DESIGN, MODELING, CONTROL, AND SIMULATION OF A
REDUNDANT, HOLONOMIC ROBOCUP GOALIE

A Thesis Presented to

The Faculty of the

Fritz J. and Dolores H. Russ
College of Engineering and Technology
Ohio University

In Partial Fulfillment

of the Requirement for the Degree

Master of Science

by

Lance J. Wilson
March, 2003

OHIO UNIVERSITY LIBRARY
<table>
<thead>
<tr>
<th>TABLE OF CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables........................................................................................................vi</td>
</tr>
<tr>
<td>List of Figures........................................................................................................vii</td>
</tr>
<tr>
<td>Chapter 1: Introduction.............................................................................................1</td>
</tr>
<tr>
<td>1.1 Background........................................................................................................1</td>
</tr>
<tr>
<td>1.2 Holonomic v. Nonholonomic Constraints.........................................................3</td>
</tr>
<tr>
<td>1.2.1 Conventional Two Wheel Robot.................................................................4</td>
</tr>
<tr>
<td>1.2.2 Omni-Directional Four Wheel Robot..........................................................5</td>
</tr>
<tr>
<td>1.2.3 Parallel Parking Analogy............................................................................6</td>
</tr>
<tr>
<td>1.3 Literature Review..............................................................................................7</td>
</tr>
<tr>
<td>Chapter 2: Mechanical Design ................................................................................13</td>
</tr>
<tr>
<td>2.1 Constraints.......................................................................................................13</td>
</tr>
<tr>
<td>2.2 Performance Specifications.............................................................................14</td>
</tr>
<tr>
<td>2.3 Layout................................................................................................................15</td>
</tr>
<tr>
<td>2.4 Omni-Directional Drive..................................................................................16</td>
</tr>
<tr>
<td>2.5 Kicking/Dribbling/Capturing Mechanism.......................................................17</td>
</tr>
<tr>
<td>Chapter 3: Modeling...............................................................................................19</td>
</tr>
<tr>
<td>3.1 Coordinate Transformation.................................................................20</td>
</tr>
</tbody>
</table>
3.2 Kinematics ................................................................. 25
3.3 Wheel Dynamics .......................................................... 27
3.4 Robot Frame Equations of Motion ....................................... 28
3.5 Joint Frame Equations of Motion ......................................... 30
3.6 Global Frame Equations of Motion ....................................... 31
3.7 Actuator Dynamics .......................................................... 33
3.8 Joint Frame Robot Model .................................................. 37
3.9 Robot Frame Model ......................................................... 38

Chapter 4: Control ................................................................. 39
4.1 Redundancy ................................................................. 39
4.2 Nonlinear Control ............................................................ 43
4.3 Multiple Input, Multiple Output ........................................... 43
4.4 Computed Torque Control .................................................. 44
  4.4.1 Local Frame Control ................................................... 45
  4.4.2 Global Frame Control ................................................... 49
  4.4.3 Joint Frame Control .................................................... 51
4.5 Joint-Actuator Control ....................................................... 54
4.6 Vision Feedback Control with Joint Actuator Computed Torque ........ 56
4.7 Practical Considerations .................................................... 57
LIST OF TABLES

Table 5.1: System Parameters..................................................................................60
LIST OF FIGURES

Figure 1.1: Conventional Two Wheel Mobile Robot ................................................. 4
Figure 1.2: Omni-Directional Four Wheel Mobile Robot ........................................ 5
Figure 1.3: Nonholonomic Parallel Parking ............................................................. 6
Figure 1.4: Holonomic Parallel Parking ................................................................. 6
Figure 1.5: Orthogonal Wheel Assembly ............................................................... 7
Figure 1.6: One-Wheel Caster and Dual-Wheel Caster .......................................... 8
Figure 1.7: Resolved Acceleration Control System ............................................... 9
Figure 2.1: Top View of OU Goalie (circular chassis) .............................................. 14
Figure 2.2: OU Goalie ......................................................................................... 16
Figure 2.3: Omniwheels ..................................................................................... 17
Figure 2.4: Close-up of Kicker Assembly .............................................................. 18
Figure 3.1: Free-Body Diagram ........................................................................... 19
Figure 3.2: Rotation Mapping ............................................................................. 21
Figure 3.3: Wheel FBD ..................................................................................... 28
Figure 3.4: DC Motor ....................................................................................... 33
Figure 3.5: Armature Circuit and Rotor FBD ....................................................... 34
Figure 4.1: Model-Based Control Scheme ......................................................... 49
Figure 4.2: Model-Based Control Scheme ........................................................... 53
Figure 4.3: Model-Based Joint Control Scheme ............................................... 56
Figure 4.4: Vision Feedback Control Scheme ..................................................... 57
Figure 5.1: Inverse Kinematics Simulation Results..........................................61

Figure 5.2: Simulink Block Diagram, Global/Local Computed Torque Control
  Simulation........................................................................................................62

Figure 5.3: Local Frame Simulation Results......................................................62

Figure 5.4: Global Frame Simulation Results....................................................64

Figure 5.5: Simulink Block Diagram, Joint/Joint-Actuator Computed Torque Control
  Simulation........................................................................................................65

Figure 5.6: Joint Frame Simulation Results......................................................65

Figure 5.7: Joint-Actuator Frame Simulation Results........................................66

Figure 5.8: Vision Feedback Control with Joint-Actuator Computed Torque........67

Figure 5.9: Mobile Robot Position Results.......................................................68

Figure 5.10: Input Current................................................................................69

Figure 5.11: 10% Increase in Payload.............................................................70

Figure 5.12: 20% Increase in Payload.............................................................70
CHAPTER 1: INTRODUCTION

Technology is advancing and robots are being used more and more in everyday life. Robots are being developed for many different and specific tasks in many different professional disciplines; including manufacturing, medical, shipping, hazardous material handling, and hazardous environment exploration. Now as computation power increases and wireless connections are readily available, the human race turns to robots once again with a new task, professional soccer. This chapter will discuss the background and the motivation for this thesis, the difference between holonomic and nonholonomic constraints, and a literature review.

1.1 Background

The RoboCup is an international competition of autonomous mobile robots that compete in soccer. The purpose of the competition is to improve on existing mobile robot technology and intelligence systems. The Ohio University RoboCup team consists of four players and one goalie. The player and the goalie each have separate mechanical, electrical and software designs to meet different performance requirements. The player is designed to give good performance while moving in any direction, while the goalie is designed to give more of a side-to-side motion, without complex movements, to better protect the goal. It is also important to note that the goalie could
be equipped with a local vision system to improve on response time. This thesis will focus on the design, modeling, control, and simulation of the goalie robot.

A mobile robot's mission can be described by a sequence of poses (position and orientation). At each pose, the robot is at rest and may perform a task. In moving from one pose to the next, the sensors on the robot may detect unexpected obstacles. If unexpected obstacles block its path, the mobile robot must find a new clear path to its next objective.

Omni-directional drive is the ability to move in any direction (0°-360°) at any given orientation. Using this type of drive system enables a mobile robot to move in a given direction while being able to rotate during linear travel. Because of this, omni-directional robots can save time by eliminating the need to rotate before translating from point A to point B. Omni-directional drive gives the goalie robot the ability to always face the ball while being able to move in any direction. In addition, a drive system such as this provides a simpler inverse kinematics solution for path planning.

Most conventional studies on omni-directional vehicles have focused on the development of mechanisms, or only on the analysis of kinematics. Hence, there are few studies on the development of dynamic models and highly accurate control systems. For these reasons, many research groups are now studying omni-directional holonomic mobile robots and vehicles [1-12].
1.2 Holonomic vs. Nonholonomic Constraints

To understand the constraints of a system, it is first and foremost important to understand generalized coordinates. The configuration of any given system may be expressed in terms of various sets of coordinates. For this reason, no specific set of coordinates is unique to the analysis of a given mechanical system. Thus, many coordinate systems are possible. Additionally, the number of coordinates used to describe any one system can vary widely, so long as there are at least as many coordinates as there are degrees of freedom. Because of this, it can be said that any set of coordinates which serve to specify the configuration of a system, are examples of generalized coordinates. For systems in motion, these generalized coordinates can vary in time and are treated as algebraic variables.

Suppose the number of generalized coordinates, \( q_j \), is \( m \) and the number of degrees of freedom is \( n \). If \( m=n \), all generalized coordinates are independent. If \( m>n \), there are \( m-n \) constraint equations. The constraint equations take the form:

\[
f_i = (q_1, q_2, ..., q_m, t) = 0 \quad i=1, 2, ..., m-n
\]  

(1.1)
In velocity form:

\[ \dot{f}_i = \sum_{j=1}^{m} a_{ij} \dot{q}_j + b_i = 0 \]

\[ a_{ij} = \frac{\partial f_i}{\partial q_i} \] (1.2).

For a given velocity constraint, Equation (1.2), if \( a_{ij} \) is integrable such that \( f_i = 0 \) can be found, it is called a holonomic constraint. If \( a_{ij} \) is not integrable, it is called a nonholonomic constraint.

**1.2.1 Conventional Two Wheel Robot**

We can describe a conventional two wheel mobile robot shown in Figure 1.1 with three generalized coordinates, \( \{X, Y, \Phi\} \).

![Conventional Two Wheel Mobile Robot](image-url)
By inspection, the velocity in the normal direction to wheels $A$ and $B$ is zero, which gives us the velocity constraint equation:

$$(V_c)_{AB} = \dot{X} \sin \Phi - \dot{Y} \cos \Phi = 0 \quad (1.3)$$

This constraint equation is not integratable; therefore the two wheel conventional mobile robot is nonholonomic.

### 1.2.2 Omni-Directional Four Wheel Robot

*Figure 1.2* shows an omni-directional mobile robot equipped with special wheels described in the next section (*Literature Review*).

![Omni-Directional Four Wheel Mobile Robot](image)

*Figure 1.2: Omni-Directional Four Wheel Mobile Robot*

This type of design eliminates the nonholonomic constraint by allowing motion in all directions. Therefore, it is a holonomic mobile robot.
1.2.3 Parallel Parking Analogy

To better understand this concept, think of parallel parking a car. In order to get the car into the desired parking space a series of complex maneuvers must be performed, refer to Figure 1.3.

![Diagram of nonholonomic parallel parking]

\begin{itemize}
  \item \textbf{(a)} Starting Position
  \item \textbf{(b)} Rotation and Translation
  \item \textbf{(c)} Final Position
\end{itemize}

\textit{Figure 1.3: Nonholonomic Parallel Parking}

This type of maneuver would be considered nonholonomic due to the rotational constraints that would be placed on the dynamic system.

A simpler approach to parallel park a car would be to directly translate the car into the parking space without any rotation. This can be done with a holonomic omni-directional car, refer to Figure 1.4.

![Diagram of holonomic parallel parking]

\begin{itemize}
  \item \textbf{(a)} Initial Position
  \item \textbf{(b)} Final Position
\end{itemize}

\textit{Figure 1.4: Holonomic Parallel Parking}
Since the system is holonomic the car can directly translate into the parking space without the need to rotate.

1.3 Literature Review

Pin [1] proposed a design to produce fully omni-directional and holonomic platforms with unconstrained, simultaneous, and independently controlled rotational and translational motions. The wheel design is based on an orthogonal wheel assembly similar to Figure 1.5.

![Orthogonal Wheel Assembly](image)

*Figure 1.5: Orthogonal Wheel Assembly*
A prototype has been constructed and its control system has been described. The proposed control system was implemented using a simple proportional gain controller. Han [2] describes an omni-directional mobile robot platform using active dual wheel caster mechanisms, shown in Figure 1.6.

![Figure 1.6: One-Wheel Caster and Dual-Wheel Caster](image)

The kinematic model of the wheel assembly was derived and used to construct a feedforward control system. A holonomic omni-directional mobile robot was then considered, the corresponding kinematic model was derived, and a resolved velocity control system was constructed by applying the model.

Shim [3] presents a miniature omni-directional mobile robot, and its kinematic and dynamics models are presented. A conventional two wheeled mobile robot serves as the base for the translational motion. A third motor controls angular motion of the body as the base turns according to the translational motion. A circular surface connector is devised to transfer signals from the base to the body without constraining angular motion of the body. The control loop of this robot is twofold. One is a low-level joint
independent proportional-integral-derivative (PID) control of the wheels and body. The other is an outer loop for angular speed control of the body.

In Wantanabe’s work [4], a dynamic model of an omni-directional mobile robot is introduced in which it is assumed that the platform is based on three orthogonal wheel assemblies similar to Figure 1.6. A resolved acceleration control system is designed using such a dynamic model, see Figure 1.7.

\[ R^{-1} \] denotes the estimated inverse dynamics; this is actually canceling out the nonlinear portion of the dynamic model. Practical experiments are presented to illustrate the full omni-directionality of the platform with decoupled rotational and translational motions. Reister [5] and Moore [6] describe wheel control systems for omni-directional mobile robots that could encounter unexpected obstacles and be required to quickly replan its
path. Because the RoboCup robots will be used in a changing environment this topic is very important. Though this topic is not discussed in this thesis, it is of great concern for the ultimate result of the RoboCup team.

An index for measuring the compatibility of pose with respect to tasks, where accurate control of small velocity and force is required, and where exertion of large velocity and force is required is presented in Chiu's [7] work. The optimization of this index presents an effective way of utilizing redundancy.

In Kalmar-Nagy's work [8], an innovative method of generating near-optimal trajectories for a robot with omni-directional capabilities is proposed, taking into account the dynamics of the actuators and the system. The relaxation of optimality results in immense computational savings, critical in dynamic environments. A decoupling strategy for each of the three degrees of freedom of the vehicle is presented, along with a method for coordinating the degrees of freedom.

