Development of Kinematic and Dynamic Models for the Argo J5 Rover

by

Erin J. E. Austen

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Aerospace Engineering

Department of Mechanical and Aerospace Engineering

Carleton University
Ottawa, Ontario

Abstract

Planetary exploration rovers are the most efficient means of off-world surface exploration. As mobile laboratories, they are used to perform various experiments and gather data semi-autonomously from remote extraterrestrial environments, both for planetary science and assessing conditions in preparation for human exploration. To accomplish the mission and access sites of scientific interest, the rover must be able to traverse various types of unstructured terrain without becoming embedded or succumbing to other hazards. Modelling of the rover is essential to understand how the rover interacts with its environment and how to select the best path. This thesis presents the development of three-dimensional kinematic and dynamic models, using MATLAB and SimMechanics, describing the Argo J5 four-wheel rover, in response to terrain elevation inputs and slip. The kinematic models describe the pose and velocity of the rover using the Denavit-Hartenberg convention, while the SimMechanics dynamic model is combined with a terramechanics model to develop accelerations and obtain the forces and torques, based on terrain properties. The kinematic analyses were performed for simulated traverses including cases of flat, inclined, side slope, and sinusoidal terrain, with varying amounts of slip in the velocity analysis. The results showed good agreement with expected trends and values for the joint displacements and rates, with the largest percent deviation for the distance travelled being approximately 0.4 %. The results of the combined dynamic and terramechanics model, incorporating slip, are limited to the conceptual development of the model due to time constraints, and are thus inconclusive at this time.

Dedication

“Remember to look up at the stars and not down at your feet. Try to make sense of what you see and wonder about what makes the universe exist. Be curious. And however difficult life may seem, there is always something you can do and succeed at.
It matters that you don't just give up.”

Dr. Stephen Hawking

Acknowledgements

Foremost, the author would like to thank her primary supervisor Dr. M. J. D. Hayes for his valued guidance, advice, and overall support throughout the progression of this project, along with Dr. M. Faragalli for special advising. Special thanks are also due to Mission Control Space Services for providing a rover to model and their input regarding the Argo J5 rover. The author also wishes to express her deep gratitude to Dr. R. Irani for the use of his terramechanics model and for his coaching on the use of SimMechanics, and for his overall patience.

Additional thanks go to the author's friends and support group: her Grad School Survival Crew, fellow Shieldmaidens and kickboxing partners, A. Pant, G. Ng, J. Hunter, M. Magnus, and S. Milton, for their unwavering belief in the author's capacity to achieve, while struggling through difficult times. Special thanks are due to G. Ng for allowing the author to bounce ideas off him.

Finally, the author would like to thank her sister, parents, and four-legged family members for always having her back and providing encouragement to reignite her passion when it was needed the most.

Table of Contents

List of Figures

List of Tables

Nomenclature

English Symbols

a_i	link length (m)
a_{wb}	length of walking beam (m)
b	wheel (tread) width (m)
c	cohesion (Pa)
d_i	link offset (m)
d_{cwb}	lateral distance from c of g to walking beam joint (m)
$D(q)$	inertia matrix
F_Z	vertical force from terrain (N)
g	gravitational constant (m/s^2)
$G(q)$	gravity vector
h_{cg}	height of rover c of g from ground (m)
J	Jacobian
J_v	translational velocity Jacobian
J_ω	rotational velocity Jacobian
$j(\theta)$	soil deformation (m)
K_j	kinetic energy (J)
k	Bekker coefficient of proportionality (N/m^{n-2})

k_c	cohesive modulus of sinkage (N/m^{n+1})
k_φ	frictional modulus of sinkage (N/m^{n+2})
K	shear modulus (m)
L	LaGrangian
$M(q)$	Mass matrix
m	mass (kg)
n	sinkage exponent
P_j	potential energy (J)
$p(z)$	pressure as a function of sinkage
q	joint variable positions
$\dot{q}$	joint variable rates
r	wheel radius (m)
R	rotation matrix
T	homogeneous transformation matrix
Δt	time step or interval (sec)
$\vec{V}$	velocity vector of the end effector
v_x	horizontal translational velocity (m/s)
v_t	tangential velocity (m/s)
$V_x(q, \dot{q})$	Coriolis and centrifugal velocities

X	coordinate in the x-direction (m)
X_{trans}	translational distance travelled in x direction (m)
$\ddot{X}$	acceleration column vector
Y	coordinate in the y-direction (m)
Y_{trans}	translational distance travelled in y direction (m)
Z	coordinate in the z-direction (m)
Z_{trans}	translational distance travelled in z direction (m)
z_j	vertical component or displacement (m)
z	sinkage (m)

Greek Symbols

α	link twist (radians)
β	½ angle of walking beams (radians)
γ	slope
δ	slope value, also rotation of wheel contact point, ie. δ_{rf} for right, front wheel
η	ratio of track depth to sinkage
ηz	track depth (m)

Λ	logical operator for joint type
φ_{pitch}	pitch of chassis, overall rover (radians)
φ_{roll}	roll of chassis, overall rover (radians)
φ_{yaw}	yaw of chassis, overall rover (radians)
$\sigma(\theta)$	normal stress at contact point (Pa)
τ	torque (Nm)
$\tau(\theta)$	shear stress at contact point (Pa)
θ	joint angle (radians)
θ_{chas}	chassis pitch, also ϕ_{pitch} (radians)
θ_{wbl}	left walking beam pitch (radians)
θ_{wbr}	right walking beam pitch (radians)
θ_f	wheel sinkage entry angle (radians)
θ_m	location of mean normal stress (radians)
θ_r	wheel sinkage exit angle (radians)
ω	angular velocity of wheel (radians/sec)

Non-dimensional Parameters

i	slip, defined as $i = 1 - \frac{v_x}{\omega r}$

Definitions/Abbreviations

CAD	Computer aided design
CoG	Centre of Gravity (also C of G)
D-H	Denavit-Hartenberg
DoF	Degrees of Freedom
DP	Drawbar Pull
MCSS	Mission Control Space Services

Pose: the position and orientation of a rigid body.

Prismatic joint: a single degree of freedom joint that describes linear translation between two bodies.

Revolute joint: a single degree of freedom joint that describes rotational motion between two bodies.

Statement of Original Contributions

The key contributions of the research shown in this thesis are:

1. A three-dimensional kinematic model of the Argo J5 rover.

2. A conceptual three-dimensional dynamic model combined with a wheel-soil interaction model (terramechanics) for the Argo J5 rover.

Chapter 1: Introduction

This thesis is an investigation of the issue of traversability facing exo-planet rovers and the means and methods available to address it. For a particular rover, the Argo J5, a number of approaches were attempted in pursuit of developing kinematic and dynamic models to assist in path planning, but also to assess the capabilities and limitations of this particular rover under varying terrain conditions. The need for the work presented herein is revealed in the greater context of extraterrestrial exploration.

In humanity's ongoing quest to reach out into space, planetary exploration rovers remain the most efficient and feasible means of exploring the surface of other bodies in our solar system. Satellites, while they have the potential to cost less, are limited mainly to mapping and imaging missions, like the Odyssey Orbiter mission [1]. Although highly useful in obtaining planet-wide data, they are limited to the resolution of their payload and cannot perform in-situ measurements. Landers such as the Phoenix Mars Lander [2], and more recently InSight [3], are also valuable tools of surface exploration and in-situ data collection but are limited to the location in which they land and their physical reach. Although the eventual goal is to place humans on Mars and other bodies in our solar system, it's more expensive to send humans into space, as it requires more fuel and infrastructure. In addition, the risk to human lives is much greater. Planetary rovers have the capability to travel between multiple sites of scientific interest without the limitations of the human body and can go where humans can't easily access while carrying various scientific payloads and instrumentation. Subsequently, rovers can improve the chances of

success for the preparation and execution of a human mission. However, it should be noted that drones are an increasing opportunity as there are plans to send one to Mars [4] and NASA has recently announced a return to Saturn's moon, Titan [5], with the Dragonfly mission to take samples and measurements of both surface and atmosphere.

Planetary rovers are the most popular means of surface exploration, contingent upon them being able to operate in remote and challenging environments. If traversability or mobility is limited, a rover can be prevented from achieving its goal and possibly realizing a mission critical failure. In order to get a sense of the complexity and importance of traversability of planetary rovers, we need to examine the challenging conditions they may face.

1.1 Martian Environment

One of the fundamental questions driving human exploration of space is to determine whether life is unique to Earth. Naturally, we look to our closest neighbours in our solar system for new insights: our Moon, Venus, Mars, Galilean moons, etc. Venus and Mars are both terrestrial (rocky) inner planets, however Venus is too hot and volatile to easily land a rover, with previous Russian landers lasting no more than two hours [6]. Mars is also relatively close, as the fourth planet from the Sun, and has potential not just for water and the building blocks of life, but as a place to establish a human colony. Smaller than Earth, with roughly 10% of Earth's mass and 37.5% of Earth's gravity,

Mars is the most similar planet to Earth since it has an atmosphere (albeit significantly thinner), a Martian day (sol) comparable to Earth at 24 hours and 37 min, and has similar weather phenomena such as dust storms, and seasons due to its axial tilt of 25.2^{o} [7, 8]. Martian temperatures are known to vary between daytime highs and lows by about 70^{o}C in winter and 100^{o}C in summer (on average) dependent upon location. Planetary observations have provided surface evidence of the existence of water by water-cut channels, along with water beneath the polar ice deposits and glaciers. These observations were recently corroborated by Orosei et al's analysis of radar profiles from the Mars Express spacecraft [9].

However, for all the similarities between Earth and Mars and its relative accessibility, it remains a challenging destination. Humans are unable to survive on Mars without significant infrastructure and environmental suits due to conditions on the surface, making a manned mission both high risk and expensive. In preparation for sending humans to Mars, there have been many observation and planetary exploration missions to gather and analyze planetary data. In addition to being safer and cheaper than sending humans, rovers are able to traverse much of the Martian terrain to investigate various scientific sites of interest and carry a suite of instruments. A significant challenge of operating rovers on the Mars is that due to the vast distances between Earth and Mars (which is variable depending on their orbital relationship), there is a significant time delay or latency period between signals transmitted and received. On average, it takes a transmission around 13 min to reach Mars, thus a roundtrip delay of 26 min [10, 11]. The orbiter itself typically communicates with the rover for approximately 8 min at a time,

per sol. This time delay makes remote operation by direct control unfeasible, requiring all planetary rovers to be semi-autonomous, which also enables more time to be dedicated to rover operations such as driving to new targets or performing its mission. Mars has received the most probes, from the Mariner satellites as the first to orbit and NASA's Viking mission with the first successful landers [12]. Sojourner was the first rover successfully landed, followed by MERs Spirit and Opportunity, and with Curiosity currently in operation. The European Space Agency's Rosalind rover (formerly ExoMars 2020) and NASA's Mars 2020 rover are scheduled for launch in the near future.

Operating a planetary rover in a widely varying terrain on a remote world has its challenges. To be successful in its mission and access the various scientific regions of interest, it is highly important to examine the rover, its traverse, and its ability to deal with obstacles.

1.1.1 Obstacles

One of the more obvious challenges for a rover in traversing from point A to point B, is that the shortest path generated is not always the best or has the shortest excursion time or is safest. Often, there are obstacles in the way that either cannot be overcome, or should not be attempted, such as Olympus Mons or the highly rippled dunes of recurring slope lineae (RSL), that are seasonal and have very steep slopes. These are called geometric obstacles which, as the name implies, are hazards that have significant volume based on geometry, such as large boulders or crevasses. Such hazards can sometimes be viewed from orbital imagery, but definitely through the rovers own visual sensors. As

such, most vision systems are quick to focus on geometric obstacles of the terrain and spend most of their effort on avoiding them or minimising contact with them. These features are commonly included in path planning algorithms [10].

However, there are other obstacles and challenges that are not as easily observed. These become classified as non-geometric hazards that do not depend on their dimensions. With regards to rover mobility, non-goemetric hazards are some of the major challenges facing a rover, slip in particular. Slip is the condition where the wheel turns but does not progress as much in the forward direction due to loss of traction. As such, slip is harder to analyse from an overall satellite map, and thus requires additional analysis to plan traverse. Slip has been a major challenge to the current and previous rovers, with it severely limiting progress and even leading to mission critical events. Slip-sinkage effect: wherein, on deformable terrain, as slip increases for a wheel, the lessening of forward motion and increasing wheel spinning excavates the ground around it [13]. An example of the slip-sinkage effect is shown in Figure 1.1, where the wheel ends up excavating the terrain surrounding it and subsequently the wheel sinks and starts to embed, making it very difficult to extract.

Figure 1.1: Testbed wheel beginning to sink and embed [15].

If the sinkage becomes significant enough to have embedding, the rover mission could fail. Spirit experienced significant slip-sinkage and embedding, which was made worse when one of its wheels failed and ended up being dragged. This failure ultimately terminated the mission when it became embedded and couldn't be extricated [15]. Figure 1.2 shows a picture taken by Spirit of its embedded wheel.

Figure 1.2: Spirit's embedded wheel (courtesy NASA/JPL).

1.1.2 Terrain Types

In order to reach science targets and gather data, the planetary rover must be capable of traversing the Martian terrain. Mars is a rocky body, with mountainous and heavily cratered terrain in the southern hemisphere and large, comparatively smooth plains comprised of basaltic lava flows and sedimentary deposits [16]. Additionally, the Martian surface has many dunes as a result of the wind which is one of the dominant forces shaping the surface [16]. From both satellite and rover data, the terrain can vary from hard bedrock to softer sands. These variations in terrain and the larger features all pose challenges which the rover must contend with. Thus, the traversability of the rover is a key concern for design. The current and previous Mars rovers all feature a rocker-

bogie suspension system which has proven advantageous in that it can negotiate obstacles 1.5 times the wheel diameter [17, 18].

Bedrock

Due to the firmness or hardness of bedrock, this type of terrain has the least amount of wheel sinkage [19]. Although this might make bedrock seem like the best surface for driving on, it is often sharp and thus results in more wear on the tires/rover wheels. Indeed, the MSL Curiosity rover opted during sol 463 to drive on some available bedrock to minimize wheel sinkage and obtain better traction, only to damage its wheel tread and significantly increase the wear/damage to the wheels as seen in Figure 1.3. In addition, smooth bedrock can be hard to gain traction on and thus result in higher amounts of slip as noted by Heverly et al [20] from the Scarecrow data and, with more texture, the tractive performance improves as the wheel grousers gain purchase on the terrain.

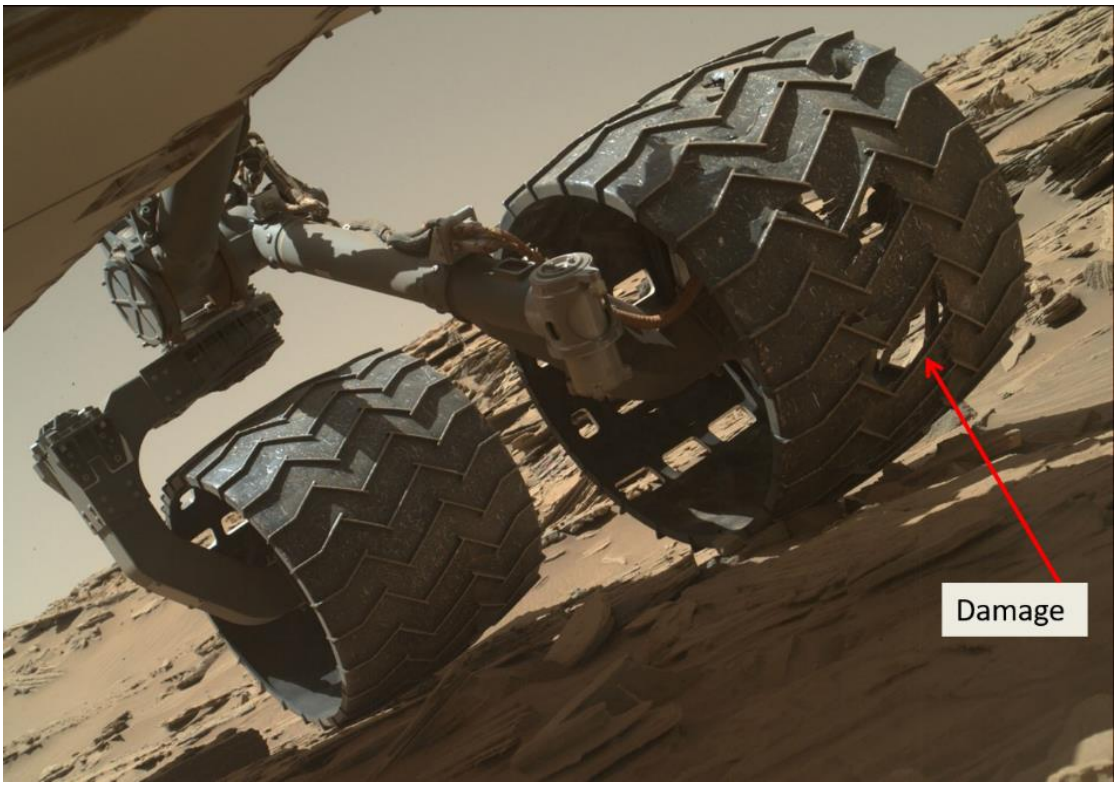


Figure 1.3: Curiosity wheel damage (courtesy NASA/JPL).

Cohesive Sand

Mars is also covered by a lot of sandy terrain. Sand can be further separated by its cohesion value, which is the degree that the individual particles or grains of sand stick together and provide a firmer surface. Cohesive soil or sand is composed of finer grain particles that want to adhere or stick to each other, even more so when wet. They have a high clay content and do not easily crumble. Such terrain is fairly easy for the rover to traverse, even uphill, as the transition from low to high slip is more continuous [20]. The adhesion between the terrain particles means that terrain deformation, and by extension sinkage of the wheels, is typically minimal for cohesive sand [20]. However, it should be

noted that this terrain type experiences the most variability due to its reliance upon other environmental factors, such as moisture content.

Cohesionless Sand

The other broad classification of terrain is cohesionless sand. As the name implies, cohesionless sand particles do not adhere to each other or pack together. As noted by Heverly [20], with the analog Scarecrow tests, this terrain poses a significant challenge to the rover and its ability to make forward progress. Due to the lack of cohesion, as slip increases, the wheels experience more sinkage since the sand gives way and is excavated from beneath the wheels. The resulting sinkage significantly impedes the forward progress of the rover and can result in embedding which can take weeks to extricate the rover [20]. It should also be noted that the slopes able to be traversed on cohesionless sand are significantly lower, and the transition to high slip occurs at a lower slope [20] such as 75% slip occurring at a slope of 16º for the Scarecrow, vs 30% slip at 28º slope.

Duricrust

Another terrain of note is duricrust, as while it may appear firm on the surface, it tends to have more porous and softer material underneath [21]. Duricrust forms via accumulation due to the presence of groundwater [21] and is of interest to planetary scientists looking for evidence of water (and life) on Mars. This particular type of terrain is challenging as it is not necessarily obvious and the hard-upper shell can be broken by the wheel, allowing the wheel to sink and embed.

1.2 Planetary Exploration Rovers

The work presented in this thesis is focused on planetary exploration rovers since they are currently the best means (including costs) of investigating the surfaces of remote worlds. For rover surface exploration, wheeled locomotion is favoured since wheels are simple, cost efficient (longer design life), and shown to be effective in negotiating smaller obstacles less than half the wheel diameter [22]. Since rovers are mobile science platforms/laboratories, it is desired that they be fairly resilient, steady, sturdy platforms upon reaching their target location in addition to traversing the challenging and varied terrain to reach their specific target. As such, to help select a path, it is needed to know the pose of the rover as it travels that potential path. Four rovers have been sent to Mars, with only Curiosity remaining operational as the MER Opportunity ended its mission June 2018 after a planet-wide dust storm [23]. To help extend the rover mission life, it is important to get an understanding of how the rover reacts to various terrains and use that information to evaluate the path. Finally, with the return to the Moon, more rover missions are expected to aid human exploration of the Moon as well.

1.2.1 Rover Configurations

Planetary exploration rovers are generally relatively small vehicles, with Spirit and Opportunity having a mass of 174 kg [24], and Curiosity being the largest at 899 kg [25]. All rovers that have been sent have been of a six-wheel configuration, as are the two near future rover missions. Additionally, these rovers all favoured what's commonly

referred to as a rocker-bogie suspension. The common six-wheel configuration is shown below in Figure 1.4 for the Curiosity rover.

Figure 1.4: Rocker-bogie configuration for the Curiosity rover (courtesy NASA/JPL) [25].

Examining Figure 1.4, one can see that two of the wheels are connected via a smaller link called the rocker [25]. The rocker connects to the larger link that is connected directly to the chassis which is call the bogie. Between the two sides of the rover there is a differential to help average out the motion of the chassis [25]. The advantage of using a rocker-bogie design with Ackermann steering is that it increases the capability of the

rover to negotiate obstacles while minimising the chances of the wheels losing contact with the terrain [25].

These rovers are well documented and make up the majority of published research. However, the rover in question for axe investigation is a different configuration.

1.2.2 Argo J5 Rover

For the work presented in this thesis, the vehicle in question is the Argo J5 rover, depicted in Figure 1.5. It is the candidate vehicle selected by Mission Control Space Services (Ottawa).

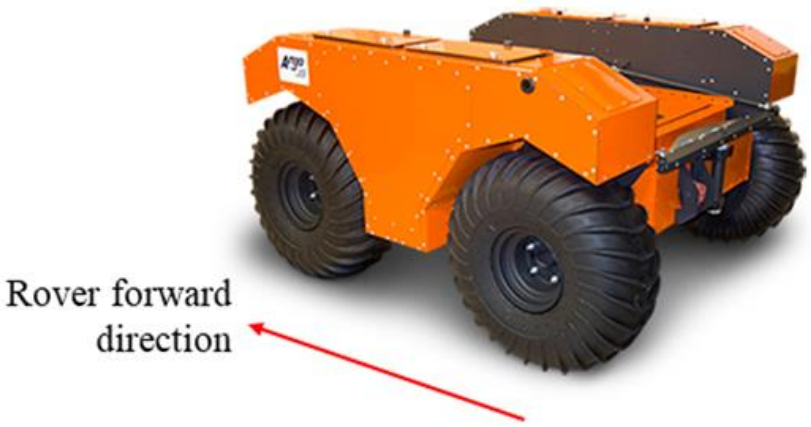


Figure 1.5: Argo J5 rover [24].

The J5 is a large platform vehicle, roughly in size of an all-terrain vehicle (ATV) and is thus capable of holding a significant payload. It has been tested with regards to terrestrial

applications in fire-fighting and amphibious scenarios [26]. Compared with the more traditional rover configuration, it is immediately obvious that the J5 rover does not have Ackermann steering, as it only possesses four wheels. The J5 operates via skid steering, meaning that a separate motor drives each side, with the wheels on one side being connected via a belt drive. Turning with skid steering, involves one side's wheels turning more slowly or opposite to the other side. Examining Figure 1.5, one can see that the rover can be considered to have three main components, the chassis and two walking beams of which each has two wheels.

In addition to differences in the number of wheels and the absence of a rocker-bogie system, the J5 has a unique rear suspension system connecting the walking beams to the centre rear of the chassis. The rear suspension acts to average the pitch of each walking beam, in order to try and keep the chassis level with respect to the terrain. Since the rear suspension connection to the walking beam is aft of the joint between the walking beam and the chassis, a pitch upwards of the left walking beam would push the left suspension rod aft, causing the back beam to rotate and push the right suspension rod forward. The result of the overall motion attempts to compensate for the left walking beam upward pitch by forcing the right walking beam to pitch downward. Finally, the J5 has the option of two different tires, rubber and metal. For the purposes of the work presented here, it is assumed to have the metal tires intended for planetary exploration.

1.3 Thesis Objectives and Outline

The overall goal of this thesis is to develop a process for assessing the traversability of a planetary rover for a given path, based on the terrain data from previous drives and other data from visual terrain maps. Specifically, to create models that can take in terrain data, in the form of digital elevation maps (from scans of the surface), to analyse the traverse of a rover along a selected path. This work involves developing a dynamic model to first characterise the reaction of the rover as it drives over terrain and then modifying it to interface with a wheel-soil interaction model including slip, to simulate potential soft terrain conditions. Empirical slip-slope data obtained from previous drives could be used as an input. In the context of this work, it is assumed that the slip data has already been incorporated in the terrain map. Eventually, the goal of this process is to use the model data to develop metrics to characterise the traversability of the vehicle for the specified terrain.

The following specific objectives have been identified for this work:

Research Objective 1*: produce a model to determine the pose of the rover in response to terrain geometry (ie. slopes & bumps, etc).*

The determination of pose will be accomplished through an inverse position kinematics model of the J5 rover. Terrain is input as the x, y, z coordinates of the respective path for each wheel. This type of data can realistically be pulled from digital elevation maps from satellite data. With the analysis, it is possible to use the pose of the rover to get a sense of

its stability at that point in time on the map. The creation of a position kinematics model is also useful as it serves as the foundation for velocity analysis and, by extension, the dynamic analysis of the rover. There is currently no publicly available model for the Argo J5 rover.

Research Objective 2*: produce a three-dimensional velocity kinematics model incorporating predicted slip to accurately determine the progression of the rover on the terrain.*

A three-dimensional inverse velocity kinematic analysis will be accomplished by extending the inverse position kinematics. It is assumed for the velocity portion that the terrain map path was first processed through terrain classification software, such as Mission Control Space Services' Autonomous Soil Assessment System (ASAS). The slopes measured from the digital elevation map were then used along with the terrain type, to determine slip values from the appropriate slip-slope curve obtained from experimental drives of the J5 rover. These slip values have been assumed and are included in the terrain maps used. The resulting inverse velocity kinematics model with slip incorporated enables the time of the traverse to be determined as a potential evaluation criterion.

Research Objective 3*: produce a dynamic model, incorporating slip and terrain geometry as inputs, allowing for other traversability metrics, such as torques and drawbar pull, to be determined.*

The three-dimensional dynamic model is included, as it is only at this level that the full effect of the rover wheels' interaction with the terrain can be determined. Since dynamics includes forces, the model is paired with a previously established terramechanics model to obtain the resulting forces acting on each wheel. The terrain elevation and slip experienced by each wheel are (again) inputs to the dynamic model from the digital terrain map.

This work is organised as follows: Following the introductory chapter, a review of traversability related studies and modeling is presented in the Literature Review of Chapter 2. Chapter 3 introduces the governing analysis methodology that is the Denavit-Hartenberg convention. The bulk of the analysis follows in Chapters 4 and 5, with Chapter 4 detailing all aspects of the kinematic analyses, leaving the dynamic analysis to be covered in Chapter 5. Both of these chapters begin with the theoretical formulation and application, and then finish with a display of representative results and discussion. Finally, this document concludes with Chapter 6 which summarises the overall results, along with comments and recommendations for future work. Appendices pertinent to each section of analysis are included at the back, containing full formulation, MATLAB scripts, and other results.

Chapter 2: Literature Review

The extreme challenges of exploring relatively unknown and uncontrolled environments posed by planetary exploration, means that it is essential to have designed as robust a vehicle as possible to maximise the probability of success. It is crucial to be able to model and understand the vehicle to get a sense of how it will perform and whether it has any specific limitations prior to its implementation. Furthermore, as unexpected challenges arise during the mission, having a reliable model aids in generating strategies likely to mitigate their outcomes. Successful mitigation has allowed several rovers to outlive their designed mission life [14, 16]. Large quantities of research and analysis have been published in various areas of rover science and development underscoring its popularity. Over the years these analyses allowed for expansion of the capabilities of rovers. As mentioned in Chapter 1, more rover missions are forthcoming with the ESA's Rosalind rover and NASA's Mars 2020, along with a return to the Moon, making ongoing rover research relevant.

Although there are many published research papers relating to planetary exploration rovers, this literature review will be focused on the following specific areas only: terramechanics and wheel-soil interaction models, rover vehicle modeling, slippage estimation, and performance metrics.

2.1 Terramechanics and Wheel-Soil Interaction Models

One of the most prevalent and oldest methods of studying the interaction between a vehicle and the terrain characteristics is terramechanics. First theorised by Bekker in the 1960's [27], the term was popularized by Wong's work in the 1980's [28]. Bekker and Wong are often referred to throughout much of the literature and are seen as the founders of Terramechanics. As defined by Wong, terramechanics is the "study of the overall performance for the machine in relation to its operational environment – the terrain [is typically considered to consist of] two main branches: terrain-vehicle mechanics and terrain-implement mechanics" [28]. Regarding the traversability or ability to traverse the terrain, planetary rovers are more concerned with their respective tractive performance while negotiating the unstructured terrain and its obstacles, which is the domain of terrain-vehicle mechanics [28]. Terramechanics as a discipline has since expanded to become its own in-depth field. Although the field is broad, this section tries to focus on research pertaining to planetary rovers and suitable non-Earth terrains.

In the field of terramechanics, Wong's expansion on Bekker's initial work has many of the content and equations referred to as the Bekker-Wong terramechanics. Wong's work was consolidated into two textbooks, *Terramechanics and Off-road Vehicle Engineering: Terrain Behavior, Off-road Vehicle Performance and Design* [29] and *Theory of Ground Vehicles* [30], and they are frequently viewed as essential texts on the subject. The content includes what are considered the basic equations of

terramechanics. Classical terramechanics uses characteristics and properties of the terrain to determine the stresses, and subsequently forces, imparted to the vehicle. The equations are semi-empirical, requiring knowledge of certain terrain properties determined through extensive in-situ lab testing. One technique is the bevameter technique which is comprised of two tests [29]. The plate penetration test allows the pressure-sinkage relationship to be obtained for a contact area the size of the plate, along with the motion resistance of the terrain. The shear test enables values related to traction, such as tractive-effort slip, to be computed using shear strength and displacement relationships. Such tests are not yet feasible on planets and moons, since limited exploration has occurred. One of the main outputs from terramechanics is the force in the longitudinal direction: drawbar pull. Drawbar pull is the force corresponding to the ability to overcome the motion resistance imparted by the terrain [29, 31]. Sinkage is the other important result from terramechanics analysis and is essential for understanding the effects of deformable terrain, which can significantly impact the rover's progress [13, 15, 19].

Classic terramechanics has since given rise to other methods, which can be categorised as empirical, analytical, or semi-empirical. Within each of these categories, the analyses have sometimes been modified for better accuracy. One such example is the Wong-Reece equation, which resulted from Reece's modifications to Wong's work on the pressure-sinkage relationship, as detailed by Ding et al [31]. Taheri et al [32] provide a good summary and comparison of other terramechanic models beyond those discussed herein.

2.1.1 Empirical Methods

Empirical models are those based on correlations determined from experimental data. Consequently, these relationships are more simplistic in nature and can only be applied to situations that match the conditions/environment in which the experiment was conducted. Hence, extrapolation to different terrains is not possible, nor are the models scalable [32]. As noted by Taheri et al [32], these models are more useful for a simple scenario such as determining a broad-based go or no-go. One example could be outfitting a rover with rotary encoders or using the visual odometry to determine the slip experienced while driving over known terrain types and slopes, such as those at a MarsYard, and producing slip-slope curves particular to the rover model, wheel design, and terrain type. Results from empirical methods can also be used as a measure to improve wheel design.

It should also be noted that, although Reece updated the pressure sinkage relationship as their major contribution to classical terramechanics, the relationship is focussed on longitudinal as opposed to lateral slip effects on sinkage as noted by Ding et al [31].

2.1.2 Analytical Methods

Due to the limited flexibility of purely empirical methods and the desire for improved accuracy, advancements in computing have allowed for the development of various analytical methods of modeling the wheel-soil interaction. Analytical or physics-based methods are ones that involve applying the appropriate principles of physics to the

system. The system in these cases refers to a discretized set of elements for both the terrain and vehicle, with interactions occurring between elements. As such, the system quickly becomes more complex with the additional components. With high complexity, the presence of discretized elements in the system requires the use of more powerful analytical methods to solve. From the overview provided by Taheri et al [32], it is noted that these methods have the potential to be more accurate and reflective of actual processes, particularly for highly deformable terrains, but they come at a cost of greater computation time. Such advancements also present the possibility of real-time capability given the right set of conditions (ie. particle/element size not too small, etc). as indicated by a surge in publications over the last decade. The main analytical techniques are Discrete Element Method (DEM) and Finite Element Method (FEM).

Discrete element method or DEM centres around modeling the soil as a system of spherical, discrete elements or particles [32]. The mechanical interactions between individual particles and those adjacent to them are included, with normal and tangential stiffness and damping applied to each force element, along with friction force in the tangential direction. For most DEM models, Coulomb damping is applied. In 2014, Smith et al [33] conducted a comparison between DEM and traditional and dynamic Bekker terramechanics modeling. They examined the three methods for a single wheel testbed, that could be applied to a rover-like small vehicle, at steady-state conditions. Upon examination of the predicted performance, the dynamic method was determined to be more realistic than traditional Bekker since it can not only accommodate multibody dynamics, but also simulate that with more complex soils. Furthermore, the results

supported that DEM is capable of being more realistic because it can model the movement of soil from the interaction along with the deformation that occurs. The discretized nature of the particles allows for modeling non-homogeneous soil characteristics. More realistic particle shapes can also be used in the model to increase the overall accuracy, however the associated increase in computational time is something that must be evaluated.

Most of the published literature only applies DEM to a single wheel test bed, rather than a full body dynamic simulation of a rover. Johnson et al [34] built upon previous work of single wheel DEM simulations for the Mars Exploration Rovers (MERs), specifically examining the accuracy of DEM simulations with regards to wheel mobility performance predictions. A DEM software called Coupi 3D DEM was used to perform various simulations for the single wheel, which were then compared to experimental results. With higher confidence in the low slip regime, the parameters of the DEM simulation were tuned accordingly, prior to generating predictions for the higher slip regimes. More realistic particle shapes were used, including trisphere, ellipsoid, and poly-ellipsoid (as opposed to spherical), so interlocking effect could occur as they would in nature. The DEM simulations were found to be most accurate for regions of high slip (slip, i = 0.7, 0.9, 0.99).

Johnson et al also published a comparison between classic terramechanics and DEM analyses for the MER wheels [35]. Again, the DEM simulations were performed using Coupi 3D DEM for a single wheel, over the entire slip range from 0 to 1. The

performance metrics used for comparison were drawbar pull and wheel sinkage. Due to the limitations of the single wheel testbed, the wheel tests were only performed up to a slip of 0.7. The results obtained show that classical terramechanics does indeed break down for high slip ($i > 0.6$) conditions. Johnson et al explain the limitation to be due to the presence of a tailings pile of regolith which contributes to a higher sinkage and is not accounted for in the classic terramechanics equations. The authors recommend a series of lookup tables based on good agreement of DEM simulations for scenarios of high slip.

Compared to empirical methods, the DEM is more realistic and accurate with nature since the discretized elements and characteristics of the analysis mean that empirical equations are unnecessary to complete the wheel simulations [33]. However, as mentioned previously, the detailed nodes significantly increase the computing process/computing time. As such, in the tuning stage there is no efficient way to tune the model and get it to match the experimental results other than via curve fitting.

Finite element method or FEM assumes a continuum rather than a discretized surface, and the continuum is comprised of a finite number of elements [29]. Due to the use of the continuum assumption, continuum mechanics are able to be applied. Unlike the spherical and semi-spherical elements used in DEM, the elements of FEM are angular, with quadrilateral and triangular shell elements. For 3D FEM, the computing cost is much higher than that of 2D, but less than that of DEM. However, for accuracy, compared to terramechanics, a more reasonable computing time is gained, although FEM

has certain limitations. Some of these limitations include that it cannot replicate large and discontinuous soil deformation, along with soil flow. According to Taheri et al. [32], some of these limitations are due to FEM being ill-equipped to handle singular boundary conditions.

Recently, there has been an increase in research interest for combining certain aspects of FEM and DEM analytical methods to try and get the best of both techniques and provide a higher degree of accuracy without exponentially increasing the computation time. Increased computation time essentially removes the possibility of real-time force and sinkage prediction for path planning. In 2016, Nishiyama et al [36] combined FEM and DEM models for a 2D analysis of the wheel-soil interaction for the wheel of a planetary rover. FEM was used for the more rigid (although elastic) body that was the wheel, whereas DEM was only applied to the top layer of the soil due to it experiencing the most deformation. Simulations analysing tractive performance were computed for flat terrain, including slip, and were then compared to experimental results from the single wheel test bed. Good agreement was found between the results, suggesting that pairing FEM with DEM is beneficial for controlling the additional computing time.

2.1.3 Semi-empirical Methods

Although there are advantages and disadvantages to empirical and analytical techniques, the majority of the published literature utilises semi-empirical models which are founded on a modified version of the Bekker-Wong equations. Semi-empirical

techniques combine experimental results and correction factors with the formulations of terramechanics equations and certain aspects of analytical methods [32]. Applications of these semi-empirical methods are popular in literature as they provide a suitable degree of accuracy without costing too much in computation time. These methods also allow for some extrapolation unlike pure empirical models. Many of the software packages fall into the semi-empirical category, such as AS^{2}TM and SCM [32].

One of the issues with the traditional Bekker-Wong terramechanics models is that they do not account for dynamic effects in the wheel-soil interactions. In 2010 and 2011, Irani et al attempted to address this issue for smooth wheels [37] and for the dynamic effects of grousers on rigid wheels [38]. Previous single wheel testbed results were showing repeating ridge tracks, which are not accounted for in traditional Bekker-Wong. The modeling of this effect was accomplished by modifying the pressure-sinkage relationship to include use of a manually tuned sinusoidal function [37] and through the generation and addition of new empirical dimensionless coefficients in the sinusoidal function [38]. Both models were compared with experimental test data for a single wheel testbed and loose sandy soil, using the usual metrics of normal force, sinkage and drawbar pull, and were found to predict the oscillations observed. This work was further expanded by Irani et al with a third publication in 2012 [39]. The model also included modifications to the Reece-Wong pressure sinkage relationship. Once again, fluctuations in drawbar pull seen in previous experiments were predicted by the model, but in this study for a range of slip values and normal loads. The results showed good agreement with the experimental results, also revealing that increases in slip and normal loads

correlated with an increase in the amplitude of the added term, with a similar relationship demonstrated for sinkage.

Ghottbi et al [40] published a sensitivity analysis for mobile robots in unstructured environments in 2016. The sensitivity analysis was computed for both the Bekker and the Wong-Reece methods for a single wheel to try and ascertain which models were more sensitive to certain parameters/conditions. Regarding soil parameters such as cohesion and internal angle of friction, the Wong-Reece was more sensitive, in particular for position and velocity analyses. The authors also found that larger wheels, both radially and widthwise, were less sensitive to the terrain and experienced lower magnitudes of stress. Knowing the sensitivity of the model being used can help in both using the model to optimise a design and overall to interpret the accuracy of the results.

One of the semi-empirical tire-soil interaction models that has been popular, is the AESCO Soft Soil Tire Model or AS^{2}TM [32]. The AS^{2}TM model is built for use in MATLAB and Simulink, modeling a solitary point of contact, for both steady state and dynamic conditions with real-time capability. Since the model was built as an add-on for Simulink, it is typically paired with a multibody dynamics simulation to analyse traction and mobility. The model is built upon the classic Bekker formulation while it also incorporates an improvement for lateral shear. It can also be made to incorporate different terrains and flexible or rigid tires at the discretion of the user, using the Bekker and Coulomb values. AS^{2}TM does have a drawback in that it cannot compute the bulldozing

force. However, it can account for other parameters such as rolling resistance, soil compaction, slip-sinkage, and multipass effect.

Another prevalent semi-empirical model is the Soil Contact Model (SCM) [32]. The SCM model is also based on the Bekker equations; however, it is a 3D model where the tire is modeled as a solid object and the solid is represented by a set of discretized columns.

Perhaps one of the most well-known applications of semi-empirical terramechanics in the field of planetary exploration rovers is the ARTEMIS system employed on both MERs and Curiosity [13, 19, 20, 41]. In 2011, Iagnemma et al [41] published their results on the terramechanic modeling included in ARTEMIS. ARTEMIS is an Adams-based dynamic simulation of the rover paired with a modified Bekker-Wong terramechanics approach that was optimised using a least-squares optimisation equation. Their paper details not only the development of ARTEMIS, but also the application of it to simulate Opportunity's drive to the Endeavour Crater, along with Spirit popping a wheelie and an embedding scenario. The results of these simulations were validated with real data. The authors also demonstrate that ARTEMIS could be used for estimating terrain parameters for the dry sand case. The results were accurate to within +/- 10% for most parameters. The only parameter for which it was not very accurate was for determining the cohesion value, which they expected due to the nature of dry sand having poor cohesion.

Most common in the application of semi-empirical terramechanics equations, focus is given to the parameters as they might affect the longitudinal progress and stability criteria of the rover, which can be seen in the some of the literature presented herein. In 2007 Ishigami et al [42] investigated the effects of terramechanics on the steering maneuvers of a four-wheel planetary rover on lunar regolith simulant. Using the terramechanics equations, Ishigami et al focused on the lateral components, including side force and bulldozing resistance, which logically have an impact on the wheel during steering maneuvers. They subsequently developed a model to predict and analyse the steering maneuvers for a given rover. The model was first validated using a single wheel test bed, prior to combining with a multibody dynamic model. The combined multibody dynamic model was evaluated for different slip cases and experimentally validated using a four-wheeled rover test bed. Comparison with the more common kinematic steering models showed improvements in accuracy predictions.

A commentary on various semi-empirical terramechanics models was compiled in 2012 by Chhaniyara et al [43], for inclusion in their review on terrain trafficability analysis. Although their review was more focused on the terrain classification side of the analysis, as opposed to trafficability, they identify that some degree of testing to obtain some parameters is unavoidable due to the nature of the models. Furthermore, they demonstrate that by using terrain classification and obtaining the slip and sinkage, some of the terrain interaction parameters are obtainable.

In 2010, Ding et al [31] released the results of their investigation into the slip-sinkage relationship from the Wong-Reece terramechanics model. The experiment was conducted for three different sized rover wheels on a single wheel testbed, which were varied for different slip ratios and lug heights to observer the effects. From the experiment, the authors were able to use the results and improve the existing Wong-Reece model by calculating sinkage based on vertical load and slip ratio, in addition to changing the sinkage exponent to a variable dependent upon the slip ratio. By doing so, the slip-sinkage takes into account more of the actual contributing factors, whereas the traditional Bekker method is limited to computing static sinkage and the Wong-Reece only accounts for longitudinal slip-sinkage. Slip-sinkage is a critical failure mode that must be understood for mission success.

Ding et al's work on sinkage continued, with another paper published in 2017 [44] focusing on modeling sinkage using terramechanics as a basis, and also to determine the moment of in-situ steering wheels. Again, they looked at cases of deformable terrain. Unlike their previous paper, this study focused more on a steering-sinkage relationship. The collected data, along with the terramechanics formulation, allowed for the sinkage relationship to be further improved.

Continuing with the problem of exploring unspecified terrains, Gallina et al [45], focused on the problem of lack of knowledge of terrain parameters required by soft soil contact models. They conducted their investigation using a multibody dynamics model paired with the SCM model mentioned above. Their results indicate that the Bayesian

approach was the best way to manage the uncertainties in these parameters; however, their results illustrate the limitation of pure models for such unknown and unstructured environments.

2.2 Rover Vehicle Modeling

The other side of the problem is the modeling of the rover vehicle itself in such a way that the terrain data can be incorporated, and the response of the rover can be determined. Terramechanics and wheel-soil interaction models can provide force inputs and terrain deformation (sinkage); however, depending on the rover size and configuration, the effects can differ significantly. In addition, the geometry of the terrain itself, will also have an effect on the rover's pose and stability. Vehicle modeling is generally classified as one of two model types – kinematics or dynamics. Kinematics describes the position and velocity of the vehicle, whereas dynamics focuses on the forces and accelerations present and can be manipulated for torques. Depending on the search parameters, vehicle kinematics often brings up material regarding position and heading (steering) of the vehicle with the application of path planning and following. Most models in existence are multibody dynamic models from multibody physics software. Consequently, many of these models require specific physics-based software platforms, such as ADAMS, to operate.

2.2.1 Kinematic Modeling

A paper by Tarokh and McDermott et al [46] published in 1999, was found to use a 3D kinematic approach with the D-H convention for a Mars rover. In their paper they apply the D-H convention to a small 6-wheel rover, the Rocky 7, which is closer to the size of Soujourner than Curiosity. Their method begins by assigning frames starting from the centre of mass and working outwards to each of the wheel contact points. From the frame assignments and corresponding parameters, the authors were able to get a full set of equations describing the forward and inverse kinematics. Additionally, for the velocity portion of the analysis, they computed specific wheel Jacobians as they found it provided a more accurate depiction of wheel roll and slip. For this paper, only the method was described and there was no experimental data presented for comparison with the results. Due to the nature of the methodology, with each contact point being treated like an end effector, the authors mention that their model should be applicable to traversing rough or uneven terrain.

Later in 2004, Chakraborty and Ghosal [47], developed a kinematic analysis for a three wheeled robot on uneven terrain. Although similar to Tarokh, Chakraborty and Ghosal chose a slightly different application of frames in their model, beginning from the contact point of each wheel and working up through the joints to meet at the common end effector, which was the chassis in this case. Rather than using the D-H convention and obtaining the corresponding parameters and transform matrices, the corresponding translations and rotations were covered using Euler angles and rotation matrices coupled

with rotations. In addition, the authors chose to model their wheels as tori, rather than thin disks, to capture the location of the assumed single point of contact. Even though slip was not included in the model as an input or as a predicted output, they used the tori wheels to generate another set of constraint equations for which the system could only be solved if its joints were able to find positions that removed slip from the scenario. Numerical solvers were employed to solve the set of ODEs.

In 2005, Tarokh and McDermott [48] published paper expanding their previous kinematic model into a more generalised kinematic analysis. As in their previous study, the analysis was again applied to the Rocky 7 rover following the same D-H approach, although using more specifically described wheels as end effectors. Similarly, the wheel Jacobians were kept and enabled full motion of the wheel. However, in this paper, more results are shown from actual simulations of the rover over given terrains. The terrains are given as a function or an elevation map, with a single contact per wheel and no sinkage or penetration of the terrain is assumed. The outputs of the simulations include pitch and roll of the rover, along with elevation and joint angles. Tarokh and McDermott's simulations were able to achieve good results for a variety of different paths (ie. straight and serpentine), even with the addition of noise. However due to the nature of the kinematic model, no different terrain types were tested.

Shortly afterwards, McDermott and Tarokh followed up with another paper [49] further generalising their approach and establishing an overall guide to the general approach to kinematic modeling of a rover. Although this work was applied again to the

6-wheel Rocky 7 rover, they applied differential kinematics to obtain the motion of the wheels and likewise wheel Jacobians. The modeling of Rocky 7 was considered successful as it was able to travel over the simulated terrain; however, this model is still somewhat limited in its application as, although it can calculate slip, it does not account for soft or deformable terrain effects. Thus, sinkage and embedding effects cannot be observed. It is also limited in that the terrain input is comprised of multiple discontinuous terrain contact points.

Building on the work of Chakraborty and Ghosal, in 2009 Auchter et al [50] produced a kinematic solution for a simple three-wheel rover on uneven terrain, and then applied it to test a passive vehicle camber (PVC) component. As such, their model also uses a torus to model the wheels, allowing for lateral tilt, and follows the idea of the parallel manipulator with multiple kinematic chains leading to a single end-effector. However, in their study, the inputs to the joints are considered known and the position and velocity of the platform are the outputs. Their model includes equations describing the wheel contact and are used to constrain the wheels to roll only, with no slip. When applied to the rover with PVC components, the program monitors the constraint equations for violation indicating there is slip occurring. The results of their study show that their design does reduce slip. It should be noted that the rover modeled, although using a more complicated wheel model, does not have complex linkages taken into effect.

In 2010, Parakh et al [51], published a different approach to kinematic modeling of a six-wheel rover. In their approach, the rover is modeled in planar form with each

joint locus given a set of coordinates and equations to describe the rigid Euclidean distances between the joint loci. This method is further detailed in Chapter 4 as part of a "working up" investigation to obtain an idea of how pose would change with respect to the terrain. However, it should be noted that the rover joint loci only give location in x and y since the model is planar.

In 2012, Tarokh et al [52] published further work on kinematic modeling of high mobility rovers, with their paper presenting a systematic approach to a general kinematic analysis that could be applied to any rover over uneven terrain and used to aid in parametric design studies. Expanding beyond their previous formulations, they produced an extended D-H table for easier formulation and highlighted their algorithm which takes the table and directly applies it to create the kinematic model. In their paper, the method and corresponding algorithm is applied to a very complicated, small, multi-tasking rover with individual leg actuators and hence capable of different angles and configurations. Different terrain topologies were used as inputs, with zero slip assumed and all wheels in contact with the ground at all times. The results presented show the model was able to follow the desired trajectory with minimal errors.

Another investigation examining 3D kinematics for wheeled mobile robots was published in 2014 by Seegmiller and Kelly [53]. Although they focused on a three-dimensional approach for their simple rover, they also acknowledge the larger number of 2D kinematic models due to their relative ease in derivation and the consequently cheaper computational costs and higher computational speeds. Often a 2D approach is acceptable

due to the slow speeds of the rovers themselves, however for uneven terrains, the 3D analysis is required. Seegmiller and Kelly used more of a velocity propagation approach as opposed to D-H convention and transform matrices which are not inefficient given the relative simplicity of their rover. Like all the kinematic models presented so far, a single point of contact was assumed for each wheel. Slip prediction for the rover body was one of the outputs included and their results, when applied, were found to have a significant improvement on the odometry of the rover.

2.2.2 Dynamic Modeling

The more common form of modeling used for planetary exploration rovers is dynamic modeling. Various software platforms and packages exist which make the modelling work a lot simpler than generating the dynamic equations using a technique like La Grange formulation. Most of the modeling literature (even some of terramechanics) relates to dynamic modeling as opposed to kinematic modeling.

One of the more frequent multibody dynamic softwares mentioned in the literature is the ARTEMIS software for dynamic modeling paired with terramechanics. In 2011, Trease et al [13], published details on ARTEMIS and its implementation. ARTEMIS stands for Adams-based Rover TErramechanics and Mobility Interaction Simulator. As the acronym implies, ARTEMIS is a multibody dynamics software based on the Adams multibody platform. Adams/View is a very expensive software package, with a steep learning curve, requiring users to undertake specific training. With ARTEMIS, the published studies don't go into kinematics as, with multibody dynamics,

it's easier to stick with dynamics and that is what most operators are more concerned about, with particular regards to forces, torques, and slip, etc. From the basic Adams/View dynamic model of the rover, it was then paired with subroutines analysing the terramechanics based off of the semi-empirical Bekker-Wong equations. The main input to ARTEMIS is a generated terrain path obtained from digital elevation maps. When applied to previous drives by both Spirit and Opportunity, ARTEMIS was able to successfully replicate the drives including other aspects of each scenario, such as the embedding experienced by both rovers. By successfully recreating the embedding, ARTEMIS enabled further analysis along with how to extricate Opportunity from its embedding. Such analysis included successful modeling of the slip-sinkage effect with ARTEMIS being able to reproduce the high wheel slip leading the increased excavation of soil and the resulting downward displacement or sinkage of the wheel. At the time of this publication, ARTEMIS did not address deformable, changing terrain, continuous contact with multiple wheels, and rolling resistance. From one publication alone, it can be seen why ARTEMIS has persisted in its popularity and also how, for non-NASA rovers, it means trying to find other solutions that aren't too expensive.

Another dynamic analysis software for rovers was developed a couple years later by Srividhya et al [54] called Software for Modelling and Analysis of Rover on Terrain or SMART. Similar to ARTEMIS, it uses Adams/View as its foundation wherein the rover itself is modeled from an imported CAD model. Simplified terramechanics equations are applied using the Adams/Solver component and uneven terrains are generated using Adams Road Definition files generated in MATLAB, including the

properties of the terrain. Macros are generated for easier user inputs, and they apply this software to a simple four-wheel rover with no suspension or differential. The model assumes that the peak normal stress occurs at the midpoint of the wheel, thereby making the location of the peak stress independent of the wheel slip. Running the model, the user obtains slip, sinkage, drawbar pull, and drive torque, with slip being obtained from drawbar pull and resistances. They were able to model slopes and associated motion resistances, obstacle negotiation and steering, with simulations performed over flat terrains both smooth and uneven. At the time of publication, the simulation results had yet to be verified.

An additional example of dynamic model paired with a wheel soil interaction model is the Rover Chassis Analysis and Simulation Tool (RCAST). An overview of the RCAST is given by Bauer et al [55] with respect to its development and then application to a 6-wheel rocker-bogie rover. RCAST examines both the terrain and the dynamics of the rover for mobility, allowing the user to designate the terrain case, soil type, and obstacles. The terramechanics aspect of the analysis is based off of the AESCO soft tire model (AS^2TM) and employs the usual linear Coulomb friction for the tractive force. RCAST was successfully validated for a single wheel negotiating a step obstacle, where the friction coefficient was observed to agree strongly between the simulation and experimental. Later testing was completed for five different wheels of same size, but the number of grousers was varied, and tested over a range of slopes. The results for drawbar pull demonstrated the expected effect where increasing the number of grousers improves drawbar pull by increasing wheel traction.

Some dynamic models have combined more than just dynamics with a wheel-soil interaction model. ROSTDyn is a rover simulation based on terramechanics and dynamics, with the multibody software, Vortex Physics Engine, providing the dynamics model. The simplified terramechanics model is based off of the work by Iagnemma et al [41] and Li et al [56], with three main equations to obtain the normal force, drawbar pull, and the resistance torque. The physical model of the terrain utilises DEM. In Li et al's overview [56] of ROSTDyn, the platform is applied to a 6-wheel rover which is then simulated to drive over different inclined slopes. Due to ROSTDyn using a simplified terramechanics model over Vortex's built-in contact model, real-time simulation speed was possible for lower display frequency. The simulation yielding results for slopes from 4 -18^{o} were compared with the experimental data from driving the rover over the same slopes. The results were found to show good agreement with each other, with both exhibiting a decrease in the normal force and increase in the total drawbar pull, in conjunction with an increase in slippage, for the increasing slope. However, ROSTDyn is somewhat limited in that it does require a preliminary soil test to obtain the necessary soil parameters for terramechanics.

In 2013 Reina et al [57], examined both kinematic and dynamic modeling (combined) techniques in applying a simplified set to evaluate the performance of rovers with rocker-bogie suspension systems with respect to locomotion. Specifically, to investigate a particular design of rocker-bogie suspension that enables the wheel camber to change and adapt to the terrain. Simulations with inputs of terrain inclination angle,

wheel elevation, rover geometry, pose, and speed, were completed and then subsequently validated experimentally. Outputs obtained included: drive motor torque, wheel load ratio where increased load corresponds to increased traction ability, and the friction coefficient wherein a decrease in value indicates a better climbing ability. For the inverse kinematics it is unclear, but does not appear to use coordinate frame transformation methods differing from D-H. Additionally, it appears that the dynamics were limited to a simple quasi-static force analysis, with the application of forces being lumped at the centre of mass and certain contributions appearing to have been neglected. Finally, this model is limited in that only firm terrain was investigated.

Returning to ARTEMIS, Zhou et al [19] published a selection of simulations on the Mars rover traverses in 2014. In comparison to the initial publication by Trease et al [13], these simulations were not only done for a single wheel, but also the full vehicle. Traverses for both bedrock and deformable soil were performed. Validation was accomplished through single wheel tests and drives with Scarecrow (Earth analogue of Curiosity) at the MarsYard. At the time of publication, ARTEMIS was capable of calculating longitudinal wheel slip, wheel sinkage, normal stress, drawbar pull, longitudinal and lateral shear stress, lateral force, and grouser forces. For the full vehicle test, slip was manually determined which then allowed the sinkage to be indirectly based on slip-sinkage relationship. The updates to ARTEMIS were demonstrated in successful replication of MER and Curiosity drives, which also included demonstrating the ability to simulate both blind drives and those run through autonav. Having both single wheel tests and full vehicle tests allowed for further tuning of parameters. Good agreement was

already observed, with the rover's being able to climb slopes up to 20^{o} before achieving 90% slip, which was expected. It should be noted that ARTEMIS is still limited by its use of classical terrramechanics and is therefore less reliable when it comes to simulating regions of high slip such as rippled dunes which rover operators currently try to avoid. The development of ARTEMIS with regards to Curiosity was further examined by Senatore et al [58] for modeling and validation. Validation performed with both single wheel tests and full vehicle tests of Scarecrow, this time on the unprepared terrain of Dumont Dunes. Simulations were found to predict mobility characteristics, such as drawbar pull, to good agreement with the experimental data.

In 2009, Schafer et al presents a multibody simulation incorporating both soft and uneven terrain for the ExoMars rover prototype [59]. The multibody system dynamics were modeled using SimPack and incorporated two different wheel-soil contact models, depending on the terrain type. For harder terrains such as bedrock, a PCM contact model is used which looks at all possible contacts through collision detection with the hard rocks and populates appropriate elements with the contact forces and torques. Softer, deformable terrains utilise the SCM contact model, which as mentioned previously utilises DEM for the soil. Simulations were run for each uneven terrain for two different wheel geometries and the results were compared with those obtained from running a breadboard chassis in a sandbox of the same simulant. Good agreement was found for the mobility characteristics obtained for drawbar pull, wheel torque, etc along with successful modeling of the multi-pass and bulldozing effects, however further work

would be needed to confirm validation for other simulant types and conditions (ie high slip, slope).

2.3 Slippage Estimation

One of the more important non-geometric hazards imparted by the terrain is slip. Slip can drastically affect a rover's progress and as specifically noted by Chakraborty et al [47], slippage can be a significant wastage of an already limited supply of power. In addition to power wastage, the lack of forward progress caused by slippage also can severely impact the rover's ability to self-localise and lead to further errors with path planning/execution [50]. Within the research around rover mobility, there are also research groups focused more on mitigation through better design of wheels, suspension, and control strategies. Indeed, Seegmiller et al [53] notes that even in the area of incorporating slip with rover vehicle modeling, most kinematic models and published rover vehicle models compute slip as it occurs for use as a usable output, and it can then be incorporated into a feedback control scheme for path following. However, even with such a scheme, there is still the possibility of a collision occurring before the feedback controller has time to react.

Gonzalez et al published an overview of current work in slippage estimation and compensation in 2017 [60]. Limitations of different techniques are also included. From their investigation, it was observed that most of the published work in slip

is in the practical detection using visual odometry to provide an estimate. Furthermore, they note that little published work exists for slip compensation for slope traversals, along with the overall area of lateral or side slip. Lateral slip would be produced more by steering maneuvers (and corresponding forces) and is often neglected in the literature as it was assumed to be minimal for most test conditions. Side slopes would likely exhibit lateral slip but require further investigation to quantify the effect. Helmick et al [61], employed the Tarokh model for the Rocky 7 rover and, from the predicted slip, applied a control loop to keep the rover on a given path.

2.4 Performance Metrics

From many of the papers previously presented, in particular rover vehicle modeling, common mobility metrics include drawbar pull, wheel torque, slip, and sinkage. Many of these have been employed since before Wong's work on terramechanics, and in his textbooks gives drawbar pull or the drawbar pull coefficient or efficiency as acceptable methods of comparing data [29, 30]. The drawbar pull coefficient is simply the ratio of the drawbar pull to the vehicle weight and allows for better comparison across different rovers. Similarly, the efficiency compares the drawbar power potential to that expected of the vehicle.

Other metrics were investigated in 2010, by Thueer and Siegwart [62], in an attempt to standardise rover mobility characteristics for cross-vehicle comparison.

Although certain characteristics are computed in many of the published investigations, they are particular to the conditions of that investigation, thus little has been done in the area of mobile robotics to create a standard. Using a static model, Thueer and Siegwart suggest the metrics should be minimum friction and torque requirements. The minimum friction requirement allows ideal torques to be obtained and knowing this value can also be useful in reduction of slip. Unsurprisingly, absolute accumulated slip is another suggested metric, due to its non-dimensionality, however the authors acknowledge that it is more effective in simulations. The final suggested metric is velocity constraint violation which as the name suggests measures the deviation from the ideal or commanded velocity, compares it to the kinematic constraints, and computes the risk of violation. These standardised metrics were first computed for simulations of three different locomotion approaches and then compared with the experimental data. Overall, good correlation was observed between the simulated/predicted values and the measurements obtained from the physical experiment.

Chapter 3: Denavit-Hartenberg Methodology

To analyse the kinematics and dynamics of a robot, the robot must first be discretized into the corresponding kinematic pathways or chains. The kinematic chain describes the rigid components (links) and the joints connecting them, which generates a motion of the end effector within the boundaries of the workspace [63]. These kinematic chains can be used to construct mathematical models which describe the motion of the system as constrained by the joints and the lengths of the links they connect. There are different classifications of kinematic chains: open and closed are one such classification. Open kinematic chains have a series of connected links with a relatively independent end effector. Closed kinematic chains have a series of connected links in one or more loops with no open attachment point, similar to a four-bar linkage.

Using the kinematic chain, there are different methods to derive the mathematical equations, employing a geometric approach (one such approach will be illustrated in Section 4.1) or by attaching reference frames and applying rotations and translations to describe how the end of one link moves relative to the previous one. While a more purely geometric approach can be quite elegant and fairly practical in a planar case, once the kinematic chain reaches a larger number of degrees of freedom, or has a parallel chain, a more streamlined and standardised approach is desirable for most cases. One of the more popular methods is the Denavit-Hartenberg convention. In this work, the Denavit-Hartenberg convention is used to derive transformation matrices to determine the motion

of an end effector (output link) with respect to a reference link or relatively fixed coordinate system.

3.1 Introduction to the Devanit-Hartenberg Convention

As mentioned, to effectively operate robots it is imperative that one understands how the motion of the robot is constrained by the joints, limiting the configurations it can enter. Particularly, the description of, or ability to predict, the pose of the robot with certainty and with respect to a reference coordinate system is necessary for effective use and placement of the end effector in the reference system.

The Denavit-Hartenberg (D-H) convention is one such method of obtaining a description of a system by attaching reference frames to the joints of the kinematic chain, following a particular procedure. The result allows for four parameters to be determined which fully describe the displacement of the system [64, 65]. Although this method was first introduced in 1955 [66], the standardised procedure is fairly easy to apply and has become (and remains) a popular method of analysis for kinematic chains. This convention allows for a standard set of equations for different serial manipulators and the results can be left in a matrix form which many operators prefer. The resulting set of vector-matrix equations enable the kinematics of the manipulator to be fully described in 3D space. These 3D kinematic equations not only describe the pose of the end effector (the forward kinematics) given the input values but allow for the various configurations

to be determined (the inverse kinematics), which is helpful in terms of workspace constraints. The end effector is used to describe the end of the kinematic chain and, for manipulators, would be the location of the tool at the end of the robotic arm [64, 65]. Furthermore, these equations can be used to solve two different types of problems, either the forward or inverse kinematics. If the active joint values are specified, the forward kinematics problem determines the position and orientation (pose) of the end effector. Conversely, if one knows the desired pose of the end effector, the inverse kinematics problem can be solved to obtain the required, active joint values [64, 65]. In addition, this set of equations can be further manipulated to produce equations for velocity and acceleration analysis in 3D, which is the focus of Chapters 4 and 5. An acceleration analysis is necessary for modeling the dynamics of the system using the LaGrange method.

Typically for serial manipulators, the inverse kinematics problem is the more difficult to solve as there can be more than one solution or configuration to produce a specific end effector pose.

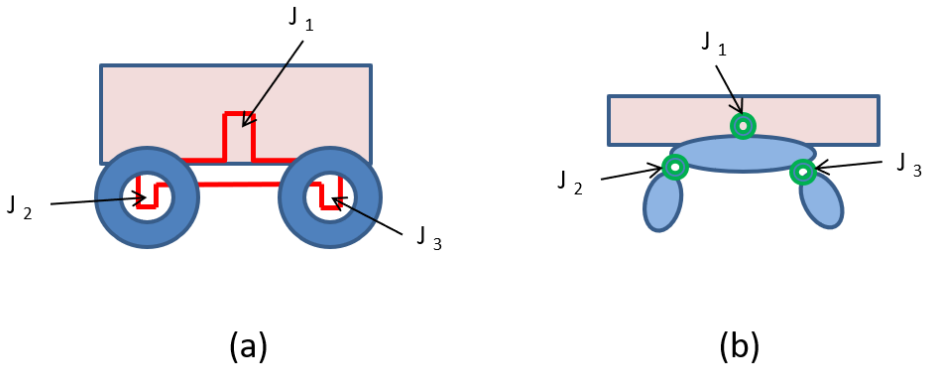


Figure 3.1: Representing a four-wheel rover (a) as a kinematic chain (b).

Given the effectiveness of the approach, the Denavit-Hartenberg convention was selected to be applied to the J5 rover as described in Chapter 1, since the rover could be visualised and analysed as a set of parallel kinematic chains. Figure 3.1 illustrates this process, with 3.1(a) depicting a simplified drawing of a rover and 3.1(b) representing the resulting kinematic chain for analysis. It can be seen that the rover is more akin to a parallel manipulator, with each of the four wheels' ground contact points being equivalent to an end effector. Although a 2D geometric approach was used initially (as detailed in Section 4.2), a 3D approach was deemed necessary with the eventual goal of describing the rover's pose with respect to an inertial world frame attached to a terrain map. With four end effectors and three dimensions, the problem becomes more complex than typical, wrist-partitioned, serial manipulators, which are well documented and for which many of the equation sets already exist [64, 65]. However as noted in the literature survey (Chapter 2), most rover kinematics and dynamics are not investigated using this method and tend to use expensive multibody software to analyze them. The Denavit-

Hartenberg method is not limited to a specific software; it only requires an initial setup based on the original D-H convention outlined below.

It should be noted that there are many variants of the original D-H method, such as the modified D-H method found in [65]. Each incorporates the same four parameters to describe the system with only slight differences in procedure and notation [64], resulting in a slightly different form of equations and transformation matrices. The method with mixed indices [65] is well-suited to velocity and acceleration propagation outward from the base to the end effector, and to force and torque propagation from the end effector back to the base.

3.2 D-H Procedure and Definition of Parameters

As noted, the original D-H convention [66] was selected for use in this work. Again, it is worth noting the underlying assumptions of this application. The original D-H method (referred to hereafter as D-H) was applied to each of the kinematic chains of the rover. Each kinematic chain was assumed to be a series of links beginning from a designated base reference frame, connected by joints with reference frames, to the final reference frame of the end effector: the wheel-ground contact point. An important assumption made is that the links are assumed to be rigid bodies and thereby experience no deformation [64, 65]. Additionally, each joint in the analysis possesses one degree of freedom [64, 65]. Each kinematic pair is one of two types of joints: revolute or prismatic.

For example, a more complicated joint such as a spherical joint, would be broken down and represented as three revolute joints with mutually orthogonal axes [65]. Revolute joints rotate about a single axis, like a hinge, whereas prismatic joints translate in a single direction, like a telescopic pole.

The first and most important step in applying the D-H convention is the assignment of the joint reference frames, from which the parameters are defined [64]. A sketch of the system of links is required. Start by identifying all the joint axes and labelling them as the z axis [64]. For prismatic joints, the z axis is in the direction of translation, whereas the joint axis for revolute joints is the axis of rotation. Figure 3.2 illustrates the assignments for three sequential, revolute joints.

It is to be seen that for the original D-H method, that there is an offset between the reference frame labelled with subscript i and the joint on which it is, which the joint is labelled $i + 1$ [64]. Looking at joint axes i and $i + 1$, identify the common perpendicular between the two axes and at the intersection of the $i + 1$ joint, is the location of the origin for that reference frame, or often noted as O_i. As such, the z_i axis will be along the joint axis for joint $i + 1$. Next, the direction of the x_i axis is assigned so it points along the common normal between the joint axes, and if the joint axes are parallel, then the x_i can remain aligned with the previous x axis. Its axis is a line at infinity, perpendicular to the direction of the p-pair. Finally, the y_i axis is selected to complete a right-hand coordinate system.

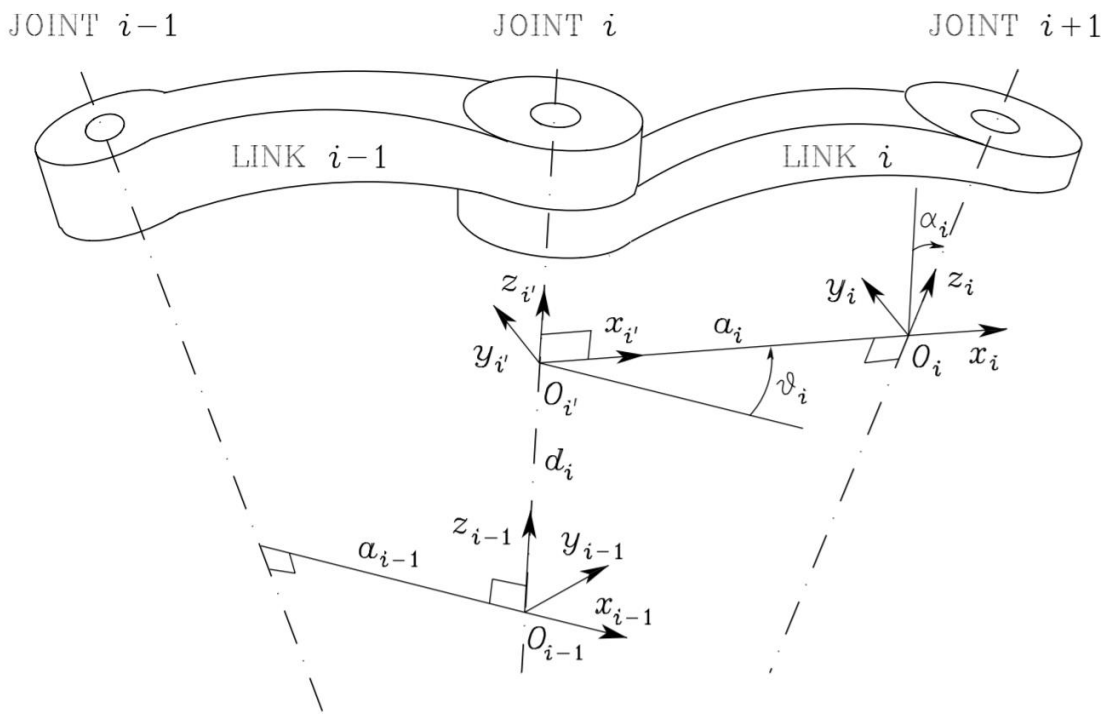


Figure 3.2: Reference frame assignment and D-H parameter definition [64].

Following the correct assignment of reference frames, the four D-H parameters can be determined as follows, see Figure 3.2. The first parameter is the link offset, d_i, which refers to the measured distance along the z_{i-1} axis from x_{i-1} to x_i. For prismatic joints, the link offset becomes the joint variable [64]. The next parameter is the joint angle, which is the variable θ_i. The joint angle is measured about the axis of rotation, z_{i-1}, and is measured from x_{i-1} to x_i. The remaining two parameters are constant values for the given link. The link twist, denoted by α_i, describes the twist angle between the two joints, measured about the x_i axis from z_{i-1} to z_i. Finally, the link length a_i, is the distance between the two joints (between z_{i-1} and z_i) measured along the x_i axis. The four parameters for each joint are collected in a table. Table 3.1 lists the D-H parameters

for each kinematic chain and is shown below with D-H parameters as illustrated in Figure 3.2. It should be noted that the additional column for coordinate frame is added to permit a sketch of the reference frame for ease in orientation. In addition, the joint column differs from the first coordinate frame column due to the notation of the reference frames in the original D-H method, wherein the integer (label) is offset by 1. The set of completed D-H tables with values for the J5 rover are provided in Appendix C.

Table 3.1: Sample D-H parameter table.

Joint	**Coordinate Frame**		**θ_i [deg]**	**d_i [m]**	**a_i [m]**	**α_i [deg]**
$i-1$		$i-1$	θ_{i-1}	d_i	a_{i-1}	α_{i-1}
i		i	θ_i	0	a_i	α_i

3.3 Homogeneous Transformation Matrices

Once the four D-H parameters have been determined, the next step in obtaining the kinematic relationships between the reference frames, and thereby joints, is obtaining the transformation matrix, T that maps point coordinates in frame i to those in frame $i-1$. In order to describe the end effector with respect to the base frame, a transformation must be sequentially concatenated from the end effector back to the base frame. One of

the main benefits of using the D-H convention, is that rather than having to apply individual rotations and translations to get from one reference frame to another, the setup and determination of D-H parameters allows for the transformation to be defined by a single homogeneous transformation matrix as in Equation 3.1.

$$T_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i\, c\alpha_i & s\theta_i\, s\alpha_i & a_i\, c\theta_i \\ s\theta_i & c\theta_i\, c\alpha_i & -c\theta_i\, s\alpha_i & a_i\, s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

From Equation 3.1, it can be observed that once the D-H parameters have been determined, the transformation is obtained by simply populating the matrix with the appropriate values from the table. Note that "*c*" and "*s*" are shorthand notation for cosine and sine, respectively. It should be noted that a transformation matrix will need to be defined for each of the individual links and will contain a single joint variable (either d for prismatic joint or Θ for revolute) [64, 65]. For a kinematic chain comprised of n links, the end effector can be described in the base reference frame by applying a series of n transformations. Again, due to the nature of the convention, once the D-H parameters are all known, one can easily obtain the overall transformation matrix by multiplying the transformation matrices for each of the links as in Equation 3.2.

$$T_n^0 = T_1^0 T_2^1 \ldots T_n^{n-1} \tag{3.2}$$

With the overall transformation matrix determined, one can either solve the forward or inverse kinematics problem as desired, depending upon the selected scenario or what information is known. The exact application of the original D-H convention for the work presented is further discussed in Chapter 4 for the kinematics along with the presentation of the dynamic model in Chapter 5.

