Out-of-Sample Mapping of a Two-Link Robotic Manipulator

Ryan Finn
Mechanical Engineering
University of New Brunswick
Fredericton, New Brunswick E3B 5A3
Email: ryan.finn@unb.ca

Dr. Rickey Dubay
Mechanical Engineering
University of New Brunswick
Fredericton, New Brunswick E3B 5A3
Email: dubayr@unb.ca

***Abstract*—An approach to reduce dimension for a set of Multiple-Input-Multiple-Output (MIMO) non-linear system equations is tested in simulation. The methodology is implemented on a vertical two-link robotic manipulator. A manifold learning (ML) algorithm is used to reduce the dimension of the system. Finally, an artificial neural network (ANN) routine is trained to act as an out-of-sample (OoS) extension algorithm, and is used to map existing and newly generated points. The initial results of the study show promise in the application of ML routines for system identification (SI) on a reduced data set and eventually application to control setups.**

I. Introduction

SI is a set of methodologies used to mathematically represent a dynamic system. For linear processes, the field of SI is considered mature with many modelling methodologies available. The most common being least-squares (LS) optimization. This method finds input/output coefficients to fit various model formats such as ARX (Auto-Regressive with eXogenous input) and ARMAX (Auto-Regressive Moving Average with eXogenous input) and CARIMA used in general predictive control [1], [2]. For practical plant identification, an online recursive method is required to update the dynamic model as the process changes over time. Methods such as recursive least-squares (RLS) can be used to learn input/output coefficients of relatively low-order systems. When the plant encountered is non-linear, pre-processing of the data may be required before methods like RLS are considered.

When complex datasets or highly non-linear systems are encountered, deriving accurate models from linear statistical techniques is very difficult. Linear methods fail to identify hidden non-linear structures that could exist in the dataset [3] (ie., swiss roll or torus embedded in 3D-Euclidean space). The application of non-linear techniques might find some hidden non-linear structure of relatively low-order, but when the process grows in dimensions so does the effort in finding any kind of relevant structure. This is referred to as the curse of dimensionality (CoD) [4]. Currently, the development of non-linear modelling methodologies is an active area of SI [5].

SI plays an important role in the field of model-based controls which lend themselves well to industrial big data trends. Processing and mining large amounts of data are important and relevant in today's industry. In an effort to realize a computational efficient and accurate identification process, the function of pre-processing complex datasets to a lower dimension representation is seen to be an important first step towards practical application in the field of SI [6]. ML techniques are used to lower the dimensionality of complex non-linear data [7]. Applications of ML can be found in a variety of disciplines such as image processing, machine learning, and computer vision [8]. Preliminary theory has been explored in [9]–[11]. These suggest that the application of ML in SI is a natural progression.

In this study, the preliminary feasibility of applying ML methods together with the future intent of closed-loop control is tested on a non-linear gain surface generated by a simulation of a vertical two-link robotic manipulator. To this end, an ANN was trained as an OoS mapping tool on the normalized gain surface produced by running a simulation through randomly generated angles for link 1 and link 2. The scope of this research is limited to the validation of the proposed methodology and will only consider the non-linear gain surface for the first manipulator link. Basic theory on SI, ML, ANN, and two-link manipulators will be presented. The mapping accuracy of the OoS is also presented. Future direction of research is proposed.

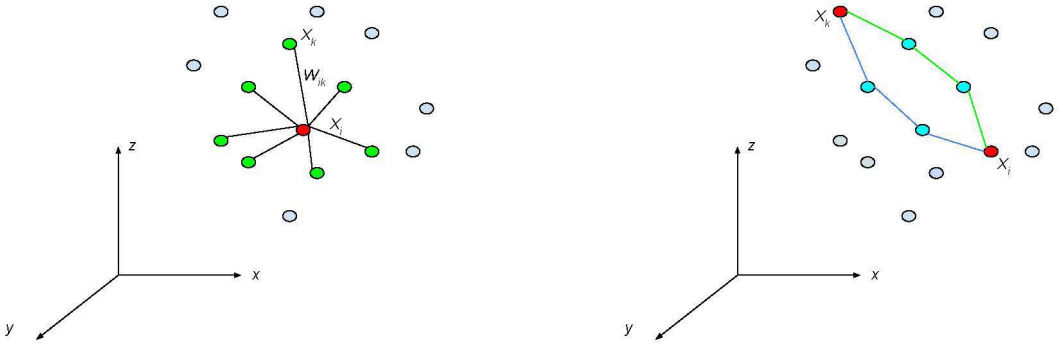
II. Manifold and Mapping Techniques

The following sections detail theory and tools used in developing a preliminary OoS mapping algorithm for the non-linear gain surface. In this section the concept of ML techniques are presented. An ANN is used to act as an OoS extension and is described in subsection II-C. The methodology is tested in simulation using a two-link manipulator model which will also be discussed.

