2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC) | 978-1-7281-1179-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/EMBC46164.2021.9630810

Control of a lower limb exoskeleton using Learning from Demonstration and an iterative Linear Quadratic Regulator Controller: A simulation study

Nathaniel Goldfarb[1], Haoying Zhou[1], Charles Bales[1], and Gregory S. Fischer[1]

Abstract—

Lower limb exoskeletons have complex dynamics that mimic human motion. They need to be able to replicate lower limb motion such as walking. The trajectory of the exoskeleton joints and the control signal generated are essential to the system's operation. Current learning from demonstration methods has only been combined with linear quadratic regulators; this limits the applicability of processes since most robotic systems have non-linear dynamics. The Asynchronous Multi-Body Framework simulates the dynamics and allows for real-time control. Eleven gait cycle demonstrations were recorded from volunteers using motion capture and encoded using Task Parameterized Gaussian mixture models. An iterative linear quadratic regulator is used to find an optimal control signal to drive the exoskeleton joints through the desired trajectories. A PD controller is added as a feed-forward control component for unmodeled dynamics and optimized using the Bayesian Information Criterion. We show how the trajectory is learned, and the control signal is optimized by reducing the required bins for learning. The framework presented produces optimal control signals to allow the exoskeleton's legs to follow human motion demonstrations.

Index Terms— **TPGMM, iLQR, learning from demonstration, optimal control**

I. Introduction

Lower limb exoskeletons have complex non-linear dynamics and are required to follow human-like trajectories. Therefore human-like motion is difficult to reproduce using hand-coded trajectories. It is advantageous to use demonstrations of the desired movement. Motion capture and Learning from Demonstration (LfD) allow for human motion to be used to generate trajectories to be encoded for robotics systems to follow [1], this enables gait cycles to be recorded and reproduced on lower limb exoskeleton systems. A controller is still required to generate control commands to follow these desired trajectories. Classical control systems are typically designed by tuning parameters to produce an acceptable response, often without attempting to minimize torques or other parameters. On the other hand, an optimal controller provides a mathematical method to minimize torques and deviations.

This paper presents a method that combines LfD and an optimal controller to control a lower limb exoskeleton's legs. A motion capture system collected gait cycle demonstrations. These demonstrations are parsed, encoded, and reproduced using Task Parameterized Gaussian mixture models (TPGMM). An iterative Linear Quadratic Regulator (iLQR) is used to calculate optimal control signals using a linear cost function to minimize the torques and reduce the path deviation error. A PD controller is added to aid with unmodeled dynamics and disturbance rejection. The controller is tested on a lower limb exoskeleton simulation, LARRE (Legged Articulated Robotic Rehab Exoskeleton). The Asynchronous Multi-Body Framework (AMBF) is used to simulate the dynamics and enable low computational overhead [2].

Manuscript received July 21, 2021
This work does not have external financial support

[1] Robotics Engineering Department at Worcester Polytechnic Institute, 50 Prescott St, Worcester, MA 01609 nagoldfarb@wpi.edu, hzhou6@wpi.edu, gfischer@wpi.edu

II. Background

A. Learning Trajectories from Human Motion

Motion capture (mocap) is one tool that can record human kinematics and kinetics. Mocap allows for the sub-millimeter measurement of markers on a body, allowing for complex motion trajectories to be captured. In [3], Chalodhorn *et. al* mapped mocap markers from a human to a robot to train the system to walk. They used the location of the markers to the motion of the body using a model-free approach. The mocap marker data was mapped from the human to the robot to calculate motor commands to move the robot. The model was optimized to follow the trajectories while satisfying the kinematic constraints of the system. In [4] they mapped human motion to a bipedal robot by passing the parameters through a classical zero moment point controller. These approaches allow for mapping either the joint angles of the marker positions to train a robot.

Learning from Demonstration (LfD) or Programming from Demonstration (PfD) is a powerful method to use human demonstrations to control a robot's movement. It has a wide range of applications in the field of robots. It can be utilized on adaptive bimanual tracking behaviors learned from a single demonstration. LfD has been used to reproduce hand-clapping [5], chess-playing, and holding objects such as sugar cubes and cups [6] based on demonstration and full-body movements [7].

