Data driven modeling and MPC Based control for pathological tremors

Subham S Samal

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

Oumar Barry, Chair
Kaveh Akbari Hamed
Steve Southward

December 3, 2024
Blacksburg, Virginia

Keywords: Exoskeletons, Optimization, Model Predictive Controller, Pathological Tremor, Data-driven modeling

Data driven modeling and MPC Based control for pathological tremors

Subham S Samal

(ABSTRACT)

Pathological tremor is a common neuromuscular disorder that significantly affects the quality of life for patients worldwide. With recent developments in robotics, rehabilitation exoskeletons serve as one of the solutions to alleviate these tremors. For improved performance of such devices, we need to solve a few problems, which include developing a model for pathological tremors, and a safe control system that can conveniently incorporate constraints on the wrist's range of motion and it's input force/torque.

Accurate predictive modeling of tremor signals can be used to provide alleviation from these tremors via various currently available solutions like adaptive deep brain stimulation, electrical stimulation and rehabilitation orthoses. Existing methods are either too general or too simplistic to accurately predict these tremors in the long term, motivating us to explore better modeling of tremors for long-term predictions and analysis. We explore towards the prediction of tremors using artificial neural networks using EMG signals, leveraging the 20-100 ms of Electromechanical Delay. The kinematics and EMG data of a publicly available Parkinson's tremor dataset is first analyzed, which confirms that the underlying EMGs have similar frequency composition as the actual tremor. 2 hybrid CNN-LSTM based deep learning architectures are then proposed to predict the tremor kinematics ahead of time using EMG signals and tremor kinematics history, and the results are compared with baseline models. This is then further extended by adding constraints-based losses in an attempt to further improve the predictions.

Then, we explore the application of model-based predictive control (MPC) for the full wrist

exoskeleton designed in our lab for the alleviation of tremors. The main motivation for using MPC here relies on its ability to incorporate state and input constraints, which are crucial for the user's safety. We employ a linear MPC methodology, in which the forearm-exoskeleton model is successively linearized at each time sample to obtain a linear state space model, which is then used to obtain the optimal input by minimizing a convex quadratic cost function. This is then integrated with the tremor model developed via BMFLC and neural networks to provide tremor suppression. Simulation studies are provided to demonstrate the effectiveness of the control schemes. The numerical simulations suggest that the MPC framework is capable of accurate trajectory tracking while providing better tremor suppression than a PD controller without using any tremor model, while the neural network model outperforms the frequency-based BMFLC model. The findings could set up for devising physics-based Neural networks for pathological tremor modeling and experimentally evaluate the performance of the developed framework.

Data driven modeling and MPC Based control for pathological tremors

Subham S Samal

(GENERAL AUDIENCE ABSTRACT)

Pathological tremors are involuntary, rhythmic, and oscillatory movements of the limbs that affect millions of people worldwide, and daily life activities like writing, eating, and object manipulation are challenging for them. In recent years, rehabilitation exoskeletons have been developed as non-invasive solutions to pathological tremor alleviation. The wrist is pivotal to human manipulation capabilities, and thus, a wrist exoskeleton (TAWE) has been developed in our lab to provide tremor alleviation. For improved performance of such devices, we need to solve a few problems, which include developing a model for pathological tremors, and a safe control system that can conveniently incorporate constraints on the wrist's range of motion and its input force/torque. We propose a deep-learning-based method for accurate modeling of tremors, along with a model predictive control framework for tremor suppression. Simulations and analyses are done to validate the tremor-modeling framework and the control framework of an exoskeleton for the tremor alleviation, and highlight shortcomings in the current methods that call for more research and advancements.

Dedication

Dedicated to my parents, and my brother whose unwavering support and encouragement made this journey possible

Acknowledgments

I would like to express my utmost gratitude to my advisor, Dr. Oumar Barry for his teachings, guidance, feedback and insights, and constant encouragement and support throughout my study. Dr. Barry's guidance and encouragement have motivated me to strive for excellence, maintain consistency, and realize my full potential, making me a better researcher and engineer in the process.

I would like to extend my thanks to Dr. Kaveh Akbari Hamed and Dr. Steve Southward for their time in reviewing this work and for their guidance during the courses I took under their instruction.

I would like to sincerely thank Dr. Jiamin Wang for his expertise and guidance throughout various aspects of this study. I am deeply grateful to Nipun Nambiar, Sudarsana Jayandan, Zijian Ding, Dr. Sunit Gupta, and other members of the VibRo Lab for their help and support in my research and daily life.

Additionally, I would like to express my gratitude to the Department of Mechanical Engineering at Virginia Tech, and the NSF Grant for partially funding my study.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

ADL Activities of Daily Life

BMFLC Band-limited multi-frequency Fourier linear combiner

CNN Convolutional Neural Network

DBS Deep brain stimulation

DOF Degree of freedom

EFE Elbow flexion-extension

EMD Electromechanical Delay

EMG Electromyography

ET Essential Tremor

IMU Inertial measurement unit

LSTM Long Short-term memory

MLP Multilayer perceptron

MPC Model Predictive Control

PC Pearson's correlation coefficient

PD Proportional-Derivative (Controller)

PS Pronation-supination (of the forearm)

PT Parkinson's Tremor

ReLU Rectified linear unit

RMS Root mean square

RUD Radial-ulnar-deviation (of the wrist)

TAWE Tremor-Alleviating Wrist Exoskeleton (a device devised in our lab)

WFE Wrist flexion-extension

Mathematical Notations

ϵ Tracking/Estimation error (for controls and optimizations)

λ Lagrange Multiplier

$\mathcal{J}$ Optimization function

ϕ Uncertain amplitude parameters (BMFLC)

$\text{col}(x_1...x_n)$ Column vector with elements $x_1...x_n$

$\text{diag}(x_1...x_n)$ N by N diagonal matrix with diagonal entries $x_1...x_n$

$\|Z\|^n$ n-norm of a matrix Z

$A \succ 0$ Positive definite square matrix A

I_n n by n Identity Matrix

C Coriolis and centripetal matrix

f General model process function

h	Generalized force vector
J	Jacobian Matrix
M	Inertia matrix that satisfies $M = M^T > 0$
P	Cost matrix associated with the terminal state
Q	Cost matrix associated with the non-terminal states
q	Generalized states
R	Cost matrix associated with inputs
S	Error weighing matrix for tremor modeling
U	Cost matrix for amplitude parameters (BMFLC)
u	Input (for dynamical systems and optimizations)
y	Output (for neural-network models)
Z	Time series

Chapter 1

Background and Motivations

1.1 Background: Pathological Tremors

Pathological tremors are involuntary rhythmic movements of varying frequencies and amplitudes that can affect one or more body parts, hindering the patients from performing activities of daily living (ADLs) like writing, eating, and object manipulation. Pathological tremors are known to affect millions of people around the world. The 2 most common adult tremors are Parkinsonian Tremor (PT), and Essential Tremor (ET). Studies in 2017 have shown that about 10 million people of the entire population suffer from pathological Parkinson's disease [39]. A study in 2022 ranged the total incident PD diagnoses in North America from approximately 60,000 to 95,000 among adults 45 years and older[89]. A study in 2020 estimated that 0.3-0.9% of the general population suffered from Essential Tremor (ET) [46].

1.1.1 Causes and Symptoms

All types of tremor in humans are caused by one of the four basic factors. They are mechanical tremor of the extremity, reflex activation leading to oscillatory activity, central oscillators, and finally the malfunction of feed-forward loops within the CNS, especially the cerebellum. Studies have indicated that Parkinson's tremor (PT) is related to the inadequacy of Dopaminergic neurons in the substantia nigra of the Central Nervous System (CNS) [51]

[32]; these are the neurons involved in information relay for motor control, while Essential Tremor (ET) could be passed down over generations [19], or may result from toxins and cerebellar overactivity [8] [20]. These different pathophysiologies could be a contributing factor to the distinctive behaviors of tremors in PT and ET [47]. For instance, the common frequency bands are 3-6 Hz and 4-12 Hz for Parkinsion's and Essential tremor respectively. ET tremors usually have variable amplitude, while PT tremors have a relatively consistent amplitude. The tremor in Parkinson's disease is typically a resting tremor [12], meaning it occurs when the muscles are at rest and diminishes with voluntary movement. This type of tremor is often described as a "pill-rolling" motion, where the thumb and index finger appear to be rolling a small object. On the other hand, essential tremor is typically an action tremor [86], meaning it occurs when the affected muscles are actively being used. For example, it becomes noticeable when a person is holding an object, writing, or trying to perform precise movements. Parkinsonian Tremors are only one symptom of the disease, as they frequently occur alongside additional symptoms, including bradykinesia (or movement slowness) [9], muscular rigidness [6], and unstable posture [57] etc. As PD advances, other non-motor symptoms such as cognitive decline [1], mood disorders[70], and autonomic dysfunction often appear. ET, although primarily is a movement disorder, it does not typically involve other neurological symptoms like bradykinesia, rigidity, or postural instability. progresses slowly over time. However, since PT and ET are connected and may co-occur[74], misdiagnosis[37] of these two tremors can still happen despite these differences. While pathological tremors don't directly lead to mortality, they can increase susceptibility to possible environmental dangers. This is especially true for the elderly population, who are the most affected group. Elderly patients often have other aging diseases like osteoporosis [38] (decrease in bone density and bone strength) and sarcopenia [53] (reduced muscle strength), making them more prone to accidents and falling, which can cause severe injuries.

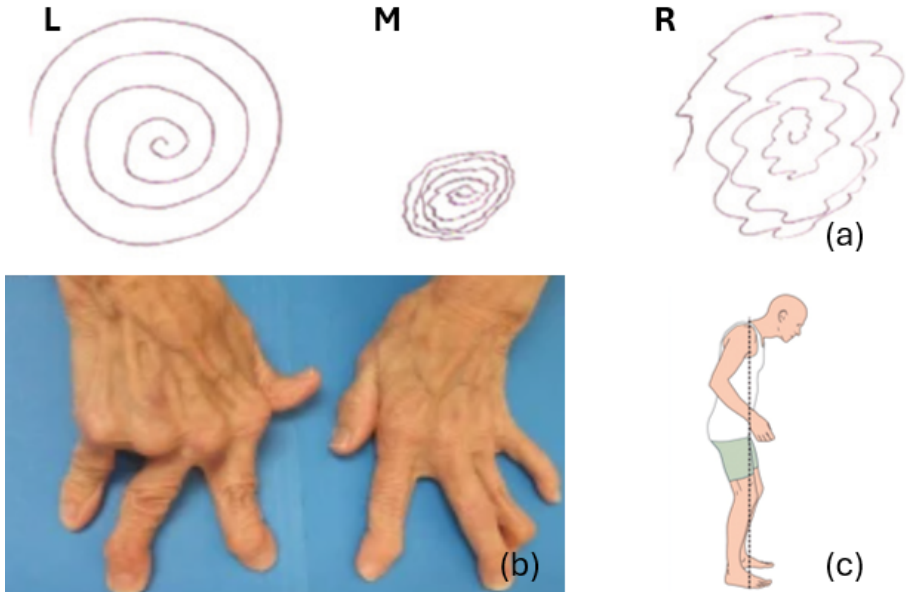


Figure 1.1: Symptoms of Pathological tremors - (a): PT (M) and ET (R) patients unable to draw a Archimedean spiral as smoothly as a healthy person (L) [3]; (b): deformities observed on the hands of a PD diagnosed patient [88]; (c): common deformity in posture for patients affected with PD [21]

1.1.2 Currently available treatments

Pharmacological therapy is often the first line of treatment for most cases. Medication aims to reduce tremor amplitude and frequency, improve functionality, and enhance quality of life. Pharmacological therapies can reduce tremors by as much as 68% for ET and 59% for Parkinson's disease [23][41]. Some common medications for ET include Beta-blockers, such as propranolol, and anticonvulsants like Primidone, but they can cause side effects like bradycardia, fatigue, and nausea [62]. Similarly, levodopa, considered one of the best medications for PT, can lead to long-term complications such as motor fluctuations and dyskinesias[67].

Surgical intervention is also an option, particularly for patients with severe, disabling tremors. The two main surgical approaches are deep brain stimulation (DBS) [5] and thalamotomy[24].

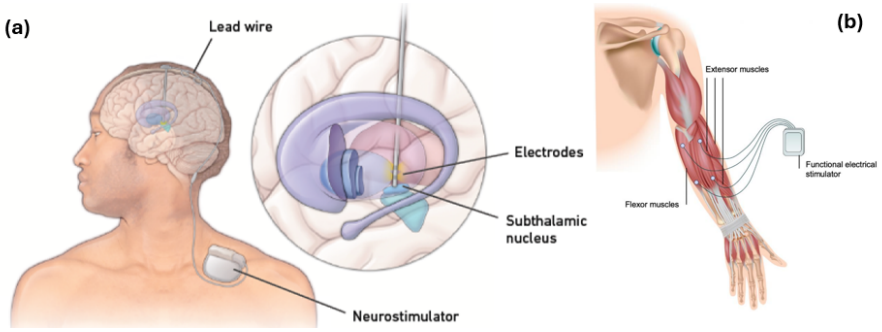


Figure 1.2: Treatments for pathological tremors - (a) Deep brain stimulation (b) Functional Electrical Stimulation[52]

DBS involves implanting electrodes into specific regions of the brain, and has been found to be 80 to 90% effective[5]. Thalamotomy is a surgical procedure in which a small, targeted lesion is created in the ventral intermediate nucleus of the thalamus, and is less frequently used due to its permanent and non-reversible nature. However, both major surgeries are expensive and aren't without risks, accompanied by surgical risks like infection, bleeding, or hardware malfunction.

Other than the mentioned surgical and pharmacological treatments, other treatment methods include different therapies [56]. Methods to stimulate the peripheral nervous system like functional electrical stimulation [63] and sensory electrical stimulation [22] can generate muscle contractions that mitigate the effects of tremors. But, FES may cause patients to feel fatigued or unable to move freely, and neural adaptation may cause electrical stimulation to lose its effectiveness over time. Cognitive behavioral therapy (CBT) may help patients develop coping mechanisms to manage anxiety and stress, which in turn may reduce tremor severity[95], but its efficacy is debatable.

1.2 Background: Exoskeletons for Tremor Suppression

Unlike previous treatments, which are invasive and often come with side effects, exoskeletons and wearable orthoses provide a non-invasive alternative as they don't require stimulating the nervous system or any modification of any part of the human body. Therefore, in recent years, they have emerged as a promising technology. These devices can be classified into 3 categories: passive, semi-active and active; based on how they are powered.

1. Passive devices do not rely on external power sources, and are purely mechanical systems that utilize springs, dampers, or other passive elements for vibration isolation and absorption

2. Semi-active devices use variable dampers or tunable stiffness mechanisms that adjust to the user's needs in real-time based on the tremor behavior, making them more efficient than passive systems but less complex than fully active models.

3. Active devices are powered systems that use motors, actuators, and sensors to provide full assistance or stabilization. They can actively counteract tremors by detecting involuntary movements and applying corrective forces and require external power.

Development of semi-active and active devices typically integrates advanced robotics, biomechanics, and machine learning algorithms, and are called exoskeletons.

Designing of tremor-alleviating exoskeletons involves analyzing tremor patterns so that the exoskeletons can predict and respond to these tremors with high precision. Therefore, the first section discusses the signal processing and prediction methods for pathological tremors. Then, previously developed exoskeletons for tremor alleviation and the current progress with newer technologies are discussed. Finally, different control methods for exoskeleton control are discussed.

1.2.1 Processing and prediction of tremor signals

Both semi-active and active tremor suppression devices require processing and prediction of tremor signals. The existing solutions discussed in section 1.1.2 like adaptive deep brain stimulation and electrical stimulations would also benefit from accurate modeling and prediction of tremor signals. Tremors are a result of some extremely complicated dynamical behavior of the human neuromusculoskeletal system [49][97]. Earlier studies have theorized presence of highly complex and nonlinear neuromusculoskeletal dynamics behind pathological tremors, which involve motor cortex [45], feedback/reflex loops (e.g., Golgi Tendon Organs, Renshaw Cells, Spindle Organs) [97], and time delays [30]. These, along with other works also indicated tremor cycles are not necessarily strictly periodic, are non-linear and involve stochasticity [27][31].

Many tremor suppression devices mainly adopt the short-term prediction of tremor signals (a few milliseconds) based on different time-series analyses. Usually, sensors like IMUs (inertial measurement units), accelerometers and encoders [77][26][15]. A few studies also include surface EMG (sEMG) sensors, which are then used to estimate the muscle forces [96]. Methods like exponential moving averages are implemented to reduce sensor noise in real time. These collected data include the users' voluntary motion superimposed with tremors. Estimation models like Auto-Regressive Moving Average (ARMA)[7], offline forward-backward filtering [7] are used to separate the intended voluntary motions from the true motions. Then, The filtered tremor movements are regressed for prediction. Time delay and harmonic/frequency models serve as the foundation for many of the tremor prediction algorithms currently in use, such as the Band-limited Multi-frequency Fourier Linear Combiner (BMFLC)[77], Weighted-frequency Fourier Linear Combiner (WFLC) [64], and Autoregressive model (AR)[73]. These models are designed to approximate the tremor dynamics for a short time using the information from a limited window of delayed time series. Kalman Filters are also implemented

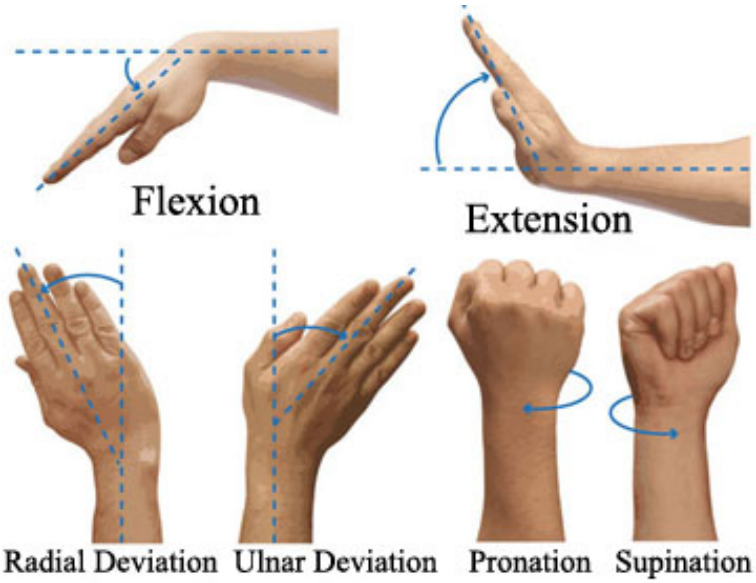


Figure 1.3: Wrist joint motions

as an online optimizer to update the parameters of the models in real-time[77].

For active exoskeletons, along with tremor prediction, often voluntary motion is also predicted to assist the user intended movement while suppressing tremors [79] [50]. For semi-active exoskeletons, tremor prediction algorithms are primarily used to tune damping parameters, but this damping can impede voluntary movement as a side effect.

1.2.2 Existing Tremor Rehabilitation Exoskeletons

The motion in the wrist joint has 3 DOFs: wrist flexion-extension (WFE), wrist abduction (radial-ulnar-deviation) (RUD) and forearm pronation-supination (FPS). For daily manipulation activities, all of these motions are crucial. At the same time, all of these motions can be affected by pathological tremors as well[54][59].

Different tremor suppression devices have been devised over time for one or many DOFs. A few wearable exoskeletons (semi-active and active) are shown in Figure 1.4.

