Development of Empirical and Virtual Tools for the Study of Bicycle Safety

Brent Kostich
DEVELOPMENT OF EMPIRICAL AND VIRTUAL TOOLS FOR THE STUDY OF BICYCLE SAFETY

by

Brent W. Kostich

A thesis submitted to the Graduate College in partial fulfillment of the requirements for the degree of Master of Science in Engineering Mechanical and Aerospace Engineering Western Michigan University August 2017

Thesis Committee:

Kaposeong Ro, Ph.D., Chair
Jennifer Hudson, Ph.D.
Jun-Seok Oh, Ph.D.
Bicycles have been used as a form of non-motorized transportation for several hundred years. Recently, their use in the transportation realm has been reinvigorated because of growing population densities and efforts to improve environmental sustainability. However, a large deterrent for public use of the bicycle is concern over its level of safety in the transportation environment. The goal of this thesis is to develop two complementary research tools that advance the study of bicycle safety factors.

First, an instrumented probe bicycle (IPB) is constructed from the ground up. The bicycle is outfitted with an array of sensors that are integrated into a unified instrumentation system. The design presented herein describes the components and capabilities of this IPB, as well as its limitation. Since the study of transportation safety often requires the identification of high-risk areas, it is necessary to classify data by geographic location. A geographic segmenting algorithm is formulated to complement the IPB system. Both the instrumented bicycle and the algorithm are evaluated from pilot testing and found to perform satisfactorily.

Second, a computation dynamic model is developed to analyze bicycle motion. Despite being a simple system, the bicycle has relatively complex dynamics. Understanding these dynamics is useful in assessing situational rider safety. To generate the model, differential equations of motion are derived from first principles. These equations are developed into a computer program for numerical integration. Model validation is performed against previous work and found to be accurate.
ACKNOWLEDGEMENTS

In the words of the poet John Donne, “No Man is an Island”. I find this to be irrevocably true, so I would like to recognize a few (of the many) who have contributed a portion of their life to my journey.

First, I would like to thank the members of my committee. I am grateful to my advisor, Dr. Kapseong Ro, and Dr. Jun-Seok Oh for allowing me to take part in their work, and exposing me to the fascinating world of bicycles. I would further like to thank Dr. Ro for his instruction in developing my MATLAB programming skills. Without his guidance over the years, the programs presented here would not exist. Thanks to Dr. Jennifer Hudson, whose patience and encouragement helped me keep a level head throughout this process.

Next, I would like to thank two people who were most helpful in critical areas of this work, Eugene Zimmerman and Dr. James Kamman. I would like to thank Eugene for his superb machining skills and willingness to teach his craft. Without his direction, the design and fabrication of the bicycle components would have suffered greatly. I would like to thank Dr. Kamman, first for exposing me to the vast and captivating world of dynamics, and second for his tireless efforts in helping me clear the final hurdles of this thesis. These efforts were paramount in the finding and correcting of my errors, and in helping me finally achieve a working computational model. Furthermore, if there is any measure of clarity in this work it is due, in large part, to his help in the preparation of this document.

Finally, I would like to thank a few members of my family. First, thanks to my parents, Kirt and Claudia, for their encouragement over the years. Their steady support has allowed me to push the limits of my knowledge and understanding. Next, thanks to my grandparents, William and Ruth, for their undying love and support. Over the years their example has given me reason to strive for excellence in all I do, knowing full well that no matter the outcome they would be proud.

Brent W. Kostich
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ............................................................................................................. ii

LIST OF TABLES ....................................................................................................................... vii

LIST OF FIGURES ..................................................................................................................... viii

1. INTRODUCTION .................................................................................................................... 1

   1.1 Project Motivation ........................................................................................................... 1

   1.2 Problem Statement ........................................................................................................... 1

   1.3 Definition of Terms .......................................................................................................... 2

2. INSTRUMENT PROBE BICYCLE ....................................................................................... 3

   2.1 Review of IPB Literature ................................................................................................. 3

   2.2 The Instrumented Bicycle ............................................................................................... 4

       2.2.1 Data Acquisition System ...................................................................................... 7

       2.2.2 IPB Sensors ............................................................................................................. 9

       2.2.3 Sensor Calibration ................................................................................................. 16

   2.3 Data Acquisition ........................................................................................................... 18

       2.3.1 Data Recording Software ...................................................................................... 18

       2.3.2 LabVIEW Program ............................................................................................... 19

       2.3.3 Challenges and Limitations ................................................................................... 22

   2.4 Data Processing .............................................................................................................. 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.1</td>
<td>Raw Data Pre-Processing</td>
<td>23</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Geographic Segmenting Algorithm</td>
<td>25</td>
</tr>
<tr>
<td>2.5</td>
<td>Results</td>
<td>27</td>
</tr>
<tr>
<td>2.5.1</td>
<td>IPB Systems</td>
<td>27</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Segmenting Algorithm Output</td>
<td>28</td>
</tr>
<tr>
<td>2.6</td>
<td>Conclusions and Future Work</td>
<td>30</td>
</tr>
<tr>
<td>3.1</td>
<td>Review of Bicycle Modeling Literature</td>
<td>31</td>
</tr>
<tr>
<td>3.2</td>
<td>Bicycle Model Description</td>
<td>32</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Overview and Focus</td>
<td>32</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Bicycle Configuration</td>
<td>33</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Notation</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Kinematics</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Rotation Matrices</td>
<td>36</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Position Vectors</td>
<td>38</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Kinematical Differential Equations</td>
<td>40</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Generalized Speeds</td>
<td>40</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Angular Velocities</td>
<td>41</td>
</tr>
<tr>
<td>3.3.6</td>
<td>Point Velocities</td>
<td>43</td>
</tr>
<tr>
<td>3.3.7</td>
<td>Motion Constraints</td>
<td>47</td>
</tr>
<tr>
<td>3.3.8</td>
<td>Angular Accelerations</td>
<td>50</td>
</tr>
</tbody>
</table>
Table of Contents—Continued

3.3.9 Point Accelerations ........................................................................................................ 51
3.4 Kinetics .......................................................................................................................... 52
  3.4.1 Mass Properties ..................................................................................................... 52
  3.4.2 Inertia Forces ........................................................................................................ 53
  3.4.3 Active Forces ........................................................................................................ 54
  3.4.4 Kane’s Equations .................................................................................................. 56
3.5 Computer Program ......................................................................................................... 57
  3.5.1 Symbolic Equation Build-up ................................................................................. 57
  3.5.2 Simulation ............................................................................................................. 59
  3.5.3 Output and Validation ........................................................................................... 64
3.6 Results ............................................................................................................................ 65
3.7 Conclusions and Future Work ....................................................................................... 69

REFERENCES ............................................................................................................................ 71

APPENDIX A: Data Sheets........................................................................................................ 73

APPENDIX B: LabVIEW Block Diagram of Data Acquisition Program................................. 77

APPENDIX C: MATLAB Program for Geographic Sorting Algorithm ................................. 78

APPENDIX D: Derivatives of the C Matrices ........................................................................ 85

APPENDIX E: Angular Acceleration Terms ............................................................................ 90

APPENDIX F: Point Acceleration Terms .............................................................................. 92
APPENDIX G: Symbolic Equations Script and Function Files ........................................... 99

APPENDIX H: Simulation Main File and Function Files .................................................. 114
LIST OF TABLES

1 Several common bicycle geometric parameters .......................................................... 5
2 IPB Sensor Array ........................................................................................................ 8
3 Overview of accuracy properties of each sensor ......................................................... 18
4 Bicycle geometry parameters ..................................................................................... 60
5 Bicycle mass parameters ........................................................................................... 60
6 Geometric parameters of current model .................................................................... 62
# LIST OF FIGURES

1. Dimensional drawing of common bicycle geometric parameters ........................................... 6
2. Completed Instrumented Probe Bicycle .................................................................................. 8
3. Close view of IMU and mounting hardware ......................................................................... 10
4. Rider position sensor ........................................................................................................... 11
5. Rider position sensor with sensing rod in place ................................................................. 12
6. Rider sensor in use during pilot testing .............................................................................. 12
7. Two views of aluminum sensing disk and mounting hardware ........................................... 13
8. Front wheel encoder and mounting brackets ...................................................................... 14
9. Two views of steering angle sensor ................................................................................... 15
10. DAQ box components ....................................................................................................... 16
11. Example of sbgCenter user interface ................................................................................ 19
12. LabVIEW front panel for IPB data collection system ......................................................... 22
13. Example output of synchronized bicycle velocity profiles ............................................... 24
14. Segmenting algorithm flowchart ....................................................................................... 26
15. Graphical user interface for segmenting algorithm ............................................................ 27
16. Route segmenting results using only distance points ....................................................... 29
17. Route segmenting results using full algorithm ................................................................ 29
18. Arbitrary configuration of the bicycle in three-dimensional space .................................... 33
19. Configuration of the bicycle with the front wheel touching the ground............................ 34
List of Figures—Continued

20 Planar view of the bicycle in its nominal configuration .......................................................... 34
21 Geometry used for parameter conversion .............................................................................. 61
22 Validation plot of bicycle dynamic response ......................................................................... 64
23 Roll rate, steer rate and forward velocity for lower unstable region (3.5 m/s) ..................... 66
24 Roll angle of bicycle in lower unstable region (3.5 m/s) ................................................... 67
25 Yaw angle of the bicycle in the lower unstable region (3.5 m/s) .................................... 67
26 Roll rate, steer rate and forward velocity for upper unstable region (8.0 m/s) .............. 68
27 Roll angle for upper unstable region (8.0 m/s) ...................................................................... 69
INTRODUCTION

1.1 Project Motivation

The inspiration for this project was born from the Transportation Research Center for Livable Communities (TRCLC). The TRCLC is a consortium of research universities whose mission is to study issues surrounding public transportation, for the improvement of infrastructure and the promotion of safety. In recent years there has been a strong push towards advancing cycling as a primary means of non-motorized transportation. Rising population densities and social sentiments toward sustainability are two of these driving forces. An increasing number of individuals consider this alternative form of transportation, but many are deterred when faced with the challenges of safety and accessibility. To recommend a solution, it is first necessary to identify the details of the problem. It is the objective of this work to develop the tools that enable such research to take place.

1.2 Problem Statement

Currently, there are no dedicated instruments for this field of research. Researchers must develop custom instruments (suitable to their needs) to study cycling transportation. This thesis aims to develop a set of these instruments for the TRCLC, and is divided into two main parts. First, an operational instrument probe bicycle (IPB) is designed, built, and tested. The IPB is a completely mobile system capable of collecting a variety of empirical data. Then, a computational dynamic model of the bicycle is developed from first principles. The model can be used to study bicycle dynamics, including rider/bicycle interaction, empirical data validation, and interactive simulator feedback.

For the first round of research the TRCLC chose to specifically study rider comfortability and rider control motion associated with on-road conditions. Rider comfortability is a metric that rates a rider’s perception of confidence for given roadway situations. This helps identify the
suitability of roadway elements by determining what features contribute to negative perceptions toward rider safety. Rider control motions, which consist of body lean and steering input, help differentiate riders of different skill level. These can be used to supplement the comfortability metric by weighting its results against skill level. The exact methodology and results of that research are tangential to the focus of this thesis. However, it is sufficient to say the IPB is useful for collecting the necessary data and the computational dynamic model for analyzing the control motions. The exact methodology and results of that research are tangential to the focus of this thesis.

1.3 Definition of Terms

To promote clarity, this section will define the meaning behind an IPB and a computational dynamic model. An IPB is any bicycle that has been modified with an instrumentation system. These systems vary greatly between IPBs based on the type of data being collected. Data collection falls into two basic categories, internal bicycle data and external bicycle data. Internal data are states of the bicycle (like its velocity or acceleration) or direct influences on the bicycle (like applied forces or torques). External data are factors that indirectly influence the motion of the bicycle and its rider, such as objects within the immediate surroundings. Most IPBs have fully mobile instrumentation systems that measure and record data directly on the bicycle. This allows for independence in motion and a large range of travel.

Computational dynamic modeling is a broad topic and is accomplished by a variety of techniques. As it relates to this thesis, the computational model will refer to the numerical evaluation of dynamical equations of motion. The computer simulation routines developed in this thesis are done so using the well-known engineering program MATLAB [1]. The simulation routines are a combination of script and function files. They allow a user to enter physical parameters and test conditions. The output is a time-history of the bicycle’s states. The propagation of these states provide insight to the physical motion of the system.
CHAPTER 2

INSTRUMENT PROBE BICYCLE

2.1 Review of IPB Literature

A number of IPBs have been built in recent years, most of which have common attributes and are used for similar purposes. However, as of now there is no standard design that researchers follow in the construction of an IPB. A goal of this project is to contribute to the process of standardizing the IPB and its systems. Standardization would allow future researchers to better compare data. To that end, a variety of IPBs are examined here to note their choice of sensors and system design. While this list is far from comprehensive, it captures features that are common among instrumented bicycles.

First, Mohanty et. al. provide a well-packed review of IPB technology, as part of the International Cycling and Safety Conference, held in 2014 [2] The direct aim of their paper is to present strategic goals for the standardization of IPB technology and collaborative IPB research. This paper is a great reference for those new to IPB research, and this thesis directly benefits from the author’s efforts and suggestions.

In 2012, Zhang, Chen, and Yi conducted a study to expand the capabilities of mobile data acquisition and study of bicycle/rider interaction [3]. The measurements of the multi-contact interaction between a rider and a bicycle is what they call pose estimation. They used an IPB equipped with two inertial measurement units (IMUs), a seat force sensor, strain gauges (on the handle bars), and an optical encoder (mounted on the rear wheel). One IMU was rigidly mounted to the bicycle frame, while the other was attached to the back of the rider. Their seat force sensor is a custom-built seat that made use of several load cells. They conducted indoor experimental trials, which enabled them to use a motion capture system to track the rider’s movements. This study helps illustrate the power of IMUs in gathering very meaningful data about rider and bicycle motion.

Another example of the usefulness of IMU technology was presented by Joo and Oh [4]. They developed a novel method of assessing the quality of a bicycling environment. This method weighed the overall contributions of different road surface grades and road features (e.g. speed
bumps, curbs, and inclines) to cycling quality. A GPS receiver was used to track the location of the bicycle during their experimental runs. This position data helped tie specific bicycle motions, like obstacle avoidance, to exact geographic locations.

One of the more advanced IPBs developed in recent years was by Dozza and Fernandez, in their 2012 study of naturalistic cyclist behavior [5]. Their research was directed toward the useful implementation of intelligent bicycle systems for everyday riders. Their bicycle is outfitted with two IMUs, one on the bicycle frame, and one on the front fork assembly. It has two cameras, one forward-facing and one facing the rider. Lastly, their bicycle has brake force sensors that monitor the pressure in the brake pads, for each wheel. The most impressive feature of this IPB is the data recording system. It is fully integrated onto the bicycle, and had its own start-up and shut-down routines that required almost no human input. The software and microcontrollers allow for motion recognition and automatic initialization of the sensor array. All data is sent to a microcomputer, mounted on the rear rack of the bicycle and housed in a waterproof case. This allows for a very compact and functional system. These advanced protocols allow the bicycle to be used in an efficient manner, and make successful data acquisition a consistent venture.

In 2014, Lee et. al. added their contributions to IPB technology by conducting a study on predictive bicycle safety and comfort models [6]. Like other researchers, they made use of a frame-mounted IMU, and forward-facing camera. They also implemented a Hall-effect sensor on the steering column, to measure steering angle, and a potentiometer on the brake handle to record its displacement. An interesting piece of technology they included on the IPB was a Microsoft Kinect sensor. It is mounted on an arm out in front of the bicycle and oriented to face the rider. The arm is used to give the Kinect enough space to provided rider tracking. All data is saved to a computer in a bag, mounted on the rear of the bicycle. This IPB is particularly interesting in its attempts of rider tracking, although the large arm out in front could limit mobility.

2.2 The Instrumented Bicycle

After examining several IPB concepts from around the world, it is clear what technologies are commonly used. The research methodologies within this realm are fairly diverse, but it seems that most seek to achieve the same goal: develop bicycling as a more legitimate form of transportation. Sharing in this goal, the design of another iteration of the IPB is presented.
This IPB is built around a 17-inch Jamis Coda Sport. This bicycle was found to be popular among individuals who ride as commuters. The solid frame and simple geometry provided good accessibility for mounting hardware components. Some of the geometry of the bicycle is given in Table 1, and a supplementary sketch is provided in Figure 1. The first four columns of Table 1 are data provided by the manufacturer [7] (shown in Appendix A) because the exact dimensions are not important to this study (no measurements were taken). However, the rear wheel rolling radius was measured because it is used to calculate the bicycle velocity. To measure this rolling radius, a point on the tire is marked and the bicycle is pushed in a straight line, allowing several revolutions of the wheel. The linear distance is then recorded (by tape measure) and divided by \( 2\pi \) times the number of wheel revolutions.

To qualitatively describe this bicycle, the head tube angle is relatively steep, which translates to a bicycle that is generally more stable, and easier to ride. This is common among commuter and cruiser bicycles, and quite different from more aggressive bicycles, like those used in mountain biking. The size, reach and wheelbase define parameters that are very important to an individual’s fit on a bicycle. Size is the length of the seat tube, it dictates how far apart the seat and pedals are. Most bicycle seats are adjustable so this helps the fit, but only within a small range. Reach is the horizontal distance from the center of the crank to the top of the head tube. As a rider, this dimension translates to how close the handlebars feel. Wheelbase is the distance between the front and rear wheel contact points while in an upright configuration (Figure 1). It is a bit subjective, but this bicycle could be classified as “medium” in size.

Table 1: Several common bicycle geometric parameters

<table>
<thead>
<tr>
<th>Size</th>
<th>Head Tube Angle</th>
<th>Wheelbase</th>
<th>Reach</th>
<th>Rear Wheel Rolling Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>17”</td>
<td>71°</td>
<td>41.38” (1.051 m)</td>
<td>15.79” (0.401 m)</td>
<td>0.341 m</td>
</tr>
</tbody>
</table>
The bicycle size is not necessarily relevant in the design of the data acquisition system, but it becomes important when considering the testing environment. The first bout of IPB research commissioned by the TRCLC was to study bicycle comfortability and safety. The exact goals of that study are outside the scope of this thesis, but they illustrate an important point. Being a naturalistic study of volunteers, a variety of riders would use the IPB. When dealing with a range of body types, especially when trying to assess safety and comfortability, there is no one-size-fits-all. So, the bicycle parameters listed above become important because they illustrate the limitations posed by using a single bicycle for IPB studies.

The mounts and location of the sensors are designed to (theoretically) be adapted to a variety of different bicycles. However, the actual implementation of this introduces a whole new set of issues. The basis of thinking about universal applications is to further the discussion on improving and standardizing IPB systems. One large hurdle is the occasional need for physical modification. For this IPB, there were two physical modifications made to the bicycle. First, an after-market cargo rack was mounted on the rear of the bicycle, above the back tire. This provided a rigid surface for mounting additional equipment. This modification is certainly reversible, but
makes the point that additional hardware was used. Second, very small holes were machined into
the metal surrounding each hub, to attach sensor pick-ups. This second modification is more
serious by virtue of permanently removing original metal, but it is very minor as modifications go.
In all, the data acquisition system demonstrates it could be easily applied to a variety of bicycles.

2.2.1 Data Acquisition System

Like previous IPBs, the data acquisition system for this bicycle is designed to be compact
and robust. This IPB is designed to exist in real-world transportation environments, not just in a
lab. Furthermore, the IPB is going to be used by volunteers, not just persons who work in a research
lab. In naturalistic studies, there is often little coaching or instruction that participants get prior to
performing an experiment. This makes it critical to generate a full system that closely maintains
the look and feel of an unmodified bicycle. The system has to be sound enough to handle any
reasonable condition that a rider would subject it to. It also has to be physically compact enough
to not restrict the mobility of a participant. Capturing the natural motion of a rider is an important
feature of these research projects, so special attention was given to this area.

The data acquisition system consists of two parts. The first part is the array of sensors. The
second is the data recording equipment. The array consists of five individual sensors and a video
camera. These sensors were chosen based on the type of data they measure, and what was of
interest to the TRCLC. A list of the sensors and a description of their general function is provided
in Table 2. The data recording system consists of three components, a small ASUS computer, a
National Instruments USB-6210 DAQ, and a USB splitter. All three of these components are
housed in a light metal case, which is secured to the bicycle’s cargo rack. The case is insulated
with thick foam to prevent motion of the internal components and protect against moderate
vibration. Holes were drilled through the case to accommodate the variety of wires that connected
to each sensor. Lastly, the case functions as a platform for the rider position sensor. The location
of the sensors and data acquisition box are shown in the profile image of the completed IPB (Figure
2).
Table 2: IPB Sensor Array

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Component(s)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front Wheel RPM</td>
<td>Bourns Optical Encoder</td>
<td>Measure the angular velocity of the front wheel</td>
</tr>
<tr>
<td>Rear Wheel RPM</td>
<td>Bourns Optical Encoder</td>
<td>Measure the angular velocity of the rear wheel</td>
</tr>
<tr>
<td>Inertial Measurement Unit/GPS</td>
<td>SBG Ellipse miniature IMU/GNSS</td>
<td>Measure bicycle linear accelerations, angular velocities, and GPS position</td>
</tr>
<tr>
<td>Steering Angle</td>
<td>Honeywell Hall-Effect Potentiometer, 180˚ of travel</td>
<td>Measure the angular displacement of the handle bar</td>
</tr>
<tr>
<td>Rider Position</td>
<td>2x Honeywell Hall-Effect Potentiometers, 90˚ of travel, combined with universal joint</td>
<td>Measure the lean and pitch angles of the rider, relative to the bicycle</td>
</tr>
<tr>
<td>Video Camera</td>
<td>Sony ActionCam</td>
<td>Capture snapshots of environment in front of bicycle</td>
</tr>
</tbody>
</table>

Figure 2: Completed Instrumented Probe Bicycle
2.2.2 IPB Sensors

Inertial Measurement Unit

The power and importance of inertial measurement units (IMUs) is evidenced by the many IPBs that use them. The unit purchased for this IPB is the Ellipse-N, produced by SBG Systems. It is a commercial research-grade product, complete with calibration records and recording software. The IMU houses eleven sensors: three single-axis accelerometers, three rate-gyros, three magnetometers, a barometric pressure sensor and a temperature sensor [8]. These sensors are high-end micro electro-mechanical systems (MEMS) components. The accelerometers, gyros and magnetometers come in sets of three to capture three-dimensional motion. SBG includes a proprietary Extended Kalman filter on this device, which provides accuracy estimation and high-fidelity output data. The Ellipse couples its internal sensors with a GNSS receiver, which is connected to a Tallysman GPS antenna. This GPS/GNSS system aids the IMU in measurement estimation and correction, thereby reducing error over time.

Figure 3 provides a close view of the mounted sensor. This location was chosen because it protects the IMU and will not interfere with a rider. The data is transferred through a single RS-232-to-USB cable, which is connected to the USB splitter inside the DAQ case. The unit is physically attached to the bicycle by a custom aluminum bracket. The rigid mounting scheme helps isolate the sensor from extraneous vibrations, the intent of which is long-term durability and accuracy in output data. The bracket assembly was built according to the specifications of this bicycle, but was designed with larger applications in mind. It takes advantage of a pressure-based clamping scheme, which could feasibly be attached to any circular tubing, so long as the diameter is within a certain range. Brackets like these can help standardize IPB builds.
Rider Position Sensor

Tracking rider motion is not a new aspect of IPB technology, but the sensor on this IPB measures rider motion in a new way. This new sensor is a common universal joint modified to measure the rotational angles of the joint arms. It is fabricated by removing the two bearing pins and replacing them with the shafts of two potentiometers. The lower arm of the universal joint is welded to a piece of extruded aluminum, which serves as a mounting platform, and is bolted to the DAQ box. The potentiometers measure the angular displacement of the upper arm, relative to the lower arm. The joint has two rotational degrees of freedom. It is fixed to the bicycle so it measures one rotation about an axis parallel to the frame, and one rotation about an axis perpendicular to the frame (Figure 4). Rotation about the parallel direction captures the lateral motion of a rider. This is classified as the rider lean angle. Rotation about the perpendicular direction captured the riders forward and back motion. This is classified as the rider pitch angle.
To gather rider motion data, a “sensing rod” is used to tether the rider to the universal joint. The sensing rod consists of two small carbon fiber tubes, one having a slightly larger diameter than the other. The larger tube is anchored to the open end of the universal joint, while the smaller tube is attached to a rider. The smaller tube rides on the inside of the larger, creating a telescoping motion. This allows the rider to move freely, without being impeded by the length of the rod. To attach the sensing rod to the rider, a chest harness and Velcro are used. The chest harness is small and has elastic straps for adjustability. One half of a Velcro square is fixed to the back of the harness, and the other half is fixed to the end of the sensing rod. Figure 5 shows the rider position sensor, with the rod in place, and Figure 6 shows the sensing rod being attached to the chest harness during a pilot test.

The design of this sensor assembly achieved three goals. First, the sensor effectively tracks the motion of a rider, relative to the bicycle. Second, it is adjustable to accommodate riders of various sizes. Third, the system provided a safe means of attachment, because in the event of a crash (or related incident) the sensor will easily break away from the rider.

Other IPBs that feature rider tracking have thus far used IMUs or video motion capture. IMUs are far more accurate, but substantially more expensive. Video motion capture systems require a lot of hardware and precise setup, which limits the range and mobility of the bicycle. In contrast, the sensor on the current IPB is discrete, of low-cost, and is easy to integrate into a data acquisition system. Many volunteers have said they were not aware of it during their ride.
Figure 5: Rider position sensor with sensing rod in place

Figure 6: Rider sensor in use during pilot testing
Wheel RPM Sensors

A common method for calculating bicycle velocity is by measuring wheel RPM. While the IMU also measures forward velocity, this additional data helps verify accuracy. This model bicycle has no convenient surfaces from which to measure wheel rotation, so an aluminum disk was attached to the hub of the wheel (Figure 7). This disk provides a surface for the encoder disk to ride on. The encoder is secured to the bicycle frame by a simple bracket assembly (Figure 8). A plastic spool covers the sensing shaft of the encoder, and its outer surface is placed in contact with the aluminum disk. The edge of this disk was given a mild knurl, to increase the friction at the contacting surfaces. Despite carefully machining and mounting, the disk is not perfectly circular. To account for the small eccentricity, a spring and pivot arm assembly helps maintain smooth, constant contact between the aluminum disk and the edge of the spool.

Figure 7: Two views of aluminum sensing disk and mounting hardware
Steering Angle Sensor

The steering angle sensor on this IPB is constructed by borrowing design features from both the IMU and encoder mounts. The actual sensing is performed by a potentiometer. A plastic spool (similar to the encoders) is placed over the potentiometers shaft, and put in contact with the steering column (Figure 9). The assembly is fixed to the bicycle with a circular pressure clamp, almost identical to the one used for the IMU, and a T-shaped intermediate piece. This clamp allows the sensor to sit very close to the steering column, and the T-shaped piece allows the sensor to be tilted so it is perpendicular to the column. Close attention was given to the size of the bracket, to ensure it would not interfere with the up and down motion of a rider’s legs. The current version has less than a half-inch of overhang on either side of the tube. A spring and pivot-arm system is included to increase contact pressure and prevent slippage. Although, occasional slippage issues were still present, so a piece of abrasive tape is wrapped around the steering column.
Data Acquisition Case

The data recording components are housed in the data acquisition case (Figure 10). The DAQ and USB splitter are placed under the bottom layer of foam, while the laptop sits on top of this layer. All the sensors, apart from the IMU, are wired into the DAQ. The DAQ not only reads the signals from each sensor, it also supplies the 5V power source. Both the DAQ and the IMU are connected to the USB splitter, which is necessary because the computer only has one USB port. The laptop supplies power for both of these devices.

The rider position sensor and GPS receiver are attached to the top of the box (Figure 4). The rider position sensor is bolted to the box (the securing hardware is hidden by the top later of foam). The GPS receiver has a magnetic bottom, so a small steel pick-up is placed between the top layer of foam and the outer shell of the case. Holes were drilled in the side of the case for sensor wiring. The case itself has a light sheet metal coating on all surfaces, reinforced metal corners, and two latches to secure it when closed. It is fixed to the bicycle’s cargo rack with six adel clamps, located underneath the case.
2.2.3 Sensor Calibration

Inertial Measurement Unit

Sensor calibration is important to ensure the data being collected is accurate. Simple procedures are used to assess the general state of each sensor. First, the IMU came factory calibrated with the calibration records, so no recalibration is necessary. However, the mounting location and orientation of the IMU is important for accurate data collection. The unit has a measurement origin, and all signals are measured with respect to that point. It is conventional to measure kinematics from the center of gravity (c.g.) of a body. But it is rarely convenient (or practical) to place a sensor at the c.g., so correction factors must be applied to the data. The IMU recording software, sbgCenter, has a menu for inputting these correction factors. There are two categories of corrections factors, misalignment angles and level arms. Misalignment angles account for the rotational orientation of the IMU. Lever arms are the linear offset distances from the body’s c.g. to the measurement origin of the IMU.

The IMU’s internal reference frame has an x-direction that points forward, a y-direction that points out to the right, and a z-direction that points down. The bicycle’s reference frame follows this same convention. The IMU clamp holds it perpendicular to the seat tube and along the centerline of the bicycle. Thus, only one misalignment angle needs to be accounted for. A
digital angle gauge was used to record the offset angle between the sensor plate and the ground plane, this was found to be 14.9°. While nothing is perfect, the other two misalignment angles are assumed to have negligible offset. The lever arms are more difficult to account for because the precise location of the bicycle c.g. is not currently well defined. The longitudinal offset is approximated by balancing the bicycle by its seat, and estimating the perpendicular distance. The vertical direction is even more difficult to measure, so an educated guess is made to approximated its location. The bicycle is assumed to be mostly symmetrical about its vertical plane, so no offset distance is included for in the lateral direction.

Encoders

Both encoders were calibrated during steady state angular velocity conditions, by manually counting wheel RPM, using a spoke marker and a stopwatch. The manual calculations were compared with the computer measured RPM and found to be ± 3 RPM in accuracy. This test was repeated multiple times for each wheel, for angular velocities ranging from 10 to 80 RPM, and the results were consistent. The accuracy of the encoder is presumed to be better, but the test procedure limits the accuracy of calibration results.

Potentiometers

The potentiometer used for steering angle has a rotational range of 180° ± 2°, and output linearity of ± 4% across that range [9]. Calibration tests reveal most of the deviation from linear behavior occurred near the extrema of this range. The calibration procedure began by rotating the sensing shaft to about 90°, or the midpoint in its travel range. The output voltage at that position was recorded and used as a reference. Multiple test iterations were conducted to evaluate the change in angular displacement versus the change in output voltage. The angular displacement was measured with a digital angle gauge that has an accuracy of ± 0.1°. The ratio was calculated to be about 50 degrees per volt. After programming this ratio into the DAQ software, more calibration tests were conducted. These tests ranged from 45° to -45° of steering angle input. This was assumed to encompass the entire operational range. The program outputs from this second round of testing were within 1° of the angle gauge readings.

The digital angle gauge was also used for calibrating the rider position sensor. Both rider position potentiometers are the same brand as the steering angle sensor. The only difference is the
rotational range. The rider position potentiometers have a rotational range of $90^\circ \pm 2^\circ$. During calibration the two arms of the universal joint were kept in-plane with each other, and each potentiometer was evaluated individually. The same procedures were followed for the steering angle potentiometer. The test results indicate an accuracy of $\pm 0.5\ degrees$ for each potentiometer.

Summary of Sensor Accuracy

Table 3 provides a summary of the accuracy of each of the sensors. Some values in this table come from the calibration procedures, and others from manufacturer data sheets [8],[9],[10]. These data sheets are found in Appendix A.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Accuracy Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMU Output</td>
<td>Velocity: Horizontal/Vertical – 0.05 m/s</td>
</tr>
<tr>
<td></td>
<td>Attitude: Roll/Pitch – 0.2°, Yaw – 0.4°</td>
</tr>
<tr>
<td>IMU GPS</td>
<td>Position: $\pm 2\ m$ (Latitude and Longitude)</td>
</tr>
<tr>
<td>Optical Encoders</td>
<td>Resolution: 64 pulses/rev</td>
</tr>
<tr>
<td></td>
<td>Wheel RPM: $\pm 3\ rev/sec$</td>
</tr>
<tr>
<td>Potentiometers</td>
<td>Resolution: 50 $^\circ/V$</td>
</tr>
<tr>
<td></td>
<td>Steering Angle: $\pm 1^\circ$</td>
</tr>
<tr>
<td></td>
<td>Rider Position Angles: $\pm 0.5^\circ$</td>
</tr>
</tbody>
</table>

2.3 Data Acquisition

2.3.1 Data Recording Software

The process of data measurement for this IPB is a combination of running two separate programs, namely LabVIEW [11] and sbgCenter [12]. All sensors wired into the DAQ use LabVIEW to read, interpret, and record data. The IMU uses sbgCenter to do the same. sbgCenter is a recording software that came with the unit. Since it not a well-known program, this section will describe some of its basic functions and why it is currently used for the IPB. The other data recording software, LabVIEW, is a large and well-known program, so no explanation of its features is given in this thesis.
The IMU’s firmware is written in C and uses the sbgECom binary protocol to communicate with computers. For the savvy programmer, this allows the IMU to be used in a variety of applications. For the everyday user, learning the language for writing these programs would be quite tedious and time-consuming. sbgCenter, however, makes it very easy to interact with the IMU. The user interface has several well-organized menus that enable a user to initialize the sensor, record, and visualize data streams (Figure 11). The program includes a unit configuration menu, where the misalignment angles and lever arms are input. It also monitors sensor accuracy and operational status. Once recorded, the saved data can be exported with user-defined presets. The amount of data can be very large, so these presets tell the program which data to export, and which to leave behind. All data is exported to a text file.

Figure 11: Example of sbgCenter user interface

2.3.2 LabVIEW Program

While sbgCenter is effectively self-sufficient, LabVIEW requires a program to read and interpret data. The program described here was built specifically for this IPB. It is designed to provide a simple interface for operating the recording software. As future research projects with this IPB will undoubtedly take place, a goal was to make the data acquisition system easy to use.
Before the program begins, the raw signals are processed by the DAQ. There are three analog signals from the potentiometers, and two digital signals from the encoders. The encoder signals are wired into ports with digital counters. These counters are configured to record the number of times the rising edge of a waveform occurred. The LabVIEW program then reads these signals from the DAQ by making use of Express VI's. One Express VI is used to read the pulse count from the digital counters, and the other is used to read the analog voltage signals. The foundation of the program is a “while loop”. It enables the user to discretize the continuous data by setting a sampling frequency. Every aspect of the program works simultaneously, but for explanation purposes it will be broken down into two parts. First, interpreting the digital signals, then interpreting the analog signals.

The DAQ records the total number of pulses (since initiation) for each encoder wheel. The number of pulses between two successive samplings is found by subtracting the number of pulses at the previous sampling from the number of pulses at the current sampling. Because sampling occurs at a fixed time increment, this result can be easily converted into a pulse rate, that is, the number of pulses per second. Then, it only takes a few simple calculations to reach the angular velocity of the wheel. The encoder outputs 64 pulses per revolution. Multiplying this by the pulse rate yields the angular velocity of the encoder wheel. From here, it is just a matter of knowing the ratio of the encoder wheel to the aluminum disk. The diameter of the encoder wheel and diameter the aluminum disk were measured with high precision calipers, and the ratio was calculated to be 0.5417. Multiplying this number by the encoder angular velocity results in wheel angular velocity. A correction for the time increment is applied, and the final units are revolutions per minute. The same procedure was use for both the front and rear wheel encoders.

The second task is interpreting the potentiometer voltages. During the calibration period, each potentiometer was tested over its rotational range, and the output voltage of the middle position was recorded. These numbers are added to the program as constant reference voltages. At each sample time, the program reads the voltage associated with each potentiometer and subtracts that value from its reference voltage. This differential voltage is then multiplied by the resolution of the potentiometer (determined during calibration) to calculate the angle associated with the potentiometer's position.
This process worked well for the steering angle sensor, as it has a well-defined zero point, and therefore a well-defined reference voltage. The rider position sensor is more complicated, however, because the placement of the sensing rod varies with rider size and chest harness placement. So, there is no set reference voltage that is applicable to all riders and tests. To account for this, the program has a control on the interface that can save a sample voltage as the reference voltage. Once saved, the program uses that voltage as the static reference, until it is reset or the program is shut down.

Because the reference voltages of the potentiometers are at (or near) the middle of their rotational range, the differential values can be positive or negative. The sign convention for this output is programmed to follow the right hand-rule of the coordinate system described in 2.2.3. Assuming all sensors are at their zero point, the sign convention is as follows. For the steering sensor, a clockwise rotation of the handle bars results in a positive signal and a counterclockwise rotation, a negative signal. For the rider lean, a lean to the right is positive and to the left, negative. For the rider pitch, rotating the upper body away from the handle bars is positive, and toward the handle bars is negative. The explicit form of the program is shown Appendix B.

During operation, the program passes all calculated values to the user interface, called the front panel, where they are displayed on numeric indicators and charts. Figure 12 shows a screenshot of the front panel (while the program is not in use). In addition to the displays, there are five controls: a sampling frequency control, the rider position zeroing controls, a save control, and a chart zeroing control. The sampling frequency is a drop-down with three options: 10 Hz, 20 Hz, or 50 Hz. The rider position controls have already been described. The save control allows the user to name the output file and select its location. Lastly, the chart zeroing control simply clears the display charts of data. When saving is turned on, all data collected by the Express VI is then written to an Excel spreadsheet.
Figure 12: LabVIEW front panel for IPB data collection system

2.3.3 Challenges and Limitations

Certain challenges arose during the design and implementation of this data acquisition system that were circumvented sufficiently, but would benefit from improvement. First, by virtue of running two separate programs for data collection, synchronization is a problem. All data is time-stamped, the IMU data by GPS time, and the LabVIEW data by the computer time. However, after a few sample runs it became clear the accuracy of the computer time did not match well with the GPS time. Furthermore, because a user must manually start both programs, there is normally a small time offset between the discrete samples of the two data sets. This is not too large an issue, since it can never be offset more than half of the sampling period. For example, if the sampling frequency is 10 Hz, the maximum this offset can be is 0.05 seconds. Both of these problems can be avoided if the entire system operates from a single software. LabVIEW has a serial communication toolbox, so it is possible to wire the IMU into a DAQ. However, the time constraints of this project did not permit exploring this possibility. Also, the current DAQ may not
have sufficient physical capabilities to accept the RS-232 wires, so a reassessment of the system hardware may be necessary.

Another challenge is in the operation of the steering angle sensor. During the systems testing phase, the sensor was observed to occasionally slip, in very small amounts, when moderate torque was applied to the handle bars. Over the course of a five-minute ride, this could translate to an error as large as 5 degrees. The abrasive tape seen in 2.2.2 was added to try and alleviate this problem, but it did not completely fix it. Thankfully, this error is easily taken care of by using a non-linear detrending function in MATLAB. It seems a permanent physical solution to this would require modification of the steering column collars, or the plastic spool.

Lastly, the LabVIEW data acquisition program suffers from what is thought to be inadequate buffer size. During operation, all data is stored in a buffer until the program is stopped. When stopped, all the data is transferred from the buffer to an Excel spreadsheet. During systems testing, it became apparent that after a certain amount of data was gathered, the program could not keep up. For example, if the sampling frequency was set to 10 Hz, the output spreadsheet should have 10 rows of data over the course of one second. But, after the critical point was hit, the data rows in a one second period would begin to slowly drop, from 10 to 9 to 8 and so on. Further investigation, revealed this problem would begin after about 6 minutes of continuous data collection when recording at 20 Hz. The first research study conducted by the TRCLC required participants to ride a route that had a completion time of anywhere from 5 to 8 minutes, so the IPB was limited to recording data at 10 Hz. According to the TRCLC, the sampling rate was acceptable for that study. Again, the time restrictions of this project prevented searching for the true cause of this problem, and its associated solution.

2.4 Data Processing

2.4.1 Raw Data Pre-Processing

The IPB is a very useful tool for collecting data, but the data is not immediately ready for analysis. In this type of transportation research, the primary way of classifying data is according to geographic segments, e.g. a portion of a route that is sidewalk versus a portion that is roadway. Therefore, the final deliverable for the IPB system is a program that applies this geographic segmentation to the data. However, before the program can run, several steps are necessary to prepare the data.
First, the text files are imported into the Excel spreadsheets, creating a single document of data. This operation is done manually for each trial. Next the data is synchronized and cleaned. The synchronization process is more of a challenge than initially anticipated. Currently, time-stamping alone does not achieve fully synchronized data because the computer time is not exactly aligned with GPS time. To finish the process, the bicycle’s forward velocity data is used. Information about the velocity is derived from two sources, the IMU and the rear wheel encoder. The IMU directly outputs the velocity components of the bicycle. The forward velocity is plotted against time to generate a velocity profile. The velocity data is then calculated and plotted using the rolling radius and rear wheel angular velocity. These two velocity profiles are very similar and are used to make the final alignment for the rows of data. Figure 13 shows the result of this process, for a single case.

