ADAPTIVE TORQUE CONTROL OF A ROBOTIC MANIPULATOR

by

DREW WALLER

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN

ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2016

Acknowledgements

The work for this project began with my first graduate class in Spring 2015. I thank Dr. Popa for initializing my interest and foundational understand of robotic systems, introducing me to the youBot device, and critical mentorship to begin and finish this thesis. Further, the technical understanding necessary to implement the controls logic would not have been possible without Dr. Lewis's instruction and text material. The Neural Network [*1*] book is a truly excellent adaptive controls resource. Also Dr. Bahare provided critical insight and support whilst I struggled with some of the more difficult controls concepts this effort required. I am very grateful for my thesis committee including Dr. Davis and Dr. Manry for taking interest and time to challenge my thesis defense and contribute questions I attempt to better address here.

Many thanks to the students and faculty of UTA's Next Generation Systems [*2*]. During their continuing mission to study and enable opportunities for robot-human interaction, these great people made space for my school and thesis work which I hope will serve them well. Oguz Yetkin, Rommel Alonzo, Raghavendra Sriram, Joshua Baptist, Sven Cremer, Fahad Mirza, Roopak Karulkar, Yathartha Tuladhar, and Cody Lundberg each provided moments of keen insight, valuable motivation, or pleasant company which certainly improved the quality of this thesis and my time at UTA.

Special thanks to Sven Cremer for pushing me into using repositories for offline code revision, as well as some of the intricacies of working with the youBot itself. Also my dad George for defeating some programming challenges, my mom Barbara for critiquing my writing, and my brother Matthew for being awesome. Mostly, huge gratitude to both my parents, who's endless sponsorship has enabled me to stumble through so many periods of study and unemployment whilst searching for a happy career path.

July 17, 2016

Abstract

ADAPTIVE TORQUE CONTROL OF A

ROBOTIC MANIPULATOR

Drew Waller, MS

The University of Texas at Arlington, 2016

Supervising Professor: Dan Popa

In this thesis, I present the theory and application of enabling intuitive control and physical interaction between a human operator and a mobile manipulator. To do so, the model of the manipulator was estimated for its kinematic and inertial properties. Specifically selected in this project was the Kuka youBot for its ease of use and wide academic availability. With thorough modeling of this device that included the arm as well as the base, and advanced control and guidance techniques, an adaptive controller was used to guide the manipulator towards a goal trajectory with great accuracy. The goal trajectory was built in cartesian reference frame was translated into the n-dimension joint parameters of the device through the kinematic techniques here discussed. These were then demonstrated with thorough virtual simulation using Matlab Simulink, and also applied to an actual device using custom ROS code written in C++. These demonstrations highlighted the difficulty of cartesian guidance with a limited manipulator whilst also showing adaptive methods can work with jacobian based error estimation. While the jacobian math suggests orientation as well as position guidance are possible, there are limiting interactions in under articulated manipulators that can be overcome using motions of the mobile base. However, until these base motions can be well observed, they cannot be used with an adaptive torque control of a mobile manipulator.

Table of Contents

List of Illustrations

List of Tables

Chapter 1 Robots in application

Motivation for an Intuitive Interface

From industry to consumer entertainment, robots are becoming ubiquitous in modern life. This is to be expected as the once cost prohibitive nature of these devices is rapidly yielding to cheap reliability. In response manufacturers, service providers, and academic researchers are creating opportunities with complex tasks or abstract goals that have very mission specific challenges. These challenges are often related to the type of mission to be executed, and they are sometimes based on technical challenges related to guiding a mobile and thoroughly articulated manipulator through the environment to accomplish the mission goals.

To help users with limited robotics experience, the next generation of robots, particularly manipulators, need to include control programs that can accept commands phrased in terms of desired world coordinates and translate them to the required internal robot coordinates. World or Cartesian coordinates are preferred as the most universal coordinate method for expressing a desired object state with three dimensions of position and three dimensions of orientation. These Cartesian commands however, must be relayed to an articulated manipulator that can only observe its joint positions through encoders, and apply appropriate joint torques through motors. Therefore, driving the robot requires translating Cartesian commands into joint vectors through the robot kinematics, while determining torques to achieve those joint vectors and commands is dependent on the dynamics of the device. The robots and control algorithms capable of accommodating both kinematics and dynamics in an adaptive manner, without user intervention and specialized technical knowledge, will enable faster and greater success in future applications.

The research presented here is aimed at formulating adaptive controllers for mobile manipulators, and was enabled by a grant from the National Science Foundation's National Robotics Initiative, "IIS #1208623 – Multimodal skin and garments for home and healthcare robots" [*3*].

Robot Control Challenges

Kinematic Guidance

In a serially linked manipulator, every joint motion creates some relative motion between the base and end effector coordinate frames. In the case of most manipulators, this motion will include quantities of both rotation and translation and rarely will any sole joint provide the desired motion of a given Cartesian task. Therefore, guidance of a manipulator requires the simultaneous coordination of every joint between a reference base and the effector frame [*4*]. The calculation of the end effector position for a known joint state, is known as forward kinematics and is a calculation based on well-known transformations using a robot model, such as the Denavit-Hartenberg notation [*5*]. Solving for the joint state corresponding to a desired effector position requires the inversion of the robot kinematic map, called inverse kinematics. This is a problem that is often fraught by a lack of analytical methods and includes multiple or infinite solutions. Solving the inverse kinematics problem is made more manageable by using the so known Jacobian method [*6*]. In this manner, rather than calculating a directly inverted kinematic solutions, the manipulator is directed iteratively toward the goal. To clarify consider equation (1) below. Here g_{st} is the forward kinematic transformation as a function of the robot joints which are indicated as a n-vector by θ. Here, n is the number of joints. This transformation matrix is translated into the six-dimension vector ***x*** which expresses all the dimensions an object's position and orientation.

$$g_{st}(\theta) \rightarrow x, J(\theta)^{-1}\dot{x}_{desired} + J(\theta)^{-1}(x_{desired} - x) = \dot{\theta}_{command} \qquad (1)$$

Assuming a task or operator expresses some desired trajectory through cartesian space, this six-dimension input would need be translated to the n-dimension θ. For this, the Jacobian J which also a function of the joints is applied relay desired behavior and bound errors through the joint velocity command calculation.

This is a simplistic expression of how the manipulator's joint encoders are observed and translated through forward kinematics into the task space of a desired trajectory. However, this method may not work if the robot does not have sufficient joints or degrees of freedom to accomplish certain tasks. Each joint in a manipulator represents a degree of freedom available to the motion of the robot. This is critical because placing any object in three space requires exactly six degrees of freedom. For this reason, a manipulator requires at least six joints to move its end effector in any arbitrary direction inside its workspace. Additionally, as a manipulator approaches a position where joint motions produce identical transformations of the end effector, i.e. the joints are redundant, an efficient manipulator will lose the ability to move in certain directions which depend on the pose of the effector. These are referred to as singularities in the inverse jacobian transformation.

Therefore, constructing the jacobian simplifies the kinematic challenge in some ways while introducing its own difficulties. A mobile manipulator may further complicate this challenge by attempting to augment a deficient manipulator of five or less joints with a base with one or more freedoms of motion. However, due the extrinsic nature of a mobile base lacking the clarity of transform provided by intrinsic joint state encoders, controllers and operators will have only a limited perception of base motion if any.

This nature of mobile devices likely restricts a roboticist from simply incorporating these freedoms of motion as joints due to the inability to observe their absolute state.

However, these base freedoms might enable jacobian solutions for end effector velocities that would be otherwise impossible for a lack of observable and actuated joints.

A method considered in this thesis incorporates the mobile base to augment the freedoms of the manipulator. Asserting mobile freedoms as having zero state while applying their velocities allows an operator to use a robust jacobian controller while also creating dependency on said operator to continuously transform desired goal coordinates into the current frame of the device. This is not unfamiliar to any remote control hobbyist since such a transformation becomes routine and subconscious as one practices control of a selected device.

To further augment the control challenge of interpreting input commands into a desired joint state, a control method is proposed that would relieve the operator of selecting position and acceleration values. In this method, the absence of input declares the current position as the desired state, otherwise the input heading is linearly filtered to determine desired position, velocity and acceleration states for guidance commands.

Torque Control

Many robots, particularly those capable of rapid movement, cannot act directly on joint state commands. While encoders allow controllers and operators to observe the joint state, joint motors through electrical current regulation are only capable of effecting torque inputs to the dynamic system. Determining the torque to guide a manipulator towards a goal with precision requires studying the general dynamics of the system. Understanding the dynamics and how a manipulator changes state with time is a mechanical challenge that has can also be solved with classical mathematics. With an exact understanding of these dynamics, a controller could use a linear filter to translate desired states into appropriate input derivatives which is typically acceleration for most devices. Knowing the exact dynamics however, is near impossible even with an exhaustively thorough measurement of the system. To avoid this exhaustive effort, methods exist to guide an unknown system whilst adapting to it [*7*]. Thus, a controller initialized with crude performance, through time and adaptive learning, will eventually become exceptionally accurate for a given device [*8*].

Description of Related Work

The Kuka youBot is a well-known academic robotic platform, which has been widely distributed and published in recent years [*9*] [*10*]. It is a relatively easy to use device, and many users have focused on very specialized or advanced applications [*11*]. The focus of our work is to enable more intuitive human machine interaction, and both the robot kinematics and dynamics have to be taken into account for adaptive controllers for the youBot.

Beginning with kinematics, many texts thoroughly describe the methods of forward kinematic analysis [*12*] [*13*]. Additionally, many works published regarding the youBot detail possible solutions to the challenge of inverse kinematics [*14*] [*15*] [*16*]. Very few, if any provide a satisfying solution to the vector space challenge of the inverse Jacobian function. This is probably due to the manipulator’s joint deficiency which will be more thoroughly discussed later. Therefore, this study will pursue a Jacobian solution to provide direct trajectory solutions for cartesian task descriptions [*17*] [*6*] [*18*] [*19*].

Further, while many texts discuss torque control methods [*20*] [*21*], applying these methods from estimation to application and validation is difficult. Although others [*11*], discuss implementing robust torque control for the youBot. This study will attempt to better elucidate the process of dynamic study and provide an adaptive method of torque control for the Kuka youBot with methods that could be applied to manipulators in general.

Contributions to Ongoing Research by this Thesis

Because of the growing interest in robot that directly interact with humans, new initiatives have begun to direct studies in robotics towards appropriate interfacing [*22*] [*23*] [*24*]. Here, this means better understanding methods by which a human operator can provide task guidance through non-technical means such as physical interaction [*25*] [*26*]. One direction of research includes creating new sensing devices to recognize and quantify physical contact between human and robots such as robot skins [*27*]. Our study however, builds on the concept that physical interaction can be recognized and quantified in terms of imparted disturbance forces that exceed known expectations [*28*] [*29*]. If a robot has high confidence in its controllers as verified by its proprioceptive encoders, sudden or even slight deviations from an expected trajectory can inform a controller of informationally significant disturbances. This proposed method is based on physical experiments in robotic interaction and is better discussed later on.

To this purpose, the Kuka youBot was quantified in terms of both its kinematics and dynamics. The kinematic nature and challenges as discussed are confronted and resolved according to published works. This understanding also provides the fundamentals to construct the dynamic model of the device known also as the joint state dependent gravity, inertial, and coriolis function matrices. Further, this thesis applies the established methods of neuro adaptive techniques to enable a controller to refine these estimation dependent models whilst retaining stable control of a physical device along a desired trajectory. All of these components are successfully combined in a virtual demonstration of a six jointed manipulator based on the youBot device. Both the jacobian guidance and neuro adaptive control are physically realized on an actual device with selected code and demonstration video appended.

The work detailed in this thesis includes several concepts of robotic manipulator characterization and control as applied to the Kuka youBot. These include:

- A formulation of the forward kinematics and Jacobian into matrix equations to utilize both the manipulator and mobile base motions. These matrices are critical to the guidance and control of this device for applications that generate task space goals during the operation of the platform.
- A method of adaptive torque control [*1*] with a preprocessed single layer neural network which was validated in numerical simulation and on the device itself.

Chapter 2 Literature and Research

Because robotic systems represent fantastic opportunities with a wide field of solvable technical challenges, there exists a great wealth of technical papers on the subject. While many papers were reviewed, only a few texts can be cited for their technical and inspirational contribution to this paper. Most critical is the mathematical introduction to robotic manipulators by Murray, Li, and Sastry [*12*]. This text provides the thorough, fundamental details necessary to mathematically describe both the kinematics and dynamic behavior of robot manipulators. The aspects relevant to this project are discussed in this chapter beginning pages fifteen and twenty respectively.

Of great inspiration and some technical insight was a thesis from ETH Zurich on the subject of Torque Control of a Kuka youBot Arm by Ben Keiser [*11*]. Discussed here are many of the same motivations and challenges. However, his solution seems light in technical detail and more relevant to low speed minimal dynamic control to demonstrate the possibilities of robotic perception. His work is cited here for his useful insights on experiments and possibilities for human interaction.

Enabling the methods of robust and high dynamic torque control required the insights of Lewis's Neural Network control of manipulators [*1*]. This text adopts the technical understanding of Lyapunov stability, robot dynamics, and neural network programming into a unified engineering applicable approach. These invaluable insights define the selected torque control methods applied by this paper. The specific insights of which are discussed in the final section of this literature review.

Kinematic Fundamentals

Beginning the design of any robotic controller is the study and understanding of the device's movement. These rules of how the motion of each joint affects the following joints and the end effector are known as the kinematics. To characterize a given robot, nearly every joint can be considered either a revolute, a prismatic or some combination of those two. Similar to a hand, the end-effector of a manipulator has a pose that is determined by its position and orientation. This pose can be determined directly from the state of the joints by the application of the forward kinematics. This kinematic function can be readily solved with mathematical descriptions of these joints described in [*12*] [*13*] as a series product of exponentials known as the kinematic chain. Specifically, each joint is described with the ξ vector which is referred to as a twist as shown in Figure 1-1.

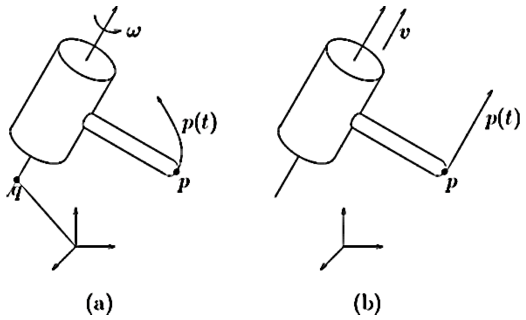


Figure 1-1 Graphical depiction of ξ vectors

$$\xi = \begin{bmatrix} -\omega \times q \\ \omega \end{bmatrix} \ (rotation) \ \ or \ \ \xi = \begin{bmatrix} v \\ 0 \end{bmatrix} \ (translation) \qquad (2)$$

A twist contains details for an axis of rotation with a magnitude or a displacement either of which is constructed as shown by equation (2) which build the transformations made by each joint of a manipulator. The product of a given displacement theta along a twist is known as a screw motion. The transformation of a coordinate frame induced by the scalar displacement of a twist is expressed mathematically as shown in equation (3).

$$e^{\hat{\xi}_i\theta_i} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{3}$$

In equation (3), ***p*** is a three-dimension vector representing the simple x, y, and z displacement of the joint transformed coordinate frame. ***R*** is a three by three matrix which expresses the rotation of this transformation. This format, allows series of transformations to be multiplied together, creating the kinematic rules of a manipulator.