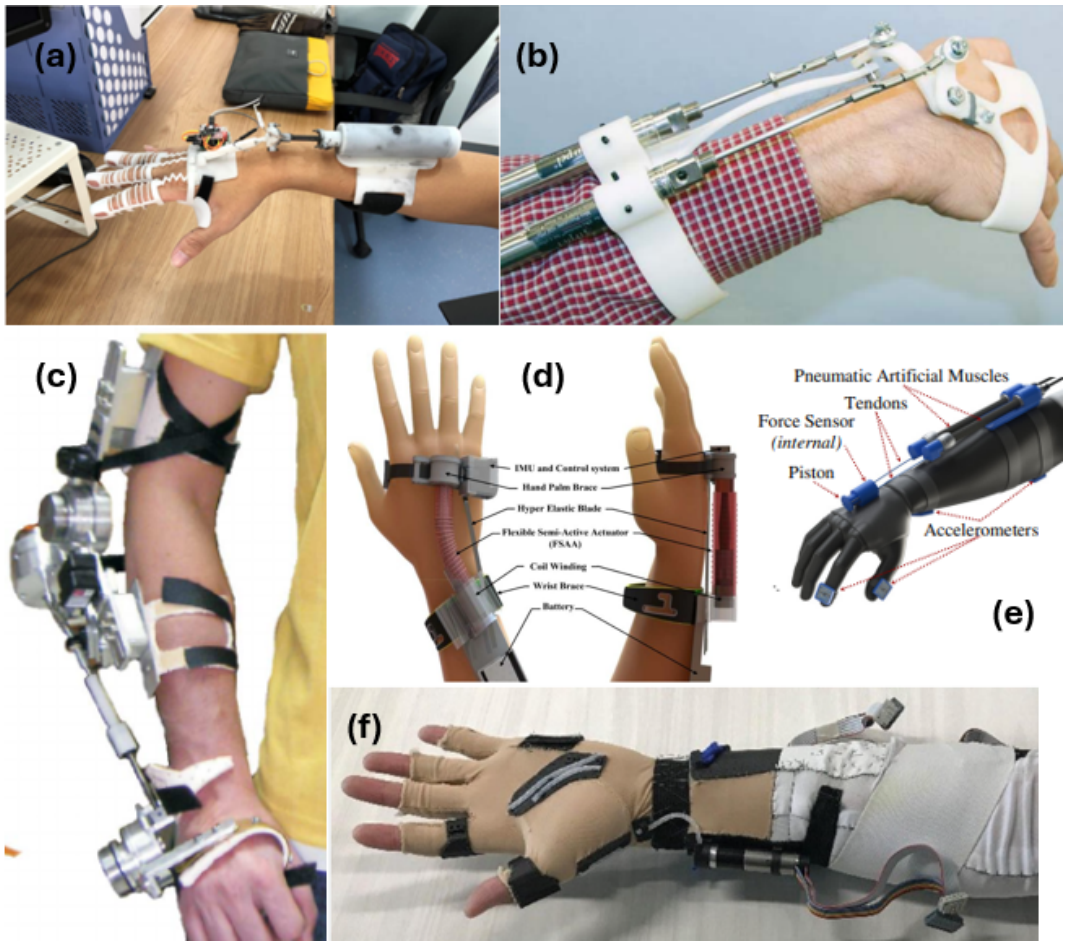


Figure 1.4: Tremor suppression exoskeleton designs and prototypes: (a) Work by Yi et al. - impedes WFE using a magnetorheological (MR) fluid damper (semi-active)[91];(b) Work by Taheri et al. - actuates FPS, WFE, and RUD with pneumatic linear actuators (semi-active)[69];(c) WOTAS by Rocon et al. - impedes WFE and FPS using DC motors (active) [65];(d) SETS by Zahedi et al. - impedes WFE and RUD using MR fluid dampers (semi-active) [94];(e) Work by Skaramagkas et al. - using pneumatic artificial muscles (active) [68];(f) WTSG by Zhou et al - impedes WFE and finger tremors using DC motors (active) [98];

The majority of the active devices (exoskeletons) use rigid mechanisms actuated by electrical motors for active tremor suppression (Figs. 1.4 (c) and (f)) and featured rigid mechanisms. Electrical motors use less energy and are simple to use. However, since these electrical motors are typically located on the upper limb, the user is subjected to heavy gravitational loads.

Some exceptions like (e) do exist, which use pneumatic systems (like artificial muscles) for actuation.

Both semi-active and active tremor suppression devices use fluid-based actuators as well, as depicted in Figs. 1.4 (a), (b), and (d). This results in significantly less gravitational strain, while also enabling higher actuation torques. Nevertheless, additional hardware is needed for pneumatic actuators, such as air compressors and valves, which produce loud operating noises. In semi-active exoskeletons, magnetorheological (MR) fluid dampers—such as those shown in Fig 1.4(d) are more frequently utilized to generate damping forces. The advantage being that the dampers' characteristics can be actively changed for better damping of tremors.

Last but not least, as soft robotics research gains traction, more tremor control exoskeletons created recently have started integrating soft actuators and structures with the aim of providing better flexibility and ergonomics. These apparatuses typically use soft MR fluid actuators, viscoelastic soft tendons, piezoelectric fabric actuators etc. One challenge still remains is the fact that the soft structures are more challenging to model and actuate. Additionally, some soft materials have restricted ranges of motion, which prevents them from fully tracking the motion of the human body. As a result, a lot of soft tremor suppression devices are still in the conceptual design phase.

1.2.3 Exoskeleton Control

The complexity and nonlinearity of the exoskeleton dynamics make controlling an exoskeleton system a difficult task. Traditional control methods, such as Proportional-Integral-Derivative (PID) control and impedance control, are often used in these systems. PID control offers simplicity but struggles with the complex, nonlinear dynamics and time delays

present in human-exoskeleton interaction. Impedance control, while good for handling interaction forces, may be less effective at providing predictive and adaptive adjustments, leading to less efficient responses in dynamic environments. To address these issues, a number of control algorithms—most notably sliding mode controllers and adaptive control techniques—have been developed in recent years. Sliding Mode Control (SMC) offers robustness against disturbances and model uncertainties by driving the system's state to a "sliding surface" and maintaining it there, ensuring stability even under varying conditions. Adaptive controllers adjust their parameters in real-time based on system performance, making them suitable for handling changes in the exoskeleton's dynamics or variations in the user's behavior. However, for both of these, incorporating state and input constraints becomes a challenge, which are crucial for the user's safety.

In recent years, researchers have been leaning towards towards model-based optimal control schemes. These methods aim to obtain the desired outcomes and simultaneously fulfilling imposed constraints by solving a finite or infinite horizon optimal control problem. They are becoming a popular choice among researchers due to their intrinsic robustness, capability of controlling complicated uncertain and disturbed systems, and ease of incorporating state and input constraints. In model-based optimal control, an optimization algorithm is used to optimize an appropriate cost function and forecast the results of potential actions in order to produce the best possible future plan. One such realization of model-based optimal control is called the model-predictive control (MPC) ([11]). The model predicts how the system will behave over a few future time steps at each sample interval. The control horizon is defined as the duration of these future time steps over which the objective function is minimized. Usually, the constrained optimization problem of this minimization function is solved at every sampling interval. Although prediction and optimization are done for the full control horizon, just the initial input sequence, or inputs for the current sampling period, are used.

At the next sampling time, this procedure is redone by moving the time window by one sampling time forward, and so on. This strategy is known as the moving or receding horizon strategy.

1.3 Contributions of the Thesis

The contributions of this thesis are listed below:

1. A review of the current methodologies used for modeling tremor and exoskeleton control, and the shortcomings and challenges faced. This review is used to highlight the need for developing and implementing state-of-the-art methods for our our in-house designed tremor suppression exoskeleton

2. Developing a deep-learning-based approach for tremor modeling. This model allows us to predict the tremor up to 100 milliseconds ahead of time, compared to a few milliseconds in previous studies; making it more convenient to apply in real-time.

3. Developing a model predictive controller for the exoskeleton for simultaneous trajectory tracking and tremor suppression. Unlike other control strategies, this methodology allows us to explicitly apply constraints based on the wrist's torque and kinematic limits, making it a safer design.

4. Integrating model predictive controller with tremor suppression. This framework allows us to combine the tremor information learned from the data-driven model with the model predictive controller, helping to achieve much better tremor suppression compared to standalone controllers.

1.4 Thesis Structure and Selected Publications

The introduction and motivation to this thesis is presented in Chapter 1. Section 2 discusses the approaches taken for data-driven modeling of tremors. We aim to make use of sEMG signals and the electromechanical delay to predict the tremors milliseconds ahead of time using deep learning approaches. The modeling and evaluation is done in a Python environment and the holistic results are presented. Chapter 3 starts off with the development of the model predictive control framework. This is followed by the integration of the data-driven models into the MPC with an aim to further improve tremor suppression performance. Simulations of the proposed methodologies are carried out in a MATLAB environment, and the numerical results are used to demonstrate the efficacy of the controller for trajectory tracking and tremor suppression, and the improvement in performances are compared. In Chapter 4, key insights of the results are discussed. Finally, some conclusions are drawn in Chapter 5 and some future work ideas are presented.

Disclaimer: Content from the following publications are used throughout this work.

Conference Proceedings:

1. **Subham Samal** and Oumar Barry. Model predictive control for tremor suppressing exoskeleton. IFAC-PapersOnLine, 56(3):337–342, 2023.

2. **Subham Samal** and Oumar Barry. Exploring Deep Learning Models for Pathological Tremors Prediction Using EMG and Kinematic Measurements. ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference 2024 (*Awaiting Publication*)

Chapter 2

Data driven modeling

This project begins with the approaches taken for modeling of pathological tremors. As discussed in section 1.2.1, tremors are a result of some extremely complicated non-linear dynamical behavior of the human neuromusculoskeletal system. Previously developed tremor suppression devices adopted short-term prediction of tremor signals, which is in orders of a few milliseconds based on different time delay and harmonic/frequency models.

Deep learning methods have developed quite rapidly in recent years. In the domain of tremor signals forecasting, various previous studies have used neural networks to predict tremors using historical kinematic signals. Ibrahim et al. [35] proposed a hybrid convolutional multilayer perceptron architecture and Wang et al [81] proposed an LSTM architecture. EMG activity from skeletal muscles precedes mechanical tension by 20 to 100 ms [14][87]. This electromechanical delay motivated us to use EMG data to develop a prediction model. Previous studies have leveraged this delay to predict joint kinematics using EMG signals and neural networks [92] [42] [25]. Here, we aim to extend this approach to predict pathological tremor kinematics from EMG. In this study, we propose to evaluate and compare different neural network architectures, using MLPs, LSTM, and combined models to predict wrist tremors using sEMG signals. An openly available dataset is used to test the efficacy of our models. Our primary aim is to be able to successfully predict about 100 ms the tremor pattern (about one-half cycle of tremor) so that we can use it later for the rehabilitation exoskeleton being developed in our lab [79].

There are a few studies that have conducted modeling of the neuromusculoskeletal (NMS) systems, particularly for tremor studies [97][44]. However, the adopted models are simplified or limited to linear models. These unsolved problems have motivated us to explore a better model for accurate long-term predictions.

In general, the musculoskeletal system can be represented as:

$$\tau = f_\eta(Z_q, Z_\tau, Z_{\eta)} \tag{2.1}$$

where τ represents the muscle force, Z_q, Z_τ, Z_η represent the delayed time series of kinematics states q, muscle load τ and neuromuscular states η.[84]. Using forward dynamics, we can write the kinematic states as:

$$\ddot{q} = f_q(q, \dot{q}, \tau, u) \tag{2.2}$$

which is a general multi-body model[29][85] of a forearm skeletal system, and u is the external inputs (gravity, ext forces). Using the above, we can infer that the current kinematic state is a function of time series of historical neuromuscular states and kinematic states, among others. In this study, using neural networks, we aim to obtain the best approximation of this unknown function for pathological tremors. Thus, the input for our neural network models are the historical EMG signals and the past kinematic signals, and the predicted output are the future kinematic signals.

2.1 Dataset Used

For this study, a publicly available dataset was obtained from a recent study by Pinheiro et al. [60]. The study consisted of 5 volunteers, 4 of whom were diagnosed with PT, and 1 in ET. Two sEMG electrodes were placed on the ECRL and FCU muscles of the most

affected arm and an Inertial Measurement Unit (IMU) (Trigno Wireless System, Delsys, Inc.) was then positioned at the hand's back, and the angular velocity signal was recorded. The acquisitions were performed at 1925.93 Hz and 148.5 Hz sampling frequencies for sEMG and IMU respectively. The volunteers were asked to extend their arms forward perpendicular to the torso, with palms down as steady as possible. About one minute of sensors' data was recorded in each trial. Since sEMG data were collected for ECRL and FCU muscles, tremors in the flexion-extension displacement angle were selected for this study, which is the IMU's measured pitch.

2.2 Experimental Tremor Time Series Observation

The raw EMG signals may encompass noise attributable to ambient environmental factors, poor electrode contact, or the inherent noise from the equipment itself. This noise can adversely impact the performance of neural networks. Thus, it is necessary to mitigate this noise to the greatest extent possible and to render the signal smoother. Thus, several preprocessing steps on the raw kinematic data and EMG signals were carried out basis recommendations in literature[48], and outlined in Figure 2.1. The EMG signals were first passed through a high pass filter to remove the DC component of the voltage, and then through a low pass filter (500 Hz) to remove unusable high-frequency noise[40]. Zero-phase 5th-order infinite impulse response (IIR) filters were used to ensure that there is no delay in the filtered signal. A notch filter with 60 Hz was also used to reduce power line noise. The filtered signals are then rectified, and the envelope is obtained by computing the root mean square (RMS) value of the signal within a sliding window of 50 ms. The envelopes are a representation of the muscle activation level[16].

For the wrist flexion-extension data, a band-pass filter (1 Hz - 20 Hz) was used to remove

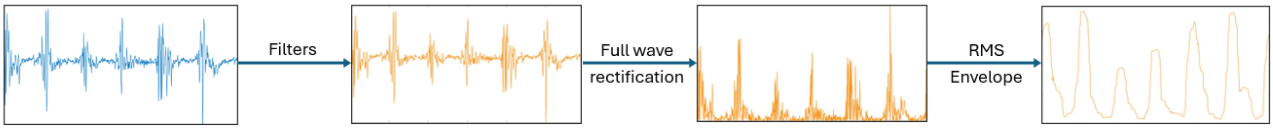


Figure 2.1: EMG signal processing steps

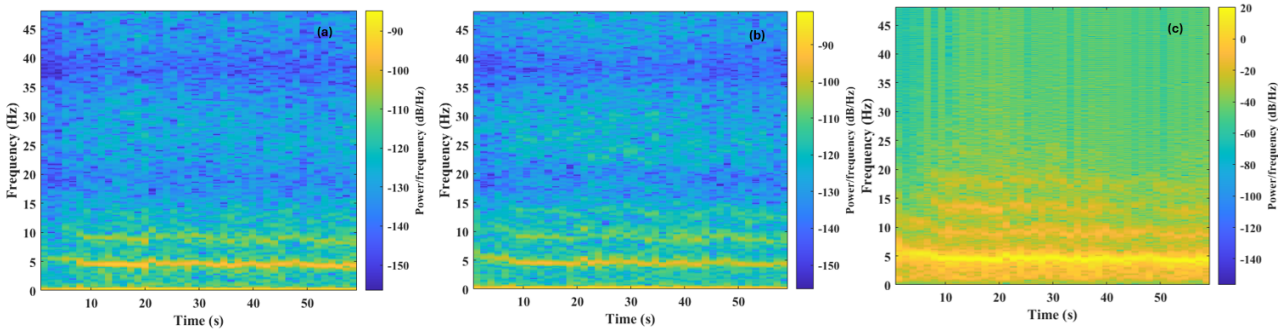


Figure 2.2: Spectrograms of Extensor EMG (a), Flexor EMG (b) and wrist flexion/deviation (c)

DC offsets and noise. Although voluntary movement components here are limited due to postural task execution, but when present, tremor can be separated from most voluntary movements through frequency-based filtering techniques [76]. All the preprocessed signals are then subsampled to 1000 Hz.

Figure 2.2 shows the spectrograms of the preprocessed joint angles and EMGs generated by short-time Fourier transformations. In general, we can observe that the frequencies of the dominant harmonic components have similar patterns for both EMGs and kinematics data. We also observe that over time, the amplitudes and frequencies of the harmonic components change, indicating that pathological tremor is a nonlinear dynamics problem, supporting findings in previous research.[81][75][17] The preprocessed data are then normalized to a 0-1 scale with the min-max feature scaler[55]; normalization has been found to aid in the improvement of the performance and training stability of deep learning models. The normalized data is then segmented into windows to be fed into the neural network models.

Each window is 1.1 seconds in length, of which the first 1 second is the input sequence and the next 0.1s (100 ms) is the prediction horizon. The first 80% of the dataset is used as training data, and the rest is test data. While training, 80% of the training data was used for training, and the rest for validation.

2.3 Neural Network approach

Different neural network architectures, using MLPs, LSTM and CNNs were created for this study. CNNs (Convolutional Neural Networks) are feedforward neural network which incorporate convolutional computation and are capable of learning spatial properties. The one-dimensional convolution operation for sequence data is shown as follows:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau \tag{2.3}$$

CNNs employ these convolutional layers that focus on continuous local regions of time series data. This helps them to capture short-term patterns effectively, which is crucial in time series forecasting.

RNN-based approaches have produced reliable results in sequence prediction challenges involving human motions, as shown in [13][71], thanks to their capacity to describe intricate temporal correlations in the input sequence data they receive. The typical RNN is unable to make reliable predictions when memorizing past lengthy sequences due to vanishing gradients[34]. LSTMs are able to solve this problem with 2 major changes; the presence of the forget gate, and the additive nature of the cell state gradients. This helps the network to update the parameters such that the gradients in the earlier layers do not converge to zero.

An LSTM's memory block is made up of one or more memory cells, and as shown in Fig. 2.3,

each memory cell shares 3 gates (input, forget and output gates) that control and protect the cell state. The input gate is in charge of managing input data contributions to the updating of the memory state. How much of which of the previous state contributes to the current state of the memory block is determined by the forget gate. Based on input and the block's memory, the output gate chooses what to transfer to the hidden state.

The memory cells in LSTM memory cells are calculated and updated each time step t basis the following equations:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f \tag{2.4a}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.4b}$$

$$\bar{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.4c}$$

$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t \tag{2.4d}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.4e}$$

$$h_t = o_t * tanh(C_t) \tag{2.4f}$$

where i_t, f_t, o_t and C_t represent the actions for the input gate, forget gate, output gate, and cell state updates at time t respectively. σ represents the sigmoid function, x_t is the input at the current timestamp, and h_{t-1} and h_t are the outputs at previous and current time-steps respectively. W_k and b_k are the weights and biases for the respective gates(k). The operations of various gating units based on the operations marked by Eqs 2.4 enables the LSTM to obtain the temporal properties of the input information when the sequence information is fed into the LSTM.

To leverage the benefits of both these structures, 2 hybrid CNN-LSTM models are also proposed. The first model is the parallel Y-CNN-LSTM network, as outlined in Figure 2.5. In this, the preprocessed EMG signals and the kinematics are fed into 2 parallel blocks,

a Convolution block and an LSTM Block. The Convolution and LSTM block consists of one or more hidden CNN and LSTM layers; the resultant tensors are then flattened and connected to a fully connected (FC) layer of 100 neurons. The outputs of the 2 blocks are concatenated, and connected to a FC layer of 100 units. The rationale behind such a structure is to combine the strengths of both LSTM and CNN using 2 parallel networks, leading to better performance. the combined In the Sequential CNN-LSTM network, the inputs are fed into the CNN layers, and the feature vectors obtained are then passed through LSTM layers. The resultant tensor is then flattened and connected to a FC layer. The advantage being that the model can support very long input sequences to extract features using the CNN layer and then send these features to the LSTM layer for further prediction. A feed-forward MLP with 2 hidden layers was created as a baseline model.

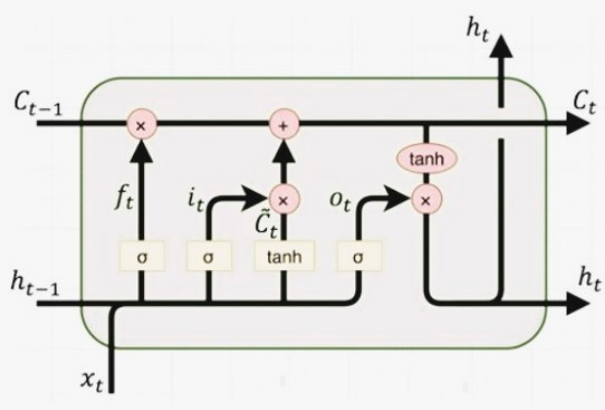


Figure 2.3: Structure of LSTM cell

The architectures used for evaluation in Table 2.1 are summarized as:

1. MLP: 2 fully connected hidden layers with 1000, 500 units
2. CNN 1: 1 hidden CNN layer of 32 filters
3. LSTM 1: 1 hidden LSTM layer of 32 units,
4. Seq CNN-LSTM 1: 1 CNN layer of 32 filters connected to a LSTM layer of 32 units

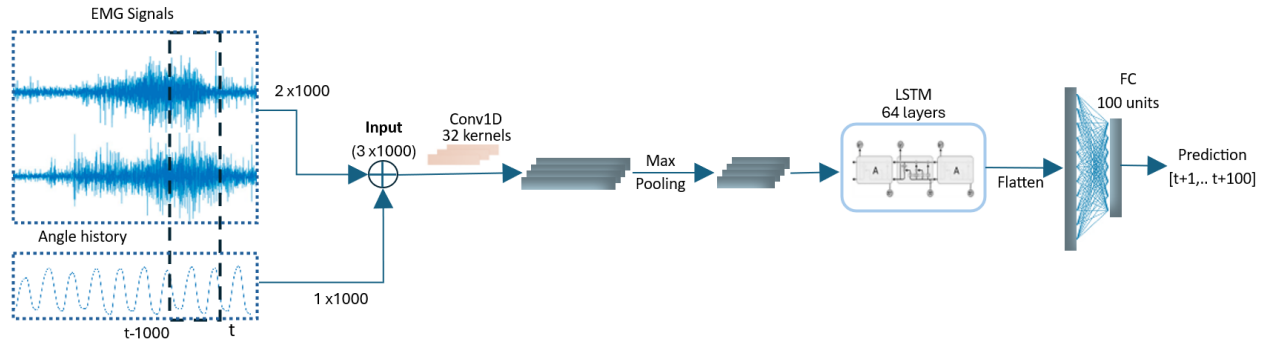


Figure 2.4: Architecture of the Sequential CNN-LSTM 1 Network as given in Table 2.1

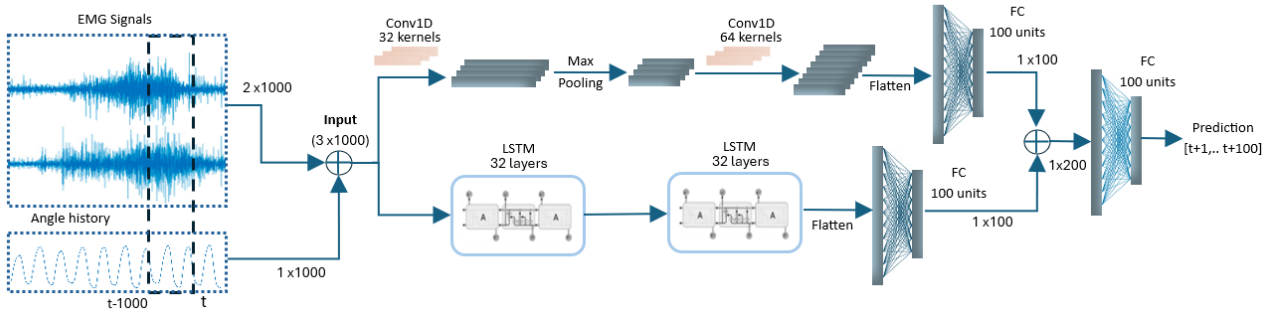


Figure 2.5: Architecture of the Parallel Y CNN-LSTM 2 Network as given in Table 2.1

5. Y CNN-LSTM 1: Parallel CNN and LSTM layers (32 units each)
6. CNN 2: 2 CNN layers of 32 and 64 filters
7. Seq CNN-LSTM 2: 2 CNN layer of 32 and 64 filters connected to first LSTM layer of 32 units, which is connected to 2nd LSTM layer with 32 units
8. Y CNN-LSTM 2: Parallel CNN (32 and 64 units) and LSTM (32 and 32 units) layers

The outer layer of each model is a dense layer of 100 units, to predict 100 timesteps (100 ms) of data. All the CNN layers are Conv1D layers, connected to a batch normalization layer[36] and MaxPooling layer [28] to reduce internal covariance shift and add regularization respectively. More details of these models are explained in Appendix A. All the models were

created in Tensorflow V2.15 environment.

