**CLASSIFICATION OF 41 HAND AND WRIST MOVEMENTS VIA SURFACE ELECTROMYOGRAM USING DEEP NEURAL NETWORK**

**ABSTRACT**

Surface electromyography is an easy, non-invasive technique that allows the user to actively control the prosthesis (sEMG). However, the categorization of hand and wrist motions using sEMG has generated a wide variety of results in earlier experiments, including the acquisition methodology and the number of classes, but not entirely. This research investigates the deep neural network strategy to categorize 41 hand and wrist movements based on the sEMG data. The Ninapro project's database, one of the largest public sEMG databases for cutting-edge hand myoelectric prosthetics, was utilized to train and evaluate the suggested models. For this study, two datasets were used: DB5 with a low-cost 16-channel, 200-Hz setup and DB7 with a 12-channel, 2 kHz design. Our technique achieved an overall accuracy of 93.87 1.49 and 91.69 4.68%, respectively, with balanced accuracy of 84.00 3.40 and 84.66 4.78% for DB5 and DB7. The Southampton Hand Assessment Procedure (SHAP), a clinically validated hand functional assessment technique, only considered the six primary hand motions based on the six prehensile patterns, but we still saw a performance improvement. Using only the SHAP movements in DB5, a balance accuracy of 94.48 2.55% and a classification accuracy of 98.82 0.58% were attained.

**Keywords:** Surface Electromyogram, Hand Movement Classification, Deep Neural Network, Prosthetic Hand, Ninapro Database.

**CHAPTER 1**

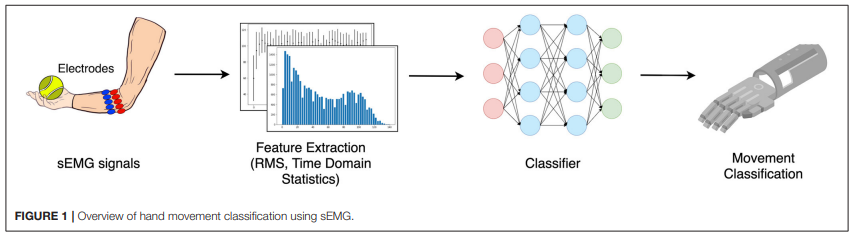
**INTRODUCTION**

Recent advancements in sensor technology, mechatronics, signal processing techniques, and edge computing hardware equipped with GPU make dexterous prosthetic hands with non-invasive sensors and control capabilities of machine learning possible. However, real-world applications of these prostheses and amputees’ receptions of them are still limited. Some of the main reasons include control difficulties, insufficient capabilities and dexterity levels, and the cost of the prosthesis. Moreover, frequent misclassifications of intended actions could lead to frustration and prostheses abandonment (Biddiss and Chau, 2007; Ahmadizadeh et al., 2017). Therefore, achieving a high level of reliability and robustness of human-machine interfaces is important for user experience and their acceptance of the prosthetic hand.

Over the last few years, multiple non-invasive control methods of prosthetic hands have been introduced and investigated; for example, surface electromyography (sEMG) (Fougner et al., 2012; Farina et al., 2014; Krasoulis et al., 2017; Pizzolato et al., 2017; Ameri et al., 2018; Li et al., 2018; Leone et al., 2019; Olsson et al., 2019; Junior et al., 2020), electroneurography (ENG) (Cloutier and Yang, 2013; Paul et al., 2018), mechanomyography (MMG) (Xiloyannis et al., 2015; Wilson and Vaidyanathan, 2017), and force myography (FMG) (Rasouli et al., 2015; Sadeghi and Menon, 2018; Ahmadizadeh et al., 2019). In particular, sEMG is a non-invasive technique for measuring the electrical activity of groups of muscles on the skin surface, which makes it a simple and straightforward way to allow the user to actively control the prosthesis. The overview of hand movement classification for the prosthetic hand using sEMG is shown in Figure 1. The muscle signals are collected as an input for the movement classification. The process typically involves feature extraction and classification process by the selected classifier.

Ameri et al. (2018) performed an sEMG classification of wrist movements based on the raw signal without any feature extraction using Support Vector Machine (SVM) and Convolutional Neural Network (CNN). The data was collected from 17 healthy individuals using eight pairs of bipolar surface electrodes with 1.2 kHz sampling rate equally spaced around the dominant forearm proximal to the elbow. A total of eight wrist movements were investigated. The classification results for the CNN and SVM were 91.61 ± 0.39 and 90.63 ± 0.31%, respectively. Li et al. (2018) investigated the use of sEMG for the classification of grasping force of a three-finger pinch movement for prosthetic control. The grasping force was separated into eight levels. A total of 15 healthy participants were recruited for the experiment. The signal was collected using a Thalmic Myo armband with 8 sEMG input channels and a 200 Hz sampling rate. Principal Component Analysis (PCA) was implemented to reduce the dimension of the extracted features to shorten the computation time. The force classification accuracy was over 95% with between-subject variations ranged from 3.58 to 1.25%. Leone et al. (2019) presented classification results for both hand or wrist gestures and forces. The algorithm was evaluated on 31 healthy participants for seven movements using commercial sEMG sensors, Ottobock 13E2000, providing six channels of input and a sampling rate of 1 kHz. The best average accuracy of 98.78% was achieved with non-linear logistic regression (NLR) algorithm. Olsson et al. (2019) experimented with a high-density sEMG (HD-sEMG) for the classification of 16 hand movements using CNN. HD-sEMG signal was collected using two of the eight-by-eight electrode arrays coated with conductive gel, for a total of 128 input channels. Fourteen healthy adults participated in this study. The input size for the CNN model was 16 × 8 × 24, 24 HD-sEMG samples. The classification accuracy was 78.13 ± 6.80% with individual subject accuracy ranged from 62 to 85%. Junior et al. (2020) investigated multiple classification techniques for six hand gestures acquired from 13 participants using eight channels sEMG armband with a sampling rate of 2 kHz. Their best result with the average accuracy of 94% was obtained from 40 features with the large margin nearest neighbor (LMNN) technique. Côté-Allard et al. (2020) presented an analysis of the features learned using deep learning for the classification of 11 hand gestures using sEMG. They concluded that handcrafted features and learned features could discriminate between the gestures but do not encode the same information. The learned features tend to ignore the most activated channel while the handcrafted features were designed to capture the amplitude information. The authors also presented an Adaptive Domain Adversarial Neural Network (ADANN) designed to study learned features that generalize well across participants. The dataset used in this study included 22 healthy participants performing ten hand and wrist gestures using the 3DC armband with ten channels at a 1 kHz sampling rate. The average accuracy was 84.43 ± 0.05% for the 10 movements. Krasoulis et al. (2017) and Pizzolato et al. (2017) performed an sEMG classification of over 40 hand and wrist movements. Krasoulis et al. (2017) reported average accuracy scores for 20 participants in the ablebodied group at 63%. For two participants in the amputee group, the average accuracy scores were 60%. Both experiments used linear discriminant analysis (LDA) classifier for movement intent decoding. Pizzolato et al. (2017) reported the best results with an accuracy of 74.01 ± 7.59% for the 41 movements in the group of 40 able-bodied participants using random forest classifier on hand-crafted features. Recent research on the sEMG classification using a deep learning approach tends to gravitate toward using CNN to automatically learn the features from a raw signal. However, training a deep neural network generally requires a large amount of training data for it to converge and discover meaningful features, especially for CNN. Moreover, CNN has a relatively high memory cost and processing time, which may pose challenges when running on embedded systems with limited resources. For our experiment, we were concerned about the limited amount of training data for the classification of 41 movements. Also, we would like to investigate the feasibility of adopting an accurate deep learning approach that would be able to run on affordable hardware. Therefore, we chose to extract hand-crafted statistical features and feed them to our deep neural network (DNN) model for the classification.