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \qquad \hat{a} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \qquad a \times b = \hat{a}b \tag{4}$$

To clarify, ***a*** in equation (4) is a generic three-dimension vector for representing the unit length axis component of a twist in the three space (x, y, z) also known as cartesian navigation. The up carrot operation known as the skew of ***a*** produces the skew symmetric matrix which when matrix multiplied by a similar vector b, produces the same result as ***a*** cross ***b***. The wedge operation of the ξ vector which is a six-dimension vector relies on this skew to create the special orthogonal matrix of four by four dimensionality which forms the exponent of a joint transformation described by the ξ matrix.

$$\xi = \begin{bmatrix} v \\ \omega \end{bmatrix}, \qquad \hat{\xi} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix}, \qquad \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{5}$$

$$g(\theta) = \begin{bmatrix} e^{\widehat{\omega}\theta} & \left(I - e^{\widehat{\omega}\theta}\right)(\omega \times v) + \omega\omega^T v\theta \\ 0 & 1 \end{bmatrix} \tag{6}$$

Here in equations (5) and (6), a transformation imposed by a single joint is produced. The product of a series of joint screw motion transformations produces the general kinematic solution for any serially linked manipulator. If it helps clarify, the vector ξ and transformation matrix ***g*** do not necessarily describe the appearance of physical joints they represent. It would be more accurate to say they describe the parameter dependent motions or transformations of coordinate frames within which points and objects might be constructed. A series of these could describe a serially linked robotic manipulator.

$$g_{st}(\theta) = e^{\hat{\xi}_1\theta_1} e^{\hat{\xi}_2\theta_2} \cdots e^{\hat{\xi}_n\theta_n} g_{st}(0) = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{7}$$

This resulting matrix function ***g*** is also known mathematically as a special Euclidean which has four by four dimensionality. Using this to navigate the six vector task space requires transforming this into a six vector of position and orientation dimensions. The position of a transformation matrix is readily expressed in the first three rows and last column '[1:3,4]' or ***p***. However, the rotation information of this kinematic function is described by the field of the first three rows and columns '[1:3,1:3]', ***R***, which is otherwise known as a special orthogonal matrix. Many methods of transforming this rotation matrix into rotational dimensions exist. For this project one of two methods were preferred depending on the situation. The first of which comes directly from MLS [*12*]. Known as exponential mapping, both of these methods consider orientation dimensions as a scalar displacement applied to a unit vector axis.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \qquad \breve{R} = \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \Big/ 2 \tag{8}$$

Recovering the orientation information from the rotation set ***R*** of the transformation matrix in (7) begins with determining the inverse skew of this set. This inverse operation as shown in (8) is referred to as the hex.

$$\theta = cos^{-1}\left(\frac{trace(R) - 1}{2}\right), \qquad \omega = \breve{R}/sin(\theta), \qquad x = \begin{bmatrix} p \\ \omega\theta \end{bmatrix} \tag{9}$$

The scalar magnitude of the rotation is the inverse cosine of half the trace reduced by one as shown in equation (9). The hex of the rotation set divided by the sine of this magnitude yields the rotation axis ω. The position vector appended with axis-magnitude produce produces the six-dimension cartesian state vector ***x***.

This method is simple and can be rapidly evaluated, however it has a severe issue with rotation matrices corresponding to poses that approach zero rotation. This is due to the denominator approaching zero creating an unreliable calculation. A more robust method is derived from the rotation definition of an arbitrary screw.

$$x = \begin{bmatrix} p \\ \omega\theta \end{bmatrix}, \qquad \widehat{\omega}\theta = \log_m R\,, \qquad \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = g_{st}(\theta) \tag{10}$$

As before, the rotation coordinates are conceived as a scalar rotation around a unit axis vector. Here. however, the axis angle product is yielded directly from the earlier described inverse skew (or vex) of the matrix logarithm of the rotation matrix shown in (10). This matrix logarithm is a specialized calculation that may be processor intensive and perhaps should be applied only when singularities in the orientation are expected.

While this produces a function ***g*** of theta, that can be evaluated for the Cartesian pose of the end-effector or any manipulator dependent location, this function is not easily inverted if it can be at all. That is to say one cannot easily if at all apply a desired pose ***x*** through some function inverse ***g*** to produce a desired joint state. Indeed, some desired poses are not reachable and are often referred to as being outside the manipulator's work space. Further, if an inverse kinematic solver is not careful, some solutions may 'toggle' the states of the kinematic chain which if signaled to a controller would be an immediate disaster. For these reasons an inverse kinematic solver was not attempted for this project, and instead the desired end effector pose is translated to desired joint states through a method known as the Jacobian.

If the forward kinematics produce a function ***g*** of the joint state θ yields the Cartesian position ***x***, then the Jacobian would simply be the derivative of that function ***g*** with respect to θ. Appling the time derivative to the kinematic function via the chain rule reveals this analytic Jacobian while a further derivative yields acceleration (11).

$$\dot{x} = J(\theta)\dot{\theta}, \qquad \ddot{x} = J(\theta)\ddot{\theta} + \frac{\partial}{\partial\theta}J(\theta)\frac{d\theta}{dt} \tag{11}$$

By rephrasing these equations in terms of the Cartesian coordinates, the inverse Jacobian can be applied to yield the desired joint state derivatives [*30*] seen in (12).

$$\dot{\theta} = J(\theta)^{-1}\dot{x}, \qquad \ddot{\theta} = J(\theta)^{-1}\ddot{x} - J(\theta)^{-1}\dot{J}(\theta)\dot{\theta} \tag{12}$$

It should be noted that an inverse Jacobian solution is incomplete for any manipulator without at least six degrees of freedom. Additionally, some joints depending on device and joint state may become redundant which effectively reduces the degree of freedoms available to a given manipulator [*31*]. This would also make some inverse solutions unattainable in such cases.

Because the desired static joint space is never exactly solved, a method to closely approximate the joint state error must be produced. When there exists a desired cartesian position, the joint state error can be determined by distributing the differential time across the jacobian equation. The forward kinematic solution provides the actual position which is removed from the desired end-effector pose as shown in (13).

$$e_{\theta} = \theta_{desired} - \theta \cong J(\theta)^{-1}(x_{desired} - x) \tag{13}$$

In the event, where the desired static position is not known, as is the case of a mobile device, the desired joint state could be constructed by integrating the desired joint velocity. The error is then found simply by removing the observed joint state.

Building the Jacobian as detailed in [*12*] begins with using the kinematic details collected earlier. The spatial Jacobian becomes a concatenation of adjoint modified twist parameters as shown by the following equations (14)(15)(16).

$$J_{st}^{s}(\theta) = [\xi_1 \quad \xi_2' \quad \cdots \quad \xi_n'] \tag{14}$$

$$\xi_i'(\theta) = Ad_{\left(e^{\hat{\xi}_1\theta_1}\cdots e^{\hat{\xi}_{i-1}\theta_{i-1}}\right)}\xi_i \tag{15}$$

$$Ad_g = \begin{bmatrix} R & \hat{p}R \\ 0 & R \end{bmatrix}, \qquad g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{16}$$

However, equation (14) only yields the spatial Jacobian. The geometric Jacobian relating joint rates to the end-effector rate requires the final adjoint of the forward kinematic displacement as shown in (17)(18). The chain rule is then used to find the acceleration by taking the time derivative of the jacobian.

$$p = g_{st}(\theta)_{1:3,4} \tag{17}$$

$$J(\theta) = Ad_{\begin{bmatrix} I & -p \\ 0 & 1 \end{bmatrix}} J_{st}^{s}(\theta) \tag{18}$$

Thus jacobian control for motion vectors reachable within the device's workspace is processed as follows. Assume desired cartesian states of position, velocity and acceleration, as well as actual joint positions and velocities are known.

$$known\colon x_{desired} = \begin{bmatrix} p \\ \omega\theta \end{bmatrix}, \dot{x}_{desired}, \ddot{x}_{desired}, \theta, \dot{\theta} \tag{19}$$

Using the mathematic methods described earlier, the inverse Jacobian matrix J^i and the cartesian position and orientation of the end effector ***x*** is produced.

$$\theta \to J(\theta)^{-1} \to J^i, \qquad \theta \to g_{st}(\theta) \to \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \to \begin{bmatrix} p \\ \omega\theta \end{bmatrix} \to x \tag{20}$$

Now, joint position error e_θ, desired velocity $\dot{\theta}_d$ and acceleration $\ddot{\theta}_d$ are produced.

$$J^i, x_d, x \to J^i(x_d - x) \to e_\theta, \qquad J^i\dot{x}_d \to \dot{\theta}_d \tag{21}$$

$$J^i, \dot{\theta}, \ddot{x}_d \to J^i\ddot{x}_d - J^i\dot{J}(\theta)\dot{\theta} \to \ddot{\theta}_d \tag{22}$$

This should be sufficient to navigate the kinematics of a plausible and reachable task vector. For a hypothetical device capable of a direct acceleration command, a linear control algorithm might function with these details and gain coefficients for proportional ***P*** and derivative ***D*** error mitigation.

$$\ddot{\theta}_d + D\left(\dot{\theta}_d - \theta\right) + P(e_\theta) \to \ddot{\theta}_{command} \tag{23}$$

Nonlinear Dynamics

Predicting the evolution of the manipulator states is made possible by constructing a thorough dynamic model of the forces acting on the device. These forces are stated generally by [*12*] as matrix equations. Inertia M, Coriolis C, gravity G, friction F, disturbances torque τ_d, and input torque τ are shown in equation (24).

$$M(\theta)\ddot{\theta} + C\left(\theta, \dot{\theta}\right)\dot{\theta} + G(\theta) + F\left(\dot{\theta}\right) + \tau_d = \tau \tag{24}$$

While this equation might appear simple enough for planar devices or those with only a few degrees of freedom, a general solution for generic serially linked manipulators can be a thoroughly intense study that deserves its own scrutiny to understand well. Regardless, the process begins with building the Lagrangian and its derivatives from which the derivation of the resulting terms can be produced [*20*]. The Lagrangian $\mathcal{L}$ which is the difference of total kinetic and potential energy of a system is stated (25).

$$\mathcal{L}\left(\theta, \dot{\theta}\right) = K\left(\theta, \dot{\theta}\right) - P(\theta) \tag{25}$$

From this the equations of motion for all mechanical systems are related by the series of differential equations for ***n*** degrees of freedom with state variables θ, and external forces or torques Υ_i.

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i} - \frac{\partial \mathcal{L}}{\partial \theta_i} = \Upsilon_i \quad i = 1, \cdots, n \tag{26}$$

Solving the potential energy of (25) is simply the sum of products each link's mass m_i, gravity g, and height h_i as function of θ. Kinect energy is the product of each link's local velocity $V_{sl_i}^b$, inertia $\mathcal{M}_i$, and velocity transpose. Substituting the yields (27).

$$\mathcal{L}\left(\theta, \dot{\theta}\right) = \sum_{i=1}^{n} \frac{1}{2}\left(V_{sl_i}^b\right)^T \mathcal{M}_i V_{sl_i}^b - m_i g h_i(\theta) \tag{27}$$

The link velocities are functions of the joint state and its derivative. Therefore, this equation cannot be reduced until the body Jacobian $J_{sl_i}^b$ is applied for each link of the manipulator chain as described in equations (28)-(31) by [*12*].

$$V_{sl_i}^b = J_{sl_i}^b(\theta)\dot{\theta} \tag{28}$$

$$J_{sl_i}^b(\theta) = [\xi_2^\dagger \quad \cdots \quad \xi_i^\dagger \quad 0 \quad \cdots \quad 0] \tag{29}$$

$$\xi_i^\dagger = Ad^{-1}_{\left(e^{\hat{\xi}_j\theta_j} \cdots e^{\hat{\xi}_i\theta_i} g_{sl_i}(0)\right)} \xi_j \; j \leq i \tag{30}$$

$$K(\theta, \dot{\theta}) = \dot{\theta}^T \sum_{i=1}^{n} J_i^T(\theta)\mathcal{M}_i J_i(\theta)\,\dot{\theta} = \dot{\theta}^T M(\theta)\dot{\theta} \tag{31}$$

Through substitution equation (27) now becomes more tractable.

$$\mathcal{L}(\theta, \dot{\theta}) = \frac{1}{2}\sum_{i,j=1}^{n} M_{ij}(\theta)\dot{\theta}_i\dot{\theta}_j - m_i g h_i(\theta) \tag{32}$$

By applying the Lagrangian derivatives of (26) to form the equations of motion, the inertial derivative terms can be expanded to yield equation (33).

$$\sum_{j=1}^{n} M_{ij}(\theta)\ddot{\theta}_j + \sum_{j,k=1}^{n}\left(\frac{\partial M_{ij}}{\partial \theta_k}\dot{\theta}_j\dot{\theta}_k - \frac{1}{2}\frac{\partial M_{kj}}{\partial \theta_i}\dot{\theta}_k\dot{\theta}_j\right) + \frac{\partial}{\partial \theta_i} m_i g h_i(\theta) = \Upsilon_i \tag{33}$$

From here, the inertia matrix can now be readily extracted and defined through the above jacobian functions with equations (34)(35)(36).

$$A_{ij} = \begin{cases} Ad^{-1}_{\left(e^{\hat{\xi}_{j+1}\theta_{j+1}} \cdots e^{\hat{\xi}_i\theta_i}\right)} & i > j \\ I & i = j \\ 0 & i < j \end{cases} \tag{34}$$

$$M_{ij}(\theta) = \sum_{l=\max(i,j)}^{n} \xi_i^T A_{li}^T \mathcal{M}_l' A_{lj}\xi_j \tag{35}$$

$$\mathcal{M}_l' = Ad^T_{g^{-1}_{sl_i(0)}} \mathcal{M}_i Ad_{g^{-1}_{sl_i(0)}} \tag{36}$$

Further the Coriolis matrix as defined by the indexed inertia derivatives can be determined through the following equations. Note the brackets in (38) here denote the mathematical lie bracket function which requires wedge and inverse wedge operations as described earlier in equation (5).

$$C_{ij}(\theta,\dot{\theta}) = \frac{1}{2}\sum_{k=1}^{n}\left(\frac{\partial M_{ij}}{\partial\theta_k} + \frac{\partial M_{ik}}{\partial\theta_j} - \frac{\partial M_{kj}}{\partial\theta_i}\right)\dot{\theta}_k \tag{37}$$

$$\frac{\partial M_{ij}}{\partial\theta_k} = \sum_{l=max(i,j)}^{n}\begin{pmatrix}[A_{ki}\xi_i,\xi_k]^T A_{lk}^T \mathcal{M}_l' A_{lj}\xi_j + \\ \xi_i^T A_{li}^T \mathcal{M}_l' A_{lk}\left[A_{kj}\xi_j,\xi_k\right]\end{pmatrix} \tag{38}$$

$$[\xi_1,\xi_2] = \left(\hat{\xi}_1\hat{\xi}_2 - \hat{\xi}_2\hat{\xi}_1\right)^{\vee} \tag{39}$$

As stated, the torque due to gravity is the joint dependent derivative of the potential energy function. Finally, the friction force is approximated as the sum of a gain on joint rate d_i and a gain of signed rate f_i to capture both damping and friction forces.

