Automatic
Control &
Systems
Engineering.

A Comparative study of classical and advanced machine learning methods for Regression and Classification on NinaPro Datasets based on s-EMG signals.

Sindhu Vasireddy

September 2021

Supervisor: Dr. Tara Baldacchino

A dissertation submitted in partial fulfilment of the requirements for the degree of MSc in Advanced Control and Systems Engineering (with a Year In Industry)

Abstract:
Techniques as varied as researchers themselves have been used to address the s-EMG pattern recognition problem. All place a conclusive judgement as to which is “The ultimate model” for regression of s-EMG signals and classification of hand movements, however all such comparisons are constrained by a combination of ‘lack of information clarity’ and a standard yardstick for comparison. Although datasets like NinaPro provide a benchmark, designed classification models lack clarity on classification imbalance problem, regression models use a different feature extraction or additional filtering stages but use the accuracy of benchmarked model from literature. In cases of comparative studies, classical techniques are not compared with modern machine learning and deep learning methods and vice-versa. In this work under regression : Ordinary least squares, Bayesian least squares, Auto Regressive models with eXogenous inputs (Linear - ARX and Non-linear - NARX), Long Short-Term Memory (LSTM) neural network and General Adversarial Network (GAN) are compared and under classification K-nearest neighbours(KNN) and LSTM are compared on a balanced training data. Bayesian Least square regression was found to be optimum in regression and KNN is best for Classification. Also classical methods against popular opinion would be the ideal modelling solution for myoelectric prosthesis owing to high computational cost of ML methods which renders them undesirable for embedded electronics even with modern computation abilities, when patient’s comfort, heat dissipation and resultant improvement in accuracy are all considered.

Keywords: myoelectric prosthetics, GAN, LSTM, ARX, NARX, Bayesian, Least Squares, KNN, sequence generation, s-EMG pattern recognition

Table of Contents

Table of Figures

Table of Tables

1 Introduction:

Research and technical advancements often propel human evolution with an intent to improve the quality of day-to-day-existence. Myoelectric prosthesis is one such novel effort towards restoring lost functionality of amputees. [1] Prosthetics are constantly improving to mimic near real human movements. Both invasive and non-invasive sensing technologies are deployed to provide this myoelectric control action[2]. Although invasive sensors have high reliability, their clinical and commercial usage is unviable; owing to its practical discomfort.[3]

Quality of control action directly depending on the sensitivity of input/stimuli measurements is a classical thought which is quickly evolving with modern system modelling techniques. Expert knowledge of signal features and system structure are compensating for learning the model parameters. In cases where there is a lack of access to signal dynamics feature engineering steps in. Similarly where there is a lack of full or partial knowledge of system structure, machine learning steps in to fill the gap.

In line with the same approach modelling and training of myoelectric systems can bridge the trade-off between user comfort and accuracy of s-EMG measurement in myoelectric prosthetics. This enables design of better control system architectures and helps in effective simulation before the usage of actual prosthetic by the patient.[4] Generated models are also increasingly used in designing human machine interfaces for amputees.

This study aims to investigate and understand the effectiveness of classical regression and classification techniques over emerging deep learning techniques.

1.1 Motivation

Pattern recognition of surface Electro Myography (s-EMG) signals is a classical problem, primarily due to the challenging stochastic nature of the signal [5] and due to its wide ranging applications including myoelectric prosthetic control[6], human-computer interfaces and its significant impact in clinical diagnosis for neuro pathological disorders [7].

Several works of regression and classification including neural networks have been conducted on EMG and s-EMG patterns. Yet due to the predominant historic confidentiality of s-EMG datasets, modern techniques like deep learning have not been extensively implemented on s-EMG signals [8] In the few cases, that are slowly gaining traction, the model parameters are heavily reliant on dataset and subjects like in [9] which is substantiated with a comprehensive literature review in [8].

This gap in availability of standard datasets is being bridged by open dataset initiatives in the last decade, yet there has not been a standard international benchmarking protocols in the research community despite wide ranging recommendations from initiatives like Surface Electromyography for the Non-Invasive Assessment of Muscles (SENIAM). First such significant benchmarking effort was the dataset NINAPRO (Non-INvasive Adaptive PROsthetics) [10] and it is now a widely referenced bench mark standard.

This open-source data repository has 3 sub-databases covering a total of 78 subjects, of which 67 are intact and 11 amputees. For this study the second database which has 40 intact subjects is chosen, comprising 28 men and 12 women. They are in the age band of 29.9±3.9 years and the dominant hand of 34 subjects was right and rest 6 was left-handed. Since this data has been statistically validated for repetition and evaluated on four standard classification models using six standard input signal features by the authors. The database has been a gold standard in hand movement analysis [11]

1.2 Problem Statement

As the world is increasingly moving towards advanced machine learning techniques in various domains, the scope of this paper is to study its efficacy in myoelectric prosthesis compared to the classical regression and classification methods.

There is a research gap when it comes to comparing, classical and deep learning regression and classification approaches on the same dataset. This work intends to bridge the gap and investigate the claim that targeted placement of electrodes can be compensated by effectiveness of pattern recognition techniques. Thereby evaluating the efforts of sub-optimal placement techniques for electrodes.

1.3 Research Questions

1) How do standard deep learning methods compare to classical methods for regression problem?
2) How do Linear and non-linear models perform in comparison for s-EMG Data?
3) How do distance-based machine learning algorithms compare to deep learning algorithms in classification problem?
4) How do supervised and unsupervised learning algorithms perform in comparison in pattern recognition problem?
5) Which of methods are effective in computational load for embedded electronics of prosthetics?

1.4 Aims and Objectives

1.5 Project Management: Gantt Chart

2 Background

2.1 Related Work:

Study and modelling of human movements and neurological motor control started with an attempt to establish a relationship between isometric contractions in human muscles based on measured action potentials back in 1952[12].A predominantly linear causal relationship was established between s-EMG signals and human muscular movement including real-time dynamic human actions. This has been significantly reviewed and presented in literature over seven decades through a plethora of experiments, improving the process through all stages from data collection to modelling. Thereby ensuring the imperative position s-EMG signals hold in aiding movement analysis.[13]

Membrane electrophysiology gained traction ever since a non-linear equivalence circuit of ionic discharge in humans that depolarizes sarcolemma to produce muscle contractions was developed in 1952 by Hodgkin and Huxley. [14] This classical electric model served as cornerstone of modern electrophysiology and histological experimentation. Now multifarious research [4], [6], [6], [15], [16] promises exemplary models for human hand movements from s-EMG signals. Models relying on non-invasive sensors are yet to be commercially and clinically widespread one among its many reasons is often attributed to lack of global open benchmark dataset as highlighted in [8], [11]

As mentioned earlier NINAPRO bridged that gap, by studying 78 subjects including 11 amputees and presented the data in two public repositories NINAPRO [17] and DYRAD [18] after initially pre-processing raw s-EMG signals as 3 sub-databases in 2014 [11]. The pre-processing involved synchronization, re-labelling and filtering in case of the second and third databases that uses 'Delsys' differential electrodes for recording s-EMG signals. Authors have expressed interest to share the raw s-EMG data upon request.

NINAPRO has demonstrated technical validation across four standard classification models built upon five common feature extraction strategies of s-EMG signals.[11] Authors have also indicated significance of mapping between signal amplitudes and movement repetitions. Hence advise caution while splitting of movement repetitions owing to this attribute [19] Additionally, they specify the lack of classification imbalance in the literature results as also a contributing factor to lack of comparative analysis. Both are addressed in this work by considering the same training (1.3.6) and test (2.5) repetitions in-line with the sample validation model and a balanced classification test data input (1945) across all 9 classes.

The dataset has been pivotal in enabling "Quantitative taxonomy of human hand grasps"[20] Thereby placing it as an unrivalled open standard for hand movement pattern recognition problems. It addresses the problem of data availability and benchmarking in intact subjects despite some room for improvement in case of amputees' data. Issue identified by the authors are fatigue/pain, discomfort leading to disruption of the activity and insufficient place or provision to place all measurement electrodes. This work intends to investigate and address this issue two-fold - primarily by modelling all 12 electrode inputs and in iterations

Despite giant research space of pattern recognition for myoelectric control, model technique comparisons over same experimental settings are still lacking. Closest estimations rely on similar data acquisition and pre-processing yet there is impact of using different although similar modelling parameters[8]. Restricted access to research data was merely one of the reasons. This has been mentioned in literature, one such instance 14 years ago in [21], but since then till now at least 33 small data sources [8] with different settings for s-EMG pattern recognition is available, although they are not as significant as NINAPRO.

However this situation hasn't improved further despite such initiatives, this is evident in recent papers where deep learning techniques have been attempted but isolated with aim of showing better recognition potential or superiority of the model.[8] In the pursuit of which a varying combination of pre-processing techniques and features are being extracted from the input signals. This has caused an imbalance and a want for comparative analysis of techniques deployed in pattern recognition, with same experimental conditions.

Advanced machine learning approaches [22] have been credited with successfully automating feature extraction thus providing the toolset to bypass dedicated feature reduction and handle raw s-EMG signals [8]. On the contrast they have enormous computational effort which makes them a disadvantage in dynamic user settings[23]. These experiments have also been placing an ever increasing demand of quality training data, with attempts at combining multiple datasets to generate more effective models.[8]

On the other hand similar window length and increments to the actual data acquisition have been used after standard filtering and pre-processing in classical methods like Linear discriminant analysis [21] These investigations have proved due simpler mathematical constructs[24] and feature engineering can go a long way in robotic prosthesis control in an embedded system implementation placing a limit on computation power too.

This is the other gap that this work aims to bridge by comparing classical methods of least squares estimation, auto-regressive modelling[25] in linear and non-linear settings with deep learning methods such as long short term memory recurrent neural networks which is popularly regarded for its regression and classification estimation in literature.

Least square estimation is done in classical setting [21] and in Bayesian framework which has been proved effective for a solving regression and classification problems in parallel for this s-EMG dataset [24] Models are estimated to solve the regression and a balanced classification using the same training, test split as per the data validation benchmark[11].

Autoregressive models [26], [27] linear and non-linear have been investigated for myoelectric control and s-EMG pattern recognition problems [28] .A few regression experiments like Akbari et. al. [29] are performed on NinaPro dataset too, (DB 5 collected using myoarm band). However, the comparison benchmark is not that against the validation model provided by the author or any other classical method, rather it is done against other neural net. In addition to this fact the NARX model parameters were pre-estimated using Levenbarg-Marqardt Algorithm in [29], a combined NARX neural network model was used for regression and was estimated as more accurate than standalone MLP NN on the same data.

The computation time which forms the pivotal element in embedded systems that propel the prosthesis was not addressed in [29]. Thereby posing a gap in its comparative advantage or disadvantage in a practical setting. Most importantly model accuracy was not revealed a quick look shows MSE value to be greater than 100% in 1 movement (Turn a screw) and 9 of 23 movements have MSE in range of 20-40% and 3 movements over 40% which could be a good measure against Neural net models[29] but it's comparison against classical methods is lacking, which is attempted in this work.

Last year Zanini et. al. [30] had attempted regression and sequence generation of tremor patterns in Parkinson's patients, using deep Convolutional Generative Adversarial Network (DCGAN). Subject 1 and exercise 3 from database 2 of NinaPro was used as the ground comparison value of real patient. As cited in [11] this work also fails to provide information of the balanced or unbalanced nature of classification, which makes it harder to compare performance with existing results.

A combination of feature extraction methods and hybrid Multi-layer perceptron (MLP) and LSTM architecture suited specifically to enhancing the tremor behaviours from the baseline s-EMG are used in [30]. There is only a comparison of loss within different AI methods in literature, but degree of fit to actual signal is not estimated before attempting data augmentation. DCGAN is executed in a highly supervised training with no information on its open loop performance. Author claims the attempt helps in "finding optimal weights for the evaluation and to better feature abstraction needed to generate a realistic sample." for hand movement s-EMG datasets like NinaPro. Author points to Time series CGAN models [31] to be explored for data augmentation of s-EMG datasets and works like [32] have demonstrated a potential for unsupervised generation of realistic samples for time series signals. Which has been used as guiding principle for GAN based investigations in this work.

2.2 GPU and Tools Background

1. NVIDIA GEFORCE GTX 1050 – This Graphic Processing Unit in GEFORCE product range manufactured by NVIDIA has 640 Compute Unified Device Architecture (CUDA) cores has a clocking speed of 1455MHz in processor clock and 1354 MHz in graphics clock during medium to full utilization.
2. MATLAB- MATLAB (2021b) offers several toolboxes for regression and classification tasks (linear and non-linear). Custom functions were created using functionalities of the following toolboxes. The data was fed as cells, generated from NINAPRO DB2 '.dat' files [17].
    a. Statistics and Machine learning Toolbox- Provides regression, classification, and machine learning algorithms along with data analysis functions from feature extraction to visualization.[33] It has the descriptive statistical tools along with data fitting capabilities, which are utilized in this work.
    b. Econometrics Toolbox- Provides general estimation and forecasting tools for economic systems, with specific functions to aid in Bayesian and Markov frameworks[34], former of which was used for Bayesian Regression models here.
    c. System Identification Toolbox- Provides functions and tools for dynamic system modelling, including online parameter estimation among other support tools for grey-box identification techniques[35]. These capabilities were utilized in ARX, NARX regression modelling.
    d. Deep Learning Toolbox- Provides functions and tools for designing and deploying deep neural network architectures for regression and classification like LSTM including GAN[36]. LSTM and GAN were designed, trained, and tested using this toolset.
    e. Parallel Computing Toolbox- Provides tools to handle computational intensity using graphic and multi-core processors and network computer clusters[37]. Various parallel enabled functions of MATLAB can be used to multi-thread processes. Iterative programming and batch modelling were supported, this was used to parallelize feature extraction and deep learning model training.

2.3 Dataset

In NINAPRO DB2 to capture muscle activity . Twelve Trigno wireless electrodes (manufactured by Delsys Inc.) were used at 12 sites – 8 were placed around the forearm at same height of radio-humeral joint , equally spaced, two at main activity spots of wrists namely flexor and extensor digitoram superficialis [38] and other two electrodes are placed in main activity spots of biceps and triceps branchii identified by palpation, This is a combinational approach of dense sampling[39]–[41] and anatomical positioning [42]–[45]due to their prevalence in literature.

Finger Force Linear Sensor (FFLS) were used to capture hand dynamics along 6 Degree of Freedom (DoF) namely the flexion and extension of 5 fingers along with abduction and adduction of the thumb. Since the sensor utilizes a strain gauge known for high repeatability and minimal hysteresis a near-linear relationship is mined[11] This signal was captured using a standard 100Hz, 12-bit National Instruments Data Acquisition card (DAQ) PCMCIA 6024E. Windows performance counter ensured precise mapping of each data with its timestamp.

Commercial electrodes were chosen by authors to record rectified signal in place of raw signals due to its impact on classification results[11]. Operational range of these electrodes were 40m and they had inbuilt rechargeable batteries, and the data was collected via a wireless base station and transferred to laptop using standard USB connection. s-EMG signals were sampled at a rate of 2KHz with a Root Mean Squared noise bounded at 70nV and the 3-axis accelerometer embedded in the electrodes had a sampling rate of 148KHz, these were placed on forearm using their adhesive bands.

The subjects were made aware of the focus which was more on mimicking the action and less on the force of movement, which was all chosen from standard daily activities thereby ensuring practical relevance for amputees and decreased discomfort for all subjects. Three exercises with 17,23 and 9 movements each was requested to be repeated 6 times by all subjects in the second database. Each movement lasted 5seconds with a rest of 3seconds in between them, and movements were sequential in repetition to reduce any effects of response time from conscious thought towards random movements.

First two exercises focused on Hand kinematics using a motion capturing data glove Cyberglove II with 22 resistive bend sensors (proprietary manufactured by Cyberglove systems LLC) to measure joint-angle movements. This was supplemented by wrist orientation measurements (range of 120º and resolution 0.15º) from a 2-axis IS40 inclinometer (by Fritz Kubler GmBH). However, it is the third exercise of 9 movements collected by FFLS to estimate hand dynamics that is pivotal to this comparative investigation.

The collected data was synchronized using nearest-neighbour interpolation in DB2 to 2KHz to match the time resolution of signal, human error in movement synchronization was corrected for offline using likelihood ratio algorithm, and since Trigno electrodes had a potential noise from power (50Hz – in Europe and Asia) and its harmonics hence it was filtered using Hampel filter before making the database available. [46].

3 Methodology:

There are broadly 4 stages to the methodology adopted in this study- the literature survey, the data pre-processing, the modelling, and the analysis, namely. The next sub-headings shall cover them in detail.

3.1 Literature survey

A critical survey of the literature available in the field of myoelectric prosthesis has shown that the current areas of research in modelling regression and classification problems, include the use of: Deep Learning, Auto Regression, and, in some cases, a blend of the two methods. Some of the previously used techniques include Linear Regression and Bayesian Linear Regression.

So, this work aims at investigating:

- The scope for Generative Adversarial Networks, in regression and sequence generation.
- The efficacy of statistical methods over deep learning approaches in solving the regression and the classification problems.

3.2 Data Pre-processing

There are 4 stages under Data Pre-processing, namely, the Data Acquisition stage, the Windowing stage, the Feature Extraction stage, the Train and Test split stage.

3.2.1 Data Acquisition

Data was acquired from DB2 of the NinaPro site. 5 subjects' finger movement data was downloaded.

3.2.2 Windowing

A window size of 800 was used in the pre-processing stage with a step increment of 20 each time.

3.2.3 Feature Extraction

Feature Extraction is also known as Feature Reduction. After Feature Extraction the size of the dataset was reduced to 1945 from an original dataset size of 39700, when a window size of 800 with a step increment of 20 was used. And, when a window size of 100 was used the dataset size went down from 39700 to 1980.

With a window size of 800 when the step increment size was increased to 100, the size of the dataset was down to 389 samples.

3.2.3.1 RMS

A technique by which only the most important features are retained by performing a square root on the mean of squares of the 12 features individually over each of the windowed time periods. This helps in linearising the data.

3.2.4 Train and Test Sets Split

How the data is split for training and testing has a lot of impact on the performance of the technique used for both Regression as well as Classification problems. A 70% and 30% split of the data was used for training and testing respectively for the course of this work. In Regression, the first four repetitions of movement were used as training data and last two repetitions as test data and in case of classification repetitions 1,3,4 and 6 were used to train the model and it was tested on repetitions 2 and 5.

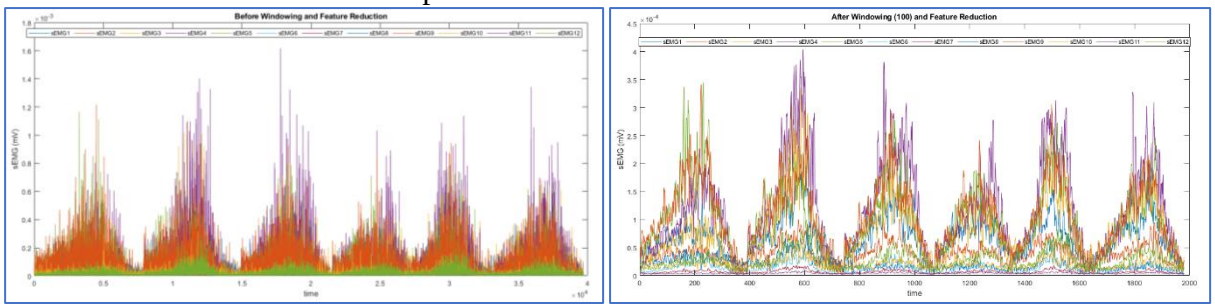


Figure 1 Subject 1 data before and after feature extraction

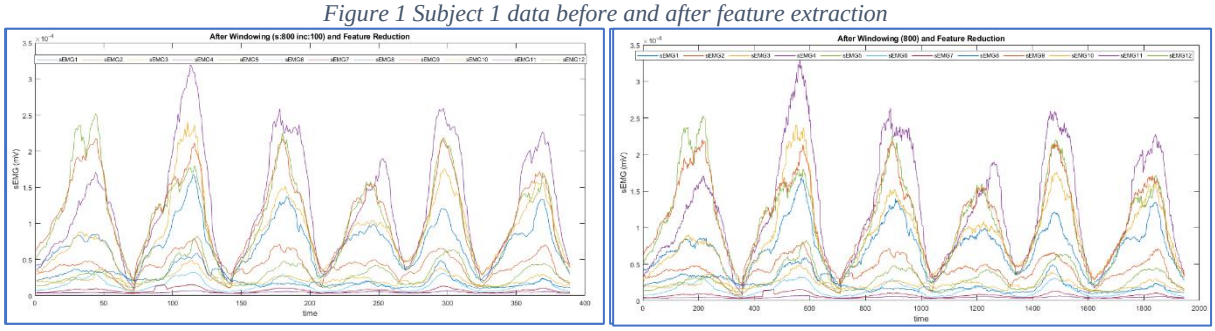


Figure 2 Subject 1 data after feature reduction using window sizes 100 and 800

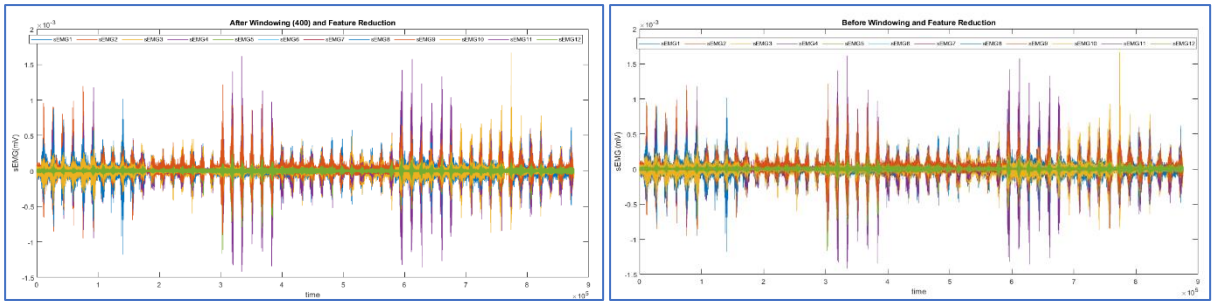


Figure 3 Subject 2 data before and after feature extraction

3.3 Modelling

3.3.1 Regression

A technique in statistics that allows for a mapping between a dependent variable and one or more independent variables. There are two types of regression techniques: linear and non-linear.

Linear Regression techniques deployed were as follows:

3.3.1.1 Ordinary Least Squares

A basic approach in statistics that relies on minimising the sum of squares of residual error between the predicted data points along the line of best fit and the actual data points to arrive at a model that best explains the relationship between the dependent variable and the independent variable(s).

$$\boldsymbol{y} = \boldsymbol{w}\boldsymbol{x} + \boldsymbol{b}$$

$$y(n) = w_1 x_1(n) + w_2 x_2(n) + \cdots + w_{12} x_{12}(n) + \xi(n)$$

$$y(n) = [x_1(1) \quad \dots \quad x_m(1)]_{1\times m} \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}_{m\times 1}$$

$$y = \boldsymbol{\phi}^T \boldsymbol{w} + \xi(n), where\ \boldsymbol{\phi}^T = [x_1(1) \quad \dots \quad x_m(1)]_{1\times m}, \boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}_{m\times 1}$$

$$\begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}_{N\times 1} = \begin{bmatrix} x_1(1) & \dots & x_m(1) \\ \vdots & \ddots & \vdots \\ x_1(N) & \dots & x_m(N) \end{bmatrix}_{N\times m} \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}_{m\times 1} + \begin{bmatrix} \xi(1) \\ \vdots \\ \xi(N) \end{bmatrix}_{N\times 1}$$

$$\boldsymbol{Y} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{\Xi}, where\ \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}^T(1) \\ \vdots \\ \boldsymbol{\phi}^T(N) \end{bmatrix}_{N\times 1}, \boldsymbol{\Xi} = \begin{bmatrix} \xi(1) \\ \vdots \\ \xi(N) \end{bmatrix}_{N\times 1}, \boldsymbol{Y} = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}$$

Cost function(J): Minimising sum of squares of errors across all N time samples.

$$\frac{1}{N}\sum_{n=1}^{N} (y(n) - \boldsymbol{\phi}^T(n)\widehat{\boldsymbol{w}})^2$$

$$J = \frac{1}{N}[(y(1) - \boldsymbol{\phi}^T(1)\widehat{\boldsymbol{w}}) \quad \dots \quad (y(N) - \boldsymbol{\phi}^T(N)\widehat{\boldsymbol{w}})]\begin{bmatrix}(y(1) - \boldsymbol{\phi}^T(1)\widehat{\boldsymbol{w}}) \\ \vdots \\ (y(N) - \boldsymbol{\phi}^T(N)\widehat{\boldsymbol{w}})\end{bmatrix}$$

$$J = \frac{1}{N}[[y(1) \quad \dots \quad y(N)] - [\widehat{\boldsymbol{w}}^T\boldsymbol{\phi}(1) \quad \dots \quad \widehat{\boldsymbol{w}}^T\boldsymbol{\phi}(N)]]\left[\begin{bmatrix}y(1) \\ \vdots \\ y(N)\end{bmatrix} - \begin{bmatrix}\boldsymbol{\phi}^T(1)\widehat{\boldsymbol{w}} \\ \vdots \\ \boldsymbol{\phi}^T(N)\widehat{\boldsymbol{w}}\end{bmatrix}\right]$$

$$J = \frac{1}{N}[(\boldsymbol{Y}^T - \widehat{\boldsymbol{w}}^T\boldsymbol{\Phi}^T)(\boldsymbol{Y} - \boldsymbol{\Phi}\widehat{\boldsymbol{w}})]$$

Since, J is quadratic in $\widehat{\boldsymbol{w}}$ it takes on a convex shape and therefore, solving for $\frac{dJ}{d\,\widehat{\boldsymbol{w}}} = 0$ yields the optimal value for $\widehat{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{Y}$, such that the loss is minimised.

The solution for the weights ($\widehat{\boldsymbol{w}}$) is solved using the Moore-Penrose Pseudoinverse.Predicted Output is calculated as $\widehat{\boldsymbol{Y}} = \boldsymbol{\Phi}\widehat{\boldsymbol{w}}$.This is also known as the Frequentist approach.

3.3.1.2Bayesian Least Squares

A prior probability distribution is assumed on the weights in addition to information available to calculate the weights in the Frequentist approach.

Gaussian Probability Density Function(pdf):

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The model equation can be written as a sum of weighted linear combinations of basis functions φ_j.

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j\varphi_j(\boldsymbol{x}) = \sum_{j=0}^{M-1} w_j\varphi_j(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})$$

The target variable is defined as:

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \varepsilon$$

Where ε is a random value chosen from a Gaussian Distribution with mean 0 and variance $\frac{1}{\beta}$.

3.3.1.2.1 Conditional Distribution of t given x

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = N(t|y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1}) = \frac{1}{\sqrt{\frac{1}{\beta}}\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{t-y(\boldsymbol{x},\boldsymbol{w})}{\sqrt{\frac{1}{\beta}}}\right)^2} = \sqrt{\frac{\beta}{2\pi}}e^{-\frac{\beta}{2}(t-y(\boldsymbol{x},\boldsymbol{w}))^2}$$

3.3.1.2.2 Joint Conditional Probability of t given x:

Also called the Likelihood function.

$$p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\beta)=\prod_{i=1}^{N}N(t_i|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i),\beta^{-1})$$

In general,

$$p(\boldsymbol{t}|\boldsymbol{\Phi},\boldsymbol{w},\sigma^2)\propto(\sigma^2)^{-\frac{N}{2}}e^{(-\frac{1}{2(\sigma^2)}(\boldsymbol{t}-\boldsymbol{\Phi w})^T(\boldsymbol{t}-\boldsymbol{\Phi w}))}$$

Log-Likelihood is quadratic in $\boldsymbol{w}$.

$$\log p(\boldsymbol{t}|\boldsymbol{w},\beta)=\frac{N}{2}\log\beta-\frac{N}{2}\log 2\pi-\beta E_D(\boldsymbol{w})$$

Sum-of-Squares Error Function:

$$E_D(\boldsymbol{w})=\frac{1}{2}\sum_{i=1}^{N}(t_i-\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i))^2=\frac{1}{2}||\boldsymbol{t}-\boldsymbol{\Phi w}||^2$$

$$\boldsymbol{\Phi}=\begin{pmatrix}\varphi_0(\boldsymbol{x}_1) & \cdots & \varphi_{M-1}(\boldsymbol{x}_1)\\ \vdots & \ddots & \vdots\\ \varphi_M(\boldsymbol{x}_N) & \cdots & \varphi_{M-1}(\boldsymbol{x}_N)\end{pmatrix}_{M\times N}$$

$\boldsymbol{\Phi}$ is known as the Design matrix.

3.3.1.2.3 Prior Distribution

It is assumed to be a conjugate to Likelihood function implying that the prior distribution also takes on a Gaussian function form. This enables us to calculate the Posterior Distribution as will be shown in the next section.

