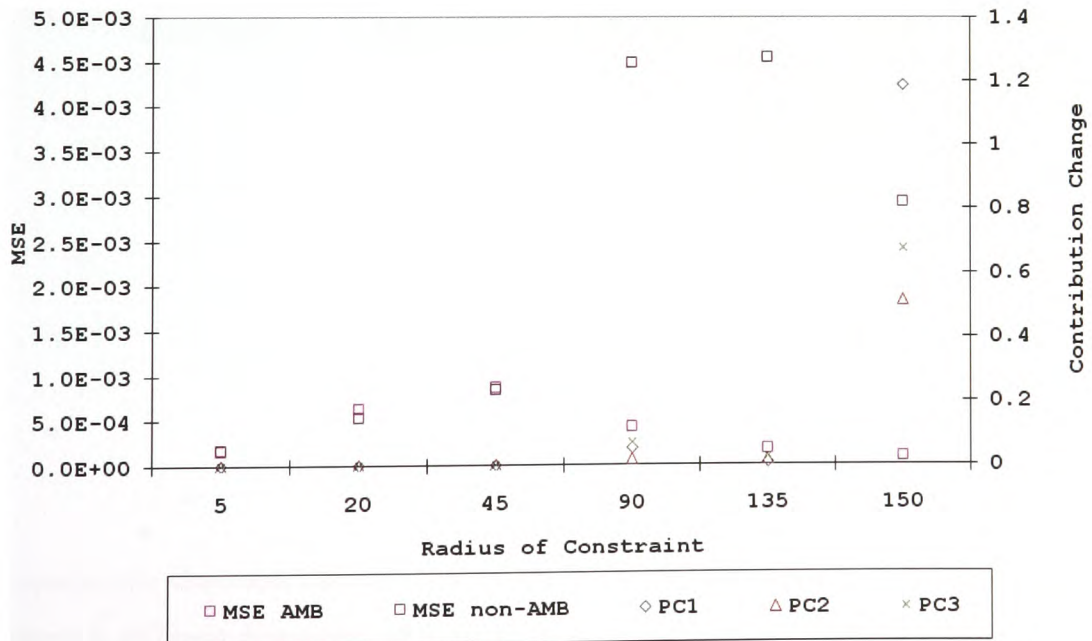


Fig. 28 -The difference in principle components contribution plotted against the average MSE for the ambiguous (AMB) dataset (with quaternions on both sides of the hyper-sphere and the non-ambiguous (non-AMB) dataset (with all quaternions on one side of the hyper-sphere.)

Mapping quaternions to one side of the hyper-sphere has an effect on the orientation of the third principle component and the contributions of all principle components. These increases are proportional to the range indicating a change in the distribution of training patterns in quaternion space.

The shape of the dataset in quaternion space described by the variance of the eigenvectors was discussed earlier. When investigating the differences in the shapes of the datasets it was found that forcing quaternion to one side of the sphere produced datasets with a less regular distribution. A marked increase in error (a positive difference) for 90 and 135 degrees in correlation with a comparative increase in eigenvector variance can be seen in TABLE IV. A large change in the difference in eigenvector contributions for the largest radius (150 degrees) is noted but a relative improvement in error, which can be attributed to the change in dataset orientation as shown in Fig. 18 above.

TABLE IV
EFFECT OF AMBIGUITY REMOVAL ON EIGAN VALUES AND
PERFORMANCE

Radii of Simulated Constraint	Difference in variance of Eigenvector contributions	Difference in MSE
5	0	1.59E-05
20	0	-9.8E-05
45	0	-2.6E-05
90	7.19	0.0040
135	4.82	0.0043
150	722.21	0.0029

Experiments also show that the effect of forcing the quaternion to one side of the hyper-sphere is different depending on the axis at the centre of the constrained region. This is shown in TABLE V.

TABLE V

THE EFFECT OF CONSTRAINT CENTRE ON QUATERNION DISTRIBUTION.

Limb Start Alignment	Difference (4D Pythagorean Distance)
Y	0.087
X	0.120
Z	0.236

4.2.2 Irregular Boundaries

In human anatomy most of the rotational boundaries encountered are irregular. Therefore the performance of this technique on such boundaries is an important consideration. In the experiments shown here irregular boundaries designed to test the capabilities of this constraint modelling approach were used.

Mathematically generating datasets for constrained regions with an irregular boundary is difficult. For these experiments a boundary and rotation recording program written in C++ using OpenGL was used. This approach produced quaternions on both sides of the quaternion hyper-sphere for all regions unlike the earlier automated test set generation.

Experiments using an ambiguous dataset, where invalid quaternions were corrected to the closer of two boundaries (one on either side of the quaternion hyper-sphere,) did not train successfully. Based on the successful regular boundary experiments, all points on the boundary (used for generating corrections) and within the valid region were forced to inhabit the same side of the quaternion hyper-sphere mimicking the distribution of the earlier regular boundary experiments.

The results show the neural network was able to learn the irregular boundary, though the error was higher than in the case of the simpler regular boundaries shown earlier. This is demonstrated by the average results shown in TABLE VI. The resultant networks are on average of higher complexity than those evolved for regular boundaries.

TABLE VI
COMPARATIVE PERFORMANCE ON TEST DATA

Boundary	Min. / Max. MSE	Avg. MSE	Avg. Hidden Nodes
Regular	4.79E-05 / 7.86E-04	3.31E-04	16
Irregular	9.41E-03 / 1.41E-03	1.24E-03	18.4

Comparing the results using the Pythagorean error metric it was observed that for the boundary shapes investigated the average error for all patterns for each of the five networks is less than 0.4 (as shown in Fig. 29). Given that the maximum possible error on the unit sphere is 2 (diametrically opposite) this gives an average error of 3.15% with a maximum error of 21.7%. Considered in terms of the virtual arm (of unit length) this gives an average error of 6.3% and a maximum error 43.4%.

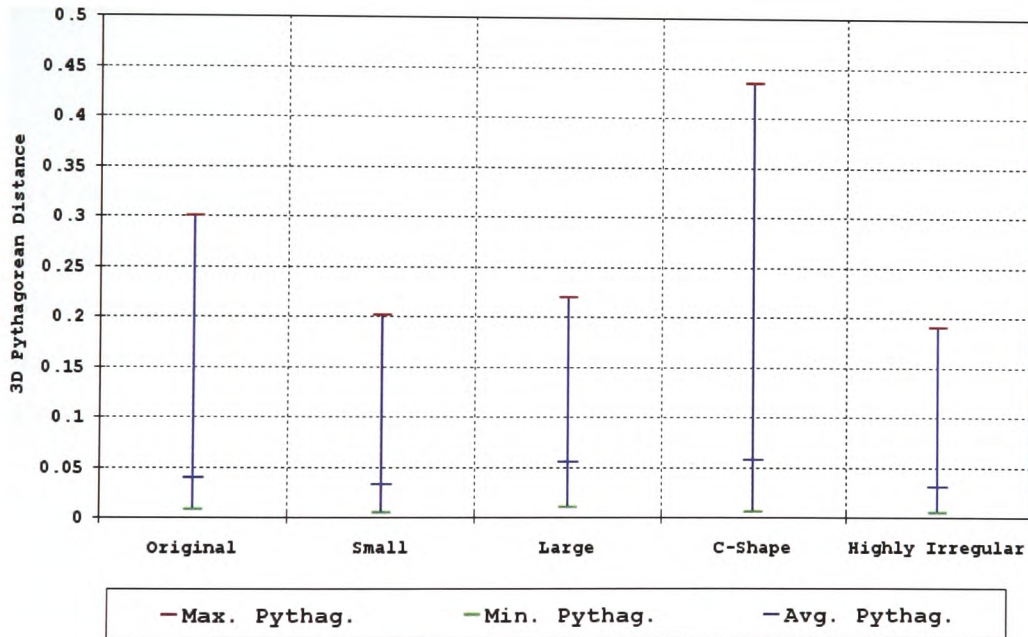


Fig. 29 – A graph showing a comparison of Pythagorean error for constraints of irregular shape.

The evolved neural networks were able to learn the boundary in most cases. Fig. 30 demonstrates the neural networks learning of the discontinuity at the irregular boundary. The chosen boundary is a continuous irregular shape and has both convex and concave

regions. The corrections are once again shown in green, and become lighter from start to finish. The boundary has been highlighted for illustrative purposes.

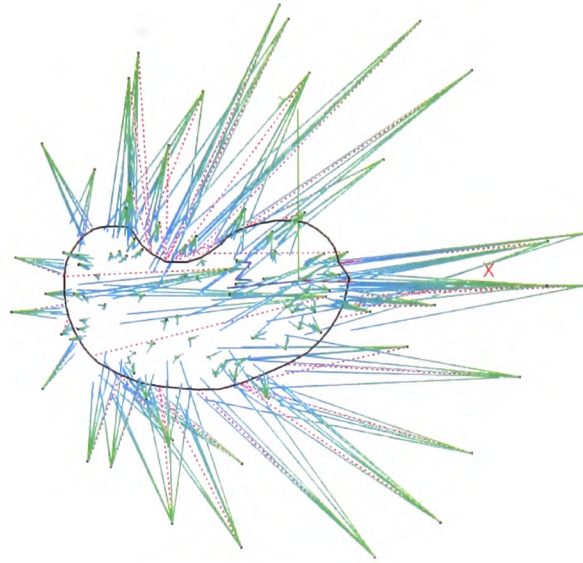


Fig. 30 -A visualization of the irregular boundary results, ideal corrections are shown as red dashed lines, neural network corrections (for each pattern) the green solid line. A fifth of the patterns are shown to improve clarity.

It is clear from the results in Fig. 30 that the neural network has performed well. Some error is present inside the boundary where valid points are very slightly adjusted. Error is also present towards the rear of the sphere and other areas. This is best highlighted using a plot of the test set quaternions graded by the Pythagorean error (using the Pythagorean distance in three-dimensions,) of the applied resultant quaternion, Fig. 31.

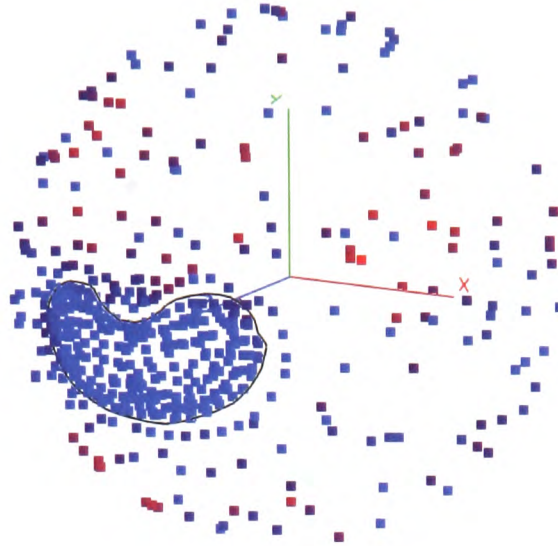


Fig. 31 -Test patterns coloured to represent the 3D Pythagorean error of the relative neural network output. The blue patterns represent low error and the red ones high error.

The pattern of error is similar to that observed for regular boundaries, with the exception of the cluster of red (high error) points around the concave region of the boundary.

The results of the experiments varying the shape of the boundary show that the neural network is remarkably accurate on the majority of boundaries as shown in TABLE VII. There is strong agreement between the MSE and the recorded Pythagorean error, with the exception of the best two in each case. The highly irregular boundary one of the more complex boundaries showed very high performance, though compared to the other boundaries it required more hidden nodes to achieve this performance. The worst performance is observed for the C-Shaped boundary, though this is not significantly worse than the large boundary.

TABLE VII
A COMPARISION OF IRREGULAR BOUNDARY SHAPES

BOUNDARY	Average 3D Pythagorean Error	AVERAGE MSE	Average Hidden Nodes
Highly Irregular	<i>0.030</i>	9.33E-04	<i>18.2</i>
Small	0.033	9.14E-04	<i>16.6</i>
Original	0.040	1.24E-03	14
Large	0.056	2.98E-03	16.8
C-Shape	0.057	3.30E-03	16.2

It is useful to visualize these boundaries to identify patterns in the distribution of the points. The results are shown in Fig. 32, and are labelled as follows; C-Shape (a), Large (b), Small (c) and Highly Irregular (d).

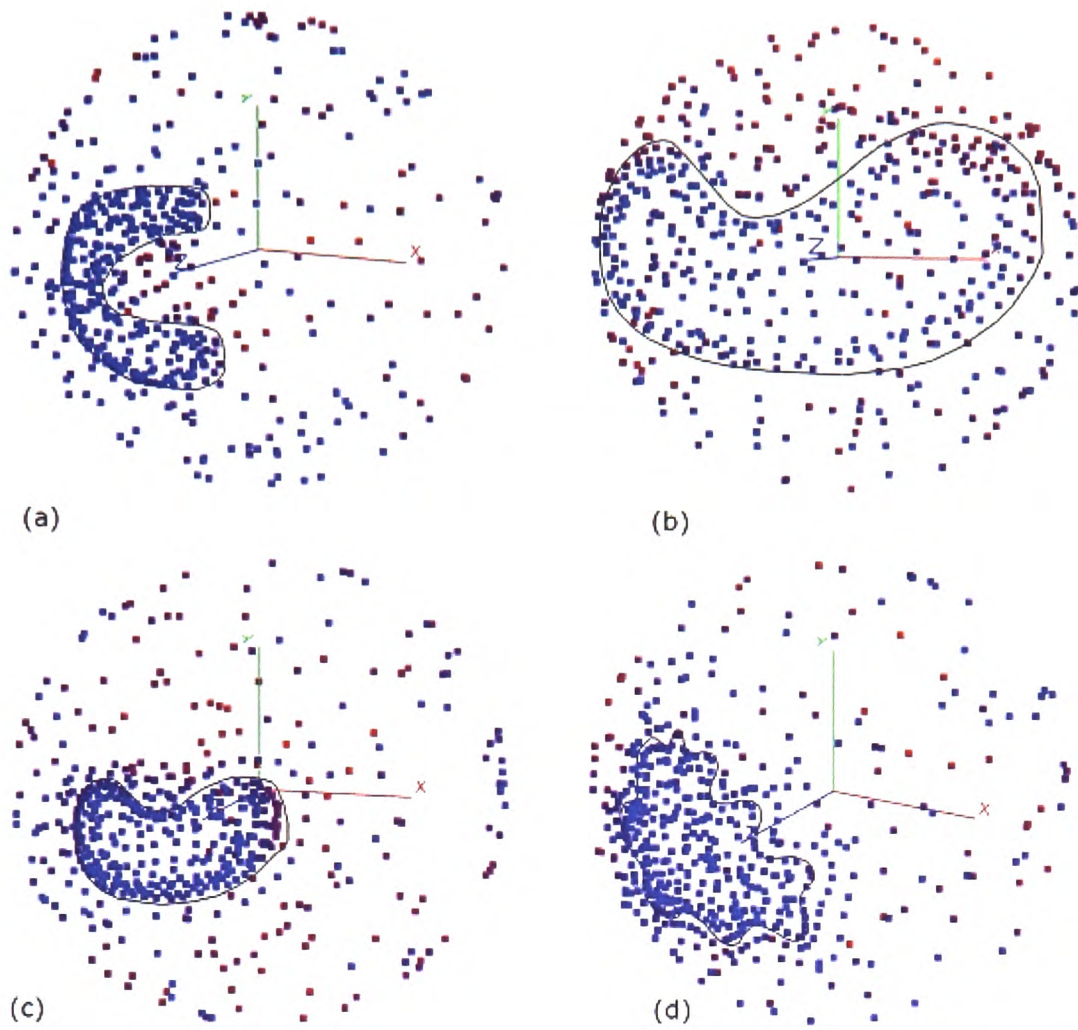


Fig. 32 -The additional irregular boundaries visualized using the colour graded points method described above.

The results show that the neural networks performed well, though difficulties were encountered with regards to concave regions. This is most noticeable in the case of the C-Shape boundary and the large boundary (Fig. 32 (a), Fig. 32 (b)). The small boundary and the highly irregular boundary give much better results. Again the highest errors are around the discontinuities. Poor neural network performance is observed for regions of the highly irregular boundary as depicted by Fig. 33 - in some regions the boundary is attenuated.

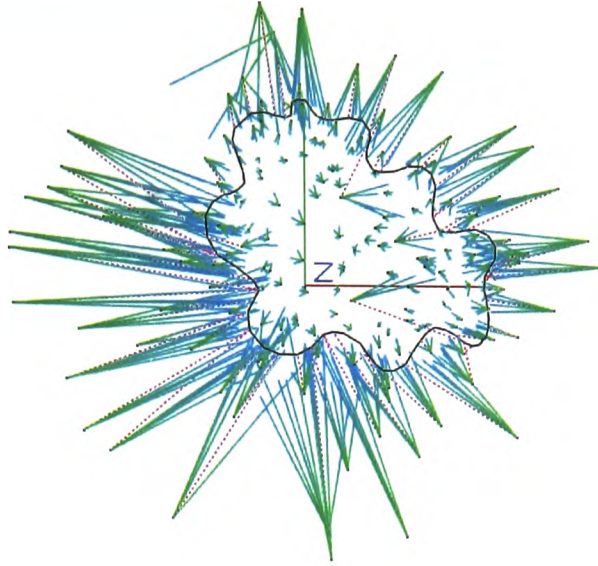


Fig. 33- A visualization showing the highly irregular boundary only 1/3 of the patterns are shown to aid clarity. Solid green lines represent the valid patterns while dashed red lines represent the ideal patterns, both get lighter from invalid to valid.

The visualization (Fig. 33) illustrates the slight errors that still occur at the boundary. Some of the concave regions between convex regions are lost while others train well and are clearly visible.

4.3 Discussion

4.3.1 Regular Boundaries

The results show that artificial neural networks can be successfully evolved and trained to correct joint rotations to a regular boundary. Boundaries similar to those of Korein [36], Engin *et al* [41] and Manurel *et al* [10] have been implemented though in quaternion space. Herda [4, 5] provided corrective constraints but reduced the dimensionality of the quaternion representation to do so, the approach presented here removes this additional complexity. Additionally the correction method used in their work was iterative and therefore inefficient in comparison to the vector field approach adopted here (Johnson [2] also used an iterative approach to correction). Both approaches reduced the dimensionality of the quaternion introducing a complex mapping and singularities, similar boundaries can be implemented by our approach

without reducing the dimensionality of the representation. The approaches of Liu and Prakash [3] (which extend Lee [6],) decompose the quaternion into two quaternions representing rotation in a single plane. This effectively gives Euler angle like constraints with a quaternion-based parameterisation.

Circular boundaries of a number of different radii were trained and the results show good performance, with an average error of less than 1% and a maximum error of only 6.3%. An increase in error matched by an increase in network size indicates an increased complexity between 20 and 90 degrees. PCA reveals that there are significant changes in the distribution of the data in quaternion space that account for this increase as for these ranges the distribution of quaternions indicated by the orientation of the principle components of the dataset change as shown in Fig. 18. These changes in the distribution of patterns in quaternion space may increase the overlap of internal distributed representations French suggests this can increase the extent of interference between patterns which inhibit learning [111].

As the complexity of the evolved networks increases the standard deviation of the number of hidden nodes evolved decreases. This indicates all the networks evolved for these ranges were close to the maximum complexity set during these experiments, (20 hidden nodes). Limiting the number of hidden nodes in combination with the regularization function has prevented an increase in complexity and contributed to the increase in error.

The Pythagorean error between the ideal correction and the neural network correction shows that the network performs well for all but a few patterns. In practice these patterns occupy regions a large distance from the constraint boundary. However, this is not a problem in that modelling anatomically constrained joints it is unlikely the joint would move far beyond the boundary before being corrected. It is important to note that these plots represent the average error over the networks created from repeating the results, observing the plots for all five separately regional performance variations are not constant.

The Pythagorean error is generally highest at the boundary due to the discontinuity in this region and at a second discontinuity present in the region opposite the boundary where proximally equal corrections to two valid boundary positions must be considered.

High Pythagorean error may be recorded in these regions despite the corrected joint reaching the boundary. Other regions of high error such as those shown in Fig. 24 can be attributed to sparse areas of training data. It is important to note however that despite the relatively high error, the correction is in all cases to a configuration within 0.126 (or 12.6% of the limb length). In practice further resources (training, hidden nodes) or iterative approaches could be used to improve on these results.

The results demonstrate that evolved neural networks of low complexity can be used to implicitly model simple spherical joint constraints similar to those modelled in exiting approaches [3]. However the evolved neural network constraints do not project the quaternion into a space with fewer dimensions and require no pre-processing.

4.3.2 Irregular Boundary

The irregular boundary results have a lot in common with the regular boundary results, though the MSE is higher due to the increased complexity of the mapping. The 3D Pythagorean errors are also higher but follow a similar pattern with regions of high error where discontinuities occur between valid and invalid configurations, diametrically opposite the boundary and in areas where training patterns are sparse.

Additionally there is high error around the concave region of the boundary caused by the complexity of the vector field in this region. In the centre of the convex region is another vector field discontinuity, as invalid configurations are of equal proximity to valid configurations on either side. In this case as in the case of the region opposite the boundary, the network may correct a point to the boundary based on training patterns in the region but correct to the wrong side of the sphere compared to the test set.

The range of irregular boundaries experimented with demonstrate the capabilities of these neural networks in learning joint constraint boundaries of the kind necessary for anatomically correct constraints, such as the knee, shoulder etc. The error in three dimensions is low and there is a good correlation with the ideal corrections, in terms of the virtual arm (which is unit length) an average error of 6.3% is reported. An interesting limitation concerning invalid boundaries occurs where the boundary is concave or convex, these local features are attenuated or lost. This can be attributed to

two factors, the first being pattern distribution and the second the learning method of the sigmoid based neural network. There is little difference in shape between the large (Fig. 32 (b)) and small (Fig. 32 (c)) boundaries yet a noticeable difference in performance is observed. This may be attributed to the density of patterns, the smaller boundary has the same number of patterns within the constraint but confined to a smaller region.

In its learning method the sigmoid-based neural network demonstrates good global learning - that is it learns large general mappings well. It is however insensitive to local features such as the boundary discontinuity and shape irregularities that are sometimes lost.

The neural networks evolved to learn the irregular boundary mapping were more complex in nature in that they had a higher number of hidden nodes. This indicates that a more complex neural network was required to train the more complex vector field.

The regular boundary experiments (which converted a random rotation to a quaternion) generated the valid and the majority of invalid quaternions on one half of the quaternion hyper-sphere. There is ambiguity in the rotations represented by the unit quaternions, in that the quaternion sphere represents 4π rotations.

In an attempt to improve performance this ambiguity was removed by limiting the distribution of points to one side of the quaternion hyper-sphere. The results deteriorated in performance for larger constraints, this is attributed to the continuity of the valid and invalid regions in quaternion space. TABLE IV shows that the change in distribution of data in quaternion space as the difference in the variance of the eigenvalues increases (indicating that the non-ambiguous dataset is becoming more elongated than its ambiguous counterpart) there is a marked decrease in performance. Because of the system used to generate the input quaternions in the ambiguous case there was no ambiguity in the valid region. Quaternions on both sides of the hyper-sphere were corrected to a boundary on one side of the hyper-sphere. Forcing the quaternions to one side of the sphere appears to have affected the continuity of valid and invalid regions, increasing the number of discontinuities the neural network must learn to approximate.

The irregular boundary experiments used a sampling dataset generator that generated a quaternion representing the rotation of a control limb being manipulated in three-dimensional space. Because of this the quaternions it generated were distributed over the whole of the quaternion sphere. The quaternion space vector field therefore contained two valid regions. Quaternion joint configurations were corrected to the nearest of these two boundaries. There are in effect two vector fields, one on each half of the hyper-sphere separated by a discontinuity. Training neural networks for such datasets failed to produce any networks with acceptable performance.

To overcome this the valid dataset and the boundary used for corrections was moved to the one side of the unit quaternion hyper sphere. Quaternions from both sides of the hyper sphere are corrected to one boundary, giving a vector field with a continuous invalid region in quaternion space. The neural network successfully learns this vector field which now contains a single valid region and a single discontinuity at the implied constraint boundary.

The choice of axis at the centre of the constraint affects the mean squared error and the actual error in three-dimensional space in different ways. Principle component analysis indicates that the distribution of patterns in quaternion space is most regular with the constraint centred on the y-axis and that as the regularity of the dataset decreases the Pythagorean error increases. According to the Pythagorean metric a dataset with a more regular shape gives superior results as in the case of quaternion ambiguity. A possible reason for this is that the increased distribution of patterns in quaternion space reduces the overlap of internal distributed representations French suggests this can reduce the extent of interference between patterns which inhibit learning [111].

Indirectly measuring the quaternion error has proved very useful and provides an insight into the behaviour of this technique when applied to simple anatomical models. Principle Component Analysis (PCA) has also proved useful in determining the shape and orientation of datasets in quaternion space.

4.4 Conclusion

In conclusion, evolved neural networks show promise in the implicit modelling of joint constraints. This chapter has demonstrated the successful learning of corrective joint constraints using quaternions, without reducing their dimensionality in order to do so, unlike the previous work of Herda *et al* [4, 5] and Johnson [2]. As this approach deals with the rotations of the limb directly (parameterised as a single quaternion) there is no decompose the quaternion as in the approaches of Lee [6] and Liu and Prakash [3].

In addition to the obvious vector field discontinuity at the joint constraint boundary other discontinuities have been identified. A discontinuity also exists at any point where two different boundary points are candidates for correction. There are two regions where this applies, the region diametrically opposite the boundary and in concave regions of the boundary. In such concave regions, the poor local learning properties of the neural network paradigm may contribute to the error. This gives strong motivation for experimentation with other paradigms which offer good global and local learning, such as mixed activation function approaches [53, 65, 83, 85] and adaptive spline activation function neural networks [63, 64, 81, 90, 93-95].

The distribution of patterns on the quaternion hyper-sphere has influence over the neural network training, more specifically the shape and orientation of the dataset. The results indicate that vector fields generated with the y-axis at the centre of the constraint may produce a more evenly distributed vector field in quaternion space, improving the results.

The neural network performs poorly if ambiguities are present in the quaternion space vector field. To overcome this, the boundary and valid points must be defined on one side of the quaternion hyper-sphere, and quaternions on both sides of the hyper-sphere corrected too this. Moving all quaternions to one side of the hyper sphere produces poor results for large constraints as the vector field becomes malformed.

In conclusion genetically evolved neural networks are applicable for joint constraint modelling using quaternion as a result of their capabilities in learning vector fields in quaternion space. These capabilities are in turn dependant on the formation of the vector field in quaternion space.

5. Increasing Performance

In previous chapters both the evolution of the neural networks and their training was limited to maximise productivity. These limitations, while reducing training times for the networks have a detrimental effect on performance.

In order to illustrate the improvements in performance possible with additional training, several key factors in the performance of evolved neural networks were investigated. In some cases the discoveries influenced the creation of earlier datasets though where training times were increased performance was sacrificed.

5.1 Methodology

There are a number of factors that influence the performance of genetic algorithm evolution and neural network training. The following experiments aim to investigate several factors identified in the literature that effect the performance of neural networks and genetic algorithms [49, 79, 134].

In the following experiments two separate discontinuous vector field mappings are investigated. The first represents a vectorial correction to a spherical constrained region of input vectors in three-dimensional space (as discussed in Chapter 3), and the second a quaternion correction to a given boundary from an initial quaternion orientation (as discussed in Chapter 4).