In many robotic applications, various performance aspects should be considered simultaneously for a given task. A redundant robot, having more actuators than degrees of freedom, has the advantage of having infinite inverse solutions in order to optimize the robots motion, singularity avoidance, obstacle avoidance, or joint limit, etc. Of each subtask one can optimize, the minimization of joint torques; i.e. determining a force vector that maximizes the work done by the bounded joint torques; and the minimization of the inverse kinematics; i.e. computing a joint velocity vector whose maximum absolute value is minimum among all joint velocity vectors yielding the desired velocity.
When considering energy consumption, the least-squares pseudo-inverse does minimize the amount of total energy the robot uses. This solution will not necessarily minimize the magnitudes of the individual joint velocities; i.e. there could be an unequal distribution of the energy resulting in a relatively high velocity for a particular joint.

Deo [9] computes an inverse kinematic solution of minimum infinity-norm for redundant manipulators. He states that minimizing the infinity-norm can be intuitively thought of as trying to distribute the load equally, so that no one joint has to have a very high velocity with respect to the others. A technique is presented that enables more direct monitoring and control of magnitudes of the individual joint velocities than does the minimization of the sum of the squares of the components.

Gravagne [10] states that generally the minimum infinity-norm solution cannot be expressed in closed form, requiring the use of an algorithm to iteratively refine an initial guess before reaching the desired solution. Gravagne shows the formulation of a new "infinity-inverse", and then uses the new inverse to explore uniqueness and continuity of a least infinity-norm solution. The new inverse is compared with the well-known pseudo-inverse solution.

The minimization of the joint torques based on the infinity norm is proposed for the dynamic control of a kinematically redundant manipulator in Shim’s work [11]. To obtain the minimum infinity norm torque solution, Shim devises a new algorithm that uses an acceleration polyhedron representing the end effector’s acceleration capability.

Although the idea of an omni-directional holonomic mobile robot is not new, the design of new platforms and highly effective dynamic control schemes is becoming more
desirable as performance requirements increase. Much of the previous work on this subject focused on the kinematic analysis and simple PID control algorithms. This thesis will describe in detail the complete kinematic and dynamic analysis of the mobile robot and will also propose a dynamic control algorithm to better control the mobile robots path.

The idea of redundancy is a concept that is utilized in many industrial robotic manipulators to better perform a task within the working environment. However, the topic of a redundant mobile robot is a new one. These concepts used to describe and optimize redundant manipulators will be used in describing and controlling the mobile robot system.
CHAPTER 2: MECHANICAL DESIGN

The goalie has two main functions. The major function is to have the ability to quickly move side-to-side to guard the goal. This has been accomplished by having the wheels aligned horizontally for quick movement. The other function is the kicking of the ball. The current design not only allows for a strong kick but also the ability of catching the ball under control.

This chapter will discuss the mechanical design of the goalie robot. The constraints and performance specifications placed on the design will be presented. The layout of the robot platform, the omnidirectional drive, and the kicking/dribbling/capturing mechanism will be explained.

2.1 Constraints

In order to compete in the Robocup tournament, robots have to meet specified size constraints. The goalie robot can be no taller than 22.5 cm if local vision is to be used, and must fit within a 18 cm diameter cylinder. Any part that may extend outward on the robot must be fully extended when placed inside the 18 cm cylinder. These size constraints greatly limit the arrangement of drive and kicking systems. The OU goalie’s kicking and drive systems are rectangular in shape and designing a layout to meet the robot size constraints results in wasted space around these systems. To utilize these irregularly shaped spaces, one has to be creative in making them useful for housing
various system components. Currently a circular chassis is being used to explore various drive/kicking system layouts that will make use of these valuable spaces (Figure 2.1).

Figure 2.1: Top View of OU Goalie (circular chassis)

2.2 Performance Specifications

Through testing various radio-controlled toy cars, experiments were performed to better visualize what kind of performance is desired for the goalie robot. High value was placed on making the robot “zippy”, which means it can move or change direction quickly but doesn’t have a high maximum velocity. The kicking mechanism used on the goalie needed to be powerful enough to propel the ball at least half the length of the playing field. This distance would clear the ball out of our defensive zone giving the defensemen time to regroup. The kicker should not be able to kick the ball so hard that it is able to roll over the boundary fence at the opposing end of the field, which would
result in a penalty. Controlling the kicking power is accomplished by using a variable speed servomotor on the kicking mechanism.

One advantage of our goalie design is the ability to capture the ball and completely enclose it. Enclosing the ball and hiding it from opposing robots operating on the global vision system can lead to confusion if they cannot locate the ball. However if the goalie is to capture the ball, teammates can be notified of the ball’s location.

2.3 Layout

The goalie layout consists of the drive system and kicking mechanism (Figure 2.2). To maintain traction on the drive wheels, the robot’s center of mass should be located on the central axes of the drive wheels. Four wheels are being used to drive the robot omni-directionally. Keeping the goalie’s center of mass near these axes becomes difficult once the kicker mechanism is added to the system. The kicker mechanism shifts the center of mass toward the geometric center of the robot, which can cause the rear wheel to slip. Shifting the center of mass to compensate for the weight of the kicker can be accomplished through shifting battery positions toward the rear of the robot. Additional ballasts can be added to maintain drive wheel traction, but care must be exercised to not impede robot performance through weight addition.
There are 6 motors utilized in the robot assembly. Four motors are placed in the wheel assemblies to drive the wheels. One is in the kicker assembly and rotates the drive bar. There is also a servomotor associated with the kicker assembly.

2.4 Omni-Directional Drive

The Kornylak Corporation manufactures the Omniwheel that appears in Figure 2.3(a). To minimize wheel width, the original Omniwheel is modified by removing material from the center and outer faces. Removal of material also allows room to insert a fabricated hub to be used for gear mounting. Figure 2.3(b) shows the modified wheel mounted to an aluminum hub.
2.5 Kicking/Dribbling/Capturing Mechanism

The kicking mechanism, Figure 2.4, is structured around one basic principle, capturing the ball. It has the ability to rotate around a pivot point with a rocking motion controlled by a servomotor. The servomotor is located on the goalie base and thus is stationary with respect to the kicker. During the game, the goalie will be located near the goal with its drive bar rotating backwards. The drive bar will be encased in rubber tubing. Initially the kicking mechanism will be at its highest point in the arc. When the ball gets near the kicker the drive bar will capture the ball via backspin. At this time the ball will be rotating between the drive bar and the idler bar located underneath the robot. Being underneath the robot will cause the ball to be hidden from the global camera. The drive bar will then reverse direction and cause the ball to spin in the opposite direction between the two bars. After the ball gains a sufficient amount of forward rotation the kicking mechanism will be rocked back by the servomotor. This will allow the ball to be released while having topspin. After releasing the ball, the
kicking mechanism will be brought up to its highest position in the rotating arc by the servomotor, to wait for the next ball.

*Figure 2.4: Close-up of Kicker Assembly*
CHAPTER 3: MODELING

This chapter will discuss the kinematic, dynamic, and electro-mechanical equations of motion. Equations of motion will be derived with respect to the global frame so the robot can be controlled with respect to a global vision system; robot frame so the robot can be controlled with a local vision system; and the joint frame so the robot can be controlled through encoder feedback. The dynamic equations for the four-wheeled Ohio University RoboCup goalie are derived from the model shown in Figure 3.1, in a manner similar to [4].

![Free-Body Diagram](image-url)

*Figure 3.1: Free-Body Diagram*
3.1 Coordinate Transformation

It is assumed the mobile robot is moving on a horizontal playing field. As it is shown, the absolute or global coordinate frame $\{^w X, \, ^w Y\}$ is fixed on the plane and the moving coordinate frame $\{^m x, \, ^m y\}$ is attached to the mass center of the mobile robot. $L_i$ is the distance from the mass-center to the center of the corresponding wheel. $\phi$ is the orientation of the mobile robot with respect to the absolute frame. As previously stated this robot is omni-directional, giving us no nonholonomic constraint equations. Also, it is important to point out that the OU RoboCup goalie has three degrees of freedom, but has four actuators meaning that the system is redundantly actuated.

First, we define the position and wrench (force and torque) vectors of the mass-center for the OU RoboCup Goalie in the absolute coordinate frame as:

\[
^w S = \begin{bmatrix} \, ^w X \, ^w Y \, \phi \end{bmatrix}^T
\]  \hspace{1cm} (3.1)

\[
^w F = \begin{bmatrix} F_x \, F_y \, T_G \end{bmatrix}^T
\]  \hspace{1cm} (3.2)

Where $T_G$ is the torque applied to the robot.

By looking at the rotation mapping shown in Figure 3.2, we can derive the coordinate rotation matrix.
In order to describe the orientation of the mobile robot, we will attach a coordinate system to the mobile robot and then give a description of this coordinate system relative to the reference system. In Figure 3.2, the coordinate system \( \{"x, "y\} \) has been attached to \( \{"X, "Y\} \) by a rotation of angle \( \phi \). A description of \( \{"x, "y\} \) relative to \( \{"X, "Y\} \) will now give the orientation of the body. Positions of points are described with vectors, and orientations of bodies are described with an attached coordinate system, shown in Figure 3.2. A set of three vectors may be used to specify an orientation as follows:

\[
{}^wP_x = {}^mP_x \cos \phi - {}^mP_y \sin \phi
\]  

(3.3)
Then the coordinate rotation matrix \( \mathcal{R} \) giving the absolute frame with respect to the orientation of the moving frame is:

\[
\begin{bmatrix}
\cos \phi & -\sin \phi & 0 \\
\sin \phi & \cos \phi & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

A special property of this matrix is that its inverse is equal to its transpose, or:

\[
\mathcal{R}^{-1} = \mathcal{R}^T
\]

Therefore, the rotation matrix \( \mathcal{R} \) is an orthogonal matrix.

The following two equations relate Cartesian pose and wrench for the moving and inertial frames:

\[
\begin{align*}
\mathcal{S} &= \mathcal{R} \mathcal{S} \\
\mathcal{F} &= \mathcal{R} \mathcal{F}
\end{align*}
\]
The pose and wrench vectors of the goalie in terms of the moving coordinates are:

\[
{^m}s = \begin{bmatrix} x^m & y^m & \phi \end{bmatrix}^T
\] (3.10),

\[
{^m}f = \begin{bmatrix} f_x & f_y & \tau_G \end{bmatrix}^T
\] (3.11).

From Newton’s 2\textsuperscript{nd} Law:

\[
M \ddot{^m}s = \ddot{^m}F
\] (3.12)

Where \(M\) is a positive-definite diagonal matrix with the mobile robots mass \(M\) and inertia \(I_m\) on the diagonal. We can then transform Equation (3.12) in terms of the moving coordinates using Equations (3.6)-(3.9).

\[
M = \begin{bmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & I_m \end{bmatrix}
\] (3.13)

\[
M \ddot{^m}s = M \left( \dot{^m}R^m s + \ddot{^m}R^m s \right) = \ddot{^m}R^m f
\] (3.14)
However, since the rotation matrix has the orthogonality property we can write this transformation as follows:

\[
M \begin{pmatrix}
\frac{w}{m} R^{-1} & \frac{w}{m} R^T \\
\frac{m}{w} s & \frac{m}{w} s + \frac{m}{w} \ddot{s}
\end{pmatrix} = f
\]  
(3.15).

Simplifying Equation (3.16), the dynamic equations of the robot are:

\[
M \begin{pmatrix}
\frac{m}{w} \dddot{x} - \frac{m}{w} \dddot{y} \\
\frac{m}{w} \dddot{x} + \frac{m}{w} \dddot{y} + \frac{m}{w} \dddot{\phi}
\end{pmatrix} = f
\]  
(3.16).

\[
M \begin{pmatrix}
\frac{m}{w} \dddot{x} - m \dddot{y} + m \dddot{\phi} \\
\frac{m}{w} \dddot{x} + m \dddot{y} - m \dddot{\phi}
\end{pmatrix} = f
\]  
(3.17)

\[
M \begin{pmatrix}
\frac{m}{w} \dddot{x} - m \dddot{y} + m \dddot{\phi} \\
\frac{m}{w} \dddot{x} + m \dddot{y} - m \dddot{\phi}
\end{pmatrix} = f
\]  
(3.18)

\[I_m \dddot{\phi} = \tau_G\]  
(3.19).

Where \(I_m\) is the mass moment of inertia for the mobile robot with respect to the mass-center.
3.2 Kinematics

To compute the relationship between the robot’s Cartesian coordinates and the wheel angles, the principle of virtual work is used. Summing the forces and moments, with respect to the mass center, from the free-body diagram shown in Figure 3.1 gives:

\[
\sum f_x = -T_2 + T_4 \\
\sum f_y = T_1 - T_3 \\
\sum \tau_G = T_1 L_1 + T_2 L_2 + T_3 L_3 + T_4 L_4
\]

Which can be written in the form:

\[
\begin{bmatrix}
{f_x} \\
{f_y} \\
{\tau_G}
\end{bmatrix} =
\begin{bmatrix}
0 & -1 & 0 & 1 \\
1 & 0 & -1 & 0 \\
L_1 & L_2 & L_3 & L_4
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4
\end{bmatrix}
\]

or

\[ f = QT \]

Where \( Q \) is the Jacobian matrix based upon the system geometry, and \( T_i \) is the traction force from each wheel. Since the virtual work done is the same for the wheel and Cartesian coordinates, the relationship can be expressed as:
By applying Equation (3.23), the virtual displacement relationship between the wheel and the Cartesian coordinates becomes:

\[ Q^T \delta^m s = \delta q \]  

(3.25).