Chapter 4: Kinematic Analysis

To understand how the terrain will interact with the rover, it is necessary to first analyse the kinematics of the rover. The kinematic analysis not only determines the pose of the rover based on the shape of the terrain but is then expanded to provide insight into the motion of the rover as it traverses the selected path. Once the analysis includes the velocity of the rover and its respective components, terrain effects such as slip can be introduced for examination. As mentioned in Chapter 3, the analysis centres around the use of D-H parameters to obtain the relative displacements between two joints representative of any motion applied. In this chapter, a brief investigation into using a geometric planar approach is presented, prior to the full three-dimensional kinematics analyses for both position and velocity.

4.1 Planar Kinematic Analysis – A Geometric Approach

During the initial stages of the investigation, a velocity based kinematic approach by Parakh et al [51] was found to be of value. The method employs a planar approach and had been applied to a six-wheel rover with a rocker-bogie linkage system, namely the Rocky 7. The rocker-bogie system is what has typically been employed on planetary exploration rovers, due to its ability to navigate obstacles that the wheels alone could not handle as well [18, 25]. The basis of the model presented by Parakh et al. is focussed on the concept of rigid links and the coordinates of the ends of each link. To confirm the

correct application of the model, the method was first applied to reproduce the results of their six-wheel rover. Sample results are presented in Appendix B.

4.1.1 Theoretical Formulation

This method focusses on the side of the rover and the pose of the links in the rocker-bogie suspension system with respect to the wheels and the shape of the terrain. As such, it is a planar method, and centres around the rigid body assumption, with x and y coordinates assigned to each of the pivot points, or joints of the system, in addition to the wheel ground contact points.

Assumption 1*: The rover is a rigid body and any parameters such as link lengths and relative joint locations can be considered rigid and therefore constant.*
In addition, certain distances, such as the distance between the front and rear wheel centres, are constant.

To begin development of the model, the rover must be projected into a simplified version of its side view. Figure 4.1 shows the simplified geometry of the right side of the Argo J5 rover analysed.

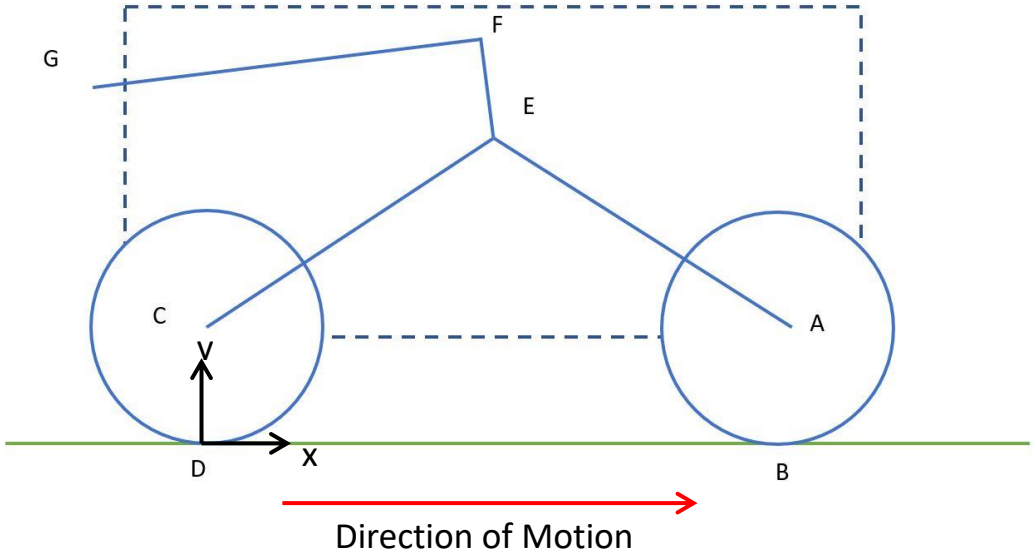


Figure 4.1: Simplified right-side profile of the Argo J5 rover with highlighted joints and contact points.

Figure 4.1 only includes the rigid links and wheels, with the broken line representing the outline of the chassis for clarity. Line FG is the side rod of the back suspension, with point F being the location of the back suspension in the walking beam. Point E is the revolute joint between the chassis and the walking beam. Each end or joint of every linkage is highlighted and labelled. These points are then assigned a pair of coordinates (x,y) to locate them in the plane.

Once all coordinates have been allocated, it can be observed from Figure 4.1 that the linkages provide rigid distances connecting some of these points. Furthermore, due to the geometry of the rocker-bogie suspension system, there are also rigid distances between other pivot points. Figure 4.2 depicts these other set distances with a broken line.

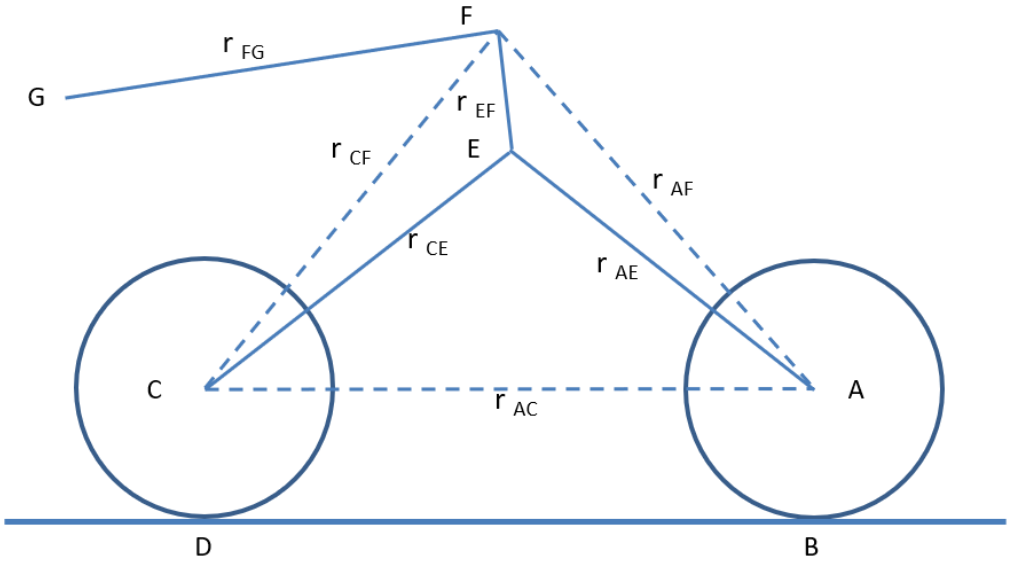


Figure 4.2: Right-side profile with labelled pivot points and rigid distances between pivot points.

Using the x,y coordinate pairs of each point, equations can be developed to determine each of the distances or rigid lengths between various coordinate pairs. The distances can easily be determined in this grid system, using the Euclidean distance between two points or Pythagorean theorem with each rigid length as the hypotenuse. Applying this method to the J5 rover, yields the following set of equations:

$$
\begin{aligned}
(x_A - x_B)^2 + (y_A - y_B)^2 &= r_{AB}^2 \\
(x_C - x_D)^2 + (y_C - y_D)^2 &= r_{CD}^2 \\
(x_A - x_C)^2 + (y_A - y_C)^2 &= r_{AC}^2 \\
(x_A - x_E)^2 + (y_A - y_E)^2 &= r_{AE}^2 \\
(x_C - x_E)^2 + (y_C - y_E)^2 &= r_{CE}^2 \\
(x_C - x_F)^2 + (y_C - y_F)^2 &= r_{CF}^2 \\
(x_E - x_F)^2 + (y_E - y_F)^2 &= r_{EF}^2 \\
(x_F - x_G)^2 + (y_F - y_G)^2 &= r_{FG}^2 \\
(x_A - x_F)^2 + (y_A - y_F)^2 &= r_{AF}^2
\end{aligned}
\tag{4.1}
$$

When reviewing the equation set it becomes apparent that for seven points or coordinate pairs (fourteen unknowns), there are only nine equations. To obtain enough equations to yield a unique solution, the wheel-ground contact points are analysed to produce more relations.

Assumption 2*: Each wheel has a single point of contact with the ground and all wheels remain in contact with the terrain throughout the traverse.*

For the planar model, the rover is constrained to follow the terrain and cannot "pop a wheelie". This constraint also means that the coordinates of the wheel's contact point must match the value of the terrain, more specifically that the y-coordinate of the contact point will match or be identical to the y-value of the terrain function.

Thus, the following additional two equations (4.2) are produced:

$$y_B - f(x_B) = 0$$
$$y_D - f(x_D) = 0 \quad (4.2)$$

Furthermore, the wheel ground contact points can be analysed with respect to the slope of the terrain it is on, as depicted in Figure 4.3.

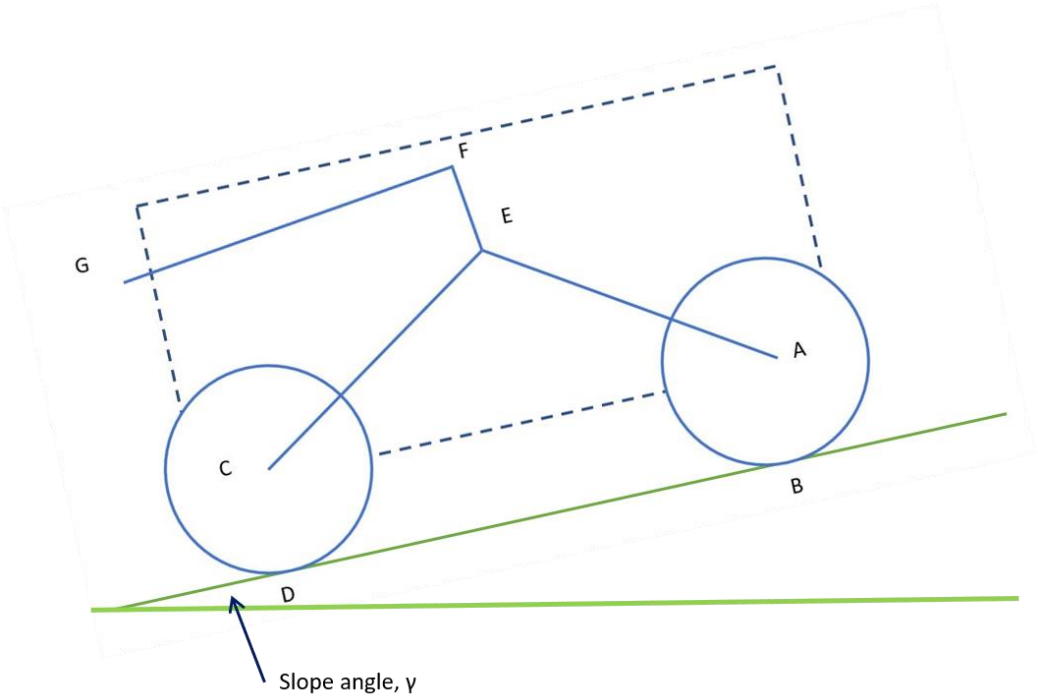


Figure 4.3: Right-side profile of rover with sloping terrain.

For each wheel another equation can be produced to describe the relation between the wheel axle and wheel-ground contact point coordinate values based on the effect of the

slope of the terrain. Using the geometry shown in Figure 4.3, the following equations were developed for each wheel:

$$m_2(y_A - y_B) + (x_A - x_B) = 0$$
$$m_4(y_C - y_D) + (x_C - x_D) = 0 \quad (4.3)$$

At this point, there are now 13 equations and 14 unknowns. In order to make the system solvable, one coordinate point is selected as an input that is specified and used to displace the rover. For this model, it was chosen to use the x variable of the back-right wheel as the input value. Using the x variable as an input yields Equation Set 4.4:

$$(x_A - x_B)^2 + (y_A - y_B)^2 - r_{AB}^2 = 0$$
$$(x_C - x_D)^2 + (y_C - y_D)^2 - r_{CD}^2 = 0$$
$$(x_A - x_C)^2 + (y_A - y_C)^2 - r_{AC}^2 = 0$$
$$(x_A - x_E)^2 + (y_A - y_E)^2 - r_{AE}^2 = 0$$
$$(x_C - x_E)^2 + (y_C - y_E)^2 - r_{CE}^2 = 0$$
$$(x_C - x_F)^2 + (y_C - y_F)^2 - r_{CF}^2 = 0$$
$$(x_E - x_F)^2 + (y_E - y_F)^2 - r_{EF}^2 = 0 \quad (4.4)$$
$$(x_F - x_G)^2 + (y_F - y_G)^2 - r_{FG}^2 = 0$$
$$(x_A - x_F)^2 + (y_A - y_F)^2 - r_{AF}^2 = 0$$
$$y_B - f(x_B) = 0$$

$$y_D - f(x_D) = 0$$

$$m_2(y_A - y_B) + (x_A - x_B) = 0$$

$$m_4(y_C - y_D) + (x_C - x_D) = 0$$

A few other important assumptions were made in the development of the planar model.

Assumption 3*: Distances between the different points of interest are correct and accurate, with lengths or distances not belonging to an actual link accurately determined from the observed geometry.*

Access was not provided to the actual CAD model and dimensions were hard to accurately measure from the physical rover. However, measurements were taken as a rough approximation and these particular distances were measured from a scaled drawing (ref. Figure 4.9 and Appendix A).

Assumption 4*: The left and right sides of the rover experience identical terrain geometry, and subsequently the same motion and displacement.*

Due to the planar nature of the model, the right and the left side are assumed to experience the same displacements due to the same terrain inputs. This situation could have been avoided with an expression for the chassis pitch based on the pitch of the two walking beams; however, it was not used for this model similar to the original paper by Parakh et al [51]. Treating the left and right separately would also have entailed determination of rover roll and yaw angles which exist in three-dimensions, beyond the scope of the planar kinematic analysis.

4.1.2 Method of Solution

From Equation Set 4.4 in the previous Section (4.1.1), it becomes apparent that the solution will be the set of the coordinates for each point, since the links and distances are by the model's definition rigid and therefore constant and have also been calculated. Since the first objective is to pre-determine the pose of the rover with respect to a set terrain path, the terrain function must be known and supplied as the input variable of the model. With a total of thirteen equations, an equation solver is needed to efficiently converge at a solution. Due to the nonlinearity of the equations, a nonlinear multivariate solver was required. The selection of the x coordinate of the rear right wheel axle as an input resulted in a system of thirteen equations and thirteen unknowns, a determinate system, which means a square Jacobian can be obtained, opening up the selection of solvers. Due to its ability to converge fairly quickly and reliably for a set on nonlinear, multivariate equations, the nonlinear multivariate Newton-Raphson method was selected. The author wrote a non-linear, multi-variate, Newton-Raphson solver (included in Appendix B) in order to solve the set of position equations. The nonlinear multivariate Newton-Raphson method can be summarised by Equation 4.5, where J_f^{-1} is the inverse of the Jacobian, x is an array of the variables to be solved for, and f(x) is the function evaluated at x.

$$\boldsymbol{F = x - J_f^{-1} f(x)} \tag{4.5}$$

A separate function file was written to perform the nonlinear multivariate Newton Raphson solution and the function file can be viewed in Appendix B, along with the rest of the code and analysis.

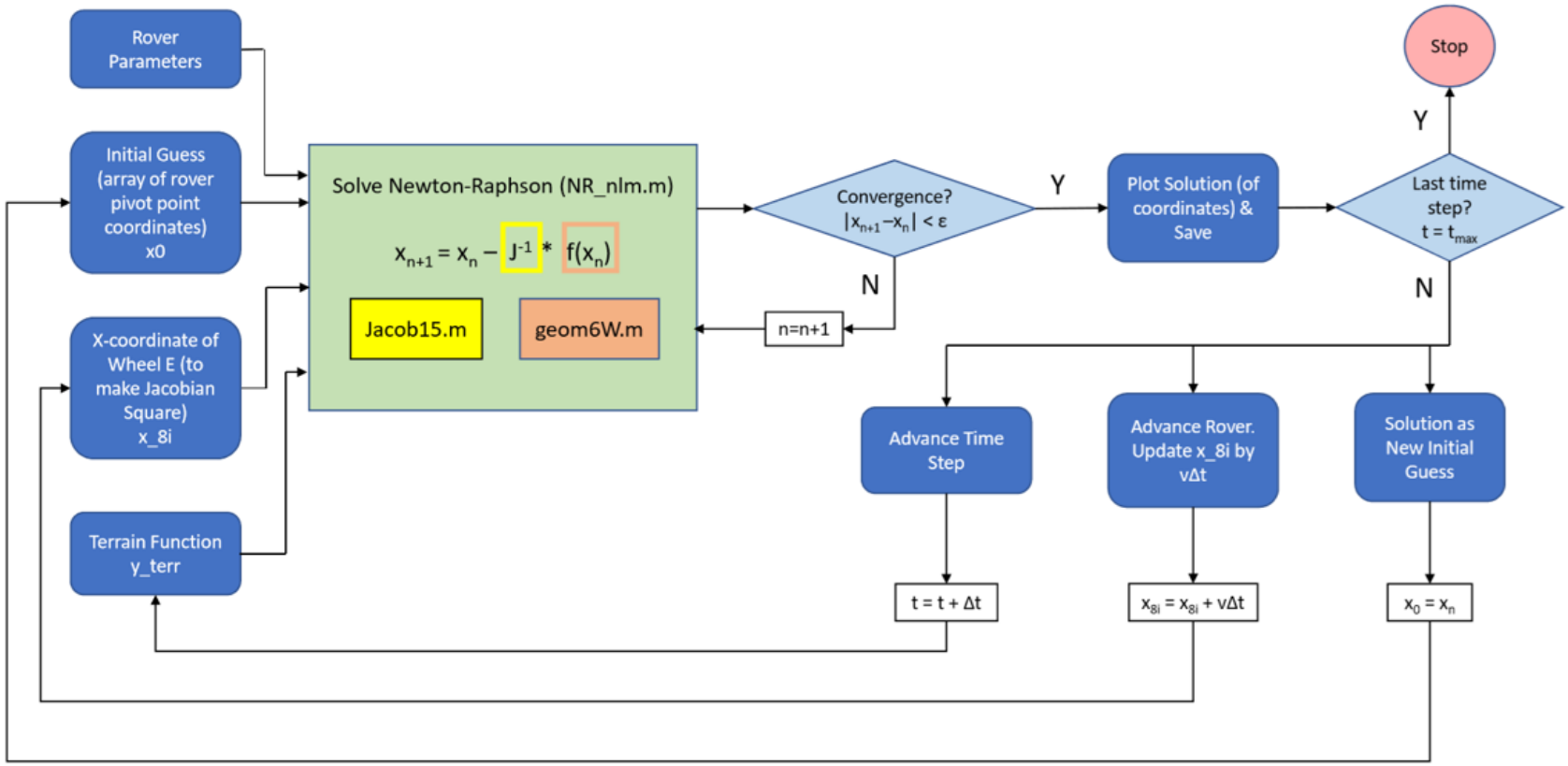


Figure 4.4: Planar position kinematic model code architecture.

MATLAB was chosen largely because of its compatibility with other programs and its ability to be easily translated to other programming languages. The algorithm shown in Figure 4.4 was implemented in MATLAB, with complete script files included in Appendix B. The algorithm takes an initial guess input by the user, along with rover parameters and the terrain function, uses the set of equations that describe the geometry of the rover, and solves them numerically using the nonlinear multivariate Newton-Raphson function file. The nonlinear multivariate Newton-Raphson function also requires that the particular Jacobian for that rover's equation set be manually determined in order to generate a separate function file to be called by the Newton-Raphson function. The Jacobian function file can also be found in Appendix B. Once the nonlinear multivariate Newton-Raphson solver converges to a solution for the given interval, the solution is then plotted to show the rover's current position. They are also stored to later show the overall progression plot of the coordinates over the full time of the traverse. It should also be noted that the while loop containing the solver is governed by time, with solutions being produced for a given interval. The rover is forced forward by the x coordinate of the right rear wheel based on the time interval, Δt, and the commanded velocity of the rover. A time step (Δt) of 1 second was used.

4.1.3 Results

Prior to deriving the equations and writing the scripts for the J5 rover, it should be noted that the process was first applied to the same Rocky 7 rover used by the original authors, to ensure that the application was correct, and their results could be replicated.

The results compared well with those of the original authors. A selection of those results, along with the algorithm/code developed, are included in Appendix B.

A selection of representative cases was chosen to demonstrate the functionality/capabilities of the method. Others are included in Appendix B. It should be noted that more cases can be generated on request, however it does not achieve the overall goal of this work and is inefficient. It should also be noted that due to the planar aspect of the method, the back suspension had to be removed to avoid introducing the necessary complications of a third dimension, rendering the equation set unsolvable.

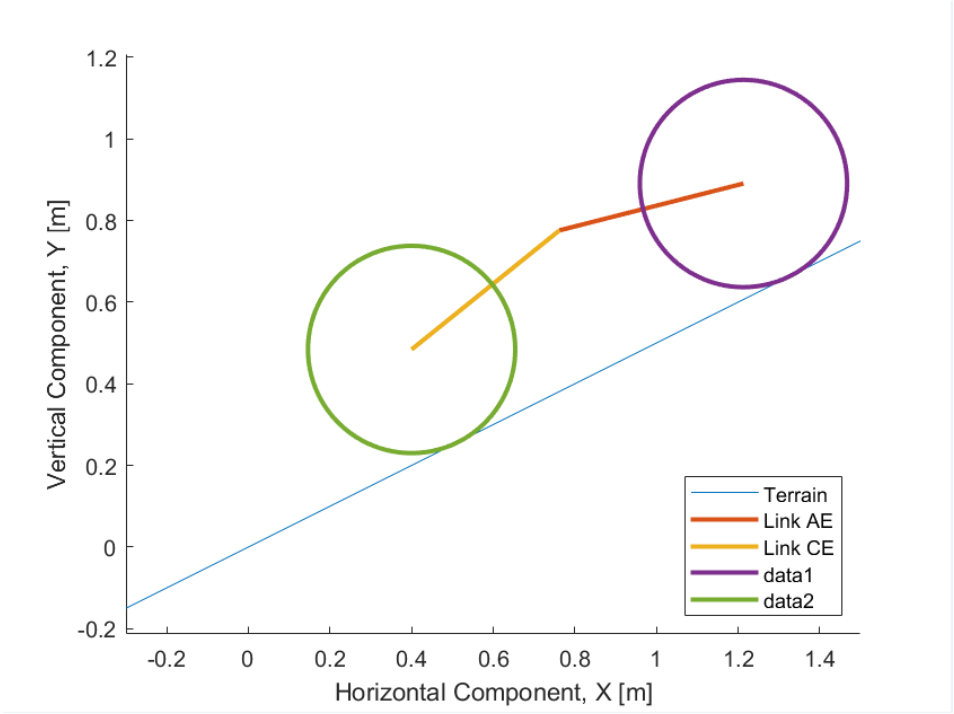


Figure 4.5: Planar inclined pose for a slope of 26.57°.

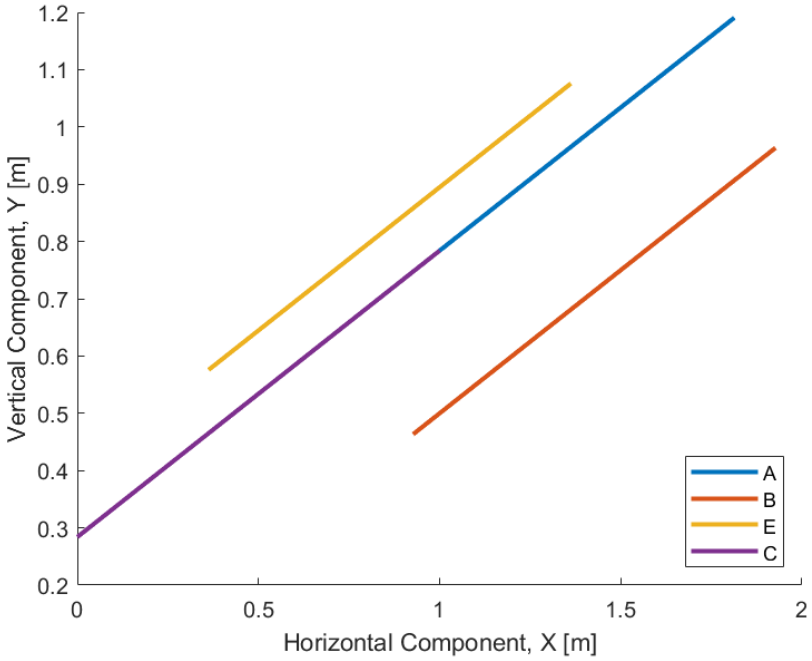


Figure 4.6: Pivot point traces for inclined terrain with slope of 26.57°.

Figures 4.5 and 4.6 detail the results obtained for an inclined terrain, driving uphill. For the uphill test case, the inclined slope is given by a terrain function of y=0.5x, or a slope of 26.57º. The planar pose in Figure 4.5 is generated for each time step and shows the rover conforming to follow the terrain as expected. Figure 4.6 details the movement of the pivot point loci (traces) for the rover as it travels along the terrain. As expected, each of the loci coordinates follows the shape of the terrain, generating lines with a slope of 0.5. Loci A and C represent the wheel axles, hence the rear wheel (C) appears connected to the front wheel (A) since eventually the rear wheel traces the path of the front.

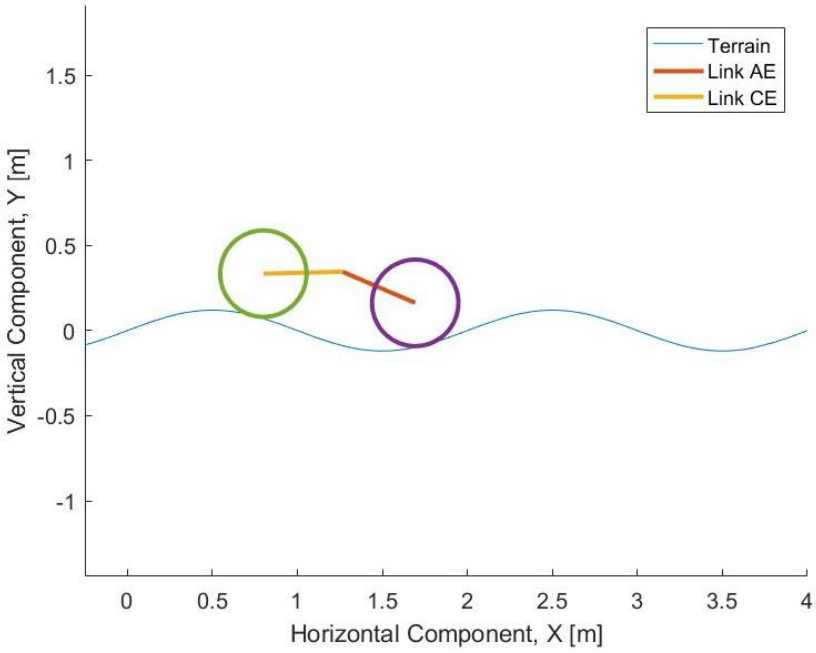


Figure 4.7: Planar pose for a sinusoidal terrain.

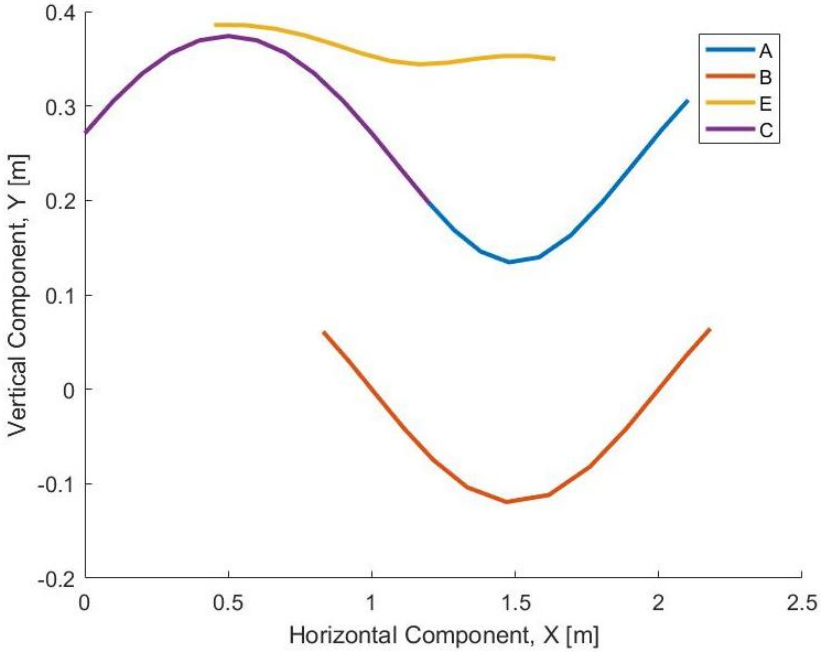


Figure 4.8: Pivot point traces for sinusoidal terrain.

A more complicated terrain function is presented in Figures 4.7 and 4.8. A sinusoidal function was chosen as it is more complex with its change in slope and would thus be closer to reality, where the terrain is not uniform. Once again, the pose was successfully determined for each time step, with the rover wheels in contact with the terrain. Also, as expected, the vehicle pivot point loci (traces) follow the terrain, each mapping out a sine function and the rear wheel following the path of the front. The specific terrain function used to generate these figures was $y=0.12 \sin(\pi x)$. It should also be noted that care must be given to choosing the terrain function, especially for sinusoidal functions, such that the peaks and valleys are not too narrow for the wheel dimensions, which would cause the rover displacement to not follow the terrain.

Overall, the planar method presented here can be used to estimate what the pose of the rover should be as it traverses a given a particular terrain function. However, it is rather limited in that only one side of the rover is examined, and it does not account well for non-smooth and rough terrains. Furthermore, the full three orientation angles of the chassis are not automatically generated and would require extra work to do so. The joint angles are of particular importance since they can be used in evaluating the stability of the current pose of the rover and whether the physical constraints of each joint have been violated.

4.2 Three-dimensional Position Kinematics (D-H Approach)

As noted from the planar kinematic analysis in Section 4.1, a three-dimensional analysis is required to accurately describe the effects of rougher terrain on rover pose. With the need for an accurate description in three dimensions, the geometric method used in 4.1 becomes cumbersome. The use of the Denavit-Hartenberg (D-H) convention described in Chapter 3 allows for determining the pose of the rover at any point in its traverse and to describe this pose in relation to the origin of its path in a world frame. As outlined in Chapter 3, applying the D-H convention allows for transformation matrices to be derived that relate the motion of the rover to the world frame.

4.2.1 Theoretical Formulation

To begin construction of a kinematic model, the particular vehicle must be examined and appropriately modelled. For a kinematic analysis, modeling specifically refers to focusing on the joints which form the kinematic pairs and the rigid lengths or links between these pairs. As introduced in Chapter 1, the rover of interest in this work is the J5.

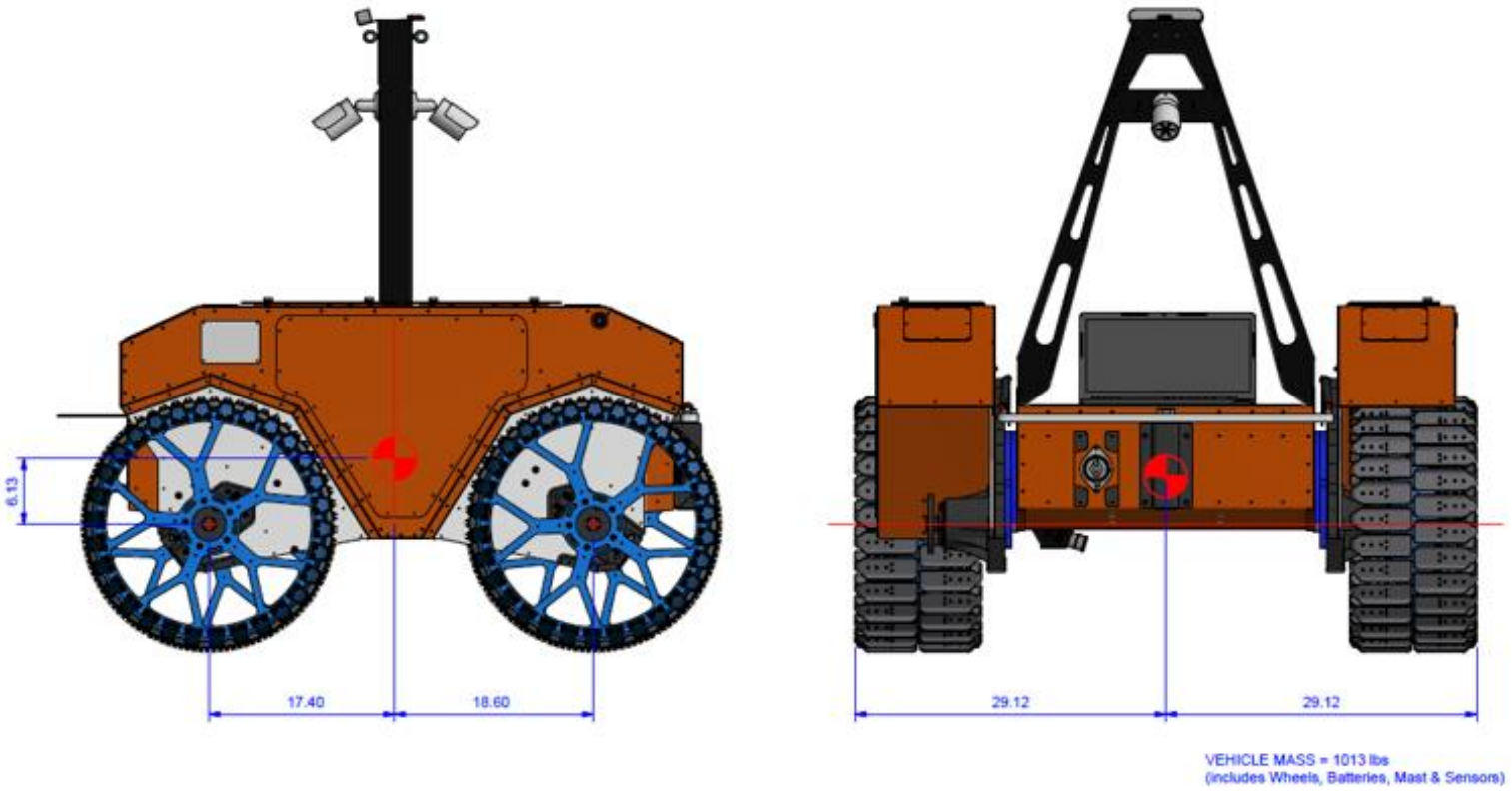


Figure 4.9: Side and rear views of the J5 rover (figure supplied by MCSS).

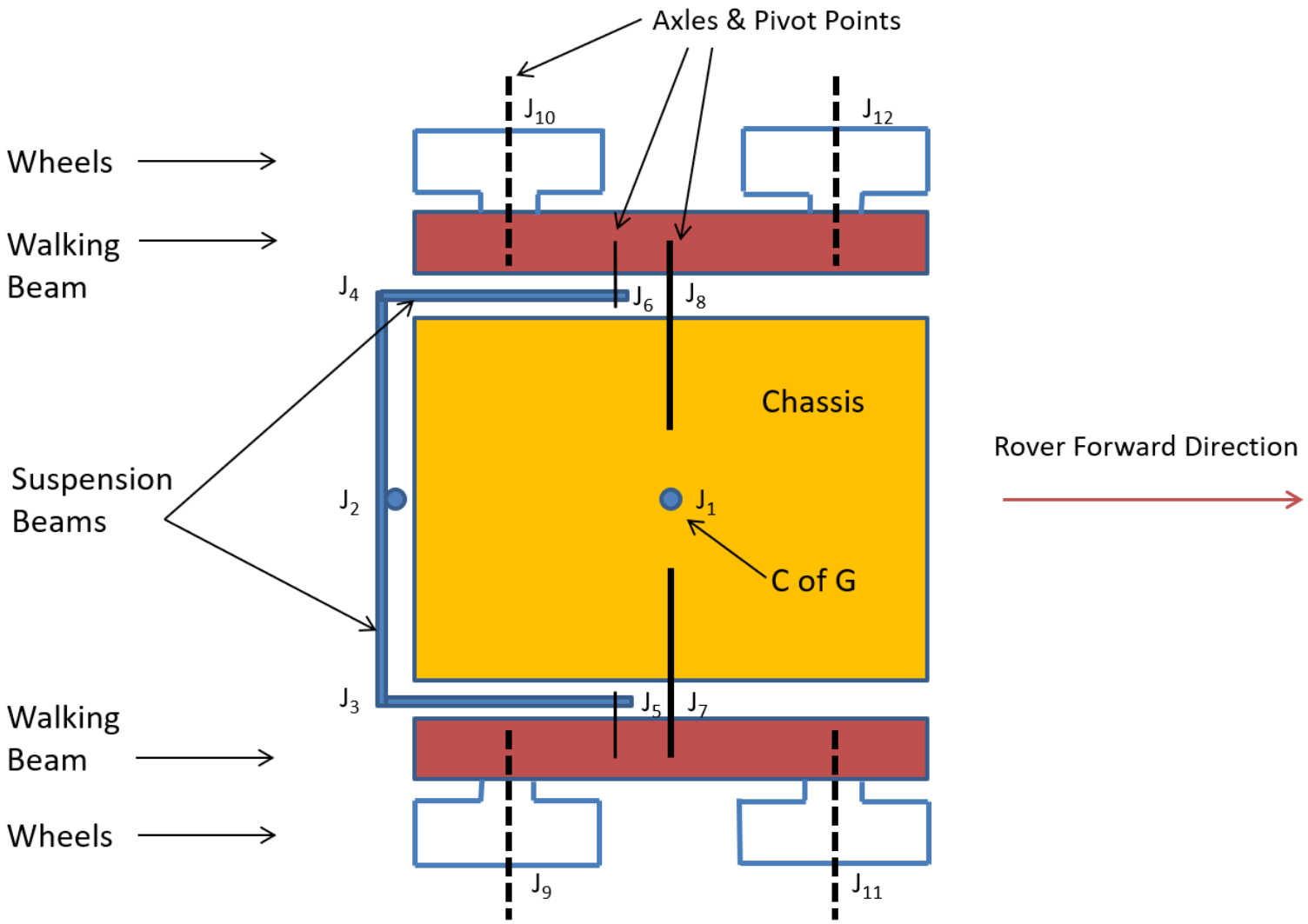


Figure 4.10: Top view of J5 with labelled joints.

From Figures 4.9 and 4.10, it is observed that the rover has a readily identifiable set of linkages in the back suspension, consisting of two side rods and a back bar. The design of the rear suspension constrains the pitch of the chassis with respect to each walking beam, by virtue of the back bar on the chassis being connected to the side rods that interface with each walking beam. The effect is such that it serves to average the pitch of walking beams to obtain the pitch of the chassis and attempt to keep the chassis level. Aside from the obvious linkage suspension system, the rest of the rover can also be discretized into kinematic pairs or joints separated by rigid lengths. These other components include the chassis, walking beams, and the wheels.

With the simplified sketch of Figure 4.10, the joints are easily identified. Starting outwards from J_1, the centre of gravity (C of G), joints are labelled sequentially, with even numbered joints on the left side of the rover and odd numbered on the right. It can further be observed that from the chassis C of G, there are two kinematic paths that can be taken to reach each of the four wheels and their respective ground contact points, either directly through the walking beams (J_7, J_8) or through the back suspension (J_2) to the walking beams (J_5, J_6). Each of the four wheel-ground contact points is analogous to the end effector of a manipulator, effectively meaning that this four-wheel rover is to be modelled as a parallel manipulator. With the establishment of the joints or kinematic (pairs, the links of each kinematic chain are more easily visualised, and one can begin to assign coordinate frames to each joint, along with the end effector and fixed reference point.

Initially, the model included all the joints for the back suspension with a spherical joint (rotates in all 3 directions) connecting the side rods to their respective walking beams. To apply the D-H convention, each of these spherical joints was decomposed into three mutually orthogonal revolute joints. This decomposition made an already complex system even more complex, as each joint adds another row of D-H parameters and subsequently another transformation matrix to be concatenated in the set from rover C of G to ground. Early attempts at solving the sets of equations developed proved to yield no results and the decision was made to simplify the back-suspension system by relating the pitch of the chassis to the two walking beams. The simplification of the back-suspension and its associated joints simplifies the rover; however, the remaining system is still effectively a parallel manipulator.

With the rover model simplified, the body can essentially be stripped away and represented purely by the individual joints and relative locations. By focusing solely on the joints, the coordinate frames are properly assigned following the rules of the D-H convention outlined in Chapter 3. Figure 4.11 depicts the final version of the rover model which forms the foundation of this kinematic analysis.

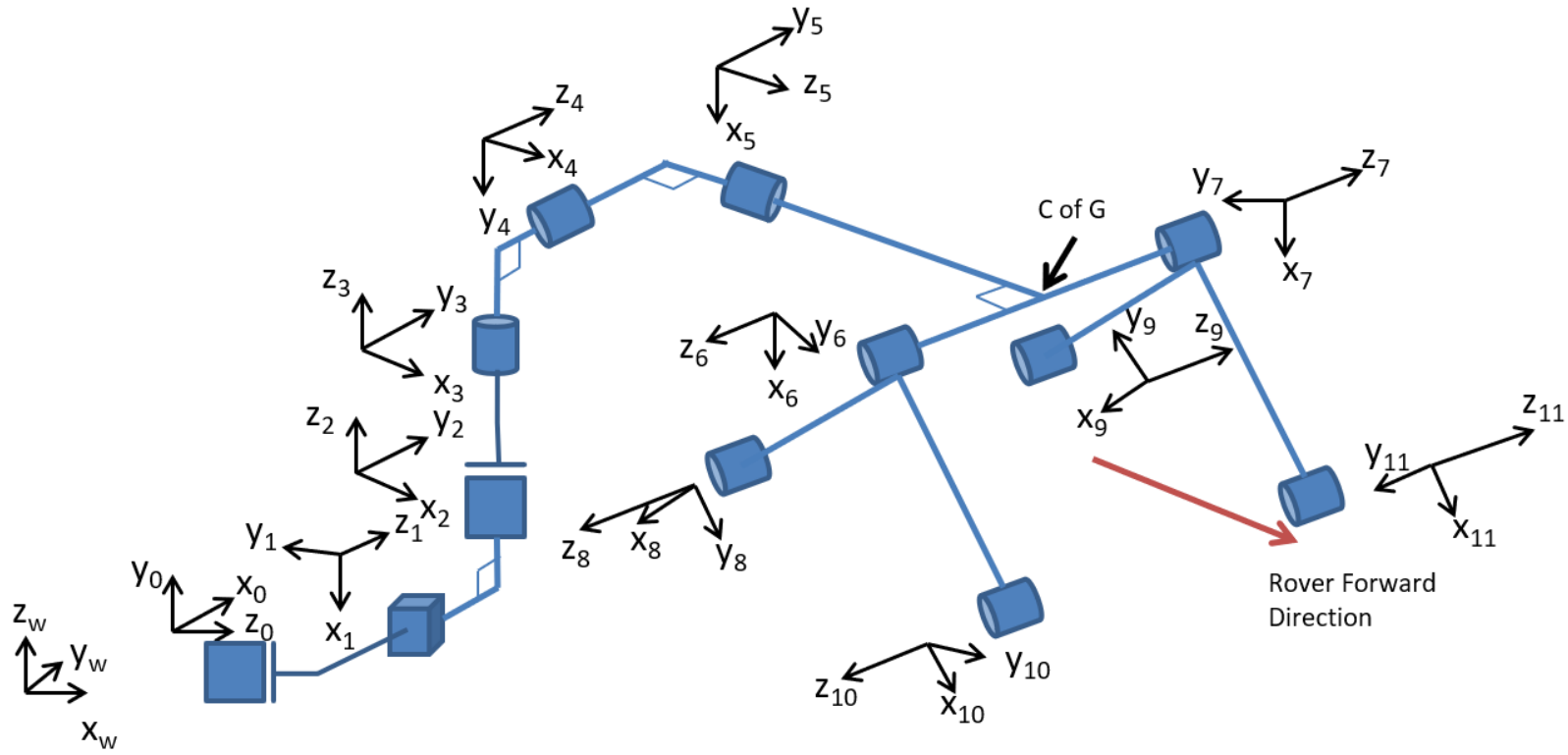


Figure 4.11: J5 joints and assigned coordinate frames.

Examination of Figure 4.11 reveals more joints than present in the physical rover, particularly the three prismatic joints (indicated by cubes) followed by three revolute joints (cylinders). In order to describe the movement of the rover, or more specifically the pose of the rover with respect to its environment, two concurrent three degree of freedom joints were added to connect the rover to the world origin or reference frame. Although these are not physical joints on the actual rover, their inclusion allows the rover's position to be tracked as it moves further from its starting position with the C of G above the world origin. Having these three degree of freedom joints is also advantageous as it means that by "actuating the joint" to get to each set of contact points on the terrain map, the model is closer to driving the rover over the terrain rather than moving the terrain under a fixed rover, like a velocity table. Similar to a spherical joint, a three degree of freedom joint must be broken down into a set of concurrent joints comprised of three prismatic components for translation and three revolute components for rotation. Each of the prismatic joints handles the rover heading in one of the three world directions, starting with the x-direction, followed by the y-direction and finally z. With the position of the rover's centre of gravity as expressed in the world frame taken care of, the orientation must follow, hence the three revolute joints. These describe the overall yaw, pitch, and roll of the chassis about its centre of gravity. Following these joints, the centre of the chassis has been reached and two branches split off to describe the left and right sides of the rover. It should be noted that the lines connecting the joints are not drawn to scale and are more to represent how each joint is oriented with respect to the world reference frame and the order of connection. The order, in turn, dictates the concatenation order of the individual transform matrices. Branching off from the centre

of gravity, the kinematic chain can proceed to the left or right side of the rover. Each walking beam can rotate or pitch up and down with respect to the world frame. Following the revolute joint of the walking beam, the kinematic chain branches again with one branch for the front wheel and its contact point, and the other for the rear. Hence, the rover can be modeled as a parallel manipulator with four distinct kinematic paths from the world reference frame to the respective wheel ground contact points.

With all the joints clearly identified as shown in Figure 4.11, the coordinate frames shown were assigned based on the rules outlined in Chapter 3. It can be seen that all joints have the z-axis aligned with their respective joint axis and that each coordinate system has its x-axis such that it is perpendicular to both its own z-axis and the z-axis of the joint preceding it. As such, the x-axis for each wheel axle is at an angle that aligns it with the link length between it and the walking beam pivot point. It is also important to note from Figure 4.11 that the left and right sides of the rover have z-axes pointing in opposite directions. This choice was made to avoid having to assign negative values to rigid distances between some of the links. Similarly, the x-axis of each walking beam joint is in the vertical, downwards or negative global z direction, to take advantage of bisecting the angle between each walking beam link length (shortest distance between the axles and the walking beam pivot point). This advantage will become more apparent as the parameters are discussed.

At this stage in the formulation, it is beneficial to introduce the assumptions made in the three-dimensional position kinematic analysis and discuss the rationale behind them.

Assumption 1: *The rover is a rigid body and any parameters such as link lengths and joint displacements can be considered rigid and therefore constant.*

It is reasonable to assume that structural flexion does not need to be accounted for. In addition, certain distances, such as the distance between the front and rear wheels, are assumed to be constant.

Assumption 2: *Dimensions of the rover are accurately determined from the drawing provided by the rover operators.*

Access was not provided to the actual CAD model and dimensions were hard to accurately measure from the physical rover. However, a drawing of the rover was provided and illustrated measurements of certain dimensions were taken from the drawing and scaled accordingly to obtain other dimensions used throughout the work presented here. The scale used was 15.8:1. The drawing provided is included in Appendix A, along with a list of dimensions and properties of the rover. These dimensions were also used in creating test terrains that would be guaranteed to line up with the wheel-ground contact points.

Assumption 3: *The rover's centre of gravity is in alignment with the walking beam joints.*

From the rover diagram mentioned in assumption 2, the checkered circle symbolising the rover's centre of gravity can be seen to be slightly off the geometric centre, where the joints connecting the chassis to each of the walking beams would be aligned. The relatively small distance (0.01524m) added unnecessary complexity to an already complex problem and did not significantly impact the D-H parameters governing the analysis. The location of the centre of gravity would have more impact in the dynamic analysis, which deals with forces (and by extension: masses and accelerations) whereas kinematics is limited to position and motion.

Assumption 4: *The back-suspension can be replaced and modeled with a function relating the pitch of each walking beam to the overall pitch of the chassis.*

Upon initial derivation of the D-H parameters and consequent transform matrices, it was determined that modelling the back suspension added too much complexity and number of equations to an already overly constrained system. Such excessive complexity and the added number of constraints would be problematic for the software to accommodate. Based on conversation with the rover owner/operator, it was determined that the appropriate function would average the pitch of each walking beam to try and keep the chassis as level as possible. As such, the following equation was developed:

$$\theta_{chas} = \frac{1}{2}(\theta_{wbr} - \theta_{wbl}) \tag{4.6}$$

Where Θ_{wbr} and Θ_{wbl} are the pitch of the right and left walking beams, respectively, with the chassis pitch represented by Θ_{chas}. The negative sign occurs due to the different orientations of the z-axes for each walking beam joint which means that the same pitch will have different directions (signs) in their respective coordinate frames, as governed by the right-hand rule.

Assumption 5: *Each wheel has a single point of contact with the terrain and is located at the same position as the axle plus a radial distance to the wheel rim.*

Each wheel is considered to be more like a disk. Although depending on the type of tire, terrain, and terra-mechanic forces involved, the actual wheel-ground contact is more of a continuum than a single point, for a kinematic analysis. The goal is the overall pose of the rover for the given terrain input and which could be used to determine whether the pose violates any constraints, or results in an unstable configuration. Thus, it was determined to be acceptable to proceed with the simplifying assumption of a single point of contact for each of the four wheels, meaning four input sets from information obtained from the terrain map.

Assumption 6: *The wheel-ground contact point is modeled via Tarokh et-al [48] due to the relatively fixed relationship between the wheel axle and the rim-ground contact point.*

This assumption simplifies the mathematical formulation for the model.

Parameters

With these governing assumptions in mind, the model above with the coordinate frames indicated was used to obtain the four D-H parameters describing the transformation between each kinematic pair. These parameters where collected in tables, with Table 4.1 as an example.

Table 4.1: D-H parameters for the kinematic chain – world origin frame to right front wheel.

n	θ [deg]	α [deg]	a [m]	d [m]
0	+ 90	+ 90	0	0
1	- 90	- 90	0	X_{trans}
2	- 90	+ 90	0	Y_{trans}
3	0	0	0	$Z_{trans} + h_{CoG}$
4	φ_{yaw}	- 90	0	0
5	$\varphi_{pitch} + 90$	+ 90	0	0
6	φ_{roll}	+ 90	0	0
10	$\theta_{wbr} + \beta$	0	a_{wb}	d_{cmwb}

The complete set of D-H parameter tables can be viewed in Appendix C. Note that there are extra rows, one between the world frame and the first joint, and also between the axle initial orientation and the orientation used in the contact transformation matrix. These additions do not describe actual joints but were necessary because an additional rotation was needed for proper alignment. As expected, the three prismatic joints (rows for n=1, 2, 3) have the joint variables representing the overall translation of the rover in the three global directions. The third joint, responsible for the global z

translation of the rover has the constant addition of the height of the rover's C of G included, so as to distinguish movement in the global z direction from the influence of the terrain and not the height of the C of G. The subsequent three revolute joints have joint variables of theta which represent the overall yaw, pitch, and roll of the chassis. The yaw angle would be the heading or steering angle of the rover as it follows the chosen path. For the joint rotation of each walking beam, an additional variable β, is included. The angle β corresponds to half the angle between the two walking beam link lengths (a_{wb}). The walking beam link lengths are the shortest distance between the walking beam joint and the wheel axles. Finally, the last row describing the transformation from the walking beam joint coordinate frame to the wheel axle also has a joint offset parameter whereas the previous row (detailing the transformation from the C of G to the walking beam joint) does not. This joint offset occurs because the coordinate frame for the walking beam joint has to be effectively translated (although still attached to the joint) to ensure it did not violate any of the rules for establishing coordinate frames. Its x-axis was not only perpendicular to its own z-axis and the z-axis of the joint before it, but also that it intersected the previous z-axis. This joint offset is the width or distance from the rover's centre of gravity to the wheel-ground contact point (or end of axle) in the global Y direction.

Equations

The final step in the formulation of the kinematic model is the generation of transform matrices between the kinematic pairs or coordinate frames, which in turn produce the equations describing the motion of an output link relative to the selected

fixed reference frame. Chapter 3 introduced the homogeneous transform matrix and how D-H parameters fit. Returning to Table 4.1, and using the D-H parameters for row 2 which describes the transform from coordinate frames 0 to 1 (or the properties for joint 1), the homogeneous transformation matrix is

$$T_1^0 = \begin{bmatrix} c\theta_1 & -s\theta_1\, c\alpha_1 & s\theta_1\, s\alpha_1 & a_1\, c\theta_1 \\ s\theta_1 & c\theta_1\, c\alpha_1 & -c\theta_1\, s\alpha_1 & a_1\, s\theta_1 \\ 0 & s\alpha_1 & c\alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c(-90) & -s(-90)c(-90) & s(-90)s(-90) & (0)c(-90) \\ s(-90) & c(-90)c(-90) & -c(-90)s(-90) & (0)s(-90) \\ 0 & s(-90) & c(-90) & X_{trans} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & X_{trans} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The full set of homogeneous transformation matrices were produced using Maple but are not included herein.

The paper with the closest approach to this work, Tarokh et al [48], noted that a better approach to the transformation matrix from the wheel axle to the contact point was by applying a simple rotation based on the terrain incline and translations of the contact joint based on that angle.

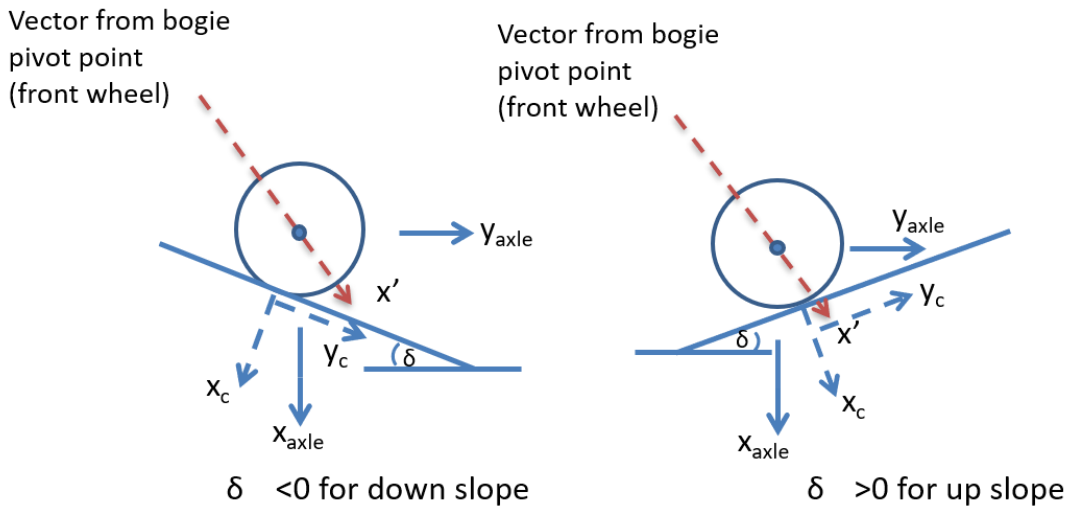


Figure 4.12: Right front wheel contact diagrams (view looking in the axle's negative z-direction).

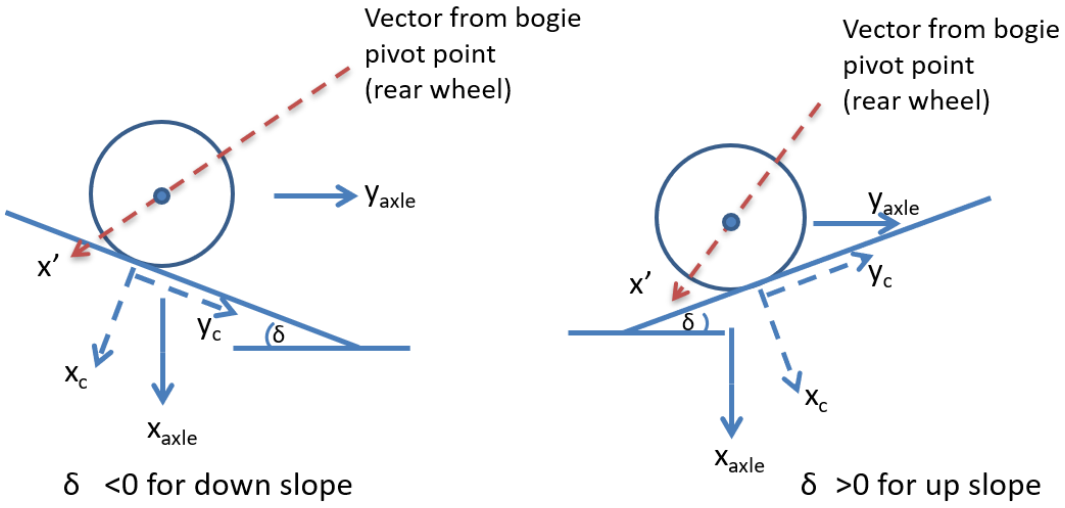


Figure 4.13: Right rear wheel contact diagrams (view looking in the axle's negative z-direction).

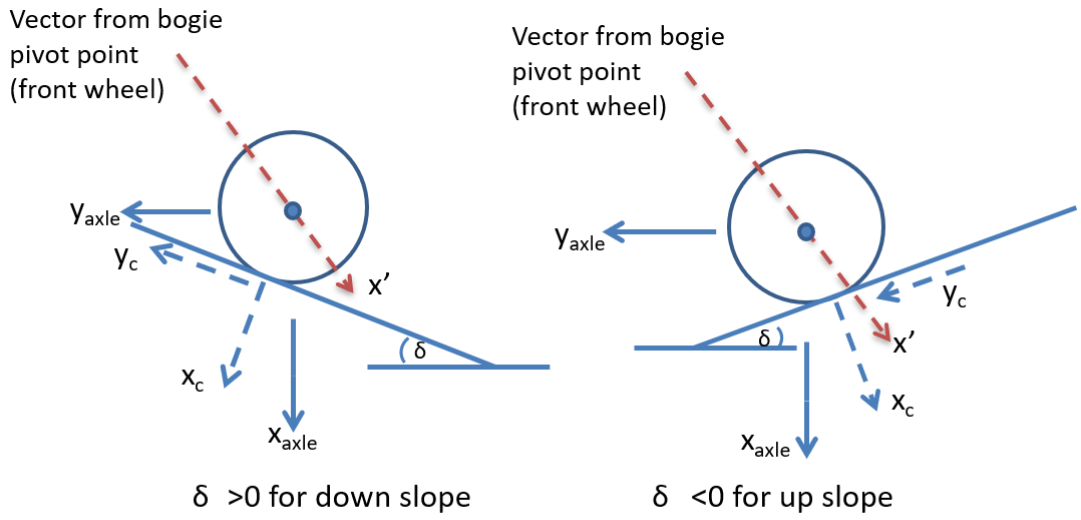


Figure 4.14: Left front wheel contact diagrams (view looking in the axle's positive z-direction).

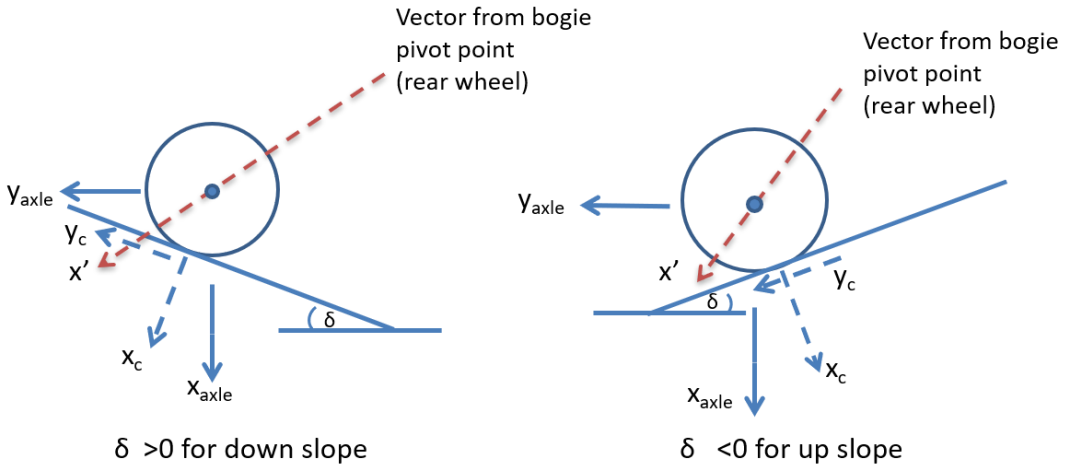


Figure 4.15: Left rear wheel contact diagrams (view looking in the axle's positive z-direction).

Figures 4.12 through 4.15 show the various wheels on a generic incline and its effect on the orientation of the coordinate frame at the contact point with respect to the axle's coordinate frame. The angle of the incline causes the contact point to rotate about the axle's z-axis by that same amount. This rotation results in the x and y location of the contact point shifting by a magnitude of the radius multiplied by the cosine or the sine of the angle respectively. By inspection, it can be seen that the transformation matrix relating the contact point to the wheel axle for the right front wheel is given by Equation 4.8.

$$T_{14}^{10} = \begin{bmatrix} c\delta_{rf} & -s\delta_{rf} & 0 & r\,c\delta_{rf} \\ s\delta_{rf} & c\delta_{rf} & 0 & r\,\delta_{rf} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Note that, in Equation 4.8, the notation on the transformation matrix, T, connects axle joint 10 with its wheel-ground contact point for that wheel, 14. After attempting a purely D-H approach, it was decided to take advantage of this simplification and include it in the model presented here.

Similarly, the overall transformation matrix describing the end effector of wheel-ground contact point with respect to the world frame, is obtained by the concatenation of transformation matrices in Equation 4.9.

$$T_{14}^{W} = T_{0}^{W} T_{1}^{0} T_{2}^{1} T_{3}^{2} T_{4}^{3} T_{5}^{4} T_{6}^{5} T_{10}^{6} T_{10a}^{10} T_{14}^{10a} \tag{4.9}$$

The order of multiplication follows the order of the joints, starting from the world frame and ending at the contact point. The concatenation of matrices for the remaining three kinematic chains (wheels) were also completed using Maple. It was decided to use Maple to obtain the overall transformation matrix for each kinematic chain because it is better built for symbolic computing.

Recalling the definition of the homogeneous transformation matrix, once the overall transformation matrix has been determined for each kinematic chain, equations expressing the pose of the end effector in the world reference can be extracted as follows:

$$\begin{aligned} X_{Ci} &= T_{Ci}^{W}[1,4] \\ Y_{Ci} &= T_{Ci}^{W}[2,4] \\ Z_{Ci} &= T_{Ci}^{W}[3,4] \\ \varphi_{X,Ci} &= \tan^{-1}\left(\frac{T_{Ci}^{W}[3,3]}{T_{Ci}^{W}[3,1]}\right) \\ \varphi_{Y,Ci} &= \sin^{-1}(-T_{Ci}^{W}[3,2]) \\ \varphi_{Z,Ci} &= \tan^{-1}\left(\frac{T_{Ci}^{W}[2,2]}{T_{Ci}^{W}[1,2]}\right) \end{aligned} \tag{4.10}$$

A brief examination of the 4^{th} and 6^{th} equations of Equation Set 4.10, which have denominators, indicates that it is extremely unlikely for the denominators to become zero, given the physical reality of the driving scenario. Once again taking advantage of the symbolic nature of the Maple environment, the resulting equations were obtained for direct implementation in the algorithm for solution. The final result was a set of a maximum of 24 equations to describe the pose of the end effector, with the added Equation 4.6 replacing the back suspension. From the D-H parameter tables it can be seen that there are 12 unknowns or joint variables for the system. Details of the algorithm, including solvers used, are presented in 4.2.2.

4.2.2 Method of Solution

With the derivation of the basic position and orientation (pose) kinematic equations complete, the algorithm to solve the set of equations given the position of each contact point on the terrain map was developed next. Figure 4.16 details the final algorithm.

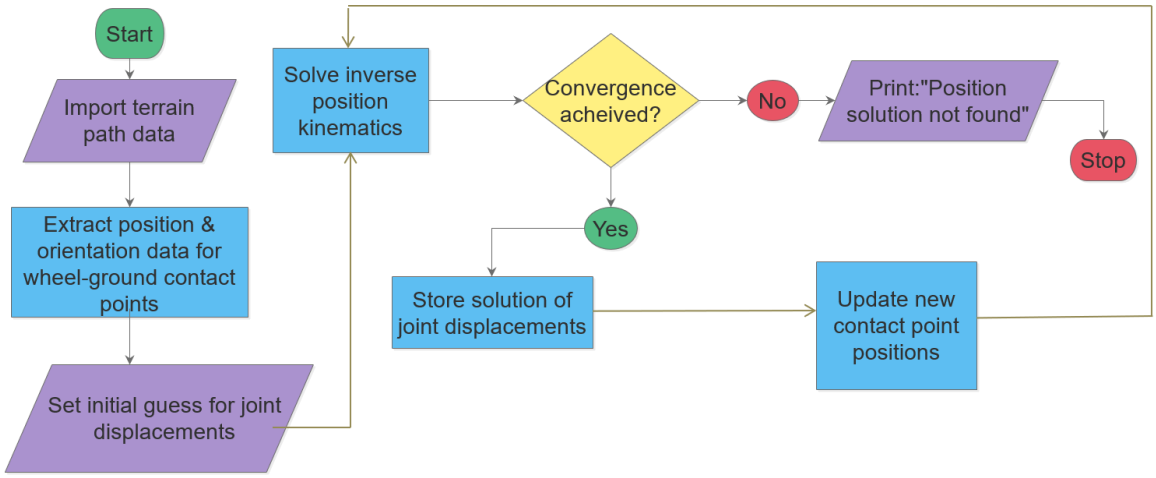


Figure 4.16: 3D position kinematics algorithm.

Since one of the objectives of this work is to predict the rover behaviour prior to executing a particular path, the inputs are designed to be representative of the scenario. As such, the main input to the position kinematic analysis is a terrain path consisting of location in X and Y, with the elevation at each set of coordinates given by Z. Alternatively, a terrain function can be used instead of a terrain map of Z(X,Y), however to try and simulate reality, the elevation terrain map was implemented. The use of a terrain map also allows for more randomness to be included in the terrain than otherwise would be. Furthermore, this choice will prove advantageous later in the velocity analysis, as it allows the terrain map to be populated with estimates for regions of slip. The first step in the algorithm imports the specified terrain path map and extracts arrays of data for the different coordinate directions for each wheel-ground contact point. These extracted arrays will then be used as inputs for the position and orientation of the wheel-ground contact points. It is important to remember that these inputs must all be expressed using the world reference frame.

A “while” loop is used to step through each set of points in the traverse, from beginning to end. With the inputs defined, the external function, J5posKin.m is called for solution. The J5posKin.m function can be thought of as the kinematic model or rover geometry/property file, since it not only contains the extracted kinematic equations but also the physical geometric properties of the particular rover. This function file could easily be swapped out for different rover models without much change to the overall algorithm, making the algorithm more versatile. It should be noted that for most numerical solvers, an initial guess of the solution is required. For simpler terrain test cases, the initial guess can be easily made to fit the actual solution, like that of flat land which can be expected to have a solution of 0 for the joint variables corresponding to the first location on its traversal path. A good initial guess gives the solver a better chance at achieving convergence. For each position iteration of the while loop, the corresponding solution of joint variables are stored for easy plotting of joint motion over the traverse. For example, the angular displacement of each walking beam over the course of the traverse is one such end output of the algorithm. The terrain path is also plotted over the terrain map to provide context and a means for comparison.

The 3D position kinematics algorithm was developed with the goal of being more software independent than other methods, including some of those highlighted in the literature presented in Chapter 2. Thus, the source scripts/code displayed in Appendix C, can easily be modified for other coding platforms and languages. The work presented here was written in MATLAB, which is relatively inexpensive compared to multibody

physics engines and other simulation software. MATLAB was also chosen for its ability to easily interface with other software applications and the familiarity of the author with its functions. The algorithm being relatively independent of the software utilized is advantageous since it means that users have more flexibility in the software they use and are not forced to invest in a software platform that can be costly in terms of both purchase and training costs.

Within MATLAB, many built-in functions are available for solving systems of equations and many of these have a choice of algorithms to use in their solution. This feature makes them more robust and increases the overall chance of convergence. After investigating some of the solvers applicable to nonlinear equations, the solver 'fsolve' was selected. Fsolve was chosen because it not only solves systems of nonlinear equations, but multivariate ones as well. More importantly, one of its potential solution methods is the Levenberg-Marquardt algorithm which can handle systems of equations where the Jacobian fails to be square. The kinematic equation set derived for the J5 rover has a maximum of 25 equations and 12 unknowns, making it an over-constrained system. Even reducing the equation set to the bare minimum still results in 13 equations for 12 unknowns. As such, the Jacobian of this equation set will never be square, hence a solver that can employ a strategy to cope with the non-square Jacobian is required. To use fsolve, the equation set must be in the form of equations that equal zero as seen in the function file, J5posKin3.m, included in Appendix C.

4.2.3 Results

A selection of representative case studies is presented below. Once again, more cases can be generated upon request, with the representative cases demonstrating the capabilities of the method. It should be noted that using the full set of 25 equations with 12 unknowns resulted in convergence issues, as the system is over-constrained. Adjusting settings on the solver did not result in convergence. However, due to the physical nature of the rover, the wheel-ground contact points remain fixed in their orientation relative to each other, with respect to yaw and roll. This relationship justified the removal of some of the orientation equations, which resulted in convergence being achieved.

Case 1: Flat Terrain

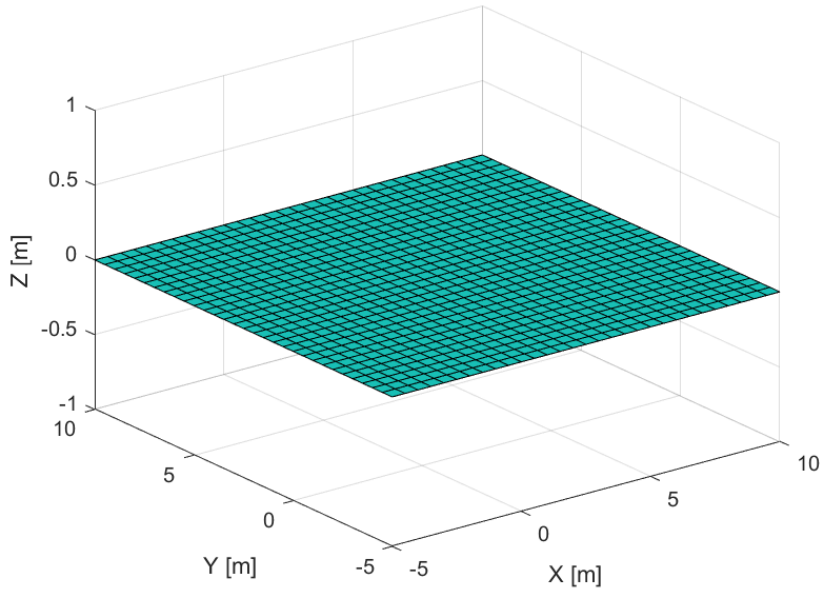


Figure 4.17: Flat terrain digital elevation map.

The first test case was for flat terrain since it was the simplest scenario. Figure 4.17 is the digital elevation map of the flat terrain and the inputted terrain path consisted of x, y, z coordinates for a straight path in the global x-direction, beginning with the chassis centred over the world origin (located at (0, 0, 0)).