A. Manifold Learning

In the context discussed here, some of the first ML algorithms developed are ISOMAP (isometric feature mapping) [12] and local linear embedding (LLE) [13]. Examples of ML routines other then the one used here are Laplacian eigenmaps, Hessian eigenmaps, and local tangent space alignment. These are extensions of ISOMAP and LLE [3]. Calculation philosophies for ISOMAP and LLE are detailed next.

978-1-5386-3664-0/18/$31.00 ©2018 IEEE



(a) Local Linear Embedding (b) ISOMAP

Fig. 1. Neighbourhood Graph Construction

1) Manifold Learning Process: All ML algorithm procedures are comprised of three steps. Steps 1) & 3) are generic for all ML algorithm while step 2) differs. The differences resides in the setup of the weight graph for the embedding step. This procedure is laid out with more information in [3].

1) A weighted graph, W_{ij}, is constructed based on neighbourhood information between data points.
2) The weighted graph from step one is transformed in a manner described by the selected ML. The transform is required to make W_{ij} suitable for the embedding step.
3) An embedding step, that involves an eigen analysis, attempts to find a lower dimension coordinate system based on calculations preformed from the first two steps.

ISOMAP estimates the true geodesic distance between all points in the dataset. In other words, ISOMAP estimates the shortest distance between each data point in the dataset while LLE considers the Euclidean distances around a neighbourhood of points [3]. Examples of neighbourhood data points for the ML algorithms are shown in Fig. 1a and Fig. 1b for LLE and ISOMAP respectively.

In Fig. 1a, the red data point is chosen as the subject for iteration i. A variable K representing how many neighbours are in the local neighbourhood is determined. The distances to each point are then multiplied by their own respective linear weight coefficients W_{ik}. Now, $\vec{X_i}$ is a linear combination of all points in the subjected data point neighbourhood. In Fig. 1b, the geodesic distance or shortest path, is calculated from all paths between all points within a neighbourhood. This neighbourhood can be described by a radius parameter ϵ or a number of neighbours K. For this study, LLE was chosen. A dataset was obtained in simulation with a vertical two-link manipulator model. In literature, it has been shown that a local approach such as LLE is better suited for data points that are well distributed over a manifold surface [3].

2) Local Linear Embedding: The Swiss roll is a dataset for testing ML algorithms and is shown in Fig. 2a. The three dimensional data set is reduced to two dimensions using LLE. This example is commonly used to show how ML methods can map higher dimension data to a lower dimensional embedding which preserves the local geometry between neighbours. The reduced dimension or unrolled data is shown in Fig. 2b. All graphs simulated/processed were all generated in the programming language Python, using code repositories SciKit and SciPy [14], [15].

In order to implement LLE, a randomly sampled data set that is large enough to generate a smooth manifold surface M is required. This M is populated with N real-valued vectors $\vec{X_i}$ of dimension D. A cost function $\epsilon(W)$, is then minimized which is defined in (1) [13].

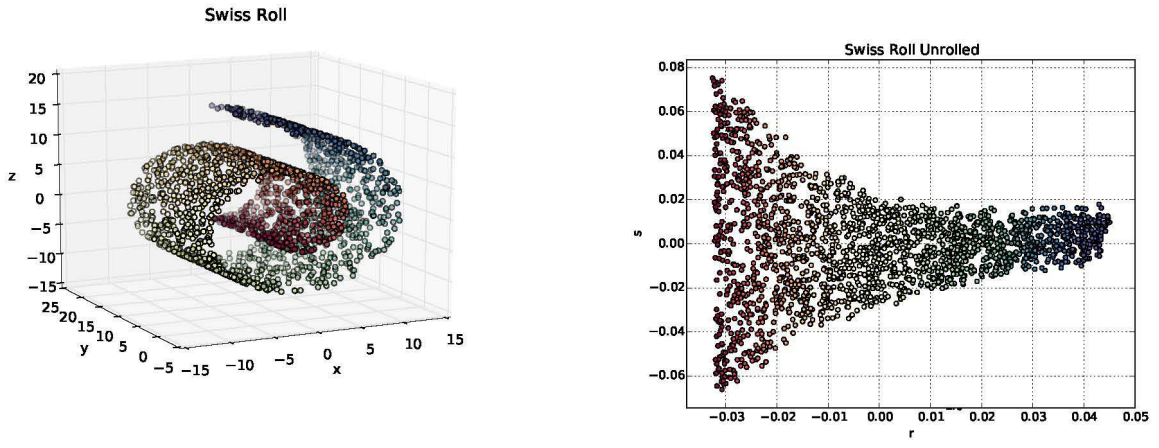
$$\epsilon(W) = \sum_{i=1}^{N} \left| \vec{X_i} - \sum_{j=1}^{N} W_{ij} \vec{X_j} \right|^2 \tag{1}$$