2.3.1 Evaluation Criteria

The prediction output from the neural network is collected and to verify the effectiveness and we use different metrics to quantify the models' performances. The first metric is the root mean square error (RMSE), which is often used for time series forecasting problems and can be calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{T}\Sigma_{t=1}^{T}(y_t - \hat{y}_t)^2} \tag{2.5}$$

where y_t and $\hat{y}_t$ indicate the true value and the corresponding predicted value respectively. The closer the RMSE term is to 0, the better the fit.

Pearson's correlation coefficient (PC) is another metric that is employed, and it can be calculated by:

$$PC = \frac{\sum_{t=1}^{T}(y_t - \bar{y}_t)(\hat{y}_t - \bar{\hat{y}}_t)}{\sqrt{\sum_{t=1}^{T}(y_t - \bar{y}_t))^2}\sqrt{\sum_{t=1}^{T}(\hat{y}_t - \bar{\hat{y}}_t)^2}} \tag{2.6}$$

where $\bar{y}_t$ and $\bar{\hat{y}}_t$ are the mean of the ground truth and the predicted value, respectively. The Pearson correlation measures the strength of the linear relationship between two variables, with the value varying between -1 to 1. The closer the value to 1, more the positive correlation; while closer to -1 indicates a negative correlation, and 0 being no correlation.

Finally, R-squared metric is also employed and is given as:

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{T}(y_t - \bar{y}_t)^2} \tag{2.7}$$

The R-Square metric ranges from 0 to 1, with a larger value indicating a better fit between prediction and actual value.

2.4 Results

In this section, the performance of the different frameworks are verified on wrist F/E angle predictions on the dataset. First, the developed models' training procedure is described. The prediction results of the developed models are then shown through overall comparisons of all the models, including representations of the predicted F/E joint angles. Effect in performance across different input sequences and prediction horizon are also studied. Finally, the effects of changing input and prediction windows, and hyper-parameters on performance are investigated.

2.4.1 Training Process

For training, the MSE loss function was considered, which is defined as:

$$Loss_{mse} = \frac{1}{T}\Sigma_{t=1}^{T}(y_t - \hat{y}_t)^2 \tag{2.8}$$

Here, y_t and $\hat{y}_t$ are the truth and predictions of the normalized F/E angle. The training was set to 100 epochs with a batch size of 64. An Early Stopping functionality was also included, which stops the training if there is no decrease in the validation loss after 7 epochs. This is implemented to stop the model from overfitting unto the training data [61]. The training of the models are carried out using Tensorflow on a workstation with 12th Gen Intel(R) Core(TM) i7-12650H (16 CPUs), 2.3GHz processor, GeForce RTX 3060 graphic cards and 16G RAM. Figure 2.6 demonstrates the convergence of the model, as we see both training loss and validation loss initially decrease and finally reach a minima after certain epochs.

Table 2.1: Performance of different architectures across subjects for the Pinheiro tremor dataset; with the 3 best models highlighted for subjects 1, 4 and 5

Subject	Model	RMSE	R-Squared	PC
S1	MLP	0.090	0.772	0.823
	CNN 1	**0.057**	**0.907**	**0.888**
	LSTM 1	**0.060**	**0.918**	**0.934**
	Seq CNN-LSTM 1	**0.056**	**0.911**	**0.875**
	Y CNN-LSTM 1	0.083	0.804	0.813
	CNN 2	0.062	0.898	0.855
	Seq CNN-LSTM 2	0.080	0.804	0.813
	Y CNN-LSTM 2	0.079	0.828	0.832
S3	MLP	0.11	0.376	0.486
	CNN 1	0.127	0.356	0.473
	LSTM 1	0.098	0.43	0.62
	Seq CNN-LSTM 1	0.168	0.356	0.473
	Y CNN-LSTM 1	0.107	0.554	0.570
	CNN 2	0.126	0.310	0.499
	Seq CNN-LSTM 2	0.132	0.377	0.481
	Y CNN-LSTM 2	0.128	0.356	0.445
S4	MLP	0.077	0.932	0.964
	CNN 1	0.045	0.978	0.983
	LSTM 1	0.042	0.978	0.989
	Seq CNN-LSTM 1	**0.035**	**0.986**	**0.994**
	Y CNN-LSTM 1	0.042	0.980	0.985
	CNN 2	0.044	0.975	0.983
	Seq CNN-LSTM 2	**0.042**	**0.980**	**0.992**
	Y CNN-LSTM 2	**0.038**	**0.983**	**0.989**
S5	MLP	0.067	0.832	0.865
	CNN 1	0.036	0.942	0.941
	LSTM 1	0.037	0.944	0.951
	Seq CNN-LSTM 1	**0.029**	**0.959**	**0.962**
	Y CNN-LSTM 1	0.044	0.881	0.931
	CNN 2	0.052	0.880	0.891
	Seq CNN-LSTM 2	**0.007**	**0.997**	**0.988**
	Y CNN-LSTM 2	**0.011**	**0.995**	**0.980**

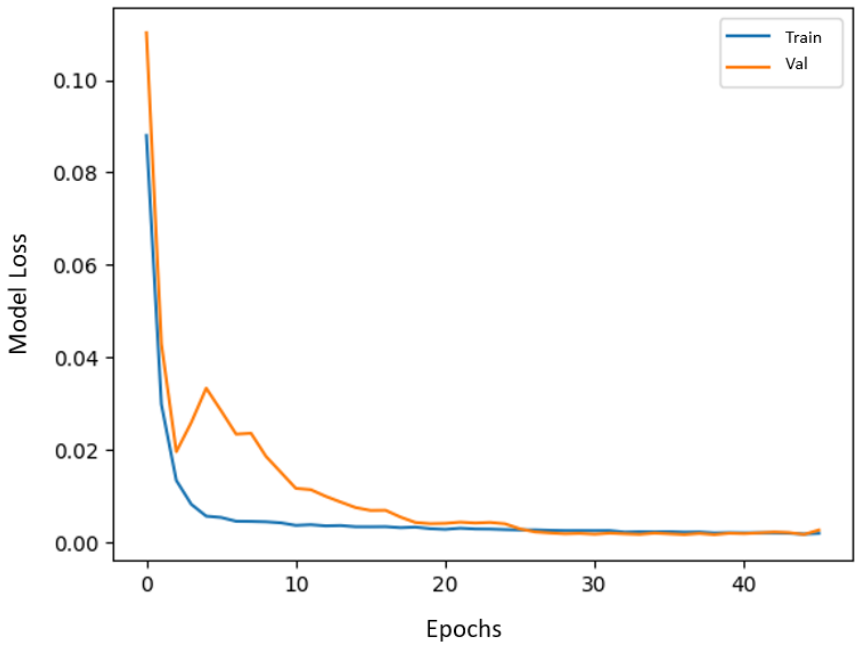


Figure 2.6: Illustration of the MSE loss over time/epochs

2.4.2 Comparison across Network Architectures

To quantitatively evaluate the performance of the different architectures, detailed comparisons are first performed across 4 Parkinson's subjects' datasets, as summarized in Tables 2.1. The metrics presented are evaluated on the test data after training was completed. Figure 2.7 shows the predictions using the proposed hybrid CNN-LSTM models. As expected, all the deep learning models performed significantly better than the baseline MLP architecture, which corroborates the fact that high-level features can be automatically extracted using deep learning-based methods. Amongst the deep learning architectures, Sequential CNN-LSTM models performed better for most cases (except subject 3, for which none of the models performed well), with the RMSE well below 0.05 in most cases. Since our data was normalized to a 0 to 1 scale, this effectively means less than 5% error. The Y CNN-LSTM, in overall performed almost similarly or slightly better than the standard LSTM model, despite having more parameters to train. It can also be observed that increasing the depth

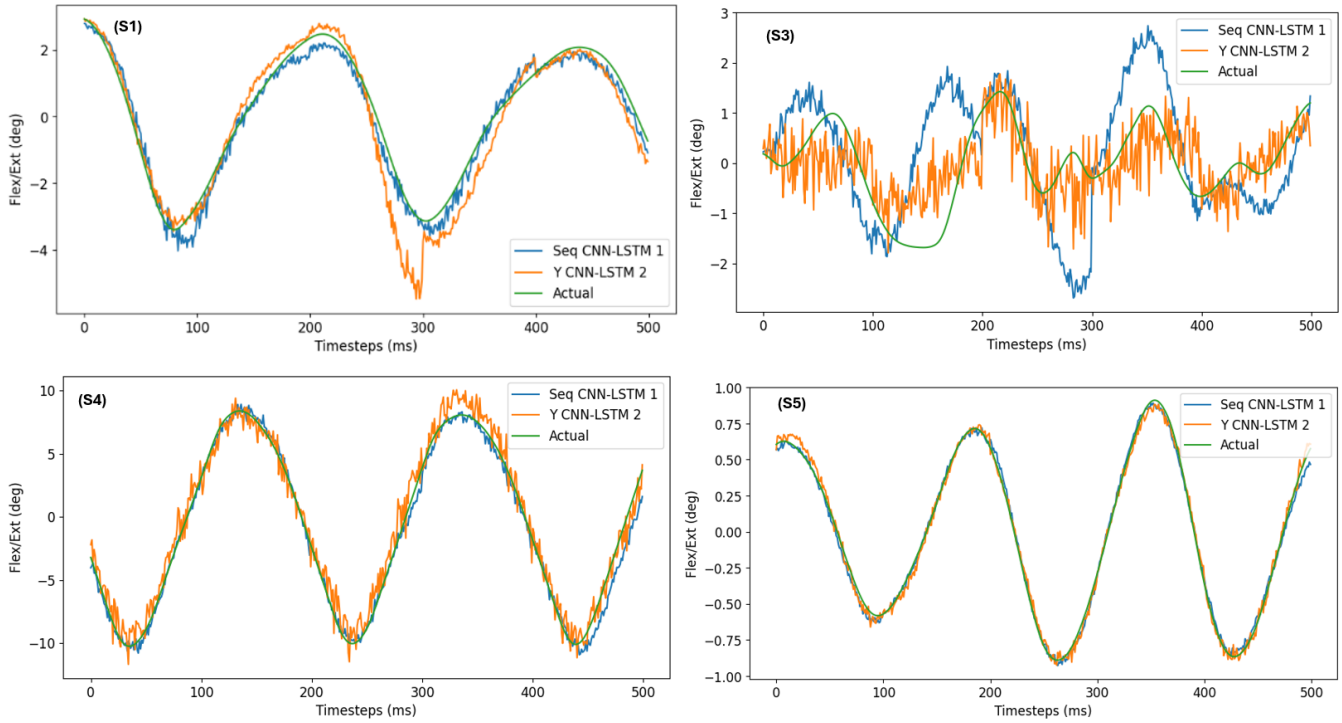


Figure 2.7: Prediction results for the Seq CNN-LSTM 1 and Y CNN-LSTM 2 models across the 4 subjects

of the networks did not necessarily improve the performances. With the trained Sequential CNN-LSTM 2 model, the prediction for 100 ms of data took only about 0.3 ms on the same workstation, and thus could be used for real-time operations. The sequential CNN-LSTM 1 model was used to study the effects of input and prediction horizon, and hyperparameters on performance in sections 2.4.3 and 2.4.4.

2.4.3 Effects of Input and Prediction horizon

We also evaluate the proposed model by varying the length of the input and prediction horizon, as detailed in Table 2.2. It was observed that the performance of the model remained quite good up to 100 ms of prediction horizon (<5% error). However, upon further increase,

Table 2.2: Comparison with different Input and Prediction Horizon

Input	Prediction	RMSE	R-Squared	PC
1000	40	0.025	0.985	0.941
1000	80	0.032	0.948	0.934
1000	100	0.040	0.952	0.944
1000	150	0.072	0.803	0.867
1500	150	0,072	0.809	0.881
1000	200	0.077	0.782	0.864
1000	400	0.096	0.664	0.824
2000	400	0.122	0.510	0.729

Table 2.3: Comparison with different Activation Functions

Activation Function	RMSE	R-Squared	PC
Tanh	0.040	0.952	0.944
ReLu	0.041	0.946	0.944
Leaky-ReLu	0.038	0.957	0.932
Sigmoid	0.084	0.868	0.922

the performance reduced drastically, and even increasing the input sequence length did not improve the performance.

2.4.4 Effects of Hyperparameters

In this section, we investigate the effects of hyperparameters on performance, i.e. different learning rates and different activation functions on the proposed Sequential CNN-LSTM 1 framework. The detailed results are shown in Table 2.4 and Table 2.3. The metrics shown are the average values obtained over S1, S4, and S5. Specifically, 3 learning rates are considered, i.e., 0.01, 0.001, and 0.0001. We can observe from table 2.4 that better performance was achieved with smaller learning rates (0.001 or 0.0001). 4 activation functions were tested, tanh, ReLU, Leaky ReLU and Sigmoid, all of which are non-linear activation functions. Observed from Table 2.3, it can be seen that tanh, ReLU and Leaky ReLU activation functions

Table 2.4: Comparison with different Learning Rate

Learning Rate	RMSE	R-Squared	PC
0.01	0.110	0.23	0.3876
0.001	0.042	0.942	0.914
0.0001	0.040	0.952	0.944

achieved similar performance and better than the sigmoid activation function.

2.5 Adding constraints to training

As observed from Figure 2.7, while the predictions are close to the actual values most of the time, the predictions are still quite noisy. While methods like exponential moving averages or Kalman Filters can be implemented in real-time, we try to explore if we can modify our model in such a way that it can learn to predict smoother outputs. Inspired by Physics-informed neural networks, where the loss functions are modified to accommodate physics-based constraints and losses, we devise loss functions based on the first-order and second-order derivatives of the actual F/E angular data. The rationale being that the 1st and 2nd derivatives give info on the continuity and smoothness, and adding these losses would help to embed the same during training.

The proposed constraint loss functions can be given as:

$$Loss_1 = \frac{1}{T}\Sigma_{t=1}^{T}((y_{t+1} - y_t) - (\hat{y}_{t+1} - \hat{y}_t))^2 \tag{2.9}$$

$$Loss_2 = \frac{1}{T}\Sigma_{t=1}^{T}((y_{t+2} + y_t - 2y_{t+1}) - (\hat{y}_{t+2} + \hat{y}_t - 2\hat{y}_t))^2 \tag{2.10}$$

Table 2.5: Comparison with adding constraint loss terms

Loss	γ_1	γ_2	RMSE	R^2	PC
MSE	0	0	0.040	0.952	0.944
MSE + Loss_1	0.01	0	0.041	0.929	0.945
	0.1	0	0.040	0.940	0.967
	1	0	0.046	0.920	0.946
Total Loss	0.1	0.01	0.042	0.925	0.963
	0.1	0.1	0.038	0.936	0.947

The total combined loss function is then given as:

$$\text{Total Loss} = Loss_{mse} + \gamma_1 Loss_1 + +\gamma_2 Loss_2 \tag{2.11}$$

From the Table 2.5, it can be observed that although some in terms of metrics, there is very slight improvement, especially in terms of correlation coefficient (PC). However, increasing γ_1 too high affects the performance negatively. Further, from 2.8, it can be observed that when the model is trained with Total Loss, the predictions are much smoother.

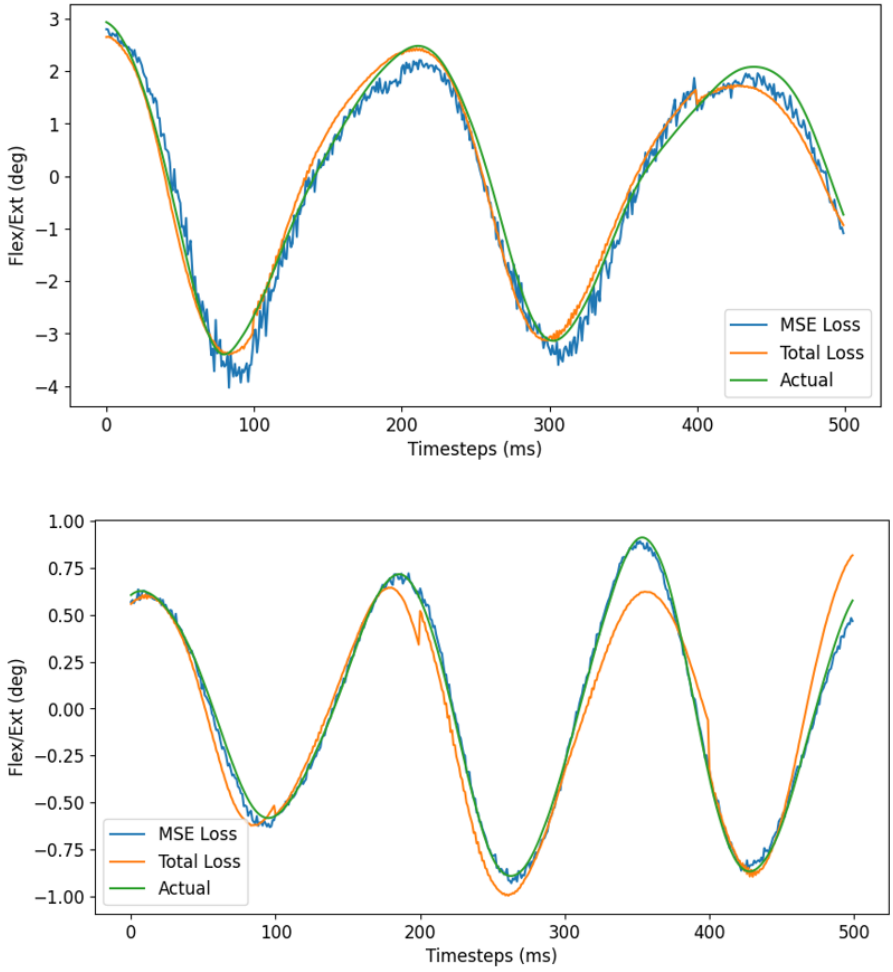


Figure 2.8: Prediction results for Subject 1 (top) and Subject 5 (bottom); after Seq CNN-LSTM 1 is trained using MSE Loss and Total Loss

Chapter 3

Control System framework

3.1 Dynamical Model and Linearization

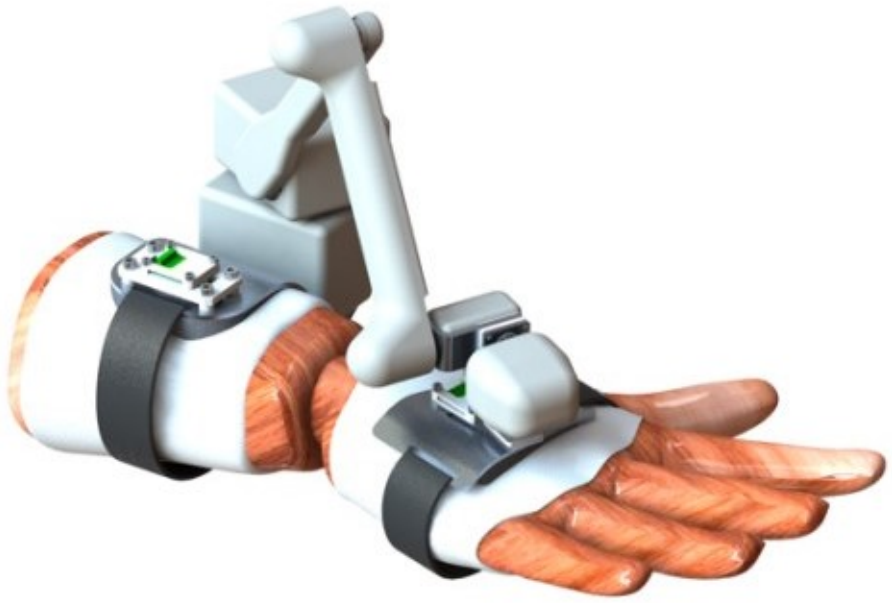


Figure 3.1: The conceptual design of TAWE attached to a right human forearm mannequin ([80])

3.1.1 Background: Human-Exoskeleton Multibody Dynamics

This section provides a brief background on the exoskeleton (TAWE) designed previously in our lab and the coupled dynamics of TAWE and the wrist via a generalized multibody

framework; this model will be used for the development of the control system in further sections. The multi-body system equations can be written as:

$$M_1\ddot{q}_1 = -C_1\dot{q}_1 - h_1 + J_{u,1}^{\mathrm{T}}u_1 + J_{w,1}^{\mathrm{T}}w_1 + J_{\lambda,1}^{\mathrm{T}}\lambda \tag{3.1a}$$

$$M_2\ddot{q}_2 = -C_2\dot{q}_2 - h_2 + J_{u,2}^{\mathrm{T}}u_2 + J_{w,2}^{\mathrm{T}}w_2 + J_{\lambda,2}^{\mathrm{T}}\lambda \tag{3.1b}$$

where "1" and "2" are labels of the human and exoskeleton subsystems, respectively denote the arm and exoskeleton subsystems respectively. q are the generalized coordinates; u are the generalized control input; M, C, and h are the inertia matrix, Coriolis and centripetal matrix, and generalized force vector respectively, while w is the system disturbance. The Jacobian matrices J_u and J_w are Jacobian matrices corresponding to u and w respectively. Lastly, λ is the Lagrange Multiplier which serves as a constraint between the exoskeleton and human subsystems. The direction of the constraint forces is indicated by the Jacobian matrix J_λ.