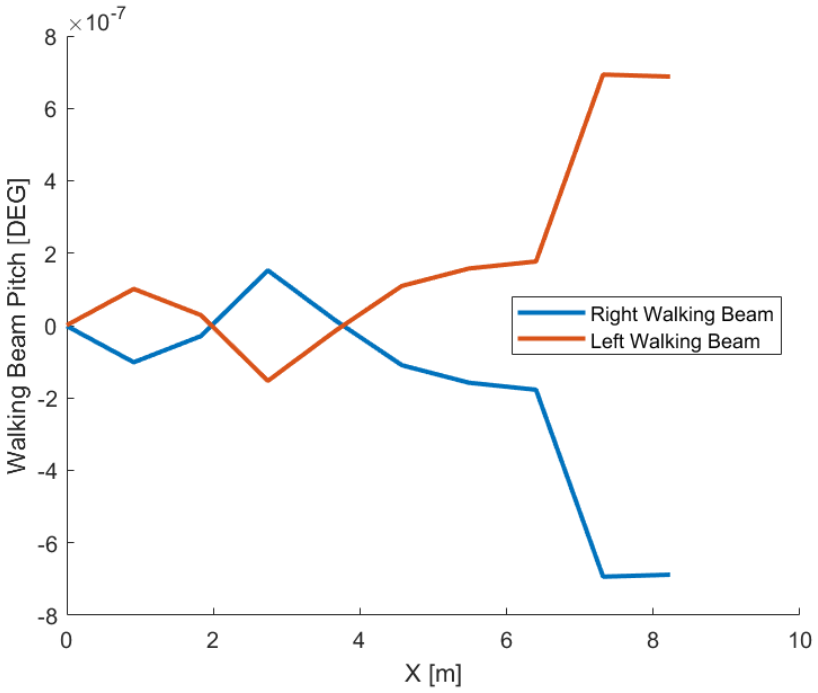


Figure 4.18: Walking beam pitch vs distance travelled (flat terrain).

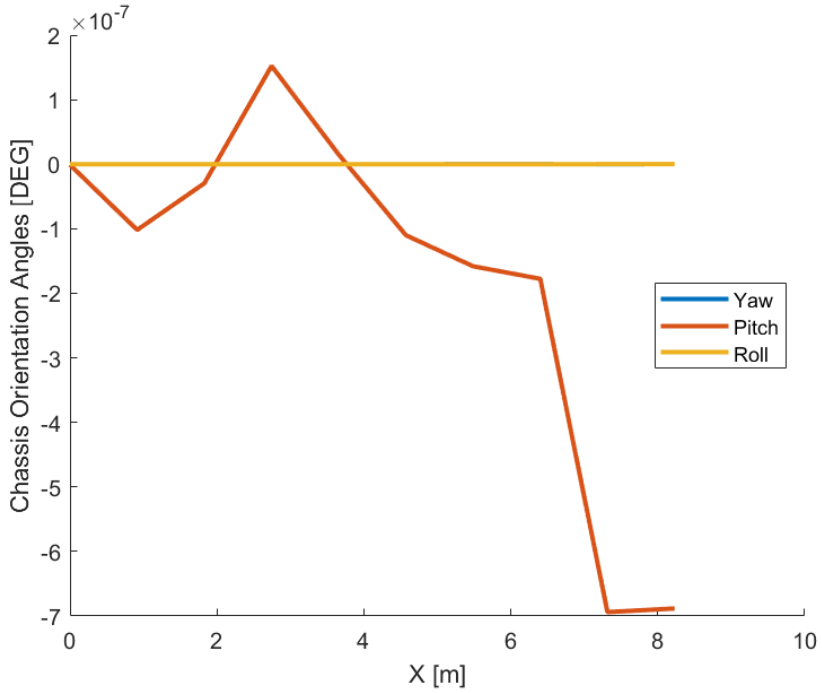


Figure 4.19: Chassis orientation angles with respect to distance travelled (flat terrain).

Figures 4.18 and 4.19 depict the results of the simulated drives. The pitch of each walking beam is plotted over the course of the rover's traverse in Figure 4.18. Examining Figure 4.18, one can observe the proper averaging of the walking beams due to the scale of the y-axis. Furthermore, the value of the y axis shows that, as expected for the flat terrain, the pitch of each walking beam is 0^{o}. Similarly, from Figure 4.19, the yaw, pitch, and roll angles are all 0 for the flat terrain which is to be expected. The scale of the y-axis shows a bit of a varied response on the pitch; however, it is off magnitude 10^{-7} and is thus considered to be 0. In both 4.18 and 4.19, these fluctuations are due to the nature of the solver as it converges toward a solution.

Case 2: Uphill Sloped Terrain

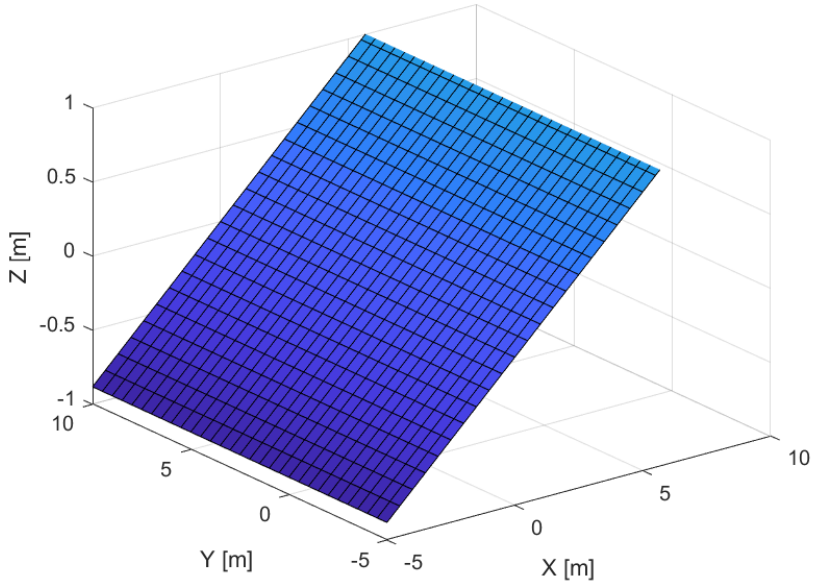


Figure 4.20: Uphill 10° inclined terrain digital elevation map.

The next test case was for moving up a sloped terrain. The slope of the terrain shown in Figure 4.20 and was assigned a slope value of 10°. The corresponding terrain path coordinates for each wheel driving in a straight line in the x-direction were generated in an Excel sheet and imported as the inputs. Subsequently, it was expected that the pitch of the walking beams and chassis would have the same value as the slope.

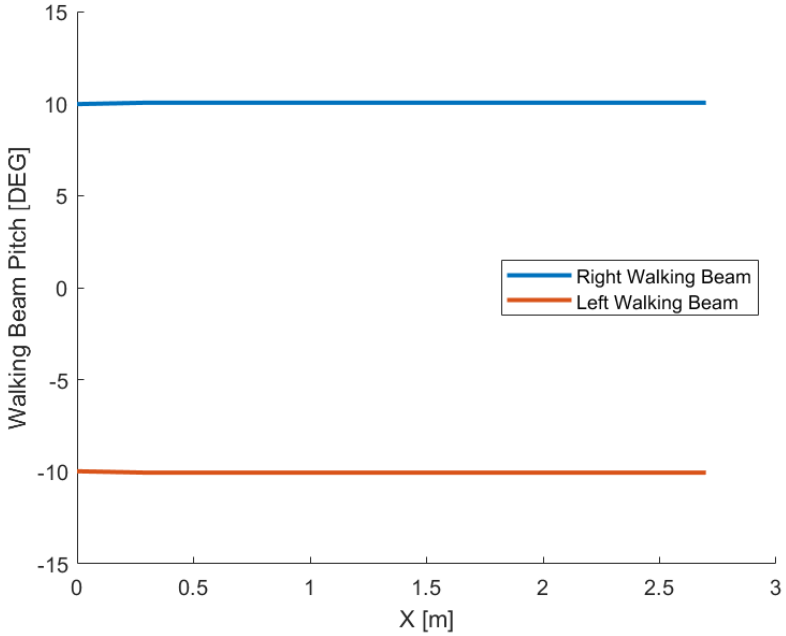


Figure 4.21: Walking beam pitch vs distance travelled (10º incline).

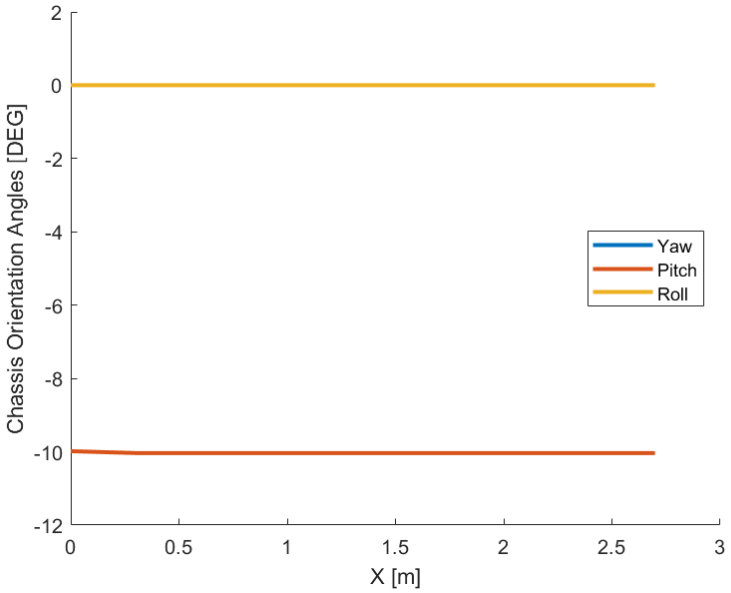


Figure 4.22: Chassis orientation angles with respect to distance travelled (10º incline).

Figures 4.21 and 4.22 detail the results for the simulated drive of a straight path on an inclined slope of 10º. It was found that for a longitudinal slope that although convergence was achieved, all pitch angles were computed and were off by half of what they should be. The apparent error was investigated through further review of the computer code, along with a complete reformulation of the overall transformation matrix by trying different order combinations for the first 6 joints. Note that the order of later joints in the sequence could not be altered as they are specified by the geometry of the rover. The application of the original D-H method in assigning coordinate frames and determining the associated parameters, was revisited multiple times to ensure correct formulation of the individual transformation matrices. Maple was used in the concatenation of all the transformation matrices to avoid the possibility of human error in matrix multiplication. The results of the investigation indicated that the various formulations all produced the same error. Therefore, due to the consistency of this offset/error, as also verified in a 15º incline test case, a correction factor of 2 was applied to the pitches. It is important that this correction be applied at this point in the model because these pitch angles are used later as inputs to the velocity analysis. The resultant walking beam pitch angles over the traversed distance are shown in Figure 4.21 and, as expected, they remain constant like the slope of the terrain, each with a corrected value of 10º. Furthermore, the walking beams continue to display the correct behaviour for an upward slope of 10º, which is negative with respect to the world frame. The right walking beam experiences a positive pitch and the left walking beam a negative pitch, based on their coordinate system definition. Thus, the walking beams average the pitch (including

the direction) to the correct value for the pitch of the chassis, which is depicted in Figure 4.22, and has a value of -10^{o} in the world frame. In addition, Figure 4.22 shows that the chassis only experiences a pitch angle in its traverse since a straight-line drive on a longitudinal incline would not encounter roll or yaw.

Case 3: Side Slope Terrain

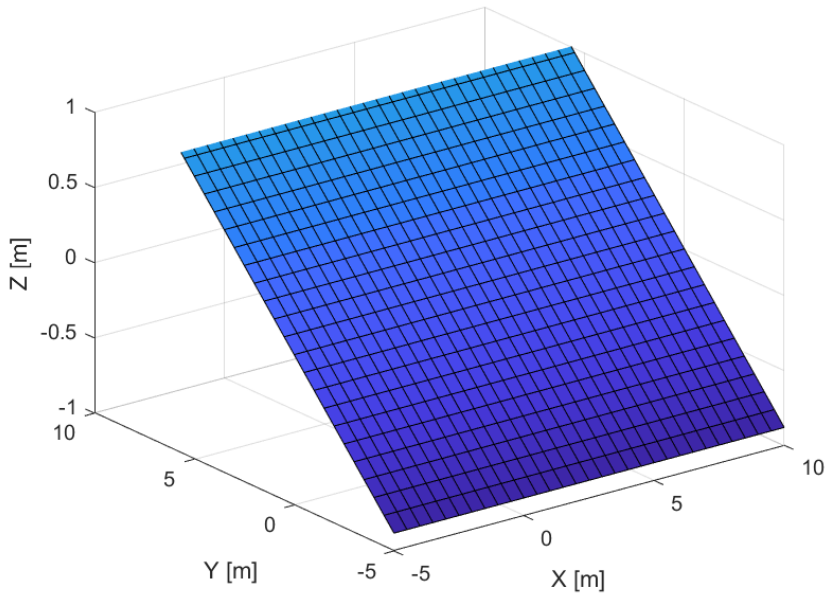


Figure 4.23: Side slope 10^{o} terrain digital elevation map.

Following the inclined uphill slope, the third test case was decided to be for driving along a side slope, given that it's a likely scenario. Figure 4.23 depicts the terrain for a side slope of 10^{o}. Once again, the rover is simulated for a straight path in the x-direction. For this case, it was expected that the chassis roll value would match that of the terrain, while pitch and yaw would have a value of 0.

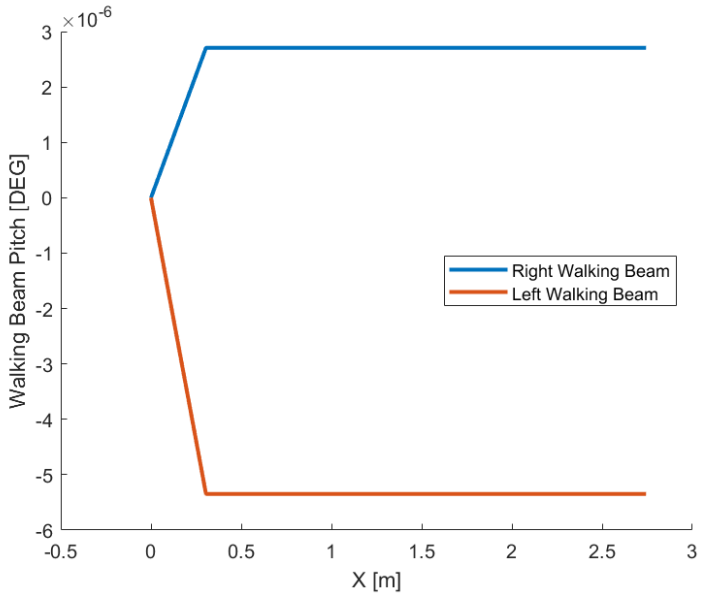


Figure 4.24: Walking beam pitch vs distance travelled (side slope 10º).

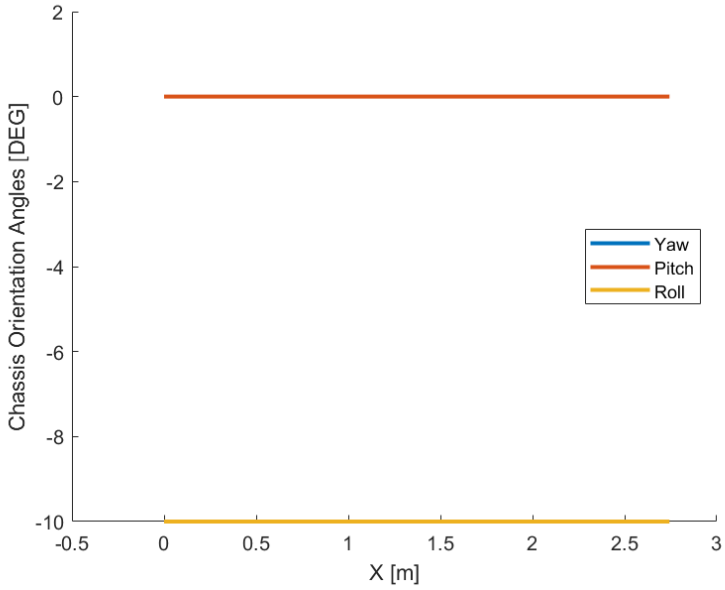


Figure 4.25: Chassis orientation angles with respect to distance travelled (side slope 10º).

The results for the side slope terrain are depicted in Figures 4.24 and 4.25. As expected, the pitch of each walking beam is essentially 0, with the scale on Figure 4.24

again showing the apparent opposite rotation of each walking beam (due to the positive z axis of rotation for each walking beam being in opposite directions). Subsequently, the chassis pitch if the average with a value of 0°, and the yaw is also 0°, both as expected. The overall chassis roll is given as -10°, which indeed matches the slope of the terrain.

Case 4: Sinusoidal Terrain

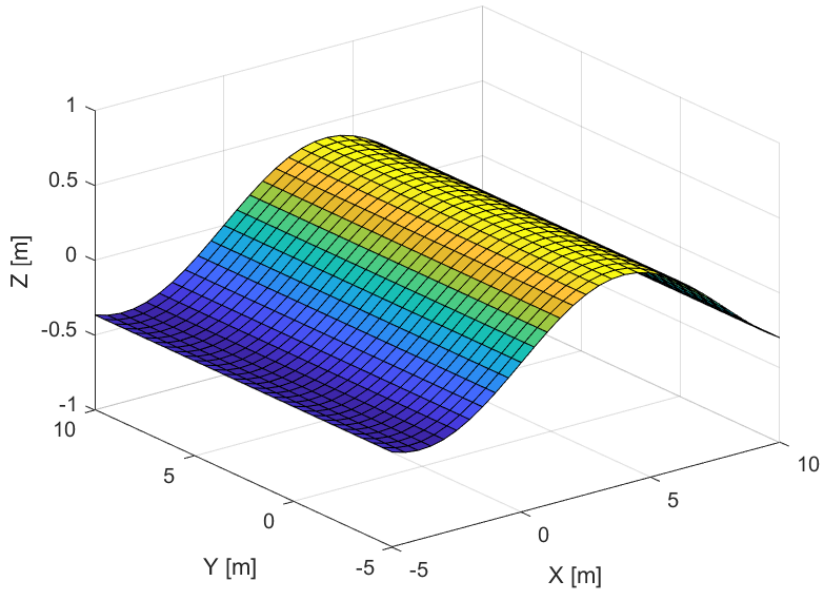


Figure 4.26: Sinusoidal terrain digital elevation map.

The final test case included in this thesis is for a slightly more complicated terrain to show the model's potential. A sinusoidal curve was chosen, with the exact function as $z=0.4\sin(0.4x)$, and is depicted in Figure 4.26. The sine curve allows for a test of a changing slope, which should result in changing pitch angles, including the existence of

the inflection point. The rover is again driven in a straight line in the x-direction, so yaw and roll angles are expected to remain zero.

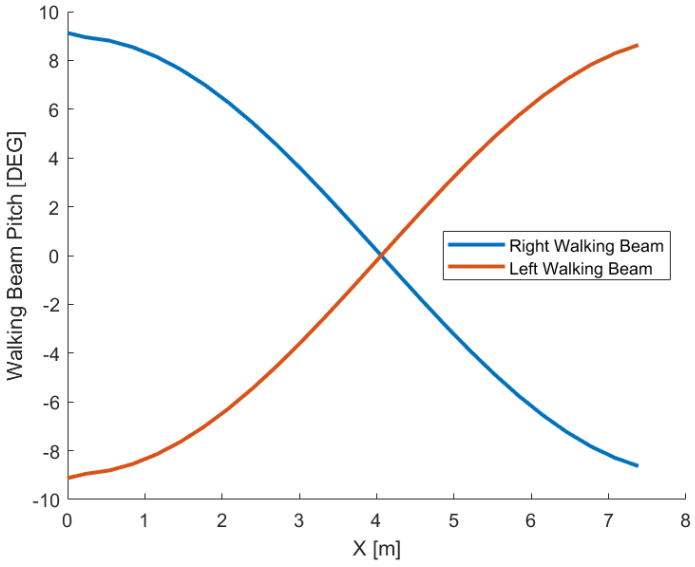


Figure 4.27: Walking beam pitch vs distance travelled (sinusoidal terrain).

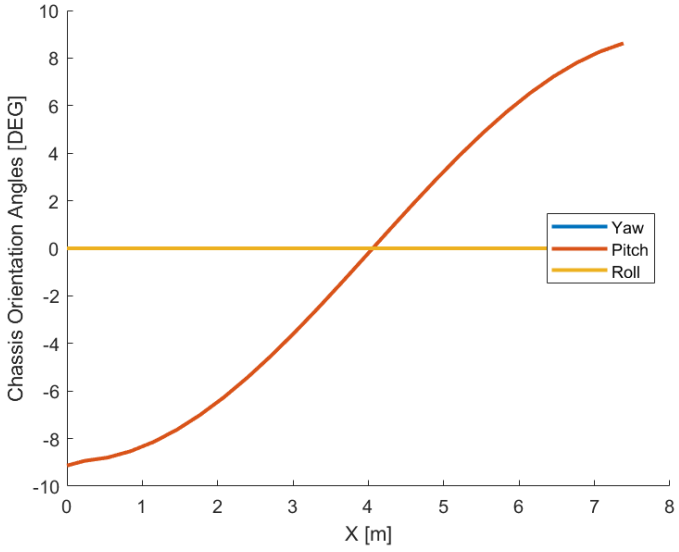


Figure 4.28: Chassis orientation angles with respect to distance travelled (sinusoidal terrain).

The results for the sinusoidal terrain are included as Figures 4.27 and 4.28. Figure 4.27 displays the walking beam pitch angles over the traversed distance. Due to the nature of the pitch in this code, the correction factors noted in simple longitudinal slope tests were maintained. The rover begins the traverse at x=0 of the sine function and as such, starts on an inclined slope. The slope can be manually determined by taking the derivative of the function, providing a means to check the numbers being produced by the MATLAB code. As expected, for an uphill slope, the right walking beam experiences a positive pitch that is the slope, whereas the left walking beam is its mirror image, with its positive axis of rotation in the opposite direction. Following the curvature of the terrain, the walking beam pitches curve towards 0 where they meet, before changing direction and moving away from each other once more. Figure 4.28 shows the chassis angles of the rover as it moves over the terrain. Due to the nature of the terrain and the path selected, the chassis should only be experiencing a change in its pitch, with both yaw and roll angles stationary at zero. Examination of Figure 4.28 shows that these expectations are met, with the yaw and roll angles remaining at a constant zero value, and the pitch angle as the average of the walking beams, meaning that the pose matches the direction and magnitude of the left walking beam. These results give confidence in the methods used to produce the code/kinematic position model, even with the correction factor.

4.3 Three-dimensional Velocity Kinematics (D-H Approach)

With the establishment of kinematic equations describing the position and orientation of the rover in the world reference frame, the method was expanded to include velocity analysis. Although being able to predict the pose of the rover as it interacts with the geometry of the terrain is crucial, having the capability to do a velocity analysis allows for the effect of non-geometric terrain characteristics, specifically slip, to be observed as well.

4.3.1 Theoretical Formulation

Since the previous 3D position kinematic model is the foundation for the 3D velocity analysis, all the previous model formulation presented in Section 4.2.1 still applies and is used here. In addition to all previous assumptions used in establishing the three-dimensional position kinematic model, some further assumptions were required for the velocity analysis.

Assumption 7: *The wheels of the rover are constrained to have the same angular velocity for a given side (ie. right vs left).*

As mentioned in previous sections, the rover under analysis is one that is equipped with skid steering. The application of skid steering in the J5 rover means that each walking beam contains a motor to drive both wheels via a belt drive. More specifically, it is assumed that there is no stretch or slip in the belt drive mechanism. In addition, the left

and right sides may be given different angular velocities to complete various steering maneuvers.

Assumption 8: *The rover wheels cannot be individually actuated or steered.*

Building on the assumption of the rover as a rigid body with no deformation and constant distances between certain joints allows the angular velocity of the contact point to be simplified to one direction, provided the rover is moving in a straight line.

Assumption 9: *Lateral (or side-slip) is minimal and can be considered negligible, not impacting rover progression.*

This assumption allows for side slip to be left out of the analysis but could be added back in should actual rover tests play a significant role. Similarly, steering angle rate slip was considered to be negligible due to the steering being skid steering as opposed to certain wheels having individual steering such as the Rocky 7 six-wheel rover analysed by Tarokh [48]. This consideration also means that the rover is assumed not to deviate laterally from its selected path and associated coordinates.

Assumption 10: *The speed of the rover is 0.10 m/s or 10 cm/s, in keeping with the speeds of other documented planetary exploration rovers (see Chapter 2).*

The selected speed is convenient as it allows the rover to travel slowly enough to maintain its assumed quasi-static status. Furthermore, this assumption enables the commanded angular velocity of the wheels to be calculated, in terms of no-slip.

Assumption 11: *Slip is obtained as part of the digital elevation terrain map and path coordinates.*

Specifically, that the digital elevation map, which could be obtained from satellite imagery, was post-processed and analysed for slope and terrain type, allowing slip-slope curves obtained from experimental drives of the J5 rover to be applied for extracting the approximate slip values. These slip values are then used to populate the map and become an additional array in the path coordinates.

In the velocity kinematics of serial manipulators, the velocity of the end effector is given by Equation 4.11, wherein the joint variable rates ($\dot{q}$) are multiplied by a matrix called the Jacobian, represented here by J.

$$\vec{V} = J\dot{q} \tag{4.11}$$

The vector, V, is comprised of three translational and three rotational velocities to fully describe the motion of the end effector, or in this case, the contact point in the world frame. The manipulator Jacobian relates the set of joint variable rates into these velocities of the wheel ground contact point. Due to the nature of the problem, with passive joints and considering information at the contact point to be known, the velocity analysis will be an inverse kinematics analysis with the joint values as the unknowns to be solved for. Hence, the manipulator Jacobian must be determined, along with developing expressions for the different components of the end effector velocity with respect to the world frame.

For the manipulator Jacobian, it is beneficial to recognise that it too, can be divided into a translational and angular/rotational part, as shown in Equation 4.12.

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \tag{4.12}$$

Regardless of how many kinematic pairs or joints are in a chain, the Jacobian will always have six rows for three-dimensional analysis due to the column vector of the end effector velocity containing six components. The number of columns will vary, however, as they are dependent upon the number of joint variables. In doing so, the number of columns also provides insight into whether the system is under- or over-actuated, with greater than six columns indicating it is over-actuated and likely to be over-constrained in the full equation set. For the J5 rover used in this work, there are eight joints in each kinematic chain from the world reference frame to that of the wheel-ground contact point, as can be observed in Figure 4.11 in the previous section. Thus, for the J5 rover, the Jacobian for each kinematic chain has the size/form of 6 x 8. The upper three rows of the Jacobian contain components for linear velocity, thus forming the linear Jacobian. Likewise, the rotational Jacobian is formed from the bottom three rows.

$$J_v = \begin{bmatrix} \frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \cdots & \frac{\partial p_x}{\partial q_n} \\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \cdots & \frac{\partial p_y}{\partial q_n} \\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \cdots & \frac{\partial p_z}{\partial q_n} \end{bmatrix} \tag{4.13}$$

The translational velocity Jacobian is determined first, using the previously determined overall T matrix and extracting the fourth column elements for the position of the end effector in the world frame. Using these three elements, the partial derivatives for each are obtained with respect to each of the joint variables as shown in Equation 4.13. For the kinematic chain of world reference frame to right rear wheel contact point, the translational velocity Jacobian is too large to include in the text of this thesis.

$$J_\omega = [\Lambda_1 h_0 \quad \Lambda_2 h_1 \quad \ldots \quad \Lambda_n h_{n-1}] \tag{4.14}$$

Obtaining the rotational velocity Jacobian is rather different. The columns for this matrix are determined using Equation 4.14, with *h* being the third column of the homogeneous transform matrix, T, from the world reference frame to each of the series of joints. The variable Λ, is merely used as a logic statement based on the joint type and is zero for prismatic joints and one for revolute joints. Thus, if the column corresponds to a prismatic joint, the result will be a column of zeros which makes intuitive sense as prismatic joints do not have a rotational component. The rotational velocity Jacobian components for the revolute joints, having a Λ of 1, are generated by extracting the third column in the orientation portion of the T matrix. The T matrix used is based on the concatenation of T matrices from the world frame to the end effector of that joint. Equation 4.15 demonstrates this process for the pitch joint of the chassis.

$$T_5^W = \begin{bmatrix} -c\theta_{yaw}s\theta_{pitch} & -s\theta_{yaw} & c\theta_{yaw}c\theta_{pitch} & X_{trans} \\ -s\theta_{yaw}s\theta_{pitch} & c\theta_{yaw} & s\theta_{yaw}c\theta_{pitch} & Y_{trans} \\ -c\theta_{pitch} & 0 & -s\theta_{pitch} & Z_{trans} + h_{CoG} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

Therefore, rows four to six, column 5 of the rotational Jacobian are $(c\theta_{yaw}c\theta_{pitch})$, $(s\theta_{yaw}c\theta_{pitch})$, and $(-s\theta_{pitch})$. Full derivation of the matrix elements was completed in Maple. Reassembling the translational and rotational Jacobians yields the overall Jacobian as shown in Equation 4.16.

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3} & \frac{\partial p_x}{\partial q_4} & \frac{\partial p_x}{\partial q_5} & \frac{\partial p_x}{\partial q_6} & \frac{\partial p_x}{\partial q_7} & \frac{\partial p_x}{\partial q_8} \\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \frac{\partial p_y}{\partial q_3} & \frac{\partial p_y}{\partial q_4} & \frac{\partial p_y}{\partial q_5} & \frac{\partial p_y}{\partial q_6} & \frac{\partial p_y}{\partial q_7} & \frac{\partial p_y}{\partial q_8} \\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3} & \frac{\partial p_z}{\partial q_4} & \frac{\partial p_z}{\partial q_5} & \frac{\partial p_z}{\partial q_6} & \frac{\partial p_z}{\partial q_7} & \frac{\partial p_z}{\partial q_8} \\ 0 & 0 & 0 & h_{3,x} & h_{4,x} & h_{5,x} & h_{6,x} & h_{7,x} \\ 0 & 0 & 0 & h_{3,y} & h_{4,y} & h_{5,y} & h_{6,y} & h_{7,y} \\ 0 & 0 & 0 & h_{3,z} & h_{4,z} & h_{5,z} & h_{6,z} & h_{7,z} \end{bmatrix} \tag{4.16}$$

Examining the complete Jacobian one can see that each yields a set of six equations describing the motion of the end effector with respect to the world frame.

With the Jacobian obtained, the next part in the formulation is to generate expressions for the end effector velocity components in the world frame. However, since these involve the wheels, and more specifically, the wheel-ground contact point, one must first familiarise themselves with the definition of slip. As mentioned earlier in this

and previous chapters, one of the benefits to going beyond the three-dimensional position kinematics and performing a velocity analysis, is that it can allow for the prediction of the effect of slip which is a non-geometric hazard. Slip is guaranteed to occur in environments such as the Moon or Mars due to the nature of the regolith or terrain, which is essentially a hard surface covered by fine particles [30, 42]. Slip is usually defined as the difference in relative motion between the wheel axle and the ground/surface it is travelling on [13, 30]. Slip is often described by Equation 4.17 as a non-dimensional variable, i, such that

$$i = 1 - \frac{v_x}{\omega r} \tag{4.17}$$

where v_x is the wheel axle translational velocity (in the direction of motion), ω is the commanded angular velocity of the wheel, and r is the wheel radius.

Typically, one associates slip with wheeled motion, but it should be noted that it can also affect legged robots as well. Usually, analysis of wheeled motion tries to adhere to what is referred to as the "no-slip" criteria, since it greatly simplifies the analysis. Wheeled motion is comprised of two parts: rotational, which is typically caused by the motor and translational representing the forward motion of the vehicle. The two components are essential for the wheel to effectively progress along a surface. Figure 4.29 depicts the two components of wheeled motion and consequent summation for the no-slip case.

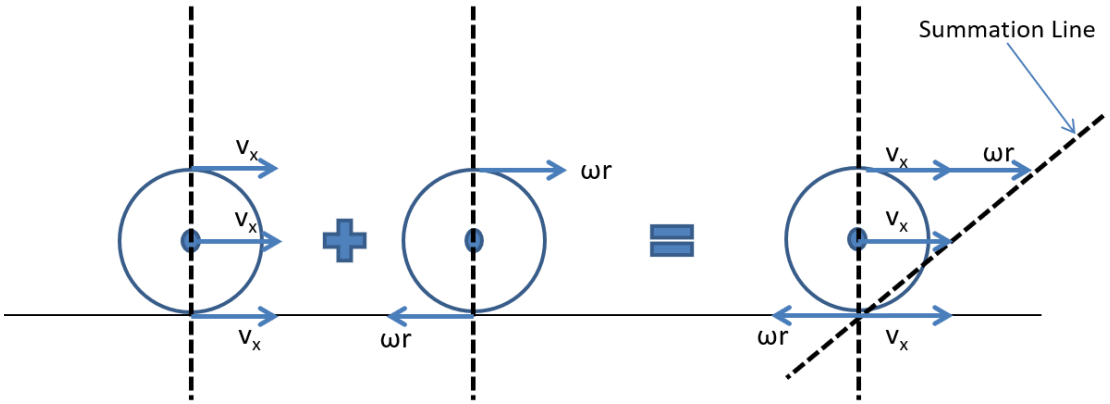
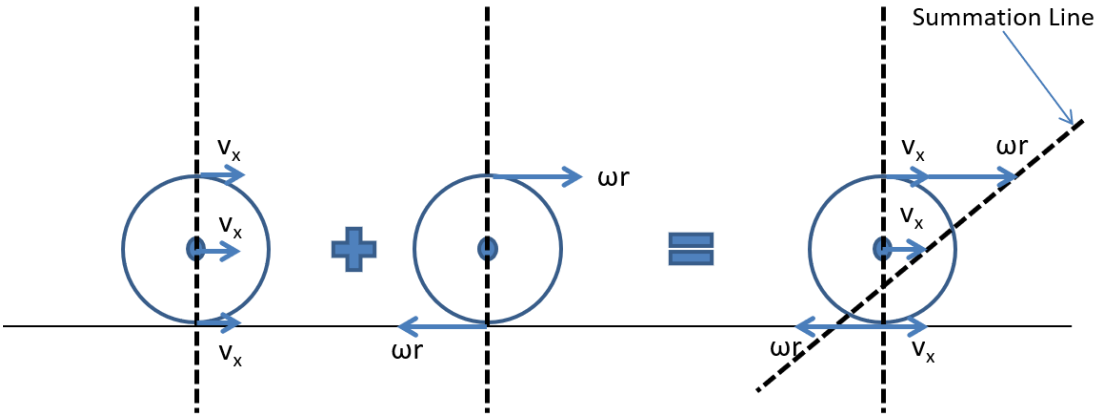


Figure 4.29: Interaction of rotational and translational velocities for wheels with no slip.

Examining Figure 4.29, it can be seen that the wheel has a rotation of ω, resulting in tangential velocity vectors along the wheel's edge with magnitude, v_t = ωr. Notice that for a clockwise rotation, the rover moves to the right, resulting in the displayed vectors. The top of the wheel has a vector, v_t, in the positive right direction and the bottom, where the contact point would be for flat, horizontal terrain, has the same magnitude of vector, but in the opposite direction. The wheel also translates, and for the condition presented of no-slip, the translation of the centre of the wheel (axle) and by extension, the whole wheel, is a set of vectors in the positive right direction, with a magnitude of the angular velocity multiplied by the wheel radius (ωr). Summation of the vectors results in the contact point vectors cancelling each other out and having a velocity of zero in that particular instant. The resultant zero velocity makes it behave like a fixed link for that movement which gives the no-slip condition. As expected for the no-slip condition, the center of the wheel advances by a velocity of ωr.



When v_x < ωr (slip), the contact point instantaneous velocity is negative.

Figure 4.30: Interaction of rotational and translational velocities for wheels with slip.

However, the no-slip condition rarely occurs. As depicted in Figure 4.30, when slip becomes involved, the translation of the wheel becomes significantly less until it approaches the case of 100% slip where the wheel spins in place and does not move. Such a case needs to be avoided, however even lower slip values can pose significant challenges as discussed in the Literature Review of Chapter 2. When slip occurs, the wheel does not make as much progress and the magnitude of the translation vectors becomes less than ωr (rotational component). Therefore, the velocity at the contact point has a negative tangential component. From Equation 4.17 for slip, and the wheeled motion diagrams, an expression of the velocities for the wheel axle and the wheel-ground contact point can be determined as follows:

$$v_{x,axle} = \omega r(1 - i)$$

$$v_{x,contact} = v_{x,axle} - \omega r \tag{4.18}$$

$$v_{x,contact} = -i\omega r$$

Equations 4.18 provide the expected results when the extreme boundaries of slip are applied (ie i = 0 vs i = 1). However, in keeping with the previous definition of the contact point to allow the inclusion of the terrain slope (γ), Equations 4.18 were further modified as follows.

$$v_{x,axle} = \omega r(1 - i)\cos\gamma$$

$$v_{z,axle} = \omega r(1 - i)\sin\gamma \tag{4.19}$$

$$v_{x,contact} = -i\omega r\cos\gamma$$

$$v_{z,contact} = -i\omega r\sin\gamma$$

These particular equations were then applied directly to produce expressions for the velocity of the end effector, which allows for the unknowns of this problem to be the joint variable rates as desired.

4.3.2 Method of Solution

Following the derivation of the velocity kinematic equations, the algorithm to solve these equations for the joint displacement rates was generated. The code

architecture flow chart describing the main functions and how it works is depicted in Figure 4.31.

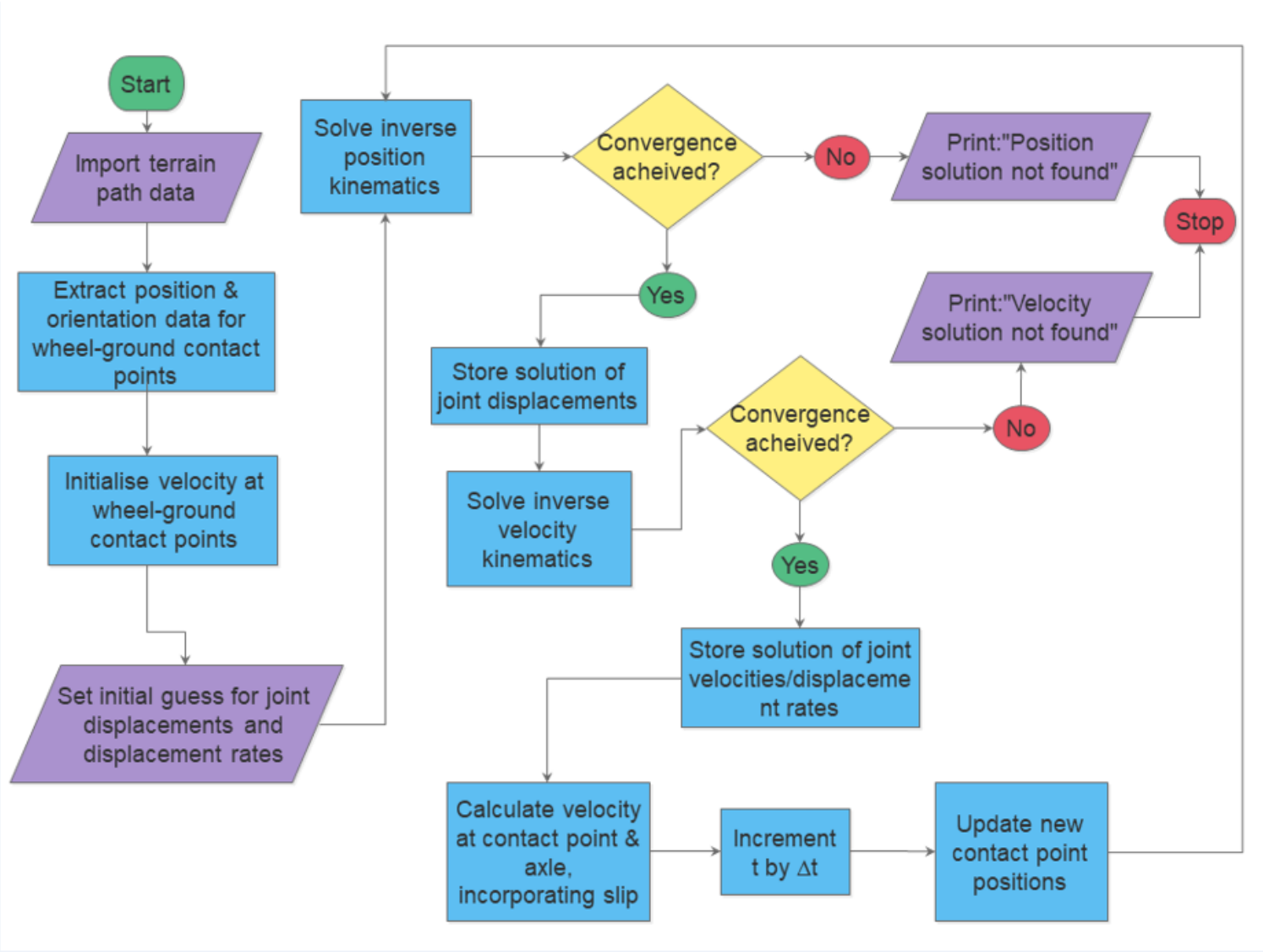


Figure 4.31: Three-dimensional inverse velocity kinematic code architecture.

Due to the nature of both the forward and inverse velocity kinematics problem, wherein the Jacobian is not only derived using position kinematics as its foundation, but also contains joint displacements, the inverse position kinematic analysis is intrinsic to being able to solve velocity. Thus it can be noticed in the code architecture of Figure 4.31, that some portions of code are very similar to that of the three-dimensional inverse position kinematics depicted in Figure 4.16, with minor adjustments that can be viewed in the MATLAB script file located in Appendix C. Identical to the position kinematics algorithm, the main input is a selected path on a digital elevation map, comprised of X, Y, Z coordinates for each wheel's contact point, however, there is also an additional array of data to be extracted for each set of coordinates containing a slip estimate. Recall that the slip values were assumed to have been extracted based on analysing the terrain for type and slope, then reading the slip off of the respective curve. These values are imported from excel spread sheets for the demonstrative test cases and are stored in arrays within the workspace of MATLAB.

Although a while loop is used again to step the rover through its selected path, the condition of the loop for this inverse velocity script is now based on time. The reason for using time is that when slip occurs, the rover will not progress as far along the path and won't coincide with the next set of coordinates on the map. The use of a time step is also more realistic as the real-life rover won't have a terrain map or path perfectly matched to rover's full set of contact coordinates. As such, it is fairly straightforward to determine the next set of solutions for a given interval (or delta) of time. The time step allows for new coordinates to be determined via the velocities and interpolation, along with the

added benefit of keeping track of the overall time in traversing the selected path. After entering the while loop, the end effector pose and velocities are determined and then the inverse kinematics problem is solved for that time step. Once the joint displacements are known, the same solver, fsolve, is applied to the set of velocity equations to perform inverse velocity kinematics with an initial guess and the previously solved for joint displacements as inputs. The full set of velocity equations to be solved is contained in the function file, J5VeloKin.m, and is attached in Appendix C. The solver selected had to be fsolve again, due to the Jacobian not being square, as well as being able to handle non-linear multivariate equations. Typically, in textbooks, when it comes to inverse velocity, the solution methods are presented as the geometric or algebraic solution where the inverse of the Jacobian matrix would be required. The non-square structure of this Jacobian makes resolution difficult plus, with computer programs such as MATLAB, it is easy to take advantage of numerical solution methods by using solvers, such as fsolve.

Once the inverse velocity kinematics has been solved, the variables are then updated before beginning the next time step. Interpolation steps are required due to the inclusion of slip in this analysis. The linear and angular velocities, along with slip and the time increment, are used to compute the new change in X, Y coordinates of each wheel axle. This change (delta) is then applied to the previous X, Y coordinates of the respective wheel contact point to obtain the new contact coordinates for each wheel in X and Y. From the terrain map, comparisons are made and then, if necessary, interpolation factors in X and Y are computed. Using these values and the terrain path data, the corresponding new Z values and slip are determined for the subsequent time step of the

loop. Finally, more visualisations of the data can be produced based upon the additional data available.

4.3.3 Results

Following the same case studies used in the 3D position kinematic analysis, a selection of results is presented for each, incorporating a range of slip values. The case studies presented were selected as being representative of the basic different terrain scenarios that could be encountered and thus the model needs to accommodate. Due to the 3D velocity analysis incorporating the solution of the position kinematic analysis, the full equation sets for the velocity Jacobians could be used with convergence achieved in the position analysis. With the rover speed of 0.1 m/s or a commanded angular wheel velocity of 0.333 rad/s, the simulation time was selected to be 75 seconds since the terrain and results did not change when compared with longer simulations. The time step used to generate the following results was 3 seconds, although other time steps could be used.

With regards to measuring accuracy, the position of the rover's centre of gravity during the traverse is compared with manually calculated position increments (for the CoG) included in the terrain maps. Since experimental validation has not occurred at this time, there are no true values of velocity against which to compare the results of the model. Therefore, velocity results for different terrain cases can only be compared against the flat terrain case for the same slip value, or the commanded translational velocity of the rover (based on the commanded angular velocity of the wheel). It should be noted that

mathematical models and processes are intended to simulate reality, and do not exhibit inherent accuracy. Model and process accuracy depends upon experimental validation to provide the true values for comparison.

Case 1: Flat Terrain

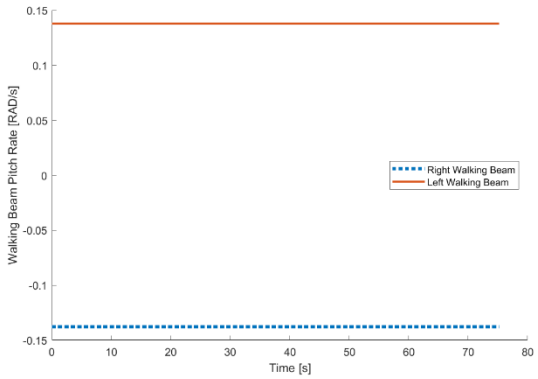
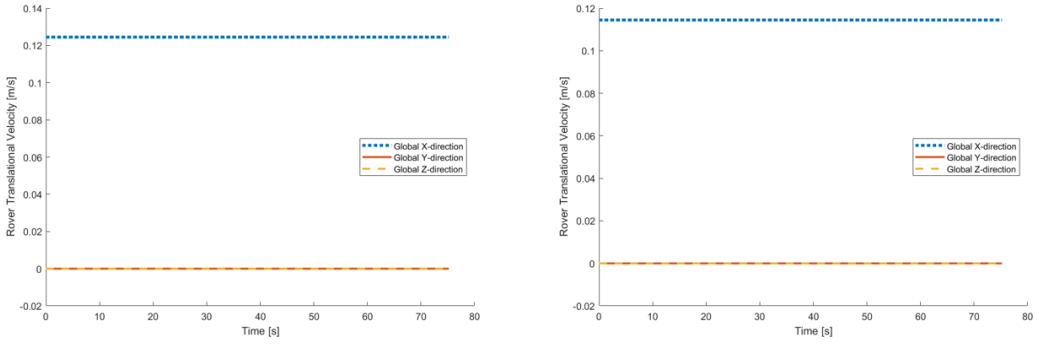
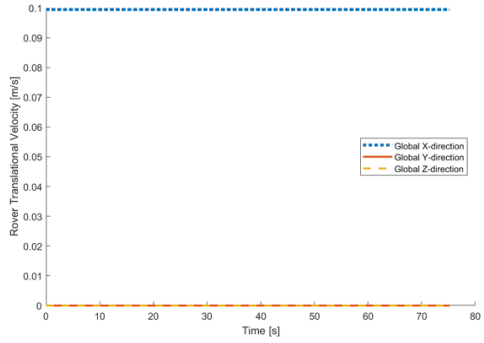


Figure 4.32: Walking beam pitch rates vs. time for a slip of 0.1.

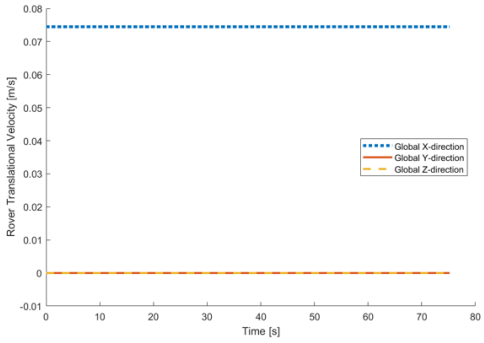


(a) No slip

(b) i = 0.1



(c) i = 0.25



(d) i = 0.5

Figure 4.33: Rover velocity components over simulated traverse.

Figures 4.32 and 4.33 detail a sample of the angular and translational velocities computed for the simplest case study of flat terrain. Building from the results of the position analysis, the resultant angles are included as an input to the velocity Jacobian. The velocity analysis converged quickly, with simulation run time under a minute for all cases of slip. The full set of results are included in Appendix C. From Figure 4.32, the walking beam pitch rates were observed for the simulation and were similar to rates for other values of slip on the flat terrain. As expected, the rates remain constant, with no change in commanded angular velocity or change in terrain. However, it was noticed that the pitch rates had a constant, non-zero value, which was not expected. This error

persisted consistently throughout all case simulations and even reformulating the Jacobian and resultant equation sets produced the same result. Thus, it is noted as a systematic error at this time. The velocity components shown in Figure 4.33 are of constant value, as expected, and only the x-component has a magnitude which is expected as that is the rover's direction of travel in the world frame. With increasing slip, the translation velocity of the rover decreases which is to be expected as the presence of slip limits the traction of the rover and the wheel does not travel as far forward. The longitudinal or x-component of translational velocity also is a bit larger in magnitude than the expected velocity (based on the no-slip case and commanded angular wheel velocities), suggesting that it may also be affected by the same systematic errors influencing the pitch rates. The velocity is higher in all slip cases by the same differential of 0.0245 m/s.

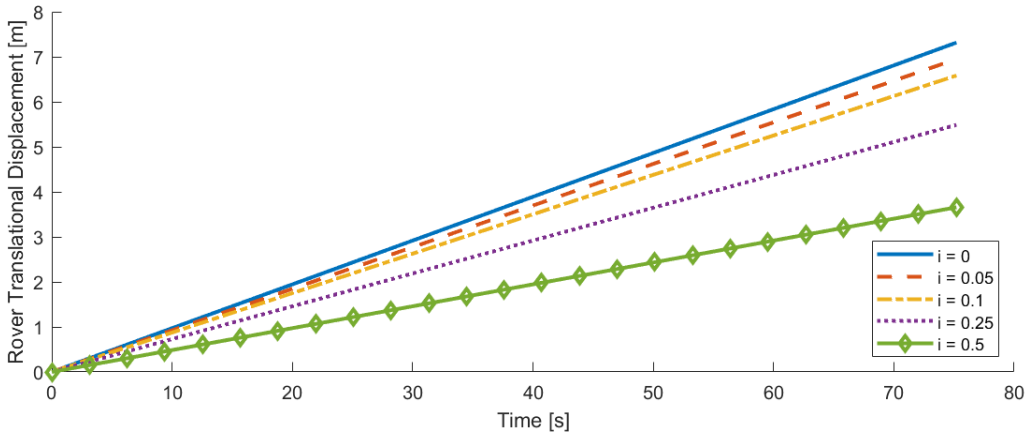


Figure 4.34: Traversed distance in the world x-direction vs simulation time, for different slip values (flat terrain).

Figure 4.34 examines the total distance travelled in the global x-direction by the rover in the given simulation time of 75 seconds for varying amounts of slip. As expected, the solid line representing the no-slip case, had the highest slope as it travelled the furthest in the x-direction. The value of x that it reached, was found to match the manually calculated value for the no-slip case exactly. Following the legend provided, as the terrain increases its slip value, the slope of the distance vs time line is observed to decrease, indicating that the rover did not travel as far. This trend was exactly as expected, as higher slip values represent a wheel that is losing forward mobility and moving towards spinning its wheels. Naturally, the highest slip value, in this case i=0.5, travels the least and barely makes it past 3 m in the 75 seconds of simulated traverse. Extending the simulation time, the difference in the distance travelled for different values of slip would grow.

Case 2: Uphill Sloped Terrain

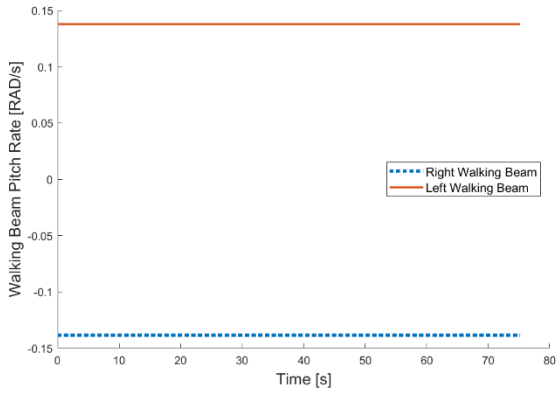
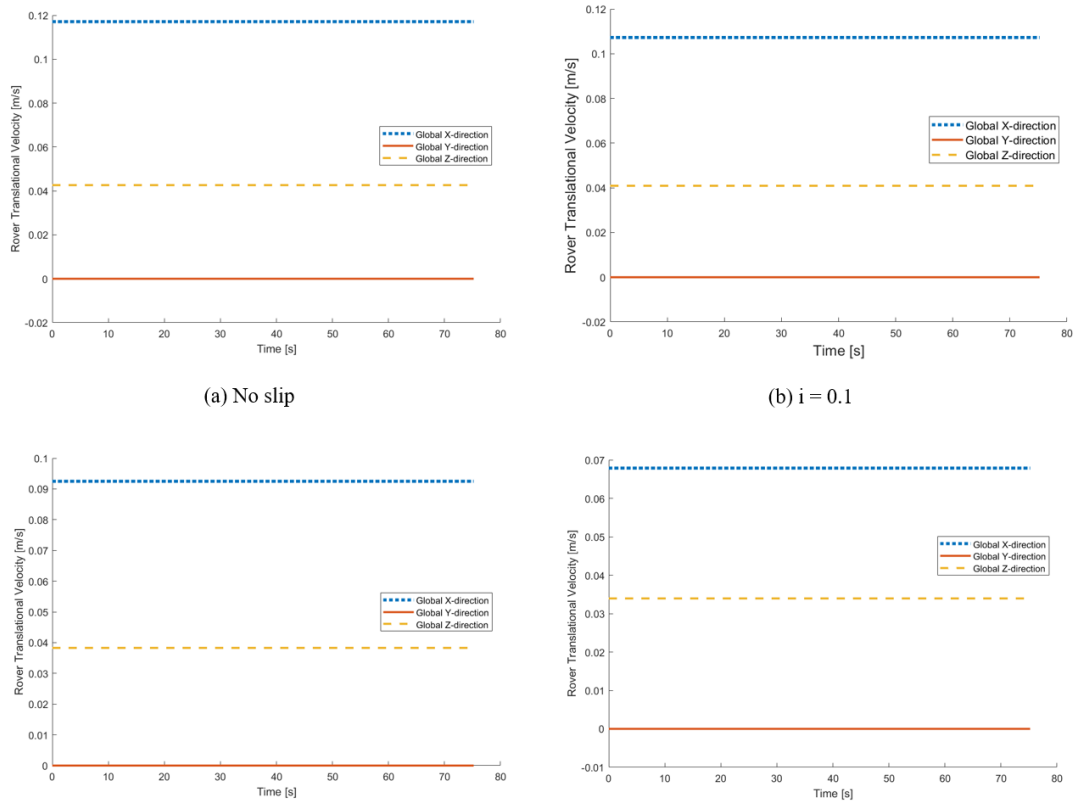


Figure 4.35: Walking beam pitch rates vs. time, for a slip of 0.1 (inclined terrain).



(a) No slip

(b) i = 0.1

(c) i= 0.25

(d) i = 0.5

Figure 4.36: Rover velocity components over simulated traverse (inclined terrain).

The results for the upslope case for a 10-degree inclined slope are displayed in Figures 4.35 and 4.36. With regards to the walking beam pitch rates in Figure 4.35, the overall trends are precise, in that the rates are constant which follows for constant commanded velocity and unchanging terrain. In addition, the walking beams exhibit the correct behaviour in that they mirror each other, which correlates with the behaviour seen in the position kinematic analysis and the established coordinates of each joint. The chassis pitch rate, which can be viewed along with the other results in Appendix C, also follows the relationship between it and the two walking beams as defined by the rear-

suspension. Again, Figure 4.35 exhibits the same inaccuracy for pitch rate as Figure 4.32. Figure 4.36 displays the rover's overall velocity components for the simulated traverse for four different slip values. Due to the shape of the inclined terrain and the direction of travel for the rover (straight line in x-direction), it was expected that the rover would have velocity components in the longitudinal and vertical directions, or for the world coordinate frame, in the x and z-directions. The results from the simulation follow the expected trend, with velocity in the x-direction being larger due to the small angle of the terrain for this particular test case. Although the values carry some inaccuracy, which may be due to the systematic error producing the inaccurate pitch rates, the model is consistent and also displays the correct decrease in both velocities for the increase in slip. Furthermore, the model is accurate in obtaining the correct translational displacement of the rover.

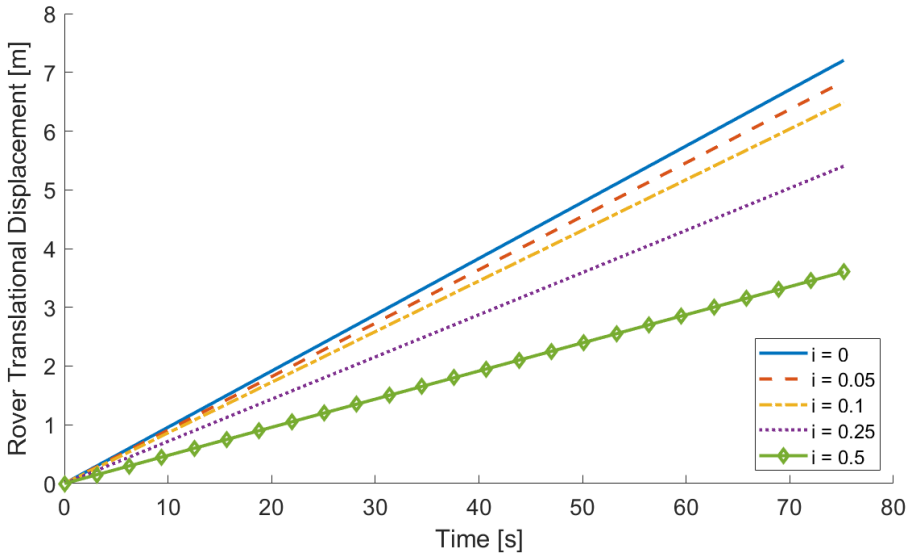


Figure 4.37: Traversed distance in the world x-direction vs simulation time, for different slip values (inclined terrain).

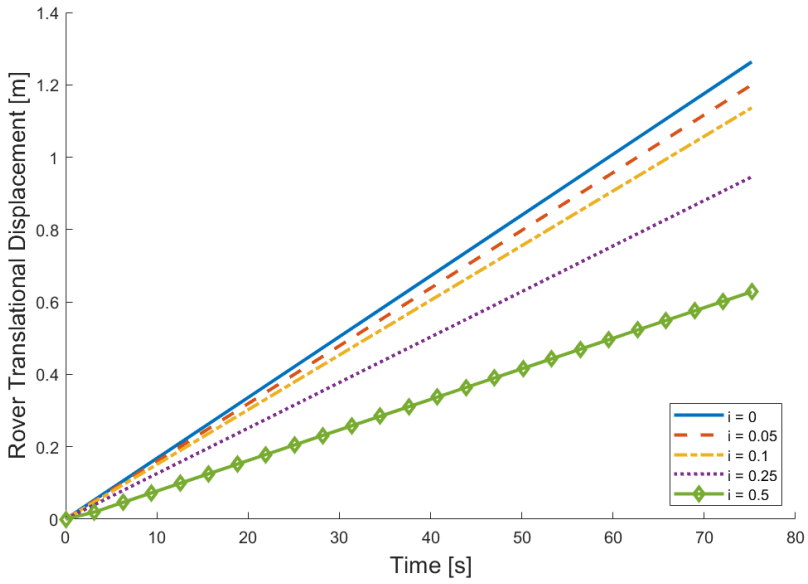


Figure 4.38: Displacement in the world z-direction vs simulation time, for different slip values (inclined terrain).

Examination of Figures 4.37 and 4.38 further demonstrate the model's accuracy in driving the rover to the correct displacement given the slip value of the terrain. The model correctly depicts the correlation between distance travelled and slip, with the no-slip case (solid line) having the highest slope and thus the largest distance travelled in the x-direction (7.204 m) and z-direction (1.2635 m). The highest slip value of 0.5 made the least progress, covering barely over 3m in the x-direction. Comparing the no-slip value to the expected value that was externally determined, the value is exact, indicating a high level of accuracy in the model being able to accurately move the rover given the terrain and its slip value.

Case 3: Side Slope Terrain

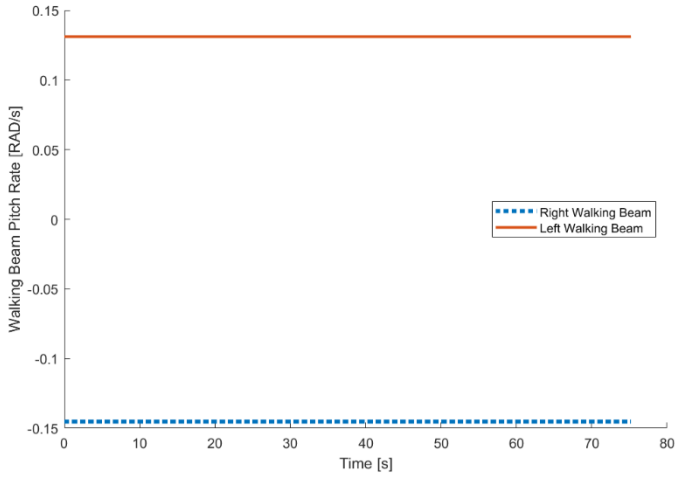
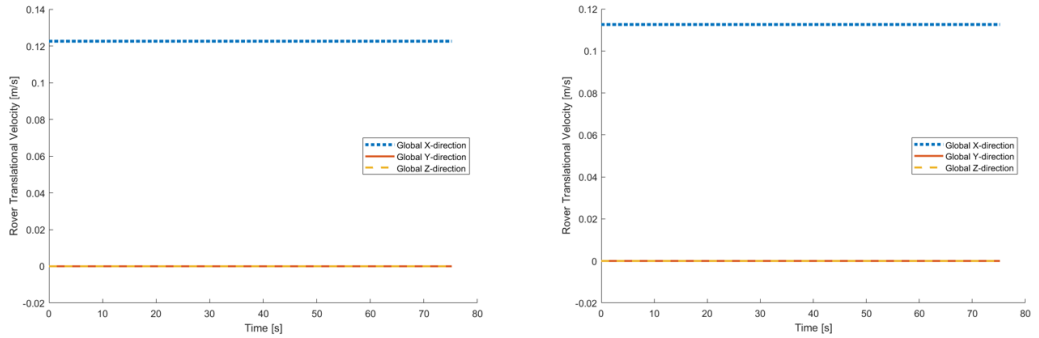
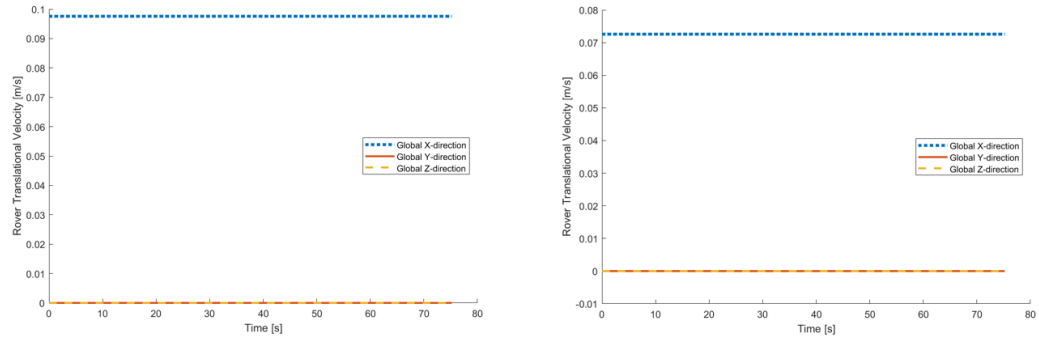


Figure 4.39: Walking beam pitch rates vs. time for a slip of 0.1 (side slope terrain).



(a) No slip

(b) i = 0.1



(c) i = 0.25

(d) i = 0.5

Figure 4.40: Rover velocity components over simulated traverse (side slope terrain).

The next case study was for a lateral or side sloped terrain of 10^{o} where, in the position analysis of Case 3 in Section 4.2.3, the rover experienced a roll to match the terrain slope and other angles had a zero value. Simulating a traverse over this terrain for the chosen simulation time of 75 seconds, the walking beam pitch rates are given in Figure 4.39. Similar to the position kinematic analysis, the results for the flat and side slope cases offer identical pitch angles, with the only difference being that the side slope provides a roll angle to the chassis. As such, it was expected that the velocity analysis should also have the pitch rates match for both the flat and side slope terrains, since pitch is not activated on either of these cases. Comparison of Figures 4.39 and 4.32 confirms the expected match, with the same pitch rate inaccuracy evident. Furthermore, the result shown in Figure 4.39 is representative of the result obtained for all the slip cases for the side terrains. Figure 4.40 highlights the translational components of velocity of the rover for different slip values. With the direction of travel being a straight line in the x-direction, and no elevation changes along its trajectory, the rover only experiences a velocity in the global x-direction as shown. Correctly incorporating the impact of slip, it is observed that the increase in slip decreases the translation velocity of the rover. Additional results and figures can be viewed in Appendix C.

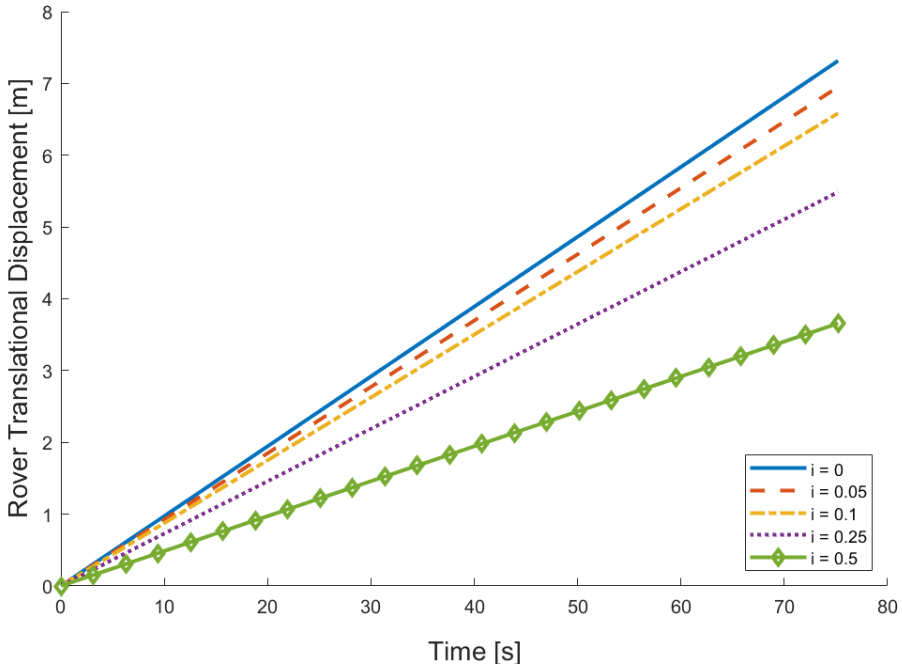


Figure 4.41: Traversed distance in the world x-direction vs simulation time, for different slip values (side slope terrain).

Continuing with the impact of slip, examination of Figure 4.41 shows the modeling of the expected behaviour of the rover progression in response to slip. The distance travelled in the global x-direction is accurately computed for the no-slip case as compared to the expected value determined during the generation of the terrain path maps. Likewise, the distance travelled for the other values of slip is shown, with the least amount of progress made on the terrain with the slip value of 0.5.

Case 4: Sinusoidal Terrain

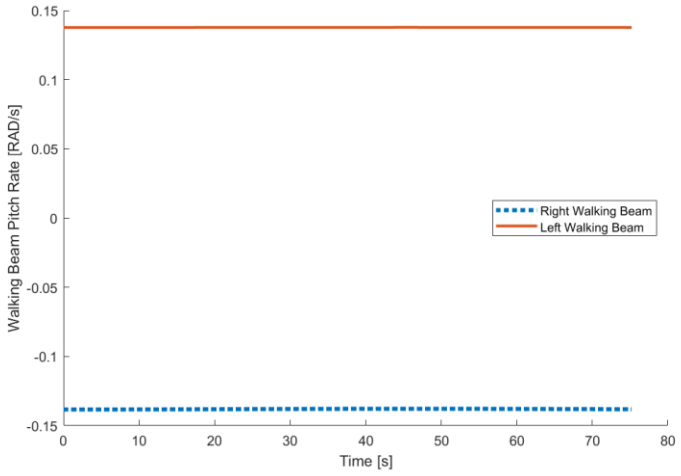
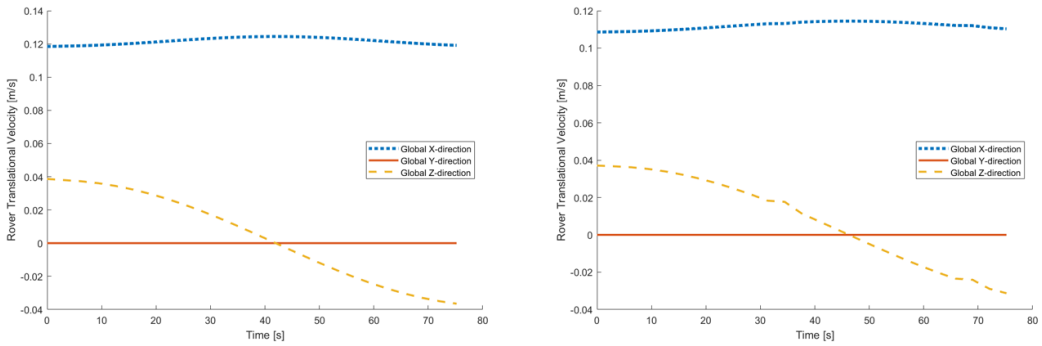


Figure 4.42: Walking beam pitch rates vs. time for a slip of 0.1 (sinusoidal terrain).



(a) No Slip

(b) i = 0.1

(c) i = 0.25

(d) i = 0.5

Figure 4.43: Rover velocity components over simulated traverse (sinusoidal terrain).

The final terrain case was a more complicated sine terrain to show the model coping with the changing slope. Figures 4.42 and 4.43 show some of the results from the velocity analysis, with more results located in Appendix C. From Figure 4.42 the walking beam pitch rates can be observed for the simulated traverse. It should be noted that the result presented in Figure 4.42 for 0.1 slip is representative of the results obtained for the other values of slip and can be viewed in the Appendix C. As well, the pitch rates are constant for the traverse and are displaying the same inaccuracy noted in the previous case studies. Figure 4.43 is interesting as it shows the translational velocity components of the rover; however, due to the nature of the changing slope of the sine curve, the velocity values are seen to fluctuate and follow somewhat of a rippling curve, most notably in the z-direction which is (the z of the terrain) dictated by the sine curve. As slip increases, each velocity component is reduced as expected; however, it is interesting to note the increased rippling in the z velocity component as slip increases. The increased rippling may be due to the cases with increased slip values relying more on interpolation based on the application of slip to obtain the correct velocity and establish the new set of wheel-ground contact points for the next time point of the analysis. Interpolation is smooth and easy on the straight-line slope terrains, however, for more complex terrains the interpolation may be improved by altering the time step of the simulation or possibly changing the method of interpolation. Subsequent testing, with time steps ranging from 1 to 7 seconds, revealed that a time step of 4 seconds eliminated the rippling for the sine function used, at moderate levels of slip. Since the change in time step did not completely

eliminate the rippling in all slip cases for the sinusoidal terrain, implementation of a non-linear interpolation function is recommended.

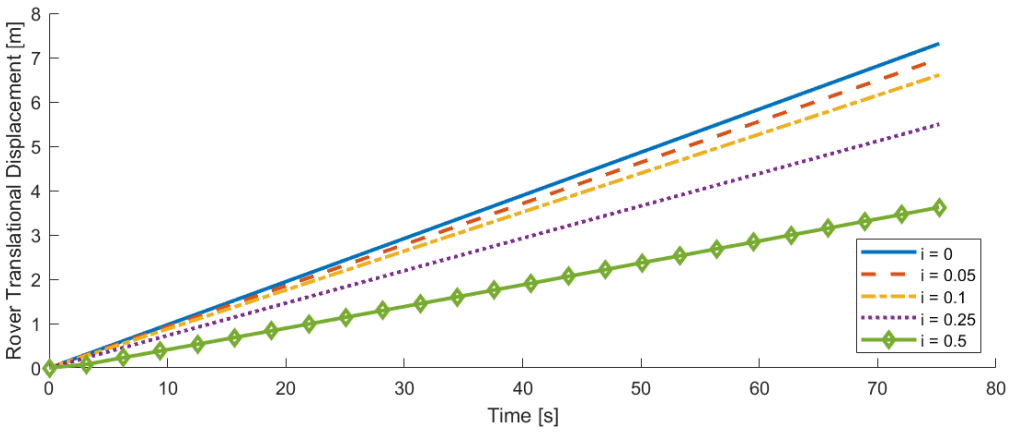


Figure 4.44: Traversed distance in the world x-direction vs simulation time, for different slip values (sinusoidal terrain).