5.1.1 Number of Hidden Layers and Nodes

In practice when utilizing Multi-Layer Perceptrons in their native form the ideal number of hidden layers and nodes is identified by trial and error [49, 81]. If not enough nodes are present then the neural network may not be powerful enough for the given task. If too many nodes are present the training time increases may be unacceptable or more seriously the neural network will loose its ability to generalise [49, 81].

This problem can be alleviated through the use of genetic algorithms, which search the space of viable networks for one suited to the current dataset. A population of neural networks is created these have the maximum number of hidden nodes, the appropriate region of the *Modified Miller Matrix* (below the main diagonal as shown in Fig. 7) is randomly populated creating the connections between the nodes. This leads to a number of hidden nodes not being connected these nodes are evolved but are not present in the phenotype. Binary markers present on both the links and nodes indicate their contribution to the phenotype, if these bits change during cross over or mutation a node may be deactivated. In which case neither the node or any associated links are represented in the phenotype [81].

A regularization term in the fitness functions of the genetic algorithm favours smaller networks. Small networks have a number of properties such as a low computational complexity and good generalisation ability that make them favourable. As part of the evolution of the networks via genetic algorithms, the number of links between the nodes is evolved.

The experiments that follow are concerned only with the maximum number of hidden nodes evolved in the hidden layer, this is an upper bound on the complexity of the network [127]. The regularization function aims to minimise the number of links and therefore the number of hidden nodes is kept below the maximum.

5.1.2 Training Epochs

The ideal number of training epochs can be determined by examining the performance of the neural network on a test set (unseen patterns) and comparing this with the training set (previously seen patterns.) At some point during training the performance on the test patterns starts to decrease. When this occurs the network starts to “memorise” the training set, losing its ability to generalise this is known as over-training [49].

Traditional approaches used a trial and error approach to identify over or under training. Evolution provides an alternative to this time consuming approach. Using the performance of the network on a unique set of validation patterns the genetic algorithm

evolves an ideal number epochs. The following experiments are concerned only with the maximum number of training epochs, the minimum being zero.

5.1.3 Number of Patterns

Several rules have been suggested for the number of training patterns given a network configuration with varying successes [59]. NetJEN has the capability to evolve the number of training patterns however this is achieved by sampling a dataset of maximum size. As pattern order appears to have a significant effect on the result an identification of the ideal training set size is attempted by experimentation. More patterns in the training set increases the length of time required to train the population of neural networks. To limit the time required for each experiment in earlier experiments the number of patterns was limited to five hundred. Dr. Helmut Mayer (a noted researcher in evolved neural networks heavily involved with both the NetJEN project and NetGEN its predecessor,) suggested this limit in a private correspondence.

5.1.4 Pattern Order

The order in which patterns are presented to the neural network has an impact on the training. Due to the global learning nature of the multi-layer perceptron patterns in one locality may affect weights associated with another. Patterns may be presented in any order, in the experiments present here these were limited to the following configurations; valid patterns followed by invalid patterns, invalid patterns followed by valid patterns and both valid and invalid patterns in a random order. The patterns from the original ordered dataset were moved to new random locations to create the random order data set. Each of the datasets contained the same set of patterns in a different order.

5.1.5 Pattern Distribution

There are a number of ways in which the patterns could be distributed (or the mapping sampled.) Distributions with points clustered in a region close to the boundary and an even distribution over the whole surface were compared. In the clustered set a dense

region of points was placed in the immediate vicinity (5 degrees) of the boundary. There were four regions in total, as illustrated in Fig. 34. In the non-clustered set the valid and invalid regions were divided into four regions containing an equal number of points. These are illustrated in Fig. 34.

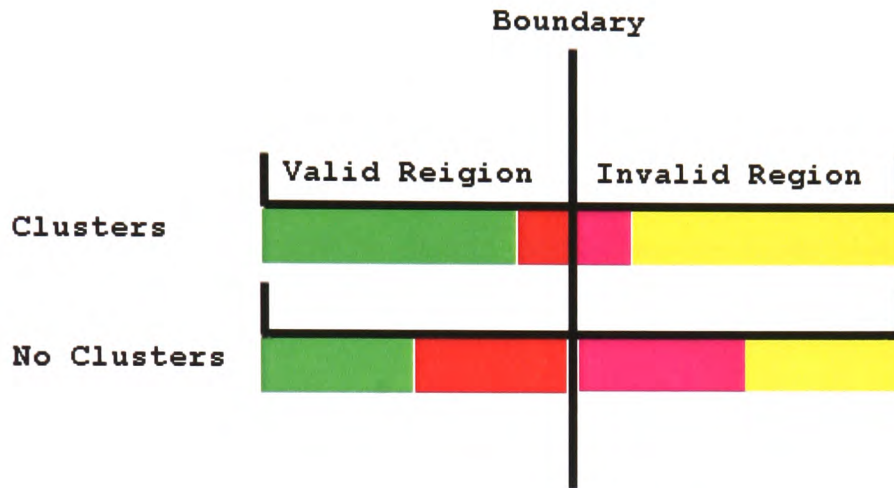


Fig. 34 - A diagram showing the distribution of patterns in the datasets. The number of patterns in each coloured region is constant irrespective of its size.

In both distributions the number of patterns within each of the regions is constant irrespective of the constraint size. This maintains independence between the size of the constraint and the distribution of patterns that may otherwise lead to small constraints having very few or no patterns in their valid region.

5.1.6 Generations

Genetic Algorithms are less susceptible to over training (local minima), though are constrained by the size of the population, number of generations, mutation rates etc. [79]. The performance of a neural network increases until the population contains a number of networks with similar genes. Further increases in performance require large numbers of generations. There is however some optimal point where increasing the number of generations still results in an increase in performance [135].

5.1.7 Population Size

The size of the population primarily effects the diversity of individuals within the gene pool. Potentially each individual carries beneficial alleles (the individuals genotype is a set of alleles,) that combined with alleles from other individuals may form a better solution. Larger populations display poor initial performance due to their slow changing populations, though their performance over a number of generations is better as their larger population allows them to maintain a more diverse gene pool and avoid allele loss leading to a better solution [134, 135]. Small populations display better initial performance as the small population changes quickly, however due to a loss of alleles from the gene pool the final solution is inferior to that of a larger population [134, 135].

It is clear that population size and the number of generations are linked. A small population may out perform a large one if the number of generations is limited [135]. A constant population size was maintained throughout the earlier experiments to minimise the training times. The effects of population size are explored in an attempt to identify an ideal population size for these experiments.

5.1.8 Activation Function

The activation function has been shown to be significant in effecting the performance of a neural network [64, 84]. The effect of activation functions can be explored using genetic algorithms, in the approach used [63] the activation function of layers of nodes are encoded as part of a genome. The genetic algorithm searches for the best of the encoded types this is described as pure activation function evolution [65]. The activation functions for the hidden and output nodes were evolved, with the following candidate functions; Gaussian, linear, sigmoid, sinus and hyperbolic tangent. These are shown in Fig. 35.

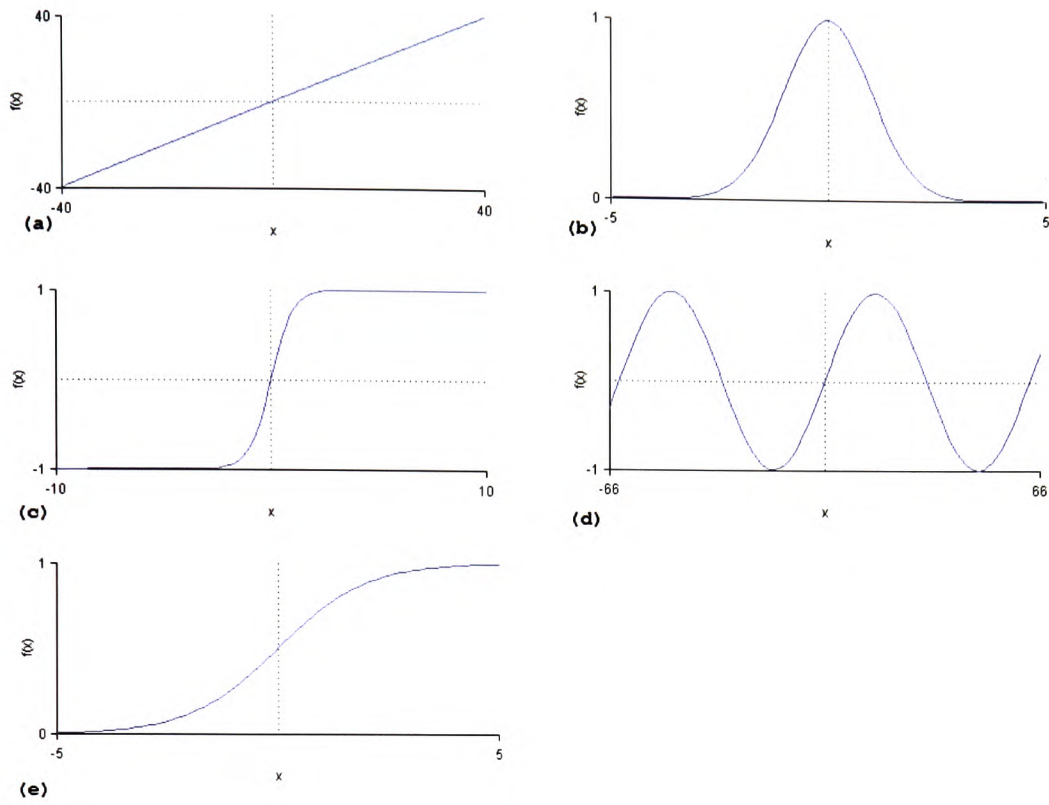


Fig. 35 - The candidate activation functions for evolution. The functions depicted are linear (a), Gaussian (b), hyperbolic tangent (c), sinus (d) and sigmoid (e).

The activation function information (as show in Fig. 6(d)) is encoded into the genome (as shown in Fig. 6(a)) and converted into a linearized binary adjacency matrix as demonstrated for the example network in Fig. 7 [83].

Alternatively a spline based function is evolved to form the shape of the activation function required [63, 64]. Early experiments utilized a neural network with sigmoid activation functions in the in both the hidden and output nodes, significant improvements were made by replacing the sigmoid activation functions in the output layer with a linear activation function. These results are included to justify the choice of activation function in the main body of this work. Template based evolved cubic spline activation functions for the hidden layer are investigated as suggested by Mayer and Schwaiger [63].

The genome representing the template spline activation functions (as show in Fig. 6(c)) is evolved separately and does not form part of the adjacency matrix. Evolution of the

activation function is effectively the same as in the previous case with an index representing each of the spline templates being evolved [63].

5.1.9 Dataset Creation

The results shown in this section are based on datasets from earlier experiments and the creation methods have remained the same. The vector-based experiments concerned a vector field representing a spherical constraint of radius 0.125 with its centre at the origin. Valid and invalid vectors are generated with components ranging from -1 to $+1$, giving a cube with the origin at its centre. In addition to the datasets created in earlier experiments, i.e. the training set, validation set and test set, a fourth dataset was created for the *number of patterns* experiments. Here it was necessary to test the performance of the networks on a common dataset, these contained three thousand patterns. The constraint is described in detail in section 3.1 and the creation of the constraint in section 3.1.1.

A discontinuous vector field in quaternion space that described a twenty-degree constraint on the surface of a unit sphere was chosen along with out initial irregular boundary. Once again additional datasets, for example a common test set with three thousand patterns for the pattern number experiments were created as required. A description of the discontinuous vector fields used to represent these constraints and their construction is contained in sections 4.1 and 4.1.1 respectively.

5.1.10 Activation Function Evolution using NetJEN

NetJEN allows the evolution of activation functions via the two methods outlined above, that is encode the activation function of nodes as part of a genome and have the genetic algorithms search for the best of the encoded types from a list of candidate functions this is described as pure activation function evolution [65]. Alternatively a spline based functions can be evolved which form the shapes of the activation functions required [63, 64].

Mayer, Strapetz and Fuchs [83] implemented pure activation function evolution in the NetGen system. The genome is discussed in section 3.1.2 (and shown in Fig. 6(d)) and

it is noted that the index of the activation function is encoded into the binary adjacency matrix (shown in Fig. 7). The number of bits required to represent the index of the activation function is flexible depending on the number of candidate networks presented [83]. The maximum size of the networks is set in advance and as the structure of the networks is changed links and nodes are switched on and off using binary markers (see Fig. 6) hence the chromosomes contain some non-coding regions [83].

The alternative involves the representation of a cubic spline within the genome the control points of which are evolved during the evolutionary process. The description which follows is based on that documented by Mayer and Schwaiger [63, 64] for the earlier system on which NetJEN is based. A fixed number of control points are used to describe the cubic spline activation function as outlined in equation 69.

$$(x_i, y_i) \in \mathfrak{R} \times \mathfrak{R} \text{ where } i = 1, \dots, n \quad (69)$$

Here n is the number of control points and defines $n-1$ intervals on the x -axis (equation 70).

$$x \in [x_i, x_{i+1}] \text{ with } x_{i+1} > x_i \quad (70)$$

For each of these intervals a function $f_i(x)$ is defined in equation 71.

$$f_i(x) = f_i(x) + a_i(x - x_i) + b_i(x - x_i)^2 + c_i(x - x_i)^3 \text{ where } a_i, b_i, c_i \in \mathfrak{R} \quad (71)$$

This demands equality of the function value and that the first and second derivative can be determined for each interval yielding a continuous and differentiable function composed of a number of cubic splines. The number of control points, range of the cubic-spline activation (sensitivity interval) and its limits during activation (activation interval) must be configured before evolution begins. Within these boundaries the spline is free to develop.

NetJEN implements template-based spline activation function evolution that relies on the encoding and evolution of the spline as indicated above. However Huber and Mayer

[63] found results were improved by evolving a set of custom spline activation functions (called templates) and simultaneously evolving the functions of the hidden layer with these templates as candidates. The spline templates are encoded and evolved as part of the genome (shown in Fig. 6(c)), the template associated with each node is evolved and encoded in the same way as the candidate functions are in the pure activation function encoding approach [63].

5.1.11 Training and Evolution

In each experiment the NetJEN system (described in 3.1.2) was configured as follows. The input layer represents the current joint either as a three-dimensional unit vector or a quaternion rotation. The output layer represents the correction either as a corrective three-dimensional vector or a corrective quaternion rotation. The number of hidden nodes and connection topology are randomised and then evolved during the learning process using Genetic Algorithms. The weights of the interconnections are initially randomised then updated using the resilient back-propagation algorithm. The evolution and training parameters, where constant, were set as shown in TABLE VIII and each experiment was repeated five times to ensure the consistency of the results. Unlike previous sections the table shows the constants used for each parameter when it was not the subject of investigation.

TABLE VIII
EVOLUTION AND TRAINING SETTINGS

Parameter	Description	Value
Regularization function	Secondary fitness function.	Number of links
Hidden Nodes	Maximum no. of hidden nodes.	20
Number of Generations	No. of generations over which the ANN were evolved.	50
Population Size	Size of the populations evolved.	20
Fitness Function	Primary fitness function.	Inverse SSE
Evolve number of Links	Networks are pruned down from fully connected networks.	On
Evolve number of Hidden Nodes	Evolves the no. of hidden nodes.	On
Evolve number of training epochs	Evolves the no. of training epochs	On
Learning Rate	Learning rate used when training the ANN.	0.1
Stopping Error	MSE at which the ANN are stopped.	0.001
Training Function	Training function used to train the weights of the ANN.	Resilient back-propagation
Max Epochs	Maximum number of training epochs	500

The neural networks used sigmoid activation functions in their hidden layer and linear activation functions in their output layer, unless otherwise stated. The justification of this decision is covered in a later section of this chapter.

The regularization weight was chosen based on publications by the authors [63], as was the learning rate [126], the stopping MSE for the networks was identified through experimentation. The size of the population, number of generations and initial limits for the number of training patterns were suggested by a co-author of the NetJEN system Dr. Helmut Mayer in private correspondence.

Additional spline specific parameters were required for the experiments involving the evolution of spline activation functions these are outlined in the following table (TABLE IX), based on those used by Mayer and Schwaiger [63].

TABLE IX
THE CONFIGURATION OF THE SPLINE EVOLUTION PARAMETERS

Parameter	Description	Value
Control Points	Number of control points used to define each spline.	8
Activation Interval	Minimum and maximum activation displayed by the function.	Min: -1.0 Max: 1.0
Sensitivity Interval	Interval over which the functions output is considered.	Min: -5.0 Max: 5.0
Number of Templates	The number of template nodes evolved.	2

5.2 Results

The majority of experiments in this section were carried out for a discontinuous vector field representing a spherical constrained region in three-dimensional space. This constraint showed high MSE and has the benefit of easily quantifiable results. Further experiments demonstrate the applicability of these results to quaternion based corrective constraints.

5.2.1 Discontinuous Vector Fields Representing Three Dimensional Constraints

Firstly the improvements are investigated on vector fields that represent the spherical constraint in three-dimensional space (discussed in Chapter 3).

5.2.1.1 Neural Network Size

Increasing the maximum number of hidden nodes that the genetic algorithm can assign to sixty nodes improves performance on both the unseen test set and on the training, as shown in Fig. 36. After this point the performance on the training set continues to increase while the performance on the test set decreases. Increasing the number of hidden nodes also increased the time required to complete the experiments, eighty hidden nodes taking just over a week to complete on a 2.4 ghz Pentium 4, experiments with twenty hidden nodes took between 16 and 18 hours on the same hardware.

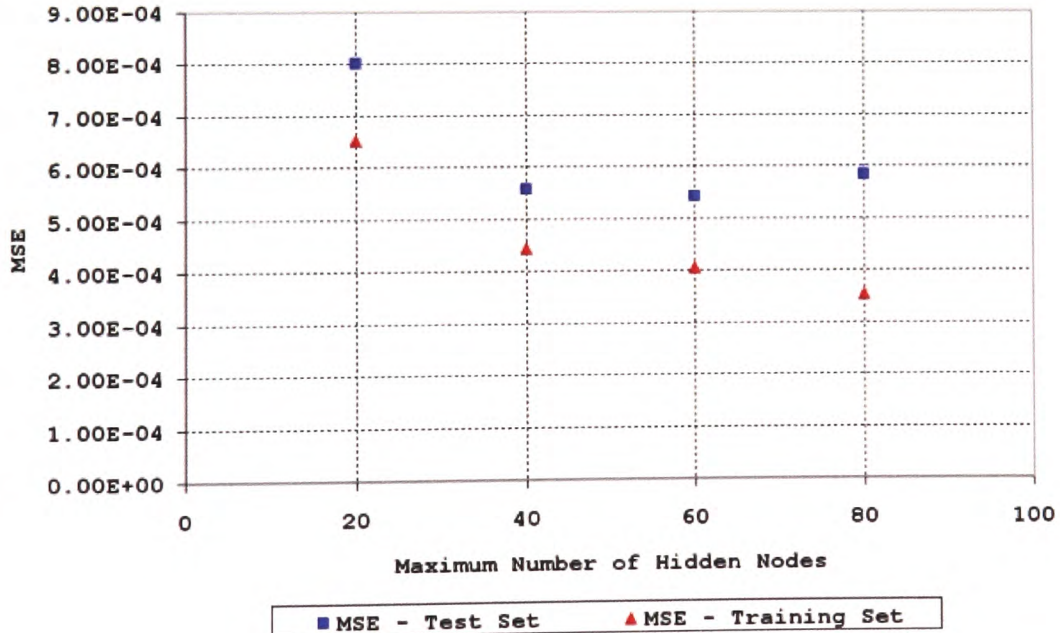


Fig. 36 - The effect of the maximum number of hidden nodes on testing and training set performance. The MSE values shown are averages of the five neural networks evolved and trained in each case.

Examining the number of hidden nodes, links and training epochs it was found that an increase the maximum number of hidden nodes results in an increase in the number of

hidden nodes evolved in the neural network. The number of training epoch evolved also increases as shown in Fig. 37.

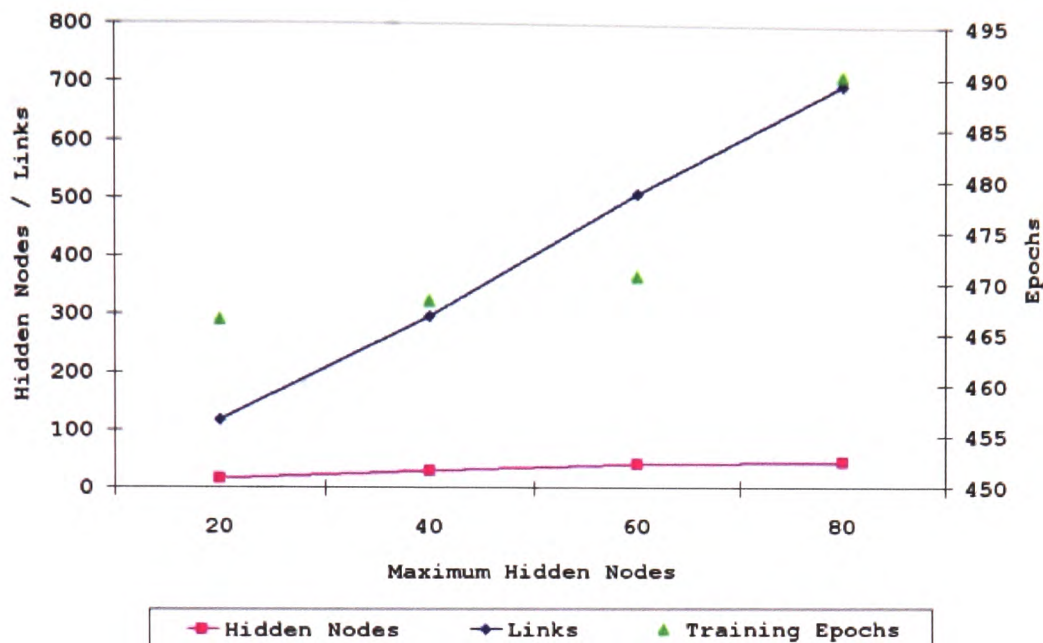


Fig. 37 - Effect of increasing the maximum number of evolved hidden nodes on the topology and evolved training requirement of the neural network.

5.2.1.2 Training Epochs

The number of training epochs has a significant effect on the performance of the neural network and incurs a significant time penalty. Five runs of the two thousand-epoch experiment required over three days to complete. Fig. 38 shows the resulting MSE, a decrease in the gradient of the MSE curve indicates that more epochs would not produce significant improvements in the result.

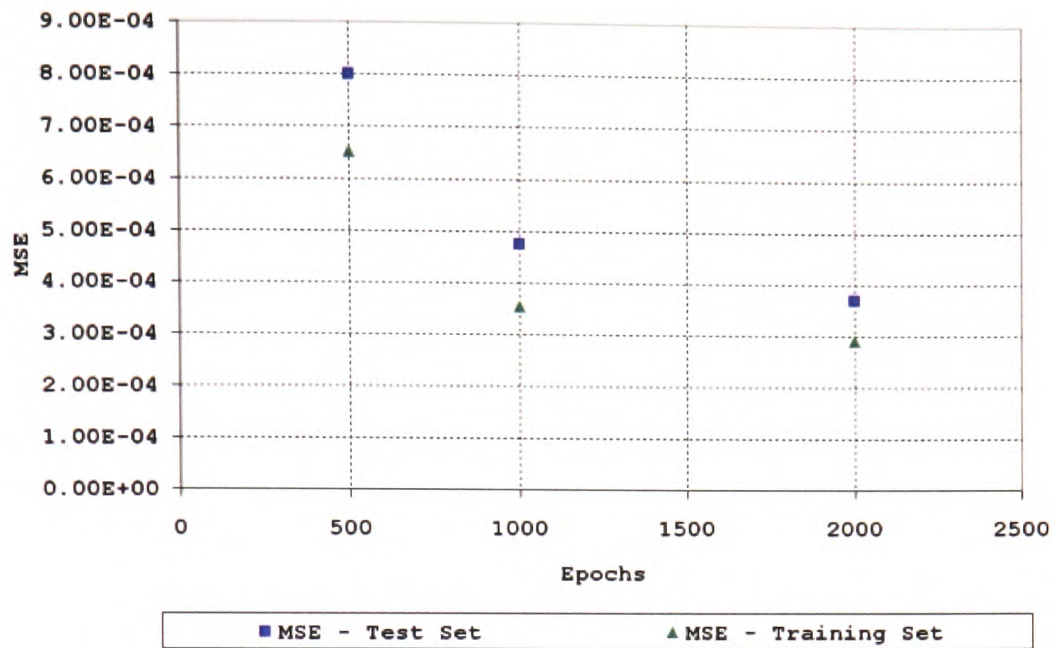


Fig. 38 – A graph showing the effect of training epochs on neural network performance.

Increasing the maximum number of training epochs for neural networks created by the genetic algorithm results in an increase in the number of training epochs evolved for each of the networks generated. However there is no significant effect on the number of hidden nodes or interconnections (links) as shown in Fig. 39.

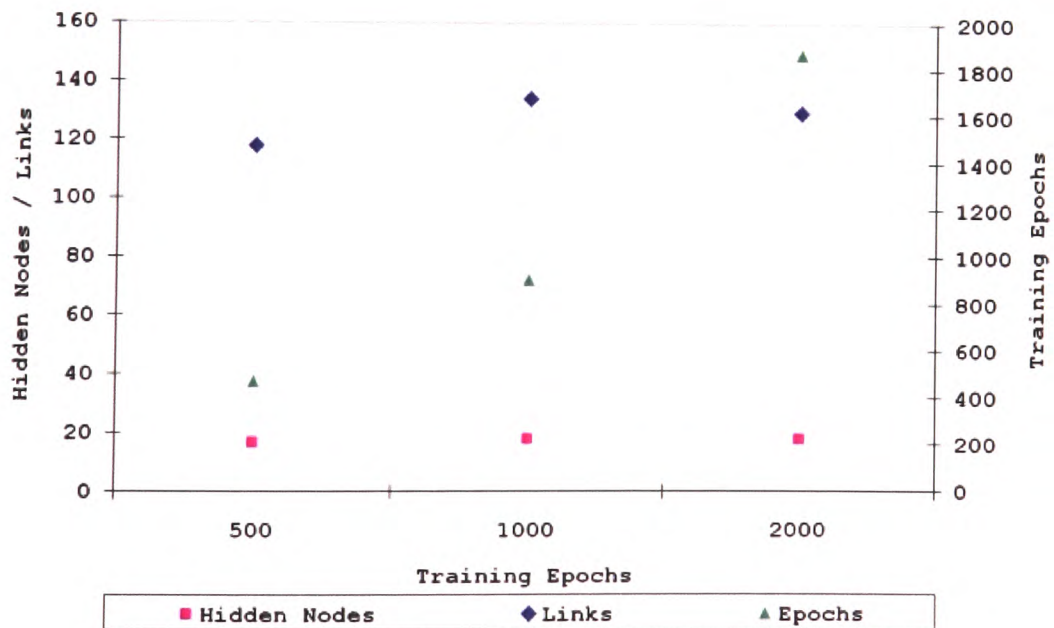


Fig. 39 – A graph showing the effect of training epochs on evolved neural network structure

5.2.1.3 Number of Training Patterns

Increasing the number of hidden nodes utilised an additional test set. As in previous experiments the performance is shown on the previously seen patterns (training set) and unseen patterns (test set) of equal size. This is shown in figure Fig. 40. A common test set containing three thousand patterns was used to give an indication of the performance of each network. The results show that performance on the common test set generally improved as the number of patterns increased. However the scale of the improvements decreases.