Task Parameterized Gaussian mixed models (TPGMM) allows for the encoding of multiple demonstrations into clusters. TPGMM probabilistically encodes the changing relevance of candidate frames throughout the task [8] similar to Gaussian mixed models. It adapts regression trajectories based on parameters: positions and orientations of

the frames. Gaussian Mixture Regression (GMR) then regresses over the TPGMM outputs to retrieve output features. TPGMM/GMR provides a robust framework for learning and encoding human motion for reproduction on robotics systems. Dynamic Movement Primitives(DMP) allows for the reproduction of tasks [9] [10] . The classical form of DMPs relies on a single demonstration for encoding and reproduction. However, this lacks the ability to build model demonstrations, which allows the reproduction model to be more versatile. By combining DMPs with TPGMM/GMR, multiple demonstrations can train and reproduce a generalized trajectory.

B. Optimal Controllers for Robotic Systems

The Linear Quadratic Regulator (LQR) is a well-known method that provides optimally controlled feedback gains to enable the closed-loop, and high-performance control [11]. The limitation of LQR is that it can only account for linear systems and linear cost functions. In [6], Calinon *et. al* used TPGMM/GMR with an LQR controller to develop a minimal intervention controller. This controller was used to control a Kuka robotic arm [12] through a series of movements. While this method worked well, the controller does not directly model the dynamics of the robot. It instead models a mass-spring-dampener system and controls this simplified system. This method leads the inspiration to use a non-linear controller instead of the linear one used previously.

The Iterative Linear Quadratic Regulator (iLQR) is a non-linear version of the LQR controller. The iLQR is an iterative process that uses a Taylor approximation of the dynamics and cost function to find a local linear model. The dynamics and cost function are linearized in the forward pass of the system using a shooting method, while like the typical LQR, the backward pass calculated the optimal gain and cost [13]. Differential Dynamic Programming(DDP) is a similar method to iLQR; in classical DDP, the second-order terms are costly operations [14] [15]. This modification to LQR allows the control of non-linear system control problems; this is useful because it expands the systems the LQR can be applied to, including biological movement system [16] and online trajectory optimization [17]. iLQR compared to ODE solvers, gradient descent methods, and differential dynamic programming converges substantially faster and finds slightly better solutions [16].

iLQR controllers have been implemented on a wide variety of systems. In [18] they used a modified form of iLQR called constrained iLQR to control the motion of a car. The car was subjected to several state and control constraints. iLQR also allows for the control of humanoid robots. Tassa *et. al* used iLQR to control an HRP-2 robot's motion by controlling the joint angles [14]. These methods show the applicability of the iLQR controllers for biological control. They provide optimal control commands for biological systems.

C. Asynchronous Multi-Body Framework

AMBF is a multi-body simulation framework based on a front-end description format. This high-performance dynamic simulation supports complex open-loop and closed-loop systems focusing on medical and industrial robotic systems. The simulated object definitions are written in the AMBF Description File (ADF) file using the YAML [1] format and allows the definition of various attributes of the robot's components [2]. The components include the bodies, joints, sensors, and actuators. Each component has attributes that could include the kinematic, inertial, collision, controller, communication, material, hierarchical attributes, etc. Some of these attributes are communicated over the ROS middleware that is then accessible via the Python Client. The AMBF environment reduces the computation overhead and allows the modeling of the complex robotic system and for controllers to be tested and implemented.

III. Methods

A. Overview

The proposed approach is split into several phases; demonstration, encoding, and optimization. During the demonstration phase, gait data is collected, and the gait cycles are parsed to extract the joint angles. The demonstrations are encoded using TPGMM/GMR. LARRE and the wearer are assumed to be non-linear and modeled by Equation 1 where M is the mass-inertia matrix, C is the Coriolis terms, G is the gravity terms. The control signal is calculated using an iLQR controller. This control signal is used to control LARRE in AMBF.

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{1}$$

B. Collection of Demonstration Data

During the demonstration phase, a trial was conducted to build a library of gait cycles. Eleven able-bodied subjects participated in the study (5F, 6M). The Institutional Review Board of Worcester Polytechnic Institute approved the study, and each subject gave written consent. The data was anonymous, so the subjects could not be identified. All the data was saved in an ASCII format file.

A Vicon Vantage V5 motion capture system[2] was used with a total of 10 cameras recording at $100Hz$. The lower-body plug-in gait model was used to record the lower body dynamics of the subject. Additional rigid body plates were placed on the thighs, shanks, and back of the participants for redundancy and to mitigate marker occlusion. The marker placement was calibrated to each subject prior to the study. The subjects were instructed to walk approximately three meters in a straight line at their own pace; this was repeated three times to ensure that a good sample trial with no marker occlusion was collected. The gait cycles of the subjects were parsed and compared. Figure 1 shows the joint angles extracted from a gait cycle of the subjects using a custom package[3]. The raw data is also provided open source for community[4].