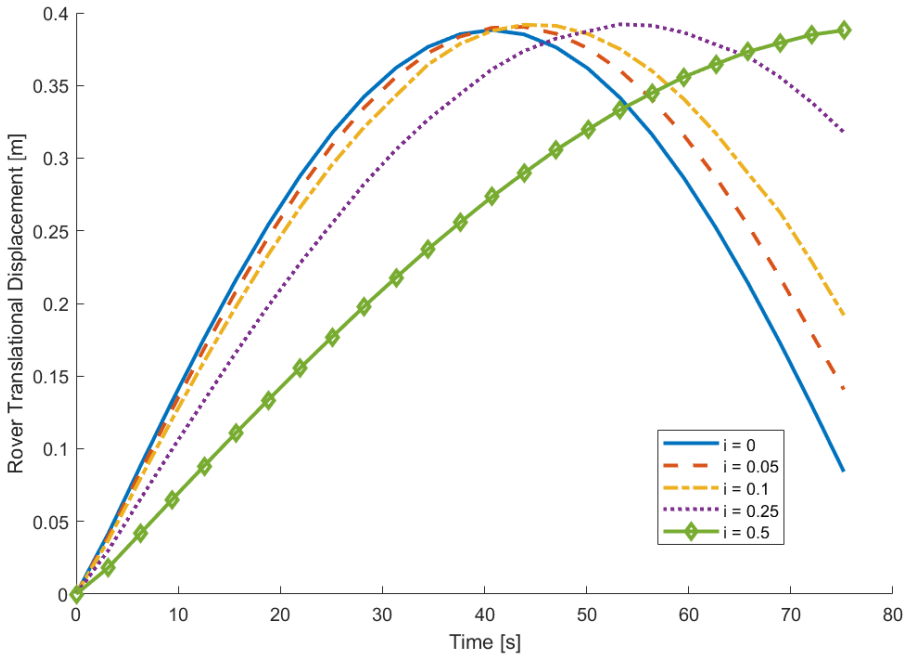


Figure 4.45: Displacement in the world z-direction vs simulation time, for different slip values (sinusoidal terrain).

Finally, the distance travelled in the global x-direction over the simulation time is examined in Figure 4.44, not only as an indication of accuracy in properly moving the rover along the terrain, but also to check for the correct inclusion of slip. Figure 4.44 shows the correct trends for increasing slip, as the rover makes less progress forward with higher values of slip. When comparing with the expected value for total distance travelled in the x-direction for the no-slip case, the model demonstrated an acceptable degree of accuracy.

The accuracy in total distance travelled is also reflected in Figure 4.45, which displays the displacement of the rover in the z-direction over the time of the simulated drive. Examination of Figure 4.45 illustrates a shift to the right of the peak displacement with an increase in slip. This shift is due to the reduced forward motion in the x-direction, which results in reduced progress along the sine curve of the terrain, thus affecting the displacement of the rover in the z-direction. Therefore, as slip increases, the rover reaches the peak of the sinusoidal terrain at a later time.

Overall, the velocity kinematic analysis model ran smoothly, with no convergence issues in all simulated traverses. The improvement in convergence is likely due to using the solution from position analysis for each time step, in addition to an initial guess, thereby requiring convergence to first occur in solving the inverse position kinematics problem. The solver had to be tuned to reduce the step tolerance several orders of magnitude from the default of 1e-6 to 1e-12, as the fsolve solver would occasionally stall

for a given time step, although it still reported successfully converging at a solution. All simulated traverses (all terrains and slip values) were able to be completed in less than a minute for each case. However, as noted in the aforementioned case studies, there are some inaccuracies in terms of the pitch rates computed. The model was able to predict the correct trends for each of the angular rates and velocities, but upon close inspection the magnitude of the walking beam pitch rates, and subsequently, the chassis pitch rate, were consistently off by the same inaccuracy across all cases and values of slip. The overall rover velocity values were also inaccurate, being 0.0245 m/s over the correct value, which for the flat no-slip case was 0.1 m/s, and this error may have been the result of the incorrect pitch rates. Yet, even with the presence of systematic errors, the three-dimensional velocity model was still able to accurately move the rover, incorporating slip, and to accurately determine its pose. The following tables examine the accuracy and precision of the model itself more quantitatively. The percentage deviations detailed in Tables 4.2, 4.3, and 4.4 are based upon Equation 4.20 and notes 1 and 2 beneath each table.

$$\%Dev = \frac{V_r - V_{x,flat}}{V_{x,flat}} * 100 \tag{4.20}$$

For further details, refer to the sample calculations provided in Appendix C.

Table 4.2: Results of rover velocities and deviation with respect to the flat case.[1,2]

Case	Slip	Rover Vx [m/s]	Rover Vz [m/s]	Percent Deviation [%]
Flat	0	0.1245	0	-
	0.05	0.1195	0	-
	0.1	0.1145	0	-
	0.25	0.0995	0	-
	0.5	0.0745	0	-
Side Slope 10°	0	0.1226	0	-1.53
	0.05	0.1176	0	-1.59
	0.1	0.1126	0	-1.66
	0.25	0.0976	0	-1.91
	0.5	0.0726	0	-2.55
Incline 10°	0	0.1171	0.0426	0.09
	0.05	0.1122	0.0418	0.20
	0.1	0.1073	0.0409	0.29
	0.25	0.0925	0.0383	0.62
	0.5	0.0679	0.0339	1.87
Sinusoidal	0	0.1191	-0.0365	0.05
	0.05	0.1151	-0.0325	0.08
	0.1	0.1103	-0.0313	0.14
	0.25	0.0926	-0.0196	-4.87
	0.5	0.0742	0.0037	-0.28

1. % deviation is calculated with respect to velocities (or distance travelled as appropriate) having the same slip on flat terrain.
2. x and z velocity components are combined using Euclidian distance theorem for comparison with uni-directional values, by magnitude only.

From Table 4.2, the velocity components of the rover are displayed for each terrain case and values of slip. The percent deviation value is that relative to the flat terrain case for that particular value of slip. The terrain with the highest percent deviation is the sinusoidal terrain, with the largest difference being less than 5.00%, which was observed at the slip case of 0.25. Both the incline and sinusoidal terrains which utilise

pitch, had small percent deviations at 1.87% and -4.87% respectively, with the largest values for both occurring for higher slip values. These values are all of a low magnitude and suggest that even with the inaccuracies occurring in the pitch rates, the model is accurate for the values it produced for other terrains when compared to the flat terrain case. It is worthwhile to note that the aforementioned change in time step for the sinusoidal terrain which eliminated the ripple effect for slip of 0.25, also reduced the resultant velocity percent deviation from -4.87% to -0.03%. The percentage deviation at other slip values and for other terrain cases remained relatively unchanged, suggesting that a nonlinear interpolation function should be implemented. It is expected that upon resolving the systematic error occurring in the flat case, that these percent deviations will remain unchanged. These results for percent deviation also indicate that the model is fairly precise in its computation, as amongst all cases the deviation values listed in Table 4.2 are all small and similar in magnitude, grouping the results.

Table 4.3: Results of axle velocities and deviation with respect to the flat case.[3,4]

Case	Slip		Axle Vx [m/s]	Axle Vz [m/s]	Percent Deviation [%]
Flat	0		0.1000	0	-
	0.05		0.0950	0	-
	0.1		0.0900	0	-
	0.25		0.0750	0	-
	0.5		0.0500	0	-
Side Slope 10°	0		0.1000	0	0
	0.05		0.0950	0	0
	0.1		0.0900	0	0
	0.25		0.0750	0	0
	0.5		0.5000	0	0
Incline 10°	0		0.0985	0.0174	0.03
	0.05		0.0936	0.0165	0.05
	0.1		0.0886	0.0156	-0.04
	0.25		0.0739	0.0130	0.05
	0.5		0.0492	0.0087	-0.07
Sinusoidal	0	Front Wheels	0.0988	-0.0155	0.01
		Rear Wheels	0.0990	-0.0139	-0.03
	0.05	Front Wheels	0.0940	-0.0139	0.02
		Rear Wheels	0.0943	-0.0112	-0.04
	0.1	Front Wheels	0.0890	-0.0132	-0.03
		Rear Wheels	0.0894	-0.0104	0.00
	0.25	Front Wheels	0.0745	-0.0083	-0.05
		Rear Wheels	0.0748	-0.0048	-0.06
	0.5	Front Wheels	0.0500	0.0000	0.00
		Rear Wheels	0.0499	0.0024	-0.08

3. % deviation is calculated with respect to velocities (or distance travelled as appropriate) having the same slip on flat terrain.
4. x and z velocity components are combined using Euclidian distance theorem for comparison with uni-directional values, by magnitude only.

Table 4.3 examines the percent deviation of the axle velocity components and the corresponding value of slip for all terrain cases simulated. Although not a quantity directly solved by solving the Jacobian velocity equation set, the velocity kinematic model involved computation of the axle velocities and incorporated slip to help with moving the rover to the correct spot for the next time step. The sinusoidal and longitudinal inclined terrains were naturally noted to have velocity components in the x and z-directions. However, the sinusoidal terrain has a slope that changes with one's position on the curving terrain, thus the analysis was computed for both front and rear wheels separately. The results of the percent deviation calculation with respect to the flat case, provide a quantitative indication of the model's level of accuracy and precision. The model is accurate in its computation of the velocities in terms of the individual results with respect to the flat case, with the side slope terrain performing the best in that no deviation was found. Yet both the inclined and sinusoidal terrains had very small values of deviation, with the largest value being -0.08%, again suggesting a high accuracy in comparison to the flat terrain as they're close to zero. These numbers are all very close in value, which demonstrates that amongst all cases there is a good level of precision in the model's computation.

Upon re-examination of Table 4.2 and its prior discussion, it is apparent that a large systematic error is included in the results for the rover translational velocities which are located at the C of G of the vehicle. The rover velocity results shown appear to be in error by 0.0245 m/s when compared with the commanded velocity of

ωr= 0.1 m/s. However, when examining Table 4.3, the translational velocities for the axles exactly match the expected results at all values of slip. The comparison of the results of Tables 4.2 and 4.3 therefore indicates that the deviation in translational velocity for the C of G of the rover is systematic and could potentially be tuned out when experimentally validating the overall kinematic model. Further investigation of the rover translational velocity error by reformulation of the system of equations yielded the same results shown in Table 4.2. As noted in Chapter 1, Objective 2 states that the 3D kinematics model is intended to accurately determine the progression of the rover on the terrain. The objective is accomplished based on the wheel axle results and the distance travelled results. The rover (C of G) velocity results are intended to provide a unified, central equivalent to the axle results and can be corrected as part of future work on the model.

Table 4.4: Results of distance travelled and deviation with respect to the flat case.[5,6]

Case	Slip	Rover X Distance [m/s]	Rover Z Distance [m/s]	Percent Deviation [%]
Flat	0	7.3152	0	-
	0.05	6.9494	0	-
	0.1	6.5837	0	-
	0.25	5.4864	0	-
	0.5	3.6576	0	-
Side Slope 10°	0	7.3152	0	0.00
	0.05	6.9494	0	0.00
	0.1	6.5837	0	0.00
	0.25	5.4864	0	0.00
	0.5	3.6576	0	0.00
Incline 10°	0	7.2041	1.2635	-0.02
	0.05	6.8439	1.2000	-0.02
	0.1	6.4837	1.1365	-0.02
	0.25	5.4031	0.9459	-0.02
	0.5	3.6021	0.6283	-0.03
Sinusoidal	0	7.3397	0.0842	0.34
	0.05	6.9709	0.1408	0.33
	0.1	6.6074	0.1920	0.40
	0.25	5.4966	0.3178	0.35
	0.5	3.6228	0.3880	-0.39

5. % deviation is calculated with respect to velocities (or distance travelled as appropriate) having the same slip on flat terrain.
6. x and z distance components are combined using Euclidian distance theorem for comparison with uni-directional values, by magnitude only.

Finally, Table 4.4 displays the percent deviations for each value of slip from the flat terrain case with regards to the distance travelled by the rover. For comparison with the flat case, the Euclidean distance is computed using the distance components and the percent deviation for that particular slip value is computed. Similar to the previous results, the side slope terrain matches the distances travelled by the rover on the flat terrain for each slip case, and is seen to be the most accurate, with a percent deviation of zero. The percent deviation for the inclined terrain was also close to zero, with the

largest percent deviation occurring at 0.5 slip. The percent deviations were larger for the sinusoidal terrain; however, they were still small in magnitude with the largest value of 0.40% for a slip of 0.1, closely followed by the slip case of 0.5 at -0.39%. Examining the individual percent deviations for each case relative to the flat terrain, there is a high accuracy in the distances computed. This level of accuracy was expected as, while running the model for each terrain and slip value, the results were checked against manually calculated values for the corresponding no-slip case for that terrain and the numbers were very exact to four decimal places, which gave confidence in the results obtained. Likewise, with the results in the other tables, these percent deviations for rover distance travelled were all very small in magnitude (less than 0.4%) and close to the other values, which again suggests that the model has a high degree of precision in the computation of these distances.

4.4 General Comments

Overall the models presented appear to be able to reproduce the Argo J5 in their own capacity. Although not particularly useful for modeling the rover with respect to three-dimensional terrain, the planar model initially used still had value in providing visual insight into how a single side of the rover was likely to move along the terrain in a constrained setting of all four-wheels on the ground. The three-dimensional model directly answers objectives 1 and 2 of the research, with the accurate determination of pose, progression, and motion of the rover in response to the shape of the terrain and its

slip value. Based on the angles computed for each interval step, one could compare them to the angle values (if known) that would constitute in the tipping of the rover and ending the mission. The over-constrained nature of the rover as a parallel manipulator, meant that convergence was an issue which was resolved by removing orientation equations based on how the contact points could actually move in relation to each other. The model was able to accurately generate the pose of the rover, with the exception of the pitch angle rates, which were consistently off by half of what they should be for any terrain with a slope. Revisiting the joint coordinate frame assignments and subsequent transform matrix derivations and formulations yielded the same equation set for solution, so a correction factor of 2 was applied, as a tuning parameter. Likewise, the velocity kinematic model also experienced a systematic offset in the pitch rates obtained, which may have contributed to the rover velocity being a little bit higher than expected. Convergence was easier to achieve, and all 25 equations could be used, with the only adjustment to the default fsolve solver settings to lower the step tolerance to limit the occasional stall (although it still landed on a solution). With regards to the pitch rates and overall rover velocity being subject to inaccuracy from systematic error, the velocity kinematic model will require tuning prior to use. Furthermore, the rates determined by the model are an ideal case and do not take into account the joint friction and damping which will lower the joint rates from the ideal calculated values. These tuning parameters must be determined via experimental validation in order to obtain the best accuracy in the model.

It should be noted that in using the original D-H convention, the most time required on the part of the user is in the establishment of the equations for solution through careful selection of coordinate frames and determination of D-H parameters. Once the equation set is established, one can easily apply it in other areas beyond pose and path progression prediction, such as using it to control the rover to stay on its commanded path. Although both models account for the shape of the terrain, with velocity incorporating the effects of the terrain's slip, neither are able to account for other effects such as the sinkage that occurs in deformable terrain. The conceptualized dynamic model is intended to account for sinkage and other factors.

Chapter 5: Dynamic Analysis

While a kinematic analysis can provide useful data on rover progression, path traversability, and traversability metrics, it can be used as the foundation of a path following control scheme as demonstrated by Helmick et al [61]. Rover pose relative to tipping and warning of any joint constraint violations based upon motion and pose only provide part of the picture. Dynamic models and analysis go beyond pose and motion, to examine accelerations, forces, and corresponding torques. Depending upon the desired metrics, dynamic simulations have the potential to provide a more detailed survey relative to the rover's interaction with terrain because they can be paired with a wheel-soil interaction model. There are several methods of modeling the dynamics of a system, from the purely mathematical formulation of the La Grange and Newton-Euler methods, to various multibody physics software platforms. In terms of analysing robotic systems and vehicles, most applications (as noted in Chapter 2) choose to use multibody physics software due to the potential for higher accuracy combined with relative ease of use through the graphic user interfaces (GUI's). However, such elaborate software comes at a high financial cost due to the years of development to produce a user-friendly interface with equations and modules applicable to any system (robotic manipulator vs vehicle).

The work presented in this thesis initially began with expansion of the three-dimensional kinematic analysis of the rover since it was the next logical progression and continued with the desire to develop the most open-source and least expensive method. In addition to creating a dynamics model of the J5 rover, a wheel-soil interaction model had

to be selected for pairing with the dynamic model to demonstrate full capability and provide more accurate picture of the forces and moments acting on the vehicle.

Terramechanics provides the forces imparted to the vehicle by the terrain and its specific properties, which can significantly impact the vehicle. Terrain type, as has been observed from previous and current rover missions, can have quite a significant impact on the rovers traversability of the extraterrestrial terrain. One of the most infamous examples being that of the Spirit rover, where it broke through a thin, seemingly solid crust to sink into the underlying soft sand [15]. Upon examination of the literature, it was decided to use a semi-empirical approach as the best compromise between the advantages and disadvantages of empirical and analytical techniques. As a result, it was decided to use a pre-existing terramechanics model to avoid the months building one from scratch would require. After consultation of both literature and resident expertise in the area of terramechanics modeling, the model of Irani et al [37] was selected.

This chapter presents the derivation and development of the dynamic model for the J5 rover, followed by a review of the terramechanics model used to complete the conceptual model. Due to the time constraints of the master's thesis, the combined dynamic-terramechanic model developed is conceptual due to the added work in tuning the model to accurately fit the rover, including some experimental work. To make up for this shortcoming, a sample test case is provided for a single rover wheel, to further demonstrate the model and the expected outcomes.

5.1 Dynamic Model Development

The dynamics of the rover were initially developed in terms of the position and time derivatives of its joint angles using the La Grange formulation [65, 67]. After some investigation, it was decided to abandon this line of pursuit due to the level of effort and simplifications required. Based upon recommendations, it was decided to use a different approach to generate a dynamic model using SimMechanics. However, the La Grange formulation still provides a sense of how SimMechanics works and the developmental work accomplished on the dynamic torque equation is provided in Appendix D.

Following the decision to pursue a more efficient means of dynamic modeling, MATLAB Simulink's SimMechanics was selected to generate the dynamic model of the J5 rover. Although not an open source code or method, MATLAB Simulink was selected due it's flexibility and compatibility with other platforms, along with its being relatively inexpensive. Furthermore, SimMechanics enables the user to remain hands-on and has the potential for interfacing with a previously generated kinematic analysis. In addition, MATLAB and Simulink are capable of autogenerating C code, which means that the models and code developed in this thesis could be converted to the programming language C, making it more open-source so that companies would not be forced to purchase software.

In order to generate a dynamic model of the rover in SimMechanics, a physical body of the rover must be generated, either in SimMechanics itself or as an externally

made computer aided design (CAD) solidmodel of the rover. Although a fully detailed CAD model of the J5 rover exists, it was not accessible due to it containing proprietary information. However, using the information provided to the author, such as the mass information and drawing included in Appendix A, rough estimates for masses could be assumed and dimensions obtained from the drawing and a determinable scale. Extrapolating dimensions from measuring the drawing and applying the scale, allowed components of the rover to be created in CAD software. Solidworks was used to produce the solid model used in this work; however, it could be easily created in Autodesk's Fusion 360 [68] which is very inexpensive.

In modeling the individual components of the J5 rover using CAD, the back suspension of the rover could easily be included, which is important in providing a necessary constraint to the system. Since dimensions and mass values were not provided for the back-suspension components, dimensions were estimated using early rough measurements of the physical rover and a material of 7075 Aluminum was assumed. When applied to the components assigned a material density, the mass of each component was obtained, along with their associated inertia matrices. This method was more accurate than the oversimplified La Grange attempt. Similarly, equivalent solid models were generated for the components of the chassis, walking beams, and wheels using dimensions and masses obtained from the drawing and other information provided in Appendix A. The wheels modeled are the metal wheels intended for use in planetary exploration, as illustrated in the drawing in Appendix A. Figure 5.1 presents various views of the CAD model created by the author.

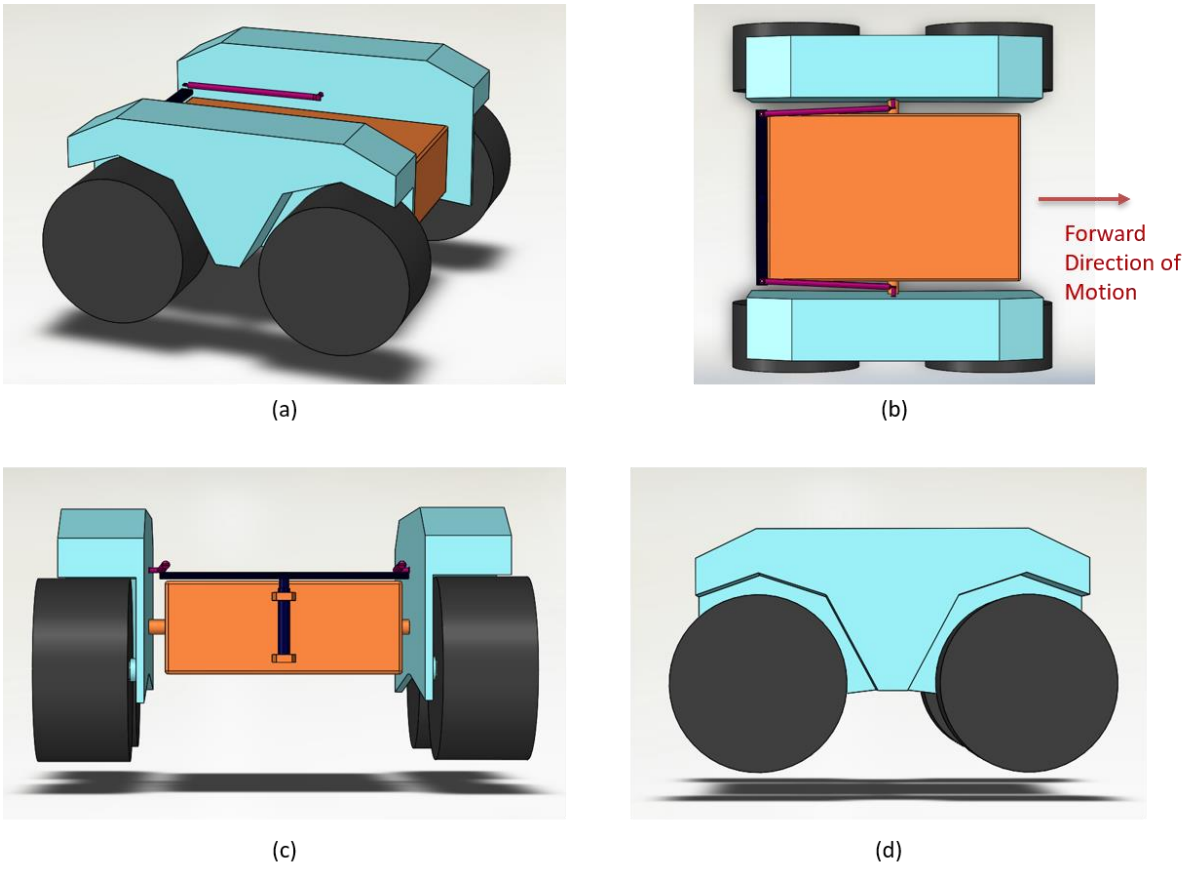


Figure 5.1: Isometric and orthographic projection of the J5 rover CAD model (isometric (a), top (b), rear (c), and right (d)).

From Figure 5.1, the rear suspension can easily be spotted in the top and rear views. It should be noted that more precise details could have been added; however, it would have been purely for aesthetics as the information important for dynamic modelling had already been incorporated (ie mass, size, etc). The resultant model was carefully assembled to properly reflect the degrees of freedom of each joint.

The solid model was then imported into SimMechanics by exporting the CAD model assembly into a .xml file which could be read by Simulink and converted into a rigid body dynamic model.

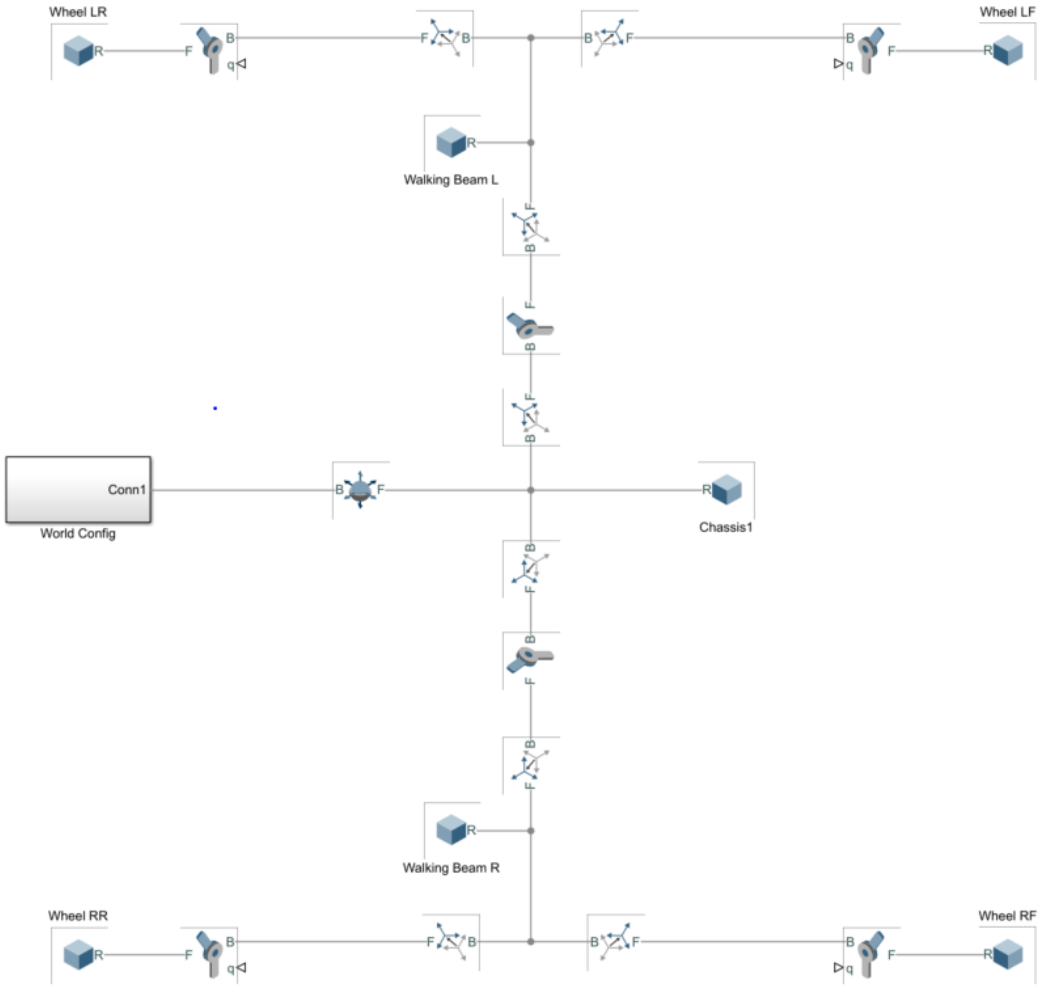


Figure 5.2: SimMechanics dynamic model of the J5 rover.

The resulting dynamic model of the J5 rover is depicted in Figure 5.2. As observed, the model is a rigid body representation with rigid components and joints. A world frame is

present to connect the rover to the world origin frame and provide context for the overall movement of the rover and its joints. With SimMechanics, rigid bodies are connected via joints and the appropriate transform blocks, forming pairs around joints, and enabling the program to determine the pose of each component similar to a 3D position kinematic analysis. Similarly, it was decided to add a 6 DoF joint to connect the chassis centre of gravity with the world frame, to allow for actuation and to travel away from the world frame. Note that although a 6 DoF joint does not exist in the real world, it is a viable tool in SimMechanics to represent the three possible translations and orientations. To actuate the model, forces and/or torques are applied to the joints themselves, as opposed to the rigid bodies. SimMechanics automatically includes the weight of each of the components in its computations. As such, forces must be applied at each of the wheel axles to provide a force in opposition to the weight of the rover and keep it from falling in space. The axles are where the forces from the terramechanics model will need to be applied. It should be noted that experimental validation with the actual rover would be needed to provide a more complete dynamic model, as damping would need to be added between the two walking beams to help conform the rover to its actual behaviour observed. Such work became beyond the scope of the time constraints of this thesis and is left as future work in need of completion.

At this point, it is necessary to import the terramechanic model and pair it with the dynamic model. The terramechanic model selected was Irani et al [37] for a single wheel testbed and was developed in Simulink.

5.2 Terramechanics Model

As defined in the literature review of Chapter 2, terramechanics is the study of forces and moments imparted by the terrain to the vehicle, and is thus dependent on terrain properties, such as soil cohesion. To understand how the terramechanic model works on its own and how it is being paired with the dynamic model, the relevant theory and equations of terramechanics are presented in this section. Figures, notation, and equations presented are in common use as detailed by Irani et al [37, 38, 39], Wong [29], Ishigami et al [42], and others (see Chapter 2). Terramechanics is not something a rover operator or designer should ignore given previous and current rover missions having demonstrated the effects of different terrain types on rover mobility.

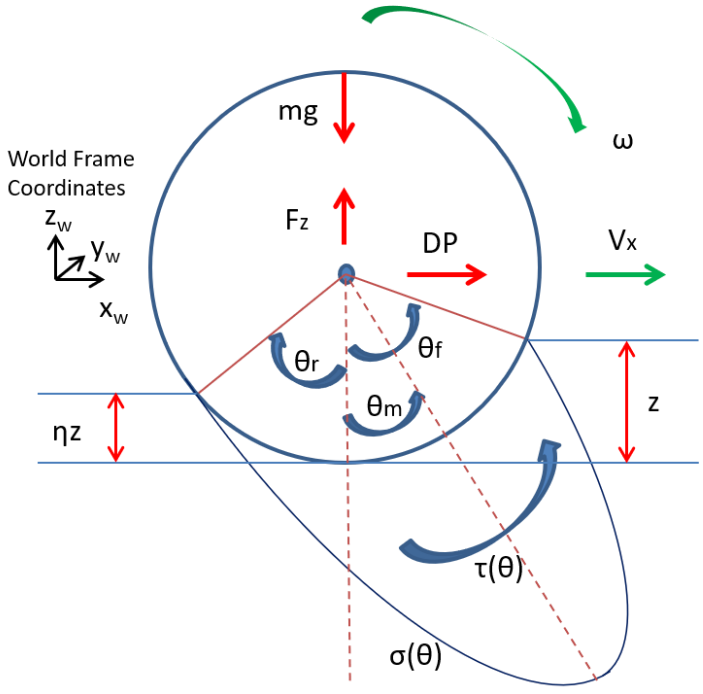


Figure 5.3: Terramechanics model representation of typical forces on the wheel [29, 37, 38].

Figure 5.3 is a common depiction of the forces and stresses resulting from the wheel-soil interaction and is reproduced in many textbooks and literature. From Figure 5.3, some of the key parameters involved can be visualised and provide a sense of how interconnected these values are. Easily identifiable are the translational velocity component, v_x, and the angular wheel velocity, ω, which provide direction context for the other components in the diagram. With most terrain, there is expected to be some amount of sinkage which, as the name implies, is the amount perpendicular to the surface that the wheel penetrates (for flat terrain), measured from the undisturbed terrain in front of the wheel. The depth of the track left behind by the wheel is simply ηz, where the sinkage value is multiplied by a ratio of track depth to sinkage. In line with sinkage and track depth, are the wheel sinkage angles listed as various θ, more specifically with θ_f as the forward or entry wheel sinkage angle, θ_r as the wheel sinkage rear or exit angle, and θ_m denoting the location of the maximum normal stress. Equations 5.1 – 5.3 describe the relation of these angles to the sinkage and wheel radius, r.

$$\theta_f = \cos^{-1}\left(1 - \frac{z}{r}\right) \tag{5.1}$$

$$\theta_r = -\cos^{-1}\left(1 - \frac{\eta z}{r}\right) \tag{5.2}$$

$$\theta_m = (b_0 + b_1 s)\theta_f \tag{5.3}$$

Examination of Equations 5.1 and 5.2 shows the dependence upon the ratio of sinkage to the wheel radius. Equation 5.3 is one of the Wong-Reece relationships [37], with the

values of b_0, b_1 as constants with values of 0.4 and between 0 and 0.3 respectively. These angles form the basis of the rest of the terramechanics equations.

Returning to Figure 5.3, the main stresses acting on the wheel are observed to be the normal and shear stress imparted by the terrain. The normal stress acts normal or perpendicular to the terrain surface, whereas the shear stress will always be tangential to the wheel-soil interface and in the opposite direction to the angular velocity of the wheel. The normal stress equation is given by Equation 5.4.

$$\sigma(\theta) = \begin{cases} r^n k \left(\cos\left(\theta_f - \left(\frac{\theta - \theta_r}{\theta_m - \theta_r} \right) (\theta_f - \theta_m) \right) - \cos\theta_f \right)^n & \begin{matrix} (\theta_r \leq \theta < \theta_m) \\ (\theta_m \leq \theta < \theta_f) \end{matrix} \\ r^n k (\cos\theta - \cos\theta_f)^n \end{cases} \tag{5.4}$$

Equation 5.4 shows a strong dependence upon the wheel sinkage angles as mentioned, along with the wheel radius and terrain parameters, k and n. The variable n represents the sinkage exponent whereas k is the Bekker coefficient of proportionality.

$$k = \frac{k_c}{b} + k_\phi \tag{5.5}$$

Equation 5.5 defines the Bekker coefficient of proportionality, where k_c and k_ϕ are soil parameters: the cohesive modulus and frictional modulus of sinkage, respectively, and

must be determined through testing. For planetary exploration rovers these values are estimated using an appropriate simulant soil. The parameter b denotes the width of the rover wheel.

$$\tau(\theta) = (c + \sigma(\theta)\tan\phi)\left(1 - e^{-\frac{j(\theta)}{K}}\right) \tag{5.6}$$

Equation 5.6 denotes the relationship for the shear stress at the wheel-soil interface and is referred to in the literature as the Janosi and Hanamoto equation, as utilised by Irani et al [37]. It can be observed that not only is the shear stress a function of the normal stress, and by extension the sinkage angles, but also depends on certain soil parameters. These include the cohesion, c, internal angle of friction, ϕ, and the shear modulus, K. Again, these parameters are estimates for planetary exploration rovers based on laboratory testing of simulant soils.

$$j(\theta) = r\left[\theta_f - \theta - (1 - i)\left(\sin\theta_f - \sin\theta\right)\right] \tag{5.7}$$

The other term in Equation 5.6 is the soil deformation term, given by $j(\theta)$, and expressed above as Equation 5.7. From Equation 5.7 it can be seen that it is highly dependent upon the entry wheel sinkage angle, along with the wheel radius and slip for that terrain. Although previously defined in Chapter 4, the equation for slip is repeated as Equation 5.8.

$$i = \frac{\omega r - v_x}{\omega r} = 1 - \frac{v_x}{\omega r} \tag{5.8}$$

Another important relationship in terramechanics is the pressure-sinkage relationship.

$$p(z) = kz^n \tag{5.9}$$

$$p(z) = \left(k_c + bk_\varphi\right)(z)^n \tag{5.10}$$

Equation 5.9 is the original pressure-sinkage relationship used by Bekker [27] and early work of Wong [28, 29], whereas most (such as Irani et al [37]) now use Equation 5.10 which is the modified Reece equation, sometimes referred to in the literature as the Wong-Reece equation for pressure-sinkage.

With the shear and normal stresses defined, the final elements of Figure 5.3 can be examined. These are some of the important metrics to be determined from a terramechanics analysis, which are the vertical force imparted by the terrain, F_z, and the drawbar pull, DP. Equations 5.11 and 5.13 detail the formulation of each respectively. Equation 5.12 describes the lateral force acting on the wheel.

$$F_z = rb \int_{\theta_r}^{\theta_f} [\tau_x(\theta) \sin\theta + \sigma(\theta) \cos\theta] d\theta \tag{5.11}$$

$$F_y = \int_{\theta_r}^{\theta_f} \left[rb\, \tau_y(\theta) + R_b(r - h(\theta) \cos\theta)\right] d\theta \tag{5.12}$$

$$DP = F_x = rb \int_{\theta_r}^{\theta_f} [\tau_x(\theta) \cos\theta - \sigma(\theta) \sin\theta] d\theta \tag{5.13}$$

Both Equations 5.11 and 5.13 show that the vertical forces and horizontal (or drawbar pull) forces rely on components of the normal and shear stresses, and consequently, the wheel sinkage angles. Furthermore, each of the integrals of the stresses is then applied to the rough contact area of the wheel given by the product of the radius and wheel width. The vertical force combats sinkage that is due to the pressure imposed by the weight of the vehicle. As will be seen in the following section, when applied to a dynamic model, the vertical force has a recursive relationship in that once calculated it also determines the value of sinkage which then enters the next iteration of the rover as moving along the terrain. Drawbar pull, as discussed in Chapter 2, is a common performance metric in both planetary rovers and vehicle performance. The drawbar pull is the horizontal force (along the direction of motion) which is accessible to the vehicle after overcoming the motion resistance of the terrain and can be used to pull a load.

5.3 Combined Dynamic and Terramechanics Model

It is advantageous to combine the previously established dynamic model for the rover with a terramechanics model, incorporating slip, to create a more realistic model of the rover in motion. Some advantages include determination of wheel sinkage relative to potential embedding and drawbar pull in terms of a safety margin on traction or potential towing capacity.

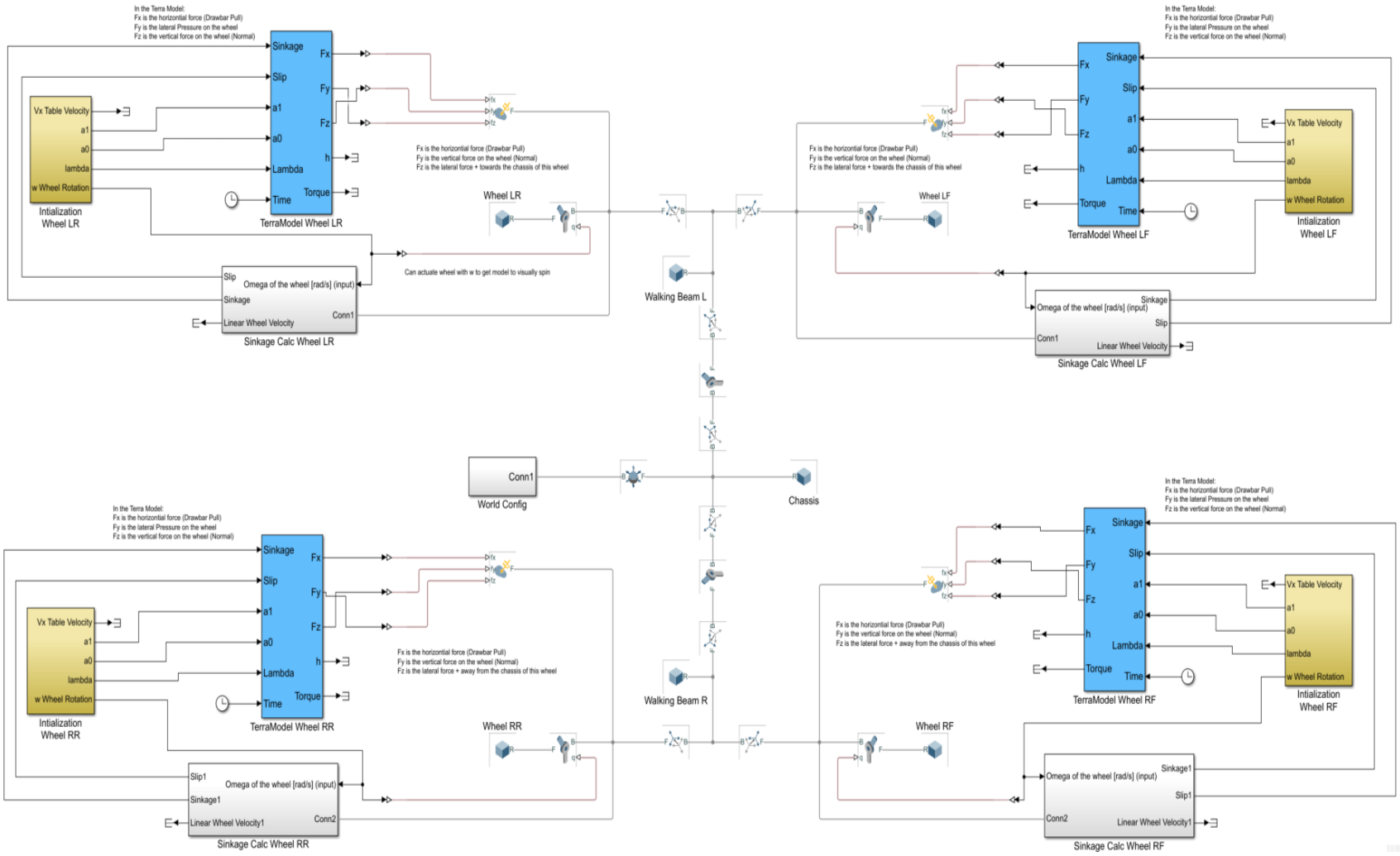


Figure 5.4: SimMechanics dynamic model combined with terramechanics model.

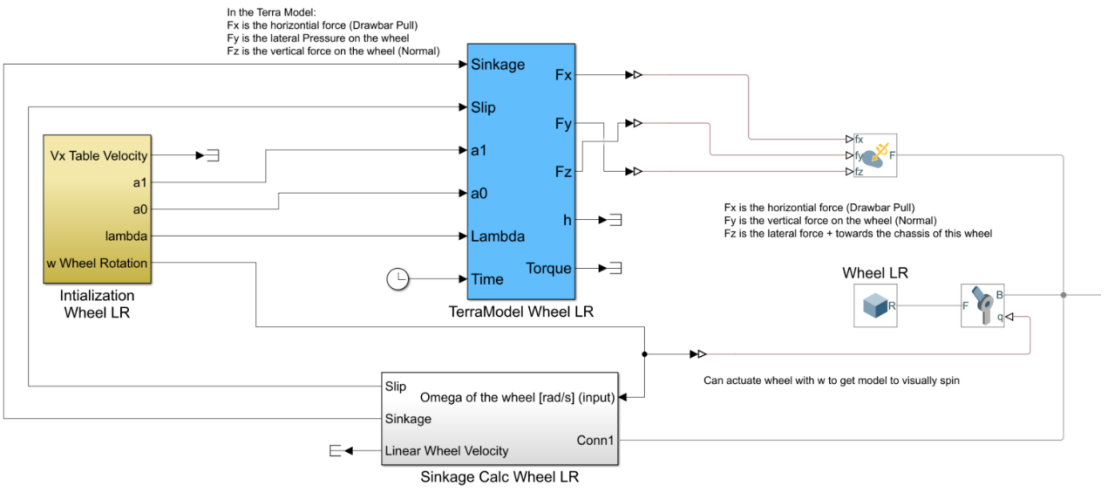


Figure 5.5: Close-up view of the added terramechanics-related blocks.

Figure 5.4 depicts the overall combined dynamic and terramechanic model. In comparison with Figure 5.2, the dynamic model can be easily identified and is the centre of the overall model itself. Recalling that the forces must be placed so as to act on the wheel joints, it is logical that the elements of the terramechanics portion are added to each of the wheel axles, making for four inputs. A close up of one of the wheels and terramechanics additions is provided in Figure 5.5, for clarity.

Examination of Figure 5.5 shows that there are three large subsystem blocks added, along with a smaller external force and torque block, which, as the name implies, allows for the application of an external force and torque to a subsequent block. The three subsystem blocks include an initialisation block shown in gold, the blue box containing the actual terramechanics model, and a white box for the sinkage and slippage calculations.

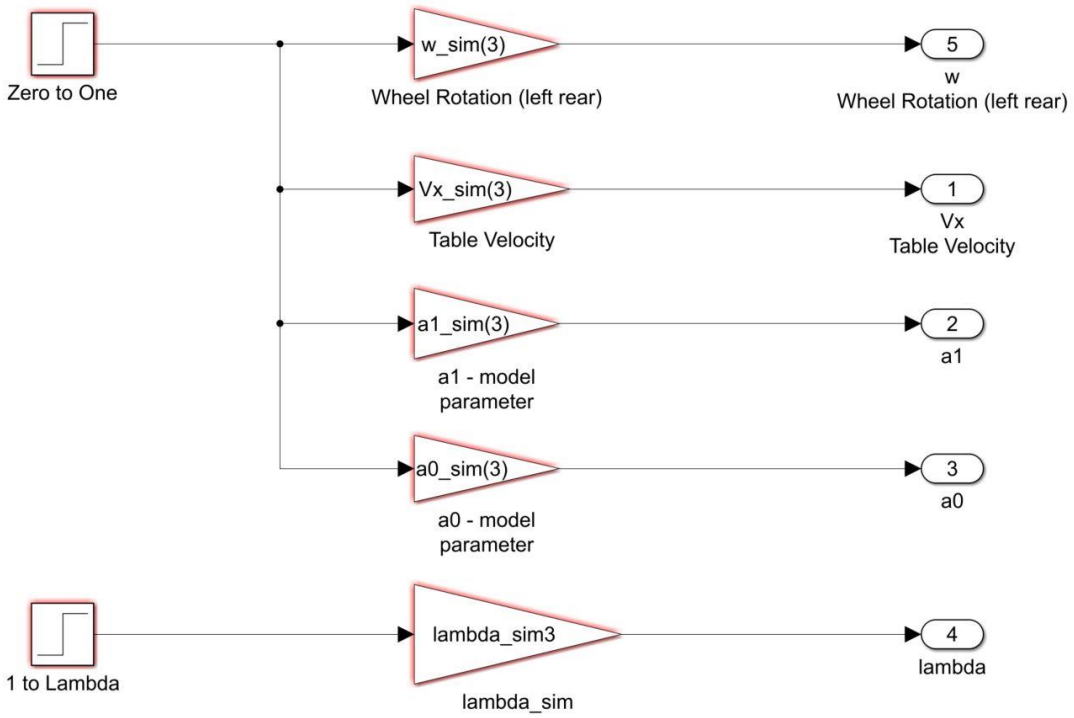


Figure 5.6: Initialisation block subsystem.

Beginning with the initialisation blocks, the inner workings the subsystem's purpose is to initialise certain parameters. Parameter initialisation is accomplished by the use of step input blocks which provide the signal that SimMechanics can interpret. For example, the step input of 0 to 1, simply turns on the signal. To then give each variable a real value, the step input is multiplied by a gain where the gain is the value of that variable. For this model, the gains include the wheel rotation or angular velocity of the wheel, along with model tuning parameters, a_0 and a_1, which were determined by Irani et al [37] in the development of this terramechanics model and represent the values b_0 and b_1 from Equation 5.3. Table velocity is included as an artefact of how Irani et al's terramechanics model was developed; however, it is irrelevant to the dynamic model of a rover traversing terrain instead of a table and is therefore not applied in the overall model.

Lambda requires a different step input from 1 to the value of lambda and is subsequently multiplied by the gain of the lambda simulation value to provide lambda or the ratio of the measured sinkage to track depth *(η*) as was discussed in Section, 5.2. Once initialised, all four parameters are then exported for use and their signals taken to the next block.

These parameters are then input to the blue terramechanics model block. However, before examining that block, there are two inputs of slip and sinkage that should be discussed. Examining the sinkage calculation box, it can be observed from the overall diagram that the required inputs are the force normal to the terrain and the angular velocity of the wheel. Figure 5.7 reveals the inner workings of the sinkage calculation box.

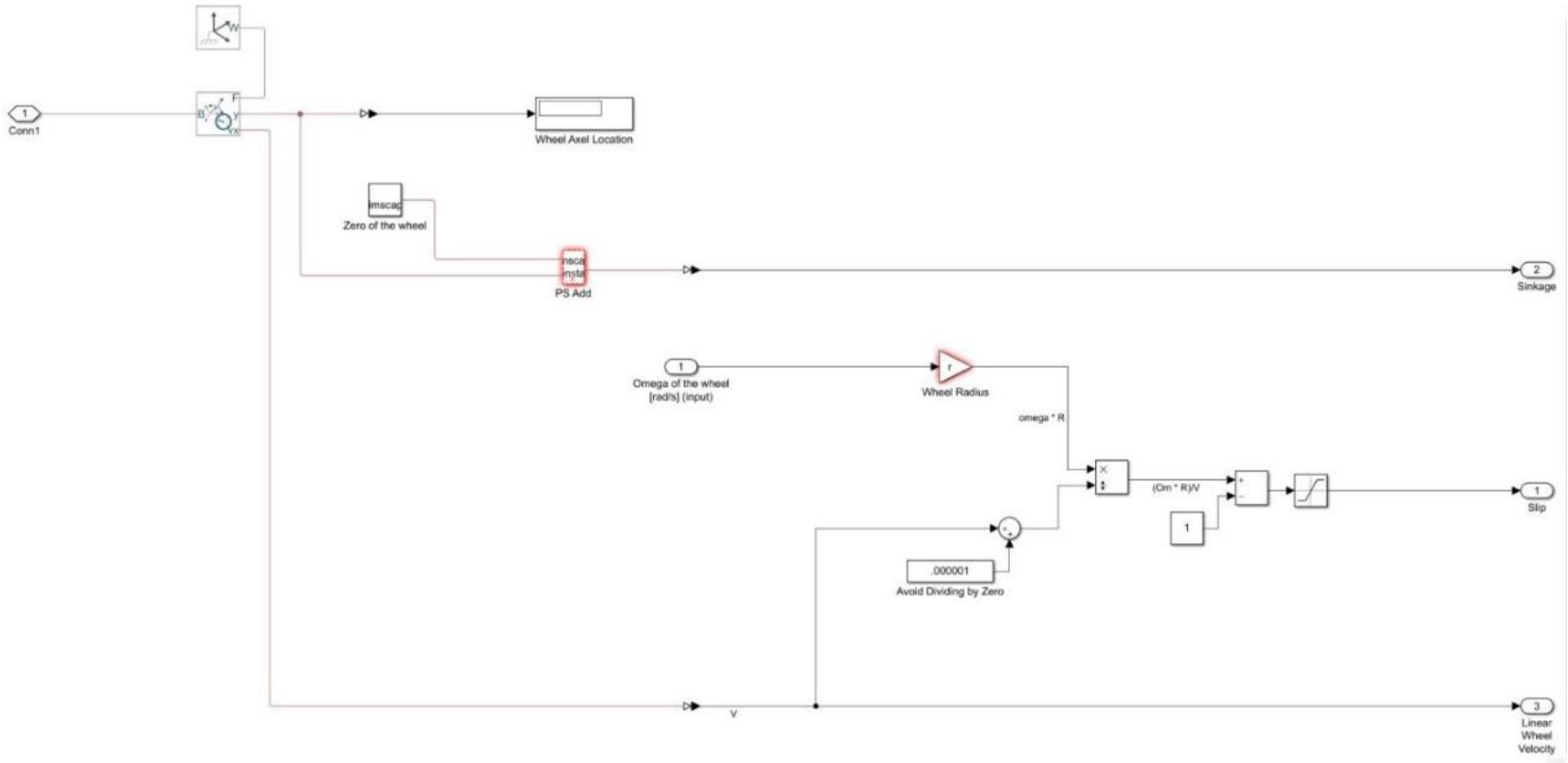


Figure 5.7: Wheel sinkage calculation subsystem.

From Figure 5.7, the first input is the normal or, in the flat terrain case, vertical force. This force gets passed through a transform sensor block which measures the vertical location of the wheel axle, along with the force in regard to the world coordinate frame and the horizontal (x direction) linear translational wheel velocity. The axle location is sent to a scope box for visualisation and is also compared to the reference or zero location of the axle, allowing the overall displacement or sinkage to be computed. Meanwhile the measured horizontal, translational velocity of the wheel is then combined with the commanded wheel velocity, which is the product of ωr, in the block diagram version of the slip equation. The version presented here is the translation velocity, v_x, as the denominator, hence the addition block where a very small decimal value (1e-6) is added to prevent a scenario of dividing by zero in the event of 100% slip. This preventative measure is also why there is a saturation block applied before the output, to effectively round the slip ratio to the correct value. The slip and sinkage values are then outputs which are sent directly to the terramechanics model.

The blue box containing the terramechanics model is not a subsystem like the other blocks, but rather represents the overall MATLAB script and nested scripts which perform the necessary recursive terramechanics analysis. The terramechanic model script was provided by the Multi-Domain-Laboratory at Carleton University and the scripts are based on the work of Irani et al [37]. Following the overview of terramechanics in the previous section, the analysis begins with computing the wheel sinkage angles as they are the foundation for the remaining equations. With the angles computed, the

terramechanics model then proceeds to compute the horizontal longitudinal force or drawbar pull (F_x), the lateral force (F_y), and the vertical force (F_z). Each of these are computed by calling upon function files for each that compute the corresponding integrand. These function files for obtaining the integrand each begin by computing the normal stress imparted by the terrain, using the appropriate case for the given angle of Equation 5.4. With the normal stress determined, the shear stress may be obtained following computation of the soil deformation according to the properties of the soil used and the slip ratio, as noted by Equations 5.6 and 5.7. Once the appropriate normal and shear stresses have been computed, the integrands for the corresponding force are obtained, and are then returned to the overall terramechanics model for integration and multiplication with the wheel radius and width. If torque is desired, it can also be computed by taking the vertical force and applying it about the radius of the wheel, provided the model parameter, a_0, matches its simulated value.

These force values then enter the external force and torque block which takes an output force of the vertical force and applies it to the wheel axle joint, but also as an input to the sinkage calculation block, to update the sinkage. When the model is fully functional, it will be necessary to initially run it with an angular velocity of zero to allow the rover to settle into the terrain and obtain its initial sinkage value. Once the model has converged on an initial sinkage value, it can then be run for the commanded angular velocity along the selected path. It should be noted that the soil properties and tuning parameters are unchanged from Irani et al's investigation [37], and as such, even using the same simulant, additional work is needed to fine tune the terramechanics model for

the J5 rover and its set of wheels. The J5 rover is much larger, with wheels of radius 0.3 m as opposed to the original Irani et al rover of 0.075 m [37]. The configuration differences require the opportunity to test the real rover in a sandbox of that simulant, or to at least take a wheel and put it in a single wheel test rig to collect data for validation and establishment of tuning parameters. Since this additional work is needed to get the overall model working in a useful way, the full model could not be completed within the time constraints of this thesis. However, a single wheel model was created using the same features as the J5 rover and the same terramechanics model and run for a sinusoidal terrain, in an attempt to further demonstrate how this model would work and what overall results are likely to occur.

5.4 Terramechanics Model Modifications

Although the previous section has discussed the combination of the terramechanics and dynamic model and how it would conceptually work, a deeper understanding can be acquired by examining the application in the context of a single wheel. Subsequent modifications are proposed for the terramechanics portion in order to accommodate the effects of non-flat terrain. By limiting the examination to a single wheel, one can more easily see the impact of the terrain shape in addition to the other parameters of the terramechanics analysis. One can recall from the classic physics problem of a box being forced up an inclined plane that the net normal and tangential

forces are impacted by the shape, specifically the slope of the incline, thus one would expect the terrain to affect the force components.

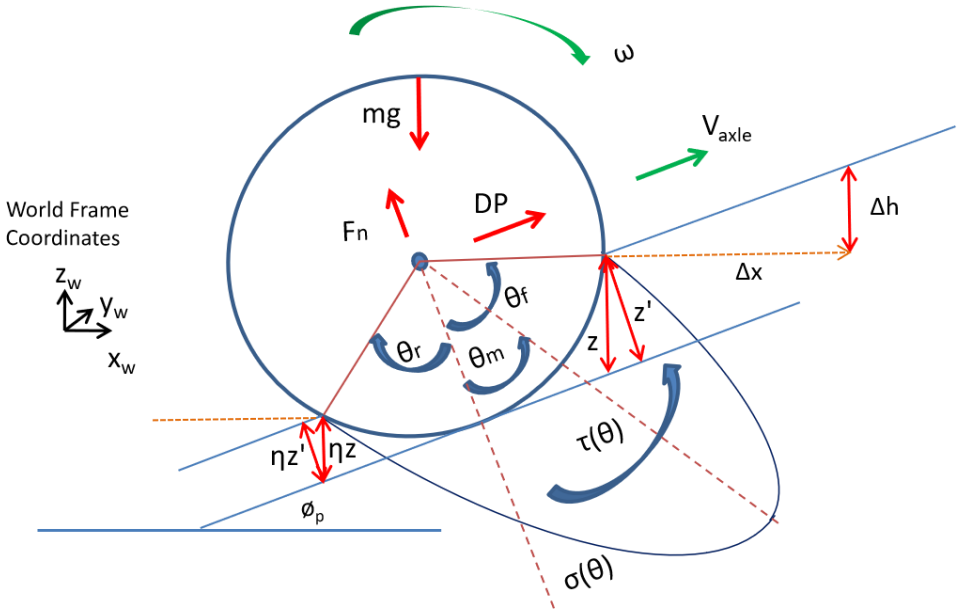


Figure 5.8: Modified terramechanics model.

Figure 5.8 displays the components of the terramechanics model for a non-flat terrain. Similar to Figure 5.3, the same parameters are present, however the addition of non-flat terrain like a sinusoidal function or an incline will affect the values of certain parameters. From the diagram it can be observed that the wheel is travelling along the more complicated terrain of a sinusoidal function, where the height and slope of the terrain are constantly changing with progression in the horizontal (global x) direction. From Section 5.3 and examination of Figure 5.8, it can be seen that the sinkage (z) is defined as the vertical distance from the top of the wheel-soil interface at the front of the wheel and to the bottom of the wheel's contact point. Subsequently, it must be

emphasized that the terramechanics formulae require a sinkage input that aligns with the normal force, as illustrated by z' in Figure 5.8. Furthermore, for non-flat terrain such as a sinusoidal curve, as the wheel progresses over the terrain and the terrain height changes, the sinkage values will be affected. Non-flat terrain will require modification to the sinkage value prior to computing the terramechanics. From the equations presented in Section 5.3, one can likewise ascertain that the wheel sinkage values will change as a result of the modification to sinkage, most notably for the front wheel sinkage angle (θ_f). If, for example, the wheel is climbing a sine curve, the increase in modified sinkage as a result of adding a positive terrain height differential, will cause the ratio of sinkage to wheel radius will increase, causing the entry wheel sinkage angle to increase (as the ratio approaches the value of 1, θ_f approaches $\frac{\pi}{2}$). With wheel sinkage values such as θ_f affected, the values determined by subsequent terramechanics equations in the sequence will also be affected because the wheel sinkage values appear in all remaining equations. For example, the location of the maximum normal stress will be increased for an increase in the front wheel sinkage angle, and both of these angles are used in the subsequent soil deformation, followed by stress computations, and finally in determining the forces. Sinkage is the value that's directly affected by terrain and, in turn, also affects the other quantities computed in a terramechanics analysis, hence sinkage is what needs to be modified for a non-flat terrain.

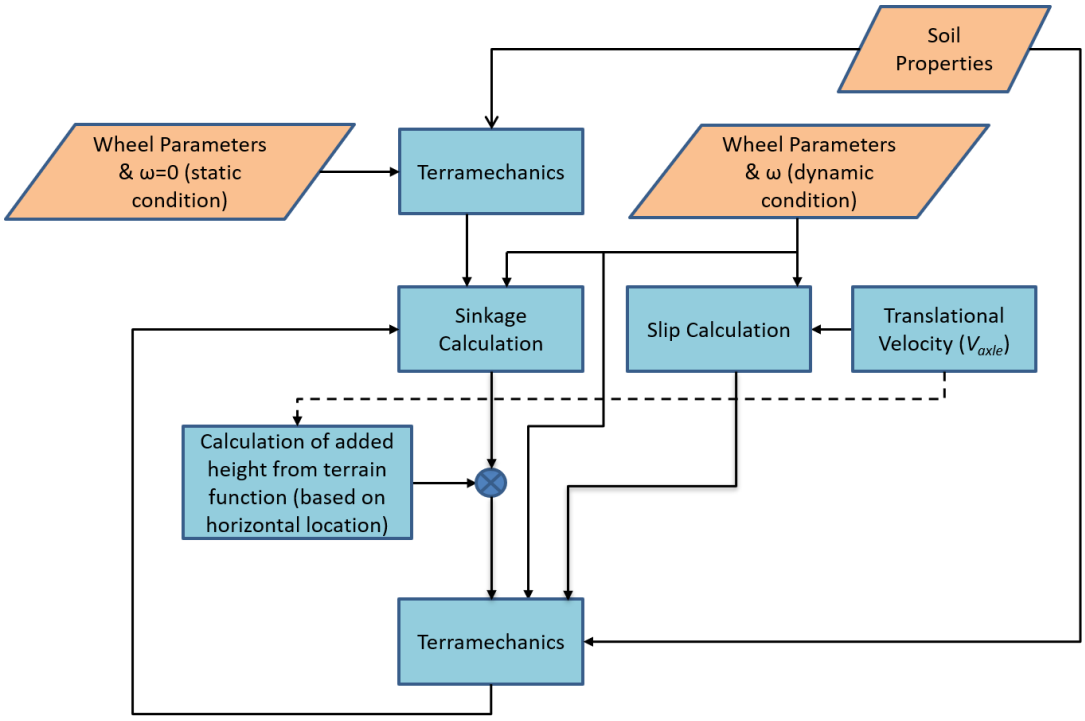


Figure 5.9: Modified single wheel terramechanics testbed model code architecture.

Figure 5.9 presents a visual walkthrough of how the terramechanics model works when applied to a single wheel and with the modifications added to sinkage. Upon review of the Figure 5.9, one should note that there are two terramechanics blocks present. These two blocks indicate that there are essentially two separate phases or applications of the terramechanics model. The first phase is for a commanded angular velocity of zero. The rationale is to let the rover wheel to sink into the terrain on account of its own weight as it would in reality prior to embarking on a traverse and establishes the static sinkage of the system. The terramechanics model in the second phase involves computing the forces and sinkage for the rover in motion, and as such is dynamic sinkage. As shown in the code architecture diagram, the increase in terrain height is added to the sinkage after the sinkage calculation (involving the vertical terrain force and weight/load on the wheel)

and prior to its input to the terramechanics model and set of equations. Following the processes outlined in the flowchart in Figure 5.9, the SimMechanics model would be initiated through a driver or MATLAB script file and the numerical solution process would be as follows:

1. SimMechanics model is initiated through the driver file. Constant variables such as soil properties/characteristics and wheel parameters are loaded into SimMechanics.
2. Through time delay settings in the step input signal generation blocks in the initialisation, the step inputs to initialise quantities such as the commanded angular velocity are delayed for a specified time (t= 3 seconds). This delay allows the model to initially run for the static case ($\omega = 0$). Thus, no slip will be calculated, and the wheel will remain at its initial coordinates, as the rover wheel sinks into the terrain as it would naturally, settling on a static sinkage value in response to the weight as opposed to the normal force imparted by the terrain.
3. Once the rover wheel has reached steady state and converged on a value for static sinkage, the rest of the input parameters are used at their dynamic value (no longer 0) and are once again fed into the terramechanics model. However, now that the velocity of the axle is non-zero, slip is computed as a non-zero input for the terramechanics equations. Prior to entering the terramechanics model in phase 2 of Figure 5.9, the sinkage (now dynamic sinkage) is calculated based on the vertical force of the terrain from the previous iteration of the terramechanics model, along with the weight of the wheel load and any damping terms. It is after this calculation that the terrain height for the given horizontal location is added to

the sinkage, in order to account for the change in terrain elevation. The modified sinkage is then input to the terramechanics model.

4. The terramechanics model computes wheel sinkage values using the modified sinkage, followed by the normal stress, soil deformation, and shear stress. The resulting normal and shear stresses are then used to compute the normal and tangential forces (ie. F_z, F_y, F_x).
5. Forces, torque (if computed), and sinkage are all outputs that can then be fed into a dynamic model if one is connected. For the full rover model, these parameters would be applied at joint blocks for the wheel axles. These values are also displayed as recorded outputs. Depending upon the dynamic model or user preference, the forces can be resolved along the terrain or in the global world frame.
6. The normal (vertical if on flat terrain) force, F_z, is then returned to the sinkage calculator where it is combined with the weight/load on the wheel and any damping included in the model to produce an updated sinkage value. The velocity of the axle is obtained and used to update the wheel axle's location on the terrain, which allows for the new contact position to be determined. Using the new location and the terrain function, the height of the terrain that needs to be added to the next calculation of sinkage is computed.
7. SimMechanics ends upon completing all of the time steps for the simulation.

With regards to expected results, it was previously mentioned that the increase in the terrain height (ie. for climbing a sine curve) would increase the sinkage value and subsequent entry sinkage angle (increases as ratio of sinkage to wheel radius approaches 1). These changes would also shift the location of the maximum normal stress, while soil deformation increases. The drawbar pull should increase with slope, as the resulting increase in entry sinkage value would alter the components of the integrals of the stress.

5.4.1 Preliminary Test Results for the Argo J5 Rover

The single wheel terramechanics model was initially run for the J5 wheel parameters and the wheel's mass only. Subsequent test runs used the ¼ vehicle mass that each wheel was expected to experience. All test results are included in Appendix D, while test results at the higher mass are included herein, for slip values of 0.05 and 0.25.

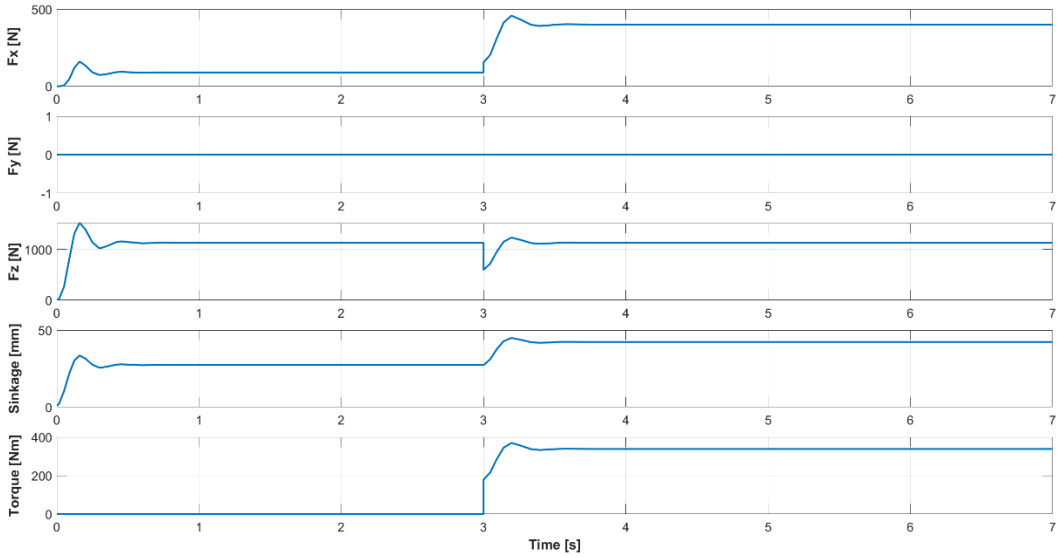


Figure 5.10: Single wheel terramechanics model for Argo J5 at 0.05 slip (flat terrain).

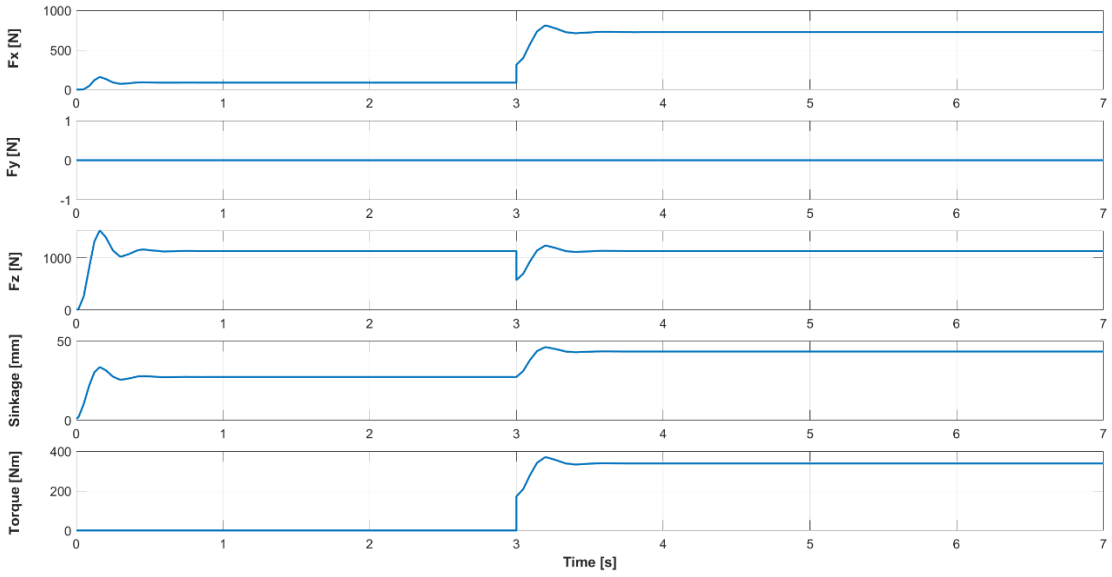


Figure 5.11: Single wheel terramechanics model for Argo J5 at 0.25 slip (flat terrain).

Figures 5.10 and 5.11 illustrate the outputs from the terramechanics model over a total simulation time of 7 seconds. The respective curves for each output parameter are

similar in shape between the two slip values, with different magnitudes. The initial settling period corresponds to 3 seconds and represents the static sinkage phase prior to motion. Following the settling period, the commanded angular velocity of the wheel initiates a transient startup, prior to reaching a steady state for each individual parameter.

Table 5.1: Terramechanics output parameters for various slip values for the Argo J5 rover (flat terrain).

Slip, i	F_x [N]	F_z [N]	Static Sinkage [mm]	Dynamic Sinkage [mm]	Torque [Nm]
0.05	401.2	1128	27.35	42.26	338.4
0.1	517.1	1128	27.35	42.66	338.4
0.25	726.0	1128	27.35	43.39	338.4
0.5	845.0	1128	27.35	44.06	338.4

Table 5.1 summarises the output of the single wheel terramechanics model for the Argo J5 rover over different slip values. The normal force, F_z, static sinkage, and torque values remain constant as expected since these results depend on the terrain properties and the weight of the vehicle, which do not change. With regards to dynamic sinkage, one can observe that this value increases with slip, as is typical of the slip-sinkage effect. The drawbar pull also increases with slip as expected, noted by Wong [29]. It should be noted that similar effects were produced for the original wheel that the model was built for. The current results require experimental validation to determine whether any tuning parameters are necessary.

The modifications proposed in Section 5.5 were implemented for a sinusoidal terrain, matching the sine function used in Chapter 4, Sections 4.2 and 4.3. Sample results for the Argo J5 rover are illustrated in Figures 5.12 and 5.13.

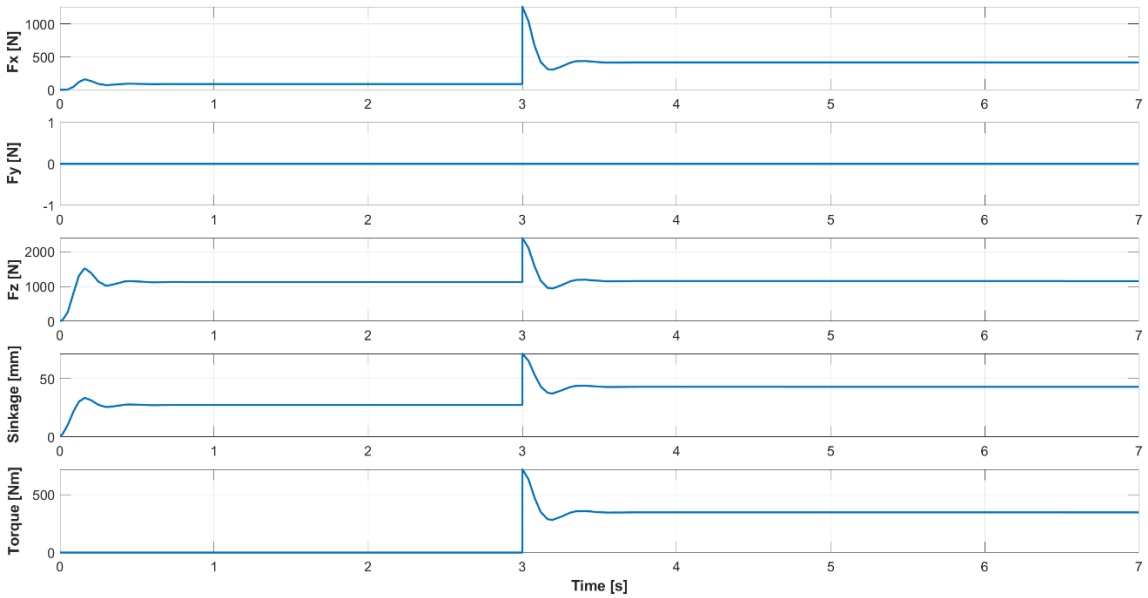


Figure 5.12: Modified single wheel terramechanics model for Argo J5 at 0.05 slip (sinusoidal terrain).