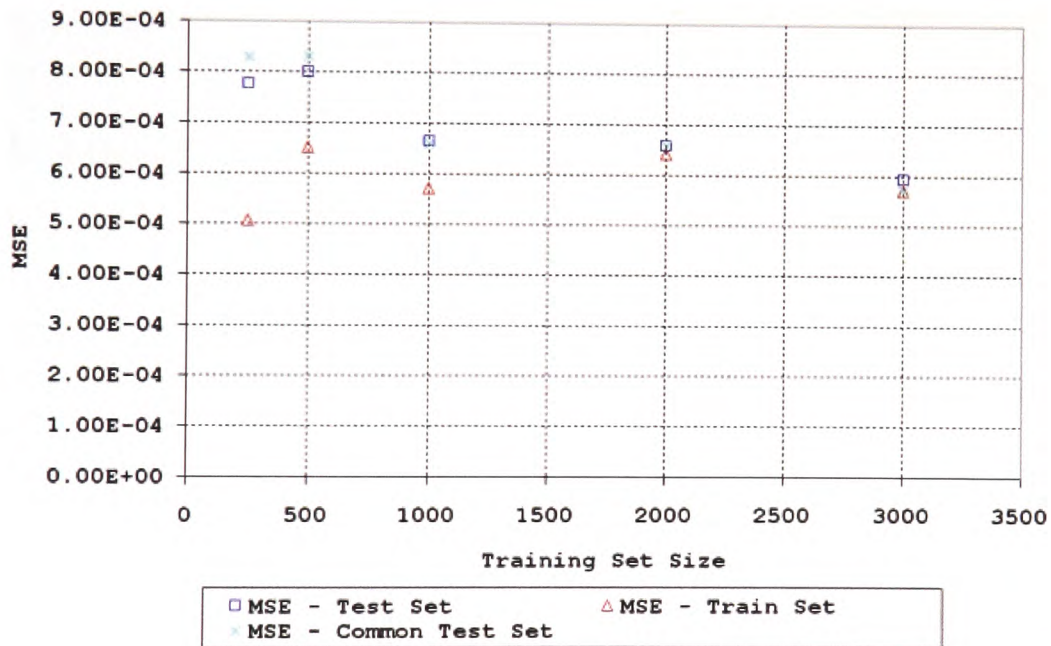


Fig. 40 – The effect of training set size on neural network performance.

Increasing the number of patterns does not appear to have a constant effect on the number of hidden nodes, links or training epochs as shown by Fig. 41. The smallest and largest training sets seem to result in large networks and a low number of training epochs, between these two extremes less neurons and more training epochs are required.

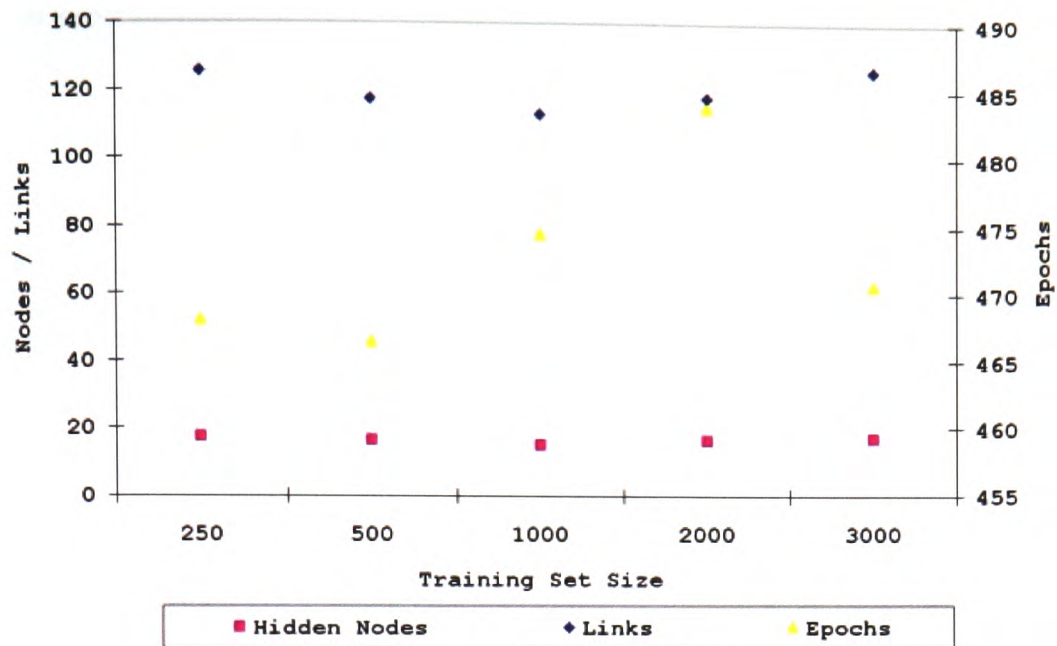


Fig. 41 - Effect of pattern density on the performance of evolved neural networks.

5.2.1.4 Generations

The results show that as the number of generations is increased the performance on the training and test sets increases as shown in Fig. 42. However the gradient of the improvements itself tends towards zero indicating that further increases in the number of generations would not produce any significant improvements in the result.

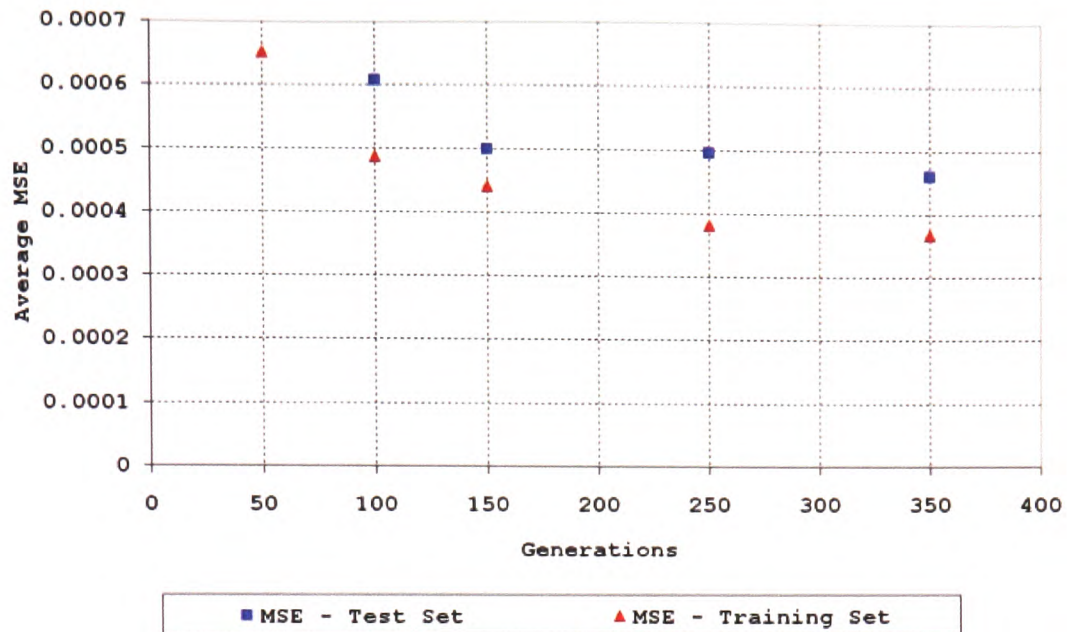


Fig. 42 – A graph showing the Number of Generations vs Average MSE. The result for fifty generations on the test set was much higher and obscured the other results.

Changing the number of generations allows the evolutionary process to continue towards a solution. As evolution is allowed to continue the number of links, hidden nodes and epochs move closer to their limits shown in Fig. 43. This seems to agree with the results observed in the individual hidden node and epoch experiments, in that improved performance requires a larger number of hidden nodes and training epochs.

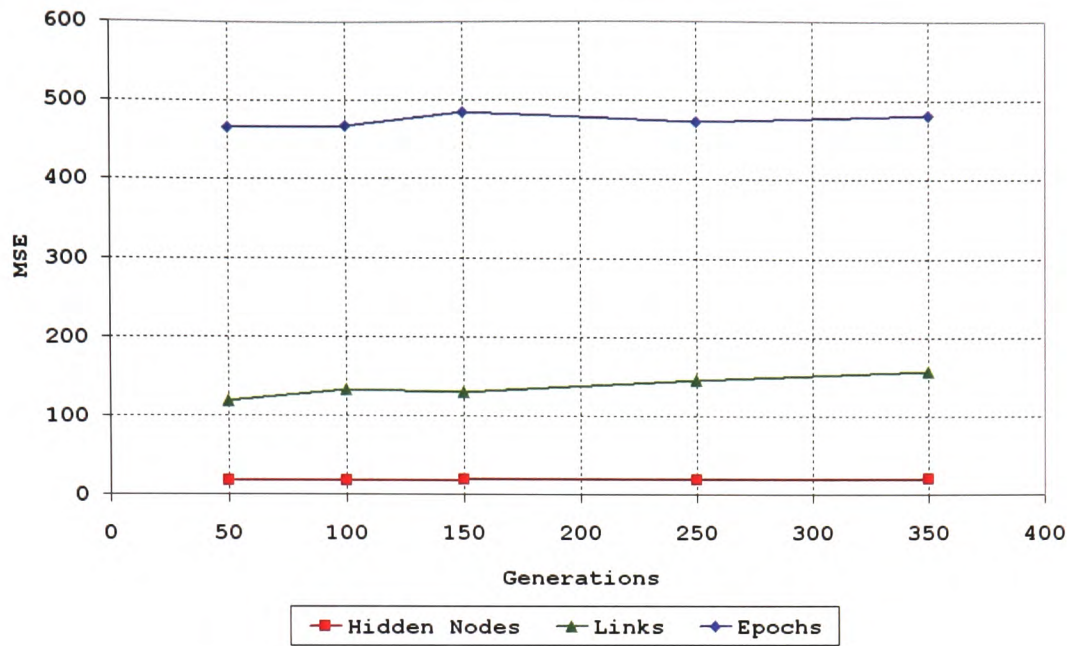


Fig. 43 - Effect of increased evolution on evolved neural networks.

5.2.1.5 Population Size

Experiments were carried out to determine the effect of varying the size of the population of neural networks used during evolution. For all other experiments a population size of fifty individuals was used. Experiments with between ten and one hundred and fifty individuals reveal that an increase in the size of the population produces an increase in performance, (as shown in Fig. 44.) There is a decrease in the gradient of improvement towards a minimum.

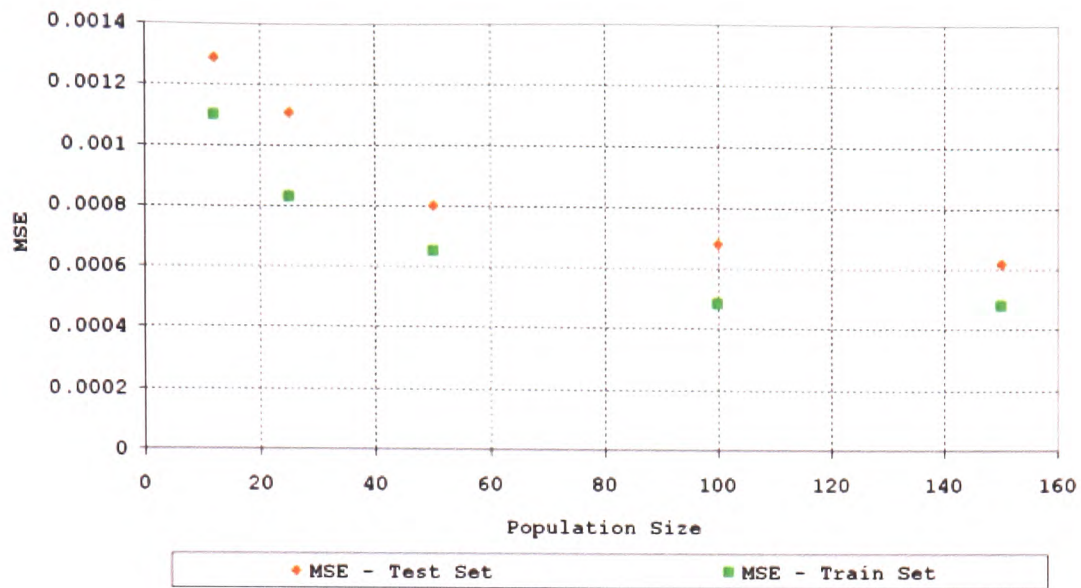


Fig. 44 – Effect of population size on the performance of the evolved neural networks.

The effect of varying the population size on the neural networks evolved is clear from the number of hidden nodes, links and epochs evolved. As the population size increases the number of hidden nodes, links and training epochs also increases, as shown in Fig. 45.

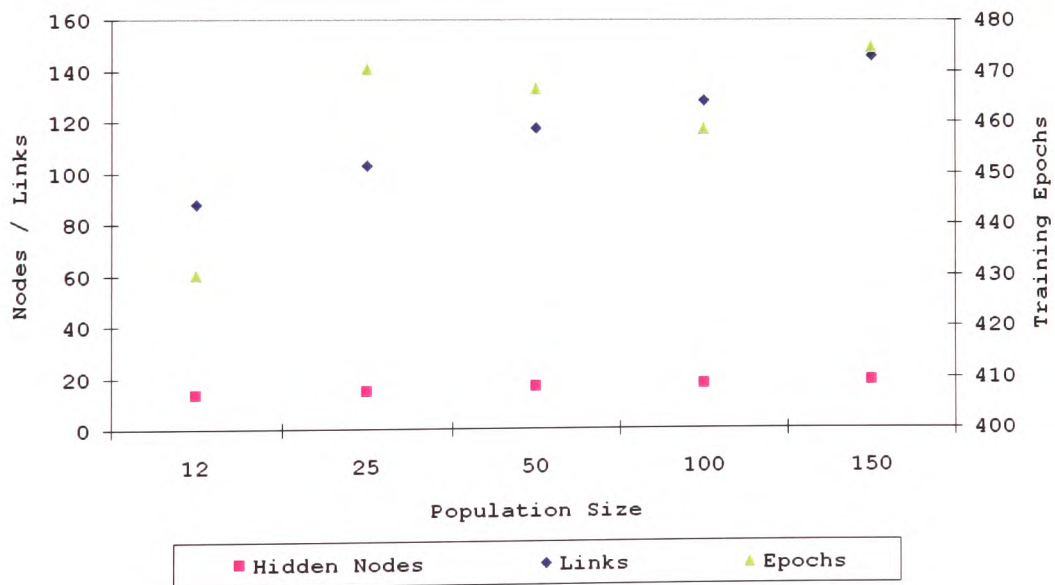


Fig. 45 - Effect of population size on evolved neural network performance.

5.2.2 Discontinuous Vector Fields Representing Regular and Irregular Quaternion Boundaries

The effects of these same factors on the more complex discontinuous vector fields representing quaternion based rotational constraints (as discussed in Chapter 4,) are examined. The results of these experiments provide evidence as to the improvements possible for practical applications of these techniques.

5.2.2.1 Number of Hidden Nodes

As in the previous experiments, increasing the maximum number of hidden nodes the genetic algorithm can evolve within the neural network reduces the MSE, as shown in Fig. 46. The degree of improvement is similar for both training and unseen patterns. Due to the time required to complete these experiments the results are insufficient to identify the point at which the neural network starts to over specialise, (when the test set performance decreases despite and increase in training set performance.) This indicates that further improvements are possible.

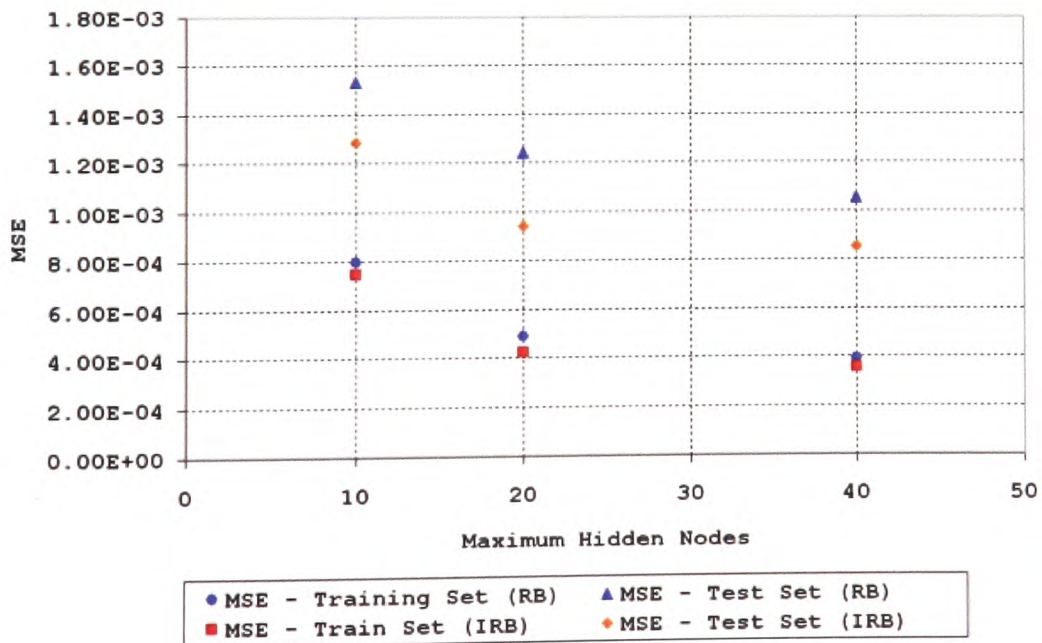


Fig. 46 - Effect of hidden node variation on regular and irregular quaternion boundary constraints.

As in the previous experiments the increase in the maximum number of hidden nodes leads to an increase in the number evolved, (Fig. 47,) and an increase in the number of training epochs evolved with respect to the increasing neural network size.

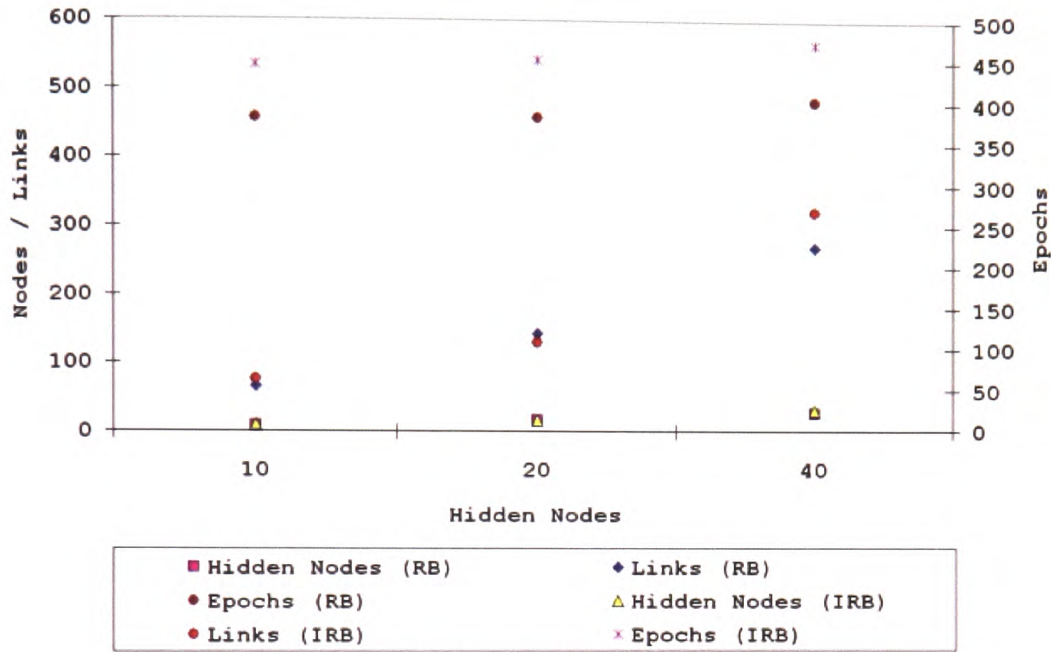


Fig. 47 - Effect of the maximum number of hidden nodes on the topology and training requirements of evolved neural networks trained to model regular (RB) and Irregular (IRB) boundaries.

5.2.2.2 Training Epochs

Variation of the maximum number of training epochs evolved as part of the neural network structure reveals that an increase in the number of epochs results in an increase in the performance of the neural networks evolved, as shown in Fig. 48. As the maximum number of training epochs increases the gradient of this increase decreases. This indicates that a saturation point exists where further neural network training would have no further benefit.

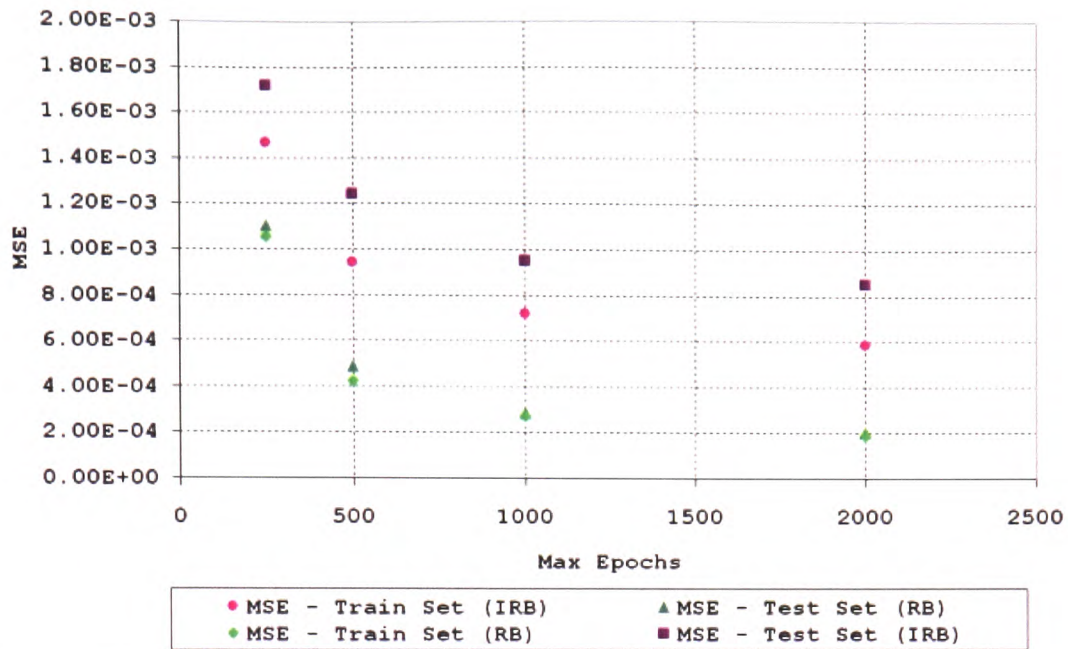


Fig. 48 - Effect of increasing the maximum number of training epochs allowed during evolution on the performance neural networks evolved to model regular (RB) and irregular (IRB) boundaries.

An increase the maximum number of training epochs which can be genetically assigned to any individual network results in no significant change in the neural network size, indicated both by the number of hidden nodes and links, as shown in figure Fig. 49.

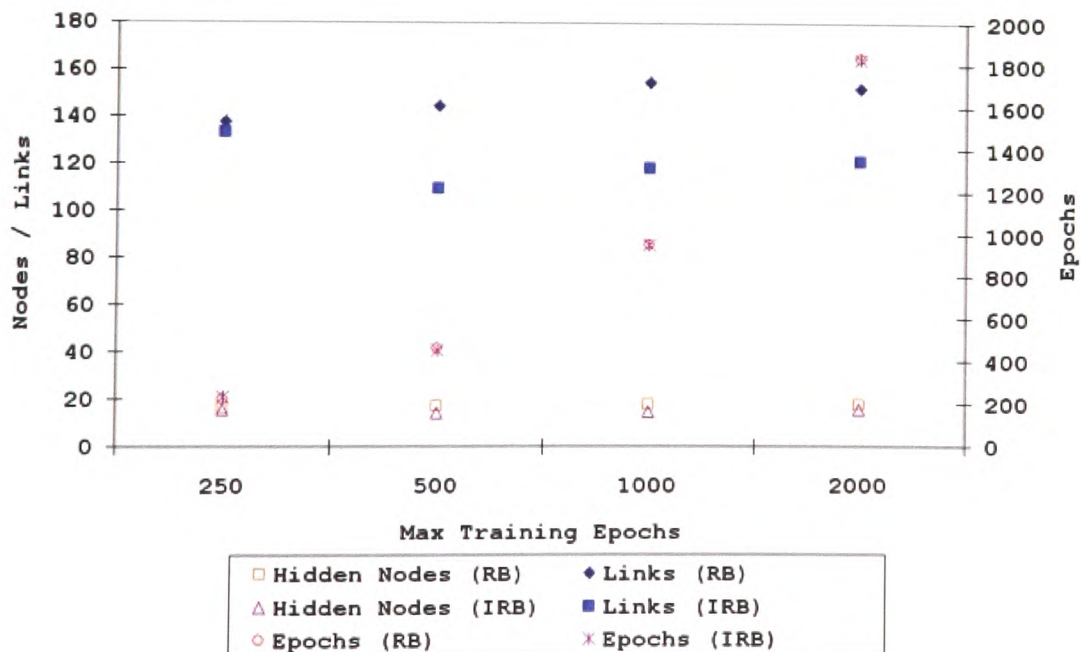


Fig. 49 - Effect of changing the upper limit for training epochs of the neural networks evolved to model regular (RB) and irregular (IRB) boundaries.

5.2.2.4 Training Patterns

In the 5.2.1 above the number of training patterns was investigated using a less complicated three-dimensional point based constraint. An increase in performance is observed in both regular and irregular boundaries, as the number of training patterns is increased. Fig. 50 shows that irregular boundaries demonstrate a marked improvement on the common test set (three thousand patterns). However the regular boundary demonstrates a very small improvement.

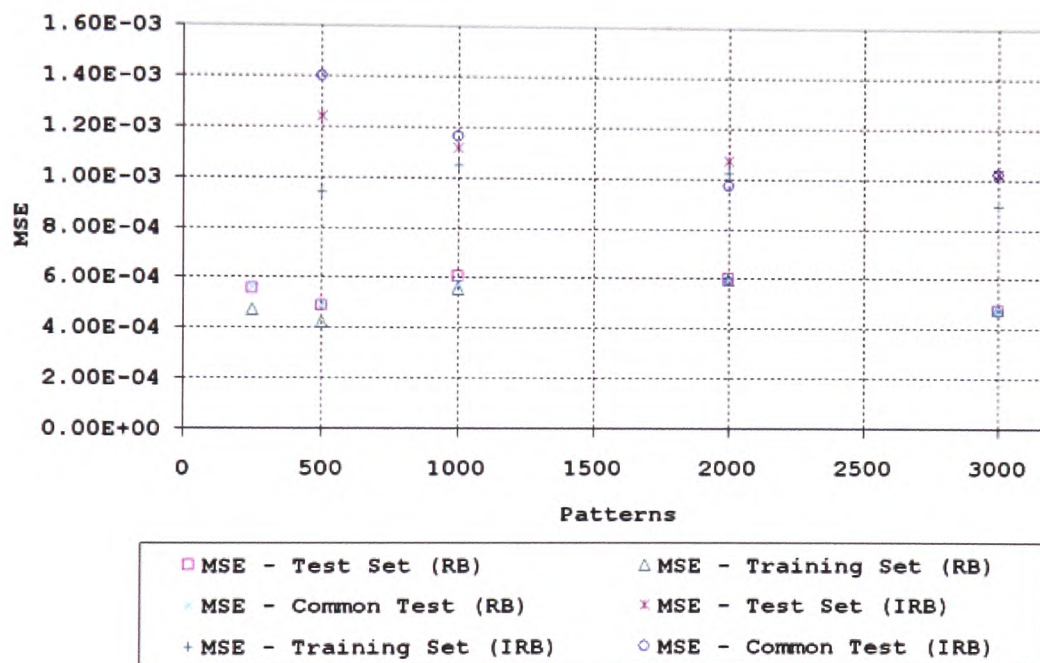


Fig. 50 – A graph showing the effect of training set size on the performance of the neural networks for both regular (RB) and irregular (IRB) boundaries.