Even though the classification of hand and wrist movements using sEMG has been investigated and reported by multiple research teams, the classification results described in the literature can vary by a large margin, ranging from 60 to 98% accuracy. The results depend on several parameters, such as the number of classes, the number of samples, the acquisition protocol and setup, and the evaluation metrics. Hence, for qualitative comparison, the experimental results should consider only studies targeting a similar number of classes, where the chance levels are comparable. The objective of this study is to investigate the DNN approach for the classification of the hand and wrist movements based on the sEMG signal. The experiments considered 41 movements of Hand, wrist, grasping, and functional hand movements. Feature extraction techniques on the sEMG signal were explored and selected for the best balance between classification performance and computational complexity. The result of the proposed deep neural network classifier was validated on the publicly accessible datasets from the Ninapro database, one of the largest public sEMG databases for advanced hand myoelectric prosthetics. The Ninapro project is an ongoing work that aims to create an accurate and comprehensive reference for scientific research on the relationship between sEMG, hand or arm kinematics, and dynamics, and clinical parameters, with the final goal of creating non-invasive, naturally controlled robotic hand prostheses for transradial amputees (Atzori et al., 2014). The data used in Krasoulis et al. (2017) and Pizzolato et al. (2017) experiments were also collected according to the protocol described in the Ninapro project and the data were included in the Ninapro database. At the time of writing, the project consists of eight datasets with a predefined set of up to 53 movements. A summary of the database is shown in Table 1. In this study, Ninapro DB5 and DB7 were chosen since they are the two newest datasets with comparable data collection protocols for 41 movements. The acquisition setup for DB5 is based on two Thalmic Myo armbands with 16 sEMG input channels and a 200 Hz sampling rate, which cost a few hundred dollars compared to several thousand dollars for other setups. The acquisition setup for DB7 is based on 12 sEMG input channels of Delsys Trigno electrodes with a 2 kHz sampling rate. The performance of our proposed technique was compared with the best performance from the previous study on the dataset. The contributions of this study are as follows: (1) performance improvement for the classification of sEMG for 41 hand and wrist movements; (2) performance comparison between the sEMG setups of low cost and low sampling rate sensors—double



**CHAPTER 2**

**LITERATURE SURVEY**

**Ahmadizadeh, C., Merhi, L.-K., Pousett, B., Sangha, S., and Menon, C. (2017). Toward intuitive prosthetic control: Solving common issues using force myography, surface electromyography, and pattern recognition in a pilot case study. IEEE Robot. Autom. Mag. 24, 102–111. doi: 10.1109/MRA.2017.2747899**Despite the appearance of advanced multi-degrees of freedom (DoF) robotic hands during the past decade, prosthetic control lacks a powerful interface to facilitate all its functionalities in a manner that is acceptable for a majority of users. In this article, we explore the feasibility of using a sensing technique called force myography (FMG) as an alternative or synergist to the traditional surface electromyography (sEMG) technique as a human-machine interface (HMI) for the control of a multi-DoF prosthetic hand, bebionic 3 by Ottobock, Austin, Texas. In this article, we present a prosthetic prototype developed for the Cybathlon 2016, a championship for racing pilots with disabilities using assistive robotic devices. The design of the prototype is discussed and the effect of two factors on its control is analyzed. These factors are 1) the impact of a multisensory approach and 2) the placement of FMG sensor strips within the prosthetic inner socket. Analysis is performed by comparing resulting pattern recognition accuracies. Results show that the use of both sensing modalities (FMG and EMG) together produced the highest pattern recognition accuracy (81.1%) for ten classes of motion (four wrist movements and six grip patterns). We demonstrated that FMG has the potential to be an HMI for control of upper-limb-powered prostheses. FMG also illustrates the potential for intuitive control through the use of pattern recognition. A multisensory approach could assist in increasing robustness of the HMI for prosthetic control.

**Summary:** Studied about Solving common issues using force myography, surface electromyography, and pattern recognition in a pilot case.

**Ahmadizadeh, C., Pousett, B., and Menon, C. (2019).** **Investigation of channel selection for gesture classification for prosthesis control using force myography: a case study. Front. Bioeng. Biotechnol. 7:331. doi: 10.3389/fbioe.2019. 00331**

**Background:** Various human machine interfaces (HMIs) are used to control prostheses, such as robotic hands. One of the promising HMIs is Force Myography (FMG). Previous research has shown the potential for the use of high density FMG (HD-FMG) that can lead to higher accuracy of prosthesis control. **Motivation:** The more sensors used in an FMG controlled system, the more complicated and costlier the system becomes. This study proposes a design method that can produce powered prostheses with performance comparable to that of HD-FMG controlled systems using a fewer number of sensors. An HD-FMG apparatus would be used to collect information from the user only in the design phase. Channel selection would then be applied to the collected data to determine the number and location of sensors that are vital to performance of the device. This study assessed the use of multiple channel selection (CS) methods for this purpose

**Summary:** Studied about Investigation of channel selection for gesture classification for prosthesis control using force myography

**Ameri, A., Akhaee, M. A., Scheme, E., and Englehart, K. (2018). Real-time,** **simultaneous myoelectric control using a convolutional neural network. PLoS ONE 13:e203835. doi: 10.1371/journal.pone.0203835**

The evolution of deep learning techniques has been transformative as they have allowed complex mappings to be trained between control inputs and outputs without the need for feature engineering. In this work, a myoelectric control system based on convolutional neural networks (CNN) is proposed as a possible alternative to traditional approaches that rely on specifically designed features. This CNN-based system is validated using a real-time Fitts’ law style target acquisition test requiring single and combined wrist motions. The performance of the proposed system is then compared to that of a standard support vector machine (SVM) based myoelectric system using a set of time-domain features. Despite the prevalence and demonstrated performance of these well-known features, no significant difference (p>0.05) was found between the two methods for any of the computed control metrics. This demonstrates the potential for automated learning approaches to extract complex and rich information from stochastic biological signals. This first evaluation of the usability of a CNN in a real-time myoelectric control environment provides a basis for further exploration.

**Summary:** Studied about simultaneous myoelectric control using a convolutional neural network

**Atzori, M., Gijsberts, A., Castellini, C., Caputo, B., Mittaz Hager, A.-G., Elsig, S., et al. (2014).** **Electromyography data for non-invasive naturally-controlled robotic hand prostheses. Nature 1:140053. doi: 10.1038/sdata.2014.53**

Intelligent reflecting surface (IRS) is a revolutionary and transformative technology for achieving spectrum and energy efficient wireless communication cost-effectively in the future. Specifically, an IRS consists of a large number of low-cost passive elements each being able to reflect the incident signal independently with an adjustable phase shift so as to collaboratively achieve three-dimensional (3D) passive beamforming without the need of any transmit radio-frequency (RF) chains. In this paper, we study an IRS-aided single-cell wireless system where one IRS is deployed to assist in the communications between a multi-antenna access point (AP) and multiple single-antenna users. We formulate and solve new problems to minimize the total transmit power at the AP by jointly optimizing the transmit beamforming by active antenna array at the AP and reflect beamforming by passive phase shifters at the IRS, subject to users' individual signal-to-interference-plus-noise ratio (SINR) constraints. Moreover, we analyze the asymptotic performance of IRS's passive beamforming with infinitely large number of reflecting elements and compare it to that of the traditional active beamforming/relaying. Simulation results demonstrate that an IRS-aided MIMO system can achieve the same rate performance as a benchmark massive MIMO system without using IRS, but with significantly reduced active antennas/RF chains. We also draw useful insights into optimally deploying IRS in future wireless systems.

**Summary:** Studied about Electromyography data for non-invasive naturally-controlled robotic hand prostheses. Nature

**Atzori, M., and Müller, H. (2015). Control capabilities of myoelectric robotic prostheses by hand amputees: a scientific research and market overview. Front. Syst. Neurosci. 9:162. doi: 10.3389/fnsys.2015.00162**

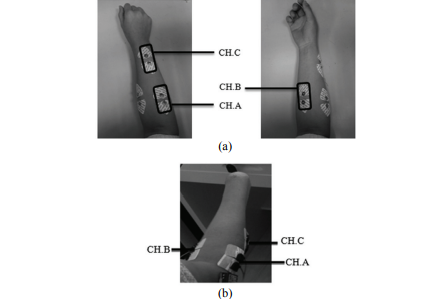
Hand amputation can dramatically affect the capabilities of a person. Cortical reorganization occurs in the brain, but the motor and somatosensorial cortex can interact with the remnant muscles of the missing hand even many years after the amputation, leading to the possibility to restore the capabilities of hand amputees through myoelectric prostheses. Myoelectric hand prostheses with many degrees of freedom are commercially available and recent advances in rehabilitation robotics suggest that their natural control can be performed in real life. The first commercial products exploiting pattern recognition to recognize the movements have recently been released, however the most common control systems are still usually unnatural and must be learned through long training. Dexterous and naturally controlled robotic prostheses can become reality in the everyday life of amputees but the path still requires many steps. This mini-review aims to improve the situation by giving an overview of the advancements in the commercial and scientific domains in order to outline the current and future chances in this field and to foster the integration between market and scientific research.