This is a least-squares optimization problem with the constraints in (2), for the solution of associated weight matrix W_{ij}.

$$\begin{cases} \sum_{j=1}^{N} W_{ij} = 1 \\ W_{ij} = 0 \text{ if } \vec{X_j} \text{ is } \notin \text{ the region of } K \end{cases} \tag{2}$$

These constraints ensures that the mapping is invariant to rotations, amplitude scaling, and axis translations when the least-squares minimization is computed. The sum-to-one restriction of the rows keeps axis translations invariant and since the weight matrix W is symmetric, the neighbourhood information remains intact [13].

The final step is an embedding formulation. This is done by solving a second least-squares constrained optimization shown in (3), and an eigenvalue problem [9]. The new generated global coordinate system centred about the origin contains all mapped $\vec{X_i}$ of dimension D to points $\vec{Y_i}$ of dimension d, where $D << d$.



(a) Swiss Roll (b) Unrolled with LLE

Fig. 2. Swiss Roll Unrolled via LLE

$$\Phi(Y) = \sum_{i=1}^{N} \left| \vec{Y_i} - \sum_{j=1}^{N} W_{ij} \vec{Y_j} \right|^2 \tag{3}$$

The constraints applied to (3) are represented in (4). The first constraint keeps the new coordinate system anchored at the origin. The second constraint is evoked to remove the possibility of degenerate solutions [13].

$$\begin{cases} \sum_{j=1}^{N} \vec{Y_i} = 0 \\ \frac{1}{N} \sum_{i=1}^{N} \vec{Y_i} \vec{Y_i^T} = I \end{cases} \tag{4}$$

In this study, a dataset similar to the Swiss roll structure was generated in simulation with a mathematical model of a vertical two-link manipulator.

B. Vertical Two-Link Manipulator

A vertical two-link manipulator simulation was used to test an OoS extension neural network algorithm for mapping new points to the lower dimensional space $(\vec{Y_i}, d)$. A basic diagram of the test apparatus can be seen in Fig. 3 with model parameters found in [16]. Experimental testing on the described manipulator is saved for future study.

The governing equation to generate simulation data to achieve a non-linear gain surface for this manipulator is in terms of angular acceleration, $\vec{\alpha}$, and is described by (5).

$$\vec{\alpha} = \mathbf{M}^{-1}(\vec{\tau} - \vec{C} - \vec{B} - \vec{G}) \tag{5}$$

Where matrix $\mathbf{M}$ is the inertia matrix, $\vec{\tau}$ is the torque generated by the motors, $\vec{C}$ represents the Coriolis forces in the spinning links, $\vec{B}$ is the torques required to overcome friction in belts and bearings, and $\vec{G}$ represents the torques

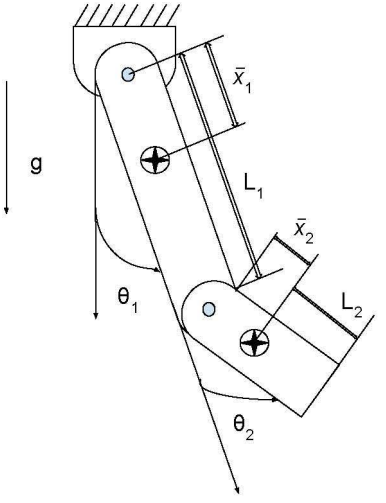


Fig. 3. Vertical Two-Link Manipulator

generated by gravity [16]. The control actions are the inputs of the system/model and are the voltages U_1 and U_2 that actuate the motors of each link of the manipulator. Equation (6) is used to solve $\vec{\tau}$.

$$\vec{\tau} = \begin{bmatrix} V_{f1} U_1 \\ V_{f2} U_2 \end{bmatrix} \tag{6}$$

The system responses for the robot manipulator were simulated with random input voltages. Steady-state angular position values of the first link, denoted as θ_1^{ss}, were recorded. Steady-states θ_1^{ss} in the simulations discussed here are the angular position values that the manipulator links remain at indefinitely given a set of voltages, U_1 and U_2. This is detailed in section III-A.