Once synchronization is complete, the data is cleaned. This cleaning process consists of removing all excess or corrupted data. The excess data is a byproduct of using two recording programs. One is started first, so it records more data than the other. This idle data is trimmed so the number of data points are equal. In some cases, small segments of data are corrupted and can be removed. In rare cases, there is too much corruption to use the trial, so it is discarded.

Figure 13: Example output of synchronized bicycle velocity profiles
2.4.2 Geographic Segmenting Algorithm

The IMU records latitude and longitude data, so it is easy to manually identify geographic segments for an individual trial. However, this becomes impractical for a large number of trials. An algorithm was developed to solve this problem, and automated with MATLAB. The goal of the algorithm is to segment all trials equally, and it does this by finding index points.

In a set of data, a specific GPS location has a unique index point. All IPB data is recorded with respect to time, so riders traveling at different speeds will reach that GPS location at different index points. Having a computer identify these index points has certain challenges, because a GPS location is comprised from two measures: latitude and longitude. If a route crosses a line of latitude or longitude more than once, there will be multiple index points associated with a single latitude or longitude value. This makes it difficult to find the correct index point. An even bigger problem is the variation in the paths. Riders rarely, if ever, traverse the exact same path. This presents the possibility that, for a geographic point of interest, no common GPS data exists. Even if the points are very close together, the computer cannot easily see this. The algorithm overcomes these issues by stepping through index values to find which data point is closest to a fixed point.

Prior to running the program, the data is imported to MATLAB and organized into structured arrays. Data is manually imported into the workspace. From there, several script files organize the individual variables into structured arrays. The algorithm has several loops at its foundation, and structured arrays are very efficient for calling data in these loops.

The program begins by integrating the forward velocity data, from every trial, to calculate the total distance traveled (Figure 14). Because the data is discrete, the integration function uses a trapezoidal method. The result is only an approximate integral, but it works sufficiently well. The distance data allows the algorithm to pick its initial test indices. After the integration completes, the program opens a graphical user interface (GUI). This GUI allows the user to define the geographic segments. It displays a plot of the route, in longitude versus latitude, and includes directions for selecting points (Figure 15). Each point is chosen by manually selecting a GPS location from the route (any number of points can be chosen). The selected points are saved as fixed GPS locations. The GUI is included to make the program easier to operate, especially for those who have no experience in MATLAB.

Once the geographic points are selected, the algorithm uses their indices to generate distance points. These distance points give the algorithm its initial guess indices. The guess indices
are used to start the index locating loop. The loop generates a fixed GPS point, from the GUI, and a test GPS point from the guess indices. Then it takes a positive index step and generates a new test GPS point. The loop compares which of these two test points is closer to the fixed point, this determines the search direction. It continues in this direction until it locates the test GPS point that is closest to the fixed GPS point. Once this point is found, the program saves its index and moves to the next fixed GPS point. This is performed on one trial at a time, once all trials are completed, the program outputs a matrix of index points. The printed code for this program is shown in Appendix C.

Figure 14: Segmenting algorithm flowchart
2.5 Results

2.5.1 IPB Systems

The IPB went through a series of preliminary tests to work out the bugs in the system. Once these tests were complete, it was used in a study by the TRCLC. In all about 60 people volunteered for the study and completed the experiment. Of this total, 8 trials had significant data corruption and were discarded. The other 52 other trials were successful, and prove the IPB works quite well. The experimental trials were conducted over a two-week period, during which time the IPB was challenged in many ways. The repeated exposure to road vibrations revealed that the USB splitter’s
connection was not very secure, which was the primary reason for data corruption in the 8 trials that were discarded. It was replaced and everything worked smoothly again.

A few of the testing days were rather windy and caused the bicycle to fall over a couple of times. None of the sensors were damaged and the only lasting marks were small scrapes on the DAQ case. Furthermore, there was no evidence of adverse effects on the collected data. This is not an ideal way to test the limits of the IPB, but the circumstances prove the system is sturdy and reliable.

2.5.2 Segmenting Algorithm Output

The output of the program is a matrix of index points. The rows of this matrix represent the number segment points, while the columns represent the number of trials. The algorithm was tested with the data from the TRCLC study. The points selected for this example are arbitrary, but illustrate the effectiveness of the algorithm, Figures 16 and 17. Figure 16 displays the location of the segment points based on the distance points. Recall, the distance points are used as guess points, and are derived from the integrated velocity. The locations of these distance points are not very accurate because of two things. First, the integration routine is only an approximate, and its error propagates over time. Second, not all riders travel the same exact distance. This is why the index locating loop is necessary. Figure 17 shows the result of this loop (the final result of the algorithm); note each fixed point is surrounded by a tight grouping of the index points.
Figure 16: Route segmenting results using only distance points

Figure 17: Route segmenting results using full algorithm
2.6 Conclusions and Future Work

The IPB presented here was carefully designed and built to serve as an effective data acquisition tool for the TRCLC. This IPB was found to be effective in its current state. The results of a first study demonstrate all systems work and provide accurate data output. However, certain features could be improved, and are recommended here as future work. First, the physical build has proven fairly robust, but the front and rear wheel encoders could benefit from a protective shielding. They are currently vulnerable to anything that might impact the side of the wheel. Second, a major improvement would be integrating the two data-recording programs into one program. This would alleviate the time-consuming process of manual data synchronization, and increase overall accuracy. Third, the current system requires a user to manually start the recording software, with the open laptop. A remote activation system, wired or wireless, would increase system efficiency and allow the DAQ box to stay closed and secured at all times. Lastly, there should be investigation into the buffer issue in LabVIEW. This would enable experiments to run at higher sampling frequencies.

The geographic segmenting algorithm worked very well. It provides an easy means of selecting geographic points, and quickly outputs the associated index points. The advantage of these index points is that they can be used universally. Index points are intrinsic to a data set, not a particular software. While the program itself uses MATLAB to function, the output can be used in any software package. Currently, the only cumbersome part of this program is the process of organizing the data into acceptable MATLAB format. A single data acquisition software would help solve this problem. If that future work was accomplished, then a script file could be developed to read the data, clean it, and organized it into structured arrays. This would totally remove a user from the current data processing loop. In turn, the IPB package would function more efficiently.
CHAPTER 3

COMPUTATIONAL DYNAMIC MODEL

3.1 Review of Bicycle Modeling Literature

The second portion of this thesis describes the computational dynamic model. This model provides a foundation for studying the motion of the bicycle. Before describing the model, it is important to define its origin. The bicycle is a simple machine that has been around for several hundred years. Yet, its complexities are still not fully understood. In the past decade, a substantial amount of research has been conducted to further this understanding. This section presents a review of several key pieces of literature that develop the groundwork necessary to derive the model.

One of the most influential articles on the dynamic modeling of the bicycle is presented by Meijaard et. al [13]. In it, the authors outline the most popular formulation of the bicycle, the Whipple model. This model defines the bicycle as a four-body system. These bodies are the rear wheel, rear frame, front frame, and front wheel. It also imposes a series of assumptions. Both wheels are axisymmetric, and both the rear frame and front frame have a plane of symmetry. The rear frame is a combination of the bicycle frame and a rigid rider. This rider contributes mass properties to the frame. The front frame is the handle bar and front fork assembly. Both wheels of this model are assumed to maintain knife-edge contact with a horizontal ground plane, and do not slip. The knife-edge assumption means there is a single point of contact between a wheel and the ground plane, and it neglects the toroidal nature of a tire.

The authors also present a set of numerical mass and geometric parameters for a bicycle. These parameters have been used by several other researchers to validate their models, and are used in the same way in this thesis. After describing the degrees of freedom and constraints on the system, the dynamical model is presented. It is comprised of a single linear equation that defines the longitudinal dynamics, and two coupled linear equations that define the lateral dynamics. The lateral dynamics were used to generate an eigenvalue plot that describes the stability characteristics of the uncontrolled bicycle. Finally, the authors use the computer program SPACAR to demonstrate the asymptotic stability and energy conservation of the nonlinear bicycle model. The
bicycle has no real damping, so the energy imposed from a perturbation is transferred to the forward velocity. However, this effect is not captured by the linear model.

A well-defined form of a nonlinear approach to the Whipple model is present by Basu-Mandal et. al., in their study of the circular motions of a bicycle [14]. This paper provides a detailed derivation of the equations of motion using the Newton-Euler approach. The authors also generate the governing equations using Lagrange’s equations, but do not display those results due to their size. Using the equations from both methods they numerically evaluate an arbitrary test case to show they have matching results.

One very important feature of the nonlinear model is the relationship between the front wheel mass center and the front wheel contact point. Because reference [1] is concerned with the nonlinear model, they provide an excellent description and visualization of this. It is expressed by a triple vector product and is critical in capturing the behavior of the bicycle.

Finally, in 2012 Jason Moore published his dissertation entitled Human Control of a Bicycle [15]. While this dissertation’s main focus is on studying the complexities of rider control, he begins his work by presenting his own derivation of the nonlinear Whipple model, using Kane’s method. This derivation is presented in phenomenal detail and clarity. His detail played a critical role in the final correctness of the model presented in this thesis. The intermediate results he displayed were scrupulously examined to help pinpoint and correct errors. The extent of this dissertation’s helpfulness cannot be overstated.

3.2 Bicycle Model Description

3.2.1 Overview and Focus

The derivation presented herein follows the Whipple Model, and its basic assumptions. The purpose of this derivation is to generate a useful computational model. With this in mind, the derivation process is tailored to the capabilities of the computer software used to build the computational model. This derivation is achieved using Kane’s method, and is built up from first principles. The process of generating the equations of motion is done using MATLAB. Script and function files are used to generate symbolic expressions for the building blocks of Kane’s equations. These expressions are then numerically evaluated to solve a set of nonlinear, first-order differential equations.
3.2.2 Bicycle Configuration

The system is fully defined by a set of eight generalized coordinates. Following convention, these generalized coordinates are represented by the variable $q$. Figure 18 illustrates the bicycle in an arbitrary configuration, in three-dimensional space. Two coordinates ($q_1$ and $q_2$) locate the rear wheel contact point, $P$, relative to the inertial origin, $O$. The bicycle is then rotated about the vertical axis. $q_3$ represents this yawing motion. $q_4$ and $q_5$ are the roll angle and pitch angle of the bicycle’s rear frame. After the pitch angle is applied, the image is exaggerated to show the front wheel contact point, $Q$, no longer touching the ground plane. In normal operation of the bicycle, $Q$ is constrained to the ground and the pitch angle is very small (Figure 19). Despite its small contribution, this variable is necessary to enforce the constraint equations. The last coordinate shown in Figure 18 is $q_7$, the steering angle. The other two coordinates are the rotation angle of the rear wheel and the rotation angle of the front wheel, $q_6$ and $q_8$, respectively. These two are omitted from Figure 18 to avoid further complicating the image, but they are shown in Figure 20. Figure 20 provides a planar view of the bicycle in its nominal configuration. The four bodies of the model are named A, B, C, and D, respectively.

![Figure 18: Arbitrary configuration of the bicycle in three-dimensional space](image-url)
Figure 19: Configuration of the bicycle with the front wheel touching the ground

Figure 20: Planar view of the bicycle in its nominal configuration
A summary of the generalized coordinates is provided below:

\[ q_1 = n_1 \text{ component of rear wheel contact point position vector} \]
\[ q_2 = n_2 \text{ component of rear wheel contact point position vector} \]
\[ q_3 = \text{rear frame yaw angle} \]
\[ q_4 = \text{rear frame roll angle} \]
\[ q_5 = \text{rear frame pitch angle} \]
\[ q_6 = \text{rear wheel rotation angle} \]
\[ q_7 = \text{steering angle} \]
\[ q_8 = \text{front wheel rotation angle} \]

### 3.2.3 Notation

The work presented in this derivation uses vector and matrix notation. This section defines the meaning behind the notations used in formulating the dynamical expressions. First, the transformation matrix \( R^B \) is the matrix that relates unit vectors fixed in the inertial frame to those fixed in the rear bicycle frame B, as shown in Equation 1.

\[
\begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix} = \begin{bmatrix}
  R^B
\end{bmatrix} \begin{bmatrix}
  N_1 \\
  N_2 \\
  N_3
\end{bmatrix}
\]  

(1)

Next, position vectors are defined by the variable \( r \). Equation 2 provides an example of a relative position vector. The under-bar identifies it as a vector. The superscript term indicated its direction. In this example, the position vector is from point \( O \) to point \( P \). The subscript on the outside of the brackets denotes which frame the vector is expressed in.

\[
L^{p/o} = \begin{bmatrix}
  q_1 \\
  q_2 \\
  0
\end{bmatrix}_R
\]  

(2)
Velocity and acceleration vectors are expressed in nearly the same way. Equation 3 shows a relative velocity expression. In this expression, $^R\mathbf{v}^G_A$ and $^R\mathbf{v}^P$ represent the velocities of points $G_A$ and $P$ in reference frame $R$. $^R\boldsymbol{\omega}^A$ represents the angular velocity of body $A$ in $R$, and $^R\mathbf{r}^G_A/\mathbf{r}^P$ represents the position vector of $G_A$ relative to $P$.

To expedite the derivation of the equations of motion of the bicycle, matrix notation is used. In matrix notation, vector cross products are represented by a matrix-vector product. In the relative velocity equation, the angular velocity vector is replaced by a skew-symmetric matrix indicated by an over-tilde (Equation 4).

$$^R\mathbf{v}^G_A = ^R\mathbf{v}^P + ^R\boldsymbol{\omega}^A \times ^R\mathbf{r}^G_A/\mathbf{r}^P$$  \hspace{1cm} (3)

$$^R\mathbf{v}^G_A = ^R\mathbf{v}^P + ^R\tilde{\boldsymbol{\omega}}^A \cdot ^R\mathbf{r}^G_A/\mathbf{r}^P$$  \hspace{1cm} (4)

### 3.3 Kinematics

#### 3.3.1 Rotation Matrices

A total of eight frames define the system. Each of the four bodies has a local frame, and the other four are: the inertial frame, the yaw frame, the roll frame, and the steering tilt frame, $R$, $Y$, $L$, and $C'$, respectively. Body $B$ is oriented with respect to the inertial frame by a 3-1-2 rotation sequence. Equations 5-7 provide the rotation matrices associated with these intermediate frames, and Equation 8 is the combined sequence.

$$^R\mathbf{R}^Y = \begin{bmatrix} C_3 & S_3 & 0 \\ -S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$  \hspace{1cm} (5)

$$^R\mathbf{R}^L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_4 & S_4 \\ 0 & -S_4 & C_4 \end{bmatrix}$$  \hspace{1cm} (6)
The rotation of body A with respect to body B is simply

\[
\begin{bmatrix}
C_5 & 0 & -S_5 \\
0 & 1 & 0 \\
S_5 & 0 & C_5
\end{bmatrix}
\]

(7)

\[
\begin{bmatrix}
C_3C_5 - S_3S_4S_5 & C_5S_3 + C_3S_4S_5 & -C_4S_5 \\
-C_4S_3 & C_3C_4 & S_4 \\
C_3S_5 + C_2S_3S_4 & S_3S_5 - C_3C_5S_4 & C_4C_5
\end{bmatrix}
\]

(8)

Working from body B to the body C requires a two-rotation sequence. This is because of the steering tilt angle, \( \lambda \), which contributes a constant angular offset (Figure 20). Rotating first about the \( b_2 \)-axis, then about the \( c_3 \)-axis, the combined rotation matrix is

\[
\begin{bmatrix}
C_6 & 0 & -S_6 \\
0 & 1 & 0 \\
S_6 & 0 & C_6
\end{bmatrix}
\]

(9)

The final rotation is body D with respect to body C, which is

\[
\begin{bmatrix}
C_8 & 0 & -S_8 \\
0 & 1 & 0 \\
S_8 & 0 & C_8
\end{bmatrix}
\]

(11)
3.3.2 Position Vectors

There are seven key points that define the geometry of the bicycle. These points also provide a means of developing the closed kinematic chain between the rear wheel contact point and the front wheel contact point. This kinematic chain is used to formulate the constraint equations on the front wheel contact point. Starting from the inertial origin, the position vectors sequentially move through the bicycle. The position vector from the inertial origin to the rear wheel contact point is

\[
\mathbf{r}^{P/O} = \begin{bmatrix} q_1 \\ q_2 \\ 0 \end{bmatrix}_R
\]  

(12)

This vector defines the global position of point \( P \). The third element is zero because the ground plane is horizontal and flat. Next is the position vector from the rear wheel contact point to the rear wheel mass center, \( G_A \).

\[
\mathbf{r}^{G_A/P} = \begin{bmatrix} 0 \\ 0 \\ -r_{rw} \end{bmatrix}_L
\]  

(13)

This position vector serves as an implicit holonomic constraint. It defines \( G_A \) as exactly one radius away from the contact point and, from Equation 12, the contact point has a zero vertical displacement. It is expressed in the lean frame, as this is the most practical way to define the constraint. \( G_A \) is a shared point on body A and body B. The other two critical points on body B are expressed as

\[
\mathbf{r}^{G_B/G_A} = \begin{bmatrix} x_B \\ 0 \\ -z_B \end{bmatrix}_B
\]  

(14)
\( G_B \) is the mass center of body B. S is a shared point between body B and body C. In transitioning to body C, the vectors are now expressed in its local frame. The locations of the mass center of body C and the mass center of body D are defined in Equations 16 and 17.

\[
\begin{align*}
\mathbf{r}^{G_{\mathcal{C}}/S} &= \begin{bmatrix} x_C \\ 0 \\ z_C \end{bmatrix}_C \\
\mathbf{r}^{G_{\mathcal{D}}/S} &= \begin{bmatrix} x_D \\ 0 \\ z_D \end{bmatrix}_C
\end{align*}
\]

(15)

(16)

(17)

The position vector that is directed from the mass center of body D to point \( Q \) is a bit more complex than the position vectors expressed thus far. The location of \( Q \) is a function of the steering angle, roll angle, and pitch angle \( (q_7, q_4 \text{ and } q_5) \). Rather than introduce additional variables to locate this point, the triple vector product from Basu-Mandal is used [14]. This expression takes the form

\[
\mathbf{r}^{Q/G_{\mathcal{D}}} = r_{FW} \left( \frac{(\mathbf{c}_2 \times n_3) \times \mathbf{c}_2}{\|(\mathbf{c}_2 \times n_3) \times \mathbf{c}_2\|} \right)
\]

(18)

Expanding the triple vector product results in

\[
\mathbf{r}^{Q/G_{\mathcal{D}}} = \begin{bmatrix} -r_{FW} (C_4 C_5 C_7 S_\lambda - S_4 S_7 + C_4 C_7 S_5 C_\lambda) / #1 \\
0 \\
r_{FW} C_4 C_{\lambda 5} / #1 \end{bmatrix}_C
\]

(19)

Where

\[
#1 = \left( (C_\lambda C_4 C_7 S_5 - S_4 S_7 + C_4 C_5 C_7 S_\lambda)^2 + C_{\lambda 5}^2 C_4^2 \right)^{1/2}
\]
3.3.3 Kinematical Differential Equations

For this derivation, the generalized speeds are simply the derivatives of the generalized coordinates. This is an easy, and common, way to define the generalized speeds, although it is not necessarily the most efficient way to express the final dynamical equations. A different set of generalized speeds may provide a more concise set, but no time was spent following this path. With that said, the set of kinematical differential equations is expressed (with $i = 1, \ldots, 8$) by

$$u_i = \dot{q}_i$$ (20)

3.3.4 Generalized Speeds

The bicycle has three degrees of freedom after the motion constraints are applied. So, of the eight generalized speeds three are independent and the remaining five are dependent. Examination of the model configuration immediately reveals that $u_1$ and $u_2$ have no contribution to the dynamical equations, so those generalized speeds are omitted from the remaining work. The choice of independent speeds is optional, but following convention, $u_4, u_6,$ and $u_7$ are chosen as the set of independent speeds [13]. This makes $u_3, u_5,$ and $u_8$ the dependent set. The matrix expressions can now be shown as

$$R^G_B = \begin{bmatrix} R^G_{yG} & \begin{bmatrix} u_4 \\ u_6 \\ u_7 \end{bmatrix} \\ \end{bmatrix} + \begin{bmatrix} R^G_{yD} \\ \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_8 \end{bmatrix}$$ (21)

In the above example, \([R^G_{yG}]_B\) is a three by three matrix, where $x = I$ or $D$. $I$ represents the matrix associated with the independent generalized speeds, and $D$ the matrix associated with the dependent speeds.
### 3.3.5 Angular Velocities

The four bodies of the bicycle are connected by a series of single-axis revolute joints. The rotational kinematics associated with a 3-1-2 rotation sequence are well defined but, for the sake of completeness, the individual parts used to build this expression are shown. Starting from the inertial frame, body B undergoes a yaw rotation, then a lean rotation, and finally a pitch rotation, expressed by

\[
^R \vec{\omega}^Y = u_3 R_3
\]  

(22)

and

\[
^Y \vec{\omega}^L = u_4 L_1
\]  

(23)

and

\[
^L \vec{\omega}^B = u_5 b_2
\]  

(24)

Summing these components, the final expression of the angular velocity of body B with respect to the inertial frame is

\[
^R \vec{\omega}^B = ^R \vec{\omega}^Y + ^Y \vec{\omega}^L + ^L \vec{\omega}^B
\]  

(25)

Expanding and transforming all the vector components to the B-frame, the final expression becomes

\[
^R \vec{\omega}^B = \begin{bmatrix} C_5 & 0 & 0 \\ 0 & 0 & 0 \\ S_5 & 0 & 0 \end{bmatrix}_B \begin{bmatrix} u_4 \\ u_6 \\ u_7 \end{bmatrix} + \begin{bmatrix} -C_4 S_5 & 0 & 0 \\ S_4 & 1 & 0 \\ C_4 C_5 & 0 & 0 \end{bmatrix}_B \begin{bmatrix} u_3 \\ u_5 \\ u_8 \end{bmatrix}
\]  

(26)

By virtue of being fixed to body B, body A rotates about the \( b_2 \) direction. Thus angular velocity of body A with respect to body B is simply

\[
^B \vec{\omega}^A = u_6 b_2
\]  

(27)
Adding this to Equation 26 results in

\[
R \omega^A = \begin{bmatrix} C_5 & 0 & 0 \\ 0 & 1 & 0 \\ S_5 & 0 & 0 \end{bmatrix}_B \begin{bmatrix} u_4 \\ u_6 \\ u_7 \end{bmatrix} + \begin{bmatrix} -C_4S_5 & 0 & 0 \\ S_4 & 1 & 0 \\ C_4C_5 & 0 & 0 \end{bmatrix}_B \begin{bmatrix} u_3 \\ u_5 \\ u_8 \end{bmatrix}
\] (28)

Body C rotations with respect to body B about the \( c_3 \) direction

\[
^B \omega^C = u_7 \xi_3
\] (29)

Transforming the angular velocity of body B from its local coordinates to C-frame coordinates, then adding in the angular velocity of body C yields

\[
R \omega^C = \begin{bmatrix} C_4 C_5 C_7 - C_7 S_2 S_5 & 0 & 0 \\ S_4 S_3 S_7 - C_4 C_5 S_7 & 0 & 0 \\ C_4 S_5 + C_5 S_4 & 0 & 1 \end{bmatrix}_C \begin{bmatrix} u_4 \\ u_6 \\ u_7 \end{bmatrix} + \begin{bmatrix} S_4 S_7 - C_4 C_5 S_7 & S_7 & 0 \\ C_7 S_4 + C_4 C_5 S_7 & C_7 & 0 \\ C_4 C_5 - C_4 S_5 S_7 & 0 & 0 \end{bmatrix}_C \begin{bmatrix} u_3 \\ u_5 \\ u_8 \end{bmatrix}
\] (31)

Finally, body D is fixed in body C. So, it rotates with respect to body C about the \( c_2 \) direction as

\[
^C \omega^D = u_8 \xi_2
\] (31)

Simply adding this to the angular velocity of body C, and the expression for the angular velocity of body D becomes

\[
R \omega^D = \begin{bmatrix} C_4 C_5 C_7 - C_7 S_2 S_5 & 0 & 0 \\ S_4 S_3 S_7 - C_4 C_5 S_7 & 0 & 0 \\ C_4 S_5 + C_5 S_4 & 0 & 1 \end{bmatrix}_C \begin{bmatrix} u_4 \\ u_6 \\ u_7 \end{bmatrix} + \begin{bmatrix} S_4 S_7 - C_4 C_5 S_7 & S_7 & 0 \\ C_7 S_4 + C_4 C_5 S_7 & C_7 & 0 \\ C_4 C_5 - C_4 S_5 S_7 & 0 & 0 \end{bmatrix}_C \begin{bmatrix} u_3 \\ u_5 \\ u_8 \end{bmatrix}
\] (32)
3.3.6 Point Velocities

Now that the angular velocities are defined, the point velocity terms can be built up. The four bodies of the bicycle are rigid, so a relative velocity approach is used to generate the expressions. Starting with the mass center of body A

$$\dot{R}V_{GA} = \dot{R}V^P + \dot{\omega}^A \times \dot{R}G_A^P$$  \quad (33)

Or in matrix notation

$$\dot{R}V_{GA} = \dot{R}V^P + \dot{\omega}^A \dot{R}G_A^P$$  \quad (34)

The velocity of point P is set to zero to enforce the rolling-without-slip constraint. The relative velocity components can then be expanded to

$$\dot{R}V_{GA} = \begin{bmatrix} 0 & -u_4S_5 & u_5 + u_6 + u_3S_4 \\ u_4S_5 + u_3C_4C_5 & 0 & -u_4C_5 + u_3S_4 \\ -u_5 - u_6 - u_3S_4 & u_4C_5 - u_3C_4S_5 & 0 \end{bmatrix} \begin{bmatrix} \dot{L}R^B \\ 0 \\ -r_{RW} \end{bmatrix}$$  \quad (35)

Rearranging this expression to the common format,

$$\dot{R}V_{GA} = \begin{bmatrix} 0 & -r_{RW}C_5 & 0 \\ r_{RW} & 0 & 0 \\ 0 & -r_{RW}S_5 & 0 \end{bmatrix}_B \begin{bmatrix} u_4 \\ u_6 \\ u_7 \end{bmatrix} + \begin{bmatrix} -r_{RW}C_5S_4 & -r_{RW}C_5 & 0 \\ 0 & 0 & 0 \\ -r_{RW}S_4S_5 & -r_{RW}S_5 & 0 \end{bmatrix}_B \begin{bmatrix} u_3 \\ u_5 \\ u_8 \end{bmatrix}$$  \quad (36)

The center point of body A is shared with body B and allows the relative velocity approach to continue. The velocity of the mass center of body B is

$$\dot{R}V_{GB} = \dot{R}V_{GA} + \dot{\omega}^B \dot{L}_{GBA}$$  \quad (37)
Using the same approach shown in Equation 35, the velocity terms can be expanded to

\[
\mathbf{v}_{\gamma}^{G_A} = \begin{bmatrix}
0 & -r_{RW} C_5 & 0 \\
r_{RW} + z_G C_5 + x_G S_5 & 0 & 0 \\
0 & -r_{RW} S_5 & 0
\end{bmatrix}
\begin{bmatrix}
u_4 \\
u_5 \\
u_7
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
-z_G S_4 - r_{RW} C_5 S_4 & -z_G - r_{RW} C_5 & 0 \\
x_G C_4 C_5 - z_G C_4 S_5 & 0 & 0 \\
-x_G S_4 - r_{RW} S_4 S_5 & -x_G - r_{RW} S_4 & 0
\end{bmatrix}
\begin{bmatrix}
u_3 \\
u_4 \\
u_8
\end{bmatrix}
\]

(38)

Next is the velocity of \( S \), the common point between body \( B \) and body \( C \). Because it is fixed in body \( B \), its position vector is expressed in the same form as the position vector to \( G_B \). This is clear when comparing Equation 14 and Equation 15. The only difference is the subscript of the elements, so the velocity of \( S \) is simply

\[
\mathbf{v}_{\gamma}^{S} = \mathbf{v}_{\gamma}^{G_A} + \frac{\mathbf{\omega}}{r} \mathbf{v}_{\gamma}^{G_C/S}
\]

(39)

Which expands, in a similar way, to

\[
\mathbf{v}_{\gamma}^{S} = \begin{bmatrix}
0 & -r_{RW} C_5 & 0 \\
r_{RW} + z_G C_5 + x_G S_5 & 0 & 0 \\
0 & -r_{RW} S_5 & 0
\end{bmatrix}
\begin{bmatrix}
u_4 \\
u_5 \\
u_7
\end{bmatrix}
\]

\[
+ \begin{bmatrix}
-z_G S_4 - r_{RW} C_5 S_4 & -z_G - r_{RW} C_5 & 0 \\
x_G C_4 C_5 - z_G C_4 S_5 & 0 & 0 \\
-x_G S_4 - r_{RW} S_4 S_5 & -x_G - r_{RW} S_4 & 0
\end{bmatrix}
\begin{bmatrix}
u_3 \\
u_4 \\
u_8
\end{bmatrix}
\]

(40)

Now that the velocity of point \( S \) is established, the rest of the velocities terms are expressed in the body \( C \) frame. The velocity of the mass center of body \( C \) is

\[
\mathbf{v}_{\gamma}^{G_C} = \mathbf{v}_{\gamma}^{S} + \frac{\mathbf{\omega}}{r} \mathbf{v}_{\gamma}^{G_C/S}
\]

(41)

At this point, the expressions become rather large, so a new format is presented to save space

\[
\mathbf{v}_{\gamma}^{G_C} = \begin{bmatrix}
S_7 (r_{RW} - z_G C_{\lambda_5} + z_G C_5 + x_G S_5) & -r_{RW} C_{\lambda_5} C_7 & 0 \\
C_7 (r_{RW} + z_G C_5 + x_G S_5) + x_G S_7 - z_G C_{\lambda_5} C_7 & r_{RW} C_{\lambda_5} S_7 & x_G \\
x_G C_{\lambda_5} S_7 & -r_{RW} S_{\lambda_5} & 0
\end{bmatrix}
\begin{bmatrix}
u_4 \\
u_6 \\
u_7
\end{bmatrix}
\]

(44)
Then in expanded form, velocity is simply the mass center of body C are the subscripts of the position vectors. Therefore, this and this:

\[
\begin{bmatrix}
  V_{C11} & V_{C12} & 0 \\
  V_{C21} & V_{C22} & 0 \\
  V_{C31} & V_{C32} & 0
\end{bmatrix}
\begin{bmatrix}
  u_3 \\
  u_5 \\
  u_8
\end{bmatrix} = \begin{bmatrix}
  V_{C11} & V_{C12} & 0 \\
  V_{C21} & V_{C22} & 0 \\
  V_{C31} & V_{C32} & 0
\end{bmatrix}_C
\]

(42)

Where

\[
V_{C11} = z_C (C_7 S_4 + C_7 C_4 S_5 + C_7 S_4 S_7) + C_7 S_4 (x_3 C_5 - z_3 S_5) - C_7 C_4 S_4 (z_5 + r_{RW} S_5) + C_7 S_4 (x_5 + r_{RW} S_5)
\]

\[
V_{C21} = z_C (C_7 C_4 S_5 - S_4 S_7) + C_7 C_4 S_4 (x_5 - z_5 S_5) + C_7 S_4 S_4 (z_5 + r_{RW} S_5) - S_4 S_4 (x_5 + r_{RW} S_5)
\]

\[
V_{C31} = -x_C (C_7 S_4 + C_7 C_4 S_5 + C_7 S_4 S_7) - C_7 S_4 (x_5 + r_{RW} S_5) - S_4 S_4 (z_5 + r_{RW} S_5)
\]

\[
V_{C12} = C_7 (z_C - r_{RW} C_{\lambda+5} - z_C C_{\lambda} + x_5 S_{\lambda})
\]

\[
V_{C22} = -S_7 (z_C - r_{RW} C_{\lambda+5} - z_C C_{\lambda} + x_5 S_{\lambda})
\]

\[
V_{C32} = -r_{RW} S_{\lambda+5} - x_3 C_{\lambda} - x_C C_7 - z_3 S_{\lambda}
\]

The velocity of the \( G_D \) is also a fixed point on body C. This is the same situation as for points \( G_B \) and S. Again, the only difference between the velocity of the mass center of body D, and the velocity of the mass center of body C are the subscripts of the position vectors. Therefore, this velocity is simply

\[
\begin{align*}
\dot{R}_{V_{G_D}} &= \dot{R}_{V_{S}} + \dot{R}_{\omega} C_{G_D/S} \\
\end{align*}
\]

(43)

Then in expanded form,

\[
\begin{bmatrix}
  V_{D11} & V_{D12} & 0 \\
  V_{D21} & V_{D22} & 0 \\
  V_{D31} & V_{D32} & 0
\end{bmatrix}
\begin{bmatrix}
  u_3 \\
  u_5 \\
  u_8
\end{bmatrix}
\]

(44)
Where

\[ VD_{11} = z_D(S_4 + C_5 S_7 + C_4 C_6 S_4 S_7 + C_4 S_4 (x_5 - z_5 S_5)) - C_4 S_4 (z_5 + r_{R_5} S_5) + C_7 S_7 4 (x_5 + r_{R_5} S_5) \]

\[ VD_{21} = z_5(S_4 C_5 + S_5 S_7 + C_4 C_5 S_4 S_7 + x_5 S_4 S_7 + C_4 S_4 (x_5 - z_5 S_5)) + C_7 S_7 4 (z_5 + r_{R_5} S_5) - S_4 S_7 (x_5 + r_{R_5} S_5) \]

\[ VD_{31} = -x_D(S_4 + C_5 C_6 S_4 S_7 + C_4 C_5 S_4 S_7 - S_4 S_7 (x_5 + r_{R_5} S_5)) - S_4 S_7 (z_5 + r_{R_5} C_5) \]

\[ VD_{12} = C_7(z_5 - r_{R_5} C_{5,5} - z_D C_5 + x_5 S_5) \]

\[ VD_{22} = -S_7(z_5 - r_{R_5} C_{5,5} - z_D C_5 + x_5 S_5) \]

\[ VD_{32} = -r_{R_5} S_{5,1} - x_5 C_5 - x_5 C_7 - z_5 S_5 \]

The velocity of the front wheel contact point, \( Q \), is the final term in the velocity kinematics. It is solved by taking advantage of the fact that \( G_D \) is a common point between body \( C \) and body \( D \), and that its position vector is fixed in \( C \)-frame coordinates. Its velocity is set to zero to enforce the non-slip condition, resulting in

\[ R_{\chi} Q = R_{\chi} G_D + \omega_{\chi} G_D / G_0 \]

Formulating this in the common form

\[
\begin{bmatrix}
0 \\
0 \\
0 \\
\end{bmatrix}
=
\begin{bmatrix}
VQ_{111} & VQ_{121} & 0 \\
VQ_{121} & VQ_{122} & VQ_{123} \\
VQ_{131} & VQ_{132} & 0 \\
\end{bmatrix}
\begin{bmatrix}
u_4 \\
u_6 \\
u_7 \\
\end{bmatrix}
+
\begin{bmatrix}
VQ_{D11} & VQ_{D12} & VQ_{D13} \\
VQ_{D21} & VQ_{D22} & 0 \\
VQ_{D31} & VQ_{D32} & VQ_{D33} \\
\end{bmatrix}
\begin{bmatrix}
u_5 \\
u_5 \\
u_8 \\
\end{bmatrix}
\]

Where

\[ VQ_{111} = S_7(r_{R_5} + z_5 C_5 + x_5 S_5) - z_D C_{5,5} S_7 - (r_{R_5} C_{5,5})^2 C_4 S_7 / #1 \]

\[ VQ_{112} = -r_{R_5} C_{5,5} C_7 \]

\[ VQ_{121} = \left( \begin{array}{c}
C_7(r_{R_5} + z_5 C_5 + x_5 S_5) + x_5 S_4 S_7 - z_D C_{5,5} C_7 - (r_{R_5} S_{5,5} (C_4 C_4 S_7 S_5 - ) \\
S_4 S_7 + C_4 C_4 C_7 S_5) / #1 - (r_{R_5} C_4^2 C_4 C_7) / #1 \\
\end{array} \right) \]

\[ VQ_{122} = r_{R_5} C_{5,5} S_7 \]

\[ VQ_{123} = x_D - (r_{R_5} (C_5 C_4 S_7 S_5 - S_4 S_7 + C_4 C_5 C_7 S_5)) / #1 \]
\[ VQ_{131} = x_D C_{4+3} S_7 - (r_{FW} C_{4+3} S_7 (C_{4} C_{4} C_{7} S_5 - S_4 S_7 + C_{4} C_{5} C_{7} C_{\lambda})) / #1 \]
\[ VQ_{132} = -r_{RW} S_{4+3} \]
\[ \#1 = ((C_{4} C_{4} C_{7} S_5 - S_4 S_7 + C_{4} C_{5} C_{7} C_{\lambda})^2 + C_{4+3}^2 C_{4}^2)^{1/2} \]

And

\[ VQ_{D11} = \left( z_D (C_{4} S_4 + C_{4} S_{4} S_7 + C_{4} C_{7} S_7) + C_{4} S_7(x_3 S_5 - z_5 S_7) - C_{4} C_{7} S_4 (z_5 + r_{RW} S_5) + \right) \]
\[ C_{7} S_{2} S_4(x_3 + r_{FW} S_3) + r_{FW} C_{4+3} C_4 (C_{7} S_4 + C_{4} C_{5} S_7 + C_{4} C_{5} S_{2} S_7) / #1 \]
\[ VQ_{D12} = z_D C_{7} - C_{4} C_{7} (z_5 + r_{RW} C_5) + C_{7} S_2 (x_3 + r_{RW} S_5) + (r_{FW} C_{4+3} C_4 C_7) / #1 \]
\[ VQ_{D13} = (r_{FW} C_{4+3} C_4) / #1 \]
\[ VQ_{D21} = \left( z_D (C_{4} C_{4} C_{7} S_5 - S_4 S_7 + C_{4} C_{5} C_{7} S_7) S_2 S_7(x_3 + r_{RW} S_5) \right) \]
\[ VQ_{D22} = C_{4} S_7(z_5 + r_{RW} C_5) - z_D S_7 - S_2 S_7(x_3 + r_{RW} S_5) - (r_{FW} C_{4+3} S_7) / #1 \]
\[ VQ_{D31} = \left( (r_{FW} C_{4} C_{4} C_{7} S_5 - S_4 S_7 + C_{4} C_{5} C_{7} S_7) C_{7} S_4 + C_{4} C_{5} S_7 + C_{4} C_{5} S_{2} S_7) / #1- \right) \]
\[ C_{4} S_4(x_3 + r_{RW} S_5) - S_2 S_4(z_5 + r_{RW} C_5) - x_D (C_{7} S_4 + C_{4} C_{5} S_7 + C_{4} C_{5} S_{2} S_7) \]
\[ VQ_{D32} = (r_{FW} C_{7} (C_{4} C_{4} C_{7} S_5 - S_4 S_7 + C_{4} C_{5} C_{7} S_7)) / #1-C_{4}(x_3 + r_{RW} S_5) - S_2(z_5 + r_{RW} C_5) x - x_D C_7 \]
\[ VQ_{D33} = (r_{FW} C_{4} C_{4} C_{7} S_5 - S_4 S_7 + C_{4} C_{5} C_{7} S_7)) / #1 \]
\[ \#1 = ((C_{4} C_{4} C_{7} S_5 - S_4 S_7 + C_{4} C_{5} C_{7} S_7)^2 + C_{4+3}^2 C_{4}^2)^{1/2} \]

### 3.3.7 Motion Constraints

Kane’s method requires taking the partial derivatives of the velocity terms with respect to the independent generalized speeds. In their current form, the velocity terms are functions of both the independent speeds and the dependent speeds. Thus, the dependent speeds must be solved as functions of the independent speeds (and the generalized coordinates). The general form of this result is

\[
\begin{align*}
u_3 &= f_3(u_4, u_6, u_7, q_4, \ldots, q_7) \\
u_5 &= f_5(u_4, u_6, u_7, q_4, \ldots, q_7) \\
u_6 &= f_6(u_4, u_6, u_7, q_4, \ldots, q_7)
\end{align*}
\]
Recall that each wheel of the bicycle is assumed to be in contact with a horizontal surface. Each contact is assumed to be at a single point and no slipping occurs. To enforce these principles analytically, constraint equations are developed. There are three constraints on the rear wheel contact point, and three on the front wheel contact point.