**SUMMARY:** Studied about Control capabilities of myoelectric robotic prostheses by hand amputees.

**CHAPTER 3**

**EXISTING METHOD**

The electromyogram (EMG) signal is a bioelectrical signal variation generated during the muscle contraction process. And the EMG signal is a complicated biomedical signal influenced by anatomical or physiological properties of the muscles and the environmental noises [12]. Thereby, the collection of EMG signal is a fundamental task that influences the next identification of hand motion commands. Generally, EMG signals can be detected by invasive or noninvasive electrodes. The EMG signals collected by invasive needles or wire electrodes are suitable to investigate deep muscle structure. However, due to their noninvasiveness and convenient operation, sEMG techniques are mainly used to detect sEMG signals in the fields of biofeedback, prosthetic hand control, movement analysis, etc. Hence, we used sEMG as original signals to control the prosthetic hand. Since we aim to recognize more hand motion commands only based on three sEMG sensors, the sensor position is crucial to the identification accuracy. Considering the kinematic physiological characteristics of human hand and the corresponding forearm muscles, we chose four commonly used hand motions at first, namely hand closing, hand opening, wrist extension, and wrist flexion. Then we need to determine the corresponding forearm muscles. Although there are deep layer and superficial layer muscles that contribute to the above hand motions, the sEMG signals are mainly influenced by the superficial layer muscles. Therefore, we selected three superficial layer muscles to fix the sEMG sensors (refer to Fig. 1). Extensor carpi radialis longus, which is on the posterior side of the forearm, acts to extend and abduct the hand at the wrist. Flexor carpi radialis assists in the wrist flexion and the wrist abduction. Extensor digitorum acts to extend the phalanges and the wrist, and it tends to separate the fingers as it extends the phalanges. In this paper, the sEMG sensors were placed on the extensor carpi radialis longus (CH.A), the flexor carpi radialis (CH.B), and the extensordigitorum (CH.C) of the subject’s forearm, respectively (refer to Fig. 1). The acquired sEMG signal is the superimposition of EMG signals from the muscles below and around the sEMG sensor. The extensor carpi radialis longus is in the upper level of the supinator, and the flexor carpi radialis is near the pronator teres. Moreover, supinator and pronator teres serve to forearm supination and forearm pronation, respectively. Consequently, we can choose forearm pronation and forearm supination as two additional hand motions without increasing the number of sEMG sensors. Totally, six hand motions can be classified by three sEMG sensors, namely hand closing, hand opening, wrist extension, wrist flexion, forearm pronation, and forearm supination (as shown in Fig. 2)

****

In order to classify more hand motions with fewer sEMG sensors, the sensor position is crucial to the identification accuracy. Therefore, we calculated RMS of sEMG signals acquired from CH. A to CH. C. As illustrated in Fig. 3, the calculated RMS values of sEMG signals detected from CH. A, CH. B, and CH. C are listed from top to bottom in turn. The identified hand motions, such as hand closing, hand opening, wrist extension, wrist flexion, forearm pronation, and forearm supination are listed from left to right. Each hand motion continues about 5 s, and the interval between two adjacent hand motions is about 10 s. According to Fig. 3, when the subject executes different hand motions, the RMS values of three sensors are different. For example, when the hand opening is executed, the RMS value of CH. C is the largest among three sensors. In the case of wrist extension, the RMS value of CH. A changes greatly. While in the case of wrist flexion, the RMS value of CH. B is more than that of the other two sensors. According to the obvious changes of RMS values, it can be concluded that the determined sensor positions contribute to the identification of the selected six hand motions. To reduce the influence of skin (e.g., impedance, superficial oil content, and dead cell layer), we wiped the skin of desired locations on the forearm with alcohol. Moreover, two channels of differential surface Ag-AgCl electrodes with 20 mm inter-electrode distance were utilized to collect sEMG signals after the skin dried. The sEMG signals were recorded with a sampled rate of 1,500 Hz using TeleMyob2400 G2 (Noraxon, USA). Six healthy subjects participated in the experiment, namely four women who were 23, 23, 25 and 26 years old, and two men who were 23 and 36 years old, respectively. Some rehabilitation experts have suggested that one should use healthy subjects for initial evaluation objectives [14, 40]. Thereby, the sEMG of healthy subjects is an appropriate emulation of the amputee’s sEMG. To further demonstrate the practicability and the validity of the proposed identification method, two amputees were required to execute the same experiment (refer to Fig. 1). All of the subjects performed above six hand movements. For each subject, we selected 100 sEMG signals of one hand movement. Since there are six hand movements that need to be classified, a total of 600 sEMG signals were collected for each subject. In order to control the prosthetic hand in real time, the response time should be less than 300 ms. Therefore, 200 ms was selected as the appropriate time window length of the sEMG signal [13]. There are 300 points for one signal. Since sEMG signals are different from one subject to another, we divided the acquired sEMG signals into two sets for each subject. One is the training set, the other is the test set, and each set has 300 sEMG signals.

After collecting sEMG signals, we need to extract the feature vectors of sEMG signals. Taking the advantages of both time domain analysis and frequency domain analysis, the WT, as a time-frequency domain analysis, was utilized in this paper. A. Wavelet Transform A wavelet , ( ) a b ψ t is derived from its mother wavelet ψ ( )γ by dilating and time-shifting. This relation can be defined as:



where a and b represent the scaling factor and the translation factor, respectively. The family of functions generated byψ ( )γ can be defined as: , { ( ), , , 0} a b Ω = ∈ℜ ∈ℜ ≠ ψ ta b a (2) Ω is called a frame of 2 L ( ) ℜ if it satisfies the admissibility condition Cψ



where ψ ( ) ω is generated from ψ ( )t by Fourier transform. There is a function 2 ft L () ( ) ∈ ℜ , and its continuous WT is defined as the inner product of functions f ( )t and , ( ) a b ψ t :

where \*( ) t b a ψ − is the conjugate function of ( ) t b a ψ − . The discrete version of WT of a discrete signal f ( ) n is defined as:



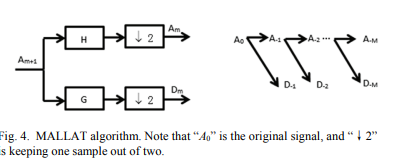
where 0 00 = = , j j a a b kb a ; j, k, and n are integers; and j is the depth of decomposition [52]. The time-frequency analysis based on the DWT is well suited to non-stationary signals whose frequency spectrum is time-variant. Some approaches have appeared in the literature that used DWT in handling EMG [52, 53]. Compared with Hudgins’ time domain approach, a wavelet-based approach exhibits superior performance [14]. To avoid complicated computations, in (5), we set 0 a = 2 and 0 b =1. Then, we have a dyadic sampling in time axes, and the following dyadic wavelet can be obtained:



where j, k and n are integers. For most signals, the analytical solution cannot be resolved by DWT, so it can only be obtained by numerical algorithm. Therefore, it requires a fast calculation method to make DWT be applied in practice. Stephane G. Mallat proposed MALLAT algorithm to realize the multiresolution signal decomposition rapidly [54, 55], which is an important part in WT. Hence, we used the MALLAT algorithm (described in Fig. 4) for DWT to process sEMG signals. The original signal A0 is decomposed into approximations A-1 and details D-1, then A-1 is decomposed into approximations A-2 and details D-2, and so on, until a predetermined level. This relationship can be expressed as:



where m=-1, -2, …, -M. In Fig. 4, G is the high-pass filter, and H is the low-pass filter. The impulse response g(k) of filter G is related to the impulse response h(k) of the filter H [54]