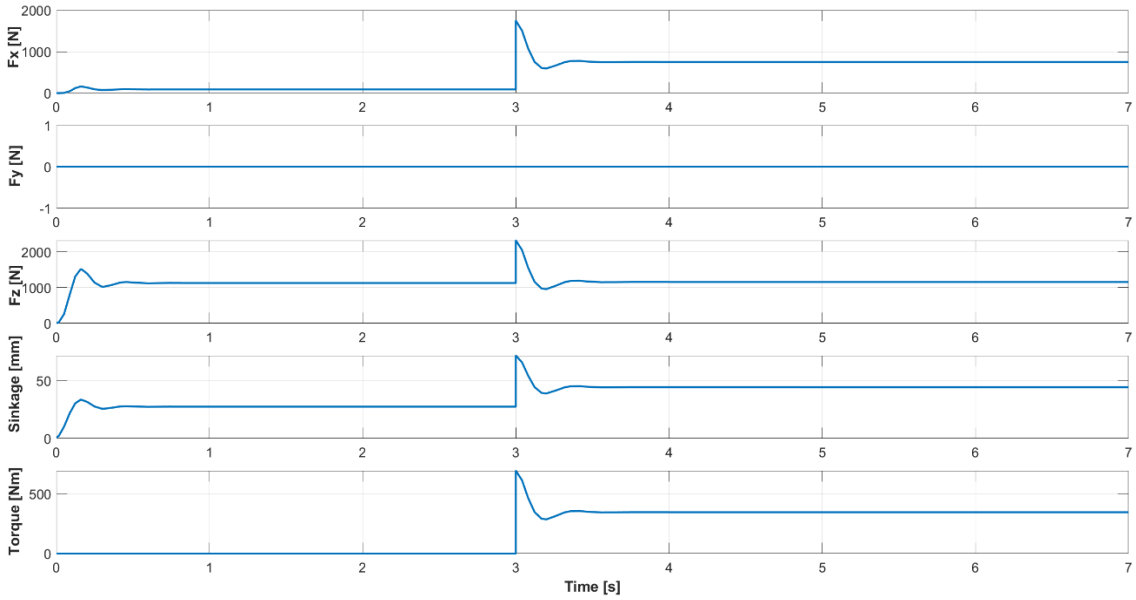


Figure 5.13: Modified single wheel terramechanics model for Argo J5 at 0.25 slip (sinusoidal terrain).

Examination of Figures 5.12 and 5.13 reveal similar results between slip values of 0.05 and 0.25 for a total simulation time of 7 seconds. Again, there is a 3 second static sinkage phase to allow the rover to settle into the terrain as it would in reality, followed by the step change to the dynamic sinkage phase where the rover begins to move. It is noted that the static to dynamic transient of the curves is significantly different in the modified model as compared to the unmodified model. The modified model appears to illustrate a transient zone with less damping and a higher initial overshoot.

It should be noted that the results illustrated in Figures 5.12 and 5.13 only cover the first 4 seconds of motion (dynamic phase). These figures have the appearance of flat curve results, in spite of a sinusoidal terrain input.

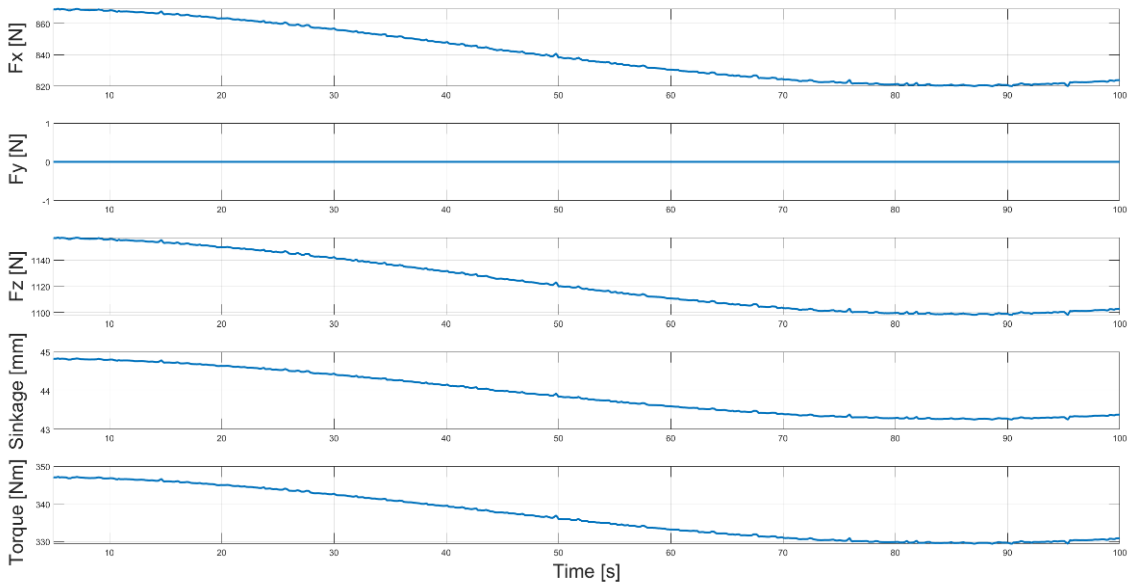


Figure 5.14: Dynamic phase of modified single wheel terramechanics model for Argo J5 at 0.25 slip (sinusoidal terrain), extended to 100 seconds.

A supplemental examination of the 0.25 slip case for the modified terramechanics model, as shown in Figure 5.14, extended the traverse duration to 100 seconds. Neglecting the static phase and the transition to the dynamic phase, Figure 5.14 illustrates the full effect of the input terrain. As expected, the sinkage varies with the shape of the terrain which, in turn, propagates through the terramechanics formulations to affect the resulting torque and forces.

Table 5.2: Modified terramechanics output parameters for various slip values for the Argo J5 rover (sinusoidal terrain).

Slip, i	F_x [N]	Static F_z [N]	Dynamic F_z [N]	Static Sinkage [mm]	Dynamic Sinkage [mm]	Torque [Nm]
0.05	416.7	1128.2	1156.8	27.35	43.00	347.0
0.1	535.5	1128.2	1156.8	27.35	43.41	347.0
0.25	748.7	1128.2	1156.7	27.35	44.14	347.0
0.5	868.9	1128.2	1156.7	27.35	44.82	347.0

Table 5.2 summarises the resulting output from the modified terramechanics model. Similar to the unmodified model, the drawbar pull and dynamic sinkage increase with slip. However, the magnitude of these parameters with respect to the same slip case for the unmodified model is slightly higher. This shift is due to the upward slope of the terrain which, from the proposed modification, adds to the sinkage.

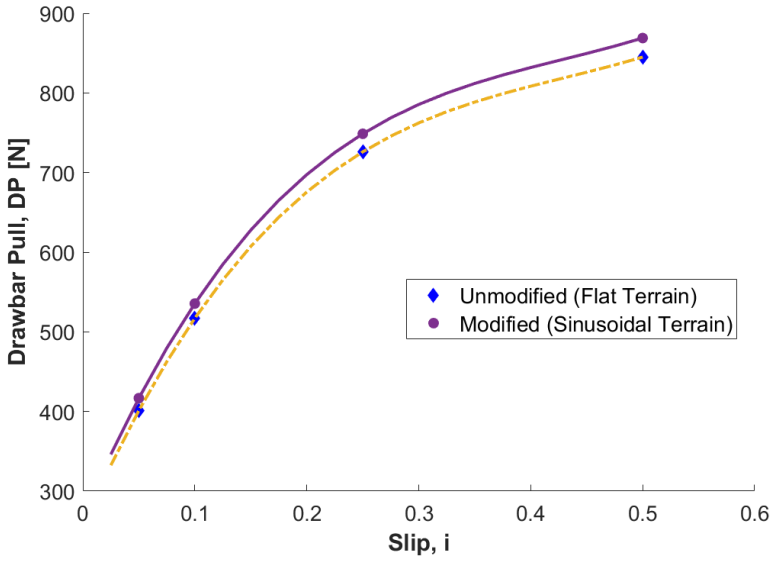


Figure 5.15: Drawbar pull vs slip.

Figure 5.15 depicts the relationship between drawbar pull and slip for both the modified and unmodified models. The curves illustrated are characteristic for drawbar pull with respect to slip and widely noted in the literature [19, 34, 42, 58]. It is noted that the curve for the modified model is higher than the unmodified model and the difference between the two increases with slip.

Overall the proposed modification to the terramechanics model of Irani et al [37], to accommodate for non-flat terrain, demonstrates the trends expected from the analysis herein. However, additional work is required to validate the modified model experimentally to incorporate any necessary tuning parameters. It should also be noted that a major limitation of using a terramechanics model such as this is that it becomes inaccurate for high levels of slip. Should one wish to model the dynamics of the J5 rover for high levels of slip, incorporating the forces and moments produced by the wheel-soil interaction for a given terrain, an alternate method such as DEM or its associated look-up tables would need to be employed.

Chapter 6: Conclusions and Future Work

Planetary exploration rovers are designed to traverse a range of terrain conditions and manage unstructured environments of another world that cannot fully be predicted. In particular, the more occluded hazards such as slip and sinkage, encompassed by the unique characteristics of the terrain, are most challenging to anticipate and require predictive modelling to assess rover capabilities. In this thesis an investigation was carried out to assess how to develop kinematic and dynamic models for the four-wheel Argo J5 rover.

The conclusions drawn from this work are organised to address each of the research objectives from Chapter 1 and are presented below. A selection of general comments are also included. It should be noted that the Maple files referred to in Chapters 4 and 5 are not included in this document due to file size limitations. They can be made available upon request.

6.1 Objective 1

Produce a model to determine the pose of the rover in response to terrain geometry (ie. slopes & bumps, etc).

Objective 1 was accomplished for varying terrain geometry, as outlined in Chapter 4, Sections 4.1 and 4.2. Due to the nature of the three-dimensional velocity

analysis and how it directly utilises the pose generated from solving the inverse three-dimensional position kinematics analysis, it was also achieved as part of the velocity kinematics model. Similarly, the dynamic model can provide information regarding the pose of the rover.

However, while the planar position model (Section 4.1) was able to visually model the pose of the J5 rover and conform to the terrain function, it is limited in application as it is planar and not suitable for actual, three-dimensional terrains without significant modifications. As such, it does not capture the full behaviour of the rover.

The three-dimensional position kinematics was used to model the full rover and demonstrate the interdependence of some joints in response to terrain disturbance input. The results obtained showed good agreement with expected trends (ie. chassis angles with respect to the terrain) and accurately predicted the total distance the centre of the chassis travelled with respect to manually calculated estimates for the selected step intervals. Other manual calculations of joint positions also showed agreement. The results of the model show agreement to within 3% for the speeds and distances on various terrains relative to the flat terrain case, for all values of slip studied. However, for best accuracy the model would need to be validated experimentally for the entire rover. Once determined, the pose for each step interval on various terrains can then be compared with the joint displacements for the specific rover that could result in tipping or exceed other functional limitations. A sub-objective was also achieved in this model as its executed code, although written in MATLAB, can be fairly easily transferred to another

programming language of choice. There were some limitations in achieving this objective as the accuracy is limited to that of manually estimated joint displacements for a given terrain. Subsequently, this limitation makes the more complex terrains harder to compute and estimate accuracy. Furthermore, it was noted that convergence was initially difficult to achieve due to the number of joints and equations causing the problem to be over constrained, requiring careful thought as to which equations could be safely removed. Finally, another limitation with regards to the solver used for this method is that the Jacobian of the system will never be square. Thus, a solver that utilises an algorithm capable of handling non-square Jacobians is required or further manipulation of the Jacobian matrix would be necessary.

6.2 Objective 2

Produce a three-dimensional velocity kinematics model incorporating predicted slip to accurately determine the progression of the rover on the terrain.

Building on the application of the original D-H convention [66] and the determined equation sets capable of describing the rover's pose, the velocity Jacobians were generated for each wheel-ground contact point, providing the equation sets required to analyse the velocity of the J5 rover as it travels over the terrain. The resulting equation sets were successfully combined with the results from the position analysis using the inputs of terrain path coordinates, along with the inclusion of slip read from the terrain path data. The model was able to simulate traverses along different test case terrains for

varying slip values, with results displaying the expected trends. Incorporating slip, the model was able to accurately drive the rover over the terrain with higher slip values resulting in less overall ground being covered, as expected. The pose was again determined to a high degree of accuracy, whereas the individual joint rates were less accurate, requiring experimental validation. Again, the program was written and executed in MATLAB, making the code flexible and more accessible. Also, the computational time is generally under a minute per time step.

However, although the objective was achieved, there are some limitations to address. As with the 3D position kinematic analysis, the solver used must be capable of handling non-square Jacobians which limits the choice of solver employed. As previously noted, the number of joints and kinematic pathways of the rover creates an over-constrained system of nonlinear equations, making convergence more difficult to achieve and resulting in some inaccuracies in the joint rates computed. The 3D velocity kinematic model requires experimental validation and tuning to account for friction and damping occurring between the joints. The actual joint rate will differ from those predicted by the 3D velocity kinematic model. Forces and accelerations are not addressed by Objective 2.

6.3 Objective 3

Produce a dynamic model, incorporating slip and terrain geometry as inputs, allowing for other traversability metrics, such as torques and drawbar pull, to be determined.

The original method of continuing with mathematical models built upon the D-H convention for describing the transformation between kinematic pairs, to include acceleration and the dynamic torque equation proved to require too many simplifications. In addition, the complexities of accurately determining the kinetic energy terms without accurate inertia information, made this method not worth pursuing. Also, the complexities would not make it user friendly in analysing the potential path for particular metrics. SimMechanics' easier generation of a dynamic model from a solid model CAD file, enables flexibility in changing the rover design and makes it easier for a generic operator to visually understand it, as opposed to complicated mathematical terms. SimMechanics also automatically generates a three-dimensional visual animation which can be manipulated for different viewpoints so there is no need to code for visualisation. The dynamic model for the J5 was able to be paired with a terramechanics model and was applied to each wheel axle, and it allowed for further inputs from the terrain geometry, or the potential to actuate other joints.

Unfortunately, Objective 3 was not fully achieved because significantly more work was needed to get the fully combined model running for inputs on all four wheels. The additional time required to troubleshoot the model, demands experimental work to determine the tuning for the correct amount of damping while also accounting for the

friction in the joints. Likewise, the terramechanics model must also be tuned for the J5 rover model (ie. wheel dimensions) and the terrains over which it will be traveling. These are direct limitations on the current model.

6.4 General Comments

Overall, the use of the D-H convention in describing the transformation between kinematic pairs, and subsequently using the transformations to build models describing the pose and motion of the rover proved to be a valid and fairly accurate approach. Although the D-H method has shown to be fairly accurate in producing three-dimensional position and velocity models that account for slip and the geometry of the terrain, this approach does require a significant time investment in the establishment of the D-H parameters to obtain the correct equation set. However, once the position-related equation set has been correctly established, these equations can be used to provide prediction of pose and progression traversability metrics to aid in path selection. They could also be applied to a feedback controller to help with path following. The models developed in this thesis examined pose, velocity, and dynamics and were developed to work in concert to provide different aspects of traversability analysis to overall rover simulation on potential paths. Depending upon the objective, the user could simply run one or more of the models to obtain the desired information. An alternative development could include all three aspects within SimMechanics alone.

6.5 Recommendations for Future Work

Based upon the literature review and work presented in this thesis, the author offers the following recommendations for possible future work:

1. Experimental validation of the kinematic models to add tuning parameters for increased accuracy of the simulation.

2. Further investigation to determine appropriate time steps for interpolation stability, or review of the potential application of non-linear interpolation techniques.

3. Activation of steering within the kinematic model, in accordance with the skid steering system of the J5 rover.

4. Perform both kinematic and dynamic simulations with complex steering maneuvers on varying complexities of terrain, such as turning on a slope, with different amounts of slip.

5. Extend the study to high values of slip, by running simulations at the higher ($i >$ 0.6) slip and validate with experimental results to best of ability.

6. Extend the study to more erratic terrain maps, including those with different terrain types (bedrock vs soft loose sand) and slip values. Also include random bumps for different sides of the rover.

7. Finalise the fully combined dynamic and terramechanics model of the Argo J5 rover. Finalisation would include the required experimental validation to generate appropriate tuning parameters, along with expanding the terrain properties in the terramechanics model to include other terrains likely to be encountered by the rover.

8. Validate the terramechanics model for the rover wheel with a single wheel testbed experiment, allowing for comparison with other models. Also generate DEM look up tables for the higher regions of slip where terramechanics breaks down.

9. Instrument a J5 rover with accelerometers to measure the forces on each wheel and selected joints during experimental drives over controlled terrain, such as that of the Canadian Space Agency's (CSA's) MarsYard.

References

1. “Odyssey Orbiter Mission Overview,” NASA, Available: https://mars.nasa.gov/odyssey/mission/overview/.
2. “Phoenix,” NASA, Available: https://mars.nasa.gov/programmissions/missions/past/phoenix/.
3. “InSight Mission Overview,” NASA, Available: https://mars.nasa.gov/insight/.
4. Northon, K., “Mars Helicopter to Fly on NASA's Next Red Planet Rover Mission,” NASA, Available: https://www.nasa.gov/press-release/mars-helicopter-to-fly-on-nasa-s-next-red-planet-rover-mission.
5. “What Is Dragonfly?”, Dragonfly, Available: http://dragonfly.jhuapl.edu/What-Is-Dragonfly/.
6. Bugby, D., Seghi, S., Kroliczek, E., and Pauken, M. “Novel Architecture for a Long-Life, Lightweight Venus Lander”, *AIP Conference Proceedings,* Vol. 1103, No. 1, United States, Mar. 2009, pp. 39 – 50. Web. doi:10.1063/1.3115545.
7. Grego, P., *Mars and how to observe it*, New York: Springer, 2012.
8. “NASA's Mars Exploration Program,” *NASA,* Available: https://mars.nasa.gov/#mars_exploration_program/3.
9. Orosei, R., Lauro, S. E., Pettinelli, E., et-al, “Radar evidence of subglacial liquid water on Mars,” *Science*, Vol. 361, No. 6401, Aug. 2018, pp. 490 – 493. doi: 10.1126/science.aar7268.

10. Carsten, J., Rankin, A., Ferguson, D., and Stentz, A., "Global Path Planning on Board the Mars Exploration Rovers," *2007 IEEE Aerospace Conference*, 2007, pp. 1–11.
11. "Moving around Mars," *NASA.* Available: https://mars.nasa.gov/mer/mission/timeline/surfaceops/navigation/
12. "Viking 1 & 2," *NASA.* Available: https://mars.nasa.gov/programmissions/missions/past/viking/.
13. Trease, B., Arvidson, R., Lindemann, R., et-al, "Dynamic Modeling and Soil Mechanics for Path Planning of the Mars Exploration Rovers," *35th Mechanisms and Robotics Conference, Parts A and B*, Vol. 6, Aug. 2011, pp. 1–11.
14. Bouguelia, M.-R., Gonzalez, R., Iagnemma, K., and Byttner, S., "Unsupervised classification of slip events for planetary exploration rovers," *Journal of Terramechanics*, Vol. 73, Sep. 2017, pp. 95–106. doi: 10.1016/j.jterra.2017.09.001.
15. Arvidson, R. E., Bell, J. F., Bellutta, P., et-al, "Spirit Mars Rover Mission: Overview and selected results from the northern Home Plate Winter Haven to the side of Scamander crater," *Journal of Geophysical Research*, Vol. 115, 2010, pp. 1–19. doi:10.1029/2010JE003633.
16. Vogt, G. L., "Chapter 1: The Two Faces of Mars," *Landscapes of Mars: A Visual Tour*, New York: Springer, 2008, pp. 27–40.
17. "Rover Wheels," *NASA.* Available: https://mars.nasa.gov/mars2020/mission/rover/wheels/

18. Lindemann, R., and Voorhees, C., “Mars Exploration Rover Mobility Assembly Design, Test and Performance,” *2005 IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2005. doi: 10.1109/ICSMC.2005.1571187.
19. Zhou, F., Arvidson, R. E., Bennett, K., Trease, B., Lindemann, R., Bellutta, P., Iagnemma, K., and Senatore, C., “Simulations of Mars Rover Traverses,” *Journal of Field Robotics*, Vol. 31, No. 1, Jan. 2014, pp. 141–160. doi: 10.1002/rob.21483.
20. Heverly, M., Matthews, J., Lin, J., Fuller, D., Maimone, M., Biesiadecki, J., and Leichty, J., “Traverse Performance Characterization for the Mars Science Laboratory Rover,” *Journal of Field Robotics*, Vol. 30, No. 6, 2013, pp. 835–846. doi: 10.1002/rob.21481.
21. Fookes, P.G., Lee, E.M., and Griffiths, J.S. *Engineering Geomorphology - Theory and Practice - 7.2 Engineering Soil Behaviour: Clays.* Whittles Publishing, 2007, pp. 47. Retrieved from: https://app.knovel.com/hotlink/pdf/id:kt00BRWZ57/engineering-geomorphology/engineering-soil-behaviour
22. Nof, S. Y., *Handbook of industrial robotics*, New York: John Wiley, 1999, p. 148.
23. “Mars Exploration Rover - Opportunity,” NASA. Available: https://www.jpl.nasa.gov/missions/mars-exploration-rover-opportunity-mer/
24. NASA Jet Propulsion Laboratory, “NASA Facts - Mars Exploration Rover,” NASA Facts - Mars Exploration Rover
25. “Rover,” NASA. Available: https://mars.jpl.nasa.gov/msl/mission/rover/.

26. "ARGO J5 XTR robot platform for extreme terrain," *ARGO Xtreme Terrain Robotics.* Available: https://www.argo-xtr.com/index.php/xtr-robots/j5-xtr/.

27. Bekker, M. G., *Off-the-road locomotion*, Ann Arbor, Michigan: The University of Michigan Press, 1960.

28. Wong, J., "An introduction to terramechanics," *Journal of Terramechanics*, Vol. 21, 1984, pp. 5–17.

29. Wong, J. Y., *Terramechanics and off-road vehicle engineering: terrain behavior, off-road vehicle performance and design*, Amsterdam: Butterworth-Heinemann, 2010.

30. Wong, J. Y., *Theory of ground vehicles*, New York: John Wiley, 2001.

31. Ding, L., Gao, H.B., Deng, Z.Q., and Tao, J.G., "Wheel slip-sinkage and its prediction model of lunar rover," *Journal of Central South University of Technology*, Vol. 17, No. 1, 2010, pp. 129–135.

32. Taheri, S., Sandu, C., Taheri, S., Pinto, E., and Gorsich, D., "A technical survey on terramechanics models for tire–terrain interaction used in modeling and simulation of wheeled vehicles," *Journal of Terramechanics*, Vol. 57, 2015, pp. 1–22.
doi: 10.1016/j.jterra.2014.08.003.

33. Smith, W., Melanz, D., Senatore, C., Iagnemma, K., and Peng, H., "Comparison of discrete element method and traditional modeling methods for steady-state wheel-terrain interaction of small vehicles," *Journal of Terramechanics*, Vol. 56, 2014, pp. 61–75.
doi: 10.1016/j.jterra.2014.08.004.

34. Johnson, J. B., Kulchitsky, A. V., Duvoy, P., Iagnemma, K., Senatore, C., Arvidson, R. E., and Moore, J., "Discrete element method simulations of Mars Exploration Rover wheel performance," *Journal of Terramechanics*, Vol. 62, 2015, pp. 31–40.
doi: 10.1016/j.jterra.2015.02.004.

35. Johnson, J. B., Duvoy, P. X., Kulchitsky, A. V., Creager, C., and Moore, J., "Analysis of Mars Exploration Rover wheel mobility processes and the limitations of classical terramechanics models using discrete element method simulations," *Journal of Terramechanics*, Vol. 73, Oct. 2017, pp. 61–71.
doi: 10.1016/j.jterra.2017.09.002.

36. Nishiyama, K., Nakashima, H., Yoshida, T., Ono, T., Shimizu, H., Miyasaka, J., and Ohdoi, K., "2D FE–DEM analysis of tractive performance of an elastic wheel for planetary rovers," *Journal of Terramechanics*, Vol. 64, Apr. 2016, pp. 23–35.
doi: 10.1016/j.jterra.2015.12.004.

37. Irani, R., Bauer, R., and Warkentin, A., "Modelling a Single-wheel Testbed for Planetary Rover Applications", *Proceedings ASME Dynamic Systems and Control Conference*, Sept. 2010, pp. 181–188.

38. Irani, R., Bauer, R., and Warkentin, A., "A dynamic terramechanic model for small lightweight vehicles with rigid wheels and grousers operating in sandy soil," *Journal of Terramechanics*, Vol. 48, No. 4, Aug. 2011, pp. 307–318.
doi: 10.1016/j.jterra.2011.05.001.

39. Irani, R. A., Bauer, R. J., and Warkentin, A., "Dynamic Wheel-Soil Model for Lightweight Mobile Robots with Smooth Wheels," *Journal of Intelligent & Robotic Systems*, Vol. 71, No.2, Aug. 2012, pp. 179–193. doi: 10.1007/s10846-012-9777-3.

40. Ghotbi, B., Azimi, A., K vecses Jozsef, and Angeles, J., "Sensitivity Analysis of Mobile Robots for Unstructured Environments," *8th International Conference on Multibody Systems, Nonlinear Dynamics, and Control, Parts A and B*, Vol. 4, 2011, pp. 1–7.

41. Iagnemma, K., Senatore, C., Trease, B., Arvidson, R., Bennett, K., Shaw, A., Zhou, F., Dyke, L. V., and Lindemann, R., "Terramechanics Modeling of Mars Surface Exploration Rovers for Simulation and Parameter Estimation," *8th International Conference on Multibody Systems, Nonlinear Dynamics, and Control, Parts A and B*, Vol. 4, Aug. 2011, pp. 1–8.

42. Ishigami, G., Miwa, A., Nagatani, K., and Yoshida, K., "Terramechanics-based model for steering maneuver of planetary exploration rovers on loose soil," *Journal of Field Robotics*, Vol. 24, No. 3, Mar. 2007, pp. 233–250. doi: 10.1002/rob.20187.

43. Chhaniyara, S., Brunskill, C., Yeomans, B., Matthews, M., Saaj, C., Ransom, S., and Richter, L., "Terrain trafficability analysis and soil mechanical property identification for planetary rovers: A survey," *Journal of Terramechanics*, Vol. 49, No. 2, Apr. 2012, pp. 115–128. doi: 10.1016/j.jterra.2012.01.001.

44. Ding, L., Yang, H., Gao, H., Li, N., Deng, Z., Guo, J., and Li, N., "Terramechanics-based modeling of sinkage and moment for in-situ steering wheels of mobile robots on deformable terrain," *Mechanism and Machine Theory*, Vol. 116, Oct. 2017, pp. 14–33. doi: 10.1016/j.mechmachtheory.2017.05.011.

45. Gallina, A., Krenn, R., and Schäfer, B., "On the treatment of soft soil parameter uncertainties in planetary rover mobility simulations," *Journal of Terramechanics*, Vol. 63, Feb. 2016, pp. 33–47. doi: 10.1016/j.jterra.2015.08.002.

46. Tarokh, M., Mcdermott, G., Hayati, S., and Hung, J., "Kinematic modeling of a high mobility Mars rover," *Proceedings 1999 IEEE International Conference on Robotics and Automation*, May 1999, pp. 992–998. doi: 10.1109/ROBOT.1999.772441.

47. Chakraborty, N., and Ghosal, A., "Kinematics of wheeled mobile robots on uneven terrain," *Mechanism and Machine Theory*, Vol. 39, No. 12, Dec. 2004, pp. 1273–1287. doi: 10.1016/j.mechmachtheory.2004.05.016.

48. Tarokh, M., and McDermott, G., "Kinematics modeling and analyses of articulated rovers," *IEEE Transactions on Robotics*, Vol. 21, No. 4, Aug. 2005, pp. 539–553. doi: 10.1109/TRO.2005.847602.

49. Mcdermott, G., and Tarokh, M., “A General Approach to Kinematics Modeling of All-Terrain Rovers,” *2005 IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2005.
doi: 10.1109/ICSMC.2005.1571445.

50. Auchter, J., Moore, C. A., and Ghosal, A., “A Novel Kinematic Model for Rough Terrain Robots,” *Lecture Notes in Electrical Engineering Advances in Computational Algorithms and Data Analysis*, Springer, Dordrecht, 2009, pp. 215–234.

51. Parakh, S., Wahi, P., and Dutta, A., “Velocity kinematics based control of rocker-bogie type planetary rover,” *TENCON 2010 - 2010 IEEE Region 10 Conference*, Nov. 2010, pp. 939–944.
doi: 10.1109/TENCON.2010.5686545.

52. Tarokh, M., Ho, H. D., and Bouloubasis, A., “Systematic kinematics analysis and balance control of high mobility rovers over rough terrain,” *Robotics and Autonomous Systems*, Vol. 61, No. 1, Jan. 2013, pp. 13–24.
doi: 10.1016/j.robot.2012.09.010.

53. Seegmiller, N., and Kelly, A., “Enhanced 3D Kinematic Modeling of Wheeled Mobile Robots,” *Robotics: Science and Systems X*, 2014.

54. Srividhya, G., Sharma, G., and Kumar, H. N. S., “Software for Modelling and Analysis of Rover on Terrain,” *Proceedings of Conference on Advances In Robotics - AIR 13*, July 2013, pp. 1–8.
doi: 10.1145/2506095.2506112.

55. Bauer, R., Barfoot, T., Leung, W., and Ravindran, G., "Dynamic Simulation Tool Development for Planetary Rovers," *International Journal of Advanced Robotic Systems*, Vol. 5, No. 3, Sept. 2008, pp. 311–314.
doi: 10.5772/5609.

56. Li, W., Ding, L., Gao, H., Deng, Z., and Li, N., "ROSTDyn: Rover simulation based on terramechanics and dynamics," *Journal of Terramechanics*, Vol. 50, No. 3, June 2013, pp. 199–210.
doi: 10.1016/j.jterra.2013.04.003.

57. Reina, G., and Foglia, M., "On the mobility of all-terrain rovers," *Industrial Robot: An International Journal*, Vol. 40, No. 2, Mar. 2013, pp. 121–131.

58. Senatore, C., Stein, N., Zhou, F., Bennett, K., Arvidson, R. E., Trease, B., Lindemann, R., Bellutta, P., Heverly, M., and Iagnemma, K., "Modeling and Validation of Mobility Characteristics of the Mars Science Laboratory Curiosity Rover," 2014.

59. Schäfer, B., Gibbesch, A., Krenn, R., and Rebele, B., "Planetary rover mobility simulation on soft and uneven terrain," *Vehicle System Dynamics*, Vol. 48, No. 1, 2010, pp. 149–169.
doi: 10.1080/00423110903243224.

60. Gonzalez, R., and Iagnemma, K., "Slippage estimation and compensation for planetary exploration rovers. State of the art and future challenges," *Journal of Field Robotics*, Vol. 35, No. 4, June 2018, pp. 564–577.
doi: 10.1002/rob.21761.

61. Helmick, D., Angelova, A., and Matthies, L., “Terrain Adaptive Navigation for planetary rovers,” *Journal of Field Robotics*, Vol. 26, No. 4, Apr. 2009, pp. 391–410.
doi: 10.1002/rob.20292.

62. Thueer, T., and Siegwart, R., “Mobility evaluation of wheeled all-terrain robots,” *Robotics and Autonomous Systems*, Vol. 58, No. 5, May 2010, pp. 508–519.
doi: 10.1016/j.robot.2010.01.007.

63. Richards, K. L., “34.3 Kinematic Definitions”, *Design Engineer's Sourcebook*, CRC Press, 2018, p. 820.
Retrieved from: https://app.knovel.com/hotlink/pdf/id:kt011MJYJG/design-engineers-sourcebook/mechanism

64. Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G., ‘Kinematics’, *Robotics Modelling, Planning and Control*, Springer, London, 2009, pp. 39–103.

65. Craig, J. J., *Introduction to Robotics: Mechanics and Control*. 2nd ed., Addison-Wesley, Reading, Mass., 1989.

66. Denavit, J., Hartenberg, R. S., ″A kinematic notation for lower-pair mechanisms based on matrices," *Trans ASME Journal of Applied Mechanics,* Vol. 23, 1955, pp. 215–221.

67. Spong, M. W., Hutchinson, S., and Vidyasagar, M., “Chapter 9 DYNAMICS,” *Robotics, Dynamics and Control*, 2nd ed., Wiley, Jan. 2004, pp. 187–224.

68. “Fusion 360: Free Software for Hobbyists,” *Autodesk.* Available: https://www.autodesk.com/campaigns/fusion-360-for-hobbyists.

Appendices

Appendix A - Argo J5 Data and Specifications

All rover data provided to the author is collected in this appendix.

A.1 Argo J5 Rover Data Summary

Rover: Argo J5 Rover (also manufactured/sold by Clearpath Robotics)[1]

***Total Mass*:** 460 kg (1013 lbs) – includes wheels, batteries, mast, sensors.

Total Envelope/Footprint: 1.52 m long x 1.48 m wide

Steering: Skid Steering

Special notes: - back suspension connecting walking beams to chassis (chassis connection at rear) averages out the pitch of each walking beams to get the pitch of the chassis.

Main Components

Chassis

- Mass: 200 kg
- Dimensions: 1.30 m long x 0.77 m wide

Walking Beams

- Mass: 50 kg (each) x2
- Dimensions: 1.32 m long x 0.29 m wide

Wheels[2]:

- Mass: 15 kg (each) x4
- Diameter: 0.60 m
- Thickness: 0.30 m

Note: In text of thesis, refer to it as the Argo J5 rover based on the language used by the company in possession of the rover (Mission Control Space Services) and special advisor to the project, Dr. M. Faragalli.

For more information, see:

Argo J5XTR website: https://www.argo-xtr.com/index.php/xtr-robots/j5-xtr/

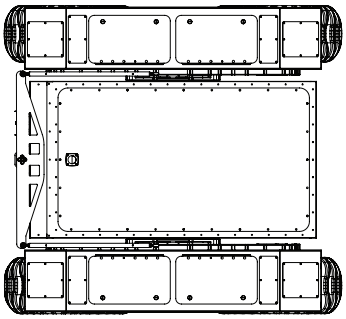
Clearpath Robotics Warthog: https://clearpathrobotics.com/warthog-unmanned-ground-vehicle-robot/

1. Rover manufacturer spec sheet & drawing attached. (https://clearpathrobotics.com/warthog-unmanned-ground-vehicle-robot/)
2. Wheels can be either be metal or rubber. Metal tires assumed for the analysis.

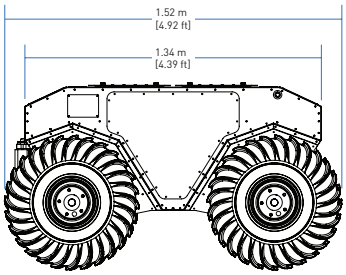
WARTHOG™

AMPHIBIOUS UNMANNED GROUND VEHICLE

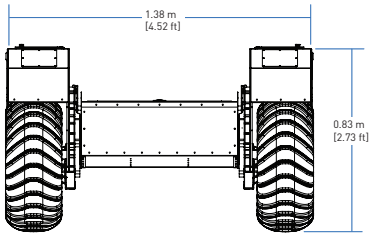
CLEARPATH ROBOTICS™



TOP



SIDE



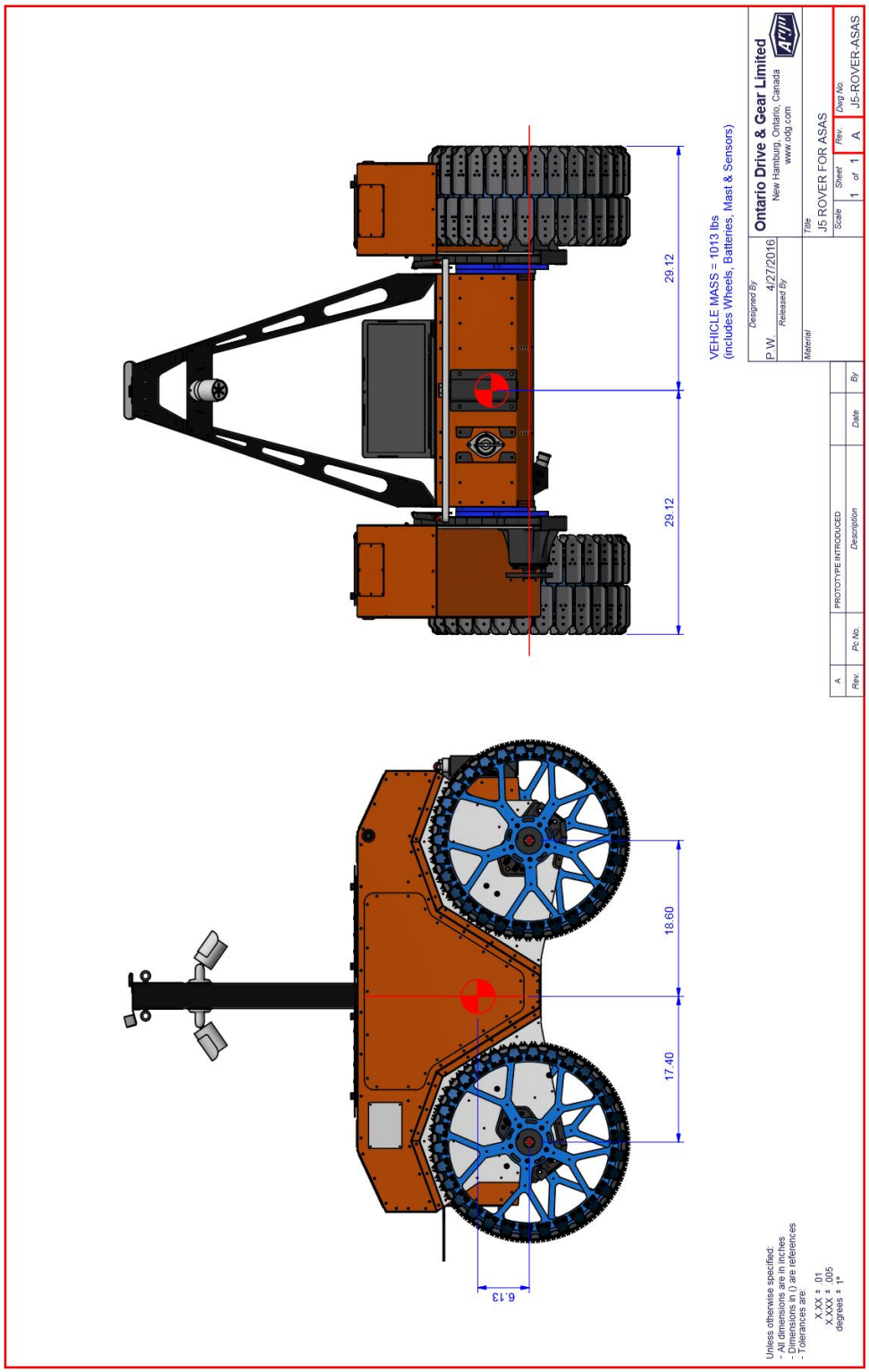
FRONT

TECHNICAL SPECIFICATIONS

SIZE AND WEIGHT	
EXTERNAL DIMENSIONS (L x W x H)	1.52 x 1.38 x 0.83 m (4.9 x 4.5 x 2.72 ft)
BASE WEIGHT (includes base battery pack)	280 kg (551 lbs)
GROSS VEHICLE WEIGHT	590 kg (1300 lbs)
GROUND CLEARANCE	254 mm (10 in)
SPEED AND PERFORMANCE	
MAX. PAYLOAD	272 kg (600 lbs)
MAX. INCLINE	35 - 45°
MAX. SPEED	18 km/h (11 mph)
SUSPENSION	Geometric Passive Articulation
TRACTION	24" Argo tire (24" Turf tire or 12" wide Quad Track System optional)
BATTERY AND POWER SYSTEM	
BATTERY CHEMISTRY	AGM sealed lead acid (Li-ion optional)
CAPACITY	105 Ah at 48 V, expandable to 110Ah with Li-ion option
CHARGE TIME	4 hrs
NOMINAL RUN TIME	Lead acid: 2.5 hrs Li-ion: 3 hrs
USER POWER	5 V, 12 V Fused (24 V, 48 V optional)
INTERFACING AND COMMUNICATION	
CONTROL MODES	Remote control, Computer controlled velocity commands (v, θ), Indoor/outdoor autonomy packages
FEEDBACK	Battery voltage, motor currents, wheel odometry, control system status, temperature, safety status
COMMUNICATION	Ethernet, USB, Remote Control, Wi-Fi
DRIVERS AND APIs	Packaged with ROS Indigo (includes RViz, Gazebo support), Matlab API available
INCLUDED HARDWARE	IMU, encoders, Onboard computer, E-Stop (hardware loop), E-Stop (software loop), removable mounting plates, bilge pumps, brakes
ENVIRONMENTAL	
OPERATING AMBIENT TEMPERATURE	-20 to 40 °C (-4 to 104 °F)
STORAGE TEMPERATURE	-40 to 50 °C (-40 to 122 °F)
IP RATING	IP65 - Vehicle is designed to float and should not be fully submerged
AMPHIBIOUS	Fully amphibious, 4 km/h (2.4 mph) maximum water speed*

Contact us today for pricing and a free 30 minute technical assessment: 1-800-301-3863

A.3 Rover Drawing



Appendix B - 2D Kinematic Analysis Data

This appendix contains the relevant formulation and results for Chapter 4, Section 4.1.

B.1 Rover Data & Analysis Setup

Rover: Argo J5 (4-wheel)

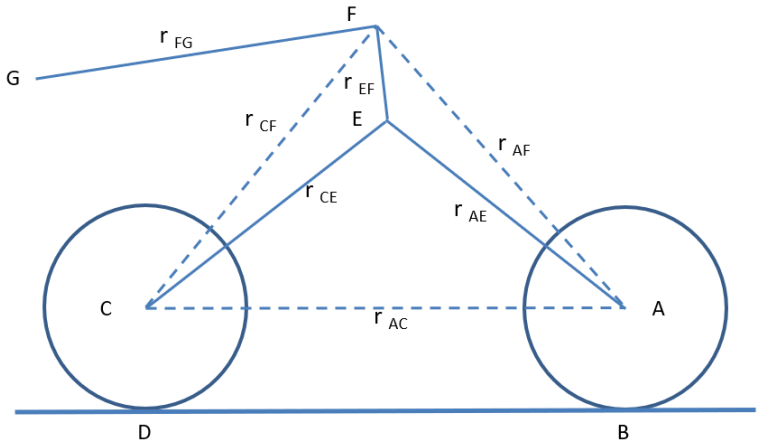


Figure B1: Right-side profile with labelled pivot points and rigid distances between pivot points.

Associated Dimensions:

$r_{AB} = r_{CD} = 0.254$ m

$r_{AC} = 0.90932$ m

$r_{AE} = 0.4653$ m

$r_{AF} = 0.632$ m

$r_{CE} = 0.465$ m

$r_{CF} = 0.5088$ m

$r_{EF} = 0.256$ m

$r_{FG} = 0.607$ m

Equation Set 1

Developed from the geometry seen in Figure B.1 and the associated dimensions.

Implemented in MATLAB function file: geomJ5_nosusp.m

$$(x_A - x_B)^2 + (y_A - y_B)^2 = r_{AB}^2$$

$$(x_C - x_D)^2 + (y_C - y_D)^2 = r_{CD}^2$$

$$(x_A - x_C)^2 + (y_A - y_C)^2 = r_{AC}^2$$

$$(x_A - x_E)^2 + (y_A - y_E)^2 = r_{AE}^2$$

$$(x_C - x_E)^2 + (y_C - y_E)^2 = r_{CE}^2$$

$$y_B - f(x_B) = 0$$

$$y_D - f(x_D) = 0$$

$$m_2(y_A - y_B) + (x_A - x_B) = 0$$

$$m_4(y_C - y_D) + (x_C - x_D) = 0$$

Equation Set 2

Independent equation set developed based on geometry as a means of validating the results and troubleshooting if need be. Note: c_x and c_y were considered known points. γ represents the slope angle and α is the angle of the walking beam between the link lengths r_{AE} and r_{CE}.

$$a_x = c_x + r_{AC}\cos\gamma$$

$$a_y = c_y + r_{AC}\sin\gamma$$

$$d_x = c_x + r_{CD} \sin \gamma$$

$$d_y = c_y - \frac{r_{CD}}{\cos \gamma}$$

$$b_x = d_x + r_{AC} \cos \gamma$$

$$b_y = d_y + r_{AC} \sin \gamma$$

$$e_x = c_x + r_{CE} \sin \left(\frac{\alpha}{2} - \gamma \right)$$

$$e_y = c_y + r_{CE} \cos \left(\frac{\alpha}{2} - \gamma \right)$$

B.2 MATLAB Script: Planar Position Kinematics for J5 Rover

(J5_KinematicModel_1.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Positional Kinematics for J5 Rover
%%
%% Based on Rocky 7 paper on velocity kinematics control. Specifcally,
%% using equations and coordinate assignment derived. Purpose of
recreating
%% their model is to serve as a baseline for kinematic model for J5
%%
%% Written by: E. Austen
%% Created on: April 20, 2018
%% Last Modified:
%%      May 3, 2018 - changed eqtn set alternative solution
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

clear all;
clc;

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



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Assumptions
%%
%% 1. All rover components (links, wheels, etc) are assumed to be rigid
%%    bodies. Thus making distances between joints (etc) constant and
%%    relative velocities btwn them zero.
%% 2. Rover travels at constant nominal speed
%% 3.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Terrain Function & Parameters
tmax=10; % Max run-time [s]
del_t=1; % Time step [s]
t=0;
% t=0:del_t:tmax;
x_terr=-0.2:1.6;
% y_ter=0.01875*sin(x);  % Overall terrain function
y_terr=0.12*sin(x_terr);
%y_terr=0.5*x_terr;       % Flat land test function
v_r=0.1;    % Rover nominal speed [m/s]
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Intitial parameters (initial guess)
% Incl. suspension
% x0 = zeros(13,1);
% x0(1) = 0.909;      x0(8) = 0.254;      % Initial coordinates of
joints & contact points
% x0(2) = 0.909;      x0(9) = 0;          % these will be used as
"guess" for Newton Rhapson
% x0(3) = 0.4545;     x0(10) = 0.352;
% x0(4) = 0.358;      x0(11) = 0.8065;     % Note that the back wheel
is used as x=0
% x0(5) = -0.246;     x0(12) = 0.747;
% x0(6) = 0;          x0(13) = 0;
% x0(7) = 0.254;

%Without Suspension
x0=zeros(9,1);
x0(1) = 0.909;      x0(6) = 0.254;      % Initial coordinates of joints
& contact points
x0(2) = 0.909;      x0(7) = 0;          % these will be used as "guess"
for Newton Rhapson
x0(3) = 0.4545;     x0(8) = 0.352;
x0(4) = 0;          x0(9) = 0;     % Note that the back wheel is used
as x=0
x0(5) = 0.254;


% Set cx as independent variable to make 13 unknowns for 13 eqtns.
x_5i=0;     % Initial position of x_7i (x-coordinate of back wheel)

% t2=v_r/x0(2);
% f2=y_ter(t2);   % Elevation of terrain at each contact point
% f4=y_ter(v_r/x0(4));
% f6=y_ter(v_r/x0(6));

% m2=diff(y_ter(v_r/x0(2)));   % Slope value of tangent at wheel-gnd
contact pt
% m4=diff(y_ter(v_r/x0(4)));
% m6=diff(y_ter(v_r/x0(6)));



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Outer time loop to move over terrain

while t<= tmax

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Perform Newton-Raphson (or nonlinear eqtn sovler)

% [xnr, iter_nr]=feval(NR_nlm,x0,y_terr,x_8i,'geom6W','Jacob15');
```

```
[xnr, iter_nr]=NR_nlm_J5(x0,x_5i,'geomJ5_nosusp','Jacob9');

% Fsolve built in function
% xnr=fsolve(geom6W,x0)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%
% Reassign x variables for ease of plotting
Ax(t+1) = xnr(1);
Bx(t+1) = xnr(2);
Ex(t+1) = xnr(3);
Dx(t+1) = xnr(4);
Cx(t+1) = x_5i;
% Reassign x variables (9-16) back to y variables for ease of plotting
Ay(t+1) = xnr(6);
By(t+1) = xnr(7);
Ey(t+1) = xnr(8);
Fy(t+1) = xnr(9);
Cy(t+1) = xnr(5);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Plot rover suspension system & animate to follow inputs
%%

% plot(x1,y1,x2,y2,x7,y7,x8,y8)
% legend('a','b','g','h')

figure
%Q=figure;
hold on

%title('J5 Rover Traversing Terrain')
xlabel('Horizontal Component, X [m]')
ylabel('Vertical Component, Y [m]')

%plot(y_terr)
% y_sine=@(x) 0.12*sin(pi*x);
% fplot(y_sine, [-0.25 4])
y_terr2= @(x) 0.5*x;
fplot(y_terr2, [-0.3 1.5]);
plot([xnr(1) xnr(3)], [xnr(6) xnr(8)])
% plot([xnr(3) xnr(4)], [xnr(10) xnr(11)])
% plot([xnr(4) xnr(5)], [xnr(11) xnr(12)])
plot([xnr(3) x_5i], [xnr(8) xnr(5)])

% legend('Terrain', 'Link AE', 'Link EF', 'Link FG', 'Link
CE','Location', 'east')
legend('Terrain', 'Link AE', 'Link CE', 'Location', 'southeast')

th = 0:pi/50:2*pi;
r = 0.254;
```

```
x_c1 = r * cos(th) + xnr(1);
y_c1 = r * sin(th) + xnr(6);
x_c2 = r * cos(th) + x_5i;
y_c2 = r * sin(th) + xnr(5);

plot(x_c1, y_c1, x_c2, y_c2)
axis equal
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off

%saveas(Q,sprintf('FIG%d.jpg',t));
saveas(gcf,'FIG%d.jpg')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% Update time step
t=t + del_t;

% Update initial guess
x0=xnr;

% Update independent & thereby move further along the terrain
x_5i = x_5i + (v_r*del_t);

end

figure
hold on
%title('Elevation of pivot points with respect to terrain traversed')
xlabel('Horizontal Component, X [m]')
ylabel('Vertical Component, Y [m]')

plot(Ax,Ay)
plot(Bx,By)
plot(Ex,Ey)
plot(Cx,Cy)

legend('A', 'B', 'E', 'C','Location', 'southeast')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off
```

B.3 MATLAB Function File: Planar Kinematic Equation Set for J5 Rover (geomJ5_nosusp.m)

```
function [F] = geomJ5_nosusp( x , x_5i )
%geomJ5 Eqtn set for based on the geometry of the rover configuration
and
%the coordinates of each node.

%% Rover fixed parameters
r_ac = 0.9093;       % Distance between wheels a & c [m]
r_ae = 0.4653;       % Link length from wheel e to chassis pivot joint
[m]
r_ce = 0.4653;       % Link length from wheel e to chassis pivot joint
[m]


r_ab = 0.254;     % Radius of wheel a. Also distance between wheel
centre and
                  % ground contact point. Units: [m]
r_cd = 0.254;     % Radius of wheel c. Units: [m]


%% Functions based on terrain
% f2=y_terr(x(2));
% f4=y_terr(x(4));

% f2=0; % Flat land case
% f4=0;
% f2=0.5*x(2); % Incline
% f4=0.5*x(4);
f2=0.12*sin(pi*x(2));
f4=0.12*sin(pi*x(4));

%% Slope values
% m2=diff(f2);
% m4=diff(f4);

% m2=0; % Flat land case
% m4=0;
% m2=0.5; % Slope case
% m4=0.5;
m2=0.12*pi*cos(pi*x(2));
m4=0.12*pi*cos(pi*x(4));

%% Equation set to be solved
F(1) = ((x(1)-x(2))^2) + ((x(6)-x(7))^2) - r_ab^2;    % Eqtns for
wheel-gnd
F(2) = ((x_5i-x(4))^2) + ((x(5)-x(9))^2) - r_cd^2;    % contact pt

F(3) = ((x(1)-x_5i)^2) + ((x(6)-x(5))^2) - r_ac^2;    % Eqtns for links
F(4) = ((x(1)-x(3))^2) + ((x(6)-x(8))^2) - r_ae^2;
F(5) = ((x_5i-x(3))^2) + ((x(5)-x(8))^2) - r_ce^2;
```

```
F(6) = x(7) - f2;     % Wheel-terrain
F(7) = x(9) - f4;

F(8) = m2.*(x(6)-x(7)) + (x(1)-x(2));
F(9) = m4.*(x(5)-x(9)) + (x_5i-x(4));

F=F.';
end
```

B.4 MATLAB Function File: Nonlinear multi-variate Newton-Raphson solver (NR_nlm_J5.m)

```
function [x, iter] = NR_nlm_J5(x0,x_5i,F,J)
%NR_nlm Newton-Raphson method for nonlinear multivariate system of
eqtns.
%   Detailed explanation goes here

N = 100;  % Number of iterations
Eps = 1e-10;     % Tolerance
Div = 1000;      % Divergence value
xx = x0;         % Load array of inital guess

% f2=0;   % Values for static flat terrain for debugging purposes
% f4=0;
% f6=0;

while N>0
    Jc = feval('Jacob9',xx, x_5i);

    if abs(det(Jc))<Eps
        error('Jacobian is singular. Change x0');
        abort;
    end

    Xn = xx - inv(Jc)*feval('geomJ5_nosusp',xx,x_5i);

    if abs(feval('geomJ5_nosusp',Xn,x_5i))<Eps
        x = Xn;
        iter  = 100 - N;
        return;
    end

    if abs(feval('geomJ5_nosusp',xx,x_5i))>Div
        iter = 100 - N;
        disp(['Iterations = ', num2str(iter)]);
        error('Solution fails to converge');
        abort;
    end

    N = N - 1;
    xx = Xn;
end
error('No convergence after 100 iterations');
abort;

end
```

B.5 MATLAB Script: Jacobian function file for use in nonlinear, multivariate, Newton-Raphson function (Jacob9.m)

```
function [J] = Jacob9(x, x_5i)
%Jacob9 Evaluates the Jacobian of a 9x9 matrix for a nonlinear system
of
%9 equations
%   Detailed explanation goes here


%% NOTE: x_5i is the x-coordinate of point 5(C), however to make the
square
%% Jacobian, it is defined as an independent variable (as per paper).
Thus,
%% x10 becomes x(5).

% Extracting f2, f6 from terrain function
% f2=y_terr(x(2));
% f6=y_terr(x(4));

% f2=0;
% f4=0;

% Slope values for test
% m2=0.5;   % Flat land case. 0.5 for slope
% m4=0.5;
m2=0.12*pi*cos(pi*x(2));
m4=0.12*pi*cos(pi*x(4));
%x_8i=0;

J = zeros(9,9);   % Initialise matrix of zeroes

J(1,1) = 2*(x(1)-x(2));     J(1,2) = 2*(x(2)-x(1));
J(1,6) = 2*(x(6)-x(7));     J(1,7) = 2*(x(7)-x(6));

J(2,4) = 2*(x(4)-x_5i);     J(2,5) = 2*(x(5)-x(9));
J(2,9) = 2*(x(9)-x(5));

J(3,1) = 2*(x(1)-x_5i);     %J(3,5) = 2*(x(5)-x(3));
J(3,5) = 2*(x(5)-x(6));    J(3,6) = 2*(x(6)-x(5));

J(4,1) = 2*(x(1)-x(3));     J(4,3) = 2*(x(3)-x(1));
J(4,6) = 2*(x(6)-x(8));    J(4,8) = 2*(x(8)-x(6));

J(5,3) = 2*(x(3)-x_5i);     %J(5,8) = 2*(x_8i-x(5));
J(5,5) = 2*(x(5)-x(8));    J(5,8) = 2*(x(8)-x(5));

J(6,2) = 0.5;
J(6,7) = 1;
% J(6,2)=-0.12*pi*cos(pi*x(2));

J(7,4) = 0.5;
J(7,9) = 1;
```

```
% J(7,4)=-0.12*pi*cos(pi*x(4));

J(8,1) = 1;      J(8,2) = -1;
J(8,6) = m2;     J(8,7) = -m2;

J(9,4) = -1;         %J(13,4) = -1;
J(9,5) = m4;       J(9,9) = -m4;


end
```

B.6 Further Results for J5 Rover

I. Flat Terrain Case

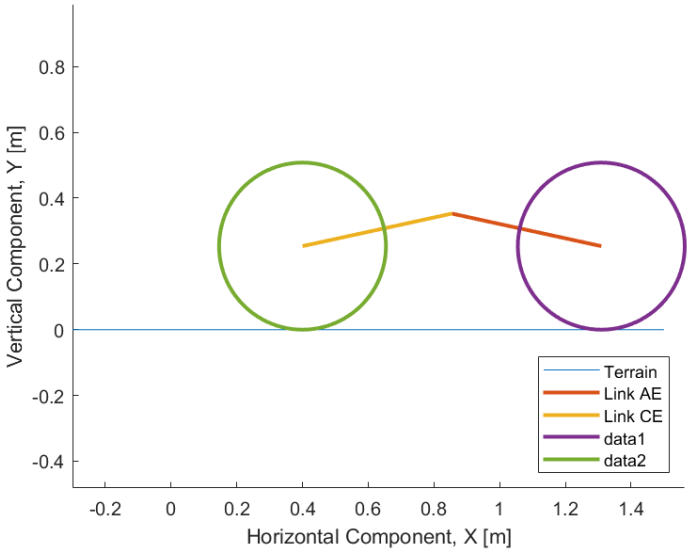


Figure B.2: Planar inclined pose for flat terrain, t=4s.

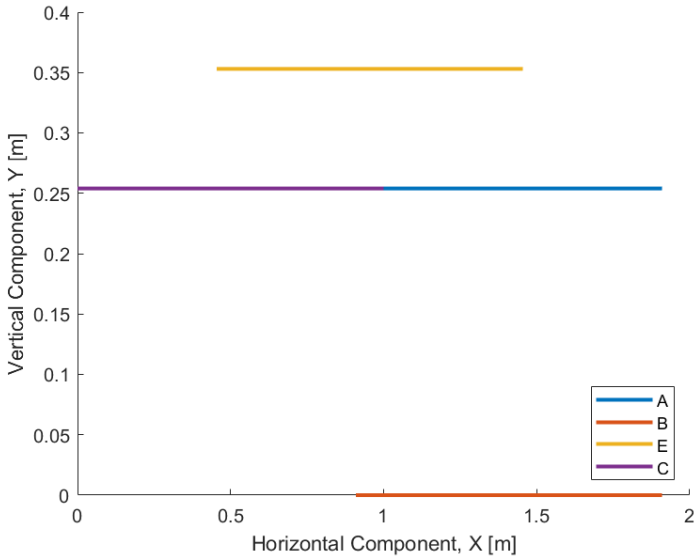


Figure B.3: Pivot point traces for flat terrain.

II. Inclined Terrain Cases

Terrain Function: $y = 0.1x$

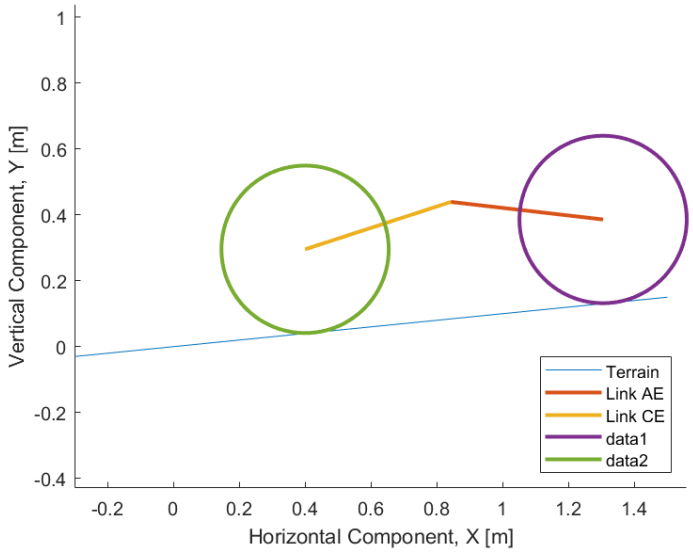


Figure B.4: Planar inclined pose for a slope of 5.71°, t=4s.

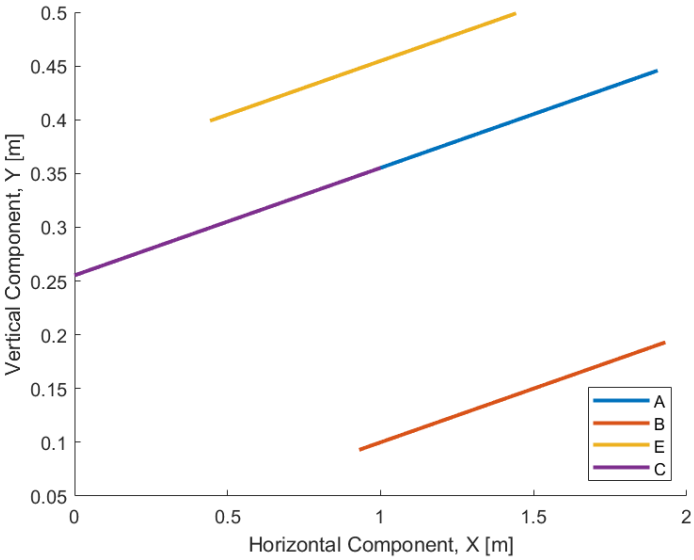


Figure B.5: Pivot point traces for inclined terrain with slope of 5.71°.

Terrain Function: $y = 0.5x$

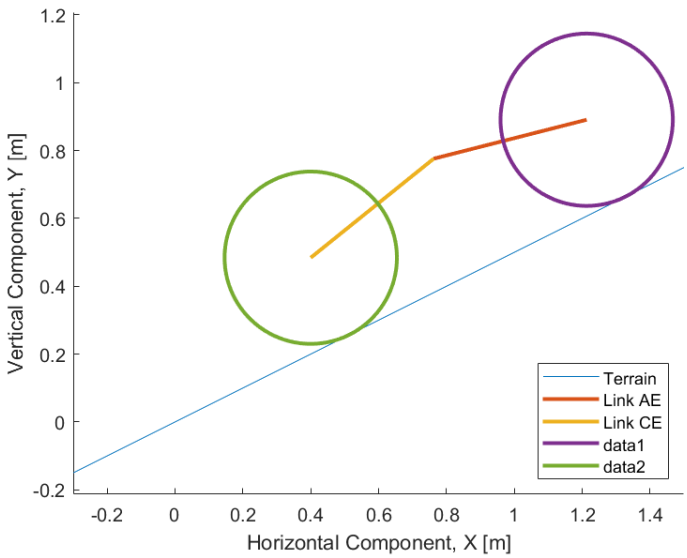


Figure B.6: Planar inclined pose for a slope of 26.57°, t=4s.

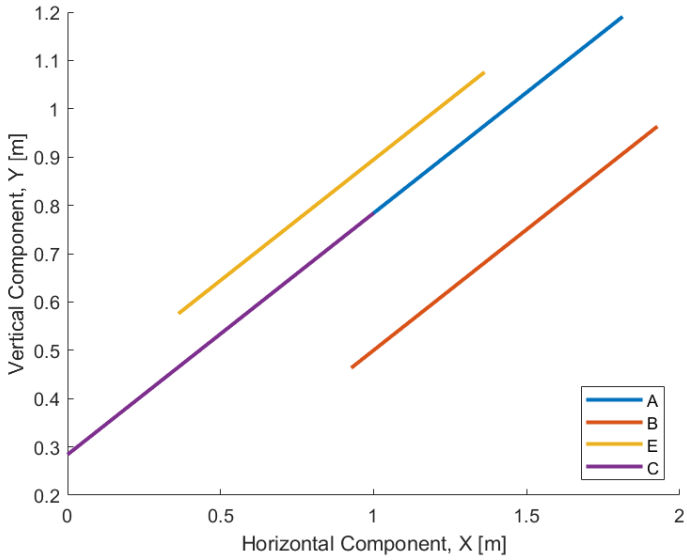


Figure B.7: Pivot point traces for inclined terrain with slope of 26.57°.

III. Sinusoidal Terrain Cases

Terrain Function: $y = 0.03 \sin\left(\frac{\pi}{2}x\right)$

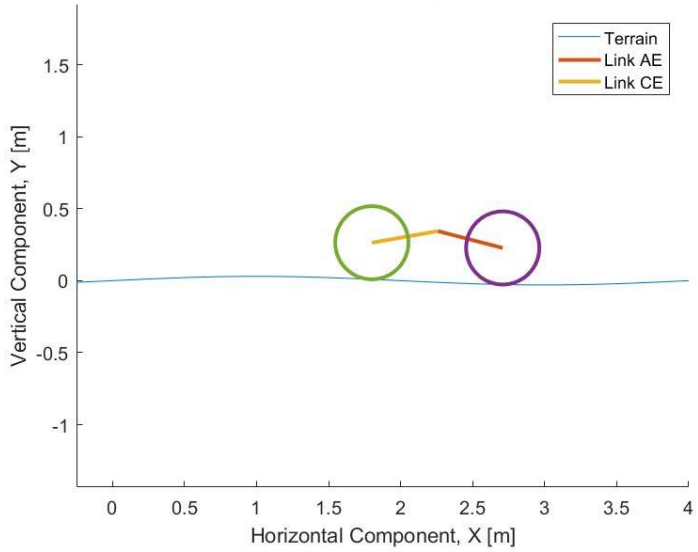


Figure B.8: Planar pose for a sine terrain of $y = 0.03 \sin\left(\frac{\pi}{2}x\right)$, t=17s.

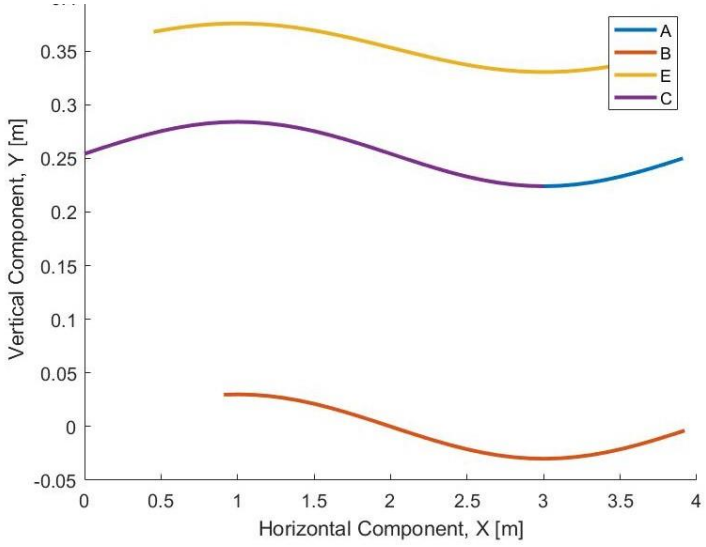


Figure B.9: Pivot point traces for a sine terrain of $y = 0.03 \sin\left(\frac{\pi}{2}x\right)$.

Terrain Function: $y = 0.12\sin(\pi x)$

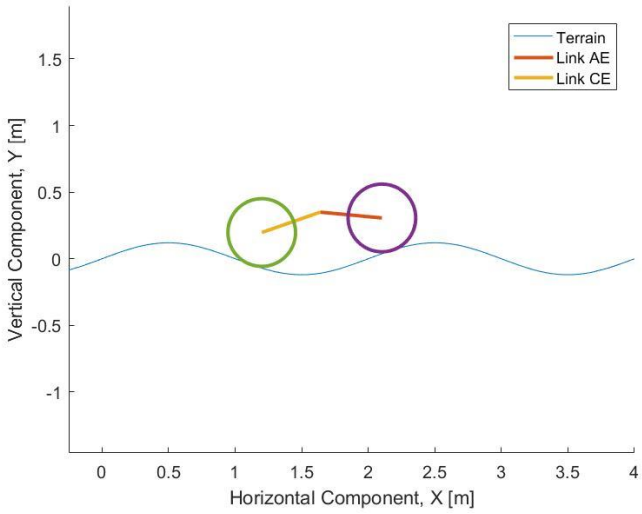


Figure B.10: Planar pose for a sine terrain of $y = 0.12\sin(\pi x)$, t=12s.

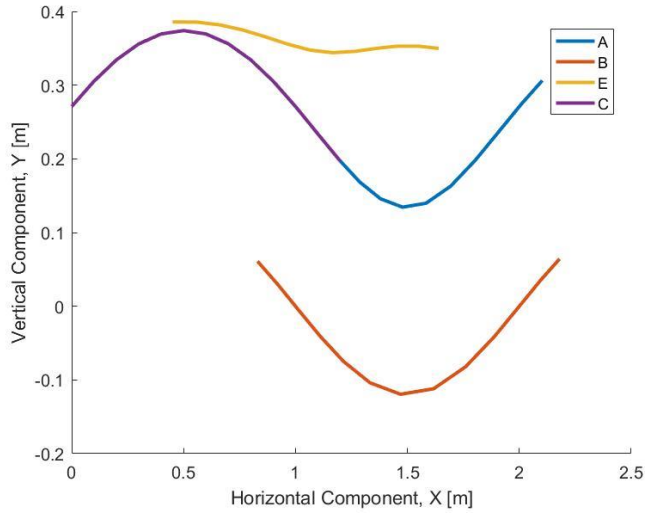


Figure B.11: Pivot point traces for a sine terrain of $y = 0.12\sin(\pi x)$.

B.7 Rover Data & Analysis Setup (from original paper)

Rover: Rocky 7 (6-wheel)

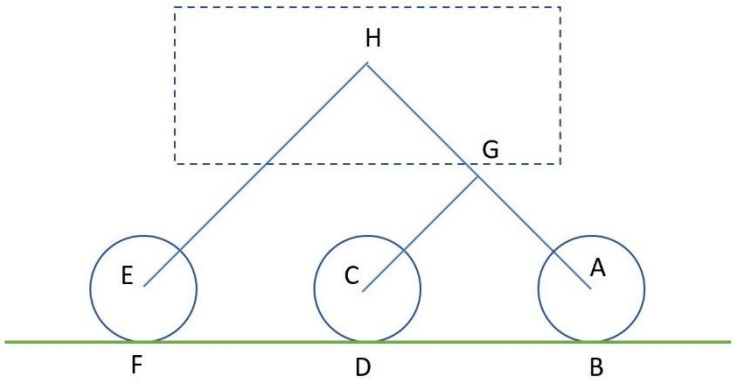


Figure B1: Right-side profile with labelled pivot points.

Associated Dimensions:

$r_{AB} = r_{CD} = r_{EF} = 0.065$ m

$r_{AC} = 0.240$ m

$r_{AG} = r_{CG} = 0.170$ m

$r_{EH} = 0.339$ m

$r_{HG} = 0.170$ m

$r_{EG} = 0.379$ m

Equation Set 1

Developed from the geometry seen in Figure B.1 and the associated dimensions.

Implemented in MATLAB function file: geom6W.m

$$(x_A - x_B)^2 + (y_A - y_B)^2 = r_{AB}^2$$

$$(x_C - x_D)^2 + (y_C - y_D)^2 = r_{CD}^2$$

$$(x_E - x_F)^2 + (y_E - y_F)^2 = r_{EF}^2$$

$$(x_A - x_C)^2 + (y_A - y_C)^2 = r_{AC}^2$$

$$(x_A - x_G)^2 + (y_A - y_G)^2 = r_{AG}^2$$

$$(x_C - x_G)^2 + (y_C - y_G)^2 = r_{CG}^2$$

$$(x_H - x_G)^2 + (y_H - y_G)^2 = r_{HG}^2$$

$$(x_E - x_G)^2 + (y_E - y_G)^2 = r_{EG}^2$$

$$(x_E - x_H)^2 + (y_E - y_H)^2 = r_{EH}^2$$

$$y_B - f(x_B) = 0$$

$$y_D - f(x_D) = 0$$

$$y_F - f(x_F) = 0$$

$$m_2(y_A - y_B) + (x_A - x_B) = 0$$

$$m_4(y_C - y_D) + (x_C - x_D) = 0$$

$$m_6(y_E - y_F) + (x_E - x_F) = 0$$

Equation Set 2

Independent equation set developed based on geometry as a means of validating the results and troubleshooting if need be. Note: e_x was considered to be a known point. γ represents the slope angle and L is the length or distance of wheel E from the start of a slope.

$$a_x = r_{EC}\cos\theta + (L - x)\sin 2\gamma + r_{AC}\cos\gamma$$

$$a_y = r_{EC}\sin\theta + (L - x)\sin 2\gamma + \frac{r_{AB}}{\cos\gamma} + r_{AC}\sin\gamma$$

$$b_x = r_{EC}\cos\theta + (L - x)\sin 2\gamma + r_{AB}\sin\gamma + r_{AC}\cos\gamma$$

$$b_y = r_{EC}\sin\theta + (L - x)\sin 2\gamma + r_{AC}\sin\gamma$$

$$c_x = r_{EC}\cos\theta + (L - x)\sin 2\gamma$$

$$c_y = r_{EC}\sin\theta + (L - x)\sin 2\gamma + \frac{r_{AB}}{\cos\gamma}$$

$$d_x = r_{EC}\cos\theta + (L - x)\sin 2\gamma + r_{AB}\sin\gamma$$

$$d_y = r_{EC}\sin\theta + (L - x)\sin 2\gamma + r_{AB}\tan\gamma\sin\gamma$$

$$e_y = \frac{r_{AB}}{\cos\gamma}$$

$$f_x = r_{AB}\sin\theta$$

$$f_y = r_{AB}\tan\theta\sin\theta$$

$$g_x = r_{EC}\cos\theta + (L - x)\sin 2\gamma + r_{CG}\sin(45 - \gamma)$$

$$g_y = r_{EC}\sin\theta + (L - x)\sin 2\gamma + \frac{r_{AB}}{\cos\gamma} + r_{CG}\cos(45 - \gamma)$$

$$h_x = r_{EC}\cos\theta + (L - x)\sin 2\gamma + r_{CG}\sin(45 - \gamma) - r_{HG}\cos\gamma$$

$$h_y = r_{EC}\sin\theta + (L - x)\sin 2\gamma + \frac{r_{AB}}{\cos\gamma} + r_{CG}\cos(45 - \gamma) + r_{HG}\cos\gamma$$

B.8 MATLAB Script: Planar Kinematic Equation Set for Rocky 7 Rover (geom6W.m)

```
function  F = geom6W( x , x_8i)
%geom6W Eqtn set for based on the geometry of the rover configuration
and
%the coordinates of each node.
%   Detailed explanation goes here

%% Rover fixed parameters
r_ac = 0.24;        % Distance between wheels a & c [m]
r_ag = 0.12*sqrt(2);        % Length of bogie link [m]
r_cg = 0.12*sqrt(2);
r_eh = 0.24*sqrt(2);        % Link length from wheel e to chassis pivot
joint
r_hg = r_eh-r_ag;        % Link length from h to g (rocker)
r_eg = sqrt((r_eh^2)+(r_hg^2));    % Distance between wheel e and bogie
pivot joint

r_ab = 0.065;      % Radius of wheel a. Also distance between wheel
centre and
                   % ground contact point. Units: [m]
r_cd = 0.065;      % Radius of wheel c. Units: [m]
r_ef = 0.065;      % Radius of wheel e. Units: [m]

%% Functions based on terrain
% f2=y_terr(x(2));
% f4=y_terr(x(4));
% f7=y_terr(x(7));
f2=0.1*sin(1.9*pi*x(2));
f4=0.1*sin(1.9*pi*x(4));
f7=0.1*sin(1.9*pi*x(7));
% f2=0;
% f4=0;
% f7=0;

%% Slope values

% m2=0; % Flat land case
% m4=0;
% m7=0;
m2=0.19*pi*cos(1.9*pi*x(2));
m4=0.19*pi*cos(1.9*pi*x(4));
m7=0.19*pi*cos(1.9*pi*x(7));
%x_8i = 0;

%% Equation set to be solved
F(1) = ((x(1)-x(3))^2) + ((x(9)-x(11))^2) - r_ac^2;    % Eqtns for
links
F(2) = ((x(1)-x(5))^2) + ((x(9)-x(13))^2) - r_ag^2;
F(3) = ((x(3)-x(5))^2) + ((x(11)-x(13))^2) - r_cg^2;
F(4) = ((x(6)-x(5))^2) + ((x(14)-x(13))^2) - r_hg^2;
F(5) = ((x_8i-x(5))^2) + ((x(8)-x(13))^2) - r_eg^2;
```

```
F(6) = ((x_8i-x(6))^2) + ((x(8)-x(14))^2) - r_eh^2;

F(7) = ((x(1)-x(2))^2) + ((x(9)-x(10))^2) - r_ab^2;    % Eqtns for
wheel-gnd
F(8) = ((x(3)-x(4))^2) + ((x(11)-x(12))^2) - r_cd^2;    % contact pt
F(9) = ((x_8i-x(7))^2) + ((x(8)-x(15))^2) - r_ef^2;

F(10) = x(10) - f2;    % Wheel-terrain
F(11) = x(12) - f4;
F(12) = x(15) - f7;

F(13) = m2.*(x(9)-x(10)) + (x(1)-x(2));
F(14) = m4.*(x(11)-x(12)) + (x(3)-x(4));
F(15) = m7.*(x(8)-x(15)) + (x_8i-x(7));

F=F.';
end
% End of function
```

B.9 MATLAB Script: Jacobian function file for use in nonlinear, multivariate, Newton-Raphson function for the Rocky 7 rover (Jacob15.m)

```
function [J] = Jacob15(x, x_8i)
%Jacob15 Evaluates the Jacobian of a 15x15 matrix for a nonlinear
system of
%15 equations
%   Rocky 7 Rover (6 wheels)

%% NOTE: x_8i is the x-coordinate of point 8(E), however to make the
square
%% Jacobian, it is defined as an independent variable (as per paper).
Thus,
%% x16 becomes x(8).