The human arm is not a rigid body system; but for this study, a few assumptions are made:

(A1) The forearm is approximated as a rigid body model, with the mass and inertia properties known and available, and the muscles' deformation is non-existent.

(A2) Muscle actuation forces are generalized into direct torque inputs at the joint directions.

The 2 subsystems in total have 21 degrees of freedom, broken down as: 6 DOF for base forearm ($q_{1,base}$), 3 DOF for the wrist joint ($q_{1,wrist} = [\theta_{\mathrm{FE}} \quad \theta_{\mathrm{RUD}} \quad \theta_{\mathrm{SP}}]^T$), 6 DOF for the exoskeleton base ($q_{2,exo}$), and 6 DOF for the exoskeleton joints ($q_{2,joints}$). In figure 3.2, they are referred to as A1, A2, E1 and (L1 to L6) respectively.

To reduce the number of degrees of freedom, we consider another assumption: (A3) The forearm motion is fixed in this study. Further, as the base of the exoskeleton is attached close to

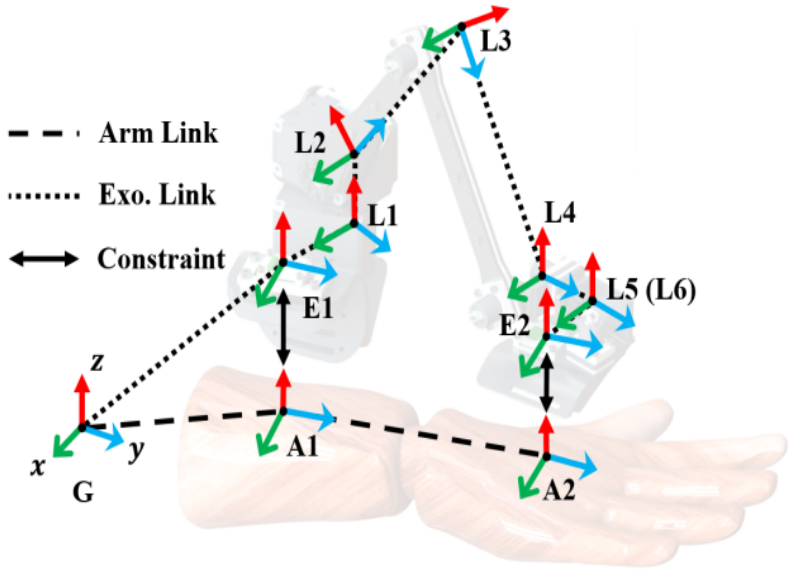


Figure 3.2: The kinematics of the forearm and exoskeleton system[79]

the tip of the forearm, the orientation difference is not affected by the supination/pronation in the forearm, which makes the supination motion of the forearm fixed in this study.

Assumption (A3) reduces 7 DOFs from the system (6 base forearm and Wrist P/S). Further, the two subsystems are coupled by the kinematic constraints which constrains the exoskeleton motion to the wrist motion. The base of the exoskeleton (E1) is constrained to the forearm base (A1), and the last exoskeleton linkage (L6) is constrained to the wrist dorsum (A2); which in total constrains 12 DOFs. The resulting 2 DOF system can now be controlled by 2 actuators, given at the first two joints of TAWE.

The constraint equation is defined as r_λ:

$$\dot{r}_\lambda = 0 = J_{\lambda,q}(q,\rho)\dot{q} + J_{\lambda,\rho}(q,\rho)\dot{\rho}; \qquad \rho = [q_{2,base}^T \quad q_{2,exo}^T]^T \tag{3.2}$$

where ρ consists of the dependent states, and q consists of the independent states (q =

$[\theta_{\text{FE}} \quad \theta_{\text{RUD}}]^T$). The dynamics of ρ can be obtained as

$$\dot{\rho} = -J_{\lambda,\rho}^{-1} J_{\lambda,q} \dot{q}; \quad \ddot{\rho} = -J_{\lambda,\rho}^{-1} \left(J_{\lambda,q} \ddot{q} + \dot{J}_{\lambda,q} \dot{q} - \dot{J}_{\lambda,\rho} J_{\lambda,\rho}^{-1} J_{\lambda,q} \dot{q} \right) \tag{3.3}$$

The 2 subsystems can be coupled to a single multibody equation for the human-exoskeleton system using recursive dynamics and Kane's method [79][72] and can written as:

$$M\ddot{q} = -C\dot{q} - h + J_u^{\mathrm{T}} u + J_w^{\mathrm{T}} w \tag{3.4}$$

where:

$$J_c = \begin{bmatrix} I_2 \\ -J_{\lambda,\rho}^{-1} J_{\lambda,q} \end{bmatrix}; \; M = J_c^{\mathrm{T}} \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} J_c; \; C = J_c^{\mathrm{T}} \begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix} J_c + J_c^{\mathrm{T}} \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \dot{J}_c;$$
$$h = J_c^{\mathrm{T}} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}; \; J_u = J_c^{\mathrm{T}} \begin{bmatrix} J_{u,1}^{\mathrm{T}} & 0 \\ 0 & J_{u,2}^{\mathrm{T}} \end{bmatrix}; \; J_w = J_c^{\mathrm{T}} \begin{bmatrix} J_{w,1}^{\mathrm{T}} & 0 \\ 0 & J_{w,2}^{\mathrm{T}} \end{bmatrix}; \; w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}; \; u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.5}$$

3.1.2 Model Linearization

The coupled hand-exoskeleton dynamical system is quite complex and nonlinear. In recent times, nonlinear model predictive control (NMPC) ([2]) methods for robotic problems have also been well developed ([90],[78]). However, it has also been noted that the amount of computing work needed is much greater than with linear MPC. Nonlinear MPC necessitates the online solution of a nonlinear programming problem, which is non-convex, consists of more decision variables, and has a global minima that is typically impossible to locate([33],[43]). Therefore, in our study, the hand-exoskeleton dynamical model is successively linearized to get a linear, time-varying dynamical model. We aim to obtain the optimal inputs that

minimize the L2 norm of the tracking error, making the control inputs are our decision variables. Since we are using L2 norm, the optimization problem becomes a convex quadratic programming (QP) problem, which is solved by numerically robust solvers to obtain global optimal solutions.

The hand-exoskeleton dynamical model in the equation 3.4 is used for formulation. For simplicity, we assume that the system disturbance w is zero. Using this equation, we define the state as $x = \text{col}(q, \dot{q})$, and therefore, we can write the state equation as:

$$\dot{x} = \begin{bmatrix} \dot{q} \\ M^{-1}(-C\dot{q} - h + J_u^T u) \end{bmatrix} \tag{3.6}$$

which is a non-linear equation of the form: $\dot{x} = f(x, u)$

By taking the Taylor expansion and discarding the higher order terms, our system can be linearized about the current operating point (x_r, u_r) as:

$$\dot{x} = f(x, u) = f(x_r, u_r) + f_{x,r}(x - x_r) + f_{u,r}(u - u_r) \tag{3.7}$$

where

$$f_{x,r} = \left.\frac{\partial f}{\partial x}\right|_{\substack{x=x_r \\ u=u_r}} \quad \text{and} \quad f_{u,r} = \left.\frac{\partial f}{\partial u}\right|_{\substack{x=x_r \\ u=u_r}} \tag{3.8}$$

Using Euler's method, for sample time Δt, Eqn. (3.7) can be converted to discrete-time form as:

$$x(t+1) = x(t) + f(x_r, u_r)\Delta t + f_{x,r}(x(t) - x_r)\Delta t + f_{u,r}(u(t) - u_r)\Delta t \tag{3.9a}$$

$$\implies x(t+1) = (I + f_{x,r}\Delta t)x(t) + (f_{u,r}\Delta t)u(t) + (f(x_r, u_r) - f_{x,r}x_r - f_{u,r}u_r)\Delta t \tag{3.9b}$$

The functions here, $f_{x,r}$ and $f_{u,r}$ are Jacobian matrices and can be obtained as:

$$f_{x,r} = \begin{bmatrix} 0 & I_{n_q} \\ A1 & A2 \end{bmatrix} \qquad f_{u,r} = \begin{bmatrix} 0 \\ B \end{bmatrix}$$

where

$$\begin{aligned} A1 &= \frac{\partial}{\partial q} M^{-1}(-C\dot{q} - h + J_u^T u) \\ &= \frac{\partial M^{-1}}{\partial q}(-C\dot{q} - h + J_u^T u) \\ &\quad + M^{-1}(-\frac{\partial C}{\partial q}(\dot{q}) - \frac{\partial h}{\partial q} + \frac{\partial J_u^T}{\partial q}(u)) \end{aligned} \tag{3.10a}$$

$$\begin{aligned} A2 &= \frac{\partial}{\partial \dot{q}} \left(M^{-1} \left(-C\dot{q} - h + J_u^T u \right) \right) \\ &= M^{-1} \left(-\frac{\partial C\dot{q}}{\partial \dot{q}} - \frac{\partial h}{\partial \dot{q}} \right) \end{aligned} \tag{3.10b}$$

$$B = \frac{\partial}{\partial u} M^{-1}(-C\dot{q} - h + J_u^T u) = M^{-1} J_u^T \tag{3.10c}$$

The ANDY symbolic toolbox developed in MATLAB in our lab [83] has been used in the modeling of the system. The process requires calculating $\frac{\partial M^{-1}}{\partial q}$; and obtaining M^{-1} first symbolically and then calculating $\frac{\partial M^{-1}}{\partial q}$ very expensive computationally. Hence instead, we calculate it as:

$$\frac{\partial M^{-1}}{\partial q} = \frac{\partial (M^{-1} \dot{M} M^{-1})}{\partial \dot{q}} \tag{3.11}$$

where M^{-1} is calculated numerically.

3.2 Model Predictive Control

Optimizing predictions of a system's behavior over a series of future control inputs is the fundamental essence of a model predictive controller. The plant is subjected to the first

element of an optimal control sequence that is produced by solving a minimization function at each sample time. The best control sequence is obtained by resolving the problem at the subsequent sampling time using the updated system state([43]).

3.2.1 MPC Formulation

The first objective of our system here is trajectory tracking; i.e. when a reference trajectory x^{des} is provided, the controller needs to obtain the optimal actuation torques basis the dynamics and defined cost function, and these torques are then applied to the human-exoskeleton system to accurately follow the set reference trajectory.

Taking the control horizon as N, the cost function is defined as:

$$\mathcal{J}_{0\to N}(x_0, U_{0\to N}) := \|x_N - x_N^{des}\|_P^2 + \sum_{k=0}^{N-1}(\|x_k - x_k^{des}\|_Q^2 + \|u_k\|_R^2) \tag{3.12}$$

subject to the following constraints based on the system dynamics and input and control constraints:

$$x_{k+1} = Ax_k + Bu_k + K \qquad \text{for k = 0 to N-1} \tag{3.13a}$$

$$X_l \leq x_k \leq X_u \qquad \text{for k = 0 to N-1} \tag{3.13b}$$

$$U_l \leq u_k \leq U_u \qquad \text{for k = 0 to N-1} \tag{3.13c}$$

Here, the matrices $P = P^T \succ 0$, $Q = Q^T \succ 0$ and $R = R^T \succ 0$ are all the cost matrices corresponding to the terminal and non-terminal states, and inputs respectively. Equation (3.13a) is just a simpler version of (3.9b), representing the state equation while (3.13b), (3.13c) are the state and input constraints.

This cost function can now be rewritten as:

$$Z_{0\to N}^{T}\begin{bmatrix}\bar{Q} & 0\\ 0 & \bar{R}\end{bmatrix}Z_{0\to N}+gZ_{0\to N} \tag{3.14}$$

where

$$Z_{0\to N}=\mathrm{col}(x_1,...x_N,u_0,...u_{N-1})$$

$$\bar{Q}=\mathrm{diag}(Q,Q...,P)$$

$$\bar{R}=\mathrm{diag}(R,...,R)$$

$$g=[-2(x_1^{des})^TQ,-2(x_2^{des})^TQ,...-2(x_N^{des})^TP,0,0...0]$$

The overall state state and input constraint matrices for horizon N can be written as:

$$\begin{bmatrix}I_{n_x} & 0 & \dots & 0 & 0 & -B & 0 & \dots & 0\\ -A & I_{n_x} & \dots & 0 & 0 & 0 & -B & \dots & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & -A & I_{n_x} & 0 & 0 & \dots & -B\end{bmatrix}[Z_{0\to N}]=\begin{bmatrix}Ax_0+K\\ K\\ \vdots\\ K\end{bmatrix} \tag{3.15a}$$

$$\begin{bmatrix}I_{2\mathrm{N}\times(n_x+n_u)}\\ -I_{2\mathrm{N}\times(n_x+n_u)}\end{bmatrix}[Z_{0\to N}]\leq\begin{bmatrix}(X_u)_{N\times1}\\ (U_u)_{N\times1}\\ (-X_l)_{N\times1}\\ (-U_l)_{N\times1}\end{bmatrix} \tag{3.15b}$$

Our cost function is quadratic, as observed in (3.14); and a variety of available quadratic solvers can be used to obtain $Z_{0\to N}$ which minimizes our cost function. The qpSWIFT solver

has been used in our studies to address this quadratic optimization problem because it has been shown to be faster than the majority of other solvers ([58]).

3.2.2 Trajectory tracking

This section shows how well the developed model predictive controller performs when tracking a specific trajectory using simulations of the forearm model and wearable wrist exoskeleton. The simulations are run using the ANDY Toolbox, and the 3D visualization of the simulation is displayed in Fig. 3.3. We also consider another assumption in this study:

(A4) The user doesn't provide any voluntary intent, while the intended motion or the tracking reference is known. This essentially means that the exoskeleton drives the entire hand-exoskeleton system.

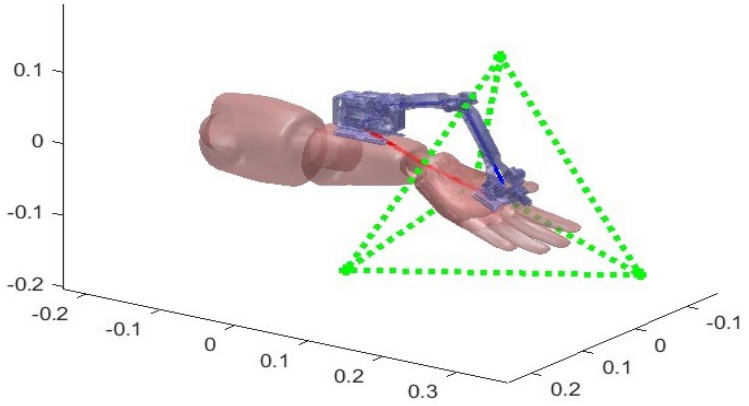


Figure 3.3: The exoskeleton simulation environment using ANDY Toolbox in MATLAB

For the simulations, the tracking references r are considered to be a combination of bounded periodic and quasi-periodic trajectories of different harmonic components ([82]) of low frequencies and generated using a script. The simulation and control sampling rate is taken as 200 Hz.

The cost matrices defined in Section 3.2.1 are considered as: $P = Q = \text{diag}(200, 1000, 0.1, 0.1)$ and $R = 0.4I_2$. The control horizon N is considered to be 20.

First, this controller was tested on a forearm (wrist) model, with the MPC algorithm implemented to obtain the optimal control torques in the Radial-Ulnar Deviation (RUD) and Flexion-Extension (FE) directions to follow the provided set trajectory.

The torques obtained are applied at the wrist joint; the state and input constraints were taken into account based on the average wrist joint strength and range of motion. According to ([18],[93]), on average the torque limits for flexion and extension were 12 and 7 Nm, respectively, while the limits for RUD were 11 and 9.5 Nm in the radial and ulnar directions, respectively.

In terms of the range of motion, the limits are $-45°$ to $25°$ (-0.79 rad to 0.44 rad) in the RUD direction and $-75°$ to $75°$ (-1.31 rad to 1.31 rad) in the FE direction ([79]). A study ([66]) of the arm's kinematics and dynamics in a variety of daily tasks found that the maximum angular velocities were 141.2 deg/s (2.46 rad/s) while flexing and 232.9 deg/s (4.06 rad/s) while extending. Similarly, it was found that the limit for ulnar and radial deviations were 180.4 deg/s (3.14 rad/s) and 203.9 deg/s (3.56 rad/s), respectively. Maintaining some buffer, the limits were kept at 270 deg/s ($3\pi/2$ rad/s) for both FE and RUD directions.

Figure 3.4 shows one example of the tracking performance. Figures 3.4(a) and 3.4(b) show that the trajectory of q is rather accurately following the set references, with the deviations in 3.4(c) and 3.4(d) being around zeros, indicating the controller's effectiveness.

The experiment with the forearm model example helps us to observe the controller's suitability and performance. The MPC Controller is then tested with the wearable exoskeleton (TAWE).

As described in section 3.1.1, TAWE constitutes a 6-DOF rigid linkage system which allows

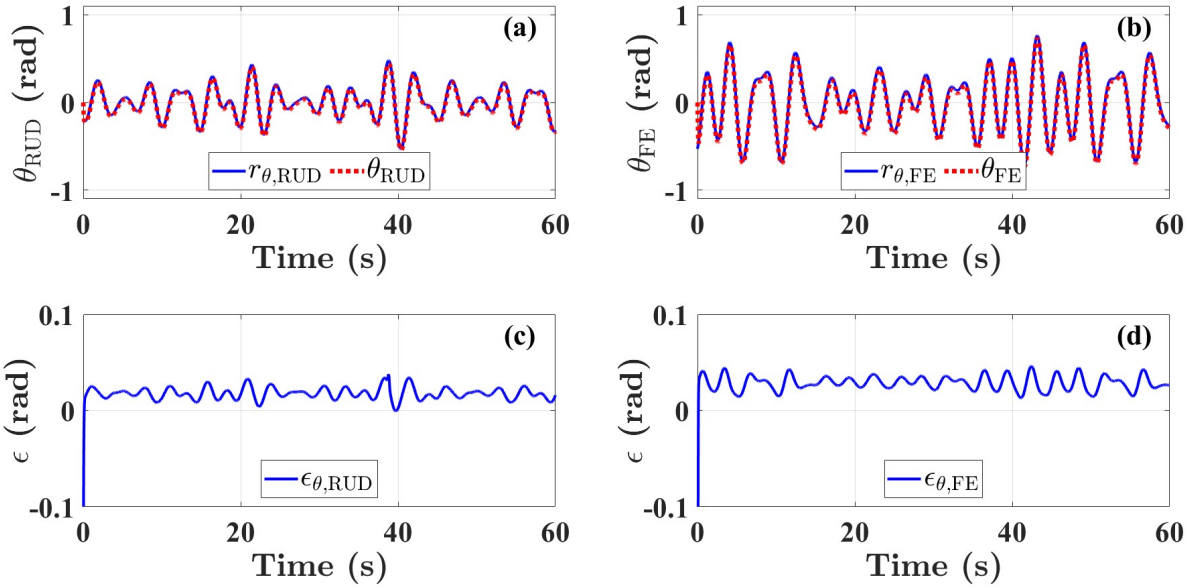


Figure 3.4: The control system performance when applied only on forearm for trajectory tracking, where (a,b) compares the trajectories followed in RUD and FE (θ) with reference (r_θ); (c-d) shows the tracking errors ($\epsilon = r_\theta - \theta$)

unconstrained wrist movement, while its kinematics is defined by the biomechanics of the wrist. For our studies, the forearm is considered fixed, and there is no deviation in the pronation-supination direction. The forearm and the exoskeleton form a closed kinematic chain, resulting the exoskeleton being fully constrained to the wrist, which constrains all but 2 DOFs of the system in total, and a 2-DOF assembled dynamical model can be obtained. The control torques are given at the first 2 joints of the exoskeleton. For the MPC controller, the weighing matrices and control horizon are kept to be the same.