[1] https://yaml.org
[2] https://www.vicon.com/
[3] https://github.com/WPI-AIM/AIM_GaitAnalysisToolkit
[4] https://github.com/WPI-AIM/AIM_GaitData

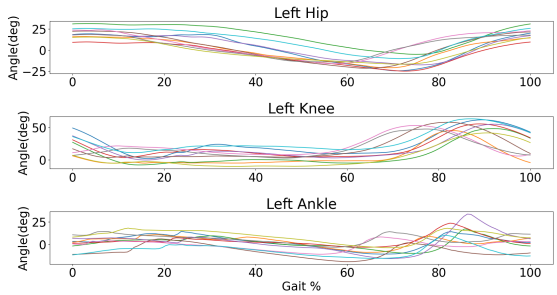

Fig. 1: Gait Cycles extracted from the subjects. Each line is a separate gait cycle from a different subject

C. Learning the Gait Motion

Equation 2 shows the basic formulation of the DMP, which defines a stable underlining model and adds a forcing function to drive the model. In this equation, β_y and α_y are the system gains, y is the system state, and g is the goal. The canonical dynamical system denoted x decreases from $1 \rightarrow 0$. The force function f is a non-linear function that pulls the canonical system along the trajectories [19].

$$\begin{aligned} \ddot{y} &= \alpha_y(\beta_y(g-x) - \dot{y}) + f \\ \dot{x} &= -\alpha_x x \end{aligned} \tag{2}$$

During the encoding phase, the demonstrations are encoded using the TPGMM algorithm. The model is defined by $\{\pi_i\{\mu_i^j, \Sigma^j\}_{j=1}^P\}$, where π_i are the mixing coefficient, μ, is the mean, and Σ is the covariance matrix of the Gaussian. The expectation-maximization (EM) algorithm is an iterative algorithm used to maximize the likelihood. K-means initializes the TPGMM algorithm, which is used to guess the values μ_{0-K}, Σ_{0-K}, where K is the number of Gaussian to place on the data set. The Bayesian Information Criterion (BIC) calculates the number of bins K [20]. The number of bins is critical to set because if too few bins are used, the model will be poorly fit; if too many are used, it will over-fit the demonstrations. Equation 3 calculates the BIC score; it is a trade-off between optimizing the likelihood and minimizing the number of states to encode. $\mathcal{L}$ is the log-likelihood, N is the number of mixture models, K is the number of components, and D is the dimension of the data points. [20] [21].

$$S_{BIC} = -\mathcal{L} + \frac{log(N)(K(D+1)(D+2)-2)}{4} \tag{3}$$

The EM algorithm is an iterative process that maximizes the likelihood value γ. The E-step (Equation 4) calculates the new likelihood with the current parameters, while the M-step (Equation 5) updates the parameters given the new likelihood.

E(xpected):

$$\gamma_{n,i} = \frac{\pi_i \prod_{j=1}^P \mathcal{N}(X_n^j | \mu_i^j, \Sigma_i^j)}{\sum_{k=1}^K \pi_k \prod_{j=1}^P \mathcal{N}(X_n^j | \mu_i^j, \Sigma_i^j)} \tag{4}$$

M(aximization):

$$\begin{aligned} \pi_i &= \frac{\sum_t^N \gamma_{t,i}}{N} \\ \mu_i^j &= \frac{\sum_t^N \gamma_{t,i} X_t^j}{\sum_t^N \gamma_{t,i}} \\ \Sigma_i^j &= \frac{\sum_t^N \gamma_{t,i}(X_t^j - \mu_i^j)(X_t^j - \mu_i^j)^T}{\sum_t^N \gamma_{t,i}} \end{aligned} \tag{5}$$

Each of the demonstrations $d \in \{1...D\}$ is a vector of time-sequenced data points creating a data structure of size $\sum_{d=1}^D T_d$. The length of each of the T_i vectors will be different for each demonstration; this results from the subjects' stride speeds. The demonstrations are aligned using the dynamic time warping algorithm; this ensures that the features demonstrations' features are aligned. The task parameters are represented as a coordinate system P defined at each time step n defined by $\{b_{n,j}, A_{n_j}\}_{j=1}^P$. The b vector is a set of basis vectors from the origin with the transformation matrix A. This formulation allows for trajectories to be transformed into any frame using Equation 6. ξ is the vector of the data points.