Re-writing Log-Likelihood such that it becomes normal in $(\boldsymbol{w}-\widehat{\boldsymbol{w}})$ from quadratic in $\boldsymbol{w}$, requires replacing:

$$(\boldsymbol{t}-\boldsymbol{\Phi w})^{\boldsymbol{T}}(\boldsymbol{t}-\boldsymbol{\Phi w})=(\boldsymbol{t}-\boldsymbol{\Phi}\widehat{\boldsymbol{w}})^{\boldsymbol{T}}(\boldsymbol{t}-\boldsymbol{\Phi}\widehat{\boldsymbol{w}})+(\boldsymbol{w}-\widehat{\boldsymbol{w}})^{\boldsymbol{T}}(\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi})(\boldsymbol{w}-\widehat{\boldsymbol{w}})$$

Proof: Consider the LHS and expand:

$$\begin{aligned}&=\boldsymbol{t't}-\boldsymbol{t'\Phi w}-\boldsymbol{w'\Phi't}+\boldsymbol{w'\Phi'\Phi w}\\&=\boldsymbol{t't}-\boldsymbol{t't}-\boldsymbol{t'\Phi w}-\boldsymbol{t't}-\boldsymbol{w'\Phi't}+2\boldsymbol{t't}+\boldsymbol{w'\Phi'\Phi w}\\&=\boldsymbol{t't}-\boldsymbol{t'}(\boldsymbol{t}+\boldsymbol{\Phi w})-(\boldsymbol{t'}+\boldsymbol{w'\Phi'})\boldsymbol{t}+2\boldsymbol{t't}+\boldsymbol{w'\Phi'\Phi w}\\&=\boldsymbol{t't}-\boldsymbol{t'}(\boldsymbol{\Phi\Phi^{-1}})(\boldsymbol{\Phi^{-1}\Phi})'(\boldsymbol{t}+\boldsymbol{\Phi w})-(\boldsymbol{t'}+\boldsymbol{w'\Phi'})(\boldsymbol{\Phi\Phi^{-1}})(\boldsymbol{\Phi^{-1}\Phi})'\boldsymbol{t}+2\boldsymbol{t't}+\boldsymbol{w'\Phi'\Phi w}\\&=\boldsymbol{t't}-\boldsymbol{t'\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'}(\boldsymbol{t}+\boldsymbol{\Phi w})-(\boldsymbol{t'}+\boldsymbol{w'\Phi'})\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi't}+2\boldsymbol{t't}+\boldsymbol{w'\Phi'\Phi w}\\&=\boldsymbol{t't}-\boldsymbol{t'}(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})'(\boldsymbol{t}+\boldsymbol{\Phi w})-(\boldsymbol{t'}+\boldsymbol{w'\Phi'})(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})'\boldsymbol{t}+2\boldsymbol{t't}+\boldsymbol{w'\Phi'\Phi w}\\&=\boldsymbol{t't}-\boldsymbol{t'}(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})'(\boldsymbol{t}+\boldsymbol{\Phi w})-(\boldsymbol{t'}+\boldsymbol{w'\Phi'})(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})\boldsymbol{t}+2\boldsymbol{t't}+\boldsymbol{w'\Phi'\Phi w}\\&=\boldsymbol{t't}-\boldsymbol{t'}(\boldsymbol{\Phi}\left(\boldsymbol{\Phi'^{\Phi})^{-1}\Phi'}\right)'\boldsymbol{t}-\boldsymbol{t'}(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})'\boldsymbol{\Phi w}-\boldsymbol{t'}(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})\boldsymbol{t}\\&\quad-\boldsymbol{w'\Phi'}(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})\boldsymbol{t}+2\boldsymbol{t'}(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})'(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})\boldsymbol{t}+\boldsymbol{w'\Phi'\Phi w}\end{aligned}$$

Since, $(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})'(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})=\boldsymbol{I}$

Assume $(\boldsymbol{\Phi}(\boldsymbol{\Phi'\Phi})^{-1}\boldsymbol{\Phi'})=\boldsymbol{K}$ for ease of computation:

$$\begin{aligned}&=\boldsymbol{t't}-\boldsymbol{t'Kt}-\boldsymbol{t'K'\Phi w}-\boldsymbol{t'Kt}-\boldsymbol{w'\Phi'Kt}+2\boldsymbol{t'K'Kt}+\boldsymbol{w'\Phi'\Phi w}\\&=\boldsymbol{t't}-\boldsymbol{t'Kt}-\boldsymbol{t'Kt}+\boldsymbol{t'K'Kt}-\boldsymbol{w'\Phi'Kt}-\boldsymbol{t'K'\Phi w}+\boldsymbol{t'K'Kt}+\boldsymbol{w'\Phi'\Phi w}\end{aligned}$$

Further re-grouping gives:

$$=(\boldsymbol{t}-\boldsymbol{Kt})'(\boldsymbol{t}-\boldsymbol{Kt})-(\boldsymbol{w}-\boldsymbol{Kt})'(\boldsymbol{\Phi'\Phi})(\boldsymbol{w}-\boldsymbol{Kt})$$

Recall that $\widehat{\boldsymbol{w}}=\boldsymbol{Kt}$,

$$=(\boldsymbol{t}-\widehat{\boldsymbol{w}})'(\boldsymbol{t}-\widehat{\boldsymbol{w}})-(\boldsymbol{w}-\widehat{\boldsymbol{w}})'(\boldsymbol{\Phi'\Phi})(\boldsymbol{w}-\widehat{\boldsymbol{w}})$$

The Likelihood function after normalization in $(\boldsymbol{w}-\widehat{\boldsymbol{w}})$ can be written as:

$$p(\boldsymbol{t}|\boldsymbol{\Phi},\boldsymbol{w},\sigma^2) \propto (\sigma^2)^{-\frac{v}{2}} e^{\left(-\frac{vs^2}{2(\sigma^2)}\right)} (\sigma^2)^{-\frac{N-v}{2}} e^{\left(-\frac{1}{2(\sigma^2)}(\boldsymbol{w}-\widehat{\boldsymbol{w}})^{\boldsymbol{T}}(\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi})(\boldsymbol{w}-\widehat{\boldsymbol{w}})\right)}$$

where, $vs^2 = (\boldsymbol{t}-\boldsymbol{\Phi}\widehat{\boldsymbol{w}})^{\boldsymbol{T}}(\boldsymbol{t}-\boldsymbol{\Phi}\widehat{\boldsymbol{w}})$ and v =N-k, k=no. of coefficients.

$$p(\sigma^2) \propto (\sigma^2)^{-\frac{v_0}{2}-1} e^{\left(-\frac{v_0 s_0^2}{2(\sigma^2)}\right)}$$

$$a_0 = \frac{v_0}{2}, b_0 = \frac{v_0 s_0^2}{2}$$

where, v_0 and s_0^2 are the prior values of v and s^2. And $p(\sigma^2)$ has an inverse-gamma distribution.

$$p(\boldsymbol{w},\sigma^2) = p(\sigma^2)p(\boldsymbol{w}|\sigma^2)$$

Conditional Prior Distribution:

$$p(\boldsymbol{w}|\sigma^2) \propto (\sigma^2)^{-\frac{k}{2}} e^{\left(-\frac{1}{2(\sigma^2)}(\boldsymbol{w}-\boldsymbol{\mu_0})^{\boldsymbol{T}}\boldsymbol{\Lambda_0}(\boldsymbol{w}-\boldsymbol{\mu_0})\right)}$$

Assuming prior distribution of weights to be an isotropic gaussian with covariance among the different weights of the regressors as 0: mean $\boldsymbol{\mu_0} = 0$ and precision $\boldsymbol{\Lambda_0} = \alpha\boldsymbol{I}$

$$p(\boldsymbol{w}|\alpha) = N(\boldsymbol{w}|0,\alpha^{-1}\boldsymbol{I})$$

3.3.1.2.4 Posterior Distribution

According to Bayes theorem, posterior beliefs about $\boldsymbol{w}, \alpha, \beta$ can be got by combining the Prior beliefs about them with the Likelihood function.

$$p(\boldsymbol{w},\sigma^2|\boldsymbol{t},\boldsymbol{\Phi}) \propto p(\boldsymbol{t}|\boldsymbol{X},\boldsymbol{w},\sigma^2)p(\boldsymbol{w}|\sigma^2)p(\sigma^2)$$

$$p(\boldsymbol{w},\sigma^2|\boldsymbol{t},\boldsymbol{\Phi})$$
$$\propto (\sigma^2)^{-\frac{N}{2}} e^{\left(-\frac{1}{2(\sigma^2)}(\boldsymbol{t}-\boldsymbol{\Phi w})^T(\boldsymbol{t}-\boldsymbol{\Phi w})\right)} (\sigma^2)^{-\frac{k}{2}} e^{\left(-\frac{1}{2(\sigma^2)}(\boldsymbol{w}-\boldsymbol{\mu_0})^{\boldsymbol{T}}\boldsymbol{\Lambda_0}(\boldsymbol{w}-\boldsymbol{\mu_0})\right)} (\sigma^2)^{-(a_0+1)} e^{\left(-\frac{b_0}{(\sigma^2)}\right)}$$

Writing the posterior mean $\boldsymbol{\mu_N}$ and $\boldsymbol{w}$ in terms of prior mean $\boldsymbol{\mu_0}$ and the least squares estimator $\widehat{\boldsymbol{w}}$,

$$\boldsymbol{\mu_N} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\Lambda_0})^{-1}(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\widehat{\boldsymbol{w}} + \boldsymbol{\Lambda_0}\boldsymbol{\mu_0})$$

Re-arranging the exponent of Posterior Distribution allowing it to be expressed as a quadratic in $(\boldsymbol{w}-\boldsymbol{\mu_N})$ gives:

$$(\boldsymbol{t}-\boldsymbol{\Phi w})^T(\boldsymbol{t}-\boldsymbol{\Phi w}) + (\boldsymbol{w}-\boldsymbol{\mu_0})^{\boldsymbol{T}}\boldsymbol{\Lambda_0}(\boldsymbol{w}-\boldsymbol{\mu_0})$$
$$= (\boldsymbol{w}-\boldsymbol{\mu_N})^{\boldsymbol{T}}(\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\boldsymbol{\Lambda_0})(\boldsymbol{w}-\boldsymbol{\mu_N}) + \boldsymbol{t}^T\boldsymbol{t} - (\boldsymbol{\mu_N})^{\boldsymbol{T}}(\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\boldsymbol{\Lambda_0})(\boldsymbol{\mu_N}) + \boldsymbol{\mu_0^T}\boldsymbol{\Lambda_0}\boldsymbol{\mu_0}$$

Replacing the exponent of Posterior Distribution with the RHS:

$$p(\boldsymbol{w},\sigma^2|\boldsymbol{t},\boldsymbol{\Phi})$$
$$\propto (\sigma^2)^{-\frac{N}{2}} e^{\left(-\frac{1}{2(\sigma^2)}(\boldsymbol{w}-\boldsymbol{\mu_N})^{\boldsymbol{T}}(\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\boldsymbol{\Lambda_0})(\boldsymbol{w}-\boldsymbol{\mu_N})\right)} (\sigma^2)^{-\frac{N}{2}-(a_0+1)} e^{\left(-\frac{1}{2(\sigma^2)}(2b_0+\boldsymbol{t}^T\boldsymbol{t}-(\boldsymbol{\mu_N})^{\boldsymbol{T}}(\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\boldsymbol{\Lambda_0})(\boldsymbol{\mu_N})+\boldsymbol{\mu_0^T}\boldsymbol{\Lambda_0}\boldsymbol{\mu_0}\right)}$$

$$p(\boldsymbol{w},\sigma^2|\boldsymbol{t},\boldsymbol{\Phi}) \propto p(\boldsymbol{w}|\sigma^2,\boldsymbol{t},\boldsymbol{\Phi})p(\sigma^2|\boldsymbol{t},\boldsymbol{\Phi})$$

The parameters are updated as follows:

$$\boldsymbol{\Lambda_N} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\Lambda_0}), \boldsymbol{\mu_N} = \boldsymbol{\Lambda_N^{-1}}(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\widehat{\boldsymbol{w}} + \boldsymbol{\Lambda_0}\boldsymbol{\mu_0})$$

$$a_N = a_0 + \frac{N}{2}, b_N = b_0 + \frac{1}{2}(\boldsymbol{t}^T\boldsymbol{t} - (\boldsymbol{\mu_N})^{\boldsymbol{T}}\boldsymbol{\Lambda_N}(\boldsymbol{\mu_N}) + \boldsymbol{\mu_0^T}\boldsymbol{\Lambda_0}\boldsymbol{\mu_0})$$

$$p(\boldsymbol{w}|\boldsymbol{t},\alpha,\beta) = N(\boldsymbol{w}|\boldsymbol{m}_N,\boldsymbol{S}_N)$$

Mean:

$$\boldsymbol{m}_N = \beta\boldsymbol{S}_N\boldsymbol{\Phi}^T t$$

Precision:

$$\boldsymbol{S}_N{}^{-1} = \alpha \boldsymbol{I} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}$$

$\boldsymbol{\Phi}$ is the Design matrix.

$$\log p(\boldsymbol{w}|\boldsymbol{t}, \alpha, \beta) = -\beta E_D(\boldsymbol{w}) - \alpha E_W(\boldsymbol{w}) + constant$$

where

$$E_D(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(t_i - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i))^2 = \frac{1}{2}||\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w}||^2$$

$$E_W(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T \boldsymbol{w}$$

3.3.1.2.5 Evidence Function:

It is the probability of data given the model's prior assumptions.

$$p(\boldsymbol{t}|m) = \int p(\boldsymbol{w}, \sigma^2) p(\boldsymbol{t}|\boldsymbol{\Phi}, \boldsymbol{w}, \sigma^2) d\boldsymbol{w} d\sigma$$

Solution for the integral:

$$p(\boldsymbol{t}|m) = \frac{1}{(2\pi)^{N/2}} \sqrt{\frac{\det(\boldsymbol{\Lambda_0})}{\det(\boldsymbol{\Lambda_N})}} \frac{b_0^{a_0}}{b_n^{a_n}} \frac{\Gamma(a_N)}{\Gamma(a_O)}$$

$$p(\boldsymbol{t}|\alpha, \beta) = \int p(\boldsymbol{t}|\boldsymbol{w}, \beta) p(\boldsymbol{w}|\alpha) d\boldsymbol{w}$$

$$\log p(\boldsymbol{t}|\alpha, \beta) = \frac{M}{2}\log \alpha + \frac{N}{2}\log \beta - E(\boldsymbol{m}_N) - \frac{1}{2}\log|\boldsymbol{S}_N^{-1}| - \frac{N}{2}\log 2\pi$$

where,

$$E(\boldsymbol{m}_N) = \frac{\beta}{2}\left||\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{w}|\right|^2 + \frac{\alpha}{2}\boldsymbol{m}_N^T \boldsymbol{m}_N$$

$$\alpha = \frac{\gamma}{\boldsymbol{m}_N{}^T \boldsymbol{m}_N}$$

$$\frac{1}{\beta} = \frac{1}{N - \gamma}\sum_{i=1}^{N}\left(t_i - \boldsymbol{m}_N{}^T \boldsymbol{\phi}(\boldsymbol{x}_i)\right)^2$$

$$\gamma = \sum_{i=0}^{M-1}\frac{\lambda_i}{\alpha + \lambda_i}$$

λ_i are the eigen values of $\beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}$.

3.3.1.2.6 Bayes Theorem:

$$p(\boldsymbol{w}, \sigma^2|\boldsymbol{t}, \boldsymbol{\Phi}, m) = \frac{p(\boldsymbol{w}, \sigma^2|m) p(\boldsymbol{t}|\boldsymbol{\Phi}, \boldsymbol{w}, \sigma^2, m)}{p(\boldsymbol{t}, m)}$$

$$p(\boldsymbol{w}|\boldsymbol{t}, \alpha, \beta) = \frac{p(\boldsymbol{t}|\boldsymbol{w}, \beta) p(\boldsymbol{w}, \alpha)}{p(\boldsymbol{t}|\alpha, \beta)}$$

3.3.1.2.7 Posterior Predictive Distribution:

$$p(t|\boldsymbol{x}, \boldsymbol{t}, \alpha, \beta) = \int p(t|\boldsymbol{x}, \boldsymbol{w}, \beta)\, p(\boldsymbol{w}|\boldsymbol{t}, \alpha, \beta) d\boldsymbol{w}$$

$$p(t|\boldsymbol{x}, \boldsymbol{t}, \alpha, \beta) = N\left(t\middle|\boldsymbol{m}_N{}^T \boldsymbol{\phi}(\boldsymbol{x}), \sigma_N{}^2(\boldsymbol{x})\right)$$

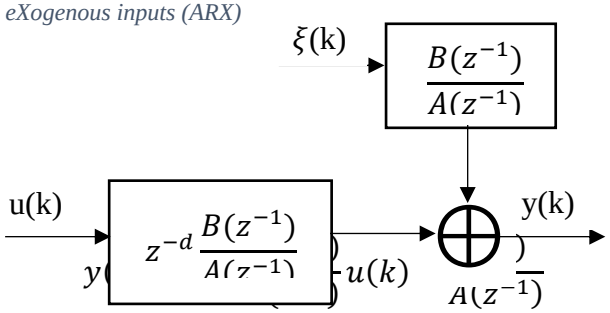
where,

$$\sigma_N{}^2(\boldsymbol{x}) = \frac{1}{\beta} + \boldsymbol{\Phi}^T(\boldsymbol{x})\boldsymbol{S}_N\boldsymbol{\Phi}(\boldsymbol{x})$$

3.3.1.3 ARX

ARX stands for Autoregressive Exogenous. Exogenous means to have an external influence. It refers to the missing input term in AR that is included here. It is a discrete time representation of a linear system.

Figure 4Block diagram of Auto Regressive model with eXogenous inputs (ARX)



$$A(z^{-1}) = a_0 + a_1 z^{-1} + \cdots + a_{na} z^{-n_a}, where\ a_0 = 1$$

$$B(z^{-1}) = b_1 + b_2 z^{-1} + \cdots + b_{nb} z^{-n_b+1}$$

$$y(k) + a_1 y(k-1) + \cdots + a_{na} y(k-n_a) = z^{-d}\big(b_1 u(k) + b_2 u(k-1) + \cdots + b_{nb} u(k-n_b+1)\big) + \xi(k)$$

It is a causal system where the output at the current time step relies on all past inputs and outputs along with the current input.

3.3.2 Non-Linear Regression

Non-Linear Regression techniques include:

3.3.2.1 LSTM

LSTM stands for Long Short-Term Memory. It is a type of Recurrent Neural Network (RNN). RNNs enable features learnt in one part of the network to be quickly generalised to other parts of the sequence as well. They are a popular choice for temporal sequences. Outputs from the previous layers are fed as input to the next layer. LSTMs were proposed as an improvement over conventional RNNs to solve the problem of Vanishing Gradients. This problem occurs during Backward Propagation when gradients computed on the cost function sequentially from back to front decrease exponentially with every iteration causing the step size of update on the weights towards the optimum to drop. It means that the neural network can't capture long-range dependencies.

As a solution to this problem the Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) architectures were proposed as modifications to the existing RNN architectures. The LSTMs have a memory cell in addition that hold the value $c^{<t-1>}$ of the activation from the previous layer. Whether or not this value is carried forward is decided by an Updating factor Γ_u and a Forgetting factor Γ_f. They are first calculated as a linear function of the activation output from the previous layer $a^{<t-1>}$ and the current time-step input $x^{<t>}$, and then a sigmoid function ($a(z) = \frac{1}{1+e^{-z}}$) is applied to scale the output to a value between 0 and 1.

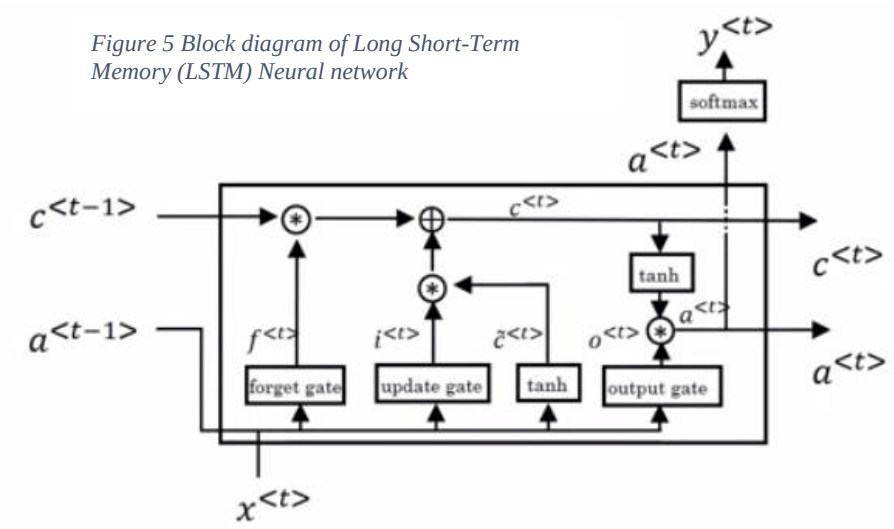


Figure 5 Block diagram of Long Short-Term Memory (LSTM) Neural network

$$\tilde{c}^{<t>} = \tanh(W_c(a^{<t-1>}, x^{<t>}) + b_c)$$

Updating Factor:

$$\Gamma_u = \sigma(W_u(a^{<t-1>}, x^{<t>}) + b_u)$$

Forgetting Factor:

$$\Gamma_f = \sigma(W_f(a^{<t-1>}, x^{<t>}) + b_f)$$

Output Factor:

$$\Gamma_o = \sigma(W_o(a^{<t-1>}, x^{<t>}) + b_o)$$

$$c^{<t>} = \Gamma_u \tilde{c}^{<t>} + \Gamma_f c^{<t-1>}$$

$$a^{<t>} = \Gamma_o \tanh(c^{<t>})$$

When the Updating factor is close to 1 and the Forgetting factor is 0, the network interprets that it must discard the previous memory cell value and update the current value $\tilde{c}^{<t>}$ also calculated from the activation output from the previous layer $a^{<t-1>}$ and the current time-step input $x^{<t>}$, and instead of a sigmoid function a tanh activation function $(a(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}})$ is applied which gives out a value between -1 and 1.

Similarly, when the Updating factor is close to 0 and the Forgetting factor is 1, the network understands that it must keep the past value from the memory cell and pass it on to the next layer.

The activation output from every layer is computed as the value obtained from applying tanh function to the memory cell value, irrespective of whether it is from the current cell or the previous cell and scaling it using an Output factor Γ_o.

is as shown:

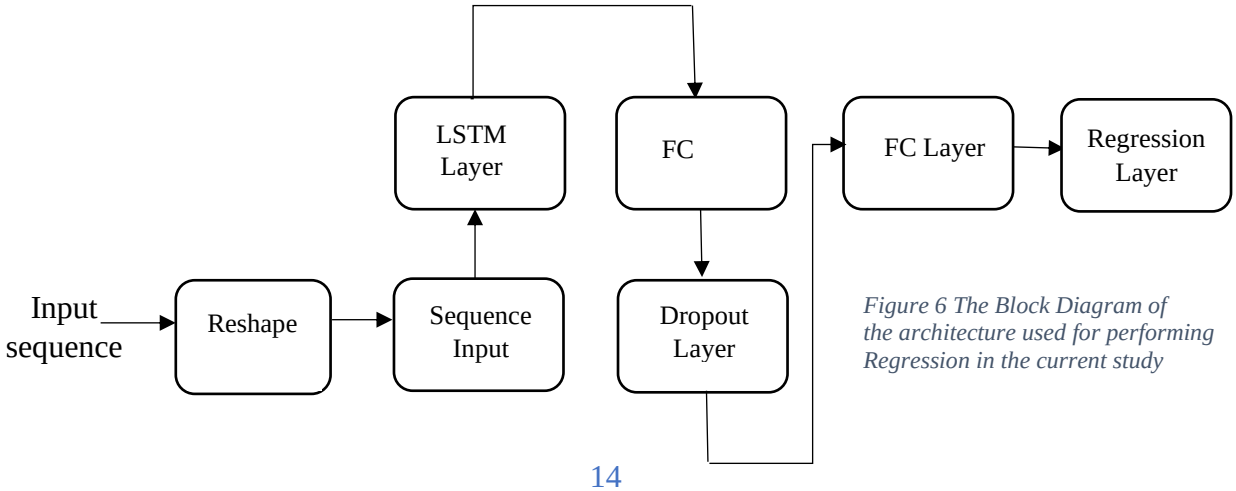


Figure 6 The Block Diagram of the architecture used for performing Regression in the current study

Reshape: Since the input to the LSTM is expecting a sequence, a reshape layer was used to convert the s-EMG input array of dimensions 1945×12 into 1945 cell arrays of size 1× 12 each.

FC Layer: A simple feedforward network layer is also called as a Fully Connected layer. Each neuron computes a linear function $z = \boldsymbol{w}^T x + b$ followed by an activation function $y = g(z)$ which is non-linear. Common choices for an activation function are- Sigmoid function $(a(z) = \frac{1}{1+e^{-z}})$, tanh function $(a(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}})$, ReLU function $(a(z) = \max(0, z))$, Leaky ReLU function $(a(z) = \max(0.01z, z))$

Dropout Layer: Deep Neural Networks esp., when training on smaller training sets can cause the problem of overfitting (also known as the high variance problem). Regularisation is a technique to reduce overfitting. The most common way of regularising a network is by penalising the weights in the cost function which will enable the weights to become sparse thereby keeping the problem of overfitting at check.
Another popular technique used for controlling the high variance problem is the Dropout Regularisation. In this approach a probability is set for eliminating a node in each of the layers of the network under consideration. A dropout probability of 0.5 was used in this case. This helps in diminishing the size of a very deep network thereby controlling the overfitting problem.
Increasing the training set size would also reduce the high variance but is very expensive to do so, therefore, L-2 Regularisation and Dropout Regularisation are the most used techniques for overfitting.

Regression Layer: This layer usually follows a Fully Connected (FC) layer. The Loss function for the network is set in this layer as half-mean-squared-error between the predicted output and the actual output in a supervised learning problem.

$$Loss\ function = \frac{1}{2}\sum_{i=1}^{M}(t_i - y_i)^2$$

where, $t_i = target\ variable, y_i = predicted\ variable, M = no.of\ responses$

Normalisation: It is important for all the input features to be on similar scales with similar variance. If the features on which the neural network is being trained are on different scales, it causes the weights to take on vastly different values giving the cost function an elongated bowl shape appearance and the gradient descent algorithm will end up taking a lot of steps oscillating in perpendicular directions along the path causing the learning to slow down and thereby requiring more time to reach the optimum.

Weights are initialised based on the activation function used to calculate the output. Since the LSTMs use a tanh activation function, the weights are initialised as values picked from a uniform distribution with mean 0 and variance $\frac{2}{n^{[l-1]}+n^{[l]}}$, where $n^{[l-1]}$ is the number of inputs into the node/neuron while $n^{[l]}$ is the number of outputs coming out of the node. This type of initialisation is called the 'Xavier Initialisation' or the 'Glorot Initialisation'.

Hyperparameters: The most important hyperparameters to tune in most applications are the learning rate, the momentum and rms prop factors, the mini-batch size, the no. of hidden units, the no. of layers, and the learning rate decay.

Mini-Batch size: This sets how many training examples are being processed by the learning algorithm per iteration. Optimum choice of mini-batch size is crucial in deciding how fast the learning algorithm can update the weights towards the minimum. For e.g., a mini-batch size of 1 causes the gradient descent problem to become stochastic since with every iteration the learning algorithm will only be training on one training example at a time thereby rendering the path to become noisy. Similarly, while a mini-batch size equal to the entire training set allows for the weights to march towards the minimum in large steps, it can increase the computational time because the entire batch will be sent for training at once with each iteration. Therefore, an ideal choice would be to choose a value in between.

One disadvantage of using a mini-batch gradient descent is that the learning algorithm will propagate the weights towards the minimum with every epoch and take it closer to the optimum but will never allow it to reach the optimum.

Learning Rate (α): Sets the rate for how fast or slow the weights and bias are updated as they approach the optimum. In this case, a learning rate of 0.01 was used.

This problem of mini-batch gradient descent not converging to a set of values at the minimum for the weights and bias can be addressed by using the technique called Learning Rate Decay where the value of learning rate is decreased with the number of epochs run so that as it reaches near the minimum the learning is slowed down allowing it a chance to converge at the minimum of the cost function.

$$\alpha = \frac{1}{1 + decay\ rate * \ epoch\ no.}\alpha_0$$

Optimization Algorithm: An Adam Optimiser was used here.
Adam is short for Adaptive Moment Estimation. It is an amalgamation of RMS Propagation and Gradient Descent with Momentum. These 3 algorithms were proposed as an improvement over the classical gradient descent algorithm for faster learning.

Advantages of using an Adam Optimiser for training:
1. Provides the best of both RMS Propagation and Gradient Descent with Momentum.
2. It works effectively across a variety of architectures.

Weights and bias in an Adam Optimiser are updated as follows:

$$w := w - \frac{V_{dw}^{corrected}}{\sqrt{S_{dw}^{corrected}} + \varepsilon}, b := b - \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}} + \varepsilon}$$

where,

$$V_{dw} = \beta_1 V_{dw} + (1 - \beta_1)dw,\ V_{db} = \beta_1 V_{db} + (1 - \beta_1)db$$

$$V_{dw}^{corrected} = \frac{V_{dw}}{(1 - \beta_1^t)},\ V_{db}^{corrected} = \frac{V_{db}}{(1 - \beta_1^t)},\ \beta_1 = first\ moment$$

$$S_{dw} = \beta_2 V_{dw} + (1 - \beta_2)dw^2,\ \ S_{db} = \beta_2 V_{db} + (1 - \beta_2)db^2$$

$$S_{dw}^{corrected} = \frac{S_{dw}}{(1 - \beta_2^t)} \quad ,S_{db}^{corrected} = \frac{S_{db}}{(1 - \beta_2^t)},\ \ \beta_2 = second\ moment$$

$V_{dw}, V_{db}, S_{dw}, S_{db}$ are all initialised as 0s to begin with.

Essentially, V_{dw} and V_{db} give the exponentially weighted average over the past $\frac{1}{1-\beta_1}$ dw and db values. While S_{dw} and S_{db} give the exponentially weighted average over the past $\frac{1}{1-\beta_2}$ dw^2 and db^2 values. This allows the algorithm to average out any oscillations in directions perpendicular to the path towards the minimum.

This whole scenario is analogous to a ball released from the top of a bowl. The momentum with which it is released determines how fast or slow it reaches the centre of the bowl which is our minimum. V_{dw} and V_{db} are analogous to velocity whereas S_{dw} and S_{db} are analogous to acceleration in the scenario described above. While β plays the role of friction between the ball and the bowl. The bowl is a very good analogy because it represents the shape of a convex shaped quadratic cost function (J).

The first moment (β_1) and second moment (β_2) parameters are also referred to as the "Gradient Decay" and "Squared Gradient Decay" factor. The default values for these factors are 0.9 and 0.999, respectively.

Hyperparameter Tuning: In Deep Learning applications it is advised to perform the tuning using a random search based on prior knowledge of which hyperparameter is likely to effect the desired change in the performance instead of going for a systematic grid search approach. For instance, the choice of learning rate (α) is going to have a more significant impact over a choice of, say, ε added in the denominator while updating weights and bias in Adam Optimisation algorithm.

Gradient Decay factor (β_1) tuning can be done by performing the search on a logarithmic scale, because a change in β_1 value of 0.0005 from 0.999 to 0.9995 could mean that the number of past values of the gradients being considered has gone up by 2-fold, i.e., from 1000 to 2000. Similarly, for Squared Gradient Decay factor (β_2).

GPU: The regression method using LSTM was run both with and without GPU.

One of the reasons why a Deep Neural Network is preferred over a Shallow Neural Network is because they can learn more complex features more efficiently i.e., they allow the starting layers to learn the low-level features and then building on those in the deeper layers to learn more complex features.

3.3.2.2 Non-Linear ARX:

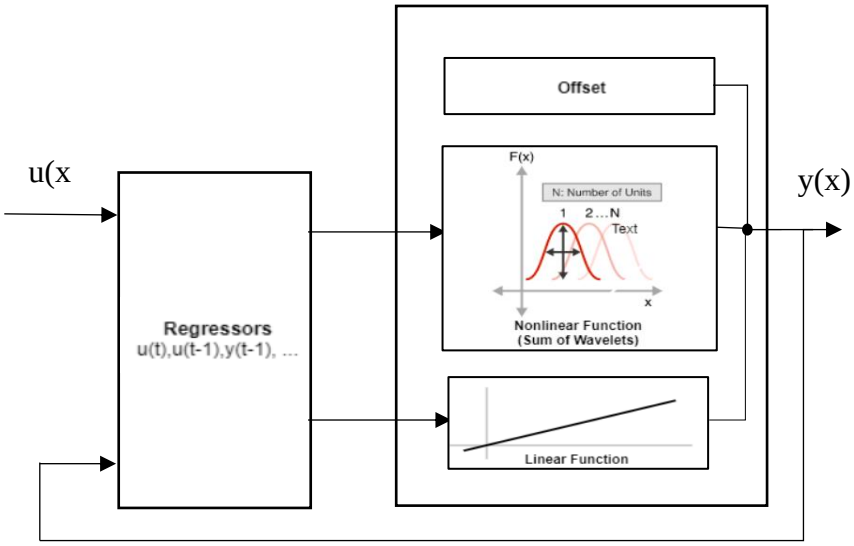


Figure 7 Block diagram of Non-Linear ARX (NARX)

A linear model predicts $y_p(t)$ as a linear summation of the weighted products of the past inputs and past outputs. Assuming the delay (n_k) as 0, it can be written as:

$$y_p(t) = [-a_1 \quad -a_2 \quad \dots \quad -a_{n_a} \quad b_1 \quad b_2 \quad \cdots \quad b_{n_b}] * \begin{bmatrix} y(t-1) \\ y(t-2) \\ \vdots \\ y(t-n_a) \\ u(t) \\ u(t-1) \\ \vdots \\ u(t-n_b-1) \end{bmatrix}$$

where, $n_a = number\ of\ past\ inputs\ used\ in\ the\ current\ output\ prediction$

$n_b = number\ of\ past\ outputs\ used\ in\ the\ current\ output\ prediction$

$n_k = delay\ from\ input\ in\ the\ current\ output\ prediction$

Whereas a non-linear model predicts $y_p(t)$ as follows:

$$y_p(t) = F(y(t-1), y(t-2), \dots, u(t), u(t-1), \dots)$$

where, $y(t-1), y(t-2), \dots, u(t), u(t-1), \dots)$ are referred to as the regressors.

The Output function (F) used in this study is a weighted sum of wavelets. It is a combination of linear weights (L), nonlinear wavelet function (W+ S) and an offset (y_0), as shown:

For a Multi-Input Single-Output system where: $\boldsymbol{X}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_m(t) \end{bmatrix}_{m\times 1}$ where x are the regressors $x_1(t) = y(t-1)$, $x_2(t) = y(t-2)$,…, $x_k(t) = u(t)$,…, $y(t) = scalar = y_p(t)$.

$$y(t) = y_0 + (\boldsymbol{X}(t) - \bar{\boldsymbol{X}})^T PL + W(\mathbf{X}(t)) + S(\mathbf{X}(t))$$

where, P= Projection Matrix of dimensions $m \times p$.

L= Weights Matrix of dimensions $p \times 1$.

W(**X**)= Translated Wavelets.

S(**X**)= Translated Scaling functions, referred to as Scaletes, in short.

$$W(\boldsymbol{X}) = \sum_{i=1}^{d_w} w_i f_w(b_i(\boldsymbol{X} - \bar{\boldsymbol{X}})^T \boldsymbol{Q} - \boldsymbol{c}_i)$$

where, w_i = wavelet coefficients, $f_w(x) = e^{-\frac{xx^T}{2}}, x$ =Linear weighted combination of inputs $\boldsymbol{X}$ with an offset $\boldsymbol{c_i}$ of dimensions $1 \times q$, b_i = wavelet dilations, Q = Projection Matrix of dimensions $m \times q$, c_i =wavelet translations of dimensions $1 \times q$.

$$S(\boldsymbol{X}) = \sum_{i=1}^{d_w} s_i f_s(d_i(\boldsymbol{X} - \bar{\boldsymbol{X}})^T \boldsymbol{Q} - \boldsymbol{e}_i)$$

where, s_i = scaling coefficients, $f_s(x) = (\dim(x) - xx^T)e^{-\frac{xx^T}{2}}, x$ =Linear weighted combinations of inputs $\boldsymbol{X}$ with an offset $\boldsymbol{c_i}$ of dimensions $1 \times q$, d_i = scaling dilations, Q = Projection Matrix of dimensions $m \times q$, c_i =wavelet translations of dimensions $1 \times q$.

w_i, b_i, s_i, d_i are all scalar coefficients.