Beginning with point $P$, the first holonomic constraint is defined in Equation 13. The two additional constraints, which are nonholonomic, are enforced by setting the velocity vector of $P$ equal to zero. This is done in Equation 33. For point $Q$, the three constraints are developed by building the velocity vectors from $P$ to $Q$. Setting $Q$ to zero enforces these final constraints (Equation 45). This velocity vector has three non-zero elements. These elements define two nonholonomic constraints and the derivative of a single holonomic constraint. Again, the two nonholonomic constraints are derived from the nonslip condition of $Q$, and the holonomic constraint from its mandatory contact with the ground plane. Equation 46 is redefined into a more compact form by introducing two new terms, the $C_1$ and $C_2$ matrices. The new equation is expressed as

$$
\begin{bmatrix}
0 \\
0 \\
0 \\
\end{bmatrix} = [C_1]_c \begin{bmatrix}
u_4 \\
u_6 \\
u_7 \\
\end{bmatrix} + [C_2]_c \begin{bmatrix}
u_3 \\
u_5 \\
u_8 \\
\end{bmatrix}
$$

Rearranging this equation solves for the dependent generalized speeds in terms of the independent ones. The final result is

$$
\begin{bmatrix}
u_3 \\
u_5 \\
u_8 \\
\end{bmatrix} = -[C_2]^{-1} [C_1] \begin{bmatrix}
u_4 \\
u_6 \\
u_7 \\
\end{bmatrix} = \begin{bmatrix}
u_4 \\
u_6 \\
u_7 \\
\end{bmatrix}
$$

Notice that after solving for the dependent speeds, the combination of $C$ matrices simplifies to the $J$ matrix, or the Jacobian matrix. The expanded expression is too large to show, and not terribly helpful to view. The goal of this derivation is to generate a computational model, so it is not
necessary to symbolically compute the $J$ matrix. The program solves the symbolic forms of the $C$ matrices numerically, and then computes the $J$ matrix. This is simpler and more efficient.

The constraint equations are then differentiated with respect to time. This is necessary to solve for the dependent $\dot{u}$ terms, expressed as

$$
\begin{align*}
\dot{u}_3 &= \dot{f}_3(u_4, \dot{u}_6, \dot{u}_7, u_4, u_6, u_7, q_4, \ldots, q_7) \\
\dot{u}_5 &= \dot{f}_5(u_4, \dot{u}_6, \dot{u}_7, u_4, u_6, u_7, q_4, \ldots, q_7) \\
\dot{u}_8 &= \dot{f}_8(u_4, \dot{u}_6, \dot{u}_7, u_4, u_6, u_7, q_4, \ldots, q_7)
\end{align*}
$$

(50)

This is accomplished by simply differentiating Equation 48 with respect to time, which yields

$$
\begin{bmatrix}
\dot{C}_1 \\
C_1 \\
\dot{C}_2 \\
C_2
\end{bmatrix}
\begin{bmatrix}
\dot{u}_4 \\
\dot{u}_6 \\
\dot{u}_7
\end{bmatrix} + [C_1]
\begin{bmatrix}
\dot{u}_4 \\
\dot{u}_6 \\
\dot{u}_7
\end{bmatrix} + [\dot{C}_2]
\begin{bmatrix}
\dot{u}_3 \\
\dot{u}_5 \\
\dot{u}_8
\end{bmatrix} + [C_2]
\begin{bmatrix}
\dot{u}_3 \\
\dot{u}_5 \\
\dot{u}_8
\end{bmatrix} =
\begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix}
$$

(51)

Rearranging this equation, and taking advantage of the $J$ matrix, results in

$$
\begin{bmatrix}
\dot{u}_3 \\
\dot{u}_5 \\
\dot{u}_8
\end{bmatrix} = -[C_2]^{-1}
\begin{bmatrix}
\dot{C}_1 \\
C_1 \\
\dot{C}_2 \\
C_2
\end{bmatrix}
\begin{bmatrix}
\dot{u}_4 \\
\dot{u}_6 \\
\dot{u}_7
\end{bmatrix} + [\dot{C}_2]
\begin{bmatrix}
\dot{u}_3 \\
\dot{u}_5 \\
\dot{u}_8
\end{bmatrix} + [J]
\begin{bmatrix}
\dot{u}_4 \\
\dot{u}_6 \\
\dot{u}_7
\end{bmatrix}
$$

(52)

Substituting in Equation 49 (to remove the dependent speeds) results in an expression that satisfies Equation 50, which is

$$
\begin{bmatrix}
\dot{u}_3 \\
\dot{u}_5 \\
\dot{u}_8
\end{bmatrix} = [\dot{J}]
\begin{bmatrix}
\dot{u}_4 \\
\dot{u}_6 \\
\dot{u}_7
\end{bmatrix} + [J]
\begin{bmatrix}
\dot{u}_4 \\
\dot{u}_6 \\
\dot{u}_7
\end{bmatrix}
$$

(53)

Where 

$$
[\dot{J}] = -[C_2]^{-1}[\dot{C}_1] + [C_2]^{-1}[\dot{C}_2][J]
$$

49
The time-derivatives of the C matrices are quite long, and will not be shown here to save space. The full expressions of these matrices are shown in Appendix D.

### 3.3.8 Angular Accelerations

The angular accelerations are calculated by simply taking the derivative of the angular velocities. These equations become very long and unlike the velocity terms they are not necessary to express in their full form. The velocities are critical in the build-up of the constraint equations, so they were shown fully. The acceleration terms are merely used to complete the dynamical equations of motion. For this reason, they will not be expressed in the body of this thesis. However, to maintain consistency these terms will be fully shown in Appendix E.

The angular velocity of body A, as developed in Equation 19, is expressed in B-frame coordinates. Therefore, by taking the derivative of this vector in its rotating reference frame, called the derivative rule [16], the expression takes the form

$$\dot{\alpha}^A = \frac{d}{dt} (\omega^A) + \alpha^B \omega^A$$  \hspace{1cm} (54)

The angular velocity of body B is expressed in its local coordinates, so the angular acceleration is developed by simply taking the time derivative. This takes the form

$$\dot{\alpha}^B = \frac{d}{dt} (\omega^B)$$  \hspace{1cm} (55)

Both body C and body D angular velocities are expressed in C-frame coordinates. For body C, this requires a simple differentiation, similar to Equation 55. Body D requires application of the derivative rule. These two angular accelerations are expressed as

$$\dot{\alpha}^C = \frac{d}{dt} (\omega^C)$$  \hspace{1cm} (56)

$$\dot{\alpha}^D = \frac{d}{dt} (\omega^D) + \omega^C \omega^D$$  \hspace{1cm} (57)
3.3.9 Point Accelerations

The final set of kinematic equations is that of the point accelerations. They are derived in a similar manner to the point velocities, using relative accelerations. These expressions become very long as well, so only the general equations are developed here. The full expressions are shown in Appendix F. The mass center of body A is first derived using direct differentiation, taking advantage of the derivative rule. This is expressed as

\[ R \alpha^G_A = \frac{b}{d} \frac{d}{dt} (R \nu^G_A) + R \omega^B R \nu^G_A \]  

(58)

The rest of the accelerations proceed in the same way as the velocities. From the mass center of body A, the acceleration of the mass center of body B is derived as

\[ R \alpha^G_B = R \alpha^G_A + R \omega^B r^{G_B/G_A} + R \omega^B r^{G_B/G_A} \]  

(59)

Calculating the acceleration of point S is done in a similar manner to Equation 59. This expression is defined as

\[ R \alpha^S = R \alpha^G_A + R \omega^B r^{S/G_A} + R \omega^B r^{S/G_A} \]  

(60)

Transitioning to body C, the two final point acceleration terms are that of the mass center of body C and the mass center of body D. These are represented in Equations 61 and 62.

\[ R \alpha^G_C = R \alpha^S + R \omega^C r^{G_C/S} + R \omega^C (R \omega^C r^{G_C/S}) \]  

(61)

\[ R \alpha^G_D = R \alpha^S + R \omega^C r^{G_D/S} + R \omega^C (R \omega^C r^{G_D/S}) \]  

(62)
3.4 Kinetics

3.4.1 Mass Properties

Before developing the final set of dynamical equations, it is important to define the mass properties of each of the four bodies. Each body has its own symbolic mass term, \( m_i \) where \( i = A,\ldots,D \). The inertia properties of each body are defined based on the assumptions of the Whipple model. Both wheels are axisymmetric and although they are considered to have knife-edge contact with the ground, they are assumed to have a non-zero thickness. This provision accounts for more realistic wheels. Because of the wheel symmetry, \( I_{Ax} = I_{Az} \) for the front wheel, and \( I_{Dx} = I_{Dz} \). The inertia tensors for the rear wheel (body A) and the front wheel (body D) are expressed as

\[
I_A = \begin{bmatrix}
I_{Ax} & 0 & 0 \\
0 & I_{Ay} & 0 \\
0 & 0 & I_{Az}
\end{bmatrix}
\] (63)

\[
I_D = \begin{bmatrix}
I_{Dx} & 0 & 0 \\
0 & I_{Dy} & 0 \\
0 & 0 & I_{Dz}
\end{bmatrix}
\] (64)

Both the rear frame (body B) and the front frame (body C) are assumed to have a vertical plane of symmetry, thereby making the \( I_{xy} \) and \( I_{yz} \) terms zeros. The resulting inertia tensors for these two bodies are

\[
I_B = \begin{bmatrix}
I_{Bx} & 0 & -I_{Bz} \\
0 & I_{By} & 0 \\
-I_{Bz} & 0 & I_{Bz}
\end{bmatrix}
\] (65)
3.4.2 Inertia Forces

For the bicycle’s four bodies, the inertia force components of Kane’s equations are expressed (from [16]) as

\[
L_e = \begin{bmatrix}
    I_{e_{xx}} & 0 & -I_{e_{xz}} \\
    0 & I_{e_{yy}} & 0 \\
    -I_{e_{xz}} & 0 & I_{e_{zz}}
\end{bmatrix}
\]

(66)

In this expression, \(u_F\) represents the independent generalized speeds, so \(k = 4,6,7\). In order to generate the inertia forces, the partial velocities and partial angular velocities are computed. The form of the velocity terms makes this a convenient operation. The constraint equations developed in Equation 49 are substituted in for the dependent speeds vector. For the velocity of the mass center of body A, the expression becomes

\[
F^* = - \left( \sum_{i=A}^D m_i g_i^G \frac{\partial v_i^G}{\partial u_k} \right) + \sum_{i=A}^D \left[ (L_i \cdot \omega_i^G) + \omega_i^G \times (L_i \cdot \omega_i^G) \right] \frac{\partial \omega_i^G}{\partial u_k}
\]

(67)

In this expression, \(u_k\) represents the independent generalized speeds, so \(k = 4,6,7\). In order to generate the inertia forces, the partial velocities and partial angular velocities are computed. The form of the velocity terms makes this a convenient operation. The constraint equations developed in Equation 49 are substituted in for the dependent speeds vector. For the velocity of the mass center of body A, the expression becomes

\[
R v^G_A = \begin{bmatrix}
    0 & -r_{RW} C_5 & 0 \\
    r_{RW} & 0 & 0 \\
    0 & -r_{RW} S_5 & 0
\end{bmatrix}_B + \begin{bmatrix}
    -r_{RW} C_5 S_4 & -r_{RW} C_5 & 0 \\
    0 & 0 & 0 \\
    -r_{RW} S_4 S_5 & -r_{RW} S_5 & 0
\end{bmatrix}_B \begin{bmatrix}
    J \\
    u_4 \\
    u_6 \\
    u_7
\end{bmatrix}
\]

(68)

And for the angular velocity of body A, the expression becomes

\[
R \omega^A = \begin{bmatrix}
    C_5 & 0 & 0 \\
    0 & S_4 & 0 \\
    S_5 & 0 & 0
\end{bmatrix}_B + \begin{bmatrix}
    -C_4 S_5 & 0 & 0 \\
    0 & C_5 & 0 \\
    0 & 0 & C_4 C_5
\end{bmatrix}_B \begin{bmatrix}
    J \\
    u_4 \\
    u_6 \\
    u_7
\end{bmatrix}
\]

(69)

This same procedure is followed for each mass center velocity and each angular velocity. Those results are not shown to save space.
The final aspect in generating these inertia forces is the consideration to reference frames. Typically, the inertia of each body is expressed in the local reference frame because that is where it is constant. In the case of the bicycle a simplification can be made. Body A is an axisymmetric wheel that rotates about the \( b_2 \)-direction, so its inertia does not change with respect to the body B reference frame. The same relationship is true between body D and body C, as body D is the wheel rotating about the \( c_2 \)-direction. Therefore, all the kinematic terms and partial velocities of body A are expressed in the B-frame coordinates, and the kinematic terms and partial velocities of body D are expressed in C-frame coordinates. This simplifies the three dynamical equations.

### 3.4.3 Active Forces

In this model, the only forces assumed present are those of gravity on the four bodies. These are expressed as

\[
\begin{align*}
F_A &= m_A g n_3 \\
F_B &= m_B g n_3 \\
F_C &= m_C g n_3 \\
F_D &= m_D g n_3
\end{align*}
\]  (70)

Because this formulation is taking advantage of the constant wheel inertias, only the B-frame and C-frame are necessary to express these forces. \( F_A \) and \( F_B \) are expressed in the B-frame as

\[
F_A = m_A g \begin{bmatrix}
-C_4 S_5 \\
S_4 \\
C_4 C_5
\end{bmatrix}_B 
\]  (71)

\[
F_B = m_B g \begin{bmatrix}
-C_4 S_5 \\
S_4 \\
C_4 C_5
\end{bmatrix}_B
\]  (72)
Then, forces $F_C$ and $F_D$ are expressed in the $C$-frame as

$$F_C = m_C g \begin{bmatrix} -(C_4C_7S_5 - S_4S_7 + C_4C_5C_7S_5) \\ C_7S_4 + C_4C_4S_5S_7 + C_4C_5S_5S_7 \\ C_4+C_5 \end{bmatrix}_C$$

(73)

$$F_D = m_D g \begin{bmatrix} -(C_4C_4C_7S_5 - S_4S_7 + C_4C_5C_7S_5) \\ C_7S_4 + C_4C_4S_5S_7 + C_4C_5S_5S_7 \\ C_4+C_5 \end{bmatrix}_C$$

(74)

There are three moments that act on the bicycle. The rear wheel moment, which simulates a pedaling torque, the roll moment and the steering moment. The rear wheel moment acts on body $A$ about the $b_2$-direction. The roll moment acts on body $B$ about the $l_1$-direction (the lean frame). The steering moment acts on body $C$ about the $c_3$-direction. The moments expressed in equation form are

$$T_{rw} = \begin{bmatrix} 0 & T_{rw} & 0 \end{bmatrix}^T_B$$

$$T_R = \begin{bmatrix} T_R & 0 & 0 \end{bmatrix}^T_L$$

$$T_S = \begin{bmatrix} 0 & 0 & T_S \end{bmatrix}^T_C$$

(75)

Body A has only the rear wheel moment acting on it, expressed as

$$M_A = \begin{bmatrix} 0 \\ T_{rw} \\ 0 \end{bmatrix}$$

(76)

Body B has the lean moment acting on it, as well as reactionary moments from the rear wheel moment, and the steering moment. The full expression of these moments on body B is
Body C has only the steering moment acting upon it, expressed as

\[ M_C = \begin{bmatrix} 0 \\ 0 \\ T_s \end{bmatrix}_{c} \]  

(78)

There are no moments that act on body D which, for completeness, take the form

\[ M_D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]  

(79)

Finally, the total contributions of all the forces and moments that act on the system are expressed as the active forces, following the form from [16] as

\[ F = \sum_{j=1}^{4} \left( F_j \cdot \frac{\partial \mathbf{v}^G_j}{\partial u_k} \right) + \sum_{j=1}^{4} \left( M_j \cdot \frac{\partial \mathbf{q}^j}{\partial u_k} \right) \]  

(80)

3.4.4 Kane’s Equations

The classic form of Kane’s equations is combination of the active forces and inertia force. They can be expressed as

\[ F + F^* = 0 \]  

(81)

From this, a minimum set of dynamical equations are derived. There are three due to the three degrees of freedom of the system. These three are supplemented with an additional six kinematical differential equations to complete the equations of motion.
From a computational standpoint, these equations are broken down and solved for in pieces. First, the computational model relies on the numerical integration of first-order differential equations. The three dynamical equations are highly nonlinear in their full form, but they are linear in terms of the time derivatives of the independent generalized speeds. Collecting the coefficients of these independent speeds and rearranging the remaining terms, the equations take the form

$$
\begin{bmatrix}
\ddot{u}_x \\
\ddot{u}_o \\
\ddot{u}_r
\end{bmatrix}
= \begin{bmatrix}
M_{11} & M_{12} & M_{13} \\
M_{21} & M_{22} & M_{23} \\
M_{31} & M_{32} & M_{33}
\end{bmatrix}
= \begin{bmatrix}
F_1 \\
F_2 \\
F_3
\end{bmatrix}
$$

(82)

The mass matrix, \([M]\), is the three-by-three collection of the coefficients. All remaining terms are grouped into the F-matrix, which is three-by-one. The mass matrix is then inverted, resulting in the necessary three first-order differential equations.

The kinematical differential equations, per their definition, are simply the time derivatives of the generalized coordinates. They are expressed as

$$
\dot{q}_i = u_i
$$

(83)

where \(i = 3, \ldots, 8\). In total, this is a complete set of nine first order differential equations that represent equations of motion for the bicycle.

3.5 Computer Program

The computational dynamic model is a set of script and function files, all coded in MATLAB. There are two main scripts in this program. The first symbolically derives the equations of motion, and the second numerically evaluates them.

3.5.1 Symbolic Equation Build-up

The equations presented in 3.3 and 3.4 are developed using the MATLAB symbolic toolbox. These equations are derived using a single script file, ‘Bicycle_EOMs_v3.m’, and two supporting function files. These files are shown in Appendix G.
The script starts by defining the symbolic variables that are used to model the bicycle. These consist of the generalized coordinates, generalized speeds, and geometric parameters. It first builds up all the individual rotation matrices and position vectors. Then, it calculates the relative position vector between the rear wheel contact point, \( P \), and the front wheel contact point, \( Q \). This is expressed as

\[
\mathbf{r}^{Q/P} = \mathbf{r}^{G_3/P} + \mathbf{r}^{S/G_3} + \mathbf{r}^{G_0/S} + \mathbf{r}^{Q/G_0}
\]

These components are defined in several reference frames, but they are transformed (as needed) so the vector is expressed in the yaw frame \([15]\).

The bicycle (based on its mathematical definition) can have several starting configurations, only one of which is correct. This correct configuration is with the bottom of both wheels touching the ground, and the bodies of the bicycle above the ground plane (See Figure 19). When running the simulation, it is critical to properly define this initial configuration. This is done by equating the third element of Equation 84 (the \( y_3 \) component) to zero. It is expressed fully as

\[
\mathbf{r}^{Q/P} \cdot y_3 = \xi_0 C_{\lambda_5} C_4 - x_0 (C_4 C_4 C_\gamma S_5 - S_4 S_\gamma + C_4 C_5 S_\lambda) - r_{R/W} C_4 - z_0 C_4 C_5
\]

\[
x_5 C_4 S_5 + r_{P/W} (C_4 C_4 C_\gamma S_5 - S_4 S_\gamma + C_4 C_5 C_\gamma S_\lambda)^2 / #1 = 0
\]

Where

\[
#1 = ((C_4 C_4 C_\gamma S_5 - S_4 S_\gamma + C_4 C_5 C_\gamma S_\lambda)^2 + C_4^2 C_\lambda S_4)^{1/2}
\]

This equation develops a nonlinear relationship between the bicycle roll angle, the pitch angle, and the steering angle \((q_4, q_5, \text{and } q_7)\). In order to define the configuration of the bicycle, two of these parameters are independently chosen, then the equation is solved for the third variable. This model is defined so the pitch angle is the dependent variable. For simulation, the initial values of the roll angle and the steering angle are used to solve for the initial condition of the pitch angle.

Next, the program calculates the angular velocities and the point velocities. The angular velocities are derived by simply summing the individual rotations of each body, and the point velocities are derived using the relative velocity approach. It finishes the kinematics by calculating the angular accelerations and point accelerations. The angular accelerations are derived using direct differentiation, and the point accelerations by using the relative acceleration approach. Next,
the $C$ matrices are generated and differentiated with respect to time. Finally, the program builds up the active forces. The script saves only Equation 85 and the symbolic variables that it define it.

The equations developed by the program are used to build function files. It was found that numerically evaluating the equations as symbolic functions is very computationally intensive and inefficient. So, these function files allow the variables in the equations to be calculated numerically instead of symbolically.

First, the kinematic vector equations are manually transformed to matrix format (defined in 3.3.4) by using the ‘SymVector2Matrix.m’ file. These output is then saved from the MATLAB command window to a text file. The $C$ matrices and active forces do not require any additional formatting and are simply output to the command window and saved to individual text files. The contents of these text files are then pasted into a series of function files. With the generation of these function files, ‘Bicycle_EOMs_v3.m’ completes its purpose.

### 3.5.2 Simulation

The simulation program operates from a single script file, ‘Bike_Sim_Main_v4’, and the set of function files generated by ‘Bicycle_EOMs_v3.m’. In its current form, the simulation is set up to be an initial value problem. The six generalized coordinates and the three independent generalized speeds comprise the nine initial values of the bicycle. Once these are defined the simulation uses a fixed-step Runge-Kutta 4th order routine to numerically integrate the equations of motion [17]. This numerical integration process produces the time-histories of the nine dynamic states of the bicycle which are then plotted for easy visualization. Finally, the script provides the option to save the simulation results to a dedicated .mat file. The simulation script and all the associated function files are shown in Appendix H.

For the purposes of this thesis, the goal was to generate a working model and verify its correctness. This verification is based on reproducing the results of a test case that is common in the bicycle modeling literature. This test case and the physical parameters of a benchmark bicycle are presented by Meijaard [13]. Before running the simulation, these bicycle parameters are converted to match the parameters of the model presented in this thesis. After the set-up is complete, the simulation uses ‘BIKE_EOM_FUN_v4.m’ to develop and evaluate the equations of motion.
Parameter Conversion

The physical parameters from Meijaard are summarized in Table 4 and Table 5 [13]. These parameters define the geometric and mass properties of the bicycle. The geometric properties are measured with the bicycle in its nominal configuration. All distances are measured from the rear wheel contact point and follow the same sign convention as the current model (x-direction positive to the right, z-direction positive down).

Table 4: Bicycle geometry parameters

<table>
<thead>
<tr>
<th>Wheelbase [m]</th>
<th>Trail [m]</th>
<th>Steering Tilt Angle ((\lambda)) [deg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.02</td>
<td>0.08</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 5: Bicycle mass parameters

<table>
<thead>
<tr>
<th>Bodies</th>
<th>Mass [kg]</th>
<th>Inertia [kg-m(^2)]</th>
<th>Location of mass center (x, z) [m]</th>
</tr>
</thead>
</table>
| Rear Wheel              | 2         | \[
|                         |           | 0.0603 0 0
|                         |           | 0 0.12 0
|                         |           | 0 0 0.0603
|                         |           | \]                                                                                  | \(0, r_{RW}\) \(0, -0.3\)        |
| Rear Bicycle Frame      | 85        | \[
|                         |           | 9.2 0 2.4
|                         |           | 0 11 0
|                         |           | 2.4 0 2.8
|                         |           | \]                                                                                  | \(x_B, z_B\) \(0.3, -0.9\)       |
| Front Bicycle Frame     | 4         | \[
|                         |           | 0.05892 0 -0.00756
|                         |           | 0 0.06 0
|                         |           | -0.00756 0 0.00708
|                         |           | \]                                                                                  | \(x_H, z_H\) \(0.9, -0.7\)       |
| Front Wheel             | 3         | \[
|                         |           | 0.1405 0 0
|                         |           | 0 0.28 0
|                         |           | 0 0 0.1405
|                         |           | \]                                                                                  | \(w, r_{FW}\) \(1.02, -0.35\)   |
Figure 21: Geometry used for parameter conversion

Figure 21 presents the basic bicycle geometry and dimensions that are used to perform the parameter conversion. Point $S$ is located such that a line connecting it to the rear wheel mass center makes a right angle with the steering axis. These lines form two legs of a triangle, and the third leg is formed by a horizontal line. Using this relationship, a series of equations are developed wherein the necessary dimensions for the current model can be calculated. The derivation of these equations is not presented here. Expressed below are the six parametric conversion equations

\[ x_S = C_\lambda \left( c + w - r_{RW} T_\lambda \right) \]  \hspace{1cm} (86)

\[ z_S = T_\lambda x_S \]  \hspace{1cm} (87)

\[ x_D = -C_\lambda \left( c - r_{FW} T_\lambda \right) \]  \hspace{1cm} (88)

\[ z_D = \frac{r_{RW} + z_S - r_{FW} + x_D S_\lambda}{C_\lambda} \]  \hspace{1cm} (89)

\[ x_C = \left( x_H - x_S \right) C_\lambda - \left( r_{RW} + z_S - z_H \right) S_\lambda \]  \hspace{1cm} (90)
\[ z_c = \frac{x_H - x_S - x_c C_\lambda}{S_\lambda} \]  \hspace{1cm} (91)

Where \( c \) is the trail and \( w \) is the wheelbase

These equations are evaluated with the parameters given in Tables 4 and 5. The geometric dimensions for the current model are presented in Table 6. Next, the mass properties are converted to the current model. While the mass values of each body can be directly used, the inertia matrices cannot. Each of the four bodies of the benchmark bicycle use the generic coordinate frame shown in Figure 21. This coordinate system is consistent with the rear wheel, rear bicycle frame and front wheel of the current model. However, the front bicycle frame in this model uses a rotated coordinate system, and therefore its inertia matrix is transformed accordingly.

<table>
<thead>
<tr>
<th>Geometric Parameters</th>
<th>Numeric Value [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_{RW} )</td>
<td>0.30</td>
</tr>
<tr>
<td>( r_{FW} )</td>
<td>0.35</td>
</tr>
<tr>
<td>( x_B )</td>
<td>0.30</td>
</tr>
<tr>
<td>( z_B )</td>
<td>0.60</td>
</tr>
<tr>
<td>( x_S )</td>
<td>0.906 791 559 062</td>
</tr>
<tr>
<td>( z_S )</td>
<td>0.294 634 437 917</td>
</tr>
<tr>
<td>( x_C )</td>
<td>0.026 100 592 803</td>
</tr>
<tr>
<td>( z_C )</td>
<td>-0.102 307 311 581</td>
</tr>
<tr>
<td>( x_D )</td>
<td>0.032 071 426 728</td>
</tr>
<tr>
<td>( z_D )</td>
<td>0.267 644 508 448</td>
</tr>
</tbody>
</table>
The front bicycle frame coordinate system is offset from the rear frame by the steering tilt angle. This is expressed by the rotation matrix

\[
^{b}R^{c} = \begin{bmatrix}
C_{\lambda} & 0 & -S_{\lambda} \\
0 & 1 & 0 \\
S_{\lambda} & 0 & C_{\lambda}
\end{bmatrix}
\] (92)

The inertia matrix of the front bicycle frame is then transformed into the coordinates of the current model by the expression

\[
I^{c}_{f} = \begin{bmatrix}
C_{\lambda} & 0 & -S_{\lambda} \\
0 & 1 & 0 \\
S_{\lambda} & 0 & C_{\lambda}
\end{bmatrix} \begin{bmatrix}
I_{C_{x}}^{c} & 0 & -I_{C_{z}}^{c} \\
0 & I_{C_{y}}^{c} & 0 \\
-I_{C_{z}}^{c} & 0 & I_{C_{y}}^{c}
\end{bmatrix} \begin{bmatrix}
C_{\lambda} & 0 & S_{\lambda} \\
0 & 1 & 0 \\
-S_{\lambda} & 0 & C_{\lambda}
\end{bmatrix}
\] (93)

Where \( I^{c}_{f} \) is the inertia matrix of the benchmark bicycle

**EOM Function File**

The equations of motion are developed and evaluated in ‘BIKE_EOM_FUN_v4.m’. First, the file calls in all geometric parameters and initial conditions. It uses these to numerically evaluate the \( C_{1} \) and \( C_{2} \) matrices (Equation 48) from the ‘C_Matrices.m’ function file. Then, it calculates the \( \dot{C}_{1} \) and \( \dot{C}_{2} \) matrices from ‘Derivs_C_Matrices.m’ function file. The file uses these four matrices to calculate the dependent generalized speeds, \( (u_{3}, u_{5}, u_{8}) \) and their derivatives \( (\dot{u}_{3}, \dot{u}_{5}, \dot{u}_{8}) \).

Once all the generalized speeds are determined, the file proceeds with generating the dynamical equations using Kane’s method. First, the partial velocities for each body are calculated using the ‘PartialVelocities.m’ function file. Next, the inertia forces are built up using the ‘CalcAccelerations.m’ and ‘CalcAlphas.m’ function files. Then, the active forces are built up using the ‘CalcActiveForces.m’ function file.

The file combines all these components to finalize a set of three dynamical equations. These equations are rearranged to match Equation 82. Then, inverting the mass matrix yields the three numerical values that represent the output of the first-order dynamical equations. The
numerical values of the six generalized speeds (having been already calculated) represent the output of the first-order kinematical equations (Equation 83). These nine values comprise the output vector of the EOM function file.

3.5.3 Output and Validation

To validate the correctness of the current model, the simulation routine was run according the test case present in Meijaard. This case consists of a forward-moving bicycle that experiences a lateral perturbation. The initial forward speed is set to 4.6 m/s and the initial roll rate is set to 0.5 rad/s [13]. The initial values for yaw angle, roll angle, steering angle, and the rotation angles for both wheels are all set to zero. Recall, the pitch angle is a dependent coordinate so its initial value is solved for using the roll angle and steering angle initial conditions. The two possible conditions are 0° or 174°, where the 0° option is correct.

After set up, the simulation evaluates the response of the bicycle over a five-second time period. The time step used for this test case was 0.01 seconds. The output of Meijaard’s test case is a plot of the forward velocity, roll rate and steering rate of the bicycle. This result is replicated and shown in Figure 22.

Figure 22: Validation plot of bicycle dynamic response
One of the interesting phenomena of the bicycle is its self-stability. This is well-illustrated from models that are linearized about straight forward motion. Time constraints did not permit linearization of the current model, but the literature reveals several characteristics of this type of linear model that are helpful in interpreting Figure 22.

The linearized equations of motion for a bicycle (following the Whipple model) have four eigenvalues that correspond with four eigenmodes [13]. These modes are often calculated over a range of forward speeds to show stability characteristics. Except at very low speeds, these four modes are comprised of two complex and two real modes. The complex modes are classified as the weave mode, and the two real modes are classified as the caster mode and the capsize mode. Positive values for the real components of the eigenvalues are associated with unstable motion and negative real components are associated with stable motion.

Analysis of these modes reveals that the benchmark bicycle exhibits self-stability for the speed range of about 4.3 m/s to 6 m/s. Below this speed range the weave mode is unstable, and above it the capsize mode is unstable. The caster mode is strongly stable in both cases. The test case (at 4.6 m/s) is just inside of the stable range. Figure 22 shows the asymptotic stability of the bicycle, and its energy conservation. The initial roll perturbation is counteracted by a steering response, and these rates attenuate over time. The system has no true damping so the energy imparted to the bicycle through the perturbation is transferred to the forward velocity of the bicycle, which is evidenced by the slightly higher final velocity.

### 3.6 Results

The computational model develops the dynamics of the bicycle from first principles. It exactly recreated the results of a common test case to prove its correctness. However, beyond this single test case the model can be exercised to demonstrate other useful responses. Similar to Figure 22, the characteristics of the linear model are helpful in interpreting the two additional results present here.

First, the model is tested below the lower bounds of the stable speed range. In the region, the roots of the linearized model show the capsize mode is stable, and the weave mode is slow and unstable. A case is evaluated using a forward velocity of 3.5 m/s and the output are shown in Figures 23 and 24.
The erratic steer and roll rates illustrate the oscillatory, unstable weave mode. The bicycle initially tries to correct the perturbation, but does not have enough energy to return to a stable equilibrium. The bicycle would fall over after the 1.5 second mark because of its large roll angle, but the contact point constraints prevent the tires from leaving the ground plane. This results in a long and slow turning motion that allows the bicycle to temporarily regain a vertical orientation before descending to another unstable roll angle. Figure 25 illustrates the turning motion of the bicycle.

Figure 23: Roll rate, steer rate and forward velocity for lower unstable region (3.5 m/s)
Figure 24: Roll angle of bicycle in lower unstable region (3.5 m/s)

Figure 25: Yaw angle of the bicycle in the lower unstable region (3.5 m/s)
The model is then tested above the upper bound of the stable speed range. In this region the linear model reveals that the weave mode becomes faster and stable and the capsize mode becomes slightly unstable. A simulation case is evaluated with a forward speed of 8.0 m/s, and the results are shown in Figures 26 and 27.

Figure 26 illustrates the fast and stable nature of the weave mode, as the roll and steer rates quickly damp out. The slow and unstable capsize mode, which is dominated by the rolling motion, drives the bicycle to slowly fall over (Figure 27).

These results display that the computational model can accurately describe a few of the complex motions of the bicycle. However, these cases demonstrate only one aspect of the model. With a small amount of additional work, the model can be extended beyond initial value problems and be used to study more advanced scenarios.

![Figure 26: Roll rate, steer rate and forward velocity for upper unstable region (8.0 m/s)](image-url)
This chapter has detailed the generation of the equations of motion for the bicycle and demonstrated how to build a computational model with these equations. The computational model was tested in various ways and the output give evidence of its correctness. This model has powerful analytical capabilities, and additional work could be done to extend these capabilities further. The primary purpose of the model is studying the motion of a bicycle due to rider interaction. Therefore, the recommendations for future work are specifically directed at augmenting this purpose.

First, in its current state the model is a very simplified version of the human/bicycle system. It treats the rider as a fixed part of the rear bicycle frame, so the first step could be extending the model to include a movable rider [18]. There are several ways to do this but the simplest is to assume the rider is a separate body that behaves as an inverted pendulum. This would match well with the IPB, as the rider position sensor measures the relative position of the upper body.

Second, a small modification to the program would allow the model to be evaluated from an inverse dynamics approach. The IPB measures the states of the bicycle, so these would be treated as model input and a time-history of the three torques acting on the bicycle would be the
output. These torques could be used to analyze a rider’s influences on the bicycle. They could also be reintroduced to the model as input to see if the model output similar results to the measured values. This process could be used in the future to evaluate the accuracy of the sensors and identify problem areas.

Finally, the program could be modified to numerically linearize the equations of motion. Linearized equations open the door for another level of analysis. Additionally, the TRCLC has built a motion simulator for studying a rider’s interaction with transportation-based scenarios. This linear model could be used to enhance real-time motion feedback and improve the quality of data gathered by the simulator [19].
REFERENCES


A Shimano drivetrain ensures the quickest, most reliable shifts every time, all the time. The 24-speed Rapidfire shifting drivetrain with Shimano M171 48/38/28 crankset, Acera rear & M191 front derailleurs and 11-32 cassette offers a much larger gear range, with much lower gearing, for easier hill climbing than standard or compact double chainring road and street bikes.

Strong-but-light double wall rims from Alex, with Formula hubs and 32 stainless steel spokes per wheel laced in a 3-cross pattern assure a reliably true & durable wheel.

Vittoria’s Randonneur 32c tires offer more air volume & a larger footprint to make for a more comfortable & stable ride. They also offer increased puncture resistance thanks to a Double Shielding casing and reflective sidewalls for enhanced visibility at night.

Tektro’s V-brakes offer plenty of stopping power (and plenty of room for the addition of fenders) with a modulator in front to ensure controlled braking. The sealed cartridge bottom bracket & sealed headset offered on Coda Sport improve performance and reduce maintenance.

**STREET - Urban**

**UPGRADES/CHANGES FROM 2015**

Same great spec and quality

**TOP 5 THINGS YOU NEED TO KNOW**

A Shimano drivetrain ensures the quickest, most reliable shifts every time, all the time. The 24-speed Rapidfire shifting drivetrain with Shimano M171 48/38/28 crankset, Acera rear & M191 front derailleurs and 11-32 cassette offers a much larger gear range, with much lower gearing, for easier hill climbing than standard or compact double chainring road and street bikes.

Strong-but-light double wall rims from Alex, with Formula hubs and 32 stainless steel spokes per wheel laced in a 3-cross pattern assure a reliably true & durable wheel.

Vittoria’s Randonneur 32c tires offer more air volume & a larger footprint to make for a more comfortable & stable ride. They also offer increased puncture resistance thanks to a Double Shielding casing and reflective sidewalls for enhanced visibility at night.

Tektro’s V-brakes offer plenty of stopping power (and plenty of room for the addition of fenders) with a modulator in front to ensure controlled braking. The sealed cartridge bottom bracket & sealed headset offered on Coda Sport improve performance and reduce maintenance.

---

**SPECIFICATIONS**

- **Frame**: Double-butted chromoly main tubes, extended head tube with reinforced collars, double tapered cromo stays, forged dropouts and fender/rack mount eyelets
- **Fork**: Chromoly uniswung, banded butted slaeve, low rider carrier mounts and forged dropouts with fender mount eyelets
- **Headset**: AheadSet, 1 1/8"
- **Wheels**: Alex ID-19 double wall alloy rims with GSW sidewall, 32H, Formula alloy hubs with QR and 14g stainless steel spokes
- **Tires**: Vittoria Randonneur with Double Shielding puncture protection and reflective sidewalls, 700 x 32c
- **Derailleurs**: Shimano Acera M360 SGS rear and M191 bottom pull front
- **Shifters**: Shimano Rapidfire Plus ST-M310, 24-Speed
- **Chain**: KMC Z7 8-Speed
- **Cassette**: SRAM PG 820 8-Speed
- **Crankset**: Shimano FC-M171 triple, 48/38/28T, 170mm (15-19”), 175mm (21-23”)
- **BB Set**: FSA sealed cartridge, 68 x 110.5mm
- **Pedals**: Platform style, steel cage/resin body, toe clip attachable
- **Brakeset**: Tektro alloy linear-pull brakes with front power modulator and Shimano brake levers
- **Handlebar**: Jamis Urban bar, 6061 T-6 aluminum, 25.4 x 10˚ sweep x 580mm
- **Stem**: Jamis Road, 3D forged 6061 alloy, 25.4 x 17˚ x 90mm (15/17”), 110mm (19/21/23”)
- **Grips**: New Anatomic double density kraton grip with end plug
- **Seat Post**: Jamis alloy micro-adjust, 27.2 x 500mm alloy clamp with QR swatpin
- **Saddle**: Jamis Touring with SL cover and satin steel rails
- **SIZES**: 15”, 17”, 19”, 21”, 23”
- **Color**: Gloss Black or Sahara Silver
1.5. System Performance

All specifications are rated to 1σ, over -40°C to +85°C (-40 to 185°F) unless otherwise stated.

These specifications have been measured based on typical mission scenarios with simulated GPS outages and compared to post processed RTK data of a high end FOG based INS.