Where \( \delta q \) is the velocity of the wheel rolling contact point assuming that there is no slip in the wheel spin direction; i.e. \( \delta q = \{r \delta q_1, r \delta q_2, r \delta q_3, r \delta q_4\}^T \). Thus, we can obtain the inverse kinematics solution of the OU RoboCup goalie from Equations (3.23) and (3.25). Given the Cartesian velocity, we may find the wheel angular velocities:

\[ \dot{q} = Q^T \delta^m s \]

or

\[
\begin{align*}
\omega_1 &= \frac{1}{r} \begin{bmatrix} 0 & 1 & L_1 \end{bmatrix} \begin{bmatrix} \dot{x}_m^* \\ \dot{y}_m^* \\ \phi \end{bmatrix} \\
\omega_2 &= \frac{1}{r} \begin{bmatrix} -1 & 0 & L_2 \end{bmatrix} \\
\omega_3 &= \frac{1}{r} \begin{bmatrix} 0 & -1 & L_3 \end{bmatrix} \\
\omega_4 &= \frac{1}{r} \begin{bmatrix} 0 & 1 & L_4 \end{bmatrix}
\end{align*}
\]  

(3.26).

Where \( r \) is the wheel radius and \( \omega_i \) is the wheel rotational rate.
Now we can transform the robot frame kinematic relationship back to the world frame
by the transformation shown by Equation (3.8); and since the rotation matrix is
orthogonal we have:

\[ q = Q^T_w R^T_w s \]

or

\[
\begin{bmatrix}
\omega_1 \\
\omega_2 \\
\omega_3 \\
\omega_4
\end{bmatrix} =
\begin{bmatrix}
0 & 1 & L_1 \\
-1 & 0 & L_2 \\
0 & -1 & L_3 \\
1 & 0 & L_4
\end{bmatrix}
\begin{bmatrix}
\cos \phi & \sin \phi & 0 \\
-\sin \phi & \cos \phi & 0 \\
-\cos \phi & -\sin \phi & L_2 \\
\sin \phi & -\cos \phi & L_3
\end{bmatrix}
\begin{bmatrix}
X_w \\
Y_w \\
\phi
\end{bmatrix} =
\begin{bmatrix}
-\sin \phi & \cos \phi & L_1 \\
-\cos \phi & -\sin \phi & L_2 \\
\sin \phi & -\cos \phi & L_3 \\
\cos \phi & \sin \phi & L_4
\end{bmatrix}
\begin{bmatrix}
\dot{X}_w \\
\dot{Y}_w \\
\dot{\phi}
\end{bmatrix}
\]

and

\[
\begin{bmatrix}
\omega_1 \\
\omega_2 \\
\omega_3 \\
\omega_4
\end{bmatrix} =
\begin{bmatrix}
-\sin \phi & \cos \phi & L_1 \\
-\cos \phi & -\sin \phi & L_2 \\
\sin \phi & -\cos \phi & L_3 \\
\cos \phi & \sin \phi & L_4
\end{bmatrix}
\begin{bmatrix}
\dot{X}_w \\
\dot{Y}_w \\
\phi
\end{bmatrix} +
\frac{1}{r}
\begin{bmatrix}
-\phi \cos \phi & -\phi \sin \phi & 0 \\
\phi \sin \phi & -\phi \cos \phi & 0 \\
\phi \cos \phi & \phi \sin \phi & 0 \\
-\phi \sin \phi & \phi \cos \phi & 0
\end{bmatrix}
\begin{bmatrix}
\dot{X}_w \\
\dot{Y}_w \\
\phi
\end{bmatrix}
\]

(3.28).

### 3.3 Wheel Dynamics

The wheel dynamics for each assembly can be derived from the following free-body
diagram, Figure 3.3.
By summing the moments about the mass center of the wheel we obtain the dynamic model for the wheel assemblies:

\[ I_w \ddot{\omega}_i + c \omega_i = k u_i - r T_i, \quad i = 1, 2, 3, 4 \]  

(3.29).

Where \( I_w \) is the mass moment of inertia for the Omniwheel, \( c \) is the viscous friction factor of the Omniwheel, \( k \) is the gear ratio, and \( u_i \) is the driving input torque.

### 3.4 Robot Frame Equations of Motion

Then combining Equations (3.17)-(3.23), (3.26), and (3.29) we have the following equations of motion written in matrix-vector form:

\[
P \{m \dddot{s}\} + N_s = B \{U\} 
\]  

(3.30).
Where:

\[
P = \begin{bmatrix}
\frac{2I_w}{r^2} + M & 0 & \frac{I_w}{r^2}(L_4 - L_2) \\
0 & \frac{2I_w}{r^2} + M & \frac{I_w}{r^2}(L_1 - L_3) \\
\frac{I_w}{r^2}(L_4 - L_2) & \frac{I_w}{r^2}(L_1 - L_3) & \frac{I_w}{r^2}(L_2^2 + L_3^2 + L_4^4) + I_m
\end{bmatrix}
\]

\[
N_s = \begin{bmatrix}
-M \ddot{y}_m + \frac{2c}{r^2} \dot{x}_m + \frac{c}{r^2} (L_4 - L_2) \ddot{\phi} \\
M \ddot{x}_m + \frac{2c}{r^2} \dot{y}_m + \frac{c}{r^2} (L_1 - L_3) \ddot{\phi} \\
\frac{c}{r^2} (L_4 - L_2) \dddot{x}_m + \frac{c}{r^2} (L_1 - L_3) \dddot{y}_m + \frac{c}{r^2} (L_2^2 + L_3^2 + L_4^4) \dddot{\phi}
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0 & -\frac{k}{r} & 0 & \frac{k}{r} \\
\frac{k}{r} & 0 & -\frac{k}{r} & 0 \\
\frac{k}{r} & \frac{k}{r} & \frac{k}{r} & \frac{k}{r} \\
\frac{k}{r} & \frac{k}{r} & \frac{k}{r} & \frac{k}{r}
\end{bmatrix}
\]

After manipulation, we have three coupled multiple input/multiple output (MIMO) nonlinear differential equations of motion with wheel torques as the inputs, and the robot's Cartesian position and orientation with respect to the local vision (robot frame) of the robot as the outputs. Vector \( N_s \) is a function of the robot's Cartesian velocity.
3.5 Joint Frame Equations of Motion

Now, a transformation of the equations of motion from the robot frame to the joint angle frame can take place. To accomplish this we solve Equation (3.26) for the forward kinematic solution as follows:

\[
\ddot{m} = (Q^T)^\dagger q
\]  

(3.31)

and since the Jacobian matrix is constant we get:

\[
\dddot{m} = (Q^T) q
\]  

(3.32).

Where \((Q^T)^\dagger\) is the pseudo-inverse of the Jacobian matrix, based upon the least-square error, and defined as:

\[
(Q^T)^\dagger = (QQ^T)^{-1} Q
\]  

(3.33).

\[
Q^r = \left( \frac{1}{L_1 + L_2 + L_3 + L_4 + 2L_1L_2 + 2L_1L_3 + 2L_2L_4} \right) \begin{bmatrix}
\frac{1}{2} (L_2 - L_4)(L_1 + L_3) \\
\frac{1}{2} (L_1 + L_3 + 2L_1L_4 + 2L_1L_3 + 2L_2L_4) \\
\frac{1}{2} (L_1 + L_2 + L_3 + L_4 + 2L_1L_2 + 2L_1L_3 + 2L_2L_4) \\
(L_1 + L_3) \\
(L_2 + L_4)
\end{bmatrix}
\]
Thus, transforming Equation (3.30) with Equations (3.31)-(3.33), we now have the Joint angle equations of motion written in matrix-vector form:

\[
P\left(Q^\top\right)\dddot{q} + N_q = B U
\]  

Vector \(N_q\) is a function of the joint velocities. After this transformation, we still have three coupled nonlinear differential equations of motion. The inputs are still the wheel torques; the outputs have become the wheel angles, which can be interpreted by the encoders. Although a detailed description of vector \(N_q\) does not appear in this section, a simplified version is shown in the next chapter, Control.

### 3.6 Global Frame Equations of Motion

Now we combine Equations (3.9), (3.12), (3.23), (3.27), and (3.29) to obtain the following equations of motion with respect to the global frame:

\[
P_g \dddot{w} + N_g = B U
\]  

(3.35)
Where:

\[
\begin{align*}
\mathbf{N}_x &= \begin{bmatrix}
\left( \frac{2c}{r^2} \cos \phi - \frac{2L_2}{r^2} \dot{\phi} \sin \phi \right) \dot{X}_x + \left( \frac{2c}{r^2} \sin \phi + \frac{2L_2}{r^2} \dot{\phi} \cos \phi \right) \dot{Y}_x + \frac{c}{r^2} (L_4 - L_2) \dot{\phi} \\
\left( \frac{2c}{r^2} \sin \phi - \frac{2L_2}{r^2} \dot{\phi} \cos \phi \right) \dot{X}_x + \left( \frac{2c}{r^2} \cos \phi + \frac{2L_2}{r^2} \dot{\phi} \sin \phi \right) \dot{Y}_x + \frac{c}{r^2} (L_3 - L_2) \dot{\phi} \\
\left( \frac{c}{r^2} (L_4 - L_2) \cos \phi + (L_3 - L_4) \sin \phi \right) + \frac{L_2}{r^2} \dot{\phi} (L_3 - L_4) \cos \phi + (L_4 - L_3) \sin \phi \right) \dot{X}_x \\
\left( \frac{c}{r^2} (L_4 - L_2) \sin \phi + (L_3 - L_4) \cos \phi \right) + \frac{L_2}{r^2} \dot{\phi} (L_3 - L_4) \sin \phi + (L_4 - L_3) \cos \phi \right) \dot{Y}_x + \frac{c}{r^2} \left( \dot{z}_1 + \dot{z}_2 + \dot{z}_3 + \dot{z}_4 \right)
\end{align*}
\]

\[
\mathbf{B} = \begin{bmatrix}
0 & k & 0 & k \\
k & 0 & -k & 0 \\
\frac{L_1}{r} & \frac{L_2}{r} & \frac{L_3}{r} & \frac{L_4}{r}
\end{bmatrix}
\]

After manipulation, we have three coupled MIMO nonlinear differential equations of motion with wheel torques as inputs, and the global frame Cartesian position and orientation as outputs. Vector \( \mathbf{N}_x \) is a function of the global frame Cartesian velocity and rotation.
3.7 Actuator Dynamics

To completely model the mobile robot system, it is necessary to add the actuator dynamics. As discussed in the chapter on the mechanical design, the actuators are DC motors. The DC motor, Figure 3.4, converts direct current (DC) electrical energy into rotational mechanical energy. A major portion of the torque generated in the rotor (armature) of the motor is available to drive an external load. Because of features like high torque, speed controllability, and adaptability over various types of control methods, DC motors are used in numerous control applications.

![DC Motor Diagram](image)

*Figure 3.4: DC Motor*

The electric circuit of the armature and the free body diagram of the rotor are shown in *Figure 3.5.*
The motor torque, $T_m$, is related to the armature current, $i$, by the armature constant, $K_t$. The back $V_{emf}$ is related to the rotational velocity by the motor constant, $K_e$. $T_L$, the load torque applied to the motor from the wheel assemblies.

\[ T_m = K_t i \]  

(3.36)

\[ V_{emf} = K_e \dot{\Theta} \]  

(3.37).

Using Newton’s 2\textsuperscript{nd} Law and Kirchhoff’s Voltage Law with the relationships above, we derive the following equations:

\[ J_m \ddot{\Theta} + b \dot{\Theta} + T_m = K_t i \]  

(3.38)
A practice common in industry is the assumption that $L$, the rotor inductance, is approximately zero $L \approx 0$. We will also make this assumption, yielding:

$$L \frac{di}{dt} + Ri + K_e \dot{\Theta} = V$$  \hspace{1cm} (3.39)$$

These equations hold true for each wheel assembly; recall that there are four wheel assemblies.

Assuming a no slip condition, we can describe the relationship between the load angle and the motor angle as:

$$Ri + K_e \dot{\Theta} = V$$  \hspace{1cm} (3.40)$$

$$\Theta = nq$$  \hspace{1cm} (3.41),

where $n$ is the gear ratio.

From this relationship we can obtain the relationship between the motor torque and the load torque.

$$T_m = \frac{1}{n} T_L$$  \hspace{1cm} (3.42).$$
Substituting *Equation (3.42)* into *Equation (3.38)* we have:

\[ J_m \ddot{\Theta} + b \dot{\Theta} + \frac{1}{n} T_L = K_i i \]

or

\[ T_L = n \left( K_i i - J_m \dddot{\Theta} - b \dot{\Theta} \right) \]

(3.43).

Substituting *Equation (3.41)* into *Equation (3.43)* we have:

\[ T_L = n \left( K_i i - J_m \dddot{q} - b \dot{q} \right) \]

or

\[ u_i = nK_i i - n^2 J_m \dddot{q}_i - n^2 b \dot{q}_i \]

(3.44),

written on a joint-by-joint basis.

Then, substituting *Equation (3.41)* into *Equation (3.40)* we have:

\[ Ri + K_e n \dot{q} = V \]

(3.45).

Using the above equations a complete robot model can be written.
Combining Equation (3.34) with Equation (3.44) we obtain the complete robot model with respect to joint frame.

\[
\begin{align*}
\left[ P(Q^T) + n^2 B J_m I \right] \dddot{q} + n^2 b B \dot{q} + N_q &= n K_i B \frac{\dot{V}}{R} \\
\end{align*}
\]  

(3.46).

Similar to Equation (3.34), there are three coupled nonlinear differential equations of motion. The inputs of this equation are four armature currents; the outputs are still the wheel angles, which can be interpreted by the encoders.

Likewise, combining Equation (3.46) with Equation (3.45) we have the complete robot model, but instead of an input current, \(i\), we have a voltage input, \(V\). The equations of motion are as follows:

\[
\begin{align*}
\left[ P(Q^T) + n^2 B J_m I \right] \dddot{q} + n^2 b B \left( b + \frac{K_e}{R} \right) \dot{q} + N_q &= \frac{n K_i}{R} B \frac{\dot{V}}{R} \\
\end{align*}
\]  

(3.47).