First round of parameter estimation is done quickly (as discussed in []) and only then an iterative algorithm is used to further refine the parameters.

3.3.2.3 Generative Adversarial Networks (GANs):

GANs fall under the category of 'generative modelling'. They comprise of two components mainly- the Generator and the Discriminator. Each component shall be discussed separately.

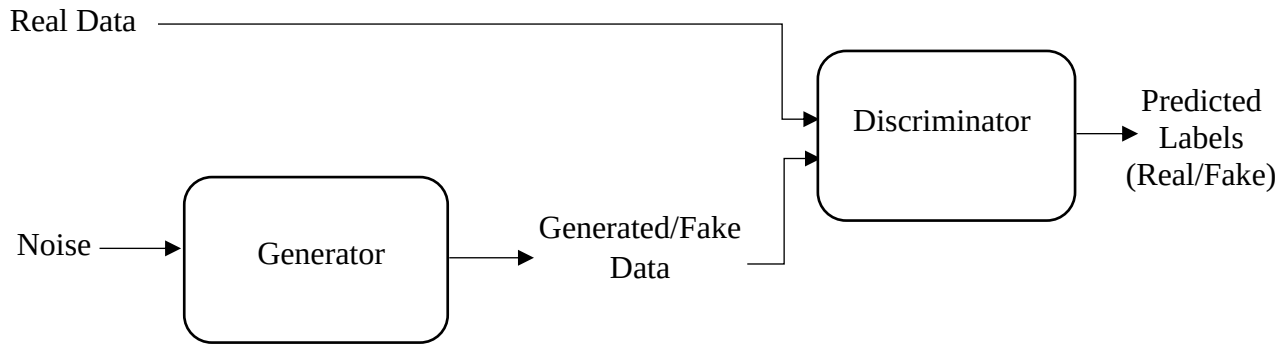


Figure 8 Block diagram of Generative Adversarial Networks

Generator:

Objective of the Generator is to generate samples to deceive the discriminator. In other words, to approximate actual inputs. A standard LSTM network was used for the Generator.

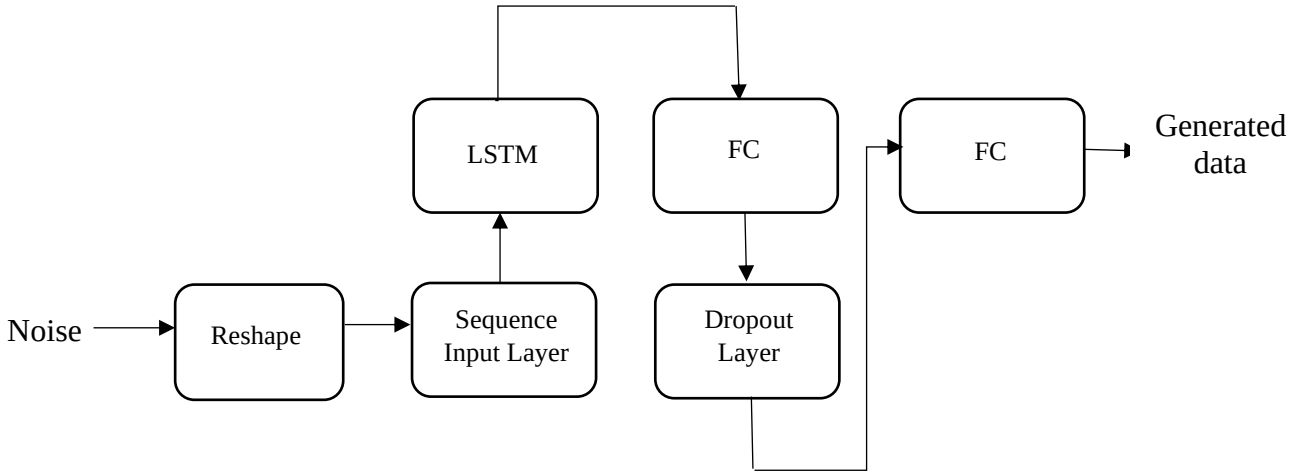


Figure 9 Block diagram of LSTM based GAN generator network

Loss of the Generator is calculated as the negative mean of log of the probability assigned by the sigmoid activation function of the output layer of the Discriminator.

$$Loss_G = -\text{mean}(\log(\text{p}_{\text{Generated}}))$$

During training, the Generator tries to reduce this loss by attempting to deceive the Discriminator into thinking that the generated data is real by making the output layer in the Discriminator assign a probability value closer to 1 which would mean that the generated data is being classed as real.

Discriminator:

Objective of the Discriminator is to discriminate between actual data and the fake data generated by the generator. A Bi-Directional LSTM was used for the Discriminator.

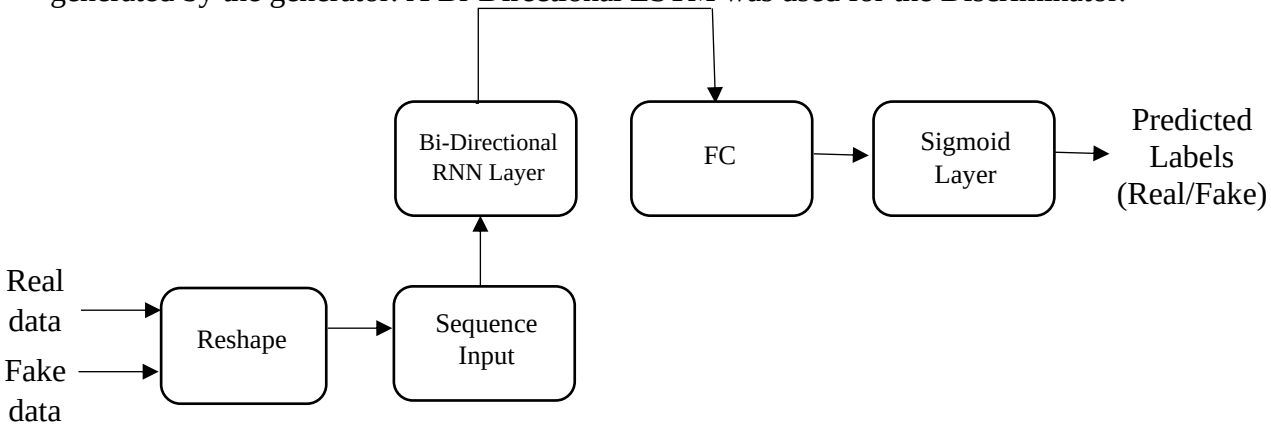


Figure 10 Block diagram of LSTM based GAN discriminator network

A sigmoid activation function is applied on the output of the feedforward calculation since it is a binary classification problem where the Discriminator must tell if the input data is real or generated.

Loss function of the Discriminator at any given time is defined as the negative sum of means of log of the probabilities assigned by the sigmoid activation function for both real and fake data, indicating to what certainty the Discriminator thinks that they are real.

$$Loss_D = -\text{mean}(\log(\text{p}_{\text{Real}})) - \text{mean}(\log(1 - \text{p}_{\text{Generated}}))$$

During training, the Discriminator tries to reduce this loss by classing the data as fake or real accurately despite the deceptions of the Generator to make the generated data seem real.

In this process of outsmarting each other, after a certain number of epochs run, the Generator gets better at generating convincingly real data while the Discriminator get better at making

out which is fake or real. At one point the generator is expected to mimic the real data so well that it replicates the model that was generating the output to begin with.
An Adam optimiser was used for training to minimise the loss function in both the Discriminator and the Generator. The Adam Optimiser and the hyperparameters involved were discussed in detail in section 3.3.4.
The hyperparameters were tuned in a similar way to that discussed in section 3.3.4.

3.4 Classification

3.4.1 K-Nearest Neighbours:

It is a non-parametric algorithm. It works on a similarity measure i.e., when a new data input is given the algorithm compares this new data to the already available data in the training set and only then makes the classification decision. It makes this decision based on the K-nearest neighbours to this new data that it has identified by calculating the Euclidian distance between the training set input and the new data input. Once it has identified its K- nearest neighbours it counts the number of data points in each category and assigns the new data point to the category that most of the K- nearest neighbours belong to.

In this study, the s-EMG signals were input to the K-NN algorithm, and the algorithm classified them into one of 10 categories based on the 9-experiments and the rest periods.

3.4.2 LSTM:

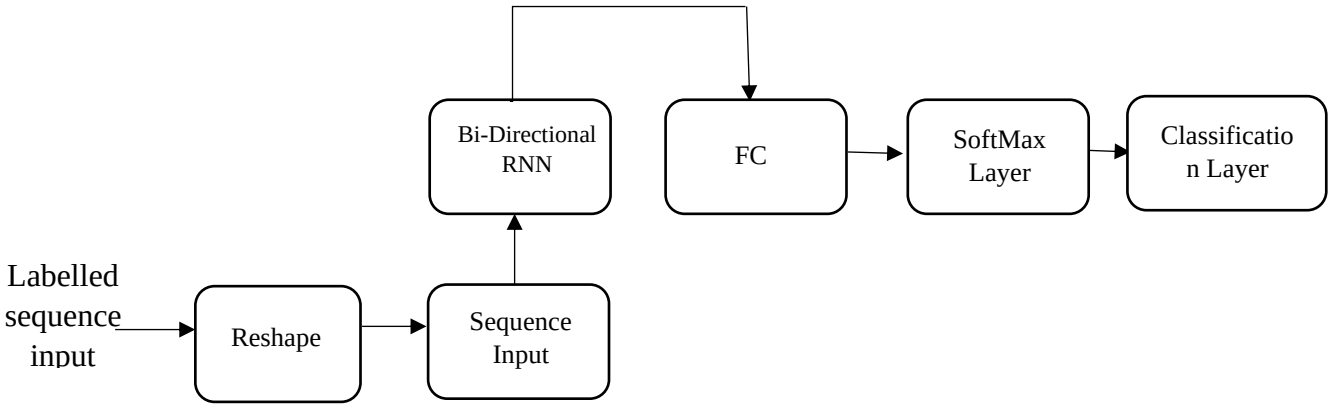


Figure 11 Block diagram of LSTM classifier

Bi-Directional RNN Layer:
This can only be used when all time steps of the input sequence are available. It allows the network to learn from both past and future input features. It has two feedforward layers: a forward layer and a backward layer.
The forward layer starts calculating the activations from $\overrightarrow{a_1}^{<1>}$, $\overrightarrow{a_2}^{<2>}$, ...,and goes up to $\overrightarrow{a_T}^{<T>}$. Whereas the backward layer starts calculating the activations from $\overleftarrow{a_T}^{<T>}$, ..., and goes up to $\overleftarrow{a_2}^{<2>}$, $\overleftarrow{a_1}^{<1>}$.
Therefore, at any timestep t, the output $y_t^{<t>}$ is calculated as:

$$\hat{y}_t^{<t>} = g(W_y[\overrightarrow{a_t}^{<t>}, \overleftarrow{a_t}^{<t>}] + b_y)$$

The architecture represents an acyclic graph.

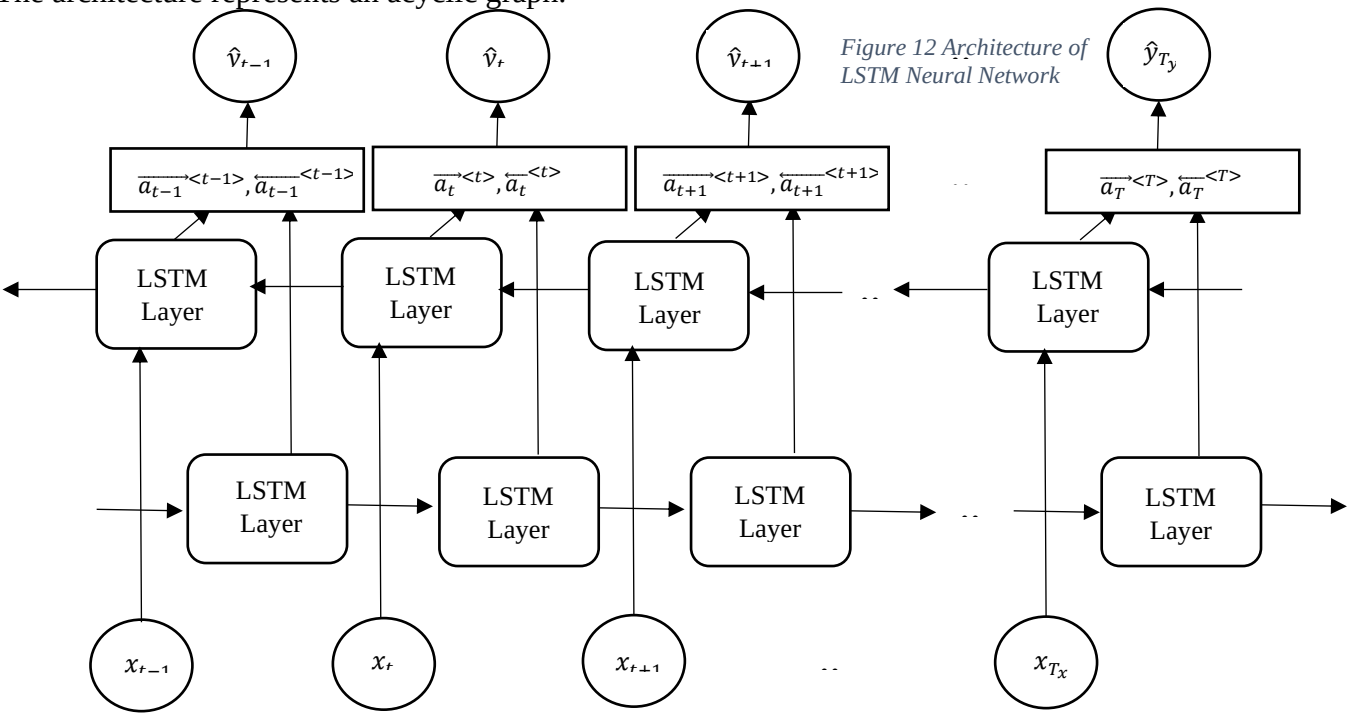


Figure 12 Architecture of LSTM Neural Network

SoftMax Layer: A SoftMax layer is used in non-binary classification problems where the classification output is not 1 of 2 classes but 1 of multiple possible classes (say, c). This is a generalisation of logistic regression. It assigns a probability for each of the c classes such that they all sum up to 1. It is usually placed in the output layer.

It is computed as: $z^{[L]} = W^{[L]}a^{[L-1]} + b^{[L]}, a^{[L]} = \frac{e^{z_j[L]}}{\sum_{j=1}^{c} e^{z_j[L]}}$.

$a^{[L]}$ is known as the SoftMax activation function. It outputs a $c \times 1$ vector of probabilities of the input belonging to each of the classes. The class corresponding to the max probability value is output as the final class label.

For an n^{th} input sample, the output from the SoftMax layer would be:

$$a_n^{[L]} = \begin{bmatrix} y_{n1} \\ \vdots \\ y_{nc} \end{bmatrix}_{c \times 1}$$

With deeper neural networks the model can even learn non-linear decision boundaries to distinguish between the different classes.

Classification Layer: Loss is calculated as cross-entropy loss in the classification layer.

$$Loss\ function = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{c} w_i t_{ni} \ln(y_{ni})$$

where, N = number of samples, c= number of classes, t_{ni}= ground truth label indicating that n^{th} sample belongs to i^{th} class, y_{ni}= predicted probability of an n^{th} input sample belonging to the i^{th} class, w_i = weight for the i^{th} class.

4 Results and Analysis:

In this section, the outcomes of performing the Regression and Classification techniques on s-EMG data mentioned in Methodology shall be discussed.

4.1 Regression Techniques:

3 linear and 3 non-linear techniques were experimented with on s-EMG data vs force from Finger Movements.

4.1.1 Linear Regression:

Within Linear Regression, 3 techniques were applied, namely, the Ordinary Least Squares, the Bayesian Least Squares, and Autoregressive with Exogenous Input. In each method, six cases were studied. The cases studied include the case with all 12 s-EMG input signals present, only 10 s-EMG signals present, and just 2 s-EMG signals present along with 1-finger force data and 2-fingers force data as output. The purpose of considering the 10 s-EMG case and the 2 s-EMG case is to include the experience of a Transradial Amputee and an Elbow Disarticulation Amputee, respectively. In specific to enable the myoelectric prosthesis to make out from the available number of sensor points, with what force the person wants to move the finger.

4.1.1.1 Least Squares:

4.1.1.1.1 No. of s-EMG signal inputs considered=2:

Modelling was first attempted by considering 2 s-EMG signals from the electrodes placed on the biceps brachii and triceps brachii of an intact person to replicate the experience of an Elbow Disarticulation Amputee. The purpose was to see if it is possible to tell with how much force a finger was moved from the limited number of sensor data available.

Two output cases were studied from the 2 s-EMG signals available.

4.1.1.1.1.1 Index Finger:

The first case was to see if it was possible to model a system of good fit that can accurately map the data to force of one finger movement (index finger, in this case)

4.1.1.1.1.2 Index and Middle Finger:

And the second case was to see how accurate the mapping was from the data to the force of 2 finger movements (index and middle finger, in this case).

The plots for the 2 cases suggest that the model was able to track the middle finger movement closely whereas for the index finger it was accurately able to follow the movement but not the force magnitude, i.e., the model was able to track the low range force magnitude and movement of the secondary finger more accurately than it was able to track the high range force magnitude of the primary finger movement.

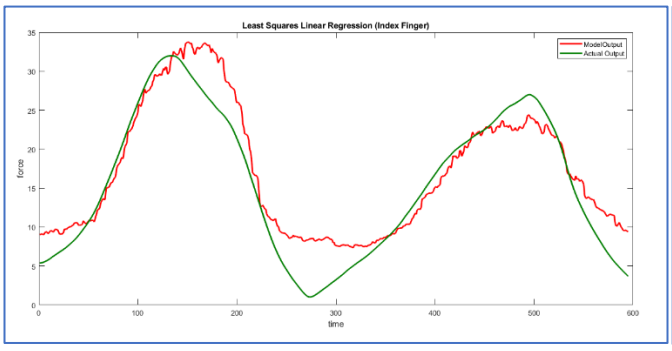


Figure 13 Least square regression (Actual vs Model) output: 2 input 1 output case

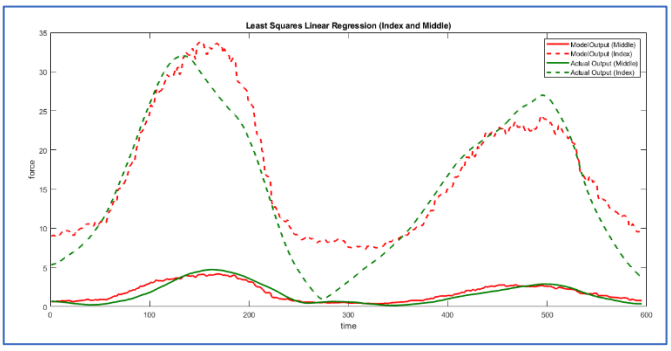


Figure 14 Least square regression (Actual vs Model) output: 2 input 2 output case

4.1.1.1.2 No. of s-EMG signal inputs considered=10:

Modelling was then attempted by considering 10 s-EMG signals from the electrodes placed around the forearm and on the biceps brachii and triceps brachii of an intact person to replicate the experience of an Transradial Amputee. The purpose was to see if it is possible to

tell with how much force a finger was moved from the limited number of sensor data available.

Two output cases were studied from the 10 s-EMG signals available.

4.1.1.1.2.1 Index Finger:

The first case was to see if it was possible to model a system of good fit that can accurately map the data to force of one finger movement (index finger, in this case)

4.1.1.1.2.2 Index and Middle Finger:

And the second case was to see how accurate the mapping was from the data to the force of 2 finger movements (index and middle finger, in this case).

The plots for the 2 cases suggest that the model was able to track the middle finger and the index finger movements closely i.e., the model was able to track the low range force magnitude of the secondary finger movement as well as it was able to track the high range force magnitude of the primary finger movement.

The performance for the index finger movement in both 1- and 2-output cases was significantly seen to improve in the 10 s-EMG case compared to when only 2 s-EMG signal data was available which is as it should be because with more signal data available tracking the finger movements becomes easier.

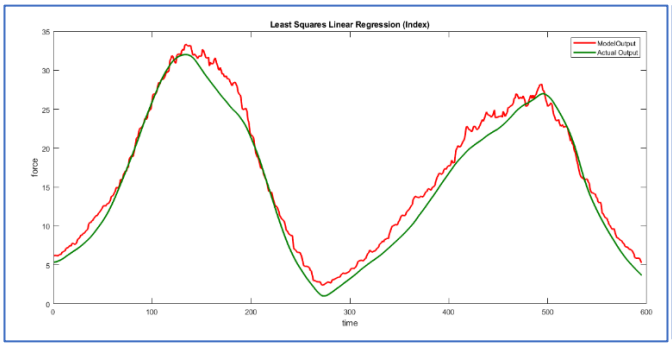


Figure 15 Least square regression (Actual vs Model) output: 10 input 1 output case

Figure 16 Least square regression (Actual vs Model) output: 10 input 2 output case

4.1.1.1.3 No. of s-EMG signal inputs considered=12:

Modelling was then attempted by considering all the 12 s-EMG signals. Two output cases were studied from all the 12 s-EMG signals available.

4.1.1.1.3.1 Index Finger:

The first case was to see if it was possible to model a system of good fit that can accurately map the data to force of one finger movement (index finger, in this case)

4.1.1.1.3.2 Index and Middle Finger:

And the second case was to see how accurate the mapping was from the data to the force of 2 finger movements (index and middle finger, in this case).

The plots for the 2 cases suggest that the model was able to track the middle finger and the index finger movements closely i.e., the model was able to track the low range force magnitude of the secondary finger movement as well as it was able to track the high range force magnitude of the primary finger movement.

The performance for the index finger movement in both 1- and 2-output cases was observed to be similar to the 10 s-EMG case when all the12 s-EMG signals were considered.

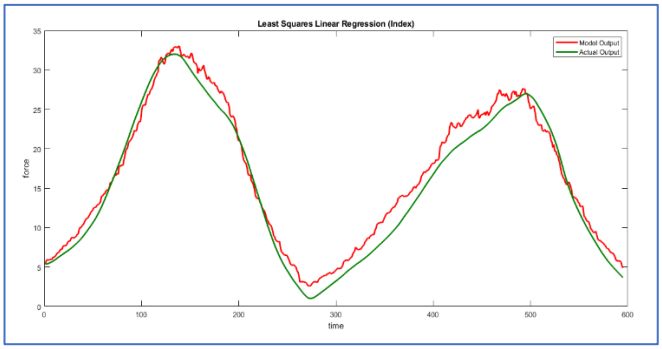


Figure 17 Least square regression (Actual vs Model) output: 12 input 2 output case

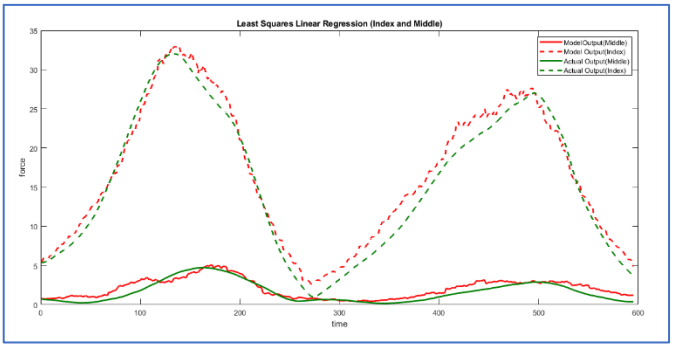


Figure 18 Least square regression (Actual vs Model) output: 12 input 2 output case

4.1.1.2Bayesian Least Squares:

A multi-variate Bayesian approach was used with the prior assumption on the weights set as a gaussian distribution with mean= $\mathbf{0}_{2n\times 1}$ and covariance=$\alpha^{-1}\boldsymbol{I}_{2n\times 2n}$, where n= number of s-EMG features. A column of 1s' was appended to the input feature matrix to account for the intercept and is denoted as $\boldsymbol{\Phi}$.

The values for α and β were got by iteratively trying to maximize the log of evidence function calculated as an integral of the posterior function weighted by likelihood function with respect to the weights variable. The range for α and β throughout the study were in between 10^{-10} to 10^{-11} and 1 to 1.5, respectively, depending on the case under study.

The model was first trained on the $1^{st}, 2^{nd}, 3^{rd}, and\ 4^{th}$ repetitions of the fingers' movements and then tested on $5^{th}, and\ 6^{th}$ repetitions.

The 6 cases considered here are the same as that in Least Squares Regression discussed in section 4.1.1.1.

4.1.1.2.1 No. of s-EMG signal inputs considered=2

4.1.1.2.1.1 Index Finger:

When 1 finger movement was modelled from just 2 s-EMG signals, the plot showed that it was able to track the finger movements quite accurately, but the force magnitude estimated by the model was slightly distorted.

The right-hand side plot was got by implementing Multivariate Bayesian Linear regression step-by-step mathematically and was later validated by using a MATLAB built-in function. As the results show, the plot was an exact match.

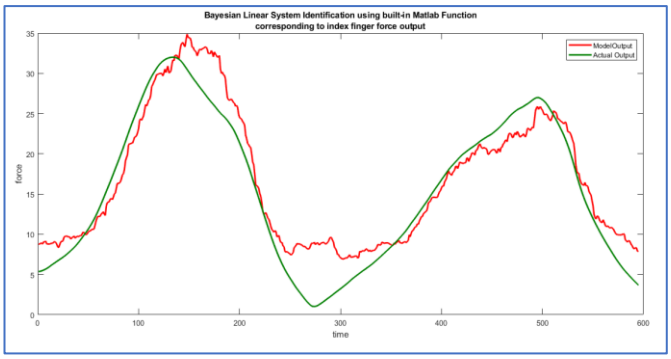


Figure 19 Bayesian System ID of index finger (Actual vs Model) output: 2 input 1 output case

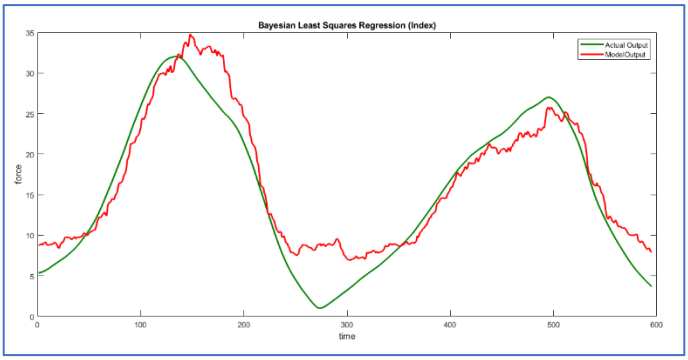


Figure 20 Bayesian Least square regression (Actual vs Model) output: 2 input 1 output case

Table 1 The prior distribution of the parameters, the intercept, and the noise variance

Parameters	Mean	Std. Dev	CI95	Positive	Distribution
Intercept	0	70.71	(-141.273, 141.273)	0.5	t(0,(57.74)^2 ,6)
s-EMG11	0	2.2361× 10^5	(-4.46× 10^5,4.46× 10^5)	0.5	t(0, (1.82 × 10^5)^2, 6)
s-EMG12	0	2.2361× 10^5	(-4.46× 10^5,4.46× 10^5)	0.5	t(0, (1.82 × 10^5)^2, 6)
Sigma2	0.5	0.5	(0.138,1.616)	1	IG(3,1)

Note: The prior table will be the same for all the cases since no changes were made to the prior assumptions case-wise.

Table 2 The posterior distribution of the 2 parameters, the intercept, and the noise variance in the 2-inputs, 1-output case for the index finger

Parameters	Mean	Std. Dev	CI95	Positive	Distribution
Intercept	-9.1086	.7589	(-10.596, -7.621)	0	t(-9.11,(0.76)^2 ,1.4× 10^3)
s-EMG11	3.6583× 10^6	2.8193× 10^5	(3.10× 10^6,4.21× 10^6)	1	t(3.65× 10^6, (2.81 × 10^5)^2, 1.4 × 10^3)
s-EMG12	3.1276× 10^5	18198.1923	(2.77× 10^5, 3.48 × 10^5)	1	t(3.12× 10^5, (18198.1923)^2, 1.4 × 10^3)
Sigma2	12.9641	.4986	(12.023,13.977)	1	IG(678,0.00011)

$1^{st}, 2^{nd}, 3^{rd}$ Rows: Contain the prior and posterior Mean($\boldsymbol{m_N}$) and Std. Dev($\sqrt{\boldsymbol{S_N}}$) for the 3 parameters: 2 corresponding to the 2 s-EMG features since this is a 2-input case and 1intercept.

4^{th} Row: Contain the prior and posterior of mean and standard deviation of the noise variance $\frac{1}{\beta}$. The posterior mean and std. dev are calculated from the prior assumption that it follows an inverse-gaussian distribution $IG(a_0, c_0 = \frac{1}{b\_0})$ whose probability density function is given as: $f(\sigma^2) = \frac{\sigma^{2^{-(a+1)}} e^{-\left(\frac{c}{\sigma^2}\right)}}{\Gamma(a)\left(\frac{1}{c}\right)^a}, where\ \sigma^2 = \frac{1}{\beta}, c = \frac{1}{b}$.

The values seen in the columns were obtained this way:

4^{th} Column:

The posterior shape a_n and scale c_n are calculated as: (Details were given in section 3.3.1.2.3)

$$\boldsymbol{\Lambda_N = (\Phi^T \Phi + \Lambda_0), \mu_N = \Lambda_N^{-1}(\Phi^T \Phi \hat{w} + \Lambda_0 \mu_0)}$$

$$a_N = a_0 + \frac{N}{2}, b_N = b_0 + \frac{1}{2}(\boldsymbol{t}^T\boldsymbol{t} - (\boldsymbol{\mu_N})^T\boldsymbol{\Lambda_N}(\boldsymbol{\mu_N}) + \boldsymbol{\mu_0}^T\boldsymbol{\Lambda_0}\boldsymbol{\mu_0}), c_N = \frac{1}{b_N}$$

where, $a_0 = \frac{v_0}{2}$, $b_0 = \frac{v_0 s_0^2}{2}$, $vs^2 = (\boldsymbol{t} - \boldsymbol{\Phi}\widehat{\boldsymbol{w}})^T(\boldsymbol{t} - \boldsymbol{\Phi}\widehat{\boldsymbol{w}})$ and v =N-k, N= Total number of samples in the Training Set, k=no. of coefficients.

In this study, the prior mean $\boldsymbol{\mu_0} = \mathbf{0}$ and precision $\boldsymbol{\Lambda_0} = \alpha\boldsymbol{I}$ of weights were used.
The shape is calculated as:

$$a_N = 3 + \frac{size(Training\ set)}{2} = 3 + \frac{1350}{2} = 678$$

Similarly, the scale is calculated as:

For $\boldsymbol{m_N} = \begin{bmatrix} -9.1086 \\ 3.6583 \times 10^6 \\ 3.1276 \times 10^5 \end{bmatrix}_{3\times1}$, $\boldsymbol{S_N^{-1}} = \begin{bmatrix} 1350.0000 & .00577 & .037924 \\ 0.00577 & .59513 \times 10^{-81} & 1.80096 \times 10^{-7} \\ 0.03792 & .80096 \times 10^{-71} & .36304 \times 10^{-6} \end{bmatrix}_{3\times1}$

$$b_N = 1 + \frac{1}{2}\left(\boldsymbol{t}^T\boldsymbol{t} - (\boldsymbol{m_N})^T\boldsymbol{S_N^{-1}}(\boldsymbol{m_N}) + \mathbf{0}\right) = 1 + \frac{1.7788 \times 10^4}{2} = 8.8948 \times 10^3$$

$$c_N = \frac{1}{b_N} = 1.1242 \times 10^{-4} \approx 0.00011$$

Therefore, the posterior Distribution for noise variance is IG(678, 0.00011).