1.5.1. Ellipse-A orientation performance

1.5.1.1. Orientation specifications

<table>
<thead>
<tr>
<th>Performance</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement range</td>
<td>360° in all axes, no mounting limitation</td>
</tr>
<tr>
<td>Roll / Pitch accuracy</td>
<td>&lt; 0.2°</td>
</tr>
<tr>
<td>Yaw Accuracy</td>
<td>0.8°</td>
</tr>
<tr>
<td></td>
<td>Medium dynamic conditions – No long term accelerations</td>
</tr>
<tr>
<td></td>
<td>Clean magnetic environment – Magnetic calibration performed</td>
</tr>
</tbody>
</table>

1.5.2. Ellipse-E/ N orientation and navigation performance

For each application, the accuracy parameters are defined in different positioning modes, explained below:

- SP refers to Single Point mode and is the default L1 GPS / GLONASS fix quality
- RTK stands for Real Time Kinematics with a typical 1 cm accuracy position
- Odometer Aiding is specified when an odometer provides velocity (automotive applications)
- DVL aided is specified when a Doppler velocity Log sensor provides sub-water velocity information (bottom tracking)

1.5.2.1. Land applications

<table>
<thead>
<tr>
<th>Outage Duration</th>
<th>Positioning Mode</th>
<th>Position Accuracy Horizontal</th>
<th>Position Accuracy Vertical</th>
<th>Velocity Accuracy Horizontal</th>
<th>Velocity Accuracy Vertical</th>
<th>Attitude Accuracy (°)</th>
<th>Roll / Pitch</th>
<th>Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Outage</td>
<td>SP</td>
<td>2 m</td>
<td>2.5 m</td>
<td>0.1 m/s</td>
<td>0.1 m/s</td>
<td>0.2°</td>
<td>0.4°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RTK</td>
<td>0.02 m</td>
<td>0.04 m</td>
<td>0.05 m/s</td>
<td>0.05 m/s</td>
<td>0.2°</td>
<td>0.4°</td>
<td></td>
</tr>
<tr>
<td>10 s</td>
<td>SP</td>
<td>2.5 m</td>
<td>3 m</td>
<td>0.2 m/s</td>
<td>0.2 m/s</td>
<td>0.2°</td>
<td>0.5°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RTK</td>
<td>0.8 m</td>
<td>0.8 m</td>
<td>0.1 m/s</td>
<td>0.1 m/s</td>
<td>0.2°</td>
<td>0.5°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Odometer aiding</td>
<td>0.1% of DT</td>
<td>0.1% of DT</td>
<td>0.1%</td>
<td>0.1%</td>
<td>0.4°</td>
<td>0.6°</td>
<td></td>
</tr>
<tr>
<td>60 s</td>
<td>SP</td>
<td>10 m</td>
<td>8 m</td>
<td>0.5 m/s</td>
<td>0.5 m/s</td>
<td>0.25°</td>
<td>0.6°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RTK</td>
<td>8 m</td>
<td>6 m</td>
<td>0.4 m/s</td>
<td>0.4 m/s</td>
<td>0.25°</td>
<td>0.6°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Odometer aiding</td>
<td>0.2% of DT</td>
<td>0.2% of DT</td>
<td>0.1%</td>
<td>0.1%</td>
<td>0.2°</td>
<td>0.6°</td>
<td></td>
</tr>
</tbody>
</table>

1.5.2.2. Marine & Subsea applications

<table>
<thead>
<tr>
<th>Outage Duration</th>
<th>Positioning Mode</th>
<th>Position Accuracy Horizontal</th>
<th>Position Accuracy Vertical</th>
<th>Velocity Accuracy Horizontal</th>
<th>Velocity Accuracy Vertical</th>
<th>Attitude Accuracy (°)</th>
<th>Roll / Pitch</th>
<th>Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-outage</td>
<td>SP</td>
<td>2 m</td>
<td>2.5 m</td>
<td>0.1 m/s</td>
<td>0.1 m/s</td>
<td>0.2°</td>
<td>0.8°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RTK / Dual</td>
<td>0.02 m</td>
<td>0.04 m</td>
<td>0.05 m/s</td>
<td>0.05 m/s</td>
<td>0.2°</td>
<td>0.2°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>antenna GPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 s</td>
<td>SP</td>
<td>3 m</td>
<td>3.5 m</td>
<td>0.2 m/s</td>
<td>0.2 m/s</td>
<td>0.3°</td>
<td>0.8°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RTK / Dual</td>
<td>1 m</td>
<td>1 m</td>
<td>0.15 m/s</td>
<td>0.15 m/s</td>
<td>0.3°</td>
<td>0.3°</td>
<td></td>
</tr>
<tr>
<td></td>
<td>antenna GPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DVL aided</td>
<td>0.5% of DT</td>
<td>0.5% of DT</td>
<td>0.5%</td>
<td>0.5%</td>
<td>0.2°</td>
<td>0.8°</td>
<td></td>
</tr>
</tbody>
</table>
## Hall-Effect Rotary Position Sensors, HRS Series

### Table 1. Electrical Specifications

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>HRS100SSAB090</th>
<th>HRS100SSAB180</th>
<th>HRS100SWAB090</th>
<th>HRSRES05A090</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solder Lugs, Unformed</td>
<td>Flying Wire Leads</td>
<td>Solder Lugs, Formed 90° Down</td>
<td></td>
</tr>
<tr>
<td><strong>Electrical angle</strong></td>
<td>90° ±2° CW</td>
<td>180° ±2° CW</td>
<td>90° ±2° CW</td>
<td></td>
</tr>
<tr>
<td><strong>Output voltage</strong></td>
<td>CW: 4.750 V min., 4.850 V max., 0.250 V max.</td>
<td>CW: 4.750 V min., 4.850 V max., 0.250 V max.</td>
<td>CW: 4.750 V min., 4.850 V max., 0.250 V max.</td>
<td>CW: 2.095 V min., 2.205 V max., 0.294 V max.</td>
</tr>
<tr>
<td><strong>Linearity</strong> (% of span)</td>
<td>±2% typ.</td>
<td>±4% typ.</td>
<td>±2% typ.</td>
<td></td>
</tr>
<tr>
<td><strong>Output current</strong></td>
<td>2 mA max.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Output type</strong></td>
<td>sink/source</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overvoltage protection</strong></td>
<td>18 Vdc. max.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Supply voltage</strong></td>
<td>5 Vdc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Supply current</strong></td>
<td>5 mA typ.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2. Mechanical Specifications

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>HRS100SSAB090</th>
<th>HRS100SSAB180</th>
<th>HRS100SWAB090</th>
<th>HRSRES05A090</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mechanical angle of rotation</strong></td>
<td>90° ±2°</td>
<td>180° ±2°</td>
<td>90° ±2°</td>
<td></td>
</tr>
<tr>
<td><strong>Rotational cycles</strong></td>
<td>10 million typ.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Rotational torque</strong></td>
<td>0.014 N m [2.0 in-oz] max. at 25 °C [77 °F]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stop torque</strong></td>
<td>0.56 N m [5 in-lb] min.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Push out</strong></td>
<td>89 N m [20 in lb] min.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pull out</strong></td>
<td>44 N m [10 in lb] min.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Shaft diameter</strong></td>
<td>6.35 mm [0.25 in]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Shaft material</strong></td>
<td>stainless steel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bush diameter</strong></td>
<td>9.5 mm [0.375 in]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bush material</strong></td>
<td>nickel-plated brass</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Housing material</strong></td>
<td>stainless steel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Terminal material: solder lug terminals</strong></td>
<td>tin-plated brass, SAC305 solder dip</td>
<td></td>
<td></td>
<td>tin-plated brass, SAC305 solder dip</td>
</tr>
<tr>
<td><strong>Flying wire leads</strong></td>
<td>—</td>
<td></td>
<td></td>
<td>XL insulated, 20 AWG stranded</td>
</tr>
<tr>
<td><strong>Terminal support material</strong></td>
<td>thermoplastic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mounting hardware material: mounting nut</strong></td>
<td>nickel-plated brass</td>
<td></td>
<td></td>
<td>nickel-plated brass</td>
</tr>
<tr>
<td><strong>lock washer</strong></td>
<td>nickel-plated brass</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

sensing.honeywell.com
Features
- Two channel quadrature output
- Bushing or servo mount
- Square wave signal
- Small size
- Resolution to 256 PPR
- CMOS and TTL compatible
- Long life
- Ball bearing option for high operating speed up to 3000 rpm
- RoHS compliant

**Technical Specifications**

**Electrical Characteristics**
- Output: 2-bit quadrature code, Channel A leads Channel B by 90° (electrical) with clockwise rotation
- Resolution: 25 to 206 cycles per revolution
- Insulation Resistance (500 VDC): 1,000 megohms
- Electrical Travel: Continuous
- Supply Voltage: 5.0 VDC ±0.25 VDC
- Output Voltage: 0.8 V maximum
- High Output: 4 V minimum
- Output Current: 25 mA maximum
- Rise/Fall Time: 200 ns (typical)
- Power Consumption: 136 mW maximum
- Pulse Width (Electrical Degrees, Each Channel): 180° ±45° typ.
- IP Rating: IP 40

**Environmental Characteristics**
- Operating Temperature Range: -40°C to +75°C (-40°F to +167°F)
- Storage Temperature Range: -40°C to +85°C (-40°F to +185°F)
- Vibration: 5 G
- Shock: 50 G
- Rotational Life: A & C Bushings (300 rpm maximum)**
  - 10,000,000 revolutions
- Y, S & T Bushings (3,000 rpm maximum)**
  - 250,000,000 revolutions

**Mechanical Characteristics**
- Mechanical Angle: 360° Continuous
- Torque (Starting and Running): 0.07 N-cm (0.1 oz-in.) maximum
- Y, S & T Bushings (Ball Bearing Shaft Support): 0.30 mm (0.012") T.I.R. maximum
- Shaft End Play: 0.12 mm (0.045") T.I.R. maximum
- Weight: 11 grs. (0.4 oz.)
- Terminals: Axial or radial pin posts or ribbon cable
- Wave Soldering:
  - 1,000°C (1,832°F) max. for 3 seconds
  - 260°C (500°F) max. for 5 seconds

**Quadrature Output Table**

<table>
<thead>
<tr>
<th>Channel A</th>
<th>Channel B</th>
</tr>
</thead>
<tbody>
<tr>
<td>25°</td>
<td>125°</td>
</tr>
<tr>
<td>50°</td>
<td>128°</td>
</tr>
<tr>
<td>64°</td>
<td>200°</td>
</tr>
<tr>
<td>100°</td>
<td>256°</td>
</tr>
</tbody>
</table>

+“For resolutions ≤ 128 quadrature cycles per shaft revolution.

**Other Information**
- For Non-Standard Resolutions—Consult Factory
- Channel B leads Channel A

---


Specifications are subject to change without notice.

The device characteristics and parameters in this data sheet can and do vary in different applications and actual device performance may vary over time. Users should verify actual device performance in their specific applications.
APPENDIX B: LabVIEW Block Diagram of Data Acquisition Program
APPENDIX C: MATLAB Program for Geographic Sorting Algorithm

--------------------- Route Segmenting Algorithm ---------------------

% IPB Project - Western Michigan University - Summer 2016
% Code written by Brent Kostich
% Last Revision: 5-11-17

% -------------------------- File Description --------------------------
% This script is used to read the collected IPB data, from any number of
% trials, and divide it up into useful segments. The user selects segment
% points using a GUI that loads automatically upon running this script.
% The program will perform the remainder of its operations and output a
% matrix of index values that correspond with the segment points.
% The columns of this matrix represent the number of segment points, and
% the rows represent the number of trials.

% The outputs from this script enter the base MATLAB workspace.
% There is also the option to save them to a .mat file.

% --------------- How to Operate this Script ---------------
% The first round of TRCLC testing was processed and saved into a
% structured array, named 'Data_Structures_3_3.mat'. This file needs to be
% in the same folder as this script in order to execute the load command
% in Line 68 (of current revision).

% Manually loading the arraying into the workspace also works, but if that
% route is taken, the user will have to omit the load command. Any other
% set of data can be processed by this script, so long as it follows the
% variable naming convention of Data_Structures_3_3.mat. Otherwise, the
% program will have to be modified by the user. This is not recommended.

% Once the variables are loaded, simply run the script. A GUI will pop
% up, wherein the user can add as many segment points as desired. The GUI
% is equipped with simple instructions for use. If an error occurs
% simply close the GUI and re-run the program.

% The naming format for the structured arrays in 'Data_Structures_3_3' is:

'Variable.All.Trial(#).Data'
- or for variables with components -

'Variable.Component.All.Trial(#).Data'

Example: --------------------------------------------------------------- %

If you desire to see the forward velocity of the bicycle, for the seventh trial, the syntax is:

>> BodyVel.X.All.T(7).Data

'BBodyVel' calls up the velocity data, 'X' is the forward direction, 'All' indicates this data contains all trials, and 'T(7)' means the seventh trial.
% -------------------------------------------------------- %

% --------------------------------------------------------------- %
%
clear % removes all workspace variables
load Data_Structures_3_3.mat % loads in the necessary data variables

----------------- Initializing Algorithm ------------------------

{%
Section Description:
This section initializes the algorithm by finding the cumulative distance traveled, for all trials, then determining which is the shortest. This shortest trial is the one loaded into the GUI for point selection.

The shortest trial is used because it has the minimum number of indices. If an intermediate trial was used, the user may select index points that are outside the range of shorter trials.
%
- Preallocating variables
num_trials = length(Yaw.All.T); % Number of trials
end_points = zeros(1,num_trials); % Total distance
dist_ints = zeros(length(tspan.All.T(1).Data),num_trials); % Distance
% integrals
% -- Approximate integration routine for first trial
for i = 1:num_trials
% - case for missing IMU data, substitution is made with encoder data
check = sum(isnan(BodyVel.X.All.T(i).Data));
if check > 0
% filter for encoder-based velocity data, to improve accuracy
h = [1/2 1/2];
binomialCoeff = conv(h,h);

for n = 1:4
    binomialCoeff = conv(binomialCoeff,h);

% a
%)
%

----------------- Initializing Algorithm ------------------------
80

end

p = WheelVel.All.T(i).Data;
fDelay = (length(binomialCoeff)-1)/2;
fwd_vel = filter(binomialCoeff, 1, p); % forward velocity from
% filtered encoder data
else
  fwdrv = BodyVel.X.All.T(i).Data; % forward velocity from
% IMU data
end

Q = cumtrapz(tspan.All.T(i).Data,fwd_vel); % approx. integral

% - saving the total distances for each trial
for j = 1:length(Q)
  dist_ints(j,i) = Q(j);
end

end_points(i) = Q(end); % total distance traveled for each trial
end

% - identifies shortest-distance trial
shortest = find(end_points == min(end_points));

%================================= Algorithm Input ==================================
%
% In order to run this program, the algorithm needs one input, which
% is a vector of segment points. These segments points provide a reference
to find distance points and establish the fixed GPS points.
%
% This GUI makes the selection of these points fast and easy.
%
seg_points = ones(2,2);
run point_select_gui;
short_trial = dist_ints(:,shortest);
ref_dist = short_trial(ref_indices);

%================================= Isolating segments by distance traveled ==========
%
% Original reference distances used for first iteration of program
% ref_dist = [20,75,110,400,610,650,750,785,930,1320,1345];

% Section Description:
% This section isolates data segments based on the distance traveled by
% the bicycle. The distance translates to a point of latitude and a point
% of longitude. Together these make a GPS point.
%
% - Preallocating
ref_points = zeros(num_trials,2,length(ref_dist)); % reference points
index_1 = zeros(num_trials,length(ref_dist)); % initial index points

% - Attaching GPS points to the reference indices
for k = 1:length(ref_dist)
    for i = 1:num_trials
        I = find(dist_ints(:,i) >= ref_dist(k),1);

        ref_points(i,1,k) = Latitude.All.T(i).Data(I);
        ref_points(i,2,k) = Longitude.All.T(i).Data(I);

        index_1(i,k) = I;
    end
end

%---------------- Finding the nearest location of segment points ----------------

%
Section Description:
This section uses the selected segment points, from the shortest trial, to and generates fixed GPS points. It then takes the guess indices from all other trials and finds the nearest GPS to the fixed ones. The goal being is to get the reference points from all trials to be as close as possible. This allows for better data comparison.
%

clear i j k

% - Initializing loop and preallocating
epsilon = 1.0e-6; % stoppage criteria
step_vec = zeros(1e2,1); % vector of closeness for each iteration
index_2 = zeros(num_trials,length(ref_dist)); % final index points
index_2(shortest,:) = index_1(shortest,:); % new index matrix, retaining shortest row

% - Index locating loop
for j = 1:length(ref_dist)
    fixed_point = index_1(shortest,j); % selecting the 'fixed point' for % each section point

    lat_fp = Latitude.All.T(shortest).Data(fixed_point); % fixed latitude
    lon_fp = Longitude.All.T(shortest).Data(fixed_point); % fixed longitude

    rem_trials = 1:num_trials; % remaining trials
    rem_trials(shortest) = []; % removes the fixed trial from the loop

    for k = 1:length(rem_trials)
        step = 1; % initializing variable
        i = 1; % initializing iteration count
        trial_num = rem_trials(k);

        while step > epsilon % index finding loop
% - Search Step Condition - %

if i == 1
    % starting with the initial index found
    % from previous approx. integral distance
    index_tp = index_l(trial_num,j));
    direction = 1;
    % move index in a guess direction if the
    % stoppage criteria is not met
endif direction == 1
    index_tp = index_tp + 1;
    % search direction is taken over by a
    % control loop, found later in code
endif direction == -1
    index_tp = index_tp - 1;
end

lat_tp = Latitude>All.T(trial_num).Data(index_tp); % test point lat
lon_tp = Longitude>All.T(trial_num).Data(index_tp); % test point lon
latdif = lat_fp - lat_tp; % taking the difference between the
    % fixed point and test point
londif = lon_fp - lon_tp;
step = sqrt((latdif)^2 + (londif)^2); % computing the
    % 'closeness' criteria

% Note: It is assumed that for the small search area, the above
% method is sufficiently accurate
step_vec(i) = step; % saving the step output

% - Pass Condition - %

% This condition is to stop the search routine if the optimal
% point is passed and the stoppage criteria is not met

if i >= 4
    if step_vec(i) == step_vec(i-2)
        break
    end
endif

% - Stoppage conditions for loop - %

if step_vec(i) <= epsilon || i == 1
    else
% - Search-direction control loop - %

if step_vec(i) < step_vec(i-1)
    direction = 1*direction;
else

82
direction = -1*direction;
end
end

i = i + 1; % increasing iteration count
end
index_2(trial_num,j) = index_tp; % saving new index point
end
end

ref_points_2 = zeros(num_trials,2,length(ref_dists)); % new GPS ref points

for k = 1:length(ref_dists) % saving new index points as lat and lon points
for i = 1:num_trials
   I = index_2(i,k);
   ref_points_2(i,1,k) = Latitude.All.T(i).Data(I);
   ref_points_2(i,2,k) = Longitude.All.T(i).Data(I);
end
end

------------------ Displaying the algorithm results ------------------

{%
Section Description:
This section plots the results of the algorithm. The figures plot all trials, and the output index points.

The first plot shows the results after the distance points are determined. This plot illustrates why just using the distance points is an insufficient method.

The second plot shows the final results, after the index locating loop has done its job. The plot show that the results are reasonably accurate.

These plots verify the effectiveness of the algorithm.
%}

close all

figure (1)

% ------ "Before" Plot ------ %

% - plotting the first trial route as a plot foundation
plot(Longitude.All.T(1).Data,...
     Latitude.All.T(1).Data,'LineWidth',1)
title('Segment Points Strictly by Distance')
grid on,hold on

% - plotting all other trials on top of existing plot
for i = 2:num_trials
   plot(Longitude.All.T(i).Data,Latitute.All.T(i).Data)
end
disp('NOTE: Check filename before saving to ensure no data is overwritten')
disp('-----------------------------------------')

filename = 'Segmented_Route_1_0';
disp(['File is named: ',filename])
disp('')

prompt = 'Do you want to save output to this file? [Y/N]: '
save_option = input(prompt,'s'))

if save_option == 'Y'
    save(filename,'ref_dists','index_2','ref_points_2')
    disp([filename,' has been saved.'])
else
    disp('File has NOT been saved.')</end
APPENDIX D: Derivatives of the C Matrices

The terms presented in this appendix follow this form

\[
\begin{bmatrix}
\dot{C}_{11} & \dot{C}_{12} & 0 \\
\dot{C}_{11} & \dot{C}_{12} & \dot{C}_{13} \\
\dot{C}_{11} & \dot{C}_{12} & 0
\end{bmatrix}
\]

Beginning with \( \dot{C}_1 \), the elements of the matrix are

\[
\begin{align*}
\dot{C}_{11} = & \left( z_D \left( u_5 (C_{\lambda} S_{\lambda} S_5 + C_5 S_{\lambda} S_7) - u_3 (C_{\lambda} C_5 C_7 + C_7 S_{\lambda} S_5) \right) + \\
& S_5 u_5 (x_5 C_{\lambda} - z_5 S_5) + u_3 C_7 (r_{FW} + z_5 C_5 + x_5 S_5) + \\
& r_{FW} \left( (C_{\lambda} C_4 C_5 - C_4 S_{\lambda} S_5)(u_5 C_{\lambda} S_5 S_7 - u_3 C_5 C_7 + u_5 C_5 S_{\lambda} S_7 + u_7 C_7 S_{\lambda} S_5) - \\
& (S_{\lambda} C_5 S_7 - C_\lambda C_5 S_7)(u_4 C_{\lambda} C_5 S_5 + u_3 C_5 C_7 + u_5 C_5 S_{\lambda} S_7 - u_7 S_{\lambda} S_5) \right) \right) / #1 + \\
& r_{FW} \left( (C_{\lambda} C_4 C_5 - C_4 S_{\lambda} S_5)(u_4 C_{\lambda} C_5 S_5 + u_3 C_5 C_7 + u_5 C_5 S_{\lambda} S_7 - u_7 S_{\lambda} S_5) + \\
& (C_{\lambda} C_4 C_5 - C_4 S_{\lambda} S_5)(u_4 C_{\lambda} C_5 S_5 + u_3 C_5 C_7 + u_5 C_5 S_{\lambda} S_7 - u_7 S_{\lambda} S_5) \right) / #1^3 + \\
& \left. u_4 S_{\lambda} S_4 S_5 (C_{\lambda} C_4 C_5 - C_4 S_{\lambda} S_5)(S_{\lambda} S_5 S_7 - C_\lambda C_5 S_7) \right)
\end{align*}
\]

\[
#1 = ((C_{\lambda} C_4 C_5 - S_{\lambda} S_5 + C_4 C_5 S_{\lambda})^2 + C^2_{\lambda} S^2_5)^{1/2}
\]

\[
\dot{C}_{12} = r_{FW} \left( u_5 C_{\lambda} C_7 S_5 + u_3 C_5 C_7 S_{\lambda} + u_7 C_{\lambda} C_5 S_7 - ru_7 S_{\lambda} S_5 S_7 \right)
\]
\[
\hat{C}_{121} = \left\{ \begin{array}{l}
x_D (u_3 C_2 C_5 - u_3 S_4 S_5) + z_D (u_5 C_4 C_5 S_3 + u_5 C_4 C_5 S_2 + u_5 C_4 C_5 S_7 - u_3 S_4 S_5) + \\
C_7 u_3 (x_3 C_5 - z_3 S_5) - u_3 S_7 (r_{FW} + z_5 C_5 + x_3 S_5) + \\
(r_{FW} (C_3 C_7 - C_3 S_3 S_5) (u_4 C_4 C_5 S_4 + u_5 C_4 C_5 S_5 + u_5 C_4 C_5 S_2 - u_4 S_4 S_5) + \\
(C_3 C_4 C_5 - C_3 S_3 S_5) (u_4 C_4 C_5 S_5 + u_5 C_4 C_5 S_4 + u_5 C_4 C_5 S_2 - u_4 S_4 S_2 S_7) + \\
(C_3 S_5 + C_3 S_2) (u_4 C_4 S_7 + u_5 C_4 S_4 - u_5 C_4 C_5 C_7 + u_4 C_4 C_7 S_5 + u_4 C_4 C_7 S_2) - \\
(u_5 C_2 C_5 - u_5 S_4 S_5) (C_3 C_4 C_5 S_3 - S_2 S_7 + C_4 C_4 C_7 S_2) \\
\end{array} \right\} \\
/ \# 1 -
\]

\[
\hat{C}_{122} = r_{FW} (u_3 (C_4 C_5 C_7 - C_3 S_3 S_5) - u_5 (C_2 S_5 S_2 + C_3 C_7 S_2))
\]

\[
\hat{C}_{123} = \left\{ \begin{array}{l}
r_{FW} (u_4 C_5 S_7 + u_7 C_4 S_4 - u_5 C_4 C_5 C_7 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_4 C_4 C_5 S_4 + \\
u_5 C_4 C_5 S_3 + u_5 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + \\
(r_{FW} (C_4 C_5 S_5 - S_3 S_7 + C_4 C_5 S_3 S_5) (u_4 C_4 S_7 + u_7 C_4 S_4 - u_5 C_4 C_5 C_7 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_4 C_4 C_5 S_4 + \\
u_5 C_4 C_5 S_3 + u_5 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + \\
(u_4 C_4 C_5 C_7 - C_3 S_5 S_2) (C_4 C_4 C_7 S_5 - S_3 S_7 + C_4 C_4 C_7 S_2) \\
\end{array} \right\} \\
/ \# 1 -
\]

\[
\hat{C}_{123} = \left\{ \begin{array}{l}
r_{FW} (u_4 C_4 S_7 + u_7 C_4 S_4 - u_5 C_4 C_5 C_7 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_4 C_4 C_5 S_4 + \\
u_5 C_4 C_5 S_3 + u_5 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + \\
(r_{FW} (C_4 C_5 S_5 - S_3 S_7 + C_4 C_5 S_3 S_5) (u_4 C_4 S_7 + u_7 C_4 S_4 - u_5 C_4 C_5 C_7 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_4 C_4 C_5 S_4 + \\
u_5 C_4 C_5 S_3 + u_5 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + \\
(u_4 C_4 C_5 C_7 - C_3 S_5 S_2) (C_4 C_4 C_7 S_5 - S_3 S_7 + C_4 C_4 C_7 S_2) \\
\end{array} \right\} \\
/ \# 1 -
\]

\[
\hat{C}_{123} = \left\{ \begin{array}{l}
r_{FW} (u_4 C_5 S_7 + u_7 C_4 S_4 - u_5 C_4 C_5 C_7 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_4 C_4 C_5 S_4 + \\
u_5 C_4 C_5 S_3 + u_5 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + \\
(C_4 C_4 C_5 S_5 - S_3 S_7 + C_4 C_5 S_3 S_5) (u_4 C_4 S_7 + u_7 C_4 S_4 - u_5 C_4 C_5 C_7 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_4 C_4 C_5 S_4 + \\
u_5 C_4 C_5 S_3 + u_5 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + u_4 C_4 C_5 S_3 + u_4 C_4 C_5 S_2 + u_5 C_4 C_5 S_4 + \\
(u_4 C_4 C_5 C_7 - C_3 S_5 S_2) (C_4 C_4 C_7 S_5 - S_3 S_7 + C_4 C_4 C_7 S_2) \\
\end{array} \right\} \\
/ \# 1 -
\]
\[
c_{1\alpha} = x_D(u_5(C_4S_5S_7 + C_3S_5S_7) - \\
+ u_7(C_4C_5S_5 - C_5S_5S_7)) + \\
\begin{pmatrix}
(C_3C_5S_5 - S_5S_7 + C_4C_5S_7)(u_5C_5S_5 - u_7C_4C_5S_5 + u_3C_3S_5S_7 - u_5C_3S_5S_7 - u_4C_3S_5S_7 - u_6C_3S_5S_7) - \\
(u_3C_3S_5S_7 + u_5C_3S_5S_7 + u_4C_3S_5S_7)
\end{pmatrix}/\#1 + \\
\begin{pmatrix}
(C_3C_5S_5 - S_5S_7 + C_4C_5S_7)(u_5C_5S_5 - u_7C_4C_5S_5 + u_3C_3S_5S_7 - u_5C_3S_5S_7 - u_4C_3S_5S_7 - u_6C_3S_5S_7) + \\
(u_3C_3S_5S_7 + u_5C_3S_5S_7 + u_4C_3S_5S_7)
\end{pmatrix}/\#1^3
\]

\[
\dot{c}_{1\alpha} = r_{fw}u_5(S_5S_7 - C_4\lambda)
\]

And for \(\dot{C}_2\)

\[
\begin{bmatrix}
\dot{C}_{21} \\
\dot{C}_{22} \\
\dot{C}_{23}
\end{bmatrix} = \begin{bmatrix}
\dot{C}_{21} \\
\dot{C}_{22} \\
\dot{C}_{23}
\end{bmatrix}
\]

\[
\dot{C}_{21} = r_{fw}\begin{pmatrix}
(C_3C_5S_5 - S_5S_7 + C_4C_5S_7)(u_5C_5S_5 - u_7C_4C_5S_5 + u_3C_3S_5S_7 - u_5C_3S_5S_7 - u_4C_3S_5S_7 - u_6C_3S_5S_7) - \\
(u_3C_3S_5S_7 + u_5C_3S_5S_7 + u_4C_3S_5S_7)
\end{pmatrix}/\#1 + \\
\begin{pmatrix}
(C_3C_5S_5 - S_5S_7 + C_4C_5S_7)(u_5C_5S_5 - u_7C_4C_5S_5 + u_3C_3S_5S_7 - u_5C_3S_5S_7 - u_4C_3S_5S_7 - u_6C_3S_5S_7) + \\
(u_3C_3S_5S_7 + u_5C_3S_5S_7 + u_4C_3S_5S_7)
\end{pmatrix}/\#1^3
\]

\[
\#1 = ((C_4C_5S_5 - S_5S_7 + C_4C_5S_7)^2 + C_5^2)^{1/2}
\]
\[
\begin{align*}
\dot{C}_{212} &= \begin{pmatrix}
\frac{u_2 C_2 S_j (z_s + r_{RW} C_7)}{u_2 z_s S_j - u_2 S_j S_s (x_s + r_{RW} S_s)} - \\
\frac{r_{RW} \left( C_7 u_4 C_4 C_5 S_4 + u_5 C_4 C_5 S_4 + u_5 C_4 C_5 S_4 - \right)}{u_4 S_j S_s (x_s + r_{RW} S_s)} - u_5 C_4 C_5 S_4 - \\
\end{pmatrix} \\
&\quad / \#1 + \\
&\quad \left( C_4 C_7 S_5 - S_4 S_j + C_4 C_5 S_5 \right)(u_4 C_4 S_7 + u_4 C_4 S_7 + u_4 C_4 C_5 S_7) + u_4 C_4 C_5 S_7 + \\
&\quad r_{FW} C_7 \left( u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 - \right) + C_4 C_5 - \\
&\quad \left( C_4 S_4 S_4 + u_4 C_4 S_4 + u_4 C_4 S_4 + \right) / \#1 + \\
&\quad r_{FW} u_4 \left( C_4 C_5 S_4 + C_4 C_5 S_4 \right) \\
#1 &= \left( (C_4 C_4 C_5 S_5 - S_4 S_j + C_4 C_5 S_5) \right)^2 + C_4^2 S_5^2 \end{align*}
\]

\[
\dot{C}_{213} = \begin{pmatrix}
\frac{u_3 C_4 S_j (z_s + r_{RW} C_7)}{u_3 z_s S_j - u_3 S_j S_s (x_s + r_{RW} S_s)} - u_3 C_4 S_j + u_3 C_4 S_j - u_3 C_4 C_4 S_4 + \\
\frac{r_{RW} \left( C_4 C_7 S_5 - S_4 S_j + C_4 C_5 S_5 \right)(u_4 C_4 S_7 + u_4 C_4 S_7 + u_4 C_4 C_5 S_7) + u_4 C_4 C_5 S_7 + \\
&\quad r_{FW} C_7 \left( u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 - \right) + C_4 C_5 - \\
&\quad \left( u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + \right) / \#1 - \\
&\quad u_4 C_4 C_5 S_4 - u_4 S_j S_s (x_s + r_{RW} S_s) + \right) \\
&\quad \left( u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 \right) - u_4 C_4 C_5 S_4 + \\
&\quad \left( u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 \right) + \right) / \#1 + \\
&\quad r_{FW} u_4 \left( C_4 C_5 S_4 + C_4 C_5 S_4 \right) \\
#1 &= \left( (C_4 C_4 C_5 S_5 - S_4 S_j + C_4 C_5 S_5) \right)^2 + C_4^2 S_5^2 \end{align*}
\]

\[
\dot{C}_{231} = \begin{pmatrix}
\frac{u_2 C_5 S_j (z_s - r_{RW} (u_2 S_j S_s - u_2 C_5 S_4)) - z_d (u_4 C_4 S_7 + u_7 C_7 S_7 - u_3 C_4 C_5 C_7 + \\
&\quad u_3 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + \right) - z_d (u_4 C_4 S_7 + u_7 C_7 S_7 - u_3 C_4 C_5 C_7 + \\
&\quad u_3 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + \right) / \#1 - \\
&\quad \left( u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 \right) - \right) \\
&\quad \left( u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 \right) + \right) / \#1 + \\
&\quad r_{RW} u_4 \left( C_4 C_5 S_4 - C_4 S_4 S_s \right) \\
#1 &= \left( (C_4 C_4 S_7 - S_4 S_s + C_4 C_5 S_4) \right)^2 + C_4^2 S_5^2 \end{align*}
\]

\[
\nonumber \dot{C}_{232} = \begin{pmatrix}
\frac{r_{RW} u_4 \left( C_2 C_5 S_7 - C_5 S_7 S_j \right) - u_2 C_5 S_7 (x_s + r_{RW} S_s) - u_7 z_D C_7 + \\
&\quad u_7 C_7 C_7 \frac{r_{FW} \left( (C_4 C_4 C_5 S_4 - S_4 S_j + C_4 C_5 S_4) (u_4 C_4 S_7 + u_7 C_7 S_7 - u_5 C_4 C_4 C_5 C_7 + \\
&\quad u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + u_4 C_4 C_5 S_4 + \right) + \right) / \#1 - \\
&\quad u_4 C_4 C_5 S_4 + \right) / \#1 + \\
&\quad \left( u_4 C_4 C_5 S_4 + \right) / \#1 + \\
&\quad r_{FW} u_4 \left( C_4 C_5 S_4 - C_4 S_4 S_s \right) \\
#1 &= \left( (C_4 C_4 C_5 S_5 - S_4 S_j + C_4 C_5 S_5) \right)^2 + C_4^2 S_5^2 \end{align*}
\]
\[
\dot{C}_{2,1} = \begin{pmatrix}
(C_A C_4 C_7 S_5 - S_4 S_7 + C_4 C_5 C_7 S_\lambda) (u_4 C_4 C_7 - u_7 S_4 S_7 + u_5 C_A C_4 C_5 S_7 + u_7 C_A C_4 C_7 S_5 + \\
u_7 C_A C_4 S_5 - u_4 C_4 S_5 S_7 - u_5 C_4 S_5 S_4 S_7 - u_5 C_A S_5 S_7 + \\
C_4 S_5 S_7) (u_4 C_4 S_7 + u_7 C_4 S_7 - u_5 C_A C_4 C_5 S_7 + u_4 C_A C_5 S_4 S_7 + \\
u_5 C_A S_4 S_7 + u_7 C_A C_4 S_5 S_7 + u_5 C_A C_4 S_5 S_7)
\end{pmatrix} / \#1 -
\]
\[
S_A (u_4 C_4 C_7 - r_{rw} u_4 S_4 S_7 + r_{rw} u_4 C_4 S_7 - x_D (u_4 C_4 C_7 - u_7 S_4 S_7 + u_5 C_A C_4 S_5 S_7) + \\
u_5 C_A C_4 S_5 + u_7 C_A C_4 C_5 S_7 - u_4 C_A S_5 S_7 - u_5 C_4 S_5 S_7 - \\
C_A (u_4 x_5 C_4 + r_{rw} u_4 C_4 S_7 + r_{rw} u_5 C_4 S_7) + \\
(C_A C_4 C_7 - S_4 S_7 + C_A C_4 C_7 S_\lambda) (u_4 C_4 S_7 + u_7 C_4 S_7 - u_4 C_A C_4 C_5 S_7 + u_4 C_A C_4 S_4 S_7 + \\
u_5 C_A C_5 S_4 S_7) C_A (C_A C_4 C_5 S_7 - u_4 C_A C_4 S_4 S_7)(C_A C_4 C_5 S_7 - \\
S_4 S_7 + C_A C_4 C_5 S_7) (C_A C_4 C_5 S_7 - C_A C_4 S_5 S_7 + C_A C_4 S_5 S_7) / \#1^3
\]

\#1 = ((C_A C_4 C_7 S_5 - S_4 S_7 + C_A C_5 C_7 S_\lambda)^2 + C_A^2 C_A^2)^{1/2}

\[
\dot{C}_{2,2} = \begin{pmatrix}
u_7 x_D S_7 + r_{rw} u_5 S_2 S_5 - \\
r_{rw} u_5 C_A C_5 - \\
r_{fw} u_5 C_4 C_5
\end{pmatrix} / \#1 +
\]
\[
(C_A C_4 C_7 S_5 - S_4 S_7 + C_A C_5 C_7 S_\lambda) (u_4 C_4 S_7 + u_7 C_4 S_7 - u_5 C_A C_4 C_5 S_7 + \\
u_5 C_A S_5 S_7 + u_4 C_A C_4 S_5 S_7 + u_4 C_A C_4 S_4 S_7 + u_5 C_A C_4 S_4 S_7 + \\
u_7 C_4 S_5 S_7 - u_7 C_A C_4 S_5 S_7 - u_7 C_A C_4 S_5 S_7 + C_A C_4 S_5 S_\lambda)
\end{pmatrix} / \#1^3
\]

\#1 = ((C_A C_4 C_7 S_5 - S_4 S_7 + C_A C_5 C_7 S_\lambda)^2 + C_A^2 C_A^2)^{1/2}

\[
\dot{C}_{2,3} = \begin{pmatrix}
(C_A C_4 C_7 S_5 - S_4 S_7 + C_A C_5 C_7 S_\lambda) (u_4 C_4 S_7 + u_7 C_4 S_7 - u_5 C_A C_4 C_5 S_7 + \\
u_5 C_A S_5 S_7 + u_4 C_A C_4 S_5 S_7 + u_4 C_A C_4 S_4 S_7 + u_5 C_A C_4 S_4 S_7 + \\
u_7 C_4 S_5 S_7 - u_7 C_A C_4 S_5 S_7 - u_7 C_A C_4 S_5 S_7 + C_A C_4 S_5 S_\lambda)
\end{pmatrix} / \#1^3 -
\]
\[
S_A (u_4 C_4 S_7 + u_4 C_4 S_7 - u_5 C_A C_4 C_5 S_7 + \\
u_5 C_A S_5 S_7 + u_4 C_A C_4 S_5 S_7 + u_4 C_A C_4 S_4 S_7 + u_5 C_A C_4 S_4 S_7 + \\
u_5 C_4 S_5 S_7 + u_7 C_A C_4 S_5 S_7 + u_7 C_A C_4 S_5 S_7) / \#1
\]

\#1 = ((C_A C_4 C_7 S_5 - S_4 S_7 + C_A C_5 C_7 S_\lambda)^2 + C_A^2 C_A^2)^{1/2}

89
APPENDIX E: Angular Acceleration Terms

The angular acceleration of body A:

\[ \mathbf{a}^A = \begin{bmatrix} C_5 & 0 & 0 \\ 0 & 1 & 0 \\ S_5 & 0 & 0 & \end{bmatrix} \mathbf{u}_4 + \begin{bmatrix} -C_4S_3 & 0 & 0 \\ S_4 & 1 & 0 \\ C_4C_5 & 0 & 0 & \end{bmatrix} \mathbf{u}_6 + \begin{bmatrix} \alpha A_{k1} \\ \alpha A_{k2} \\ \alpha A_{k3} \end{bmatrix} \]

Where

\[ \alpha A_{k1} = (u_z + u_3S_4)(u_4S_5 + u_3C_4C_5) - (u_4S_5 + u_3C_4C_5)(u_5 + u_6 + u_3S_4) - u_5u_zS_3 + u_3u_4S_4S_3 - u_3u_5C_4C_5 \]
\[ \alpha A_{k2} = u_3u_4C_4 \]
\[ \alpha A_{k3} = (u_4C_5 - u_5C_5S_4)(u_5 + u_6 + u_3S_4) - (u_5 + u_3S_4)(u_4C_5 - u_5C_5S_4) + u_4u_5C_5 - u_3u_4C_4 - u_3u_5C_4S_5 \]

The angular acceleration of body B:

\[ \mathbf{a}^B = \begin{bmatrix} C_5 & 0 & 0 \\ 0 & 0 & 0 \\ S_5 & 0 & 0 & \end{bmatrix} \mathbf{u}_4 + \begin{bmatrix} -C_4S_3 & 0 & 0 \\ S_4 & 1 & 0 \\ C_4C_5 & 0 & 0 & \end{bmatrix} \mathbf{u}_6 + \begin{bmatrix} \alpha B_{k1} \\ \alpha B_{k2} \\ \alpha B_{k3} \end{bmatrix} \]

Where

\[ \alpha B_{k1} = u_3u_4S_4S_5 - u_4u_5S_3 - u_3u_5C_4C_5 \]
\[ \alpha B_{k2} = u_3u_4C_4 \]
\[ \alpha B_{k3} = u_4u_5C_5 - u_3u_4C_5S_4 - u_3u_5C_4S_5 \]