For the purposes of control and simulation, Equation (3.46) will be used. We have only shown one case where the actuator dynamics have been applied to the equations of motion to obtain a complete robot model. We have chosen the most complex and the most important equation to completely model; the joint frame model. The procedure for adding the actuator dynamics to the other equations of motion is the same.
3.9 Robot Frame Model

Combining *Equation (3.30)* with *Equation (3.44)* we obtain the complete robot model with respect to the robot frame:

\[
\begin{bmatrix}
P + n^2 B J_n \dot{Q}^T \end{bmatrix} \ddot{\hat{s}} + n^2 b B Q^T \dot{\hat{s}} + N_s = nK, B \hat{\xi} \tag{3.48}
\]

Similar to *Equation (3.30)*, there are three coupled nonlinear differential equations of motion. The inputs of this equation are four armature currents; the outputs are still Cartesian position and orientation with respect to the robot (local) frame.
CHAPTER 4: CONTROL

The fundamental tasks of this mobile robot are to intercept the ball and block the goal whenever a shot is taken. To accomplish these tasks, a motion controller must be designed such that the robot moves with minimum steady-state error, rise time, settling time, and overshoot with respect to the desired path. These performance specifications are important to the overall response of the system. If any performance specification were ignored, it would take away from the overall response of the system. For instance, if the overshoot was too high the robot might not intercept the ball; the opposing team would easily take advantage of this weakness. If rise time were not fast enough the robot would not get to the ball in time to intercept it. In other words, the overall quality of the system is measured by the design of the controller. In this chapter a proposed controller design will be described for the nonlinear, multiple-input multiple-output, redundant mobile robot.

4.1 Redundancy

As previously stated, this mobile robot is redundant. Redundancy is caused from having more actuators than degrees of freedom. In this case we have three degrees of freedom and four actuators. Redundant robots have several advantages over other types of non-redundant mobile robots. Some of these advantages include: better mobility and the ability to complete tasks even in the event of failure of one or more of the actuators.
Contrasting these advantages, we have several disadvantages of redundant robots: their structure is more complex, they are bulkier, and the system equations may be more complex. Another disadvantage would be that the control algorithm becomes more complex, leading to increased computations.

The reason for the redundancy of the OU RoboCup Goalie is that in order to have the fastest response to guard the goal, the robot must be able to move in front of the goal without having too many complex movements. This has been accomplished by aligning the wheels parallel to the goal box. The wheels perpendicular to these give the mobile robot a full range of motion to better defend the goal and more mobility to get the best defensive stance against the opposing team.

Since our robot is redundant, a viable solution method is to formulate the problem as a constrained linear optimization problem in order to determine a force vector that maximizes the work done by the bounded joint torques. From our equations of motion we have the relationship:

\[ m \mathbf{f} = \mathbf{B[U]} \]  \hspace{1cm} (4.1).

It is desired to find the solutions to \( U \) that satisfy the linear equation \((4.1)\) and minimize the quadratic cost function of joint torques:

\[ g = \frac{1}{2} \mathbf{U}^t \mathbf{WU} \]  \hspace{1cm} (4.2)
where $W$ is a suitable positive definite weighting matrix. This problem can be solved using the method of Lagrangian multipliers. Consider the modified cost function:

$$ g = \frac{1}{2} U^T W U + \lambda^T (\mathbf{f} - B U) $$

where $\lambda$ is a vector of unknown multipliers that allows incorporating the constraint (4.1) in the function to minimize. The requested solution has to satisfy the necessary conditions:

$$ \left( \frac{\partial g}{\partial U} \right)^T = 0 $$

and

$$ \left( \frac{\partial g}{\partial \lambda} \right)^T = 0 $$

From constraint (4.4) we have $W U - B^T \lambda = 0$ and thus:

$$ U = W^{-1} B^T \lambda $$
where the inverse of $W$ exists. From (4.5), the constraint (4.1) is recovered. Combining
the two conditions gives:

\[ \lambda = \left( BW^{-1} B^T \right)^{-1} m f \]  

(4.8).

Which can be substituted into (4.6) which gives the optimal solution:

\[ U = W^{-1} B^T \left( BW^{-1} B^T \right)^{-1} m f \]  

(4.9).

A particular case occurs when the weighting matrix $W$ is the identity matrix $I$ and the
solution simplifies into:

\[ U = B^+ m f \]  

(4.10)

where:

\[ B^+ = B^T \left( BB^T \right)^{-1} \]  

(4.11).
is the pseudo-inverse of $B$ which locally minimizes the norm of the joint torques.

4.2 Nonlinear Control

As derived in the chapter on modeling, the equations of motion are coupled nonlinear differential equations. The source of the nonlinearity is attributed to the coupled translation and rotation of the equations of motion. Solutions to nonlinear equations are more difficult to obtain. In some instances when a system is not severely nonlinear, local linearization may be used to derive linear models, which are approximations of the nonlinear equations about an operating point. However, to accurately control the nonlinear system, a nonlinear control technique must be used in order to get the desired output from the robot. Consequently, the major focus of this chapter will be deriving and understanding one particular method, the computed torque method.

4.3 Multiple-Input, Multiple-Output

The OU RoboCup Goalie has four inputs, a vector of motor torques, $\{U\}$; and three outputs, Cartesian position in the plane, and orientation with respect to the inertial frame, $\{"S\}$. Applying different motor torques to each wheel will give us a response. It is the job of the controller to give us the desired response. In order to design a
controller that suits the needs of this MIMO system, a control law must be able to handle all inputs and outputs. This control law takes the form:

\[ F = \alpha F' + \beta \]  

(4.12).

Where, for a system of \( n \) degrees of freedom, \( F, F' \), and \( \beta \) are \( mx1 \) vectors in joint space; and \( \alpha \) is an \( mxm \) matrix. Note that matrix \( \alpha \) is chosen in order to decouple the \( n \) equations of motion. If \( \alpha \) and \( \beta \) are chosen correctly, the system appears to have \( m \) independent \( F' \) inputs. For this reason, the model-based portion of the control law is referred to as linearizing and decoupling control law. The servo law for this system yields:

\[ F' = \ddot{X}_d + K_d \dot{E} + K_p E \]  

(4.13).

Where \( K_d \) and \( K_p \) are now \( nxn \) matrices, which are generally chosen with constant gains on the diagonal. \( \dot{E} \) and \( E \) are \( nx1 \) vectors of errors in velocity and position.

### 4.4 Computed Torque Control

In the chapter on modeling, the equations of motion were developed. It is important to note that in order to use the computed torque method, it is assumed that the derived
model is the exact model. We will assume that \( L_1=L_3 \) and \( L_2=L_4 \) to simplify the equations of motion and to reduce the number of computations.

### 4.4.1 Local Frame Control

For the local control, the equations of motion with respect to the robot frame with wheel torques as inputs, and the Cartesian position as the outputs with respect to the robot frame, will be used. The simplified equations are shown below:

\[
P\left\{ \begin{array}{c}
\ddot{x}_s \\
\ddot{y}_s \\
\ddot{\phi}
\end{array} \right\} + N_s = B \{ \tau \}
\]

(4.14).

Where:

\[
P = \begin{bmatrix}
\frac{2I_w}{r^2} + M & 0 & 0 \\
0 & \frac{2I_w}{r^2} + M & 0 \\
0 & 0 & \frac{I_w}{r^2} \left( 2L_1^2 + 2L_2^2 \right) + I_m
\end{bmatrix}
\]

\[
N_s = \begin{bmatrix}
-M y_m \phi + \frac{2c}{r^2} x_m \\
M x_m \phi + \frac{2c}{r^2} y_m \\
\frac{c}{r^2} \left( 2L_1^2 + 2L_2^2 \right) \phi
\end{bmatrix}
\]
Since the form of these equations does not match the form of the control law, we must manipulate them so that they do. In order to achieve this the pseudo-inverse of matrix $B$ must be taken. Since matrix $B$ has a size of 3x4 and satisfies $\text{rank}(B) = 3$, the pseudo-inverse of this matrix, $B^+$, based upon the minimum norm becomes:

$$B^+ = B^T(BB^T)^{-1}$$ (4.15).

Thus, the equations of motion take the form:

$$B^+ P \ddot{\theta} + B^* N_z = U$$ (4.16).
The problem of controlling this complicated system can be handled by the partitioned controller scheme introduced in this chapter. In this case we have:

\[ U = \alpha U' + \beta \]  

(4.17),

where \( U \) is the 4x1 vector of motor torques. We choose:

\[ \alpha = B^* P \]  

(4.18)

\[ \beta = B^* N_s \]  

(4.19).

Resulting in a unit mass system dynamics:

\[ I^{\text{m}} \dddot{s} = U' \]  

(4.20).

With the servo law:

\[ U' = \ddot{s}_d + K_d E + K_p E \]  

(4.21),

where
Substituting Equations (4.16)-(4.22), it is easily seen that the system is linearized.

The linearized equation is shown in Equation (4.23):

\[
\begin{bmatrix}
\dddot{s} \\
\ddot{s}
\end{bmatrix} =
\begin{bmatrix}
m & 0 \\
0 & m
\end{bmatrix}
\begin{bmatrix}
\dddot{s} \\
\ddot{s}
\end{bmatrix} +
K_d \dddot{s} + K_p \ddot{s}
\]

(4.23).

Using Equations (4.16)-(4.22), it is easily seen that the closed loop system is characterized by the error equation:

\[\dddot{E} + K_d \ddot{E} + K_p E = 0\]

(4.24).

Note that this vector equation is decoupled since the matrices \(K_d\) and \(K_p\) are diagonal, so that Equation (4.24) could be written in a joint-by-joint basis as:

\[\dddot{e}_i + K_{d_i} \ddot{e}_i + K_{p_i} e_i = 0\]

(4.25).

The resulting control system is shown in Figure 4.1:
4.4.2 Global Frame Control

For the global "outer-loop" control the equations of motion, with respect to the robot frame with wheel torques as inputs and the robots Cartesian position as the outputs, will be used. The simplified equations are shown below:

\[ P_x \tilde{\mathbf{S}} + N_x = B\{U\} \]

(4.26).

Where:
Again, the form of these equations does not match the form of the control law. To achieve the correct form we use the minimum norm pseudo-inverse of $B$ calculated in the previous section. Following manipulation we have:

$$B^* P_g \begin{bmatrix} \ddot{w} \\ \dot{w} \end{bmatrix} + B^* N_g = U$$  \hspace{1cm} (4.27)$$

Using the partitioned controller scheme introduced earlier, we choose:
\[ \alpha = B^+ P_g \]  
(4.28)

\[ \beta = B^+ N_g \]  
(4.29)

By doing this we have linearized and decoupled the mobile robot system. We can now apply the above equations to the control scheme presented in Figure 4.1.

### 4.4.3 Joint Frame Control

For the local "inner-loop" control, the equations of motion with respect to the robot joint frame will be used. The equations of motion have wheel torques as inputs and wheel angular position as outputs. The simplified equations are shown below:

\[ P(Q^T) \{ \ddot{q} \} + N_q = B^T U \]  
(4.30)

Where:

\[
P = \begin{bmatrix}
\frac{2I_w}{r^2} + M & 0 & 0 \\
0 & \frac{2I_w}{r^2} + M & 0 \\
0 & 0 & \frac{I_w}{r^2} \left(2L_1^2 + 2L_2^2 \right) + I_m
\end{bmatrix}
\]
Yet again the form of these equations does not match the form of the control law. To achieve the correct form we use the minimum norm pseudo-inverse of $B$ calculated in an earlier section. Following manipulation we have:

$$B^* P(Q^T) \{q^{**}\} + B^* N_q = U$$

(4.31).

Using the partitioned controller scheme introduced earlier, we choose:
By doing this we have linearized and decoupled the mobile robot system. We can now apply the above equations to the model-based control scheme presented in Figure 4.2.

\[
\alpha = B^* P (Q^T)
\]

(4.32)

\[
\beta = B^* N_q
\]

(4.33)

By doing this we have linearized and decoupled the mobile robot system. We can now apply the above equations to the model-based control scheme presented in Figure 4.2.

*Figure 4.2: Model-Based Control Scheme*


4.5 Joint-Actuator control

For the local “inner-loop” control the equations of motion with respect to the robot joint frame, with armature current inputs and wheel angels as outputs, will be used. These equations of motion are repeated below for convenience:

\[
\begin{aligned}
\left[ P(Q^T) + n^2BJ_nI \right] \ddot{q} + n^2bB \dot{q} + N_q &= nK_B \dot{q} \\
\end{aligned}
\] (4.34)

The same method used to derive the global control scheme will be utilized in this section.

Since the form of these equations does not match the form of the control law, manipulation of the equations will be necessary so that they do. To complete this task, the pseudo-inverse of matrix \( B, B^\dagger \), defined in the previous section of this chapter, will be needed. Using \( B^\dagger \), the equations of motion take the form:

\[
\begin{aligned}
\frac{1}{nK_i} B^\dagger P(Q^T) + n^2J_nI \ddot{q} + \frac{nb}{K_i} \dot{q} + \frac{1}{nK_i} B^\dagger N_q &= \dot{q} \\
\end{aligned}
\] (4.35)

The problem of controlling this highly complex system can be handled by use of the partitioned controller scheme introduced in this chapter. In this case we have:

\[
\{i\} = \alpha \{i\} + \beta 
\] (4.36)
where $i$ is the 4x1 vector of input currents. We choose

$$
\alpha = \frac{1}{nK_i} \left[ B^T P \left( Q^T \right) + n^2 J_m i \right]
$$

(4.37)

$$
\beta = \frac{nb}{K_i} \left\{ \dot{q} \right\} - \frac{1}{nK_i} B^T N_q
$$

(4.38)

Resulting in a unit mass system dynamics:

$$
I \dddot{q} = \left\{ \ddot{q} \right\} \quad (4.39)
$$

With the servo law:

$$
\left\{ \dot{q} \right\} = \dddot{q} + K_d \dot{E} + K_p E
$$

(4.40)

where

$$
E = q_d - q \quad (4.41)
$$
Substituting Equations (4.26)-(4.30), it is quite easy to see that the system is linearized.