$$G_i(\theta) = \frac{\partial}{\partial\theta_i} m_i g h_i(\theta) \tag{40}$$

$$F_i(\dot{\theta}) = d_i\dot{\theta}_i + f_i sgn(\dot{\theta}_i) \tag{41}$$

With these equations, the dynamic model of the serially linked manipulator could be applied to evaluate the joint acceleration due to a Torque vector at the current state with the following method. Assume, an input torque vector T and the joint states of position θ and velocity $\dot{\theta}$ are known. An integrable signal $\ddot{\theta}$ is produced.

$$T,\theta,\dot{\theta} \to \left[T - F(\dot{\theta}) - G(\theta) - C_{ij}(\theta,\dot{\theta})\dot{\theta}\right]/M(\theta) = \ddot{\theta} \tag{42}$$

Adaptive Control

These thoroughly modeled dynamics make for an adequate dynamic simulator. However, they are even more valuable in the construction of the controller for the manipulator. According to [*1*] [*32*] [*33*], there are few ways of building an adaptive controller, some of which require little knowledge of the robot's dynamics. The proof begins with the general dynamic equation of manipulator motion, the assumption of a desired trajectory, and a new state variable ***r*** combines the static error from that trajectory and its first time dependent derivative. This sum uses a coefficient matrix Λ to manage units and effectively replicates a linear proportional-derivative control gain.

$$e_\theta = \theta_d - \theta, \qquad r = \Lambda e + \dot{e} \tag{43}$$

$$N(\theta, \dot{\theta}) = G(\theta) + F(\dot{\theta}) \tag{44}$$

$$M(\theta)(\ddot{\theta}_d + \Lambda\dot{e} - \dot{r}) + C(\theta, \dot{\theta})(\dot{\theta}_d + \Lambda e - r) + N(\theta, \dot{\theta}) + \tau_d = \tau \tag{45}$$

Beginning in (46), ***x*** represents the concatenation of all desired states and actual states, and the new state variable ***r*** becomes the core expression of the error state.

$$x = [\theta^T \quad \dot{\theta}^T \quad \theta_d^T \quad \dot{\theta}_d^T \quad \ddot{\theta}_d^T] \tag{46}$$

$$M\dot{r} = -Cr + f(x) + \varepsilon(x) + \tau_d - \tau \tag{47}$$

$$f(x) + \varepsilon(x) \equiv M(\ddot{\theta}_d + \Lambda\dot{e}) + C(\dot{\theta}_d + \Lambda e) + N(\theta, \dot{\theta}) \tag{48}$$

Beginning with (47) an unknown nonlinear function, f, plus some intractable approximation error, ε, is determined by the actual system dynamics and dependent on some or all of the variables included within the newly declared ***x*** vector.

$$\tau = \hat{f}(x) + K_v r \tag{49}$$

A control method (49) that would minimize the error vector ***r*** should begin by estimating the nonlinear function approximation $\hat{f}$ as well as including some proportional error feedback gain K_v. Through substitution, the dynamic error equation becomes (50).

$$\tilde{f}(x) = f(x) - \hat{f}(x)$$
$$M\dot{r} = -Cr - K_v r + \tilde{f}(x) + \tau_d(t) + \varepsilon(x) \tag{50}$$

Note the new function label $\tilde{f}$ merely represents the error between some ideal approximation f and the current estimate $\hat{f}$. Because the dynamics are believed to be well modelled, the preferred method of approximation would be to create a set of basis functions $\phi(x)$. These basis functions, many of which would be nonlinear, would be proportionally adjusted through some integration method for each respective joint until the weights $\widehat{W}$ and basis product perfectly replicate the nonlinear dynamics of f.

$$\hat{f}(x) = \widehat{W}^T\phi(x), \qquad \widetilde{W} = W - \widehat{W} \tag{51}$$

Similar to the function error $\tilde{f}$ earlier, W is the weight matrix of an ideal function approximation f, while $\widehat{W}$ indicates the current estimate and $\widetilde{W}$ is there error of this estimate. The illustration *Figure 2-1* generated by my colleague [*34*] [*35*] [*36*] is structurally identical in that a task is first generated in cartesian space ***x***. The nonlinear function approximation negates the dynamics while the linear control gains of K_v and Λ bound the error to the desired trajectory.

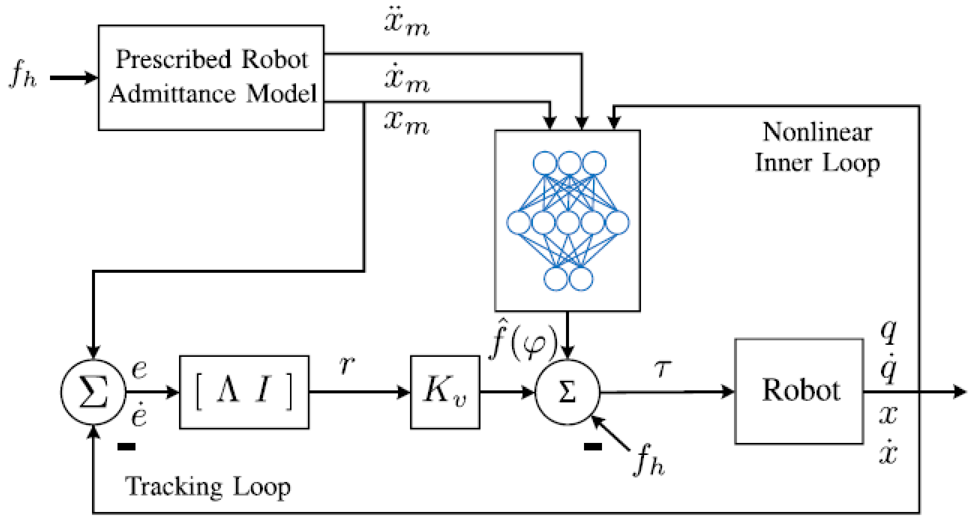


Figure 2-1 This Layout is structurally similar to that proposed here.

For the system to improve and adapt, it must continuously integrate in a way that reduces the trajectory error ***r*** and the function approximation error $\tilde{f}$. Ideally, this will occur quickly and smoothly. Of course, the most universally recognized method for nonlinear systems such as this, is the Lyapunov method [*5*] [*37*] [*38*]. This begins with the positive definite scalar sum of the relevant state variables which are the error ***r*** and basis weights error $\widetilde{W}$.

$$L = \frac{1}{2} r^T M r + \frac{1}{2} tr\left\{\widetilde{W}^T F^{-1} \widetilde{W}\right\} \tag{52}$$

$$\frac{dL}{dt} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + tr\left\{\widetilde{W}^T F^{-1} \dot{\widetilde{W}}\right\} \tag{53}$$

This is known as the Lyapunov candidate. Determined here as the norm of a weight matrix deviation $\widetilde{W}$ and error ***r*** which are positive definite and monotonically increases with either input. If selected parameters are designed such that the time rate of change is always negative, then the sum will approach zero with time. To observe and design these selectable parameters, the dynamics established earlier are applied to the Lyapunov derivative.

$$\frac{dL}{dt} = r^T \left(-Cr - K_v r + \widetilde{f}(x) + \tau_d(t) + \varepsilon(x)\right) + \frac{1}{2} r^T \dot{M} r + tr\left\{\widetilde{W}^T F^{-1} \dot{\widetilde{W}}\right\} \tag{54}$$

$$\frac{dL}{dt} = -r^T K_v r + r^T \left(\widetilde{W}^T \phi(x) + \tau_d + \varepsilon\right) + tr\left\{\widetilde{W}^T F^{-1} \dot{\widetilde{W}}\right\} \tag{55}$$

By the skew symmetric property of twice the coriolis matrix removed from the time derivative of inertia $\left(\dot{M} - 2C\right)$, this term is known to be zero [*12*] which yields (55).

$$\frac{dL}{dt} = -r^T K_v r + tr\left\{\widetilde{W}^T \left(\phi(x) r^T + F^{-1} \dot{\widetilde{W}}\right)\right\} + r^T(\tau_d + \varepsilon) \tag{56}$$

The basis term $\phi(x)$ can now be evaluated within trace to further simplify the candidate function (56). From here, a weights derivative function can be proposed (57).

$$\dot{\widetilde{W}} = -F\phi(x) r^T, \qquad \tau_d(t) \leq d_B \tag{57}$$

$$\dot{L} = -r^T(K_v r - \tau_d - \varepsilon) \leq -K_{v_{min}}\|r\|^2 + (d_B + \varepsilon)\|r\| \tag{58}$$

Substituting this proposed derivative function, the final statement (58) indicates the behavior of the Lyapunov function [*39*] [*40*]. However, the Lyapunov function is only negative semidefinite where the error variable r is non-zero, which may satisfy some applications. Regardless, Barbalat's Lemma shows the system can not only bound the error of the joint states, but also attenuate error in the nonlinear function approximation.

$$\ddot{L} = -\dot{r}^T(2K_v r - \tau_d) \tag{59}$$

$$\ddot{L} = (2K_v r - \tau_d)^T M^{-1}\big(Cr + K_v r - \widetilde{W}^T\phi - \tau_d - \varepsilon\big) \tag{60}$$

Given the bounded assumptions of τ_d, r, ϕ, as well as the non-zero value of any device's inertia, the second derivative of ***L*** is bounded, which implies that $\dot{L}$ is uniformly continuous and ***L*** itself must approach zero eventually. This requires not only bounded error of the joint states but also bounded error of the weight nonlinear function approximations. This only leaves the basis vectors to be designed which has further opportunity to expedite the training process with structured networks as in *Figure 2-2*.

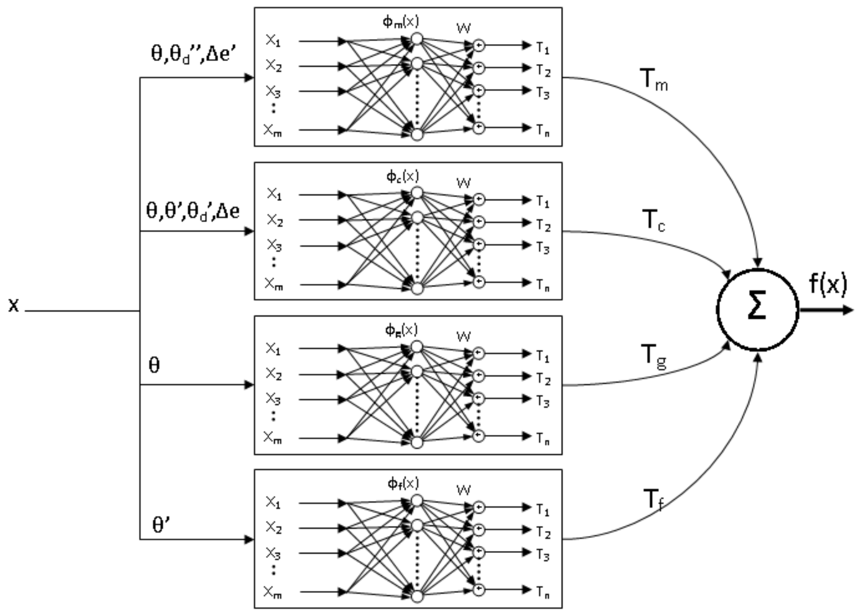


Figure 2-2 Functionally organized subnets

$$M(\theta)\left(\ddot{\theta}_d + \Lambda\dot{e}\right) = W_M^T\phi_M(x) + \varepsilon_M \tag{61}$$

$$C\left(\theta,\dot{\theta}\right)\left(\dot{\theta}_d + \Lambda e\right) = W_C^T\phi_C(x) + \varepsilon_C \tag{62}$$

$$G(\theta) = W_G^T\phi_G(x) + \varepsilon_G \tag{63}$$

$$F\left(\dot{\theta}\right) = W_F^T\phi_F(x) + \varepsilon_F \tag{64}$$

Rather than a single network spanning all aspects of the nonlinear approximation function f, four distinct networks are processed independently. This structured approximation function builds parallel weight matrices and basis sets for each of the inertia, coriolis, gravity, and friction influences. Each subnet of *Figure 2-2* shows two nodes where the ***x*** vector is preprocessed into the basis function sets which is then matrix multiplied with the approximation matrix ***W***. The n-vector outputs of each is summed together which ideally would mitigate the nonlinear forces acting on the device.

Each basis set is constructed through a process that begins with summing together all the components of a selected matrix function from the dynamic model. The sum is then processed into separate input functions and scalar weights. These functions are the basis set from which guides the initial weight matrix of the nonlinear network.

So to clarify, each subnet must produce a n dimension torque vector. Here, n refers to the number of joints in the manipulator. For each functionally organized net, this vector could be referred to as $T_i(x)\ where\ i = 1, \dots n$. To derive the ϕ_i function vectors, the dynamics models constructed earlier are directly applied.

$$Recall\colon\ x = [\theta^T \quad \dot{\theta}^T \quad \theta_d^T \quad \dot{\theta}_d^T \quad \ddot{\theta}_d^T]$$

$$M(\theta)\left(\ddot{\theta}_d + \Lambda\dot{e}\right) \rightarrow M_i(x)\ inertial\ torque\ vector$$

$$C\left(\theta, \dot{\theta}\right)\left(\dot{\theta}_d + \Lambda e\right) \rightarrow C_i(x)\ coriolis\ torque\ vector$$

The sum of these vectors including gravity and friction will be a scalar function dependent on numerous linear and nonlinear functions of ***x*** components. The ϕ_i vectors are composed by identifying a complete list of these input functions while removing the scalar gains S that will later be replaced by the adapting weight matrix ***W*** as follows.

$$\sum_{i=1}^{n} M_i(x) \rightarrow S \times \phi_M(x)\ \ a\ scalar\ sum\ of\ independent\ functions$$

$$list\ independent\ functions \rightarrow \phi_M(x)\ a\ vector\ function$$

Repeating this process will yield an appropriate basis set for each subnet. This basis or ϕ_i function vector should be capable of perfectly replicating the properties of the dynamic model they represent, i.e. $\varepsilon_s = 0$; for at least a virtual simulation.

Chapter 3 Platform Description: The Kuka youBot

Critical to the demonstration and research of the topics in this thesis is the platform upon which these methods will be applied. Manufactured by Kuka [*41*], a German robotics company, the youBot is a simple example of a customizable mobile manipulator. Its mission seems to be purely academic and is an ideal platform to demonstrate fundamental robotics control methods. The youBot includes an omnidirectional base with all three freedoms of planar motion and a five jointed manipulator with a two state gripper. Detailed in this section will be the mathematical description of the platform, control design, and overall system signal path.

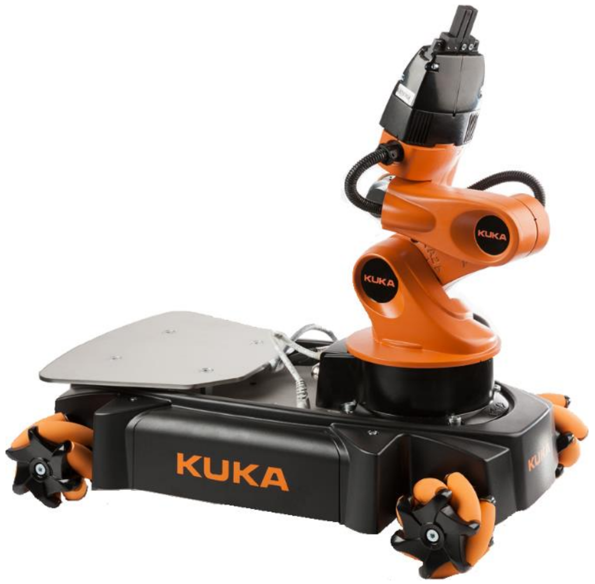


Figure 3-1 YouBot Device [42]

The kinematic description begins with the ξ vectors or collectively the ξ matrix.