% Extracting f2, f4, f7 from terrain function
% f2=y_terr(x(2));
% f4=y_terr(x(4));
% f7=y_terr(x(7));
% f2=0;
% f4=0;
% f7=0;

% Slope values for test
% m2=0;     % Flat land case
% m4=0;
% m7=0;
m2=0.19*pi*cos(1.9*pi*x(2));
m4=0.19*pi*cos(1.9*pi*x(4));
m7=0.19*pi*cos(1.9*pi*x(7));
%x_8i=0;

J = zeros(15,15);   % Initialise matrix of zeroes

J(1,1) = 2*(x(1)-x(3));     J(1,3) = 2*(x(3)-x(1));
J(1,9) = 2*(x(9)-x(11));     J(1,11) = 2*(x(11)-x(9));

J(2,1) = 2*(x(1)-x(5));     J(2,5) = 2*(x(5)-x(1));
J(2,9) = 2*(x(9)-x(13));     J(2,13) = 2*(x(13)-x(9));

J(3,3) = 2*(x(3)-x(5));     J(3,5) = 2*(x(5)-x(3));
J(3,11) = 2*(x(11)-x(13));    J(3,13) = 2*(x(13)-x(11));

J(4,5) = 2*(x(5)-x(6));     J(4,6) = 2*(x(6)-x(5));
J(4,13) = 2*(x(13)-x(14));    J(4,14) = 2*(x(14)-x(13));

J(5,5) = 2*(x(5)-x_8i);     %J(5,8) = 2*(x_8i-x(5));
J(5,13) = 2*(x(13)-x(8));    J(5,8) = 2*(x(8)-x(13));

J(6,6) = 2*(x(6)-x_8i);     %J(6,8) = 2*(x_8i-x(6));
J(6,14) = 2*(x(14)-x(8));    J(6,8) = 2*(x(8)-x(14));

J(7,1) = 2*(x(1)-x(2));     J(7,2) = 2*(x(2)-x(1));
```

```
J(7,9) = 2*(x(9)-x(10));      J(7,10) = 2*(x(10)-x(9));

J(8,3) = 2*(x(3)-x(4));      J(8,4) = 2*(x(4)-x(3));
J(8,11) = 2*(x(11)-x(12));    J(8,12) = 2*(x(12)-x(11));

J(9,7) = 2*(x(7)-x_8i);      %J(9,8) = 2*(x_8i-x(7));
J(9,15) = 2*(x(15)-x(8));     J(9,8) = 2*(x(8)-x(15));

% J(10,2) = 0;
J(10,10) = 1;
J(10,2)=0.19*pi*cos(1.9*pi*x(2));

% J(11,4) = 0;
J(11,12) = 1;
J(11,4)=0.19*pi*cos(1.9*pi*x(4));

% J(12,7) = 0;
J(12,15) = 1;
J(12,7)=0.19*pi*cos(1.9*pi*x(7));

J(13,1) = 1;      J(13,2) = -1;
J(13,9) = m2;     J(13,10) = -m2;

J(14,3) = 1;         J(14,4) = -1;
J(14,11) = m4;       J(14,12) = -m4;

J(15,7) = -1;         %J(15,8) = 1;
J(15,15) = -m7;       J(15,8) = m7;


end
% End of function
```

B.10 Further Results for Rocky 7 Rover

Results were generated for the Rocky 7 Rover to confirm correct application and coding of the process used in the original paper by Parakh et al [50].

I. Flat Case

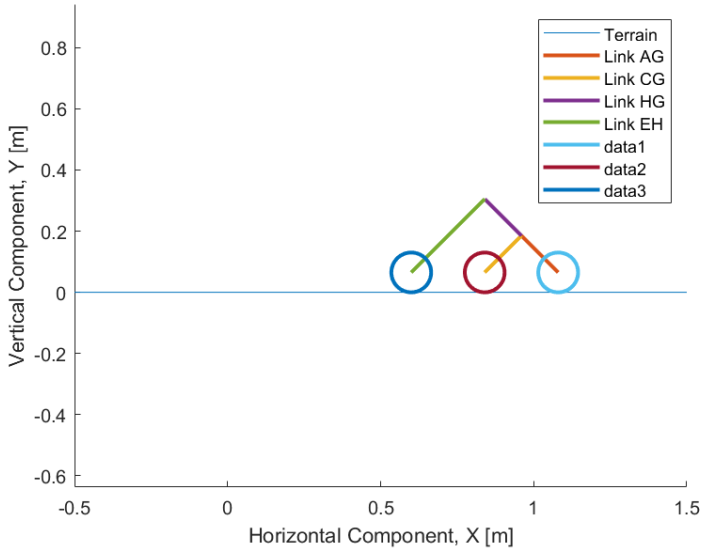


Figure B.12: Planar inclined pose for flat terrain, t=6s.

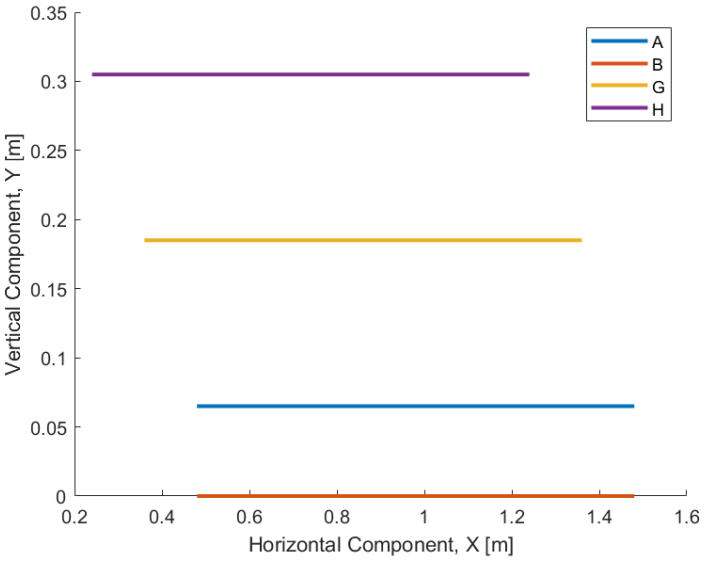


Figure B.13: Pivot point traces for flat terrain.

II. Upslope Case

Terrain Function: $y = 0.5x$

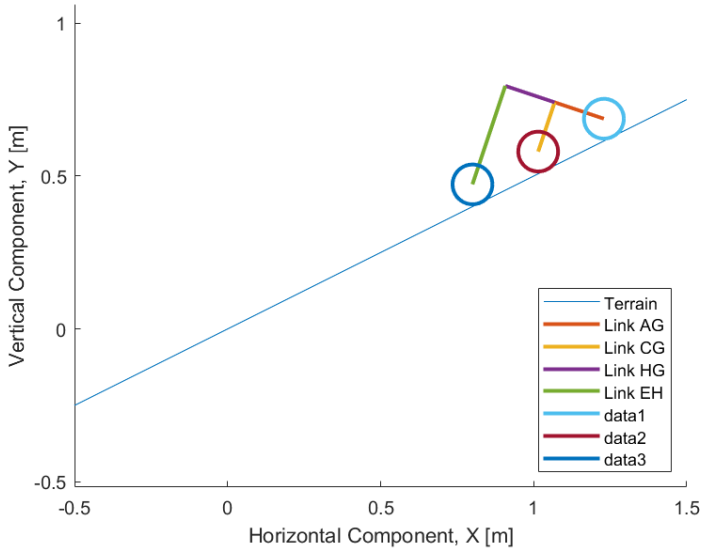


Figure B.16: Planar inclined pose for a slope of 26.57°, t=8s.

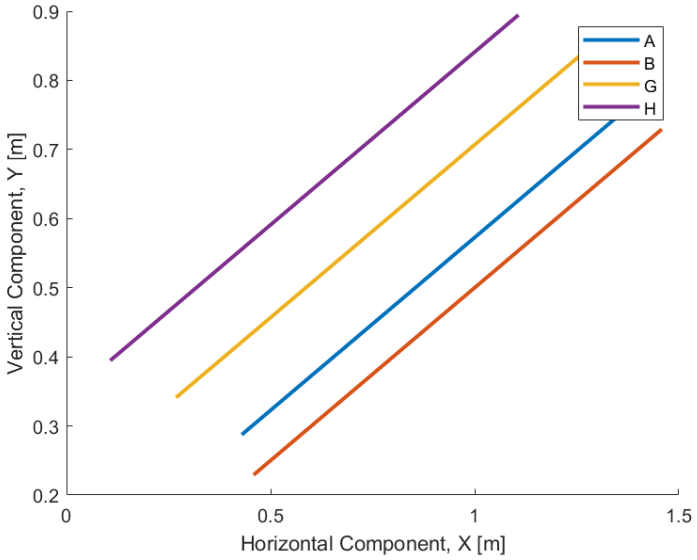


Figure B.17: Pivot point traces for inclined terrain with slope of 26.57°.

III. Sinusoidal Terrain

Terrain Function: $y = 0.1\sin(2\pi x)$

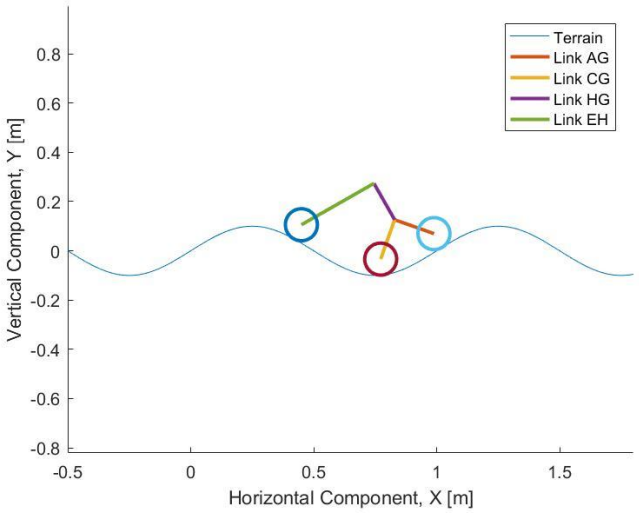


Figure B.18: Planar pose for a sine terrain of $y = 0.1\sin(2\pi x)$, t=4.5s.

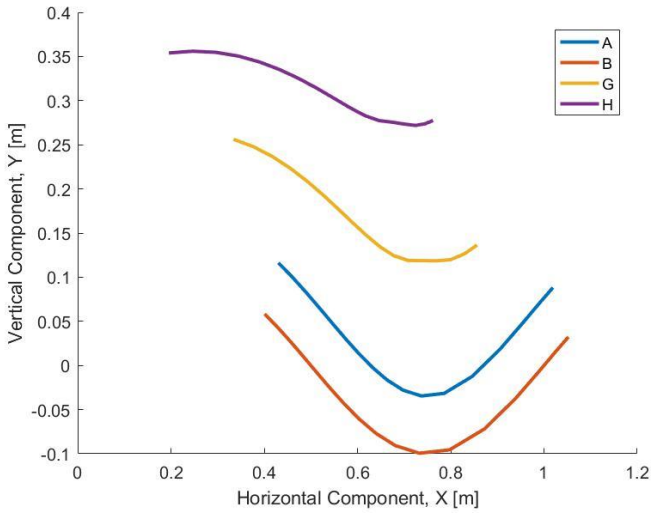


Figure B.19: Pivot point traces for a sine terrain of $y = 0.1\sin(2\pi x)$.

IV. Sinusoidal Terrain

Terrain Function: $y = 0.12\sin(\pi x)$

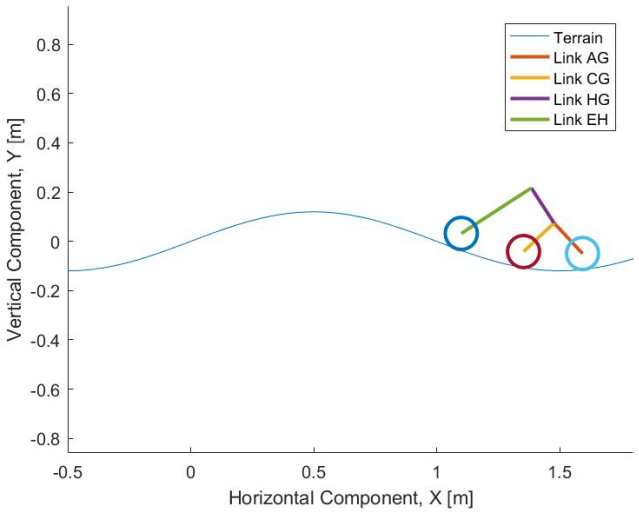


Figure B.20: Planar pose for a sine terrain of $y = 0.12\sin(\pi x)$, t=11s.

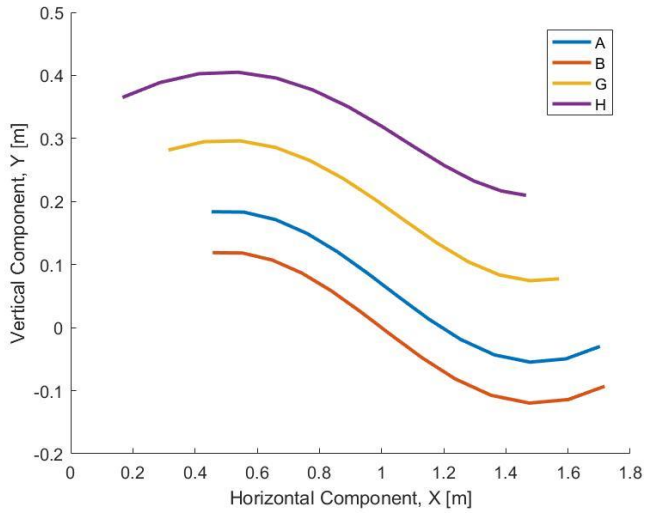


Figure B.21: Pivot point traces for a sine terrain of $y = 0.12\sin(\pi x)$.

Appendix C - 3D Kinematic Analysis Data

Appendix C contains material pertinent to the development of the 3D position and velocity kinematic models, including additional results.

C.1 D-H Tables

The following are the full set of D-H parameters describing all four kinematic chains of the rover.

Table C.1: D-H Parameters for the Kinematic Chain – World Origin Frame to Right Rear Wheel.

n	θ [deg]	α [deg]	a [m]	d [m]
0	+ 90	+ 90	0	0
1	- 90	- 90	0	X_{trans}
2	- 90	+ 90	0	Y_{trans}
3	0	0	0	$Z_{trans} + h_{CoG}$
4	φ_{yaw}	- 90	0	0
5	$\Phi_{pitch} + 90$	+ 90	0	0
6	φ_{roll}	+ 90	0	0
8	$\theta_{wbr} - \beta$	0	a_{wb}	d_{cmwb}

Table C.2: D-H Parameters for the Kinematic Chain – World Origin Frame to Right Front Wheel.

n	θ [deg]	α [deg]	a [m]	d [m]
0	+ 90	+ 90	0	0
1	- 90	- 90	0	X_{trans}
2	- 90	+ 90	0	Y_{trans}
3	0	0	0	$Z_{trans} + h_{CoG}$
4	φ_{yaw}	- 90	0	0
5	$\Phi_{pitch} + 90$	+ 90	0	0
6	φ_{roll}	+ 90	0	0
10	$\theta_{wbr} + \beta$	0	a_{wb}	d_{cmwb}

Table C.3: D-H Parameters for the Kinematic Chain – World Origin Frame to Left Rear Wheel.

n	θ [deg]	α [deg]	a [m]	d [m]
0	+ 90	+ 90	0	0
1	- 90	- 90	0	X_{trans}
2	- 90	+ 90	0	Y_{trans}
3	0	0	0	$Z_{trans} + h_{CoG}$
4	φ_{yaw}	- 90	0	0
5	$\Phi_{pitch} + 90$	+ 90	0	0
7	φ_{roll}	- 90	0	0
9	$\theta_{wbl} + \beta$	0	a_{wb}	d_{cmwb}

Table C.4: D-H Parameters for the Kinematic Chain – World Origin Frame to Left Front Wheel.

n	θ [deg]	α [deg]	a [m]	d [m]
0	+ 90	+ 90	0	0
1	- 90	- 90	0	X_{trans}
2	- 90	+ 90	0	Y_{trans}
3	0	0	0	$Z_{trans} + h_{CoG}$
4	φ_{yaw}	- 90	0	0
5	$\Phi_{pitch} + 90$	+ 90	0	0
7	φ_{roll}	- 90	0	0
11	$\theta_{wbl} - \beta$	0	a_{wb}	d_{cmwb}

C.2 MATLAB Script File: 3D Position Kinematic Model

(J5_3DPositionKinematics.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% 3D Position Kinematics
%%
%% Written by: E. Austen
%%
%% This script solves the inverse position kinematics in 3D for the J5
%% rover, for a given set of wheel-ground contact positions. These
contact
%% positions can be either obtained from a DEM of the terrain or
extraction
%% from a specified terrain function. The kinematic equations used were
%% derived using the original Denavit-Hartenberg convention.
%%
%% Created on: May 24, 2019
%% Last Modified:
%% July 15, 2019 - added chassis angle orientation plot
%% June 2, 2019 - fixed orientation eqtns
%% May 29, 2019 - eqtn set
%% May 28, 2019 - terrain variables & loop conditions
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

clear all;
clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Initialisation of parameters & importing the terrain path
coordinates

% Establish global variables
global X_lr Y_lr Z_lr i_lr X_rr Y_rr Z_rr i_rr X_lf Y_lf Z_lf i_lf X_rf
Y_rf Z_rf i_rf
global phiX_lr phiY_lr phiZ_lr phiX_rr phiY_rr phiZ_rr phiX_lf phiY_lf
phiZ_lf phiX_rf phiY_rf phiZ_rf
global delta_rr delta_lr delta_rf delta_lf

% Rover speed
w=1/3;  % Commanded angular velocity of wheels [rad/s]
w_r=w;  % Skid steering, thus assume right wheels have same w, and
likewise
        % for left wheels
w_l=w;

% Import terrain path map
% pathdata=readtable('FlatTerrain_NoSlip_Test2.xlsx');
% pathdata=readtable('UpslopeTerrain_10deg_NoSlip.xlsx');
% pathdata=readtable('UpslopeTerrain_15deg_NoSlip.xlsx');
```

```
% pathdata=readtable('DownslopeTerrain_10deg_NoSlip.xlsx');
% pathdata=readtable('SideslopeTerrain_10deg_NoSlip.xlsx');
pathdata=readtable('SineTerrain_NoSlip.xlsx');


x_lr=pathdata{1:30,{'x_lr'}};   % Terrain path data for left rear wheel
y_lr=pathdata{1:30,{'y_lr'}};
z_lr=pathdata{1:30,{'z_lr'}};
I_lr=pathdata{1:30,{'i_lr'}};

x_rr=pathdata{1:30,{'x_rr'}};   % Terrain path data for right rear
wheel
y_rr=pathdata{1:30,{'y_rr'}};
z_rr=pathdata{1:30,{'z_rr'}};
I_rr=pathdata{1:30,{'i_rr'}};

x_lf=pathdata{1:30,{'x_lf'}};   % Terrain path data for left front
wheel
y_lf=pathdata{1:30,{'y_lf'}};
z_lf=pathdata{1:30,{'z_lf'}};
I_lf=pathdata{1:30,{'i_lf'}};

x_rf=pathdata{1:30,{'x_rf'}};   % Terrain path data for right front
wheel
y_rf=pathdata{1:30,{'y_rf'}};
z_rf=pathdata{1:30,{'z_rf'}};
I_rf=pathdata{1:30,{'i_rf'}};

% Initial guess for solver
q0 = zeros(8,1);
% q0=[0; 0; 0; 0; 0.1746; 0; -0.1746; 0.1746];

% Set loop conditions
%x_lrmax=x_lr(10);
u=1;
u_max=25;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Start of while loop to solve eqtn set for each position
while u<= u_max


    %Pose of RR wheel-gnd contact pt
      X_rr=x_rr(u);
      Y_rr=y_rr(u);
      Z_rr=z_rr(u);
      i_rr=I_rr(u);


    %Pose of LR wheel-gnd contact pt
      X_lr=x_lr(u);
      Y_lr=y_lr(u);
      Z_lr=z_lr(u);
```

```
      i_lr=I_lr(u);

    %Pose of RF wheel-gnd contact pt
      X_rf=x_rf(u);
      Y_rf=y_rf(u);
      Z_rf=z_rf(u);
      i_rf=I_rf(u);

    %Pose of LF wheel-gnd contact pt
      X_lf=x_lf(u);
      Y_lf=y_lf(u);
      Z_lf=z_lf(u);
      i_lf=I_lf(u);


    % Orientation of contact points
    if u==1
%         phiX_rf=(pi/2)+asin((Z_lf-Z_rf)/1.1786);    % Roll case
        phiX_rf=(pi/2);
%         phiX_rf=0;
        phiX_lf=-phiX_rf;
%         phiX_rr=(pi/2)+asin((Z_lr-Z_rr)/1.1786);
        phiX_rr=(pi/2);
%         phiX_rr=0;
        phiX_lr=-phiX_rr;

        phiY_rf=-asin((Z_rf-Z_rr)/0.9144);  % Pitch
%         phiY_rf=(pi/2)-asin((Z_rf-Z_rr)/0.9144);    % Other
coordinate defn
        phiY_rr=phiY_rf;
        phiY_lf=-asin((Z_lf-Z_lr)/0.9144);
%         phiY_lf=(pi/2)-asin((Z_rf-Z_rr)/0.9144);    % Other
coordinate defn
        phiY_lr=phiY_lf;

        phiZ_rf=-(pi/2);    % Yaw
%         phiZ_rf=0;      % For other coordinate defn
        phiZ_rr=phiZ_rf;
        phiZ_lf=(pi/2);
%         phiZ_lf=0;      % For other coordinate defn
        phiZ_lr=phiZ_lf;

%         delta_rf=asin((Z_rf-Z_rr)/0.9144);  % Pitch
        delta_rf=0;
        delta_rr=delta_rf;
%         delta_lf=-asin((Z_lf-Z_lr)/0.9144);
        delta_lf=0;
        delta_lr=delta_lf;
    else
        if y_rf(u)-y_rf(u-1)==0         %Roll
        phiX_rf=(pi/2);   % Non-side slope
%         phiX_rf=0;
%         phiX_rf=(pi/2)+asin((Z_lf-Z_rf)/1.1786);    % Side slope case
        phiX_lf=-phiX_rf;
        else
```

```
        phiX_rf=atan2((z_rf(u)-z_rf(u-1)),(y_rf(u)-y_rf(u-1)));  % Roll
        phiX_lf=atan2((z_lf(u)-z_lf(u-1)),(y_lf(u)-y_lf(u-1)));
        end

        if y_rr(u)-y_rr(u-1)==0
         phiX_rr=(pi/2);   % Non-slide slope
%         phiX_rr=0;
%         phiX_rr=(pi/2)+asin((Z_lr-Z_rr)/1.1786);    % Side slope
        phiX_lr=-phiX_rr;
        else
        phiX_rr=atan2((z_rr(u)-z_rr(u-1)),(y_rf(u)-y_rr(u-1)));
        phiX_lr=atan2((z_lr(u)-z_lr(u-1)),(y_lr(u)-y_lr(u-1)));
        end

        phiY_rf=-atan((z_rf(u)-z_rf(u-1))/(x_rf(u)-x_rf(u-1))); % Pitch
        phiY_rr=-atan((z_rr(u)-z_rr(u-1))/(x_rr(u)-x_rr(u-1)));
        phiY_lf=-atan((z_lf(u)-z_lf(u-1))/(x_lf(u)-x_lf(u-1)));
        phiY_lr=-atan((z_lr(u)-z_lr(u-1))/(x_lr(u)-x_lr(u-1)));
%         phiY_rf=(pi/2)-atan2((z_rf(u)-z_rf(u-1)),(x_rf(u)-x_rf(u-
1))); % Pitch for other coordinate defn
%         phiY_rr=(pi/2)-atan2((z_rr(u)-z_rr(u-1)),(x_rr(u)-x_rr(u-
1)));
%         phiY_lf=(pi/2)-atan2((z_lf(u)-z_lf(u-1)),(x_lf(u)-x_lf(u-
1)));
%         phiY_lr=(pi/2)-atan2((z_lr(u)-z_lr(u-1)),(x_lr(u)-x_lr(u-
1)));

        phiZ_rf=-(pi/2);    % Yaw
%         phiZ_rf=0;      % Other coordinate defn
        phiZ_rr=phiZ_rf;
        phiZ_lf=(pi/2);
%         phiZ_lf=0;      %Other coordinate defn
        phiZ_lr=phiZ_lf;

%         delta_rf=atan((z_rf(u)-z_rf(u-1))/(x_rf(u)-x_rf(u-1))); %
Pitch
%         delta_rr=atan((z_rr(u)-z_rr(u-1))/(x_rr(u)-x_rr(u-1)));
%         delta_lf=-atan((z_lf(u)-z_lf(u-1))/(x_lf(u)-x_lf(u-1)));
%         delta_lr=-atan((z_lr(u)-z_lr(u-1))/(x_lr(u)-x_lr(u-1)));
        delta_rf=0;
        delta_rr=delta_rf;
        delta_lf=0;
        delta_lr=delta_lf;
    end


    %% Nonlinear Eqtn Solver
    options1 = optimoptions('fsolve');
    options1.Algorithm = 'levenberg-marquardt';
    options1.FunctionTolerance = 1e0;  %e-4 for flat, e-1 for slope, e-
2 for side slope

    qSol=fsolve(@J5posKin3, q0, options1)
```

```
    % Reassign variables for ease of plotting
Q1(u) = qSol(1);  % Xtrans or distance travelled in global X [m]
Q2(u) = qSol(2);  % Ytrans or distance travelled in global Y [m]
Q3(u) = qSol(3);  % Ztrans or distance travelled in global Z [m]
Q4(u) = qSol(4);  % Yaw of the chassis [RAD]
Q5(u) = qSol(5);  % Pitch of the chassis [RAD]
Q6(u) = qSol(6);  % Roll of the chassis [RAD]
Q7(u) = qSol(7);  % Pitch of the right walking beam [RAD]
Q8(u) = qSol(8);  % Pitch of the left walking beam [RAD]
% Q9(u) = qSol(9);  % Contact angle right rear wheel [RAD]
% Q10(u) = qSol(10);    % Contact angle left rear wheel [RAD]
% Q11(u) = qSol(11);    % Contact angle right front wheel [RAD]
% Q12(u) = qSol(12);    % Contact angle left front wheel [RAD]

%if u==1
qSol(1)=qSol(1)-Q1(1);

qSol(2)=qSol(2)-Q2(1);
qSol(3)=qSol(3)-Q3(1);

% if u==1
qSol(5)=2*Q5(u);
qSol(7)=2*Q7(u);
qSol(8)=2*Q8(u);

Q5(u)=qSol(5);
Q7(u)=qSol(7);
Q8(u)=qSol(8);
% end

%end
Q1(u)=qSol(1);
Q2(u)=qSol(2);
Q3(u)=qSol(3);

qSol

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Update parameters prior to exiting the loop

% Upate time step or condition of the loop
% t=t + del_t;
u=u+1;

% Update initial guess
q0=qSol;



end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Display outputs
```

```
% Plot terrain as DEM
[X,Y] = meshgrid(-5:0.5:10,-5:0.5:10);
Z = zeros([31 31]);
figure
%hold on
%title('Terrain map with the selected path')
xlabel('X [m]')
ylabel('Y [m]')
zlabel('Z [m]')

surf(X,Y,Z)

% set(findall(gca, 'Type', 'Line'),'LineWidth',2)
%hold off

% Plot walking beam/bogie angles vs time??
figure
hold on
%title('Walking Beam Pitch') %Plotted with respect to Xdistance
travelled
xlabel('X [m]')
ylabel('Walking Beam Pitch [DEG]')

plot(Q1,(Q7*180/pi))
plot(Q1,(Q8*180/pi))
legend('Right Walking Beam', 'Left Walking Beam','Location', 'east')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off

% Plot chassis angles vs time??
figure
hold on
%title('Chassis Orientation Angles') %Plotted with respect to Xdistance
travelled
xlabel('X [m]')
ylabel('Chassis Orientation Angles [DEG]')

plot(Q1,(Q4*180/pi))    % Yaw
plot(Q1,(Q5*180/pi))    % Pitch
plot(Q1,(Q6*180/pi))    % Roll
legend('Yaw', 'Pitch', 'Roll', 'Location', 'east')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off
```

C.3 MATLAB Function File: 3D Position Equation Set for Solution (J5posKin3.m)

```
function Fval = J5posKin3(q)
%J5posKin This function uses the extracted position kinematics eqtns
for
% the pose of each wheel-gnd contact point to solve for the joint
angles &
% displacements.
%   Each eqtn is extracted from the transform or T-matrix for the
%   corresponding kinematic chain. Details on the derivation of the
%   T-matrices and subsequent pose eqtn sets can be viewed in the
%   corresponding Maple files. The full eqtn set for all wheels are
%   structured in the form F=0,as per the conditions for using fsolve.
%   Global variables determined in the outer loop are brought in as
%   additional inputs to the expected
%   vectors q (joint displacements) and initial guess q0.

%% Declaration of global variables based on terrain path in main script
global X_lr Y_lr Z_lr X_rr Y_rr Z_rr X_lf Y_lf Z_lf X_rf Y_rf Z_rf
global phiX_lr phiY_lr phiZ_lr phiX_rr phiY_rr phiZ_rr phiX_lf phiY_lf
phiZ_lf phiX_rf phiY_rf phiZ_rf
global delta_rr delta_lr delta_rf delta_lf

%% Assignment of constant rover parameters
beta=(71.67*pi/180); % Angle of walking beam, formed btwn the two
wheels [RAD]
h_cog=0.4515;     % Height to centre of gravity [m].
d_comwb=0.5644;  % Distance from CoG to centre of walking beam [m].
a_wb=0.4816;     % Link length from walking beam/bogie pivot pt to
wheel axle [m].
r=0.3;      % Wheel radius [m]

%% Assigning joint variables to the vector q
X_trans=q(1,1);
Y_trans=q(2,1);
Z_trans=q(3,1);
phi_yaw=q(4,1);
phi_pitch=q(5,1);
phi_roll=q(6,1);
theta_wbr=q(7,1);
theta_wbl=q(8,1);
% delta_rr=q(9,1);
% delta_lr=q(10,1);
% delta_rf=q(11,1);
% delta_lf=q(12,1);
% delta_rr=0;
% delta_lr=0;
% delta_rf=0;
% delta_lf=0;
```

```
%% Full set of kinematic eqtns to be solved. See Maple for full
derivation
% Back Suspension Eqtn
Fval(1,1)=theta_wbl-theta_wbr-(2*phi_pitch);

% Right rear wheel eqtns
Fval(2,1)=((-cos(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(be
ta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos(phi_yaw)-
sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*cos(-theta_wbr+beta)+((-
r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))*sin(p
hi_pitch)-cos(phi_pitch)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*cos(phi_yaw)-
r*sin(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(del
ta_rr)))*sin(-theta_wbr+beta)-
sin(phi_pitch)*cos(phi_yaw)*sin(phi_roll)*d_comwb+sin(phi_yaw)*cos(phi_
roll)*d_comwb+X_trans - X_rr;
Fval(3,1)=((-cos(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(be
ta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*sin(phi_yaw)+cos(phi_yaw)*s
in(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*cos(-theta_wbr+beta)+((-
r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))*sin(p
hi_pitch)-cos(phi_pitch)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*sin(phi_yaw)+r*cos(phi_yaw)*sin(phi_ro
ll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*sin(-
theta_wbr+beta)-sin(phi_pitch)*sin(phi_yaw)*sin(phi_roll)*d_comwb-
cos(phi_roll)*cos(phi_yaw)*d_comwb+Y_trans - Y_rr;
Fval(4,1)=(-cos(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*cos(phi_pitch)-
r*sin(phi_pitch)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos
(-theta_wbr+beta)+(-
r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))*cos(p
hi_pitch)+sin(phi_pitch)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*sin(-theta_wbr+beta)-
cos(phi_pitch)*sin(phi_roll)*d_comwb+h_cog+Z_trans - Z_rr;

% Fval(6,1)=asin(sin(phi_roll))-phiX_rr;
Fval(5,1)=(asin((-
cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))*cos(phi
_pitch)-sin(phi_pitch)*(sin(beta)*sin(delta_rr)-
cos(beta)*cos(delta_rr)))*cos(-theta_wbr+beta)-sin(-
theta_wbr+beta)*(cos(phi_roll)*(sin(beta)*sin(delta_rr)-
cos(beta)*cos(delta_rr))*cos(phi_pitch)-
sin(phi_pitch)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))))-
phiY_rr;
%
Fval(6,1)=(asin(((((cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitc
h)*cos(beta))*cos(delta_rr)+sin(delta_rr)*(cos(phi_roll)*sin(phi_pitch)
*cos(beta)-cos(phi_pitch)*sin(beta)))*sin(phi_yaw)-
sin(phi_roll)*cos(phi_yaw)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta
_rr)))*cos(-theta_wbr+beta)-(((cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta))*cos(delta_rr)-
sin(delta_rr)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*co
```

```
s(beta)))*sin(phi_yaw)-sin(phi_roll)*cos(phi_yaw)*(-
sin(beta)*sin(delta_rr)+cos(beta)*cos(delta_rr)))*sin(-
theta_wbr+beta))/cos(phi_pitch)))-phiZ_rr;

% Right front wheel
Fval(6,1)=((-
cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb
)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*cos(phi_yaw)-
sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(d
elta_rf)+a_wb))*cos(theta_wbr+beta)+((r*cos(phi_roll)*(cos(beta)*sin(de
lta_rf)-
sin(beta)*cos(delta_rf))*sin(phi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos
(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb))*cos(phi_yaw)+r*sin(phi_yaw)
*sin(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*sin(theta_wbr+beta)-
sin(phi_pitch)*cos(phi_yaw)*sin(phi_roll)*d_comwb+sin(phi_yaw)*cos(phi_
roll)*d_comwb+X_trans - X_rf;
Fval(7,1)=((-
cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb
)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*sin(phi_yaw)+cos(phi_yaw)*sin(phi_roll)*(r*co
s(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb))*cos(theta_wbr+be
ta)+((r*cos(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf))*sin(phi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos
(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb))*sin(phi_yaw)-
r*cos(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*sin(theta_wbr+beta)-
sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)*d_comwb-
cos(phi_yaw)*cos(phi_roll)*d_comwb+Y_trans - Y_rf;
Fval(8,1)=(-
cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb
)*cos(phi_pitch)-r*sin(phi_pitch)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*cos(theta_wbr+beta)+(r*cos(phi_roll)*(cos(bet
a)*sin(delta_rf)-sin(beta)*cos(delta_rf))*cos(phi_pitch)-
sin(phi_pitch)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_w
b))*sin(theta_wbr+beta)-
cos(phi_pitch)*sin(phi_roll)*d_comwb+h_cog+Z_trans - Z_rf;

% Fval(11,1)=(asin(sin(phi_roll)))-phiX_rf;
Fval(9,1)=(-asin((cos(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf))*cos(phi_pitch)-
sin(phi_pitch)*(sin(beta)*sin(delta_rf)+cos(beta)*cos(delta_rf)))*cos(t
heta_wbr+beta)+sin(theta_wbr+beta)*(cos(phi_roll)*(sin(beta)*sin(delta_
rf)+cos(beta)*cos(delta_rf))*cos(phi_pitch)+sin(phi_pitch)*(cos(beta)*s
in(delta_rf)-sin(beta)*cos(delta_rf)))))-phiY_rf;
% Fval(11,1)=(asin(((((-
cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta))*cos(de
lta_rf)+sin(delta_rf)*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_p
itch)*sin(beta)))*sin(phi_yaw)+sin(phi_roll)*cos(phi_yaw)*(sin(beta)*co
s(delta_rf)-
cos(beta)*sin(delta_rf)))*cos(theta_wbr+beta)+sin(theta_wbr+beta)*(((co
s(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos(delt
a_rf)+sin(delta_rf)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta)))*sin(phi_yaw)-
```

```
sin(phi_roll)*cos(phi_yaw)*(sin(beta)*sin(delta_rf)+cos(beta)*cos(delta
_rf))))/cos(phi_pitch)))-phiZ_rf;

%Left rear wheel
Fval(10,1)=((-
cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb
)*sin(phi_pitch)-r*cos(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*cos(phi_yaw)-
sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(d
elta_lr)+a_wb))*cos(theta_wbl+beta)+((r*cos(phi_roll)*(cos(beta)*sin(de
lta_lr)-sin(beta)*cos(delta_lr))*sin(phi_pitch)-
cos(phi_pitch)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_w
b))*cos(phi_yaw)+r*sin(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*sin(theta_wbl+beta)+sin(phi_pitch)*cos(phi_ya
w)*sin(phi_roll)*d_comwb-sin(phi_yaw)*cos(phi_roll)*d_comwb+X_trans -
X_lr;
Fval(11,1)=((-
cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb
)*sin(phi_pitch)-r*cos(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*sin(phi_yaw)+cos(phi_yaw)*sin(phi_roll)*(r*co
s(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb))*cos(theta_wbl+be
ta)+((r*cos(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*sin(phi_pitch)-
cos(phi_pitch)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_w
b))*sin(phi_yaw)-r*cos(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*sin(theta_wbl+beta)+sin(phi_yaw)*sin(phi_pitc
h)*sin(phi_roll)*d_comwb+cos(phi_yaw)*cos(phi_roll)*d_comwb+Y_trans -
Y_lr;
Fval(12,1)=(-
cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb
)*cos(phi_pitch)+r*sin(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*cos(theta_wbl+beta)+(r*cos(phi_roll)*(cos(bet
a)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*cos(phi_pitch)+sin(phi_pitch)*(r*cos(beta)*cos
(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb))*sin(theta_wbl+beta)+cos(phi
_pitch)*sin(phi_roll)*d_comwb+h_cog+Z_trans - Z_lr;

% Fval(16,1)=(asin(sin(phi_roll)))-phiX_lr;
Fval(13,1)=(asin((cos(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*cos(phi_pitch)+sin(phi_pitch)*(sin(delta_lr)*s
in(beta)+cos(beta)*cos(delta_lr)))*cos(theta_wbl+beta)+sin(theta_wbl+be
ta)*(cos(phi_roll)*(sin(delta_lr)*sin(beta)+cos(beta)*cos(delta_lr))*co
s(phi_pitch)-sin(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))))-phiY_lr;
%
Fval(16,1)=(asin(((((cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pit
ch)*cos(beta))*cos(delta_lr)-
sin(delta_lr)*(cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta)))*sin(phi_yaw)-
sin(phi_roll)*cos(phi_yaw)*(sin(beta)*cos(delta_lr)-
cos(beta)*sin(delta_lr)))*cos(theta_wbl+beta)-
sin(theta_wbl+beta)*(((cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta))*cos(delta_lr)+sin(delta_lr)*(cos(phi_roll)*si
n(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta)))*sin(phi_yaw)-
```

```
sin(phi_roll)*cos(phi_yaw)*(sin(delta_lr)*sin(beta)+cos(beta)*cos(delta
_lr))))/cos(phi_pitch)))-phiZ_lr;

% Left front wheel
Fval(14,1)=((-cos(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*sin(phi_pitch)-
r*cos(phi_pitch)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*cos
(phi_yaw)-sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*cos(-theta_wbl+beta)+((-
r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))*sin(p
hi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*cos(phi_yaw)-
r*sin(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(del
ta_lf)))*sin(-
theta_wbl+beta)+cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)*d_comwb-
cos(phi_roll)*sin(phi_yaw)*d_comwb+X_trans - X_lf;
Fval(15,1)=((-cos(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*sin(phi_pitch)-
r*cos(phi_pitch)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*sin
(phi_yaw)+cos(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*cos(-theta_wbl+beta)+((-
r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))*sin(p
hi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*sin(phi_yaw)+r*cos(phi_yaw)*sin(phi_ro
ll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*sin(-
theta_wbl+beta)+sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)*d_comwb+cos(p
hi_yaw)*cos(phi_roll)*d_comwb+Y_trans - Y_lf;
Fval(16,1)=(-cos(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*cos(phi_pitch)+r*sin(phi_pitch)*(cos(be
ta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*cos(-theta_wbl+beta)+(-
r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))*cos(p
hi_pitch)-sin(phi_pitch)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*sin(-
theta_wbl+beta)+cos(phi_pitch)*sin(phi_roll)*d_comwb+h_cog+Z_trans -
Z_lf;
% Fval(21,1)=(asin(sin(phi_roll)))-phiX_lf;
Fval(17,1)=(-asin((-
cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))*cos(phi
_pitch)+sin(phi_pitch)*(sin(beta)*sin(delta_lf)-
cos(beta)*cos(delta_lf)))*cos(-theta_wbl+beta)-sin(-
theta_wbl+beta)*(cos(phi_roll)*(sin(beta)*sin(delta_lf)-
cos(beta)*cos(delta_lf))*cos(phi_pitch)+sin(phi_pitch)*(cos(beta)*sin(d
elta_lf)+sin(beta)*cos(delta_lf)))))-phiY_lf;
% Fval(21,1)=(-asin(((((cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_lf)+sin(delta_lf)*(cos(phi_roll)*si
n(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta)))*sin(phi_yaw)-
sin(phi_roll)*cos(phi_yaw)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta
_lf)))*cos(-theta_wbl+beta)-
(((cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos
(delta_lf)-sin(delta_lf)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta)))*sin(phi_yaw)-
sin(phi_roll)*cos(phi_yaw)*(cos(beta)*cos(delta_lf)-
sin(beta)*sin(delta_lf)))*sin(-theta_wbl+beta))/cos(phi_pitch)))-
phiZ_lf;
end
```

C.4 MATLAB Script File: 3D Velocity Kinematic Model

(J5_3DVelocityKinematics_v1.m)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% 3D Velocity Kinematics
%%
%% Written by: E. Austen
%%
%% This script solves the inverse velocity kinematics in 3D for the J5
%% rover, for a given set of wheel-ground contact positions and the
%% the solution to the inverse kinematics problem. These contact
%% positions can be either obtained from a DEM of the terrain or
extraction
%% from a specified terrain function. The kinematic equations used were
%% derived using the original Denavit-Hartenberg convention.
%%
%% Created on: June 2, 2019
%% Last Modified:
%% Aug 10, 2019 - new plots
%% Aug 08, 2019 - new Jacobian
%% July 24, 2019 - added index j
%% July 01, 2019 - changed parameter update for better interpolation
%% June 30, 2019 - r
%% June 21, 2019 - updated velocity
%% June 10, 2019 - updated outline in script file
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

clear all;
clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Initialisation of parameters & importing the terrain path
coordinates

% Establish global variables
global X_lr Y_lr Z_lr i_lr X_rr Y_rr Z_rr i_rr X_lf Y_lf Z_lf i_lf X_rf
Y_rf Z_rf i_rf
global phiX_lr phiY_lr phiZ_lr phiX_rr phiY_rr phiZ_rr phiX_lf phiY_lf
phiZ_lf phiX_rf phiY_rf phiZ_rf
global delta_rr delta_lr delta_rf delta_lf

global Xdot_lr Ydot_lr Zdot_lr Xdot_rr Ydot_rr Zdot_rr Xdot_lf Ydot_lf
Zdot_lf Xdot_rf Ydot_rf Zdot_rf
global wX_lr wY_lr wZ_lr wX_rr wY_rr wZ_rr wX_lf wY_lf wZ_lf wX_rf
wY_rf wZ_rf
global deltadot_rr deltadot_lr deltadot_rf deltadot_lf A

% Rover speed
w=1/3;  % Commanded angular velocity of wheels [rad/s]
```

```
w_r=w;  % Skid steering, thus assume right wheels have same w, and
likewise
        % for left wheels
w_l=w;
r=0.3;

% Import terrain path map
% pathdata=readtable('FlatTerrain_NoSlip_Test2.xlsx');
% pathdata=readtable('FlatTerrain_NoSlip.xlsx');
% pathdata=readtable('FlatTerrain_i0.5.xlsx');
% pathdata=readtable('SideslopeTerrain_10deg_NoSlip.xlsx');
% pathdata=readtable('SideslopeTerrain_10deg_i0.5.xlsx');
pathdata=readtable('UpslopeTerrain_10deg_NoSlip.xlsx');
% pathdata=readtable('UpslopeTerrain_10deg_i0.5.xlsx');
% pathdata=readtable('SineTerrain_NoSlip2.xlsx');
% pathdata=readtable('SineTerrain_i0.5.xlsx');


x_lr=pathdata{1:31,{'x_lr'}};   % Terrain path data for left rear wheel
y_lr=pathdata{1:31,{'y_lr'}};
z_lr=pathdata{1:31,{'z_lr'}};
I_lr=pathdata{1:31,{'i_lr'}};   %Note: I is slip value

x_rr=pathdata{1:31,{'x_rr'}};   % Terrain path data for right rear
wheel
y_rr=pathdata{1:31,{'y_rr'}};
z_rr=pathdata{1:31,{'z_rr'}};
I_rr=pathdata{1:31,{'i_rr'}};

x_lf=pathdata{1:31,{'x_lf'}};   % Terrain path data for left front
wheel
y_lf=pathdata{1:31,{'y_lf'}};
z_lf=pathdata{1:31,{'z_lf'}};
I_lf=pathdata{1:31,{'i_lf'}};

x_rf=pathdata{1:31,{'x_rf'}};   % Terrain path data for right front
wheel
y_rf=pathdata{1:31,{'y_rf'}};
z_rf=pathdata{1:31,{'z_rf'}};
I_rf=pathdata{1:31,{'i_rf'}};

% Initial guess for solver
q0 = zeros(8,1);
qdot0 = zeros(8,1);
A=zeros(8,1);


% Set loop conditions, time in secs.
t=0;
del_t=3.048;
tmax=75;
u=1;
j=(t+del_t)/del_t;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Start of while loop to solve eqtn sets for position & velocity for
each
%% time step.

while t<=tmax



    % Orientation of contact points
    if t==0 && u==1

        %Pose of RR wheel-gnd contact pt
          X_rr=x_rr(u);
          Y_rr=y_rr(u);
          Z_rr=z_rr(u);
          i_rr=I_rr(u);
          x_rrt(j)=X_rr;
          y_rrt(j)=Y_rr;
          z_rrt(j)=Z_rr;
          i_rrt(j)=i_rr;
        %Pose of LR wheel-gnd contact pt
          X_lr=x_lr(u);
          Y_lr=y_lr(u);
          Z_lr=z_lr(u);
          i_lr=I_lr(u);
          x_lrt(j)=X_lr;
          y_lrt(j)=Y_lr;
          z_lrt(j)=Z_lr;
          i_lrt(j)=i_lr;
        %Pose of RF wheel-gnd contact pt
          X_rf=x_rf(u);
          Y_rf=y_rf(u);
          Z_rf=z_rf(u);
          i_rf=I_rf(u);
          x_rft(j)=X_rf;
          y_rft(j)=Y_rf;
          z_rft(j)=Z_rf;
          i_rft(j)=i_rf;
        %Pose of LF wheel-gnd contact pt
          X_lf=x_lf(u);
          Y_lf=y_lf(u);
          Z_lf=z_lf(u);
          i_lf=I_lf(u);
          x_lft(j)=X_lf;
          y_lft(j)=Y_lf;
          z_lft(j)=Z_lf;
          i_lft(j)=i_lf;

        %phiX_rf=(pi/2)+asin((Z_lf-Z_rf)/1.1786);    % Roll
        phiX_rf=(pi/2);
        phiX_lf=-phiX_rf;
        %phiX_rr=(pi/2)+asin((Z_lr-Z_rr)/1.1786);
        phiX_rr=(pi/2);
```

```
        phiX_lr=-phiX_rr;

        phiY_rf=-asin((z_rf(u)-z_rr(u))/0.9144);  % Pitch
        phiY_rr=phiY_rf;
        phiY_lf=-asin((z_lf(u)-z_lr(u))/0.9144);
        phiY_lr=phiY_lf;

        phiZ_rf=-(pi/2);    % Yaw
        phiZ_rr=phiZ_rf;
        phiZ_lf=(pi/2);
        phiZ_lr=phiZ_lf;

%         delta_rf=asin((z_rf(u)-z_rr(u))/0.9144);  % Pitch
        delta_rf=0;
        delta_rr=delta_rf;
%         delta_lf=-asin((z_lf(u)-z_lr(u))/0.9144);
        delta_lf=0;
        delta_lr=delta_lf;
    else
        if y_rft(j)-y_rft(j-1)==0         %Roll
        phiX_rf=(pi/2);
%         phiX_rf=(pi/2)+asin((Z_lf-Z_rf)/1.1786);    % Side slope case
        phiX_lf=-phiX_rf;
        else
        phiX_rf=atan((z_rft(j)-z_rft(j-1))/(y_rft(j)-y_rft(j-1)));  %
Roll
        phiX_lf=atan((z_lft(j)-z_lft(j-1))/(y_lft(j)-y_lft(j-1)));
        end

        if y_rrt(j)-y_rrt(j-1)==0
        phiX_rr=(pi/2);
%         phiX_rr=(pi/2)+asin((Z_lr-Z_rr)/1.1786);    % Side slope
        phiX_lr=-phiX_rr;
        else
        phiX_rr=atan((z_rrt(j)-z_rrt(j-1))/(y_rft(j)-y_rrt(j-1)));
        phiX_lr=atan((z_lrt(j)-z_lrt(j-1))/(y_lrt(j)-y_lrt(j-1)));
        end

        phiY_rf=-atan((z_rft(j)-z_rft(j-1))/(x_rft(j)-x_rft(j-1))); %
Pitch
        phiY_rr=-atan((z_rrt(j)-z_rrt(j-1))/(x_rrt(j)-x_rrt(j-1)));
        phiY_lf=-atan((z_lft(j)-z_lft(j-1))/(x_lft(j)-x_lft(j-1)));
        phiY_lr=-atan((z_lrt(j)-z_lrt(j-1))/(x_lrt(j)-x_lrt(j-1)));

        phiZ_rf=-(pi/2);    % Yaw
        phiZ_rr=phiZ_rf;
        phiZ_lf=(pi/2);
        phiZ_lr=phiZ_lf;

%         delta_rf=atan((z_rft(j)-z_rft(j-1))/(x_rft(j)-x_rft(j-1))); %
Pitch
%         delta_rr=atan((z_rrt(j)-z_rrt(j-1)/(x_rrt(j)-x_rrt(j-1))));
%         delta_lf=-atan((z_lft(j)-z_lft(j-1))/(x_lft(j)-x_lft(j-1)));
%         delta_lr=-atan((z_lrt(j)-z_lrt(j-1))/(x_lrt(j)-x_lrt(j-1)));
        delta_rf=0;
```

```
    delta_rr=0;
    delta_lf=0;
    delta_lr=0;

    delta_rft(j)=delta_rf;
    delta_rrt(j)=delta_rr;
    delta_lft(j)=delta_lf;
    delta_lrt(j)=delta_lr;
end

%% End effector velocity inputs
% Right rear wheel
Xdot_rr=-i_rr*w_r*r*cos(-phiY_rr);
Ydot_rr=0;
Zdot_rr=-i_rr*w_r*r*sin(-phiY_rr);
wX_rr=0;
wY_rr=w_r;
wZ_rr=0;

%Left rear wheel
Xdot_lr=-i_lr*w_l*r*cos(-phiY_lr);
Ydot_lr=0;
Zdot_lr=-i_lr*w_l*r*sin(-phiY_lr);
wX_lr=0;
wY_lr=w_l;
wZ_lr=0;

% Right front wheel
Xdot_rf=-i_rf*w_r*r*cos(-phiY_rf);
Ydot_rf=0;
Zdot_rf=-i_rf*w_r*r*sin(-phiY_rf);
wX_rf=0;
wY_rf=w_r;
wZ_rf=0;

% Left front wheel
Xdot_lf=-i_lf*w_l*r*cos(-phiY_lf);
Ydot_lf=0;
Zdot_lf=-i_lf*w_l*r*sin(-phiY_lf);
wX_lf=0;
wY_lf=w_l;
wZ_lf=0;

% Rate of contact angle change
if t==0
    deltadot_rf=0;
    deltadot_rr=0;
    deltadot_lf=0;
    deltadot_lr=0;
else
    deltadot_rf=(delta_rft(j)-delta_rft(j-1))/del_t;
    deltadot_rr=(delta_rrt(j)-delta_rrt(j-1))/del_t;
    deltadot_lf=(delta_lft(j)-delta_lft(j-1))/del_t;
    deltadot_lr=(delta_lrt(j)-delta_lrt(j-1))/del_t;
end
```

```
    %% Nonlinear Eqtn Solver
    options1 = optimoptions('fsolve');
    options1.Algorithm = 'levenberg-marquardt';
    options1.FunctionTolerance = 1e-3;


    qSol=fsolve(@J5posKin3, q0, options1)    % Solves IK for joint
displacements



    %% Reassign and store solutions for ease of plotting

    % Position
Q1(j) = qSol(1);  % Xtrans or distance travelled in global X [m]
Q2(j) = qSol(2);  % Ytrans or distance travelled in global Y [m]
Q3(j) = qSol(3);  % Ztrans or distance travelled in global Z [m]
Q4(j) = qSol(4);  % Yaw of the chassis [RAD]
Q5(j) = qSol(5);  % Pitch of the chassis [RAD]
Q6(j) = qSol(6);  % Roll of the chassis [RAD]
Q7(j) = qSol(7);  % Pitch of the right walking beam [RAD]
Q8(j) = qSol(8);  % Pitch of the left walking beam [RAD]
% Q9(j) = qSol(9);  % Contact angle right rear wheel [RAD]
% Q10(j) = qSol(10);    % Contact angle left rear wheel [RAD]
% Q11(j) = qSol(11);    % Contact angle right front wheel [RAD]
% Q12(j) = qSol(12);    % Contact angle left front wheel [RAD]

% Correction Factors
if j==1
qSol(1)=qSol(1)-Q1(1);

qSol(2)=qSol(2)-Q2(1);
qSol(3)=qSol(3)-Q3(1);

Q1(j)=qSol(1);
Q2(j)=qSol(2);
Q3(j)=qSol(3);
end

% if u==1
qSol(5)=2*Q5(j);
qSol(7)=2*Q7(j);
qSol(8)=2*Q8(j);

Q5(j)=qSol(5);
Q7(j)=qSol(7);
Q8(j)=qSol(8);
% end



qSol
```

```
A=qSol;

    %% Nonlinear Eqtn Solver Inverse Velocity
    options2 = optimoptions('fsolve');
    options2.Algorithm = 'levenberg-marquardt';
    options2.FunctionTolerance = 1e-1;
    options2.StepTolerance = 1e-12;

    qdotSol=fsolve(@J5veloKin2, qdot0, options2)   % Solves IK velocity


    % Velocity
Qdot1(j) = qdotSol(1);  % Xdot or velocity in global X [m/s]
Qdot2(j) = qdotSol(2);  % Ydot of velocity in global Y [m/s]
Qdot3(j) = qdotSol(3);  % Zdot or velocity in global Z [m/s]
Qdot4(j) = qdotSol(4);  % Yaw rate of the chassis [RAD/s]
Qdot5(j) = qdotSol(5);  % Pitch rate of the chassis [RAD/s]
Qdot6(j) = qdotSol(6);  % Roll rate of the chassis [RAD/s]
Qdot7(j) = qdotSol(7);  % Pitch rate of the right walking beam [RAD/s]
Qdot8(j) = qdotSol(8);  % Pitch rate of the left walking beam [RAD/s]
% Qdot9(j) = qdotSol(9);  % Contact angle rate right rear wheel [RAD/s]
% Qdot10(j) = qdotSol(10);    % Contact angle rate left rear wheel
[RAD/s]
% Qdot11(j) = qdotSol(11);    % Contact angle rate right front wheel
[RAD/s]
% Qdot12(j) = qdotSol(12);    % Contact angle rate left front wheel
[RAD/s]

% Plot rover suspension sides for later animation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Update parameters prior to exiting the loop

% Upate time step or condition of the loop
t=t + del_t;
j=int16((t+del_t)/del_t);

% Update initial guesses
q0=qSol;
qdot0=qdotSol;

% Update position of contact points

%% Right rear wheel
v_xaxrr=w_r*r*(1-i_rr)*cos(-phiY_rr);
v_yaxrr=0;
v_zaxrr=w_r*r*(1-i_rr)*sin(-phiY_rr);

x_rrt(j)=x_rrt(j-1) + (v_xaxrr*del_t);
y_rrt(j)=y_rrt(j-1) + (v_yaxrr*del_t);
%z_rrt(j)=z_rrt(j-1) + (v_zaxrr*del_t); % Check for even terrain

if x_rrt(j)==x_rr(u+1) && y_rrt(j)==y_rr(u+1)
    z_rrt(j)=z_rr(u+1);
```

```
    i_rrt(j)=I_rr(u+1);
elseif x_rr(u+1)>x_rrt(j)
        if x_rr(u+1)==x_rrt(j-1)
            ifx=0;
        else
            ifx=(x_rrt(j)-x_rrt(j-1))/(x_rr(u+1)-x_rrt(j-1));
        end
        if y_rr(u+1)==y_rrt(j-1)
            ify=0;
        else
            ify=(y_rrt(j)-y_rrt(j-1))/(y_rr(u+1)-y_rrt(j-1));
        end

        factor=sqrt((ifx^2)+(ify^2));

        z_rrt(j)=(factor*(z_rr(u+1)-z_rrt(j-1)))+z_rrt(j-1);
        i_rrt(j)=(factor*(I_rr(u+1)-i_rrt(j-1)))+i_rrt(j-1);
else
    if x_rr(u+2)==x_rrt(j-1)
            ifx=0;
    else
        ifx=(x_rrt(j)-x_rrt(j-1))/(x_rr(u+2)-x_rrt(j-1));
    end
    if y_rr(u+2)==y_rrt(j-1)
        ify=0;
    else
        ify=(y_rrt(j)-y_rrt(j-1))/(y_rr(u+2)-y_rrt(j-1));
    end

    factor=sqrt((ifx^2)+(ify^2));

    z_rrt(j)=(factor*(z_rr(u+2)-z_rrt(j-1)))+z_rrt(j-1);
    i_rrt(j)=(factor*(I_rr(u+2)-i_rrt(j-1)))+i_rrt(j-1);

end

% if x_rrt(j)>=x_rr(u+1)
%     u=u+1;
% end

%% Right front wheel
v_xaxrf=w_r*r*(1-i_rf)*cos(-phiY_rf);
v_yaxrf=0;
v_zaxrf=w_r*r*(1-i_rf)*sin(-phiY_rf);

x_rft(j)=x_rft(j-1) + (v_xaxrf*del_t);
y_rft(j)=y_rft(j-1) + (v_yaxrf*del_t);
%z_rft(j)=z_rft(j-1) + (v_zaxrf*del_t); % Check for even terrain

if x_rft(j)==x_rf(u+1) && y_rft(j)==y_rf(u+1)
    z_rft(j)=z_rf(u+1);
    i_rft(j)=I_rf(u+1);
elseif x_rf(u+1)>x_rft(j)
        if x_rf(u+1)==x_rft(j-1)
            ifx=0;
```

```
        else
            ifx=(x_rft(j)-x_rft(j-1))/(x_rf(u+1)-x_rft(j-1));
        end
        if y_rf(u+1)==y_rft(j-1)
            ify=0;
        else
            ify=(y_rft(j)-y_rft(j-1))/(y_rf(u+1)-y_rft(j-1));
        end

        factor=sqrt((ifx^2)+(ify^2));

        z_rft(j)=(factor*(z_rf(u+1)-z_rft(j-1)))+z_rft(j-1);
        i_rft(j)=(factor*(I_rf(u+1)-i_rft(j-1)))+i_rft(j-1);
else
    if x_rf(u+2)==x_rft(j-1)
            ifx=0;
    else
        ifx=(x_rft(j)-x_rft(j-1))/(x_rf(u+2)-x_rft(j-1));
    end
    if y_rf(u+2)==y_rft(j-1)
        ify=0;
    else
        ify=(y_rft(j)-y_rft(j-1))/(y_rf(u+2)-y_rft(j-1));
    end

    factor=sqrt((ifx^2)+(ify^2));

    z_rft(j)=(factor*(z_rf(u+2)-z_rft(j-1)))+z_rft(j-1);
    i_rft(j)=(factor*(I_rf(u+2)-i_rft(j-1)))+i_rft(j-1);

end

%% Left rear wheel
v_xaxlr=w_l*r*(1-i_lr)*cos(-phiY_lr);
v_yaxlr=0;
v_zaxlr=w_l*r*(1-i_lr)*sin(-phiY_lr);

x_lrt(j)=x_lrt(j-1) + (v_xaxlr*del_t);
y_lrt(j)=y_lrt(j-1) + (v_yaxlr*del_t);
%z_lrt(j)=z_lrt(j-1) + (v_zaxlr*del_t); % Check for even terrain

if x_lrt(j)==x_lr(u+1) && y_lrt(j)==y_lr(u+1)
    z_lrt(j)=z_lr(u+1);
    i_lrt(j)=I_lr(u+1);
elseif x_lr(u+1)>x_lrt(j)
        if x_lr(u+1)==x_lrt(j-1)
            ifx=0;
        else
            ifx=(x_lrt(j)-x_lrt(j-1))/(x_lr(u+1)-x_lrt(j-1));
        end
        if y_lr(u+1)==y_lrt(j-1)
            ify=0;
        else
            ify=(y_lrt(j)-y_lrt(j-1))/(y_lr(u+1)-y_lrt(j-1));
        end
```

```
        factor=sqrt((ifx^2)+(ify^2));

        z_lrt(j)=(factor*(z_lr(u+1)-z_lrt(j-1)))+z_lrt(j-1);
        i_lrt(j)=(factor*(I_lr(u+1)-i_lrt(j-1)))+i_lrt(j-1);
else
    if x_lr(u+2)==x_lrt(j-1)
            ifx=0;
    else
        ifx=(x_lrt(j)-x_lrt(j-1))/(x_lr(u+2)-x_lrt(j-1));
    end
    if y_lr(u+2)==y_lrt(j-1)
        ify=0;
    else
        ify=(y_lrt(j)-y_lrt(j-1))/(y_lr(u+2)-y_lrt(j-1));
    end

    factor=sqrt((ifx^2)+(ify^2));

    z_lrt(j)=(factor*(z_lr(u+2)-z_lrt(j-1)))+z_lrt(j-1);
    i_lrt(j)=(factor*(I_lr(u+2)-i_lrt(j-1)))+i_lrt(j-1);

end

%% Left front wheel
v_xaxlf=w_l*r*(1-i_lf)*cos(-phiY_lf);
v_yaxlf=0;
v_zaxlf=w_l*r*(1-i_lf)*sin(-phiY_lf);

x_lft(j)=x_lft(j-1) + (v_xaxlf*del_t);
y_lft(j)=y_lft(j-1) + (v_yaxlf*del_t);
%z_lft(j)=z_lft(j-1) + (v_zaxlf*del_t); % Check for even terrain

if x_lft(j)==x_lf(u+1) && y_lft(j)==y_lf(u+1)
    z_lft(j)=z_lf(u+1);
    i_lft(j)=I_lf(u+1);
elseif x_lf(u+1)>x_lft(j)
        if x_lf(u+1)==x_lft(j-1)
            ifx=0;
        else
            ifx=(x_lft(j)-x_lft(j-1))/(x_lf(u+1)-x_lft(j-1));
        end
        if y_lf(u+1)==y_lft(j-1)
            ify=0;
        else
            ify=(y_lft(j)-y_lft(j-1))/(y_lf(u+1)-y_lft(j-1));
        end

        factor=sqrt((ifx^2)+(ify^2));

        z_lft(j)=(factor*(z_lf(u+1)-z_lft(j-1)))+z_lft(j-1);
        i_lft(j)=(factor*(I_lf(u+1)-i_lft(j-1)))+i_lft(j-1);
else
    if x_lf(u+2)==x_lft(j-1)
            ifx=0;
```

```
    else
        ifx=(x_lft(j)-x_lft(j-1))/(x_lf(u+2)-x_lft(j-1));
    end
    if y_lf(u+2)==y_lft(j-1)
        ify=0;
    else
        ify=(y_lft(j)-y_lft(j-1))/(y_lf(u+2)-y_lft(j-1));
    end

    factor=sqrt((ifx^2)+(ify^2));

    z_lft(j)=(factor*(z_lf(u+2)-z_lft(j-1)))+z_lft(j-1);
    i_lft(j)=(factor*(I_lf(u+2)-i_lft(j-1)))+i_lft(j-1);

end

% Update u when necessary
if x_rrt(j)>=x_rr(u+1)
    u=u+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% Update global variable positions
    %Pose of RR wheel-gnd contact pt
      X_rr=x_rrt(j);
      Y_rr=y_rrt(j);
      Z_rr=z_rrt(j);
      i_rr=i_rrt(j);
    %Pose of LR wheel-gnd contact pt
      X_lr=x_lrt(j);
      Y_lr=y_lrt(j);
      Z_lr=z_lrt(j);
      i_lr=i_lrt(j);
    %Pose of RF wheel-gnd contact pt
      X_rf=x_rft(j);
      Y_rf=y_rft(j);
      Z_rf=z_rft(j);
      i_rf=i_rft(j);
    %Pose of LF wheel-gnd contact pt
      X_lf=x_lft(j);
      Y_lf=y_lft(j);
      Z_lf=z_lft(j);
      i_lf=i_lft(j);

end % End of loop


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Display outputs

tplot = linspace(0,(t-1),(j-1));

% Plot terrain as DEM
```

```
[X,Y] = meshgrid(-5:0.5:10,-5:0.5:10);
Z = zeros([31 31]);
figure
% hold on
% title('Terrain map with the selected path')
xlabel('X [m]')
ylabel('Y [m]')
zlabel('Z [m]')

surf(X,Y,Z)

% set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off


% Plot walking beam/bogie angles vs distance
figure
hold on
% title('Walking Beam Pitch') %Plotted with respect to Xdistance
travelled
xlabel('X [m]')
ylabel('Walking Beam Pitch [RAD]')

plot(Q1,Q7)
plot(Q1,Q8)
legend('Right Walking Beam', 'Left Walking Beam','Location', 'east')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off


% Plot walking beam angular rate vs time??
figure
hold on
% title('Walking Beam Pitch Rate') %Plotted with respect to time
travelled
xlabel('Time [s]')
ylabel('Walking Beam Pitch Rate [RAD/s]')

plot(tplot,Qdot7)
plot(tplot,Qdot8)
legend('Right Walking Beam', 'Left Walking Beam','Location', 'east')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off


% Plot chassis angles vs time??
figure
hold on
% title('Chassis Orientation Angles wrt Time') %Plotted with respect to
time travelled
xlabel('Time [s]')
ylabel('Chassis Orientation Angles [RAD]')

plot(tplot,Q4)    % Yaw
plot(tplot,Q5)    % Pitch
```

```
plot(tplot,Q6)    % Roll
legend('Yaw', 'Pitch', 'Roll', 'Location', 'east')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off


% Plot chassis angles vs time??
figure
hold on
% title('Chassis Orientation Angular Rates wrt Time') %Plotted with
respect to time travelled
xlabel('Time [s]')
ylabel('Chassis Angular Rates [RAD/s]')

plot(tplot,Qdot4)    % Yaw
plot(tplot,Qdot5)    % Pitch
plot(tplot,Qdot6)    % Roll
legend('Yaw', 'Pitch', 'Roll', 'Location', 'east')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off

% Plot distance travelled vs time
figure
hold on
% title('Rover Distance Travelled') %Plotted with respect to time
travelled
xlabel('Time [s]')
ylabel('Rover Distance Travelled [m]')

plot(tplot,Q1)
plot(tplot,Q2)
plot(tplot,Q3)
legend('Global X-direction', 'Global Y-direction', 'Global Z-
direction', 'Location', 'east')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off


% Plot translational velocity vs time
figure
hold on
% title('Rover Translational Velocity') %Plotted with respect to time
travelled
xlabel('Time [s]')
ylabel('Rover Translational Velocity [m/s]')

plot(tplot,Qdot1)
plot(tplot,Qdot2)
plot(tplot,Qdot3)
legend('Global X-direction', 'Global Y-direction', 'Global Z-
direction', 'Location', 'east')
set(findall(gca, 'Type', 'Line'),'LineWidth',2)
hold off
```

C.5 MATLAB Function File: 3D Velocity Equation Set for Solution (J5veloKin.m)

```
function V = J5veloKin(qdot)
%J5veloKin This function contains the extracted velocity kinematics for
%each wheel-gnd contact point to enable solution of individual joint
%angle and displacement rates.
%   Each eqtn is extracted from the velocity kinematics eqtn involving
the
%   Jacobian matrix. Details on the derivation of the Jacobian and
%   subsequent vector eqtn sets can be viewed in the corresponding
Maple
%   files. The full eqtn set for all wheels are structured in the form
%   V=0,as per the conditions for using fsolve. Global variables
determined
%   in the outer loop are brought in as additional inputs to the
expected
%   vectors q (joint angles & displacements) and initial guess qdot0
%   (joint angular & displacement rates).