The linearized equation:

\[
\begin{align*}
\ddot{\mathbf{q}} &= \dot{\mathbf{q}}_d + K_d \dot{E} + K_p E
\end{align*}
\]  

(4.42).

The resulting control system is shown below, Figure 4.3:

---

**Figure 4.3: Model-Based Joint Control Scheme**

---

**4.6 Vision Feedback Control with Joint Actuator Computed Torque**

Now we create an “outer-loop” control scheme, similar to the one described by Luo [16], to be used with our global vision system. For the mobile robot block, we use the previous joint actuator computed torque control scheme as presented in section 4.5. The vision system block will feedback the actual position of the mobile robot with
respect to itself, the global frame. Once this is done, the difference, desired less actual, can be applied to a proportional gain controller to adjust the position and decrease the amount of error during the robots translation. The resulting feedback control scheme is shown below, Figure 4.4:

![Vision Feedback Control Scheme](image)

**4.4: Vision Feedback Control Scheme**

### 4.7 Practical Considerations

In considering control strategy, we implicitly assume that the entire system is running in continuous time, and that the computations in the control scheme require zero time for their computation. Given any amount of computation, it is a reasonable assumption that we can do the computations sufficiently fast with a large enough computer. In the case such as this mobile robot, the entire dynamic equations of motion must be computed in the control scheme. These computations are quite involved and consequently there has been a great deal of interest in developing a fast computational scheme to compute them in an efficient way.
Almost all control schemes are now performed in digital circuitry and are run at a certain sampling rate. This means that the encoders are read at discrete points in time. Based on the value read, an actuator command is computed and sent to the actuator. Thus, reading the encoders and sending actuator commands are not done continuously, rather at a finite sampling rate. In discrete time, differential equations turn into difference equations.

Although we have managed to write complicated differential equations of motion for the mobile robot, a discrete time equivalent is impossible to obtain in general. This is because the only way to solve for the motion of the mobile robot for a given set of initial conditions, the inputs, and a finite interval, is by numerical integration. We will generally assume that the computations can be performed quickly enough and often enough that the continuous time approximation is valid.

A potential problem when computing a torque control algorithm is that the dynamic model is not accurately known. This is particularly true for the effects that friction will have on the system. It is extremely difficult to know the parameters of the friction model. Another problem is that vision updates may not be uniformly spaced in time and may lead to significant latencies.
The major advantage of modeling a system and analyzing its response is that it allows us to predict the behavior of the system before it is built; this is sometimes referred to as "virtual prototyping". We also can analyze an existing system with the intent of improving the behavior of the system, or we can determine what might happen to a system with an unusual input or condition without exposing the actual system to danger. A good mathematical model can tell us the performance characteristics of the system. Of course, our solutions to differential equations will not be an exact representation of the actual system. In many cases, simulation by digital computation is favored over the use of analytic solutions to differential equations for several reasons. First, given the right software, it does not require a lot of effort, experience, or time to set up a simulation. Second, we can make changes in the model and quickly obtain the response using the same computational techniques as before, whereas by analytical methods, a change in the differential equations might require a completely different analytic solution to develop. Third, and perhaps the most important, nonlinear systems can be handled with relative ease, whereas analytic solutions may not be possible.

It is important to note that a numerical solution of a set of nonlinear system equations gives only a numerical result for that particular circumstance; a large number of simulations may be required to detect performance trends corresponding to variations in the values of parameters.
This chapter will describe the controlled system responses for the robot’s equations of motion described in the previous chapters, Chapter 3: Modeling and Chapter 4: Control. Table 5.1 shows the system parameters used for the simulations in this chapter.

### Table 5.1: System Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gear Ratio (k)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Gear Ratio (n)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$L_i$ for $i=1, 2, 3, 4$</td>
<td>.1</td>
<td>m</td>
</tr>
<tr>
<td>Motor Inertia ($J_m$)</td>
<td>2.7003</td>
<td>N-m</td>
</tr>
<tr>
<td>Motor Friction (b)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Robot Inertia ($I_m$)</td>
<td>2.26</td>
<td>Kg</td>
</tr>
<tr>
<td>Torque Constant ($K_t$)</td>
<td>.014497</td>
<td>N-m/A</td>
</tr>
<tr>
<td>Viscous Friction Factor (c)</td>
<td>.0113</td>
<td>N/m</td>
</tr>
<tr>
<td>Wheel Inertia ($I_w$)</td>
<td>1.46E-5</td>
<td>Kg-m^2</td>
</tr>
<tr>
<td>Wheel Radius (r)</td>
<td>.0254</td>
<td>m</td>
</tr>
</tbody>
</table>

### 5.1 Kinematic Simulation

The inverse kinematics solution is simulated using Matlab. The robot follows an arbitrary straight-line path from (-0.5, 0) m to (0.5, 2.5) m in 5 seconds while maintaining the orientation as shown in Figure 5.1(a). The trajectory and velocity profile were described using a third-order polynomial with zero initial and final velocity. Figure 5.1(c) shows the corresponding wheel angular position and velocity via the inverse kinematics solution.
5.2 Local Frame Computed Torque

Using the computed torque control algorithm described in Chapter 4: Control the local dynamics of the system was simulated using Simulink, Matlab's graphical interface. The robots' path is still the arbitrary straight-line defined in the previous section. Proportional and derivative gains were designed using a second order ITAE performance index. ITAE is used to simultaneously optimize competing requirements. The results from this simulation using the control block diagram, Figure 5.2, are shown in Figure 5.3.
Figure 5.2: Simulink Block Diagram, Global /Local Computed Torque Control Simulation

Figure 5.3: Local Frame Simulation Results
The result is from planned path inputs, using a third order polynomial and solving the boundary value problem. As can be seen in Figure 5.3, the simulation using the computed torque method closely mimics the desired path. The $X$ and $Y$ results overlap the desired path, while the $\phi$ result differs from the desired by approximately one-degree. This simulation deals only with mechanical factors like inertia and not any electrical components or digital control effects. Wheel slip has not been modeled; thus the real-world results may not mimic the simulated results.

### 5.3 Global Frame Computed Torque

Similarly the results from the global dynamics have been simulated using Simulink. The same straight-line path used for the local dynamics defined by the boundary value problem was the input for the control scheme shown in Figure 5.2. The results from this simulation are shown below, Figure 5.4.
The $X$ and $Y$ results overlap the desired path, while the $\phi$ result differs from the desired by approximately one-degree. Again, this simulation deals only with mechanical factors like inertia and not any electrical components or digital control effects.

### 5.4 Joint Frame Computed Torque

The joint frame equations were simulated using the block diagram shown in Figure 5.5. This simulation also uses the same arbitrary straight-line path defined previously, except this path gets converted to the joint frame by applying the inverse kinematics derived in Chapter 3: Modeling.
The results from this simulation, shown in Figure 5.6, very closely mimic the desired.
5.5 Joint-Actuator Computed Torque

Now, we have added the actuator dynamics to the joint frame equations of motion. We still use the same arbitrary straight-line path and the same Simulink block diagram shown in Figure 5.5. Similarly to the joint frame simulation results, the results from this simulation very closely mimic the desired, refer to Figure 5.7.

Figure 5.7: Joint-Actuator Frame Simulation Results
5.6 Vision Feedback Control with Joint-Actuator Computed Torque

Using the joint-actuator Simulink block diagram, a new simulation is solved with the addition of a feedback loop. This feedback loop adds the effects of the global vision system, which is represented by a 20Hz-sampling rate between the encoder feedback and the vision feedback. This technique is used to reduce error in the position of the mobile robot. The resulting Simulink block diagram is shown in Figure 5.8. The results can be seen in Figure 5.9.

![Figure 5.8: Vision Feedback Control with Joint-Actuator Computed Torque](image-url)
The input to the system is a new arbitrary straight-line path from \((0, 0)\) m to \((-1, 2.5)\) m while maintaining the orientation as shown. The results shown in Figure 5.9 more closely mimic the desired results. In the previous position simulations there was some error associated with the orientation of the mobile robot. With the addition of the vision feedback “outer-loop” this error has been corrected.

In addition to showing position results it is necessary to look at the input current using this control scheme, Figure 5.10.
Motor Current vs Time

<table>
<thead>
<tr>
<th>time (sec)</th>
<th>0</th>
<th>0.5</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3</th>
<th>3.5</th>
<th>4</th>
<th>4.5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>current (amps)</td>
<td>0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.10: Input Current

It is important to note that this simulation does not take into account amplifier dynamics, encoder feedback delay and other electronic unknowns; however the results are within acceptable ranges for current.

5.7 Controller Robustness

As components are placed on the mobile robot and configurations are adjusted it is important for the controller to be accurate and deliver the robot to its specified location. This concept is known as robustness. We now look at this idea with respect to the goalie robot. Two cases will be discussed; a 10% (Figure 5.11) and then a 20% (Figure
A 10% increase in payload will be applied and simulated using the global frame control block diagram.

**Figure 5.11: 10% Increase in Payload**

**Figure 5.12: 20% Increase in Payload**
The results from these simulations show that changing the payload has little effect over the performance of the controller. However, the computed torque control law is based on the assumption that the exact model is known in order to linearize and decouple the nonlinear equations of motion. Only experimental results can prove or disprove these simulated results.
CHAPTER 6: CONCLUDING REMARKS and CONTRIBUTION

This thesis focused on the design, modeling, control, and simulation of a redundant omni-directional holonomic mobile robot. The mobile robot platform was developed for the Robocup, an international competition of autonomous mobile robots that compete in soccer. This mobile robot was specifically designed for the position of goalie. The design intent was to give the robot more of a side-to-side motion, without complex movements, to better protect the goal than the current player design.

This chapter will discuss the conclusions of this thesis, as well as its contributions to the academic world, and finally the future work.

6.1 Conclusions

Contrasting traditional mobile robot designs that must rotate before translation, a mobile robot platform was designed to be omni-directional. Omni-directional drive is the ability to move in any direction at any given orientation. Using this type of drive system, it enables the mobile robot to rotate during translation. This type of drive system allows the robot to always face the ball in order to stop a potential goal. Additionally, this type of drive system provides simpler inverse kinematic solutions.

During competition, the robots will be controlled using a global vision system. This system will allow the robots to move strategically in order to make the best possible attempt on goal while avoiding collisions with other robot players. Additionally, the
goalie robot can be equipped with a local vision system to improve on response time to better protect the goal. Since the mobile robot has these vision systems as well as encoders to determine wheel angular position there were several possibilities for modeling and controlling the system.

Equations of motion have been developed with respect to the robot (local) frame, the global frame, and the joint angle frame. These equations were first derived without the actuator dynamics and then with them. The addition of the actuator dynamics greatly increases the complexity of the equations of motion. However, in order to abridge these equations of motion a simplification of the link lengths was applied to reduce the number of computations during simulation.

However, even after the simplification, the equations of motion have coupled translational and rotational motions. Because of this, a multiple input multiple output control law was implemented called the computed torque method. By means of a linearizing and decoupling control law, this control method decouples and cancels out the nonlinear equations of motion resulting in a unit mass system characterized by an error equation.

A vision feedback control scheme was also incorporated to improve position and orientation error during motion. This vision feedback control scheme makes use of the computed torque control of the dynamic system as part of the “inner-loop”.

In order to utilize the redundancy of the system an optimization problem was defined to determine a force vector that maximizes the work done by the bounded joints. This optimization minimizes the norm of the joint torques. This can be intuitively thought of
as distributing the load equally, so that no one joint has to have a very high velocity with respect to the others.

The equations of motion and control schemes were simulated using Simulink. The results from these simulations closely mimic the desired path. There was, however, some error with the rotational position of the robot. With the addition of vision feedback applied to the computed torque control scheme error in the results greatly decreased.

There is an inherent flaw in the four-wheel robot design. Only three points, not four, define a plane. That means that unless machining tolerances are zero one wheel will not be touching the ground. Because of this, experimental results could not obtained with the current design. An alternate design that includes a suspension system for the wheel assemblies would have to be built.

6.2 Contribution

Previously, conventional studies on omni-directional mobile robots have focused on the development of mechanisms, or only on the analysis of kinematics. Hence, there are few studies on the development of dynamic models and highly accurate control schemes. This thesis does just that, it shows the derivation of the dynamic model and uses a highly accurate control scheme. Also, the system described in this thesis is redundant, having more actuators than degrees of freedom. Currently, the majority of papers on omni-directional mobile robots describe a three-wheeled system. This thesis
explained an optimization of joint torques described as distributing the load equally between the joints.