B. Features Selection The approximate part mainly consists of low frequency signals; whereas, the details mainly consist of high frequency signals. The value of coefficients represents the fitting extent of the signals, and higher values indicating greater similarity. Therefore, we can choose the maximum absolute value of the coefficients as the feature vector of sEMG signals. One method is to select the maximum absolute value of approximate coefficients of every level, marked as AAA; the other method is to choose the maximum absolute value of approximate coefficients of the last level and the maximum absolute value of detail coefficients of every level, marked as ADDD. The decomposition level is represented by n. Because the sEMG signals were detected from three sensors, there is a 3n-dimensional feature vector of AAA and a 3(n+1)-dimensional feature vector of ADDD. Considering the computational complexity, we cannot decompose sEMG signals into many levels. Hence, the obtained sEMG signals were decomposed by DWT level-3 decomposition. For example, the level-3 decomposition process of sEMG signals detected from CH.C is illustrated in Fig. 5. The feature vector selection influences the classification results significantly [16], and the suitable feature vector should be selected by using DWT to deal with sEMG signals. Therefore, the contrast experiment was done to choose the better feature vector from AAA and ADDD. In this experiment, we collected three healthy subjects’ sEMG signals. Each subject performed the proposed six hand movements, and the results are shown in Fig. 6. All three subjects’ average classification accuracy rates of ADDD are above 90%, and the average classification accuracy rate of ADDD is much higher than that of AAA. Consequently, a 12-dimensional feature vector of ADDD was selected as the input of the WNN C. The Best Mother Wavelet Selection Different wavelet functions have different time-frequency characteristics and affect the performance of WT, so it is important to select an appropriate wavelet function for the wavelet analysis. To construct a feature set through DWT，we need to determine the type of mother wavelet. On the basis of the feature selection results, we used the feature vector of ADDD to investigate the classification accuracy rates of eleven different dominant mother wavelets, such as Haar, biorthogonal, coiflet, Daubechies, and symlet with different orders. Above five types of mother wavelets all have biorthogonality and compact support, and they are commonly used in DWT. In the feature extraction part, the feature should contain enough information to maintain the classification accuracy of the hand motion commands. Hence, we determined the best mother wavelet according to the classification accuracy rate. In this paper, eleven different mother wavelets are marked as “bior1.5”, “bior3.5”, “bior3.9”, “coif3”, “coif5”, “db2”, “db9”, “haar”, “sym3”, “sym5”, and “sym7”. Fig. 7 depicts the average accuracy rates of three subjects and the standard deviation of each type of mother wavelet. In Fig. 7, “coif5”, with the maximum classification accuracy rate and smaller standard deviation, is selected as the best mother wavelet for sEMG feature extraction. In this paper, we chose the wavelet coefficients as feature vector of sEMG signals, and the wavelet coefficients reflect the degree of similarity between the wavelet function and the decomposed sEMG signals. The larger the wavelet coefficient is, the higher the degree of similarity is. Thus, choosing the mother wavelet with higher degree of similarity is beneficial to the analysis of sEMG signals. The original sEMG signals (refer to Fig. 5) are non-stationary and non-decaying, so the mother wavelets with decay (as “bior3.9”, “db9”, “sym7”) are not appropriate, according to the mother wavelets’ waveforms in Fig. 8. Moreover, the Haar wavelet is discontinuous in time domain, so it is also not appropriate. The coiflet wavelets’ waveforms are similar to the sEMG signals, so the average accuracy rate of six hand motions using “coif5” is high. The coiflet wavelet with L orders is the compactly supported orthogonal wavelet which satisfies the following two vanishing moment conditions in (10) and (11) [56]. Equation (10) means that the vanishing moment of the scaling function φ( )t equals to 0. Equation (11) means that the vanishing moment of the mother wavelet ψ ( )t equals to 0 equal to 0 and a support of length 6L-1. For a given support width, the mother wavelet “coif L” is a compactly supported wavelet with the highest number of vanishing moments for both mother wavelet (psi) and scaling function (phi) [57]. In this paper, L is set as 5 .

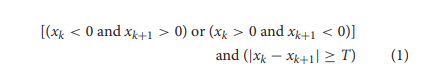
**CHAPTER 4**

**PROPOSED METHOD**

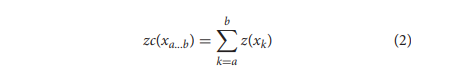
2.1. Database and Acquisition Setup The database of the Ninapro project was used in this study. Ninapro DB5 and DB7, two of their newest datasets acquired using the same data acquisition protocols, were selected for comparison. For the data acquisition protocol, participants were instructed to repeat several hand movements by following videos shown on a laptop screen. The recording of each movement took 5 s, with 3 s of rest to avoid errors from muscular fatigue. For every hand movement recording, participants performed six repetitions, to account for slight variations of the exact hand muscle movements within the same movement class. DB5 has a total of 53 movements while DB7 has 41 movements. The same movements that were collected in both datasets are from two movement groups: isometric and isotonic hand configurations and basic wrist movements (17 exercises), and grasping and functional movements (23 exercises), as illustrated in Figure 2. According to the Ninapro project, all of the movements were selected from the hand taxonomy as well as from hand robotics literature. Following previous studies on the Ninapro database, we used repetitions 1, 3, 4, and 6 as training data, while repetition 2 and 5 were used for evaluation (Atzori et al., 2014; Atzori and Müller, 2015; Pizzolato et al., 2017). For DB5, the low cost and low sampling rate dataset, the sEMG was recorded with two Thalmic Myo armbands. Each Myo armband has eight sEMG electrodes with a sampling rate of 200 Hz. The upper armband is placed closer to the elbow with the first electrode on the radio humeral joint. The lower armband is tilted by 22.5◦ and placed next to the upper one to fill in the gaps between its electrodes. This setting provides an extended uniform muscle mapping at the most affordable cost. The participants in this dataset are 10 intact participants, eight males and two females, with an average age of 28.00 ± 3.97 years. On the other hand, the high cost and high sampling rate DB7 dataset used 12 Delsys Trigno electrodes for sEMG recording with a sampling rate of 2 kHz. Eight sensors were placed around three centimeters below the elbow and equally spaced around the forearm. Two sensors were placed for the extrinsic hand muscles; Extensor Digitorum Communis (EDC) and Flexor Digitorum Superficialis (FDS). The last two sensors were placed on the biceps and triceps brachii muscles. The dataset contains 20 intact participants and 2 amputee participants, with an average age of 27.73 ± 6.53 years. The first amputee participant was a transradial 28 years old male with 6 years of right limb loss due to a car accident. The second amputee participant was a transradial 54 years old male with 18 years of right limb loss due to epithelioid sarcoma cancer. Figure 3 shows the relationship between the amplitude of the sEMG signal and different experimental conditions, namely movement repetition, movement class, and subject. The data from the first channel of each dataset is illustrated and the outliers are omitted for readability purposes. Even though the subjects performed each repetition under the same environment, the signals may differ between repetitions due to physiological factors such as muscular characteristics, skin impedance, sweat, muscular tone, or fatigue. To statistically validate the differences in each repetition, we conducted a one-way analysis of variance (ANOVA) statistical test on the concatenation of all signal channels in each repetition. Our analysis showed that there are no significant differences between different movement repetitions

**2.2. Data Preprocessing and Feature Extraction**

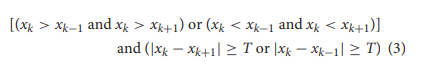
To process real-time sEMG data, the raw signals were sectioned using a sliding window. To introduce time variation as well as add more samples, the stride between each window was set to be smaller than the window size, resulting in some overlap between consecutive samples. We extracted from each window the following features: the root mean square (RMS), and time-domain statistics as described by Hudgins et al. (1993); mean absolute value, mean absolute value slope, zero crossings, slope sign changes, and waveform length. Each feature was standardized into a normal distribution to make sure no feature is favored unequally over the others due to scale or range. Out of all the time domain features, zero crossings and slope sign changes were noted to require a noise threshold. Due to being features based on counting occurrences of, for example, the values crossing zero, one must exclude any occurrences caused by low-valued noise. We restate these features more formally as follows. Given a window of data xa...b , the zero crossing function z(xk ) is equal to 1 when:

****

where T is the noise threshold, and 0 otherwise. The zero crossing feature zc(xa...b ) itself is an accumulation, or a summation of said function:



The slope sign change feature is defined similarly. The condition for the slope sign change function s(xk ) being equal to 1 is:

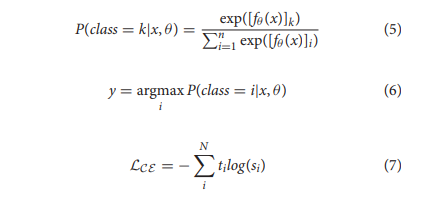


and the slope sign change feature ssc(xa...b ):