%% Declaration of global variables based on terrain path in main script
global Xdot_lr Ydot_lr Zdot_lr Xdot_rr Ydot_rr Zdot_rr Xdot_lf Ydot_lf
Zdot_lf Xdot_rf Ydot_rf Zdot_rf
global wX_lr wY_lr wZ_lr wX_rr wY_rr wZ_rr wX_lf wY_lf wZ_lf wX_rf
wY_rf wZ_rf
global delta_rr delta_lr delta_rf delta_lf
global deltadot_rr deltadot_lr deltadot_rf deltadot_lf A

%% Assignment of constant rover parameters
beta=(71.67*pi/180); % Angle of walking beam, formed btwn the two
wheels [RAD]
h_cog=0.4515;     % Height to centre of gravity [m].
d_comwb=0.5644;  % Distance from CoG to centre of walking beam [m].
a_wb=0.4816;     % Link length from walking beam/bogie pivot pt to
wheel axle [m].
r=0.3;      % Wheel radius [m]

%% Assigning joint variables to the vector q
X_trans=A(1,1);
Y_trans=A(2,1);
Z_trans=A(3,1);
phi_yaw=A(4,1);
phi_pitch=A(5,1);
phi_roll=A(6,1);
theta_wbr=A(7,1);
theta_wbl=A(8,1);

%% Assigning joint variables to the vector qdot
Xdot_trans=qdot(1,1);
Ydot_trans=qdot(2,1);
Zdot_trans=qdot(3,1);
phidot_yaw=qdot(4,1);
```

```
phidot_pitch=qdot(5,1);
phidot_roll=qdot(6,1);
thetadot_wbr=qdot(7,1);
thetadot_wbl=qdot(8,1);


%% Full set of kinematic eqtns to be solved. See Maple for full
derivation

% Right rear wheel eqtns
V(1,1)=(1)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(((cos(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*sin(phi_pitch)-
r*cos(phi_pitch)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*sin
(phi_yaw)-cos(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*cos(-
theta_wbr+beta)+((r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*co
s(delta_rr))*sin(phi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*sin(phi_yaw)-
r*cos(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(del
ta_rr)))*sin(-
theta_wbr+beta)+d_comwb*(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(
phi_yaw)*cos(phi_roll)))*phidot_yaw + (-
cos(phi_yaw)*((cos(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*cos(phi_pitch)+r*sin(phi_pitch)*(cos(be
ta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos(-
theta_wbr+beta)+(r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos
(delta_rr))*cos(phi_pitch)-sin(phi_pitch)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*sin(-
theta_wbr+beta)+cos(phi_pitch)*sin(phi_roll)*d_comwb))*phidot_pitch +
((cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*cos(-
theta_wbr+beta)+r*(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delt
a_rr))*sin(-theta_wbr+beta)-
d_comwb*(cos(phi_yaw)*sin(phi_pitch)*cos(phi_roll)+sin(phi_yaw)*sin(phi
_roll)))*phidot_roll + (((r*(-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos(de
lta_rr)+r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(be
ta))*sin(delta_rr)-
cos(phi_roll)*sin(phi_pitch)*a_wb)*cos(phi_yaw)+sin(phi_yaw)*sin(phi_ro
ll)*(r*sin(beta)*sin(delta_rr)-r*cos(beta)*cos(delta_rr)-a_wb))*sin(-
theta_wbr+beta)+((r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pit
ch)*cos(beta))*cos(delta_rr)-r*(-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*sin(de
lta_rr)+cos(phi_pitch)*a_wb)*cos(phi_yaw)+r*sin(phi_yaw)*sin(phi_roll)*
(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos(-
theta_wbr+beta))*thetadot_wbr
+(((((cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta))*
cos(delta_rr)+sin(delta_rr)*(cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta)))*cos(phi_yaw)+sin(phi_yaw)*sin(phi_roll)*(cos
(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos(-theta_wbr+beta)-
(((cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta))*cos(delta_rr)-
```

```
sin(delta_rr)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*co
s(beta)))*cos(phi_yaw)+sin(phi_yaw)*sin(phi_roll)*(cos(beta)*cos(delta_
rr)-sin(beta)*sin(delta_rr)))*sin(-theta_wbr+beta))*r)*deltadot_rr -
Xdot_rr;
V(2,1)=(0)*Xdot_trans + (1)*Ydot_trans + (0)*Zdot_trans + (((-
cos(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(be
ta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos(phi_yaw)-
sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*cos(-theta_wbr+beta)+((-
r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))*sin(p
hi_pitch)-cos(phi_pitch)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*cos(phi_yaw)-
r*sin(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(del
ta_rr)))*sin(-theta_wbr+beta)-
d_comwb*(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll)))*phidot_yaw + (-
sin(phi_yaw)*((cos(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*cos(phi_pitch)+r*sin(phi_pitch)*(cos(be
ta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos(-
theta_wbr+beta)+(r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos
(delta_rr))*cos(phi_pitch)-sin(phi_pitch)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*sin(-
theta_wbr+beta)+cos(phi_pitch)*sin(phi_roll)*d_comwb))*phidot_pitch +
((sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))
*(r*cos(beta)*cos(delta_rr)-r*sin(beta)*sin(delta_rr)+a_wb)*cos(-
theta_wbr+beta)+r*(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_ya
w)*cos(phi_roll))*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))*sin
(-theta_wbr+beta)+(-
sin(phi_yaw)*sin(phi_pitch)*cos(phi_roll)+cos(phi_yaw)*sin(phi_roll))*d
_comwb)*phidot_roll + (((r*(-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos(de
lta_rr)+r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(be
ta))*sin(delta_rr)-cos(phi_roll)*sin(phi_pitch)*a_wb)*sin(phi_yaw)-
cos(phi_yaw)*sin(phi_roll)*(r*sin(beta)*sin(delta_rr)-
r*cos(beta)*cos(delta_rr)-a_wb))*sin(-theta_wbr+beta)-cos(-
theta_wbr+beta)*((-
r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta))*cos
(delta_rr)+r*(-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*sin(de
lta_rr)-
cos(phi_pitch)*a_wb)*sin(phi_yaw)+r*cos(phi_yaw)*sin(phi_roll)*(cos(bet
a)*sin(delta_rr)+sin(beta)*cos(delta_rr))))*thetadot_wbr +(-((((-
cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_rr)-
sin(delta_rr)*(cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta)))*sin(phi_yaw)+sin(phi_roll)*cos(phi_yaw)*(cos
(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos(-theta_wbr+beta)-
sin(-theta_wbr+beta)*(((-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos(de
lta_rr)+sin(delta_rr)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_p
itch)*cos(beta)))*sin(phi_yaw)+sin(phi_roll)*cos(phi_yaw)*(cos(beta)*co
s(delta_rr)-sin(beta)*sin(delta_rr))))*r)*deltadot_rr - Ydot_rr;
V(3,1)=(0)*Xdot_trans + (0)*Ydot_trans + (1)*Zdot_trans +
(0)*phidot_yaw + ((-cos(phi_roll)*(r*sin(beta)*sin(delta_rr)-
```

```
r*cos(beta)*cos(delta_rr)-a_wb)*sin(phi_pitch)-
r*cos(phi_pitch)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*cos
(-
theta_wbr+beta)+(r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos
(delta_rr))*sin(phi_pitch)-cos(phi_pitch)*(r*sin(beta)*sin(delta_rr)-
r*cos(beta)*cos(delta_rr)-a_wb))*sin(-
theta_wbr+beta)+sin(phi_pitch)*sin(phi_roll)*d_comwb)*phidot_pitch +
((-sin(phi_roll)*(r*sin(beta)*sin(delta_rr)-r*cos(beta)*cos(delta_rr)-
a_wb)*cos(-
theta_wbr+beta)+r*sin(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(
delta_rr))*sin(-theta_wbr+beta)-
cos(phi_roll)*d_comwb)*cos(phi_pitch))*phidot_roll +
((r*cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))*cos
(phi_pitch)-sin(phi_pitch)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb))*cos(-theta_wbr+beta)-
(cos(phi_roll)*(r*cos(beta)*cos(delta_rr)-
r*sin(beta)*sin(delta_rr)+a_wb)*cos(phi_pitch)+r*sin(phi_pitch)*(cos(be
ta)*sin(delta_rr)+sin(beta)*cos(delta_rr)))*sin(-
theta_wbr+beta))*thetadot_wbr
+(((cos(phi_roll)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))*cos
(phi_pitch)+sin(phi_pitch)*(sin(beta)*sin(delta_rr)-
cos(beta)*cos(delta_rr)))*cos(-theta_wbr+beta)+sin(-
theta_wbr+beta)*(cos(phi_roll)*(sin(beta)*sin(delta_rr)-
cos(beta)*cos(delta_rr))*cos(phi_pitch)-
sin(phi_pitch)*(cos(beta)*sin(delta_rr)+sin(beta)*cos(delta_rr))))*r)*d
eltadot_rr - Zdot_rr;

V(4,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans + (-
sin(phi_yaw))*phidot_yaw + (cos(phi_yaw)*cos(phi_pitch))*phidot_pitch +
(-
cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+sin(phi_yaw)*cos(phi_roll))*p
hidot_roll + (-
cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+sin(phi_yaw)*cos(phi_roll))*t
hetadot_wbr + (-
cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+sin(phi_yaw)*cos(phi_roll))*d
eltadot_rr - wX_rr;
V(5,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(cos(phi_yaw))*phidot_yaw + (sin(phi_yaw)*cos(phi_pitch))*phidot_pitch
+ (-sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
cos(phi_yaw)*cos(phi_roll))*phidot_roll + (-
sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
cos(phi_yaw)*cos(phi_roll))*thetadot_wbr + (-
sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
cos(phi_yaw)*cos(phi_roll))*deltadot_rr - wY_rr;
V(6,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(0)*phidot_yaw + (-sin(phi_pitch))*phidot_pitch + (-
cos(phi_pitch)*sin(phi_roll))*phidot_roll + (-
cos(phi_pitch)*sin(phi_roll))*thetadot_wbr + (-
cos(phi_pitch)*sin(phi_roll))*deltadot_rr - wZ_rr;

% Right front wheel eqtns
V(7,1)=(1)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(((cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a
_wb)*sin(phi_pitch)-r*cos(phi_pitch)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*sin(phi_yaw)-
```

```
cos(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(d
elta_rf)+a_wb))*cos(theta_wbr+beta)+((-
r*cos(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf))*sin(phi_pitch)-
cos(phi_pitch)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_w
b))*sin(phi_yaw)+r*cos(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*sin(theta_wbr+beta)+d_comwb*(sin(phi_yaw)*sin
(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll)))*phidot_yaw + (-
cos(phi_yaw)*((cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin
(delta_rf)+a_wb)*cos(phi_pitch)+r*sin(phi_pitch)*(cos(beta)*sin(delta_r
f)-sin(beta)*cos(delta_rf)))*cos(theta_wbr+beta)+(-
r*cos(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf))*cos(phi_pitch)+sin(phi_pitch)*(r*cos(beta)*cos
(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb))*sin(theta_wbr+beta)+cos(phi
_pitch)*sin(phi_roll)*d_comwb))*phidot_pitch +
((cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(
delta_rf)+a_wb)*cos(theta_wbr+beta)-
r*(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf))*sin(theta_wbr+beta)-
d_comwb*(cos(phi_yaw)*sin(phi_pitch)*cos(phi_roll)+sin(phi_yaw)*sin(phi
_roll)))*phidot_roll +
(((r*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*
cos(delta_rf)+r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*sin(delta_rf)+sin(phi_pitch)*cos(phi_roll)*a_
wb)*cos(phi_yaw)+sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_rf)+
r*sin(beta)*sin(delta_rf)+a_wb))*sin(theta_wbr+beta)-
cos(theta_wbr+beta)*((r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_rf)-
r*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*sin
(delta_rf)-
cos(phi_pitch)*a_wb)*cos(phi_yaw)+r*sin(phi_yaw)*sin(phi_roll)*(sin(bet
a)*cos(delta_rf)-cos(beta)*sin(delta_rf))))*thetadot_wbr +(-
((((cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_rf)-
sin(delta_rf)*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*si
n(beta)))*cos(phi_yaw)+sin(phi_yaw)*sin(phi_roll)*(sin(beta)*cos(delta_
rf)-cos(beta)*sin(delta_rf)))*cos(theta_wbr+beta)-
(((cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos
(delta_rf)+sin(delta_rf)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta)))*cos(phi_yaw)+sin(phi_yaw)*sin(phi_roll)*(sin
(beta)*sin(delta_rf)+cos(beta)*cos(delta_rf)))*sin(theta_wbr+beta))*r)*
deltadot_rf - Xdot_rf;
V(8,1)=(0)*Xdot_trans + (1)*Ydot_trans + (0)*Zdot_trans + (((-
cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb
)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*cos(phi_yaw)-
sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(d
elta_rf)+a_wb))*cos(theta_wbr+beta)+((r*cos(phi_roll)*(cos(beta)*sin(de
lta_rf)-
sin(beta)*cos(delta_rf))*sin(phi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos
(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb))*cos(phi_yaw)+r*sin(phi_yaw)
*sin(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*sin(theta_wbr+beta)-
```

```
d_comwb*(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll)))*phidot_yaw + (-
sin(phi_yaw)*((cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin
(delta_rf)+a_wb)*cos(phi_pitch)+r*sin(phi_pitch)*(cos(beta)*sin(delta_r
f)-sin(beta)*cos(delta_rf)))*cos(theta_wbr+beta)+(-
r*cos(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf))*cos(phi_pitch)+sin(phi_pitch)*(r*cos(beta)*cos
(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb))*sin(theta_wbr+beta)+cos(phi
_pitch)*sin(phi_roll)*d_comwb))*phidot_pitch +
((sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))
*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_wb)*cos(theta_w
br+beta)-
r*(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll)
)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf))*sin(theta_wbr+beta)+d_comwb*(-
sin(phi_yaw)*sin(phi_pitch)*cos(phi_roll)+cos(phi_yaw)*sin(phi_roll)))*
phidot_roll +
(((r*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*
cos(delta_rf)+r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*sin(delta_rf)+sin(phi_pitch)*cos(phi_roll)*a_
wb)*sin(phi_yaw)-
cos(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(d
elta_rf)+a_wb))*sin(theta_wbr+beta)+((-
r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_rf)+r*(cos(phi_roll)*sin(phi_pitch)
*cos(beta)+cos(phi_pitch)*sin(beta))*sin(delta_rf)+cos(phi_pitch)*a_wb)
*sin(phi_yaw)+r*cos(phi_yaw)*sin(phi_roll)*(sin(beta)*cos(delta_rf)-
cos(beta)*sin(delta_rf)))*cos(theta_wbr+beta))*thetadot_wbr +(r*((((-
cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta))*cos(de
lta_rf)+sin(delta_rf)*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_p
itch)*sin(beta)))*sin(phi_yaw)+sin(phi_roll)*cos(phi_yaw)*(sin(beta)*co
s(delta_rf)-cos(beta)*sin(delta_rf)))*cos(theta_wbr+beta)-
sin(theta_wbr+beta)*(((-cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta))*cos(delta_rf)-
sin(delta_rf)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta)))*sin(phi_yaw)+sin(phi_roll)*cos(phi_yaw)*(sin
(beta)*sin(delta_rf)+cos(beta)*cos(delta_rf)))))*deltadot_rf - Ydot_rf;
V(9,1)=(0)*Xdot_trans + (0)*Ydot_trans + (1)*Zdot_trans +
(0)*phidot_yaw +
((cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_
wb)*sin(phi_pitch)+cos(phi_pitch)*r*(sin(beta)*cos(delta_rf)-
cos(beta)*sin(delta_rf)))*cos(theta_wbr+beta)+(r*cos(phi_roll)*(sin(bet
a)*cos(delta_rf)-cos(beta)*sin(delta_rf))*sin(phi_pitch)-
cos(phi_pitch)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_w
b))*sin(theta_wbr+beta)+sin(phi_pitch)*sin(phi_roll)*d_comwb)*phidot_pi
tch +
((sin(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_
wb)*cos(theta_wbr+beta)+r*sin(phi_roll)*(sin(beta)*cos(delta_rf)-
cos(beta)*sin(delta_rf))*sin(theta_wbr+beta)-
cos(phi_roll)*d_comwb)*cos(phi_pitch))*phidot_roll + ((-
r*cos(phi_roll)*(sin(beta)*cos(delta_rf)-
cos(beta)*sin(delta_rf))*cos(phi_pitch)-
sin(phi_pitch)*(r*cos(beta)*cos(delta_rf)+r*sin(beta)*sin(delta_rf)+a_w
b))*cos(theta_wbr+beta)+(cos(phi_roll)*(r*cos(beta)*cos(delta_rf)+r*sin
(beta)*sin(delta_rf)+a_wb)*cos(phi_pitch)-
```

```
r*sin(phi_pitch)*(sin(beta)*cos(delta_rf)-
cos(beta)*sin(delta_rf)))*sin(theta_wbr+beta))*thetadot_wbr
+(((cos(phi_roll)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf))*cos(phi_pitch)-
sin(phi_pitch)*(sin(beta)*sin(delta_rf)+cos(beta)*cos(delta_rf)))*cos(t
heta_wbr+beta)+(cos(phi_roll)*(sin(beta)*sin(delta_rf)+cos(beta)*cos(de
lta_rf))*cos(phi_pitch)+sin(phi_pitch)*(cos(beta)*sin(delta_rf)-
sin(beta)*cos(delta_rf)))*sin(theta_wbr+beta))*r)*deltadot_rf -
Zdot_rf;

V(10,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans + (-
sin(phi_yaw))*phidot_yaw + (cos(phi_yaw)*cos(phi_pitch))*phidot_pitch +
(-
cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+sin(phi_yaw)*cos(phi_roll))*p
hidot_roll + (-
cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+sin(phi_yaw)*cos(phi_roll))*t
hetadot_wbr + (-
cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+sin(phi_yaw)*cos(phi_roll))*d
eltadot_rf - wX_rf;
V(11,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(cos(phi_yaw))*phidot_yaw + (sin(phi_yaw)*cos(phi_pitch))*phidot_pitch
+ (-sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
cos(phi_yaw)*cos(phi_roll))*phidot_roll + (-
sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
cos(phi_yaw)*cos(phi_roll))*thetadot_wbr + (-
sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
cos(phi_yaw)*cos(phi_roll))*deltadot_rf - wY_rf;
V(12,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(0)*phidot_yaw + (-sin(phi_pitch))*phidot_pitch + (-
cos(phi_pitch)*sin(phi_roll))*phidot_roll + (-
cos(phi_pitch)*sin(phi_roll))*thetadot_wbr +(-
cos(phi_pitch)*sin(phi_roll))*deltadot_rf - wZ_rf;

% Left rear wheel eqtns
V(13,1)=(1)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(((cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a
_wb)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*sin(phi_yaw)-
cos(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(d
elta_lr)+a_wb))*cos(theta_wbl+beta)+((-
r*cos(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*sin(phi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos
(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb))*sin(phi_yaw)+r*cos(phi_yaw)
*sin(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*sin(theta_wbl+beta)-
(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))*
d_comwb)*phidot_yaw + (-
cos(phi_yaw)*((cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin
(delta_lr)+a_wb)*cos(phi_pitch)-
r*sin(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*cos(theta_wbl+beta)+(-
r*cos(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*cos(phi_pitch)-
sin(phi_pitch)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_w
b))*sin(theta_wbl+beta)-
```

```
cos(phi_pitch)*sin(phi_roll)*d_comwb))*phidot_pitch +
((cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(
delta_lr)+a_wb)*cos(theta_wbl+beta)-
r*(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*sin(theta_wbl+beta)+d_comwb*(cos(phi_yaw)*sin(
phi_pitch)*cos(phi_roll)+sin(phi_yaw)*sin(phi_roll)))*phidot_roll +
(((-r*(-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos(de
lta_lr)+r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(be
ta))*sin(delta_lr)+cos(phi_roll)*sin(phi_pitch)*a_wb)*cos(phi_yaw)+sin(
phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta
_lr)+a_wb))*sin(theta_wbl+beta)-
cos(theta_wbl+beta)*((r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi
_pitch)*cos(beta))*cos(delta_lr)+r*(-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*sin(de
lta_lr)+cos(phi_pitch)*a_wb)*cos(phi_yaw)+r*sin(phi_yaw)*sin(phi_roll)*
(sin(beta)*cos(delta_lr)-cos(beta)*sin(delta_lr))))*thetadot_wbl +(-
((((cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta))*co
s(delta_lr)-sin(delta_lr)*(cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta)))*cos(phi_yaw)+sin(phi_yaw)*sin(phi_roll)*(sin
(beta)*cos(delta_lr)-cos(beta)*sin(delta_lr)))*cos(theta_wbl+beta)-
sin(theta_wbl+beta)*(((cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta))*cos(delta_lr)+sin(delta_lr)*(cos(phi_roll)*si
n(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta)))*cos(phi_yaw)+sin(phi_
yaw)*sin(phi_roll)*(sin(beta)*sin(delta_lr)+cos(beta)*cos(delta_lr))))*
r)*deltadot_lr - Xdot_lr;
V(14,1)=(0)*Xdot_trans + (1)*Ydot_trans + (0)*Zdot_trans + (((-
cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb
)*sin(phi_pitch)-r*cos(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*cos(phi_yaw)-
sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(d
elta_lr)+a_wb))*cos(theta_wbl+beta)+((r*cos(phi_roll)*(cos(beta)*sin(de
lta_lr)-sin(beta)*cos(delta_lr))*sin(phi_pitch)-
cos(phi_pitch)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_w
b))*cos(phi_yaw)+r*sin(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*sin(theta_wbl+beta)+(cos(phi_yaw)*sin(phi_pit
ch)*sin(phi_roll)-sin(phi_yaw)*cos(phi_roll))*d_comwb)*phidot_yaw + (-
sin(phi_yaw)*((cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin
(delta_lr)+a_wb)*cos(phi_pitch)-
r*sin(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*cos(theta_wbl+beta)+(-
r*cos(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*cos(phi_pitch)-
sin(phi_pitch)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_w
b))*sin(theta_wbl+beta)-
cos(phi_pitch)*sin(phi_roll)*d_comwb))*phidot_pitch +
((sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))
*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb)*cos(theta_w
bl+beta)-
r*(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll)
)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*sin(theta_wbl+beta)-d_comwb*(-
sin(phi_yaw)*sin(phi_pitch)*cos(phi_roll)+cos(phi_yaw)*sin(phi_roll)))*
```

```
phidot_roll + (((-r*(-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos(de
lta_lr)+r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(be
ta))*sin(delta_lr)+cos(phi_roll)*sin(phi_pitch)*a_wb)*sin(phi_yaw)-
cos(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(d
elta_lr)+a_wb))*sin(theta_wbl+beta)+cos(theta_wbl+beta)*((-
r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta))*cos
(delta_lr)-r*(-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*sin(de
lta_lr)-
cos(phi_pitch)*a_wb)*sin(phi_yaw)+r*cos(phi_yaw)*sin(phi_roll)*(sin(bet
a)*cos(delta_lr)-cos(beta)*sin(delta_lr))))*thetadot_wbl +(((((-
cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_lr)+sin(delta_lr)*(cos(phi_roll)*si
n(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta)))*sin(phi_yaw)+sin(phi_roll)*cos(phi_yaw)*(sin
(beta)*cos(delta_lr)-cos(beta)*sin(delta_lr)))*cos(theta_wbl+beta)-
sin(theta_wbl+beta)*(((-
cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos(de
lta_lr)-
sin(delta_lr)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*co
s(beta)))*sin(phi_yaw)+sin(phi_roll)*cos(phi_yaw)*(sin(beta)*sin(delta_
lr)+cos(beta)*cos(delta_lr))))*r)*deltadot_lr - Ydot_lr;
V(15,1)=(0)*Xdot_trans + (0)*Ydot_trans + (1)*Zdot_trans +
(0)*phidot_yaw +
((cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_
wb)*sin(phi_pitch)-cos(phi_pitch)*r*(sin(beta)*cos(delta_lr)-
cos(beta)*sin(delta_lr)))*cos(theta_wbl+beta)+(r*cos(phi_roll)*(sin(bet
a)*cos(delta_lr)-
cos(beta)*sin(delta_lr))*sin(phi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos
(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb))*sin(theta_wbl+beta)-
sin(phi_pitch)*sin(phi_roll)*d_comwb)*phidot_pitch +
((sin(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(delta_lr)+a_
wb)*cos(theta_wbl+beta)+r*sin(phi_roll)*(sin(beta)*cos(delta_lr)-
cos(beta)*sin(delta_lr))*sin(theta_wbl+beta)+cos(phi_roll)*d_comwb)*cos
(phi_pitch))*phidot_roll + ((-r*cos(phi_roll)*(sin(beta)*cos(delta_lr)-
cos(beta)*sin(delta_lr))*cos(phi_pitch)+sin(phi_pitch)*(r*cos(beta)*cos
(delta_lr)+r*sin(beta)*sin(delta_lr)+a_wb))*cos(theta_wbl+beta)+sin(the
ta_wbl+beta)*(cos(phi_roll)*(r*cos(beta)*cos(delta_lr)+r*sin(beta)*sin(
delta_lr)+a_wb)*cos(phi_pitch)+r*sin(phi_pitch)*(sin(beta)*cos(delta_lr
)-cos(beta)*sin(delta_lr))))*thetadot_wbl
+(((cos(phi_roll)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr))*cos(phi_pitch)+sin(phi_pitch)*(sin(beta)*sin(d
elta_lr)+cos(beta)*cos(delta_lr)))*cos(theta_wbl+beta)+(cos(phi_roll)*(
sin(beta)*sin(delta_lr)+cos(beta)*cos(delta_lr))*cos(phi_pitch)-
sin(phi_pitch)*(cos(beta)*sin(delta_lr)-
sin(beta)*cos(delta_lr)))*sin(theta_wbl+beta))*r)*deltadot_lr -
Zdot_lr;

V(16,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans + (-
sin(phi_yaw))*phidot_yaw + (cos(phi_yaw)*cos(phi_pitch))*phidot_pitch +
(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*phidot_roll +
(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*thetadot_wbl
```

```
+(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*deltadot_lr - wX_lr;
V(17,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(cos(phi_yaw))*phidot_yaw + (sin(phi_yaw)*cos(phi_pitch))*phidot_pitch
+
(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))*
phidot_roll +
(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))*
thetadot_wbl
+(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))
*deltadot_lr - wY_lr;
V(18,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(0)*phidot_yaw + (-sin(phi_pitch))*phidot_pitch +
(cos(phi_pitch)*sin(phi_roll))*phidot_roll +
(cos(phi_pitch)*sin(phi_roll))*thetadot_wbl
+(cos(phi_pitch)*sin(phi_roll))*deltadot_lr - wZ_lr;

% Left front wheel eqtns
V(19,1)=(1)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(((cos(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(be
ta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*sin(phi_yaw)-
cos(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*cos(-
theta_wbl+beta)+((r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*co
s(delta_lf))*sin(phi_pitch)-cos(phi_pitch)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*sin(phi_yaw)-
r*cos(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(del
ta_lf)))*sin(-theta_wbl+beta)-
(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))*
d_comwb)*phidot_yaw + (-((cos(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*cos(phi_pitch)-
r*sin(phi_pitch)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*cos
(-
theta_wbl+beta)+(r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos
(delta_lf))*cos(phi_pitch)+sin(phi_pitch)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*sin(-theta_wbl+beta)-
cos(phi_pitch)*sin(phi_roll)*d_comwb)*cos(phi_yaw))*phidot_pitch +
((cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*cos(-
theta_wbl+beta)+r*(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delt
a_lf))*sin(-
theta_wbl+beta)+d_comwb*(cos(phi_yaw)*sin(phi_pitch)*cos(phi_roll)+sin(
phi_yaw)*sin(phi_roll)))*phidot_roll + (((-
r*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos
(delta_lf)+r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*sin(delta_lf)-
cos(phi_roll)*sin(phi_pitch)*a_wb)*cos(phi_yaw)+sin(phi_yaw)*sin(phi_ro
ll)*(r*sin(beta)*sin(delta_lf)-r*cos(beta)*cos(delta_lf)-a_wb))*sin(-
theta_wbl+beta)+((r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_lf)+r*(cos(phi_roll)*sin(phi_pitch)
*cos(beta)+cos(phi_pitch)*sin(beta))*sin(delta_lf)-
cos(phi_pitch)*a_wb)*cos(phi_yaw)+r*sin(phi_yaw)*sin(phi_roll)*(cos(bet
```

```
a)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*cos(-
theta_wbl+beta))*thetadot_wbl
+(((((cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_lf)+sin(delta_lf)*(cos(phi_roll)*si
n(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta)))*cos(phi_yaw)+sin(phi_
yaw)*sin(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*c
os(-theta_wbl+beta)-sin(-
theta_wbl+beta)*(((cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch
)*sin(beta))*cos(delta_lf)-
sin(delta_lf)*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta)))*cos(phi_yaw)+sin(phi_yaw)*sin(phi_roll)*(cos
(beta)*cos(delta_lf)-sin(beta)*sin(delta_lf))))*r)*deltadot_lf -
Xdot_lf;
V(20,1)=(0)*Xdot_trans + (1)*Ydot_trans + (0)*Zdot_trans + (((-
cos(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*sin(phi_pitch)-
r*cos(phi_pitch)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*cos
(phi_yaw)-sin(phi_yaw)*sin(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*cos(-theta_wbl+beta)+((-
r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))*sin(p
hi_pitch)+cos(phi_pitch)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*cos(phi_yaw)-
r*sin(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(del
ta_lf)))*sin(-
theta_wbl+beta)+(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*d_comwb)*phidot_yaw + (-
((cos(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*cos(phi_pitch)-
r*sin(phi_pitch)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf)))*cos
(-
theta_wbl+beta)+(r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos
(delta_lf))*cos(phi_pitch)+sin(phi_pitch)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*sin(-theta_wbl+beta)-
cos(phi_pitch)*sin(phi_roll)*d_comwb)*sin(phi_yaw))*phidot_pitch +
((sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))
*(r*cos(beta)*cos(delta_lf)-r*sin(beta)*sin(delta_lf)+a_wb)*cos(-
theta_wbl+beta)+r*(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_ya
w)*cos(phi_roll))*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))*sin
(-theta_wbl+beta)-d_comwb*(-
sin(phi_yaw)*sin(phi_pitch)*cos(phi_roll)+cos(phi_yaw)*sin(phi_roll)))*
phidot_roll + (((-
r*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*cos
(delta_lf)+r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*sin(delta_lf)-
cos(phi_roll)*sin(phi_pitch)*a_wb)*sin(phi_yaw)-
cos(phi_yaw)*sin(phi_roll)*(r*sin(beta)*sin(delta_lf)-
r*cos(beta)*cos(delta_lf)-a_wb))*sin(-theta_wbl+beta)-cos(-
theta_wbl+beta)*((-r*(cos(phi_roll)*sin(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta))*cos(delta_lf)-
r*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*sin(beta))*sin
(delta_lf)+cos(phi_pitch)*a_wb)*sin(phi_yaw)+r*cos(phi_yaw)*sin(phi_rol
l)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))))*thetadot_wbl +(-
r*((((-
cos(phi_roll)*sin(phi_pitch)*sin(beta)+cos(phi_pitch)*cos(beta))*cos(de
lta_lf)-
```

```
sin(delta_lf)*(cos(phi_roll)*sin(phi_pitch)*cos(beta)+cos(phi_pitch)*si
n(beta)))*sin(phi_yaw)+cos(phi_yaw)*sin(phi_roll)*(cos(beta)*sin(delta_
lf)+sin(beta)*cos(delta_lf)))*cos(-theta_wbl+beta)-sin(-
theta_wbl+beta)*(((-cos(phi_roll)*sin(phi_pitch)*cos(beta)-
cos(phi_pitch)*sin(beta))*cos(delta_lf)+sin(delta_lf)*(cos(phi_roll)*si
n(phi_pitch)*sin(beta)-
cos(phi_pitch)*cos(beta)))*sin(phi_yaw)+cos(phi_yaw)*sin(phi_roll)*(cos
(beta)*cos(delta_lf)-sin(beta)*sin(delta_lf)))))*deltadot_lf - Ydot_lf;
V(21,1)=(0)*Xdot_trans + (0)*Ydot_trans + (1)*Zdot_trans +
(0)*phidot_yaw + ((-cos(phi_roll)*(r*sin(beta)*sin(delta_lf)-
r*cos(beta)*cos(delta_lf)-
a_wb)*sin(phi_pitch)+r*cos(phi_pitch)*(cos(beta)*sin(delta_lf)+sin(beta
)*cos(delta_lf)))*cos(-
theta_wbl+beta)+(r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos
(delta_lf))*sin(phi_pitch)+cos(phi_pitch)*(r*sin(beta)*sin(delta_lf)-
r*cos(beta)*cos(delta_lf)-a_wb))*sin(-theta_wbl+beta)-
sin(phi_pitch)*sin(phi_roll)*d_comwb)*phidot_pitch + ((-
sin(phi_roll)*(r*sin(beta)*sin(delta_lf)-r*cos(beta)*cos(delta_lf)-
a_wb)*cos(-
theta_wbl+beta)+r*sin(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(
delta_lf))*sin(-
theta_wbl+beta)+cos(phi_roll)*d_comwb)*cos(phi_pitch))*phidot_roll +
((r*cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))*cos
(phi_pitch)+sin(phi_pitch)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb))*cos(-theta_wbl+beta)-sin(-
theta_wbl+beta)*(cos(phi_roll)*(r*cos(beta)*cos(delta_lf)-
r*sin(beta)*sin(delta_lf)+a_wb)*cos(phi_pitch)-
r*sin(phi_pitch)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))))*th
etadot_wbl
+(r*((cos(phi_roll)*(cos(beta)*sin(delta_lf)+sin(beta)*cos(delta_lf))*c
os(phi_pitch)-sin(phi_pitch)*(sin(beta)*sin(delta_lf)-
cos(beta)*cos(delta_lf)))*cos(-theta_wbl+beta)+sin(-
theta_wbl+beta)*(cos(phi_roll)*(sin(beta)*sin(delta_lf)-
cos(beta)*cos(delta_lf))*cos(phi_pitch)+sin(phi_pitch)*(cos(beta)*sin(d
elta_lf)+sin(beta)*cos(delta_lf)))))*deltadot_lf - Zdot_lf;

V(22,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans + (-
sin(phi_yaw))*phidot_yaw + (cos(phi_yaw)*cos(phi_pitch))*phidot_pitch +
(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*phidot_roll +
(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*thetadot_wbl
+(cos(phi_yaw)*sin(phi_pitch)*sin(phi_roll)-
sin(phi_yaw)*cos(phi_roll))*deltadot_lf - wX_lf;
V(23,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(cos(phi_yaw))*phidot_yaw + (sin(phi_yaw)*cos(phi_pitch))*phidot_pitch
+
(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))*
phidot_roll +
(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))*
thetadot_wbl
+(sin(phi_yaw)*sin(phi_pitch)*sin(phi_roll)+cos(phi_yaw)*cos(phi_roll))
*deltadot_lf - wY_lf;
V(24,1)=(0)*Xdot_trans + (0)*Ydot_trans + (0)*Zdot_trans +
(0)*phidot_yaw + (-sin(phi_pitch))*phidot_pitch +
```

```
(cos(phi_pitch)*sin(phi_roll))*phidot_roll +
(cos(phi_pitch)*sin(phi_roll))*thetadot_wbl
+(cos(phi_pitch)*sin(phi_roll))*deltadot_lf - wZ_lf;