The angular acceleration of body C:

\[ \mathbf{a}^C = \begin{bmatrix} C_4C_4C_7 - C_4S_3S_3 & 0 & 0 \\ S_4S_7 - C_4C_4C_7 & 0 & 0 \\ C_4S_7 + C_4C_4C_7 & 0 & 1 & \end{bmatrix} \mathbf{u}_4 + \begin{bmatrix} S_6S_7 - C_4C_4C_7S_5 - C_4C_4C_7S_5 & S_7 & 0 & \end{bmatrix} \mathbf{u}_6 + \begin{bmatrix} C_7S_4 + C_4C_4C_7S_4 + C_4C_4C_7S_4 & C_7 & 0 & \end{bmatrix} \mathbf{u}_7 + \begin{bmatrix} \alpha C_{k1} \\ \alpha C_{k2} \\ \alpha C_{k3} \end{bmatrix} \]
And finally, the angular acceleration of body D:

\[ \alpha_{D1} = \begin{bmatrix} 0 & 0 & S_7 \lambda & 0 & \dot{u}_4 & S_7 & 0 & \dot{u}_3 & \dot{u}_8 & \alpha_{D1} \end{bmatrix} \]

\[ \alpha_{D2} = \begin{bmatrix} 0 & 0 & S_7 \lambda & 0 & \dot{u}_4 & S_7 & 0 & \dot{u}_3 & \dot{u}_8 & \alpha_{D2} \end{bmatrix} \]

Where

\[ \alpha_{D1} = \left( (u_1 + C_1(u_4S_5 + u_5C_3) + S_2(u_7C_3 - u_3C_2S_5))(C_7 + (u_5 + u_3S_5) - S_2S_7(u_2C_3 - u_3C_4S_5) + 
S_2S_7(u_4S_5 + u_5C_4S_5)) - (u_7 + C_2(u_4S_5 + u_5C_3) + S_2(u_4C_5 - u_5C_4S_5))(u_8 + C_7(u_5 + u_3S_5) - 
C_2S_7(u_4C_5 - u_5C_4S_5) + S_2S_7(u_4S_5 + u_5C_4S_5)) + u_2C_5(u_5 + u_3S_5) - C_2C_7(u_5S_5 - u_3C_4S_5) + 
\dot{u}_4C_3 + C_3S_7(u_4C_5 - u_5C_4S_5) + u_4S_5C_3 + u_3C_4S_5) + S_7 \lambda(u_5 + u_3S_5) - S_7 \lambda(u_5S_5 - u_3S_5); + 
\dot{u}_4C_4S_5 + u_4S_5C_3 + u_3C_4S_5) + u_4S_5C_3 + u_3C_4S_5) + S_7 \lambda(u_5 + u_3S_5) - S_7 \lambda(u_5S_5 - u_3S_5) \right) \]

\[ \alpha_{D2} = \left( (u_1 + C_1(u_4S_5 + u_5C_3) + S_2(u_7C_3 - u_3C_2S_5))(C_7 + (u_5 + u_3S_5) - S_2S_7(u_2C_3 - u_3C_4S_5) + 
S_2S_7(u_4S_5 + u_5C_4S_5)) - (u_7 + C_2(u_4S_5 + u_5C_3) + S_2(u_4C_5 - u_5C_4S_5))(u_8 + C_7(u_5 + u_3S_5) - 
C_2S_7(u_4C_5 - u_5C_4S_5) + S_2S_7(u_4S_5 + u_5C_4S_5)) + u_2C_5(u_5 + u_3S_5) - C_2C_7(u_5S_5 - u_3C_4S_5) + 
\dot{u}_4C_3 + C_3S_7(u_4C_5 - u_5C_4S_5) + u_4S_5C_3 + u_3C_4S_5) + S_7 \lambda(u_5 + u_3S_5) - S_7 \lambda(u_5S_5 - u_3S_5); + 
\dot{u}_4C_4S_5 + u_4S_5C_3 + u_3C_4S_5) + u_4S_5C_3 + u_3C_4S_5) + S_7 \lambda(u_5 + u_3S_5) - S_7 \lambda(u_5S_5 - u_3S_5) \right) \]
APPENDIX F: Point Acceleration Terms

The acceleration of the rear wheel mass center is

\[ \mathbf{a}_{G_k} = \begin{bmatrix} 0 & -r_{RW}C_5 & 0 \\ r_{RW} & 0 & 0 \\ 0 & -r_{RW}S_5 & 0 \end{bmatrix}_{B} \begin{bmatrix} \dot{u}_4 \\ \dot{u}_6 \\ \dot{u}_7 \end{bmatrix} + \begin{bmatrix} -r_{RW}C_5S_4 & -r_{RW}C_5 & 0 \\ 0 & 0 & 0 \\ -r_{RW}S_4S_5 & -r_{RW}S_5 & 0 \end{bmatrix}_{B} \begin{bmatrix} \dot{u}_3 \\ \dot{u}_5 \\ \dot{u}_8 \end{bmatrix} + \begin{bmatrix} aA_{k1} \\ aA_{k2} \\ aA_{k3} \end{bmatrix}_{B} \]

Where

\[ aA_{k1} = -r_{RW}(u_3^2S_4 + u_5^2S_5 - u_3^2C_4^2S_4 + u_3u_5S_4S_5 + u_4u_6S_4S_5 + 2u_3u_4C_4C_5) \]
\[ aA_{k2} = -r_{RW}u_5(u_3S_{2q_4})/2 + u_3C_4 + u_6C_4 \]
\[ aA_{k3} = r_{RW}(u_3^2C_5 + u_4^2C_5 - u_3^2C_4^2C_5 - 2u_4u_4C_4S_5 + u_4u_5C_5S_4 + u_6C_5S_4) \]

The acceleration of the rear bicycle frame mass center is

\[ \mathbf{a}_{G_k} = \begin{bmatrix} 0 & -r_{RW}C_5 & 0 \\ r_{RW} + z_BC_5 + x_BS_5 & 0 & 0 \\ 0 & -r_{RW}S_5 & 0 \end{bmatrix}_{B} \begin{bmatrix} \dot{u}_4 \\ \dot{u}_6 \\ \dot{u}_7 \end{bmatrix} + \begin{bmatrix} -z_BS_4 - r_{RW}C_5S_4 & -z_B - r_{RW}C_5 & 0 \\ x_BC_5 - z_BS_5 & z_BC_5S_4 & 0 \\ -x_BS_5 - r_{RW}S_4S_5 & -x_B - r_{RW}S_5 & 0 \end{bmatrix}_{B} \begin{bmatrix} \dot{u}_3 \\ \dot{u}_5 \\ \dot{u}_8 \end{bmatrix} + \begin{bmatrix} aB_{k1} \\ aB_{k2} \\ aB_{k3} \end{bmatrix}_{B} \]

\[ aB_{k1} = \left( \frac{r_{RW}u_5S_4(u_5 + u_6 + u_3S_4) - x_B(u_5 + u_3S_4)^2}{2} - r_{RW}u_5(u_4S_5 + u_5C_4C_5) - (u_4S_5 + u_5C_4C_5)(u_4z_BC_5 + u_4C_5S_4) \right) \]
\[ aB_{k2} = \left( \frac{2u_4u_5x_BC_5 - r_{RW}u_5u_6C_4 - r_{RW}u_5u_6C_4}{2} - \frac{(r_{RW}u_3S_{2q_4})}{2} - 2u_4u_5z_BS_5 - u_3^2z_BC_4C_5S_4 - u_3^2x_BC_4S_4S_5 - 2u_3z_Bx_BC_4C_5 - 2u_3u_5x_BC_4S_5 \right) \]
\[ aB_{k3} = \left( \frac{(u_4C_5 - u_3C_4S_4)(u_4z_BC_5 + u_4x_BC_4 - u_3z_BC_4S_5 + u_3x_BC_4C_5) + z_B(u_5 + u_3S_4)^2 + r_{RW}u_4(u_4C_5 - u_3C_4S_5)}{2} - r_{RW}u_5C_5(u_5 + u_3S_4)(u_5 + u_6 + u_5S_4) - u_3u_5x_BC_4 - r_{RW}u_5u_4C_4S_5 \right) \]
The acceleration of point S is

$$\mathbf{a}^S = \begin{bmatrix} 0 & -r_{RW} C_5 & 0 \\ r_{RW} + z_5 C_5 + x_5 S_5 & 0 & 0 \\ 0 & -r_{RW} S_5 & 0 \end{bmatrix} \begin{bmatrix} \dot{u}_4 \\ \dot{u}_5 \\ \dot{u}_6 \end{bmatrix}$$

+ \begin{bmatrix} -z_5 S_4 - r_{RW} C_5 S_4 & -z_5 - r_{RW} C_5 \\ x_5 C_4 C_5 - z_5 C_5 S_5 & 0 \\ -x_5 S_4 - r_{RW} S_5 S_5 & -x_5 - r_{RW} S_5 \end{bmatrix} \begin{bmatrix} \ddot{u}_4 \\ \ddot{u}_5 \\ \ddot{u}_6 \end{bmatrix} + \begin{bmatrix} a_{S_{i1}} \\ a_{S_{i2}} \\ a_{S_{i3}} \end{bmatrix}_B$$

Where

$$a_{S_{i1}} = \begin{bmatrix} r_{RW} u_5 S_5 (u_5 + u_6 + u_5 S_4) - x_5 (u_5 + u_3 S_4)^2 - r_{RW} u_4 (u_3 S_5 + u_3 C_4 C_5) - (u_3 S_5 + u_3 C_4 C_5)(u_4 S_5 + ) \\ u_4 x_5 S_5 - u_5 z_5 C_4 S_5 + u_3 x_5 C_4 C_5 - r_{RW} S_5 (u_5 + u_3 S_4)(u_5 + u_6 + u_3 S_4 - u_3 u_4 S_5 C_4 - r_{RW} u_3 u_4 C_4 C_5 \end{bmatrix}$$

$$a_{S_{i2}} = \begin{bmatrix} 2u_4 u_5 x_5 C_5 - r_{RW} u_5 u_5 C_4 - r_{RW} u_3 u_5 C_4 - (r_{RW} u_3^2 S_{q1}) / 2 - 2u_4 u_5 z_5 S_5 - \\ u_5^2 z_4 C_4 S_4 - u_5 x_5 C_4 S_4 - 2u_3 u_5 z_5 C_4 C_5 - 2u_4 u_5 x_5 C_4 S_4 \end{bmatrix}$$

$$a_{S_{i3}} = \begin{bmatrix} (u_4 S_5 - u_3 C_4 S_5)(u_4 z_5 C_5 + u_4 x_5 S_5 - u_5 z_4 C_4 S_5 + u_5 x_5 C_4 C_5) + z_5 (u_5 + u_3 S_4)^2 + r_{RW} u_4 (u_3 C_5 - u_3 S_5) - \\ r_{RW} u_3 C_5 (u_5 + u_6 + u_3 S_4) + r_{RW} C_5 (u_5 + u_3 S_4)(u_5 + u_6 + u_3 S_4) - u_3 u_4 x_5 C_4 - r_{RW} u_3 u_4 C_4 S_5 \end{bmatrix}$$

The acceleration of the front frame mass center is

$$\mathbf{a}^G = \begin{bmatrix} a_{C_{i11}} & a_{C_{i12}} & 0 \\ a_{C_{i12}} & a_{C_{i22}} & a_{C_{i23}} \\ a_{C_{i13}} & a_{C_{i32}} & 0 \end{bmatrix} \begin{bmatrix} \dot{u}_4 \\ \dot{u}_5 \\ \dot{u}_6 \end{bmatrix}$$

+ \begin{bmatrix} a_{C_{i11}} & a_{C_{i12}} & 0 \\ a_{C_{i21}} & a_{C_{i22}} & 0 \\ a_{C_{i31}} & a_{C_{i32}} & 0 \end{bmatrix} \begin{bmatrix} \ddot{u}_4 \\ \ddot{u}_5 \\ \ddot{u}_6 \end{bmatrix} + \begin{bmatrix} a_{C_{k1}} \\ a_{C_{k2}} \\ a_{C_{k3}} \end{bmatrix}_C$$

Where

$$a_{C_{i11}} = S_7 (r_{RW} - z_5 C_5 S_5 + z_3 C_5 + x_5 S_5)$$

$$a_{C_{i12}} = -r_{RW} C_5 - S_7$$

$$a_{C_{i13}} = C_7 (r_{RW} + z_5 C_5 + x_5 S_5) + x_5 S_5 S_7 - -z_5 C_5 - S_7$$

$$a_{C_{i21}} = C_7 (r_{RW} + z_5 C_5 + x_5 S_5) + x_5 S_5 S_7 - -z_5 C_5 - S_7$$

$$a_{C_{i22}} = r_{RW} C_5 S_7$$

$$a_{C_{i23}} = x_5$$
\[ aC_{i31} = x_c C_{\lambda s} S_7 \]

\[ aC_{i32} = -r_{RW} S_{\lambda s} \]

And

\[ aC_{d11} = \left( z_c (C_s S_4 + C_s C_4 S_3 S_7 + C_s C_5 S_\lambda S_7) + C_s S_7 (x_3 C_5 - z_3 S_3) - C_s C_7 S_4 (z_3 + r_{RW} C_3) + C_s S_2 S_4 (x_3 + r_{RW} S_5) \right) \]

\[ aC_{d12} = C_s (z_c - r_{RW} C_{\lambda s} - z_3 C_\lambda + x_3 S_3) \]

\[ aC_{d21} = \left( z_c (C_s C_4 C_7 S_5 - S_4 S_7 + C_s C_5 C_7 S_\lambda + x_3 C_{\lambda s} S_4 + C_s S_7 (x_3 C_5 - z_3 S_3)) + C_s S_2 S_4 (x_3 + r_{RW} S_5) \right) \]

\[ aC_{d22} = -S_7 (z_c - r_{RW} C_{\lambda s} - z_3 C_\lambda + x_3 S_3) \]

\[ aC_{d31} = -x_c (C_s S_4 + C_s C_4 S_3 S_7 + C_s C_5 S_\lambda S_7) - C_s S_3 (x_3 + r_{RW} S_5) - S_\lambda S_4 (z_3 + r_{RW} C_3) \]

\[ aC_{d32} = -r_{RW} S_{\lambda s} - x_3 C_\lambda - x_3 C_7 - z_3 S_\lambda \]

And

\[ aC_{e1} = \left\{ z_c (S_s (u_3 + u_3 S_3) + C_s C_4 (u_4 C_5 - u_4 C_3 S_s) - C_s S_3 (u_3 S_5 + u_3 C_s C_3)) - x_c (u_3 + (u_3 C_{\lambda s}) / 2 + u_3 S_{\lambda s}) + (u_3 S_{\lambda s}) / 2)(u_4 + (u_3 C_{\lambda s}) / 2 + u_3 S_{\lambda s}) / 2 - S_7 (r_{RW} C_3 S_7 + r_{RW} u_3 C_4 + r_{RW} u_3 C_4 - 2 u_3 u_3 C_4) - u_3 C_s C_3 C_s + u_3 S_3 C_3 C_3 + u_3 S_3 C_3 C_3 - 2 u_3 u_3 C_3 C_3 + 2 u_3 u_3 C_3 C_3 - z_c (u_3 (u_3 + u_3 S_3) - C_3 S_s (u_4 S_5 - u_4 C_3 C_3) + u_3 S_3 (u_3 S_5 + u_3 C_s C_3)) + C_s S_3 (u_3 S_5 + u_3 C_s C_3) - u_3 S_3 (u_3 S_5 + u_3 C_s C_3) - x_c (C_s (u_3 + u_3 C_3) - C_s S_3 (u_4 S_5 - u_4 C_3 C_3) + u_3 C_s C_3 C_s + u_3 S_3 C_3 C_3) \right\}
\[
aC_{s2} = \left\{ \begin{array}{l}
C_\lambda S_7 ((u_4 S_5 + u_3 C_4 C_7) (u_4 z_5 C_5 + u_4 x_5 S_5 - u_3 z_5 C_4 S_5 + u_3 x_5 C_4 C_5) + x_5 (u_5 + u_3 S_4)^2 + \\
r_{RW} u_4 (u_3 S_5 + u_3 C_4 C_5) - r_{RW} u_5 S_5 (u_5 + u_6 + u_3 S_4) + r_{RW} S_5 (u_5 + u_5 S_4)(u_5 + u_6 + u_3 S_4) + \\
u_t u_4 z_4 C_4 + r_{RW} u_t u_4 C_4 C_5) - C_7 (r_{RW} u_3^2 C_4 S_4 + r_{RW} u_t u_4 C_4 + r_{RW} u_t u_4 C_4 - 2 u_t u_4 x_4 C_5 + \\
2 u_4 u_5 z_5 S_5 + u_5^2 z_5 C_4 S_4 + u_4^2 x_5 C_4 S_4 + 2 u_4 u_5 S_5 C_4 S_5 + 2 u_t u_4 x_4 C_4 S_5 - \\
z_c (u_4 C_7 (u_5 + u_3 S_4) - C_7 C_5 (u_4 u_3 S_5 - u_3 u_4 S_5) + u_4 u_3 C_4 S_5) + C_7 S_5 (u_4 u_3 C_5 - \\
u_t u_4 C_5 + u_4 u_3 C_5) + u_4 u_3 S_5 - u_3 C_5 S_5 (u_4 C_5 - u_3 C_4 S_5) + u_t S_5 (u_5 + u_3 S_4) - \\
x_c (C_5 u_4 S_5 - u_t u_4 C_5 + u_t u_4 C_5 S_5) + S_5 (u_4 u_3 S_5 - u_3 u_4 S_5 + u_t u_4 C_5) + \\
S_5 S_7 ((u_4 C_5 - u_3 C_4 S_5) (u_4 z_5 C_5 + u_4 x_5 S_5 - u_3 z_5 C_4 S_5 + u_3 x_5 C_4 C_5) + z_c (u_5 + u_3 S_4)^2 + \\
r_{RW} u_4 (u_3 C_5 - u_3 C_4 S_5) - r_{RW} u_5 C_5 (u_5 + u_6 + u_3 S_4)(u_5 + u_6 + u_3 S_4) - u_3 u_4 x_4 C_4 - \\
r_{RW} u_4 u_3 C_4 S_5 + x_c (S_5 (u_5 + u_3 S_4) + C_7 C_5 (u_4 C_5 - u_3 C_4 S_5) - \\
C_7 S_5 (u_4 S_5 + u_4 C_5))(S_5 (u_5 + u_3 S_4) - C_7 S_7 (u_4 C_5 - u_3 C_4 S_5) + S_7 S_5 (u_5 + u_4 C_5) - \\
z_c (C_7 (u_5 + u_3 S_4) - C_7 S_7 (u_4 C_5 - u_3 C_4 S_5) + S_7 S_5 (u_5 + u_4 C_5))(u_7 + (u_7 C_5) / 2 + \\
(u_4 S_4^2 + u_4^2 C_5) / 2) / 2) \end{array} \right. \\
\end{array}
\]

\[
aC_{s3} = \left\{ \begin{array}{l}
C_\lambda ((u_4 C_5 - u_3 C_4 S_5) (u_4 z_5 C_5 + u_4 x_5 S_5 - u_3 z_5 C_4 S_5 + u_3 x_5 C_4 C_5) + z_c (u_5 + u_3 S_4)^2 + r_{RW} u_4 (u_4 C_5 - \\
u_3 C_4 S_5) - r_{RW} u_4 C_5 (u_5 + u_6 + u_3 S_4) + r_{RW} C_5 (u_5 + u_6 + u_3 S_4)(u_5 + u_6 + u_3 S_4) - r_{RW} u_4 u_3 C_4 S_5) - \\
S_5 ((u_4 C_5 + u_4 C_4 S_5) (u_4 z_5 C_5 + u_4 x_5 S_5 - u_3 z_5 C_4 S_5 + u_3 x_5 C_4 C_5) + x_c (u_5 + u_3 S_4)^2 + r_{RW} u_4 (u_4 C_5 - \\
u_3 C_4 S_5) - r_{RW} u_3 C_4 S_5 (u_5 + u_6 + u_3 S_4) + r_{RW} S_5 (u_5 + u_6 + u_3 S_4)(u_5 + u_6 + u_3 S_4) + u_t u_4 z_4 C_4 + r_{RW} u_t u_4 C_4 C_5 - \\
z_c (C_7 (u_5 + u_3 S_4) - C_7 S_5 (u_4 C_5 - u_3 C_4 S_5) + S_5 S_7 (u_4 S_5 + u_4 C_5)) - z_c (S_5 (u_5 + u_3 S_4) + \\
C_7 C_5 (u_4 C_5 - u_3 C_4 S_5) - C_7 S_7 (u_4 S_5 + u_4 C_5))(u_7 + (u_7 C_5) / 2 + \\
(u_4 S_4^2 + u_4^2 C_5) / 2) / 2) (S_5 (u_5 + u_3 S_4) + C_7 C_5 (u_4 C_5 - u_3 C_4 S_5) - C_7 S_7 (u_4 S_5 + u_4 C_5)) + \\
x_c (u_4 S_7 (u_5 + u_3 S_4) - C_7 S_7 (u_4 S_5 + u_4 C_5))(u_4 S_5 + u_4 C_5) + S_2 S_7 (u_4 u_3 C_5 - u_4 u_3 C_5 + \\
(u_3 u_4 C_5) + u_t C_7 C_7 (u_5 + u_3 S_5 - u_4 C_5 - u_3 C_4 S_5) - u_t C_7 S_7 (u_4 S_5 + u_4 C_5) - u_3 u_4 C_5) + u_t C_7 C_7 (u_5 + u_3 S_5 - u_4 C_5 - u_3 C_4 S_5) - u_t C_7 S_7 (u_4 S_5 + u_4 C_5) - u_3 u_4 C_5) + \end{array} \right. \\
\end{array}
\]

95
And finally, the acceleration of the front wheel mass center is

\[
\frac{g}{a_{GW}} = \begin{bmatrix}
    a_{d_{i1}} & a_{d_{i2}} & 0 \\
    a_{d_{i1}} & a_{d_{i2}} & a_{d_{i3}} \\
    a_{d_{i1}} & a_{d_{i2}} & 0
\end{bmatrix}
\begin{bmatrix}
    \dot{u}_4 \\
    \dot{u}_6 \\
    \dot{u}_7
\end{bmatrix}
+ \begin{bmatrix}
    a_{d_{d1}} & a_{d_{d2}} & 0 \\
    a_{d_{d1}} & a_{d_{d2}} & a_{d_{d3}} \\
    a_{d_{d1}} & a_{d_{d2}} & 0
\end{bmatrix}
\begin{bmatrix}
    \dot{u}_3 \\
    \dot{u}_5 \\
    \dot{u}_8
\end{bmatrix}
+ \begin{bmatrix}
    a_{D_1} \\
    a_{D_2} \\
    a_{D_3}
\end{bmatrix}
\]

Where

\[
a_{d_{i1}} = S_7 (r_{RW} - z_D C_{\lambda+5} + z_3 C_5 + x_5 S_5)
\]

\[
a_{d_{i2}} = -r_{RW} C_{\lambda+5} C_7
\]

\[
a_{d_{i3}} = C_7 (r_{RW} + z_3 C_5 + x_5 S_5)
\]

\[
a_{d_{i4}} = r_{RW} C_{\lambda+5} S_7
\]

\[
a_{d_{i5}} = x_D
\]

\[
a_{d_{i6}} = x_D C_{\lambda+5} S_7
\]

\[
a_{d_{i7}} = -r_{RW} S_{\lambda+5}
\]

And

\[
a_{d_{d1}} = \begin{cases}
    z_D (C_7 S_4 + C_4 C_7 S_7) + C_4 S_4 (x_5 C_5 - z_5 S_5) - C_4 C_7 S_4 (z_5 + r_{RW} C_5) & \\
    C_7 S_4 (x_5 + r_{RW} S_5)
\end{cases}
\]

\[
a_{d_{d2}} = C_7 (z_D - r_{RW} C_{\lambda+5} - z_5 C_\lambda + x_5 S_\lambda)
\]

\[
a_{d_{d3}} = \begin{cases}
    z_D (C_4 C_7 S_7) + C_4 C_7 S_7 & \\
    C_4 C_7 (x_5 C_5 - z_5 S_5) + C_4 C_7 (x_5 C_5 - z_5 S_5) & \\
    C_4 S_4 S_7 (z_5 + r_{RW} C_5) - S_4 S_7 (x_5 + r_{RW} S_5)
\end{cases}
\]

\[
a_{d_{d4}} = -S_7 (z_D - r_{RW} C_{\lambda+5} - z_5 C_\lambda + x_5 S_\lambda)
\]

\[
a_{d_{d5}} = -x_D (C_7 S_4 + C_4 C_7 S_7 + C_4 C_5 S_2 S_7) - C_4 S_4 (x_5 + r_{RW} S_5) - S_4 S_7 (x_5 + r_{RW} C_5)
\]

\[
a_{d_{d6}} = -r_{RW} S_{\lambda+5} - x_5 C_{\lambda} - x_5 C_{\lambda} - z_5 S_\lambda
\]
\[
\begin{align*}
\text{And} \\
\left(z_0(S_5(u_5 + u_5S_5) + C_3C_7(u_5C_5 - u_5C_5S_5) - C_7S_5(u_5S_5 + u_5C_5C_5)) - x_0(u_5 + (u_5C_{\lambda + 4})/2 + u_5S_{\lambda + 5}) + \\
(u_5C_{\lambda + 5})/2)(u_5 + (u_5C_{\lambda + 4})/2 + u_5S_{\lambda + 5} + (u_5C_{\lambda + 4})/2) - S_7(r_{\text{RW}}u_5^2C_5S_5 + r_{\text{RW}}u_5u_5C_5 + r_{\text{RW}}u_5u_5^2C_5 - \\
2u_5u_5x_5C_5 + 2u_5u_5z_5S_5 + u_5^2z_5C_5S_5 + u_5^2x_5C_5S_5 + 2u_5u_5z_5^2C_5 + 2u_5u_5x_5C_5S_5) - \\
z_0(u_5S_5(u_5 + u_5S_5) - C_5S_5(u_5u_5S_5 - u_5u_5S_5S_5 + u_5u_5C_5S_5) + S_5S_5(u_5u_5S_5 - u_5u_5S_5S_5 + u_5u_5C_5S_5) + \\
u_5C_5C_5C_5(u_5C_5 - u_5C_5S_5) - u_5C_5C_5S_5 + u_5C_5S_5 - u_5C_5^2S_5)(C_5(u_5 + u_5S_5) - C_5S_5) - \\
x_5(u_5 + u_5S_5)^2 + r_{\text{RW}}u_5(u_5S_5 + u_5C_5S_5) - r_{\text{RW}}u_5S_5(u_5 + u_5S_5) + r_{\text{RW}}S_5(u_5 + u_5S_5) + r_{\text{RW}}C_5(u_5 + u_5S_5) - \\
u_5u_5C_5C_5 + r_{\text{RW}}u_5C_5C_5 - C_5S_5((u_5S_5 + u_5C_5S_5) + u_5x_5S_5 - u_5z_5C_5S_5 + u_5x_5C_5S_5) + \\
z_5(u_5 + u_5S_5)^2 + r_{\text{RW}}u_5(u_5C_5 - u_5C_5S_5) - r_{\text{RW}}u_5S_5(u_5 + u_5S_5) + r_{\text{RW}}C_5(u_5 + u_5S_5)(u_5 + u_5S_5) - \\
u_5u_5x_5C_5 - r_{\text{RW}}u_5u_5C_5S_5) \\
\right)
\end{align*}
\]

\[
\begin{align*}
\text{As} \\
\left(C_5C_7((u_5S_5 + u_5C_5S_5) + u_5x_5S_5 - u_5z_5S_5C_5 + u_5x_5C_5C_5) + x_5(u_5 + u_5S_5)^2 + \\
r_{\text{RW}}u_5(u_5S_5 + u_5C_5S_5) - r_{\text{RW}}u_5S_5(u_5 + u_5S_5) + r_{\text{RW}}S_5(u_5 + u_5S_5)(u_5 + u_5S_5) + \\
u_5u_5z_5S_5 + r_{\text{RW}}u_5u_5C_5S_5 - C_5((r_{\text{RW}}u_5^2C_5S_5 + r_{\text{RW}}u_5u_5C_5 + r_{\text{RW}}u_5u_5^2C_5 - 2u_5u_5x_5C_5 + \\
2u_5u_5z_5S_5 + u_5^2x_5S_5C_5 + u_5^2x_5C_5S_5 + 2u_5u_5z_5C_5S_5 + 2u_5u_5x_5S_5C_5S_5 - \\
z_0(u_5C_5(u_5 + u_5S_5) - C_5C_7(u_5u_5S_5 - u_5u_5C_5S_5 + u_5u_5C_5S_5) + C_7S_5(u_5u_5C_5 - \\
u_5u_5C_5 + u_5u_5C_5S_5 + u_5u_5C_5S_5 - u_5C_5S_5(u_5S_5 - u_5S_5S_5 + u_5S_5C_5) - \\
x_0C_5(u_5C_5 - u_5C_5S_5) + S_5((u_5C_5 - u_5C_5S_5) + u_5x_5S_5 - u_5z_5S_5 + u_5C_5S_5) + z_5(u_5 + u_5S_5)^2 + \\
r_{\text{RW}}u_5(u_5C_5 - u_5S_5)(u_5 + u_5S_5) + u_5u_5S_5(u_5 + u_5S_5) + r_{\text{RW}}u_5S_5(u_5 + u_5S_5) - \\
u_5u_5x_5C_5 + x_5(u_5 + u_5S_5) + C_5C_7(u_5C_5 - u_5C_5S_5) - \\
C_7S_5(u_5C_5 + u_5C_5S_5)(C_5(u_5 + u_5S_5) - C_5C_7(u_5C_5 - u_5C_5S_5) + S_5S_5(u_5S_5 + u_5C_5S_5) + \\
z_5(u_5 + u_5S_5)(u_5 + u_5S_5) - C_5C_7(u_5C_5 - u_5C_5S_5) + S_5S_5(u_5C_5 + u_5C_5S_5)(u_5 + (u_5C_{\lambda + 4})/2 + \\
u_5C_{\lambda + 5}(u_5C_{\lambda + 4})2) \\
\right)
\end{align*}
\]
\[
\begin{align*}
\text{AD}_3 &= \left\{ C_2 ((u_4C_5 - u_3C_4S_2)(u_4z_3C_5 + u_4x_3S_5 - u_3z_3S_4C_5 + u_3x_3C_4C_5) + z_3(u_5 + u_3S_4)^2 + r_{\text{rw}}u_4(u_4C_5 - \\
u_3C_4S_5) - r_{\text{rw}}u_3C_5(u_5 + u_6 + u_3S_4) + r_{\text{rw}}C_5(u_5 + u_6 + u_3S_4) - u_3u_4x_3S_4 - r_{\text{rw}}u_3u_4C_4S_5) - \\
S_1((u_4S_5 + u_3C_5C_5)(u_4z_3C_5 + u_4x_3S_5 - u_3z_3C_5 + u_3x_3C_4C_5) + x_3(u_5 + u_3S_4)^2 + r_{\text{rw}}u_4(u_4S_5 + \\
u_3C_4C_5) - r_{\text{rw}}u_3S_5(u_5 + u_6 + u_3S_4) + r_{\text{rw}}S_5(u_5 + u_6 + u_3S_4) + u_3u_4z_3C_4 + r_{\text{rw}}u_3u_4C_4C_5) - \\
z_{\text{D}}(C_7(u_5 + u_3S_4) - C_\lambda S_7(u_4C_5 - u_4C_4S_5) + S_2S_7(u_4S_5 + u_3C_4C_5))^2 - (z_{\text{D}}(S_7(u_5 + u_3S_4) + \\
C_2C_7(u_4C_5 - u_4C_4C_5) - C_\gamma S_7(u_4S_5 + u_3C_4C_5)) - x_{\text{D}}(u_5 + (u_3C_\lambda - 4 + 5)^2 + u_4S_\lambda + \\
(u_5C_\lambda + 5)/2))(S_7(u_5 + u_3S_4) + C_\gamma C_7(u_4C_5 - u_4C_4S_5) - C_\gamma S_7(u_4S_5 + u_3C_4C_5) + \\
x_{\text{D}}(u_5S_7(u_5 + u_3S_4) - C_\lambda S_7(u_4u_5S_5 - u_3u_4S_5 + u_3u_3C_4C_5) + S_2S_7(u_3u_4C_5S_4 - u_4u_3C_5 + \\
u_3u_4C_4S_5) + u_5C_2C_7(u_4C_5 - u_4C_4S_5) - u_5C_3S_7(u_4S_5 + u_3C_4C_5) - u_5u_4C_4C_5) \\
\right\}
\end{align*}
\]
Symbolic Equations File

Bicycle EOMs Version 3

% Western Michigan University
% Spring 2017
% Code by Brent Kostich

Section 1: Model Description

{%
    ================== Section Description =================

    The bicycle itself has four bodies, each of which has a local coordinate frame.

    % ------- Frames ------- %
    R - Inertial frame (Newtonian)
    Y - Yaw Frame
    L - Lean Frame
    A - Rear Wheel
    B - Rear Frame
    C - Front Frame
    D - Front Wheel

    The bicycle has an eight-dimensional configuration space, which is described by the following generalized coordinates. Of these eight, the first two are ignored in the derivation of the dynamical equations. They are unnecessary because they merely define the global positon of the rear wheel contact point. They are defined so that later adaptations of this model can calculate them.
%}
% Generalized Coordinates
q1 = xp - position of rear wheel contact point
q2 = yp - position of rear wheel contact point
q3 = psi - yaw angle
q4 = phi - roll angle
q5 = theta - pitch angle
q6 = theta_r - rear wheel rotation angle
q7 = delta - steering angle
q8 = theta_f - front wheel rotation angle

Symbolic variables are created as functions of time for each
generalized coordinate. The time dependence allows them to be properly
differentiated during the derivation of the system kinematics.

clear

Section 2: Reference Frames and Orientation Angles

% Section Description
This section covers the various rotation matrices that define the
orientation of the different bicycle frames. The transpose of each matrix
is also included, for ease of use in later computations.

The rotation is defined so that is describes how the unit vectors relate
to each other. For example, BtoC means transforming the B-frame unit
vectors to C-frame unit vectors.

Rotation Sequences

A 3-1-2 rotation sequence was used to relate the orientation of the
bicycle rear frame to the inertial frame.

The intermediate frames are Y and L, the yaw frame and lean frame,
respectively. These are explicitly included because of their contribution
to transforming several critical components of the equations.

Y rotates relative to R about the y3 axis (q3)
L rotates relative to Y about the l1 axis (q4)
B rotates relative to L about the b2 axis (q5)
A rotates relative to B about the b2 axis (q6)

There is a two sequence rotation to get from the B frame to the C frame.
This is due to the constant steering tilt axis angle that is found
between the rear frame (B) and the front frame (C) of the bicycle.

C rotates relative to B by first taking a rotation about the b2 axis
(lamba, the steering tilt angle), then by taking a rotation about the
c3 axis (q7)
D rotates relative to C about the c2 axis (q8)

```

* % - Symbolic variables
  syms Q1(t) Q2(t) Q3(t) Q4(t) Q5(t) Q6(t) Q7(t) Q8(t)
  syms U1(t) U2(t) U3(t) U4(t) U5(t) U6(t) U7(t) U8(t)
  syms x_B x_B x_S x_S x_C x_C x_D x_D r_RW r_FW lambda

% -- Inertial frame to Yaw frame -- %
YtoR = [cos(Q3) -sin(Q3) 0;
       sin(Q3)  cos(Q3) 0;
       0       0      1];

RtoY = YtoR.';

% -- Yaw frame to Lean frame -- %
LtoY = [1 0 0;
       0  cos(Q4) -sin(Q4);
       0  sin(Q4)  cos(Q4)];

YtoL = LtoY.';

% -- Lean frame to B frame -- %
BtoL = [cos(Q5) 0  sin(Q5);
        0  1  0;
       -sin(Q5) 0  cos(Q5)];

LtoB = BtoL.';

% -- B frame to A frame -- %
AtoB = [cos(Q6) 0  sin(Q6);
        0  1  0;
       -sin(Q6) 0  cos(Q6)];

BtoA = AtoB.';

% -- B frame to C prime frame (steering tilt frame) -- %
C_ptoB = [cos(lambda) 0  sin(lambda);
          0  1  0;
       -sin(lambda) 0  cos(lambda)];

BtoC_p = C_ptoB.';

% -- C prime frame to C frame -- %
CtoC_p = [cos(Q7) -sin(Q7) 0;
          sin(Q7)  cos(Q7) 0;
          0       0      1];

C_ptoC = CtoC_p.';

% -- D frame to C frame -- %
DtoC = [cos(Q8) 0  sin(Q8);
        0  1  0;
       -sin(Q8) 0  cos(Q8)];
```
CtoD = DtoC.';

% ---- Useful Rotation Combinations ---- %
% -- C frame to B frame -- %
CtoB = C_ptoB*CtoC_p;
BtoC = CtoB.';
% -- R frame to B frame -- %
RtoB = LtoB*YtoL*RtoY;
BtoR = RtoB.';

Section 3: Position Vectors

{%
    ------------------------------- Section Description -------------------------------
This section defines six of the seven points used to define the geometry of the bicycle. The seventh is defined in the next section. These points are related through the use of relative position vectors.