Table 3-1: The estimated ξ vectors of the complete youBot with extended manipulator

1	0	0	0	-267	-422	-557	0
0	1	0	-160	0	0	0	-193
0	0	0	0	193	193	193	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0
0	0	1	1	0	0	0	1

Note the first three columns represent of the movement capabilities of the mobile base the manipulator sits on, which are two translation axes and one rotation axis. However, the youBot lacks any built in odometry or global reference system with which to observe the state of these joints. Therefore, while the motions the base provides are critical to a full range kinematic freedom, for practical control purposes, the base reference frame must travel with the base. This effectively means joints one, two, and three will be hard coded with zero position at all times regardless of velocity.

The remaining columns describe the joints of the manipulator arm. All of which are rotation axes, with positions determined by assuming the zero joint state has the links of the manipulator extended vertically along the z axis. As described earlier, this provides all the detail necessary for the forward kinematics and jacobian. Since these are extensive trigonometric matrix functions, they will be listed in the appendix.

Continuing with the description of the device, the weights, inertia, and dynamic properties must now be estimated. Many of these values will be imperfect since some are outright guesses. Fortunately, the nature of adaptive control enables these to be constantly improved and replaced as the manipulator moves [*8*]. Applying this freedom of approximation, a construction of the robot's dynamics is made according to:

- each link's center of mass is near its volumetric center
- the first three links are not considered because they are not torque controlled
- the inertia of each manipulator link is proportional to its prismatic dimensions
- manipulator zero position aligns all inertial frames parallel to the base

Listed in Table 3-2 are the suggested coordinates for the mass centers of the torque controlled links. Each vector set top to bottom indicates the millimeter distance fore, left, and above the base center respectively.

Table 3-2 The suggested manipulator link mass centers in mobile platform frame

$$\begin{bmatrix}160\\10\\300\end{bmatrix}, \quad \begin{bmatrix}193\\-30\\395\end{bmatrix}, \quad \begin{bmatrix}193\\20\\540\end{bmatrix}, \quad \begin{bmatrix}193\\-12\\645\end{bmatrix}, \quad \begin{bmatrix}193\\0\\730\end{bmatrix}$$

The following table lists the suggested inertias in kg·mm^2 for the same joints as table two with longitudinal, lateral, and vertical axes going top to bottom respectively.

Table 3-3 Inertial value estimations for manipulator links

6200	4400	2600	1200	340
6400	4400	2500	1100	280
3900	820	540	870	160

With these details, the inertia gravity and coriolis matrices are constructed as described earlier. These are extensive matrix functions the programmatic code for which should be found in attached materials.

From the dynamics described by these matrices, the basis equations would need to be extracted. Some of these would be slightly complicated as they require the product of the error vector and dynamic matrix as described by the unknown nonlinear function earlier. Fortunately, appropriate attention to detail and symbolic math tools makes this a simple task. For the initial six degree of freedom simulation, this produced three gravity basis functions, ninety-eight inertia basis functions, four hundred and thirty-five coriolis functions, and six friction functions. These basis sets will vary for other kinematic scenarios described later. For clarity, the gravity basis is described here.

$$\phi_g(x) = [\,\sin(q_3 + q_4 + q_5)\,,\ \sin(q_3 + q_4)\,,\ \sin(q_3)\,]^T \tag{65}$$

$$W_g^T = 1e7 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -.2531 & -.4442 & -3.0232 \\ -.2531 & -.4442 & 0 \\ -.2531 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{66}$$

$$G(\theta) = W_G^T \phi_G(x) + \varepsilon_G(x) \tag{67}$$

Unless the robot is placed on a non-level floor and/or some weight is added non-coincident with the wrist axis, there should be no torque due to gravity on joints 1,2 or 6.

Chapter 4 Simulation and Experimentation

Virtual Analysis with Matlab/Simulink

Validating this control theory began with a virtual simulation. This simulation requires an input that engages as many joints as possible while avoiding workspace excursions or singularities. For this, the trajectory of a circle was generated in six dimensions, which includes three static dimensions of orientation.

Next, all the pieces would be assembled in a simulation environment. For this project, matlab Simulink was selected and the layout for the overall simulation is shown below in *Figure 4-1*. In order to follow the trajectory without an operator, this simulation assumes the base rotation is observable and torque controlled while translation of the base is negated. Joint state error is the product of the cartesian error with the inverse jacobian. This enables simulations without a human “in the loop”.

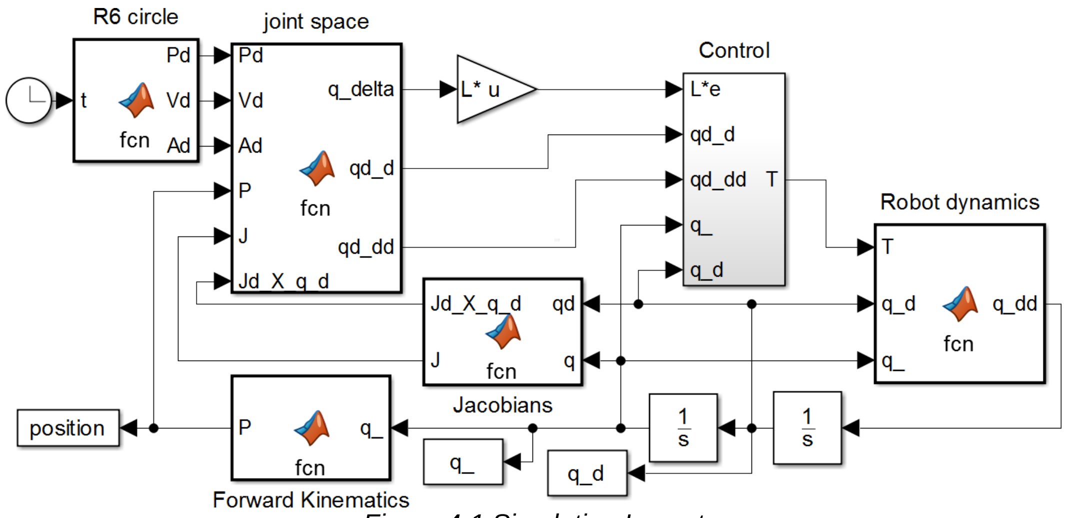


Figure 4-1 Simulation Layout

Simulink is a powerful tool that readily simplifies the task of organizing and implementing dynamic simulation and signal integration. Here each of the ‘fcn’ blocks contain only a few lines matlab scripting according to their title and context. The Control subsystem is somewhat more complex. Visible in *Figure 4-2* below depicts a Simulink implementation of the adaptive torque control described earlier.

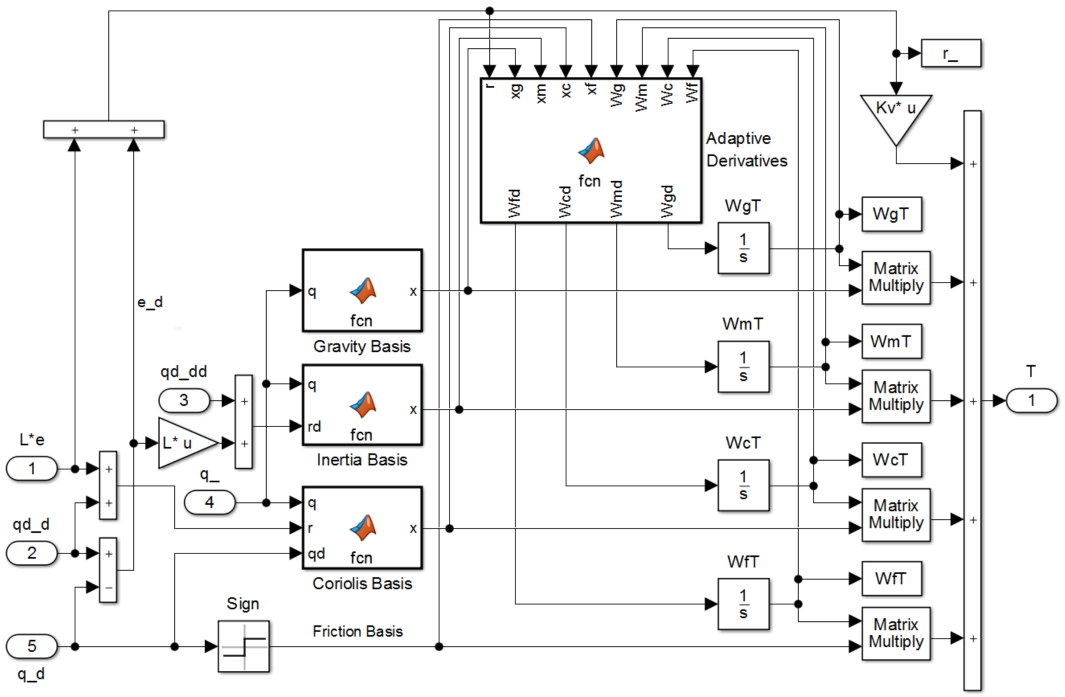


Figure 4-2 Torque control subsystem

To ensure the feasibility of the of jacobian and the success of the adaptive control method, only the six rotational joints off the mobile manipulator are included in this simulation. Remember, the actual manipulator has five joints. Here, the rotation of the mobile base is included as a sixth fully observable and torque controlled joint. Because the manipulator base joint is offset from the platform center, this offsets the world frame of a purely manipulator based reference while providing additional motion ability necessary for the full rank jacobian.

To begin, the device simulation would be initialized in an arbitrary position. This position would seed the path generation and hopefully avoid singularities and workspace excursions. If these should occur, the simulation will quickly fail. Additionally, because the end-effector points arbitrarily away from any specific axis, the exponential coordinates of the rotation matrix can be reliably calculated. This calculation will become a troublesome issue later. There are other parameters to be considered such as proportional derivative gain and weight matrix derivative gain. Determining these requires a thorough analysis of the expected trajectory, disturbances, and basis bounds. Fortunately, these can be rapidly revealed with a series of simulation attempts.

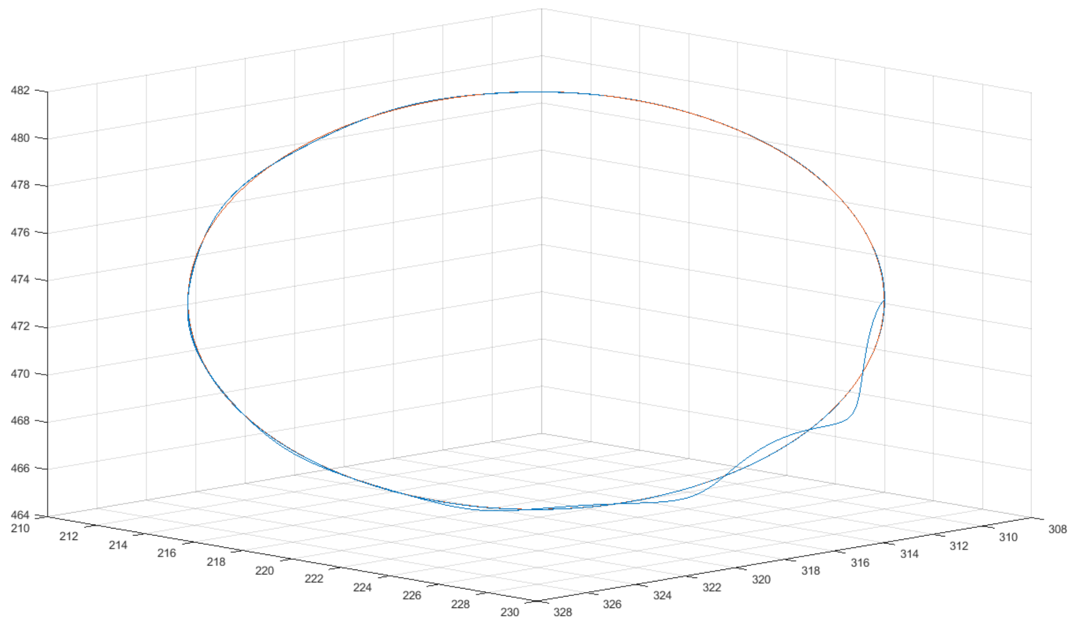


Figure 4-3 20mm Cartesian trace history

Shown in *Figure 4-3* is the task space or cartesian positon of the end effector coordinate frame throughout this simulation. Each axis indicates two millimeter increments of x, y, or z displacement. From these visual results the end-effector in blue stabilizes along the desired circle path in red very well. While visual satisfying, the history of the error state history shown in *Figure 4-4* is more informative. In this figure, each color indicates the error state of a specific joint as labeled in the upper corner from base to wrist. Overall, the error which is unit less begins at something less than .005 and attenuates to something much less than .0001. While tracing approximately three revolutions of the circle trajectory, the error attenuating performance is disturbed in approximately the same location along the circle. Since increasing the circle diameter readily crashes the simulation in the same region due to a workspace excursion, this disturbance is likely due to the desired trajectory exceeding the workspace limits of the manipulator.

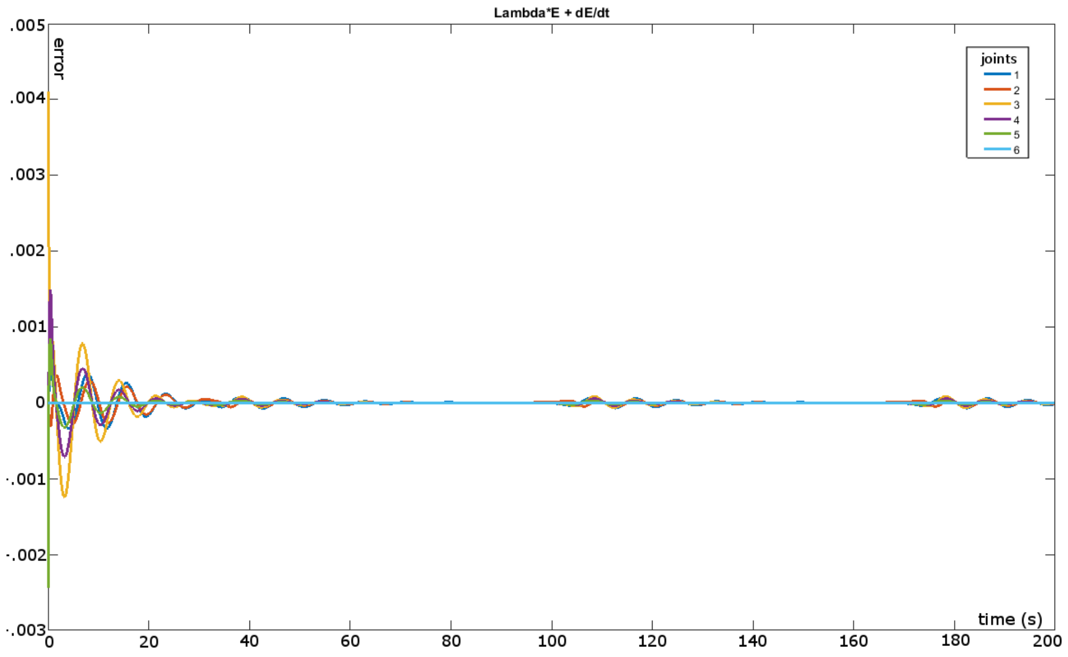
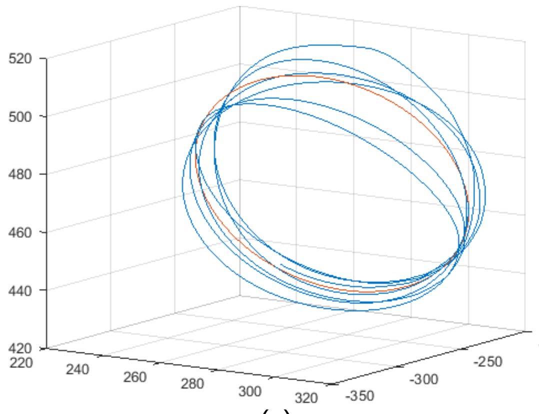
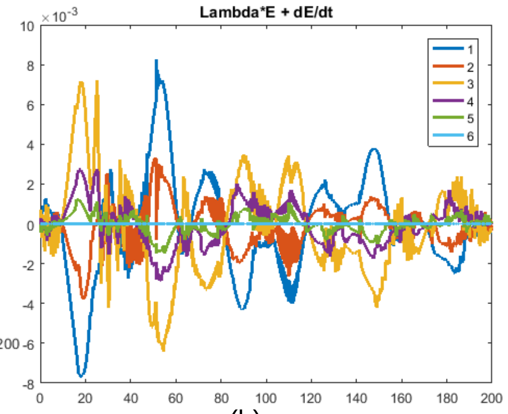


Figure 4-4 Error state history

While this simulation effectively demonstrated the feasibility of this control method, the dynamics were slow and the weight matrices were initialized with their expected final vales. To demonstrate the stabilization of the neural network weight matrices and particularly coriolis, a simulation with more displacement and more speed was required. This lead to the follow results where the start position and velocity were reevaluated and the circle diameter is increased from 20 millimeters to 90 millimeters. Linear control gains and weight integration factors were selected empirically after a series of simulations.