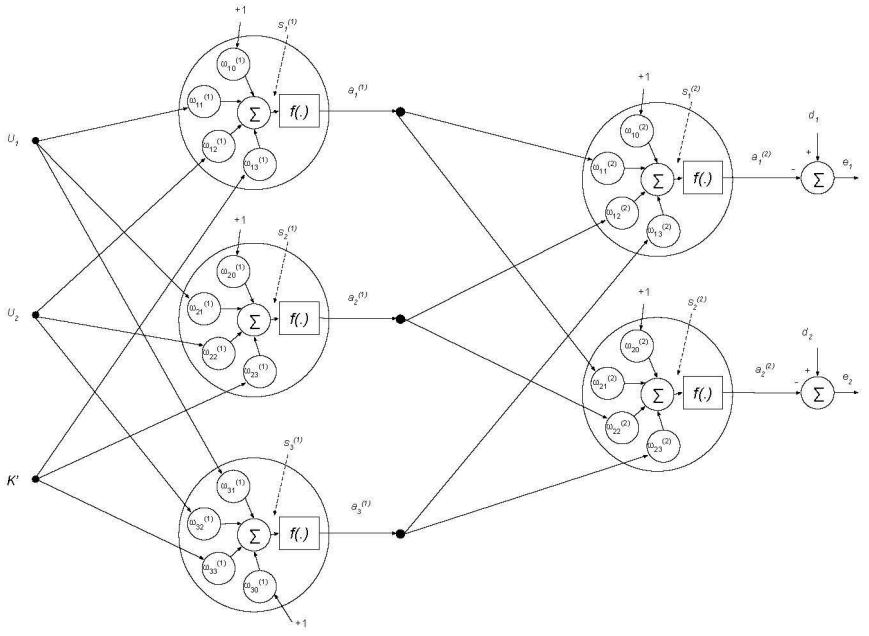


Fig. 4. Neural Network Structure

C. Neural Network

A basic ANN was developed and is shown in Fig. 4. The variables U_1 and U_2 are input voltages to the two-link manipulator detailed in II-B. The third input is K', the normalized gain value reached after all transient behaviours become negligible. Normalized gain is detailed in section III-A. The two outputs of the ANN represent the new global coordinate system, (r, s), generated from LLE on the two-link manipulator. For this study the ANN output dimension is $d = 2$. When applying ANN algorithms it is important to consider training. There are numerous ANN training algorithms available that are based on local and global optimization methods. Backpropagation (BP) is used to train the ANN for this problem.

The ANN structure adopted in Fig. 4 is constructed with a variable hidden layer. It is important to note that there are various configurations of ANN architectures that span a number of research avenues. In general an ANN structure is chosen by a designer based on problem dependent requirements. Other types of ANN include feed-forward, radial basic function, and recurrent networks. Each different design of neural network differs in internal connections and performance features. For example, recurrent neural networks have internal memory, self-feedback connections, and time delay blocks. This allows data to be stored resulting in a network able to identify dependencies [17].

1) Backpropagation Training: When training with BP, weights are updated in a patternwise fashion or in an epochwise fashion. Patternwise updates are done iteratively by propagating the error back through the network and updating all weights on each pattern presentation. Epochwise updates are conducted by presenting every pattern in your training set and accumulating the errors. This allows just one global weight update after N' pattern presentations. Epochwise updates offer a smoother convergence then patternwise updates. In this case, epochwise BP is used to train the ANN to act as an OoS extension for the ML LLE routine. To begin training, the first input pattern $(U_{(1,i)}, U_{(2,i)}, K'_{(i)})$ is sent through the ANN, resulting in an output (r, s) which is then backpropagated through the ANN.

The average-squared error $\xi(W)$ seen in (7), is the error criteria for minimizing the weights of the ANN structure in the epochwise fashion. Where N'_i is the number of patterns in the training set, and p represents the particular pattern being presented at iteration i. As an example, the update equation will be derived for the 1^{st} output node weight, $\omega_{11}^{(2)}$, shown in Fig. 5.

$$\xi(W) = \frac{1}{N'_i} \sum_{p=1}^{N'_i} \sum_{m=1}^{2} \left[e_m^{(p)}(W) \right]^2 \tag{7}$$

The error is the difference between the desired response vector $\vec{d}$ and the output of the network. See relation (8).

$$\vec{e} = \begin{bmatrix} d_1 - a_1^{(2)} \\ d_2 - a_2^{(2)} \end{bmatrix} \tag{8}$$

Considering weight $\omega_{11}^{(2)}$, the $\nabla\xi(W)$ over the change in weight $\omega_{11}^{(2)}$ is determined. More specifically in (9), the partial derivative wrt. each signal operation is preformed.