The effect of increasing the number of patterns on the number of hidden nodes appears to be more pronounced in the irregular boundary results. Fig. 51 shows both an increase in hidden nodes and epochs indicating an increased difficulty associated with the inclusion of new patterns. In the case of the regular boundary this increase is not present, indicating that in the regular boundary case the problem does not increase in complexity as the number of patterns increases.

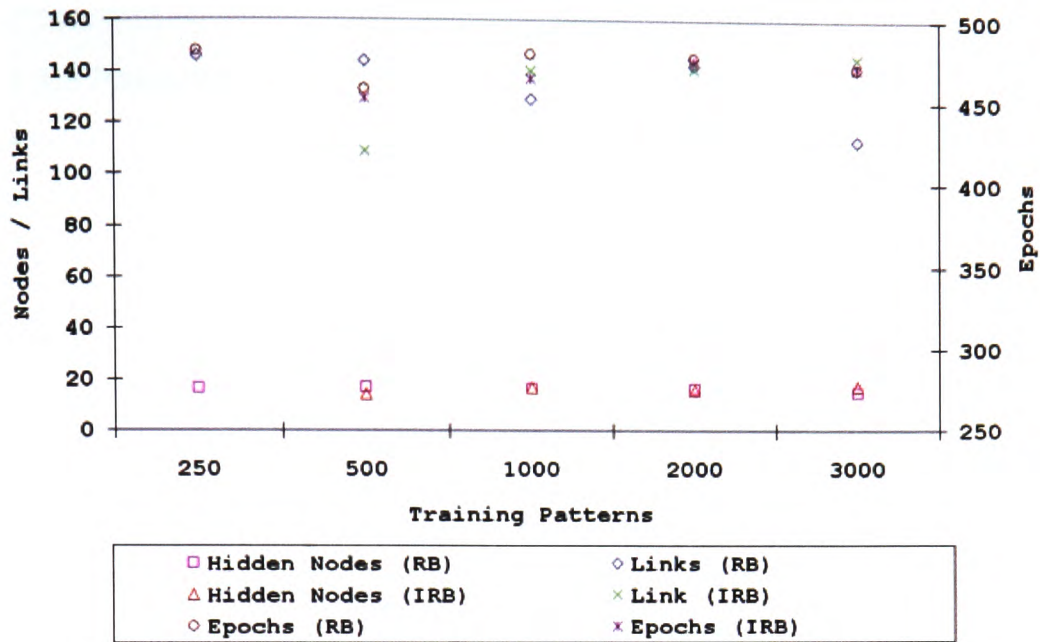


Fig. 51 – A graph showing the effect of changing the number of training patterns on the evolved neural networks for both regular (RB) and irregular (IRB) boundaries.

5.2.2.5 Pattern Order

The results of the pattern order experiment showed the configuration used throughout the experiments, that is, valid patterns followed by invalid patterns, was superior to both the random order and reversed order (invalid patterns first) training sets. These results are shown in TABLE X.

TABLE X
THE RESULTS OF THE PATTERN ORDER EXPERIMENT

	Average	Standard Deviation
Normal	4.88E-04	1.61E-04
Random	6.43E-04	1.10E-04
Reversed	6.49E-04	1.58E-04

5.2.2.6 Pattern Distribution

Comparing the clustered and evenly distributed (non-clustered) datasets (outlined in 5.1.5,) there is a clear improvement in the results for evenly distributed patterns. This can be observed in a comparison of the average network error as shown in TABLE XI.

TABLE XI
THE ERROR AND EVOLVED NODE CONSTRUCTION FOR CLUSTERED AND
NON-CLUSTERED DATASETS

	Avg. MSE	Avg. Hidden Nodes	Avg. Links	Avg. Epochs
No Clusters	4.88E-04	17	144.2	458
Clusters At Boundaries	7.85E-04	17.8	164.6	492.8

The neural networks evolved were of comparable size and required a similar amount of training epochs as shown in TABLE XI. As shown by the number of hidden nodes, links and training epochs evolved by the genetic algorithm over the five test networks also shown in TABLE XI.

5.2.2.7 Activation Function Evolution

All experiments studying the effect of the activation function on neural network performance are presented in the following section. This holistic approach gives the results context.

The section begins with a general overview of the results obtained for the network architectures investigated. These results and the evolution of activation functions concern only the four-dimensional vector fields representing the quaternion based constraints and the earlier constraints of lower dimensionality. Later the template based

spline activation function neural networks are compared to sigmoid-linear neural networks for one, two, three and four dimensional vector fields.

The performances of evolved topology neural networks with a number of hidden and output layer activation functions were examined. Sigmoid activation functions in both the hidden and output layer produced very poor performance. This was countered by the introduction of a linear output layer (this is described as the sigmoid-linear neural network.) Several authors have used bipolar sigmoid (or hyperbolic tangent) neural networks for vector field approximations [54, 55, 57].

The use of evolution in assigning the activation functions produced some improvement over the sigmoid-linear neural network in terms of the average MSE over five evolved neural networks. Sigmoid linear neural networks produced a lower minimum error than both pure evolved activation function and evolved template based spline activation function neural networks, but also produces a higher maximum error. These results are shown in TABLE XII.

TABLE XII

TABLE DETAILING THE PERFORMANCE OF NEURAL NETWORKS WITH
DIFFERENT ACTIVATION FUNCTIONS

Network Construction Hidden / Output	Avg. MSE	Std. MSE	Max MSE	Min MSE
Cubic Spline / Linear	5.46E-04	7.64E-05	6.72E-04	4.42E-04
Evolved / Evolved	6.19E-04	2.25E-04	9.18E-04	3.68E-04
Sigmoid / Linear	6.22E-04	2.12E-04	9.67E-04	3.18E-04
Sigmoid / Sigmoid	1.01E-01	1.26E-04	1.01E-01	1.01E-01

The topologies of the neural networks evolved by the genetic algorithm were varied. The neural network with sigmoid hidden and output layers produced the smallest network but the high error makes them of little use. The sigmoid-linear neural network evolved a number of nodes close to the maximum of twenty hidden nodes to produce

reasonable performance (TABLE XIII). The evolved and cubic spline neural networks give better performance on average and require fewer hidden nodes to achieve this, as shown in TABLE XIII.

TABLE XIII

TABLE DETAILING THE TOPOLOGY OF NEURAL NETWORKS WITH
DIFFERENT ACTIVATION FUNCTIONS

Network Construction			
Hidden / Output	Avg. Hidden Nodes	Avg. Links	Avg. Epochs
Cubic Spline / Linear	14.4	124.8	485.6
Evolved / Evolved	14.6	119.6	433.2
Sigmoid / Linear	17	129.6	475.8
Sigmoid / Sigmoid	8.6	56.6	430.6

When evolving the activation functions of the hidden and output nodes via pure evolution, some patterns in the evolved activation functions can be identified. The performance of the five neural networks is shown in Fig. 52 along with the distribution of the evolved functions. There is correlation between the number of patterns and the performance of the network, with larger hidden layers showing better performance. The results also suggest that an increase in the number of hidden nodes with Gaussian, bipolar sigmoid (or hyperbolic tangent) and Sinus functions improved the performance of the neural network, this coincides with a decrease in the number of hidden nodes with sigmoid functions.

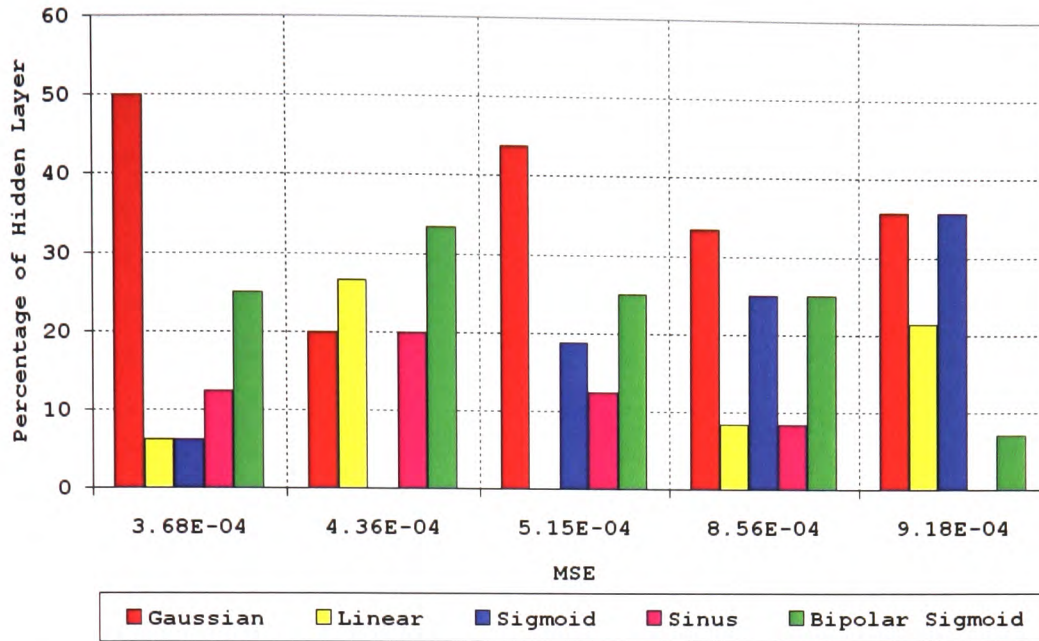


Fig. 52 - Graph detailing the comparative performance of activation functions in the hidden layers of the evolved networks.

Evolving the output layer seems to have less influence. A wide variety of activation function combinations are observed but there is no correlation between the activation functions and the performance of the evolved neural networks. Sinus and linear activation functions are most prevalent, few Gaussian or sigmoidal activation functions were evolved.

The dataset used in these experiments has four distinct regions (discussed in some detail in section 5.1.5) and provide a clear indication of the performance in each of these regions. The results (shown in Fig. 53) indicate that all three networks demonstrate similar performance on the inner constrained region (shown as the green region in Fig. 34) though the pure evolved activation function network is least capable. In the second region inside the constraint but adjacent to the boundary (shown in red in Fig. 34) Sigmoid linear and spline linear networks display similar performance, and purely evolved neural network give lower error. These regions are within the constraint boundary and so the corrective response in both cases is the identity quaternion.

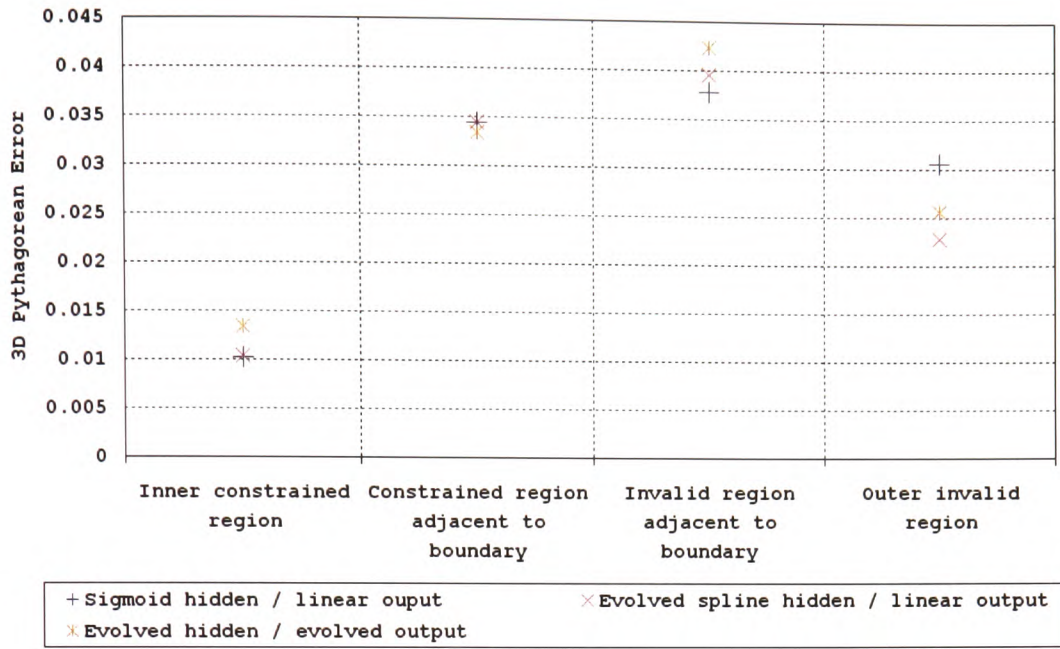


Fig. 53 - The average Pythagorean per region of the dataset.

The invalid regions where a more complex mapping exists i.e. from the invalid rotation to the correction produced more interesting results. The sigmoid-linear neural network out performs the purely evolved activation function neural network by 0.044 and the spline activation function neural by 0.028 in the region adjacent to the boundary on the invalid side (the region depicted in pink in Fig. 34.). The evolved spline neural network outperforms the evolved activation function neural network by 0.027 and the sigmoid-linear neural network by 0.049 in the invalid region furthest from the boundary (as shown in Fig. 53.) In general the spline activation function gives very similar results to the sigmoid-linear but out performs it in the outer region (Fig. 53). Despite this fewer nodes are required as shown in TABLE XIII.

The comparison of sigmoid and (evolved template based) cubic spline activation functions in the hidden layer (as shown in Fig. 54,) indicates that in several cases networks with a cubic spline hidden layer were out performed (on average) by their sigmoid based counterparts. These include the spherical boundary in three-dimensional space (a difference of $1.61\text{E-}04$) and both quaternion examples (a difference of $7.59\text{E-}05$ for the regular boundary and $1.52\text{E-}04$ for the irregular boundary).

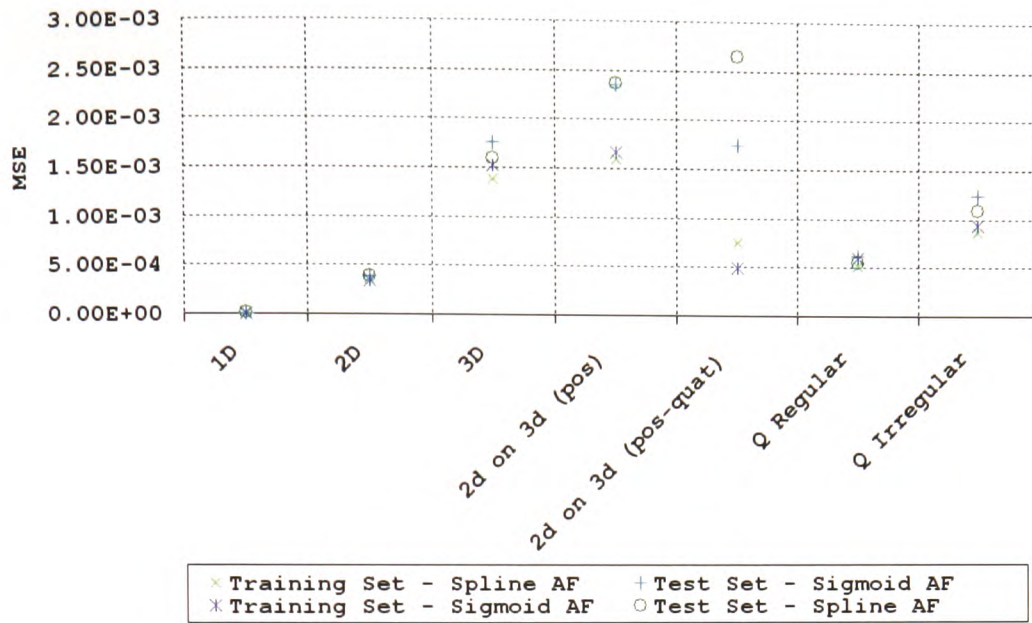


Fig. 54 – A comparison of evolved spline and sigmoid activation functions in the hidden layer.

The manifestation of this improvement in performance can be clearly identified by observing the error around the boundary, paying close attention to the region that corrects to the discontinuity. This is shown in Fig. 55.

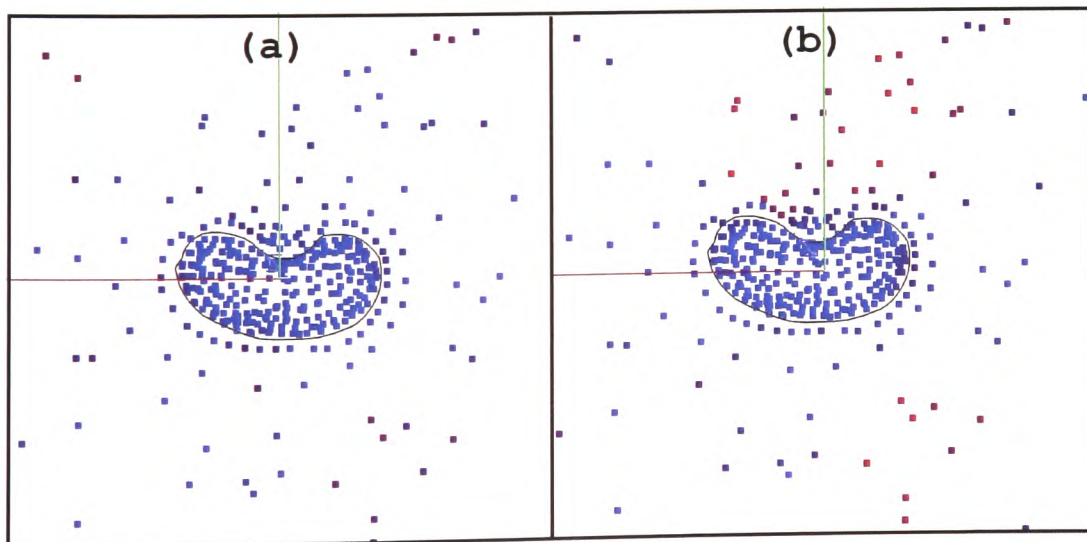


Fig. 55 - The figure on the left (a) represents the spline activation function neural network while that on the right (b) represents the sigmoid linear neural network. The same datasets were used in each case the reduction in the number of red points indicates that a significant improvement has taken place in the area around the discontinuity.

Comparing the neural networks evolved by the genetic algorithms it is noted that the number of hidden nodes in the cubic spline activation function network has in each case fewer nodes than its sigmoid-linear counterpart, as shown in Fig. 56.

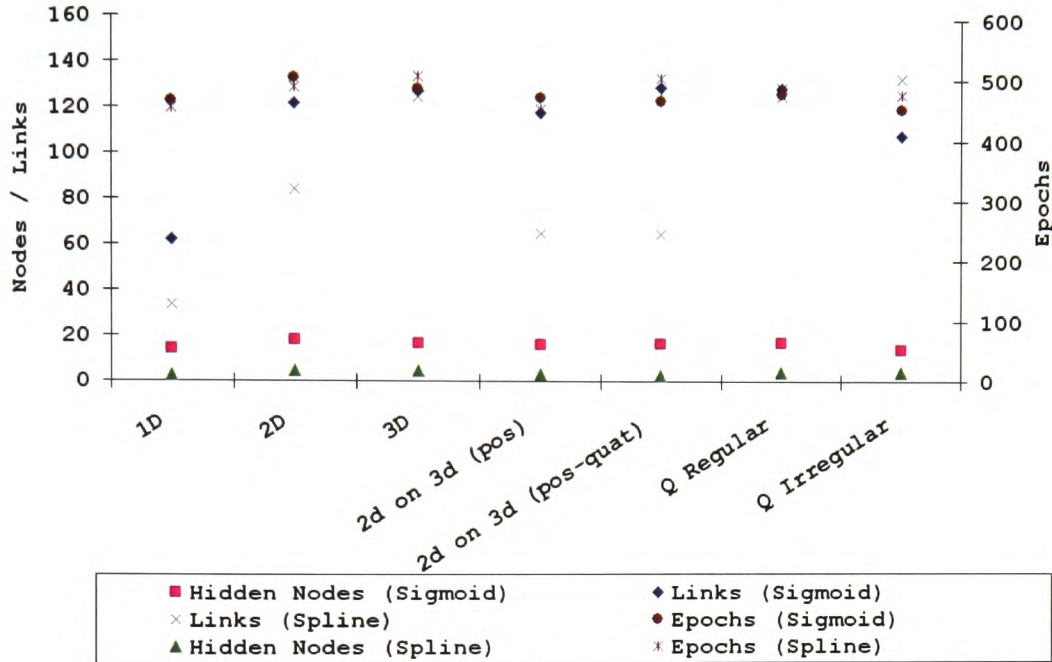


Fig. 56 – A graph showing the effect of activation function variation on the topologies and training requirements of the evolved neural networks.

5.3 Discussion

5.3.1 Discontinuous Vector Fields Representing Three-Dimensional Constraints

The results of varying the maximum number of hidden nodes evolved by the genetic algorithm indicate that increasing the size of the neural network increased performance on the training and test set until the maximum number of evolved hidden nodes reached around sixty. Here the neural network began to ‘over train’. Mehrotra, Mohan, and Ranker describe this as the neural network ‘memorizing’ the test set after which it is unable to generalize when faced with fresh examples [49].

The increase in the number of hidden nodes leads an increase in the number of links. This is to be expected as the number of links is related to the number of hidden nodes. There is also an increase in the number of training epochs evolved. This indicates that as the number of hidden nodes increased the amount of training required increased this confirms a connection between these two factors highlighted by Huber, Mayer and Schwaiger [81].

Varying the number of epochs evolved by the genetic algorithm demonstrates that increasing the training epochs also leads to an increase in performance. The shape of the plot (shown in Fig. 49) indicates that the gradient of performance decreases as the number of epochs increases. It can be inferred from this that at some point increasing the size of the network will no longer lead to a productive gain in performance. The network would over train giving further increases in performance on the training set only indicating a reduction in the neural networks generalisation ability. A trade off exists between the time taken to training the network (greatly affected by the number of epochs evolved) and the network performance.

Increasing the maximum number of epochs that the genetic algorithm can evolve leads to an increase in the number of epochs evolved in each case. This indicates that additional epochs would yield networks with higher performance. There is also a corresponding increase in the numbers of nodes and links, indicating that bigger networks were not previously evolved due to a lack of epochs to train them. This confirms the earlier results indicating a link between the number of hidden nodes and the volume of training required.

Increasing the number of generations increases the length of the genetic algorithms search and with each additional generation it approaches a maximum fitness, (the best network it can find to suit the problem.) In support of this the results show that the as the number of generations increases the performance of the neural networks evolved on both the test and training sets is increased. The increase in performance however is less each time indicating that a saturation point exists where an increase in the number of generations will have little effect on the performance.

An increase in the number of links, hidden nodes and training epochs evolved is observed as the number of generations over which the neural network is evolved

increases. Evidently as the evolutionary process continues the networks evolve towards the evolutionary constraints and an improvement in performance. These results reinforce those discussed earlier regarding the number of hidden nodes, and epochs.

An increase in the number of training patterns provided to train the three-dimensional discontinuous vector field representing a spherical constraint in three-dimensional space appears to improve the results in terms of their average MSE. As the number of training patterns was increased the improvement in performance decreased indicating a saturation point. This was accompanied by an attenuated increase in the number of training epochs required.

Increasing the size of the population with a fixed number of generations appears to improve neural network performance up to a saturation point. At this point the number of generations becomes the limiting factor and an increase in population size no longer produces an increase in performance. A large population requires more generations to evolve but produces better neural networks, as it does not suffer from allele loss. Increasing the size of the population increases the performance as allele loss is reduced while there are sufficient generations to evolve the population.

5.3.2 Discontinuous Vector Fields Describing Regular and Irregular Boundary Quaternion Constraints

Investigating performance issues for these vector fields representing quaternion based joint constraints with complex boundaries provide an opportunity to assess the performance increases and to compare the regular and irregular boundaries based on the complexities of the neural network evolved to learn them.

Increasing the maximum number of hidden nodes evolved seems to improve both the training set and the test set, indicating that the point where over training occurs has not been reached and further performance increases may be possible. A significant difference in the average MSE between the regular and irregular boundaries is observed and that this distance is maintained, further demonstrating that the irregular boundary is more complex than its regular counterpart.

An increase in the maximum number of evolved hidden nodes, leads to a proportional increase in the number of links, accompanied by an increase in the number of evolved training epochs required. This confirms that an increase in nodes results in an increased requirement for neural network training epochs.

Increasing the number of training epochs increases the performance, as for three-dimensional vector based constraints. The gradient of the improvement decreases, indicating a saturation point at this point the network will over train that is, the performance on the training set will continue to increase while the performance on the test set will decrease as the neural network loses its ability to generalise. The results for the four-dimensional vector fields representing the regular and irregular quaternion boundaries remain an almost identical distance apart, indicating that the improvement to both networks is comparable. There is little or no effect on the size of the networks evolved as observed for the three-dimensional vector fields representing spherical constraints.

The effect of the number of training patterns appears to be different for discontinuous vector fields representing regular and irregular quaternion rotational boundaries. For irregular boundaries the results are similar to the vector fields representing spherical constraints described earlier in this chapter. There appear to be some anomalies in the results attributed to averaging multiple neural networks results. In the irregular boundary case the number of hidden nodes and links increase as the number of patterns increases. Increasing the number of patterns increases the complexity of the constraint, due to the irregularity of the boundary. As more patterns are added the boundary is more clearly defined and the genetic algorithm evolves networks capable of dealing with this higher level of complexity.

In the case of regular boundaries there is no significant performance improvement as the number of patterns increases and there is no increase in the number of hidden nodes, links or epochs. The number of patterns does not affect the complexity in this case.

The performance of the neural network was significantly affected by the order of the training patterns, attributed to the global learning properties of the multi-layer feed-forward perceptron. Presenting the invalid patterns first led to poor performance, indicating that the valid patterns affected the learning that had taken place for invalid

patterns. Similar results are observed in the random case for the same reason. The performance is greatly improved when the valid patterns are presented first, indicating that the valid patterns have a disruptive effect on the learned invalid patterns but not vice versa. Disruption of previously learned patterns to this extent is termed 'catastrophic interference', it is a radical manifestation of a more general problem termed the 'plasticity-stability' problem [111]. This is summarised by French [111] as being the problem of designing "a system that is simultaneously sensitive to, but not radically disrupted by, new input." A reduction in the overlap of internal distributed representations reduces the extent of catastrophic interference [111]. As the back propagation-learning algorithm updates all nodes, not just those associated with the erroneous response it appears the magnitude of the updates required in the case of the valid inputs are sufficient to erase previously learned patterns but not vice versa.

In distributing the training patterns it was discovered that an even distribution gives a much better result than clustering the points at the boundary, which led to sparse regions away from the clusters. The evenly distributed data set provides clear representation of the mapping as a whole without the focus on individual sections, improving the generalisation of the neural networks evolved. This indicates that the high concentration of patterns depicting the local features had an effect on the global learning of the network.

Experiments investigating the effect of neuron activation function on neural network training show that pure evolution of the activation function can provide distinct advantages over neural networks with fixed sigmoid activation functions. Experiments show that in some regions the pure evolution neural network is better suited than the other approaches investigated (Fig. 53), though overall its MSE is higher than that of the evolved sigmoid hidden layer neural networks.

The activation functions evolved in the pure evolution approach are interesting in themselves some correlation can be identified between the functions evolved and the result. Results improve with the inclusion of both Gaussian and bipolar sigmoid (hyperbolic tangent) functions, researchers have found mixed activation function networks with two layers composed of these functions to perform well [53, 65, 83, 85]. Researchers have used the hyperbolic tangent, or bi-polar sigmoid function in the field of function approximation [54, 55, 57].