**2.3. Deep Neural Network Classifier**

A deep neural network (DNN) has been chosen for dealing with real-world signal processing tasks, due to its outstanding performance compared to other machine learning algorithms (Park and Lee, 2016; Chen et al., 2017; Orjuela-Cañón et al., 2017; Tsinganos et al., 2018; Chaiyaroj et al., 2019). Motivated by this fact and considering our aim for a real-time system, we implemented a simple feed-forward neural network model. The model consists of three hidden layers, which are fully connected layers with 512, 256, and 256 neurons, respectively. All layers were initially assigned random weights using the He uniform initialization scheme (He et al., 2015). Each layer is followed by the rectified linear unit (ReLU) activation function, to address the vanishing gradient problem (Nair and Hinton, 2010; Dahl et al., 2013; Nwankpa et al., 2018). For regularization, we applied batch normalization to increase the numerical stability of the neural network, and 20% dropout to prevent over-fitting by forcing the model not to rely on the same patterns all the time (Srivastava et al., 2014; Ioffe and Szegedy, 2015). The output layer uses the softmax activation function to simulate a probability vector, as our task is a multi-class classification. The model was optimized with the Adam optimizer, with a learning rate of 0.005 and decay of 0.00001. Our proposed model is illustrated in Figure 5. In Equation (5), given a vector of preprocessed signal input x and trained weights θ, [fθ (x)]k is an output value obtained from passing the input through our DNN model. This output value can be described as a score it assigns to whether the input belongs to each class k. To derive a probability vector from the output values of all classes, we added the softmax function in our last output layer. After that, according to Equation (6), the class with the highest probability is selected as the final output



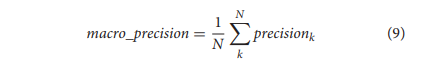
where ti is either 1 in case the sample’s ground truth label is i, or otherwise 0, while si is the probability score of the sample about which class the model predicts it to belong to.

**2.4. Evaluation Metrics**

According to the data acquisition protocol from Ninapro, data for the rest class was collected after every hand movement exercise to avoid errors from muscular fatigue due to that particular exercise. Therefore, with approximately half of the samples belonging to the rest class, robust and efficient evaluation metrics are necessary to deal with the imbalanced data Otherwise, the result will not reflect the real performance of the model; the model might perform well just because it outputs only the majority class. For binary classification tasks, a distinction is often made between overall accuracy, and balanced accuracy. Overall accuracy, often simply referred to as accuracy, is one of the most commonly used metrics, reflecting the number of all correctly identified samples out of all samples. However, this metric does not distinguish samples between classes; thus, it may not show the true performance of the classifier when a class imbalance is present in the data. On the other hand, balanced accuracy, also known as the Balanced Classification Rate (BCR) (Hardison et al., 2008; Brodersen et al., 2010; Tharwat, 2018), can mitigate the imbalance’s effect by normalizing the accuracy of each class with the number of samples of the class. In the case of multi-class classification, taking the average of recall values can be generalized as the macro recall:



where recallk is the percentage of total relevant results correctly classified by our algorithm for class k, and N is the number of classes. While balanced accuracy is not defined for multiclass classification, we may refer to macro recall as such due to the similarity and to be more in line with other studies. Since macro recall can represent a classifier’s performance on each class equally, we have included it along with accuracy as metrics by which the classifiers will be evaluated. To facilitate any comparisons in further studies, we have also included other macro-averaged metrics: macro precision, and macro F1 score. Macro precision is defined as:



where precisionk is the percentage of predictions that come from class k. Macro F1 score is simply the harmonic mean of precision and macro recall:



**2.5. Usage Simulation**

Since our experiments used public datasets from the Ninapro project, we currently do not have a similar acquisition setup for online testing. Therefore, we examine how our system would perform in a real use case using the full sequence from each repetition to simulate usage. Adhering to Ninapro’s sEMG data acquisition procedure, we tested our model on the entire lengths of the test repetitions from each intact subject. This procedure illustrates whether the model’s predictions will translate into smooth and uninterrupted hand movements. Ideally, the predictions on this dataset should be a period of rest, followed by a continuous sequence of one of the specified hand movements. Any wrong classification in the middle of the sequence indicates that our system will execute an incorrect movement and interrupt the user’s intended movement

**CHAPTER 5**

**ADVANTAGES AND APPLICATIONS**

**Advantages:**

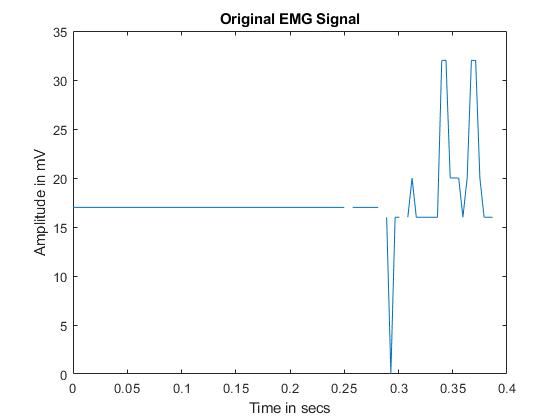
* Proposed CNN based classifier is very good at classification.
* Proposed classifier takes less time for training.
* Proposed classifier is very accurate compared to existing classifiers.
* Feature extraction is includes even the minute details which results in more accurate

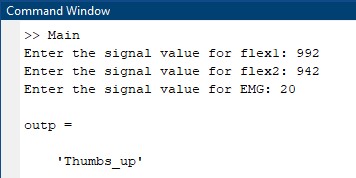
**Applications:**

* Bio-Medical applications
* Signal Processing
* Machine Learning environments
* Image Processing

**CHAPTER 6**

**RESULTS**





**CHAPTER 7**

**CONCLUSION**

This study presents an application of a deep neural network model for classifying 41 hand movements based on surface electromyogram. The public datasets Ninapro DB5 and DB7 were used as low sampling rate data and high sampling rate data for our experiment. The acquisition setup for DB5 was based on two Thalmic Myo armbands with 16 channels of input and 200 Hz sampling rate, while DB7 was recorded by Delsys Trigno electrodes with 12 channels of input and 2 kHz sampling rate. Following the Southhampton Hand Assessment Procedure (SHAP), we also performed experiments for the classification of the six movements based on six prehensile patterns for hand functionality evaluation. Compared to other studies’ classification results, our proposed model outperformed the best results of the previous studies from Pizzolato et al. (2017) and Krasoulis et al. (2017). This is a promising result, though some confirmation from a larger experiment with more data samples would certainly be beneficial. Experimentation on the window size shows that the larger the window size is, the higher the performance gain the proposed model achieves, which is expected. Lastly, we measured the running time of our proposed model to compare the feasibility of using different window sizes. We believe that given sufficient data, our proposal could be a feasible approach for controlling advanced prosthetic hands.

**REFERENCES**

[1] Q. Wu and R. Zhang, “Towards smart and reconfigurable environment: Intelligent reflecting surface aided wireless network,” IEEE Commun. Mag., pp. 1-7, 2019.

[2] E. Basar, M. Di Renzo, J. De Rosny, M. Debbah, M.-S. Alouini and R. Zhang, “Wireless communications through reconfigurable intelligent surfaces,” IEEE Access, vol. 7, pp. 116753-116773, 2019.

[3] M. Di Renzo, et al., “Smart radio environments empowered by reconfigurable AI meta-surfaces: An idea whose time has come,” EURASIP J. Wireless Commun. Netw., vol. 2019, p. 129, May 2019.

[4] Q. Wu and R. Zhang, “Intelligent reflecting surface enhanced wireless network via joint active and passive beamforming,” IEEE Trans. Wireless Commun., vol. 18, no. 11, pp. 5394-5409, Nov. 2019.

[5] P. Wang, J. Fang, X. Yuan, Z. Chen, H. Duan, and H. Li, “Intelligent reflecting surface-assisted millimeter wave communications: Joint active and passive precoding design,” [Online]. Available: https://arxiv.org/abs/1908.10734

[6] C. Pradhan, A. Li, L. Song, B. Vucetic, and Y. Li, “Hybrid precoding design for reconfigurable intelligent surface aided mmWave communication systems,” [Online]. Available: <https://arxiv.org/abs/1912.00040>

[7] D. Mishra and H. Johansson, “Channel estimation and low-complexity beamforming design for passive intelligent surface assisted MISO wireless energy transfer,” IEEE ICASSP, May 2019, pp. 46594663.