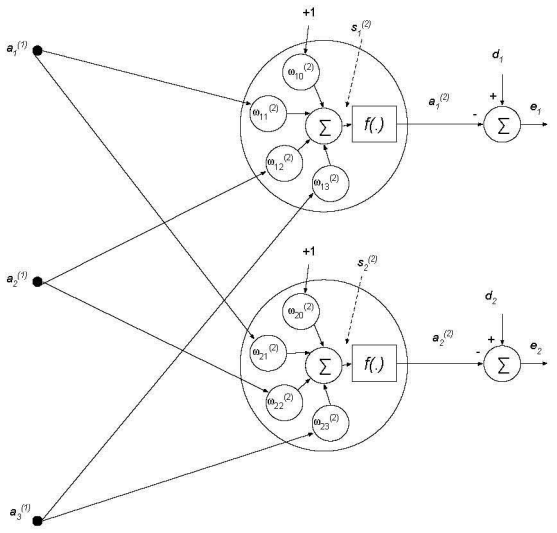


Fig. 5. Backpropagation - Outer layer

$$\frac{\partial \xi}{\partial \omega_{11}^{(2)}} = \frac{\partial \xi}{\partial e_1}\frac{\partial e_1}{\partial a_1^{(2)}}\frac{\partial a_1^{(2)}}{\partial s_1^{(2)}}\frac{\partial s_1^{(2)}}{\partial \omega_{11}^{(2)}} \tag{9}$$

The value for $\frac{\partial \xi}{\partial e_1}$ is derived to be e_1, and is obtained from (7). The term $\frac{\partial e_1}{\partial a_1^{(2)}}$ is derived from (8) as (-1). The third term is the signal $s_1^{(2)}$ passed through a non-linear activation function and its value is denoted as $f'(s_1^{(2)})$.

The 1^{st} node in the output layer is then associated with $\delta_1^{(2)}$ as described in (10), where $\delta_1^{(2)}$ is a placeholder . All other nodes are processed analogously. When updating the weights in a particular output node, the δ values for each corresponding node remains constant.

$$\frac{\partial \xi}{\partial e_1}\frac{\partial e_1}{\partial a_1^{(2)}}\frac{\partial a_1^{(2)}}{\partial s_1^{(2)}} = e_1(-1)f'(s_1^{(2)}) = \delta_1^{(2)} \tag{10}$$

The final term $\frac{\partial s_1^{(2)}}{\partial \omega_{11}^{(2)}}$ from (10) is then multiplied by $a_1^{(1)}$ as shown in Fig. 5. All other weight updates are done in a similar fashion in the output layer.

III. Results

Simulation results carried out on the vertical two-link manipulator described in section II-B are presented here. The simulations were carried out using Euler's method with a time step of $T_s = 0.01\ sec$. Input voltages U_1 and U_2 were randomly generated to actuate the two-link manipulator in various configurations. Each of the N tests were simulated until a steady-state value for θ_1^{ss} was reached. Initial conditions were reset to zero for each new trail preformed. This θ_1^{ss} value was then recorded along with the associated inputs to give $(U_1, U_2, \theta_1^{ss})$. In this preliminary investigation on the MIMO two-link manipulator, the dynamics of the first link were simulated, θ_1. The dynamics of the second link can be determined similarly in order to implement a control algorithm, this is saved for future study.

A. Two-Link Manipulator Simulation

An important SI parameter when considering the dynamics of a system is the gain denoted K'. The K' for a MIMO system is a function of all inputs for each output. The manifold surface created here is an expression of K' for the first link of the manipulator based on both U_1 and U_2. In order to calculate K' already seen, the virtual system was excited and monitored for each random combination of U_1 and U_2 until θ_1^{ss} was reached. This value was normalized to obtain K' as shown in (11).

$$K' = \frac{\theta_1^{ss}}{U_1 + U_2} \tag{11}$$

Recording and plotting the value of K' for all tests generated the manifold surface seen in Fig. 6. To ensure a well populated manifold surface a total of 7500 tests were conducted, $N = 7500$.

In Fig. 6 clusters of data points are shown outside of the stable operating range of the manipulator, this is where link one reaches $\theta_1 = \pi\ rads$. When only stable actuating voltages are considered, $\{-3 \leq (U_1, U_2) \leq 3\}$, the manifold surface representing the operating region of interest is shown in Fig. 7.