Shibata [85] used a mixture of Gaussian and Sigmoid activation functions and found the Gaussian function added a local learning component to the global learning of the feed forward sigmoid neural network. The results show that evolution tends towards an increase in Gaussian activation functions to improve performance, though an increase in local learning cannot be identified. This is possibly due to the increased number of free variables the genetic algorithm needs to optimise an increase in generations may provide this improvement.

The results of the evolved spline activation function experiments demonstrate that on average, template based evolved cubic spline activation functions offered some small improvements over their sigmoid hidden layered counterparts. This seems to be reversed when error is very high (when the mapping is at its most complex). In isolation this slight improvement in performance means very little, however combined with a lower network size for each of the spline networks this result becomes significant.

The regular boundary results with varied activation functions (shown in Fig. 53,) indicate that evolving cubic spline activation functions reduces the three-dimensional error in some regions but not in others. This may be an indication of better local learning in this region, however in the regions where local learning should have the largest influence (those regions closest to the boundary) the results do not support this.

In earlier results regarding irregular boundaries it was found that performance in the concave section was particularly low these were attributed in part to the poor local learning capabilities of the sigmoid activation function neural network. Spline activation function neural networks have been shown to have better local learning properties than their sigmoid-based counterparts [63, 64, 81, 90, 93-95]. The results support this with the evolved spline activation function neural network providing an improvement in the neural network performance reflected both by the MSE (Fig. 54) and the Pythagorean error (visualised as coloured points in Fig. 55).

Several researchers have identified improvements in performance when using spline activation functions, however there seems to be some disagreement on the origin of this performance increase. A number of authors [84, 87, 97] indicate that the adaptive spline

activation functions provide improved local learning as in the case of mixed activation function neural networks such as the gauss-sigmoid neural network [85].

Huber, Mayer and Schwaiger [63, 64] found that using spline activation functions reduced network complexity and increased performance for simple examples. They attribute this to a shift in complexity from the neural network (number of hidden nodes and links) to the activation functions of the hidden layer. In this case where the size of the hidden layer is less than optimal (due to the constraints imposed to reduce training times) the complex hidden layer neurons of the spline activation function neural networks give them a slight advantage over the sigmoid linear neural network.

5.4 Conclusions

Significant improvements can be made by removing the constraints placed on the evolution and training of the networks to minimise training times. The effect of each of these parameters on the evolved neural network is dictated by the complexity of the mapping.

The neural network and genetic algorithm performance was improved towards some maximum by increasing the duration of each, i.e. the number of epochs the neural networks were trained for and the number of generations over which the neural networks were evolved. This was also the case with regards to the population size - a larger population with more diverse individuals when provided with sufficient generations over which to evolve produced an improvement in the results.

In each experiment the number of hidden nodes tended towards the maximum, despite the secondary fitness function (fewer links) attempting to limit this rise. This seems to indicate that many more neurons are needed to obtain the maximum performance from evolved neural networks with sigmoid activation functions in their hidden neurons. Increases in the number of hidden layer neurons were accompanied by increases in the number of training epochs, indicating that an increased number of weight updates were required to sufficiently train the additional neurons.

Increasing the number of training patterns in most cases improves the results by providing clarification of the mapping. The network complexity and training requirement (evolved training epochs) did not increase, demonstrating that the mapping complexity was not changed. In the case of irregular boundaries, clarification of the mapping led to an increased complexity as the irregularities of the boundary became more clearly defined.

The pattern order selected for use in earlier experiments - valid patterns followed by invalid patterns, out performs both a random and an invalid patterns first approach. Learning one region has a disruptive effect on the other this is caused by the single set of weights used within the network and is described as catastrophic interference [111]. It was found that learning the invalid patterns after the valid patterns is less disruptive than learning the invalid first or the patterns in random order, this suggests that this arrangement of patterns produces a reduction in the overlap of internal distributed representations reduces the extent of catastrophic interference thus improving performance as suggested by French [111].

This chapter also shows that improvements can be made by using evolving neuron activation functions to suit the purpose. What the results do not clarify is the source of these improvements. Does the specialisation of the neurons improve the local learning [84, 87, 97] or the scope for additional complexity created when some of the network complexity is transferred to the activation functions [63, 64, 81]? Further investigation is needed to explore this question.

In a practical context a neural networks with smaller computational cost can be trained with the benefit of good generalization [81] an important consideration in any practical application. Further improvements could be made by the use of a gating network as used in the work of several other researchers [136-140]. Here the approximation of discontinuous functions is achieved by a number of continuous approximations separated at the discontinuities. However to achieve this an appropriate expert is required to differentiate between valid and invalid regions. Such an expert may be useful in improving the performance of the neural networks by limiting their application to invalid constraints. The following chapter will investigate the training of neural networks to group rotations as valid or invalid.

6. Binary Constraints in S^3 Space

Though the focus of this work lies in creating corrective constraints, binary constraints are implemented by a number of authors, either alone [6] or as a precursor to corrective constraints [2, 4, 5, 36]. These are simple equality type constraints in the case of Euler angles and more complex point in polygon tests in the case of two and three-dimensional polygon representations (for example [36]). Lee [6] implemented a set of simple binary quaternion constraints which could easily be combined into more complex constraints. Herda [4, 5] and Johnson [2] implemented binary quaternion constraints as a precursor to corrective constraints. These approaches all required the projection of the unit quaternions to a lower dimensional space, requiring additional processing and introducing singularities.

From a neural network point of view the mapping of an invalid constraint to a valid one can be considered as the learning of a discontinuous vector field. The problem of identifying valid and invalid constraints may be considered a classification problem. There are many machine learning techniques capable of solving multi-dimensional classification problems [49, 141]. Binary constraints in a number of dimensions were implemented using a Support Vector Machine (SVM) neural network.

6.1 Methodology

A number of experiments were designed to evaluate the use of SVMs for configuration classification. Initial experiments concerned the classification of one, two and three-dimensional vectors representing constraints of various sizes. With each increase in spatial dimensionality there was an accompanying increase in the complexity of the constraint, one-dimensional equality constraints were followed with circular constraints then spherical constraint. Encouraged by the results of these more complex quaternion based constraints, limited to regular shaped constraint boundaries similar to those of Lee [6] were attempted.

Experiments were also carried out towards finding the most appropriate kernel function and improving on the results of the quaternion-based experiments by increasing the number of patterns used in training.

6.1.1 Dataset Generation

The dataset generation software created to model corrective constraints was modified in each case such that it produced a single binary output indicating which of the two groups (valid or invalid) the input data represented. The dataset creation processes for one, two and three-dimensional constraints are detailed in sections 3.1 and those for the quaternion based constraints in section 4.1.

Two datasets were created for each experiment, the training set was used to train the SVM and the test set provided measurement of the SVMs generalisation capabilities. SVMLight provides a plethora of statistics regarding its performance on each test set and the success of the training.

6.1.2 SVMLight

SVMLight is a state of the art SVM implementation, it is based on the original ideas of Cotes and Vapnic [73] refined in conjunction with other researchers [66, 67, 71, 73]. The SVM methodology was discussed in section 2.3.1, SVMLight implements these principles with extensions to improve computation efficiency.

These improvements allow the use of a larger set of training patterns which would otherwise be limited by the size of the matrix containing the training patterns and more conventional computational improvements such as caching [72]. The development of SVMLight is discussed in detail by Joachims [72] whose training algorithm;

- Decomposes the training set into manageable chunks avoiding the problems of memory allocation with large training sets.
- Successively reduces the size of the training set, by removing those patterns most unlikely to become support vectors.

- Implements computational improvements such as caching.

SVMLight was developed by T. Joachims who has published substantial material on improving the performance of the SVM [72]. He has very kindly made his work available for non-commercial purposes.

6.1.3 Training Configuration

Unlike the earlier experiments with neural networks and genetic algorithms there is no random component in the training process and so two SVMs trained with the same training set will give the same result hence there is no requirement to repeat the results to obtain an average.

SVMLight provides several different kernel functions including linear, polynomial, sigmoid and radial basis (Gaussian). Earlier research discussed the power of different activation functions in a neural network setting this is also true in terms of kernel functions for SVMs [71].

6.2 Results

The results show that the SVM is able to classify the joint configurations as valid or invalid to a high degree of accuracy. In the case of one-dimensional constraints shown in Fig. 57 it is noted that for small constraints the linear kernel out performed the other kernel types this trend however was reversed above sixty degrees. The results for polynomial, radial basis, and sigmoid kernel types were very similar though where the results differ significantly the sigmoid kernel function appears to correctly classify the highest percentage of patterns.

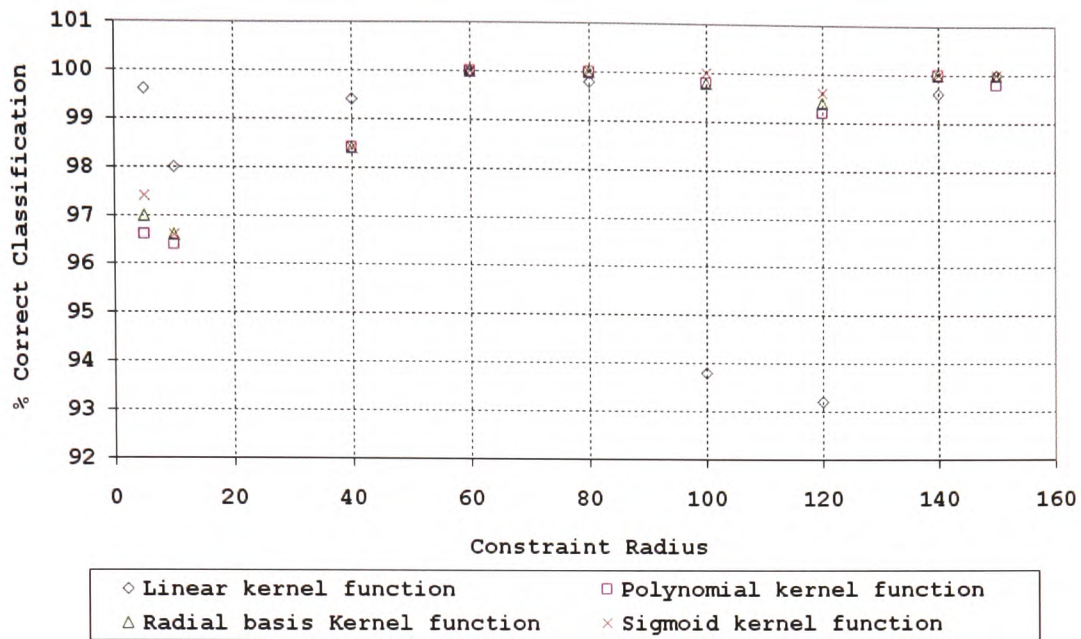


Fig. 57 - Performance of the SVM on a one-dimensional constraint.

The performance of the SVM decreases between one-dimensional and two-dimensional vector based constraints. However the pattern of results remains the same, there is a steady increase in performance as the size of the constraint increases for the linear, sigmoid and polynomial kernels. The opposite is true in the case of the linear kernel where performance decreases as the size of the constraint increases. The performance of the polynomial and radial basis kernel functions are very similar to each other perhaps indicating some advantage provided by a common aspect of their shape. These results are shown in Fig. 58.

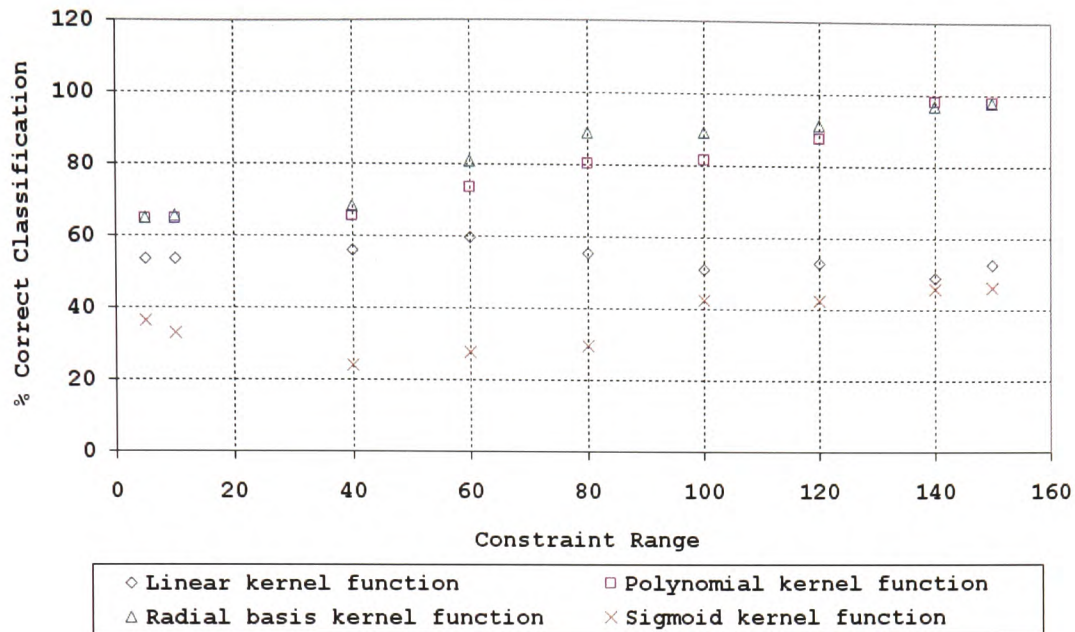


Fig. 58 - Performance of the SVM on a two-dimensional (spherical) constraint.

Moving to three dimensions there is a further decrease in the performance of the SVM compared to two and three-dimensional constraints. The polynomial and radial basis kernel functions again out perform the sigmoid and linear kernel types. Though for all kernel functions the results have decreased compared to earlier results as shown in Fig. 59. For the polynomial and radial basis kernel functions an increase in performance is observed as the size of the constraint region increases. However for linear and sigmoid kernel functions there are less definite variations, though there are indications of an overall increase in the sigmoid performance with range and a decrease in the linear kernel function performance.

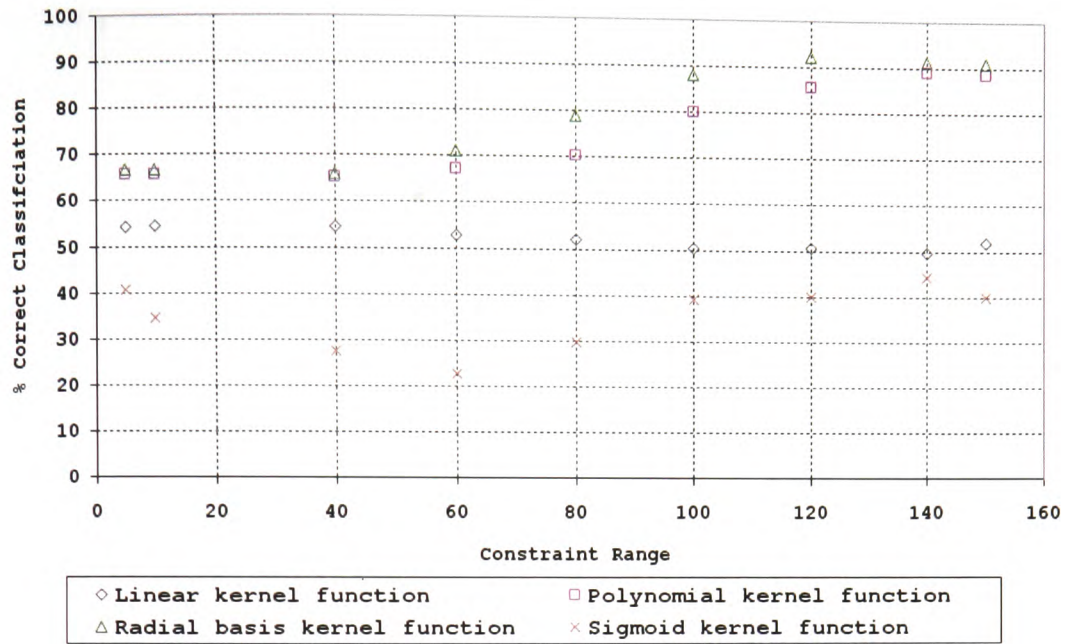


Fig. 59 - Performance of the SVM on a 3D constraint

Moving to more complex quaternion based constraints an improvement is observed in performance, a similar improvement is observed in neural networks trained for corrective constraints between constraints of the same complexity. The performance for the quaternion-based constraint (a two dimensional regular boundary on the surface of a three dimensional sphere, described using quaternion) demonstrates a significant improvement in performance. The sigmoid kernel function does not perform very well with a maximum correct classification of less than 60%. The results for linear, polynomial and radial basis kernel functions demonstrate much better performance and all follow a similar pattern. Their results are almost symmetrical around a ninety degrees radius as the constraint covers half the sphere hence the size of the regions is equal. These results are shown in Fig. 60.

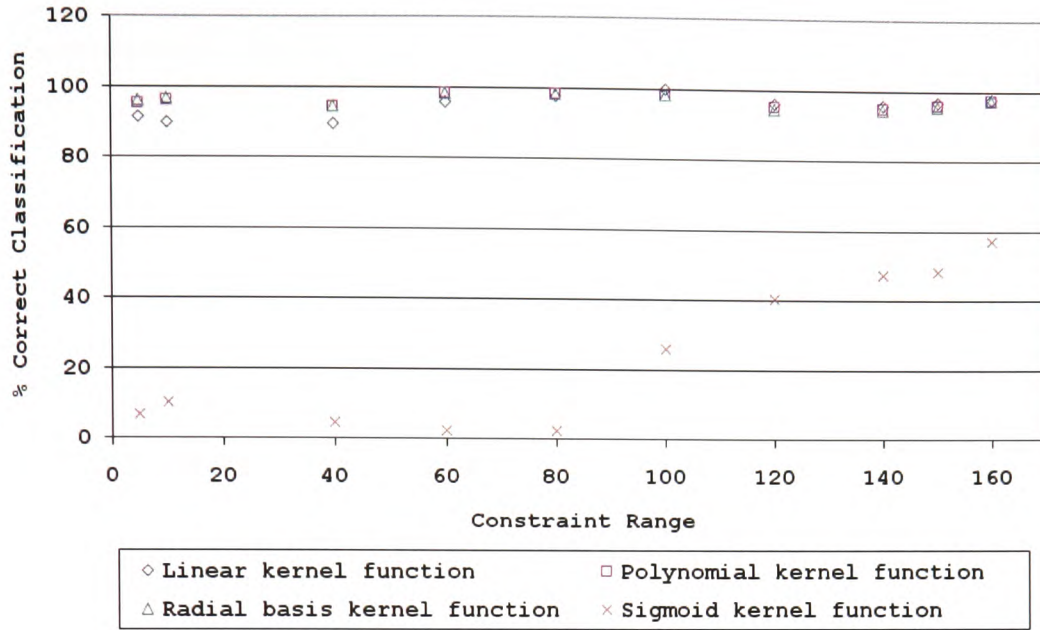


Fig. 60 - Performance of the SVM on a quaternion based constraint with a regular boundary.

Attempts were made to improve on these results by increasing the number of patterns used in training. A constraint of twenty degrees radius was selected as the results showed scope for improvement. The results show that increasing the number of training patterns does make some improvement in the case of the linear, polynomial and radial basis kernel functions, but not however in the case of the sigmoid function performance decreases as the number of patterns increases. The increase in these results seems to attenuate as the number of patterns increases hence the benefits of increasing the number of patterns are negligible above a threshold.

6.3 Discussion

The results show that SVMs are capable of classifying valid and invalid vectors in a vector field in one, two and three dimensions and indicate an increase in problem complexity (by their decrease in performance,) as the dimensionality of the constraint and the problem space increase.

Polynomial and radial basis kernel functions perform consistently well for each however there is a decrease in their performance as the number of dimensions and

therefore complexity increases. Sigmoid kernel functions perform poorly for all but the one-dimensional constraint and the performance of the linear kernel function (effectively not using a kernel transform) decreases in its effectiveness as the number of dimensions increases.

Considered in terms of the distribution of the points in their respective problem space, it is clear why the performance of the linear separator decreases, moving to two and then three dimensions the boundaries formed are circular and spherical respectively and therefore better separated when moved into a higher dimensional space.

The local learning capabilities exhibited by neural networks which used Gaussian or polynomial activation functions is well established in the literature [85, 96], and likewise the poor local learning exhibited by feed forward neural networks with sigmoid activation functions [85]. The limitations of this kernel type have been acknowledged, though due to their global learning capabilities in neural networks their development and inclusion in SVMs is an open issue [142].

There is a definite contrast between the results of the one, two and three-dimensional constraints and those of the quaternion based constraints. Though there are only two dimensions related by the constraint several additional constraints are required by the quaternion representation. It appears however that the neural network finds it easier to classify the quaternion constraints than any of the previous constraints with fewer dimensions. This indicates that the increase in dimensionality of initial mapping provided better results when moved to a higher dimensional space by the kernel functions.

There is a strong indication that the density or distribution of data changes significantly between three and four dimensions giving a decrease in classification error. Current results do not provide a basis for quantifying these factors and this may be considered in future work.

With regards to the performance of the SVM it was found that the polynomial and RBF kernel types were superior and provided the best results in all cases. It was also found that the effect of increasing the number of patterns improved classification results

though there were no significant gains after two thousand patterns. This is the result of increasing the number of support vectors present in the dataset.

6.4 Conclusions

In conclusion SVMs are capable of implementing constraints in one, two and three dimensions to a reasonable degree of accuracy. More importantly they can classify valid and invalid quaternion based orientation constraints like those suggested by Lee [6] to a very high degree of accuracy. Unlike the approach of Lee no decomposition or reformatting of the quaternion representing the joints rotation is required. SVMs provide a significant advantage in that they can be created based on subject data rather than being a combination of abstract shapes.

Furthermore it can be concluded that the quaternion rotational representation offers further advantages in this case as due to its higher dimensionality the SVM is more successful in defining a boundary between valid and invalid points in the high dimensional kernel space.

7. Discussion

This work has focused on neural network learning of discontinuous multidimensional vector fields applicable to the implementation of corrective angular constraints to simulate anatomical joints. Initially constraints represented by vector fields in one, two and three dimensions were studied as a precursor to quaternion representations. An initial investigation was also carried out into the neural network classification of valid and invalid vectors representing joint configurations, building from low dimensional representations to those in quaternion space.

7.1 Binary Constraints

It has been found that Support Vector Machines (SVMs,) are capable of classifying valid and invalid vectors in a vector field in one, two and three dimensions. The results here indicate an increasing complexity as the number of related dimensions in the constraint increase.

As the complexity of the constraints increases in terms of the related dimensions that describe the constrained region, it was found that the suitability of the tested SVM transfer functions changes. One-dimensional constraints can be separated using a support vector machine with linear kernel functions, though these kernels are incapable of separating more complex constraints in two and three dimensions. Radial Basis (Gaussian) and polynomial activation functions have shown the best performance indicating that applying these kernel transformations makes the data easier to separate in the more complex cases.

The Support Vector Machine classifies the quaternion based binary constraints more accurately than constraints with fewer dimensions. This indicates that the higher dimensionality of the quaternion representation provides better results when moved to a higher dimensional space by the kernel functions. This improvement may indicate that the density or distribution of data changes significantly between three and four

dimensions giving a decrease in classification error. The quantification of these factors is a subject for further study.

The performance of the SVM was improved by increasing the number of patterns though no significant increases were observed after two thousand patterns. This is the result of increasing the number of support vectors present in the dataset, which in turn improves the positioning of the separating function.

7.2 Corrective Constraints

Corrective constraints in all cases involve genetically evolved neural networks learning a discontinuous vector field. In initial experiments simple one, two and three-dimensional discontinuous vector fields representing constraints with continuous boundaries were trained. A discussion of these simple cases follows moving towards quaternion-based constraints.

Each increase in the dimensionality of the discontinuous vector field results in the inclusion of additional relationships between dimensions these define both the vector field and implicitly the discontinuity. The neural network requires sufficient patterns to learn both the vector field and the discontinuity. As increases in the dimensionality of the vector field and constraint were not matched by increases the number of training patterns there are fewer patterns present representing each relationship and consequently the neural network performance decreases, as shown in Fig. 9. Experiments undertaken to improve the performance of the neural network showed that increasing the number of patterns increased performance (Fig. 40).

Monitoring the evolution of the networks formed in each case showed that as the vector field and discontinuity increased in complexity more complex networks were required to maintain performance. This increase is less pronounced between results for discontinuous vector fields representing constraints in two and three-dimensional space with constraints of equal dimensionality to the problem space, as shown in Fig. 9. This is attributed to the constraints imposed on the maximum hidden nodes evolved (imposed to restrict the temporal cost of experiments.) Removing the hidden node constraint

where more complex networks were required resulted in an increase in performance Fig. 36.

The decrease in performance in relation to the increasing size of the discontinuous region (representing the constrained region,) can be attributed to the distribution of training patterns. Large valid regions are easily learned as the majority of inputs map to a single vector however, the reduction in exposure to patterns outside this region (where the input vector is mapped to a correction) reduces neural network exposure to complex inter-relationships between vector elements and between input and correction vectors. The technique is applicable to various sizes of constrained region and that despite the increase in error with the size of the constraint as the overall MSE of the results is low, the results are shown Fig. 9.

The evolution and training parameters of the neural networks were initially limited and these limitations were found to affect performance and became the subject of further investigation (detailed in Chapter 5). Several parameters with a direct effect on performance were identified. Increasing the number of generations allowed further evolution and improving results (in Fig. 43.) Several limiting factors such as the number of nodes and epochs evolved further towards their constrained maximums.

Increasing the limit on the number of hidden nodes increased performance. Resulting in an increase in the number of training epochs required these additional epochs being required to refine the additional nodes. Increasing the number of training patterns and training epochs improved the neural networks performance though the scale of the improvements decreased. Extending the adaptive processes, i.e. evolution via the number of generations and neural network training via the number of epochs leads to an increase in performance. Increasing the population size produces a steady increase in performance due to the reduction of allele loss until the number of generations required to evolve the population became a limiting factor.

Having considered the training of less complicated vector fields (those with fewer dimensions,) focus moves towards vector fields representing regular two-dimensional boundaries on the surface of a unit sphere. Here vectors represent both the initial position and correction. A high rate of correct approximation was observed for a range of constraint radii (as shown in Fig. 9). This technique can be used to implement simple

angular constraints similar to spherical polygon [36] and cone based [41] constraints considered in the literature review (Chapter 2).

The error recorded for these neural networks is higher than that of the circular boundary in two-dimensional space and that of the spherical boundary in three-dimensional space. Despite the dimensionality of the input space being the same as that of the output space the constraint itself is more complex. This confirms that the complexity of the vector field discontinuity has an effect on the neural networks performance. It is also the case that an increase in dimensionality of the problem space has an effect on performance for networks of limited size as indicated by Grzeszczuk, Terzopoulos and Hinton [56].

The experiments for vector fields representing a circular constraint on the surface of a sphere were extended such that the corrective component was no longer a vector but a quaternion representing the required corrective rotation. A decrease in MSE is observed when the output component is encoded as a quaternion and the error is more consistent (as shown in Fig. 9.) Examination of the three-dimensional Pythagorean error (as shown in Fig. 11,) identifies isolated patterns of high error. These errors can be attributed to a lack of test data in the region of an additional discontinuity by visualising them in the context of the training and validation patterns.