[8] Z. He and X. Yuan, “Cascaded channel estimation for large intelligent metasurface assisted massive MIMO,” IEEE Wireless Commun. Lett., vol. 9, no. 2, pp. 210-214, Feb. 2020.

[9] P. Wang, J. Fang, H. Duan and H. Li, “Compressed channel estimation for intelligent reflecting surface-assisted millimeter wave systems,” in IEEE Signal Processing Lett., May. 2020.

[10] B. Zheng and R. Zhang, “Intelligent reflecting surface-enhanced OFDM: Channel estimation and reflection optimization,” IEEE Wireless Commun. Lett., Dec. 2019.

[11] A. Taha, M. Alrabeiah, and A. Alkhateeb, “Deep learning for large intelligent surfaces in millimeter wave and massive MIMO systems,” 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6.

**BIBLIOGRAPHY**

**Introduction To Matlab**

What Is MATLAB?

The name MATLAB stands for Matrix Laboratory. The software is built up around vectors and matrices. This makes the software particularly useful for linear algebra but MATLAB is also a great tool for solving algebraic and differential equations and for numerical integration. MATLAB has powerful graphic tools and can produce nice pictures in both 2D and 3D. It is also a programming language, and is one of the easiest programming languages for writing mathematical programs. These factors make MATLAB an excellent tool for teaching and research.

MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems.

MATLAB abilities a family of add-on software program utility software application software program software utility software-unique solutions called toolboxes. Very essential to maximum customers of MATLAB, toolboxes assist you to studies and observe specialized technology. Toolboxes are entire collections of MATLAB abilities (M-files) that increase the MATLAB surroundings to remedy precise schooling of problems. Areas in which toolboxes are to be had embody signal processing, manipulate systems, neural networks, fuzzy correct judgment, wavelets, simulation, and hundreds of others.

It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

**Brief History of MATLAB:**

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s. The first MATLAB® was not a programming language; it was a simple interactive matrix calculator. There were no programs, no toolboxes, no graphics and no ODEs or FFTs. He designed it to give his student’s access to LINPACK and EISPACK without them having to learn FORTRAN. It soon spread to other universities and found a strong audience within the applied mathematics community. The mathematical basis for the first version of MATLAB was a series of research papers by J. H. Wilkinson and 18 of his colleagues, published between 1965 and 1970 and later collected in Handbook for Automatic Computation, Volume II, Linear Algebra*,* edited by Wilkinson and C. Reinsch. These papers present algorithms, implemented in Algol 60, for solving matrix linear equation and Eigen value problems.

In the 1970s and early 1980s, I was teaching Linear Algebra and Numerical Analysis at the University of New Mexico and wanted my students to have easy access to LINPACK and EISPACK without writing FORTRAN programs. By “easy access,” I meant not going through the remote batch processing and the repeated edit-compile-link-load-execute process that was ordinarily required on the campus central mainframe computer. Jack little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded Math Works in 1984 to continue its development. These rewritten libraries were known as JACKPAC. In 2000, MATLAB was rewritten to use a newer set of libraries for matrix manipulation, LAPACK. MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved in video processing**.**

## **EISPACK and LINPACK**:

In 1970, a group of researchers at Argonne National Laboratory proposed to the U.S. National Science Foundation (NSF) to “explore the methodology, costs, and resources required to produce, test, and disseminate high-quality mathematical software and to test, certify, disseminate, and support packages of mathematical software in certain problem areas.” The group developed EISPACK (Matrix Eigen system Package) by translating the Algol procedures for Eigen value problems in the handbook into FORTRAN and working extensively on testing and portability. The first version of EISPACK was released in 1971 and the second in 1976.

In 1975, four of us Jack Dongarra, Pete Stewart, Jim Bunch, and myself proposed to the NSF another research project that would investigate methods for the development of mathematical software. A byproduct would be the software itself, dubbed LINPACK, for Linear Equation Package. This project was also centered at Argonne. LINPACK originated in FORTRAN; it did not involve translation from Algol. The package contained 44 subroutines in each of four numeric precisions. In a sense, the LINPACK and EISPACK projects were failures. We had proposed research projects to the NSF to “explore the methodology, costs, and resources required to produce, test, and disseminate high-quality mathematical software.” We never wrote a report or paper addressing those objectives. We only produced software.

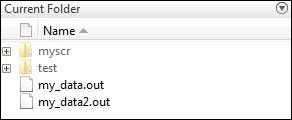
So, I studied Niklaus Wirth’s book Algorithms + Data Structures *=* Programs and learned how to parse programming languages. I wrote the first MATLAB an acronym for Matrix Laboratory in FORTRAN, with matrix as the only data type. The project was a kind of hobby, a new aspect of programming for me to learn and something for my students to use. There was never any formal outside support, and certainly no business plan. This first MATLAB was just an interactive matrix calculator. This snapshot of the start-up screen shows all the reserved words and functions. There are only 71. To add another function, you had to get the source code from me, write a FORTRAN subroutine, add your function name to the parse table, and recompile MATLAB.

**Starting MATLAB:**

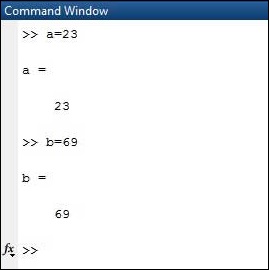
After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon (MATLAB 7.0.4) on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

* The Command Window
* The Command History
* The Workspace
* The Current Directory
* The Help Browser

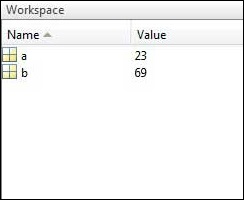
**Current Folder:** This panel allows you to access the project folders and files.



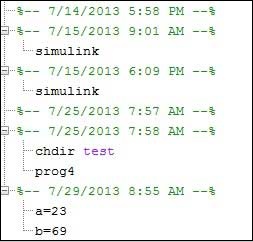
**Command Window:** This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).



**Workspace:**  The workspace shows all the variables created and/or imported from files.



**Command History:** This panel shows or return commands that are entered at the command line.



**Help Browser:**

The critical way to get assist online is to use the MATLAB help browser, opened as a separate window every through clicking at the question mark photograph (?) on the computing tool toolbar, or through manner of typing assist browser on the spark off in the command window. The assist Browser is an internet browser blanketed into the MATLAB computing tool that shows a Hypertext Markup Language (HTML) files. The Help Browser consists of panes, the help navigator pane, used to find out information, and the show pane, used to view the information. Self-explanatory tabs apart from navigator pane are used to performs are searching out.

**MATLAB language:**

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

**MATLAB working environment:**

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

**MATLAB mathematical function library:**

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

**MATLAB Application Program Interface (API):**

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

**MATLAB DESKTOP:**

MATLAB Desktop is the precept MATLAB utility window. The computing tool includes five sub home windows, the command window, the workspace browser, the modern-day-day list window, the command records window, and one or greater decide domestic windows, which is probably confirmed high-quality on the identical time due to the truth the client suggests a photo. The command window is in which the character types MATLAB instructions and expressions at the spark off (>>) and in which the output of these commands is displayed. MATLAB defines the workspace because the set of variables that the client creates in a bit consultation. The workspace browser suggests those variables and some facts about them. Double clicking on a variable within the workspace browser launches the Array Editor, which may be used to gain statistics and profits instances edit exceptional homes of the variable.

The modern-day-day-day Directory tab above the workspace tab suggests the contents of the cutting-edge list, whose path is shown inside the modern-day list window. For example, in the home windows on foot machine the path is probably as follows: C: MATLAB Work, indicating that listing “artwork” is a subdirectory of the number one list “MATLAB”; WHICH IS INSTALLED IN DRIVE C. Clicking on the arrow within the modern list window suggests a listing of these days used paths. Clicking at the button to the right of the window permits the individual to trade the present day listing. MATLAB uses a seeking out path to find out M-documents and one-of-a-type MATLAB associated documents, which can be put together in directories within the computer document tool. Any report run in MATLAB need to be dwelling in the modern-day-day listing or in a list that is on is looking for course. By default, the documents supplied with MATLAB and math works toolboxes are included inside the searching out direction. The first-rate manner to look which directories are on the searching out route. The satisfactory manner to appearance which directories are speedy the quest route, or to characteristic or regulate a searching for course, is to pick out outset path from the File menu the computing device, and then use the set course talk discipline. It is proper exercise to feature any generally used directories to the hunt route to avoid again and again having the exchange the cutting-edge-day listing.