1^{st}, 2^{nd} Columns:
The Mean and standard deviation are calculated from the posterior shape a_N and scale c_N, by substituting the values got previously as:

$$Mean = \frac{b_N}{(a_N - 1)} = \frac{8.8948 \times 10^3}{(678 - 1)} \approx 12.964(in\ this\ case),$$

$$Variance = \frac{b_N^2}{(a_N - 1)^2(a_N - 2)} = \frac{\left(8.8948 \times 10^3\right)^2}{(678 - 1)^2(678 - 2)} \approx 0.2486$$

$$std.dev = \sqrt{Variance} \approx 0.4986(in\ this\ case),$$

3^{rd} Row: Contains the prior and posterior confidence intervals for the population means of the 4 random variables in consideration, which are the intercept, the two weights corresponding to the 2 input features and the noise variance. The range starts of wide in the prior assumptions table and gets narrower in the posterior table.

The plot below shows a comparison of the prior and posterior distributions of the intercept, the 2 weights corresponding to the 2-input s-EMG features and the noise variance based on the calculations above.

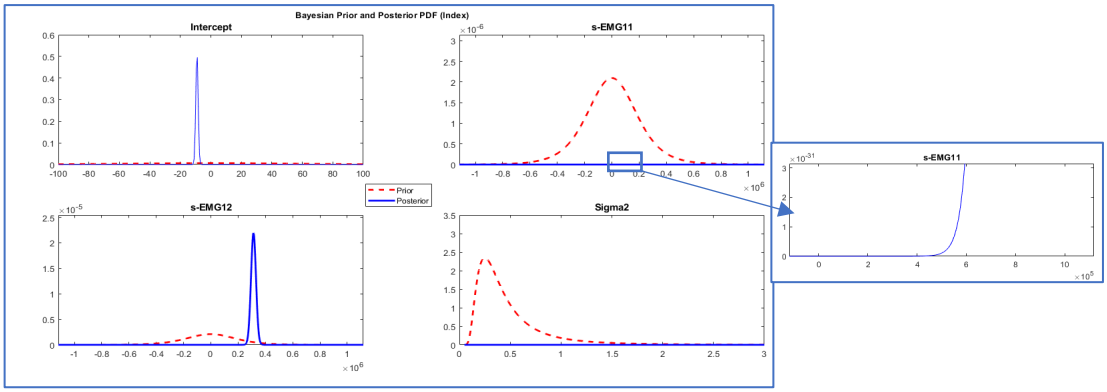


Figure 21 Bayesian Prior and Posterior PDF of index finger for 2 input 1 output case

The following CI plot shows the range around the most frequently occurring sample mean values of the 595 random variables where the population means of the 595 output predictions can be found with 95% certainty. It is calculated as: $\mu \pm \frac{\sigma}{\sqrt{N}}$, where μ and σ are the sample mean and standard deviation, N is the sample size. $\frac{\sigma}{\sqrt{N}}$ is known as the standard error. As N increases the standard error decreases narrowing the range for where the population mean could lie with 95% probability thereby increasing the certainty that the frequently occurring sample mean is very close to the population mean. The CI plot in the 2-inputs, 1-output case shows that the interval is wider to what will be seen in the 10-inputs and 12-inputs cases.

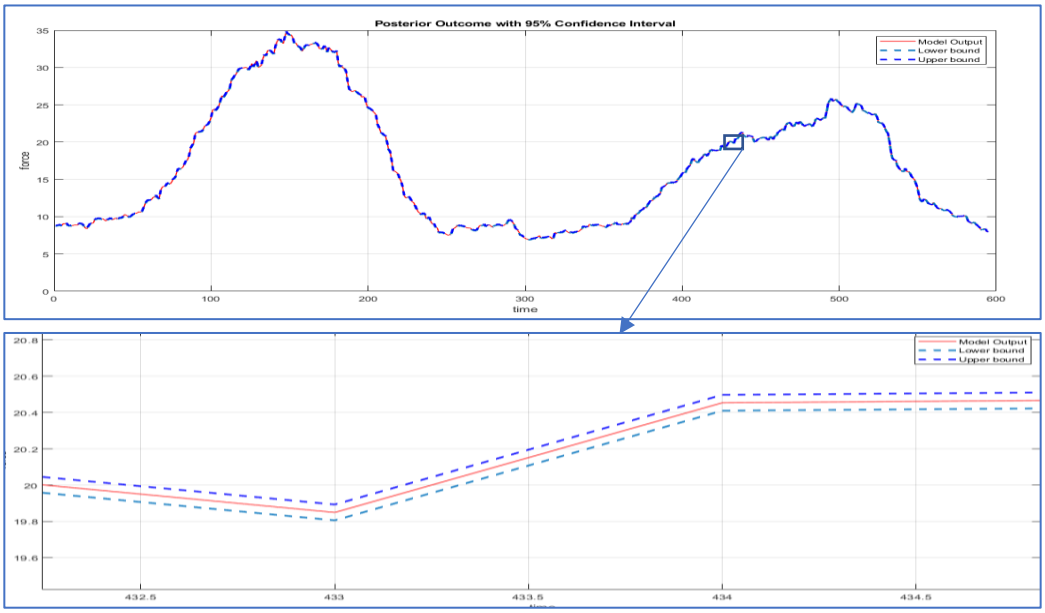


Figure 22 Bayesian Posterior predictive distribution with 95% confidence interval (2 input 1 output case)

4.1.1.2.1.2 Index and Middle Finger:

When 2 finger movements were modelled from just 2 s-EMG signals, the plot showed that it was able to track the finger movements quite accurately for both the fingers, but the force magnitude estimated by the model was slightly more distorted for the index finger.

The top plot was got by implementing Multivariate Bayesian Linear regression step-by-step mathematically and was later validated by using a MATLAB built-in function(shown in the bottom plot). As the results show, the plot was an exact match.

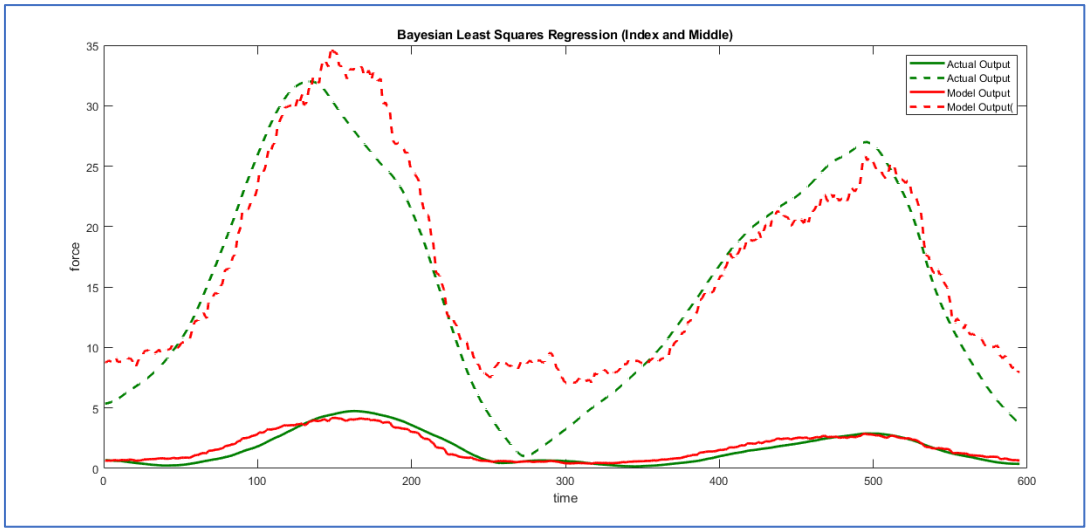


Figure 23 Bayesian Least square regression (Actual vs Model) output: 2 input 2 output case

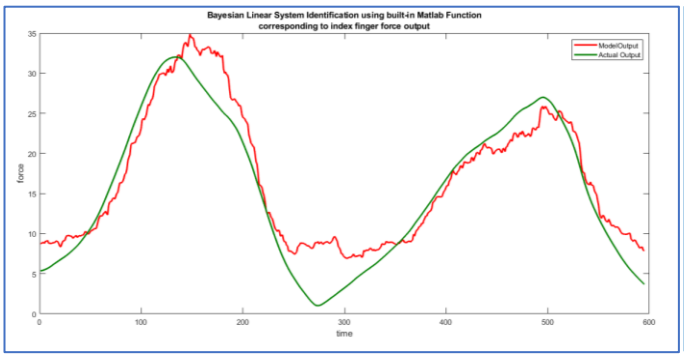


Figure 24 Bayesian System ID of index finger (Actual vs Model) output: 2 input 2 output case

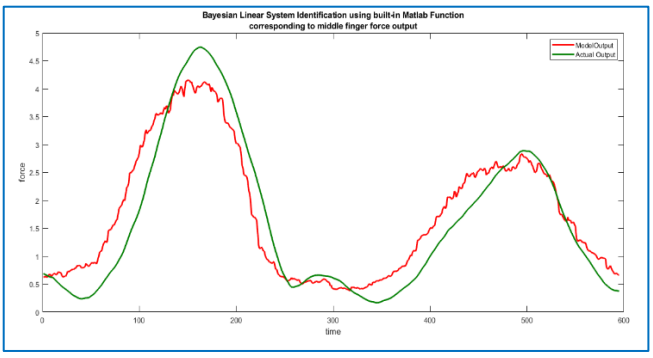


Figure 25 Bayesian System ID of middle finger (Actual vs Model) output: 2 input 2 output case

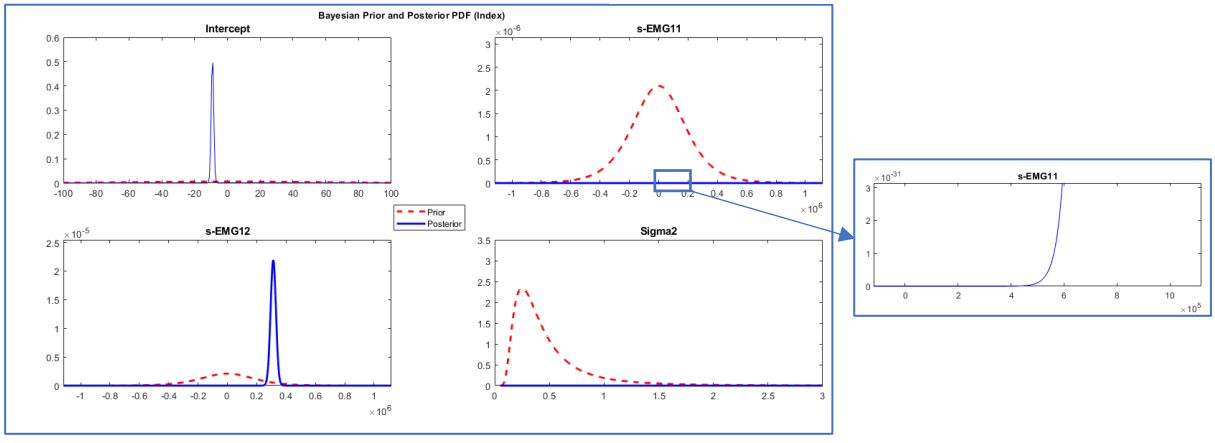


Figure 26 Bayesian Prior and Posterior PDF of index finger for 2 input 2 output case

Table 3 The posterior distribution of the 2 parameters, the intercept, and the noise variance in the 2-inputs, 2-outputs case for the middle finger

Parameters	Mean	Std Dev	CI95	Positive	Distribution
Intercept	-0.7786	0.1709	(-1.114,-0.444)	0	$t(-0.78,(0.17)^2,1.4x10^3)$
sEMG1	$1.01\ x10^5$	$6.34\ x10^4$	$(-2.32\ x10^4,2.25\ x10^5)$	0.944	$t(1.01\ x10^5,(6.34\ x10^5)^2,1.4x10^3)$
sEMG2	$6.72\ x10^4$	4098.77	$(5.92\ x10^4,7.52\ x10^4)$	1	$t(6.72\ x10^4,(4095.75)^2,1.4x10^3)$
Sigma2	0.6576	0.0253	(0.61,0.709)	1	IG(678,0.00022)

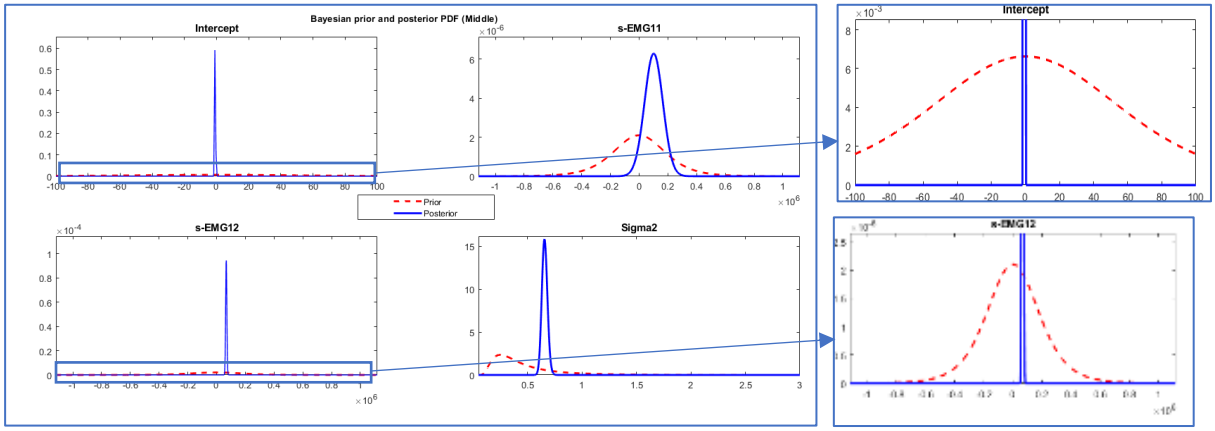


Figure 27 Bayesian Prior and Posterior PDF of middle finger for 2 input 2 output case

The figure shows the 95% CI plot for the 2-inputs, 2-outputs case when both the input finger and the middle finger are considered. The left plot shows the zoomed in range for the index finger and the right plot shows the zoomed in range for the middle finger. The CI in the left plot is narrower compared to that in the right plot indicating that the output prediction certainty was more for the index finger movement than it was for the middle finger movement case.

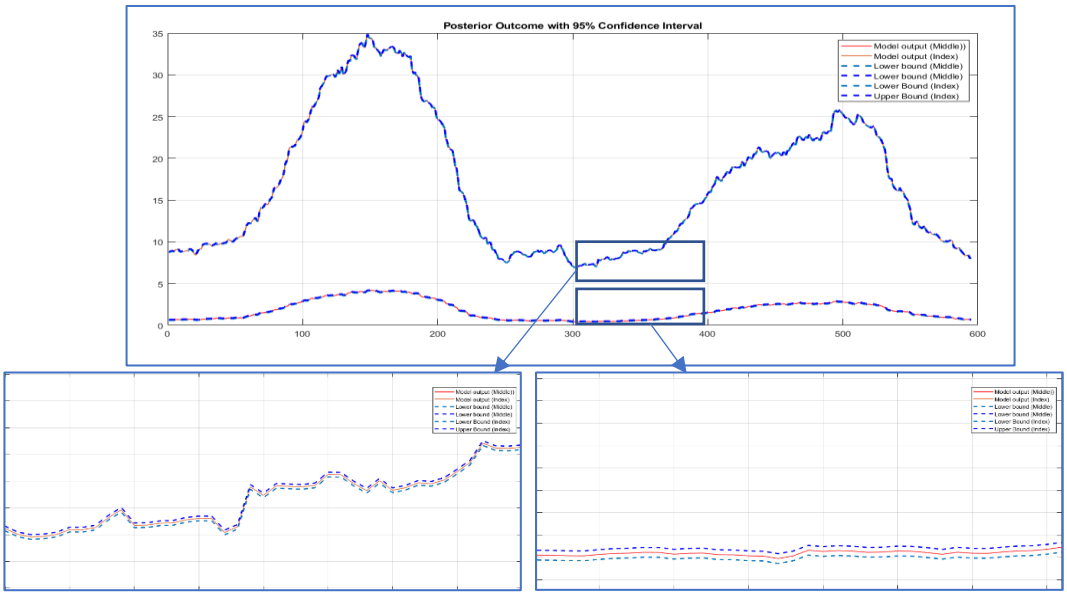


Figure 28 Bayesian Posterior predictive distribution with 95% confidence interval (2 input 2 output case)

4.1.1.2.2 No. of s-EMG signal inputs considered=10

4.1.1.2.2.1 Index Finger:

When 1 finger movement was modelled from 10 s-EMG signals, the plot showed that it was able to track the finger movements quite accurately, and the force magnitudes estimated by the model were also close to the actual values despite a few distortions.

The right-hand side plot was got by implementing Multivariate Bayesian Linear regression step-by-step mathematically and was later validated by using a MATLAB built-in function. As the results show, the plot was an exact match.

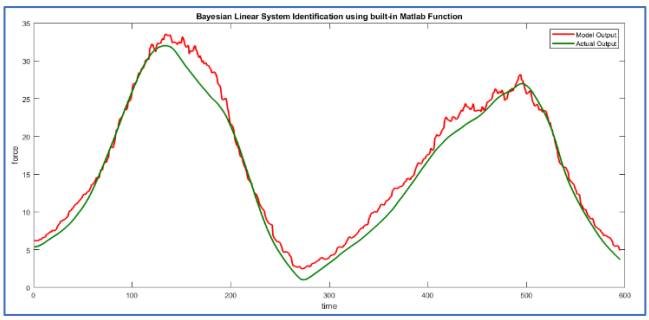


Figure 30 Bayesian System ID of index finger (Actual vs Model) output: 10 input 1 output case

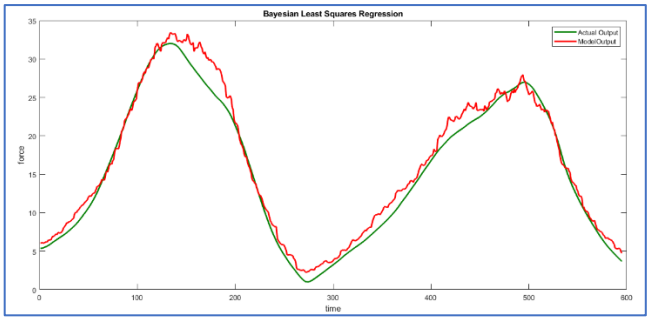


Figure 29 Bayesian Least square regression (Actual vs Model) output: 10 input 1 output case

Table 4 The posterior distribution of the 10 parameters, the intercept, and the noise variance in the 10-inputs,1-output case for the index finger

Parameters	Mean	Std Dev	CI95	Positive	Distribution
Intercept	0.4754	0.1616	(0.159,0.792)	0.998	$t(0.48,(0.16)^2,1.4x10^3)$
sEMG1	$1.70\ x10^4$	3023.03	$(1.10\ x10^4,2.29\ x10^4)$	1	$t(1.70\ x10^4,(3020.81)^2,1.4x10^3)$
sEMG2	$-2.82\ x10^4$	3143.94	$(-3.44\ x10^4,-2.21\ x10^4)$	0	$t(-2.82\ x10^4,(3141.62)^2,1.4x10^3)$
sEMG3	-7961.6989	1691.84	$(1.12\ x10^4,-4645.22)$	0	$t(-7961.6989,(1690.60)^2,1.4x10^3)$
sEMG4	$1.30\ x10^4$	906.36	$(1.12\ x10^4,1.47\ x10^4)$	1	$t1.30\ x10^4,(905.70)^2,1.4x10^3)$
sEMG5	-4788.85	1429.34	(-7590.756,-1986.952)	0	$t(-4788.85,(1428.29)^2,1.4x10^3)$
sEMG6	$-4.86x10^5$	$1.51\ x10^4$	$(-7.83\ x10^5,-1.9\ x10^4)$	0.001	$t-4.86x10^5,(\ 1.50\ x10^4)^2,1.4x10^3)$
sEMG7	$-1.23x10^5$	$1.16\ x10^4$	$(-1.46x10^5,-1.007\ x10^5)$	0	$t(-1.23x105,(1.16x10^4)^2,1.4x10^3)$
sEMG8	$-1.47\ x10^4$	2437.49	$(-1.95\ x10^4,-9996.226)$	0	$t(-1.47\ x104,(2435.7)^2,1.4x10^3)$
sEMG11	$4.46\ x10^5$	$5.72\ x10^4$	$(-6.75x10^4,1.56\ x10^5)$	0.782	$t(4.46\ x10^5,(5.71x10^4)^2,1.4x10^3)$
sEMG12	$8.42x10^4$	4390.21	$(7.56\ x10^4,9.28\ x10^4)$	1	$t(8.42x10^4,(\ 4386.97)^2,1.4x10^3)$
Sigma2	0.3146	0.0121	(1.477,0.339)	1	IG(678,0.0047)

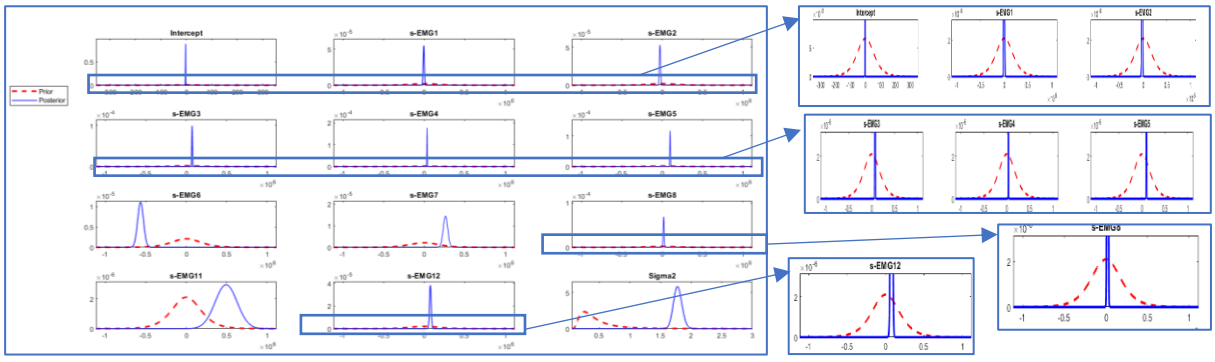


Figure 31 Bayesian Prior and Posterior PDF of index finger for 10 input 1 output case

The figure shows the 95% CI plot for the 10-inputs, 1-output case for the index finger. The bottom plot shows the zoomed in range for the index finger. The CI as seen in the plot is quite narrow indicating that the output prediction certainty was more for the index finger movement.

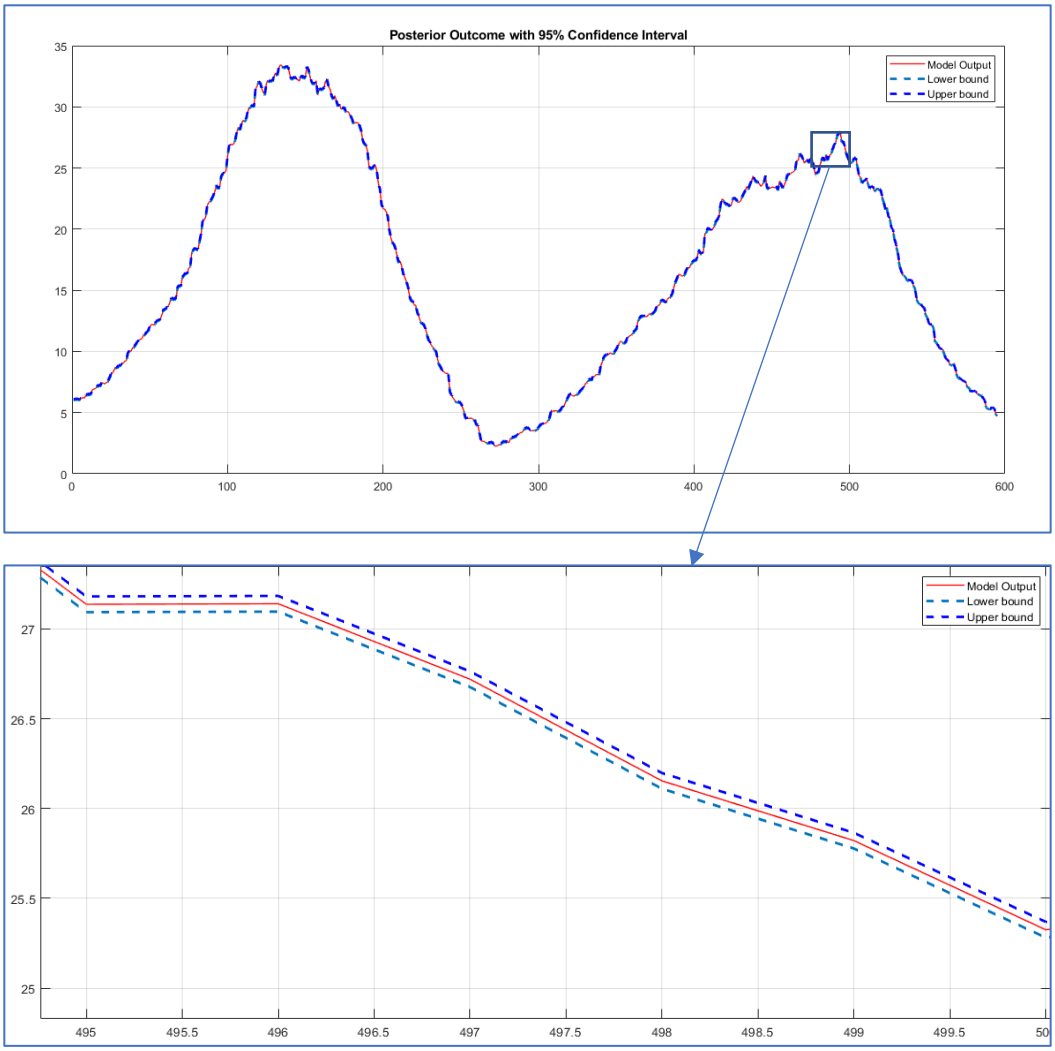


Figure 32 Bayesian Posterior predictive distribution with 95% confidence interval (10 input 1 output case)

4.1.1.2.2.2 Index and Middle Finger:

When 2 finger movements were modelled from 10 s-EMG signals, the plot showed that it was able to track the finger movements quite accurately for both fingers, but the force magnitude estimated by the model was more distorted for the middle finger.

The top plot was got by implementing Multivariate Bayesian Linear regression step-by-step mathematically and was later validated by using a MATLAB built-in function(shown in the bottom plot). As the results show, the plot was an exact match.

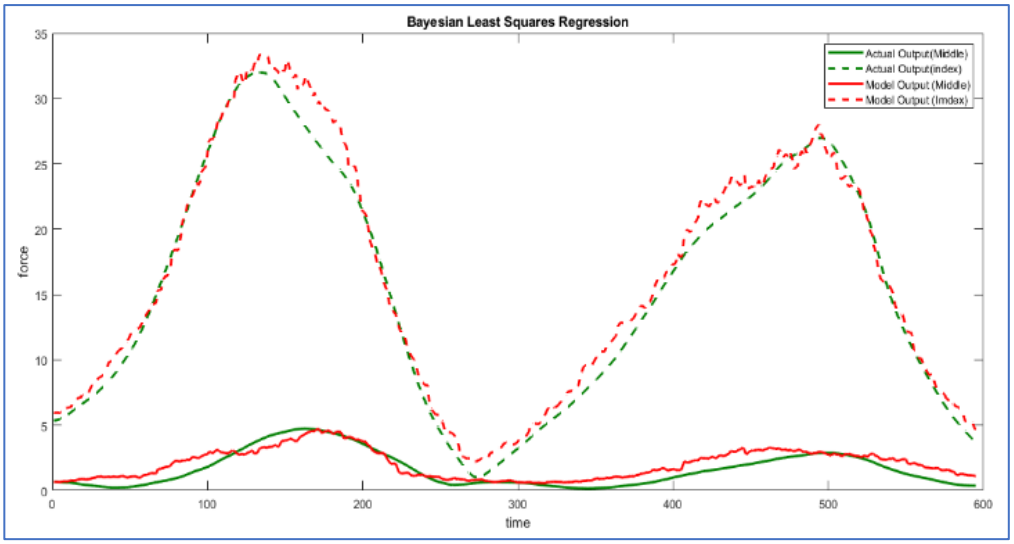


Figure 33 Bayesian Least square regression (Actual vs Model) output: 10 input 2 output case

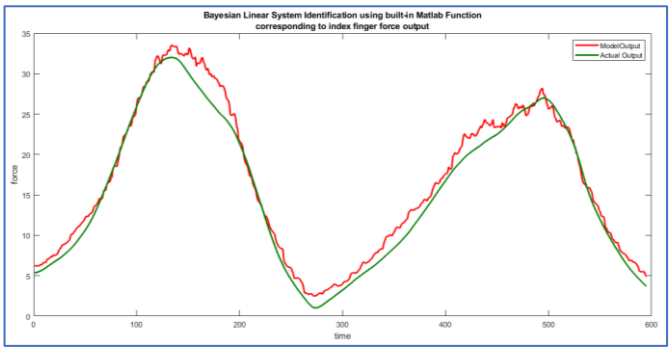


Figure 34 Bayesian System ID of index finger (Actual vs Model) output: 10 input 2 output case

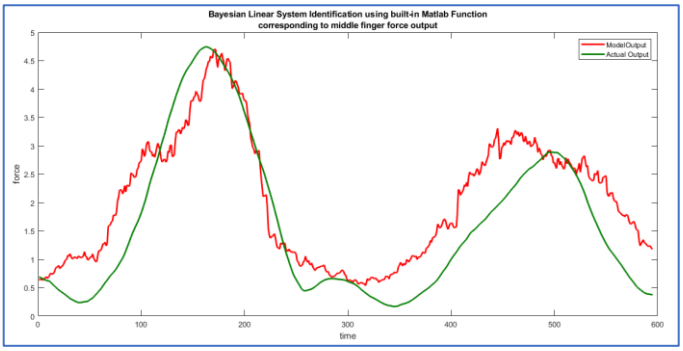


Figure 35 Bayesian System ID of middle finger (Actual vs Model) output: 10 input 2 output case

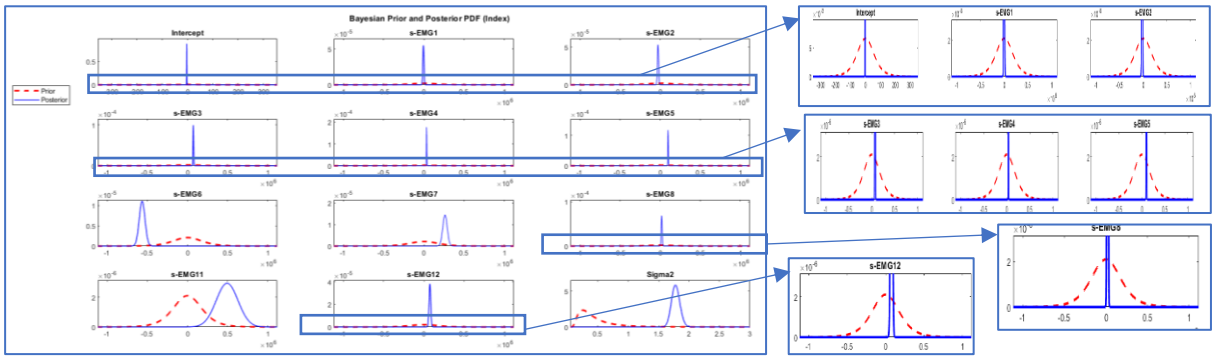


Figure 36 Bayesian Prior and Posterior PDF of index finger for 10 input 2 output case

The posterior distribution of the 10 parameters, the intercept, and the noise variance in the 10-inputs, 2-outputs case for the index finger was found to be the same as in the 10-inputs, 1-output case.