This additional discontinuity is opposite the valid region where points are equally close to opposite sides of the spherical boundary. To simplify future discussion discontinuity between the valid and invalid region is described as the *boundary discontinuity* and the discontinuity in the region opposite the valid region equidistant to two positions on the boundary as the *correctional discontinuity*. The correctional discontinuity results from the corrections in three-dimensional space and is implied in quaternion space like the *boundary discontinuity*. The true effect of this discontinuity is difficult to judge from the results presented as the metrics used (MSE and Pythagorean distance between corrected virtual limbs) both measure against the test set and not the proximity to the boundary of the corrected orientation.

This explains why the high error patterns are isolated each is similar to neighbouring patterns corrected to the other side of the sphere. The neural network successfully corrects the point to the boundary, however, as this is not the boundary indicated by the test set a large error is reported.

Orientations effected by the correctional discontinuity are some distance from the boundary, in a practical application it is unlikely that the limb would reach these extremes before being corrected. Improvement in neural network training may partially eliminate these errors and the creation of an error metric that recorded error in relation to the distance of the virtual limb from the boundary would give a more accurate representation of network error. The networks created here could be used to implement corrective constraints for some of the spherical polygon and cone boundaries described by earlier researchers [36, 40, 41].

Continuing with the two-dimensional constraint boundary on the surface of a unit sphere quaternions are used to represent both current rotation of the limb and the required correction. Though the complexity of the discontinuity represented remains constant the dimensionality of the vector field increases. A direct comparison of the MSE of the discontinuous vector fields in S^2 with those in S^3 (quaternion) is meaningless as the latter represent rotational and the former proximal error. An increase in the MSE of the neural network is observed as the complexity of the vector field increases.

Neural networks can be successfully evolved and trained to learn discontinuous vector fields in quaternion space, which produce quaternion rotations to correct a given quaternion rotation to a regular constraint boundary. Thus implementing similar boundaries to those of Gyi *et al* [40], Korein [36], and Lee [6]. The approach introduced here does not require the dimensionality of the quaternion to be reduced unlike other joint constraint approaches [4, 5, 46]. Reducing the dimensionality of the quaternion representation incurs a similar penalty to converting between rotational formats for constraint and introduces singularities. Also unlike the approaches of Lee [6] and Liu and Prakash [3] there is no requirement to decompose the quaternion into quaternions representing planar rotations, this again incurs a similar penalty to converting between parameterisations.

Discontinuous vector fields trained to imply a constrained circular region of various radii produces positive results with average errors less than one percent using networks with less than twenty hidden nodes. An increase in error matched by an increase in network size indicates an increased complexity between constraint sizes of 45 and 135

degrees in radius (Fig. 18). PCA reveals that there are significant changes in the distribution of the quaternion in quaternion space that account for the increase for this angular range, the orientation of the principle components of the dataset change as shown in Fig. 18. These changes in the distribution of patterns in quaternion space may increase the overlap of internal distributed representations French suggests this can increase the extent of interference between patterns which inhibit learning [111].

The regions of the vector field representing constraint correction and the constrained region are continuous with a single discontinuity between them. This is with the exception of the vector field representing the circular constraint boundary on the surface of a unit sphere. Here a discontinuity is present in the region of the vector field representing the correction due the corrective discontinuity discussed earlier. This discontinuity is implied within the problem space and so the neural network reports a high error, as the network results do not match the ideal. Despite the high error reported by both MSE and three-dimensional metrics the corrections made by the neural network are to configurations close to the boundary in most cases.

As the complexity of the evolved networks increases with respect to the range, the standard deviation of the number of hidden nodes evolved decreases. This combined with the high average number of hidden nodes indicates all the networks evolved for these ranges were close to the constrained maximum. Limiting the number of hidden nodes in combination with the regularization function prevented an increase in complexity and contributed to the increase in error.

The axis chosen to mark the centre of the constraints has an effect on the results. PCA can be used to link this to the distribution of patterns in quaternion space this becomes more regular as the constraint centre is moved from the y-axis to x-axis and from x-axis to the z-axis. According to the Pythagorean metric a dataset with a more regular shape gives superior results as in the case of quaternion ambiguity. A possible reason for this is that the more regular datasets are better distributed in quaternion space French [111] suggests that increasing the distribution of patterns reduces interference.

The order in which the patterns were presented to the neural network provided an interesting insight into the learning process. The patterns were presented with valid patterns followed by invalid patterns, and results (TABLE X) have shown that this out

performs datasets with both a random and an invalid patterns first ordering. Learning one region clearly has a disruptive effect on the other this is caused by the single set of weights used within the network and is described as catastrophic interference [111]. Learning the invalid patterns after the valid patterns is less disruptive than learning the invalid first or the patterns in random order, this suggests that this arrangement of patterns produces a reduction in the overlap of internal distributed representations reduces the extent of catastrophic interference thus improving performance as suggested by French [111].

The Pythagorean error between the ideal correction and the neural network correction shows that the network performs well for all but a few patterns. Only $3.7 \times 10^{-4}\%$ of patterns demonstrated an average error greater than 2% of the maximum possible error (with an average neural network size of 16 hidden neurons, and an average of 471 training epochs). In practice where these patterns occupy regions a large distance from the constraint this is not critical as it is unlikely in kinematics systems that the joint would move far beyond the boundary between corrections. It is important to note that these plots (shown in Fig. 21) represent the average error over the networks, when observing the plots for all five separately individual neural network performances vary from region to region.

The Pythagorean error is generally highest at the boundary discontinuity and at the correctional discontinuity. Similar results were observed in the case of circular boundaries on the surface of a unit sphere described using vectors (shown in Fig. 11,) and that ambiguity is the cause of many of the problems associated with neural network training.

Sparse data offers an explanation for a number of individual high error results (highlighted in Fig. 24). However despite the high error the correction is to a configuration close to or inside the boundary, in practice iterative approaches could be used to improve these results. Sparse regions of data had a similar impact on performance in earlier experiments of lower dimensional order.

The results demonstrate that evolved neural networks of low complexity can be used to implicitly model simple spherical joint constraints similar to those modelled in other approaches [6]. However the evolved neural network constraints described here are able

to provide the necessary correction without reducing the dimensionality of the quaternion like contemporary approaches [2-5].

Having considered discontinuous vector fields representing regular boundaries (circular boundaries) on the surface of a unit sphere the discussion turns to constraints with irregular boundaries on the surface of a unit sphere. An increase in both neural network error and three-dimensional Pythagorean error is observed compared with regular boundaries. High error is once more observed between valid and invalid configurations (at the boundary discontinuity), opposite the boundary (at the correctional discontinuity,) and in areas of sparse training patterns.

High error recorded around the convex region of the irregular boundary is attributed to the complexity of the vector field in this region. In the centre of each concave region of the boundary is another vector field discontinuity as quaternions are proximally equidistant from valid configurations on either side, (these are referred to as *concave region discontinuities*.) As in the case of the correctional discontinuity the network may produce a correction that returns the limb to the boundary but not to the side of the boundary indicated by the test set, resulting in a high Pythagorean error.

Additional irregular boundaries demonstrate the capabilities of neural networks in learning discontinuous vector fields to represent anatomical boundaries. The error in three dimensions is low and there is a good correlation with the ideal corrections. An interesting limitation is identified, where the boundary of the discontinuity is concave or convex these local features are sometimes lost. This can be attributed to two factors, the first being pattern distribution. There is little difference in shape between the large (Fig. 32 (b)) and small (Fig. 32 (c)) boundaries yet a noticeable difference in performance is observed. This is attributed to the density of patterns, the smaller boundary has the same number of patterns within the constraint but confined to a smaller region.

The second factor that may affect the error at the discontinuous boundary is the learning method of the sigmoid-based neural network. This demonstrates good global learning that is it learns large general mappings well. It is however insensitive to local features such as the concave and convex regions of the boundary, this results in the learned boundary being an attenuated version of the original.

An interesting discovery was made regarding the vector field representation of quaternion rotational constraints. The regular boundary experiments (which generated data sets by converting a random rotation to a quaternion) generated all valid and the majority of invalid quaternion on one half of the quaternion hyper-sphere. There was no ambiguity in quaternion space as there was only one boundary to which all the quaternions were corrected.

There is ambiguity in the rotations represented by unit quaternions, in that the quaternion sphere represents 4π rotations. In an attempt to improve performance this ambiguity was removed by forcing the quaternion to one side of the quaternion hyper-sphere. The results deteriorated in performance for larger constraints, this is attributed to a change in the continuity of the continuous parts of the vector field and or the implied boundary in quaternion space. The increase in error is accompanied by a change in the distribution of data in quaternion space.

A comparable case in two dimensions can be visualised (Fig. 61), where a group of valid and invalid points lie across the centre of the region. If the size of this region is halved and the points projected to their equals on the opposite side the distribution of the data changes reflected the principle components. The divergence of results at a given constraint radius (reflected in Fig. 26,) may indicate that below a given radius the continuity of the regions is not affected. Above this radius a number of points key to the implicit representation of the boundary are moved, as in the two-dimensional case shown in Fig. 61.

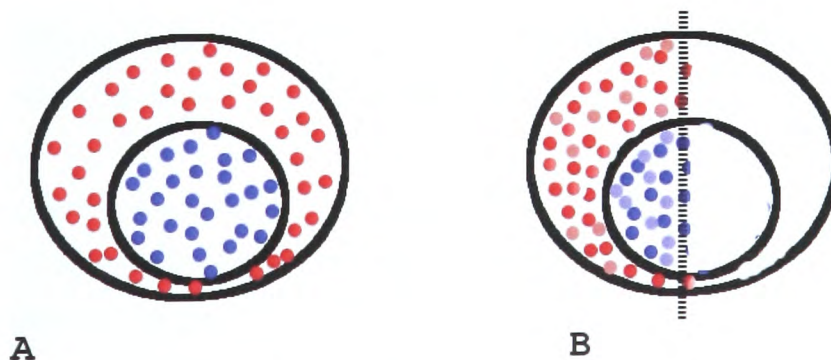


Fig. 61 – The diagram presents a simplified demonstration of the effect of moving patterns to one side of the quaternion hyper sphere. A shows the points distributed evenly, B shows the points forced to one side.

The irregular boundary experiments used a sampling dataset generator that generated a quaternion representing the rotation of a control limb being manipulated in three-dimensional space. The dataset generated is more widely distributed over the surface of the quaternion hyper-sphere. This caused ambiguity, as there were two valid regions on either side of the quaternion hyper-sphere. The quaternion space vector field now contained a second discontinuity, the network had to perform mapping of quaternions to the appropriate boundary on the quaternion hyper-sphere. Neural network training failed to produce any networks with acceptable performance for these datasets.

To overcome this, the valid dataset and the boundary used to calculate corrections were moved to one side of the quaternion hyper-sphere. The neural network is capable of learning the vector field which now contains a single discontinuity, between invalid and valid quaternion, that is correction and no correction.

The choice of constraint centre seems to affect the mean squared error and the actual error in three-dimensional space in different ways. It is clear from the principle component analysis that the distribution of patterns in quaternion space is more regular with the constraint centred on the y-axis than on the x-axis or z-axis. More regularly shaped dataset appear to give better results according to the three-dimensional metric.

Increasing the maximum number of hidden nodes evolved by the genetic algorithm produced an increase in performance. Though for discontinuous quaternion vector fields the experiments do not identify the point at which over-training occurred due to the time required to complete the experiments. Increasing the number of training epochs evolved also increased performance to a saturation point, with uniform increases in performance observed for both regular and irregular boundaries.

Regular boundaries were found to be insensitive to an increase or decrease in the number of patterns used. This indicates fewer patterns were sufficient for the neural network to learn the regular case and that in this case the number of patterns is not the limiting factor. In the case of irregular boundaries the increase in the definition of the boundary irregularities improved the neural networks learning of these complex structures. Earlier experiments with different shaped boundaries identified that the increasing density of patterns describing the implied boundary as being significant.

Global and local learning were briefly mentioned earlier in this chapter regarding the global learning nature of multi-layer perceptron type networks and its possible adverse effects on the results. As indicated in the literature survey (Chapter 2) several researchers have attempted to introduce local learning to the multi-layer perceptron via evolutionary techniques. In Chapter 5 two of these approaches are investigated, the evolution of static neural network functions in the hidden layer of a neural network (similar to [65, 86]) and the template based evolution of cubic spline activation function in hidden layer neurons (similar to [63]).

On average, template based evolved cubic-spline activation functions offered some small improvements over their sigmoid counterparts. This seems to be reversed when the vector field is at its most complex. In isolation this slight improvement in performance means very little, however combined with a reduction in network size this result becomes significant as smaller networks have better generalisation capabilities.

Mayer and Schwaiger [63, 64] found that using spline activation functions reduced network complexity and increased performance for simple examples. This is because the complexity shifts from the neural network (number of hidden nodes and links) to the activation functions of the hidden layer. In this case where the size of the hidden layer is less than optimal (due to the constraints imposed to reduce training times) the complex hidden layer neurons of the spline activation function neural networks give them a slight advantage over their competitors.

Indirectly measuring the quaternion error has proved very useful and provides an insight into the behaviour of this technique when applied to simple anatomical models. For regular boundaries errors in quaternion space are proportional to those in three-dimensional space. Principle Component Analysis (PCA) has also proved useful in determining the shape and orientation of datasets in quaternion space.

8. Conclusions

This thesis details an investigation into the use of Evolved Topology Neural Networks for anatomical constraints. The conclusion, and contribution to knowledge, is that evolved generalised multi-layer perceptrons are capable of modelling vector fields in quaternion space suitable implementation of correctional rotational constraints on a virtual limb. The main findings of this work that support this conclusion are outlined below...

- Evolved topology neural networks can model implicit boundaries (discontinuities) between continuous regions within vector fields in a number of dimensions, specifically one, two, three and four-dimensional quaternion space. In terms of vector fields suitable for the creation of corrective joint constraints it was found that a number of three-dimensional factors needed to be taken into consideration. An additional discontinuity was identified where a point was equally close to more than one point on the boundary. These must be taken into account when creating the datasets and evaluating neural network performance.
- The distribution of the training data in quaternion space had a significant effect on the performance of evolved neural networks in learning the discontinuous vector field. It was found that the implied boundary between continuous regions must be located on one side of the hyper sphere (despite the equality of its polar equivalent), to maintain continuity of the both valid and invalid regions. A number of factors concerning the dataset were found to influence neural network training these were primarily concerned with the distribution of the training set in quaternion space. Appropriately orientated evenly distributed datasets produce an improvement in performance this is attributed to a reduction in the overlap of internal distributed representations which reduces the extent of interference [111].
- The implicit boundaries between continuous regions can be applied to the representation of anatomical constraints this research focuses on rotational constraints concerning rotation of (but not along) a virtual limb in three-dimensional space. The technique has been shown to be successful for two-

dimensional constraints on a unit sphere represented by both vectors (representing the free end of a virtual limb) and quaternion (representing the orientation of a virtual limb). Constraints can be trained in quaternion space utilising the quaternion representation for both regular and irregular boundaries, an important consideration if anatomical boundaries are to be considered. The vector fields in these cases are similar despite an increase in the dimensionality of both the problem space and the constraint.

- The complexity of the quaternion vector field mapping is affected by the distribution (shape and orientation) of the dataset in quaternion space, resulting in fluctuations in the error recorded. Despite which, neural networks were trained (using limited training) such that error in three-dimensions is on average 0.99%, by neural networks with less than twenty hidden nodes. Representational ambiguity must be removed such that the neural network corrects to only one boundary from any point on the quaternion hyper sphere.
- Evolved topology neural networks were found to be capable of modelling constraints of equal dimensionality to the problem space, and of lower dimensionality. Increases in dimensionality result in an increase in the complexity of the network, if network complexity is limited the network error increases. Where the dimensionality of the constraint is lower than the dimensionality of the problem space, the error is significantly lower than when the constraint and problem dimensionality are the same.
- Evolved topology neural networks successfully trained a number of regular boundaries in quaternion space with different ranges the success of the training is dependent on the distribution of patterns in quaternion space. These results produced average errors of 0.99% in three-dimensions, with the highest error at 11.67% and the lowest at 0.0063%. Only $3.7 \times 10^{-4}\%$ of patterns resulted in errors greater than 2% with an average neural network size of 16 hidden nodes.
- A number of irregular boundaries were also investigated performance on these boundaries was dependent on the boundary shape. The presents of concave regions on the boundary introduced boundary correction ambiguities decreasing performance. These results produced average errors of 2.15% in three-

dimensions, with the highest error at 24.71% and the lowest at 0.017%. Only 0.032% of patterns resulted in errors greater than 7.5% with an average network size of 18.4 hidden nodes.

These findings have implications in a number of areas where joint constraints of high accuracy and performance are required. In animation a requirement for more sophisticated joint specific constraints was identified by Shao and Ng-Thow-Hing [1]. Such standard body constraints are however limited in scenarios where other factors affect a joints range of motion. For example, an animated character designed with heavy shoulder armour may be constrained to avoid contact with the armour during the animation process. Using present techniques an irregular boundary could be defined by an animator, training data could then be generated manually or automatically and the network trained. Unlike other approaches [3-6, 46] no prior processing of the quaternion is required at use time.

In terms of execution time it was found that in the one two and three-dimensional case the correction of the dataset generator (written in C) was faster than the Java based neural network. In the quaternion case the C based generator for the regular boundary in quaternion space was faster than the Java based neural network. Re-writing the regular boundary correction generator in Java made little difference to the execution time. The java based neural network was however significantly faster than the dataset generator for irregular boundaries. As most anatomical joints have non-spherical joint constraint limits this is the most significant.

The neural network manager used in NetJEN is BOONE (Basic Object Orientated Neural Evaluator) which was written and designed by August Mayer and is available under the GNU Public Licence (GPL). Viewing the code it is apparent that rather than firing the layers sequentially to obtain an output the network fires the nodes sequentially until there are no further changes to the activations of any of the nodes. Through correspondence with the author it was found that this method was selected over the more efficient alternative to allow BOONE the flexibility to deal with recurrent and cyclic neural networks.

In summary evolved generalised multi-layer perceptrons are capable of modelling discontinuous vector fields in quaternion space suitable for the correction of rotational

constraints on a virtual limb subject to a number of conditions concerning the distribution of patterns. Their ability to model irregular rotational boundaries gives them an advantage of approaches using coarse spherical approximations [6]. Their ability to utilise quaternions without pre-processing (conversion or dimensional reduction) and their potential for hardware [121, 143, 144] and vector based [145] implementations gives them the potential for performance increases over existing approaches.

9. Future Work

9.1 Introduction

Previous chapters have demonstrated the capabilities of evolved topology neural networks in modelling discontinuous vector fields suitable for the representation of joint constraints using a quaternion representation. In this section a number of possible extensions to the work undertaken are considered. These are intended to overcome limitations of the current study and look towards applying the constraints developed in other areas.

9.2 Development of the Current Work

9.2.1 Performance of Spline Based Neural Networks

In Chapter 5 improvements in performance were identified when using spline activation functions. This has been previously demonstrated by a number of researchers however there seems to be some disagreement on the origin of this performance increase.

Shen *et al* [87], Guarnieri, Piazza and Uncini [97], and Vecchi, Piazza and Uncini [84] indicate that the adaptive spline activation functions provide improved local learning as in the case of mixed activation function neural networks such as the gauss-sigmoid neural network [85].

Huber, Mayer and Schwaiger [63, 64] found that using template based spline activation functions reduced network complexity and increased performance for simple examples. They attribute this to a shift in complexity from the neural network (number of hidden nodes and links) to the activation functions of the hidden layer. The results presented in Chapter 5 supports both and further work is required to resolve the exact reason for the performance increase.

Catastrophic interference was significantly reduced by changing the alignment of the centre of the constrained region, Seipone [129] has suggested that evolution can reduce

interference between patterns. Further research into the effects of pattern order on the structure of the neural networks evolved may give some insight into the changes in the internal representations described by French [111].

Recent work by Bullinaria [146] suggests that learning strategies may have a significant effect on the performance of neural network evolution, advances have also been made in the constraint of the genetic algorithms search domain limiting it to feasible individuals using genetic techniques similar to RNA repair [147].

9.2.2 Performance Metrics for Joint Constraint Vector Fields

Current methods for assessing the error produced by the neural networks developed may not reflect the networks true performance in each case. At present the measurements used, SSE (used to assess neural network fitness during evolution), MSE (used to calculate network error during training,) and the Pythagorean error metric (used in reporting the results and comparing representations,) all depend on the comparison of the current output with the test set. A more useful error metric would be the distance of the corrected virtual limb from the boundary. This would provide more representative results in cases such as the *correctional discontinuity* where the vector is corrected to a diametrically opposite position on the boundary. Future work may consider the development of an error metric of this kind and possibly the development of a backpropagation based learning algorithm based on this error metric.

9.2.3 The Constraint of Rotation around the Limb

The previous chapters have focused on the development of quaternion-based constraints for the rotation of the limb in three-dimensional space. It has not however considered the rotation around the limb itself. The rotations being performed are better described as the swing and twist respectively [37]. This is an important consideration if the approach presented here is to be used in the constraint of anatomical limbs in three-dimensional space. Preliminary research has been carried out into this area with the training of a simple one-dimensional constraints representing rotation around the limb and combining these with constraints on the rotation of the limb.

9.2.3.1 Methodology

As in previous experiments a Generalised Multi-Layer Perceptron (GMLP) was used to model discontinuous vector fields representing quaternion constraints. Vector fields were trained to model both constraints on the rotation around the limb and combining both rotation of the limb and rotation around the limb. Initially the rotation around the limb was considered alone with the rotation of the limb constrained to a radius of twenty degrees. These constraints were then combined with those on the rotation of the limb. The datasets were created using similar method to that described in 4.1.1 with the addition of a second quaternion generated and combined with the first to represent the rotation around the limb. Its negation was stored and used as the initial component of the correction. The parameters for training and evolution were as shown in TABLE I.

9.2.3.2 Results

The results of preliminary experiments show that the neural network successfully learned discontinuous vector fields representing corrective quaternion based constraints on both the rotation of and around the limb. The results for a constraint on the rotation around the limb with a constant rotation of the limb show invariable results Fig. 62.

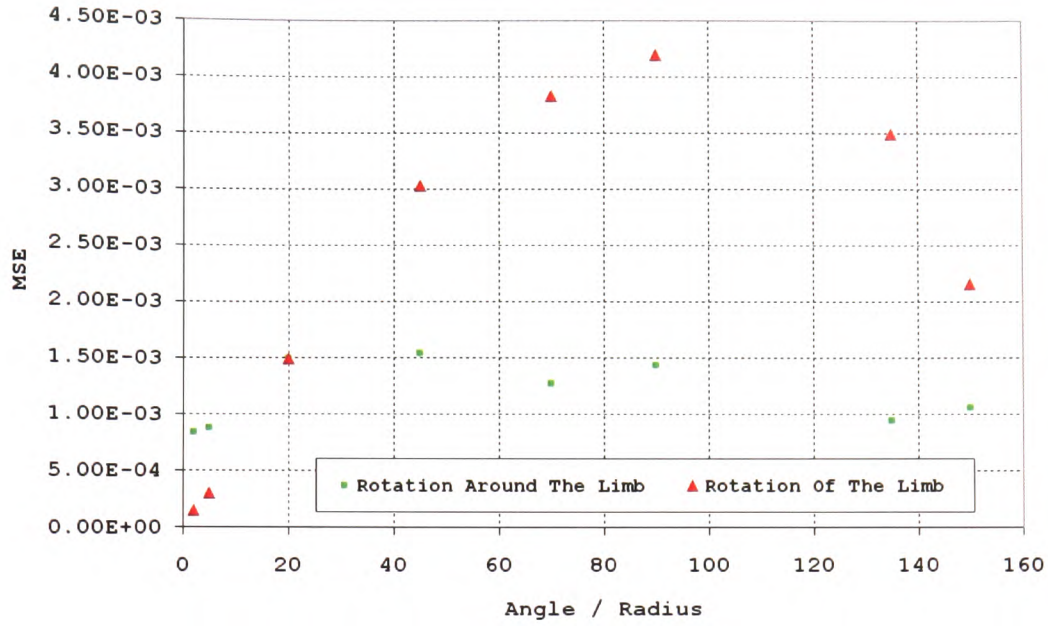


Fig. 62 - A graph showing the effect of the additional constraint on the rotation around the axis. The red triangle marker shows the effect of changing the limb rotation constraint with a constant rotational constraint around the limb of twenty degrees. The green square marker shows the error resulting from the increase in the constraint on rotation around the axis with a constant constraint on the rotation of the limb.

Distinct changes in the pattern of error are observed when a constant rotation around the limb (of 20°,) and varying the rotation of the limb are considered. An increase occurs as the constraint increases in size (Fig. 62) up to 90° followed by a decrease the error is to some extent symmetrical around 90°. In both cases there is an increase in the error compared to earlier constraints where no constraint on the rotation around the limb was enforced (shown in Fig. 18).

9.2.3.3 Discussion & Conclusions

The results indicate that the introduction of a secondary constraint on the rotation around the limb has a significant effect on the performance of the neural network, possibly caused by a further increase in the complexity of the discontinuous vector field that the neural network is required to learn. As the size of the constraint on the rotation of the limb increases error increases towards a radius of ninety degrees above this it decreases. Based on previous observations it is postulated that the shape and orientation of the dataset in quaternion space becomes more difficult to learn at this point. More

work is required to understand how the distribution of the quaternion vector-field changes and its effect on the performance of the neural network.

The results suggest that while there is an increase in error when the rotation of the limb is increased it may be possible to implement constraints on the rotations around the limb separately. It is observed (in Fig. 62,) that where the constraint on rotation of the limb is very small there is an increase in performance, indicating if the rotation of the limb was zero and a system considering only the rotation around the limb a suitable constraint of any size could be trained with high performance. Hence a quaternion based constraints system could be employed to constrain the rotation of and rotation around the limb separately. However one of the benefits of using quaternion based constraints, the ability to model these the relationships between these constraints, is lost.

Future work may investigate the factors the shape and orientation of the quaternion space vector field and how the errors observed can be reduced possibly investigating some of the performance related factors uncovered in Chapter 5. More complex networks may form part of any future solution.

9.2.4 Reduced Coordinate Encoding

Previous approaches to the constraint of both the rotation of and around the limb have reduced the dimensionality of the quaternion representation. Herda *et al* [5, 8] reduced the dimensionality of the sampled rotations by ensuring all scalar components were positive and omitting them as the quaternion is unit length (and the scalar positive,) these can be recovered from the three remaining components. Johnson [2] who projected one half of the unit quaternion hyper sphere onto a three-dimensional tangent space. Chapter 3 demonstrated the capabilities of evolved topology neural networks in the approximation of vector fields in three-dimensional space. It may be feasible to apply these techniques to vector fields representing joint constraints using reduced coordinate mappings in order to improve results or include rotation around the limb.

9.2.5 Multiple Dependent Joints

The internal biomechanical constraints of anatomical joint are often linked to the orientation of other joints as muscles and tendons often contributed to the constraint of more than one joint. For example what is often referred to as the ankle is anatomically two joints the ankle itself that provides anterior and posterior movement and the subtalar joint that provides medial and lateral movement [23, 148].

This work has only considered the constraint of individual joints however inter-joint dependencies have been modelled in other approaches [1]. It is feasible to model multiple joints using the presented approach however the input space would have to include the orientations of all related joints in the system, a significant increase in dimensionality. Research into dynamics systems for animated multi-jointed limbs suggests a hierarchical approach to combat the disproportionate increase in network size with problem space dimensionality [56]. A number of previous approaches have reduced the dimensionality of the problem space to three dimensions [2, 4, 5] though this would not significantly reduce problem space dimensionality. Research also suggests that complex vector fields may be simplified by conversion to basis fields [105] this may reduce the complexity of the vector field to be learned in each case.