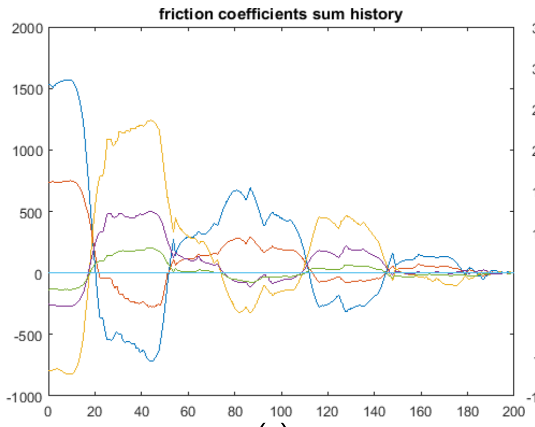
(a)



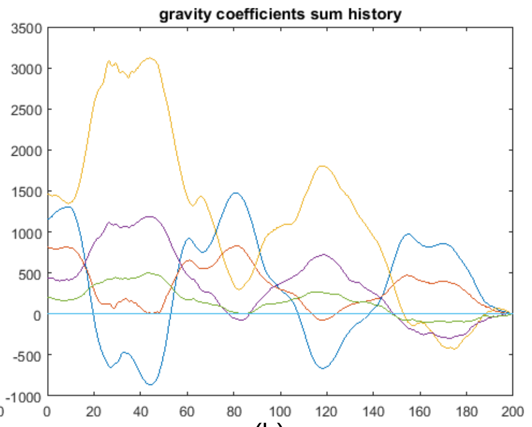
(b)

Figure 4-5 90mm Circle tracing. (a) xyz position and (b) joint state error.

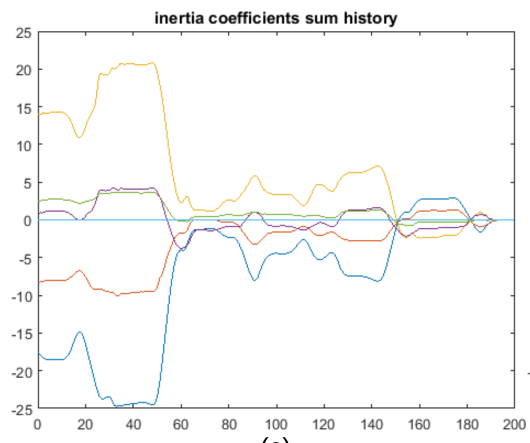
Although initial errors are small, because the dynamics are initialized with desired position and velocity, errors still emerge from the tracing attempt because the weight matrices of the phi functions for this simulation are purposely perturbed from their prescribed ideal states. To demonstrate how the controller, corrects and guides these values to better represent the nature of the system, a summation is performed during the simulation to express the evolution of these matrices, some with several hundred values included. This summary condenses each matrix intro six values representing the six joints of the hypothetical device. While these values shown in *Figure 4-6* don't contribute to the system, they hopefully illuminate the functioning workings of the controller.



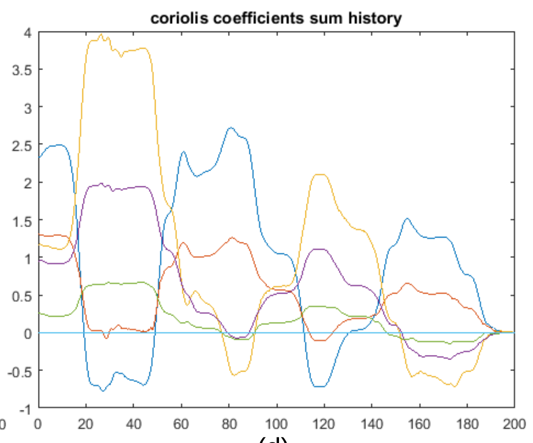
(a)



(b)



(c)



(d)

Figure 4-6 Basis weights. (a) friction (b) gravity (c) inertia (b) coriolis

While successfully demonstrating the possibility of adaptive torque control with guidance through jacobian inversion, this does not yet provide an opportunity for human interaction nor can it be directly applied to this project's robotic platform due its dependency on joint capabilities that will simply not be available. Instead a new mission with the possibility of human guidance while accepting robot limitations is devised.

To demonstrate these capabilities while respecting the limitations of the actual device, a new mission is here described. Instead of a tracing a circle in six dimensions requiring a six jointed manipulator, a seemingly simpler task of tracking and collecting a small passing object is now considered. This requires initializing the manipulator near the ground alongside the base and near its foremost reachability. The procedure here is the robot and operator work collectively to collect objects from a cluttered area. The operator targets an object and directs the base to drive next to it. The robot base continues while the end-effector temporarily matches speed with the target. The operator then delivers fine adjustments to the end-effector position and signals capture. When the effector approaches the opposite end of the manipulator's reachability, any captured object is autonomously delivered to a collection bin. Meanwhile, the operator directs the base to the next target and adjusts speed if the manipulator arm needs more time to prepare and reinitialize.

Having shown the manipulator should perform well with joint trajectories defined to avoid workspace excursions, the next challenge here to guide the joints along a motion vector that negates the movement of the base. With regards to reachability, there exists a circular area around the base joint the effector can easily reach in a z inverted, or end effector pointed down, orientation for object capture. A kinematic solver should show that much of this area can be reached with two positions of the base joint. Further, a dual of elbow joint positions can reach the entire region within the perimeter. These are the partial freedoms that might confuse a simple kinematic solver. Instead, reliance on a now rank deficient jacobian must continue. Using the starting position of the manipulator, this deficiency is well illustrated by the jacobian product with its inverse (68). The near identity appearance indicates that cartesian velocities and z rotations can translated into joint commands that perfectly recover the desired inputs. The x and y rotations however are coupled, which means their instantaneous movements are proportional to each other.

$$J(\theta_i)J^{-1}(\theta_i) \cong \begin{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} & [0] \\ [0] & \begin{bmatrix} .25 & -.433 & 0 \\ -.433 & .75 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix} \tag{68}$$

Fortunately, this is not an immediate problem for the mission selected as no rotations are desired. However, correcting on rotation errors may complicate guidance and adaptive torque stability. For proof of concept a new simulation was devised with appropriate basis functions, initial weights, and guidance calculation.

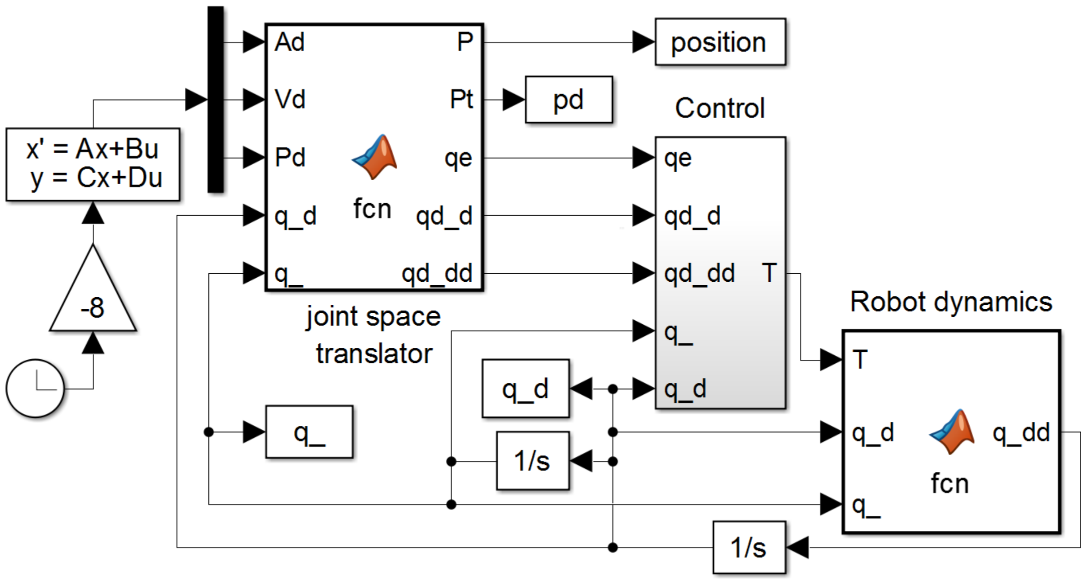


Figure 4-7 Line tracing Simulink layout

Much here is the same as the original six-jointed simulation. The kinematic and jacobian calculation blocks are now incorporated within the joint space translator block, the robot dynamics and integrators are now reduced to only consider the five joints of the actual manipulator, and much of the math within the control block has been almost entirely condensed into a matlab block as shown in *Figure 4-8*.

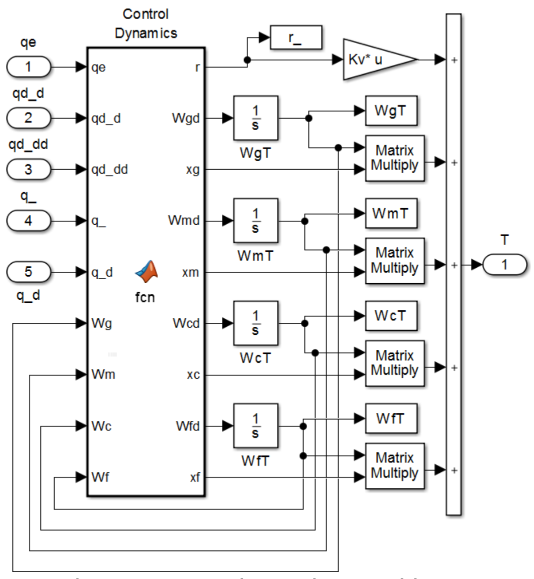


Figure 4-8 condensed control layout

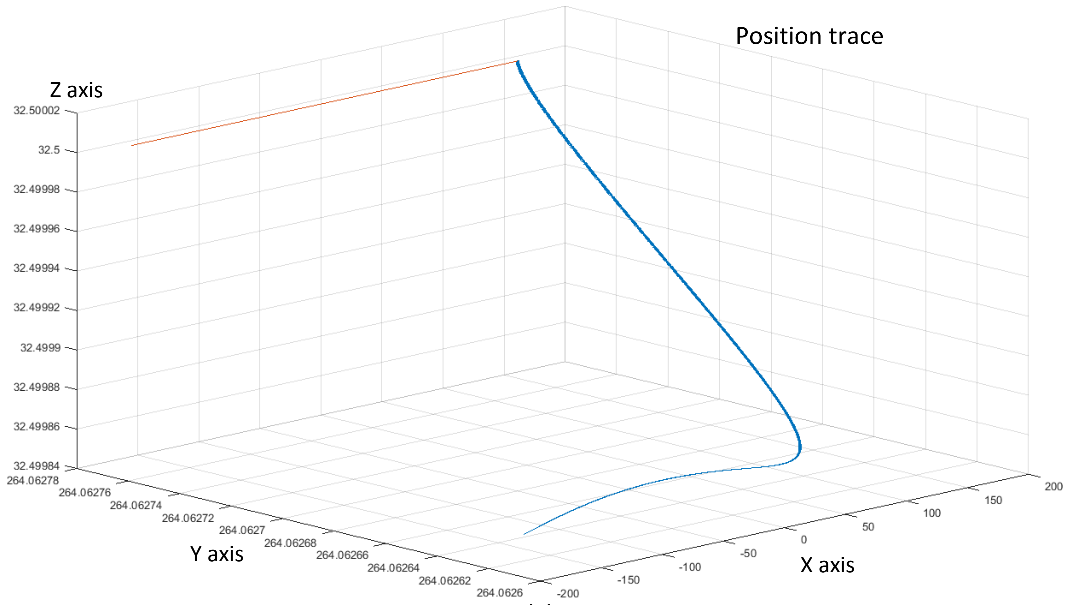
However, initial attempts at this simulation revealed a new challenge. To regulate and maintain a desired orientation, the kinematics must calculate the rotation coordinates precisely. The exponential method described earlier works well when the angle of the end-effector can be described as non-zero. In this task however, the position of the effector is as near zero as the control algorithm can sustain. Buried in the coordinate calculation is a near zero divided by near zero that yielded large rotation errors that greatly confused every other aspect of the simulation. With some creativity, a solution to solve orientation navigation with Euler angles and unit vector based saturation logic was applied. Euler angles effectively describe all rotation matrix sets as the product of elevation, azimuth, and rotation transformations. The resulting rotation matrix in (69) is comprised of trigonometric functions of the orientation angles that must be recovered. Here the letters ***s*** and ***c*** represent the sine and cosine functions with subscripted inputs.

$$R = rot_z(\alpha)rot_y(\beta)rot_z(\gamma) = \begin{bmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta \\ s_\alpha c_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha c_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta \\ s_\beta c_\gamma & s_\beta s_\gamma & c_\beta \end{bmatrix} \tag{69}$$

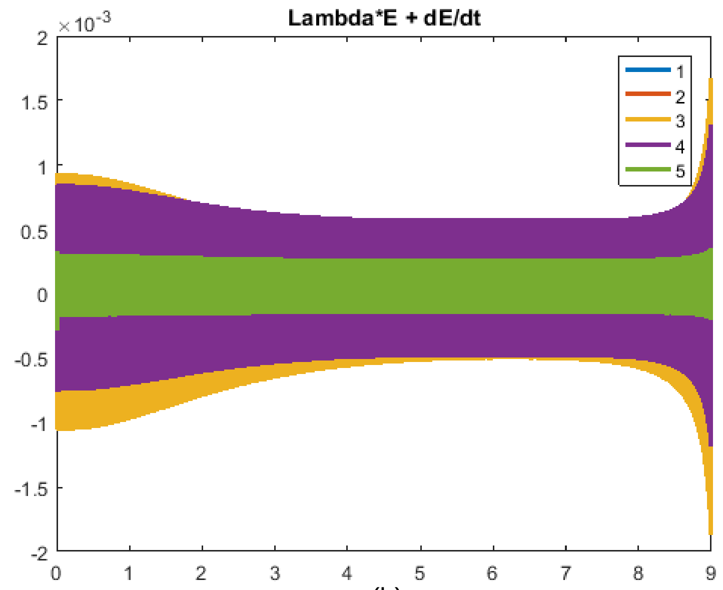
$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \qquad \beta = atan2\left(\sqrt[2]{r_{13}^2 + r_{23}^2}, r_{33}\right), \qquad tol = \sqrt[2]{pi^2 - \beta^2} \tag{70}$$

$$\alpha = \left(-tol \le atan2\left(\frac{r_{23}}{s_\beta}, \frac{r_{13}}{s_\beta}\right) \le tol\right), \gamma = \left(-tol \le atan2\left(\frac{r_{32}}{s_\beta}, -\frac{r_{31}}{s_\beta}\right) \le tol\right) \tag{71}$$

This may not be an ideal solution, but it has been an effective one. With this, the line trace simulation was enabled with cartesian and error state results in *Figure 4-9* a. Similar to the earlier visual results of the circle trace, the units are again in millimeters, however while the x position moves almost 300 millimeters, extremely little motion is seen on any other axis. *Figure 4-9* b shows the joint error state history of the more realistic youBot manipulator.