The Command History Window contains a file of the instructions a person has entered in the command window, together with every contemporary-day and former MATLAB periods. Previously entered MATLAB instructions can be determined on and re-completed from the command statistics window thru proper clicking on a command or series of commands. This movement launches a menu from which to select numerous options similarly to executing the commands. This is useful to select out abilities options in addition to executing the instructions. This is a beneficial feature at the equal time as experimenting with numerous commands in a piece session.

**Using the MATLAB Editor to create M-Files:**

The MATLAB editorial manager is a literary substance proofreader particular for growing M-facts and a graphical MATLAB debugger. The supervisor can seem in a window through command facts technique for itself, or it is probably a right-clicking inside the PC. M-information this gadget signified through the use of the expansion .M, as in pixel up.M. The MATLAB editorial supervisor window has a few draws down menus for obligations collectively with sparing, seeing, and troubleshooting facts. Since it plays more than one easy test and furthermore affects utilization of shade to separate among exclusive variables of code, this article editorial supervisor is often supported due to reality the system of a need for composing and altering M-talents. To open the manager, type at enact opens the M-document filename. M in a supervisor window, sorted out for enhancing. As stated earlier than, the file should be inside the cutting-edge posting, or in a posting in the seeking out direction.

## **Features of MATLAB:**

Following are the basic features of MATLAB.

* It is a high-level language for numerical computation, visualization and application development.
* It also provides an interactive environment for iterative exploration, design and problem solving.
* It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
* It provides built-in graphics for visualizing data and tools for creating custom plots.
* MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.
* It provides tools for building applications with custom graphical interfaces.
* It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

## **Uses of MATLAB:**

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including

* Signal Processing and Communications
* Video and Video Processing
* Control Systems
* Test and Measurement
* Computational Finance
* Computational Biology

**Applications of MATLAB:**

MATLAB can be used as a tool for simulating various electrical networks but the recent developments in MATLAB make it a very competitive tool for Artificial Intelligence, Robotics, Video processing, Wireless communication, Machine learning, Data analytics and whatnot. Though it’s mostly used by circuit branches and mechanical in the engineering domain to solve a basic set of problems its application is vast. It is a tool that enables computation, programming and graphically visualizing the results. The basic data element of MATLAB as the name suggests is the Matrix or an array. MATLAB toolboxes are professionally built and enable you to turn your imaginations into reality. MATLAB programming is quite similar to C programming and just requires a little brush up of your basic programming skills to start working with.

Below are a few applications of MATLAB –

* **Statistics and machine learning (ML)**

This toolbox in MATLAB can be very handy for the programmers. Statistical methods such as descriptive or inferential can be easily implemented. So is the case with machine learning. Various models can be employed to solve modern-day problems. The algorithms used can also be used for big data applications.

* **Curve fitting**

The curve fitting toolbox helps to analyze the pattern of occurrence of data. After a particular trend which can be a curve or surface is obtained, its future trends can be predicted. Further plotting, calculating integrals, derivatives, interpolation, etc. can be done.

* **Control systems**

Systems nature can be obtained. Factors such as closed-loop, open-loop, its controllability and observability, Bode plot, NY Quist plot, etc. can be obtained. Various controlling techniques such as PD, PI and PID can be visualized. Analysis can be done in the time domain or frequency domain.

* **Signal Processing**

Signals and systems and digital signal processing are taught in various engineering streams. But MATLAB provides the opportunity for proper visualization of this. Various transforms such as Laplace, Z, etc. can be done on any given signal. Theorems can be validated. Analysis can be done in the time domain or frequency domain. There are multiple built-in functions that can be used.

* **Mapping**  
  Mapping has multiple applications in various domains. For example, in Big Data, the Map Reduce tool is quite important which has multiple applications in the real world. Theft analysis or financial fraud detection, regression models, contingency analysis, predicting techniques in social media, data monitoring, etc. can be done by data mapping.
* **Deep learning**

It’s a subclass of machine learning which can be used for speech recognition, financial fraud detection, and medical video analysis. Tools such as time-series, Artificial neural network (ANN), Fuzzy logic or combination of such tools can be employed.

* **Financial analysis**

An entrepreneur before starting any endeavor needs to do a proper survey and the financial analysis in order to plan the course of action. The tools needed for this are all available in MATLAB. Elements such as profitability, solvency, liquidity, and stability can be identified. Business valuation, capital budgeting, cost of capital, etc. can be evaluated.

* **Video processing**

The most common application that we observe almost every day are bar code scanners, selfie (face beauty, blurring the background, face detection), video enhancement, etc. The digital video processing also plays quite an important role in transmitting data from far off satellites and receiving and decoding it in the same way. Algorithms to support all such applications are available.

* **Text analysis**

Based on the text, sentiment analysis can be done. Google gives millions of search results for any text entered within a few milliseconds. All this is possible because of text analysis. Handwriting comparison in forensics can be done. No limit to the application and just one software which can do this all.

* **Electric vehicles designing**

Used for modeling electric vehicles and analyze their performance with a change in system inputs. Speed torque comparison, designing and simulating of a vehicle, whatnot.

* **Aerospace**

This toolbox in MATLAB is used for analyzing the navigation and to visualize flight simulator.

* **Audio toolbox**

Provides tools for audio processing, speech analysis, and acoustic measurement. It also provides algorithms for audio and speech feature extraction and audio signal transformation.

**COMMUNICATION:**

Communications System Toolbox™ offers algorithms and gear for the layout, simulation, and analysis of communications systems. These capabilities are furnished as MATLAB ® features, MATLAB System gadgets™, and Simulink ® blocks. The machine toolbox includes algorithms for source coding, channel coding, interleaving, modulation, equalization, synchronization, and channel modeling. Tools are supplied for bit blunders charge evaluation, producing eye and constellation diagrams, and visualizing channel characteristics. The machine toolbox additionally provides adaptive algorithms that allow you to version dynamic communications structures that use OFDM, OFDMA, and MIMO techniques. Algorithms support fixed-point facts arithmetic and C or HDL code era.

**Key Features**

▪ Algorithms for designing the physical layer of communications systems, which includes supply coding, channel coding, interleaving, modulation, channel fashions, MIMO, equalization, and synchronization

▪ GPU-enabled System objects for computationally intensive algorithms together with Turbo, LDPC, and Viterbi decoders

▪ Interactive visualization equipment, consisting of eye diagrams, constellations, and channel scattering capabilities

▪ Graphical tool for evaluating the simulated bit mistakes rate of a machine with analytical outcomes

▪ Channel models, consisting of AWGN, Multipath Rayleigh Fading, Rician Fading, MIMO Multipath Fading, and

LTE MIMO Multipath Fading

▪ Basic RF impairments, along with nonlinearity, section noise, thermal noise, and section and frequency offsets

▪ Algorithms available as MATLAB features, MATLAB System objects, and Simulink blocks

▪ Support for fixed-point modeling and C and HDL code technology

**System Design, Characterization, and Visualization:**

The layout and simulation of a communications gadget requires analyzing its reaction to the noise and interference inherent in real-world environments, reading its behavior the usage of graphical and quantitative manner, and determining whether the resulting overall performance meets requirements of acceptability. Communications System Toolbox implements a selection of obligations for communications machine layout and simulation. Many of the functions, System objects™, and blocks inside the device toolbox perform computations associated with a specific thing of a communications gadget, consisting of a demodulator or equalizer. Other talents are designed for visualization or evaluation.

**System Characterization**

The system toolbox offers several standard methods for quantitatively characterizing system performance:

▪ Bit error rate (BER) computations

▪ Adjacent channel power ratio (ACPR) measurements

▪ Error vector magnitude (EVM) measurements

▪ Modulation error ratio (MER) measurements

Because BER computations are fundamental to the characterization of any communications system, the system toolbox provides the following tools and capabilities for configuring BER test scenarios and accelerating BER simulations:

**BER tool**— A graphical user interface that enables you to analyze BER performance of communications systems. You can analyze performance via a simulation-based, semi analytic, or theoretical approach.