9.2.6 Training from Sampled Data

To date only topologically evolved feed forward neural networks that undergo supervised learning have been considered. This is suitable for a number of applications where a dataset can be constructed detailing the whole of the vector field in the case of joint constraints this means the valid and invalid regions. In animation systems this is not a problem, as the animator would typically create the constraint boundary in an editor from which training data could be generated. In other application areas such as medical research where patient biomechanics are being recorded this is almost certainly impossible, as acquisition of data relies on motion capture techniques or mechanical measuring.

In order to utilise neural networks for applications where a complete vector field cannot be measured recurrent neural networks may be utilised. One-dimensional discontinuous functions have been approximated using recurrent neural network and reinforcement learning [88, 101, 102]. Pose constraints whose underlying representation is comparable to joint constraints were learned by a Scaled Gaussian Process Latent Variable Model (SGPLVM) in an approach which uses exponential maps rotated to avoid singularities [47].

Research has shown the inclusion of domain knowledge in learning algorithms can produce significant improvements in the learning of continuous two dimensional vector fields [104, 105]. Other techniques have considered the simplification of the vector field using such techniques as basis vectors [105]. These techniques may be applied within the framework of the evolved neural networks where full vector fields and a traditional back-propagation training algorithm were used.

9.3 Application of the Techniques Developed

9.3.2 Kinematic Modelling

The generation of character specific joint constraints may be a useful addition to character animation tools. It is feasible using the techniques described for creating irregular boundaries to create a character specific joint boundary limited for example by clothing or injury. Such constraints may have advantages in maintaining consistency

where procedural animation is used and where multiple individuals animate a single model. The approach also gives the potential advantage of a quaternion representation for all aspects of rotation within the system as indicated by a number of authors [6, 46]. The potential improvements in performance offered by these techniques may add to these benefits.

9.3.3 Biomechanical Modelling

Recent studies suggest that current joint constraint approaches are only capable of modelling gross motion and are unsuitable for surgical purposes such as simulation of surgical outcomes [149, 150]. Proposed solutions rely on moving frame implementations [149] leading to difficulties in constraint interrogation, increased complexity and lacking the benefits of a quaternion based approach, (such as the avoidance of singularities).

Skeletal models with joint constraints are used extensively in the study of lower body motion. At present simplified skeletons with various mechanical joints and crude constraints are advocated for cleaning up motion capture data [150]. Accurate joint modelling in this case is essential especially in the case of pathological patients who may have abnormal joint motions [150].

The techniques proposed here may be applied in this area creating more accurate constraints based on patient specific data. Using current techniques a rotational boundary could be recorded by the patient and training data created based on the techniques used for irregular boundaries.

9.3.4 Dynamics Modelling

The description of joints in dynamics simulations is often associated with a number of constraints that produce specified joint behaviour. These constraints provide forces which prevent the limbs from drifting apart and may provide angular constraint by reducing the degrees of freedom of a joint [151, 152]. Constraint forces are also used when objects intersect a joint is formed between the two components and repulsive forces calculated to reduce intersection to zero [151, 152].

In terms of constraint of the rotational movement, of for example, a virtual limb connected to a ball and socket joint whose constrained region of rotational motion is described as a bounding polygon, (with the body of this polygon describing the invalid region). It may be feasible to treat the intersection of the bounding polygon and virtual limb as a collision and generate forces to reduce the intersection. This would produce a force on the virtual limb that would translate into rotational force being applied to the virtual limb validating the constraint. Using the approach presented here it may be possible to develop a neural network based constraint that produces a corrective rotational force based on the position of the limb end point or orientation. This would reduce the computational complexity of the implementation with the additional advantage that the constraint boundary could be trained using data from various sources.

9.3.5 Pose Constraints

There has been some research into the combination of pose and joint constraints these include approaches which only consider pose constraints and assume (not unreasonably) that a favoured pose would not contain invalid joint configurations [47]. Johnson [2] presents a system of statistically based quaternion constraints which enforces pose and joint constraints in quaternion space. The inclusion of pose constraints may be considered in future work simplifying the modelling of joint constraints significantly as the set of poses that for example maintain balance is smaller than the set of all possible configurations for the limbs concerned. Such techniques however are not applicable to falling bodies where there may be no appropriate pose to maintain.

9.3.6 Camera Constraint

The discontinuous vector fields modelled in this work were created specifically to represent joint constraints. There are however other applications where these techniques may prove useful. One of these is the stabilization of a camera being rotated between orientations via spherical linear interpolation (SLERP), without a constraint it is impossible to ensure that the camera remains upright a limitation in the use of quaternions for camera control [153]. It may be possible to employ the constraints developed in this work to solve this problem.

9.4 Conclusion

The work undertaken in this thesis provides a building block for further research into the use of neural networks for anatomical joint constraint. A number of areas of future work have been identified based around both the improvement of the current techniques and their application in various domains. The application of the techniques is particularly important in order to obtain insight into the relationship between experimental performance and performance within an application.

References

- [1] W. Shao and V. Ng-Thow-Hing, "A General Joint Component Framework for Realistic Articulation in Human Characters," presented at 2003 Symposium on 3D Graphics, Monterey, California, USA, 2003.
- [2] M. P. Johnson, "Exploiting Quaternions to Support Expressive Interactive character Motion." Massachusetts: Massachusetts Institute of Technology, 1995.
- [3] Q. Liu and E. Prakash, C, "The Parameterization of Joint Rotation with the Unit Quaternion," presented at 7th Digital Image Computing: Techniques and Applications, Sydney, 2003.
- [4] L. Herda, R. Urtasun, and P. Fua, " Hierarchical Implicit Surface Joint Limits for Human Body Tracking," Computer Vision Lab, Ecole Polytechnique Federal de Lausanne (EPFL), Lusanne CH-1015, 2004.
- [5] L. Herda, R. Urtasun, P. Fua, and A. Hanson, "Automatic determination of shoulder joint limits using quaternion field boundaries," *International Journal of Robotics Research*, vol. 22, pp. 419-444, 2003.
- [6] J. Lee, "A Hierarchical Approach to Motion Analysis and Synthesis for Articulated Figures," in *Department of Computer Science*. Daejeon: Korean Advanced Institute of Science and Technology, 2000, pp. 93.
- [7] F. E. Zajac, "Biomechanics and muscle coordination of human walking Part 2: Lessons from dynamical simulations and clinical implications," *Gait and Posture*, vol. 17, pp. 1-17, 2003.
- [8] A. Watt and M. Watt, "Forward vs Inverse Kinematics in Computer Animation," in *Advanced Animation and Rendering Techniques*. New York: ACM Press, 1992, pp. 371-384.
- [9] K. Manal, X. Lu, M. K. Nieuwenhuis, P. J. M. Helders, and T. S. Buchanan, "Force transmission through the juvenile idiopathic arthritic wrist: a novel approach using a sliding rigid body spring," *Journal of Biomechanics*, vol. 35, pp. 203-218, 2002.
- [10] W. Manurel and D. Thalmann, "Human shoulder joint modelling including scapulo-thoracic constraints and joint sinus cones," *Computers and Graphics*, vol. 24, pp. 203-218, 2000.
- [11] J. D. Feikes, J. J. O'Connor, and A. B. Zavatsky, "A constraint-based approach to modelling the mobility of the human knee joint," *Journal of Biomechanics*, vol. 36, pp. 125-129, 2003.
- [12] Y. Zhang and J. Wang, "A dual neural network for constrained torque optimization of kinematically redundant manipulators," *IEEE Transactions on System, Man and Cybernetics: Part B*, vol. 32, pp. 654-662, 2002.

- [13] A. D'Souza, V. S. and S. Stefan, "Learning Inverse Kinematics," presented at International Conference on Intelligent Robots and System, Maui, Hawaii, USA, 2001.
- [14] A. J. Hamel, N. A. Sharkey, F. L. Buczek, and J. Michelson, "Relative motions of the tibia, talus and calcareous during the stance phase of gait," *Gait and Posture*, vol. 20, pp. 153-157, 2003.
- [15] D. N. Yang, D. N. Condie, M. H. Granat, J. P. Paul, and D. I. Rowley, "Effects of joint motion constraints on the gait of normal subjects and their implications on the further developments of hybrid FEZ orthosis for paraplegic persons.," *Journal of Biomechanics*, vol. 29, pp. 217-226, 1996.
- [16] R. R. Selmic and L. L. Lewis, "Deadzone Compensation in Motion Control Systems Using Neural Networks," *IEEE Transactions on Automatic Control*, vol. 45, pp. 602-613, 2000.
- [17] A. E. Engin and S. T. Tumer, "Improvised dynamic model of human knee joint and its response to loading," *Journal of Biomechanical Engineering*, vol. 115, pp. 137-142, 1993.
- [18] M. G. Ishac, D. A. Winter, and J. J. Engin, "Limb segment model validation: The power imbalance story," *Gait and Posture*, vol. 4, pp. 167-208, 1996.
- [19] D. Popovic, R. B. Stein, M. N. Oguztörel, M. Lebedowska, and S. Jonić, "Optimal control of walking with functional electrical stimulation a computer simulation study," *IEEE Transactions of Rehabilitation Engineering*, vol. 7, pp. 69-79, 1999.
- [20] J. Feikes, "Articular surface representation in a 3D model of the knee mobility," *Journal of Biomechanics*, vol. 31, pp. 148, 1998.
- [21] D. R. Wilson, J. D. Feikes, and J. J. O'Connor, "Ligaments and articular contact guide passive knee flexion," *Journal of Biomechanics*, vol. 31, pp. 1127-1136, 1998.
- [22] T. J. A. Mommersteeg, R. Huiske, L. Blankevoort, J. G. M. Kooloos, and J. M. G. Kauer, "An inverse dynamics modelling approach to determine the restraining function of the human knee ligament bundles," *Journal of Biomechanics*, vol. 30, pp. 139-146, 1997.
- [23] F. Gray, *Gray's Anatomy, the classic collectors edition*, 15 ed. New York: Bounty Books, 1977.
- [24] A. J. Van Soest and G. P. Van Galen, "Coordination of multi-joint movements: An introduction to emerging views.," *Human Movement Science*, vol. 14, pp. 391-400, 1995.
- [25] M. G. Pandy and B. Nici, "Synthesis of human walking: a planar model for single support," *Journal of Biomechanics*, vol. 21, pp. 1053-1063, 1988.

- [26] M. G. Pandy and B. Nicip, "Quantitative assessment of gait determinants during single stance via a three-dimensional model - part 1 normal gait," *Journal of Biomechanics*, vol. 22, pp. 69-79, 1989.
- [27] K. B. Shelburne, M. G. Pandy, and M. R. Torry, "Comparison of shear forces and ligament loading in the healthy and ACL-deficient knee during gait," *Journal of Biomechanics*, vol. 37, pp. 313-319, 2004.
- [28] J. H. Heegaard, P. F. Leyvraz, and C. B. Hovey, "A computer model to simulate patellar biomechanics following total knee replacement: the effects of femoral component alignment," *Clinical Biometrics*, vol. 16, pp. 415-423, 2001.
- [29] J. J. Faraway, X. Zhang, and D. B. Chaffin, "Rectifying postures reconstructed from joint angles to meet constraints," *Journal of Biomechanics*, vol. 32, pp. 733-736, 1999.
- [30] C. B. Phillips, J. Zhao, and N. Badler, "Interactive Real-Time Articulated Figure Manipulation using Multiple Kinematics Constraints," *SIGGRAPH 90 Course notes (Human figure animation : approaches and applications)*, 1990.
- [31] J. Eng and D. A. Winter, "Kinematic analysis of the lower limbs during walking: what information can be gained from a 3D model," *Journal of Biomechanics*, vol. 28, pp. 753-758, 1995.
- [32] T. Furuta, T. Tawara, Y. Okumura, M. Shimizu, and K. Tomiyama, "Design and construction of a series of compact humanoid robots & development of bipedal walking control strategies," *Robotics and Autonomous Systems*, vol. 37, pp. 81-100, 2001.
- [33] A. Watt and M. Watt, *Advanced Rendering Techniques*. New York: ACM Press, 1992.
- [34] P. Baerlocher, "Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures," in *Department D'Informatique*. Lausanne: Ecole Polytechnique Federal De Lausanne, 2001, pp. 156.
- [35] A. Maciel, L. P. Nedel, and C. M. D. S. Freitas, "Anatomy Based Joint Models for Virtual Human Skeletons," presented at IEEE Computer Animation, Geneva, Switzerland., 2002.
- [36] J. U. Korein, *A geometric investigation of reach*. Massachusetts: MIT Press, 1984.
- [37] P. Baerlocher and R. Boulic, "Parameterization and Range of Motion of the Ball and Socket Joint," presented at IFIP TC5/WG5.10 DEFORM' 2000 Workshop and AVATARS' 2000 Workshop on Deformable Avatars, 2000.
- [38] F. S. Grassia, "Practical Parameterization of Rotations Using the Exponential Map," *Journal of Graphics Tools*, vol. 3, 1998.

- [39] P. M. Isaacs and M. F. Cohen, "Controlling dynamic simulation with kinematic constraints," *ACM SIGGRAPH Computer Graphics*, vol. 21, pp. 215 - 224, 1987.
- [40] D. E. Gyi, R. E. Sims, J. M. Porter, R. Marshall, and K. Case, "Representing Older and Disabled People in Virtual User Trials: Data Collection Methods," *Applied Ergonomics*, vol. 35, pp. 443-451, 2004.
- [41] A. E. Engin and S. T. Tumer, "Three dimensional kinematic modelling of the human shoulder complex part 1: physical model & determination of joint sinus cone," *Journal of Biomechanical Engineering*, vol. 111, pp. 107-112, 1989.
- [42] P. M. Issacs, Micheal, F C, "Controlling Dynamic Simulation with Kinematic Constraints, Behaviour Functions and Inverse Dynamics," *ACM Transactions on Computer Graphics*, vol. 21, pp. 215-223, 1987.
- [43] A. Watt and M. Watt, "Parameterisation of Rotation," in *Advanced Animation and Rendering Techniques*. New York: ACM Press, 1992, pp. 356-368.
- [44] J. J. Kuffner, "Effective Sampling and Distance Metrics for 3D Ridgid body Path Planning," presented at IEEE International Conference on Robotics and Automation, New Orleans, Los Angels, USA, 2004.
- [45] S. L. Altermann, *Rotations, Quaternions and Double Groups*. New York: Dover Publications Inc., 2005.
- [46] M. P. Johnson, "Exploiting Quaternions to Support Expressive Interactive character Motion," in *MIT Media Lab*. Massachusetts: Massachusetts Institute of Technology, 2003.
- [47] K. Grochow, S. Martin, L. , A. Hertzmann, and Z. Popovic, "Style-Based Inverse Kinematics," presented at ACM Transactions on Graphics, 2004.
- [48] The-Open-University-Course-Team, "Complex Numbers, Mathematics Foundation Course, Block VI Mathematical Structures, Unit 1.," The Open University, Milton Keynes 1993.
- [49] K. Mehrotra, C. K. Mohan, and S. Ranka, "Introduction," in *Elements of Artificial Neural Networks*. Massachusetts: MIT Press, 1997, pp. 1-40.
- [50] D. W. Coit, J. Billa, D. Lenoard, A. Smith, W. Clark, and A. El-Jarovd, "Wave soldering process control modelling using a neural network approach," in *Intelligent Engineering Systems through artificial neural networks*, vol. 4, C. H. Dagli, B. R. Fernandez, J. Ghosh, and R. T. S. Kumara, Eds.: ASME Press, New York, 1994, pp. 999-1004.
- [51] J. Kamruzzaman and R. R. Begg, "Support Vector Machines and Other Pattern Recognition Approaches to the Diagnosis of Cerebral Palsy Gait," *IEEE Transactions on Biomedical Engineering*, vol. 53, pp. 2479-2490, 2006.

- [52] P. Janik and T. Lobos, "Automated classification of power-quality disturbances using SVM and RBF networks," *IEEE Transactions on Power Delivery*, vol. 21, pp. 1663-1669, 2006.
- [53] X. Yao and Y. Liu, "A New Evolutionary System for Evolving Artificial Neural Networks," *IEEE Transactions on Neural Networks*, vol. 8, pp. 694-712, 1997.
- [54] N. H. R. Goerke and R. Eckiller, "A Neural Network that Generates Attractive Vector Fields for Robot Control," presented at Fourth European Congress on Intelligent Techniques and Soft Computing, Aachen, 1996.
- [55] N. H. R. Goerke, F. Kintzler, A. Rabe, D. Roggisch, and R. Eckmiller, "Controlling the Khepera Robot by Neural Network Modules," presented at First International Khepera Workshop, Paderborn: HNI-Verglasschriftenreihe, 1999.
- [56] R. Grzeszczuk, Terzopoulos, D. Hinton, G, "NuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models," presented at 25th Annual Conference on Computer Graphics and Interactive Techniques, 1998.
- [57] Y. N. Kulchin and A. V. Panova, "Neural Networks for Reconstruction of Signal from Distributed Measuring System of Optical Amplitude Sensors," *Pacific Science Review*, vol. 3, pp. 1-4, 2001.
- [58] R. R. Selmic and L. L. Lewis, "Neural Network Approximation of Piecewise Continuous Functions: Application to Friction Compensation," in *Soft Computing and Intelligent Systems: Theory and Applications*. New York: Academic, 2000.
- [59] K. Mehrotra, C. K. Mohan, and S. Ranka, "Supervised Learning: Multi-layer Networks," in *Elements of Artificial Neural Networks*. Massachusetts: MIT Press, 1997, pp. 63-105.
- [60] M. Riedmiller and H. Braun, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," presented at IEEE International Conference on Neural Networks, San Francisco, CA, 1993.
- [61] M. Hargan and M. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm," *IEEE Transactions on Neural Networks*, vol. 5, pp. 989-993, 1994.
- [62] D. E. Rumelheart and J. L. McClelland, "Parallel distributed processing: Explorations in the microstructure of cognition," *IEEE Transactions on Neural Networks*, vol. 1, 1986.
- [63] H. A. Mayer and R. Schwaiger, "Differentiation of Neuron Types by Evolving Activation Function Templates for Artificial Neural Networks," presented at World Congress on Computational Intelligence, International Joint Conference on Neural Networks, Honolulu, Hawaii, USA, 2002.
- [64] H. A. Mayer and R. Schwaiger, "Evolution of Cubic Spline Activation Functions for Artificial Neural Networks," presented at 10th Portuguese Conference on Artificial Intelligence (EPIA 2001), 2001.

- [65] Y. Liu and X. Yao, "Evolutionary Design of Artificial Neural Networks with Different Nodes," presented at The Third International Conference on Evolutionary Computation, 1996.
- [66] V. Vapnik, *The Nature of Statistical Learning Theory*, 2 ed. New York: Springer, 1995.
- [67] A. Autret, "Modular Neural Networks for Analysis of Flow Cytometry Data," in *School of Computing*. Treforest: University of Glamorgan, 2003.
- [68] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.
- [69] S. Zomer, R. G. Brereton, J. F. Carter, and C. Eckers, "Support Vector Machines for the Discrimination of Analytical Chemical Data: Applications to the Determination of Tablet Production by Pyrolysis-gas Chromatography-mass Spectrometry," *The Analyst*, vol. 129, pp. 175-181, 2004.
- [70] R. Collobert and S. Bengio, "Links between Perceptrons, MLPs and SVMs," presented at Twenty-first International Conference on Machine Learning, Banff, Alberta, Canada, 2004.
- [71] V. Vapnik, S. E. Golowich, and A. J. Smola, "Support Vector Method for Function Approximation, Regression Estimation and Signal Processing," *Advances in Neural Information Processing*, vol. 9, pp. 281-287, 1997.
- [72] T. Joachims, "Making Large-Scale SVM Learning Practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Scholkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge: MIT Press, 1999.
- [73] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, pp. 273-297, 1995.
- [74] X. Zhang and H. Ke, "ALL/AML Cancer Classification by Gene Expression Data Using SVM and CSVM Approach," *Genome Informatics*, vol. 11, pp. 237-239, 2000.
- [75] B. Scholkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New Support Vector Algorithms," *Neural Computation*, vol. 12, pp. 1207-1245, 2000.
- [76] S. Vijayakumar and S. Wu, "Sequential Support Vector Classifiers and Regression," presented at International Conference on Soft Computing, 1999.
- [77] T. Masuyama and H. Nakagawa, "Two Step POS Selection for SVM Based Text Categorization," *IECE Transactions on Information and Systems*, vol. E87-D, pp. 1-7, 2004.
- [78] D. E. Goldberg, "A Gentle Introduction to Genetic Algorithms," in *Genetic Algorithms in Search, Optimization, and Machine Learning*, D. Edward, Ed.

- Reading, Massachusetts: Addison-Wesley Publishing Company Inc., 1989, pp. 1-26.
- [79] D. E. Goldberg, "Genetic Algorithms Revisited: Mathematical Foundations," in *Genetic Algorithms in Search, Optimization, and Machine Learning*, D. Edward, Ed. Reading, Massachusetts: Addison-Wesley Publishing Company Inc., 1989.
 - [80] H. A. Mayer, "A Taxonomy of the Evolution of Artificial Neural Systems," presented at Scientific Computing in Salzburg, 2005.
 - [81] R. Huber, H. A. Mayer, and R. Schwaiger, "netGEN - A Parallel System Generating Problem-Adapted Topologies of Artificial Neural Networks by means of Genetic Algorithms," presented at Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3, Dortmund, 1995.
 - [82] S. H. Lane, D. A. Handelman, and J. Gelfand, J., "Theory and development of higher order CMAC neural networks," *Control Systems Magazine, IEEE theory and development of higher order CMAC neural networks*, vol. 12, 1992.
 - [83] H. A. Mayer, M. Strapetz, and R. Fuchs, "Simultaneous Evolution of Structure and Activation Function Types in Generalized Multi-Layer Perceptrons," presented at WSES International Conference on Neural Networks and Applications, Puerto De La Cruz, Tenerife, Spain, 2000.
 - [84] L. Vecchi, F. Piazza, and A. Uncini, "Learning and Approximation Capabilities of Adaptive Spline Activation Function Neural Networks," *Neural Networks*, vol. 11, pp. 259-270, 1998.
 - [85] K. Shibata and K. Ito, "Gauss-Sigmoid Neural Network," presented at International Joint Conference on Neural Networks, 1999.
 - [86] A. L. V. Coelho, D. Weingaertner, and F. J. Von Zuben, "Evolving Heterogeneous Neural Networks for Classification Problems," presented at Genetic and Evolutionary Computation Conference, San Francisco, 2001.
 - [87] Y. Shen, B. Wang, F. Chen, and L. Cheng, "A new multi-output neural model with tuneable activation function and its applications," *Neural Processing Letters*, vol. 20, pp. 85-104, 2004.
 - [88] S. W. Lee and J. H. Kim, "Control of system deadzones using neural-network based learning controller," presented at IEEE International Conference on Neural Networks, 1994.
 - [89] R. Garziera, "Recursive Formulation of the Inverse Kinematics of Redundant Robots Performing Tasks with Priority Order," *ASME Journal of Mechanical Design*, vol. 116, pp. 337-338, 1994.
 - [90] F. Piazza, S. Smerilli, A. Uncini, M. Griffò, and R. Zunino, "Fast Spline Neural Networks for Image Compression," presented at 8th Italian Workshop on Neural Nets, Vietri Sul Mare, Salerno, Italy, 1996.

- [91] K. Hlavackova and M. Verleysen, "Placing spline knots in neural networks using splines as activation functions," *Neurocomputing*, vol. 17, pp. 159-166, 1997.
- [92] S. Fiori, "Hybrid independent component analysis by adaptive LUT activation function neurons," *Neural Networks*, vol. 15, pp. 85-94, 2002.
- [93] M. Solazzi and A. Uncini, "Artificial Neural Networks with Adaptive Multidimensional Spline Activation Functions," presented at IEEE International Joint Conference on Neural Networks (IEEE-INNS-ENNS), Como, Italy, 2000.
- [94] M. Solazzi and A. Uncini, "Adaptive multidimensional spline neural network for digital equalization," presented at IEEE International Conference on Acoustics, Speech and Signal Processing, Istanbul, 2000.
- [95] M. Solazzi and A. Uncini, "Regularising neural networks using flexible multivariate activation function," *Neural Networks*, vol. 17, pp. 247-260, 2004.
- [96] K. Huang, Y. Haiqin, I. King, and M. R. Lyu, "Local Learning vs. Global Learning: An Introduction to Maxi-Min Margin Machine," in *Support Vector Machines: Theory and Applications*, vol. 177/2005, *Studies in Fuzziness and Soft Computing*: Springer-Verlag, 2005.
- [97] S. Guarnieri, F. Piazza, and A. Uncini, "Multilayer Feedforward Networks with Adaptive Spline Activation Function," *IEEE Transactions on Neural Networks*, vol. 10, pp. 672-683, 1999.
- [98] G. Arfken, "Vector Analysis," in *Mathematical Methods for Physicists*, G. B. Arfken and H. J. Weber, Eds., 3rd ed. Orlando: Academic Press, 1985.
- [99] R. A. Crawfis, B. Becker, B. Cabral, and N. Max, "Volume Rendering of 3D Scalar and Vector Fields at LLNL," presented at Supercomputing, Portland, Oregon, 1993.
- [100] S. N. Huang, K. Tan, K. and T. Lee, H, "Adaptive Motion Control using Neural Network Approximations," *Automatica*, vol. 38, pp. 227-233, 2002.
- [101] R. Rastko, R. R. Selmic, and L. L. Lewis, "Neural Net Backlash Compensation with Hebbian Tuning using Dynamic Inversion," *Automatica, Special Issue on Neural Networks for Feedback Control*, 1999.
- [102] T. Taware and G. Tao, "Neural-hybrid control of systems with sandwidge dead-zones," *International Journal of Adaptive Control and Signal Processing*, vol. 16, pp. 473-469, 2002.
- [103] C. W. Anderson and Z. Hong, "Reinforcement Learning with Modular Neural Networks for Control," presented at IEEE International Workshop on Neural Networks Applied to Control and Image Processing, 1994.
- [104] Y. Kuzo, M. Mitsui, H. Kawakimi, and T. Mori, "A Learning Method for Vector Field Approximation by Neural Networks," presented at IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 1998.