Table 5 The posterior distribution of the 10 parameters, the intercept, and the noise variance in the 10-inputs, 2-outputs case for the middle finger

Parameters	Mean	Std Dev	CI95	Positive	Distribution
Intercept	0.4754	0.1616	(0.159,0.792)	0.998	$t(0.48,(0.16)^2,1.4x10^3)$
sEMG1	$1.70\ x10^4$	3023.03	$(1.10\ x10^4,2.29\ x10^4)$	1	$t(1.70\ x10^4,(3020.81)^2,1.4x10^3)$
sEMG2	$-2.82\ x10^4$	3143.94	$(-3.44\ x10^4,-2.21\ x10^4)$	0	$t(-2.82\ x10^4,(3141.62)^2,1.4x10^3)$
sEMG3	-7961.6989	1691.84	$(1.12\ x10^4,-4645.22)$	0	$t(-7961.6989,(1690.60)^2,1.4x10^3)$
sEMG4	$1.30\ x10^4$	906.36	$(1.12\ x10^4,1.47\ x10^4)$	1	$t1.30\ x10^4,(905.70)^2,1.4x10^3)$
sEMG5	-4788.85	1429.34	(-7590.756,-1986.952)	0	$t(-4788.85,(1428.29)^2,1.4x10^3)$
sEMG6	$-4.86x10^5$	$1.51\ x10^4$	$(-7.83\ x10^5,-1.9\ x10^4)$	0.001	$t-4.86x10^5,(\ 1.50\ x10^4)^2,1.4x10^3)$
sEMG7	$-1.23x10^5$	$1.16\ x10^4$	$(-1.46x10^5,-1.007\ x10^5)$	0	$t(-1.23x105,(1.16x10^4)^2,1.4x10^3)$
sEMG8	$-1.47\ x10^4$	2437.49	$(-1.95\ x10^4,-9996.226)$	0	$t(-1.47\ x104,(2435.7)^2,1.4x10^3)$
sEMG11	$4.46\ x10^5$	$5.72\ x10^4$	$(-6.75x10^4,1.56\ x10^5)$	0.782	$t(4.46\ x10^5,(5.71x10^4)^2,1.4x10^3)$
sEMG12	$8.42x10^4$	4390.21	$(7.56\ x10^4,9.28\ x10^4)$	1	$t(8.42x10^4,(\ 4386.97)^2,1.4x10^3)$
Sigma2	0.3146	0.0121	(1.477,0.339)	1	IG(678,0.0047)

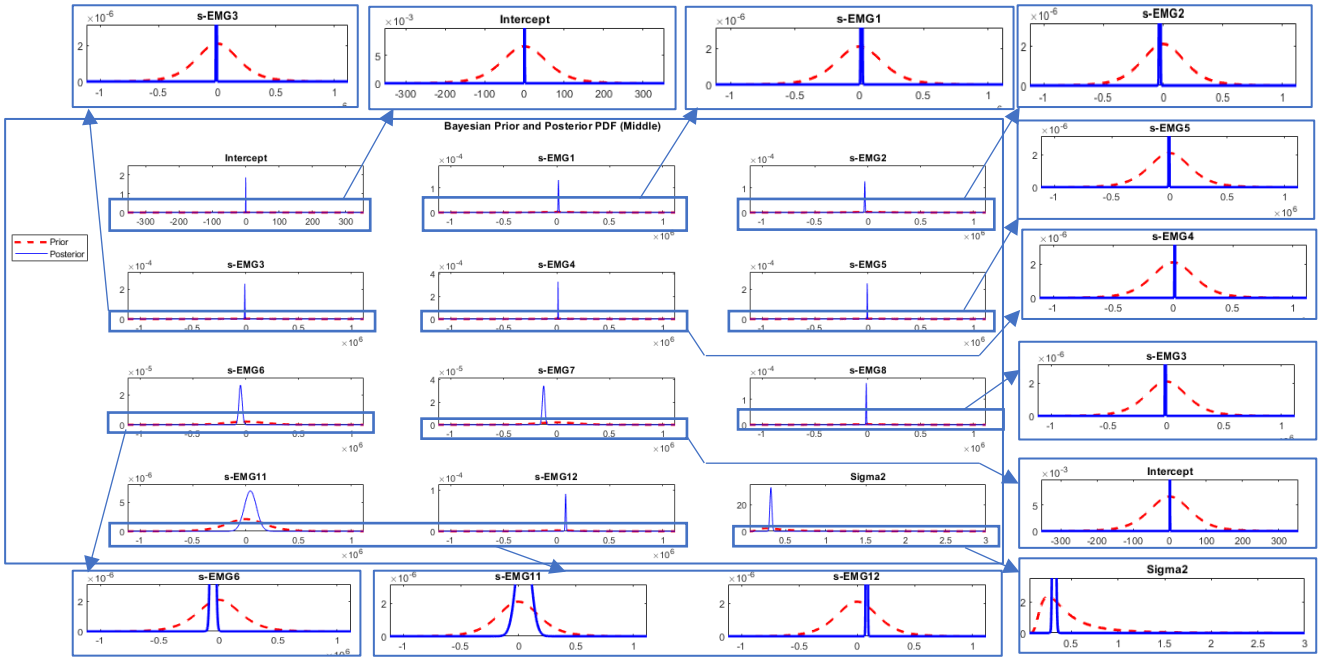


Figure 37 Bayesian Prior and Posterior PDF of middle finger for 10 input 2 output case

The figure shows the 95% CI plot for 10-inputs, 2 outputs case when both the input finger and the middle finger are considered. The bottom plot shows the zoomed in range for both the index finger and middle finger. The CI in the plot shows a narrow range for both the index and the middle finger cases suggesting that the prediction was quite good in both.

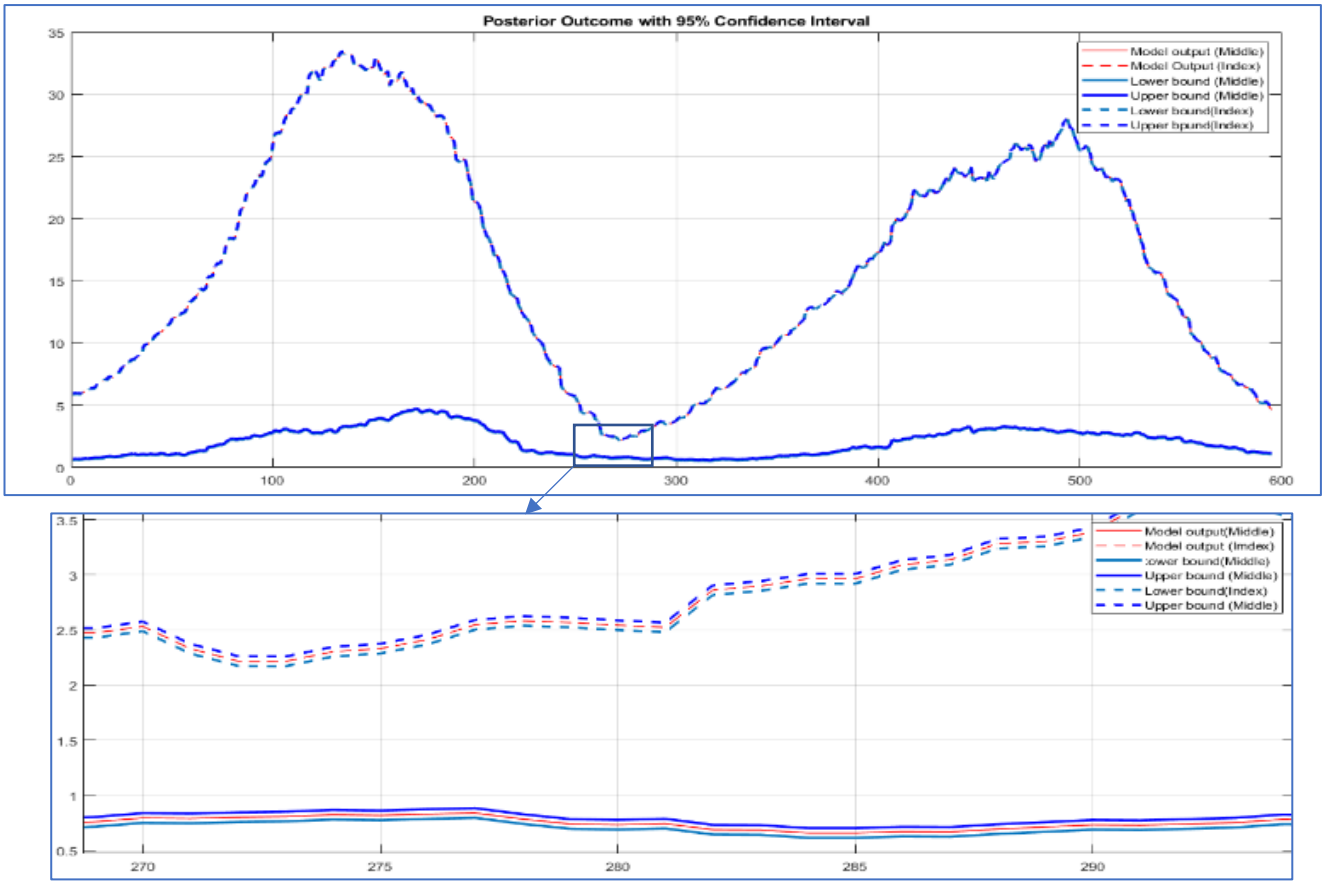


Figure 38 Bayesian Posterior predictive distribution with 95% confidence interval (10 input 2 output case)

4.1.1.2.3 No. of s-EMG signal inputs considered=12

4.1.1.2.3.1 Index Finger:

When 1 finger movement was modelled from all the12 s-EMG signals, the plot showed that it was able to track the finger movements accurately, and the force magnitudes estimated by the model were also close to the actual values despite a few distortions.

The right-hand side plot was got by implementing Multivariate Bayesian Linear regression step-by-step mathematically and was later validated by using a MATLAB built-in function. As the results show, the plot was an exact match.

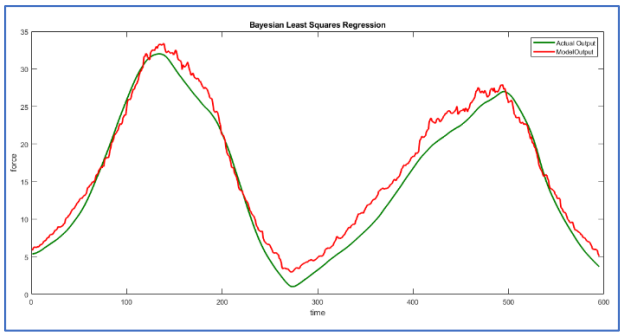


Figure 40 Bayesian Least square regression (Actual vs Model) output: 12 input 1 output case

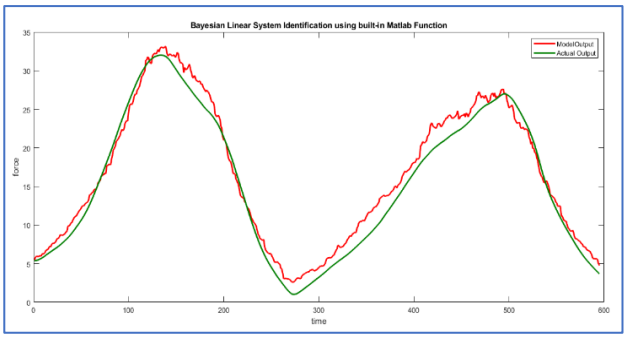


Figure 39 Bayesian System ID of index finger (Actual vs Model)output: 12 input 1 output case

:

Table 6 The posterior distribution of the 12 parameters, the intercept, and the noise variance in the 12-inputs, 1-output case for the index finger

Parameters	Mean	Std Dev	CI95	Positive	Distribution
Intercept	-1.025	0.3957	(-0.343,0.331)	0.005	$t(-1.03,(0.40)^2,1.4x10^3)$
sEMG1	1.14 $x10^4$	7076.07	(916.831,2.86 $x10^4$)	0.982	$t(1.14\ x10^4,(7070.85)^2,1.4x10^3)$
sEMG2	-6.12 $x10^4$	7621.27	(-7.61 $x10^4$,-4.62 $x10^4$)	0	$t(-6.12\ x10^4,(7615.65)^2,1.4x10^3)$
sEMG3	8.27 $x10^4$	3884.00	(7.51 $x10^4$,-9.03 $x10^4$)	1	$t(8.27\ x10^4,(3881.14)^2,1.4x10^3)$
sEMG4	3.31 $x10^4$	2047.98	(2.91 $x10^4$,3.71 $x10^4$)	1	$t(3.31\ x10^4,(2046.47)^2,1.4x10^3)$
sEMG5	6.41 $x10^4$	4540.14	(5.52 $x10^4$,7.30 $x10^4$)	1	$t(6.41\ x10^4,(4536.79)^2,1.4x10^3)$
sEMG6	-5.19 $x10^5$	3.48 $x10^4$	(-5.88 $x10^5$,-4.51 $x10^4$)	0	$t(-5.19\ x10^5,(\ 3.47\ x10^4)^2,1.4x10^3)$
sEMG7	2.79 $x10^5$	2.64 $x10^4$	(2.27$x10^5$,3.31 $x10^4$)	1	$t(2.79\ x10^5,(2.64x10^4)^2,1.4x10^3)$
sEMG8	4.20 $x10^4$	6105.37	(3.00 $x10^4$,5.40 $x10^4$)	1	$t(4.20\ x10^4,(6100.87)^2,1.4x10^3)$
sEMG9	4.25 $x10^4$	4642.50	(3.34 $x10^4$,5.16 $x10^4$)	1	$t(4.25\ x10^4,(4369.08)^2,1.4x10^3)$
sEMG10	-1.03 $x10^5$	1.11 $x10^4$	(-1.25 $x10^5$,-8.16 $x10^4$)	0	$t(-1.03\ x10^5,(1.11\ x10^4)^2,1.4x10^3)$
sEMG11	2.06 $x10^5$	1.3074	(-4.94$x10^4$,4.63 $x10^5$)	0.943	$t(2.06\ x10^5,(1.30\ x10^5)^2,1.4x10^3)$
sEMG12	4.79$x10^4$	1.01 $x10^4$	(2.81 $x10^4$,6.78 $x10^4$)	1	$t(4.79\ x10^4,(\ 1.10\ x10^4)^2,1.4x10^3)$
Sigma2	1.5992	0.0612	(1.477,1.717)	1	IG(678,0.0093)

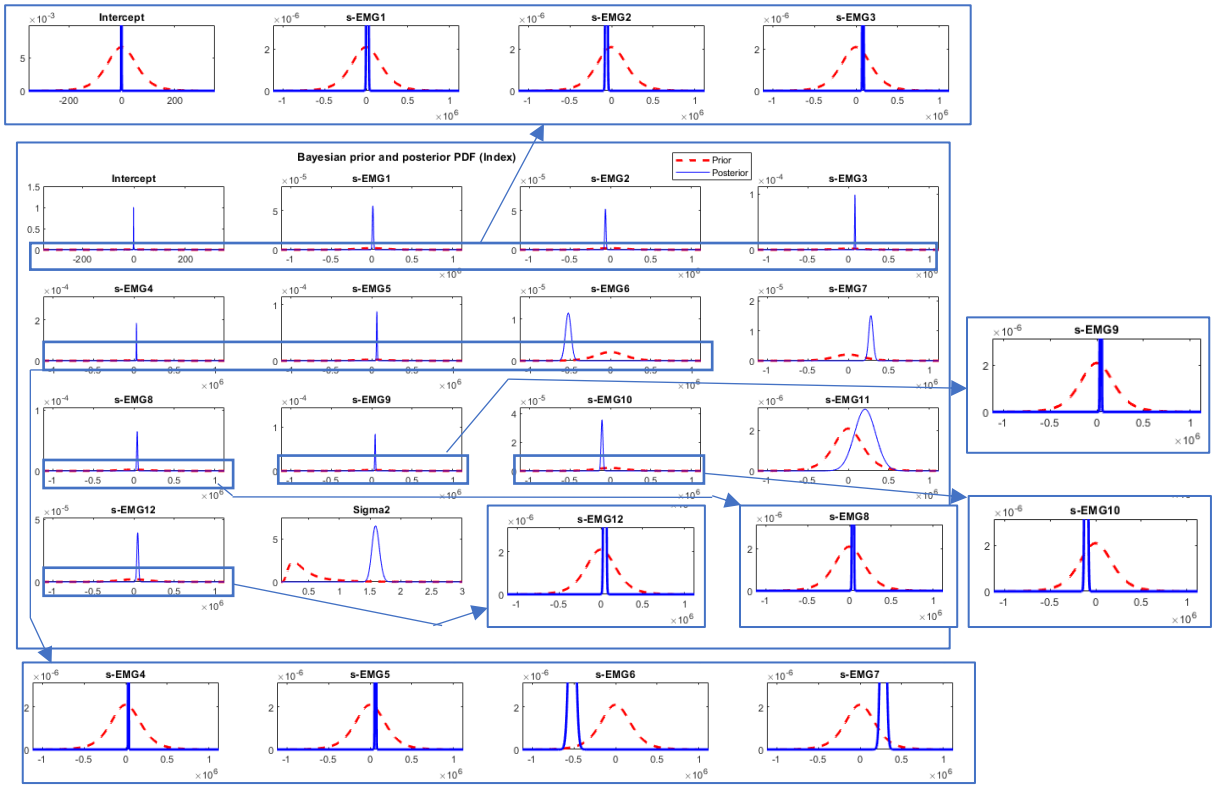


Figure 41 Bayesian Prior and Posterior PDF of index finger for 12 input 1 output case

The figure shows the 95% CI plot for the 12-inputs,1-output case for the index finger. The bottom plot shows the zoomed in range for the index finger. The CI in the plot is quite compact just as was the case in the 10-inputs scenario for the index finger movement implying a good fit.

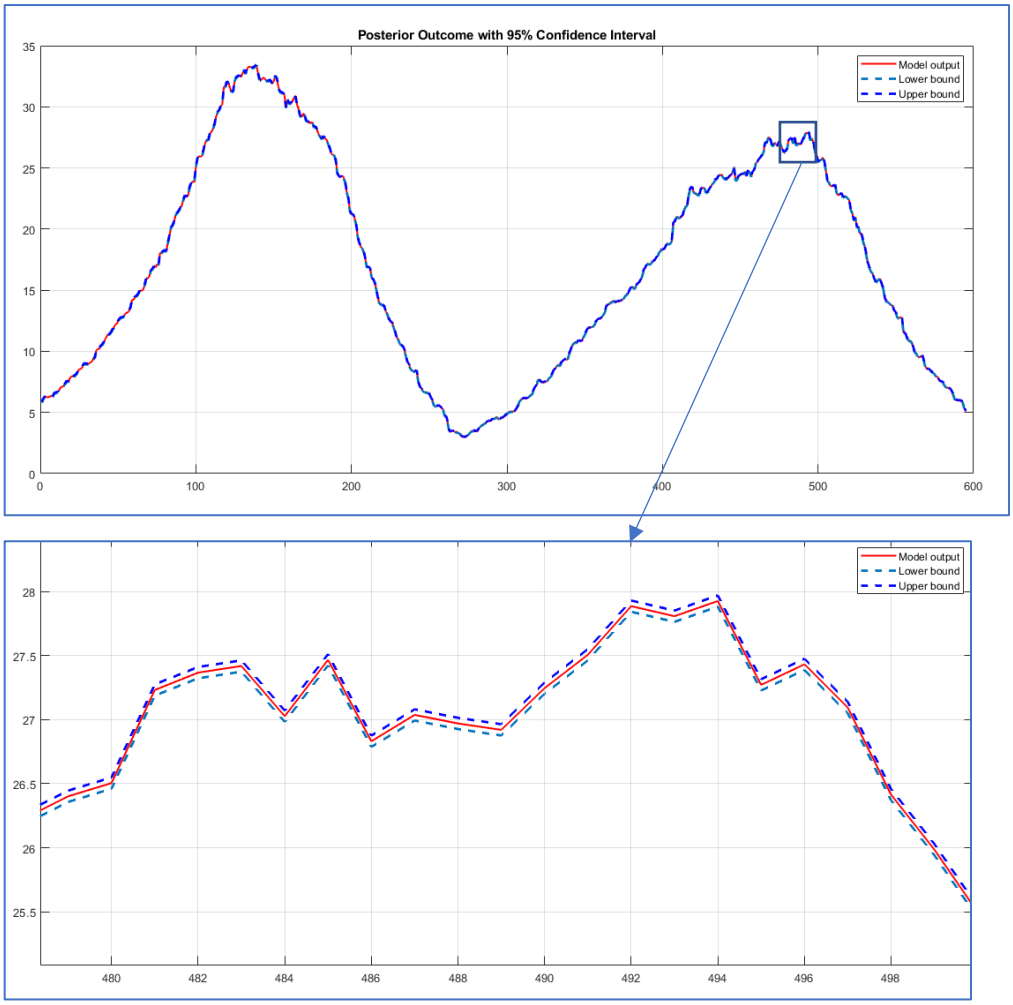


Figure 42 Bayesian Posterior predictive distribution with 95% confidence interval (12 input 1 output case)

4.1.1.2.3.2 Index and Middle Finger:

When 2 finger movements were modelled from all the 12 s-EMG signals, the plot showed that it was able to track the finger movements quite accurately for both fingers, but the force magnitude estimated by the model was more distorted for the middle finger and was found to have performed like how the model in the 10-input 2-output case did.

The top plot was got by implementing Multivariate Bayesian Linear regression step-by-step mathematically and was later validated by using a MATLAB built-in function(shown in the bottom plot). As the results show, the plot was an exact match.

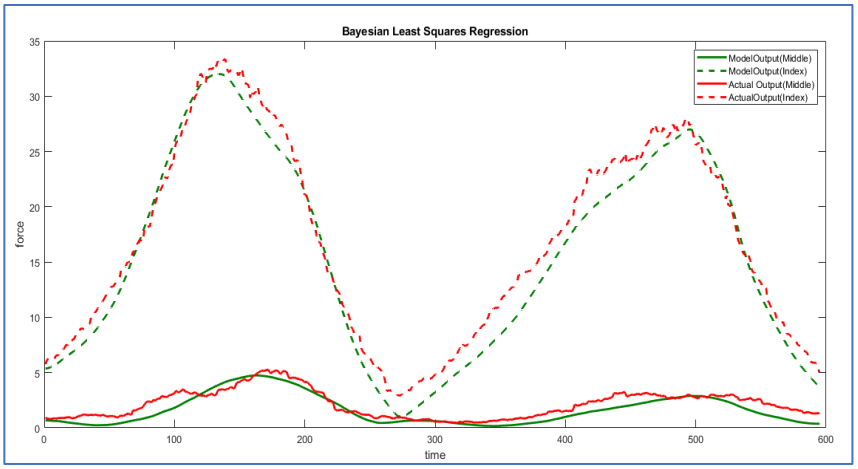


Figure 43 Bayesian Least square regression (Actual vs Model) output: 12 input 2 output case

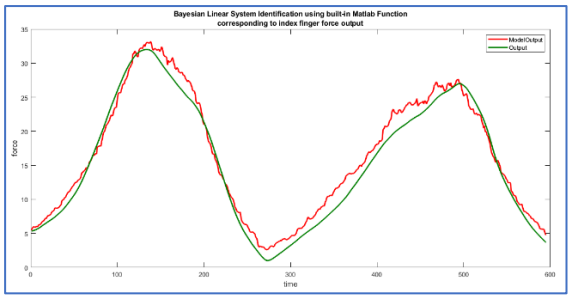


Figure 44 Bayesian System ID of middle finger (Actual vs Model) output: 12 input 2 output case

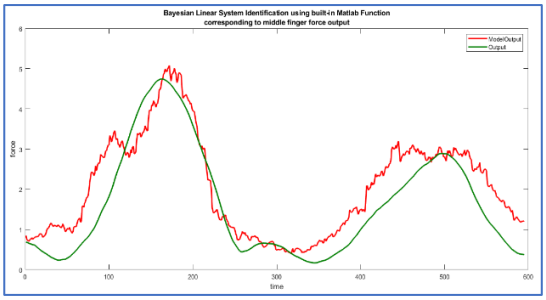


Figure 45 Bayesian System ID of index finger (Actual vs Model) output: 12 input 2 output case

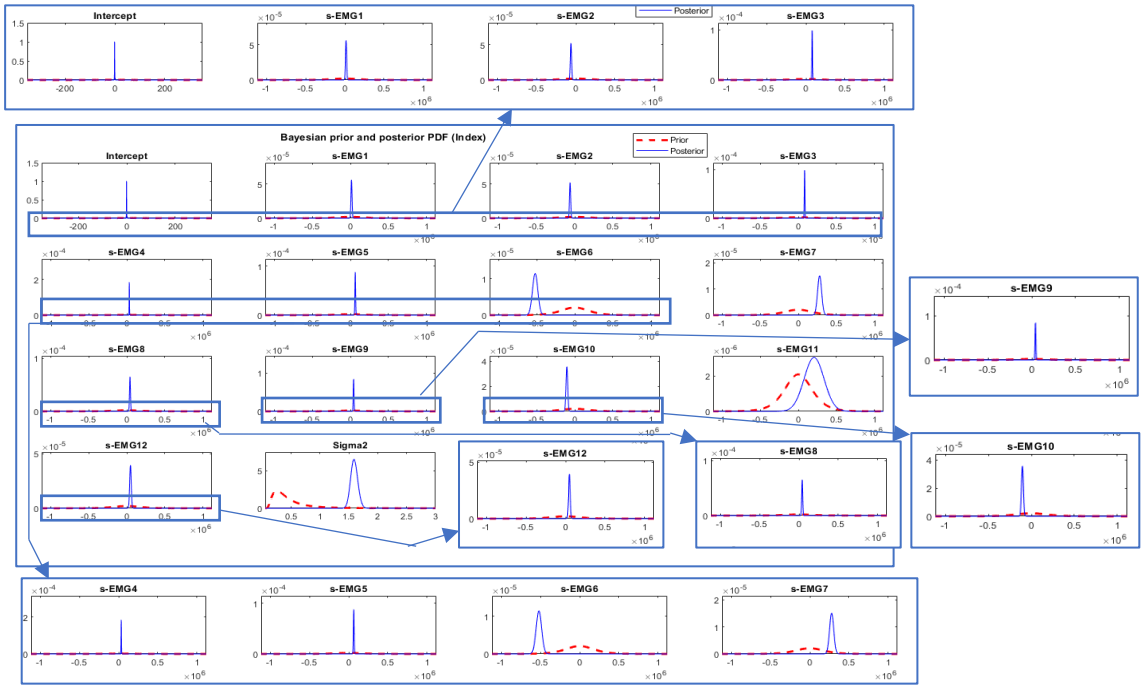


Figure 46 Bayesian Prior and Posterior PDF of index finger for 12 input 2 output case

The posterior distribution of the 12 parameters, the intercept, and the noise variance in the 12-inputs, 2-outputs case for the index finger was found to be the same as in the 12-inputs, 1-output case.

Table 7 The posterior distribution of the 12 parameters, the intercept, and the noise variance in the 12-inputs, 2-outputs case for the middle finger

Parameters	Mean	Std Dev	CI95	Positive	Distribution
Intercept	-0.0059	0.1719	(-0.343,0.331)	0.486	$t(-0.01,(0.17)^2,1.4x10^3)$
sEMG1	$1.13\ x10^4$	3073.76	$(5.35\ x10^3,1.74\ x10^4)$	1	$t(1.13\ x10^4,(3071.50)^2,1.4x10^3)$
sEMG2	$-1.87\ x10^4$	3310.60	$(-2.52\ x10^4,-1.22\ x10^4)$	0	$t(-1.87\ x10^4,(3308.16)^2,1.4x10^3)$
sEMG3	$-1.05\ x10^4$	1687.17	$(-1.38\ x10^4,-7274.48)$	0	$t(-1.05\ x10^4,(1685.93)^2,1.4x10^3)$
sEMG4	$1.35\ x10^4$	889.62	$(1.18\ x10^4,1.53\ x10^4)$	1	$t(1.35\ x10^4,(888.96)^2,1.4x10^3)$
sEMG5	$4.33\ x10^3$	1972.18	(467.093,8199.131)	0.986	$t(4.33\ x10^3,(1970.73)^2,1.4x10^3)$
sEMG6	$-6.32\ x10^4$	$1.51\ x10^4$	$(-9.28\ x10^4,-3.36\ x10^4)$	0	$t(-6.32\ x10^4,(\ 1.51\ x10^4)^2,1.4x10^3)$
sEMG7	$-1.29\ x10^5$	$1.14\ x10^4$	$(-1.51\ x10^5,-1.06\ x10^5)$	0	$t(-1.29\ x10^5,(1.14\ x10^4)^2,1.4x10^3)$
sEMG8	$-2.15\ x10^4$	2652.11	$(-2.67\ x10^4,-1.63\ x10^4)$	0	$t(-2.15\ x10^4,(2650.15)^2,1.4x10^3)$
sEMG9	$-1.04\ x10^4$	2016.65	$(-1.44\ x10^4,-6540.32)$	0	$t(-1.04\ x10^4,(2015.17)^2,1.4x10^3)$
sEMG10	$3.06\ x10^4$	4844.09	$(2.11\ x10^4,4.01\ x10^4)$	1	$t(3.06\ x10^4,(4840.52)^2,1.4x10^3)$
sEMG11	$1.22\ x10^5$	$5.67\ x10^4$	$(1.14\ x10^4,2.34\ x10^5)$	0.985	$t(1.22\ x10^5,(5.67\ x10^4)^2,1.4x10^3)$
sEMG12	$9.18\ x10^4$	4398.11	$(8.32\ x10^4,1.005\ x10^4)$	1	$t(9.18\ x10,(4394.87)^2,1.4x10^3)$
Sigma2	0.3004	0.0116	(0.279,0.324)	1	IG(678,0.0049)

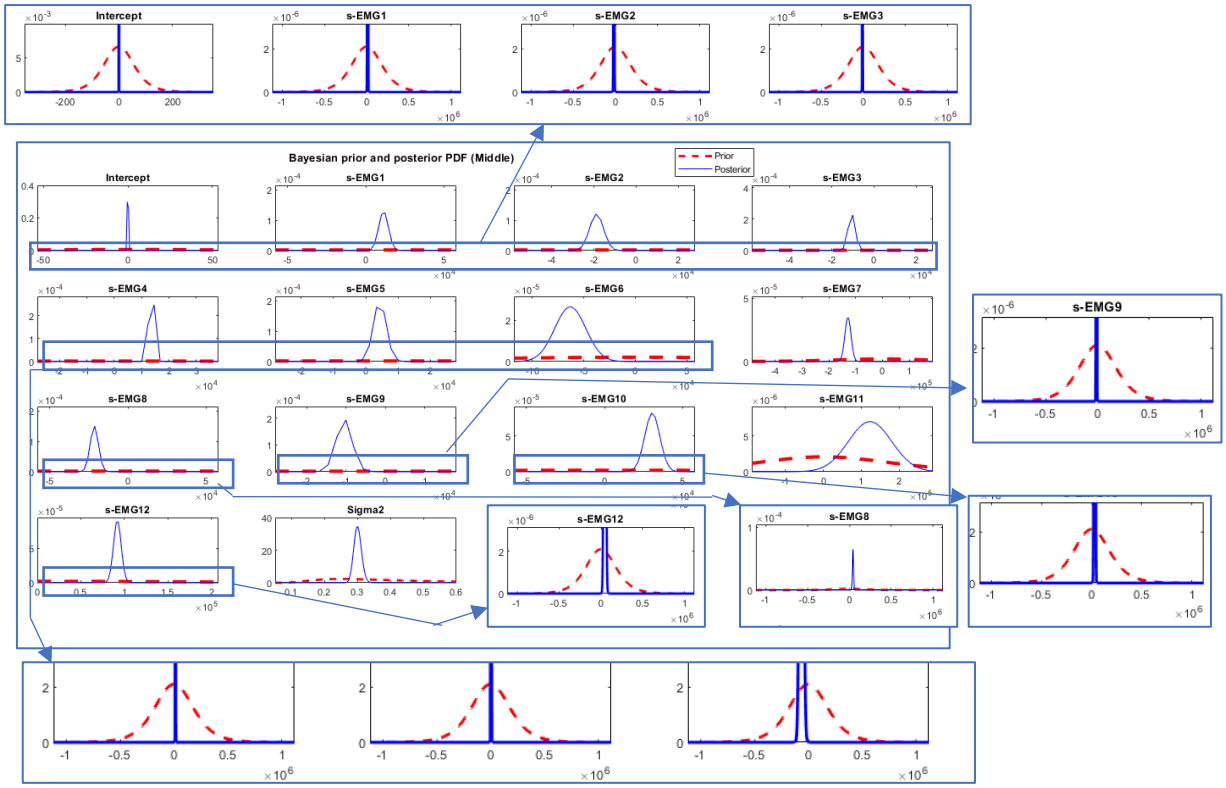


Figure 47 Bayesian Prior and Posterior PDF of middle finger for 12 input 2 output case

The figure shows the 95% CI plot for the 12-inputs, 2-outputs case when both the input finger and the middle finger are considered. The bottom plot shows the zoomed in range for the index finger and for the middle finger. The CI in the plot is very close to the actual force values for both the finger movements.