First, as illustrated in Fig. 3.5, the tracking performance is evaluated. Figure 3.5 shows that the TAWE-Forearm model's trajectory closely resembles the Forearm model's trajectory, which means it closely follows the set trajectory as well. Figure 3.6 shows the control torques acquired for the trajectory tracking problem with the forearm-exoskeleton system.

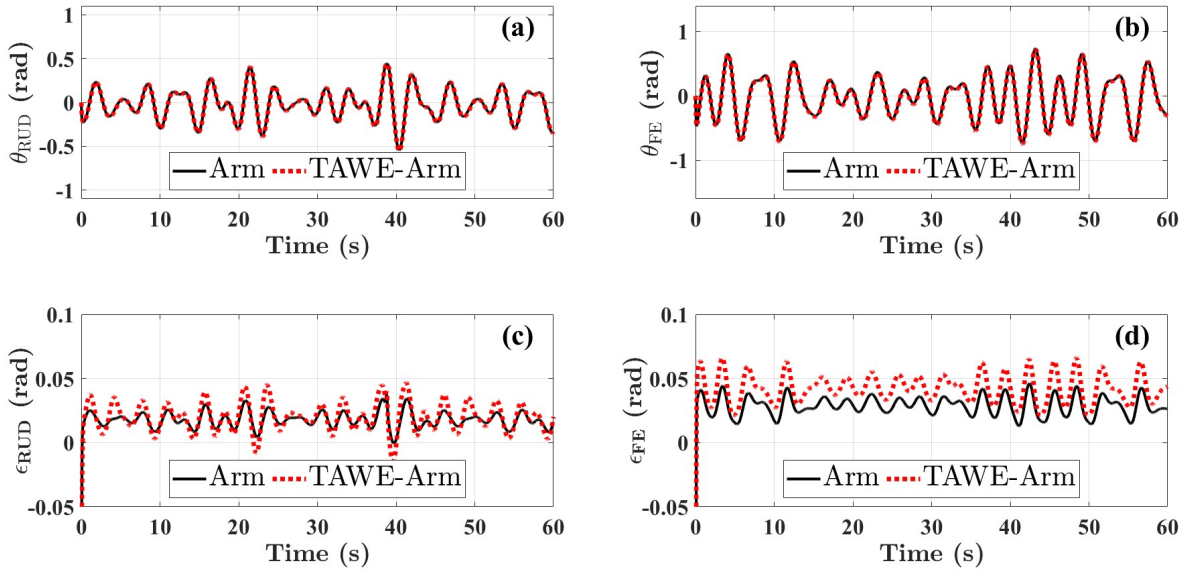


Figure 3.5: Comparing the control system performance when applied to Forearm-TAWE (red) and Forearm (black) model for trajectory tracking, where (a, b) show the followed trajectories (θ) and (c,d) shows the tracking errors ($\epsilon = r_\theta - \theta$)

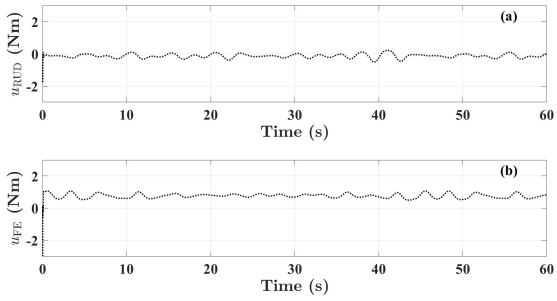


Figure 3.6: The obtained input torques for trajectory tracking with TAWE; where (a) and (b) show the control inputs in RUD and FE respectively

3.2.3 Tremor Suppression Without tremor model

In the above section, we discussed on the effectiveness of the MPC Controller for the trajectory tracking goal. In this section, we discuss if the controller can suppress tremors when

they are introduced, without using any model for the tremor. To perform this, tremors are added to the system as a model uncertainty. The synthetic tremor excitation introduced to the model consists of harmonic waves with various frequencies between 3-6 Hz, and the forearm is aimed to be kept stationary with the palm facing down parallel to the ground initially, i.e. the set reference to follow is kept to be 0 in both FE and RUD directions.

A PD controller with a feed-forward term is used to compare the MPC Controller's outcomes. The PD part of the controller pushes the system back to [0,0] as the tremors operate to move it away from [0,0], while the feed-forward input, which is constant, is provided to compensate for the torque caused by gravity. Figure 3.7 demonstrates that the deviation is very big when the PD controller is applied. But Figures 3.8(a) and 3.8(b) demonstrate that the erroneous oscillation amplitudes are greatly reduced when MPC is applied. This suggests that MPC can suppress tremors as well without using any tremor model.

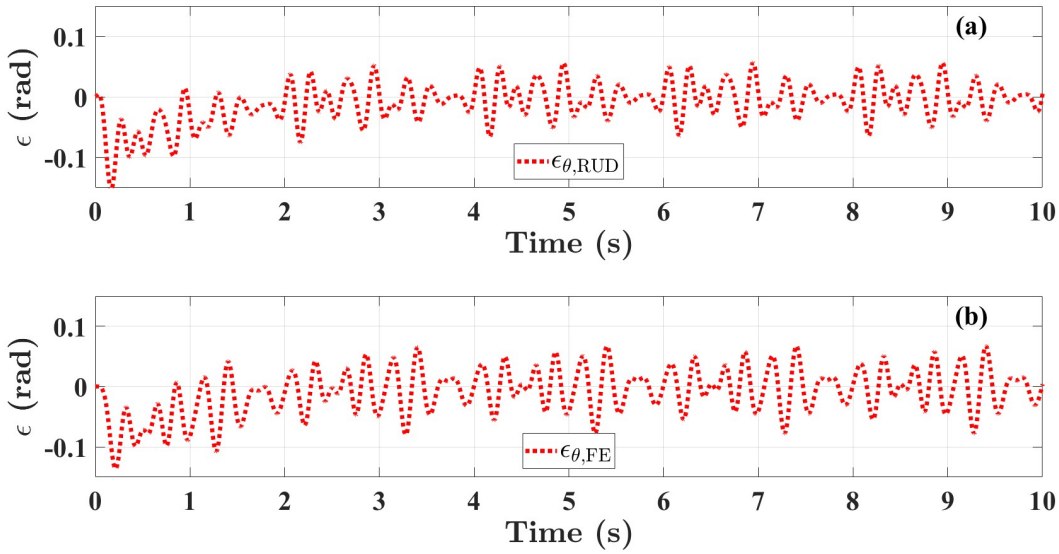


Figure 3.7: The performance for tremor suppression with a PD Controller for the stationary forearm-exoskeleton; where (a,b) shows the deviations in RUD and FE ($\epsilon = \mathrm{r}_\theta - \theta$)

Since the MPC cost term includes terms for both the proportional and derivative errors,

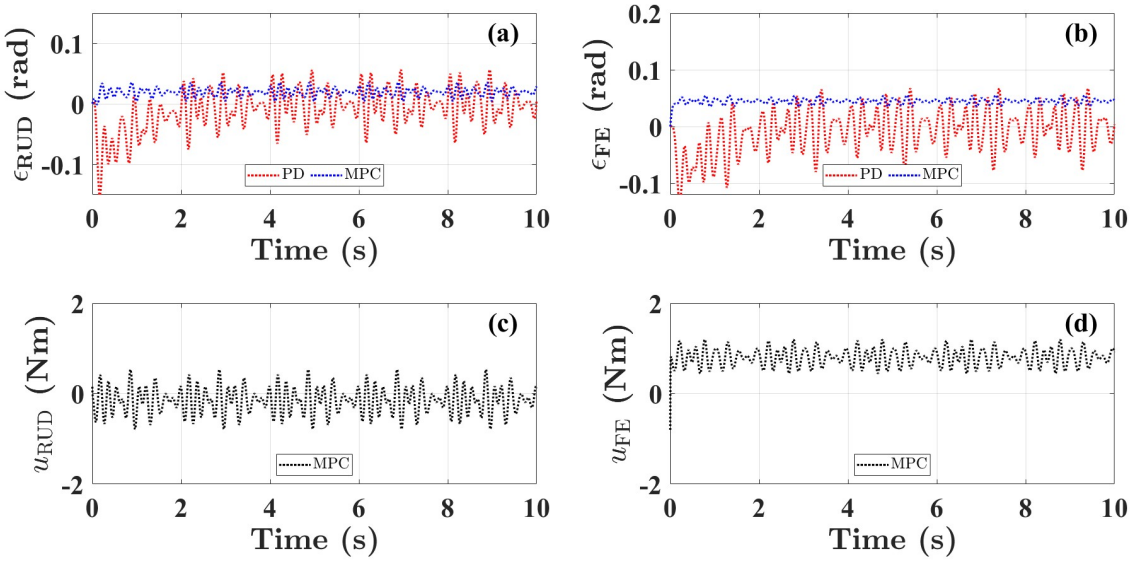


Figure 3.8: Comparing tremor suppression between PD and MPC controllers (a-b) ($\epsilon = r_\theta - \theta$). The MPC control input obtained is shown in (c-d)

the controller can suppress tremors by imitating the behavior of mechanical impedance, resembling the effect of spring and damper respectively. Next, instead of keeping the arm-exoskeleton stationary, we aim to make the arm follow a specific trajectory using the exoskeleton, while tremors act upon it. Figure 3.9 shows one simulation result, and it can be observed that, compared to the case without any tremors, the tracking errors are more tremorous.

3.2.4 Tremor suppression with BMFLC Model for tremor

In the last section, we observed that the MPC controller is able to suppress tremors fairly well, without any modeling of the tremor. In this section, we explore different methods of using a tremor model for tremor suppression in a quest to improve our performance. This, thus adds another assumption:

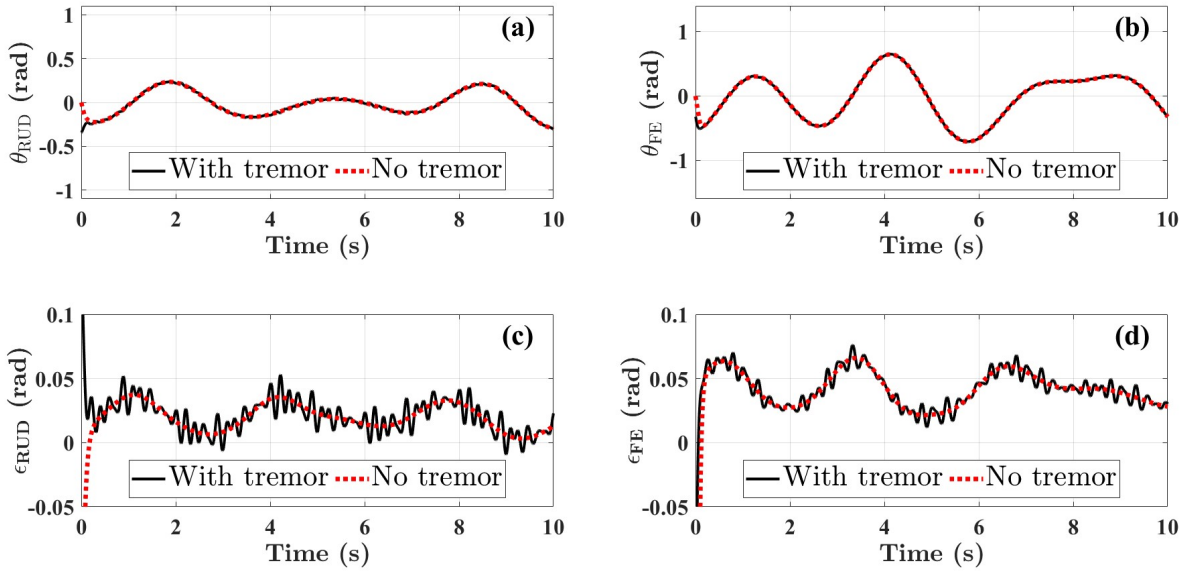


Figure 3.9: Comparison of the forearm-exoskeleton system for trajectory tracking with and without tremor, where (a,b) shows the trajectories followed (θ) and reference (r_θ); (c,d) shows the tracking errors ($\epsilon = r_\theta - \theta$)

(A5) The past tremor data is available explicitly for modeling, i.e. the tremor data has filtered out from the voluntary motions overlaid with tremors, using methods discussed in 1.2.1

The synthetic tremor excitation introduced to the model consists of harmonic waves with various frequencies within a certain bandwidth. For example, for Parkinsonian tremor, this range is 3–6 Hz as discussed in section 1.1.1. As discussed in Section 1.2.1, previous studies investigated the regression of tremor signals based on data-driven models such as the Autoregressive (AR) model, the weighted-frequency Fourier linear combiner (WFLC), band-limited multi-frequency Fourier linear combiner (BMFLC) model etc. We explore the band-limited multi-frequency Fourier linear combiner (BMFLC) ([77]) to model the tremor; specifically to approximate or predict tremor signals for the control horizon. A BMFLC

model with n_{BMFLC} frequency components can be written as:

$$\mu_{\text{BMFLC}}(t) = \sum_{i=1}^{n_{\text{BMFLC}}} (p_{\mu,\text{BMFLC},i} sin(c_{\mu,\text{BMFLC},i} t) + p_{\mu,\text{BMFLC},i+n} cos(c_{\mu,\text{BMFLC},i} t)) \tag{3.16}$$

where $p_{\mu,\text{BMFLC},i}$ is the i^{th} uncertain amplitude parameter and $c_{\mu,\text{BMFLC},i}$ is the i^{th} constant frequency. The frequencies $c_{\mu,\text{BMFLC},1}$ and $c_{\mu,\text{BMFLC},n_{\text{BMFLC}}}$ is the bandwidth, and n_{BMFLC} decides the frequency-domain resolution of the model ([79]).

This can also be rewritten as:

$$\mu_{\text{BMFLC}}(t) = J^T_{\phi,\text{BMFLC}}(t) \times \phi \tag{3.17}$$

where

$$J_{\phi,\text{BMFLC}}(t) = \begin{bmatrix} sin(\text{c}_{\mu,1,\text{RUD}} t) & 0 \\ \vdots & \vdots \\ sin(\text{c}_{\mu,n,\text{RUD}} t) & 0 \\ cos(\text{c}_{\mu,n+1,\text{RUD}} t) & 0 \\ \vdots & \vdots \\ cos(\text{c}_{\mu,2n,\text{RUD}} t) & 0 \\ 0 & sin(\text{c}_{\mu,1,\text{FE}} t) \\ \vdots & \vdots \\ 0 & sin(\text{c}_{\mu,n,\text{FE}} t) \\ 0 & cos(\text{c}_{\mu,n+1,\text{FE}} t) \\ \vdots & \vdots \\ 0 & cos(\text{c}_{\mu,2n,\text{FE}} t) \end{bmatrix} \tag{3.18}$$

and $\phi \in R^{4n_{\text{BMFLC}} \times 1}$ is the uncertain amplitude parameter vector given as :

$$\phi = [p_{\mu,1,\text{RUD}}, .. \; p_{\mu,2n_{\text{BMFLC}},\text{RUD}}, p_{\mu,1,\text{FE}}, .. \; p_{\mu,2n_{\text{BMFLC}},\text{FE}}]^T \tag{3.19}$$

The model equation in 3.13a now changes as

$$x_{k+1} = Ax_k + Bu_k + (J^T_{\phi,\text{BMFLC}})_k \phi + K \tag{3.20}$$

$(J^T_{\phi,\text{BMFLC}})_k \phi$ now acts as another input to the dynamical model, while u_k now would change accordingly to resist this input which causes tremor.

The goal is to get these uncertain amplitude parameters via optimization, and to do this, another cost term is introduced to (3.12) which would update these parameters at every time sample. With this, the modified cost function can now be rewritten as:

$$\begin{aligned} \mathcal{J}_{0\to N}(x_0, U_{0\to N}) := \|x_N - x_N^{des}\|_P^2 \\ + \sum_{k=0}^{N-1} (\|x_k - x_k^{des}\|_Q^2 + \|u_k\|_R^2) + \phi^T S \phi \end{aligned} \tag{3.21}$$

where $S = S^T \succ 0$ is the cost matrix corresponding to the amplitude vector. Considering these modifications, the new decision variables $Z_{0\to N}$ are given as
$\text{col}(x_1, x_2, ...x_N, u_0, u_1, ...u_{N-1}, \phi)$

Considering these modified formulations, the equations (3.15a and 3.15b) are changed accordingly.

As observed in figure 3.10, the performance in the FE direction improves only somewhat when the BMFLC component is included, but no improvement is observed in the RUD direction. This is probably because previous tremor data should be used to predict the

tremor, which hasn't been done here. In this case, only the weighted amplitude parameters were added to the cost function as a regularization term, leading to an incomplete model.

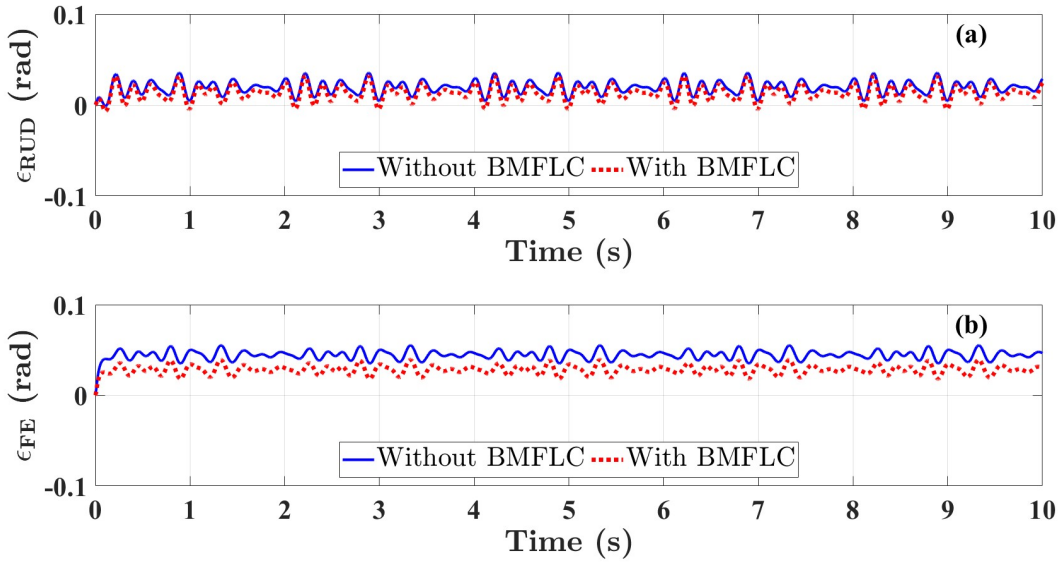


Figure 3.10: Comparing the performances for tremor suppression for stationary exoskeleton without BMFLC and with BMFLC using tracking errors ($\epsilon = \mathrm{r}_\theta - \theta$)

Thus, in our next study, the information about the past tremors is considered for estimating the tremor using the BMFLC model. Now, considering M > 0 is the delay window for tremor modeling, we can write a cost function as:

$$J_{tr}(tr,\phi) = \sum_{k=-M}^{0} \left(\left\| tr_k - (J_{\phi,BMFLC})_k^T \phi \right\|_U^2 \right) + \|\phi\|_S^2 \tag{3.22}$$

The aim here is to find the optimal ϕ which minimizes the above cost function. tr_k refers to the actual tremor, and U and S are weighing matrices. The $\|\phi\|_S^2$ term, again acts as a regularization term. This equation can be further expanded as:

$$\mathcal{J}_{tr}(\mathrm{tr},\phi) = \sum_{k=-M}^{0} \left[tr_k^T U tr_k + \phi^T (J_{\phi,BMFLC})_k U (J_{\phi,BMFLC})_k^T \phi - 2 tr_k^T U (J_{\phi,BMFLC})_k^T \phi \right] + \phi^T S \phi \tag{3.23}$$

Ignoring $tr_k^T U tr_k$ as its constant and making some modifications as:

$$TR = [tr_{-M}; tr_{-M+1}; ...tr_0] \tag{3.24a}$$

$$U_{big} = diag(U, U, ...U) \tag{3.24b}$$

$$J_{\phi,BMFLC} = [(J_{\phi,BMFLC})_{-M}, (J_{\phi,BMFLC})_{-M+1}, ...(J_{\phi,BMFLC})_0] \tag{3.24c}$$

The cost function is given as:

$$\mathcal{J}_{tr} = \phi^T (J_{\phi,BMFLC} U_{big} J_{\phi,BMFLC}^T + S)\phi - 2TR^T U_{big} J_{\phi,BMFLC}^T \phi \tag{3.25}$$

which is a quadratic function of the form $x^T Gx + \frac{1}{2}Fx$, and can be solved using quadratic solvers like quadprog or QPSwift. Once we have obtained ϕ from the quadratic solver, we can model the tremor for future timesteps as:

$$tr_{model}(t) = (J_{\phi,BMFLC})_t^T \phi \text{ for } t = 1, ..., hor \tag{3.26}$$

Where *hor* is the MPC control horizon.