The notation 'R_p_o' is the position vector of P with respect to O. It can also be thought of as going from point O to point P.
%
%
R_p_o = [Q1;Q2;0];  % In R frame
R_rw_p = [0;0;-r_RW]; % In L frame
R_rf_rw = [x_B;0;-z_B]; % In B frame
R_s_rw = [x_S;0;-z_S]; % In B frame
R_fw_s = [x_D;0;x_D]; % In C frame
R_ff_s = [x_C;0;x_C]; % In C frame
%
% ---- Useful Position Vector Combinations ---- %
R_rw_o = (YtoL*RtoY)*R_p_o + R_rw_p;  % In L frame
R_p_rw = -R_rw_p;  % In L frame
%
%
Section 4: Triple Vector Product for Front Wheel Contact Point

{%
    ------------------------------- Section Description -------------------------------
This section determines the final position vector. The position vector points from the front wheel c.g. to the front wheel contact point. A triple vector product is used to establish the correct orientation.
%
%
RtoB_t = RtoB(t);
BtoC_t = BtoC(t);
\begin{verbatim}
RtoC  = BtoC_t*RtoB_t;  % Generating a rotation matrix from the R frame to the C frame
Z_c  = RtoC(:,3).';      % Expressing the 'Z' unit vector direction into C frame coordinates
c2   = [0;1;0];          % Vector representing the c2 unit vector direction
c2_tilde = SymCrossMatrix(c2,t);
beta  = c2_tilde*Z_c.';  % First vector cross product
beta_tilde = SymCrossMatrix(beta,t);
WCP = beta_tilde*c2;     % Complete triple vector product
mag = (WCP(1,:)*WCP(1,:) + WCP(2,:)*WCP(2,:)) + WCP(3,:)*WCP(3,:))^(1/2);
WCP = WCP/mag;           % The result is divided by its magnitude to create a unit vector set
R_q_fw = r_FW*WCP;       % Final expression of the front wheel contact point location, in the C frame
\end{verbatim}

Section 5: Front Wheel Holonomic Constraint

\%
%--------------------------------------------------------------------------
% This section develops the vector chain between the rear wheel contact point and the front wheel contact point. This relationship represents the holonomic constraint on the front wheel. It is used to solve for the initial value of the bicycle frame pitch angle. This is necessary to properly configure the bicycle prior to any simulation.
%--------------------------------------------------------------------------
%
\R_Q_P = LtoY*R_rw_pv + (LtoY*BtoL)*R_rw_pv + (LtoY*BtoL*CtoB)*R_rw_pv + ...
        (LtoY*BtoL*CtoB)*R_q_fw;
\R_Q_P = simplify(R_Q_P);
\R_Q_P = R_Q_P(t);
\R_Q_P_z = R_Q_P(3) == 0;
\R_QP_z = R_Q_P(3);

Section 6: Angular Velocities

\%
%--------------------------------------------------------------------------
% This section defines the angular velocities of the bicycle. The bicycle is connected by a series of single-axis revolute joints. These joints relate the rotations of one body relative to another. By summing these simple angular velocities, the program builds up the complex relation
%--------------------------------------------------------------------------
%
ships that express the angular velocities of each of the four bodies.

```matlab
omegaY_R = [0;0;U3]; % In Y frame
omegaL_Y = [U4;0;0]; % In L frame
omegaRF_L = [0;U5;0]; % In B frame
omegaRW_RF = [0;U6;0]; % In B frame
omegaFF_RF = [0;U7]; % In C frame
omegaFW_FF = [0;U8;0]; % In D frame

omegaRF_R = omegaRF_L + (LtoB)*omegaL_Y + (LtoB*YtoL)*omegaY_R; % In B frame
omegaRW_R = omegaRW_RF + omegaRF_R; % In B frame
omegaFF_R = omegaFF_RF + BtoC*omegaRF_R; % In C frame
omegaFW_R = omegaFW_FF + omegaFF_R; % In C frame
```

**Section 7: Point Velocities**

```matlab

%{  
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  Section Description %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  This section develops the velocities for all key points in the bicycle system. It takes advantage of the fact that the system is comprised of four rigid bodies with a few points that are shared between bodies. This allows for the use of relative velocity relationships.

At the velocity level, there are two constraints at each wheel contact point. These are enforced by setting the contact point velocity to zero.

%}

Q_dot_sym = [diff(Q1,t), diff(Q2,t), diff(Q3,t), diff(Q4,t),
             diff(Q5,t), diff(Q6,t), diff(Q7,t), diff(Q8,t)];

U_sym = [U1, U2, U3, U4, U5, U6, U7, U8];

% -- Velocity of Rear Wheel c.g. -- %
omegaRW_R_tilde = SymCrossMatrix(omegaRW_R,t);

V_RW_R = 0 + omegaRW_R_tilde*(LtoB*R_rw_p);  % In B frame

% -- Velocity of Rear Frame c.g. -- %
omegaRF_R_tilde = SymCrossMatrix(omegaRF_R,t);

V_RF_RW_R = omegaRF_R_tilde*R_rf_rw;
```
\[ V_{RF_R} = V_{RW_R} + V_{RF_RW_R}; \quad \text{\% In B frame} \]

\[ \text{\% -- Velocity of Bearing Point, S -- \%} \]
\[ V_{S_RW_R} = \omega_{RF_R} \tilde{t} \omega_{R_S_RW}; \]

\[ V_{S_R} = V_{RW_R} + V_{S_RW_R}; \quad \text{\% In B frame} \]

\[ \text{\% -- Velocity of Front Wheel c.g. -- \%} \]
\[ \omega_{FF_R} \tilde{t} \omega_{R_{FW_s}} = \text{SymCrossMatrix}(\omega_{FF_R}, t); \]

\[ V_{FW_S_R} = \omega_{FF_R} \tilde{t} \omega_{R_{FW_s}}; \]
\[ V_{FW_R} = BtoC*V_{S_R} + V_{FW_S_R}; \quad \text{\% In C frame} \]

\[ \text{\% -- Velocity of Front Frame c.g. -- \%} \]
\[ V_{FF_S_R} = \omega_{FF_R} \tilde{t} \omega_{R_{FF_s}}; \]
\[ V_{FF_R} = BtoC*V_{S_R} + V_{FF_S_R}; \quad \text{\% In C frame} \]

\[ \text{\% ---- Contact PointVelocities ---- \%} \]

\[ \text{\% -- Velocity of Front Wheel contact point, Q -- \%} \]
\[ \omega_{FW_R} \tilde{t} \omega_{R_{Q_{FW}}}; \quad \text{\% In C frame} \]

\[ \text{\% -- Velocity from Holonomic Constraint -- \%} \]
\[ \omega_{Y_R} \tilde{t} \omega_{R_{Y_R}} = \text{SymCrossMatrix}(\omega_{Y_R}, t); \]

\[ V_{Q_P} = \text{diff}(R_{QP}(t)) + \omega_{Y_R} \tilde{t} \omega_{R_{Q_P}}; \quad \text{\% In Y frame} \]
\[ V_{Q_P_Z} = V_{Q_P}(3); \]
\[ V_{Q_P_Z} = \text{subs}(V_{Q_P_Z}, Q_{dot\_syms}, U_{sym}); \]

---

**Section 8: Angular Accelerations**

\[ \text{\%\%} \]

\[ \text{------------- Section Description -------------} \]
This section develops the angular accelerations for each body of the bicycle. This is done with direct differentiation of the angular velocities. Two of the angular velocity vectors are expressed in a frame that is not local to their body, so the derivative rule is applied.

\[ \text{\%\%} \]

\[ \omega_{FW_R} \tilde{t} \omega_{R_{FW}} = \text{SymCrossMatrix}(\omega_{FW_R}, t); \]

\[ \alpha_{RF_R} = \text{diff}(\omega_{RF_R}, t); \quad \text{\% In B frame} \]
\[ \alpha_{RF_R} = \text{subs}(\alpha_{RF_R}, Q_{dot\_syms}, U_{sym}); \]

\[ \alpha_{RW_R} = \text{diff}(\omega_{RW_R}, t) + \omega_{RF_R} \tilde{t} \omega_{R_{FW}}; \quad \text{\% In B frame} \]
\[ \alpha_{RW_R} = \text{subs}(\alpha_{RW_R}, Q_{dot\_syms}, U_{sym}); \]

\[ \alpha_{RF_R} = \text{diff}(\omega_{RF_R}, t); \quad \text{\% In C frame} \]
\[ \alpha_{RF_R} = \text{subs}(\alpha_{RF_R}, Q_{dot\_syms}, U_{sym}); \]

\[ \alpha_{FW_R} = \text{diff}(\omega_{FW_R}, t) + \omega_{RF_R} \tilde{t} \omega_{R_{FW}}; \quad \text{\% In C frame} \]
alphaFW_R = subs(alphaFW_R,Q_dot_sym,U_sym);

**Section 9: Point Accelerations**

```
%  % Section Description  %  %
This section develops the accelerations of the mass centers of each of the 
fronds bodies. Similar to the velocity terms, this section uses relative 
accelerations to work between points.

However, the angular acceleration of the rear wheel mass center is 
computed by direct differentiation, using the derivative rule.

% Acceleration of Rear Wheel c.g.
alphaRW_R_tilde = SymCrossMatrix(alphaRW_R,t);

A_RW_R = diff(V_RW_R) + omegaRF_R_tilde*V_RW_R;    % In B frame

% Acceleration of Rear Frame c.g.
alphaRF_R_tilde = SymCrossMatrix(alphaRF_R,t);

A_RF_RW_R = alphaRF_R_tilde*R_rf_rw + ...
  omegaRF_R_tilde*(omegaRF_R_tilde*R_rf_rw);

A_RF_R = A_RW_R + A_RF_RW_R;   % In B frame

% Acceleration of Bearing Point, S
A_S_RW_R = alphaRF_R_tilde*R_s_rw ...
  omegaRF_R_tilde*(omegaRF_R_tilde*R_s_rw);

A_S_R = A_RW_R + A_S_RW_R;   % In B frame

% Acceleration of the Front Wheel c.g.
alphaFF_R_tilde = SymCrossMatrix(alphaFF_R,t);

A_FW_S_R = alphaFF_R_tilde*R_fw_s ...
  omegaFF_R_tilde*(omegaFF_R_tilde*R_fw_s);

A_FW_R = BtoC*A_S_R + A_FW_S_R;   % In C frame

% Acceleration of the Front Frame c.g.
A_FF_S_R = alphaFF_R_tilde*R_ff_s ...
  omegaFF_R_tilde*(omegaFF_R_tilde*R_ff_s);

A_FF_R = BtoC*A_S_R + A_FF_S_R;   % In C frame
```

**Section 10: Organizing Vectors**

```
%  % Section Description  %  %
This section goes through the process of reordering the necessary vectors 
to remove time dependence from the symbolic functions. This is purely a
```
programming step that is necessary to perform the functions of later sections.

})

-% Introducing new syms for time derivatives of generalized speeds
syms q1 q2 q3 q4 q5 q6 q7 q8 u1 u2 u3 u4 u5 u6 u7 u8
syms u1DT u2DT u3DT u4DT u5DT u6DT u7DT u8DT

-% Old symbolic variables to be replaced
SVars_old = [diff(U1,t),diff(U2,t),diff(U3,t),diff(U4,t),diff(U5,t),...
diff(U6,t),diff(U7,t),diff(U8,t),...
diff(Q1,t),diff(Q2,t),diff(Q3,t),diff(Q4,t),diff(Q5,t),...
diff(Q6,t),diff(Q7,t),diff(Q8,t),...
U1,U2,U3,U4,U5,U6,U7,U8,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8];

SVars_old = SVars_old(t);

-% New symbolic variables to be used
SVars_new = [u1DT,u2DT,u3DT,u4DT,u5DT,u6DT,u7DT,u8DT,....
u1,u2,u3,u4,u5,u6,u7,u8,...
u1,u2,u3,u4,u5,u6,u7,u8,q1,q2,q3,q4,q5,q6,q7,q8];

% ------------------ Angular Velocities ----------------------------- %
omegaRW_R_q = omegaRW_R(t); % Stays in B frame
omegaRF_R_q = omegaRF_R(t); % Stays in B frame
omegaFF_R_q = omegaFF_R(t); % Stays in C frame
omegaFW_R_q = omegaFW_R(t); % Stays in C frame

% ------------------ Velocities ----------------------------- %
V_RW_R_q = V_RW_R(t); % Stays in B frame
V_RF_R_q = V_RF_R(t); % Stays in B frame
V_FF_R_q = V_FF_R(t); % Stays in C frame
V_FW_R_q = V_FW_R(t); % Stays in C frame
V_S_R_q = V_S_R(t); % Stays in B frame
V_Q_R_q = V_Q_R(t); % Stays in C frame

% ------------------ Angular Accelerations ----------------------------- %
alphaRW_R_q = alphaRW_R(t); % Stays in B frame
alphaRF_R_q = alphaRF_R(t); % Stays in B frame
alphaFF_R_q = alphaFF_R(t); % Stays in C frame
alphaFW_R_q = alphaFW_R(t); % Stays in C frame
Section 11: Calculating the Coefficient Matrices

%! 
%----------------------- Section Description -----------------------
%This section calculates the C1 and C2 matrices that define the constraint equations. These matrices are exported into function files and used to compute the partial velocities necessary in Kane’s method.

The section also takes the time derivatives of the C1 and C2 matrices. These matrices are necessary in solving the derivatives of the dependent generalized speeds. This is done so the acceleration-level terms can be properly evaluated.
%

SVars_T = [Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,... % Symbolic variables with U1,U2,U3,U4,U5,U6,U7,U8]; % time dependence
Section 12: Active Forces

% ------------------------------- Section Description -------------------------------
This section covers the build-up of the active forces used to generate Kane's Equations.

The only forces that are acting on the system are the weight forces from
each body. These forces act through the center of mass of each body.

There are three torques that are defined to act on the system. These are
the roll torque, rear wheel torque and steering torque. These are considered to be actuated by a rider.

In a simplified sense, roll torque results from the rider's lateral
motion. Rear wheel torque generated from pedaling forces. Steering
torque is a result of the rider applying force to the handle bars.

The roll torque acts through the y1 axis of the yaw frame. The rear wheel
torque acts through the b2 axis for the rear frame. The steering torque
acts through the c3 axis of the front frame.

The active forces for Kane's Equations follow the form:

\[
\begin{vmatrix}
\text{SUM} & \text{F} & \text{M} \\
\text{dv} & \text{d(omega)} & \\
\text{du} & \\
\end{vmatrix}
\]

where,
- \(F\) = force acting on a body
- \(dv/du\) = partial velocity of the point through which the force
  is acting, w.r.t. a generalized speed
- \(M\) = torque acting on a body
- \(d(omega)/du\) = partial angular velocity of a body
- \(*\) = dot product

% ------------------ Forces ------------------ %
syms g m_RW m_RF m_FF m_FW % Syms for Forces
Fa_RW = [0;0;m_RW*g]; % In R frame
Fa_RW = simplify(RtoB*Fa_RW); % Altered to B frame
Fa_RW = Fa_RW(t);

Fa_RF = [0;0;m_RF*g]; % In R frame
Fa_RF = simplify(RtoB*Fa_RF); % Altered to B frame
Fa_RF = Fa_RF(t);

Fa_FF = [0;0;m_FF*g]; % In R frame
Fa_FF = simplify((BtoC*RtoB)*Fa_FF); % Altered to C frame
Fa_FF = Fa_FF(t);

Fa_FW = [0;0;m_FW*g]; % In R frame
Fa_FW = simplify((BtoC*RtoB)*Fa_FW); % Altered to C frame
Fa_FW = Fa_FW(t);

% ---------------- Torques ---------------- %

syms T_R T_RW T_S % Variables for torque
T4 = [T_R;0;0]; % In L frame
T6 = [0;T_RW;0]; % In B/A frame
T7 = [0;0;T_S]; % In B/C frame
Ta_RW = T6; % In A frame
Ta_RF = LtoB*T4 - T6 - T7; % In B frame
Ta_RF = Ta_RF(t);
Ta_FF = T7; % In C frame
Ta_FW = [0;0;0];

SVars_old_q = [Q3,Q4,Q5,Q6,Q7,Q8];
SVars_old_q = SVars_old_q(t);
SVars_new_q = [q3,q4,q5,q6,q7,q8];

Fa_RW_q = subs(Fa_RW,SVars_old_q,SVars_new_q);
Fa_RF_q = subs(Fa_RF,SVars_old_q,SVars_new_q);
Fa_FF_q = subs(Fa_FF,SVars_old_q,SVars_new_q);
Fa_FW_q = subs(Fa_FW,SVars_old_q,SVars_new_q);

Ta_RW_q = Ta_RW;
Ta_RF_q = subs(Ta_RF,SVars_old_q,SVars_new_q);
Ta_FF_q = Ta_FF;
Ta_FW_q = Ta_FW;

Section 13: Save Option

%}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Section Description %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
This section provides the option to save the symbolic variables developed in the program to a .mat file. This file is used in the simulation routine. This is done to provide the workspace with only the necessary variables.
Section 14: Geometry

%{
    %------------------- Section Description -------------------
    %
    % This section is primarily used for internal verifications. The expression
developed in this script are rather large and complex. So numerical
evaluation allows for better examination.

    % This section is peripheral to the equations of motion.
}

%{
%    %----- Common Vector Formats ------ %
%
% -- Geometry Vector
r_RW = Geo(1);
r_FW = Geo(2);
x_B = Geo(3);
z_B = Geo(4);
x_S = Geo(5);
z_S = Geo(6);
x_C = Geo(7);
z_C = Geo(8);
x_D = Geo(9);
z_D = Geo(10);

% -- Input Vector
q3 = x(1);
q4 = x(2);
q5 = x(3);
q6 = x(4);
q7 = x(5);
q8 = x(6);
%

R2D = 180/pi;
D2R = pi/180;
% - Geometric parameters from the MeiJaard benchmark bicycle to be used in
% model validation

112

% - Calculating the length of my position vector components, based on the
% geometry of the MeiJaard bicycle

x_S_n = cos(lambda_n)*2*(c + w - r_RW_n*tan(lambda_n));
z_S_n = tan(lambda_n)*x_S_n;

x_D_n = -cos(lambda_n)*(c - r_FW_n*tan(lambda_n));
z_D_n = (r_RW_n + z_S_n - r_FW_n + x_D_n*sin(lambda_n))/cos(lambda_n);

x_C_n = (x_H - x_S_n)*cos(lambda_n) - ...
        (r_RW_n + z_S_n - z_H)*sin(lambda_n);
z_C_n = (x_H - x_S_n - x_C_n*cos(lambda_n))/sin(lambda_n);

x_B_n = 0.3;
z_B_n = 0.6;

% Geo = [r_RW_n r_FW_n x_B_n z_B_n x_S_n z_S_n ...]
%       x_C_n z_C_n x_D_n z_D_n lambda_n];

Geo = [0.3 0.35 0.3 0.6 0.8 0.5 0.059767247746024 0.128171276411158...
        0.052823312076349 0.498106110885306 20*D2R];

Input = [30*D2R 20*D2R 5*D2R 0 10*D2R 0];
**SymCrossMatrix.m**

```matlab
function [matrix] = SymCrossMatrix(vector,t)

checksize = length(vector);

if checksize == 1
    V = vector(t);
else
    V = vector;
end

A = V(1);
B = V(2);
C = V(3);

matrix = [ 0   -C   B;   
            C 0   -A;   
           -B   A  0 ];
end
```

**SymVector2Matrix.m**

```matlab
function [output_matrix] = SymVector2Matrix(vector,variables)

A_matrix = vpa(zeros(length(variables)));

for i = 1:length(vector)
    for j = 1:length(variables)
        A_matrix(i,j) = diff(vector(i,:),variables(j));
    end
end

output_matrix = A_matrix;
```
APPENDIX H: Simulation Main File and Function Files

Simulation Main File

Bike Simulation Main File - Version 4

% Western Michigan University
% Spring 2017
% Code by Brent Kostich

clear
load EOM_Syms % loading in the syms called in this script

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Global Variables %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
These global variables consist of the three symbolic variables and the
geometric and mass properties that the EOM function files needs to run.
As they are constant, it is safe to define them as global instead of
explicitly passing them into the function file.
%

global u4DT u6DT u7DT
global r_RW_n r_FW_n x_B_n z_B_n x_S_n z_S_n
global x_C_n z_C_n x_D_n z_D_n lambda_n
global m_RW_n m_RF_n m_FP_n m_FW_n
global I_RW_matrix I_RF_matrix I_PP_matrix I_FW_matrix grav

null_vec = [u4DT,u6DT,u7DT]; % in place to satisfy MATLAB's code analyzer

Section 1: Bicycle Physical Parameters

{%

The section defines the geometry and mass properties of the bicycle.

As this current script file is designed to validate the model
correctness, the parameters shown are from the benchmark bicycle defined
by MeiJaard et. al. (2007).

These parameters are designated for a linear model, so they are converted
to suit the nonlinear model used here. Furthermore, the inertia matrices
are adjusted (when necessary) to match the coordinate systems used in the
current model
%
}
R2D = 180/pi; \quad % \text{Radians to degrees}
D2R = pi/180; \quad % \text{Degrees to radians}
grav = 9.81; \quad % \text{Gravity [m/s^2]}

% ------------------- Geometric Parameters of the Bicycle ------------------- %

% -- Parameters from the Meijaard bicycle
\text{c} = 0.08; \quad % \text{trail}
w = 1.02; \quad % \text{wheel base}
\text{r}_{\text{RW}_n} = 0.3; \quad % \text{radius of rear wheel}
\text{r}_{\text{FW}_n} = 0.35; \quad % \text{radius of front wheel}
\lambda_n = \pi/10; \quad % \text{steer axis tilt (\pi/2 - head angle, in radians)}
x_H = 0.9; \quad % \text{x-position of front frame C.G.}
z_H = 0.7; \quad % \text{z-position of front frame C.G.}

% -- Conversion to the parameters current bicycle model
\text{x}_{\text{S}_n} = \cos(\lambda_n)2(\text{c} + \text{w} - \text{r}_{\text{RW}_n}\tan(\lambda_n));
\text{z}_{\text{S}_n} = \tan(\lambda_n)\text{x}_{\text{S}_n};

\text{x}_{\text{D}_n} = -\cos(\lambda_n)\text{(c} - \text{r}_{\text{FW}_n}\tan(\lambda_n));
\text{z}_{\text{D}_n} = (\text{r}_{\text{RW}_n} + \text{z}_{\text{S}_n} - \text{r}_{\text{FW}_n} + \text{x}_{\text{D}_n}\sin(\lambda_n))/\cos(\lambda_n);

\text{x}_{\text{C}_n} = (\text{x}_H - \text{x}_{\text{S}_n})\cos(\lambda_n) - (\text{r}_{\text{RW}_n} + \text{z}_{\text{S}_n} - \text{z}_H)\sin(\lambda_n);
\text{z}_{\text{C}_n} = (\text{x}_H - \text{x}_{\text{S}_n} - \text{x}_{\text{C}_n}\cos(\lambda_n))/\sin(\lambda_n);

\text{x}_{\text{B}_n} = 0.3;
\text{z}_{\text{B}_n} = 0.6;

% ------------------- Mass Properties of Bicycle ------------------- %

% -- Parameters from the Meijaard bicycle

% - Rear Wheel
\text{m}_{\text{RW}_n} = 2;
\text{I}_{\text{RWxx}_n} = 0.0603;
\text{I}_{\text{RWyy}_n} = 0.12;

% - Rear Frame
\text{m}_{\text{RF}_n} = 85;
\text{I}_{\text{RFxx}_n} = 9.2;
\text{I}_{\text{RFyy}_n} = 11;
\text{I}_{\text{RFzz}_n} = 2.8;
\text{I}_{\text{RFxz}_n} = -2.4;

% - Front Frame
\text{m}_{\text{FF}_n} = 4;
\text{I}_{\text{FFxx}_n} = 0.05892;
\text{I}_{\text{FFyy}_n} = 0.06;
\text{I}_{\text{FFzz}_n} = 0.00708;
\text{I}_{\text{FFxz}_n} = 0.00756;

% - Front Wheel
\text{m}_{\text{FW}_n} = 3;
\text{I}_{\text{FWxx}_n} = 0.1405;
\text{I}_{\text{FWyy}_n} = 0.28;

% ------------------- Inertia Tensors ------------------- %
Section 2: Simulation Initial Conditions

%!}
%--------------------------- Section Description ---------------------------
The current simulation is designed to evaluate the equations of motion as an initial value problem. This section defines these initial values.

The generalized coordinates and the three independent generalized speeds are the nine parameters that define the system, and therefore need initial values.

Of the six generalized cooordinatea, only five are independent. The holonomic constraint on the front wheel contact point creates a relationship between the roll angle, pitch angle, and steering angle such that only two are independent. In this model, the pitch angle is defined as the dependent coordinate.

The value of the pitch angle is determined using the constraint equation from 'Bicycle_EOM_v3.'. Mathematically, there are two configurations the bicycle can be in and still satisfy the constraint equations. It is up to the user to pick the correct initial value of the pitch angle.

The forward speed is based on the rotational rate of the rear wheel, and
thus it is used to define U6.

All other generalized coordinates and generalized speeds can be set independently.
%

% - Forward speed of bicycle
Speed = 4.6; % [m/s]

% - Initial values for generalized coordinates and speeds
Q3 = 0; % yaw angle
Q4 = 0; % roll angle
Q5 = 0; % rear wheel angle
Q7 = 0; % steering angle
Q8 = 0; % front wheel angle
U4 = 0.5; % lean rate
U6 = -Speed/r_RW_n; % rear wheel angular velocity
U7 = 0; % steering rate

% - Torques
T_R_n = 0; % Roll torque
T_RW_n = 0; % Rear Wheel torque
T_S_n = 0; % Steering torque

% ---- Solving for pitch angle initial condition ---- %
%
Here the program uses the initial condition of the roll angle and steering angle to determine what the initial condition of the pitch angle should be.

The user should check the value of PIC and choose the option that properly fits the bicycle configuration.

For the validation case, PIC was evaluated and the two solution were 0 deg and 174 deg. The correct initial condition is 0 deg.
%

SymVars = [r_RW ,z_FW, x_B, z_B, x_S, z_S, x_C, z_C, x_D, z_D, lambda];

NumVars = [r_RW_n ,r_FW_n, x_B_n, z_B_n, x_S_n, z_S_n, x_C_n, z_C_n, x_D_n, z_D_n, lambda_n];

R_QP = subs(R_QP,Z,SymVars,NumVars);

PIC = subs(R_QP,[q4,q7],[04,07]);
PIC = vpa(solve(PIC==0,q5),5);
Q5 = 0; % pitch angle initial condition

Section 3: Simulation Routine
%

-------------------------- Section Description --------------------------
This section numerically evaluates the equations of motion, based on the
previously defined initial conditions. It uses a Runge-Kutta 4th order routine.

The user sets the time step and the duration of the simulation time.

The simulation routine has a stop condition if the bicycle falls over. This happens when the roll angle reach plus or minus 90 degrees.

The output is a matrix of the time-histories of the nine bicycle states. Each column is a single state, and the rows correspond with the time step and total simulation duration.

```matlab
% Simulation Parameters

del_time = 0.01; % time step [s]
duration = 5; % duration of simulation [s]

% Runge-Kutta 4th order routine

x0 = [Q3;Q4;Q5;Q6;Q7;Q8;U4;U6;U7]; % initial conditions vector
tspan = 0:del_time:duration; % timespan [s]
yout = zeros(length(tspan),length(x0)); % initializing output vector
early_end = 'N'; % condition for plotting

for i = 1:length(tspan)
    if i == 1
        y = x0;
        disp(y)
        yout(i,:) = y;
    else
        t_half = tspan(i) + del_time/2;
        t_plus = tspan(i) + del_time;
        k1 = del_time*Bike_EQM_FUNv4(tspan(i),y);
        k2 = del_time*Bike_EQM_FUNv4(t_half,y + 1/2*k1);
        k3 = del_time*Bike_EQM_FUNv4(t_half,y + 1/2*k2);
        k4 = del_time*Bike_EQM_FUNv4(t_plus,y + k3);
        y = y + 1/6*k1 + 1/3*k2 + 1/3*k3 + 1/6*k4;
        disp(y)
        yout(i,:) = y;
    end
    if y(2) > pi/2
        disp('Bicycle fell over')
        early_end = 'Y';
        break
    elseif y(2) < -pi/2
        disp('Bicycle fell over')
        early_end = 'Y';
        break
```
Section 4: Monitoring Front Contact Point Displacement

{%
  \------------------- Section Description \-------------------
  This section calculates the displacement of the front wheel contact point by evaluating its holonomic constraint equation.

  This helps verify that the constraint is being enforced, and reveals the amount of noise that the RK4 method is generating.
%
Z_disp = zeros(1,length(yout));
for i = 1:length(yout)
    Z_disp(i) = subs(R_QF,[q4 q5 q7 lambda],...
                        [yout(i,2) yout(i,3) yout(i,5) lambda_n]);
end

Section 5: Output Plots

{%
  \------------------- Section Description \-------------------
  This section is used for visualization of the simulation results.

  Each of the states is individually plotted, except for the rear wheel and front wheel rotation angles, as these merely increase over time.

  Two additional plots are included: (1) the front wheel contact point displacement, and (2) the Meijaard validation plot
%
Yaw_ang   = yout(:,1);
Roll_ang  = yout(:,2);
Pitch_ang = yout(:,3);
RW_ang    = yout(:,4);
Steer_ang = yout(:,5);
FW_ang    = yout(:,6);
Roll_r    = yout(:,7);
RW_r      = yout(:,8);
Steer_r   = yout(:,9);
```matlab
figure (1)
plot(tspan(1:end_point),Z_disp(1:end_point))
title('Displacement of Q with respect to P, in the Z-direction')
xlabel('Time [sec]'),ylabel('Displacement [m]')

figure (2)
plot(tspan(1:end_point),Yaw_ang(1:end_point)*360/2)
title('Yaw Angle of Bicycle')
xlabel('Time [sec]'),ylabel('Angle [deg]')

figure (3)
plot(tspan(1:end_point),Roll_ang(1:end_point)*360/2)
title('Roll Angle of Bicycle')
xlabel('Time [sec]'),ylabel('Angle [deg]')

figure (4)
plot(tspan(1:end_point),Pitch_ang(1:end_point)*360/2)
title('Pitch Angle of Bicycle')
xlabel('Time [sec]'),ylabel('Angle [deg]')

figure (5)
plot(tspan(1:end_point),Steer_ang(1:end_point)*360/2)
title('Steering Angle')
xlabel('Time [sec]'),ylabel('Angle [deg]')

figure (6)
plot(tspan(1:end_point),Roll_r(1:end_point))
title('Roll Rate')
xlabel('Time [sec]'),ylabel('Angular Velocity [rad/s]')

figure (7)
plot(tspan(1:end_point),RW_r(1:end_point)*(-r_RW_n))
title('Velocity of Bicycle')
xlabel('Time [sec]'),ylabel('Velocity [m/s]')

figure (8)
plot(tspan(1:end_point),Steer_r(1:end_point))
title('Steering Rate')
xlabel('Time [sec]'),ylabel('Angular Velocity [rad/s]')

figure (9)
yyaxis left
plot(tspan(1:end_point),...
    [Roll_r(1:end_point) Steer_r(1:end_point)])
ylabel('Angular Rate [rad/s]'),xlabel('Time [sec]')

axL = gca;
axL.YLim = [-0.5 0.9];
```
Section 6: Save Option

{%

================================== Section Description ===============================
The section offers the option to save the output matrix of a simulation case to a dedicated .mat file.

It is important to check the filename before saving so no previously saved data is accidentally overwritten.
%

% Warning: **** CHECK FILENAME BEFORE RUNNING THIS SECTION ****

prompt = 'Do you want to save this file? [Y/N]: ';
save_case = input(prompt,'s');

if save_case == 'Y'

    filename = 'Bike_check_v3_7';

    save(filename,'tspan','yout','end_point','r_RW_n','z_disp',... 'Speed','x0')

    disp([filename,' has been saved.'])
else

    disp('File has NOT been saved.')
end

%}
function [x,dt] = Bike_EOM_FUNv4(-x)

% -- global variables for constant parameters --
global u4DT u6DT u7DT

global r_RW_n r_FW_n x_B_n z_B_n x_S_n z_S_n

global x_C_n z_C_n x_D_n z_D_n lambda_n

global m_RW_n m_RF_n m_FF_n m_FW_n

global I_RW_matrix I_RF_matrix I_FF_matrix I_FW_matrix grav

% -- Defining the input vector --
Q3 = x(1);
Q4 = x(2);
Q5 = x(3);
Q6 = x(4);
Q7 = x(5);
Q8 = x(6);

U4 = x(7);
U6 = x(8);
U7 = x(9);

% -- Creating the geometry vector --
geo = [r_RW_n r_FW_n x_B_n z_B_n x_S_n z_S_n
      x_C_n z_C_n x_D_n z_D_n lambda_n];

m_RW = m_RW_n;
m_RF = m_RF_n;
m_FF = m_FF_n;
m_FW = m_FW_n;

I_RW = I_RW_matrix;
I_RF = I_RF_matrix;
I_FF = I_FF_matrix;
I_FW = I_FW_matrix;

% =============== Calculating Dependent Generalized Speeds ==============

Input_x = [Q3 Q4 Q5 Q6 Q7 Q8];

[C1,C2] = C_Matrices(geo,Input_x);

J = -C2\C1;

U3 = J(1,:)*[U4;U6;U7];
U5 = J(2,:)*[U4;U6;U7];
U8 = J(3,:)*[U4;U6;U7];

% =============== Calculating Derivs of Dependent Generalized Speeds ==

[C1dt,C2dt] = Derivs_C_Matrices(geo,Input_x,J,[U4;U6;U7]);

Uds = -C2\(C1dt*[U4;U6;U7] + C2dt*[U3;U5;U8]) + J*[u4DT;u6DT;u7DT];

U3dt = Uds(1,:);
U5dt = Uds(2,:);
U8dt = Uds(3,1);

% ------------------------ Partial Velocities ------------------------ %

[FWa,FWb,FWc,FWd,FVa,FVb,FVc,FVd] = PartialVelocities(Geo,Input_x,J);

% Body A
FWa_u4 = FWa(:,1); % partial angular velocity, w.r.t u4
FWa_u6 = FWa(:,2); % partial angular velocity, w.r.t u6
FWa_u7 = FWa(:,3); % partial angular velocity, w.r.t u7
FVa_u4 = FVa(:,1); % partial velocity of mass-center, w.r.t u4
FVa_u6 = FVa(:,2); % partial velocity of mass-center, w.r.t u6
FVa_u7 = FVa(:,3); % partial velocity of mass-center, w.r.t u7

% Body B
FWb_u4 = FWb(:,1);  
FWb_u6 = FWb(:,2);  
FWb_u7 = FWb(:,3);  
FVb_u4 = FVb(:,1);  
FVb_u6 = FVb(:,2);  
FVb_u7 = FVb(:,3);  

% Body C
FWc_u4 = FWc(:,1);  
FWc_u6 = FWc(:,2);  
FWc_u7 = FWc(:,3);  
FVc_u4 = FVc(:,1);  
FVc_u6 = FVc(:,2);  
FVc_u7 = FVc(:,3);  

% Body D
FWd_u4 = FWd(:,1);  
FWd_u6 = FWd(:,2);  
FWd_u7 = FWd(:,3);  
FVd_u4 = FVd(:,1);  
FVd_u6 = FVd(:,2);  
FVd_u7 = FVd(:,3);  

% --------------------- Calculating Components of Inertia Forces for Each Body --------------------- %

% - Mass times acceleration

[AA_i,AA_d,AB_i,AB_d,AC_i,AC_d,AD_i,AD_d] = ...
  CalcAccelerations(Geo,Input_x,J,[U4;U6;U7],[U3dt;U5dt;U8dt]);

% Body A
MA_RWi = m_RW*AA_i; % coeffs. of indep. gen. speeds
MA_RWd = m_RW*AA_d; % all remaining terms
MA_RW = MA_RWi*[u4DT;u6DT;u7DT] + MA_RWd; % combined terms

% Body B
MA_RFi = m_RF*AB_i;
MA\_RFd = m\_RF*Ab\_d;

MA\_RF = MA\_RFi*[u4\_DT;u6\_DT;u7\_DT] + MA\_RFd;

% Body C
MA\_FFi = m\_FF*Ac\_i;
MA\_FFd = m\_FF*Ac\_d;

MA\_FF = MA\_FFi*[u4\_DT;u6\_DT;u7\_DT] + MA\_FFd;

% Body D
MA\_FWi = m\_FW*Ad\_i;
MA\_FWd = m\_FW*Ad\_d;

MA\_FW = MA\_FWi*[u4\_DT;u6\_DT;u7\_DT] + MA\_FWd;

% - Inertia matrix dotted with angular acceleration

[AL\_i,AL\_a\_d,AL\_b\_i,AL\_b\_d,AL\_c\_i,AL\_c\_d,AL\_d\_i,AL\_d\_d] = ...
CalcAlphas(Geo,Input_x,J,[U4;U6;U7],[U3\_dt;U5\_dt;U8\_dt]);

% Body A
IA\_RW1 = I\_RW*AL\_a\_i; % coeffs. of indep. gen. speeds
IA\_RWd = I\_RW*AL\_a\_d; % all remaining terms

IA\_RW = IA\_RW1*[u4\_DT;u6\_DT;u7\_DT] + IA\_RWd; % combined terms

% Body B
IA\_RFi = I\_RF*AL\_b\_i;
IA\_RFd = I\_RF*AL\_b\_d;

IA\_RF = IA\_RFi*[u4\_DT;u6\_DT;u7\_DT] + IA\_RFd;

% Body C
IA\_FFi = I\_FF*AL\_c\_i;
IA\_FFd = I\_FF*AL\_c\_d;

IA\_FF = IA\_FFi*[u4\_DT;u6\_DT;u7\_DT] + IA\_FFd;

% Body D
IA\_FWi = I\_FW*AL\_d\_i;
IA\_FWd = I\_FW*AL\_d\_d;

IA\_FW = IA\_FWi*[u4\_DT;u6\_DT;u7\_DT] + IA\_FWd;

% - Angular velocity crossed with angular momentum

% Body A
omega\_RW = P\_W\_a*[U4;U6;U7];
omega\_RW\_tilde = SymCrossMatrix(omega\_RW);

WH\_RW = omega\_RW\_tilde*(I\_RW*omega\_RW);

% Body B
omega\_RF = P\_W\_b*[U4;U6;U7];
omega\_RF\_tilde = SymCrossMatrix(omega\_RF);
\[
WH_{RF} = \omega_{RF \, \tilde{t}}(I_{RF} \cdot \omega_{RF});
\]

\%
Body C
\[
\omega_{FF} = PW_4[U_4; U_6; U_7];
\]
\[
\omega_{FF \, \tilde{t}} = \text{SymCrossMatrix}(\omega_{FF});
\]
\[
WH_{FF} = \omega_{FF \, \tilde{t}}(I_{FF} \cdot \omega_{FF});
\]

\%
Body D
\[
\omega_{FW} = PWd[U_4; U_6; U_7];
\]
\[
\omega_{FW \, \tilde{t}} = \text{SymCrossMatrix}(\omega_{FW});
\]
\[
WH_{FW} = \omega_{FW \, \tilde{t}}(I_{FW} \cdot \omega_{FW});
\]

\%
------------------------------- Build-up of Inertia Forces -------------------------------

\%
- U4
\[
F_i_{P u4} = MA_{RW}.'*PV_{a u4} + MA_{RF}.'*PV_{b u4} + ...
\]
\[
MA_{FF}.'*PV_{c u4} + MA_{FW}.'*PV_{d u4};
\]
\[
F_i_{M u4} = (IA_{RW} + WH_{RW}).'*PW_{a u4} + ...
\]
\[
(IA_{RF} + WH_{RF}).'*PW_{b u4} + ...
\]
\[
(IA_{FF} + WH_{FF}).'*PW_{c u4} + ...
\]
\[
(IA_{FW} + WH_{FW}).'*PW_{d u4};
\]
\[
F_{i u4} = \text{vpa}(F_i_{P u4} + F_i_{M u4}); \quad \% \text{Total inertia forces w.r.t u4}
\]

\%
- U6
\[
F_i_{P u6} = MA_{RW}.'*PV_{a u6} + MA_{RF}.'*PV_{b u6} + ...
\]
\[
MA_{FF}.'*PV_{c u6} + MA_{FW}.'*PV_{d u6};
\]
\[
F_i_{M u6} = (IA_{RW} + WH_{RW}).'*PW_{a u6} + ...
\]
\[
(IA_{RF} + WH_{RF}).'*PW_{b u6} + ...
\]
\[
(IA_{FF} + WH_{FF}).'*PW_{c u6} + ...
\]
\[
(IA_{FW} + WH_{FW}).'*PW_{d u6};
\]
\[
F_{i u6} = \text{vpa}(F_i_{P u6} + F_i_{M u6}); \quad \% \text{Total inertia forces w.r.t u6}
\]

\%
- U7
\[
F_i_{P u7} = MA_{RW}.'*PV_{a u7} + MA_{RF}.'*PV_{b u7} + ...
\]
\[
MA_{FF}.'*PV_{c u7} + MA_{FW}.'*PV_{d u7};
\]
\[
F_i_{M u7} = (IA_{RW} + WH_{RW}).'*PW_{a u7} + ...
\]
\[
(IA_{RF} + WH_{RF}).'*PW_{b u7} + ...
\]
\[
(IA_{FF} + WH_{FF}).'*PW_{c u7} + ...
\]
\[
(IA_{FW} + WH_{FW}).'*PW_{d u7};
\]
\[
F_{i u7} = \text{vpa}(F_i_{P u7} + F_i_{M u7}); \quad \% \text{Total inertia forces w.r.t u7}
\]

\%
------------------------------- Calculating Components of Active Forces -------------------------------

\%
 \{ Props vector values
\]
\[
m_{RW} = \text{props}(1);
\]
\[
m_{RF} = \text{props}(2);
\]
\[
m_{FF} = \text{props}(3);
\]
m_FW = props(4);  
g = props(5);  
lambda = props(6);  
%
Props = [m_RW m_RF m_FF m_FW grav lambda_n];

% Input torques
Torques = [0 0 0];

[Fa_RW,Fa_RF,Fa_FF,Fa_FW,Ta_RW,Ta_RF,Ta_FF,Ta_FW] = ...
    CalcActiveForces(x,Props,Torques);

% ================ Build-up of Active Forces =============== %

% - U4
Fa_P_u4 = Fa_RW.*PV_a_u4 + Fa_RF.*PV_b_u4 + ...%  
                 Fa_FF.*PV_c_u4 + Fa_FW.*PV_d_u4;
Fa_T_u4 = Ta_RW.*PW_a_u4 + Ta_RF.*PW_b_u4 + ...%  
                 Ta_FF.*PW_c_u4 + Ta_FW.*PW_d_u4;
FA_u4 = vpa(Fa_P_u4 + Fa_T_u4);

% - U6
Fa_P_u6 = Fa_RW.*PV_a_u6 + Fa_RF.*PV_b_u6 + ...%  
                 Fa_FF.*PV_c_u6 + Fa_FW.*PV_d_u6;
Fa_T_u6 = Ta_RW.*PW_a_u6 + Ta_RF.*PW_b_u6 + ...%  
                 Ta_FF.*PW_c_u6 + Ta_FW.*PW_d_u6;
FA_u6 = vpa(Fa_P_u6 + Fa_T_u6);

% - U7
Fa_P_u7 = Fa_RW.*PV_a_u7 + Fa_RF.*PV_b_u7 + ...%  
                 Fa_FF.*PV_c_u7 + Fa_FW.*PV_d_u7;
Fa_T_u7 = Ta_RW.*PW_a_u7 + Ta_RF.*PW_b_u7 + ...%  
                 Ta_FF.*PW_c_u7 + Ta_FW.*PW_d_u7;
FA_u7 = vpa(Fa_P_u7 + Fa_T_u7);

% ================ Evaluating Kane's Equations ================ %

% - Kane's Equations
Keq_u4 = PA_u4 - FI_u4;
Keq_u6 = PA_u6 - FI_u6;
Keq_u7 = PA_u7 - FI_u7;