6.3 Future Work

Although this thesis is concluded the work on this subject is not. An innovative design was introduced that could play a major role in the future of the Ohio University Robocup Team. Currently this is a theoretical design, this design should be built and tested to obtain good experimental results. An enhancement to the design would be the addition of a suspension system. Adding a suspension system would help to maintain even, level, contact with the four wheels and the surface. Once the system is built a friction model could be derived to help reduce error. This friction model will add more complexity to the system but it will help to understand the effects that friction plays on the system.
REFERENCES


APPENDICES

- Function to perform smooth motion path computation based on pose and orientation...81
- Goalie Kinematics (inverse solution)...82
- Goalie Kinematics (forward solution with path planning for smooth motion)...84
- Goalie Simulation...87
- Symbolic Derivation of Jacobian Transpose PsuedoInverse, Based Upon Minimum Norm, Simplified Case L1 = L2...91
- Symbolic Derivation of Jacobian Transpose PsuedoInverse, Based Upon Least Squares Error, Generic and Simplified Cases...92
- Symbolic Dynamic EOMs for the 3 dof Redundant Holmomic Mobile Robot wrt Global Vision...95
- Symbolic Dynamic EOMs for the 3 dof Redundant Holmomic Mobile Robot wrt Local Vision...96
- Simplified Equations for use in DEE and Computed Torque Control Scheme, wrt Joint Frame with actuator dynamics...98
- Simplified Equations for use in DEE and Computed Torque Control Scheme, wrt Local Vision with actuator dynamics ...100
- Simplified Equations for use in DEE and Computed Torque Control Scheme, wrt Global Vision...102
• Simplified Equations for use in DEE and Computed Torque Control Scheme, wrt Joint Frame…104

• Simplified Equations…106

• Simplified Equations for use in DEE and Computed Torque Control Scheme, wrt Local Vision…107
function [X,Xdot] = smpath(Xo,Yo,THo,Xf,Yf,THf,tf);

% function to perform smooth motion path computation based on pose and orientation

t0 = 0; % initial time
dt = .1; % discrete time step
t = [t0:dt:tf]; % time range

% path planning
X = Xo + 3*(Xf-Xo).*t.^2/tf^2-2*(Xf-Xo).*t.^3/tf^3;
Y = Yo + 3*(Yf-Yo).*t.^2/tf^2-2*(Yf-Yo).*t.^3/tf^3;
TH = THo + 3*(THf-THo).*t.^2/tf^2-2*(THf-THo).*t.^3/tf^3;
X = [X;Y;TH]; %POSE Vector

Xdot = 6*(Xf-Xo).*t/tf^2 - 6*(Xf-Xo).*t.^2/tf^3;
Ydot = 6*(Yf-Yo).*t/tf^2 - 6*(Yf-Yo).*t.^2/tf^3;
THdot = 6*(THf-THo).*t/tf^2 - 6*(THf-THo).*t.^2/tf^3;
Xdot = [Xdot;Ydot;THdot]; % Velocity Vector
clc; clear; clf;
DR = pi/180;  % degrees to radians conversion

B = .18;  % robot dimension
L = sqrt((B/2)^2+(B/2)^2);  % robot dimension
alp = 90*DR;  % robot dimension
r = 1;  % wheel radius
L1=1; L2=1; L3=1; L4=1;  % robot dimension
J = 1/r*[0,-1,0,1;...
    1,0,-1,0;...
    L1, L2, L3, L4];  % Jacobian Matrix
M = J';  % transpose of the jacobian matrix

% Forward Solution: given robot and cartesian rates, find wheel velocities
% Xdot =
% Qdot = M*Xdot

% Inverse Solution: given robot and wheel velocities, find cartesian rates
Qdot1 = input('Enter wheel velocity of wheel #1 (rad/sec):');
Qdot2 = input('Enter wheel velocity of wheel #2 (rad/sec):');
Qdot3 = input('Enter wheel velocity of wheel #3 (rad/sec):');
Qdot4 = input('Enter wheel velocity of wheel #4 (rad/sec):');
Qdot = [Qdot1;Qdot2;Qdot3;Qdot4];  % Wheel Velocity Vector
X = input('Enter initial robot position in the x-direction:');
Y = input('Enter initial robot position in the y-direction:');
TH = input('Enter initial rotation position (deg):');
TH = TH*DR;

t0 = 0; dt = .1; tf = 2;
t = [t0:dt:tf];  % time
N = (tf-t0)/dt+1;

for i = 1:N,
    Xdot = pinv(M)*Qdot;
    X = X+Xdot(1)*dt;  % integration
    Y = Y+Xdot(2)*dt;
    TH = TH+Xdot(3)*dt;

    % animation of robot
xc = [0 X];
yc = [0 Y];
x1 = [X+L*cos(TH+alp/2) X+L*cos(TH+alp/2+alp)];
y1 = [Y+L*sin(TH+alp/2) Y+L*sin(TH+alp/2+alp)];
x2 = [X+L*cos(TH+alp/2+alp) X+L*cos(TH+alp/2+2*alp)];
y2 = [Y+L*sin(TH+alp/2+alp) Y+L*sin(TH+alp/2+2*alp)];
x3 = [X+L*cos(TH+alp/2+2*alp) X+L*cos(TH+alp/2+3*alp)];
y3 = [Y+L*sin(TH+alp/2+2*alp) Y+L*sin(TH+alp/2+3*alp)];
x4 = [X+L*cos(TH+alp/2+3*alp) X+L*cos(TH+alp/2)];
y4 = [Y+L*sin(TH+alp/2+3*alp) Y+L*sin(TH+alp/2)];
xcoordx = [X X+B*cos(TH)];
ycoordx = [Y Y+B*sin(TH)];
xcoordx = [X X+B*cos(TH+pi/2)];
ycoordx = [Y Y+B*sin(TH+pi/2)];

plot(X,Y,'o',x1,y1,x2,y2,x3,y3,x4,y4,xcoordx,ycoordx,xcoordx,ycoordx,xcoor,ycoor);
title('Goalie Position');
xlabel('X (itm)'); ylabel('Y (itm)');
axis([-1.5 1.5 0 2.74]);
axis('normal');
axis('equal');
axis('manual');
grid;
pause(1/8);
end
% RoboCup: Goalie Kinematics (forward solution with path planning for smooth motion)
% Lance Wilson
% 11/26/00

clear; clf; % clear
DR = pi/180; % degrees to radians conversion

B = .18; % robot dimension
L = sqrt((B/2)^2+(B/2)^2); % robot dimension
alp = 90*DR; % robot dimension
r = .0254; % wheel radius
L1=.1; L2=.1; L3=.1; L4=.1; % robot dimension

J = 1/r*[0,-1,0,1;...
     1,0,-1,0;...
     L1,L2,L3,L4]; % Jacobian Matrix
M = J'; % transpose of the jacobian matrix

Xo = 0; % input('Enter initial robot position in the x-direction:');
Yo = 0; % input('Enter initial robot position in the y-direction:');
PHIo = 0; % input('Enter initial rotation position (deg):');
PHIo = PHIo*DR;
Xf = -1; % input('Enter final robot position in the x-direction:');
Yf = 2.5; % input('Enter final robot position in the y-direction:');
PHIf = 0; % input('Enter final rotation position (deg):');
PHIf = PHIf*DR;

tf = 5; % input('Enter time for the robot to move to desired position (sec):');
t0 = 0; dt = 5/57;
t = [t0:dt:tf]; % time
N = (tf-t0)/dt+1;

% initial wheel rotation
q1(1)=0; q2(1)=0; q3(1)=0; q4(1)=0;

for i = 1:N,
    X = Xo+3*(Xf-Xo).*t(i).^2/tf^2-2*(Xf-Xo).*t(i).^3/tf^3;
    Y = Yo+3*(Yf-Yo).*t(i).^2/tf^2-2*(Yf-Yo).*t(i).^3/tf^3;
    PHI = PHIo+3*(PHIf-PHIo).*t(i).^2/tf^2-2*(PHIf-PHIo).*t(i).^3/tf^3;
    x(i) = X; y(i) = Y; phi(i) = PHI; % storing values for plots

    Xdot = 6*(Xf-Xo).*t(i)/tf^2-6*(Xf-Xo).*t(i).^2/tf^3;
    Ydot = 6*(Yf-Yo).*t(i)/tf^2-6*(Yf-Yo).*t(i).^2/tf^3;
PHIdot = 6*(PHIf-PHIo).*t(i)/tf^2-6*(PHIf-PHIo).*t(i).*^2/tf^3;
Xdot = [Xdot;Ydot;PHIdot];
xdot(i) = Xdot(1); ydot(i) = Xdot(2); phidot(i) = Xdot(3); % storing values for plots

Forward Solution: given robot and cartesian rates, find wheel velocities
Qdot = M*Xdot;
qdot1(i) = Qdot(1); qdot2(i) = Qdot(2); qdot3(i) = Qdot(3); qdot4(i) = Qdot(4);
q1(i+1) = q1(i)+qdot1(i)*dt; % integration to find wheel rotation
q2(i+1) = q2(i)+qdot2(i)*dt;
q3(i+1) = q3(i)+qdot3(i)*dt;
q4(i+1) = q4(i)+qdot4(i)*dt;

% animation of robot
xc = [0 X];
yc = [0 Y];
x1 = [X+L*cos(PHI+alp/2) X+L*cos(PHI+alp/2+alp)];
y1 = [Y+L*sin(PHI+alp/2) Y+L*sin(PHI+alp/2+alp)];
x2 = [X+L*cos(PHI+alp/2+alp) X+L*cos(PHI+alp/2+2*alp)];
y2 = [Y+L*sin(PHI+alp/2+alp) Y+L*sin(PHI+alp/2+2*alp)];
x3 = [X+L*cos(PHI+alp/2+2*alp) X+L*cos(PHI+alp/2+3*alp)];
y3 = [Y+L*sin(PHI+alp/2+2*alp) Y+L*sin(PHI+alp/2+3*alp)];
x4 = [X+L*cos(PHI+alp/2+3*alp) X+L*cos(PHI+alp/2)];
y4 = [Y+L*sin(PHI+alp/2+3*alp) Y+L*sin(PHI+alp/2)];
xcoordx = [X X+B*cos(PHI)];
ycoordx = [Y Y+B*sin(PHI)];
xcoordy = [X X+B*cos(PHI+pi/2)];
ycoordy = [Y Y+B*sin(PHI+pi/2)];
xfill = [X+L*cos(PHI+alp/2+alp) X+L*cos(PHI+alp/2+2*alp) X+L*cos(PHI+alp/2+3*alp) X+L*cos(PHI+alp/2)];
yfill = [Y+L*sin(PHI+alp/2+alp) Y+L*sin(PHI+alp/2+2*alp) Y+L*sin(PHI+alp/2+3*alp) Y+L*sin(PHI+alp/2)];

plot(x1,y1,'k',x2,y2,'k',x3,y3,'k',x4,y4,'k',xcoordx,ycoordx,'r',xcoordy,ycoordy,'r',
x,y,'r+',X,Y,'ro');

%patch(xfill,yfill,'b');
patch(X,Y,'ro');
title('litGoalie Position');
xlabel('litX(litm)'); ylabel('litY(litm)');
axis([-1.525/2 1.525/2 0 2.74]);
axis('normal');
axis('equal');
axis('manual');
grid;
pause(1/8);
end

%% PLOTS
% t = [0:5/59:5];
% tout = [0:5/5000:5];
% figure;
% subplot(4,1,1), plot(t,q1(2:59)/DR,tout,q1a/DR), ylabel('#1 (deg)'); grid;
% title('Desired vs Simulated Wheel Position');
% subplot(4,1,2), plot(t,q2(2:59)/DR,tout,(q2a-19.6850)/DR), ylabel('#2'); grid; %
% wheels 2 and 4 already rotated by .5/(.0254*2*pi)*2*pi
% subplot(4,1,3), plot(t,q3(2:59)/DR,tout,q3a/DR), ylabel('#3'); grid;
% subplot(4,1,4), plot(t,q4(2:59)/DR,tout,(q4a+19.6850)/DR), xlabel('time (sec)'),
% ylabel('wheel #4'); grid;

% figure;
% subplot(4,1,1), plot(t,qdot1), ylabel('#1 (rad/s)'); grid;
% title('Wheel Velocities vs Time');
% subplot(4,1,2), plot(t,qdot2), ylabel('#2'); grid;
% subplot(4,1,3), plot(t,qdot3), ylabel('#3'); grid;
% subplot(4,1,4), plot(t,qdot4), xlabel('time (sec)'), ylabel('wheel #4'); grid;

% figure;
% subplot(3,1,1), plot(t,x,tout,xa,'-'), ylabel('X');
% title('Desired vs Simulated Position');
% subplot(3,1,2), plot(t,y,tout,ya,'-'), ylabel('Y');
% subplot(3,1,3), plot(t,phi/DR,tout,phiafDR,'-'), xlabel('time (sec)'), ylabel('PH1 (deg)');
% grid;

% figure;
% subplot(3,1,1), plot(t,xdot), ylabel('Xdot'); grid;
% title('Velocity vs Time');
% subplot(3,1,2), plot(t,ydot), ylabel('Ydot'); grid;
% subplot(3,1,3), plot(t,phidot), xlabel('time (sec)'), ylabel('PH1dot (rad/s)'); grid;

figure;
subplot(3,1,1), plot(t,x), ylabel('X'); grid;
title('Desired vs Simulated Position');
subplot(3,1,2), plot(t,y), ylabel('Y'); grid;
subplot(3,1,3), plot(t,phi/DR), xlabel('time (sec)'), ylabel('PH1 (deg)'); grid;
% Goalie Simulation
% Lance J. Wilson
% Robocup/Thesis

clear; clf;

DR = pi/180;  % degress to radians conversion

B = .18;  % robot dimension
L = sqrt((B/2)^2 + (B/2)^2);  % robot dimension
alp = 90*DR;  % robot dimension

J = [0,1,0,1;...
   1,0,1,0;...
L*cos(alp/2),-L*sin(alp/2),L*cos(alp/2),L*sin(alp/2)];  % Jacobian Matrix
M = J';  % transpose of the jacobian matrix