- [105] F. A. Mussa-Ivaldi and S. F. Griszter, "Vector Field Approximations: A Computational Paradigm for Motor Control and Learning," *Biological Cybernetics*, vol. 67, pp. 491-500, 1992.
- [106] Y. Kuroe, M. Mitsui, H. Kawakimi, and T. Mori, "A Learning Method for Vector Field Approximation by Neural Networks," presented at IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 1998.
- [107] I. Kimura, Y. Susaki, R. Kiyohara, A. Kaga, and Y. Kuroe, "Gradient-based PIV Using Neural Networks," *Journal of Visualization*, vol. 5, pp. 363-370, 2002.
- [108] Y. J. Kim, D. H. Kim, and J. H. Kim, "Evolutionary Programming-Based Uni-Vector Field Method for Fast Mobile Robot Navigation," *Lecture Notes in Computer Science*, vol. 1585, pp. 154-156, 1999.
- [109] E. P. Tsang and C. J. Wang, "A generic neural network approach for constraint satisfaction problems.," in *Neural Network Applications*, J. G. Taylor, Ed.: Springer-Verlag, 1992, pp. 12-22.
- [110] C. J. Wang and E. P. Tsang, "Solving the constraint satisfaction problem using neural networks," presented at IEEE Second International Conference on Artificial Neural Networks, 1991.
- [111] R. M. French, "Catastrophic Forgetting in Connectionist Networks: Causes, Consequences and Solutions," *Trends in Cognitive Sciences*, vol. 3, pp. 128-135, 1999.
- [112] J. E. Jackson, "Introduction," in *A Users Guide To Principal Components*: John Wiley & Sons, Inc, 1991, pp. 4-25.
- [113] A. F. Siegel, Morgan, C J, "Describing Distributions," in *Statistics and Data Analysis: An Introduction*. Chichester: John-Whiley, 1996, pp. 67-141.
- [114] F. Daly, D. J. Hand, M. C. Jones, A. D. Lunn, and K. J. McConway, "Models for Data II," in *Elements of Statistics*, J. K. Brown, Ed. Workingham: Addison-Wesley Publishing Company, 1995.
- [115] M. S. Srivastava, "Multivariate Normal Distributions," in *Models of Multivariate Statistics*. New York: John Wiley and Sons Inc, 2002, pp. 20-56.
- [116] M. S. Srivastava, "Eigenvectors and Eigenvalues," in *Models of Multivariate Statistics*. New York: John Wiley and Sons Inc, 2002, pp. 632-633.
- [117] The-Open-University-Course-Team, "Complex Numbers, Mathematics Foundation Course, Block IV Matrices, Unit 4.," The Open University, Milton Keynes 1993.
- [118] E. Kreyszig, "Linear Algebra: Matrix Eigenvalue Problems," in *Advanced Engineering Mathematics*, 9th ed. Singapore: John Wiley & Sons, 2006, pp. 333-364.

- [119] J. E. Jackson, "Computational Methods," in *A Users Guide To Principal Components*: John Wiley & Sons, Inc, 1991, pp. 450-455.
- [120] J. Huang and E. C. Prankash, "Sinus cone a theta-phi algorithm for human arm animation," presented at Proceedings for 2000 IEEE Conference on Information Visualization, 2000.
- [121] B. Girau and A. Tisserand, "On-line Arithmetic based Reprogrammable Hardware Implementation of Multi-layer Perceptron Back-Propagation," presented at Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems, 1996.
- [122] J. Dayhoff, "Applications and Future Directions," in *Neural Network Architectures*. New York: Van Nostrand Reinhold, 1990, pp. 217-245.
- [123] G. F. Luger, "Machine Learning: Connectionist," in *Artificial Intelligence*, 5th ed. Essex, England: Pearson Education Limited, 2005, pp. 435-507.
- [124] J. Dayhoff, "Back-Error Propagation," in *Neural Network Architectures*. New York: Van Nostrand Reinhold, 1990, pp. 58-79.
- [125] H. A. Mayer and P. Maier, "Evolution of Neural Go Players," *Osterreichische Gesellschaft fur Artificial Intelligence*, vol. 24, pp. 8-16, 2005.
- [126] H. A. Mayer, R. Huber, and R. Schwaiger, "Lean Artificial Neural Networks - Regularization Helps Evolution," presented at 2nd Nordic Workshop on Genetic Algorithms and their Applications, Vaasa, Finland, 1996.
- [127] H. A. Mayer and R. Schwaiger, "Evolutionary and Coevolutionary Approaches to Time Series Prediction Using Generalized Multi-Layer Perceptrons," presented at Congress on Evolutionary Computation, Washington DC, 1999.
- [128] H. A. Mayer, "Symbiotic Co evolution of Artificial Neural Networks and Training Data Sets," presented at 5th International Conference on Problem Solving from Nature, Amsterdam, The Netherlands, 1998.
- [129] T. Seipone and J. A. Bullinaria, "Evolving Improved Incremental Learning Schemes for Neural Network Systems.," presented at 2005 IEEE Congress on Evolutionary Computing, Piscataway, NJ, 2005.
- [130] G. F. Miller, P. M. Todd, and S. Hegde, U., "Designing neural networks using genetic algorithms," presented at Third International Conference on Genetic Algorithms, San Mateo, California, 1989.
- [131] A. Zell, G. Marmier, M. Vogt, N. Mach, R. Huebner, K. U. Herrmann, T. Soye, M. Schmalzl, T. Sommer, A. Hatzigeorgiou, S. Doering, and D. Posselt, "Stuttgart Neural Network Simulator, User Manual," University of Stuttgart, Stuttgart 1994.
- [132] R. E. Smith, D. E. Goldberg, and J. A. Earickson, "Sga-c: A c-language implementation of a simple genetic algorithm.," University of Alabama, Tuscaloosa TCGA 91002, May 1991.

- [133] D. E. Goldberg, "Computer Implementation of a Genetic Algorithm," in *Genetic Algorithms in Search, Optimization, and Machine Learning*, D. Edward, Ed. Reading, Massachusetts: Addison-Wesley Publishing Company Inc., 1989.
- [134] K. A. De Jong, "An analysis of the behaviour of a class of genetic adaptive systems." Michigan: University of Michigan, 1975.
- [135] D. E. Goldberg, "De Jong and Function Optimization," in *Genetic Algorithms in Search, Optimization, and Machine Learning*, D. Edward, Ed. Reading, Massachusetts: Addison-Wesley Publishing Company Inc., 1989, pp. 106-120.
- [136] E. Oyama, A. Arvin, K. F. MacDorman, T. Maeda, and S. Tachi, "A modular neural network architecture for inverse kinematics model learning," *Neurocomputing*, vol. 38-40, pp. 797-805, 2001.
- [137] E. Oyama and S. Tachi, "Modular neural net system for inverse kinematics learning," presented at International Conference on Robotics and Automation, 2000.
- [138] E. Oyama and S. Tachi, "Inverse Kinematics Learning by Modular Architecture Neural Networks," presented at International Joint Conference on Neural Networks, Washinton D.C., 1999.
- [139] D. DeMers and K. Kreutz-Delgado, "Learning Global Direct Inverse Kinematics," in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. Lippmann, Eds.: Morgan Kaufmann Publishers Inc., 1992, pp. 589-594.
- [140] M. Toussaint and S. Vijayakumar, "Learning Discontinuities for Switching between Local Models," presented at 19th International Conference on Artificial Intelligence, Edinburgh, UK, 2005.
- [141] M. F. Wilkins, "Neural network analysis of multivariate flow cytometric data from phytoplankton," in *School of Pure and Applied Biology*. Cardiff: University of Wales, 1995.
- [142] G. Camps-Valls, J. D. Martin-Guerreo, J. L. Rojo-Alvarez, and E. Soria-Olivas, "Fuzzy sigmoid kernel for support vector classifiers," *Neurocomputing*, vol. 62, pp. 501-506, 2004.
- [143] M. Skrbek, "Fast Neural Network Implementation," *Neural Network World*, vol. 9, pp. 375-391, 1999.
- [144] F. Boussaid, A. Bouzerdoun, and D. Chai, "VLSI Implementation of a Skin Detector based on Neural Network," presented at 5th International Conference on Information, Communications and Signal Processing, Bangkok, Thailand, 2005.
- [145] A. Janin and N. Morgan, "SpeechCorder, the portable meeting recorder," presented at International Workshop on Hands-Free Speech Communication, Kyoto, Japan, 2001.

- [146] J. A. Bullinaria, "Ensemble Techniques for Avoiding Poor Performance in evolved Neural Networks," presented at International Joint Conference on Neural Networks, Piscataway, NJ, 2006.
- [147] P. Rohlfshagen and J. A. Bullinaria, "An Exonic Genetic Algorithm with RNA Editing Inspired Repair Function for the Multiple Knapsack Problem," presented at UK Workshop on Computational Intelligence, Leeds, UK, 2006.
- [148] K. F. Wells and K. Luttgens, *Kinesiology, scientific basis of human motion*. East Sussex: W B Saunders Company, 1979.
- [149] D. Gattamelata, P. Engenio, and P. P. PValentini, "Accurate Geometrical Constraints for the Computer Aided Modelling of the Hummer Upper Body," *Computer-Aided Design*, vol. 39, pp. 540-547, 2007.
- [150] R. Ward, R. Baker, and A. Schache, "Anatomical constraints of the lower limb joints: Implications for kinematic modelling for quantitative gait analysis," *Gait and Posture*, vol. 24, pp. 103-104, 2006.
- [151] R. Smith, "How to make new joints in ODE," vol. 2008, 2002.
- [152] R. Smith, "Constraints in Ridgid Body Dynamics," in *Games Programming Gems: 3*, A. Kirmse, Ed., 1st ed. Hingham, Massachusetts: Charles River Media, 2004, pp. 241-253.
- [153] J. D. Foley, A. Van Dam, S. Feiner, and J. F. Hughes, *Computer Graphics Principles and Practice*: Addison Wesley, 1995.

Appendix A.

Published Papers

G. Jenkins and P. Angel, "Evolved Topology Generalized Multi-Layer Perceptron (GMLP) for Joint Constraint Modeling," presented at 9th International Conference on Computer Modeling and Simulation, Oriel Collage Oxford, 2006.

EVOLVED TOPOLOGY GENERALIZED MULTI-LAYER PERCEPTION (GMLP) FOR JOINT CONSTRAINT MODELLING.

GLENN JENKINS, DR. PAUL ANGEL
School of Computing, University of Glamorgan,
Pontypridd (Cardiff),
CF37 1DL,
Wales
Email: <gjenkins><pangel>@glam.ac.uk

ABSTRACT

The accurate simulation of anatomical joint models is becoming increasingly important for both medical and animation applications. We propose the use of Artificial Neural Networks to accurately simulate joint constraints based on recorded data. This paper describes the application of Genetic Algorithm approaches to neural network training in order to model corrective piece-wise linear / discontinuous functions required to maintain valid joint configurations. The results show that Artificial Neural Networks are capable of modelling continuous boundary shapes for a range of constraint sizes.

KEYWORDS: anatomical joint constraint, NetJEN, GMLP, piece-wise linear, discontinuous, neural networks

INTRODUCTION

Anatomical joint models are important constituents of anatomical models, they are used in simulation to retain anatomically correct movement and ensure limbs do not separate or intersect. Anatomical models are used in both medicine and animation to create model humans as characters, teaching aids or to evaluate the benefits of surgical or prosthetic intervention (Zajac 2003, Watt and Watt 1992, Manal, et al. 2002).

Many current techniques are limited by their underlying representation or their abstraction of the joint function and there is increasing demand for anatomically correct joints for both animation and medicine. However in current applications, increasing accuracy leads to increasing complexity which requires additional computation (Watt and Watt 1992, Zhang and Wang 2002, D'Souza, et al. 2001).

The long term aim of this work is to create an anatomically correct joint model trained using person specific data (from non-invasive (Hamel, et al. 2003) or invasive (Yang, et al. 1996) sources). This will provide an accurate representation of an individual's mobility. The accurate representation of joint constraints by Artificial Neural Networks (ANN) has advantages over methods which use coarse approximations and computationally expensive recursive or iterative techniques.

This paper investigates the application of ANN techniques to model a joint constraint system. From the training data given, the network learns discontinuous corrective functions which model the behaviour of the joint and ensure the joint configuration remains valid during movement. Using evolutionary techniques based on genetic algorithms, the topology of the network is configured dynamically to approximate the piece-wise linear properties inherent in discontinuous functions (Selmic and Lewis 2000).

BACKGROUND

Inverse Kinematics (IK) techniques attempt to resolve one or more constraints which constitute a constraint system. This problem is compounded by the existence of zero or more solutions (Watt and Watt 1992). Numerical techniques are favoured over analytical techniques as the inversion of forward kinematics functions becomes more difficult as the systems complexity increases. Common approaches are based on resolved motion rate (Madhavapeddy and Ferguson 1998, Baerlocher and Boulic 2004) and optimization techniques (Badler, et al. 1993, Zhao and Badler 1994, Nelson 1988).

Speed and complexity limitations associated with IK have been overcome using ANNs (Guez and Ahmad 1988). Recurrent neural networks have been used to identify optimum solutions to inverse kinematics problems (Ding and Wang 1999, Ding and Tso 1999, Zhang, et al. 2003, Zhang and Wang 2002, Zhang, et al. 2002, Xia and Wang 2001, D'Souza, et al. 2001) and have

also been developed to overcome problems in numerical techniques which use Jacobian inversion (Wang 1997). This operation is difficult, especially when the matrix is non-square as in the case of redundant manipulators (Watt and Watt 1992).

The problem of constructing anatomical joints has been approached in several ways. Engin (Engin and Tumer 1993) classifies these as 'anatomically based' and 'phenomenological' joints. Anatomically-based joints represent the joint through the interaction of geometrical models that represent the physical components of the joint whereas phenomenological joints use mathematical models to describe the behavior of the joint without reference to its constituent parts.

Primitive joint constraints have been parameterized using Euler angles (Faraway, et al. 1999, Eng and Winter 1995, Furuta, et al. 2001). Inter-dimensional dependencies cannot be easily represented using Euler angles (Baerlocher 2001), and singularities or "Gimbal Lock" are encountered. Feikes et al (Feikes, et al. 2003) used special orthogonal matrices, a rotational parameterization not susceptible to "Gimbal Lock", to overcome this.

N-dimensional boundary representations preserve the relationships between rotational degrees of freedom. Conceptually a number of points along the boundary are obtained through measurement, and then approximated to an n-dimensional shape. Pioneered by Korein (Korein 1984) whose 2D spherical polygons constrained the movement of robotic arms, this technique has also been employed to constrain the 'swing' component in a swing-twist parameterization specifically for ball and socket joints (Baerlocher and Boulic 2000, Korein 1984). Cone based polygons using one (Engin and Tumer 1989) or more (Manurel and Thalmann 2000) cones have also been suggested for the complex shoulder joint. The quaternion iso-surface approach of Herda et al both preserves the relationship between the degrees of freedom and avoids singularities found in Euler angles (Herda, et al. 2003, Watt and Watt 1992). Here a subject's arm movements were recorded and represented in quaternion space. A boundary between valid and invalid rotations of the arm was then defined on the surface of the unit sphere in quaternion space. Iterative approaches were then employed to resolve invalid joint configurations.

Artificial neural networks are inspired by the structure of the human brain. Like biological neural networks they are composed of

neurons which are linked together to form complex networks. However, they are significantly different in terms of complexity and the way nodes in the network communicate. There are many types of network architecture, from auto-associative memories such as the Hopfield network to unsupervised networks such as Kohonen's SOM (Self-Organising Map)(Mehrotra, et al. 1997). The most popular type of architecture is the feed-forward network such as the Multi-layer Perception. These are trained to give certain outputs in response to given inputs by repeatedly adjusting the strengths of the interconnections between neurons within the network. Typically, neural networks use an optimization process to learn the best boundary to delineate regions within a multi-dimensional feature space. Recent developments have introduced the use of genetic algorithms to find the optimum network configuration and topology for a given network (Huber, et al. 1995).

EXPERIMENTS

This paper describes the application of genetic algorithm approaches to neural network training in order to model piece-wise linear / discontinuous functions that approximate the behaviour of anatomically correct joint constraints.

We trained a Generalised Multi-Layer Perceptron (GMLP) model to learn joint behaviour in one (Euler angle), two and three dimensions. Each point in the feature space represents the rotation of a given joint model. For each point, we want to model the appropriate correction to map a given joint configuration to the nearest valid joint configuration. So for valid rotations, there is no correction, while for invalid rotations, a vector is stored to move the rotation to the nearest valid rotation. Discontinuities arise at the joint constraint boundary where the valid and invalid joint configurations meet. A range of experiments were undertaken to model different rotational constraint sizes in 1, 2 and 3 dimensions.

The NetJEN system used in our work is a Java based application which grew out of NetGEN (Huber, et al. 1995) developed for research purposes at the University of Salzburg. NetJEN boasts several impressive features and provides an implementation of Huber et al's (Huber, et al. 1995) work in topology evolution. A hybrid system is employed using genetic algorithms with the back-propagation learning algorithm.

In each experiment the network was configured as follows. The input layer represents the current joint rotation, while the output layer represents the correction vector. The number of hidden

nodes and connection topology are randomized and then evolved during the learning process using Generic Algorithms. The weights of the interconnections are randomized and updated using the back-propagation algorithm. In each case the inputs and outputs were mapped to the range -1 to +1, the evolution and training parameters were set as shown in Table 1. We restricted the number of generations and training cycles to reduce training times. Each experiment was repeated five times to ensure the consistency of the results.

Three datasets were prepared for each of the experiments; a training set, used to train each generation of ANN, a validation set, used to assess the fitness of the ANN for genetic selection and a test set which provided an unseen set of data on which to test the ANN. In creating the datasets we aimed to cluster patterns around the boundary representing the discontinuity between the valid and invalid joint regions.

Table 1 : Evolution and Training Settings

Parameter	Description	Setting
Regularization function	Secondary fitness function.	Number of links
Hidden Nodes	Maximum no. of hidden nodes.	20
Number of Generations	No. of generations over which the ANN were evolved.	50
Population Size	Size of the populations evolved.	20
Fitness Function	Primary fitness function.	Inverse SSE
Evolve number of Links	Networks are pruned down from fully connected networks.	On
Evolve number of Hidden Nodes	Evolves the no. of hidden nodes.	On
Evolve number of training cycles	Evolves the no. of training cycles	On
Learning Rate	Learning rate used when training the ANN.	0.1
Stopping Error	MSE at which the ANN are stopped.	0.001
Training Function	Training function used to train the weights of the ANN.	Resilient back-propagation
Max Epochs	Maximum number of training epochs	500

RESULTS

The results of the first experiment show that though each of the networks trained successfully and the Mean Squared Error (MSE) of the network was low, the

performance of the network decreased as the number of dimensions increased, demonstrated by the increase in the MSE, (figure1.) In each case the functions describe both continuous and discontinuous regions. These are of comparable size so the decrease in accuracy is proportional to the number of degrees of freedom being modelled. There was little difference in the performance on test and training sets, suggesting the network performed well on unseen patterns.

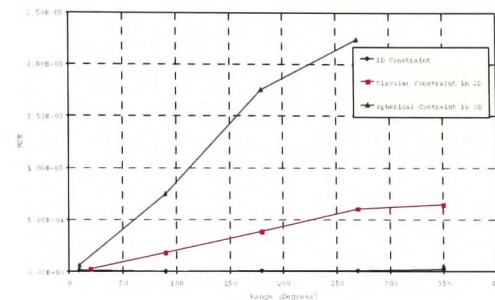


Figure 1: Average MSE vs. Constraint Size

The number of hidden nodes and the number of inter-connections, which are to a certain extent linked, also increased as the number of dimensions increased, though only between the one and two dimension constraints. This indicates that the number of inter-connections and nodes required to approximate a constraint in two dimensions was sufficient also to approximate a constraint in three dimensions.

The second experiment varied the size of the constraint. In each case the network performed well. However, it was noted that in each case the network performance decreased as the size of the constrained region increased, (figure 1.) For each of the ranges tested the size of the evolved networks varied little. The increase in the MSE in relation to the size of the constraint can be attributed to the distribution of training patterns. This shows that the technique is applicable to various sizes of constrained region, which is important if we are to model anatomical joints.

CONCLUSION AND FUTURE WORK

Our results show that a Generalised Multi-Layer Perceptron (GMLP) with evolved structure can model a corrective joint function in 1, 2 and 3 dimensions to a reasonable degree of accuracy. Experimental results are encouraging with regards to the use of such networks for neural network constraint modelling. Through experimentation, we found that the best results were obtained when the network evolved a hidden layer with sigmoid transfer functions and an output layer with linear transfer functions.

Future work will look at applying this network architecture to more general rotation models using the quaternion representation.

ACKNOWLEDGEMENTS

The authors would like to thank; Helmut Mayer (University of Salzburg) for all his advice in optimizing the configuration of the NetJEN system; August Mayer (University of Salzburg) for his advice and continued development and support of the NetJEN system and Carl Davies (ISELS – University of Glamorgan) for additional hardware and technical support.

REFERENCES

- Badler, N. I.; C. B. Phillips and B. L. Webber. 1993. *Simulating Humans: Computer Graphics, Animation, and Control.*, Oxford University Press, Oxford.
- Baerlocher, P. and R. Boulic. 2000. "Parameterization and Range of Motion of the Ball and Socket Joint" In IFIP TC5/WG5.10 DEFORM' 2000 Workshop and AVATARS' 2000 Workshop on Deformable Avatars
- Baerlocher, P. 2001. "Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures". PhD thesis from Department D'Informatique, Ecole Polytechnique Federal De Lausanne, Lausanne
- Baerlocher, P. and R. Boulic. 2004. "An Inverse Kinematics Architecture Enforcing an Arbitrary Number of Strict Priority Levels" *The Visual Computer: International Journal of Computer Graphics*, 20, No 6, 402-217.
- Ding, H. and S. K. Tso. 1999. "A Fully Neural Network-Based Planning Scheme for Torque Minimization of Redundant Manipulators" *IEEE Transactions on Industrial Electronics*, 46, No 1, 199-206.
- Ding, H. and J. Wang. 1999. "Recurrent Neural Networks for Minimum Infinity-Norm Kinematic Control of Redundant Manipulators" *IEEE Transactions on System, Man and Cybernetics: Part A*, 29, No 3, 269-276.
- D'Souza, A.; V. S. and S. Stefan. 2001. "Learning Inverse Kinematics" In *International Conference on Intelligent Robots and System*, (Maui, Hawaii, USA)
- Eng, J. and D. A. Winter. 1995. "Kinematic Analysis of the Lower Limbs During Walking: What Information Can Be Gained from a 3d Model" *Journal of Biomechanics*, 28, No 6, 753-758.
- Engin, A. E. and S. T. Tumer. 1989. "Three Dimensional Kinematic Modelling of the Human Shoulder Complex Part 1: Physical Model & Determination of Joint Sinus Cone." *Journal of Biomechanical Engineering*, 111, 107-112.
- Engin, A. E. and S. T. Tumer. 1993. "Improved Dynamic Model of Human Knee Joint and Its Response to Loading" *Journal of Biomechanical Engineering*, 115, No 2, 137-142.
- Faraway, J. J.; X. Zhang and D. B. Chaffin. 1999. "Rectifying Postures Reconstructed from Joint Angles to Meet Constraints" *Journal of Biomechanics*, 32, No 7, 733-736.
- Feikes, J. D.; J. J. O'Connor and A. B. Zavatsky. 2003. "A Constraint-Based Approach to Modelling the Mobility of the Human Knee Joint" *Journal of Biomechanics*, 36, No 1, 125-129.
- Furuta, T.; T. Tawara; Y. Okumura; M. Shimizu and K. Tomiyama. 2001. "Design and Construction of a Series of Compact Humanoid Robots & Development of Bipedal Walking Control Strategies" *Robotics and Autonomous Systems*, 37, 81-100.
- Guez, A. and Z. Ahmad. 1988. "Solution to the Inverse Problem in Robotics by Neural Network" In *IEEE International Conference on Neural Networks*, 2, (San Diego, CA), 617-624
- Hamel, A. J.; N. A. Sharkey; F. L. Buczek and J. Michelson. 2003. "Relative Motions of the Tibia, Talus and Calcaneous During the Stance Phase of Gait" *Gait and Posture*, 20, No 2, 153-157.
- Herda, L.; R. Urtasun; P. Fua and A. Hanson. 2003. "Automatic Determination of Shoulder Joint Limits Using Quaternion Field Boundries" *International Journal of Robotics Research*, 22, No 6, 419-444.
- Huber, R.; H. A. Mayer and R. Schwaiger. 1995. "Netgen - a Parallel System Generating Problem-Adapted Topologies of Artificial Neural Networks by Means of Genetic Algorithms" In *Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3*, (Dortmund), 91-98
- Korein, J. U. 1984. *A Geometric Investigation of Reach*, MIT Press, Massachusetts.
- Madhavapeddy, N. and S. Ferguson. 1998. "Specialised Constraints for an Inverse

Kinematics Animation System Applied to Articulated Figures" In Eurographics'98, (Leeds, United Kingdom), 215-223

Manal, K.; X. Lu; M. K. Nieuwenhuis; P. J. M. Helders and T. S. Buchanan. 2002. "Force Transmission through the Juvenile Idiopathic Arthritic Wrist: A Novel Approach Using a Sliding Rigid Body Spring" Journal of Biomechanics, 35, 203-218.

Manurel, W. and D. Thalmann. 2000. "Human Shoulder Joint Modelling Including Scapulo-Thoracic Constraints and Joint Sinus Cones." Computers and Graphics, 24, 203-218.

Mehrotra, K.; C. K. Mohan and S. Ranka. 1997. Elements of Artificial Neural Networks, MIT Press, Massachusetts.

Nelson, D. 1988. "Constraint Jacobians for Constant-Time Inverse Kinematics and Assembly Optermization". Report No. University of Utah, Utah

Selmic, R. R. and L. L. Lewis. 2000. "Deadzone Compensation in Motion Control Systems Using Neural Networks" IEEE Transactions on Automatic Control, 45, No 4, 602-613.

Wang, J. 1997. "Recurrent Neural Networks for Computng Pseudoinverses of Rank-Deficient Matrices" SIAM Journal of Scientific Computing, 18, No 5, 1479-1493.

Watt, A. and M. Watt. 1992. Advanced Animation and Rendering Techniques, ACM Press, New York.

Xia, Y. and J. Wang. 2001. "A Dual Neural Network for Kinematic Control of Redundant Robot Manipulators" IEEE Transactions on System, Man and Cybernetics: Part B, 31, No 1, 147-154.

Yang, D. N.; D. N. Condie; M. H. Granat; J. P. Paul and D. I. Rowley. 1996. "Effects of Joint Motion Constraints on the Gait of Normal Subjects and Their Implications on the Further Developments of Hybrid Fez Orthosis for Paraplegic Persons." Journal of Biomechanics, 29, No 2, 217-226.

Zajac, F. E. 2003. "Biomechanics and Muscle Coordination of Human Walking Part 2: Lessons from Dynamical Simulations and Clinical Implications" Gait and Posture, 17, 1-17.

Zhang, Y. and J. Wang. 2002. "A Dual Neural Network for Constrained Torque Optimization of Kinematically Redundant Manipulators" IEEE Transactions on System, Man and Cybernetics: Part B, 32, No 5, 654-662.

Zhang, Y.; J. Wang and Y. Xu. 2002. "A Dual Neural Network for Bi-Criteria Kinematic Control of Redundant Manipulators" IEEE Transactions on Robotics and Automation, 18, No 6, 923-931.

Zhang, Y.; J. Wang and Y. Xia. 2003. "A Dual Neural Network for Redundancy Resolution of Kinematically Redundant Manipulators Subject to Joint Limits and Joint Velocity Limits" IEEE Transactions on Neural Networks, 14, No 3, 658-667.

Zhao, J. and N. I. Badler. 1994. "Inverse Kinematics Positioning Using Non-Linear Programming for Highly Articulated Figures" Transactions on Graphics, 14, No 4, 313-336.

AUTHOR BIOGRAPHIES

Mr. Glenn Jenkins is a PhD Research Student at the University of Glamorgan, studying towards a PhD in Computer Science focusing on the Simulation of Anatomical Joint Constraints using Neural Networks. His research interests include artificial neural network evolution, spline and mixed activation function artificial neural networks, the simulation of anatomical joint constraints and their implementation.

Dr. Paul Angel is a Senior Lecturer in the School of Computing at the University of Glamorgan, specialising in Computer Graphics, Visualisation and software design. His research interests include volumetric modelling and visualisation, image analysis, artificial neural networks and concurrent / parallel programming techniques. He obtained his PhD in Computer Science focusing on the application of wavelet feature extraction techniques applied to biological image data.