% - Mass matrix
M_matrix = zeros(3);  
M_matrix(1,1) = diff(FI_u4,u4DT);  
M_matrix(1,2) = diff(FI_u4,u6DT);  
M_matrix(1,3) = diff(FI_u4,u7DT);  
M_matrix(2,1) = diff(FI_u6,u4DT);  
M_matrix(2,2) = diff(FI_u6,u6DT);  
M_matrix(2,3) = diff(FI_u6,u7DT);  

126
M_matrix(3,1) = diff(FL_u7,u6DT);
M_matrix(3,2) = diff(FL_u7,u6DT);
M_matrix(3,3) = diff(FL_u7,u7DT);

% - Remaining nonlinear terms
F_matrix  = zeros(3,3);
F_matrix(1) = subs(Req_u4,[u4DT u6DT u7DT],[0 0 0]);
F_matrix(2) = subs(Req_u6,[u4DT u6DT u7DT],[0 0 0]);
F_matrix(3) = subs(Req_u7,[u4DT u6DT u7DT],[0 0 0]);

% -- Output vector -- %
xdot     = zeros(9,1);

% kinematical equations
xdot(1) = U3;
xdot(2) = U4;
xdot(3) = U5;
xdot(4) = U6;
xdot(5) = U7;
xdot(6) = U8;

% dynamical equations
xdot(7:9) = M_matrix\F_matrix;
```matlab
function [C1,C2] = C_Matrices(Geo,x)

% -- Geometry Vector
r_RW = Geo(1);
r_FW = Geo(2);
x_B = Geo(3);
z_B = Geo(4);
x_S = Geo(5);
z_S = Geo(6);
x_C = Geo(7);
z_C = Geo(8);
x_D = Geo(9);
z_D = Geo(10);
lambda = Geo(11);

% -- Input Vector
q3 = x(1);
q4 = x(2);
q5 = x(3);
q6 = x(4);
q7 = x(5);
q8 = x(6);

============= Calculating C1 Matrix ==============

C1_11 = sin(q7)*(r_RW*cos(q5)^2 + z_S*cos(q5) + r_RW*sin(q5)^2 + ...
          x_S*sin(q5)) + z_D*(sin(lambda)*sin(q5)*sin(q7) - ...
          cos(lambda)*cos(q5)*sin(q7)) + (r_FW*(cos(lambda)*cos(q4)*cos(q5) ...
          - cos(q4)*sin(lambda)*sin(q5))*(sin(lambda)*sin(q5)*sin(q7) - ...
          cos(lambda)*cos(q5)*sin(q7)))/((cos(lambda)*cos(q4)*cos(q7)*sin(q5) ...
          - sin(q4)*sin(q7) + cos(q4)*cos(q5)*cos(q7)*sin(lambda))^2 + ...
          (cos(lambda)*cos(q4)*cos(q5) - cos(q4)*sin(lambda)*sin(q5))^2)^1/2);

C1_12 = r_RW*cos(q7)*sin(lambda)*sin(q5) - r_RW*cos(lambda)*cos(q5)*cos(q7);

C1_13 = 0;

C1_21 = cos(q7)*(r_RW*cos(q5)^2 + z_S*cos(q5) + ...
          r_RW*sin(q5)^2 + x_S*sin(q5)) - z_D*(cos(lambda)*cos(q5)*cos(q7) - ...
          cos(q7)*sin(lambda)*sin(q5)) + x_D*(cos(lambda)*sin(q5) + ...
          cos(q5)*sin(lambda) - (r_FW*cos(lambda)*sin(q5) + ...
          cos(q5)*sin(lambda))*(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - ...
          sin(q4)*sin(q7) + ...
          cos(q4)*cos(q5)*cos(q7)*sin(lambda)))/...
          ((cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) + ...
          cos(q4)*cos(q5)*cos(q7)*sin(lambda))^2 + ...
          (cos(lambda)*cos(q4)*cos(q5) - ...
\[
\cos(q4)\sin(lambda)\sin(q5) + r_{FW}\cos(lambda)\cos(q4)\cos(q5) - ... \\
(r_{FW}\cos(lambda)\cos(q4)\cos(q5) - ... \\
\cos(q4)\sin(lambda)\sin(q5))\cos(lambda)\cos(q5)\cos(q7) - ... \\
\cos(q7)\sin(lambda)\sin(q5)\cos(q7)\sin(lambda)\cos(q5)\cos(q7) - ... \\
\sin(q4)\sin(q7) + \cos(q4)\cos(q5)\cos(q7)\sin(lambda)\sin(q7))\cos(lambda)\cos(q5)\cos(q7) - ... \\
\cos(lambda)\cos(q4)\cos(q5) - \cos(q4)\sin(lambda)\sin(q5))\cos(lambda)\cos(q5)\cos(q7) - ... \\
(1/2);
\]

\[
C1_{22} = r_{RW}\cos(lambda)\cos(q5)\sin(q7) - r_{RW}\sin(lambda)\sin(q5)\sin(q7);
\]

\[
C1_{23} = x_{D} - (r_{FW}\cos(lambda)\cos(q4)\cos(q7)\sin(q5) - ... \\
\sin(q4)\sin(q7) + \cos(q4)\cos(q5)\cos(q7)\sin(lambda))\cos(lambda)\cos(q4)\cos(q7)\sin(lambda)\sin(q5) - ... \\
\cos(lambda)\cos(q4)\cos(q7)\sin(lambda)\sin(q5)\cos(q7)\sin(lambda)\cos(q5)\cos(q7)\sin(lambda)\sin(q7) - ... \\
(1/2);
\]

\[
C1_{31} = (r_{FW}\sin(lambda)\sin(q5)\sin(q7) - ... \\
\cos(lambda)\cos(q5)\sin(q7))\cos(lambda)\cos(q4)\cos(q7)\sin(q5) - ... \\
\sin(q4)\sin(q7) + \cos(q4)\cos(q5)\cos(q7)\sin(lambda))\cos(lambda)\cos(q4)\cos(q7)\sin(lambda)\sin(q5) - ... \\
\cos(lambda)\cos(q4)\cos(q5)\cos(q7)\sin(lambda))\cos(lambda)\cos(q4)\cos(q5)\cos(q7)\sin(lambda)\sin(q7) - ... \\
(r_{FW}\sin(lambda)\sin(q5)\sin(q7) - \cos(q4)\sin(lambda)\sin(q5)\sin(q7))\cos(lambda)\cos(q5)\cos(q7) - ... \\
(1/2);
\]

\[
C1_{32} = -r_{RW}\cos(lambda)\sin(q5) - r_{RW}\cos(q5)\sin(lambda);
\]

\[
C1_{33} = 0;
\]

\% - combined matrix

\[
C1 = [C1_{11} C1_{12} C1_{13}; \\
C1_{21} C1_{22} C1_{23}; \\
C1_{31} C1_{32} C1_{33}];
\]

\(-- Calculating C2 Matrix --

\[
C2_{11} = \sin(q7)\times(x_{S}\cos(q4)\cos(q5) - z_{S}\cos(q4)\sin(q5)) + ... \\
z_{D}\times(\cos(q7)\sin(q4) + \cos(lambda)\cos(q4)\sin(q5)\sin(q7) + ... \\
\cos(lambda)\cos(q7)\times(z_{S}\sin(q4) + r_{RW}\cos(q5)\sin(q4)) + ... \\
\cos(q7)\sin(lambda)\times(x_{S}\sin(q4) + r_{RW}\sin(q4)\sin(q5)) + ... \\
(r_{FW}\times(\cos(lambda)\cos(q4)\cos(q5) - ... \\
\cos(lambda)\cos(q4)\sin(lambda)\sin(q5))\times(\cos(q7)\sin(q4) + ... \\
\cos(lambda)\cos(q4)\sin(q5)\sin(lambda)\sin(q7)) + ... \\
\cos(lambda)\cos(q4)\cos(q5)\cos(q7)\sin(lambda))\cos(lambda)\cos(q5)\cos(q7) - ... \\
(1/2);
\]

\[
C2_{12} = z_{D}\cos(q7) - \cos(lambda)\cos(q7)\times(z_{S} + r_{RW}\cos(q5)) + ... \\
\cos(q7)\sin(lambda)\times(x_{S} + r_{RW}\sin(q5)) + ... \\
(r_{FW}\times(\cos(lambda)\cos(q4)\cos(q5) - ... \\
\cos(lambda)\cos(q4)\sin(lambda)\sin(q5))\times(\cos(lambda)\cos(q4)\cos(q7)\sin(lambda)\sin(q5) - ... \\
\sin(q4)\sin(q7) + \cos(q4)\cos(q5)\cos(q7)\sin(lambda))\cos(lambda)\cos(q5)\cos(q7) - ... \\
(1/2);
\]

\[
C2_{13} = (r_{FW}\times(\cos(lambda)\cos(q4)\cos(q5) - ... \\
\cos(lambda)\cos(q4)\sin(lambda)\sin(q5))\times(\cos(lambda)\cos(q4)\cos(q7) - ... \\
(1/2));
\]
\[
\cos(q_4)\sin(\lambda)\sin(q_5))}/(\cos(\lambda)\cos(q_4)\cos(q_7)\sin(q_5) - ...
\sin(q_4)\sin(q_7) + \cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda))^{2} + ...
\cos(\lambda)\cos(q_4)\cos(q_5) - \cos(q_4)\sin(\lambda)\sin(q_5))^{2}^{(1/2)};
\]

\[
C_{21} = x_{D}^{}(\cos(\lambda)\cos(q_4)\cos(q_5) - \cos(q_4)\sin(\lambda)\sin(q_5)) + ...
\cos(q_7)\sin(q_7) + \cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7) + ...
\cos(\lambda)\cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7) + ...
\cos(\lambda)\cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7))^{2}^{(1/2)};
\]

\[
C_{22} = \cos(\lambda)\sin(q_7)\sin(q_7)\sin(q_7) + \cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7))^{2}^{(1/2)};
\]

\[
C_{23} = 0;
\]

\[
C_{24} = (x_{RW}^{}(\cos(\lambda)\cos(q_4)\cos(q_5) - \cos(q_4)\sin(\lambda)\sin(q_5)) + ...
\cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7)) + ...
\cos(\lambda)\cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7))^{2}^{(1/2)};
\]

\[
C_{31} = (x_{S}^{} + r_{RW}^{}\cos(q_5)) - \cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7))^{2}^{(1/2)};
\]

\[
C_{32} = (x_{RW}^{}(\cos(\lambda)\cos(q_4)\cos(q_5) - \cos(q_4)\sin(\lambda)\sin(q_5)) + ...
\cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7))^{2}^{(1/2)};
\]

\[
C_{33} = (r_{FW}^{}\cos(q_7))^{2}^{(1/2)};
\]

\[
C_{34} = (x_{S}^{} + r_{RW}^{}\cos(q_5)) - \cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)\sin(q_7))^{2}^{(1/2)};
\]

\[
\% - Combined Matrix
\]

\[
C_{2} = [C_{21} \ C_{22} \ C_{23} \ C_{24};
C_{21} \ C_{22} \ C_{23} \ C_{24};
C_{31} \ C_{32} \ C_{33} \ C_{34}];
\]
Derivs_C_Matrices.m

```matlab
function [C1dt,C2dt] = Derivs_C_Matrices(Geo,x,J,U1)

% -- Geometry Vector
r_RW = Geo(1);
r_FW = Geo(2);
x_B = Geo(3);
z_B = Geo(4);
x_S = Geo(5);
z_S = Geo(6);
x_C = Geo(7);
z_C = Geo(8);
x_D = Geo(9);
z_D = Geo(10);
lambda = Geo(11);

% -- Input Vector
q3 = x(1);
q4 = x(2);
q5 = x(3);
q6 = x(4);
q7 = x(5);
q8 = x(6);

% -- Independent Generalized Speeds
u4 = U1(1);
U6 = U1(2);
U7 = U1(3);

% -- Dependent Generalized Speeds
u3 = J{1,:}*U1;
U5 = J{2,:}*U1;
U8 = J{3,:}*U1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

C1dt_11 = z_D*(u5*cos(lambda)*sin(q5)*sin(q7) - ...
    u7*cos(lambda)*cos(q5)*cos(q7) + u5*cos(q5)*sin(lambda)*sin(q7) + ...
    u7*cos(q7)*sin(lambda)*sin(q5) + sin(q7)*(u5*x_S*cos(q5) - ...
    u5*z_S*sin(q5)) + u7*cos(q7)*(r_RW*cos(q5)^2 + z_S*cos(q5) + ...
    r_RW*sin(q5)^2 + x_S*sin(q5)) + (r_FW*cos(lambda)*cos(q4)*cos(q5) - ...
    cos(q4)*sin(lambda)*sin(q5))*(u5*cos(lambda)*sin(q5)*sin(q7) - ...
    u7*cos(lambda)*cos(q5)*cos(q7) + u5*cos(q5)*sin(lambda)*sin(q7) + ...
    u7*cos(q7)*sin(lambda)*sin(q5))/...
    (cos(lambda)*cos(q4)*cos(q7)*sin(q5) - ...
    sin(q4)*sin(q7) + cos(q4)*cos(q5)*cos(q7)*sin(lambda))^2 + ...
```
r_RW * u7 * cos(q7) * sin(lambda) * sin(q5);

Cldt_23 = (r_FW * (u4 * cos(q4) * sin(q7) + u7 * cos(q7) * sin(q4) - ...
  u5 * cos(lambda) * cos(q4) * cos(q7) * cos(q5) * sin(q7) + ...
  u4 * cos(lambda) * cos(q7) * sin(q4) * sin(q5) + ...
  u4 * cos(q5) * cos(q7) * sin(lambda) * sin(q4) + ...
  u5 * cos(q4) * cos(q7) * sin(lambda) * sin(q5) + ...
  u7 * cos(lambda) * cos(q4) * sin(q5) * sin(q7) + ...
  u7 * cos(q4) * cos(q5) * sin(lambda) * sin(q7))) / ...
  (cos(lambda) * cos(q4) * cos(q7) * sin(q5) - sin(q4) * sin(q7) + ...
  cos(q4) * cos(q5) * cos(q7) * sin(lambda)) / 2 + ...
  (cos(lambda) * cos(q4) * cos(q5) - cos(q4) * sin(lambda) * sin(q5)) ^ 2) ^ (1/2) - ...
  (r_FW * (u4 * cos(lambda) * cos(q4) * cos(q7) * sin(q5) - sin(q4) * sin(q7) + ...
  cos(q4) * cos(q5) * cos(q7) * sin(lambda)) * (u4 * cos(q4) * sin(q7) + ...
  u7 * cos(q7) * sin(q4) - u5 * cos(lambda) * cos(q4) * cos(q5) * cos(q7) + ...
  u4 * cos(lambda) * cos(q7) * sin(q4) * sin(q5) + ...
  u4 * cos(q5) * cos(q7) * sin(lambda) * sin(q4) + ...
  u5 * cos(q4) * cos(q7) * sin(lambda) * sin(q5) + ...
  u7 * cos(lambda) * cos(q4) * sin(q5) * sin(q7) + ...
  u7 * cos(q4) * cos(q5) * sin(lambda) * sin(q7)) + ...
  2 * (cos(lambda) * cos(q4) * cos(q5) - cos(q4) * sin(lambda) * sin(q5)) ^ 2 + ...
  cos(q4) * sin(lambda) * sin(q5)) * (u4 * cos(lambda) * cos(q5) * sin(q4) + ...
  u5 * cos(lambda) * cos(q4) * sin(q5) + u5 * cos(q4) * cos(q5) * sin(lambda) - ...
  u4 * sin(lambda) * sin(q4) * sin(q5)) * (cos(lambda) * cos(q4) * cos(q7) * sin(q5) - ...
  sin(q4) * sin(q7) + cos(q4) * cos(q5) * cos(q7) * sin(lambda))) / ...
  (2 * (cos(lambda) * cos(q4) * cos(q7) * sin(q5) - sin(q4) * sin(q7) + ...
  cos(q4) * cos(q5) * cos(q7) * sin(lambda)) / 2 + ...
  (cos(lambda) * cos(q4) * cos(q5) - cos(q4) * sin(lambda) * sin(q5)) ^ 2) ^ (3/2));

Cldt_31 = (r_FW * (cos(lambda) * cos(q4) * cos(q7) * sin(q5) - sin(q4) * sin(q7) + ...
  cos(q4) * cos(q5) * cos(q7) * sin(lambda)) * (u5 * cos(lambda) * sin(q5) * sin(q7) - ...
  u7 * cos(lambda) * cos(q5) * cos(q7) + u5 * cos(q5) * sin(lambda) * sin(q7) + ...
  u7 * cos(q7) * sin(lambda) * sin(q5)) / ...
  (cos(lambda) * cos(q4) * cos(q7) * sin(q5) - sin(q4) * sin(q7) + ...
  cos(q4) * cos(q5) * cos(q7) * sin(lambda)) / 2 + ...
  (cos(lambda) * cos(q4) * cos(q5) - cos(q4) * sin(lambda) * sin(q5)) ^ 2) ^ (1/2) - ...
  x * (u5 * cos(lambda) * sin(q5) * sin(q7) - ...
  u7 * cos(lambda) * cos(q5) * cos(q7) + u5 * cos(q5) * sin(lambda) * sin(q7) + ...
  u7 * cos(q7) * sin(lambda) * sin(q5)) - (r_FW * (sin(lambda) * sin(q5) * sin(q7) - ...
  cos(lambda) * cos(q5) * sin(q7)) * (u4 * cos(q4) * sin(q7) + u7 * cos(q7) * sin(q4) - ...
  u5 * cos(lambda) * cos(q4) * cos(q7) * cos(q5) * sin(q7) + ...
  u4 * cos(lambda) * cos(q7) * sin(q4) * sin(q5) + ...
  u4 * cos(q5) * cos(q7) * sin(lambda) * sin(q4) + ...
  u5 * cos(q4) * cos(q7) * sin(lambda) * sin(q5) + ...
  u7 * cos(lambda) * cos(q4) * sin(q5) * sin(q7) + ...
  u7 * cos(q4) * cos(q5) * sin(lambda) * sin(q7))) / ...
  (cos(lambda) * cos(q4) * cos(q7) * sin(q5) - sin(q4) * sin(q7) + ...
  cos(q4) * cos(q5) * cos(q7) * sin(lambda)) / 2 + ...
  (cos(lambda) * cos(q4) * cos(q5) - cos(q4) * sin(lambda) * sin(q5)) ^ 2) ^ (1/2) + ...
  (r_FW * (2 * (cos(lambda) * cos(q4) * cos(q7) * sin(q5) - sin(q4) * sin(q7) + ...
  cos(q4) * cos(q5) * cos(q7) * sin(lambda)) * (u4 * cos(q4) * sin(q7) + ...
  u7 * cos(q7) * sin(q4) - u5 * cos(lambda) * cos(q4) * cos(q5) * cos(q7) + ...
  u4 * cos(lambda) * cos(q7) * sin(q4) * sin(q5) + ...
  u4 * cos(q5) * cos(q7) * sin(lambda) * sin(q4) + ...
  u5 * cos(q4) * cos(q7) * sin(lambda) * sin(q5) + ...)
```matlab
Cldt_32 = r_RW*u5*sin(lambda)*sin(q5) - r_RW*u5*cos(lambda)*cos(q5); 
Cldt_33 = 0; 

% - combined C1_dt matrix

Cldt = [Cldt_11 Cldt_12 Cldt_13; 
        Cldt_21 Cldt_22 Cldt_23; 
        Cldt_31 Cldt_32 Cldt_33];
```

------------------- Calculating C2_dt Matrix -------------------

```matlab
C2dt_11 = z_D*(u4*cos(q4)*cos(q7)) - u7*sin(q4)*sin(q7) + ... 
          u5*cos(lambda)*cos(q5)*cos(q5)*sin(q7) + ... 
          u7*cos(lambda)*cos(q5)*cos(q7)*sin(q5) + ... 
          u7*cos(q4)*cos(q5)*cos(q7)*sin(lambda) - ... 
          u4*cos(lambda)*sin(q4)*sin(q5)*sin(q7) - ... 
          u4*cos(q5)*sin(lambda)*sin(q4)*cos(q5) - ... 
          u5*cos(lambda)*sin(q5)*sin(q5)*sin(q7) - ... 
          u5*cos(q5)*sin(lambda)*sin(q5)*cos(q5) - ... 
          sin(q7)*(u4*x_S*cos(q5)*sin(q4) + u5*x_S*cos(q4)*sin(q5) - ... 
          u4*z_S*sin(q4)*sin(q5) + 5*x_S*cos(q4)*cos(q5) + ... 
          u7*cos(q7)*(x_S*cos(q4)*cos(q5) - r_RW*u5*cos(q5)*sin(q5) - ... 
          cos(lambda)*cos(q7)*(u4*z_S*cos(q4) - r_RW*u5*cos(q5)*sin(q5) - ... 
          r_RW*u4*cos(q4)*cos(q5) + cos(q7)*sin(lambda)*(u4*x_S*cos(q4) + ... 
          r_RW*u4*cos(q4)*sin(q5) + r_RW*u5*cos(q5)*sin(q4)) + ... 
          u7*cos(lambda)*sin(q7)*(z_S*sin(q4) + r_RW*cos(q5)*sin(q4)) - ... 
          u7*sin(lambda)*sin(q7)*(x_S*sin(q5) + r_RW*sin(q5) + ... 
          r_FW*(cos(q7)*sin(q4) + cos(lambda)*cos(q4)*sin(q5)*sin(q7) + ... 
          cos(q4)*cos(q5)*sin(lambda)*sin(q7)) + (u4*cos(lambda)*cos(q5)*sin(q4)) + ... 
          u5*cos(lambda)*cos(q4)*sin(q5) + u5*cos(q4)*cos(q5)*cos(q5)*sin(lambda) - ... 
          u4*sin(lambda)*sin(q4)*sin(q5); ... 
```

```matlab
((cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7)) + ... 
  cos(q4)*cos(q5)*cos(q7)*sin(lambda)) + (1/2) + ... 
  (r_FW*(cos(lambda)*cos(q4)*cos(q5) - ... 
  cos(q4)*sin(lambda)*sin(q5)) + (u4*cos(q4)*cos(q7) - ... 
  u7*sin(q4)*sin(q7) + u5*cos(lambda)*cos(q4)*cos(q5)*sin(q7) + ... 
  u7*cos(lambda)*cos(q4)*cos(q7)*sin(q5) + ... 
  u7*cos(q4)*cos(q5)*cos(q7)*sin(lambda) - ... 
  u4*cos(lambda)*sin(q4)*sin(q5)*sin(q7) - ... 
  u4*cos(q5)*sin(lambda)*sin(q4)*sin(q7) - ... 
  u5*cos(q4)*sin(lambda)*sin(q5)*sin(q7))/... 
((cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) + ... 
  cos(q4)*cos(q5)*cos(q7)*sin(lambda)) + (1/2) + ... 
  (r_FW*(cos(lambda)*cos(q4)*cos(q5) - ... 
  cos(q4)*sin(lambda)*sin(q5)) + (u4*cos(q4)*cos(q7) - ... 
  u7*sin(q4)*sin(q7) + u5*cos(lambda)*cos(q4)*cos(q5)*sin(q7) + ... 
  u7*cos(lambda)*cos(q4)*cos(q7)*sin(q5) + ... 
  u7*cos(q4)*cos(q5)*cos(q7)*sin(lambda) - ... 
  u4*cos(lambda)*sin(q4)*sin(q5)*sin(q7) - ... 
  u4*cos(q5)*sin(lambda)*sin(q4)*sin(q7) - ... 
  u5*cos(q4)*sin(lambda)*sin(q5)*sin(q7)))/... 
((cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) + ... 
  cos(q4)*cos(q5)*cos(q7)*sin(lambda)) + (1/2) + ... 
  `
\begin{verbatim}
(cos(lambda) * cos(q4) * cos(q5) - ... 
  cos(q4) * sin(lambda) * sin(q5)) / 2 \cdot \cos(1/2) + ...
(r_FW * (2 * cos(lambda) * cos(q4) * cos(q7) * sin(q5) - ...
  sin(q4) * sin(q7) + ... 
  cos(q4) * cos(q5) * cos(q7) * sin(lambda) * (u4 * cos(q4) * sin(q7) + ...
  u7 * cos(q7) * sin(q4) - u5 * cos(lambda) * cos(q4) * cos(q5) * cos(q7) + ...
  u4 * cos(lambda) * cos(q7) * sin(q4) * sin(q5) + ... 
  u4 * cos(q5) * cos(q7) * sin(lambda) * sin(q4) + ... 
  u5 * cos(q4) * cos(q7) * sin(lambda) * sin(q5) + ... 
  u7 * cos(lambda) * cos(q4) * sin(q5) * sin(q7) + ... 
  u7 * cos(q4) * cos(q5) * sin(lambda) * sin(q7) + ... 
  2 * (cos(lambda) * cos(q4) * cos(q5) - ... 
  cos(q4) * sin(lambda) * sin(q5)) * (u4 * cos(lambda) * cos(q5) * sin(q4) + ... 
  u5 * cos(lambda) * cos(q4) * sin(q5) + ... 
  u4 * cos(lambda) * sin(q4) * sin(q5) + ... 
  u4 * sin(lambda) * sin(q4) * sin(q5) + ... 
  u5 * cos(q4) * cos(q5) * sin(lambda) * sin(q7) + ... 
  u7 * cos(lambda) * sin(q4) * sin(q5) + ... 
  u7 * cos(q4) * sin(lambda) * sin(q5) + ... 
  u7 * cos(lambda) * sin(lambda) * sin(q5) + ... 
  u7 * sin(lambda) * sin(q5) + ... 
  u7 * z_D * sin(q7) + ... 
  r_FW * u7 * sin(q7) * (r_S + r_RW * cos(q5)) - u7 * z_D * sin(q7) - ...
  u7 * sin(lambda) * sin(q7) * (x_S + r_RW * sin(q5)) - ...
  (r_FW * cos(q7) * (u4 * cos(lambda) * cos(q5) * sin(q4) + ...
  u5 * cos(lambda) * cos(q4) * sin(q5) + ... 
  u4 * sin(lambda) * sin(q4) * sin(q5) + ... 
  u4 * sin(lambda) * sin(q4) * sin(q5) + ... 
  u5 * cos(q4) * cos(q5) * sin(lambda) * sin(q7) + ... 
  u7 * cos(lambda) * sin(q4) * sin(q5) + ... 
  u7 * cos(q4) * sin(lambda) * sin(q5) + ... 
  (cos(lambda) * cos(q4) * cos(q5) - ... 
  cos(q4) * sin(lambda) * sin(q5)) / 2 \cdot \cos(1/2) - ...
  (r_FW * u7 * sin(q7) * (cos(q4) * cos(q5) - ...
  cos(q4) * sin(lambda) * sin(q5)) / 2 + ... 
  cos(q4) * sin(lambda) * sin(q5)) / 2 \cdot \cos(1/2) + ...
  (r_FW * cos(q7) * (2 * cos(lambda) * cos(q4) * cos(q7) * sin(q5) - ...
  sin(q4) * sin(q7) + ... 
  cos(q4) * cos(q5) * cos(q7) * sin(lambda) * (u4 * cos(q4) * sin(q7) + ...
  u7 * cos(lambda) * cos(q7) * sin(q4) - u5 * cos(lambda) * cos(q4) * cos(q5) * cos(q7) + ...
  u4 * cos(lambda) * cos(q7) * sin(q4) * sin(q5) + ... 
  u4 * cos(q5) * cos(q7) * sin(lambda) * sin(q4) + ... 
  u5 * cos(q4) * cos(q7) * sin(lambda) * sin(q5) + ... 
  u7 * cos(lambda) * cos(q4) * sin(q5) * sin(q7) + ... 
  u7 * cos(q4) * cos(q5) * sin(lambda) * sin(q7) + ... 
  2 * (cos(lambda) * cos(q4) * cos(q5) - ... 
  cos(q4) * sin(lambda) * sin(q5)) * (u4 * cos(lambda) * cos(q5) * sin(q4) + ... 
  u5 * cos(lambda) * cos(q4) * sin(q5) + ... 
  u4 * sin(lambda) * sin(q4) * sin(q5) + ... 
  u4 * sin(lambda) * sin(q4) * sin(q5) + ... 
  u5 * cos(q4) * cos(q5) * sin(lambda) * sin(q7) + ... 
  u7 * cos(lambda) * sin(q4) * sin(q5) + ... 
  u7 * cos(q4) * sin(lambda) * sin(q5) + ... 
  u7 * sin(lambda) * sin(q5) + ... 
  u7 * z_D * sin(q7) + ... 
  r_FW * u5 * cos(lambda) * cos(q7) * sin(q5) + ... 
  r_RW * u5 * cos(q5) * cos(q7) * sin(lambda) + ...
\end{verbatim}
C2dt_13 = \((rFW \cdot 2 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q7) \cdot \sin(q5) - ... \sin(q4) \cdot \sin(q7) + ... \cos(q4) \cdot \cos(q5) \cdot \cos(q7) \cdot \sin(\text{lambda}) \cdot (u4 \cdot \cos(q4) \cdot \sin(q7) + ... \u7 \cdot \cos(q7) \cdot \sin(q4) - u5 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) \cdot \cos(q7) + ... \u4 \cdot \cos(\text{lambda}) \cdot \cos(q7) \cdot \sin(q4) \cdot \sin(q5) + ... \u4 \cdot \cos(q5) \cdot \cos(q7) \cdot \sin(\text{lambda}) \cdot \sin(q4) + ... \u5 \cdot \cos(q4) \cdot \cos(q7) \cdot \sin(\text{lambda}) \cdot \sin(q5) + ... \u7 \cdot \cos(q4) \cdot \cos(q5) \cdot \sin(q4) \cdot \sin(q5) + ... \u7 \cdot \cos(q4) \cdot \cos(q5) \cdot \sin(\text{lambda}) \cdot \sin(q7) + ... 2 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) - ... \cos(q4) \cdot \sin(\text{lambda}) \cdot \sin(q5)) \cdot (u4 \cdot \cos(\text{lambda}) \cdot \cos(q5) \cdot \sin(q4) + ... \u5 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \sin(q5) + u5 \cdot \cos(q4) \cdot \cos(q5) \cdot \sin(\text{lambda}) - ... \u4 \cdot \sin(\text{lambda}) \cdot \sin(q4) \cdot \sin(q5)) \cdot (\cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) - ... \cos(q4) \cdot \sin(\text{lambda}) \cdot \sin(q5)) / ... (2 \cdot (\cos(q4) \cdot \cos(q7) \cdot \sin(q5) - \sin(q4) \cdot \sin(q7) + ... \cos(q4) \cdot \cos(q5) \cdot \cos(q7) \cdot \sin(\text{lambda})) \cdot 2 + ... (\cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) - ... \cos(q4) \cdot \sin(\text{lambda}) \cdot \sin(q5)) \cdot 2^3 \cdot 2^3 - ... (rFW \cdot u4 \cdot \cos(\text{lambda}) \cdot \cos(q5) - \sin(q4) + ... \u5 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \sin(q5) + u5 \cdot \cos(q4) \cdot \cos(q5) \cdot \sin(\text{lambda}) - ... \u4 \cdot \sin(\text{lambda}) \cdot \sin(q4) \cdot \sin(q5)) \cdot (\cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) - ... \cos(q4) \cdot \sin(\text{lambda}) \cdot \sin(q5)) \cdot 2 + ... (\cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) - \cos(q4) \cdot \sin(\text{lambda}) \cdot \sin(q5)) \cdot 2^3 \cdot 2^3 /
\end{equation}

C2dt_21 = \(\cos(\text{lambda}) \cdot \sin(q7) \cdot (u4 \cdot z_S \cdot \cos(q4) - r RW \cdot u5 \cdot \sin(q4) \cdot \sin(q5) + ... \quad rRW \cdot u4 \cdot \cos(q4) \cdot \cos(q5) - z_D \cdot (u4 \cdot \cos(q4) \cdot \sin(q7) + ... \u7 \cdot \cos(q7) \cdot \sin(q4) - u5 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) \cdot \cos(q7) + ... \u4 \cdot \cos(q4) \cdot \cos(q7) \cdot \sin(q4) \cdot \sin(q5) + ... \u4 \cdot \cos(q5) \cdot \cos(q7) \cdot \sin(\text{lambda}) \cdot \sin(q4) + ... \u5 \cdot \cos(q4) \cdot \cos(q7) \cdot \sin(q4) \cdot \sin(q5) + ... \u7 \cdot \cos(q4) \cdot \sin(q5) \cdot \sin(q4) \cdot \sin(\text{lambda}) - ... \quad x_D \cdot (u4 \cdot \cos(\text{lambda}) \cdot \cos(q5) \cdot \sin(q4) + ... \u5 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \sin(q5) + u5 \cdot \cos(q4) \cdot \cos(q5) \cdot \sin(\text{lambda}) - ... \u4 \cdot \sin(\text{lambda}) \cdot \sin(q4) \cdot \sin(q5) - u7 \cdot \sin(q7) \cdot (x_S \cdot \cos(q4) \cdot \cos(q5) - ... z_S \cdot \cos(q4) \cdot \sin(q5)) - \cos(q7) \cdot (u4 \cdot x_S \cdot \cos(q5) \cdot \sin(q4) + ... \u5 \cdot x_S \cdot \cos(q4) \cdot \sin(q5) - u4 \cdot z_S \cdot \sin(q4) \cdot \sin(q5) + ... \u5 \cdot x_S \cdot \cos(q4) \cdot \cos(q5) - \sin(\text{lambda}) \cdot \sin(q7) \cdot (u4 \cdot x_S \cdot \cos(q4) + ... rRW \cdot u4 \cdot \cos(q4) \cdot \sin(q5) + rRW \cdot u5 \cdot \cos(q5) \cdot \sin(q4) + ... \u7 \cdot \cos(\text{lambda}) \cdot \cos(q7) \cdot (z_S \cdot \sin(q4) + rRW \cdot \cos(q5) \cdot \sin(q4)) = ... \u7 \cdot \cos(q7) \cdot \sin(q7) \cdot (x_S \cdot \sin(q4) + rRW \cdot \sin(q5) \cdot \sin(q5)) \end{equation}

C2dt_22 = \((rFW \cdot \sin(q7) \cdot (u4 \cdot \cos(\text{lambda}) \cdot \cos(q5) \cdot \sin(q4) + ... \u5 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \sin(q5) + u5 \cdot \cos(q4) \cdot \cos(q5) \cdot \sin(\text{lambda}) - ... \u4 \cdot \sin(\text{lambda}) \cdot \sin(q4) \cdot \sin(q5)) / ... \quad (\cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q7) \cdot \sin(q5) - \sin(q4) \cdot \sin(q7) + ... \cos(q4) \cdot \cos(q5) \cdot \cos(q7) \cdot \sin(\text{lambda})) \cdot 2 + ... \quad (\cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) - ... \cos(q4) \cdot \sin(\text{lambda}) \cdot \sin(q5)) \cdot 2^3 \cdot 2^3 - ... \quad (rFW \cdot \sin(q7) \cdot (x_S + rRW \cdot \sin(q5)) + u7 \cdot \cos(q7) \cdot (z_S + rRW \cdot \cos(q5)) + ... \u7 \cdot \cos(\text{lambda}) \cdot \cos(q7) \cdot (z_S + rRW \cdot \cos(q5)) = ... \quad \sin(q4) \cdot \sin(q7) + ... \cos(q4) \cdot \cos(q5) \cdot \cos(q7) \cdot \sin(\text{lambda}) \cdot (u4 \cdot \cos(q4) \cdot \sin(q7) + ... \u7 \cdot \cos(q7) \cdot \sin(q4) - u5 \cdot \cos(\text{lambda}) \cdot \cos(q4) \cdot \cos(q5) \cdot \cos(q7) + ... \u4 \cdot \cos(\text{lambda}) \cdot \cos(q7) \cdot \sin(q4) \cdot \sin(q5) + ...\)
u4*cos(q5)*cos(q7)*sin(lambda)*sin(q4) + ... 
138

u5*cos(q4)*cos(q7)*sin(lambda)*sin(q5) + ... 

u7*cos(lambda)*cos(q4)*sin(q5)*sin(q7) + ... 

u7*cos(q4)*cos(q5)*sin(lambda)*sin(q7)) + ... 2*(cos(lambda)*cos(q4)*sin(q5) - ... 

138

cos(q4)*sin(lambda)*sin(q5))*u4*cos(lambda)*cos(q5)*sin(q4) + ... 

u5*cos(lambda)*cos(q4)*sin(q5) + u5*cos(q4)*cos(q5)*sin(lambda) - ... 

u4*sin(lambda)*sin(q4)*sin(q5))*cos(lambda)*cos(q4)*cos(q5) - ... 

cos(q4)*sin(lambda)*sin(q5)) ));/... 

2*(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) + ... 

(cos(q4)*cos(q5)*cos(q7)*sin(lambda))*2 + ... 

(cos(lambda)*cos(q4)*cos(q5) - ... 

cos(q4)*sin(lambda)*sin(q5)))*2*(3/2)) -... 

r_RW*u5*cos(lambda)*sin(q5)*sin(q7) -... 

r_RW*u5*cos(q5)*sin(lambda)*sin(q7) -... 

r_RW*u5*cos(q5)*sin(lambda)*sin(q7) -... 

r_RW*u5*cos(q5)*sin(lambda)*sin(q7) -... 

(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) +... 

(cos(q4)*cos(q5)*cos(q7)*sin(lambda))*2 +... 

(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - cos(q4)*sin(lambda)*sin(q5))^2)^{(1/2)};

C2dt_31 = 0;

C2dt_31 = (r_FW*(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) +... 

(cos(q4)*cos(q5)*cos(q7)*sin(lambda))*u4*cos(q4)*cos(q7) -... 

u7*sin(q4)*sin(q7) + u5*cos(lambda)*cos(q4)*cos(q5)*sin(q7) +... 

u5*cos(lambda)*cos(q4)*cos(q5)*sin(q7) -... 

u7*cos(q4)*cos(q5)*cos(q7)*sin(lambda) -... 

u4*cos(lambda)*sin(q4)*sin(q5)*sin(q7) -... 

u4*cos(q5)*sin(lambda)*sin(q4)*sin(q7) -... 

u5*cos(q4)*sin(lambda)*sin(q5)*sin(q7)))/... 

(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) +... 

(cos(q4)*cos(q5)*cos(q7)*sin(lambda)))*2 +... 

(cos(lambda)*cos(q4)*cos(q5) -... 

cos(q4)*sin(lambda)*sin(q5)))*2*(3/2)) -... 

sin(lambda)*u4*cos(q4) - r_RW*u5*sin(q4)*sin(q5) +... 

r_RW*u4*cos(q4)*cos(q5) - x_D*(u4*cos(q4)*cos(q7) -... 

u7*sin(q4)*sin(q7) + u5*cos(lambda)*cos(q4)*cos(q5)*sin(q7) +... 

u5*cos(lambda)*cos(q4)*cos(q5)*sin(q7) -... 

u7*cos(q4)*cos(q5)*cos(q7)*sin(lambda) -... 

u4*cos(lambda)*sin(q4)*sin(q5)*sin(q7) -... 

u4*cos(q5)*sin(lambda)*sin(q4)*sin(q7) -... 

u5*cos(q4)*sin(lambda)*sin(q5)*sin(q7)) - (r_FW*(cos(q7)*sin(q6) +... 

(cos(lambda)*cos(q4)*sin(q5))*sin(q7) +... 

(cos(q4)*cos(q5)*sin(lambda)*sin(q7))*u4*cos(q4)*sin(q7) +... 

u7*cos(q4)*sin(q4) - u5*cos(lambda)*cos(q4)*cos(q5)*cos(q7) +... 

u4*cos(lambda)*cos(q7)*sin(q4)*sin(q5) +... 

u4*cos(q5)*cos(q7)*sin(lambda)*sin(q4) +... 

u5*cos(q4)*cos(q7)*sin(lambda)*sin(q5) +... 

u7*cos(lambda)*cos(q7)*sin(q4)*sin(q5) +... 

u7*cos(q4)*cos(q5)*sin(lambda)*sin(q7)))/... 

(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) +... 

(cos(q4)*cos(q5)*cos(q7)*sin(lambda)))*2 +... 