The manifold surface shown in Fig. 6 is limited to a range of $[-3.0 \leq K' \leq 3.0]$, this results in a more uniform spread of the majority of the data points represented in Fig. 8. It is common practice to limit the manifold space size before processing it [14]. This is the surface that will be subjected to LLE.

B. Local Linear Embedding Simulation

To reduce the manifold surface shown in Fig. 8 to a two dimensional data set, LLE was implemented. When applying

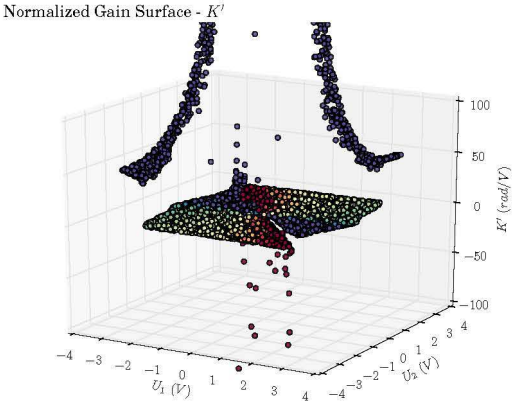


Fig. 6. Normalized Gain Surface - Regions of Instability

LLE, a neighbourhood size K must be selected. This is a problem dependent tuning parameter. Different values of K were attempted in order to get a reconstruction error as close to zero as possible. See Figs. 9 and 10 of LLE results for different values of K.

From these figures, greater values of K generate a more distributed data plot, which suggest a more accurate mapping. For $K = 5$, most points are clustered together in the new global coordinate system (r, s). In the cases of $K = 30$ and $K = 50$, a more uniform mapping exists. A value of $K = 30$ was chosen as the training data set for the ANN. Using $K = 50$ is computationally more expensive with insignificant improvement for (r, s).

C. Neural Network Training

An ANN was trained using a normalized gain surface K' generated from angular position data, θ_1, of a vertical two-link manipulator. Open-loop tests through randomly generated voltages (U_1, U_2) were conducted until $\theta_1 \to \theta_1^{ss}$ (angular position steady-state). A total of 7500 tests where used to construct the dataset consisting of $(\vec{U}_1, \vec{U}_2, \vec{\theta}_1^{ss})$. Finally, each $\theta_{(1,i)}^{ss}$ was normalized as seen in relation (11) to extract the gain K_i' for corresponding values of $(U_{(1,i)}, U_{(2,i)})$. This resulted in a dataset $(\vec{U}_1, \vec{U}_2, \vec{K}')$.

From the $N = 7500$ trials that make up the dataset, $N' = 5835$ elements existing in the RoS were used to train the ANN. New patterns were then sent through to test the mapping capabilities of the trained ANN. Therefore, the training set $TR\{\cdot\}$ will be denoted as seen in relation (12).

$$TR \doteq \left\{ (\vec{U}_{(1,i)}, \vec{U}_{(2,i)}, \vec{K}'_{(i)}) \right\}_{i=1,2,\cdots,N_i'} \tag{12}$$

The ANN considered here uses a non-linear, $\tanh(\cdot)$, activation function in the hidden layer and linear activation units in the output layer. Due to the $\tanh(\cdot)$ functions, the