Xo = input('Enter initial robot position in the x-direction:');
Yo = input('Enter initial robot position in the y-direction:');
THo = input('Enter initial rotation position (deg):');
THo = THo*DR;
Xf = input('Enter final robot position in the x-direction:');
Yf = input('Enter final robot position in the y-direction:');
THf = input('Enter final rotation position (deg):');
THf = THf*DR;
tf = input('Enter time for the robot to move to desired position (sec):');

t0 = 0; dt = .1;
t = [t0:dt:tf];
N = (tf-t0)/dt+1;
[X,Xdot] = smpath(Xo,Yo,THo,Xf,Yf,THf,tf);  % path planning function

Qdot = M*Xdot;

for i = 1:N,
    % storing values for plots
    x(i) = X(1,i); y(i) = X(2,i); th(i) = X(3,i);
    xdot(i) = Xdot(1,i); ydot(i) = Xdot(2,i); thdot(i) = Xdot(3,i);
    qdot1(i) = Qdot(1,i); qdot2(i) = Qdot(2,i); qdot3(i) = Qdot(3,i); qdot4(i) = Qdot(4,i);
    % animation of robot
xc = [0 X(1,i)];
yc = [0 X(2,i)];
x1 = [X(1,i)+L*cos(X(3,i)+alp/2) X(1,i)+L*cos(X(3,i)+alp/2+alp)];
y1 = [X(2,i)+L*sin(X(3,i)+alp/2) X(2,i)+L*sin(X(3,i)+alp/2+alp)];
x2 = [X(1,i)+L*cos(X(3,i)+alp/2+2*alp) X(1,i)+L*cos(X(3,i)+alp/2+2*alp)];
y2 = [X(2,i)+L*sin(X(3,i)+alp/2+2*alp) X(2,i)+L*sin(X(3,i)+alp/2+2*alp)];
x3 = [X(1,i)+L*cos(X(3,i)+alp/2+2*alp) X(1,i)+L*cos(X(3,i)+alp/2+2*alp)];
y3 = [X(2,i)+L*sin(X(3,i)+alp/2+2*alp) X(2,i)+L*sin(X(3,i)+alp/2+2*alp)];
x4 = [X(1,i)+L*cos(X(3,i)+alp/2+3*alp) X(1,i)+L*cos(X(3,i)+alp/2)];
y4 = [X(2,i)+L*sin(X(3,i)+alp/2+3*alp) X(2,i)+L*sin(X(3,i)+alp/2)];
xcoordx = [X(1,i) X(1,i)+B*cos(X(3,i))];
ycoordx = [X(2,i) X(2,i)+B*sin(X(3,i))];
xcoordy = [X(1,i) X(1,i)+B*cos(X(3,i)+pi/2)];
ycoordy = [X(2,i) X(2,i)+B*sin(X(3,i)+pi/2)];
xfill = [X(1,i)+L*cos(X(3,i)+alp/2+2*alp) X(1,i)+L*cos(X(3,i)+alp/2+2*alp)]
X(1,i)+L*cos(X(3,i)+alp/2+3*alp) X(1,i)+L*cos(X(3,i)+alp/2)];
yfill = [X(2,i)+L*sin(X(3,i)+alp/2+2*alp) X(2,i)+L*sin(X(3,i)+alp/2+2*alp)]
X(2,i)+L*sin(X(3,i)+alp/2+3*alp) X(2,i)+L*sin(X(3,i)+alp/2)];

plot(x1,y1,'k',x2,y2,'k',x3,y3,'k',x4,y4,'k',xcoordx,ycoordx,'r',xcoordy,ycoordy,'r',
x,y,'r+',X(1,i),X(2,i),'ro');

 PATCH(X(1),X(2),'ro');

 title('itGoalie Position');
 xtitle('itX(itm)'); ytitle('itY(itm)');
 axis([-1.525/2 1.525/2 0 2.74]);
 axis('normal');
 axis('equal');
 axis('manual');

 grid;
 pause(1/8);
 end


%%% PLOTS

 figure;
 subplot(4,1,1), plot(t,qdot1), ylabel('#1 (rad/s)');
 subplot(4,1,2), plot(t,qdot2), ylabel('#2');
 subplot(4,1,3), plot(t,qdot3), ylabel('#3');
 subplot(4,1,4), plot(t,qdot4), xtitle('time (sec)'), ytitle('wheel #4');

 figure;
 subplot(3,1,1), plot(t,x), ylabel('X');

 subplot(3,1,2), plot(t,y), ylabel('Y');

 subplot(3,1,3), plot(t,qdot), ytitle('Wheel Velocity');

 subplot(3,1,4), plot(t,q), ytitle('Wheel Acceleration');

 subplot(3,1,5), plot(t,qdot2), ytitle('Wheel Acceleration');

 subplot(3,1,6), plot(t,qdot3), ytitle('Wheel Acceleration');

 subplot(3,1,7), plot(t,qdot4), ytitle('Wheel Acceleration');

 figure;
 subplot(3,1,1), plot(t,x), ylabel('X');
 subplot(3,1,2), plot(t,y), ylabel('Y');
 subplot(3,1,3), plot(t,q), ytitle('Wheel Velocity');
 subplot(3,1,4), plot(t,qdot), ytitle('Wheel Acceleration');
 subplot(3,1,5), plot(t,qdot2), ytitle('Wheel Acceleration');
 subplot(3,1,6), plot(t,qdot3), ytitle('Wheel Acceleration');
 subplot(3,1,7), plot(t,qdot4), ytitle('Wheel Acceleration');
title('Position vs Time');
subplot(3,1,2), plot(t,y), ylabel('Y'); grid;
subplot(3,1,3), plot(t,th/DR), xlabel('time (sec)'), ylabel('TH (deg)'); grid;

figure;
subplot(3,1,1), plot(t,xdot), ylabel('Xdot'); grid;
title('Velocity vs Time');
subplot(3,1,2), plot(t,ydot), ylabel('Ydot'); grid;
subplot(3,1,3), plot(t,thdot), xlabel('time (sec)'), ylabel('THdot (rad/s)'); grid;
clc; clear;

% symbol declaration
syms L1 L2

B = k/r*[0,-1,0,1;...
    1,0,-1,0;...
    L1,L2,L1,L2];
Bt = k/r*[0,1,L1;...
       -1,0,L2;...
       0,-1,L1;...
       1,0,L2];
Bpinv = Bt*inv(B*Bt);  % Pseudo Inverse, Minimum Norm
Bpinv = simple(Bpinv);
pretty(Bpinv)
clc; clear;

% symbol declaration
syms L1 L2 L3 L4

% Generic Case
Q = [0, -1, 0, 1; ...]; % Jacobian
  1, 0, -1, 0; ...
  L1, L2, L3, L4];
Qt = [0, 1, L1; ...]; % Jacobian Transpose
  -1, 0, L2; ...
  0, -1, L3; ...
  1, 0, L4];
Qtpinv = inv(Q*Qt)*Q; % Pseudoinverse, least squares error
Qtpinv = simple(Qtpinv);
pretty(Qtpinv) % multiply by 'r'

% Simplified Case L1 = L3 & L2 = L4
Q = [0, -1, 0, 1; ...]; % Jacobian
  1, 0, -1, 0; ...
  L1, L2, L1, L2];
Qt = [0, 1, L1; ...]; % Jacobian Transpose
  -1, 0, L2; ...
  0, -1, L1; ...
  1, 0, L2];
Qtpinv = inv(Q*Qt)*Q; % Pseudoinverse
Qtpinv = simple(Qtpinv);
pretty(Qtpinv) % multiply by 'r'
% Lance Wilson
% Symbolic Dynamic EOMs for the 3 dof Redundant Holonomic Mobile Robot wrt
Global Vision

clc; clear;

syms L1 L2 L3 L4 r t k c lw u1 u2 u3 u4 M Im
X = sym('x(t)');
Y = sym('y(t)');
phi = sym('phi(t)');
Xd = diff(X,t);
Yd = diff(Y,t);
phid = diff(phi,t);
Xdd = diff(Xd,t);
Ydd = diff(Yd,t);
phidd = diff(phid,t);

% Kinematics
Qt = 1/r*[0,1,L1;...
    -1,0,L2;...
    0,-1,L3;...
    1,0,L4];
Rt = [cos(phi),sin(phi),0;...
    -sin(phi),cos(phi),0;...
    0,0,1];
Sw = [Xd;Yd;phid];
qdot = Qt*Rt*Sw;
qddot = diff(qdot,t);

% Wheel Dynamics
T1 = (k*u1-c*qdot(1)-lw*qddot(1))/r;
T2 = (k*u2-c*qdot(2)-lw*qddot(2))/r;
T3 = (k*u3-c*qdot(3)-lw*qddot(3))/r;
T4 = (k*u4-c*qdot(4)-lw*qddot(4))/r;

% Robot Frame Dynamics
fx = -T2+T4;
fy = T1-T3;
Mg = T1*L1+T2*L2+T3*L3+T4*L4;
fm = [fx;fy;Mg];
R = [cos(phi),-sin(phi),0;...
    sin(phi),cos(phi),0;...
    0,0,1];
% World Frame Dynamics
Fw = R*fm;

% Equations of Motion!
EOM1 = M*(Xdd)-fx;
EOM1 = simple(EOM1);
EOM1 = collect(EOM1,r);
EOM1 = collect(EOM1,c);
EOM1 = collect(EOM1,k);
EOM1 = collect(EOM1,lw);
EOM1 = collect(EOM1,M);
EOM1 = collect(EOM1,phid);
EOM1 = collect(EOM1,Yd);
EOM1 = collect(EOM1,Xd);
EOM1 = collect(EOM1,phidd);
EOM1 = collect(EOM1,Ydd);
EOM1 = collect(EOM1,Xdd);
pretty(EOM1)

EOM2 = M*(Ydd)-fy;
EOM2 = simple(EOM2);
EOM2 = collect(EOM2,r);
EOM2 = collect(EOM2,c);
EOM2 = collect(EOM2,k);
EOM2 = collect(EOM2,lw);
EOM2 = collect(EOM2,M);
EOM2 = collect(EOM2,Yd);
EOM2 = collect(EOM2,Xd);
EOM2 = collect(EOM2,phid);
EOM2 = collect(EOM2,phidd);
EOM2 = collect(EOM2,Ydd);
EOM2 = collect(EOM2,Xdd);
pretty(EOM2)

EOM3 = I*m*(phidd)-Mg;
EOM3 = simple(EOM3);
EOM3 = collect(EOM3,r);
EOM3 = collect(EOM3,c);
EOM3 = collect(EOM3,k);
EOM3 = collect(EOM3,lw);
EOM3 = collect(EOM3,M);
EOM3 = collect(EOM3,phid);
EOM3 = collect(EOM3,Yd);
EOM3 = collect(EOM3,Xd);
EOM3 = collect(EOM3,phidd);  
EOM3 = collect(EOM3,Ydd);  
EOM3 = collect(EOM3,Xdd);  
pretty(EOM3)
clc; clear;

% symbol declaration
syms t L1 L2 L3 L4 alpi k c u1 u2 u3 u4 Iw Im M
x = sym('x(t)');
y = sym('y(t)');
phi = sym('phi(t)');
w1 = sym('w1(t)');
w2 = sym('w2(t)');
w3 = sym('w3(t)');
w4 = sym('w4(t)');

Dx = diff(x,t);
D2x = diff(Dx,t);
Dy = diff(y,t);
D2y = diff(Dy,t);
Dphi = diff(phi,t);
D2phi = diff(Dphi,t);

% solving for angular wheel rates from inverse kinematics
w1 = solve('r*w1=Dy+Dphi*L1');
w1 = subs(w1);
w2 = solve('r*w2=-Dx+Dphi*L2');
w2 = subs(w2);
w3 = solve('r*w3=-Dy+Dphi*L3');
w3 = subs(w3);
w4 = solve('r*w4=Dx+Dphi*L4');
w4 = subs(w4);

% change in angular wheel rates
Dw1 = diff(w1,t);
Dw2 = diff(w2,t);
Dw3 = diff(w3,t);
Dw4 = diff(w4,t);

% Wheel Dynamics
T1 = (k*u1-c*w1-Iw*Dw1)/r;
T2 = (k*u2-c*w2-Iw*Dw2)/r;
T3 = (k*u3-c*w3-Iw*Dw3)/r;
T4 = (k*u4-c*w4-Iw*Dw4)/r;
Robot Dynamics

\[ f_x = -T_2 + T_4; \]
\[ f_y = T_1 - T_3; \]
\[ M_g = T_1 * L_1 + T_2 * L_2 + T_3 * L_3 + T_4 * L_4; \]

Equations of Motion!

\[ \text{EOM} 1 = M \left( D^2 x - D_y D \phi \right) - f_x; \]
\[ \text{EOM} 1 = \text{simple}(\text{EOM} 1); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, r); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, c); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, k); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, lw); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, M); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, Dphi); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, Dy); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, Dx); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, D2phi); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, D2y); \]
\[ \text{EOM} 1 = \text{collect}(\text{EOM} 1, D2x); \]
\[ \text{pretty}(\text{EOM} 1); \]

\[ \text{EOM} 2 = M \left( D^2 y + D_x D \phi \right) - f_y; \]
\[ \text{EOM} 2 = \text{simple}(\text{EOM} 2); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, r); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, c); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, k); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, lw); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, M); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, Dphi); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, Dy); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, Dx); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, D2phi); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, D2y); \]
\[ \text{EOM} 2 = \text{collect}(\text{EOM} 2, D2x); \]
\[ \text{pretty}(\text{EOM} 2); \]

\[ \text{EOM} 3 = I_m D^2 \phi \text{-} M_g; \]
\[ \text{EOM} 3 = \text{simple}(\text{EOM} 3); \]
\[ \text{EOM} 3 = \text{collect}(\text{EOM} 3, I_m); \]
\[ \text{EOM} 3 = \text{collect}(\text{EOM} 3, lw); \]
\[ \text{EOM} 3 = \text{collect}(\text{EOM} 3, c); \]
\[ \text{EOM} 3 = \text{collect}(\text{EOM} 3, r^2); \]
\[ \text{EOM} 3 = \text{collect}(\text{EOM} 3, k); \]
\[ \text{EOM} 3 = \text{collect}(\text{EOM} 3, r); \]
EOM3 = collect(EOM3,Dphi);
EOM3 = collect(EOM3,Dy);
EOM3 = collect(EOM3,Dx);
EOM3 = collect(EOM3,D2phi);
EOM3 = collect(EOM3,D2y);
EOM3 = collect(EOM3,D2x);
pretty(EOM3)
% Lance Wilson
% Simplified Equations for use in DEE and Computed Torque Control Scheme, wrt
Joint Frame

clear;