**Error Rate Test Console** — A MATLAB object that runs simulations for communications systems to measure error rate performance. It supports user-specified test points and generation of parametric performance plots and surfaces. Accelerated performance can be realized when running on a multi core computing platform.

**Multi core and GPU acceleration** — A capability provided by Parallel Computing Toolbox™ that enables you to accelerate simulation performance using multi core and GPU hardware within your computer.

**Distributed computing and cloud computing support** — Capabilities provided by Parallel Computing Toolbox and MATLAB Distributed Computing Server™ that enable you to leverage the computing power of your server farms and the Amazon EC2 Web service. Performance Visualization. The system toolbox provides the following capabilities for visualizing system performance:

**Channel visualization tool** — For visualizing the characteristics of a fading channel

**Eye diagrams and signal constellation scatter plots** — for a qualitative, visual understanding of system behavior that enables you to make initial design decisions

**Signal trajectory plots** — for a continuous picture of the signal’s trajectory between decision points

**BER plots** — for visualizing quantitative BER performance of a design candidate, parameterized by metrics such as SNR and fixed-point word size

**Analog and Digital Modulation**

Analog and digital modulation strategies encode the facts circulation into a sign this is appropriate for transmission. Communications System Toolbox presents some of modulation and corresponding demodulation abilities. These talents are available as MATLAB features and gadgets, MATLAB System Modulation sorts provided by the toolbox are:

**Source and Channel Coding**

Communications System Toolbox affords source and channel coding talents that can help you develop and compare communications architectures fast, enabling you to discover what-if eventualities and avoid the need to create coding competencies from scratch.

**Source Coding**

Source coding, also referred to as quantization or signal formatting, is a manner of processing facts a good way to lessen redundancy or prepare it for later processing. The system toolbox offers a diffusion of styles of algorithms for imposing source coding and interpreting, inclusive of:

▪ Quantizing

▪ Companding (*µ*-law and A-law)

▪ Differential pulse code modulation (DPCM)

▪ Huffman coding

▪ Arithmetic coding

**Channel Coding**

▪ orthogonal area-time block code (OSTBC) (encoder and decoder for MIMO channels)

▪ Turbo encoder and decoder examples

The gadget toolbox offers application functions for developing your personal channel coding. You can create generator polynomials and coefficients and syndrome deciphering tables, in addition to product parity-take a look at and generator matrices.

The system toolbox additionally presents block and convolutional interleaving and deinters leaving functions to reduce facts errors as a result of burst mistakes in a conversation machine:

**Block,** including General block interleaver, algebraic interleaver, helical scan interleaver, matrix interleaver, and random interleaver.

**Convolutional,** including General multiplexed interleaver, convolutional interleaver, and helical interleaver

**Channel Modeling and RF Impairments**

Channel Modeling

Communications System Toolbox provides algorithms and tools for modeling noise, fading, interference, and different distortions which might be commonly found in communications channels. The system toolbox supports the subsequent styles of channels:

▪ Additive white Gaussian noise (AWGN)

▪ Multiple-enter multiple-output (MIMO) fading

▪ Single-enter single-output (SISO), Rayleigh, and Rician fading

▪ Binary symmetric

A MATLAB channel object provides a concise, configurable implementation of channel models, enabling you to

specify parameters such as:

▪ Path delays

▪ Average path gains

▪ Maximum Doppler shifts

▪ K-Factor for Rician fading channels

▪ Doppler spectrum parameters

For MIMO systems, the MATLAB MIMO channel object expands these parameters to also include:

▪ Number of transmit antennas (up to 8)

▪ Number of receive antennas (up to 8)

▪ Transmit correlation matrix

▪ Receive correlation matrix

To combat the effects noise and channel corruption, the system toolbox provides block and convolutional coding and decoding techniques to implement error detection and correction. For simple error detection with no inherent correction, a cyclic redundancy check capability is also available. Channel coding capabilities provided by the system toolbox include:

▪ BCH encoder and decoder

▪ Reed-Solomon encoder and decoder

▪ LDPC encoder and decoder

▪ Convolutional encoder and Viterbi decoder

****

**RF Impairments**

To model the effects of a non-ideal RF front end, you can introduce the following impairments into your communications system, enabling you to explore and characterize performance with real-world effects:

▪ Memory less nonlinearity

▪ Phase and frequency offset

▪ Phase noise

▪ Thermal noise

You can include more complex RF impairments and RF circuit models in your design using SimRF™.

****

**Equalization and Synchronization**

Communications System Toolbox lets you discover equalization and synchronization strategies. These techniques are usually adaptive in nature and tough to design and symbolize. The machine toolbox affords algorithms and tools that will let you swiftly select the proper approach on your communications machine. Equalization To compare one-of-a-kind techniques to equalization, the device toolbox offers you with adaptive algorithms which include:

▪ LMS

▪ Normalized LMS

▪ Variable step LMS

▪ Signed LMS

▪ MLSE (Viterbi)

▪ RLS

▪ CMA

These adaptive equalizers are available as nonlinear decision feedback equalizer (DFE) implementations and as

Linear (symbol or fractionally spaced) equalizer implementations.

**Synchronization**

The device toolbox provides algorithms for each service segment synchronization and timing phase synchronization. For timing section synchronization, the machine toolbox presents a MATLAB Timing Phase Synchronizer object that offers the following implementation techniques:

▪ Early-late gate timing method

▪ Gardner’s method

▪ Fourth-order nonlinearity method

**Stream Processing in MATLAB and Simulink**

Most verbal exchange structures cope with streaming and frame-primarily based statistics using a aggregate of temporal processing and simultaneous multi frequency and multichannel processing. This form of streaming multidimensional processing can be visible in superior communication architectures consisting of OFDM and MIMO. Communications System Toolbox enables the simulation of advanced communications structures via helping move processing and frame-based simulation in MATLAB and Simulink. In MATLAB, circulate processing is enabled by way of System items™, which use MATLAB objects to symbolize time-based and facts-driven algorithms, sources, and sinks. System objects implicitly manipulate many information of flow processing, including information indexing, buffering, and management of set of rules state. You can mix System gadgets with fashionable MATLAB functions and operators. Most System items have a corresponding Simulink block with the identical abilities. Simulink handles circulation processing implicitly with the aid of coping with the float of information thru the blocks that make up a Simulink model. Simulink is an interactive graphical environment for modeling and simulating dynamic systems that uses hierarchical diagrams to symbolize a machine version. It includes a library of widespread-reason, predefined blocks to represent algorithms, resources, sinks, and device hierarchy.

**Implementing a Communications System**

Fixed-Point Modeling Many communications systems use hardware that requires a fixed-point representation of your design.

Communications System Toolbox supports fixed-point modeling in all relevant blocks and System objects™ with tools that help you configure fixed-point attributes.

Fixed-point support in the system toolbox includes:

▪ Word sizes from 1 to 128 bits

▪ Arbitrary binary-point placement

▪ Overflow handling methods (wrap or saturation)

▪ Rounding methods: ceiling, convergent, floor, nearest, round, simplest, and zero

Fixed-Point Tool in Simulink Fixed Point™ facilitates the conversion of floating-point data types to fixed point. For configuration of fixed-point properties, the tool tracks overflows and maxima and minima.

**Code Generation**

Once you've got advanced your set of rules or communications device, you can robotically generate C code from it for verification, rapid prototyping, and implementation. Most System gadgets, functions, and blocks in Communications System Toolbox can generate ANSI/ISO C code the use of MATLAB Coder™, Simulink Coder™, or Embedded Coder™. A subset of System gadgets and Simulink blocks also can generate HDL code. To leverage present highbrow belongings, you can choose optimizations for specific processor architectures and integrate legacy C code with the generated code.

You can also generate C code for both floating-point and fixed-point data types.

DSP Proto typing DSPs are used in communication system implementation for verification, rapid prototyping, or final hardware implementation. Using the processor-in-the-loop (PIL) simulation capability found in Embedded Coder, you can verify generated source code and compiled code by running your algorithm’s implementation code on a target processor. FPGA Prototyping

FPGAs are used in communication systems for implementing high-speed signal processing algorithms. Using the FPGA-in-the-loop (FIL) capability found in HDL Verifier™, you can test RTL code in real hardware for any existing HDL code, either manually written or automatically generated HDL code.