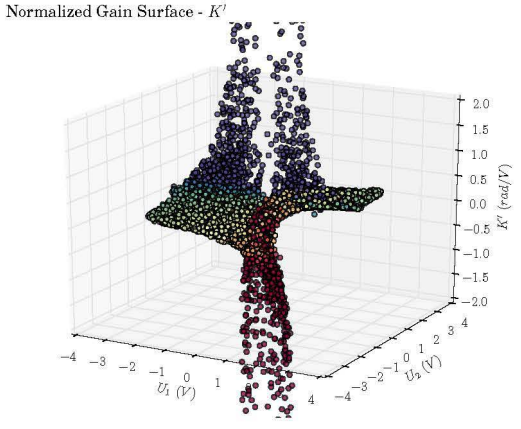


Fig. 7. Normalized Gain Surface - Link Output 1

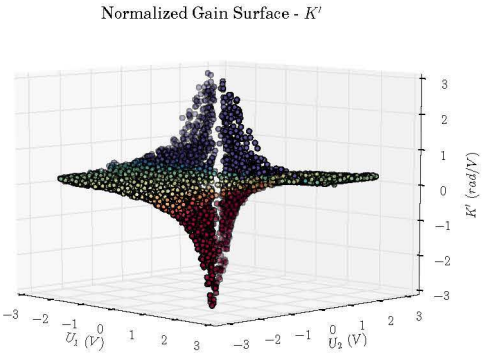


Fig. 8. Normalized Gain Surface - Region of Stability

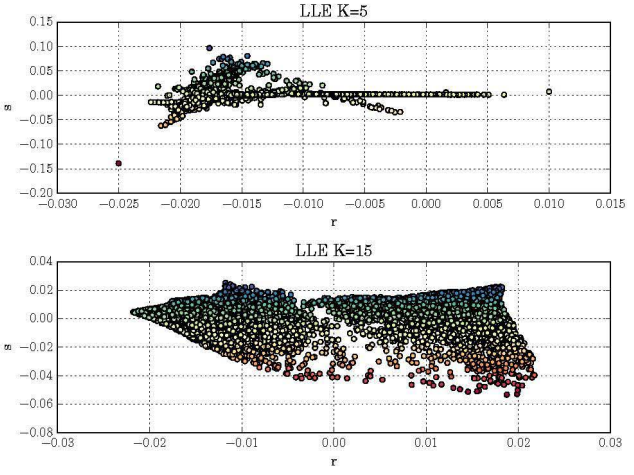


Fig. 9. Local Linear Embedding - $K = 5$ & $K = 15$

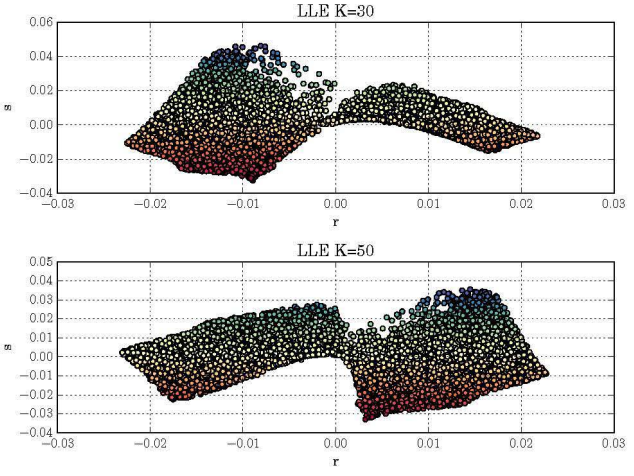


Fig. 10. Local Linear Embedding - $K = 30$ & $K = 50$

training and test data used to train the ANN, $(\vec{U}_1, \vec{U}_2)$, had to be normalized to the interval [-1,1]. This normalization was computed with repository [14], with implementation for the required quantities listed as Equ. (13) on $\vec{U}_1$.

$$\vec{U}_1' = \frac{\vec{U}_1}{\left| \max_{1 \le i \le N'} \vec{U}_{(1,i)} \right|} \tag{13}$$

Now for $\vec{U}_2$ in relation (14).

$$\vec{U}_2' = \frac{\vec{U}_2}{\left| \max_{1 \le i \le N'} \vec{U}_{(2,i)} \right|} \tag{14}$$

Once the normalized training set $\left\{ (\vec{U}_1', \vec{U}_2', \vec{K}') \right\}$ was achieved, the ANN was trained with various input configurations to determine the amount of hidden layers required. A hidden layer size consisting of 150 neurons was found to be the configuration needed for mapping to the required coordinate range (r, s) as seen before in Fig. 10. Other ANN structures were attempted but higher node counts did not result in improved mappings. A mapping generated from the trained ANN can be seen in Fig. 11. The general shape and scale of the plot in Fig. 10 can be seen in Fig. 11.

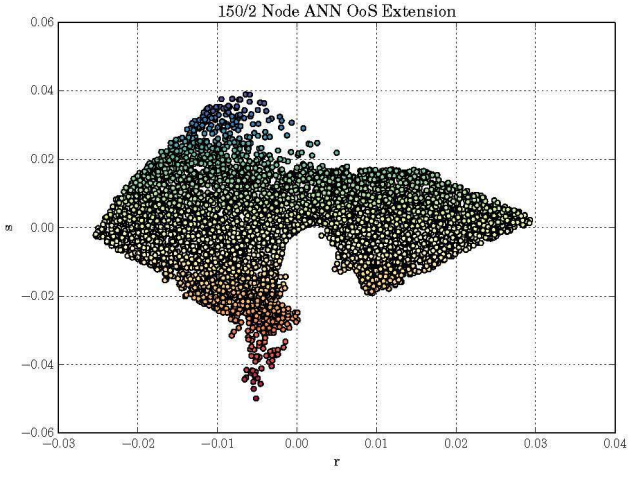


Fig. 11. Trained ANN - Training Set

D. Output Mapping Results

Next, mapping from a new set of randomly generated data obtained from the two-link manipulator simulation was used to test the ANN. A hidden layer of 150 neurons was implemented to evaluate the performance of the mapping for OoS test samples. The goal is to match the ANN output shape from Fig. 11.