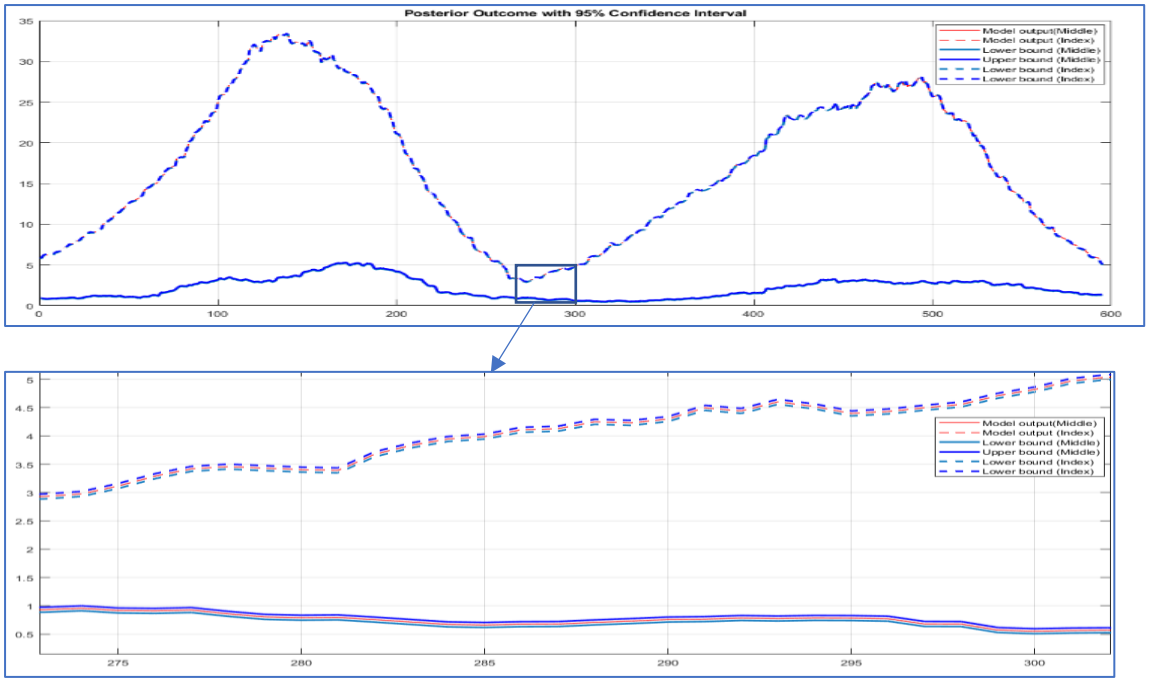


Figure 48 Bayesian Posterior predictive distribution with 95% confidence interval (12 input 2 output case)

4.1.1.3 ARX

Discrete-time ARX model: of the form A(z)y(t) = B(z)u(t) + e(t) with a sampling time of 0.01 seconds was considered for 1-output cases. In the case of 2 output fingers (index and middle) each output is modelled with input as 2 models:- In case of 10 and 12 sEMG inputs A(z)y1(t) = - A2(z)y2(t) + B(z)u(t) + e1(t), and Model for output "y2": A(z)y2(t) = - A1(z)y1(t) + B(z)u(t) + e2(t) (Note:A1(z) and A2(z) are 0 for 2 sEMG input case).

4.1.1.3.1 No. of s-EMG signal inputs considered=2

4.1.1.3.1.1 Index Finger

$A(z) = 1 - 1.009\ z^{-1}$

$B1(z) = 2.328x10^{5} + 4551\ z^{-1} - 2.081x10^{5}\ z^{-2}$

$B2(z) = 2.478x10^{4} - 5022\ z^{-1} + 7026\ z^{-2} + 5875\ z^{-3} + 928.3\ z^{-4}$
$- 3782\ z^{-5} - 2410\ z^{-6} - 5023\ z^{-7} + 6189\ z^{-8} - 3.758e04\ z^{-9}$

Polynomial orders provide were na=1, nb=[3 10], nk=[0 0] and Number of free coefficients in the model were 14. The model performance was as follows:
The model performance was as follows: Fit to estimation data: 98.79% (prediction focus) FPE: 0.01182 and a mean squared error value of 0.01155.

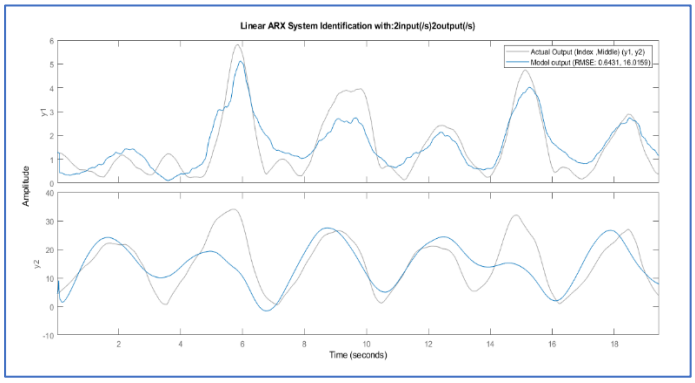


Figure 50 ARX regression (Actual vs Model) output: 2 input 2 output case

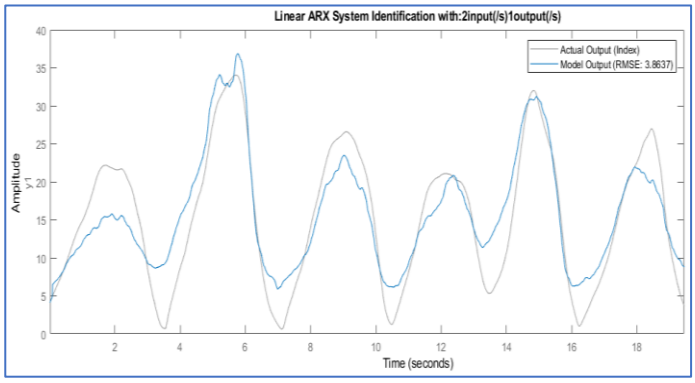


Figure 50 ARX regression (Actual vs Model) output: 2 input 1 output case

4.1.1.3.1.2 Index and Middle Finger

Model for output "y1": middle finger

$A(z) = 1 - 0.9963\ z^{-1}$

$B1(z) = 3.849x10^{4} + 4129\ z^{-1} - 4.699\ x10^{4}\ z^{-2}$

$B2(z) = 6049 - 1708\ z^{-1} + 1464\ z^{-2} + 1114\ z^{-3} + 227.6\ z^{-4}$
$- 677.8\ z^{-5} - 434\ z^{-6} - 647.7\ z^{-7} + 417\ z^{-8} - 4943\ z^{-9}$

Model for output "y2": index finger

$A(z) = 1 - 2.391\ z^{-1} + 1.633\ z^{-2} - 0.1295\ z^{-3} - 0.06294\ z^{-4} - 0.07809\ z^{-5} - 0.001854\ z^{-6} + 0.08386\ z^{-7} - 0.05289\ z^{-8}$

$B1(z) = 865.7$

$B2(z) = 184.9\ z^{-1} - 117.9\ z^{-2} - 1.369\ z^{-3} - 244.2z^{-4} + 237.5\ z^{-5} - 138.9\ z^{-6}$

Polynomial orders provide were na=[1 0;0 8] nb=[3 10;1 6] nk=[0 0;0 1] Number of free coefficients in the model were 29 The model performance was as follows: Fit to estimation data: [98.42;99.93]% (prediction focus) FPE: 2.019e-08and a mean squared error value of 0.0004953.
The plots for the 2-outputs case and the 1-output case show an acceptable tracking of the finger movements but the force magnitude ranges are not met at all places, but this is expected since it was modelled with access to only 2 s-EMG signals as input.

4.1.1.3.2 No. of s-EMG signal inputs considered=10

4.1.1.3.2.1 Index Finger

$A(z) = 1 - 1.982\ z^{-1} + 0.9831\ z^{-2}$
$B1(z) = 482.6 + 3.679\ z^{-1} - 235.1\ z^{-2} - 45.21\ z^{-3} - 122.2\ z^{-4} - 64.22\ z^{-5}$
$B2(z) = 783 - 432.3\ z^{-1} + 221.4\ z^{-2} - 206.4\ z^{-3} - 50.44\ z^{-4} - 200.9\ z^{-5}$
$B4(z) = 239 - 153.4\ z^{-1} - 107.2\ z^{-2}$
$B5(z) = B6(z) = B8(z) = B9(z) = B10(z) = 0$
$B3(z) = 0.1799$ and $B7(z) = 874.4$

Polynomial orders provide were na=2 nb=[6 6 1 3 0 0 1 0 0 0] nk=[0 0 0 0 1 1 0 1 1 1]
Number of free coefficients in the model were 19
The model performance was as follows: Fit to estimation data: 99.91% (prediction focus)
FPE: 5.867e-05and a mean squared error value of 5.724e-05

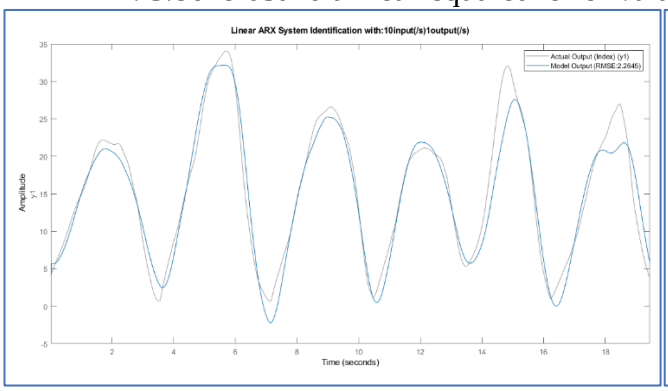


Figure 51 ARX regression (Actual vs Model) output: 10 input 1 output case

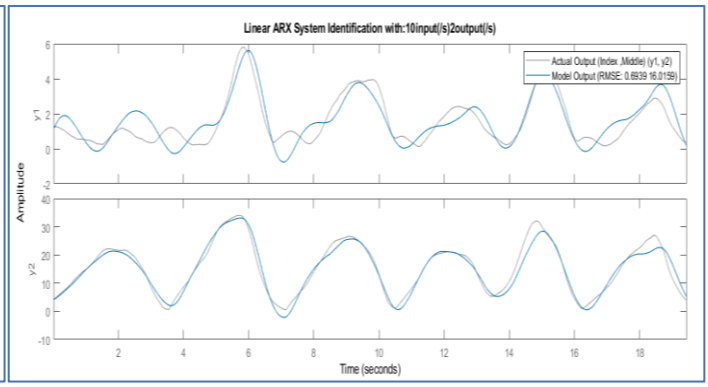


Figure 52 ARX regression (Actual vs Model) output: 10 input 2 output case

4.1.1.3.2.2 Index and Middle Finger

Model for output "y1": middle finger
$A(z) = 1 - 1.984\ z^{-1} + 0.9854\ z^{-2}$
$A\_2(z) = 0.001668\ z^{-1} - 0.001552\ z^{-2}$
$B1(z) = 22.74 - 5.581\ z^{-1} - 28.92\ z^{-2} + 85.32\ z^{-3} + 27.63\ z^{-4} - 83.86\ z^{-5}$
$B2(z) = 101.7 - 152.4\ z^{-1} + 65.21\ z^{-2} - 171.3\ z^{-3} + 97.64\ z^{-4} + 38.55\ z^{-5}$
B3(z) = 11.68 B4(z) = 90.81 - 41.18 z-1 - 35.43 z-2
$B5(z) = B6(z) = B8(z) = B9(z) = B10(z) = 0$
$B7(z) = 24.26$

Model for output "y2": index finger
$A(z) = 1 - 1.982\ z^{-1} + 0.9828\ z^{-2}$
$A\_1(z) = 0.01306\ z^{-1} - 0.01219\ z^{-2}$
$B1(z) = 457.7 + 8.047\ z^{-1} - 238.2\ z^{-2} - 39.65\ z^{-3} - 122.2\ z^{-4} - 33.4\ z^{-5}$
$B2(z) = 805.8 - 434.9\ z^{-1} + 223.1\ z^{-2} - 214.3\ z^{-3} - 52.65\ z^{-4} - 237.6\ z^{-5}$
$B3(z) = 9.975$
$B4(z) = 246.6 - 153.3\ z^{-1} - 94.44\ z^{-2}$
$B5(z) = B6(z) = B8(z) = B9(z) = B10(z) = 0$
$B7(z) = 806.9$

Polynomial orders provide were na=[2 2;2 2] nb=[6 6 1 3 0 0 1 0 0 0;6 6 1 3 0 0 1 0 0 0] nk=[0 0 0 0 1 1 0 1 1 1;0 0 0 0 1 1 0 1 1 1] Number of free coefficients in the model were 42
The model performance was as follows: Fit to estimation data: [99.82;99.92]% (prediction focus) FPE: 3.64e-10and a mean squared error value of 6.318e-05

The plots for the 2-outputs case and the 1-output case with 10 s-EMG signals as input show an improved tracking of the finger movements compared to the 2-inputs case but the force magnitude ranges are not met at all places and the finger movements are not tracked properly in the 2- outputs case for the middle finger.

4.1.1.3.3 No. of s-EMG signal inputs considered=12

4.1.1.3.3.1 Index Finger

$A(z) = 1 - 1.982\ z^{-1} + 0.9831\ z^{-2}$
$B1(z) = 482.6 + 3.679\ z^{-1} - 235.1\ z^{-2} - 45.21\ z^{-3} - 122.2\ z^{-4} - 64.22\ z^{-5}$
$B2(z) = 783 - 432.3\ z^{-1} + 221.4\ z^{-2} - 206.4\ z^{-3} - 50.44\ z^{-4} - 200.9\ z^{-5}$
$B4(z) = 239 - 153.4\ z^{-1} - 107.2\ z^{-2}$
$B5(z) = B6(z) = B8(z) = B9(z) = B10(z) = B11(z) = B12(z) = 0$
$B3(z) = 0.1799$ and $B7(z) = 874.4$

Polynomial orders provide were na=2 nb=[6 6 1 3 0 0 1 0 0 0 0 0] nk=[0 0 0 0 1 1 0 1 1 1 1 1] Number of free coefficients in the model were 19 The model performance was as follows: Fit to estimation data: 99.91% (prediction focus) FPE: 5.867e-05and a mean squared error value of 5.724e-05

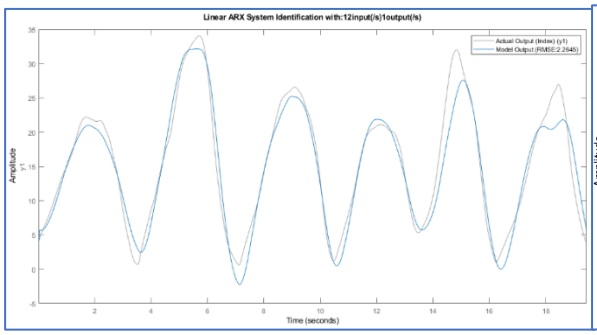


Figure 49 ARX regression (Actual vs Model) output:12 input 1 output case

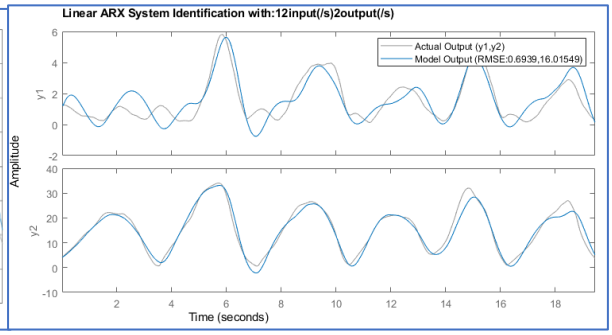


Figure 53 ARX regression (Actual vs Model) output: 12 input 2 output case

4.1.1.3.3.2 Index and Middle Finger

Model for output "y1": middle finger
$A(z) = 1 - 1.984\ z^{-1} + 0.9854\ z^{-2}$
$A\_2(z) = 0.001668\ z^{-1} - 0.001552\ z^{-2}$
$B1(z) = 22.74 - 5.581\ z^{-1} - 28.92\ z^{-2} + 85.32\ z^{-3} + 27.63\ z^{-4} - 83.86\ z^{-5}$
$B2(z) = 101.7 - 152.4\ z^{-1} + 65.21\ z^{-2} - 171.3\ z^{-3} + 97.64\ z^{-4} + 38.55\ z^{-5}$
$B4(z) = 90.81 - 41.18\ z^{-1} - 35.43\ z^{-2}$
$B5(z) = B6(z) = B8(z) = B9(z) = B10(z) = B11(z) = B12(z) = 0$
$B3(z) = 11.68$ and $B7(z) = 24.26$

Model for output "y2": index finger
$A(z) = 1 - 1.982\ z^{-1} + 0.9828\ z^{-2}$
$A\_1(z) = 0.01306\ z^{-1} - 0.01219\ z^{-2}$
$B1(z) = 457.7 + 8.047\ z^{-1} - 238.2\ z^{-2} - 39.65\ z^{-3} - 122.2\ z^{-4} - 33.4\ z^{-5}$
$B2(z) = 805.8 - 434.9\ z^{-1} + 223.1\ z^{-2} - 214.3\ z^{-3} - 52.65\ z^{-4} - 237.6\ z^{-5}$
$B3(z) = 9.975$ and $B7(z) = 806.9$
$B4(z) = 246.6 - 153.3\ z^{-1} - 94.44\ z^{-2}$
$B5(z) = B6(z) = B8(z) = B9(z) = B10(z) = B11(z) = B12(z) = 0$
Polynomial orders provide were na=[2 2;2 2] nb=[6 6 1 3 0 0 1 0 0 0 0 0;6 6 1 3 0 0 1 0 0 0 0 0] nk=[0 0 0 0 1 1 0 1 1 1 1 1;0 0 0 0 1 1 0 1 1 1 1 1] Number of free coefficients in the model

were 42 The model performance was as follows: Fit to estimation data: [99.82;99.92]% (prediction focus) FPE: 3.64e-10and a mean squared error value of 6.318e-05

The plots for the 2-outputs case and the 1-output case with all the 12 s-EMG signals as input show a similar performance to that of the 10 s-EMG cases.
The plots below show the step response of the predicted models.

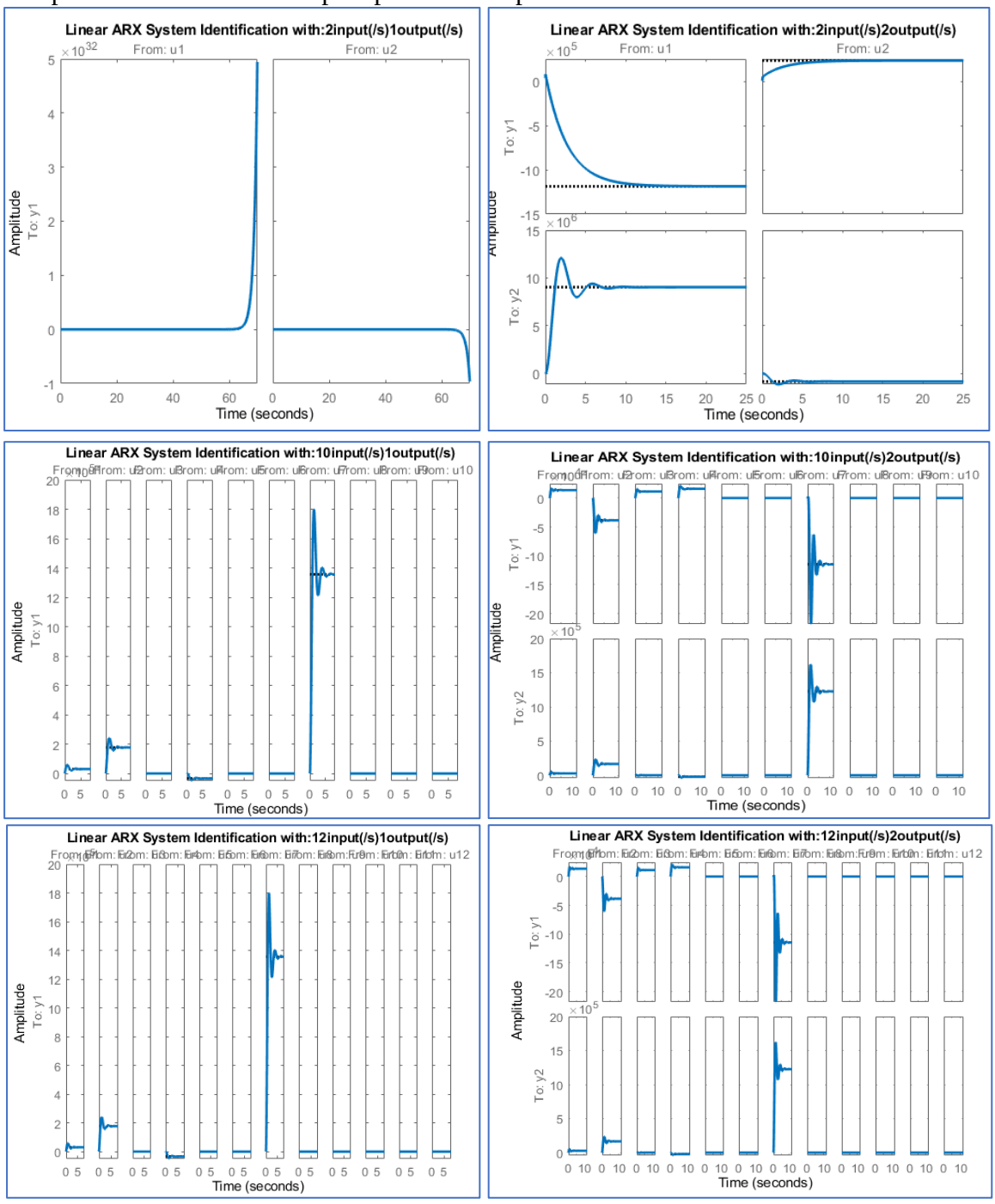


Figure 54 step response of ARX models corresponding to each input and output combination

From the 10-inputs and 12-inputs case step plots with single output, it is evident that the 1st 2nd and 7th feature inputs were good indicators for the index finger movement. Among the sensors placed around the forearm, the 7th sensor seems to be the most active muscle

responsible for the index finger movement. Similarly, in the 2-outputs case with 10 s-EMG signals and 12 s-EMG signals, the 7th sensor signal only seems to be the biggest contributor for the middle finger movement as well.

In the 2-outputs case, the 7th sensor step response for the middle finger showed lot of oscillations before settling into a steady state indicating high damping ratio. Also, the rise time was low, and the overshoot was high. All these properties indicate to the system having a high sensitivity to the 7th sensor input when the middle finger is moved. The system is more prone to picking up noise as well. In the index finger case, the step response has a slightly lower damping ratio and a slower rise time with a lower overshoot. The settling time is also lower compared to that for the middle finger case indicating that the system is more stable.

Where the attempt was to model with only 2 s-EMG signals, the modelled system was found to be unstable in the 1-output case, whereas in the 2-outputs case 11th sensor signal collected from Biceps Brachii showed more muscle activity for the index finger movement. Similarly, the 12th sensor signal collected from Triceps Brachii showed more muscle activity for the middle finger movement.

The step response from the 12th sensor for the index finger in the 2-input case indicates an underdamped system implying that the system is highly selective.

4.1.1.4 Comparison among the linear regression techniques used:

The study so far has indicated towards Bayesian Linear regression technique to be the best likely because it does not base its parameter estimation on available data alone like the other techniques but also assigns a prior distribution weighted by a likelihood function and treats the parameters to be estimated as random variables. This additional information can aid in designing better regression models. This is what primarily sets apart the Bayesian approach from the other Frequentist approaches such as the Ordinary Least Squares and the ARX.

Another advantage of using the Bayesian technique for regression is that it not only gives the predicted output but also the uncertainty surrounding the prediction.

The ARX on the other hand considers the impact of the past inputs and outputs on the current output but fails to perform nearly as well as the Ordinary Least Squares and the Bayesian approach because of its model design where the poles of noise model and the dynamic model are shared. Recall, how the denominator $A(z^{-1})$ is same for both the terms.

$$y(k) = z^{-d}\frac{B(z^{-1})}{A(z^{-1})}u(k) + \frac{\xi(k)}{A(z^{-1})}$$

This inherent noise in ARX biases the parameter estimation.

Table 8 Goodness of fit (R2 values) for different linear regression tasks

R^2 Values (goodness of fit)						
I/O Case / **Model Type**	**12 Input 1 Output**	**12 Input 2 Output**	**10 Input 1 Output**	**10 Input 2 Output**	**2 Input 1 Output**	**2 Input 2 Output**
LS	80.615025	93.004032, 93.902129	98.780781	80.370176, 98.780781	80.615025	76.060226, 80.615025
BLS	81.204603	100.076891 95.067131	101.989067	79.421044, 101.376975	81.204603	79.437045, 81.631030
ARX	93.7664	97.1455 244.9232	93.7664	97.1455 244.9232	69.6899	58.1267 244.9232

Table 9 RMSE Values for different linear regression tasks

RMSE Values						
I/O Case / **Model Type**	**12 Input 1 Output**	**12 Input 2 Output**	**10 Input 1 Output**	**10 Input 2 Output**	**2 Input 1 Output**	**2 Input 2 Output**
LS	3.308447	0.644694,1.619250	1.556891	0.622037, 1.556891	3.308447	0.493967,3.308447
BLS	3.071978	0.685781, 1.711379	1.390852	0.658785, 1.265203	3.070897	0.469916,3.070897
ARX	2.2645	0.6939 16.0159	2.2645	0.6939 16.0159	3.8637	0.6431 16.0159

Bayesian least square regression had the highest goodness of fit and least RMSE value within linear regression as explained.

4.1.2 Non- Linear Regression

4.1.2.1 NARX

Discrete-time Non-linear ARX model with a sampling time of 0.01 seconds was considered for all 6 input-output cases. Model output is linear in their regressors. Standard wavelet regressors of specified orders were used in each case.

4.1.2.1.1 No. of s-EMG signal inputs considered=2

4.1.2.1.1.1 Index Finger

The model performance was as follows: Fit to estimation data: [98.33;98.73]% (prediction focus) FPE: 1.329e-07 and a mean squared error value of 0.01326

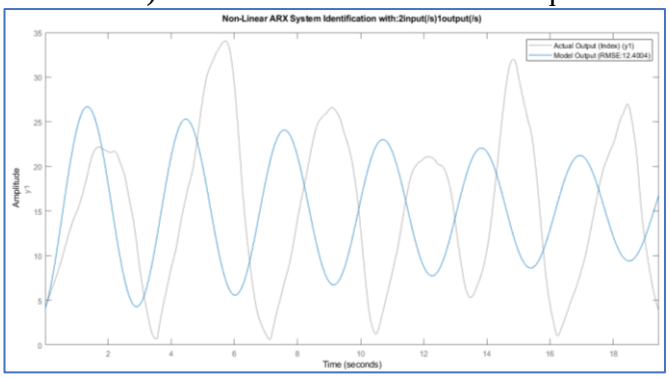


Figure 55 Non-linear ARX regression (Actual vs Model) output: 2 input 1 output case

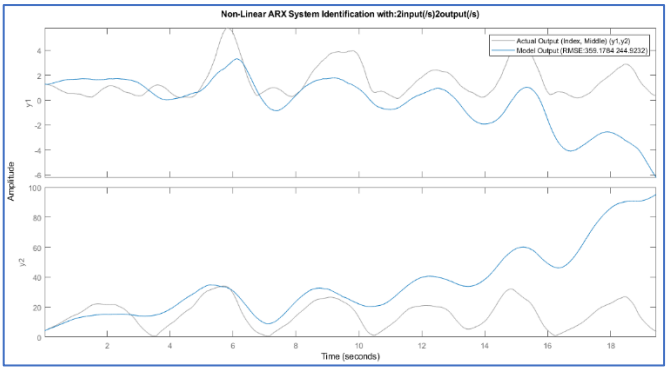


Figure 56 Non-linear ARX regression (Actual vs Model) output: 2 input 1 output case

4.1.2.1.1.2 Index and Middle Finger

Standard regressors corresponding to the orders: na = [1 2; 2 1], nb = [3 10; 3 10], nk = [0 0; 0 0] The model performance was as follows: Fit to estimation data: [98.33;98.73]% (prediction focus) FPE: 1.329e-07and a mean squared error value of 0.01326

The 1-output plot based on 2 s-EMG signals shows a terrible lag in tracking the index finger movement. While in the 2-output case the system seems to be doing a better job of tracking in the initial time steps but worsens progressively as time passes.

4.1.2.1.2 No. of s-EMG signal inputs considered=10

4.1.2.1.2.1 Index Finger

Standard regressors corresponding to the orders: na = [2], nb = [6 1 1 0 0 0 1 0 0 0], nk = [0 0 0 1 1 1 0 1 1 1]. The model performance was as follows: Fit to estimation data: 99.92% (prediction focus) FPE: 5.732e-05and a mean squared error value of 5.647e-05

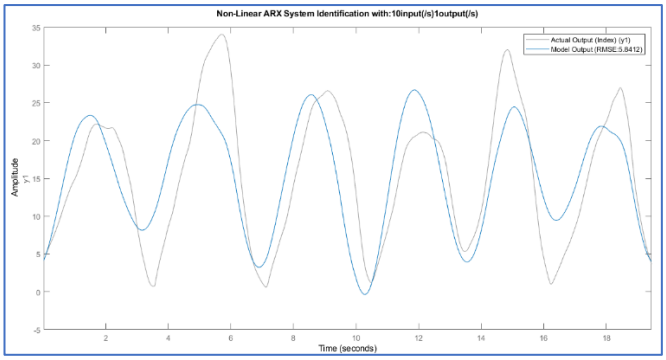


Figure 58 Non-linear ARX regression (Actual vs Model) output: 10 input 1 output case

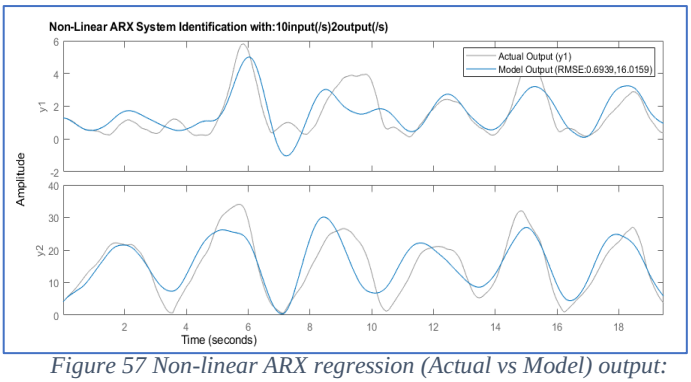


Figure 57 Non-linear ARX regression (Actual vs Model) output: 10 input 2 output case

4.1.2.1.2.2 Index and Middle Finger

Standard regressors corresponding to the orders: na = [2 2; 2 2], nb = [6 1 1 0 0 0 1 0 0 0; 6 1 1 0 0 0 1 0 0 0], nk = [0 0 0 1 1 1 0 1 1 1; 0 0 0 1 1 1 0 1 1 1]. The model performance was as follows: Fit to estimation data: [99.81;99.92] FPE: 3.603e-10and a mean squared error value of 6.274e-05

The 1-output and 2- outputs plots based on 10 s-EMG signals show much better performance in tracking the index finger movement compared to when it was based on only 2 s-EMG signals. However, the force magnitude range is not met at all places.

4.1.2.1.3 No. of s-EMG signal inputs considered=12

4.1.2.1.3.1 Index Finger

Standard regressors corresponding to the orders: na = [2] nb = [6 1 1 0 0 0 1 0 0 0 0 0] nk = [0 0 0 1 1 1 0 1 1 1 1 1] The model performance was as follows: Fit to estimation data: 99.92% (prediction focus) FPE: 5.732e-05and a mean squared error value of 5.647e-05

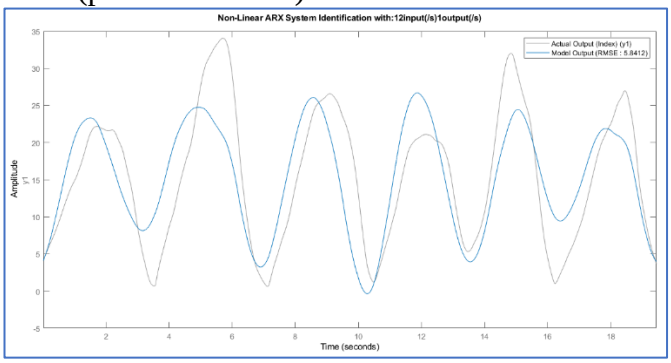


Figure 80 Non-linear ARX regression (Actual vs Model) output:

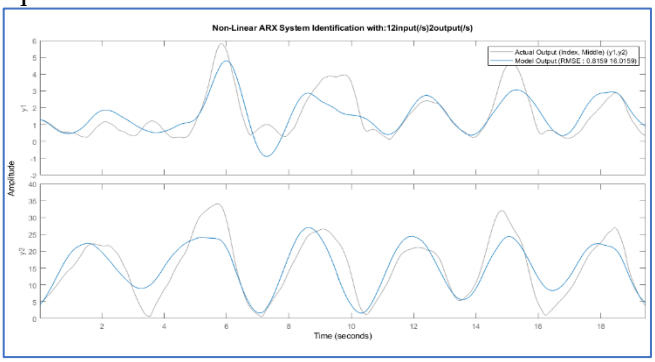


Figure 59 Non-linear ARX regression (Actual vs Model) output: 12 input 2 output case

4.1.2.1.3.2 Index and Middle Finger

Standard regressors corresponding to the orders:na = [2 2; 2 2] nb = [6 1 1 0 0 0 1 0 0 0 0 0; 6 1 1 0 0 0 1 0 0 0 0 0] nk = [0 0 0 1 1 1 0 1 1 1 1 1; 0 0 0 1 1 1 0 1 1 1 1 1] The model performance was as follows: Fit to estimation data: [99.81;99.92]% (prediction focus) FPE: 3.603e-10and a mean squared error value of 6.274e-05.