$$X_t^j = A_{t,j}^{-1}(\xi_t - b_{t,j}) \tag{6}$$

GMR models the regression function from the joint probabilities calculated by TPGMM. GMR regress over the basis function, creating the forcing function (f) needed for the seen in the DMP formulation. Equation 7 calculates the likelihood, and Equation 8 calculates the covariance and mean, where ξ is a multidimensional array, μ_i^o, and Σ_t^o are vectors of the output mean, and covariance, μ_i^I, and Σ_t^I are vectors of the input mean and covariance.

$$P(\xi_t^O | \xi_t^I) \sim \sum_i^K h_i(\xi_t^I) \mathcal{N}(\hat{\mu_i^o}, \hat{\Sigma_t^o}) \tag{7}$$

where,

$$\begin{aligned} \hat{\mu_i^o} &= \mu_i^o + \Sigma_i^{OI} \Sigma_i^{I-1}(\xi_t^I - \mu_i^I) \\ \hat{\Sigma_i^O} &= \Sigma_i^O - \Sigma_i^{OI} \Sigma_i^{I-1}(\xi_t^I - \mu_i^I) \\ h_i &= \frac{\pi_i \mathcal{N}(\xi_t^I | \mu_i^j, \Sigma_i^j)}{\sum_k^K \pi_k \mathcal{N}(\xi_t^I | \mu_k^j, \Sigma_k^j)} \end{aligned} \tag{8}$$

D. Optimization of the Joint Control Commands

During the optimization phase, the control inputs calculated drive the system along the trajectory. There are two steps in the iLQR algorithm; a forward pass and a backward pass. In this forward pass, the simulated forward LARRE and wearer system is simulated forward along the trajectory using a dynamic model. LARRE's dynamics are defined using the Rigid Body Dynamics Library (RBDL) [22]. This library uses a Newton-Euler approach and Featherstone dynamics to solve the forward, and inverse dynamics of a system [23].

Runge Kutta 4 (RK4) integrates the system forward to obtain the next state of the system [24]. RK4 is a numerical integration method that perturbs the system around a point

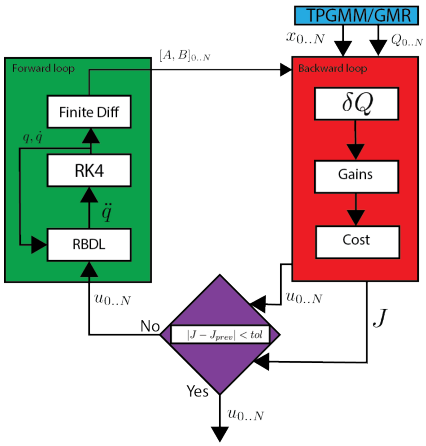

Fig. 2: Diagram of how the iLQR algorithm works with the forward pass and backward pass. The loop exists when the difference in current cost and previous cost is below a tolerance.

and uses an average response average value. In the backward pass, the system is solved backward to update the control parameters. A modified version of the open-source library[5] was used and implemented. Figure 2 shows a diagram of how the iLQR algorithm works. The algorithm continues back and forth until the cost J converges, indicating that the control signal u can drive the system along the desired trajectory.

Equation 9 defines the generalized non-linear system dynamics. The cost function is the sum of the running cost and the terminal cost shown in Equation 10. This paper uses a linear cost function to take advantage of the TPGMM/GMR process shown in Equation 11, where $\tilde{x}_i = x_i - x_i^d$. This paper's cost function is designed to follow a reference trajectory x^d using GMR. The Q_i varies along the path and are calculated by the TPGMM algorithm ($Q_i = \Sigma_i^{-1}$). The Q matrix is the weight for transiting, and the R matrix is the weight of the control.

$$x_{i+1} = f(x_i, u_i) \tag{9}$$

$$J(x, U) = \ell_f(x_N) + \sum_{i=0}^{N-1} \ell(x_i, u_i) \tag{10}$$

where,

$$\begin{aligned} \ell(x_i, u_i) &= \tilde{x}_i^T Q_i \tilde{x}_i + u_i^T R u_i \\ \ell_f(x_N) &= x_N^T Q_N x_N \end{aligned} \tag{11}$$

The value function is found using Equation 12 which is minimized over the entire control sequence. Using calculus of variations Equation 13 is found which is decomposed into Equation 14, where $A = \partial f/\partial x$ and $B = \partial f/\partial u$. The particles derivatives are calculated in the forward pass at each time step using a finite difference method [15].