%syms t L1 L2 r k c u1 u2 u3 u4 Iw Im M
%syms x y pd
%qd1 = sym('x(5)'); qd2 = sym('x(6)'); qd3 = sym('x(7)'); qd4 = sym('x(8)');
%qd1 = sym('x5'); qd2 = sym('x6'); qd3 = sym('x7'); qd4 = sym('x8');
qu = sym('u(1)'); qd2 = sym('u(2)'); qd3 = sym('u(3)'); qd4 = sym('u(4)');
%syms u1 u2 u3 u4
L1 = .1; % m
L2 = .1; % m
r = .0254; % m
k = 1; % gear ratio
c = .0113; % viscous friction factor N/m
Iw = 1.46*10^-5; % kg-m^2 wheel inertia
Im = .013114; % kg-m^2 robot inertia
M = 2.26; % kg robot mass
n = 1; % gear ratio
Jm = 2.7003; % N-m robot inertia
b = 0; % motor friction
Kt = .014497; % N-m/A torque constant

P = [2*Iw/r^2+M,0,0,...
    0,2*Iw/r^2+M,0,...
    0,0,1w/r^2*(2*L1^2+2*L2^2)+Im];

% Simplified Case L1 = L3 & L2 = L4
Q = 1/r*[0,-1,0,1;...
    1,0,-1,0;...
    L1,L2,L1,L2];
Qt = 1/r*[0,1,L1;...
    -1,0,L2;...
    0,-1,L1;...
    1,0,L2];
Qtpinv = inv(Q*Qt)*Q; % Pseudo Inverse, least squares

% kinematics
Smd = Qtpinv*[qd1,qd2;qd3;qd4];
xd = Smd(1);
yd = Smd(2);
pd = Smd(3);
\[
Nq = [-M*yd*pd+2*c/r^2*xd;...
M*xd*pd+2*c/r^2*yd;...
c/r^2*(2*L1^2+2*L2^2)*pd];
\]

\[
B = k/r*[0,-1,0,1;...
1,0,-1,0;...
L1,L2,L1,L2];
Bt = k/r*[0,1,L1;...
-1,0,L2;...
0,-1,L1;...
1,0,L2];
Bpinv = Bt*inv(B*Bt); % Psuedo Inverse, Minimum Norm
\]

% matrices for control
% alpha
alpha = 1/(n*Kt)*(Bpinv*P*Qtpinv+n^2*Jm*eye(4))
% beta
qd = [qd1;qd2;qd3;qd4];
beta = n*b/Kt*qd+1/(n*Kt)*Bpinv*Nq

% equations for DEE
U = [u1;u2;u3;u4];
Ppinv = (P*Qtpinv+n^2*B*Jm*eye(4));
Ppinv = Ppinv*inv(Ppinv*Ppinv);
sdd = Ppinv*(n*Kt*B*U-Nq-n^2*b*B*qd)
Lance Wilson
Simplified Equations for use in DEE and Computed Torque Control Scheme, wrt Local Vision

clc; clear;
% symstLl L2r k c u1 u2 u3 u4 lW lm M
% symst xd yd pd u1 u2 u3 u4
xd = sym('x(4)'); yd = sym('x(5)'); pd = sym('x(6)');
% xd = sym('u(1)'); yd = sym('u(2)'); pd = sym('u(3)');
u1 = sym('u(1)'); u2 = sym('u(2)'); u3 = sym('u(3)'); u4 = sym('u(4)');
% symst u1 u2 u3 u4
L1 = .1; % m
L2 = .1; % m
r = .0254; % m
k = 1; % gear ratio
c = .0113; % viscous friction factor N/m
lW = 1.46*10^-5; % kg-m^2 wheel inertia
lm = .013114; % kg-m^2 robot inertia
M = 2.26; % kg robot mass
n = 1; % gear ratio
Jm = 2.7003; % N-m motor inertia
b = 0; % motor friction
Kt = .014497; % N-m/A torque constant

P = [2*Iw/r^2+M,0,0;...
    0,2*Iw/r^2+M,0;...
    0,0,Iw/r^2*(2*L1^2+2*L2^2)+lm];

N = [-M*yd*pd+2*c/r^2*xd;...
    M*xd*pd+2*c/r^2*yd;...
    c/r^2*(2*L1^2+2*L2^2)*pd];

B = k/r*[0,-1,0,1;...
      1,0,-1,0;...
      L1,L2,L1,L2];
Bt = k/r*[0,1,L1;...
      -1,0,L2;...
      0,-1,L1;...
      1,0,L2];
Bpinv = Bt*inv(B*Bt); % Psuedo Inverse, Minimum Norm

% Simplified Case L1 = L3 & L2 = L4
Q = 1/r*[0,-1,0,1;... % Jacobian
      1,0,-1,0;...
L1,L2,L1,L2];
Qt = 1/r*[0,1,L1;...  % Jacobian Transpose
     -1,0,L2;...
     0,-1,L1;...
     1,0,L2];
Qtpinv = inv(Q*Qt)*Q;  % Psuedo Inverse, least squares

% matrices for control
% alpha
%alpha = Bpinv*P
% beta
%beta = Bpinv*N

sd = [xd;yd;pd];

% equations for DEE
U = [u1;u2;u3;u4];
Ppinv = (P+n^2*B*Jm*eye(4)*Qt);
Ppinv = Ppinv*inv(Ppinv*Ppinv');
sdd = Ppinv*(n*Kt*B*U-N-n^2*b*B*Qt*sd)
clc; clear;
% syms t L1 L2 r k c u1 u2 u3 u4 Iw Im M phi
% syms xd yd pd u1 u2 u3 u4
% xd = sym('x(4)'); yd = sym('x(5)'); pd = sym('x(6)');
% u1 = sym('u(1)'); u2 = sym('u(2)'); u3 = sym('u(3)'); u4 = sym('u(4)');
% syms u1 u2 u3 u4
L1 = .1; % m
L2 = .1; % m
r = .0254; % m
k = 1; % gear ratio
c = .0113; % viscous friction factor N/m
Iw = 1.46*10^-5; % kg-m^2 wheel inertia
Im = .013114; % kg-m^2 robot inertia
M = 2.26; % kg robot mass
phi = 0; % radian angle

B = k/r*[0,-1,0,1;1,0,-1,0;L1,L2,L1,L2];
Bt = k/r*[0,1,L1;1,0,-1,L1;0,-1,L1;1,0,L2];
Bpinv = Bt*inv(B*Bt); % Psuedo Inverse, Minimum Norm
%Bpinv = simple(Bpinv);

P = [2*Iw/r^2*cos(phi)+M,2*Iw/r^2*sin(phi),0,...
    -2*Iw/r^2*sin(phi),2*Iw/r^2*cos(phi)+M,0,...
    0,0,Iw/r^2*(2*L1^2+2*L2^2)+Im];

N = [(2*c/r^2*cos(phi)-
    2*Iw/r^2*p*d*sin(phi))*xd+(2*c/r^2*sin(phi)+2*Iw/r^2*p*d*cos(phi))*yd;...
    (-2*c/r^2*sin(phi)-
    2*Iw/r^2*p*d*cos(phi))*xd+(2*c/r^2*cos(phi)+2*Iw/r^2*p*d*sin(phi))*yd;...
    c/r^2*(2*L1^2+2*L2^2)*pd];
% beta
beta = Bpinv*N

% equations for DEE
U = [u1;u2,u3,u4];
sdd = inv(P)*(-N)+inv(P)*B*U
clc; clear;
%syms t L1 L2 r k c u1 u2 u3 u4 Iw Im M
%syms xd yd pd
%qd1 = sym('x(5)'); qd2 = sym('x(6)'); qd3 = sym('x(7)'); qd4 = sym('x(8)');
%qd1 = sym('x5'); qd2 = sym('x6'); qd3 = sym('x7'); qd4 = sym('x8');
qd1 = sym('u(1)'); qd2 = sym('u(2)'); qd3 = sym('u(3)'); qd4 = sym('u(4)');
%u1 = sym('u(1)'); u2 = sym('u(2)'); u3 = sym('u(3)'); u4 = sym('u(4)');
%syms u1 u2 u3 u4
L1 = .1; % m
L2 = .1; % m
r = .0254; % m
k = 1; % gear ratio
c = .0113; % viscous friction fator N/m
Iw = 1.46*10^-5; % kg-m^2 wheel inertia
Im = .013114; % kg-m^2 robot inertia
M = 2.26; % kg robot mass
P = [2*Iw/r^2+M,0,0,...
 0,2*Iw/r^2+M,0,;...
 0,0,Iw/r^2*(2*L1^2+2*L2^2)+Im];

% Simplified Case L1 = L3 & L2 = L4
Q = 1/r*[0,-1,0,1;...
 1,0,-1,0;...
 L1,L2,L1,L2];
Qt = 1/r*[0,1,L1;...
 -1,0,L2;...
 0,-1,L1;...
 1,0,L2];
Qtpinv = inv(Q*Qt)*Q;% Psuedo Inverse, least squares

% kinematics
Smd = Qtpinv*[qd1;qd2;qd3;qd4];
xd = Smd(1);
yd = Smd(2);
pd = Smd(3);
Nq = [-M*yd*pd+2*c/r^2*xd;...
  M*xd*pd+2*c/r^2*yd;...
  c/r^2*(2*L1^2+2*L2^2)*pd];
\[ B = k/r*[0,-1,0,1;...\]
\[ 1,0,-1,0;...
\[ L1,L2,L1,L2]\];
\[ Bt = k/r*[0,1,L1;...\]
\[ -1,0,L2;...
\[ 0,-1,L1;...
\[ 1,0,L2]\];
\[ Bpinv = Bt*inv(B*Bt); \% Psuedo Inverse, Minimum Norm \]

\% matrices for control
\% alpha
\% beta
\% equations for DEE
U = [u1,u2,u3,u4];
sdd = Bq*U-Nq
Lance Wilson

Simplified Equations

clc; clear;
syms t L1 L2 r k c u1 u2 u3 u4 Iw Im M

syms xd yd pd qd1 qd2 qd3 qd4

\[
P = \begin{bmatrix} 2*Iw/r^2+M,0,0; \\ 0,2*Iw/r^2+M,0; \\ 0,0,Iw/r^2*(2*L1^2+2*L2^2)+Im \end{bmatrix};
\]

% Simplified Case L1 = L3 & L2 = L4

\[
Q = 1/r*[0,-1,0,1; \\ 1,0,-1,0; \\ L1,L2,L1,L2];
\]

\[
Qt = 1/r*[0,1,L1; \\ -1,0,L2; \\ 0,-1,L1; \\ 1,0,L2];
\]

\[
Qtpinv = inv(Q*Qt)*Q; \quad \text{% Pseudo Inverse, least squares}
\]

% kinematics

\[
Smd = Qtpinv*[qd1;qd2;qd3;qd4];
\]

\[
xd = Smd(1);
\]

\[
yd = Smd(2);
\]

\[
pd = Smd(3);
\]

\[
Ns = [-M*yd*pd+2*c/r^2*xd; \\ M*xd*pd+2*c/r^2*yd; \\ c/r^2*(2*L1^2+2*L2^2)*pd];
\]

\[
Nq = Qt*inv(P)*Ns;
\]

\[
B = k/r*[0,-1,0,1; \\ 1,0,-1,0; \\ L1,L2,L1,L2];
\]

\[
Bq = Qt*inv(P)*B;
\]
% Lance Wilson
% Simplified Equations for use in DEE and Computed Torque Control Scheme, wrt
  Local Vision

clc; clear;
%syms t L1 L2 r k c u1 u2 u3 u4 Iw Im M
%syms xd yd pd ul u2 u3 u4
%xd = sym('x(4)'); yd = sym('x(5)'); pd = sym('x(6)');
%u1 = sym('u(1)'); u2 = sym('u(2)'); u3 = sym('u(3)'); u4 = sym('u(4)');
L1 = .1; % m
L2 = .1; % m
r = .0254; % m
k = 1; % gear ratio
C = .0113; % viscous friction factor N/m
Iw = 1.46*10^-5; % kg*m^2 wheel inertia
Im = .013114; % kg*m^2 robot inertia
M = 2.26; % kg robot mass
P = [2*Iw/r^2+M,0,0;...
0,2*Iw/r^2+M,0;...
0,0,Iw/r^2*(2*L1^2+2*L2^2)+Im];

N = [-M*yd*pd+2*c/r^2*xd;...
M*xd*pd+2*c/r^2*yd;...
c/r^2*(2*L1^2+2*L2^2)*pd];

B = k/r*[0,-1,0,1;...
1,0,-1,0;...]
  L1,L2,L1,L2];
Bt = k/r*[0,1,L1;...
-1,0,L2;...
0,-1,L1;...
1,0,L2];
Bp Inv = Bt*inv(B*Bt); % Pseudo Inverse, Minimum Norm
%Bp Inv = simple(Bp Inv);

% matrices for control
% alpha
alpha = Bp Inv*P
% beta
beta = Bp Inv*N
% equations for DEE
U = [u1;u2;u3;u4];
sdd = inv(P)*(-N)+inv(P)*B*U