Mapping with the new dataset presents similar shape and range to the training plot seen in Fig. 11. This shows that it is possible to use a ANN to act as an OoS extension algorithm. This shows promise for using this methodology for system identification with more refinement required.

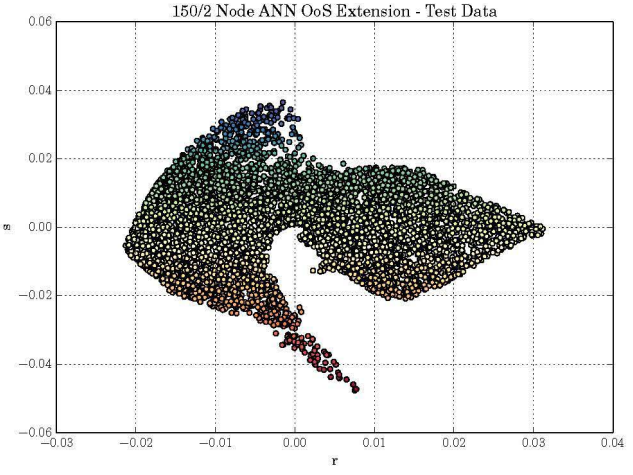


Fig. 12. ANN Mapping - Test Set

IV. Conclusion

In this study, an ANN was constructed to act as an OoS extension algorithm for a ML mapping function. Results from training the ANN show that this methodology provides the possibility to aid in the analysis of non-linear system data being read through mediums such as plant sensors. The implementation of the ML routine reduces dimensionality of complex non-linear problems, which in turn, reduces processing time and lends itself well to online processing on the reduced dimensional dataset. The initial investigation did not consider noise or extend the ANN to system identification with the reduced dataset. These are saved for future study. Different ANN methodologies could also improve the results obtained with this general methodology.

References

[1] E. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed. Springer 2007

[2] J. Juang, *Applied System Identification*, 1st ed. Prentice Hall PTR 1994

[3] A. Izenman, *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*, 1st ed. Springer 2013

[4] H. Ohlsson, J. Roll, T. Glad, and L. Ljung, *Using Manifold Learning for Non-linear System Identification*, 7th IFAC Symposium on Non-linear Controls 2007

[5] L. Ljung.,*Perspectives on system identification*, Annual Reviews in Controls, Vol:34, pp. 1-12, 2010

[6] S. Zhang, R. Dubay, M. Charest, *A principal component analysis model-based predictive controller for controlling part warpage in plastic injection molding*, Expert Systems with Applications: An International Journal, Vol:42, pp. 2912-2927, 2015

[7] Z. Zhang, J. Wang, H. Zha, *Adaptive Manifold Learning*, IEEE Transactions on Pattern Analysis and Machine Intelligence,pp. 253-265, 2012

[8] Y. Ma and Y. Fu, *Manifold Learning Theory and Applications*, CRC Press 2011

[9] H. Ohlsson, J. Roll, T. Glad and L. Ljung, *Using Manifold Learning for Non-linear System Identification*, 7th IFAC Symposium on Non-linear Control Systems 2007

[10] H. Ohlsson, J. Roll and L. Ljung, *Manifold-Constrained Regressors in System Identification*, 47th IEEE Conference on Decision and Control 2008

[11] B. Boots and G. Gordon, *Two-Manifold Problem with Applications to Non-linear System Identification*, 29th International Conference on Machine Learning, Edinburgh, Scotland 2012

[12] J. Tenenbaum, V. Silva and J. Langford, *A Global Geometric Framework for Non-linear Dimensionality Reduction*, Science, Vol:290, pp. 2319:2322

[13] S. Roweis and L. Saul, *Non-linear Dimensionality Reduction by Local Linear Embedding*, Science, Vol:290, pp. 2323:2326

[14] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, Vol:12, pp. 2825-2830, 2011

[15] S. Walt, S. Colbert and G. Varoquaux, *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science & Engineering, Vol:13, pp. 22-30, 2011

[16] J. Wilson, M. Charest and R. Dubay, *Non-linear model predictive control schemes with application on a 2 link vertical robot manipulator*, Robotics and Computer-Integrated Manufacturing, Vol:41, pp. 23-30, 2016

[17] B. Li. et al., *Large Scale Recurrent Neural Network on GPU*, International Joint Conference on Neural Networks, pp. 4062:4069, 2014