$$\begin{aligned} Q(x, u) &= \ell(x, u) + V(f(x, u, i+1) \\ V(x, i) &= \min_u Q(x, u) \end{aligned} \tag{12}$$

[5]https://github.com/anassinator/ilqr

$$\delta Q = \frac{1}{2} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix} \tag{13}$$

$$\begin{aligned} Q_x &= \ell_x + A^T V_x' \\ Q_u &= \ell_u + B^T V_x' \\ Q_{xx} &= \ell_{xx} + A^T V_{xx}' A \\ Q_{uu} &= \ell_{uu} + B^T V_{xx}' B \\ Q_{ux} &= \ell_{ux} + B^T V_{xx}' A \end{aligned} \tag{14}$$

The optimization finds the total cost and the optimal control gains for the system. The control sequence is found using Equation 15, where $K = -Q_{uu}^{-1} Q_{ux}$ and $k = -Q_{uu}^{-1} Q_{ux}$. The α term is a linear search term to ensure convergence of the system, and $\hat{u}_i$ is the nominal control input. Here k is a feed-forward term, and K is the gain for the feedback term.

$$u_i = K_i(x_i - \hat{x}_i) + \alpha k_i + \hat{u}_i \tag{15}$$

A PD controller was added into the loop to allow for error tracking in real-time; the PD controller handles errors in un-modeled dynamics suck as joint friction and damping [14]. The iLQR torque acted as a feed-forward term driving the system, and the PD controller handled path deviation errors. Figure 3 show the control diagram.

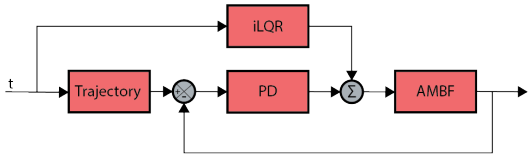

Fig. 3: Control diagram of the exoskeleton. The trajectory is found using TPGMM. The iLQR provides a feedforward control input. The PD controller removes un-modeled dynamics of the system

E. Training R matrix

As discussed above, the TPGMM process finds optimal values for the Q_i along the trajectory, which is the weighting of the system's state. However, this does not provide insight into the form of the R matrix, which is the weighting of the control input into the system. These values are critical because they determine the amount of effort applied at every point along the trajectory. The shape of the R matrix for this application 6×6 diagonal matrix. The first three diagonal elements of the R matrix are correlated to the control input of the left leg (*hip*, *knee*, *ankle*). The other three diagonal elements are related to the control input of the right leg (*hip*, *knee*, *ankle*). Since the mechanical structures of both legs are identical, therefore it can assume that the exoskeleton has mechanical symmetry of the joints for the left and right legs i.e. hip_R == hip_L. This assumption is made because each leg would have similar masses and controlled with identical motors. In addition, the minor differences can be supplemented by the Q matrix during the iLQR training.

To find the R matrix's values; the values were iteratively changed to find a matrix that minimizes cost J defined in Equation 16, where N is the number of points for output, x_i are the points on the desired trajectory, and $\hat{x}_i$ are the points on the actual trajectory. [25].

$$J = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}} \tag{16}$$

The high dimensionality and non-linear dynamics make it challenging to weight values of the matrix [26]. The complexity of the motor behaviors also increases the difficulty of finding the R matrix. Therefore, a brute-force method was implemented to go through values in different magnitudes and select the optimal result. The control signal was tested by forward integrating using RK4. Figure 4 show the optimal trajectory, which uses the optimal R values, and one poor case, which uses different R values that cannot make the generated trajectory fit the desired trajectory well. This trajectory did not follow the desired motion and was deemed unacceptable. Changing the values of the R matrix has a significant effect on the replication of the trajectory. The R matrix must be tuned in order to replicate the desired trajectory.

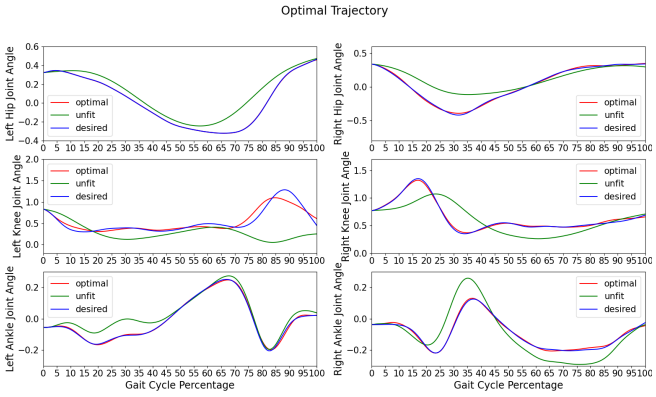

Fig. 4: Joint Angles for the trajectories altered by changing values of the R matrix. Showing the importance of the R matrix in the iLQR algorithm. The blue line is the desired motion, the red line is well fit trajectory found, and the green line is a poorly fit trajectory.