(a)



(b)

Figure 4-9 Line tracing. (a) xyz position and (b) joint state error.

From the results, the system does not converge on the desired state as closely as one might expect. This may be due to the nature of orientation logic and/or the deficient jacobian guidance. Still, the y and z cartesian error is approximately 0.14 µm which is thoroughly below any human perception and easily accepted. More concerning is the divergent nature of the error in the final second of simulation. This behavior is almost certainly due to the approaching workspace limit. In an object capture scenario, this would be the moment to abort capture and reinitialize for the next target.

Some opportunity for proactive troubleshooting remains. For example, tracing a line may not be the best opportunity to refine the weight matrix estimation. To further validate the torque control with jacobian guidance performance, a new simulation was considered to draw continuous circles within the manipulator's workspace. Due to the limited nature of the manipulator, a flat circle directly adjacent the device base defined the details of the challenge. This proved incredibly more difficult than tracing the line.

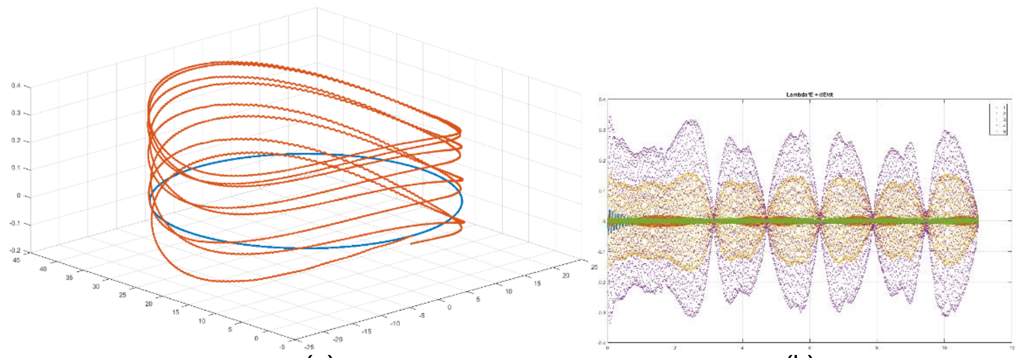


Figure 4-10 Circle tracing. (a) xyz position and (b) joint state error.

From results shown in *Figure 4-10* the error results show magnitudes greater than the line trace. This may be due to the deficient jacobian matrix or the poor angular coordinate calculation. Whichever the case, the nature of simulation requires the neural net to continuously retrain to keep the system somewhat stable. This continuous training is evident in the line trace as well though it is much subtler. It was simply not possible for the controller to maintain stability with a static network. That's to say, while the theory indicates that if the joint trajectory and physics model is correct, then the network should stabilize onto some static weight set that together with the linear controller gains maintains a stable behavior. Further, the development of these simulations revealed an extremely narrow range of control gains that retained stability. Because the virtual physics is completely known, an error is suspected to exist in the trajectory calculation, namely the jacobian and/or angular coordinates.

A late epiphany resulted in an alternative method of orientation navigation which used the matrix logarithm function (10). With this method, a series of purely kinematic simulations of the five jointed manipulator were performed using equation (23).

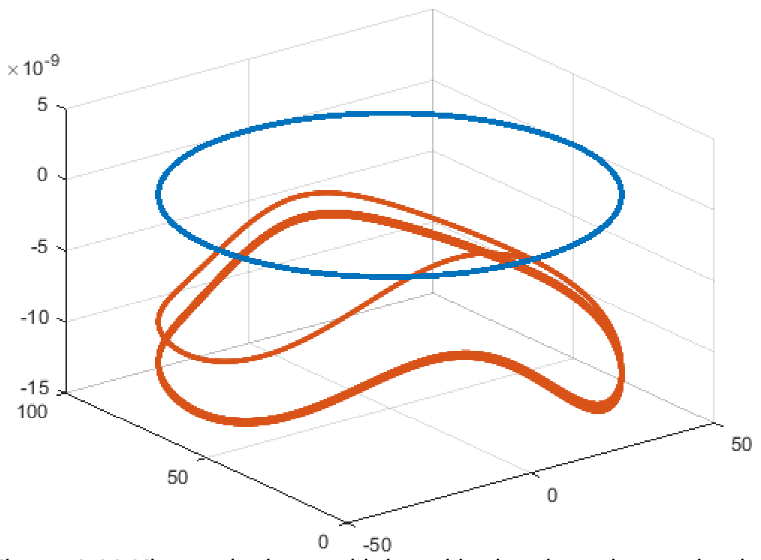


Figure 4-11 Kinematic demo with logarithmic orientation navigation

This navigation method was simulated for twenty-five revolutions of a one-hundred-millimeter circle while demonstrating errors less than 10^{-8} as shown in *Figure 4-11*. However, the error history shown below in *Figure 4-12* seems to indicate some growing instability. It is possible that the system is over-actuated system because the simulated device traces a desired path defined in six dimensions whilst utilizing only five joints. Because of this, error signals into the controller may signal conflicting correction information, which may be what is allowing these errors to accumulate. The orientation navigation is the likely source of this error, as the position history seems stable.

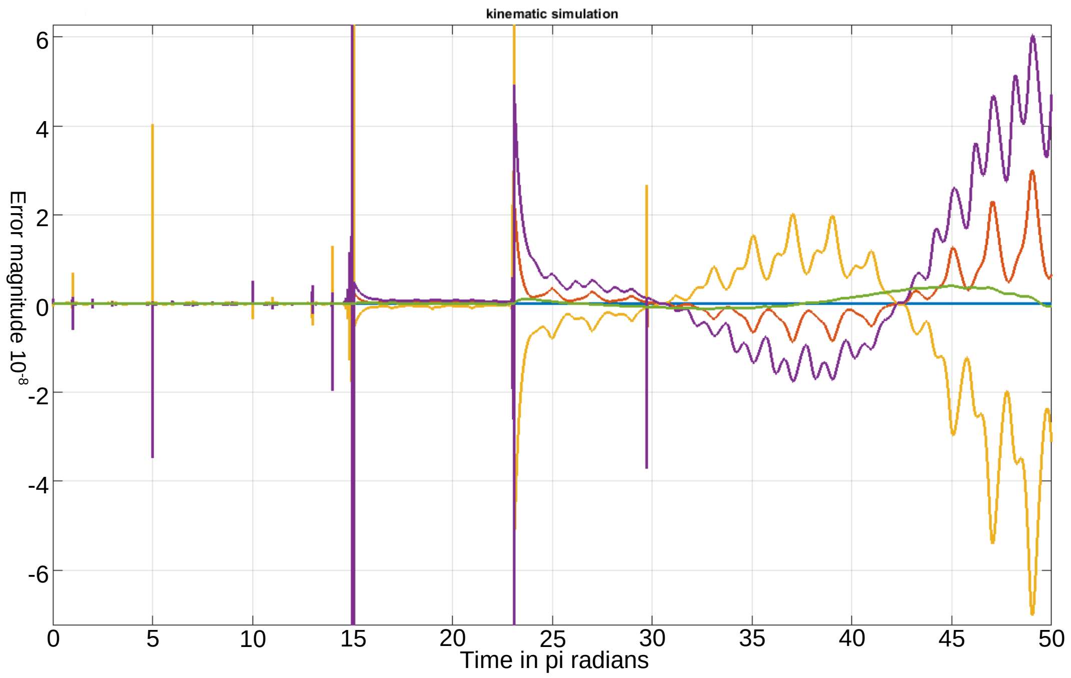


Figure 4-12 Error history of logarithmic orientation navigation

This history of the errors for each joint never exceeds 10^{-7} units. Further, until thirty-five pi radians, they generally remain below ten to the minus eight in magnitude. Of course, this is ignoring these regularly occurring spikes. These seem to occur every pi radian and have something to do with to orientation navigation toggling. To clarify, in the inverted position the manipulator is attempting to maintain, the rotation angles should be $[0\ pi\ 0]^T\ or\ -[0\ pi\ 0]^T$. Because of this 'wrapping' of the angular position at the trigonometric extremes, the controller effectively pulses an extreme error into the simulation. In light of this and considering there are only five degrees of freedom to control a method to mitigate this wrap effect is proposed by means of a gain matrix between the cartesian error and inverse jacobian (72).

$$q_e = J^i K(P_d - P), \qquad K = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & k & \\ & & & & & 1 \end{bmatrix}, \qquad k = 0 \tag{72}$$

Through this method, inputs to one of the coupled motions of the end effector is entirely negated. The thought is, if only five joints can be actuated then perhaps only five error signals should be regulated. The results of this kinematic simulation are shown in *Figure 4-13*.

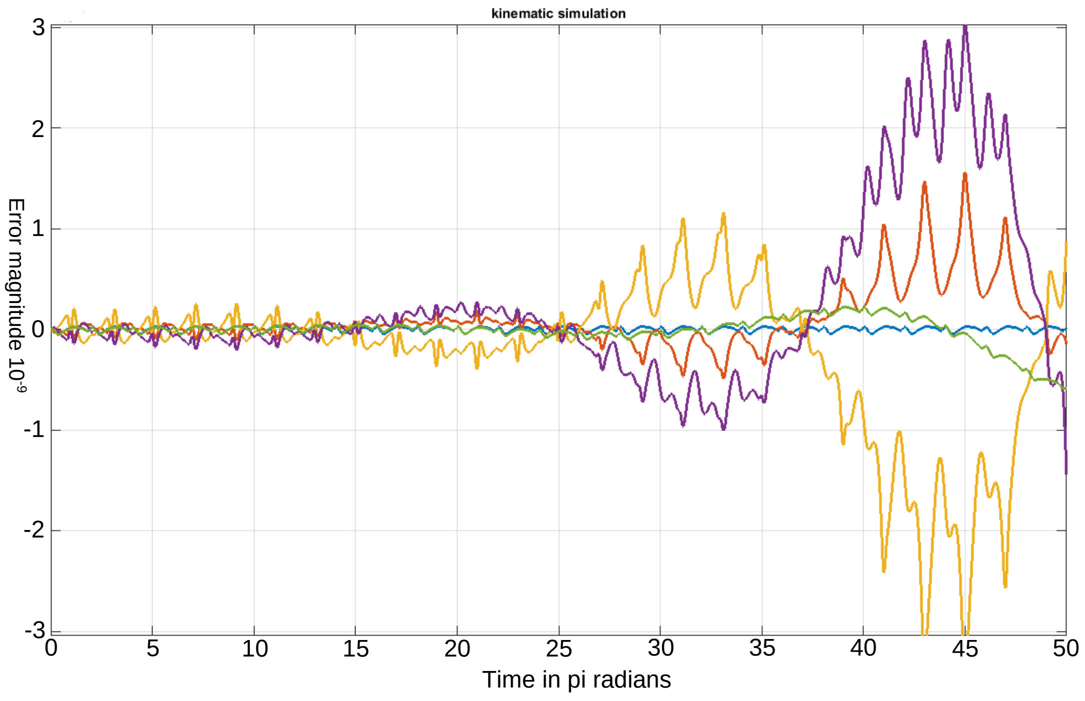


Figure 4-13 Error history with k=0

Here, because the k value effectively nullifies this wrapping, the pulsing of the error signal by the controller is virtually nonexistent at first but begins to re-assert itself later. This must be because the remaining two dimensions of orientation are also wrapping and signaling confusing corrections to the control method. Thus, rather than simply zero the influence of the magnitude of this primary vector, a series of experiments were attempted to rapidly explore a means balancing these dimensional extremes against each other. To clarify, this kinematic simulation was reevaluated numerous times to find an empirical factor to maintain a stable error performance. This ultimately yielded an extremely small and negative k factor. *Figure 4-14* shows the results of the latest simulation. This position trace like many earlier is visual satisfying with a trend that seems to suggest stabilizing around a static error of 4e-8. Apparent from *Figure 4-15*, though the error is extremely small, less than 10^{-9}, it is not attenuating. However, it is difficult to ascertain stability since the dynamics of kinematic navigation have not been formulated. Qualitatively, while the initial error is substantially larger than before, it now seems to hold a stable magnitude for twenty-five revolutions of this circle which suggests but not proves a stable system.