(cos(lambda)*cos(q4)*cos(q5) -... 

cos(q4)*sin(lambda)*sin(q5))^2)^{(1/2)};
\[
\begin{align*}
C_{2dt \_32} &= u_7 * \cos(q_7) - u_5 * \cos(\lambda) * \cos(q_4) * \cos(q_5) * \cos(q_7) + \\
& u_4 * \cos(\lambda) * \cos(q_7) * \sin(q_4) * \sin(q_5) + \\
& u_4 * \cos(q_5) * \cos(q_7) * \sin(\lambda) * \sin(q_4) + \\
& u_5 * \cos(q_4) * \cos(q_7) * \sin(\lambda) * \sin(q_5) + \\
& u_7 * \cos(\lambda) * \cos(q_4) * \sin(q_5) * \sin(q_7) + \\
& 2 * (\cos(\lambda) * \cos(q_4) * \cos(q_5) - \\
& \cos(q_4) * \sin(\lambda) * \sin(q_5)) * (u_4 * \cos(\lambda) * \cos(q_5) * \sin(q_4) + \\
& u_5 * \cos(\lambda) * \cos(q_4) * \sin(q_5) + u_5 * \cos(q_4) * \cos(q_5) * \sin(\lambda) - \\
& u_4 * \sin(\lambda) * \sin(q_4) * \sin(q_5) + \cos(\lambda) * \cos(q_4) * \cos(q_7) * \sin(q_5) - \\
& \sin(q_4) * \sin(q_7) + \\
& \cos(q_4) * \cos(q_5) * \sin(q_7)) * \sin(\lambda) * \sin(q_4) + \\
& \cos(\lambda) * \cos(q_4) * \sin(q_5) * \sin(q_7) + \\
& \cos(q_4) * \cos(q_5) * \sin(\lambda) * \sin(q_7)) / ... \\
& (2 * (\cos(\lambda) * \cos(q_4) * \cos(q_7) * \sin(q_5) - \sin(q_4) * \sin(q_7) + \\
& \cos(q_4) * \cos(q_5) * \cos(\lambda) * \sin(q_7)) / ... \\
& \cos(q_4) * \sin(\lambda) * \sin(q_5)) / 2 + ... \\
& \cos(q_4) * \sin(\lambda) * \sin(q_5)) / (1/2) - ... \\
& r_{RW} * u_5 * \cos(\lambda) * \cos(q_5) + ... \\
& (r_{FW} * \cos(q_7) * (2 * (\cos(\lambda) * \cos(q_4) * \cos(q_7) * \sin(q_5) - \\
& \sin(q_4) * \sin(q_7) + ... \\
& \cos(q_4) * \cos(q_5) * \cos(q_7) * \sin(\lambda)) + (u_4 * \cos(q_4) * \sin(q_7) + \\
& u_7 * \cos(\lambda) * \sin(q_4) - u_5 * \cos(\lambda) * \cos(q_4) * \cos(q_5) * \cos(q_7) + ... \\
& u_4 * \cos(\lambda) * \cos(q_7) * \sin(q_4) * \sin(q_5) + \\
& u_4 * \cos(q_5) * \cos(q_7) * \sin(\lambda) * \sin(q_4) + \\
& u_5 * \cos(q_4) * \cos(q_7) * \sin(\lambda) * \sin(q_5) + \\
& u_7 * \cos(\lambda) * \cos(q_4) * \sin(q_5) * \sin(q_7) + \\
& u_7 * \cos(q_4) * \cos(q_5) * \sin(\lambda) * \sin(q_7)) + \\
& 2 * (\cos(\lambda) * \cos(q_4) * \cos(q_5) - \\
& \cos(q_4) * \sin(\lambda) * \sin(q_5)) / (3/2)) - ... \\
& (r_{FW} * u_7 * \sin(q_7) * \cos(\lambda) * \cos(q_4) * \cos(q_7) * \sin(q_5) - ... \\
& \sin(q_4) * \sin(q_7) + \cos(q_4) * \cos(q_5) * \cos(q_7) * \sin(\lambda)) / ... \\
& (\cos(\lambda) * \cos(q_4) * \cos(q_7) * \sin(q_5) - \sin(q_4) * \sin(q_7) + \\
& \cos(q_4) * \cos(q_5) * \cos(q_7) * \sin(\lambda)) / 2 + ... \\
& \cos(\lambda) * \cos(q_4) * \cos(q_5) - \cos(q_4) * \sin(\lambda) * \sin(q_5)) / 2) / (1/2); \\
C_{2dt \_33} &= (r_{FW} * (2 * (\cos(\lambda) * \cos(q_4) * \cos(q_7) * \sin(q_5) - ... \\
& \sin(q_4) * \sin(q_7) + ... 
\end{align*}
\]
\[
\begin{align*}
\cos(q_4)\cos(q_5)\cos(q_7)\sin(\lambda)) & = (u_4\cos(q_4)\sin(q_7) + ... \\
u_7\cos(q_7) & = u_5\cos(\lambda) \cos(q_4) \cos(q_5) \cos(q_7) + ... \\
u_4\cos(\lambda) & = u_5\cos(q_4) \cos(q_5) \cos(q_7) + ... \\
u_4\cos(q_5) & = u_5\cos(q_7) \sin(q_4) \sin(q_5) + ... \\
u_5\cos(q_4) & = u_5\cos(q_7) \cos(q_5) \sin(q_4) + ... \\
u_7\cos(q_4) & = u_5\cos(q_7) \sin(q_5) \sin(q_4) + ... \\
2\cos(\lambda) & = u_5\cos(q_7) \sin(q_4) \sin(q_5) + ... \\
u_5\cos(\lambda) & = u_5\cos(q_7) \cos(q_5) \sin(q_4) \sin(q_5) + ... \\
u_7\cos(q_7) \sin(q_4) \sin(q_5) & = u_5\cos(q_7) \sin(q_4) \cos(q_5) \sin(q_4) \sin(q_5) + ... \\
2\cos(\lambda) & = u_5\cos(q_7) \cos(q_5) \sin(q_4) \sin(q_5) + ... \\
u_7\cos(q_7) \sin(q_4) \sin(q_5) & = u_5\cos(q_7) \cos(q_5) \sin(q_4) \sin(q_5) + ... \\
\end{align*}
\]

\begin{verbatim}
C2dt = [C2dt_11 C2dt_12 C2dt_13; \\
        C2dt_21 C2dt_22 C2dt_23; \\
        C2dt_31 C2dt_32 C2dt_33];
\end{verbatim}
function [PWa,PWB,PWC,PWD,PVA,PVB,PVC,PVD] = PartialVelocities(Geo,x,J)

% -- Geometry Vector
r_RW = Geo(1);
r_PW = Geo(2);
x_B = Geo(3);
z_B = Geo(4);
x_S = Geo(5);
z_S = Geo(6);
x_C = Geo(7);
z_C = Geo(8);
x_D = Geo(9);
z_D = Geo(10);
lambda = Geo(11);

% -- Input Vector
q3 = x(1);
q4 = x(2);
q5 = x(3);
q6 = x(4);
q7 = x(5);
q8 = x(6);

{%
Body A = Rear Wheel
Body B = Rear Frame
Body C = Front Frame
Body D = Front Wheel
%

% --------------------- Partial Angular Velocity of Rear Wheel --------------------- %
W_A_i = zeros(3);
W_A_i(1,1) = cos(q5);
W_A_i(2,2) = 1;
W_A_i(3,1) = sin(q5);
W_A_d = zeros(3);
W_A_d(1,1) = -cos(q4)*sin(q5);
W_A_d(2,1) = sin(q4);
W_A_d(2,2) = 1;
W_A_d(3,1) = cos(q4)*cos(q5);

PW_a = W_A_i + W_A_d*J;

% --------------------- Partial Angular Velocity of Rear Frame --------------------- %
W_B_i = zeros(3);
W_B_i(1,1) = cos(q5);
W_B_i(3,1) = sin(q5);

W_B_d = zeros(3);
W_B_d(1,1) = -cos(q4)*sin(q5);
W_B_d(2,1) = sin(q4);
W_B_d(2,2) = 1;
W_B_d(3,1) = cos(q4)*cos(q5);
FWd = W_B_i + W_B_d*J;

% ----------------- Partial Angular Velocity of Front Frame ----------------- %
W_C_i = zeros(3);
W_C_i(1,1) = cos(lambda)*cos(q5)*cos(q7) - cos(q7)*sin(lambda)*sin(q5);
W_C_i(2,1) = sin(lambda)*sin(q5)*sin(q7) - cos(lambda)*cos(q5)*sin(q7);
W_C_i(3,1) = cos(lambda)*sin(q5) + cos(q5)*sin(lambda);
W_C_i(3,3) = 1;

W_C_d = zeros(3);
W_C_d(1,1) = sin(q4)*sin(q7) - cos(lambda)*cos(q4)*cos(q7)*sin(q5) - ... 
             cos(q4)*cos(q5)*cos(q7)*sin(lambda);
W_C_d(2,1) = sin(q7);
W_C_d(2,2) = cos(q7)*sin(q4) + cos(lambda)*cos(q4)*sin(q5)*sin(q7) + ... 
             cos(q4)*cos(q5)*sin(lambda)*sin(q7);
W_C_d(3,1) = cos(lambda)*cos(q4)*cos(q5) - cos(q4)*sin(lambda)*sin(q5);

FWC = W_C_i + W_C_d*J;

% ----------------- Partial Angular Velocity of Front Frame ----------------- %
W_D_i = zeros(3);
W_D_i(1,1) = cos(lambda)*cos(q5)*cos(q7) - cos(q7)*sin(lambda)*sin(q5);
W_D_i(2,1) = sin(lambda)*sin(q5)*sin(q7) - cos(lambda)*cos(q5)*sin(q7);
W_D_i(3,1) = cos(lambda)*sin(q5) + cos(q5)*sin(lambda);
W_D_i(3,3) = 1;

W_D_d = zeros(3);
W_D_d(1,1) = sin(q4)*sin(q7) - cos(lambda)*cos(q4)*cos(q7)*sin(q5) - ... 
             cos(q4)*cos(q5)*cos(q7)*sin(lambda);
W_D_d(2,1) = sin(q7);
W_D_d(2,2) = cos(q7)*sin(q4) + cos(lambda)*cos(q4)*sin(q5)*sin(q7) + ... 
             cos(q4)*cos(q5)*sin(lambda)*sin(q7);
W_D_d(3,1) = cos(lambda)*cos(q4)*cos(q5) - cos(q4)*sin(lambda)*sin(q5);

FWd = W_D_i + W_D_d*J;

% ----------------- Partial Velocity of Rear Wheel Mass Center ----------------- %
V_A_i = zeros(3);
V_A_i(1,2) = -r_RW*cos(q5);
V_A_i(2,1) = r_RW*cos(q5)^2 + r_RW*sin(q5)^2;
V_A_i(3,2) = -r_RW*sin(q5);

V_A_d = zeros(3);
V_A_d(1,1) = -r_RW*cos(q5)*sin(q4);
V_A_d(1,2) = -r_RW*cos(q5);
V_A_d(3,1) = -r_RW*sin(q4)*sin(q5);
V_A_d(3,2) = -r_RW*sin(q5);

FWa = V_A_i + V_A_d*J;

% ----------------- Partial Velocity of Rear Frame Mass Center ----------------- %
V_B_i = zeros(3);
\begin{align*}
V_{B, i}(1, 2) &= -r_{RW} \cos(q_5); \\
V_{B, i}(2, 1) &= r_{RW} \cos(q_5)^2 + z_B \cos(q_5) + r_{RW} \sin(q_5)^2 + x_B \sin(q_5); \\
V_{B, i}(3, 2) &= -r_{RW} \sin(q_5); \\
V_{B, d} &= zeros(3); \\
V_{B, d}(1, 1) &= -z_B \sin(q_4) - r_{RW} \cos(q_5) \sin(q_4); \\
V_{B, d}(1, 2) &= -z_B - r_{RW} \cos(q_5); \\
V_{B, d}(1, 2) &= x_B \cos(q_4) \cos(q_5) - z_B \cos(q_4) \sin(q_5); \\
V_{B, d}(1, 3) &= -x_B \sin(q_4) - r_{RW} \sin(q_4) \sin(q_5); \\
V_{B, d}(2, 2) &= -x_B - r_{RW} \sin(q_5); \\
\mathbf{P}_{vb} &= V_{B, i} + V_{B, d} + J; \\
\% \text{ Partial Velocity of Front Frame Mass Center} \% \\
V_{C, i} &= zeros(3); \\
V_{C, i}(1, 1) &= \sin(q_7) \left( r_{RW} \cos(q_5)^2 + z_B \cos(q_5) + r_{RW} \sin(q_5)^2 + x_B \sin(q_5) \right) + z_C (\sin(lambda) \sin(q_5) \sin(q_7) - \cdots) \cos(lambda) \sin(q_5) \sin(q_7); \\
V_{C, i}(1, 2) &= r_{RW} \cos(q_5) \sin(lambda) \sin(q_5) - \cdots \cos(lambda) \cos(q_5) \sin(q_7); \\
V_{C, i}(2, 1) &= \cos(q_7) \left( r_{RW} \cos(q_5)^2 + z_B \cos(q_5) + r_{RW} \sin(q_5)^2 + x_B \sin(q_5) \right) - z_C (\cos(lambda) \cos(q_5) \cos(q_7) - \cdots) \cos(q_7) \sin(lambda) \sin(q_5) + x_C (\cos(lambda) \sin(q_5) + \cdots) \cos(q_5) \sin(lambda)); \\
V_{C, i}(2, 2) &= r_{RW} \cos(lambda) \cos(q_5) \sin(q_7) - \cdots \cos(lambda) \sin(q_5) \sin(q_7); \\
V_{C, i}(2, 3) &= x_C; \\
V_{C, i}(3, 1) &= -x_C (\sin(lambda) \sin(q_5) \sin(q_7) - \cdots) \cos(lambda) \cos(q_5) \sin(q_7)); \\
V_{C, i}(3, 2) &= -r_{RW} \cos(lambda) \sin(q_5) - r_{RW} \cos(q_5) \sin(lambda); \\
V_{C, d} &= zeros(3); \\
V_{C, d}(1, 1) &= \sin(q_7) (x_S \cos(q_4) \cos(q_5) - z_S \cos(q_4) \sin(q_5)) + \cdots z_C (\cos(q_7) \sin(q_4) + \cos(lambda) \cos(q_4) \sin(q_5) \sin(q_7) + \cdots) \cos(q_4) \cos(q_5) \sin(lambda) \sin(q_7)) - \cdots \cos(lambda) \cos(q_7) (z_S \sin(q_4) + r_{RW} \cos(q_5) \sin(q_4)) + \cdots \cos(q_7) \sin(lambda) (x_S \sin(q_4) + \cdots) \cos(lambda) \sin(q_4) \sin(q_5)); \\
V_{C, d}(1, 2) &= z_C \cos(q_7) - \cos(lambda) \cos(q_7) (z_S + r_{RW} \cos(q_5)) + \cdots \cos(q_7) \sin(lambda) (x_S + r_{RW} \sin(q_5)); \\
V_{C, d}(2, 1) &= x_C \cos(lambda) \cos(q_4) \cos(q_5) - \cdots \cos(q_4) \sin(lambda) \sin(q_5)); \\
V_{C, d}(2, 2) &= \cos(lambda) \sin(q_7) (z_S + r_{RW} \cos(q_5)) - z_C \sin(q_7) - \cdots \end{align*}
\[
\sin(\lambda) \sin(q7) \cdot (x_S + r_{RW} \sin(q5)) ;
\]

\[
V_{C_d}(3,1) = -\cos(\lambda) \cdot (x_S \sin(q4) + r_{RW} \sin(q4) \sin(q5)) - ... \\
\sin(\lambda) \cdot (x_S \sin(q4) + r_{RW} \cos(q5) \sin(q4)) - ... \\
x_C \cdot (\cos(q7) \sin(q4) + \sin(\lambda) \cos(q4) \sin(q5) \sin(q7) + ... \\
\cos(q4) \cos(q5) \sin(\lambda) \sin(q7)) ;
\]

\[
V_{C_d}(3,2) = -x_C \cos(q7) - \cos(\lambda) \cdot (x_S + r_{RW} \sin(q5)) - ... \\
\sin(\lambda) \cdot (x_S + r_{RW} \cos(q5)) ;
\]

\[ PV_c = V_{C_i} + V_{C_d} \cdot \bar{J} ; \]

\% ---------------- Partial Velocity of Front Frame Mass Center ----------------- %

\[
V_{D_i} = \text{zeros}(3) ;
\]

\[
V_{D_i}(1,1) = \sin(q7) \cdot (r_{RW} \cos(q5) \cdot 2 + z_S \cos(q5) + r_{RW} \sin(q5) \cdot 2 + ... \\
x_S \sin(q5)) + z_D \cdot (\sin(\lambda) \sin(q5) \sin(q7) - ... \\
\cos(\lambda) \cos(q5) \sin(q7)) ;
\]

\[
V_{D_i}(1,2) = r_{RW} \cos(\lambda) \sin(\lambda) \sin(q5) - ... \\
r_{RW} \cos(q5) \cos(q7) ;
\]

\[
V_{D_i}(2,1) = \cos(q7) \cdot (r_{RW} \cos(q5) \cdot 2 + z_S \cos(q5) + r_{RW} \sin(q5) \cdot 2 + ... \\
x_S \sin(q5)) - z_D \cdot (\cos(\lambda) \cos(q5) \cos(q7) - ... \\
\cos(q7) \sin(\lambda) \sin(q5)) + x_D \cdot (\cos(\lambda) \sin(q5) + ... \\
\cos(q5) \sin(\lambda)) ;
\]

\[
V_{D_i}(2,2) = r_{RW} \cos(\lambda) \cos(q5) \sin(q7) - ... \\
r_{RW} \sin(\lambda) \sin(q5) \sin(q7) ;
\]

\[
V_{D_i}(2,3) = x_D ;
\]

\[
V_{D_i}(3,1) = -x_D \cdot (\sin(\lambda) \sin(q5) \sin(q7) - ... \\
\cos(\lambda) \cos(q5) \sin(q7)) ;
\]

\[
V_{D_i}(3,2) = -r_{RW} \cos(\lambda) \sin(q5) - r_{RW} \cos(q5) \sin(\lambda) ;
\]

\[
V_{D_d} = \text{zeros}(3) ;
\]

\[
V_{D_d}(1,1) = \sin(q7) \cdot (x_S \cos(q4) \cos(q5) - z_S \cos(q4) \sin(q5)) + ... \\
z_D \cdot (\cos(q7) \sin(q4) + \sin(\lambda) \cos(q4) \sin(q5) \sin(q7) + ... \\
\cos(q4) \cos(q5) \sin(\lambda) \sin(q7)) - ... \\
\cos(\lambda) \cos(q7) \cdot (z_S \sin(q4) + r_{RW} \cos(q5) \sin(q4)) + ... \\
\cos(q7) \sin(\lambda) \cdot (x_S \sin(q4) + r_{RW} \sin(q4) \sin(q7)) ;
\]

\[
V_{D_d}(1,2) = z_D \cdot (\cos(q7) \cdot (\cos(\lambda) \cos(q5) \cdot (x_S + r_{RW} \cos(q5)) + ... \\
\cos(q7) \sin(\lambda) \sin(q5) \cdot (x_S + r_{RW} \sin(q5)) ;
\]

\[
V_{D_d}(2,1) = x_D \cdot (\cos(\lambda) \cos(q4) \cos(q5) - ... \\
\cos(q4) \sin(\lambda) \sin(q5)) + \cos(q7) \cdot (x_S \cos(q4) \cos(q5) - ... \\
z_S \cos(q4) \sin(q5)) + ... \\
z_D \cdot (\cos(\lambda) \cos(q4) \cos(q7) \cdot (z_S \sin(q4) + r_{RW} \cos(q5) \sin(q4)) + ... \\
\cos(q4) \cos(q5) \sin(\lambda) \sin(q7)) - ... \\
\cos(\lambda) \sin(q7) \cdot (z_S \sin(q4) + r_{RW} \cos(q5) \sin(q4)) + ... \\
\sin(\lambda) \sin(q7) \cdot (x_S \sin(q4) + r_{RW} \sin(q4) \sin(q5)) ;
\]

\[
V_{D_d}(2,2) = \cos(\lambda) \sin(q7) \cdot (z_S + r_{RW} \cos(q5)) - z_D \cdot (\sin(q7) - ... \\
\sin(\lambda) \sin(q7) \cdot (x_S + r_{RW} \sin(q5)) ;
\]

144
\[ V_{D_d}(3,1) = -\cos(\lambda)(x_S\sin(q4) + r_{RW}\sin(q4)\sin(q5)) - \ldots \]
\[ \sin(\lambda)(z_S\sin(q4) + r_{RW}\cos(q5)\sin(q4)) - \ldots \]
\[ x_D(\cos(q7)\sin(q4) + \cos(\lambda)\cos(q4)\sin(q5)\sin(q7) \pm \ldots \]
\[ \cos(q4)\cos(q5)\sin(\lambda)\sin(q7)) \]

\[ V_{D_d}(3,2) = -x_D(\cos(q7) - \cos(\lambda)\cos(q5)\cos(q5) - \ldots \]
\[ \sin(\lambda)(z_S + r_{RW}\sin(q5)) \]

\[ F_{V_d} = V_{D_d} + V_{D_d}J; \]
function [Aa_i,Aa_d,Ab_i,Ab_d,Ac_i,Ac_d,Ad_i,Ad_d]...
    = CalcAccelerations(Geo,x,J,Ui,Ud_dt)

% -- Geometry Vector
r_RW  = Geo(1);
x_B   = Geo(3);
z_B   = Geo(4);
x_S   = Geo(5);
z_S   = Geo(6);
x_C   = Geo(7);
z_C   = Geo(8);
x_D   = Geo(9);
z_D   = Geo(10);
lambda = Geo(11);

% -- Input Vector
q3   = x(1);
q4   = x(2);
q5   = x(3);
q6   = x(4);
q7   = x(5);
q8   = x(6);

% -- Independent Generalized Speeds
u4   = Ui(1);
u6   = Ui(2);

% -- Dependent Generalized Speeds
u3   = J(1,:)*Ui;

% -- Dependent Generalized Speed Derivatives
u3dt = Ud_dt(1);

% Body A = Rear Wheel
% Body B = Rear Frame
% Body C = Front Frame
% Body D = Front Wheel

% ========= Acceleration of Rear Wheel Mass Center =========== %
Aa_i = [ 0, -r_RW*cos(q5), 0;
        r_RW, 0, 0;
        0, -r_RW*sin(q5), 0];
Aa_d = vpa(zeros(3,1));
% Acceleration of Rear Frame Mass Center

\[
\begin{align*}
\text{Ab}_i &= \begin{bmatrix}
0, & -r_{\text{RW}}\cos(q_5), & 0; \\
r_{\text{RW}} + z_B\cos(q_5) + x_B\sin(q_5), & 0, & 0; \\
0, & -r_{\text{RW}}\sin(q_5), & 0;
\end{bmatrix}; \\
\text{Ab} &= \text{vpa}(\text{zeros}(3,1)); \\
\text{Ab}_d(1) &= r_{\text{RW}}u_5\sin(q_5)*(u_5 + u_6 + u_3\sin(q_4)) - z_B*(u_5D + ... \\
& - u_3\cos(q_5) + u_3\sin(q_4)) - z_B*(u_5\cos(q_4) + u_3\sin(q_4)) - ... \\
& - (u_4\sin(q_5) + u_3\cos(q_4)\cos(q_5))*(r_{\text{RW}}\cos(q_5)*(u_4\cos(q_5) - ... \\
& - u_3\cos(q_4)\sin(q_5)) + r_{\text{RW}}\sin(q_5)*(u_4\sin(q_5) + ... \\
& - u_3\cos(q_4)\cos(q_5)); \\
\text{Ab}_d(2) &= r_{\text{RW}}\sin(q_5)*(u_4u_5\cos(q_5) + u_3\cos(q_4)\cos(q_5) - ... \\
& - u_3u_4\cos(q_4)*\sin(q_4) - u_3u_5\cos(q_4)*\sin(q_5)) - ... \\
& - (u_4\sin(q_5) + u_4\cos(q_4)*\sin(q_5))*(r_{\text{RW}}\cos(q_5)*(u_4\cos(q_5) - ... \\
& - u_3\cos(q_4)\sin(q_5)) + r_{\text{RW}}\sin(q_5)*(u_4\sin(q_5) + ... \\
& - u_3\cos(q_4)\cos(q_5)) - (u_5 + u_6 + u_3\sin(q_4)) + ... \\
& - r_{\text{RW}}\cos(q_5)*(u_5 + u_3\sin(q_4))*(u_5 + u_6 + u_3\sin(q_4)); \\
\text{Ab}_d(3) &= (u_4\cos(q_5) - u_3\cos(q_4)\sin(q_5))*(r_{\text{RW}}\cos(q_5)*(u_4\cos(q_5) - ... \\
& - u_3\cos(q_4)\sin(q_5)) + r_{\text{RW}}\sin(q_5)*(u_4\sin(q_5) + ... \\
& - u_3\cos(q_4)\cos(q_5)) - r_{\text{RW}}\sin(q_5)*(u_5D + u_3\sin(q_4)) + ... \\
& - u_3u_4\cos(q_4) - (x_B*(u_4\sin(q_5) + u_3\cos(q_4)\cos(q_5)) + ... \\
& - z_B*(u_4\cos(q_5) - u_3\cos(q_4)*\sin(q_5)))*(u_4\sin(q_5) + ... \\
& - u_3\cos(q_4)\cos(q_5) - r_{\text{RW}}\sin(q_5)*(u_5 + u_3\sin(q_4)))*(u_5 + u_6 + ... \\
& - u_3\cos(q_4)\sin(q_4)); \end{align*}
\]
\( \text{Ab}_d(3) = (x_B*(u4*sin(q5) + u3*cos(q4)*cos(q5)) + z_B*(u4*cos(q5) - ... \)
\( u3*cos(q4)*sin(q5))*u4*cos(q5) - u3*cos(q4)*sin(q5)) - x_B*(u5DT + ... \)
\( u3DT*sin(q4) + u3*cos(q4)) + z_B*(u5 + u3*sin(q4))^2 + ... \)
\( u4*cos(q5) - u3*cos(q4)*sin(q5))*r_RW*cos(q5))*(u4*cos(q5) - ... \)
\( u3*cos(q4)*sin(q5) + r_RW*sin(q5)*(u4*sin(q5) + ... \)
\( u3*cos(q4)*cos(q5)) - r_RW*sin(q5)*(u5DT + u3DT*sin(q4) + ... \)
\( u3*u4*cos(q4)) - r_RW*u5*cos(q5)*(u5 + u6 + u3*sin(q4)) + ... \)
\( r_RW*cos(q5)*(u5 + u6 + u3*sin(q4))); \)

\% =============== Acceleration of Front Frame Mass Center =============== \%

\text{Ac}_i = vpa(zeros(3));

\text{Ac}_i(1,1) = sin(q7)*r_RW*cos(q5)^2 + z_S*cos(q5) + r_RW*sin(q5)^2 + ... \)
\( x_S*sin(q5)) + x_C*(sin(lambda)*sin(q5)*sin(q7) - ... \)
\( \cos(lambda)*cos(q5)*sin(q7)); \)

\text{Ac}_i(1,2) = r_RW*cos(q7)*sin(lambda)*sin(q5) - ...
\( r_RW*cos(lambda)*cos(q5)*cos(q7); \)

\text{Ac}_i(2,1) = cos(q7)*r_RW + z_S*cos(q5) + x_S*sin(q5)) - ...
\( z_C*(cos(lambda)*cos(q5)*cos(q7) - ...
\( \cos(q7)*sin(lambda)*sin(q5)) + x_C*(cos(lambda)*sin(q5) + ...
\( \cos(q5)*sin(lambda)); \)

\text{Ac}_i(2,2) = r_RW*cos(lambda)*cos(q5)*sin(q7) - ...
\( r_RW*sin(lambda)*sin(q5)*sin(q7); \)

\text{Ac}_i(2,3) = x_C;

\text{Ac}_i(3,1) = -x_C*(sin(lambda)*sin(q5)*sin(q7) - ...
\( \cos(lambda)*cos(q5)*sin(q7)); \)

\text{Ac}_i(3,2) = -r_RW*cos(lambda)*sin(q5) - ...
\( r_RW*cos(q5)*sin(lambda); \)

\text{Ac}_d = vpa(zeros(3,1));

\text{Ac}_d(1) = z_C*(cos(q7)*(u5DT + u3DT*sin(q4) + u3*u4*cos(q4)) + ...
\( \sin(lambda)*sin(q7)*(u4*u5*cos(q5) + u3DT*cos(q4)*cos(q5)) - ...
\( u3*u4*cos(q5)*sin(q4) - u3*u5*cos(q4)*sin(q5); + ...
\( \cos(lambda)*sin(q7)*(u4*u5*sin(q5) + u3DT*cos(q4)*sin(q5) - ...
\( u3*cos(q4)*sin(q5) + u3*cos(q4)*cos(q5)) - u7*sin(q7)*(u5 + ... \)
\( u3*cos(q4)*sin(q5) + u7*cos(q7)*sin(lambda)*(u4*sin(q5) + ...
\( u3*cos(q4)*cos(q5)) - (x_C*(u7 + cos(lambda))*(u4*sin(q5) + ...
\( u3*cos(q4)*cos(q5)) + sin(lambda)*(u4*cos(q5) - ... \)
\( u3*cos(q4)*sin(q5)) - z_C*(sin(q7)*(u5 + u3*sin(q4)) + ...
\( \cos(lambda)*cos(q7)*(u4*cos(q5) - u3*cos(q4)*sin(q5)) - ...
\( \cos(q7)*sin(lambda)*(u4*sin(q5) + u3*cos(q4)*cos(q5)))*(u7 + ... \)
\( \cos(lambda)*(u4*cos(q5) - u3*cos(q4)*sin(q5)) + u3*cos(q4)*cos(q5)) + ...
\( \sin(lambda)*(u4*cos(q5) - u3*cos(q4)*sin(q5)) - x_C*(cos(q7)*(u5 + ...
\( u3*sin(q4)) - cos(lambda)*sin(q7)*(u4*cos(q5) - ... \)
\( u3*cos(q4)*sin(q5) + sin(lambda)*sin(q7)*(u4*sin(q5) + ...
\( u3*cos(q4)*cos(q5)) + 2 + sin(q7)*(x_S*(u4*u5*cos(q5) + ... \)
\[
\begin{aligned}
\cos(q_7)\sin(\lambda)\sin(q_5) = &\ ...
\cos(q_7)\sin(q_5) = &\ ...
\end{aligned}
\]
cos(lambda)*cos(q7)*(u4*cos(q5) - u3*cos(q4)*sin(q5)) - ... 
cos(q7)*sin(lambda)*(u4*sin(q5) + u3*cos(q4)*cos(q5)) - ... 
X_D*(cos(q7)*(u5*D + u3*D*sin(q4) + u3*u4*cos(q4)) + ... 
sin(lambda)*sin(q7)*(u4*u5*cos(q5) + u3*D*cos(q4)*cos(q5)) - ... 
u3*u4*cos(q5)*sin(q4) - u3*u5*cos(q4)*sin(q5)) + ... 
cos(lambda)*sin(q7)*(u4*u5*sin(q5) + u3*D*cos(q4)*sin(q5)) - ... 
u3*u4*sin(q4)*sin(q5) + u3*u5*cos(q4)*cos(q5)) - ... 
u7*sin(q7)*(u5 + u3*sin(q4)) - u7*cos(lambda)*cos(q7)*(u4*cos(q5) - ... 
u3*cos(q4)*sin(q5)) + u7*cos(q7)*sin(lambda)*(u4*sin(q5) + ... 
u3*cos(q4)*cos(q5));
function [ALa_i, ALa_d, ALa_b_i, ALa_b_d, ALa_c_i, ALa_c_d, ALa_d_i, ALa_d_d]...
    = CalcAlphas(Geo, x, J, Ui, Ud_dt)

% -- Geometry Vector
x_RW = Geo(1);
x_RW = Geo(2);
x_B = Geo(3);
z_B = Geo(4);
x_S = Geo(5);
z_S = Geo(6);
x_C = Geo(7);
z_C = Geo(8);
x_D = Geo(9);
z_D = Geo(10);
lambda = Geo(11);

% -- Input Vector
q3 = x(1);
q4 = x(2);
q5 = x(3);
q6 = x(4);
q7 = x(5);
q8 = x(6);

% -- Independent Generalized Speeds
u4 = Ui(1);
u6 = Ui(2);
u7 = Ui(3);

% -- Dependent Generalized Speeds
u3 = J(1,:) * Ui;
u5 = J(2,:) * Ui;
u8 = J(3,:) * Ui;

% -- Dependent Generalized Speed Derivatives
u3dt = Ud_dt(1);
u5dt = Ud_dt(2);
u8dt = Ud_dt(3);

{%
    Body A = Rear Wheel
    Body B = Rear Frame
    Body C = Front Frame
    Body D = Front Wheel
%}

% ============== Angular Acceleration of Rear Wheel ============== %

ALa_i = [ cos(q5), 0, 0;
      0, 1, 0;
      sin(q5), 0, 0];

ALa_d = vpa(zeros(3,1));
\[
\begin{align*}
\text{ALa}_d(1) &= (u5 + u3*\sin(q4))*(u4*\sin(q5) + u3*\cos(q4)*\cos(q5)) - \\
&\quad (u4*\sin(q5) + u3*\cos(q4)*\cos(q5))*((u5 + u6 + u3*\sin(q4)) - \\
&\quad u4*u5*\sin(q5) - u3DT*\cos(q4)*\sin(q5) + u3*u4*\sin(q4)*\sin(q5) - \\
&\quad u3*u5*\cos(q4)*\cos(q5)); \\
\text{ALa}_d(2) &= u5DT + u3DT*\sin(q4) + u3*u4*\cos(q4); \\
\text{ALa}_d(3) &= (u4*\cos(q5) - u3*\cos(q4)*\sin(q5))*(u5 + u6 + u3*\sin(q4)) - \\
&\quad (u5 + u3*\sin(q4))*(u4*\cos(q5) - u3*\cos(q4)*\sin(q5)) + u4*u5*\cos(q5) + \\
&\quad u3DT*\cos(q4)*\cos(q5) - u3*u4*\cos(q4)*\sin(q5) - u3*u5*\cos(q4)*\sin(q5); \\
\% \text{ --------------- Angular Acceleration of Rear Frame ---------------} \%
\text{ALb}_i &= [\cos(q5), 0, 0; \\
&\quad 0, 0, 0; \\
&\quad \sin(q5), 0, 0]; \\
\text{ALb}_d &= vpa(zeros(3,1)); \\
\text{ALb}_d(1) &= u3*u4*\sin(q4)*\sin(q5) - u3DT*\cos(q4)*\sin(q5) - u4*u5*\sin(q5) - \\
&\quad u3*u5*\cos(q4)*\cos(q5)); \\
\text{ALb}_d(2) &= u5DT + u3DT*\sin(q4) + u3*u4*\cos(q4); \\
\text{ALb}_d(3) &= u4*u5*\cos(q5) + u3DT*\cos(q4)*\cos(q5) - u3*u4*\cos(q5)*\sin(q5) - \\
&\quad u3*u5*\cos(q4)*\sin(q5); \\
\% \text{ --------------- Angular Acceleration of Front Frame ---------------} \%
\text{ALc}_i &= [\cos(\lambda_1)*\cos(q5)*\cos(q7) - \cos(q7)*\sin(\lambda_1)*\sin(q5), 0, 0; \\
&\quad \sin(\lambda_1)*\sin(q5)*\sin(q7) - \cos(\lambda_1)*\cos(q5)*\sin(q7), 0, 0; \\
&\quad \cos(\lambda_1)*\sin(q5) + \cos(q5)*\sin(\lambda_1), 0, 1]; \\
\text{ALc}_d &= vpa(zeros(3,1)); \\
\text{ALc}_d(1) &= \sin(q7)*(u5DT + u3DT*\sin(q4) + u3*u4*\cos(q4)) - \\
&\quad \cos(q7)*\sin(\lambda_1)*(u4*u5*\cos(q5) + u3DT*\cos(q4)*\cos(q5)) - \\
&\quad u3*u4*\cos(q5)*\sin(q4) - u3*u5*\cos(q4)*\sin(q5)) - \\
&\quad \cos(\lambda_1)*\cos(q7)*(u4*u5*\sin(q5) + u3DT*\cos(q4)*\sin(q5)) - \\
&\quad u3*u4*\sin(q4)*\sin(q5) + u3*u5*\cos(q4)*\cos(q5) + u7*\cos(q7)*(u5 + \\
&\quad u3*\sin(q4)) - u7*\cos(\lambda_1)*\sin(q7)*(u4*\cos(q5) - \\
&\quad u3*\cos(q4)*\sin(q5)) + u7*\sin(\lambda_1)*\sin(q7)*(u4*\sin(q5) + \\
&\quad u3*\cos(q4)*\cos(q5)); \\
\text{ALc}_d(2) &= \cos(q7)*(u5DT + u3DT*\sin(q4) + u3*u4*\cos(q4)) + \\
&\quad \sin(\lambda_1)*\sin(q7)*(u4*u5*\cos(q5) + u3DT*\cos(q4)*\cos(q5)) - \\
&\quad u3*u4*\cos(q5)*\sin(q4) - u3*u5*\cos(q4)*\sin(q5)) + \\
&\quad \cos(\lambda_1)*\sin(q7)*(u4*u5*\sin(q5) + u3DT*\cos(q4)*\sin(q5)) - \\
&\quad u3*u4*\sin(q4)*\sin(q5) + u3*u5*\cos(q4)*\cos(q5) - u7*\cos(q7)*(u5 + \\
&\quad u3*\sin(q4)) - u7*\cos(\lambda_1)*\sin(q7)*(u4*\cos(q5) - \\
&\quad u3*\cos(q4)*\sin(q5)) + u7*\sin(\lambda_1)*\sin(q7)*(u4*\sin(q5) + \\
&\quad u3*\cos(q4)*\cos(q5)); \\
\text{ALc}_d(3) &= \cos(\lambda_1)*(u4*u5*\cos(q5) + u3DT*\cos(q4)*\cos(q5) - \\
&\quad u3*u4*\cos(q5)*\sin(q4) - u3*u5*\cos(q4)*\sin(q5)) - \\
&\quad \sin(\lambda_1)*(u4*u5*\sin(q5) + u3DT*\cos(q4)*\sin(q5)) - \\
&\quad u3*\cos(q4)*\cos(q5));
\end{align*}
\]
\[ u_3 \times u_4 \times \sin(q_4) \times \sin(q_5) + u_3 \times u_5 \times \cos(q_4) \times \cos(q_5) \]
function [Fa_RW,Fa_RF,Fa_FF,Fa_FW,...
    Ta_RW,Ta_RF,Ta_FF,Ta_FW] = CalcActiveForces(x,props,T)

% -- Input Vector
q3 = x(1);
q4 = x(2);
q5 = x(3);
q6 = x(4);
q7 = x(5);
q8 = x(6);

% -- Properties Input
m_RW = props(1);
m_RF = props(2);
m_FF = props(3);
m_FW = props(4);
g = props(5);
lambda = props(6);

% -- Torques
T_RW = T(1);
T_R = T(2);
T_S = T(3);

{%
    Body A = Rear Wheel
    Body B = Rear Frame
    Body C = Front Frame
    Body D = Front Wheel
%}

% =----------------------------------- Active Forces =----------------------------------- %

% - Forces
Fa_RW = [-g*m_RW*cos(q4)*sin(q5);
          g*m_RW*sin(q4);
          g*m_RW*cos(q4)*cos(q5)];

Fa_RF = [-g*m_RF*cos(q4)*sin(q5);
          g*m_RF*sin(q4);
          g*m_RF*cos(q4)*cos(q5)];

Fa_FF = [-g*m_FF*(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) + ... 
            cos(q4)*cos(q5)*cos(q7)*sin(lambda));
          g*m_FF*cos(q7)*sin(q5) + cos(lambda)*cos(q4)*sin(q5)*sin(q7) + ...
          cos(q4)*cos(q5)*sin(lambda)*sin(q7));
          g*m_FF*cos(lambda + q5)*cos(q4)];

Fa_FW = [-g*m_FW*(cos(lambda)*cos(q4)*cos(q7)*sin(q5) - sin(q4)*sin(q7) + ... 
            cos(q4)*cos(q5)*cos(q7)*sin(lambda));

\[ g \cdot m \cdot \text{FW} \cdot (\cos(q_7) \cdot \sin(q_4) + \cos(\lambda) \cdot \cos(q_4) \cdot \sin(q_5) \cdot \sin(q_7) + \ldots \cos(q_4) \cdot \cos(q_5) \cdot \sin(\lambda) \cdot \sin(q_7)) \]
\[ g \cdot m \cdot \text{FW} \cdot \cos(\lambda + q_5) \cdot \cos(q_4) \}; \]

% - Torques
Ta_RW = [0; T_RW; 0];

Ta_RF = [T_R*\cos(q_5); -T_RW; T_R*\sin(q_5) - T_S];

Ta_FF = [0; 0; T_S];

Ta_FW = [0; 0; 0];