F. Simulation of the system

The LARRE model and all its components were designed in CAD. The exoskeleton's model consists of various sub-assemblies simplified into the hip, thigh, shank, and foot components for both the left and right sides, respectively, and exported as STLs. The inertia and mass of each of the segments were also calculated using CAD. A scaled human model inserted into the exoskeleton allowed for joint alignment and sizing of the limb lengths. We assumed that the human had a mass of $80Kg$ [27]; this is roughly the mass for an average North American male. The human joints' mass and inertia were calculated using human anatomical data [28]. The exported STLs were brought into Blender[6], and the ambf_addon[7] were used to create the joints and assign the dynamics properties to the bodies. The human model and exoskeleton were connected. A pair of passive crutches were added to the system to help support and balance the body trunk. The ADF file, exported using the ambf_addon, contains all the kinematics and dynamic information to initialize the model in AMBF. Figure 5 shows LARRE standing in the AMBF environment.

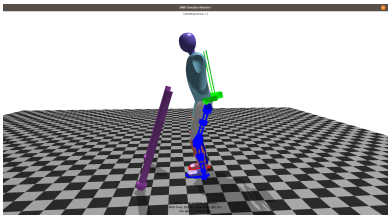

Fig. 5: AMBF model of exoskeleton

A modular control interface was used to control LARRE in AMBF. This control interface is beyond the scope of this paper. The simulation environment was set to run at $1000Hz$, and the controller runs at $500Hz$. LARRE's joint position and velocity are collected in each iteration of the loop, and a torque command is sent back, allowing for feedback control.

IV. Results

The number of bins was calculated using the BIC score, which was calculated several times to ensure converges. Figure 6 shows the BIC score calculation calculated over several iterations using 5-30 bins, this range of bins sizes was experimentally determined. Due to the K-means randomization step in the EM algorithm, there is some expected variation. However, it has been shown that the optimal number of bins converges to be around 15 bins. Using the results of BIC, the initial TPGMM algorithm used 15 bins. The output of the TPGMM/GMR process is shown in Figure 7. In this figure, the blue lines are the underlining demonstrations used to train the forcing function. The red dots and green ovals are the mean and covariances of bins, respectively. The TPGMM algorithm calculates the size and placement of the bins. Smaller major and minor axis represent groupings of smaller variation. Figure 8 show the comparison of the learned model compared to the training demonstrations. The thin lines are the training demonstrations; the thick line is the trained model.

The path and Qs are generated by the TPGMM/GMR process and initialize the iLQR controller algorithm. As the name implies, the iLQR algorithm is an iterative possesses that breaks when the cost J converges. Figure 9 shows the converges of the cost for each iteration. The cost coverage's from $\sim 47.5 \rightarrow \sim 27.5$ after 6 iterations.

Figure 10 shows a comparison of the exoskeleton joints to the reference trajectory. The orange line is referencing trajectory, and the blue line is the path the exoskeleton joints

[6]https://www.blender.org/
[7]https://github.com/WPI-AIM/ambf_addon

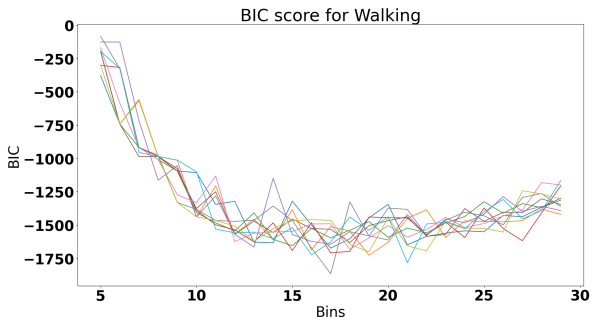

Fig. 6: The bayesian information criterion score that determines the number of bins. Each line indicates an iterations of measuring different bins sizes. Multiple trials were conducted to locate the score and ensure converges.