The 1-output and 2- outputs plots based on all the 12 s-EMG signals show a similar performance in tracking the index finger movement just as in the 10 s-EMG cases. In the 2-outputs case, however,

the performance on the index finger movement case is noticeably better here than it was in the previous case with 10-inputs.

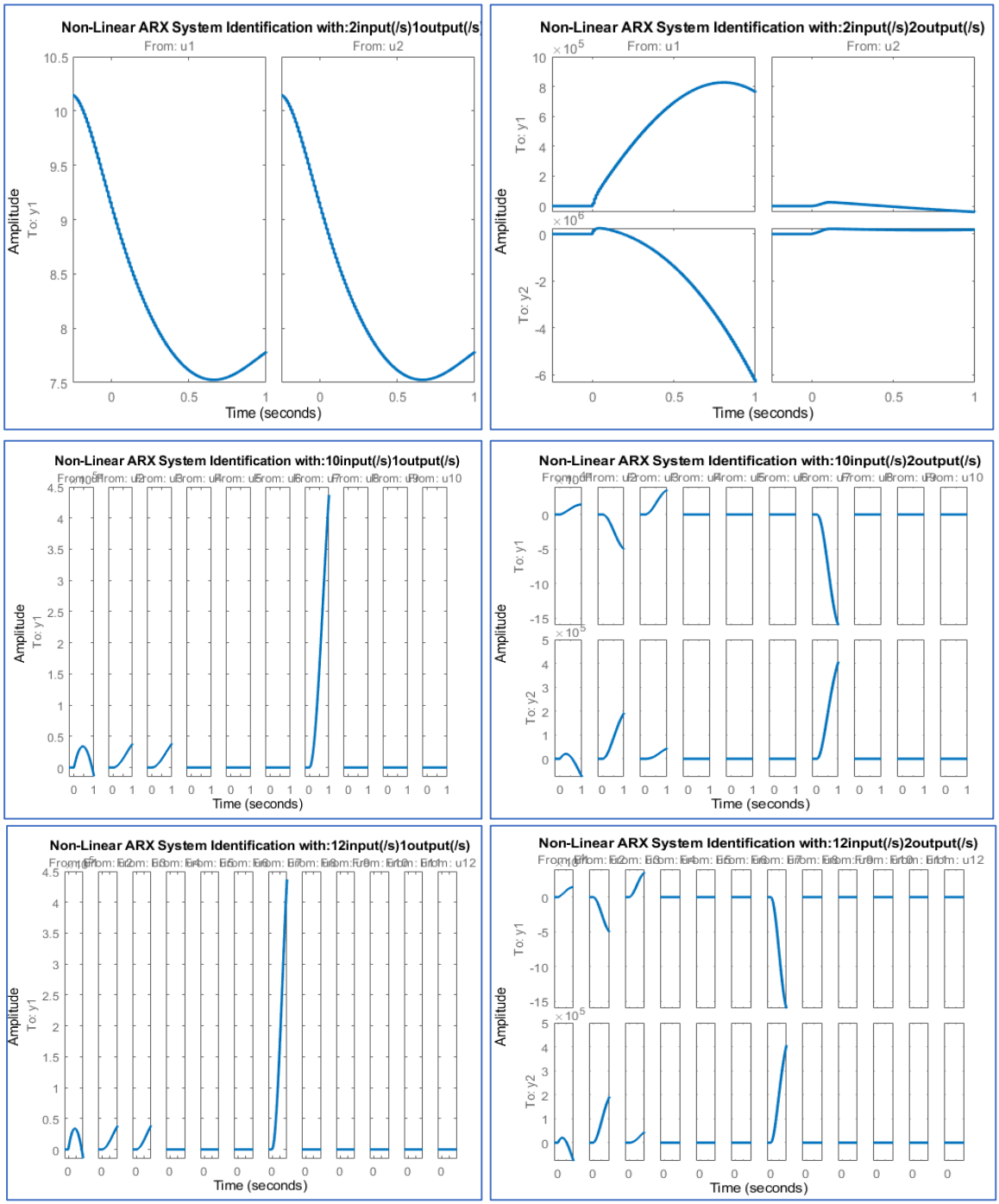


Figure 60 step response of Non- linear ARX models corresponding to each input and output combination

The step-responses show the impact of the 1st, 2nd, 3rd, and 7th sensors around the forearm on the index and middle fingers but indicate towards a system that is highly unstable.

4.1.2.2 LSTM

A mini-batch size that is between 1 and the size of the entire training set, is not expected to decrease with every iteration because splitting a training set into mini-batches is as if training on a different training set each iteration so it is going to be noisy but not as chaotic as it would get if a mini-batch size of 1 were used. Overall, despite the noisiness, the loss trends downwards.

4.1.2.2.1 No. of s-EMG signal inputs considered=2

4.1.2.2.1.1 Index Finger:

The index finger movement tracking based on 2 s-EMG signals plots were obtained as shown below. The left plot was obtained when the LSTM regression was performed on a system with GPU and the right plot was obtained on a system without GPU enabled.

Overall, the models can track the finger movement, but the force range is visibly better matched to the actual output in the plot taken from system without GPU compared to the one that was obtained from a system that had GPU enabled. In other words, a slower system enabled the neural network to learn the pattern better than a faster system that took nearly half the time that it took to perform the same task on a system without GPU.

The plots below show the progress during training in terms of loss and Root Mean Squared Error(RMSE). As the training progress they can be seen to be dropping indicating how the LSTM network is trying to minimise the loss on the cost function. However, after a certain point the network stagnates on the learning.

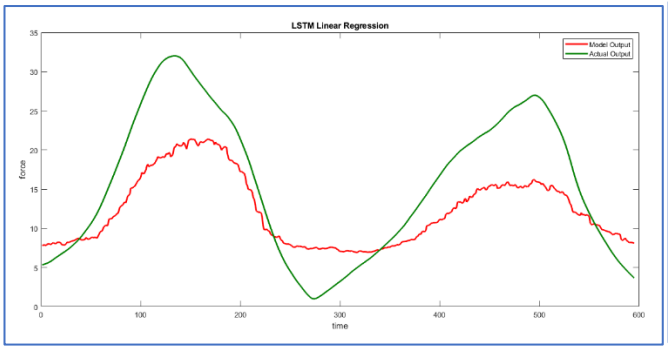


Figure 62 LSTM regression (Actual vs Model) output: 2 input 1 output case (with GPU)

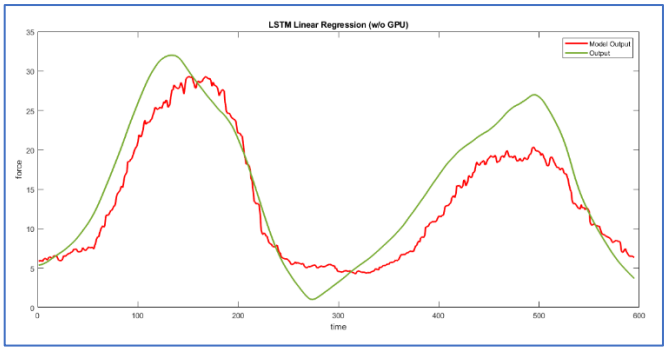


Figure 61 LSTM regression (Actual vs Model) output: 2 input 1 output case (without GPU)

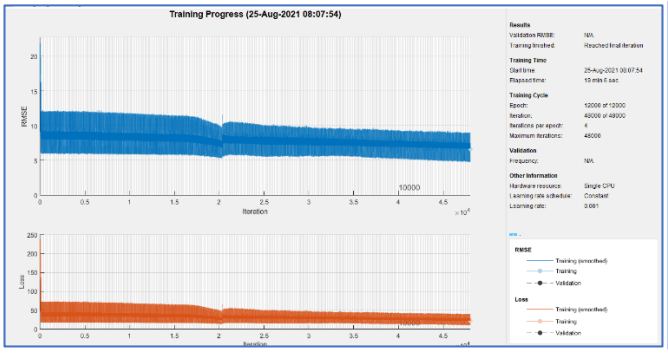


Figure 64 LSTM Model training accuracy and loss plots : 2 input 1 output case (with GPU)

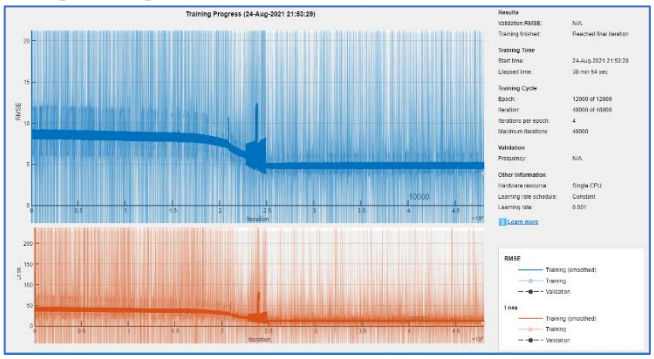


Figure 63 LSTM Model training accuracy and loss plots : 2 input 1 output case (without GPU)

4.1.2.2.1.2 Index and Middle Finger:

In another case, where 2-finger movements were predicted based on the same 2 s-EMG signals, the output plots again show how a system without GPU improved the performance of the predicted outputs drastically. However, when compared to the 1-output case the

performance of the model on the index finger in the 2-outputs case is worser in the plots taken on a system without GPU.

The training progress plots are again on a monotonic decent indicating that the system is making progress and improving the system further by decreasing the loss.

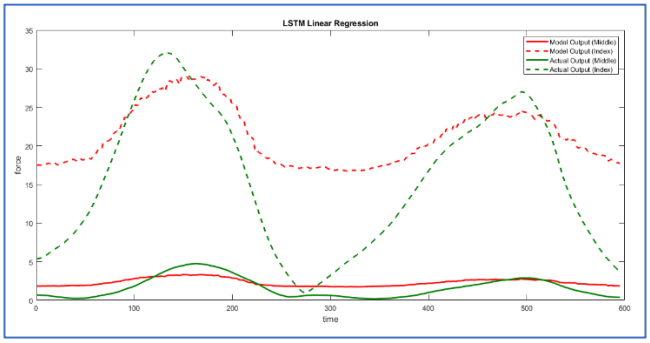


Figure 66 LSTM regression (Actual vs Model) output: 2 input 2 output case (with GPU)

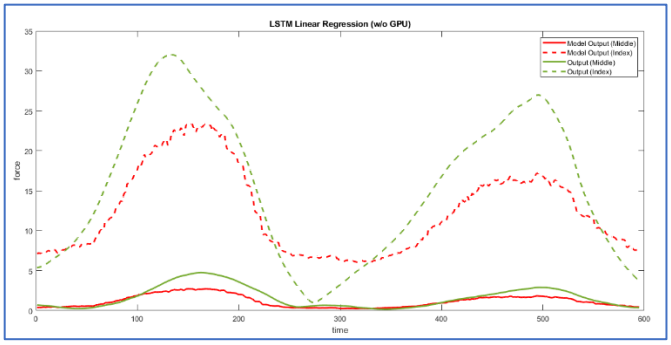


Figure 65 LSTM regression (Actual vs Model) output: 2 input 2 output case (without GPU)

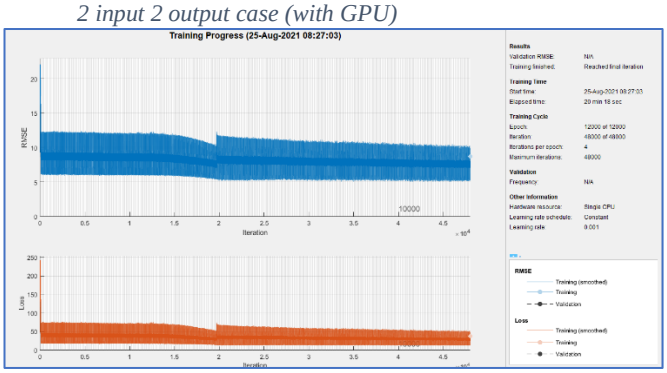


Figure 68 LSTM Model training accuracy and loss plots : 2 input 2 output case (with GPU)

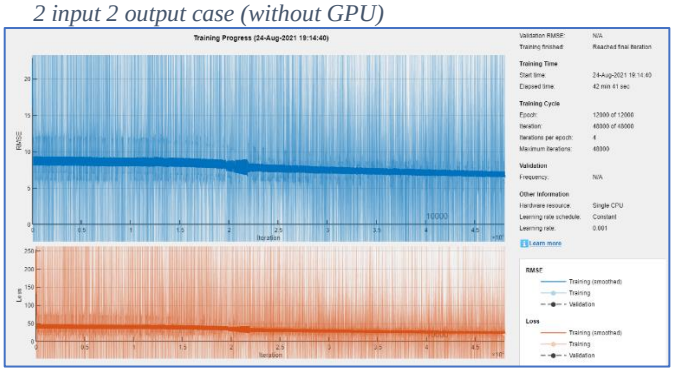


Figure 67 LSTM Model training accuracy and loss plots : 2 input 2 output case (without GPU)

4.1.2.2.2 No. of s-EMG signal inputs considered=10

4.1.2.2.2.1 **Index Finger**:

The predicted output based on 10 s-EMG signals shows a significant improvement compared to before, just as it should, and the performance of the LSTM network on a system with GPU also seems to be picking up.

Similarly, the training progress plots show a sudden drop in the first few iterations of the training and continue dropping but at a very slow pace indicating that the neural network learnt more in the first few iterations than it did in the rest of the duration.

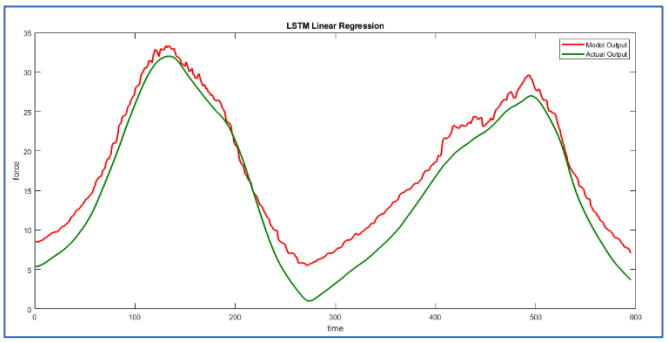


Figure 70 LSTM regression (Actual vs Model) output: 10 input 1 output case (with GPU)

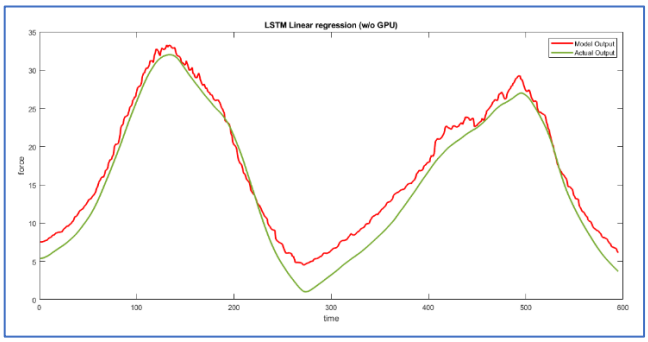


Figure 69 LSTM regression (Actual vs Model) output: 10 input 1 output case (without GPU)

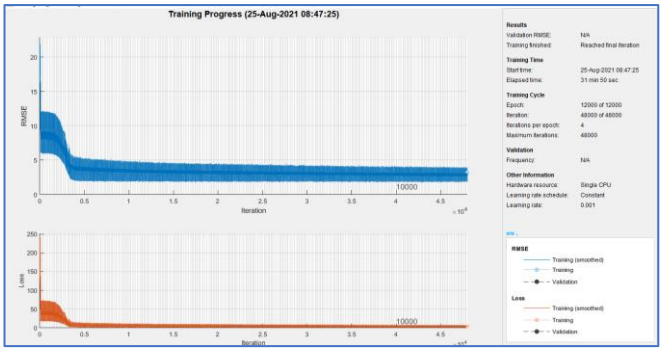


Figure 72 LSTM Model training accuracy and loss plots : 10 input 1 output case (with GPU)

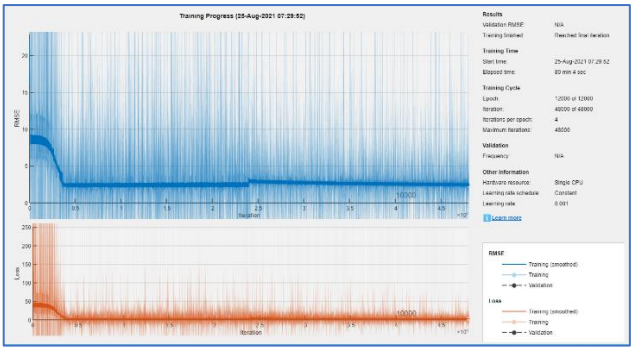


Figure 71 LSTM Model training accuracy and loss plots : 10 input 1 output case (without GPU)

4.1.2.2.2 Index and Middle Finger:

The performance on the system with GPU is now on par with that on the system without GPU in the 2- outputs case based on 10 s-EMG signals. The predicted force outputs for both finger movements are reasonably close to the actual values.

The loss and RMSE plots show a similar behaviour in pattern to that observed in the 1-output case.

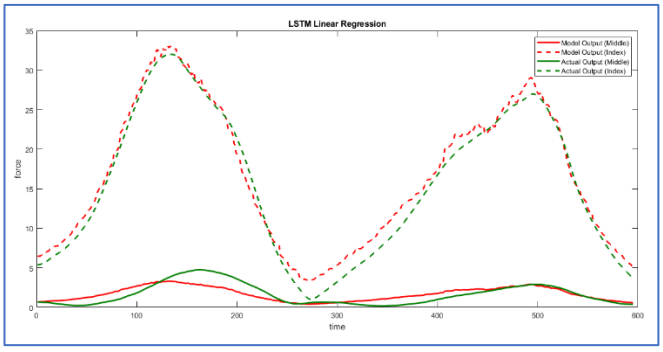


Figure 74 LSTM regression (Actual vs Model) output: 10 input 2 output case (with GPU)

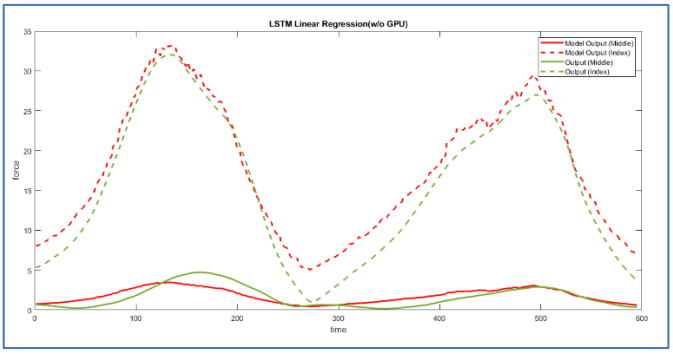


Figure 73 LSTM regression (Actual vs Model) output: 10 input 2 output case (without GPU)

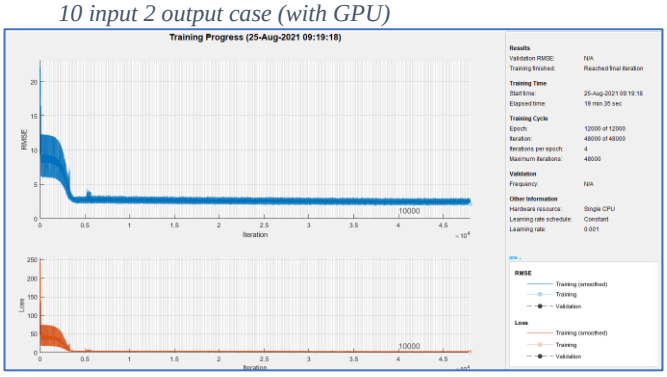


Figure 76 LSTM Model training accuracy and loss plots : 10 input 2 output case (with GPU)

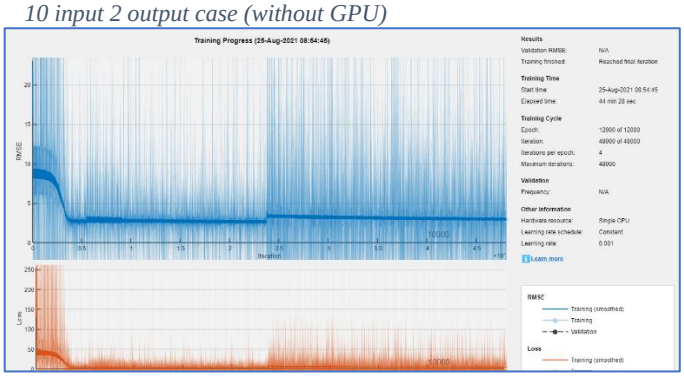


Figure 75 LSTM Model training accuracy and loss plots : 10 input 2 output case (without GPU)

4.1.2.2.3 No. of s-EMG signal inputs considered=12

4.1.2.2.3.1 Index Finger:

Contrary to what is expected the plot on the system with GPU shows better tracking than the one taken from a system with GPU in the 12 s-EMG case when only 1-output is being considered. Overall, the neural network is seen to be tracking the finger movement.

The training progress plot shows how the LSTM network learns very quickly in the first few iterations and stagnates on learning beyond a point.

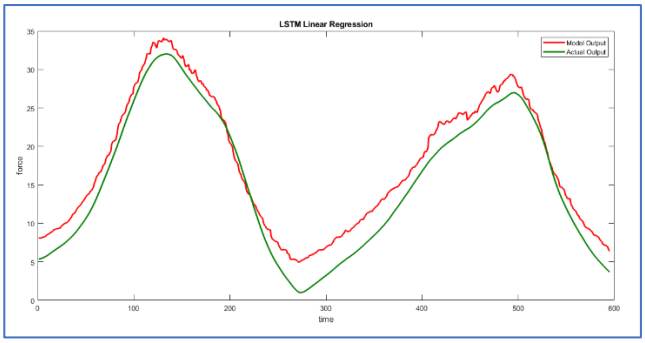


Figure 78 LSTM regression (Actual vs Model) output: 12 input 1 output case (with GPU)

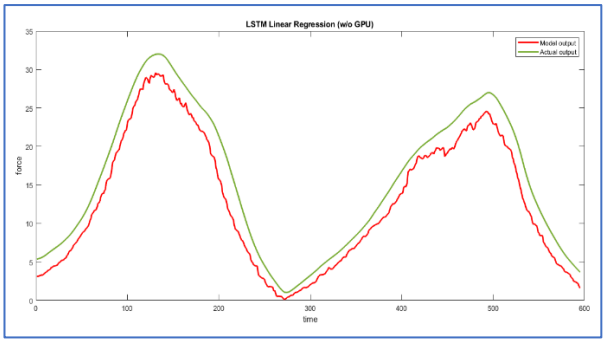


Figure 77 LSTM regression (Actual vs Model) output: 12 input 1 output case (without GPU)

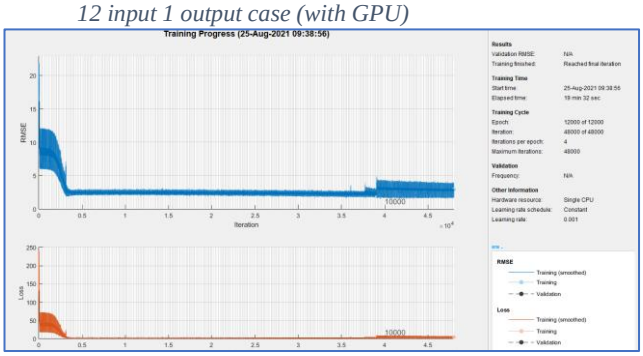


Figure 80 LSTM Model training accuracy and loss plots : 12 input 1 output case (with GPU)

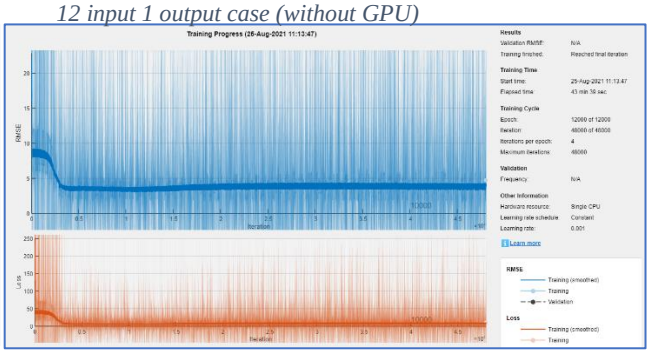


Figure 79 LSTM Model training accuracy and loss plots : 12 input 1 output case (without GPU)

4.1.2.2.3.2 Index and Middle Finger:

The 2-outputs case based on 12 s-EMG signals however displays the best performance seen in all the cases until now. In the middle finger case alone, the plot looks slightly distorted in comparison to the actual output plot.

The training progress made is similar to the 1-output case.

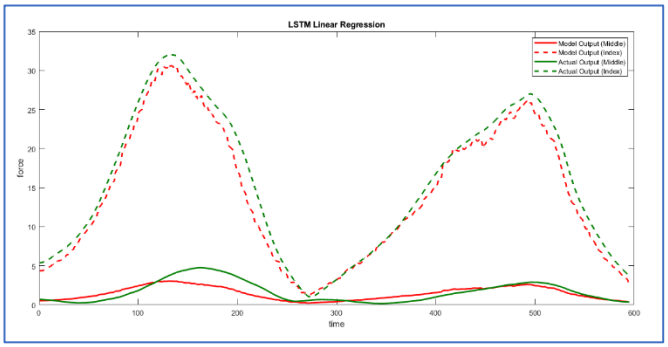


Figure 82 LSTM regression (Actual vs Model) output: 12 input 2 output case (with GPU)

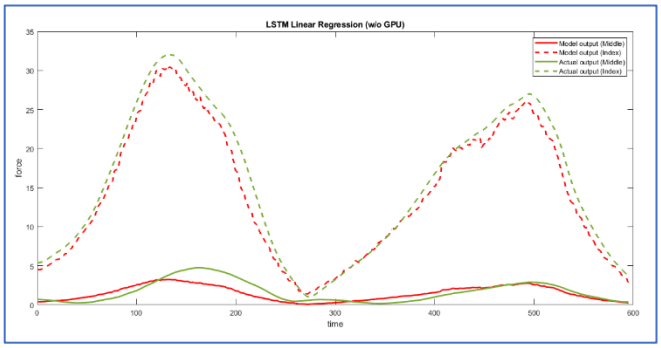


Figure 81 LSTM regression (Actual vs Model) output: 12 input 2 output case (without GPU)

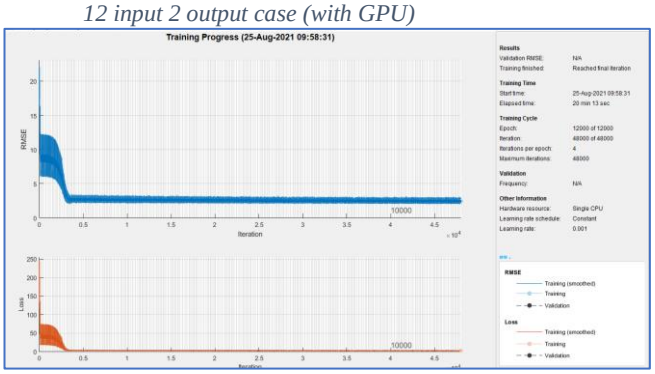


Figure 84 LSTM Model training accuracy and loss plots : 12 input 2 output case (with GPU)

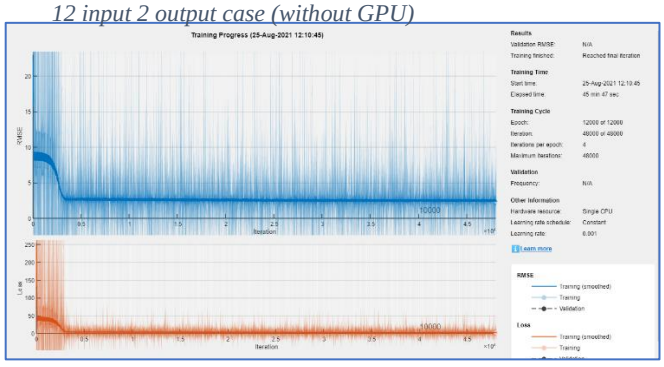


Figure 83 LSTM Model training accuracy and loss plots : 12 input 2 output case (without GPU)

4.1.2.3 GAN

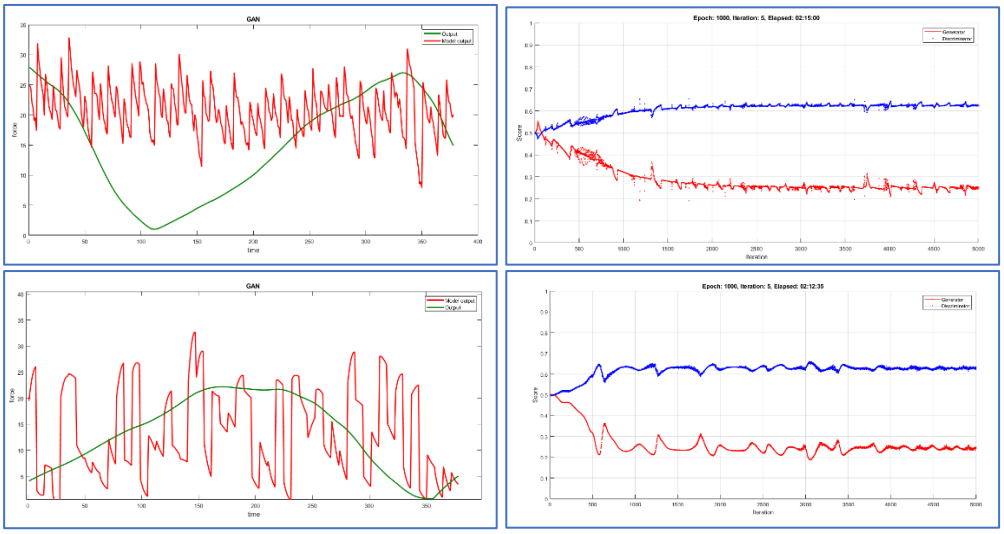


Figure 85 (Top) GAN regression output(Left) and score plot(Right) for case A hyper parameters; (Bottom) GAN regression output(Left) and score plot(Right) for case B hyper parameters

The GAN technique does not rely on the number of inputs to the system. The generator component takes in noise and gives out fake data that is made to look exactly like the actual data(discussed in detail in section 3.3.2.3). In this process of trying to replicate the actual data the generator improves at deceiving the discriminator which must tell if the data it just received was real or fake.

Unfortunately, however, the study conducted, with the architecture shown in section 3.3.2.3 on GANs, showed that the discriminator was convinced that the generator generated data was actual data because the generated data was in the range of the actual data. In other words, it was unable to capture the pattern in the data after hours of training. It could merely capture the range.

A wide range of hyperparameter tuning was experimented with by trying different values for the learning rate, the number of hidden units in the generator and the discriminator, the number of hidden units in the LSTM network, the gradient decay and the squared gradient decay used by the Adam Optimiser, the number of epochs, the mini-batch size, to name a few, based on theoretical knowledge.

For instance, the learning rate was decreased in an attempt to slow down the learning thereby allowing more time for the generator to pick up the pattern of the actual data so that it can better deceive the discriminator and likewise not allowing the discriminator to learn too quickly to tell apart between real and fake data.

The mini-batch size was increased until a balance was found between computational complexity and speed of reaching the optimum.

The number of epochs was increased to see if it led to any sudden pick up in the training progress.

The number of hidden units in the generator and discriminator were increased to see if this improved the pattern recognition and then decreased in an attempt to make the system less complex for the small data size(1^{st} , 2^{nd} ,3^{rd} ,4^{th}, 5^{th} repetitions) on which the training was being conducted to avoid overfitting on noise input in case of the generator.

The gradient decay factor and the scaled gradient decay factor were explained in section 3.3.2.3. They are used to update the weights in an Adam Optimiser. Increasing them would mean considering a greater number of past dw values and dw^2 values in updating the weights and bias each time, through use of an exponential moving average.