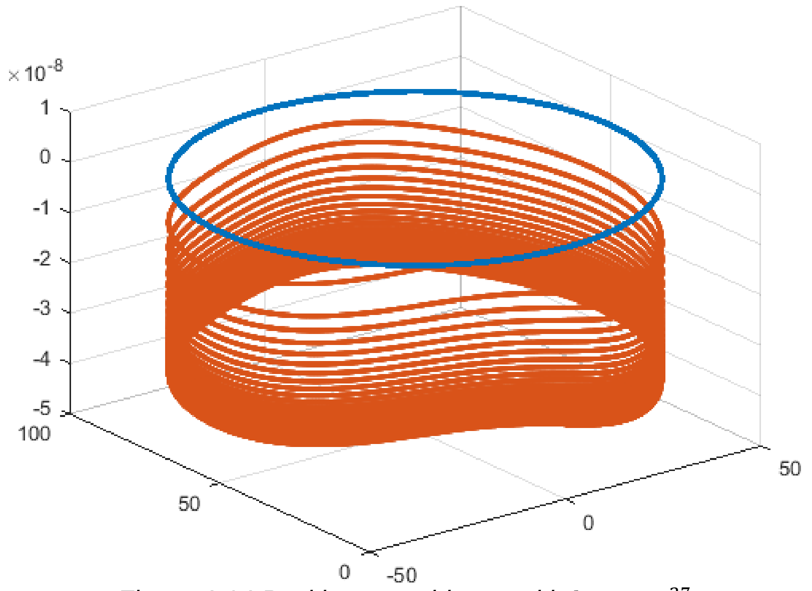


Figure 4-14 Position trace history with $k = -2^{-27}$

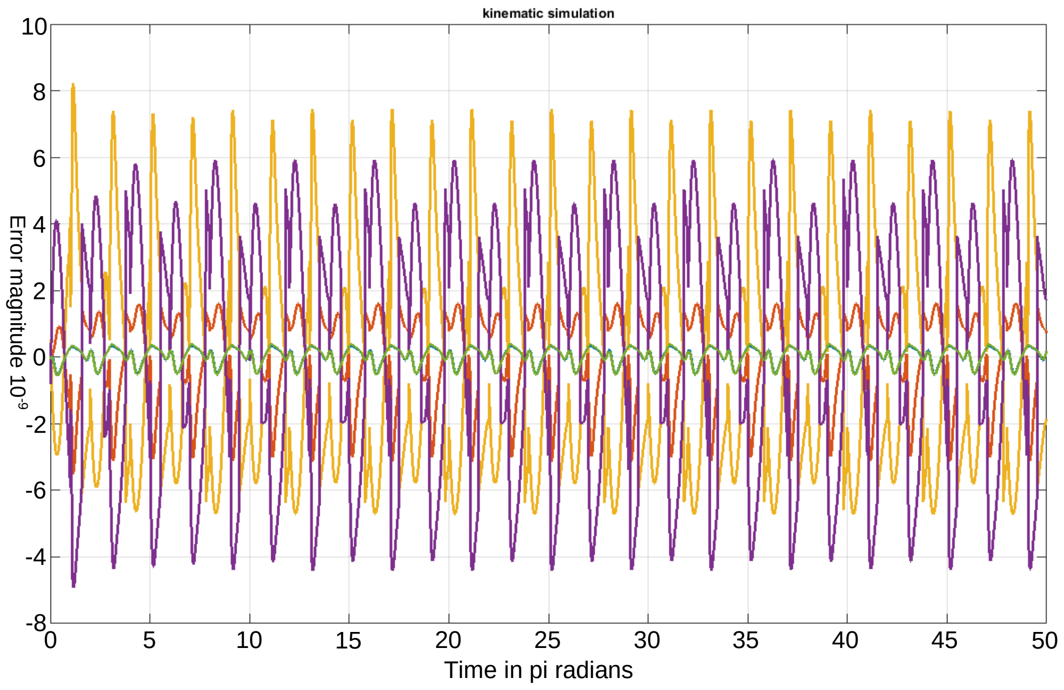


Figure 4-15 Error history with $k = -2^{-27}$

Physical Experimentation

To begin constructing an extensive demonstration on a physical device, a plan to program and validate small pieces might help accelerate the troubleshooting and debugging process of novel code development. As a simple introduction into torque based control, a project to implement gravity mitigation only, served as an ideal introduction for working with the youBot device itself. Before trajectories, or jacobian guidance, or adaptive methods are developed on the platform, the method of simple robust torque control was implemented. Based on the dynamic analysis earlier, only three individual trig functions need to be evaluated to mitigate the influence of gravity. The difficult part here is only learning the nature of programming the device which required extensive assistance from online communities in similar fields [*44*] [*45*]. As with most software, the simple step of computing the robust torque value must be preceded by numerous initialization commands and also followed by a safe shutdown procedure. For clarity, only the code relevant to this concept is stated below.

```
q2 = armSensedAngles[2-1].angle.value() -  65  *3.141592654/180;
q3 = armSensedAngles[3-1].angle.value() + 146  *3.141592654/180;
q4 = armSensedAngles[4-1].angle.value() - 102.5*3.141592654/180;

t4 =  0 - ( 77970861*sin(q2 + q3 + q4))/ 50000000;
t3 = t4 - (692273061*sin(q2 + q3)     )/250000000;
t2 = t3 - (493626447*sin(q2)          )/100000000;

armSetTorques[2-1].torque = t2 * newton_meters;
armSetTorques[3-1].torque = t3 * newton_meters;
armSetTorques[4-1].torque = t4 * newton_meters;
```

This code, shows the key steps for this simple torque control application. The joint positions are observed and adjusted by the difference between the natively zero ‘nested’ configuration and the kinematic known extended position. The torque is evaluated based on the dynamic model as expressed by the three trigonometric functions of the gravity model. Finally, these torques are imposed on the motors through the driver functions. Since the programing has no desired pose or trajectory, the execution merely holds the manipulator aloft against the force of gravity. This behavior could be applied as a means of physically programming the device using a series of motions described by the operator to the device by physically configuring the manipulator.

The next critical component would be to demonstrate the feasibility of the jacobian guidance. Fortunately, the nature of this platform’s API [*45*] [*46*] enables direct position control of the actuators. Therefore, a simple and critically unique component is to demonstrate the Jacobian guidance. Following the task concept detailed earlier, the end effector begins near the front of the device and using only joint velocity commands is made to track alongside the platform base. Some of the code is available in appendix A as well as information to find a video record. The code reveals some adjustments were made to accommodate each joint’s zero position and apparent rotation direction.

Subjectively, the end effector appeared to stray from desired linear path. It should be noted that the code includes some state feedback to observe and correct the manipulator into the desired direction. However, the performance is ultimately dependent on the application program interface included a motion control which has done well enough to validate the Jacobian potential at this point. Unfortunately, no data is available since the development immediately continued into the next phase.

Because the dynamics of the device are poorly known, a sensible approach to begin the torque control validation should be to develop a routine to engage and refine the neural network weights that characterizes the program’s understanding of these properties. To this end, an arbitrary trajectory is constructed to purposely challenge the device with seemingly frantic motion. A portion of the code constructed for this phase is available in appendix B.

Critical to the performance in this phase as well as many feedback dependent applications is the ability to observe and update the system input with some minimum regularity. Something that would be a fascinating and useful to future control projects might be quantify what this rate needs to be. Unfortunately, the stability proofs are constructed in continuous time while practical application of the math requires discrete steps. As the programming was originally developed, an arbitrary cycle rate of one millisecond was selected to complete the necessary calculations while providing the continuous feedback necessary for stable performance. This was later revealed to be unachievable. Despite this, the controller updated the torque input roughly every two and half milliseconds which proved robust enough to maintain the stability of the system. Details to find a video record of this code in action are available in the appendix.

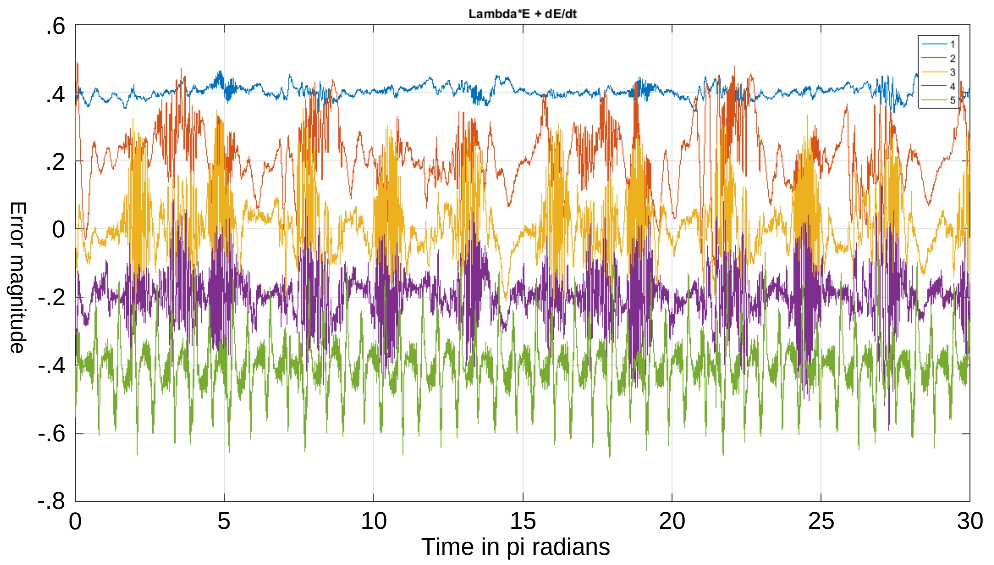


Figure 4-16 Adaptive demo error state history with traces offset

Shown in *Figure 4-16* is the only data from the physical testing of the adaptive controller. To improve the clarity of the graph, each error trace has a unique zero offset in .2 unit increments. This performance is admittedly disappointing however despite the very frantic motion and seemingly extreme dynamics and error excursions, the manipulator never goes unstable and maintains a generally smooth behavior. As the robot pursues the desired trajectory, it visibly stumbled and shuddered under close scrutiny. This may be due to frictional type phenomena such as damping and static friction that have not been anticipated. Because the control is based on a single layer functional level neural network, the program's ability to adapt to unanticipated phenomena is extremely limited. This performance demonstrates the feasibility of adaptive torque control and enables the possibility of thoroughly robust control modes for physical interaction programs.

Chapter 5 Conclusion

Discussion

Despite many difficulties, the methods detailed here can be applied to rapidly quantify and deploy manipulators of serial joint chains. It was noted during physical execution that the performance of the actual device in the limited scenarios demonstrated surprising controller gain tolerance where the virtual device did not. Specifically, the device demonstrated stability across a wider range of selected proportional derivative feedback gain as well as the neural network dynamic factor. This may be due to the un-modeled friction phenomena creating some stabilizing influence. Thus far, demonstration of physical human machine interaction, is shown through a robust torque mode to apply gravity compensation only. This is a pertinent beginning for physical human machine interaction, since this could allow an operator to program a device by physically guiding a manipulator through a series of motions which it would record and repeat however its instructions require. Because this functional layer NN allows separation of the various forces acting on the device, and adapted controller could enable its gravity network to hold its limbs aloft while a human guides and programs it. Conceivably, a static controller could also enable an artificial inertia for massive robots operated by relatively light humans.

A greater aspiration is that complex tasks with high dynamic motion can be made to recognize operator input with entirely proprioceptive sensing. Assuming a given desired trajectory with a stable neural network definition, external disturbances beyond an arbitrary threshold can be recognized as physical human commands. The magnitude and direction of such error vectors could, though appropriate logic, guide revision of the desired trajectory.

Future Work

Of immediate interest is to find and apply a better method of angular coordinate calculation, or perhaps better understand the nature of the Jacobian. There may also be tools of inverse kinematic solution that are potentially more robust than the jacobian method. Further, an improved frictional physics model of the device could greatly improve the torque control performance. This might be achieved by constructing a multi-layer sigmoid activation network parallel to the established subnets. While this method would not directly reveal the underlying nature, it could well imitate an empirical model. This model could either be left to simply improve the control or extracted and studied virtually to perhaps recognize the classical physics components that might further improve understanding and performance.

Additionally, opportunities remain to demonstrate purposeful disturbance recognition. This would be the device following a known trajectory, could recognize disturbances as described earlier and demonstrate goal revision based on the nature of the disturbance. This could yet be accomplished with jacobian guidance for a high degree of freedom manipulator. Since the original simulation still indicates, stable error and network behavior is possible where the kinematics are well away from singularities and the manipulator joints are capable of a full rank jacobian.

Appendix A

Matlab Jacobian Construction with and youBot C++ code

```
%%  Jacobian via product of exponentials [12] [13]
% from rvctools import skew, vex, t2r

IAC = 'IgnoreAnalyticConstraints';
x = [1 0 0]; y = [0 1 0]; z = [0 0 1]; e = [0 0 0]; n = 5;
wedge   = @( twist )[ skew(twist(4:6)), twist(1:3); e 0 ];
adjoint = @( tf ) ...
[ t2r(t), skew(t(1:3,4))*t2r(t); zeros(3), t2r(t)];
theta   = @( tf ) ...
simplify( acos( (trace( t2r(tf))-1)/2), IAC, true);
vee     = @( tf ) ...
 [ tf(1:3,4); sin(theta(g))\theta(g)*vex( t2r(tf) ) ];

q  = sym( 'q',[1,n]); assume( q, 'real');
qd = sym('qd',[1,n]); assume(qd, 'real');

Xi= [   [-cross(z,   0*x+   0*z),z]', ...
        [-cross(y,  33*x+ 267*z),y]', ...
        [-cross(y,  33*x+ 422*z),y]', ...
        [-cross(y,  33*x+ 557*z),y]', ...
        [-cross(z,  33*x+ 724*z),z]'];
g  = [ eye(3), x'* 33 + z'*724; e 1 ];

Js = sym(zeros(6,n)); A = eye(4);
for i = 1:n; fprintf('.');
    Js(:,i) = simplify( adjoint( A )*Xi(:,i), IAC, true);
    A = A*expm(wedge(Xi(:,i))*q(i));
end; fprintf('\n') % spatial jacobian

Jb = sym(zeros(6,n)); A = g;
for i = n:-1:1; fprintf('.');
    A = expm( wedge( Xi(:,i))*q(i))*A;
    Jb(:,i) = simplify( adjoint(A)\Xi(:,i), IAC, true);
end; fprintf('\n') % body jacobian

T = eye(4);
for i = 1:n; fprintf('.');
    T = T*expm( wedge( Xi(:,i))*q(i) );
end; fprintf('\n');
T = simplify( T*g, IAC, true); Tt = vee(T*g);
disp('The Tranformation twist:'); disp(Tt)

pd = simplify( wedge(Js*qd)*T(:,4) );
    % partial solution
Jw = adjoint( [eye(3) -T(1:3,4); e 1] )*Js;
    % final PoE method for analytic jacobian

Jv = jacobian( vee(T), q );
disp('The analytic Jacobian:'); disp(Jv)
```

```
// 6x5 jacobian in c [47] translated to MS word clearest I could
J <<  -sin(q_(1))*(167*sin(q_(2) + q_(3) + q_(4)) +
       cos(q_(1))*(167*sin(q_(2) + q_(3) + q_(4)) +


135*sin(q_(2) + q_(3)) + 155*sin(q_(2)) + 33),
135*sin(q_(2) + q_(3)) + 155*sin(q_(2)) + 33),
                                            0,
                                            0,
                                            0,
                                            1,

cos(q_(1))*(167*cos(q_(2) + q_(3) + q_(4)) + 135*cos(q_(2) +
sin(q_(1))*(167*cos(q_(2) + q_(3) + q_(4)) + 135*cos(q_(2) +
          - 167*sin(q_(2) + q_(3) + q_(4)) - 135*sin(q_(2) +


q_(3)) + 155*cos(q_(2))), cos(q_(1))*(167*cos(q_(2) + q_(3) +
q_(3)) + 155*cos(q_(2))), sin(q_(1))*(167*cos(q_(2) + q_(3) +
q_(3)) - 155*sin(q_(2)) ,            - 167*sin(q_(2) + q_(3) +
            -sin(q_(1)) ,
            -cos(q_(1)) ,
                       0,

q_(4)) + 135*cos(q_(2) + q_(3))), (167*cos(q_(1) + q_(2) + q_(3)
q_(4)) + 135*cos(q_(2) + q_(3))), (167*sin(q_(1) + q_(2) + q_(3)
q_(4)) - 135*sin(q_(2) + q_(3)) ,
                    -sin(q_(1)) ,
                    -cos(q_(1)) ,
                               0,

+ q_(4)))/2 + (167*cos(q_(2) - q_(1) + q_(3) + q_(4)))/2,
+ q_(4)))/2 - (167*sin(q_(2) - q_(1) + q_(3) + q_(4)))/2,
              -167*sin(q_(2)        + q_(3) + q_(4) )   ,
                                          -sin(q_(1)),
                                           cos(q_(1)),
                                                    0,

                                  0,
                                  0,
                                  0,
sin(q_(2) + q_(3) + q_(4))*cos(q_(1)),
sin(q_(2) + q_(3) + q_(4))*cos(q_(1)),
           cos(q_(2) + q_(3) + q_(4));
```

https://youtu.be/0wNv2FnbS7o

Appendix B

Selected Code for Adaptive Torque

```
// <- setup and initialization steps for adaptive torque
while (running) {
usleep( (int)(tn - (t-t0)) ); // first loop should wait zero

// observe robot joint states
observe_joints( &myYouBotManipulator );

// static joint error, method varies with jacobian guidance
qe = qd_ - q_;

// calculate control vectors
control_vectors();

// calculate torque vector
Trq = Kv*r + Wf*phi_f + Wg*phi_g + Wm*phi_m + Wc*phi_c;

// for loop -> push torques in armSetTorques
myYouBotManipulator.setJointData(armSetTorques);

// update weights matrices, integrate: W_T = F*phi_*rT
controller_dynamics();

// run logging ...
// ready next desired joint states
tn += delta_t*1e+06;
trajectory( 1e-06*tn );

// time to wait until time next (tn)
t = getMicrotime();

// run for _ seconds
running = ((t < (t0 + 1e7)) & (t > t0));
}

void controller_dynamics(){
// weight matrix derivative integration
Wf = Wf + delta_t/2 * s*(r*phi_f.transpose() - k*r.norm()*Wf);

Wg = Wg + delta_t/2 * s*(r*phi_c.transpose() - k*r.norm()*Wg);

Wm = Wm + delta_t/2 * s*(r*phi_m.transpose() - k*r.norm()*Wm);

Wc = Wc + delta_t/2 * s*(r*phi_c.transpose() - k*r.norm()*Wc);
// k set to a very small value, typ k=1e-08
```

https://youtu.be/Lk1DX9lSe9s

References

[1] Frank L Lewis, S Jagannathan, and A Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Philadelphia: Taylor & Francis, 1999.