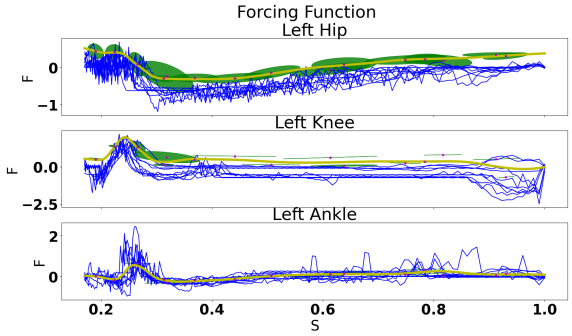

Fig. 7: Forcing Function Learned for each of the trajectories. The red dots are the means and the green ovals are the covariances.

traveled. LARRE's joints were able to track the desired motion. Figure 11 shows a comparison of the joint effort over the trajectory. Both the iLQR feed-forward term and the total torque (iLQR+PD) are presented. Additionally, the effort of a pure PD controller is presented for comparison of effort. This is not the same PD effort used for the total torque (*orange* + *green* $\neq$ *blue*) The iLQR controller encodes the known non-linear dynamics of LARRE and the person, where a pure PD controller does not.

V. Discussion

This approach in developing a controller has taken advantage of several different algorithms. The TPGMM process offers a comprehensive method to learn complex human motions from multiple demonstrations, using the GMR algorithm to extract the motion model to replicate the desired motion. This method bypasses the limitations of DMPs trained from a single demonstration. The number of bins used in TPGMM is optimized using the BIC algorithm. Additionally, the bins' placement and size are calculated using the EM steps of the TPGMM algorithm.

The iLQR algorithm allows for the generation of optimal control sequences for non-linear systems. This modification of the vanilla LQR controller is vital for robotics systems since robotic systems are usually modeled as non-linear systems. The iLQR algorithm is also a natural extension of

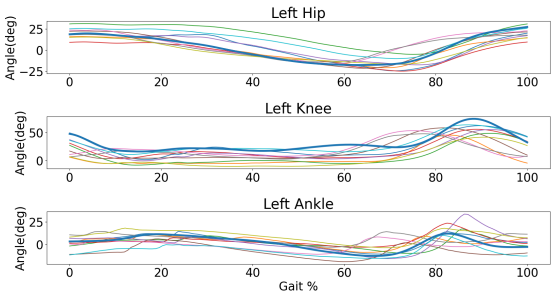

Fig. 8: Learned model compared to the demonstrations. The thin lines are the training demonstrations and the thick line is the learned model.

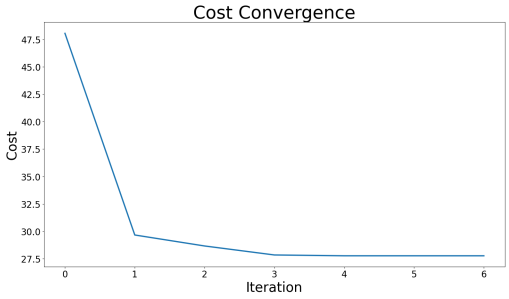

Fig. 9: Converges of the iLQR controller cost at each loop iteration. The cost coverage's from $\sim 47.5 \rightarrow \sim 27.5$ after 6 iterations.

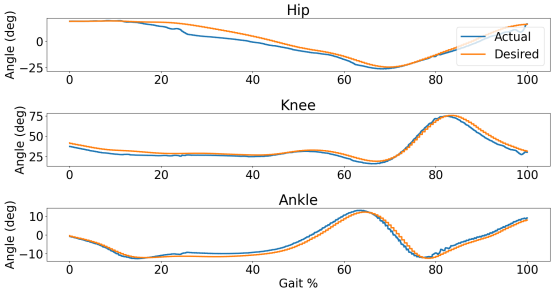

Fig. 10: The trajectory followed using the iLQR controller compared to the reference trajectory. The blue line is the actual motion and the orange is the reference trajectory

the TPGMM/GMR algorithm. It provides a comprehensive method of building a trajectory and the values of the Q_k matrix along the trajectories. Without the TPGMM step, the Q_k matrices would have to be hand-tuned. The control signal generated by the iLQR controller was able to drive the joints of LARRE along the desired trajectories. The PD controller in the loop allowed for feedback to eliminate any system errors or unmodeled dynamics. This result is best shown in the first 20% of the gait cycle of the knee and ankle efforts; the PD controller primarily generates the total torque; however, the iLQR tracks the total effort over the remaining control sequence. The large PD controller torque at the beginning of the trajectory is due to misalignment in the starting condition. The PD controller can drive the