4.1.2.4 Comparison among the non-linear regression techniques used.

The LSTM Regression technique showed a better performance over the other non-linear regression methods considered such as the NARX and the GAN. The reason for the GANs' under-performance can be attributed to the open architecture used in this study implying that there was no feedback to the generator whatsoever on how it was doing other than the loss minimisation it was attempting to do on the negative log of the number of generated data that the discriminator wrongly guessed as true data. Overall, an 'Unsupervised Learning' was attempted on the force data using GANs. A close-looped mode where the generator has access to how it did in comparison to the actual data would significantly improve the performance of the generator without it having to rely solely on the number of fake data that the discriminator classes incorrectly. In short, the maximum likelihood-based training on an open-loop system is highly inefficient and therefore a closed-loop system could be an alternative.

A non-linear ARX on the other hand gave a poor fit in some cases(esp. the 2-input case) because of its model complexity compared to the size of the training data set. Also, a Wavelet non-linear function was used to model the system instead of which a neural network based non-linear function could be experimented with.

The LSTM network was able to perform better because of the advantage it has of being able to capture long-term dependencies in time-series data well using the concept of memory cells(explained in detail in section 3.3.2.1).

Table 10 RMSE Values for non-linear regression tasks

RMSE Values						
I/O Case **Model Type**	**12 Input** **1 Output**	**12 Input** **2 Output**	**10 Input** **1 Output**	**10 Input** **2 Output**	**2 Input** **1 Output**	**2 Input** **2 Output**
NARX	5.8412	0.8159 16.0159	5.8412	0.8159 16.0159	12.4004	2.5523 16.0159
LSTM w/o GPU	2.7983	0.8025 1.6475	2.0309	0.8114 2.3854	3.7449	0.8200 5.4606
LSTM w GPU	2.4351	0.8305, 1.6684	2.7030	0.8123, 1.3598	6.1396	1.1155, 8.0005

Table 11 Goodness of fit (R^2 square value) for non-linear regression tasks

R^2 Values (goodness of fit)						
I/O Case **Model Type**	**12 Input** **1 Output**	**12 Input** **2 Output**	**10 Input** **1 Output**	**10 Input** **2 Output**	**2 Input** **1 Output**	**2 Input** **2 Output**
NARX	66.6863	62.9577 244.9232	66.6863	62.9577 244.9232	46.2361	359.1784 244.9232
LSTM w/o GPU	97.0048	51.3679 90.1716	89.154396	46.6206 87.3417	75.7308	45.9761, 46.0626
LSTM w GPU	92.7058	40.5317, 92.6275	87.3223	41.079163, 92.7785	37.2928	39.5167, 54.8020

NARX had the least goodness of fit and highest RMSE values amongst non-linear regression tasks and over all among all tried regression methods as illustrated.

Table 12 Computation time for LSTM with and without GPU

I/O Case	12 Input 1 Output	12 Input 2 Output	10 Input 2 Output	2 Input 1 Output	2 Input 2 Output	10 Input 1 Output
LSTM With GPU	19 min 32 Sec	20 min 13 sec	19 min 35 Sec	19 min 6 sec	20 min 8 sec	31 min 50 sec
LSTM Without GPU	49 min 39 sec	45 min 47 sec	44 min 28 sec	38 min 54 sec	42 min 41Sec	80 min 4 sec

As individually explained with 6 cases the LSTM computation times are high when a GPU was not utilized, as the model learnt slowly utilizing the processor cores without any parallelization which had indicated higher efficiency in modelling.

4.2 Classification Techniques

4.2.1 KNN:

A K-Nearest Neighbours technique with K=10 was used for classification of the s-EMG signals into one of 10 movements, which included Flexion of Little Finger(41), Flexion of the Ring Finger(42), Flexion of the Middle Finger(43), Flexion of the Index Finger(44), Abduction of the Thumb(45), Flexion of the Thumb(46), Flexion of the Index and Little Finger(47), Flexion of Ring and Middle Finger(48), Flexion of Index and Thumb(49) and Rest periods(0), with their respective experiment numbers within brackets.

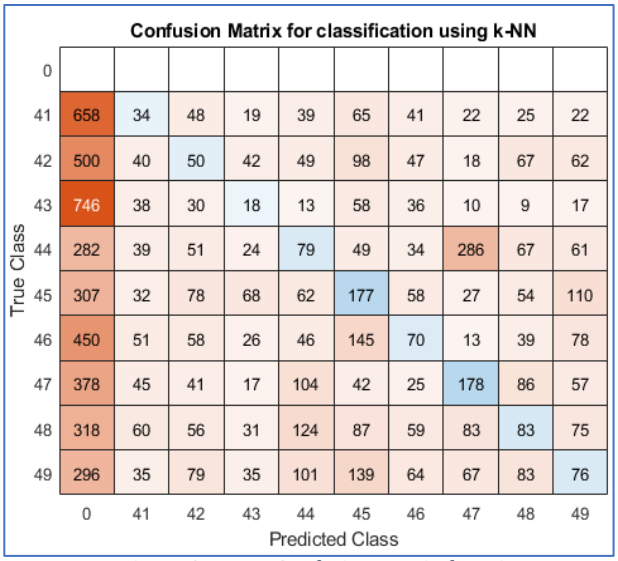


Figure 86 KNN Confusion matrix for 2 inputs

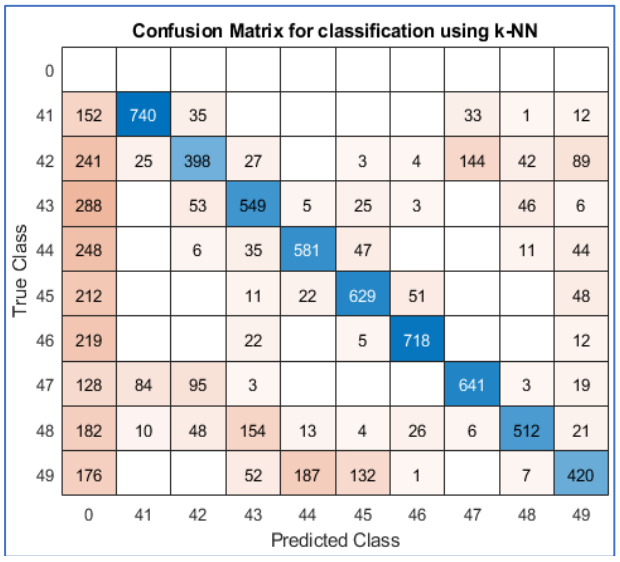


Figure 87 Confusion matrix of KNN classifier for 10 inputs case

4.2.1.1 No. of s-EMG signal inputs considered=2

The first plot shows the case where classification was attempted based on 2-inputs. The purpose of this was to study how feasible it was to discern what movement an amputee who can only have 2 sensors on is trying to make. As is evident from the Confusion Matrix the accuracy of the plots is poor. Summation of the diagonal numbers gives the total number of inputs that were labelled correctly. The ratio of correctly classified movements to incorrectly classified movements is very low.

4.2.1.2 No. of s-EMG signal inputs considered=10

The second plot shows the case where classification was attempted based on 10-inputs. The purpose of this was to study how feasible it was to discern what movement an amputee who can only have 10 sensors on is trying to make. As is evident from the Confusion Matrix the accuracy of the plots is significantly better. More number of correctly classed movements can be found along the diagonal.

4.2.1.3 No. of s-EMG signal inputs considered=12

The third plot shows the case where classification was attempted based on 12-inputs. The purpose of this was to study how feasible it was to discern what movement a fully intact person is trying to make. As is evident from the Confusion Matrix the accuracy of the plots is improved. The ratio of correctly classified movements to incorrectly classified labels is greater than 1.Increasing the K value beyond 10, worsened the performance but not by a very large degree. K=10 gave the best accuracy for classification into different kinds of finger movements that the subjects were made to do during the subjective tests, based on the s-EMG signals data.

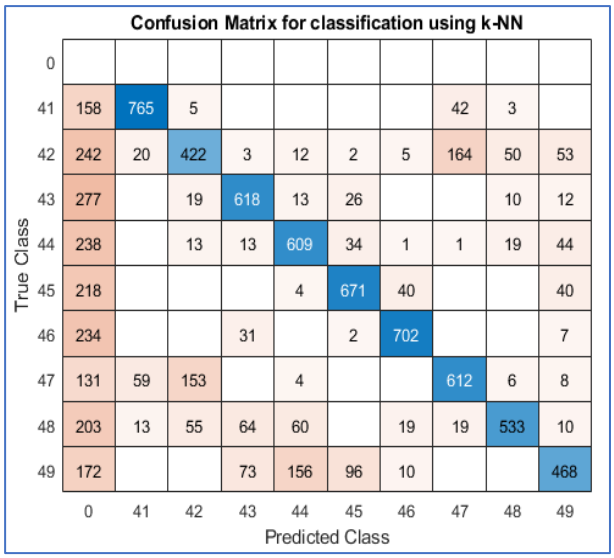


Figure 88 Confusion matrix of KNN classification models for 12 inputs

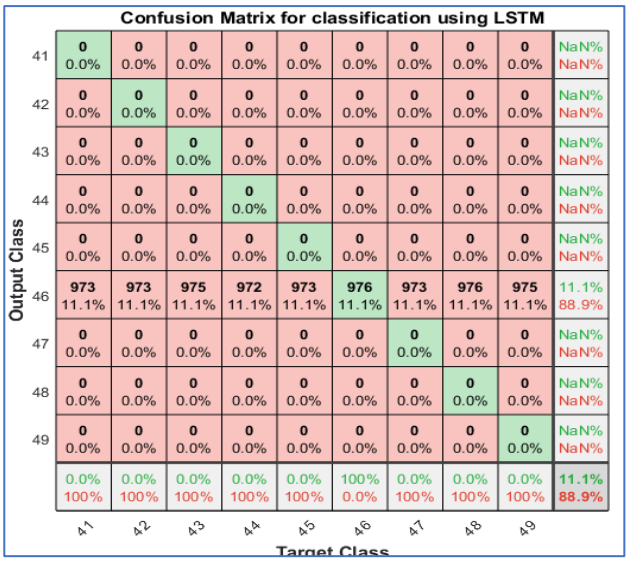


Figure 89 Confusion matrix for LSTM for 2 inputs case

4.2.2 LSTM:

A bi-directional LSTM architecture was used because of the availability of the entire time-series sequence at the beginning of training because it draws its information from both feedforward layers, namely, the forward layer and the backward layer(details in section 3.4.2) in deciding for the output at the current time step. To avoid a Class bias, rest periods were removed, and the classification was only performed across 9 finger movements. A SoftMax layer was used to classify the movements into 9 classes.

4.2.2.1 No. of s-EMG signal inputs considered=2:

The Confusion matrix shows the poor performance of this architecture in the case where only 2 s-EMG signals data is available to make the classification.

The training progress plot shows the accuracy and loss of the neural network throughout training and the loss can be seen to be gradually decreasing.

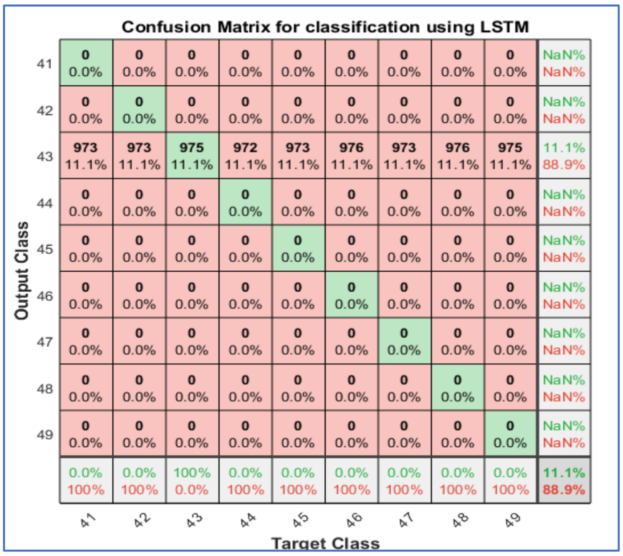


Figure 90 Confusion matrix for LSTM for 10 input case

4.2.2.2 No. of s-EMG signal inputs considered=10:

In the case where 10 s-EMG signals data is available to make the classification, the Confusion matrix displays a very slight improvement in performance compared to before but not a significant improvement as one would expect given how 5 times greater number of input data is available to the neural network for making the decision of Classification.

The training progress plot shows the accuracy and loss of the neural network throughout training and the loss can be seen to be gradually decreasing.

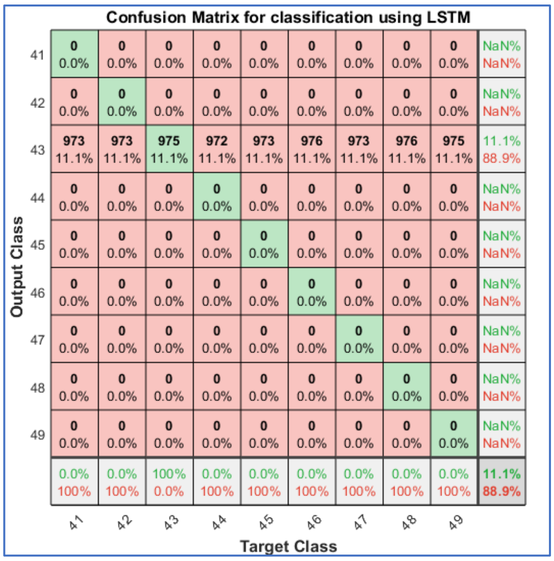


Figure 92 Confusion matrix for LSTM for 12 input case

4.2.2.3 No. of s-EMG signal inputs considered=12:

The 12 s-EMG signal case again shows an even slighter improvement in performance compared to the 10 s-EMG case but is still not a significant improvement.

The training progress plot shows the accuracy and loss of the neural network throughout training and the loss can be seen to be gradually decreasing with the accuracy plot remaining nearly the same throughput the training.

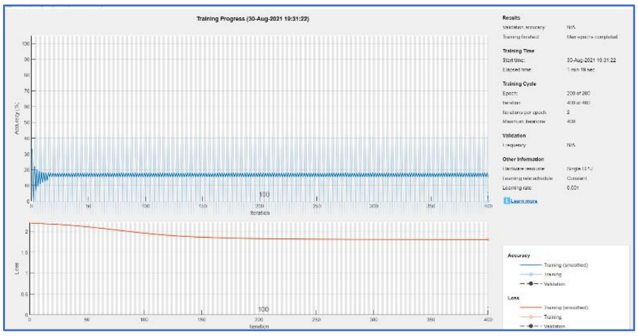


Figure 91 LSTM Classification Model training accuracy and loss plots : 10 inputs

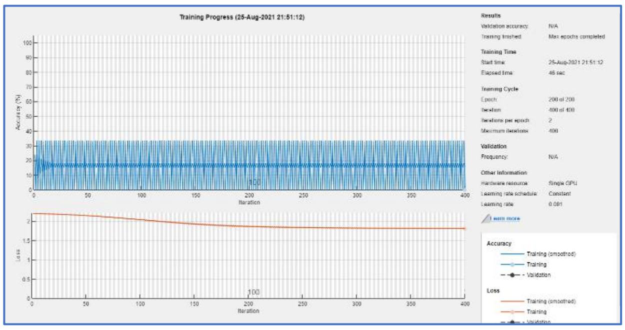


Figure 93 LSTM Classification Model training accuracy and loss plots : 12 inputs

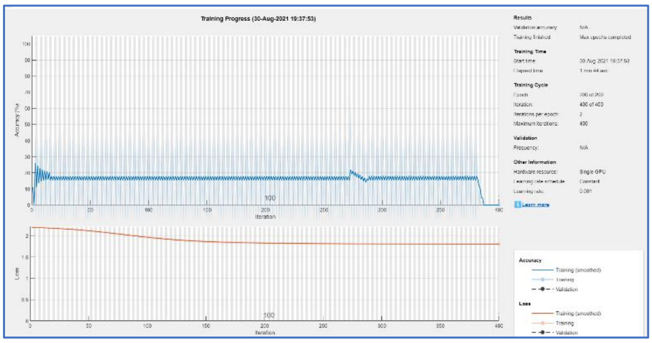


Figure 94 LSTM Classification Model training accuracy and loss plots : 2 inputs

4.2.3 Comparison among the Classification techniques used:

A compilation of the accuracy values for both the Classification techniques used shows an improvement as the number of s-EMG values available increases. However, LSTMs' overall performance is very poor despite the strength of a bi-directional LSTM at capturing dependencies from anywhere in the sequence before making a prediction for the current output. The data was given to the LSTM classification network in the same format as how it was given to the K-NN algorithm for training. It was also made sure that there are same number of data in the training set of each class to avoid bias. However, the performance ended up being poor. The LSTM kept classifying all the inputs into just 1 class and a different one each time.

A way to correct this problem with LSTM would be to shuffle the data a bit to see if it improves the performance.

Table 13 Model accuracy for different classification tasks

Classification Accuracy		
Input	**KNN**	**LSTM**
2 sEMG	8.72%	11.09%
10 sEMG	59.18%	11.12%
12 sEMG	61.60%	11.13%

5 Conclusion:

5.1 Answers to research questions

1. Classical methods like ordinary least squares, Bayesian least squares and even auto regressive models such as ARX have better regression fitness in comparison to their deep learning counterparts. ML methods also introduce a massive increase in computational complexity.
2. Linear models have a better prediction of force output from s-EMG signals in comparison to non-linear algorithms, this could be attributed to the quasi-linear relationship that has been documented in literature for s-EMG signals.
3. Classification problem is solved with a moderate accuracy by the distance learning ML algorithm of KNN which is however comparatively five times better than the best case of deep learning architectures like LSTM when the same balanced data is provided as input to both methods.
4. The unsupervised deep learning model GAN has estimated the range of output but massively failed to capture the dynamic trend of output, which can be improved in a supervised closed loop setting, thereby proving supervised approach better for pattern recognition of s-EMG.
5. All classical models used have negligible computation times and a few cases of deep learning such as the unsupervised GAN is a massive load on an embedded chip onboard a myoelectric prosthesis. Hence simpler methods like least squares can be used to compensate computation load and their efficiency can be improved by utilizing mathematical constructs such as Bayesian statistics.

5.2 Inference

There is a place and use for every method, even in the modern day. Fields like machine learning may be rich in their computational ability to map all potential relationships and have

their own advantages in modelling highly complex systems, but their need stems from the lack of simpler mathematical frameworks, and they are sought after as an alternative.

But the spotlight of near accurate estimates, has fuelled the addition of machine learning techniques in more than necessary spheres of life, as demonstrated here for the case of s-EMG in the context of prosthesis, introducing a walking supercomputer in the place of a human limb will be a burden. This is the cost for the advertised accuracy that can ensure near real action. The claim is evident from the ratio of number of quality literature published in the field vs clinical and commercially available prosthetics.

6 Future Work:

NinaPro dataset now has 10 databases, the latest database (2020) addition is also referred to as MeganePro[47], [48] which has eye-tracking and computer vision datasets augmenting s-EMG signal data. Thereby making it an ideal candidate for blend of image processing and deep learning techniques. A premise that shall be well suited for complex architectures like GAN, for regression and sequence generation problems, but in a closed loop setting. There are 4 sub-datasets within this database 10 [49]–[52]. Also it is the first multi-modal dataset within robotic hand prosthesis aimed at handling the hand-eye coordination effectively.

Multimodal systems can be handled effectively by Bayesian Least squares as, the inherent advantage of Bayesian is to perform better in multivariate inputs, this capability can be expanded to handle multimodal systems easily. Simultaneous classification and regression for the databases 1-9 of NinaPro has been demonstrated earlier in [24]. Thus these modern refinements in datasets in this s-EMG pattern recognition problem and parallel evolution of pattern recognition strategies along with computational advancements pave the carpet for future work. The real-time improvement of quality of life for amputees stemming from this, would be ultimate reward of such results.

REFERENCES

[1] J. W. Sensinger and S. Dosen, “A Review of Sensory Feedback in Upper-Limb Prostheses From the Perspective of Human Motor Control,” *Front. Neurosci.*, vol. 14, p. 345, Jun. 2020, doi: 10.3389/fnins.2020.00345.

[2] I. Kyranou, S. Vijayakumar, and M. S. Erden, “Causes of Performance Degradation in Non-invasive Electromyographic Pattern Recognition in Upper Limb Prostheses,” *Front. Neurorobotics*, vol. 12, p. 58, Sep. 2018, doi: 10.3389/fnbot.2018.00058.

[3] P. Svensson, U. Wijk, A. Björkman, and C. Antfolk, “A review of invasive and non-invasive sensory feedback in upper limb prostheses,” *Expert Rev. Med. Devices*, vol. 14, no. 6, pp. 439–447, Jun. 2017, doi: 10.1080/17434440.2017.1332989.

[4] M. R. Dawson, J. P. Carey, and F. Fahimi, “Myoelectric training systems,” *Expert Rev. Med. Devices*, vol. 8, no. 5, pp. 581–589, Sep. 2011, doi: 10.1586/erd.11.23.

[5] P. Padmanabhan and S. Puthusserypady, “Nonlinear analysis of EMG signals - a chaotic approach,” in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Francisco, CA, USA, 2004, vol. 3, pp. 608–611. doi: 10.1109/IEMBS.2004.1403231.

[6] E. Scheme and K. Englehart, “Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use,” *J. Rehabil. Res. Dev.*, vol. 48, no. 6, p. 643, 2011, doi: 10.1682/JRRD.2010.09.0177.

[7] I. Campanini, C. Disselhorst-Klug, W. Z. Rymer, and R. Merletti, “Surface EMG in Clinical Assessment and Neurorehabilitation: Barriers Limiting Its Use,” *Front. Neurol.*, vol. 11, p. 934, Sep. 2020, doi: 10.3389/fneur.2020.00934.

[8] A. Phinyomark and E. Scheme, “EMG Pattern Recognition in the Era of Big Data and Deep Learning,” *Big Data Cogn. Comput.*, vol. 2, no. 3, p. 21, Aug. 2018, doi: 10.3390/bdcc2030021.

[9] E. N. Kamavuako, E. J. Scheme, and K. B. Englehart, “Determination of optimum threshold values for EMG time domain features; a multi-dataset investigation,” *J. Neural Eng.*, vol. 13, no. 4, p. 046011, Aug. 2016, doi: 10.1088/1741-2560/13/4/046011.

[10] M. Atzori *et al.*, “Building the Ninapro database: A resource for the biorobotics community,” in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, Rome, Italy, Jun. 2012, pp. 1258–1265. doi: 10.1109/BioRob.2012.6290287.

[11] M. Atzori *et al.*, “Electromyography data for non-invasive naturally-controlled robotic hand prostheses,” *Sci. Data*, vol. 1, no. 1, p. 140053, Dec. 2014, doi: 10.1038/sdata.2014.53.

[12] O. Lippold, “The relation between integrated action potentials in a human muscle and its isometric tension,” *J Physiol*, vol. 117, pp. 492–499, 1952.

[13] R. Merletti and P. Parker, *Electromyography : physiology, engineering, and noninvasive applications*. Hoboken, New Jersey: Wiley-Interscience, 2005.

[14] H. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *J Physiol*, vol. 117, pp. 500–544, 1952.

[15] A. Phinyomark, P. Phukpattaranont, and C. Limsakul, “Feature reduction and selection for EMG signal classification,” *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7420–7431, Jun. 2012, doi: 10.1016/j.eswa.2012.01.102.

[16] K. Englehart and B. Hudgins, “A robust, real-time control scheme for multifunction myoelectric control,” *IEEE Trans. Biomed. Eng.*, vol. 50, no. 7, pp. 848–854, Jul. 2003, doi: 10.1109/TBME.2003.813539.

[17] M. Atzori, “Data from: Electromyography data for non-invasive naturally controlled robotic hand prostheses.” Ninapro Repository, 2014. [Online]. Available: http://ninapro.hevs.ch
[18] M. Atzori *et al.*, “Data from: Electromyography data for non-invasive naturally controlled robotic hand prostheses.” Dryad Repository, Dec. 19, 2014.
[19] A. Gijsberts, M. Atzori, C. Castellini, H. Muller, and B. Caputo, “Movement Error Rate for Evaluation of Machine Learning Methods for sEMG-Based Hand Movement Classification,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 4, pp. 735–744, Jul. 2014, doi: 10.1109/TNSRE.2014.2303394.
[20] F. Stival, S. Michieletto, M. Cognolato, E. Pagello, H. Müller, and M. Atzori, “A quantitative taxonomy of human hand grasps,” *J. NeuroEngineering Rehabil.*, vol. 16, no. 1, p. 28, Dec. 2019, doi: 10.1186/s12984-019-0488-x.
[21] A. Chan and G. Green, “Myoelectric Control Development Toolbox,” *Conf. Can. Med. Biol. Eng. Soc.*, 2007.
[22] Y. Du, W. Jin, W. Wei, Y. Hu, and W. Geng, “Surface EMG-Based Inter-Session Gesture Recognition Enhanced by Deep Domain Adaptation,” *Sensors*, vol. 17, no. 3, p. 458, Feb. 2017, doi: 10.3390/s17030458.
[23] H. Shim and S. Lee, “Multi-channel electromyography pattern classification using deep belief networks for enhanced user experience,” *J. Cent. South Univ.*, vol. 22, no. 5, pp. 1801–1808, May 2015, doi: 10.1007/s11771-015-2698-0.
[24] T. Baldacchino, W. R. Jacobs, S. R. Anderson, K. Worden, and J. Rowson, “Simultaneous Force Regression and Movement Classification of Fingers via Surface EMG within a Unified Bayesian Framework,” *Front. Bioeng. Biotechnol.*, vol. 6, p. 13, Feb. 2018, doi: 10.3389/fbioe.2018.00013.
[25] A. Sebastian, P. Kumar, M. Anugolu, M. P. Schoen, A. Urfer, and D. S. Naidu, “Optimization of Bayesian Filters and Hammerstein-Wiener Models for EMG-Force Signals Using Genetic Algorithm,” in *ASME 2009 Dynamic Systems and Control Conference, Volume 1*, Hollywood, California, USA, Jan. 2009, pp. 713–720. doi: 10.1115/DSCC2009-2658.
[26] M. Ison and P. Artemiadis, “The role of muscle synergies in myoelectric control: trends and challenges for simultaneous multifunction control,” *J. Neural Eng.*, vol. 11, no. 5, p. 051001, Oct. 2014, doi: 10.1088/1741-2560/11/5/051001.
[27] S. Karimimehr, P. Ghaderi, and M. E. Andani, “Hand kinematics estimation using non-invasive surface sensors: a linear system identification approach,” in *2015 22nd Iranian Conference on Biomedical Engineering (ICBME)*, Tehran, Iran, Nov. 2015, pp. 239–244. doi: 10.1109/ICBME.2015.7404149.
[28] A. A. Akbari and M. Talasaz, “Prediction of Above-elbow Motions in Amputees, based on Electromyographic(EMG) Signals, Using Nonlinear Autoregressive Exogenous (NARX) Model,” *Iran. J. Med. Phys.*, vol. 11, no. Issue 2,3, Aug. 2014, doi: 10.22038/ijmp.2014.3095.
[29] Z. Gao, R. Tang, Q. Huang, and J. He, “A Multi-DoF Prosthetic Hand Finger Joint Controller for Wearable sEMG Sensors by Nonlinear Autoregressive Exogenous Model,” *Sensors*, vol. 21, no. 8, p. 2576, Apr. 2021, doi: 10.3390/s21082576.
[30] R. A. ZANINI, “Parkinson EMG signal prediction and generation with neural networks,” *Previs. E Geração Sinais EMG Park. Com Redes Neurais 2020*, Jul. 2020, [Online]. Available: http://www.repositorio.unicamp.br/handle/REPOSIP/346211
[31] R. Fu, J. Chen, S. Zeng, Y. Zhuang, and A. Sudjianto, “Time Series Simulation by Conditional Generative Adversarial Net,” *ArXiv190411419 Cs Eess Stat*, Apr. 2019, Accessed: Aug. 29, 2021. [Online]. Available: http://arxiv.org/abs/1904.11419

[32] J. Yoon, D. Jarrett, and M. V. der Schaar, “Time-series Generative Adversarial Networks,” *Adv. Neural Inf. Process. Syst. 32*, no. 32, 2019.
[33] *Statistics and Machine Learning Toolbox*. Mathworks Inc. [Online]. Available: https://uk.mathworks.com/products/statistics.html
[34] *Econometrics Toolbox*. Mathworks Inc. [Online]. Available: https://uk.mathworks.com/products/econometrics.html
[35] *System Identification toolbox*. Mathworks Inc. [Online]. Available: https://uk.mathworks.com/products/sysid.html
[36] *Deep Learning toolbox*. Mathworks Inc. [Online]. Available: https://uk.mathworks.com/products/deep-learning.html
[37] *Parallel Computing Toolbox*. Mathworks Inc. [Online]. Available: https://uk.mathworks.com/products/parallel-computing.html
[38] M. Atzori *et al.*, “Characterization of a Benchmark Database for Myoelectric Movement Classification,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 23, no. 1, pp. 73–83, Jan. 2015, doi: 10.1109/TNSRE.2014.2328495.
[39] O. Fukuda, T. Tsuji, M. Kaneko, and A. Otsuka, “A human-assisting manipulator teleoperated by EMG signals and arm motions,” *IEEE Trans. Robot. Autom.*, vol. 19, no. 2, pp. 210–222, Apr. 2003, doi: 10.1109/TRA.2003.808873.
[40] F. V. G. Tenore, A. Ramos, A. Fahmy, S. Acharya, R. Etienne-Cummings, and N. V. Thakor, “Decoding of Individuated Finger Movements Using Surface Electromyography,” *IEEE Trans. Biomed. Eng.*, vol. 56, no. 5, pp. 1427–1434, May 2009, doi: 10.1109/TBME.2008.2005485.
[41] G. Li, A. E. Schultz, and T. A. Kuiken, “Quantifying Pattern Recognition—Based Myoelectric Control of Multifunctional Transradial Prostheses,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 2, pp. 185–192, Apr. 2010, doi: 10.1109/TNSRE.2009.2039619.
[42] C. J. De Luca, “The Use of Surface Electromyography in Biomechanics,” *J. Appl. Biomech.*, vol. 13, no. 2, pp. 135–163, May 1997, doi: 10.1123/jab.13.2.135.
[43] P. M. Hopkins, “Skeletal muscle physiology,” *Contin. Educ. Anaesth. Crit. Care Pain*, vol. 6, no. 1, pp. 1–6, Feb. 2006, doi: 10.1093/bjaceaccp/mki062.
[44] K. Mukund and S. Subramaniam, “Skeletal muscle: A review of molecular structure and function, in health and disease,” *WIREs Syst. Biol. Med.*, vol. 12, no. 1, Jan. 2020, doi: 10.1002/wsbm.1462.
[45] C. Castellini, A. E. Fiorilla, and G. Sandini, “Multi-subject/daily-life activity EMG-based control of mechanical hands,” *J. NeuroEngineering Rehabil.*, vol. 6, no. 1, p. 41, Dec. 2009, doi: 10.1186/1743-0003-6-41.
[46] I. Kuzborskij, A. Gijsberts, and B. Caputo, “On the challenge of classifying 52 hand movements from surface electromyography,” in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Diego, CA, Aug. 2012, pp. 4931–4937. doi: 10.1109/EMBC.2012.6347099.
[47] F. Giordaniello *et al.*, “Megane Pro: Myo-electricity, visual and gaze tracking data acquisitions to improve hand prosthetics,” in *2017 International Conference on Rehabilitation Robotics (ICORR)*, London, Jul. 2017, pp. 1148–1153. doi: 10.1109/ICORR.2017.8009404.
[48] G. Saetta *et al.*, “Gaze, behavioral, and clinical data for phantom limbs after hand amputation from 15 amputees and 29 controls,” *Sci. Data*, vol. 7, no. 1, p. 60, Dec. 2020, doi: 10.1038/s41597-020-0402-1.
[49] M. Cognolato *et al.*, “MeganePro dataset 1 (MDS1).” Harvard Dataverse, 2019. doi: 10.7910/DVN/1Z3IOM.

[50] G. Saetta *et al.*, “MeganePro dataset 2 (MDS2).” Harvard Dataverse, 2020. doi: 10.7910/DVN/78QFZH.
[51] G. Saetta *et al.*, “MeganePro Dataset "Clinical Interview and Neurocognitive Tests in Amputees” (MDSInfo).” Harvard Dataverse, 2020. doi: 10.7910/DVN/EJJ91H.
[52] G. Saetta *et al.*, “MeganePro dataset 4 (MDS4).” Harvard Dataverse, 2020. doi: 10.7910/DVN/F9R33N.

Appendix