The state equation is then modified, as given in 3.20, and the input obtained after solving the quadratic optimization aids to suppress the tremor.

The overall methodology can be summarized below in figure 3.11

This methodology adds some new parameters to tune, and the trends observed from simulations by changing them are given below:

1. **M**: This is the delay window basis which tremor is modeled. We observed better performances as we increased this initially (till about 0.3 s), and as we increased this

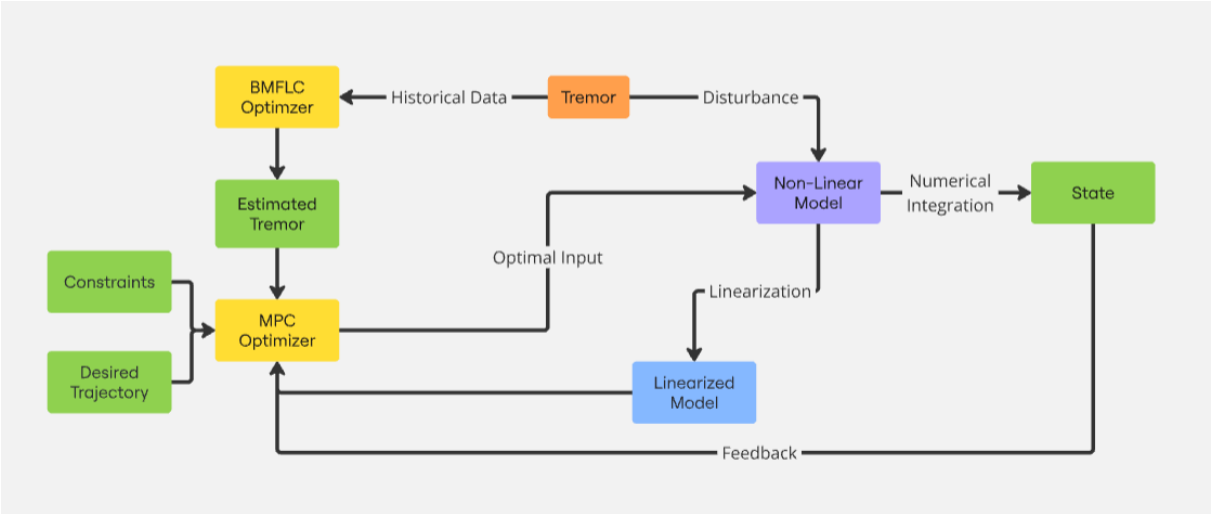


Figure 3.11: The MPC + BMFLC flowchart for tracking and tremor suppression

further, the increase in performance was very less, and instead, the optimization process gets very slow as the matrix size increases.

2. **U and S**: Error weighing matrix and the weighing matrix for the uncertain amplitude parameters. Increasing S compared to U improved performance till about $\|S\| = 50\,\|U\|$, beyond which the performance worsened. This is consistent as we know that by increasing the regularization parameter too much, the model will ignore the data completely and the norm of the parameters will be forced arbitrarily close to zero. The matrix U consists of errors of both in RUD and FE directions, and since both are equally important, the values were kept the same.

3. Δ: This is the gap between 2 consecutive time series points to be considered for tremor modeling (example: if M=200, and $\Delta = 10$, then the modeling will be done by taking $200/10 = 20$ terms, or every 10th term). Since the tremor data frequency is 200, we can implement this to reduce the optimization time while keeping most of the information. It was observed that till $\Delta = 5$, the performance was very close to when $\Delta = 1$.

Increasing further leads to loss of data, affecting performance.

4. n_{BMFLC}: Number of frequency components in BMFLC modeling. Increasing beyond 16 didn't yield significant improvements, instead slowing down the optimization.

5. $c_{\mu,1}$ **and** $c_{\mu,n_{BMFLC}}$: Bandwidth of the frequency components. Usually, tremor frequency is about 5-6 Hz, so the upper and lower limits of the frequency components were selected accordingly.

The final obtained tuned parameters were $M = 150, U = U = 1 * I_2, S = 50, \Delta = 5, n_{BMFLC} = 16, c_{\mu,1} = 2, c_{\mu,n_{BMFLC}} = 7$.

Figures 3.12 and 3.13 show the performance of the exoskeleton simulation for the stationary case and for the trajectory tracking case respectively by implementing this BMFLC based tremor modeling based on historical tremor data. In both cases, it can be observed that the amplitude of the tremors has been significantly reduced compared to the case when MPC was applied without any tremor model.

3.3 Real Tremor and using neural-network model

So far in this section, we have dealt with simulated tremors to test our controller efficacy. While so far, our results have been pretty good, the next step would be to test this on actual tremors. In section 2, we used the dataset from [60] to explore data-driven modeling methods for tremors. Thus, we would now use the same data for our next simulations on the exoskeleton.

We first implement the BMFLC to model the tremors, and then use the Sequential CNN-LSTM 1 model from section 2.3, which was our best-performing model overall to model the

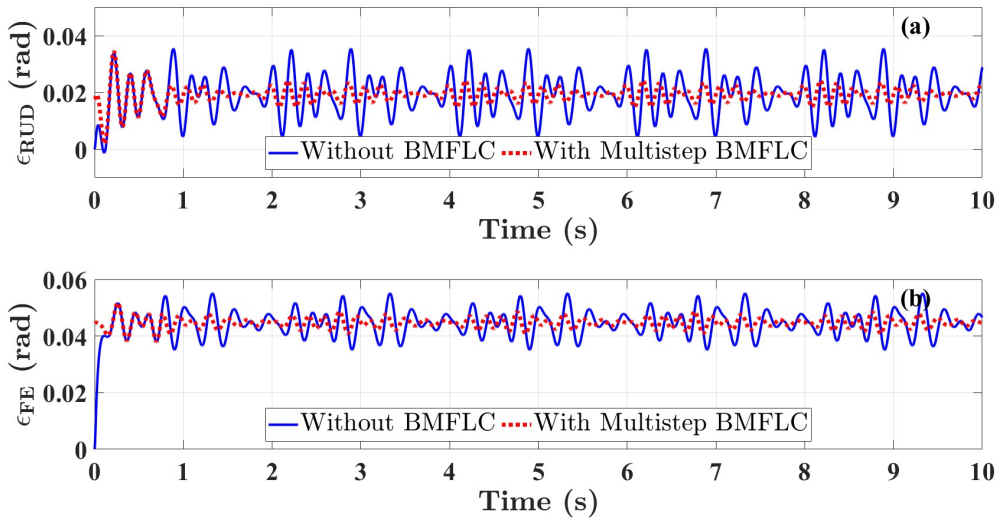


Figure 3.12: The comparison for performances for tremor suppression using tracking errors ($\epsilon = r_\theta - \theta$) for stationary exoskeleton for suppression without BMFLC and suppression with tremor modeling with BMFLC using historical tremor data. It should be noted that for the 1st 1 second or so, the tremor suppression doesn't improve as we don't have enough historical tremor data to fit our BMFLC model.

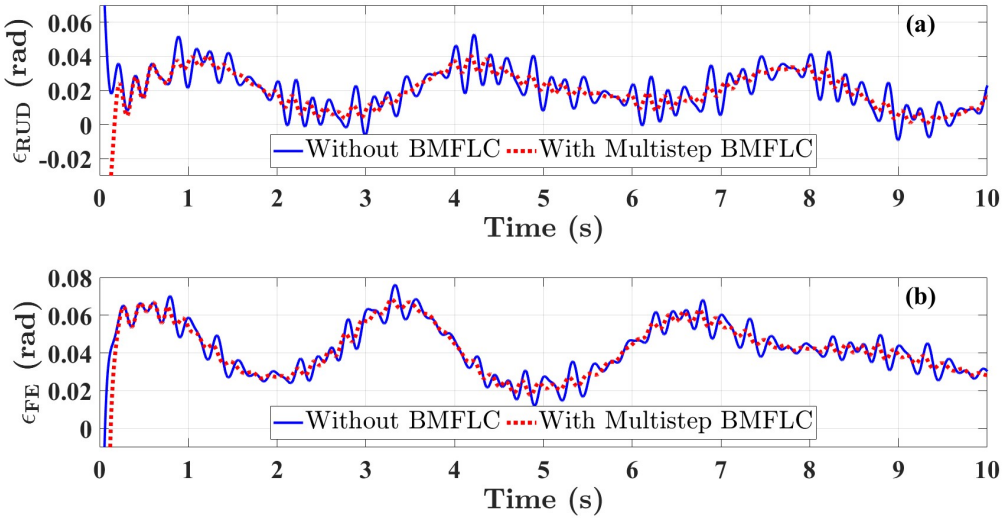


Figure 3.13: The comparison for performances for tremor suppression using with both tremor suppression and trajectory tracking for suppression without BMFLC and suppression with multi-step BMFLC model.

tremors. As mentioned in section 2.3, the data was divided into train, validation and test, from which train and validation data were used for training; hence, for this evaluation, only the test data was used. The model equation in 3.13a now slightly changes as

$$x_{k+1} = Ax_k + Bu_k + f_{seq1}(k) + K \tag{3.27}$$

where f_{seq1} is the neural network model used to predict the tremor. With our simulation frequency of 200 Hz, and the control horizon for MPC as 20, we need the neural network model to be able to model the tremor for 100 ms ahead of time, which was exactly the prediction horizon of our neural network model in section 2.3! It should be noted that the training was done offline in Python, and the trained model with weights was imported into the MATLAB environment to model the tremors. Further, since the tremors used were only in the WFE direction, the RUD tremors were considered to be 0 for this case. Figure 3.17 shows the comparisons of the results obtained when the tremors are modeled with BMFLC and Sequential 1. Along expected lines, it can be observed that the suppression performance is mostly better when the Sequential 1 model is used; although the BMFLC model performs appreciably well as well. However, it should be noted that for the BMFLC model, we are optimizing a second quadratic problem in each loop to obtain the amplitude parameters, making the process considerably slower than using a pre-trained neural network.

3.4 Robustness and Real-time Feasibility Analysis

In section 3.1.1, we had taken an assumption (A1) that the forearm is a rigid body, with the mass and inertia properties known and available. The mannequin model of the wrist was created based on the mean wrist dimensions in [10], as summarized in table 3.1. The density was considered to be uniform, with the total mass as 400g [4].

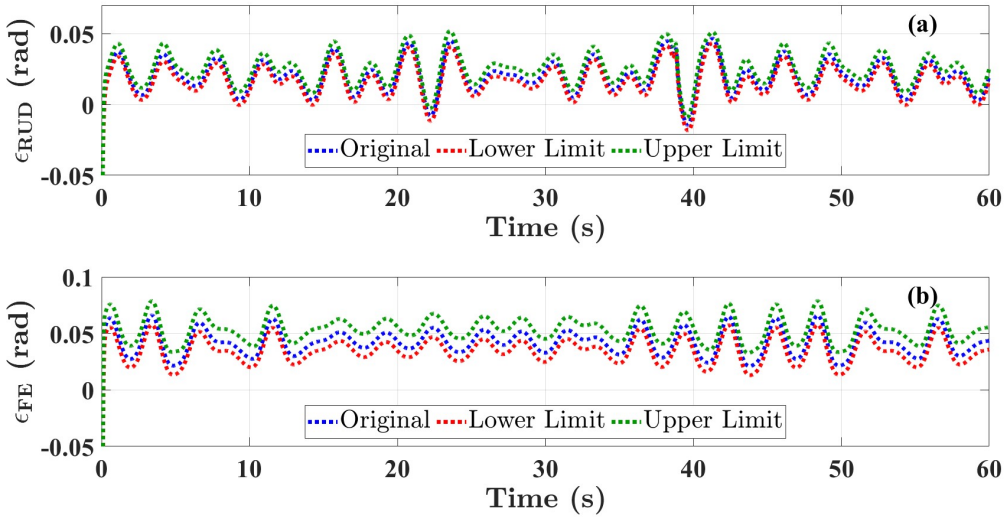
However, as described in table 3.1, the dimensions, along with the weight of the hand can vary significantly between individuals. Thus, to test the robustness of our controller, we consider the 95th male percentile dimensions as our upper limit and the 5th female percentile as our lower limit. The hand's weight was reduced based on the reduction in dimensions, keeping the density the same. Using this method, the lower and upper limits of the hand obtained are 58% and 162% of the original weight. These lower and upper limits (or the actual dimensions) were only used for integrating the non-linear model, while the initial mean dimensions were used for the controller (linearization and constraints).

Table 3.1: Statistics of dimensions of the human hand, based on a study of 92 males and 73 females in [10], the dimensions are in mm

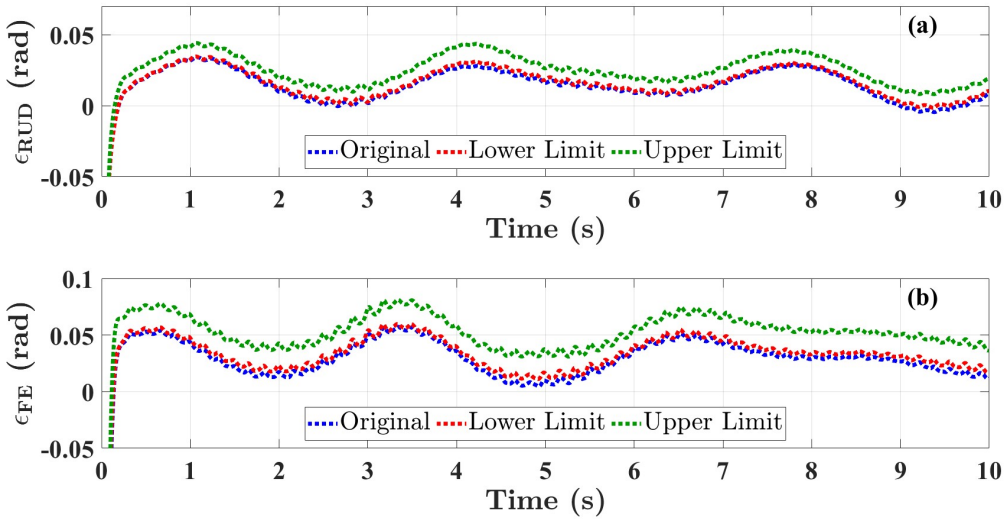
Men				**Women**			
Dim	**Mean**	**5th perc**	**95th perc**	**Dim**	**Mean**	**5th perc**	**95th perc**
Length	104.3	94.05	115.69	Length	91.45	83.45	99.52
Breadth	87.32	79.95	94.4	Breadth	76.06	68.78	84.1
Depth	42.89	36.6	47.82	Depth	37.32	32.62	44.03

Figure 3.14 shows the results obtained by considering these upper and lower limits of the hand's weight and dimensions. Figure 3.14a shows the performance for only trajectory tracking, without any tremors. It was observed that while lowering the weight and dimensions didn't significantly affect the performance, an increase in weight led to significantly higher tracking errors. This is especially true in the WFE tracking, as it is along the direction of the gravitational force. Figure 3.14b shows the performance for both trajectory tracking and tremor suppression, where the tremors used are the tremors from Subject 5 in 2.3, and the Sequential CNN-LSTM 1 model is used as the tremor model for tremor suppression. Here too, it can be observed that the tracking errors increased significantly in the FE direction when the upper limit was considered. However, there isn't much change in the amplitude of the resultant tremors, either for the upper limit or the lower limit inertia parameters; from which we can confidently say that the tremor suppression performance doesn't get affected

much with the change in the wrist's inertia parameters.



(a) Robustness analysis: Only trajectory tracking



(b) Robustness analysis: Trajectory tracking and tremor suppression

Figure 3.14: Robustness analysis considering only trajectory tracking (a), and both trajectory tracking and tremor suppression with Neural network model(b)

Second, so far for all of our studies, we have considered the user control to be non-existent; i.e. the user doesn't voluntarily try to move his wrist in the given trajectory, and the entire support is given by the exoskeleton. But, in a real-life scenario, it would be more of a

collaboration task. The user would create the intent and would move his wrist inefficiently due to tremors and reduced strength, while the exoskeleton would aim to add additional support while suppressing tremors. Just like pathological tremors, it is hard to explain the neuromuscular dynamics of the user controller analytically. Thus, for simplicity, we assume that the user controller is a weak PD controller. This is based on the reasoning that due to reduced strength and agility, the response would be slower between the intent and the movement and insufficient. Thus, the user input is now given as:

$$\begin{aligned} u_{user} &= K_P(x_r - y) + K_D(\dot{x}_r - \dot{y}) \\ &= K_P\epsilon_{user} + K_D\dot{\epsilon}_{user} \end{aligned} \tag{3.28}$$

where $K_P > 0$ and $K_D > 0$ are respectively the scalar proportional and derivative control gains. The exoskeleton inputs, disruptions, and any time-varying generalized forces are not taken into account in the design of this user controller.

It should be noted that during simulations, u_{user} is only applied to the original non-linear dynamics in equation 3.4, and not while formulating the MPC Controller. Basically, the MPC Controller has no information on this user input, and the user input acts like a disturbance to the MPC controller, and we want to verify if it's robust enough to adapt accordingly.

Figure 3.15 shows the performance after adding the user input; (a) and (b) show how the tracking improves when the MPC controller is applied along with user input (no tremors), while figures (c) and (d) show that the MPC+BMFLC controller works just as good when user input is considered as well.

Finally, since our entire study has been based on simulations, we need to ensure that the numerical calculations required for each step are fast enough to run the simulations, and in the future, the experiments at 200 Hz. To run at 200 Hz, each step needs to be completed in

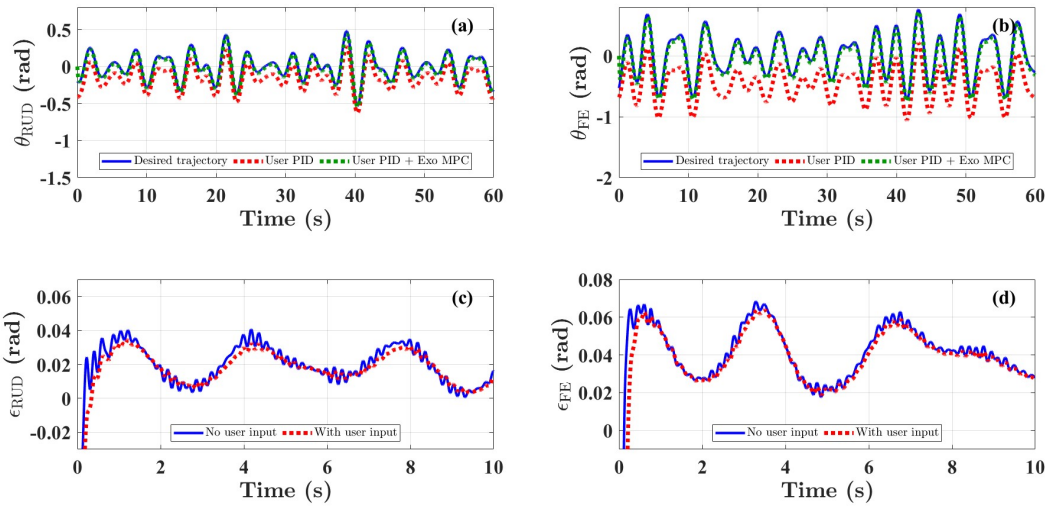


Figure 3.15: The comparison for performances considering user input to be a PD controller; (a) and (b) show the comparison of performance for trajectory tracking without tremors; (c) and (d) show the comparison for tremors with and without considering user input

<5 ms. We considered the MPC with BMFLC model execution and the MPC with neural net to record the timings required for each step of computation for the simulation. The MPC with BMFLC model would theoretically have the highest computation time, since it involves solving 2 quadratic optimizations in a single loop. It should be noted that all the computation processes (linearization, quadratic solver, numerical integration of the non-linear model) were initialized before the loop. As shown in figure 3.16, all the steps were comfortably completed in less than 5 ms, with the typical time being 1-1.5 ms for each loop. This should be fast enough for real-time applications; however, it would also depend on the delays of the hardware involved, based on which the frequency may have to be changed. As expected, the MPC with BMFLC took longer, as it involved 2 quadratic optimizations in a single loop (with a sudden spike when the BMFLC model first gets activated after we have sufficient tremor history data for modeling), while the MPC with NN only had 1 optimization as the neural net model has been already pretrained.