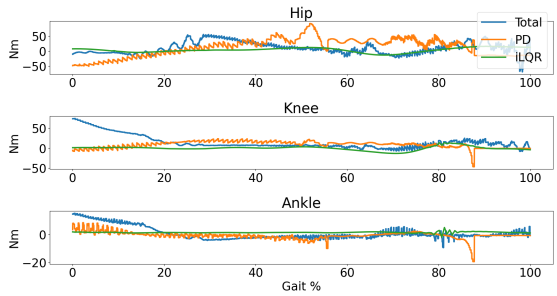

Fig. 11: Comparison of the PD controller to the iLQR torque and the total torque. (*orange* + *green* $\neq$ *blue*)

system towards the iLQR control input and maintain the lower control input. This result indicates that most of the torque is generated by the iLQR controller over the PD controller. Compared to the vanilla PD controller, the control sequence is smooth, with fewer spikes in the control input.

VI. Conclusion

This paper introduced a method of integrating an optimal controller with learning from demonstration to control a lower limb exoskeleton. This method allows a trajectory to be learned from multiple demonstrations and uses a non-linear dynamic model to learn an optimal control signal. Using a non-linear model allows for the complex dynamics to be encoded into the control signal. The PD controller allows feedback to account for non-modeled dynamics. This work can be used for building a gait controller for lower limb exoskeletons. Using human demonstrations and integrating them with a controller allows a seamless pipeline from demonstration to control. Future work includes using an online model predictive controller to allow for optimal feedback, allowing for online control, and finding optimal values for the R matrix. This work will allow every controller process to be optimized and for the controller to be used on physical systems.

Acknowledgement

Nathaniel Goldfarb is supported by the SMART fellowship. The authors don't have any personal or financial conflicts of interest.

References

[1] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97. Citeseer, 1997, pp. 12–20.

[2] A. Munawar, Y. Wang, R. Gondokaryono, and G. S. Fischer, "A real-time dynamic simulator and an associated front-end representation format for simulating complex robots and environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 1875–1882.

[3] R. Chalodhorn, D. B. Grimes, K. Grochow, and R. P. Rao, "Learning to walk through imitation."

[4] K. Hu, C. Ott, and D. Lee, "Online human walking imitation in task and joint space based on quadratic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3458–3464.

[5] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 323–329.

[6] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[7] R. Chalodhorn, D. B. Grimes, K. Grochow, and R. P. Rao, "Learning to walk through imitation." in *IJCAI*, vol. 7, 2007, pp. 2084–2090.

[8] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3339–3344.

[9] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[10] S. Schaal, J. Peters, J. Nakanishi, and A. J. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," in *Workshop on Bilateral Paradigms on Humans and Humanoids: IEEE International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003, pp. 1–21.

[11] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.

[12] G. Schreiber, A. Stemmer, and R. Bischoff, "The fast research interface for the kuka lightweight robot," in *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA 2010)*. Citeseer, 2010, pp. 15–21.

[13] B. Jackson, "AL-iLQR Tutorial," https://bjack205.github.io/papers/AL_iLQR_Tutorial.pdf, 2019.

[14] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[15] Z. Manchester and S. Kuindersma, "Derivative-free trajectory optimization with unscented dynamic programming," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 3642–3647.

[16] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems."

[17] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[18] J. Chen, W. Zhan, and M. Tomizuka, "Constrained iterative lqr for on-road autonomous driving motion planning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–7.

[19] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[20] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[21] A. G. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 370–384, 2006.

[22] M. L. Felis, "Rbdl: an efficient rigid-body dynamics library using recursive algorithms," *Autonomous Robots*, vol. 41, no. 2, pp. 495–511, 2017.

[23] R. Featherstone and D. Orin, "Robot dynamics: equations and algorithms," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 826–834.

[24] J. A. dit Sandretto, "Runge-kutta theory and constraint programming," *Reliable Computing*, vol. 25, pp. 178–201, 2017.

[25] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[26] I.-W. Park, K.-B. Lee, and J.-H. Kim, "Multi-objective evolutionary algorithm-based optimal posture control of humanoid robots," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–7.

[27] S. C. Walpole, D. Prieto-Merino, P. Edwards, J. Cleland, G. Stevens, and I. Roberts, "The weight of nations: an estimation of adult human biomass," *BMC public health*, vol. 12, no. 1, pp. 1–6, 2012.

[28] R. Drillis, R. Contini, and M. Bluestein, "Body segment parameters," *Artificial limbs*, vol. 8, no. 1, pp. 44–66, 1964.