[2] University of Texas at Arlington. (2016, July) Next Generation Systems. [Online]. http://www.uta.edu/ee/ngs/

[3] National Science Foundation, "NRI-Small: Multi-modal sensor skin and garments for healthcare and home robots," University of Texas at Arlington, Standard Grant 1208623, 2013.

[4] J. Norberto Pires, "Robot Manipulators and Control Systems," in *Industrial Robots Programming*.: Springer US, 2007, pp. pgs: 35-107.

[5] University of Texas at Arlington, Electrical Engineering Dept. (2016, April) Systems and Controls Course Sequence. [Online]. http://www.uta.edu/utari/acs/controls/controlcrse.htm

[6] C. C. Cheah, C. Liu, and H. C. Liaw, "Stability of inverse Jacobian control for robot manipulator," in *International Conference on Control Applications*, Taipei, Taiwan, 2004, pp. 321 - 326 Vol.1.

[7] Ghassan M. Atmeh et al., "Implementation of an adaptive, model free, learning controller on the Atlas robot," in *American Control Conference*, Portland, OR, 2014, pp. 2887 - 2892.

[8] Hamidreza Modares, A. Karimpour, and A. Rowhanimanesh, "A novel adaptive neural sliding mode control for systems with unknown dynamics," in *Third International Workshop on Advanced Computational Intelligence (IWACI)*, Suzhou, Jiangsu, 25-27 Aug. 2010, pp. 40 - 45.

[9] Rainer Bischoff, Erwin Prassler, and Ulrich Huggenberger, "KUKA youBot - a mobile manipulator for research and education," in *International Conference on Robotics and Automation (ICRA)*, Shanghai, 9-13 May 2011, pp. 1 - 4.

[10] Andy Chang, "Mobile manipulators go mainstream," *Machine Design*, no. 00249114, Oct 2013.

[11] Benjamin Keiser, "Torque Control of a KUKA youBot Arm," Zurich, 2013.

[12] Richard M. Murray, Zexiang Li, and S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulation*. Berkeley: CRC Press, 1994.

[13] F. C. Park and D. J. Pack, "Motion control using the product-of-exponentials kinematic equations," in *International Conference on Robotics and Automation (ICRA)*, Sacramento, CA, 9-11 Apr 1991, pp. 2204 - 2209 vol.3.

[14] Shashank Sharma, Gerhard K. Kraetzschmar, Rainer Bischoff, and Christian Scheurer, "Unified Closed Form Inverse Kinematics for the KUKA youBot," in *Robotics; Proceedings of ROBOTIK*, Munich, Germany, 21-22 May 2012, pp. 1 - 6.

[15] Yaolun Zhang, Xiao Xiao, and Yangmin Li, "A novel kinematics analysis for a 5-DOF manipulator based on KUKA youBot," in *International Conference on Robotics and Biomimetics*, Zhuhai, 2015, pp. 1477 - 1482.

[16] Carla González, Luis Moreno, and Dolores Blanco, "A memetic approach to the inverse kinematics problem," in *International Conference on Mechatronics and Automation*, Chengdu, 5-8 Aug. 2012, pp. 180 - 185.

[17] Ignacy Duleba and Michal Opalka, "A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators," *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 2, p. 373, Jun 2013.

[18] C. C. Cheah and Y. Zhao, "Inverse Jacobian regulator for robot manipulator: theory and experiment," in *Conference on Decision and Control*, 14-17 Dec. 2004, pp. 1252 - 1257 Vol.2.

[19] Ross L. Hatton, "Geometric Mechanics of Locomotion and Optimal Coordinate Choice," Carnegie Mellon University, Pittsburgh, PA, Thesis 3455985, 2011.

[20] Shinobu Sasaki, "A complete description of robot manipulator dynamics based on Lagrangian mechanics," Japan Atomic Energy Research Inst, Tokyo, DE90-741467 1989.

[21] Suresh Sampathkumar, "Real time motion control for natural human robot interaction," Arlington, Texas, Master of Science Thesis 2013.

[22] Sven Cremer et al., "Investigation of human-robot interface performance in household environments," in *SPIE Vol. 9859, Sensors for Next-Generation Robotics III*, 13 May 2016, p. 13 pages.

[23] Chandan Datta, "Programming Behaviour of Personal Service Robots with Application to Healthcare," The University of Auckland, New Zealand, Dissertations & Theses 3648403, 2014.

[24] M. Nokata, H. Ishii, and K. Ikuta, "Safety-optimizing method of human-care robot design and control," in *International Conference on Robotics and Automation*, 2002, pp. pp. 1991-1996.

[25] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [Grand Challenges of Robotics]," *IEEE Robotics & Automation Magazine*, pp. vol. 14, no. 1, pp. 20-29, 2007.

[26] Martin Wassink and Stefano Stramigioli, "Towards a novel safety norm for domestic robotics," in *International Conference on Intelligent Robots and Systems*, San Diego, CA, 2007, pp. pp. 3354-3359.

[27] Rommel Alonzo, Sven Cremer, Fahad Mirza, Sandesh Gowda, and Larry Mastromoro, "Multi-modal sensor and HMI integration with applications in personal robotics," in *SPIE*, Baltimore, Maryland, 2015.

[28] Sven Cremer, Isura Ranatunga, and Dan O. Popa, "Robotic waiter with physical co-manipulation capabilities," in *International Conference on Automation Science and Engineering (CASE)*, Taipei, 2014, pp. 1153-1158.

[29] Shaikh Md Rubayiat Tousif, "Physical Human Robot Interaction using model reference neuroadaptive control," Arlington, Texas, Master of Science Thesis 2014.

[30] Pablo Sanchez-Sanchez and Fernando Reyes-Cortes, "A New Cartesian Controller for Robot Manipulators," in *International Conference on Intelligent Robots and Systems*, 2005, pp. 3733 - 3739.

[31] Q. -L. Huang, J. Wu, and R. Xiong, "A solution of inverse kinematics for 7-DOF manipulators and its application," in *World Congress on Intelligent Control and Automation (WCICA)*, Beijing, 2012, pp. 3711 - 3717.

[32] S. Jagannathan and Frank L. Lewis, "Multilayer neural network controller for a class of nonlinear systems," in *International Symposium on Intelligent Control*, Monterey, CA, 27-29 Aug 1995, pp. 427 - 432.

[33] Jean-Jacques Slotine and Weiping Li, *Applied Nonlinear Control*. University of Michigan: Prentice Hall, 1991.

[34] Isura Ranatunga, Sven Cremer, Frank L. Lewis, and Dan O. Popa, "Intent aware adaptive admittance control for physical Human-Robot Interaction," in *International Conference on Robotics and Automation (ICRA)*, Seattle, WA, 26-30 May 2015, pp. 5635 - 5640.

[35] Isura Ranatunga, Sven Cremer, Dan O. Popa, and Frank L. Lewis, "Neuroadaptive control for safe robots in human environments: A case study," in *International Conference on Automation Science and Engineering (CASE)*, Gothenburg, 24-28 Aug. 2015, pp. 322 - 327.

[36] Isura Ranatunga, Shaikh M. Tousif, Dan O. Popa, and Frank L. Lewis, "Adaptive Admittance Control for Human-Robot Interaction Using Model Reference Design and Adaptive Inverse Filtering," *Transactions on Control Systems Technology*, pp. 1 - 8, May 2016.

[37] John T. Wen, "A Unified Perspective on Robot Control: the Energy Lyapunov Function Approach," in *Conference on Oeclslon and Control*, Honolulu, Hawall, 1990, pp. 1968 - 1973 vol.3.

[38] Rafael Kelly and Victor Santibanez, "Strict Lyapunov Functions for Global Regulation of Robot Manipulators," in *International Conference on Robotics and Automation (ICRA)*, Nagoya, 21-27 May 1995, pp. 2758 - 2763 vol.3.

[39] Paul N. Vernaza, "Efficient learning and inference for high-dimensional Lagrangian systems," University of Pennsylvania, Pennsylvania, Dissertations & Theses 3463096, 2011.

[40] Hamidreza Modares, Dan O. Popa, Frank L. Lewis, and Isura Ranatunga, "Optimized Assistive Human–Robot Interaction Using Reinforcement Learning," *Transactions on Cybernetics*, pp. 655 - 667, February 2015.

[41] (2016) Kuka youBot product page. website. [Online]. http://www.kuka-robotics.com/germany/en/products/education/youbot/

[42] (2016) KUKA youBot Online Exhibition. [Online]. http://www.expo21xx.com/automation21xx/20317_st3_mobile-robots/contact.htm

[43] Creative Commons Attribution. (2016, July) ROS Hydro Medusa. [Online]. http://wiki.ros.org/hydro

[44] (2016, July) youBot Driver Documentation. [Online]. https://janpaulus.github.io/

[45] Linsalata Robert S., "Development of a Universal Robotics API for Increased Classroom Collaboration within Robotics Education," Tufts University, Massachusetts, Dissertations & Theses 1512750, 2012.

[46] Benoît Jacob and Gaël Guennebaud. (2016, June) Eigen is a C++ template library. [Online]. http://eigen.tuxfamily.org/

[47] KUKA, *KUKA youBot User Manual*. Stuttgart: Locomotec, December 6, 2012.

[48] Tzyh-Jong Tarn, Zuofeng. Li, Xiaoping Yun, and Anta1 K. Bejczy, "Effect of motor dynamics on nonlinear feedback robot arm control," *IEEE Transactions on Robotics and Automation (Volume:7 , Issue: 1)*, pp. 114 - 122, Feb 1991.

[49] Y. H. Kim and F. L. Lewis, "Hamilton-Jacobi-Bellman optimal design of CMAC neural network controller for robot manipulators," in *International Conference on Systems, Man, and Cybernetics*, Orlando, FL, 12-15 Oct 1997, pp. 1361 - 1366 vol.2.

[50] Giovanni Buizza Avanzini, Paolo Rocco, and Andrea Maria Zanchettin, "Constraint-based Model Predictive Control for holonomic mobile manipulators," in *International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Sept. 28 2015-Oct. 2 2015, pp. 1473 - 1479.

[51] R. Fierro and Frank L. Lewis, "Control of a nonholonomic mobile robot using neural networks," in *International Symposium on Intelligent Control*, Monterey, CA, 27-29 Aug 1995, pp. 415 - 421.

[52] (2016, July) Kuka youBot Store. [Online]. http://www.youbot-store.com/

[53] Ioan Alexandru Sucan, "Task and motion planning for mobile manipulators," Rice University, Texas, Dissertation/Thesis 3521301, 2011.

[54] Thien Nguyen Duc, Annalisa Terracina, and Massimo Mecella, "Robotic Teaching Assistant for "Tower of Hanoi" Problem," in *Academic Conferences International Limited*, Reading, 2014, pp. 723-731.

[55] Lluis Ribas-Xirgo, "Emerging Technology and Factory Automation," in *Emerging Technology and Factory Automation*, Barcelona, Spain, 2014, pp. 1-4.

[56] Matthew Luciw, "Reinforcement and shaping in learning action sequences with neural dynamics," in *The Institute of Electrical and Electronics Engineers*, Genoa, Italy, 2014, pp. 48-55.

[57] Lorenzo Peppoloni, Filippo Brizzi, and Carlo Alberto Avizzano, "Immersive ROS-integrated framework for robot teleoperation," in *Symposium on 3D User Interfaces*, Arles, France, 2015, pp. 177-178.

[58] Joanna Ratajczak, "Design of inverse kinematics algorithms: extended Jacobian approximation of the dynamically consistent Jacobian inverse," *Archives of Control Sciences*, vol. 25, no. 1, pp. 35-50, 2015.

[59] J M Hollerbach, "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," Massachusetts Inst. of Tech, Cambridge, Report AD-A078067, 1979.

[60] Burhanettin Durmus, Hasan Temurtas, Nejat Yumusak, and Fevzullah Temurtas, "A study on industrial robotic manipulator model using model based predictive controls," *Journal of Intelligent Manufacturing*, vol. 20, no. 2, pp. 233-241, Apr 2009.

[61] W M. Silver, "Representation of Angular Velocity and Its Effect on the Efficiency of Manipulator Dynamics Computation," *National Technical Information Service*, p. 30 pp, 1981.

[62] Oguz Yetkina, Kristi Wallacea, Joseph D. Sanford, and Dan Popa, "Control of a Powered Prosthetic Device via a Pinch Gesture Interface," in *Next-Generation Robotics II; and Machine Intelligence and Bio-inspired Computation: Theory and Applications IX*, Baltimore, Maryland, United States, 2015, p. Volume 9494.

Biographical Information

Drew started at UTA in 2004 to pursue interests in both mechanical and electrical engineering fields. After just two years and deep involvement with the local Formula student team, he committed to mechanical engineering. Serving as the team's suspension engineer for the remaining two years of his bachelor degree, he graduated in 2008 with his bachelors.

After two years of working, a passion for race cars would lead him to pursue a specialized degree of Automotive Motorsports in England. While the experience was uniquely satisfying, the racing career never emerged and eventually Drew would return home and in 2014 begin a Master's degree at UTA.

Reviving an old interest in electrical science, Drew pursued this most recent degree with focus on its application to dynamic systems. Almost immediately after resuming his studies, he attracted the recruiting pipeline of Bell Helicopter and began what would be a long onboarding process that provided convenient opportunity to find a fulfilling career while completing this latest degree.

Now he looks forward to having time to idle, be with friends and family, achieve financial independence, and perhaps someday, flying a personal aircraft.