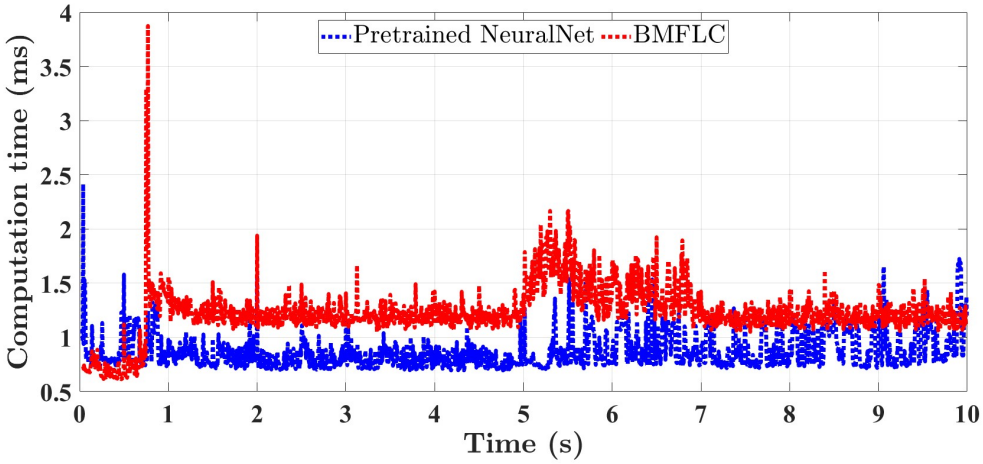
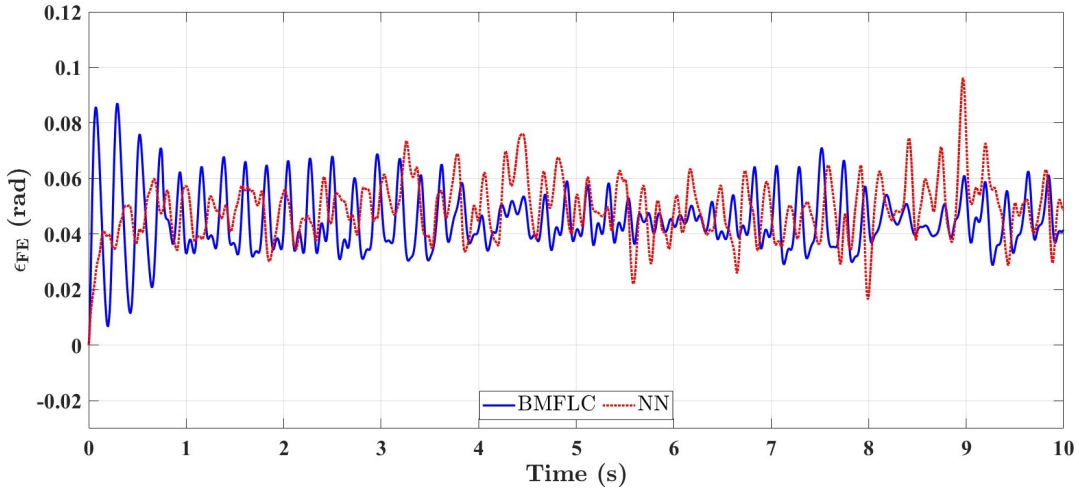
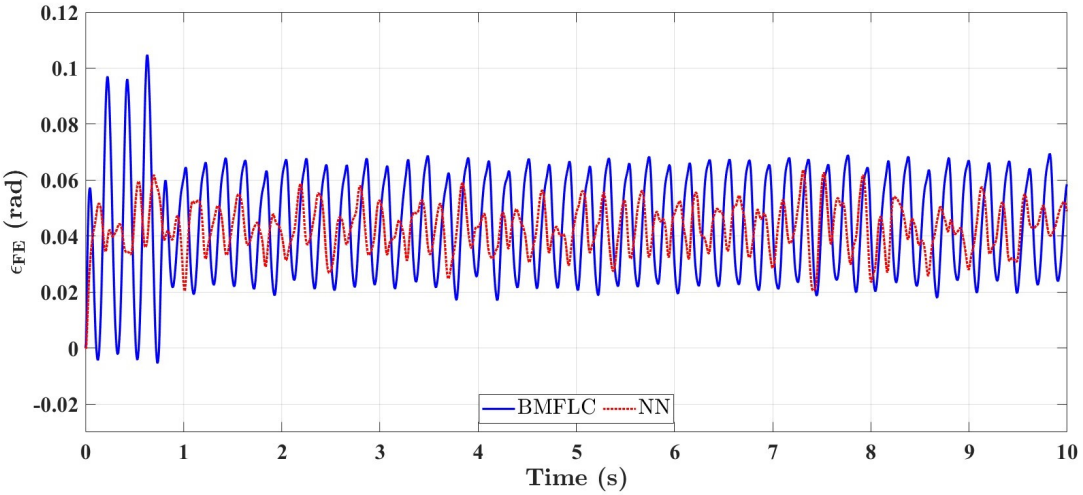


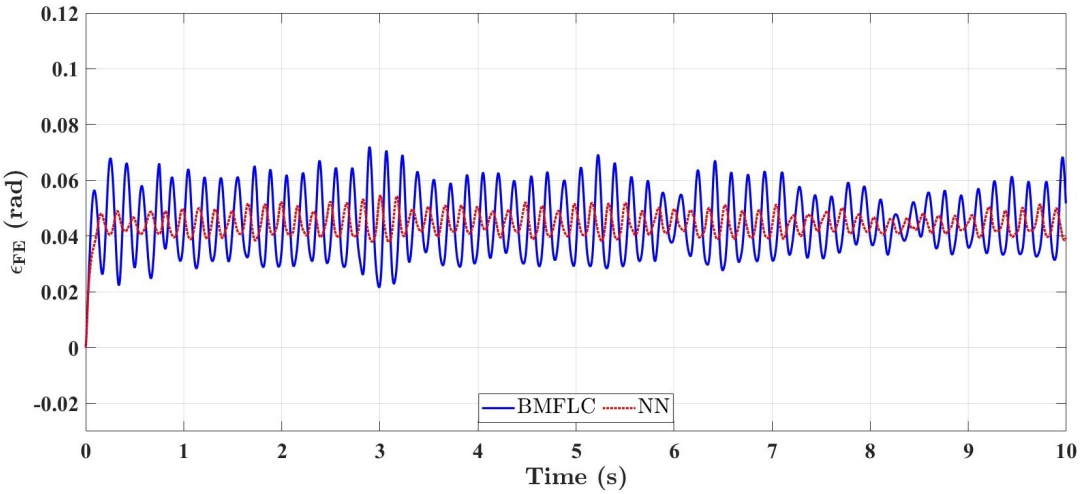
Figure 3.16: Total computation time for each cycle for MPC with BMFLC model and MPC with Neural-net model



(a) Subject 1



(b) Subject 4



(c) Subject 5

Figure 3.17: Performance using a BMFLC model vs Neural-Network model for real tremor suppression for tremors from subjects 1, 4 and 5 from 2.4

Chapter 4

Discussion

In this chapter, the results of the thesis are summarized. We started off with pathological tremors modeling. An open-source dataset of sEMG and motion signals was collected and pre-processed. Initial spectrograms of experimental tremor data showed that the frequencies of the dominant harmonic components have similar patterns for both EMGs and kinematics data, and that the frequencies and amplitudes of these harmonic components change over time, indicating that pathological tremor is a nonlinear dynamics problem. We then developed different neural network models, involving MLPs, LSTM, CNNs and combinations of all to obtain the best approximation of the unknown non-linear function to predict pathological tremors up to 100 ms. From the developed models, it was observed that the Sequential CNN-LSTM models performed better for most cases, with effectively less than 5% error. Further studies were conducted to study the effects of input and prediction horizon, and hyperparameters on performance. These studies showed that increasing the learning rate or the prediction horizon too much affected the performance negatively. Most non-linear activation functions worked well, except for sigmoid. Next, we also explored adding constraints to our model by adding developed loss terms of the 1st-order and 2nd order derivatives of the kinematics data. The visualizations showed that the resultant predictions were much smoother in this case.

Next, we developed a model-predictive controller framework for the exoskeleton developed in our lab for trajectory tracking and tremor alleviation. Our initial testing, without considering

any tremors showed very good results for trajectory tracking, with the tracking errors being less than 0.05 radians. We first added simulated tremors, and the MPC controller also performed some tremor suppression, which was significantly better than a PD controller. We then adapted BMFLC as an adaptive model for tremor modeling and integrated the same with MPC to provide better tremor suppression. Parametric analysis was also performed for the BMFLC modeling to obtain the optimal delay window, weighing matrices, and frequency and bandwidth of the BMFLC model. The simulation results obtained showed a significant decrease in the resultant tremor's amplitude, thus effectively suppressing tremors. This was also tested by considering the user's input as a weak PD controller to similar results.

Finally, we added actual tremors, and our evaluations suggest that integrating the trained neural network models into our controller resulted in better tremor suppression than the BMFLC model. Furthermore, we also evaluated the robustness of our controller by changing the wrist inertia parameters and observed that an increase in inertia particularly leads to a decrease in our performance. Finally, we also analyzed the real-time feasibility of employing our methodology on the actual hardware.

Chapter 5

Conclusions and Future Work

In this thesis, we first presented a deep-learning-based long-term pathological tremors prediction using EMG signals and historical motion data. An open-source dataset of sEMG and motion signals was collected and preprocessed. Various deep learning architectures were proposed, including hybrid CNN-LSTM models to predict pathological tremor signals from EMG data with fairly good results. However, the study is limited in many aspects. The study for the tremor modeling is currently limited in many aspects. First, only one direction (F/E) of the tremors is studied, but in practical cases, all the 3 wrist movements are affected. Second, while neural networks perform appreciable predictions, the physical meanings of the models can't really be interpreted, giving us no information on the tremor dynamics. Thus, with regards to tremor-modeling for future work, the following tasks need to be completed.

1a. Exploring Physics-Constrained Data-driven Modeling: These models would employ a combination of model-based (MBR) and model-free (MFR) regressions to obtain neuromusculoskeletal models that can be used for both analytical studies and real-time applications. Currently, experiments are being performed to gather a holistic collection of measurement data from studies, such as kinematics, EEG, and EMG, and use this data to build more precise models of pathological tremors. The collected datasets would consist of multi-directional movement measurements and thus should be used as the database for this study.

1b. Online Training for data-driven model: The training for our neural networks is done offline with limited data. Employing online training with new data that becomes available

could be a useful method to fine-tune models from a base model for more accurate modeling.

Next, we introduced a model predictive controller designed to suppress tremors in the exoskeleton. The state space form was obtained by linearizing the non-linear multibody system at each time sample, and the control inputs were obtained by minimizing a convex quadratic cost function with constraints. The control simulations of the forearm and TAWE systems for performance tracking were then used to validate the developed controller. The BMFLC and Neural network models were then integrated for better tremor suppression. However, we also considered a lot of simplifications compared to the actual real-world scenario. We considered that the states were known and could be directly obtained, whereas in reality, these would have to be obtained from sensors, which would be noisy. We also considered the voluntary intent/trajectory and tremor history is readily available; but usually, the voluntary motions would be overlaid with tremors, which need to be filtered out. Thus, with regard to controller development, the following tasks need to be completed.

2a. So far, with our MPC Control system, we haven't encountered any case where the quadratic problem is infeasible. However, methods should be explored to deal with such situations when they arise. One method could be using the obtained input from the previous optimization cycle. Another way could be increasing the control horizon even higher, although it would slow down the process. Further tests should also be done with different trajectories to test the feasibility of the proposed solutions.

2b. Integrating real-time state estimation methods like EKFs to get the states from inertia measurement units (IMU) and encoders on TAWE.

2c. Developing methods for real-time voluntary movement estimation ahead of time from full movement signal which contains both voluntary and tremor components. The voluntary motion signal needs to be separated from the tremor signals using tremor filtering algorithms,

and if neuromuscular sensors (i.e., sEMG sensors) are available in the exoskeleton, these along with past movement signals can be used for estimating voluntary movement to generate the reference trajectory.

2d. Finally, methods should be explored to add more robustness to our controller since we are dealing with uncertain inertia. Further, the connection positions of the exoskeleton to the arm could vary with time as well. While methods like Tube-MPC could be explored, a better strategy could be to employ methods like Model reference adaptive control to identify the unknown dynamical properties by persistently exciting the system using exoskeleton input only.

3. Following the aforementioned advancements, TAWE must be carefully assessed using a large number of human trials. To investigate the effectiveness of TAWE in tremor suppression and user movement compliance, volunteer individuals with or without pathological tremors must be invited. The methods would have be further optimized with the help of user input and data collection, and it may eventually develop into a viable product that will help those with pathological tremors live better lives.

Bibliography

[1] Dag Aarsland, Byron Creese, Marios Politis, K Ray Chaudhuri, Dominic H Ffytche, Daniel Weintraub, and Clive Ballard. Cognitive decline in parkinson disease. *Nature Reviews Neurology*, 13(4):217–231, 2017.

[2] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[3] Jane E Alty and Peter A Kempster. A practical guide to the differential diagnosis of tremor. *Postgraduate medical journal*, 87(1031):623–629, 2011.

[4] Joseph T Belter, Jacob L Segil, and BS Sm. Mechanical design and performance specifications of anthropomorphic prosthetic hands: a review. *Journal of rehabilitation research and development*, 50(5):599, 2013.

[5] Alim Louis Benabid. Deep brain stimulation for parkinson's disease. *Current opinion in neurobiology*, 13(6):696–706, 2003.

[6] Alfredo Berardelli, AF Sabra, and M Hallett. Physiological mechanisms of rigidity in parkinson's disease. *Journal of Neurology, Neurosurgery & Psychiatry*, 46(1):45–53, 1983.

[7] Antonio Padilha Lanari Bo, Philippe Poignet, and Christian Geny. Pathological tremor and voluntary motion modeling and online estimation for active compensation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(2):177–185, 2010.

[8] Henning Boecker and David J Brooks. Functional imaging of tremor. *Movement disorders*, 13(S3):64–72, 1998.

[9] Matteo Bologna, Giorgio Leodori, Paola Stirpe, Giulia Paparella, Donato Colella, Daniele Belvisi, Alfonso Fasano, Giovanni Fabbrini, and Alfredo Berardelli. Bradykinesia in early and advanced parkinson's disease. *Journal of the neurological sciences*, 369: 286–291, 2016.

[10] Erman Cakit, Behice Durgun, Oya Cetik, and Oguz Yoldas. A survey of hand anthropometry and biomechanical measurements of dentistry students in turkey. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 24(6):739–753, 2014.

[11] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[12] Carmen Camara, Pedro Isasi, Kevin Warwick, Virginie Ruiz, Tipu Aziz, John Stein, and Eduard Bakštein. Resting tremor classification and detection in parkinson's disease patients. *Biomedical Signal Processing and Control*, 16:88–97, 2015.

[13] Yuwen Chen, Kunhua Zhong, Ju Zhang, Qilong Sun, and Xueliang Zhao. Lstm networks for mobile human activity recognition. In *2016 International conference on artificial intelligence: technologies and applications*, pages 50–53. Atlantis Press, 2016.

[14] Daniel M Corcos, Gerald L Gottlieb, Mark L Latash, Gil L Almeida, and Gyan C Agarwal. Electromechanical delay: An experimental artifact. *Journal of Electromyography and Kinesiology*, 2(2):59–68, 1992.

[15] Houde Dai, Pengyue Zhang, and Tim C Lueth. Quantitative assessment of parkinsonian tremor based on an inertial measurement unit. *Sensors*, 15(10):25055–25071, 2015.

[16] T D'Alessio and S Conforto. Extraction of the envelope from surface emg signals. *IEEE Engineering in Medicine and Biology Magazine*, 20(6):55–61, 2001.

[17] Olivier Darbin, Elizabeth Adams, and Daniel Dees. Non-linear dynamics in parkinsonism. *Frontiers in Neurology*, 4:61229, 2013.

[18] Scott L Delp, Anita E Grierson, and Thomas S Buchanan. Maximumisometric moments generated by the wrist muscles in flexion-extension and radial-ulnar deviation. *Journal of Biomechanics*, 29(10):1371–1375, 1996.

[19] Hao Deng, Weidong Le, and Joseph Jankovic. Genetics of essential tremor. *Brain*, 130 (6):1456–1464, 2007.

[20] Günther Deuschl, Jan Raethjen, Michael Lindemann, and Paul Krack. The pathophysiology of tremor. *Muscle & Nerve: Official Journal of the American Association of Electrodiagnostic Medicine*, 24(6):716–735, 2001.

[21] Karen M Doherty, Bart P van de Warrenburg, Maria Cecilia Peralta, Laura Silveira-Moriyama, Jean-Philippe Azulay, Oscar S Gershanik, and Bastiaan R Bloem. Postural deformities in parkinson's disease. *The Lancet Neurology*, 10(6):538–549, 2011.

[22] Strahinja Dosen, Silvia Muceli, Jakob Lund Dideriksen, Juan Pablo Romero, Eduardo Rocon, Jose Pons, and Dario Farina. Online tremor suppression using electromyography and low-level electrical stimulation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(3):385–395, 2014.

[23] Rodger Elble and Günther Deuschl. Milestones in tremor research. *Movement Disorders*, 26(6):1096–1105, 2011.

[24] W Jeffrey Elias, Diane Huss, Tiffini Voss, Johanna Loomba, Mohamad Khaled, Eyal Zadicario, Robert C Frysinger, Scott A Sperling, Scott Wylie, Stephen J Monteith, et al. A pilot study of focused ultrasound thalamotomy for essential tremor. *New England Journal of Medicine*, 369(7):640–648, 2013.

[25] Aberham Genetu Feleke, Luzheng Bi, and Weijie Fei. Emg-based 3d hand motor intention prediction for information transfer from human to robot. *Sensors*, 21(4):1316, 2021.

[26] JA Gallego, E Rocon, JO Roa, JC Moreno, AD Koutsou, and Jose L Pons. On the use of inertial measurement units for real-time quantification of pathological tremor amplitude and frequency. *Procedia Chemistry*, 1(1):1219–1222, 2009.

[27] Jianbo B Gao. Analysis of amplitude and frequency variations of essential and parkinsonian tremors. *Medical and Biological Engineering and Computing*, 42:345–349, 2004.

[28] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020.

[29] Jianqiao Guo, Junpeng Chen, Jing Wang, Gexue Ren, Qiang Tian, and Chuanbin Guo. Emg-assisted forward dynamics simulation of subject-specific mandible musculoskeletal system. *Journal of Biomechanics*, 139:111143, 2022.

[30] David Hajdu, John Milton, and Tamas Insperger. Extension of stability radius to neuromechanical systems with structured real perturbations. *IEEE transactions on neural systems and rehabilitation engineering*, 24(11):1235–1242, 2016.

[31] B Hellwig, P Mund, B Schelter, B Guschlbauer, J Timmer, and CH Lücking. A longitudinal study of tremor frequencies in parkinson's disease and essential tremor. *Clinical Neurophysiology*, 120(2):431–435, 2009.

[32] Rick C Helmich. The cerebral basis of parkinsonian tremor: a network perspective. *Movement Disorders*, 33(2):219–231, 2018.

[33] Michael A Henson. Nonlinear model predictive control: current status and future directions. *Computers & Chemical Engineering*, 23(2):187–202, 1998.

[34] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[35] Anas Ibrahim, Yue Zhou, Mary E Jenkins, Ana Luisa Trejos, and Michael D Naish. The design of a parkinson's tremor predictor and estimator using a hybrid convolutional-multilayer perceptron neural network. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 5996–6000. IEEE, 2020.

[36] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[37] Samay Jain, Steven E Lo, and Elan D Louis. Common misdiagnosis of a common neurological disorder: how are we misdiagnosing essential tremor? *Archives of neurology*, 63(8):1100–1104, 2006.

[38] F Jakob, L Seefried, and M Schwab. Age and osteoporosis: effects of aging on osteoporosis, the diagnostics and therapy. *Der Internist*, 55:755–761, 2014.

[39] Spencer L James, Degu Abate, Kalkidan Hassen Abate, Solomon M Abay, Cristiana Abbafati, Nooshin Abbasi, Hedayat Abbastabar, Foad Abd-Allah, Jemal Abdela, Ahmed Abdelalim, et al. Global, regional, and national incidence, prevalence, and years lived with disability for 354 diseases and injuries for 195 countries and territories, 1990–2017: a systematic analysis for the global burden of disease study 2017. *The Lancet*, 392 (10159):1789–1858, 2018.

[40] Paulina Kieliba, Peppino Tropea, Elvira Pirondini, Martina Coscia, Silvestro Micera, and Fiorenzo Artoni. How are muscle synergies affected by electromyography pre-

processing? *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26 (4):882–893, 2018.

[41] William C Koller. Pharmacologic treatment of parkinsonian tremor. *Archives of neurology*, 43(2):126–127, 1986.

[42] Nili E Krausz, Denys Lamotte, Iason Batzianoulis, Levi J Hargrove, Silvestro Micera, and Aude Billard. Intent prediction based on biomechanical coordination of emg and vision-filtered gaze for end-point control of an arm prosthesis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(6):1471–1480, 2020.

[43] Felipe Kuhne, Walter Fetter Lages, and J Gomes da Silva Jr. Model predictive control of a mobile robot using linearization. In *Proceedings of mechatronics and robotics*, pages 525–530. Citeseer, 2004.

[44] Martin Lakie, Carlijn A Vernooij, Timothy M Osborne, and Raymond F Reynolds. The resonant component of human physiological hand tremor is altered by slow voluntary movements. *The Journal of physiology*, 590(10):2471–2483, 2012.

[45] David Lindenbach and Christopher Bishop. Critical involvement of the motor cortex in the pathophysiology and treatment of parkinson's disease. *Neuroscience & Biobehavioral Reviews*, 37(10):2737–2750, 2013.

[46] Praween Lolekha, Pornpatr Dharmasaroja, Nattaphol Uransilp, Puchit Sukphulloprat, Sombat Muengtaweepongsa, and Kongkiat Kulkantrakorn. The differences in clinical characteristics and natural history between essential tremor and essential tremor plus. *Scientific reports*, 12(1):7669, 2022.

[47] Elan D Louis and Duarte G Machado. Tremor-related quality of life: A comparison of

essential tremor vs. áparkinson's disease patients. *Parkinsonism & related disorders*, 21 (7):729–735, 2015.

[48] Kurt Manal and Thomas S Buchanan. A one-parameter neural activation to muscle activation model: estimating isometric joint moments from electromyograms. *Journal of biomechanics*, 36(8):1197–1202, 2003.

[49] CD Marsden. Origins of normal and pathological tremor. In *Movement disorders: tremor*, pages 37–84. Springer, 1984.

[50] Yuya Matsumoto, Masatoshi Seki, Takeshi Ando, Yo Kobayashi, Hiroshi Iijima, Masanori Nagaoka, and Masakatsu G Fujie. Tremor frequency based filter to extract voluntary movement of patients with essential tremor. In *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1415–1422. IEEE, 2012.

[51] Hideto Miwa. Rodent models of tremor. *The Cerebellum*, 6:66–72, 2007.

[52] Jiancheng Mo and Ronny Priefer. Medical devices for tremor suppression: current status and future directions. *Biosensors*, 11(4):99, 2021.

[53] John E Morley, Angela Marie Abbatecola, Josep M Argiles, Vickie Baracos, Juergen Bauer, Shalender Bhasin, Tommy Cederholm, Andrew J Stewart Coats, Steven R Cummings, William J Evans, et al. Sarcopenia with limited mobility: an international consensus. *Journal of the American Medical Directors Association*, 12(6):403–409, 2011.

[54] Steven Morrison and Karl M Newell. Postural and resting tremor in the upper limb. *Clinical neurophysiology*, 111(4):651–663, 2000.

[55] Nazri Mohd Nawi, Walid Hasen Atomi, and Mohammad Zubair Rehman. The effect

of data pre-processing on optimized training of artificial neural networks. *Procedia Technology*, 11:32–39, 2013.

[56] Rory J O'Connor and Manohar U Kini. Non-pharmacological and non-surgical interventions for tremor: a systematic review. *Parkinsonism & related disorders*, 17(7): 509–515, 2011.

[57] Bhavana Palakurthi and Sindhu Preetham Burugupally. Postural instability in parkinson's disease: a review. *Brain sciences*, 9(9):239, 2019.

[58] Abhishek Goud Pandala, Yanran Ding, and Hae-Won Park. qpswift: A real-time sparse quadratic program solver for robotic applications. *IEEE Robotics and Automation Letters*, 4(4):3355–3362, 2019.

[59] Adam C Pigg, Johanna Thompson-Westra, Karin Mente, Carine W Maurer, Dietrich Haubenberger, Mark Hallett, and Steven K Charles. Distribution of tremor among the major degrees of freedom of the upper limb in subjects with essential tremor. *Clinical Neurophysiology*, 131(11):2700–2712, 2020.

[60] Wellington C Pinheiro, Henrique B Ferraz, Maria Claudia F Castro, and Luciano L Menegaldo. An opensim-based closed-loop biomechanical wrist model for subject-specific pathological tremor simulation. *IEEE transactions on neural systems and rehabilitation engineering: a publication of the IEEE Engineering in Medicine and Biology Society*.

[61] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 2002.

[62] BNC Prichard and PMS Gillam. Treatment of hypertension with propranolol. *Br Med J*, 1(5635):7–16, 1969.

[63] Arthur Prochazka, Josef Elek, and Manouchehr Javidan. Attenuation of pathological tremors by functional electrical stimulation i: Method. *Annals of biomedical engineering*, 20:205–224, 1992.

[64] Cameron N Riviere, Stephen G Reich, and Nitish V Thakor. Adaptive fourier modeling for quantification of tremor. *Journal of neuroscience methods*, 74(1):77–87, 1997.

[65] Eduardo Rocon, Juan Manuel Belda-Lois, AF Ruiz, Mario Manto, Juan C Moreno, and Jose L Pons. Design and validation of a rehabilitation robotic exoskeleton for tremor assessment and suppression. *IEEE Transactions on neural systems and rehabilitation engineering*, 15(3):367–378, 2007.

[66] Jacob Rosen, Joel C Perry, Nathan Manning, Stephen Burns, and Blake Hannaford. The human arm kinematics and dynamics during daily activities-toward a 7 dof upper limb powered exoskeleton. In *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pages 532–539. IEEE, 2005.

[67] Emanuela Santini, Emmanuel Valjent, and Gilberto Fisone. Parkinson's disease: Levodopa-induced dyskinesia and signal transduction. *The FEBS Journal*, 275(7):1392–1399, 2008.

[68] Vasileios Skaramagkas, George Andrikopoulos, and Stamatis Manesis. An experimental investigation of essential hand tremor suppression via a soft exoskeletal glove. In *2020 European Control Conference (ECC)*, pages 889–894. IEEE, 2020.

[69] Behzad Taheri. *Real-time pathological tremor identification and suppression in human arm via active orthotic devices.* PhD thesis, Southern Methodist University, 2013.

[70] Louis CS Tan. Mood disorders in parkinson's disease. *Parkinsonism & related disorders*, 18:S74–S76, 2012.

[71] Yongyi Tang, Lin Ma, Wei Liu, and Weishi Zheng. Long-term human motion prediction by modeling motion context and enhancing motion dynamic. *arXiv preprint arXiv:1805.02513*, 2018.

[72] Tzyh Jong Tarn, GA Shoults, and SP Yang. A dynamic model of an underwater vehicle with a robotic manipulator using kane's method. *Autonomous robots*, 3:269–283, 1996.

[73] Sivanagaraja Tatinati, Kalyana C Veluvolu, Sun-Mog Hong, Win Tun Latt, and Wei Tech Ang. Physiological tremor estimation with autoregressive (ar) model and kalman filter for robotics applications. *IEEE Sensors Journal*, 13(12):4977–4985, 2013.

[74] Mary Ann Thenganatt and Joseph Jankovic. The relationship between essential tremor and parkinson's disease. *Parkinsonism & related disorders*, 22:S162–S165, 2016.

[75] Jens Timmer, Siegfried Häußler, Michael Lauk, and C-H Lücking. Pathological tremors: Deterministic chaos or nonlinear stochastic oscillators? *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 10(1):278–288, 2000.

[76] Christopher A Vaz and Nitish V Thakor. Adaptive fourier estimation of time-varying evoked potentials. *IEEE Transactions on Biomedical Engineering*, 36(4):448–455, 1989.

[77] Kalyana C Veluvolu and Wei Tech Ang. Estimation of physiological tremor from accelerometers for real-time applications. *Sensors*, 11(3):3020–3036, 2011.

[78] Stavros G Vougioukas. Reactive trajectory tracking for mobile robots based on non linear model predictive control. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3074–3079. IEEE, 2007.

[79] Jiamin Wang. *Design and Control of an Ergonomic Wearable Full-Wrist Exoskeleton for Pathological Tremor Alleviation*. PhD thesis, Virginia Tech, 2023.

[80] Jiamin Wang and Oumar R Barry. Multibody analysis and control of a full-wrist exoskeleton for tremor alleviation. *Journal of Biomechanical Engineering*, 142(12), 2020.

[81] Jiamin Wang and Oumar R Barry. Exploring data-driven modeling and analysis of nonlinear pathological tremors. *Mechanical Systems and Signal Processing*, 156:107659, 2021.

[82] Jiamin Wang and Oumar R Barry. Inverse optimal robust adaptive controller for upper limb rehabilitation exoskeletons with inertia and load uncertainties. *IEEE Robotics and Automation Letters*, 6(2):2171–2178, 2021.

[83] Jiamin Wang, Vinay R Kamidi, and Pinhas Ben-Tzvi. A multibody toolbox for hybrid dynamic system modeling based on nonholonomic symbolic formalism. In *Dynamic Systems and Control Conference*, volume 51913, page V003T29A003. American Society of Mechanical Engineers, 2018.

[84] Jiamin Wang, Sunit K Gupta, and Oumar Barry. Towards data-driven modeling of pathological tremors. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 83914, page V002T02A030. American Society of Mechanical Engineers, 2020.

[85] Xinyue Wang, Jianqiao Guo, and Qiang Tian. A forward-inverse dynamics modeling framework for human musculoskeletal multibody system. *Acta Mechanica Sinica*, 38 (11):522140, 2022.

[86] Thomas Welton, Francisco Cardoso, Jonathan A Carr, Ling-Ling Chan, Günther Deuschl, Joseph Jankovic, and Eng-King Tan. Essential tremor. *Nature reviews Disease primers*, 7(1):83, 2021.

[87] Ferdinan Widjaja, Cheng Yap Shee, Wing Lok Au, Philippe Poignet, and Wei Tech Ang. Using electromechanical delay for real-time anti-phase tremor attenuation system using functional electrical stimulation. In *2011 IEEE International Conference on Robotics and Automation*, pages 3694–3699. IEEE, 2011.

[88] Subhashie Wijemanne and Joseph Jankovic. Hand, foot, and spine deformities in parkinsonian disorders. *Journal of Neural Transmission*, 126:253–264, 2019.

[89] AW Willis, E Roberts, JC Beck, B Fiske, W Ross, R Savica, SK Van Den Eeden, CM Tanner, C Marras, and Parkinson's Foundation P4 Group Alcalay Roy Schwarzschild Michael Racette Brad Chen Honglei Church Tim Wilson Bill Doria James M. Incidence of parkinson disease in north america. *npj Parkinson's Disease*, 8(1):170, 2022.

[90] Jacob Wilson, Meaghan Charest, and Rickey Dubay. Non-linear model predictive control schemes with application on a 2 link vertical robot manipulator. *Robotics and Computer-Integrated Manufacturing*, 41:23–30, 2016.

[91] Andong Yi, Ahmad Zahedi, Yansong Wang, U-Xuan Tan, and Dingguo Zhang. A novel exoskeleton system based on magnetorheological fluid for tremor suppression of wrist joints. In *2019 IEEE 16th International conference on rehabilitation robotics (ICORR)*, pages 1115–1120. IEEE, 2019.

[92] Chunzhi Yi, Feng Jiang, Shengping Zhang, Hao Guo, Chifu Yang, Zhen Ding, Baichun Wei, Xiangyuan Lan, and Huiyu Zhou. Continuous prediction of lower-limb kinematics from multi-modal biomedical signals. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):2592–2602, 2021.

[93] Yuichi Yoshii, Hiroshi Yuine, Ohashi Kazuki, Wen-lin Tung, and Tomoo Ishii. Measure-

ment of wrist flexion and extension torques in different forearm positions. *Biomedical engineering online*, 14:1–10, 2015.

[94] Ahmad Zahedi, Bin Zhang, Andong Yi, and Dingguo Zhang. A soft exoskeleton for tremor suppression equipped with flexible semiactive actuator. *Soft robotics*, 8(4):432–447, 2021.

[95] Kirsten E Zeuner and Christos Sidiropoulos. Cognitive behavioral therapy in functional tremor: A promising treatment approach, 2019.

[96] Dingguo Zhang and Wei Tech Ang. Reciprocal emg controlled fes for pathological tremor suppression of forearm. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4810–4813. IEEE, 2007.

[97] Dingguo Zhang, Philippe Poignet, Antonio PL Bo, and Wei Tech Ang. Exploring peripheral mechanism of tremor on neuromusculoskeletal model: A general simulation study. *IEEE transactions on biomedical engineering*, 56(10):2359–2369, 2009.

[98] Yue Zhou, Anas Ibrahim, Kenneth G Hardy, Mary E Jenkins, Michael D Naish, and Ana Luisa Trejos. Design and preliminary performance assessment of a wearable tremor suppression glove. *IEEE Transactions on Biomedical Engineering*, 68(9):2846–2857, 2021.

Appendices

Appendix A

Neural Network models and code snippets for Data-driven modeling

This appendix presents detailed information about the neural network models built in section 2.3

The first model is the baseline model, which is a standard MLP (multilayer perceptron) model. The neural network has 2 hidden layers, with 1000 neurons and 500 neurons. The sequential API in Tensorflow was used to build this.

```
#optimizer and learning rate
opt = keras.optimizers.Adam(learning_rate=0.001)

model_dense = Sequential()
model_dense.add(Flatten(input_shape=(n_steps_in, n_features)))
model_dense.add(Dense(1000, activation='tanh'))
model_dense.add(Dense(500, activation='tanh'))
model_dense.add(Dense(n_steps_out, activation='tanh'))
model_dense.compile(loss='mse', optimizer=opt)
```

Listing A.1: Standard MLP Model

The CNN models involve Conv1D layers, a convolution kernel convolved with its input over a single spatial (or temporal) dimension. Batch normalization and max pooling layers are

also included for training stability, faster convergence, and adding regularization to avoid overfitting. The batch normalization layer normalizes the activations of a layer within a mini-batch during the training, avoiding an internal covariance shift. Max pooling is a downsampling method that reduces the spatial dimensions by selecting the maximum value within each small window or region. The flatten layer before the final output layer reshapes the tensor to 1D, to be connected to the final dense layer. It should be noted that when we use CNN or LSTM layers, they return a matrix. The final dense layer can not process a matrix, instead it has to be a vector. Thus the flatten layer is applied just to open up the 2D matrix and represent it as a 1D vector.

```
model_cnn1 = Sequential()
model_cnn1.add(Conv1D(filters=32, kernel_size=5, activation='tanh',
    input_shape=(n_steps_in, n_features)))
model_cnn1.add(BatchNormalization())
model_cnn1.add(MaxPooling1D(pool_size=2))
model_cnn1.add(Flatten())
model_cnn1.add(Dense(n_steps_out,activation='tanh'))
```

Listing A.2: CNN 1 Model

```
model_cnn2 = Sequential()
model_cnn2.add(Conv1D(filters=32, kernel_size=5, activation='tanh',
    input_shape=(n_steps_in, n_features)))
model_cnn2.add(BatchNormalization())
model_cnn2.add(MaxPooling1D(pool_size=2))
model_cnn2.add(Conv1D(filters=64, kernel_size=5, activation='tanh'))
model_cnn2.add(BatchNormalization())
model_cnn2.add(MaxPooling1D(pool_size=2))
model_cnn2.add(Flatten())
model_cnn2.add(Dense(n_steps_out,activation='tanh'))
```

Listing A.3: CNN 2 Model

```
model_LSTM.add(LSTM(32, input_shape=(n_steps_in, n_features), activation='tanh
    ', return_sequences=True))
model_LSTM.add(Flatten())
model_LSTM.add(Dense(n_steps_out, activation='tanh'))
```

Listing A.4: LSTM 1 Model

For the sequential CNN-LSTM models described below, the outputs of the CNN networks are connected to further layers of LSTM networks.

```
model_seq1 = Sequential()
model_seq1.add(Conv1D(filters=32, kernel_size=5, activation='tanh',
    input_shape=(n_steps_in, n_features)))
model_seq1.add(BatchNormalization())
model_seq1.add(MaxPooling1D(pool_size=2))
model_seq1.add(LSTM(32, activation='tanh', return_sequences=True))
model_seq1.add(Flatten())
model_seq1.add(Dense(n_steps_out,activation='tanh'))
model_seq1.compile(loss='mse', optimizer=opt)
```

Listing A.5: Sequential CNN-LSTM 1 Model

```
model_seq2 = Sequential()
model_seq2.add(Conv1D(filters=32, kernel_size=5, activation='tanh',
    input_shape=(n_steps_in, n_features)))
model_seq2.add(BatchNormalization())
model_seq2.add(MaxPooling1D(pool_size=2))
model_seq2.add(Conv1D(filters=64, kernel_size=5, activation='tanh'))
model_seq2.add(BatchNormalization())
model_seq2.add(MaxPooling1D(pool_size=2))
```

```
model_seq2.add(LSTM(32, activation='tanh', return_sequences=True))
model_seq2.add(LSTM(32, activation='tanh', return_sequences=True))
model_seq2.add(Flatten())
model_seq2.add(Dense(n_steps_out,activation='tanh'))
model_seq2.compile(loss='mse', optimizer=opt)
```

Listing A.6: Sequential CNN-LSTM 2 Model

Constructing the Y-networks involved use of both the functional and sequential APIs of tensorflow. The functional API is necessary here to combine the 2 branches to give a single output.

```
# Define left branch of the Y-network
left = Sequential()
left.add(Conv1D(filters=32, kernel_size=5, activation='tanh', input_shape=(
    n_steps_in, n_features)))
left.add(BatchNormalization())
left.add(MaxPooling1D(pool_size=2))
left.add(Dropout(0.2))
left.add(Flatten())
left.add(Dense(n_steps_out))

# Define right branch of the Y-network
right = Sequential()
right.add(LSTM(32, activation='tanh',input_shape=(n_steps_in, n_features),
    return_sequences=True))
right.add(Flatten())
right.add(Dense(n_steps_out, activation = 'tanh'))

# Merge the branches
concatenated = Concatenate()([left.output, right.output])
top = Dense(100, activation='tanh')(concatenated)
```

```
model_y1 = Model(inputs=[left.input, right.input], outputs=top)
```

Listing A.7: Y CNN-LSTM 1 Model

```
# Define left branch of the Y-network
left = Sequential()
left.add(Conv1D(filters=32, kernel_size=5, activation='tanh', input_shape=(
    n_steps_in, n_features)))
left.add(BatchNormalization())
left.add(MaxPooling1D(pool_size=2))
left.add(Dropout(0.2))
left.add(Conv1D(filters=64, kernel_size=5, activation='tanh'))
left.add(BatchNormalization())
left.add(MaxPooling1D(pool_size=2))
left.add(Flatten())
left.add(Dense(n_steps_out))

# Define right branch of the Y-network
right = Sequential()
right.add(LSTM(32, activation='tanh',input_shape=(n_steps_in, n_features),
    return_sequences=True))
right.add(LSTM(32, activation='tanh', return_sequences=True))
right.add(Flatten())
right.add(Dense(n_steps_out, activation = 'tanh'))

# Merge the branches
concatenated = Concatenate()([left.output, right.output])
top = Dense(100, activation='tanh')(concatenated)
model_y2 = Model(inputs=[left.input, right.input], outputs=top)
```

Listing A.8: Y CNN-LSTM 2 Model

The below code snippet shows the development of the custom loss function used to embed the first-order and second-order derivatives of the time-series into the training. During training, the standard mean-squared error loss is replaced by this custom loss.

```
def custom_loss(y_true, y_pred):

  mse_loss = K.mean(K.square(y_true[0:10,0:100] - y_pred[0:10,0:100]))
  lamb = 1
  vel_loss = 0
  for i in range(10):
    for j in range(99):
      vel_loss = vel_loss + K.square(((y_true[i,j+1] - y_true[i,j])-(y_pred[i,
   j+1] - y_pred[i,j])))
  mean_vel_loss = vel_loss/990

  lamb2 = 0.1
  acc_loss = 0
  for i in range(10):
    for j in range(98):
      acc_loss = acc_loss + K.square(((y_true[i,j+2] - y_true[i,j+1])-(y_pred[
   i,j+2] - y_pred[i,j+1]))-((y_true[i,j+1] - y_true[i,j])-(y_pred[i,j+1] -
   y_pred[i,j])))
  mean_acc_loss = acc_loss/980

  return mse_loss + lamb*mean_vel_loss + lamb2*mean_acc_loss
```

Listing A.9: Custom Loss