End
```

C.6 Additional Results – 3D Position Kinematic Analysis

Case 2: Uphill Sloped Terrain

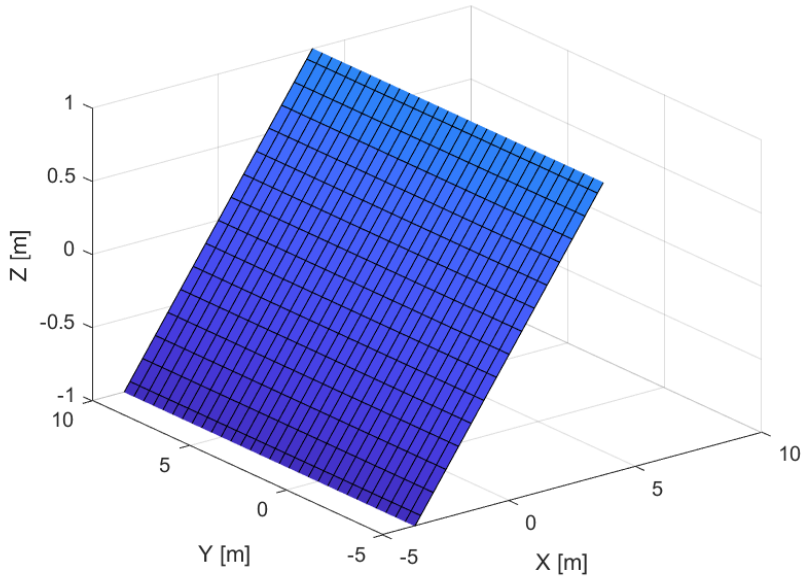


Figure C.1: Uphill 15º inclined terrain digital elevation map.

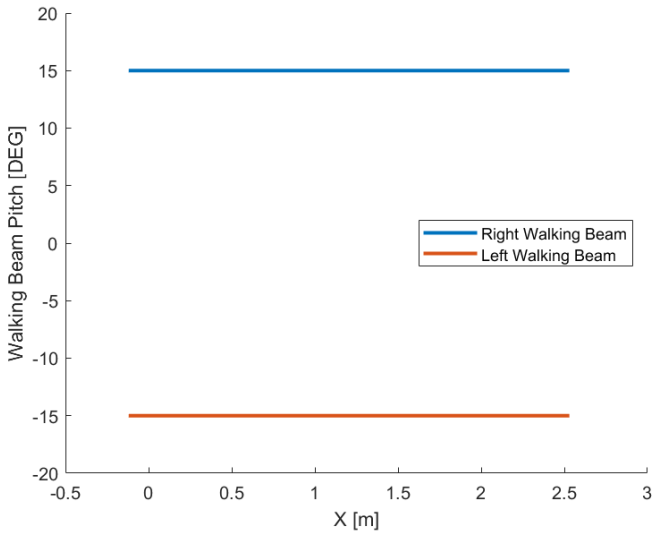


Figure C.2: Walking beam pitch vs distance travelled (15º Incline).

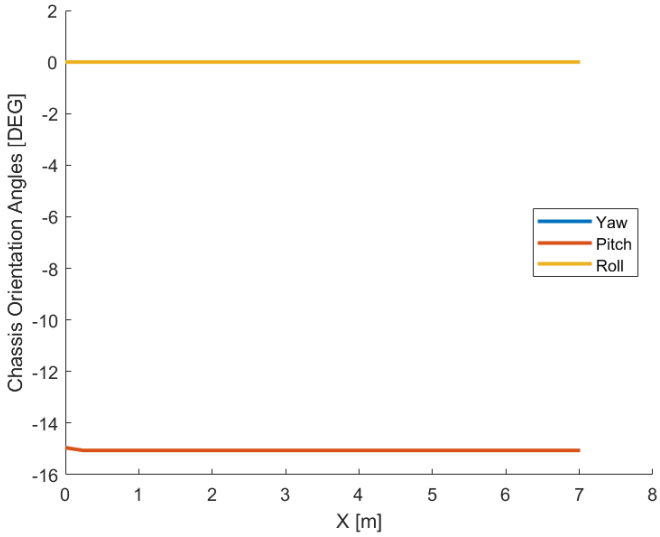


Figure C.3: Chassis orientation angles with respect to distance travelled (15º incline).

C.7 Additional Results – 3D Velocity Kinematic Analysis

Case 1: Flat Terrain

I. No-Slip

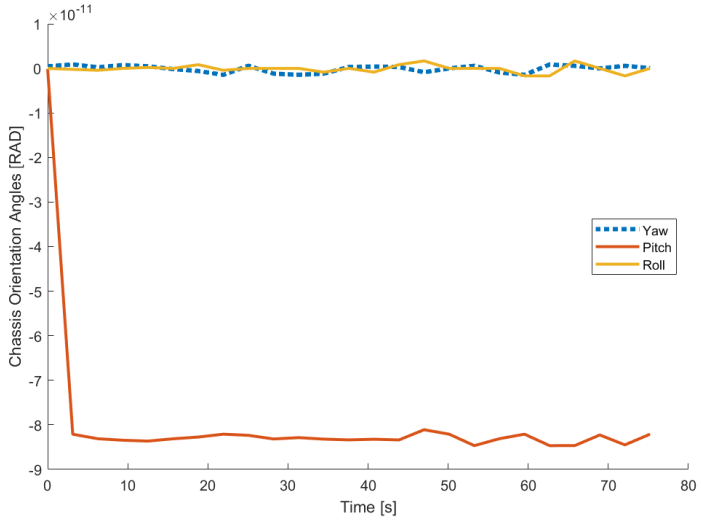


Figure C.4: Chassis angles vs time (flat terrain), for slip, i=0.

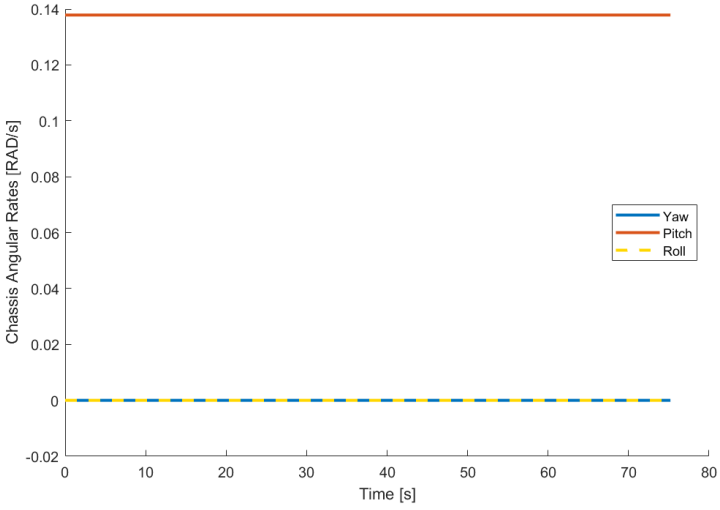


Figure C.5: Chassis angle rates vs time (flat terrain), for slip, i=0.

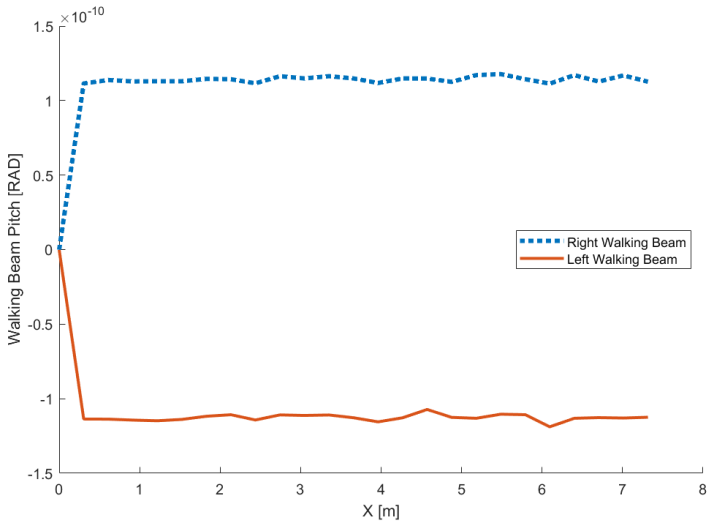


Figure C.6: Walking beam pitch vs distance (flat terrain), for slip, i=0.

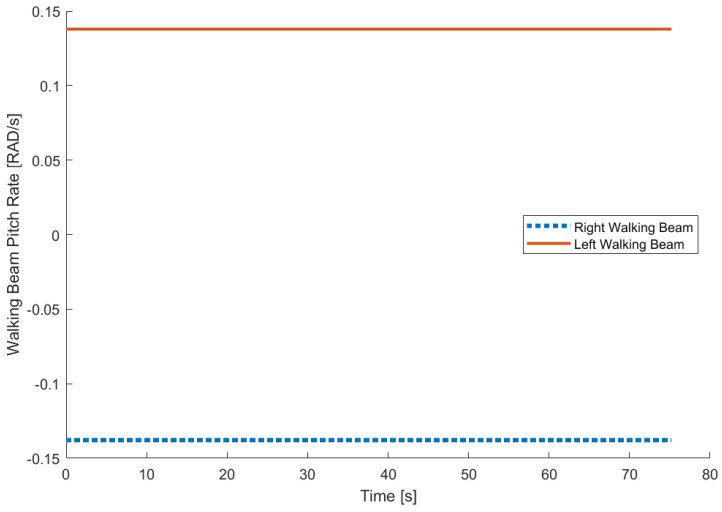


Figure C.7: Walking beam pitch rates vs time (flat terrain), for slip, i=0.

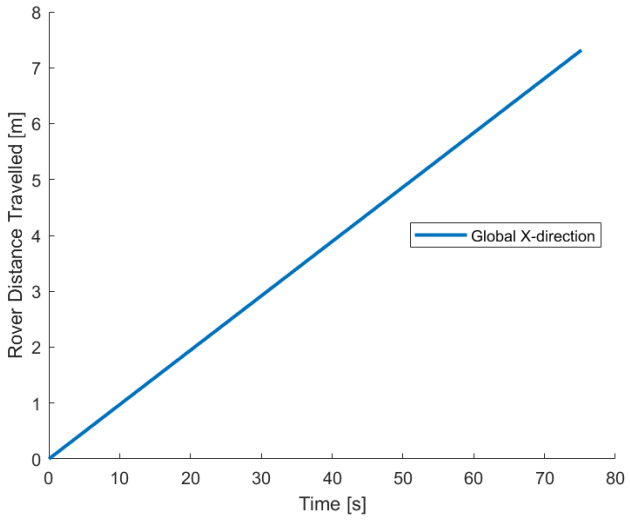


Figure C.8: Displacement vs time (flat terrain), for slip, i=0.

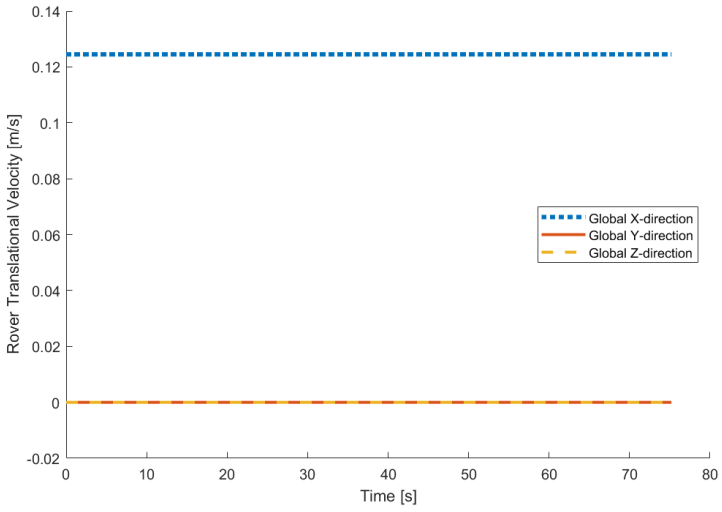


Figure C.9: Rover translational velocities vs time (flat terrain), for slip, i=0.

II. 0.05 Slip

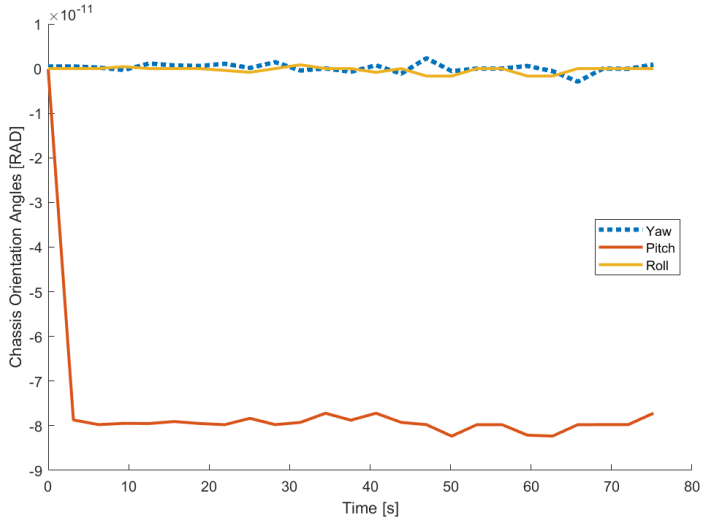


Figure C.10: Chassis angles vs time (flat terrain), for slip, i=0.05.

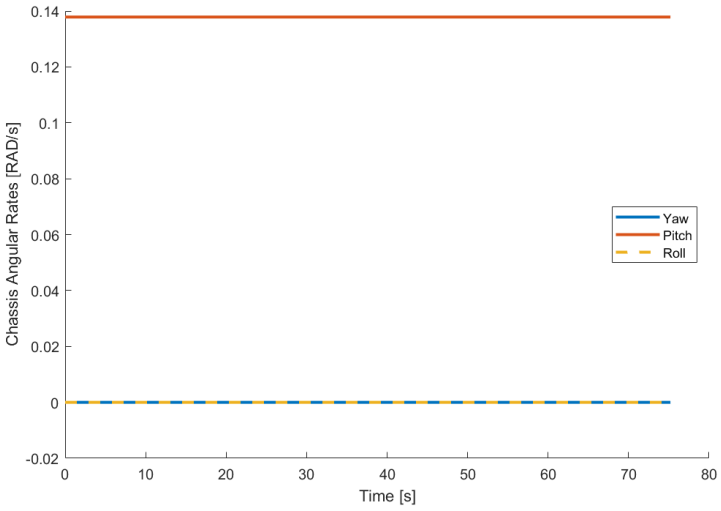


Figure C.11: Chassis angle rates vs time (flat terrain), for slip, i=0.05.

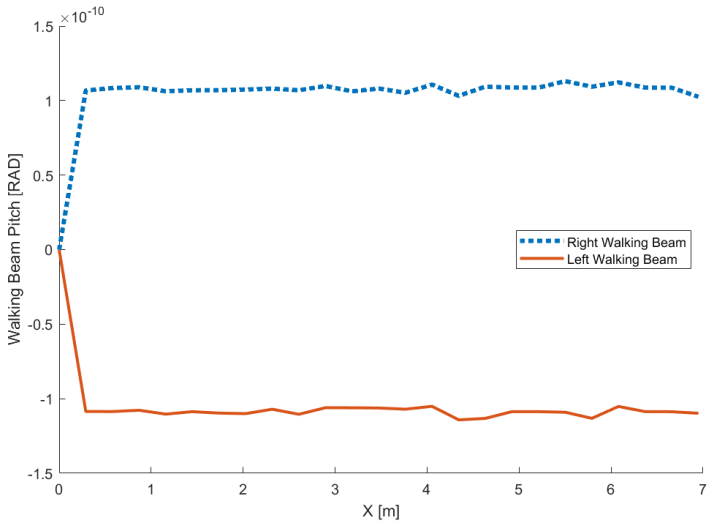


Figure C.12: Walking beam pitch vs distance (flat terrain), for slip, i=0.05.

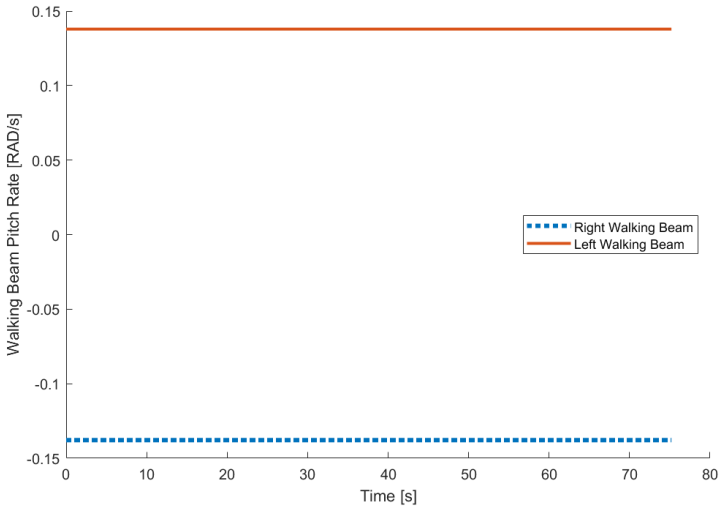


Figure C.13: Walking beam pitch rates vs time (flat terrain), for slip, i=0.05.

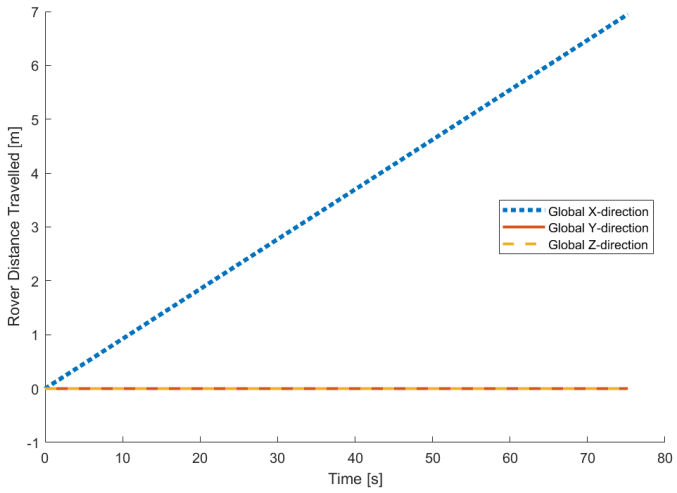


Figure C.14: Displacement vs time (flat terrain), for slip, i=0.05.

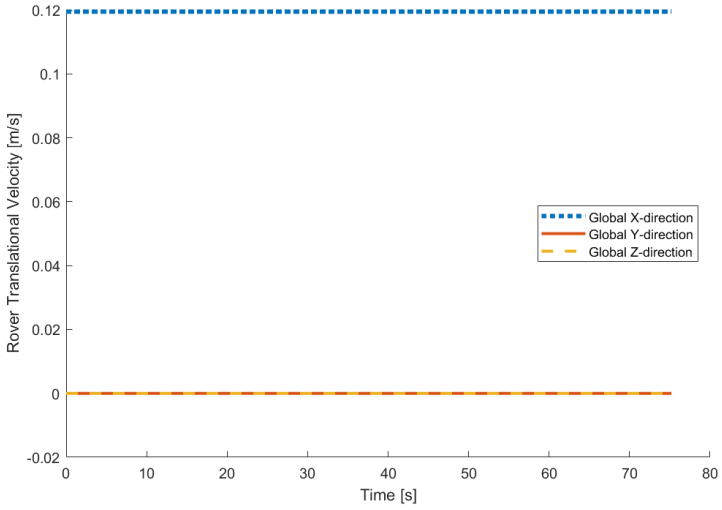


Figure C.15: Rover translational velocities vs time (flat terrain), for slip, i=0.05

III. 0.1 Slip

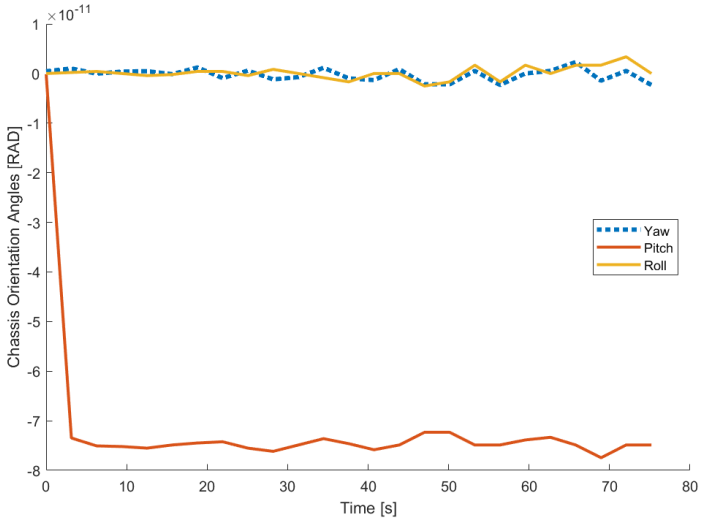


Figure C.16: Chassis angles vs time (flat terrain), for slip, i=0.1.

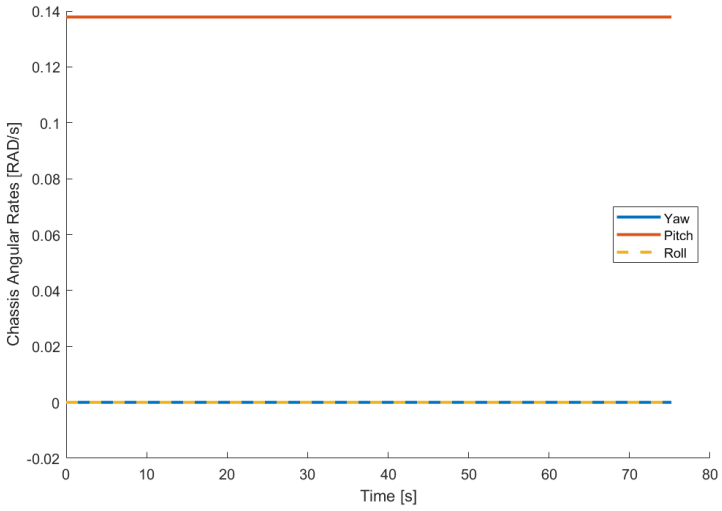


Figure C.17: Chassis angle rates vs time (flat terrain), for slip, i=0.1.

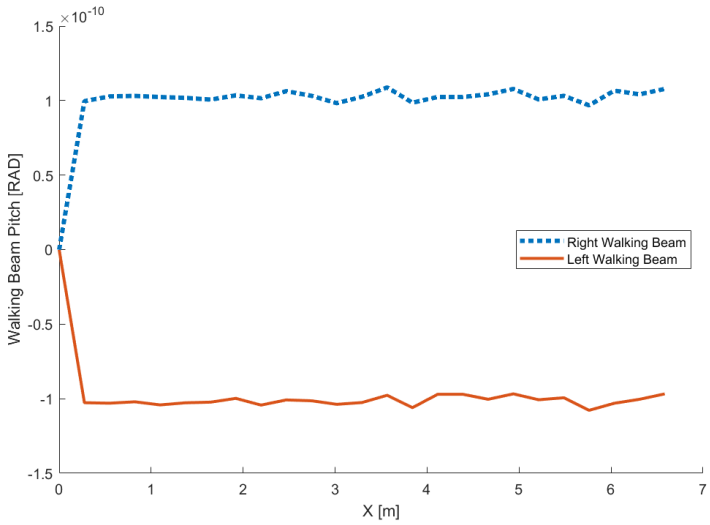


Figure C.18: Walking beam pitch vs distance (flat terrain), for slip, i=0.1.

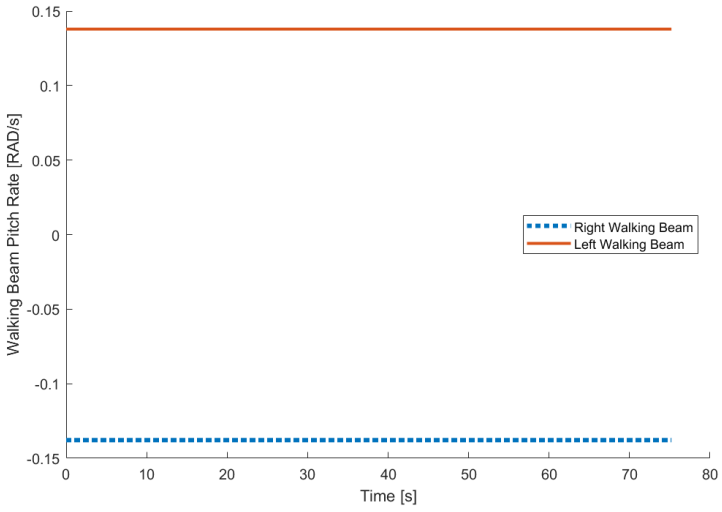


Figure C.19: Walking beam pitch rates vs time (flat terrain), for slip, i=0.1.

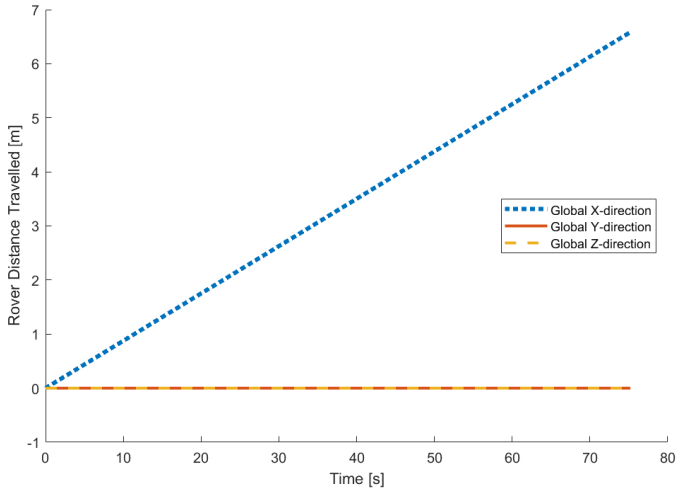


Figure C.20: Displacement vs time (flat terrain), for slip, i=0.1.

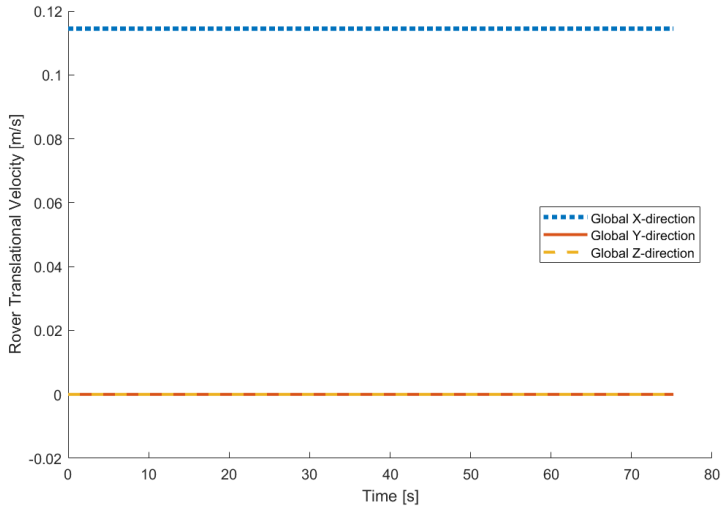


Figure C.21: Rover translational velocities vs time (flat terrain), for slip, i=0.1.

IV. **0.25 Slip**

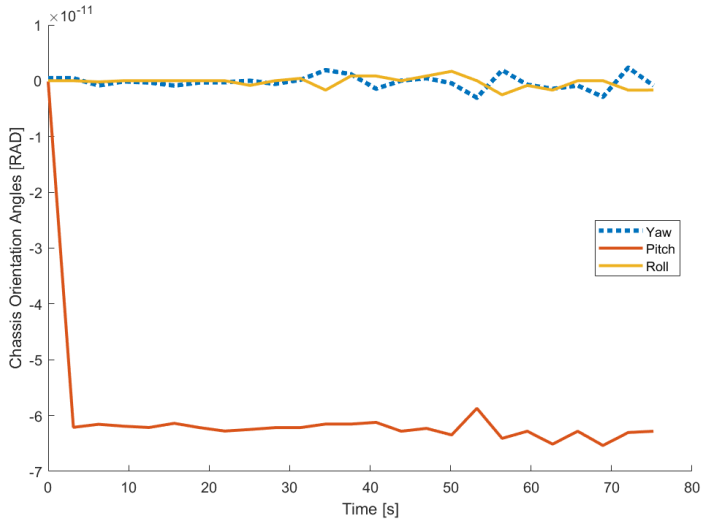


Figure C.22: Chassis angles vs time (flat terrain), for slip, i=0.25.

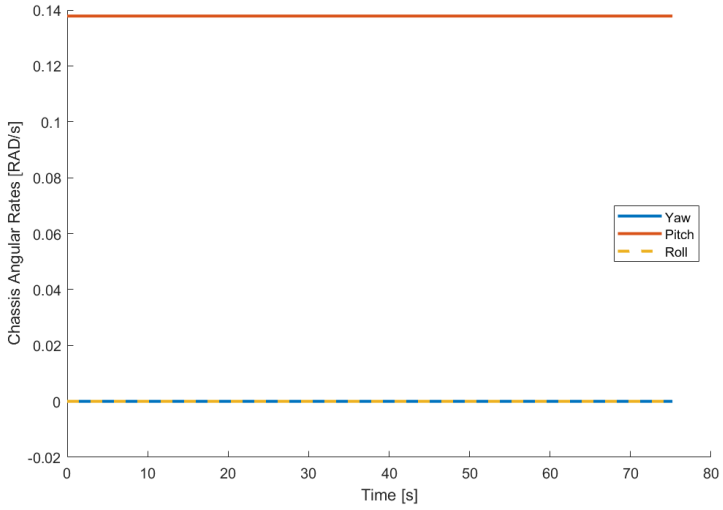


Figure C.23: Chassis angle rates vs time (flat terrain), for slip, i=0.25.

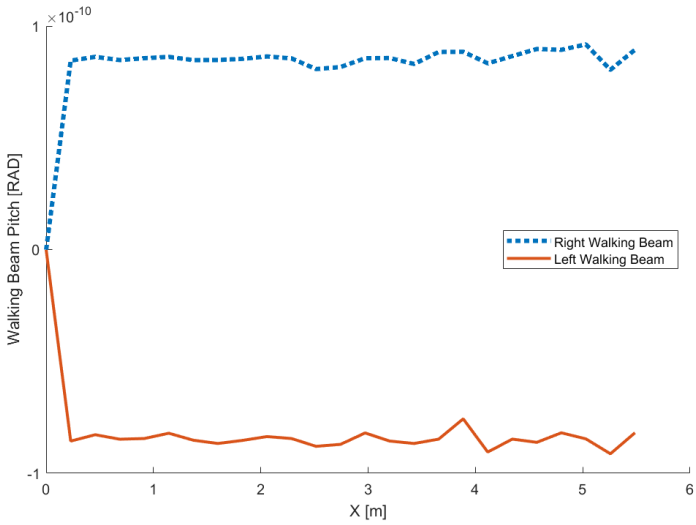


Figure C.24: Walking beam pitch vs distance (flat terrain), for slip, i=0.25.

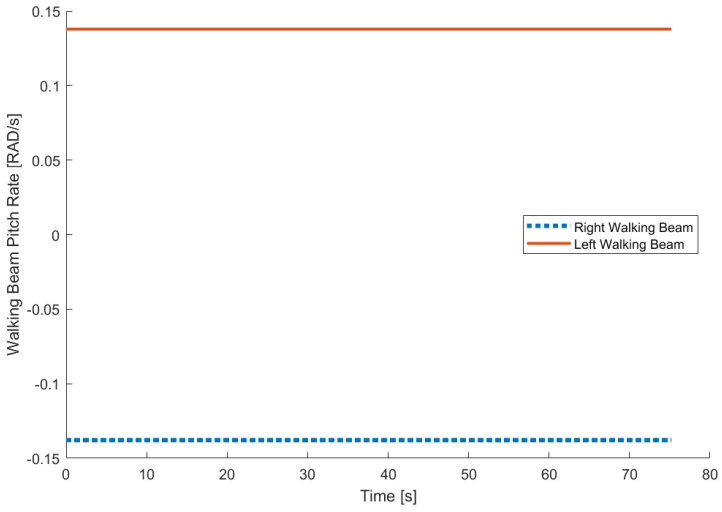


Figure C.25: Walking beam pitch rates vs time (flat terrain), for slip, i=0.25.

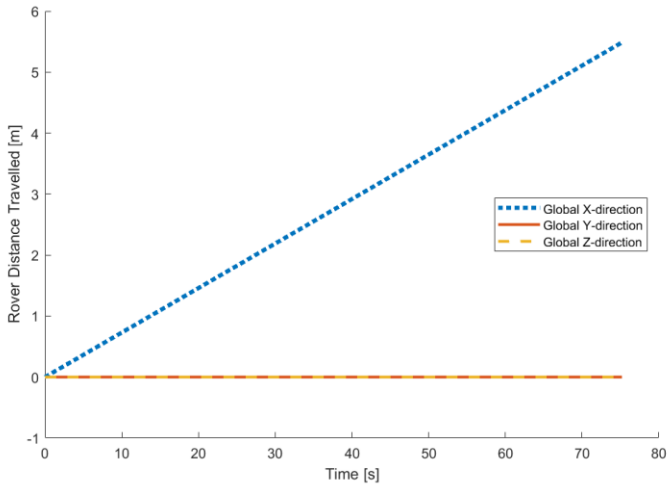


Figure C.26: Displacement vs time (flat terrain), for slip, i=0.25.

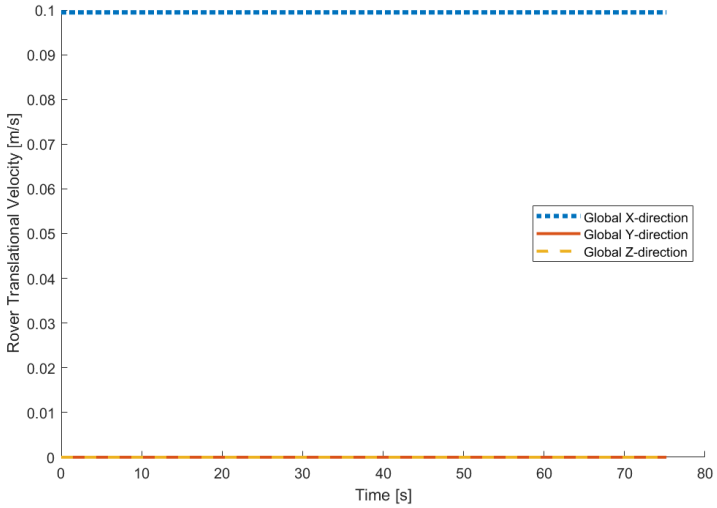


Figure C.27: Rover translational velocities vs time (flat terrain), for slip, i=0.25.

V. 0.5 Slip

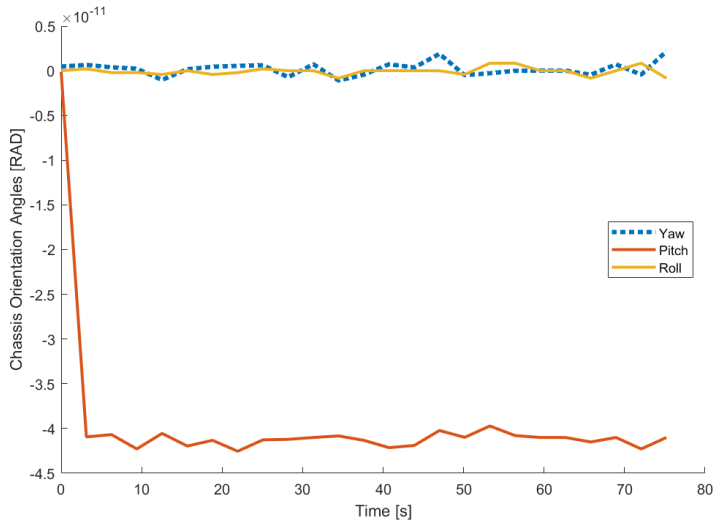


Figure C.28: Chassis angles vs time (flat terrain), for slip, i=0.5.

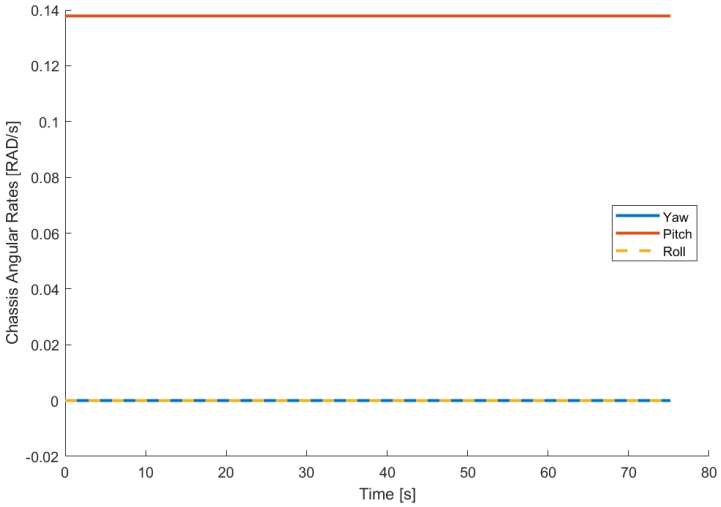


Figure C.29: Chassis angle rates vs time (flat terrain), for slip, i=0.5.

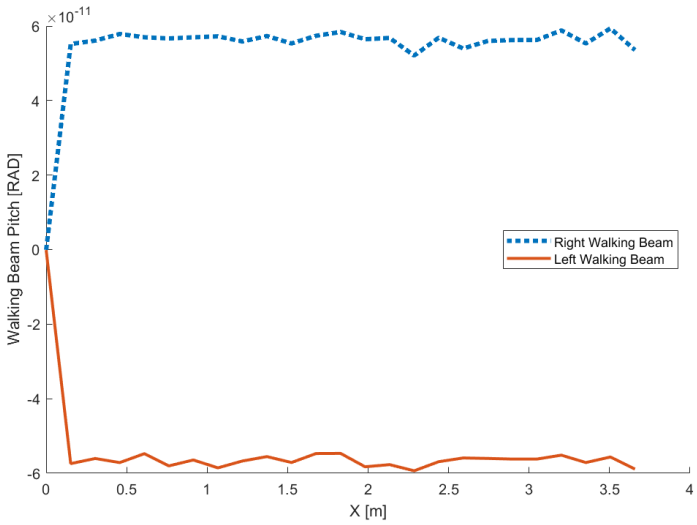


Figure C.30: Walking beam pitch vs distance (flat terrain), for slip, i=0.5.

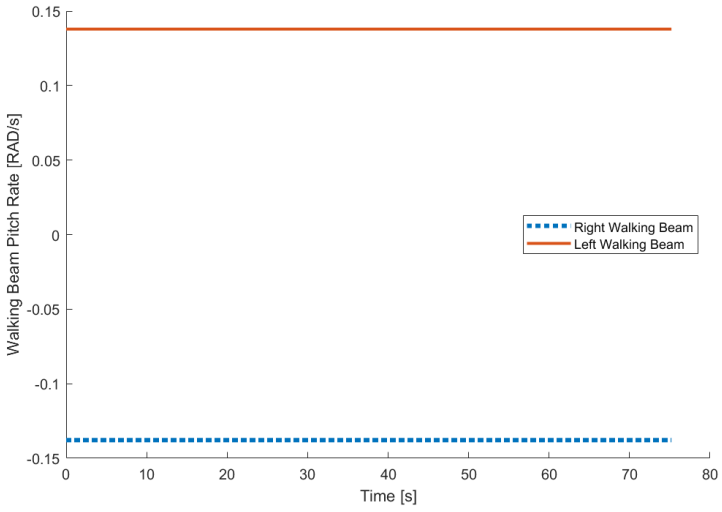


Figure C.31: Walking beam pitch rates vs time (flat terrain), for slip, i=0.5.

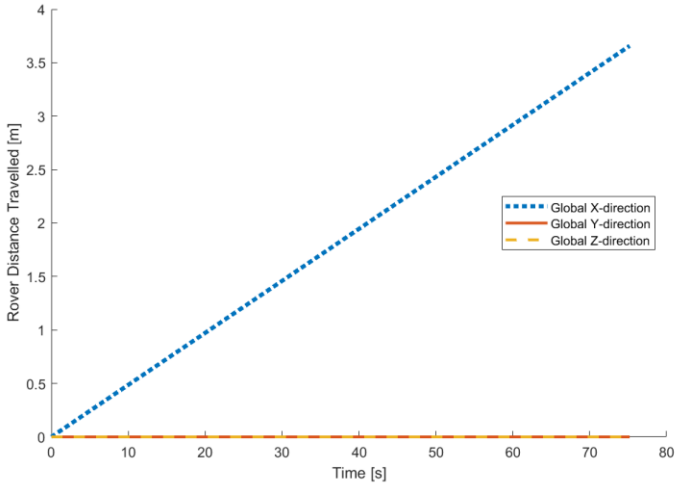


Figure C.32: Displacement vs time (flat terrain), for slip, i=0.5.

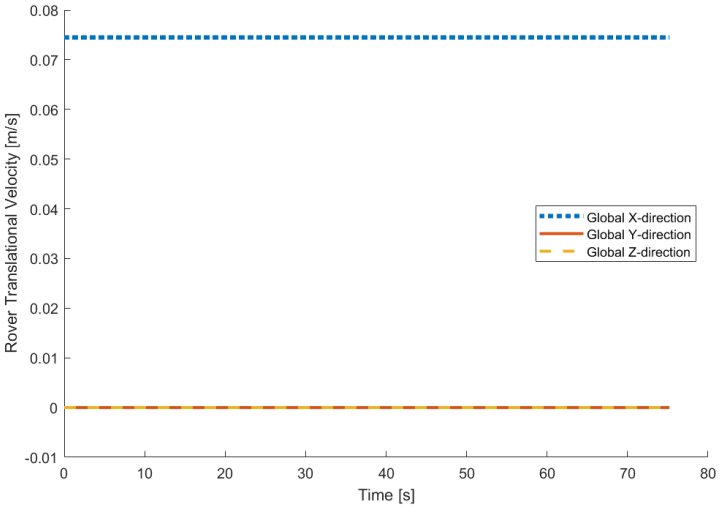


Figure C.33: Rover translational velocities vs time (flat terrain), for slip, i=0.5.

Case 2: Uphill Sloped Terrain (10º)

I. **No-Slip**

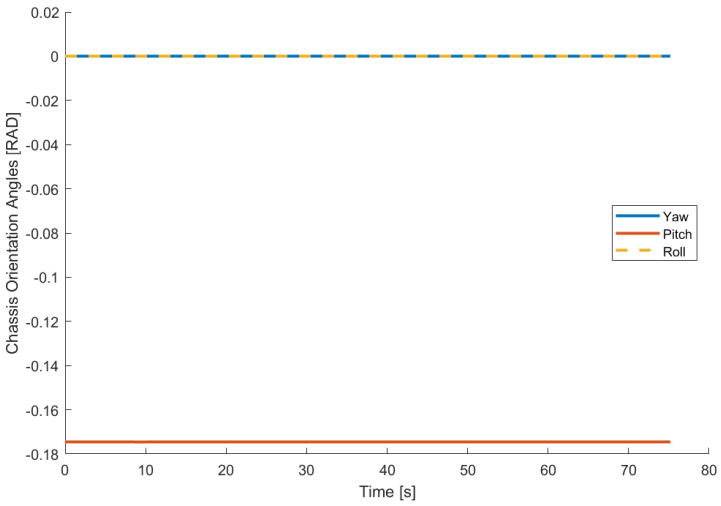


Figure C.34: Chassis angles vs time (10º incline), for slip, i=0.

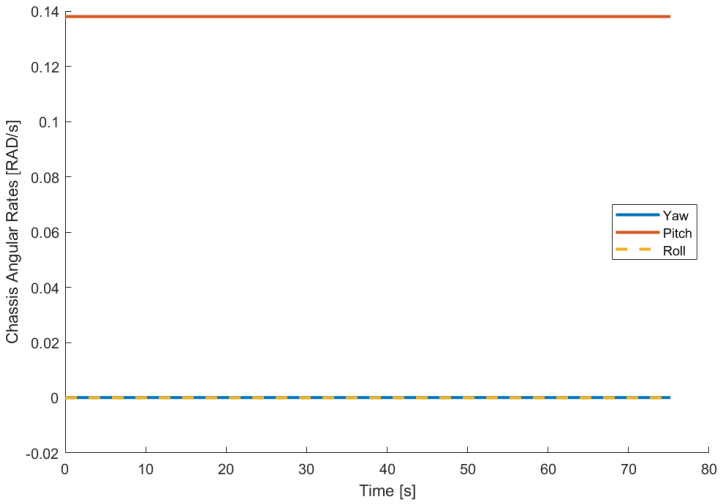


Figure C.35: Chassis angle rates vs time (10º incline), for slip, i=0.

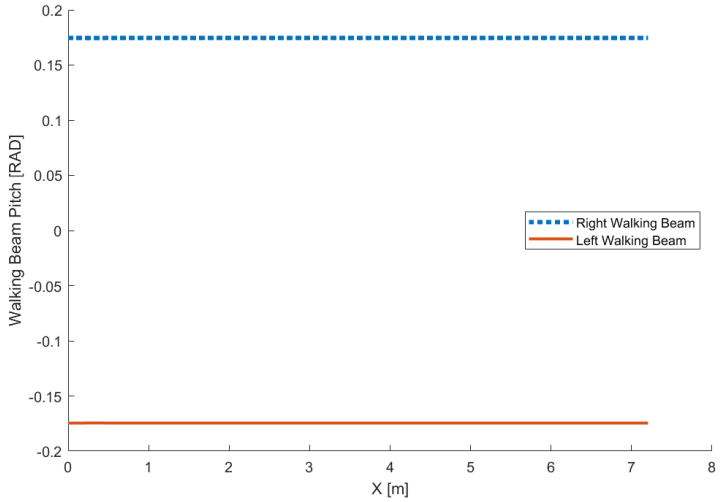


Figure C.36: Walking beam pitch vs distance (10º incline), for slip, i=0.

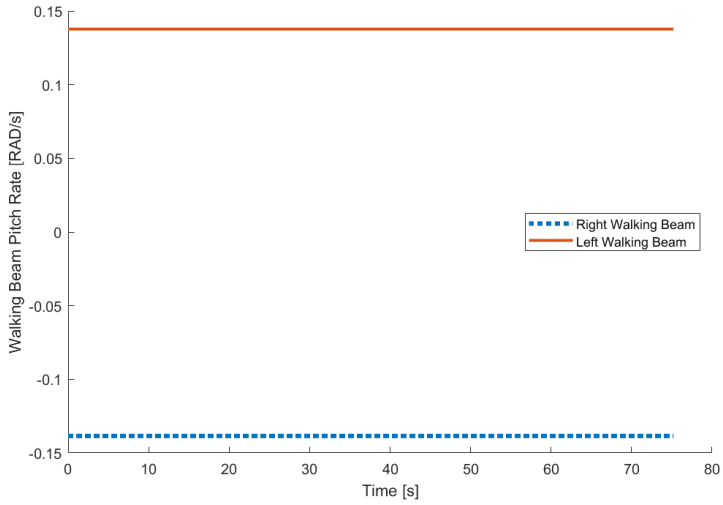


Figure C.37: Walking beam pitch rates vs time (10º incline), for slip, i=0.

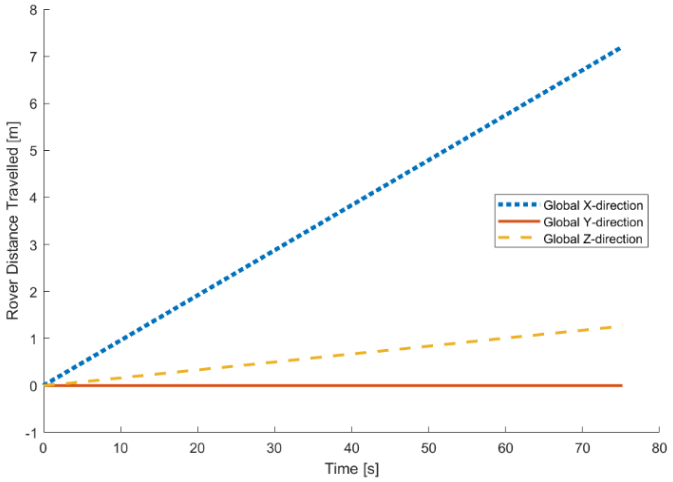


Figure C.38: Displacement vs time (10º incline), for slip, i=0.

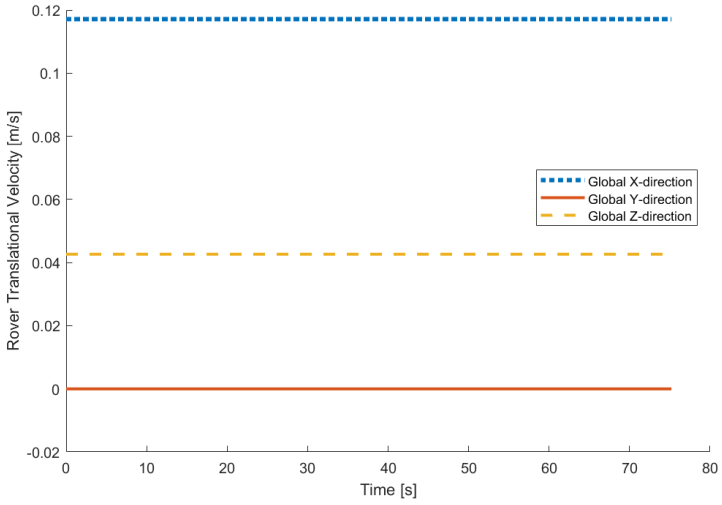


Figure C.39: Rover translational velocities vs time (10º incline), for slip, i=0.

II. 0.05 Slip

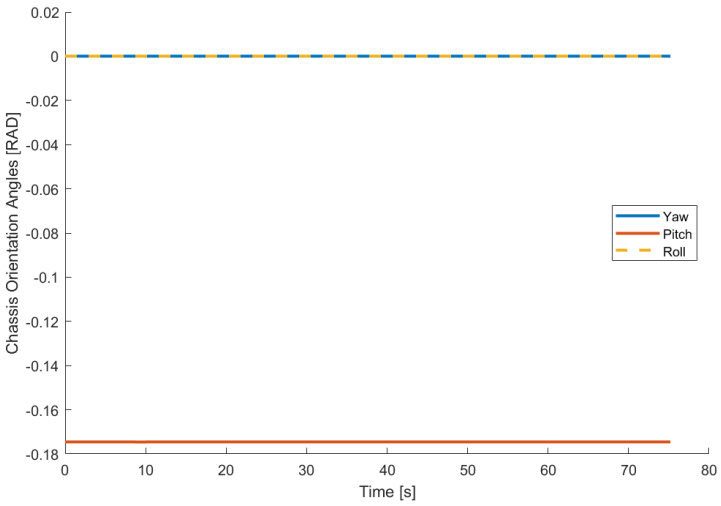


Figure C.40: Chassis angles vs time (10º incline), for slip, i=0.05.

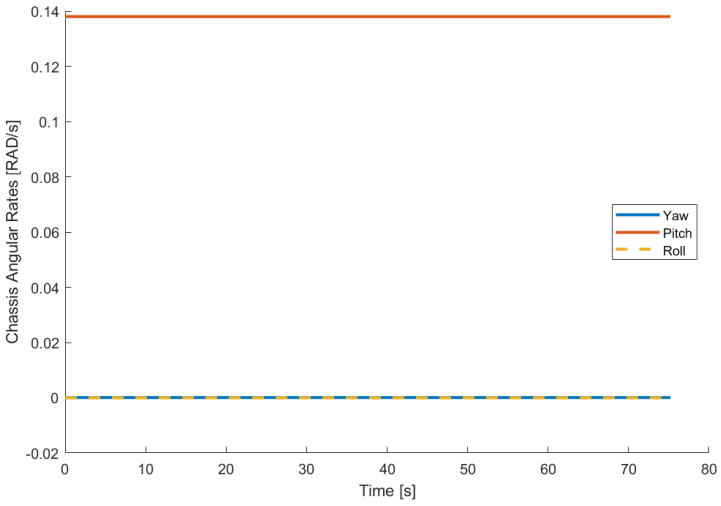


Figure C.41: Chassis angle rates vs time (10º incline), for slip, i=0.05.

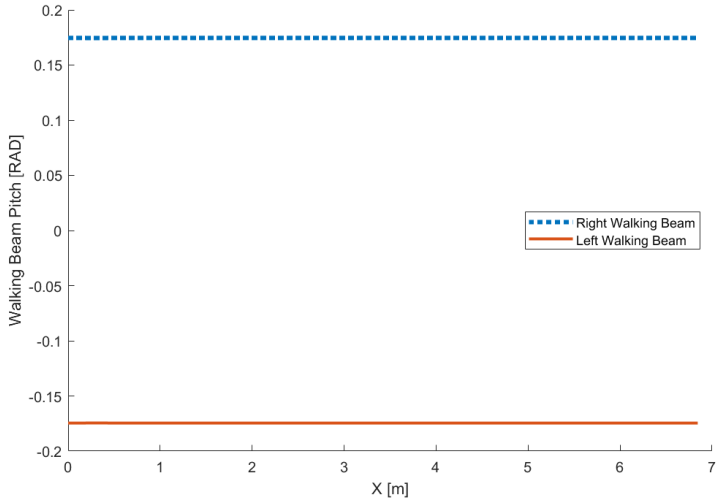


Figure C.42: Walking beam pitch vs distance (10º incline), for slip, i=0.05.

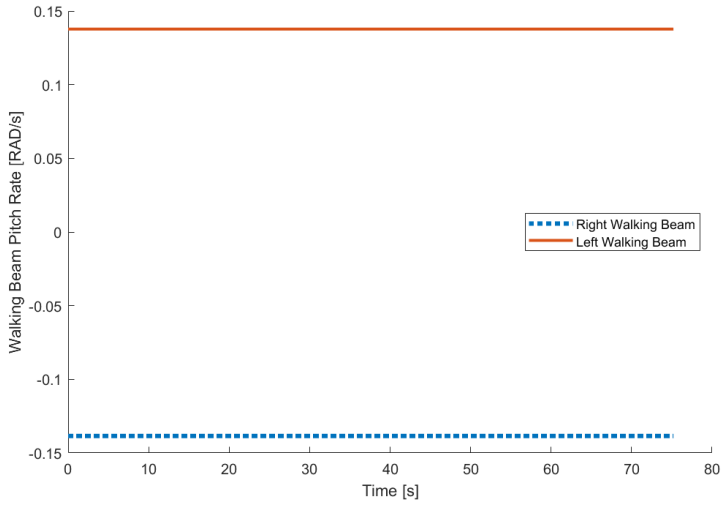


Figure C.43: Walking beam pitch rates vs time (10º incline), for slip, i=0.05.

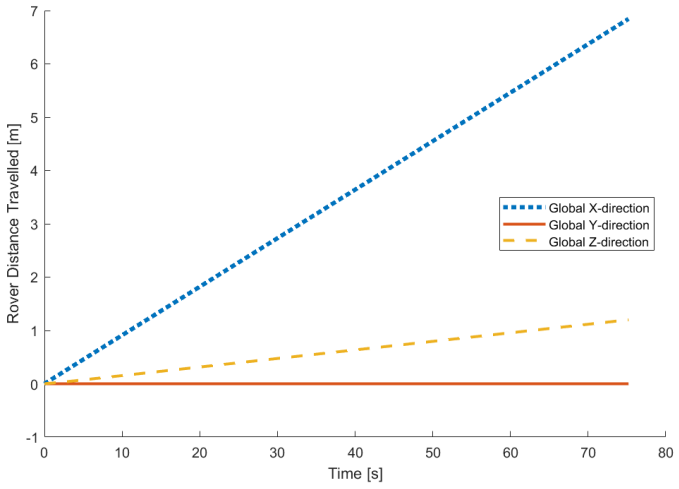


Figure C.44: Displacement vs time (10º incline), for slip, i=0.05.

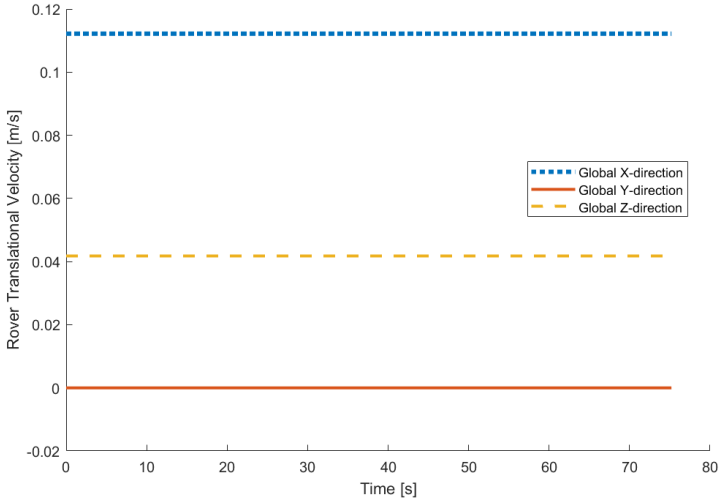


Figure C.45: Rover translational velocities vs time (10º incline), for slip, i=0.05

III. 0.1 Slip

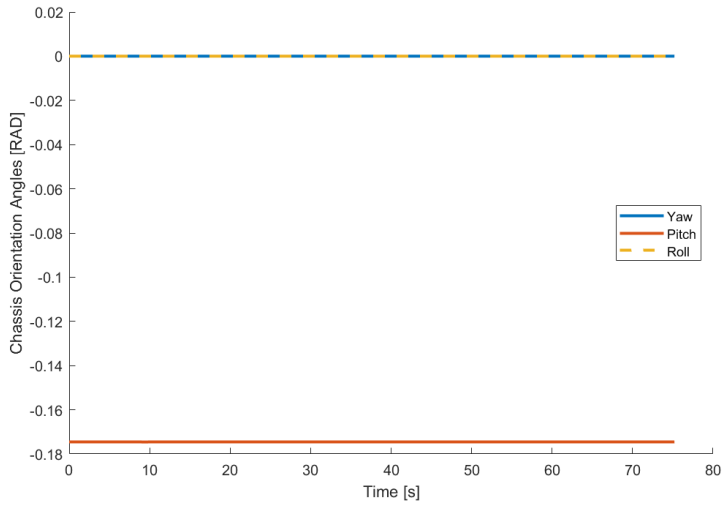


Figure C.46: Chassis angles vs time (10º incline), for slip, i=0.1.

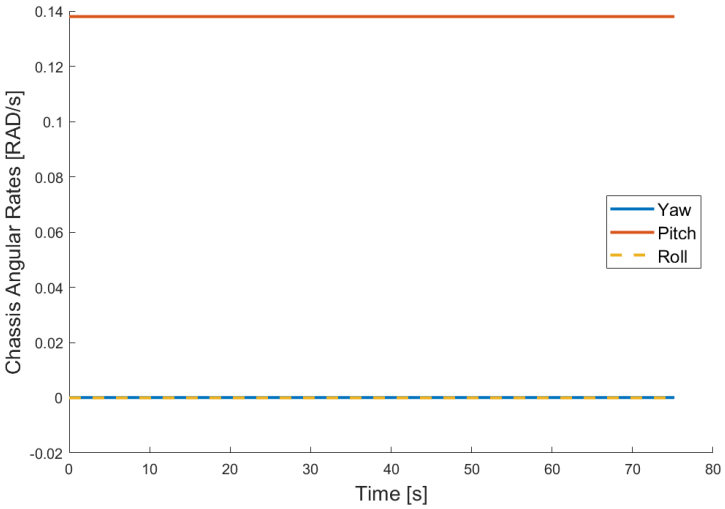


Figure C.47: Chassis angle rates vs time (10º incline), for slip, i=0.1.

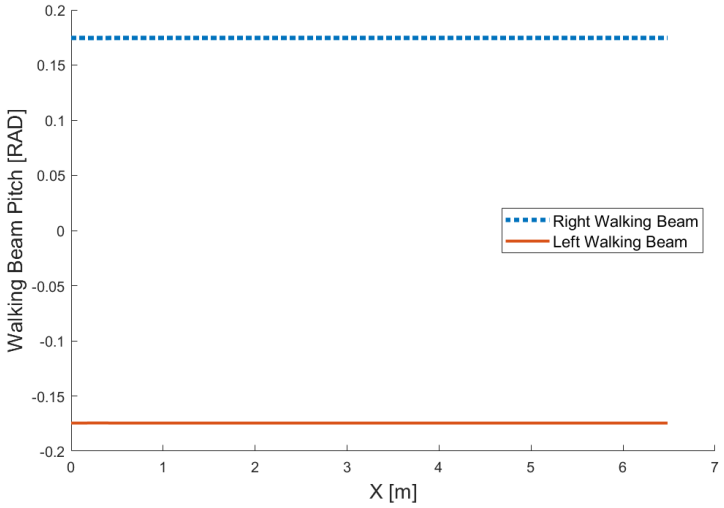


Figure C.48: Walking beam pitch vs distance (10º incline), for slip, i=0.1.

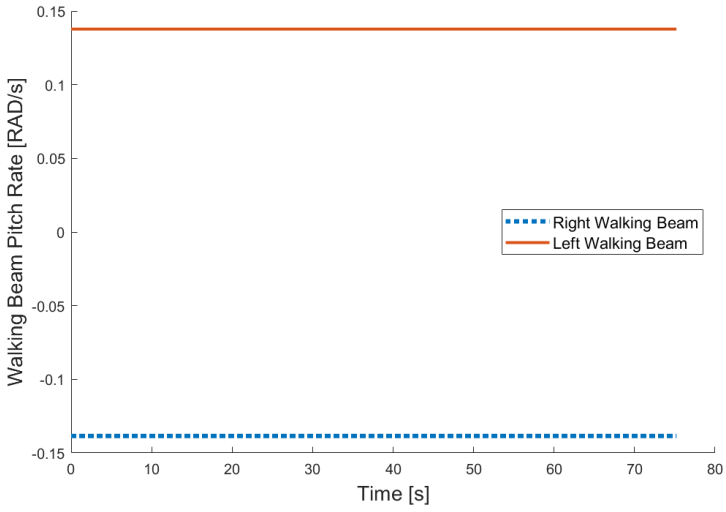


Figure C.49: Walking beam pitch rates vs time (10º incline), for slip, i=0.1.

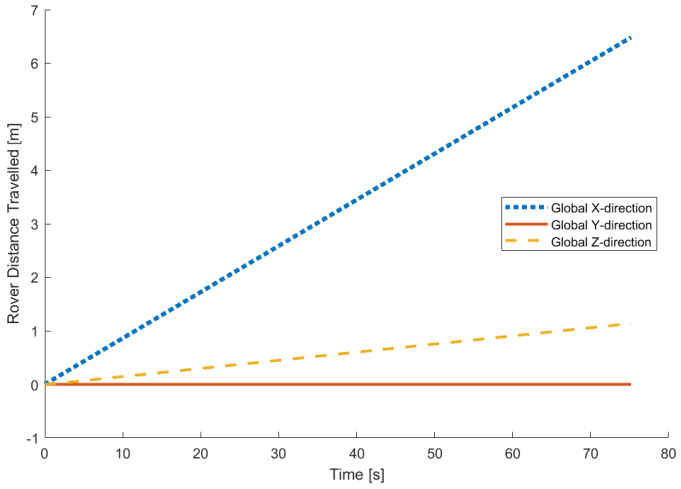


Figure C.50: Displacement vs time (10º incline), for slip, i=0.1.

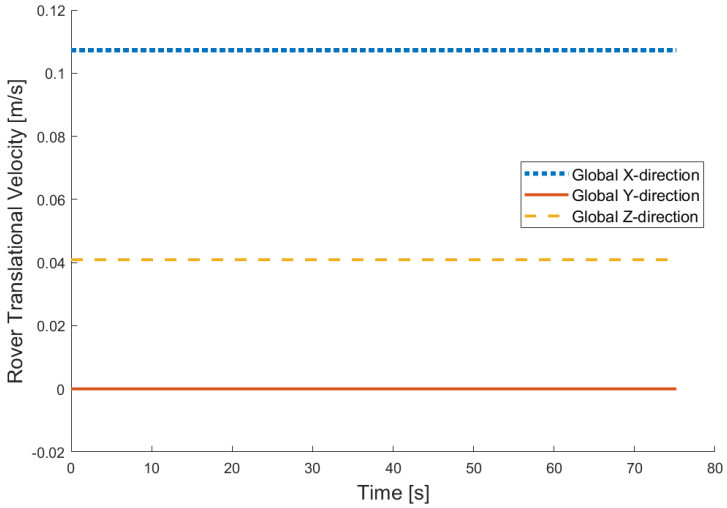


Figure C.51: Rover translational velocities vs time (10º incline), for slip, i=0.1.

IV. 0.25 Slip

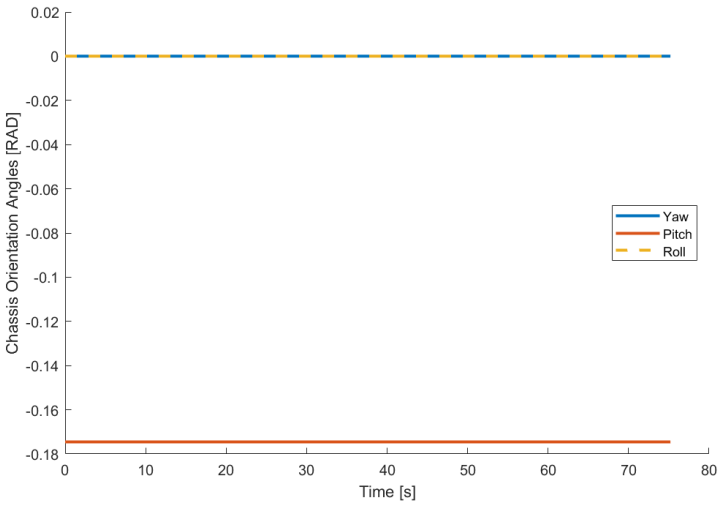


Figure C.52: Chassis angles vs time (10º incline), for slip, i=0.25.

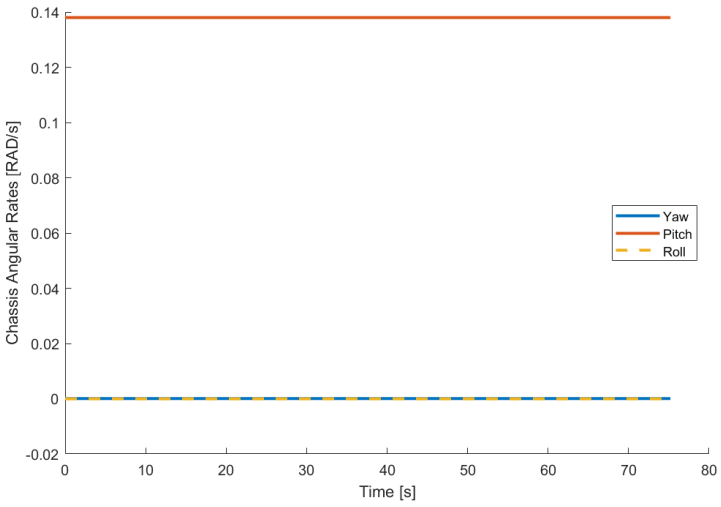


Figure C.53: Chassis angle rates vs time (10º incline), for slip, i=0.25.

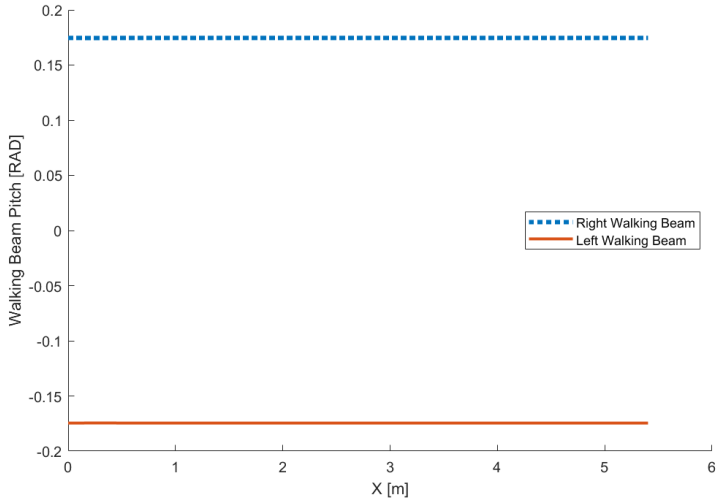


Figure C.54: Walking beam pitch vs distance (10º incline), for slip, i=0.25.

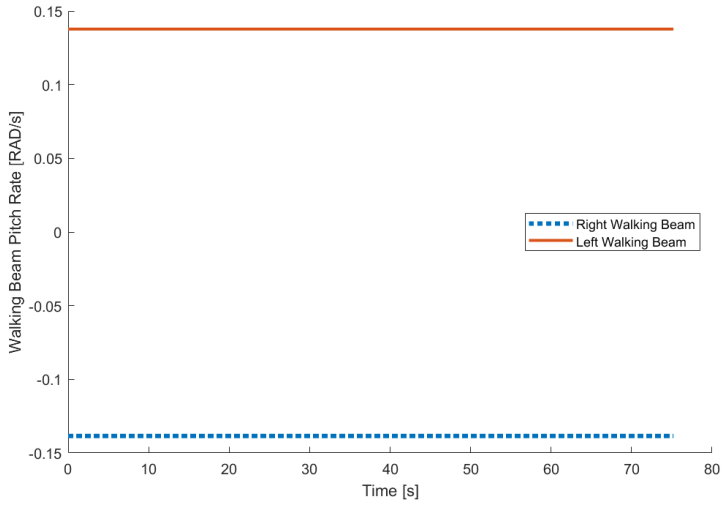


Figure C.55: Walking beam pitch rates vs time (10º incline), for slip, i=0.25.

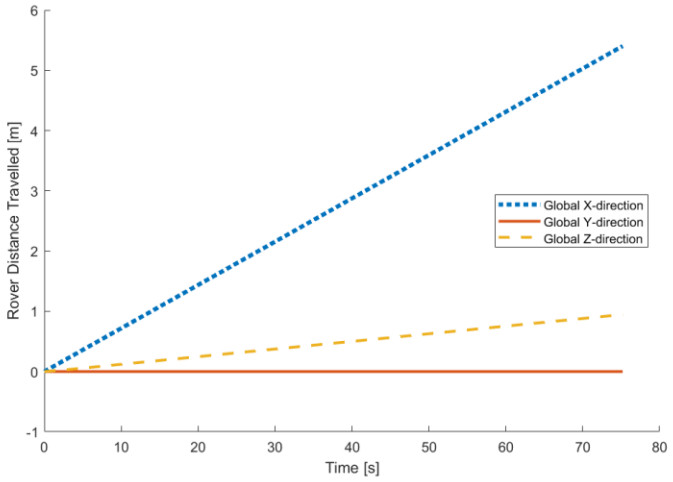


Figure C.56: Displacement vs time (10º incline), for slip, i=0.25.

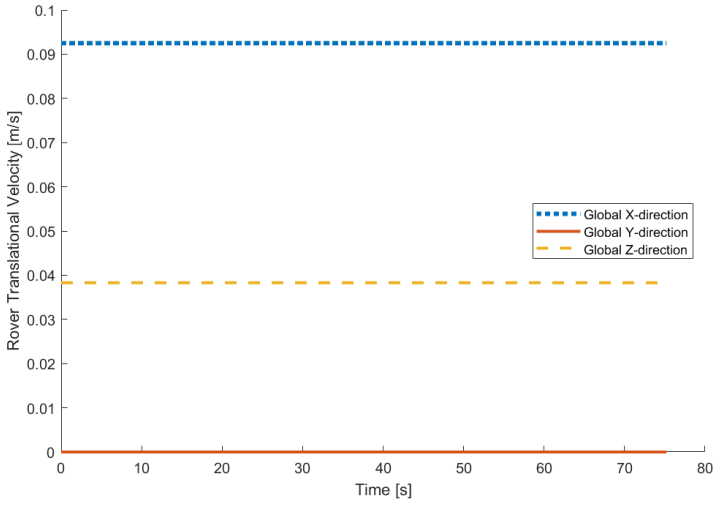


Figure C.57: Rover translational velocities vs time (10º incline), for slip, i=0.25.

V. 0.5 Slip

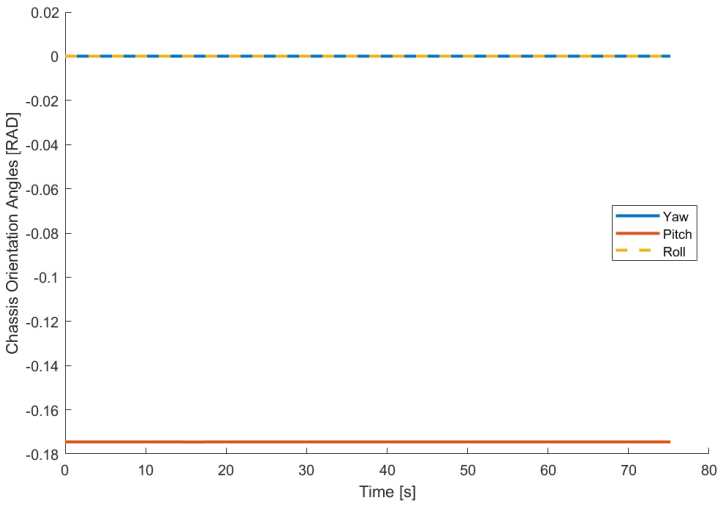


Figure C.58: Chassis angles vs time (10º incline), for slip, i=0.5.

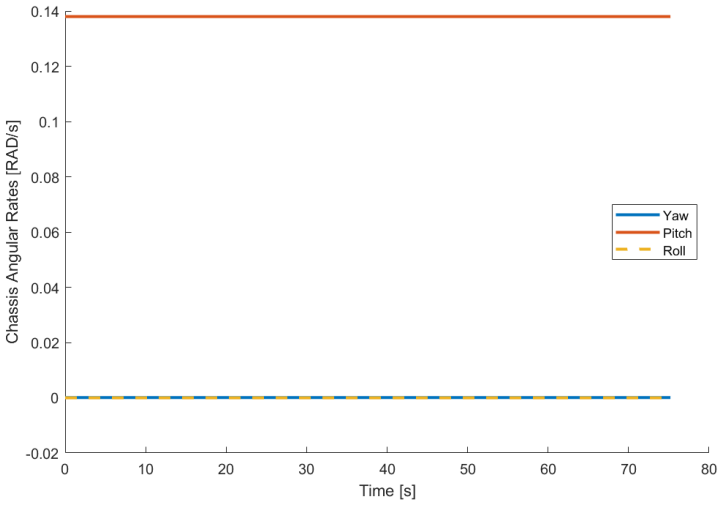


Figure C.59: Chassis angle rates vs time (10º incline), for slip, i=0.5.

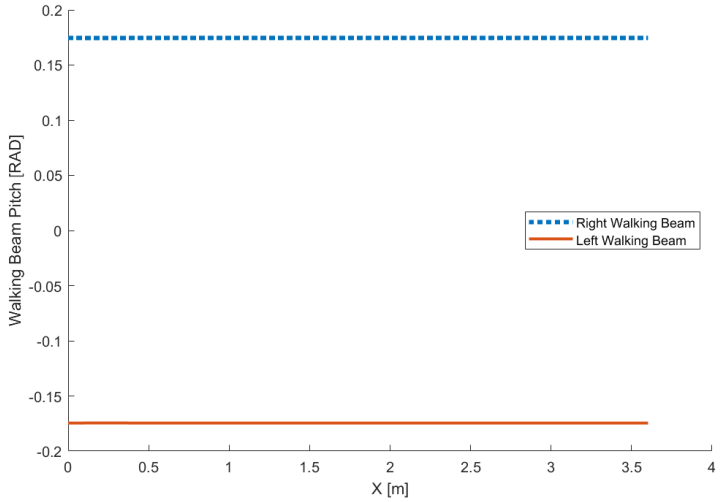


Figure C.60: Walking beam pitch vs distance (10º incline), for slip, i=0.5.

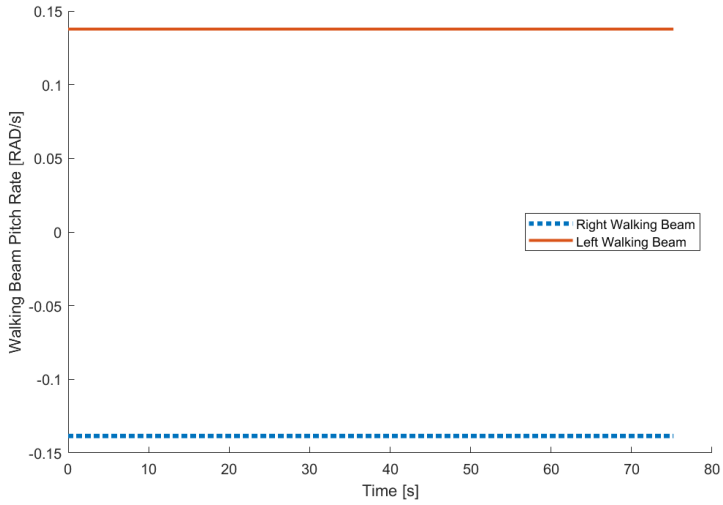


Figure C.61: Walking beam pitch rates vs time (10º incline), for slip, i=0.5.

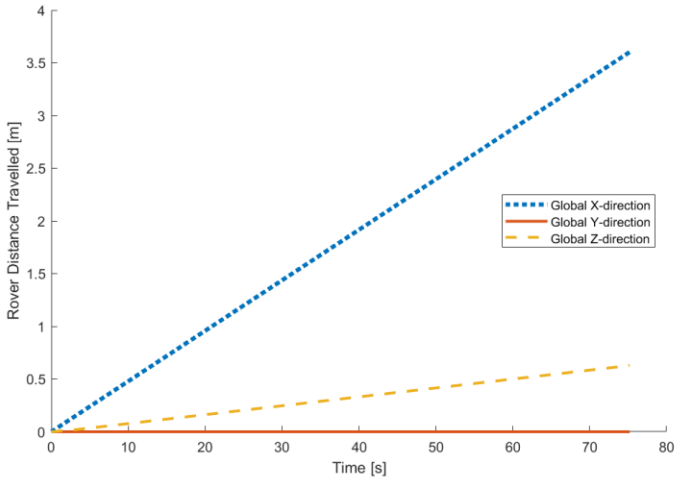


Figure C.62: Displacement vs time (10° incline), for slip, i=0.5.

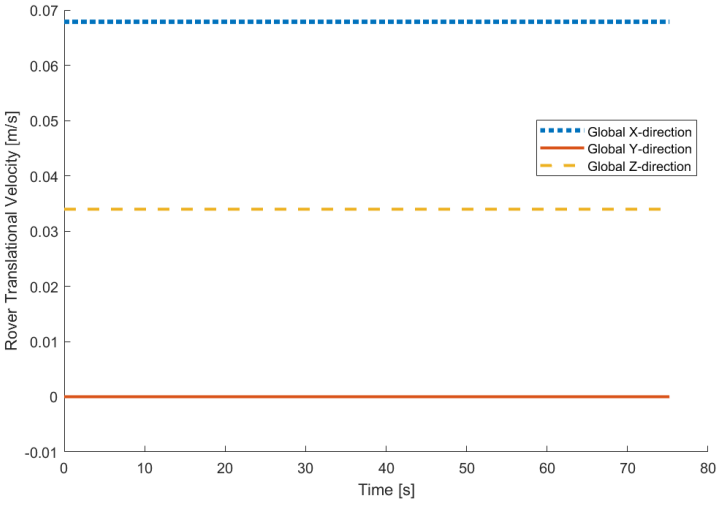


Figure C.63: Rover translational velocities vs time (10° incline), for slip, i=0.5.

Case 3: Side-slope Terrain (10º)

I. No-Slip

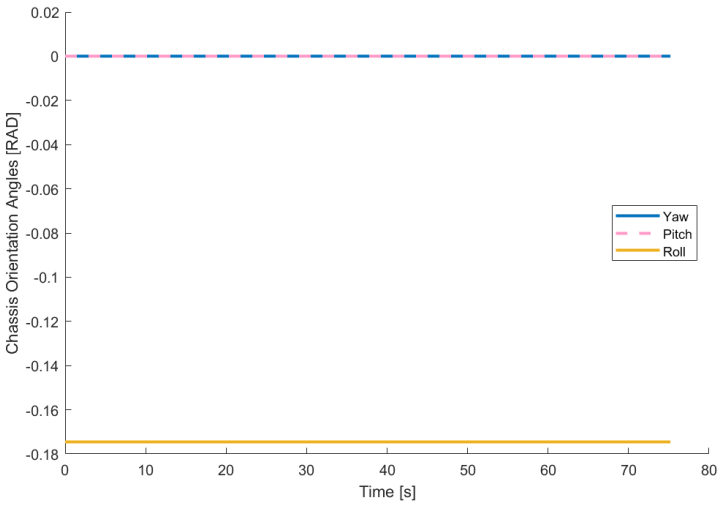


Figure C.64: Chassis angles vs time (side-slope 10º), for slip, i=0.

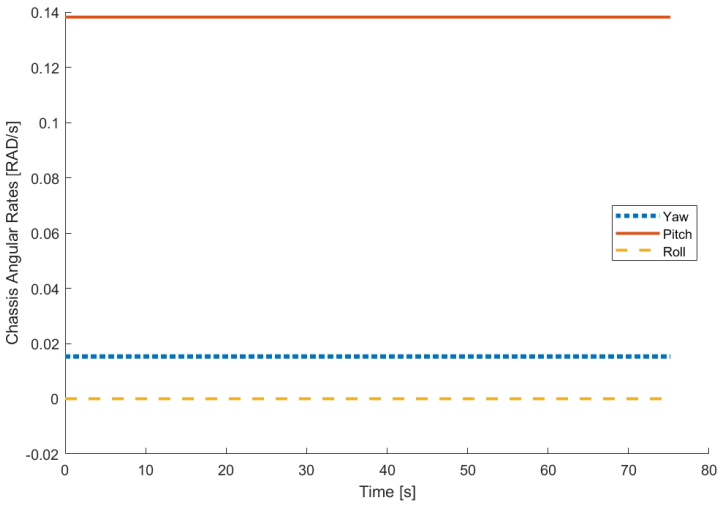


Figure C.65: Chassis angle rates vs time (side-slope 10º), for slip, i=0.

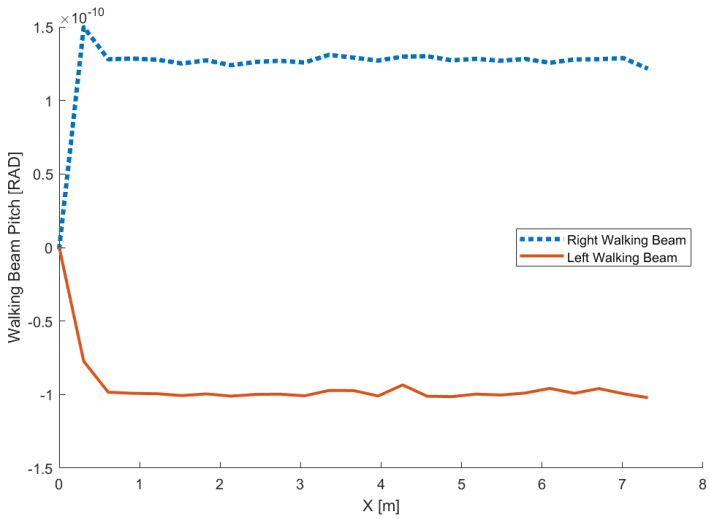


Figure C.66: Walking beam pitch vs distance (side-slope 10°), for slip, i=0.

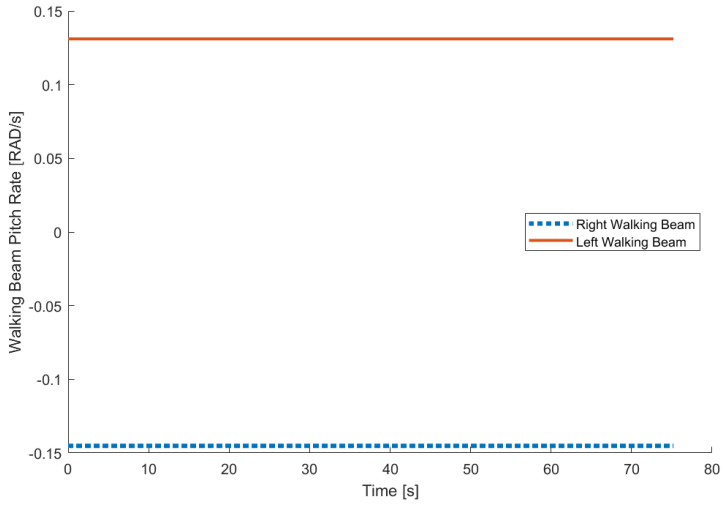


Figure C.67: Walking beam pitch rates vs time (side-slope 10°), for slip, i=0.

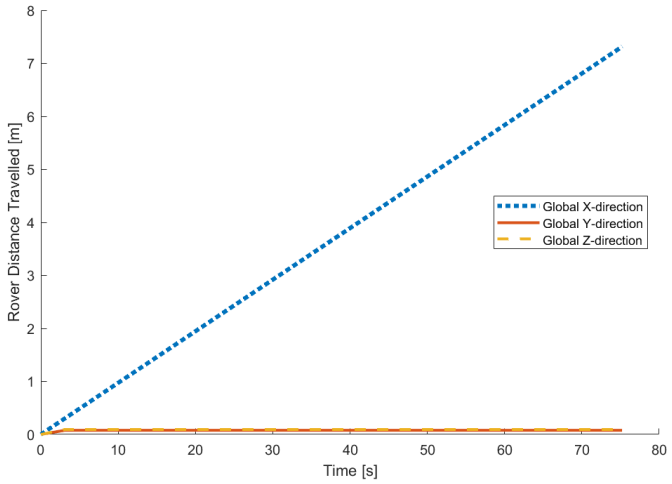


Figure C.68: Displacement vs time (side-slope 10º), for slip, i=0.

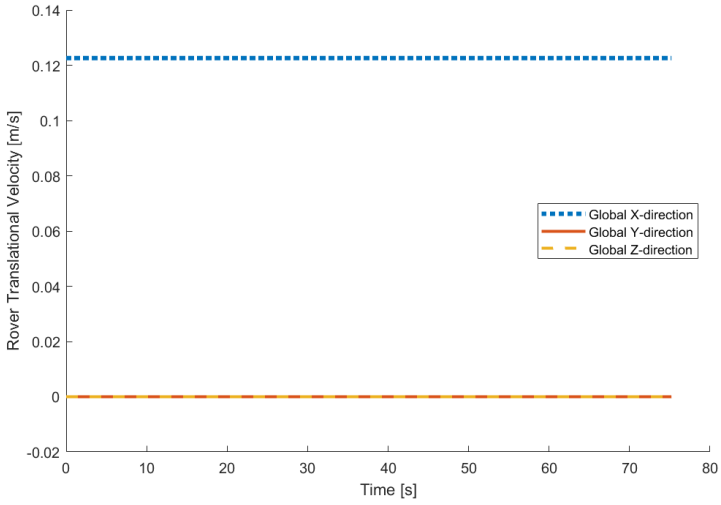


Figure C.69: Rover translational velocities vs time (side-slope 10º), for slip, i=0.

II. 0.05 Slip

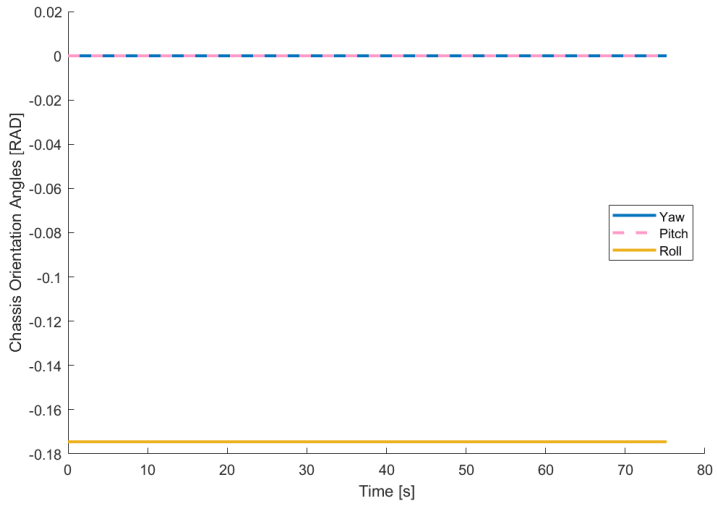


Figure C.70: Chassis angles vs time (side-slope 10º), for slip, i=0.05.

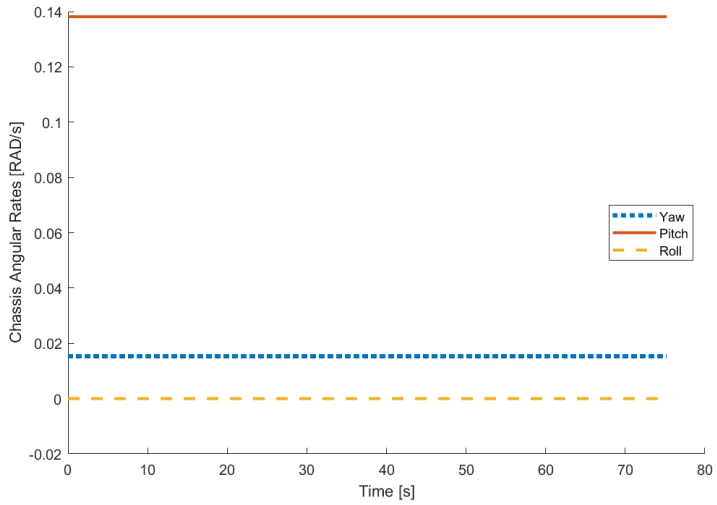


Figure C.71: Chassis angle rates vs time (side-slope 10º), for slip, i=0.05.

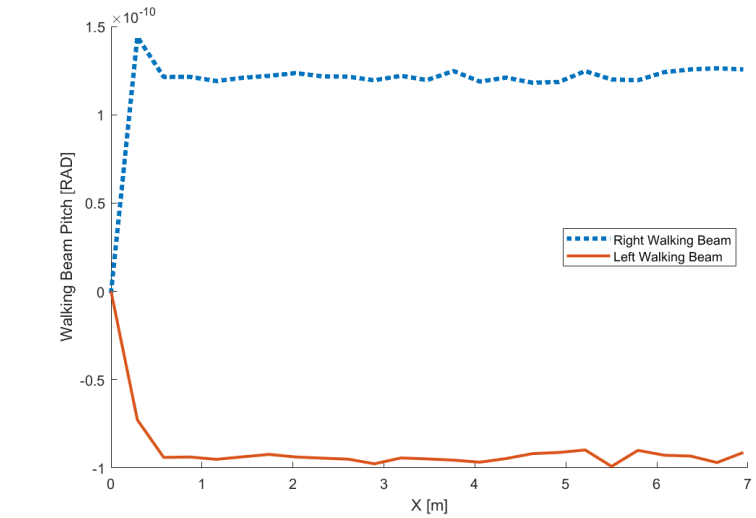


Figure C.72: Walking beam pitch vs distance (side-slope 10º), for slip, i=0.05.

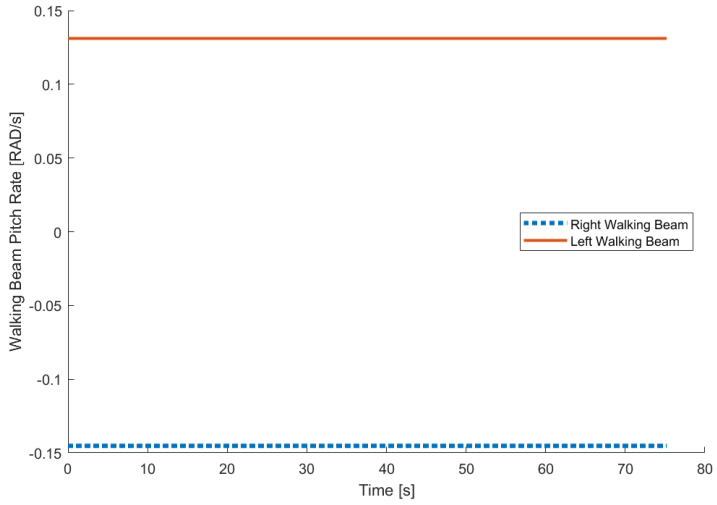


Figure C.73: Walking beam pitch rates vs time (side-slope 10º), for slip, i=0.05.

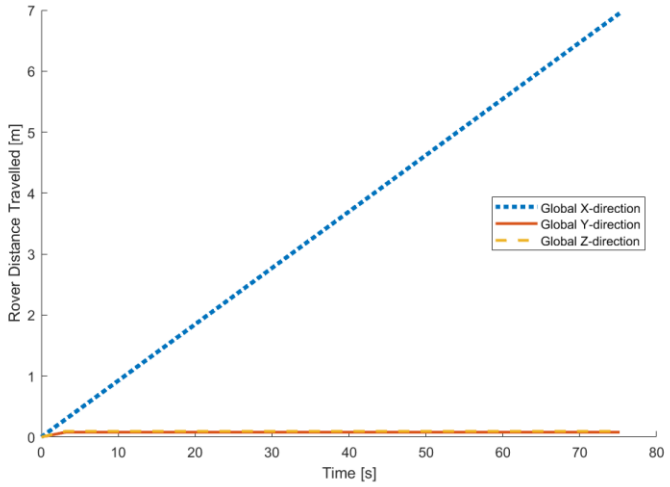


Figure C.74: Displacement vs time (side-slope 10º), for slip, i=0.05.

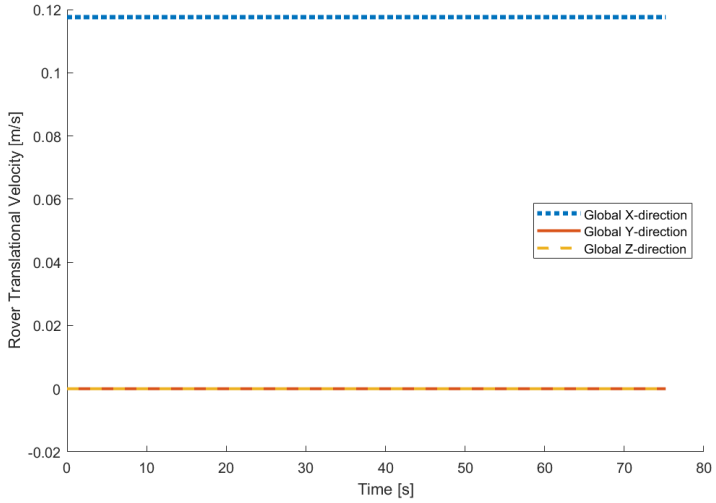


Figure C.75: Rover translational velocities vs time (side-slope 10º), for slip, i=0.05.

III. 0.1 Slip

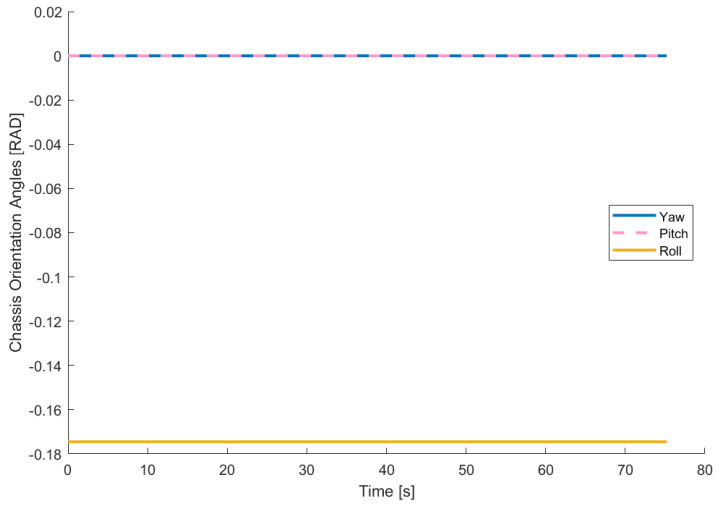


Figure C.76: Chassis angles vs time (side-slope 10º), for slip, i=0.1.

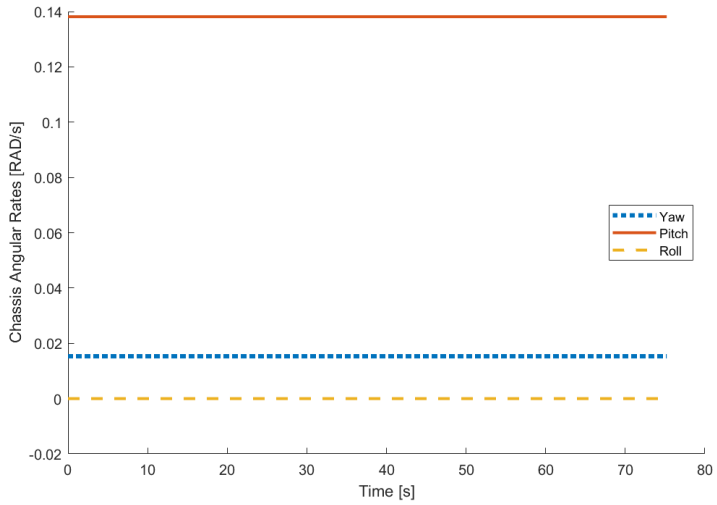


Figure C.77: Chassis angle rates vs time (side-slope 10º), for slip, i=0.1.

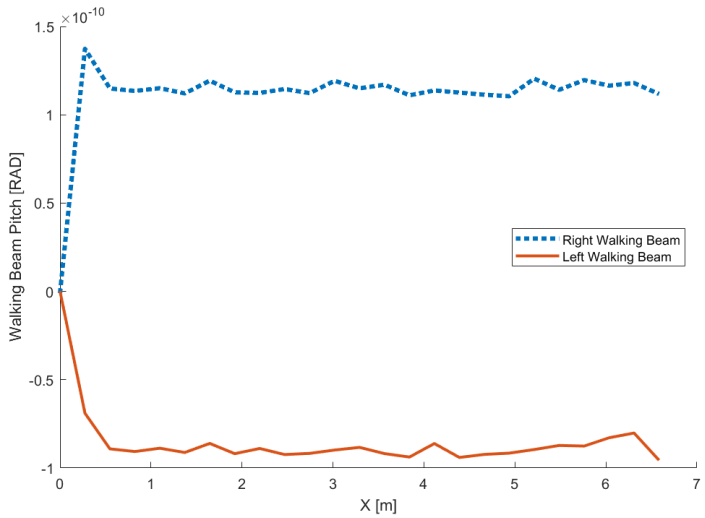


Figure C.78: Walking beam pitch vs distance (side-slope 10º), for slip, i=0.1.

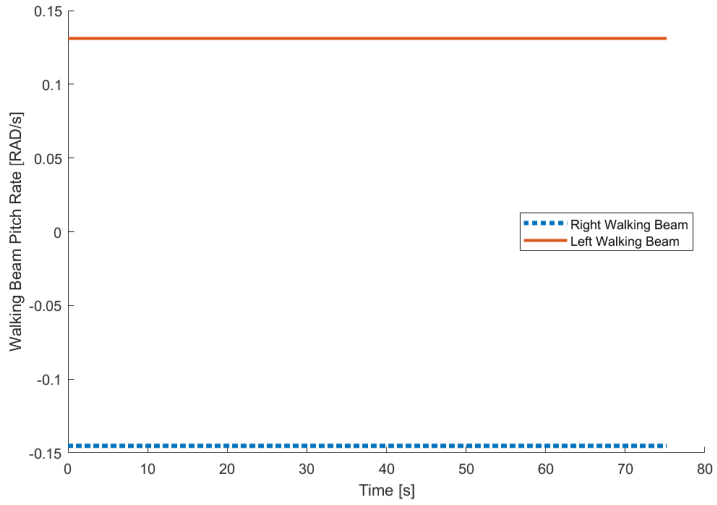


Figure C.79: Walking beam pitch rates vs time (side-slope 10º), for slip, i=0.1.

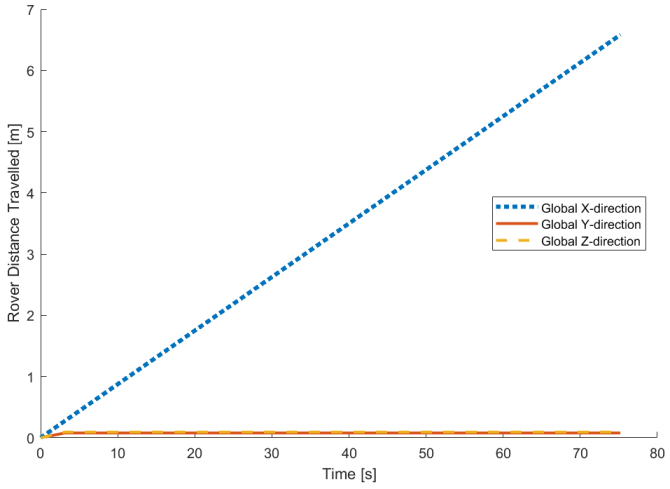


Figure C.80: Displacement vs time (side-slope 10°), for slip, i=0.1.

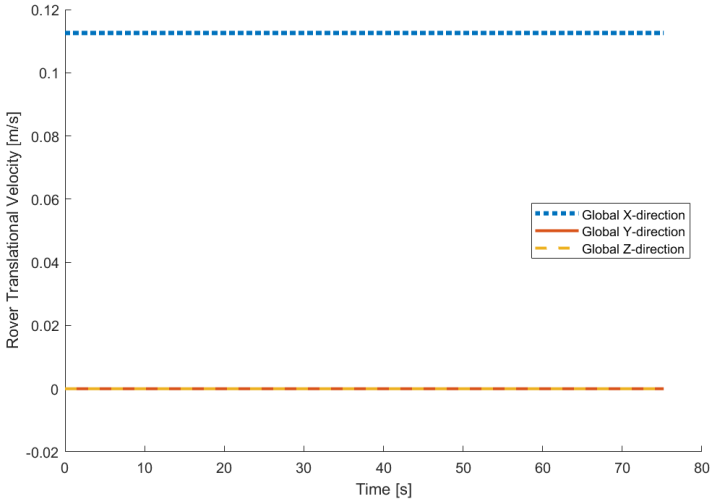


Figure C.81: Rover translational velocities vs time (side-slope 10°), for slip, i=0.1.

IV. 0.25 Slip

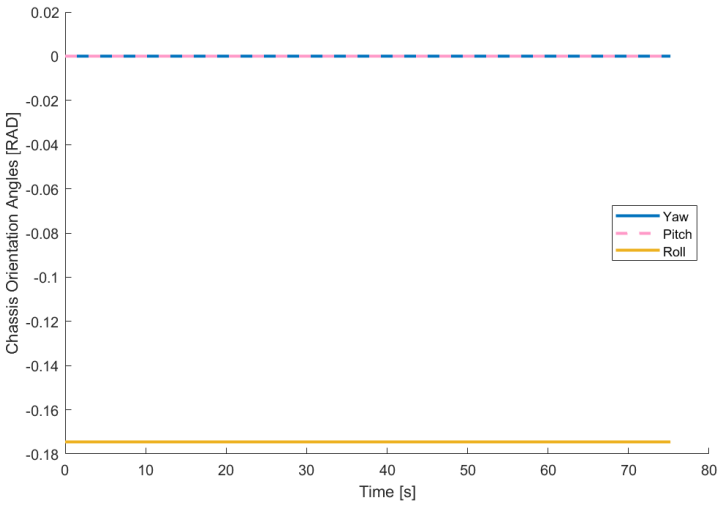


Figure C.82: Chassis angles vs time (side-slope 10º), for slip, i=0.25.

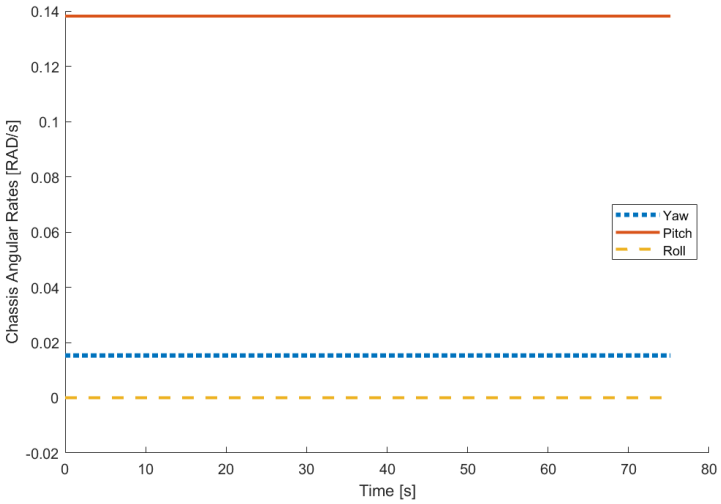


Figure C.83: Chassis angle rates vs time (side-slope 10º), for slip, i=0.25.

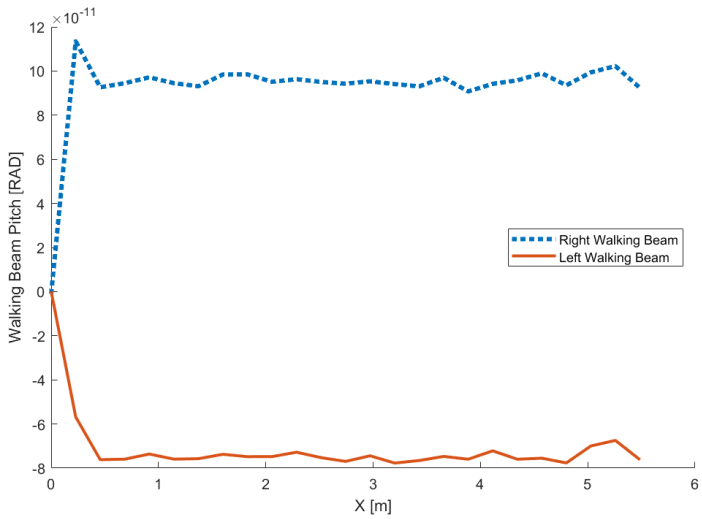


Figure C.84: Walking beam pitch vs distance (side-slope 10º), for slip, i=0.25.

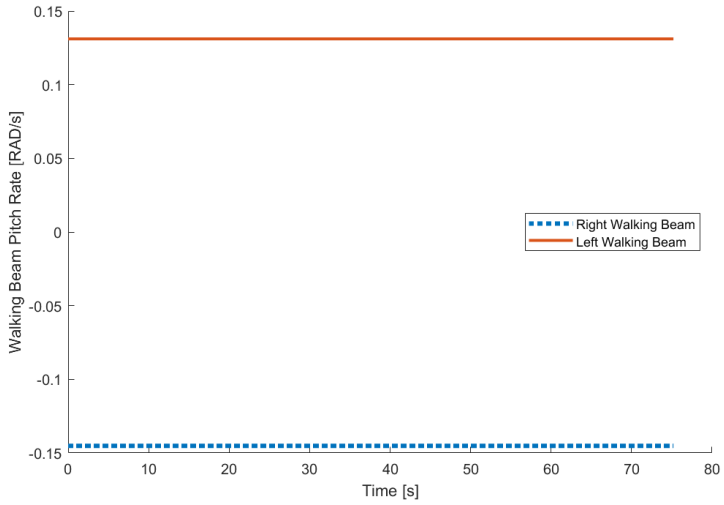


Figure C.85: Walking beam pitch rates vs time (side-slope 10º), for slip, i=0.25.

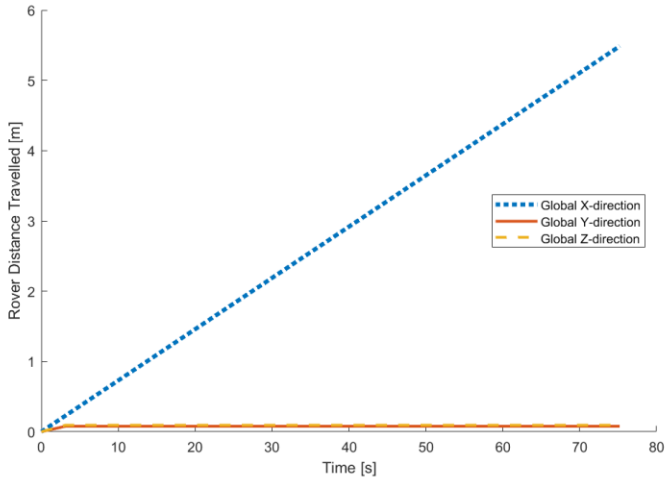


Figure C.86: Displacement vs time (side-slope 10º), for slip, i=0.25.

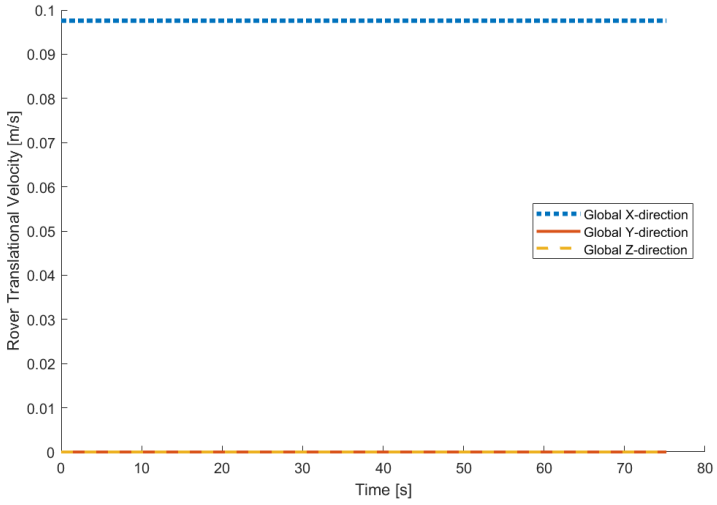


Figure C.87: Rover translational velocities vs time (side-slope 10º), for slip, i=0.25.

V. 0.5 Slip

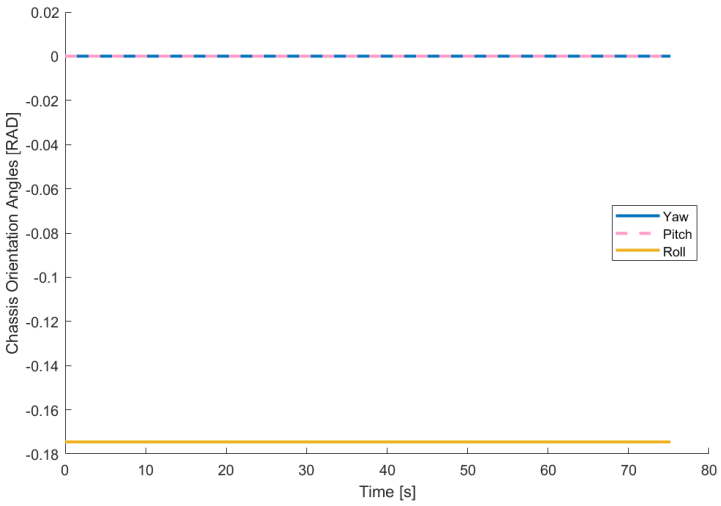


Figure C.88: Chassis angles vs time (side-slope 10º), for slip, i=0.5.

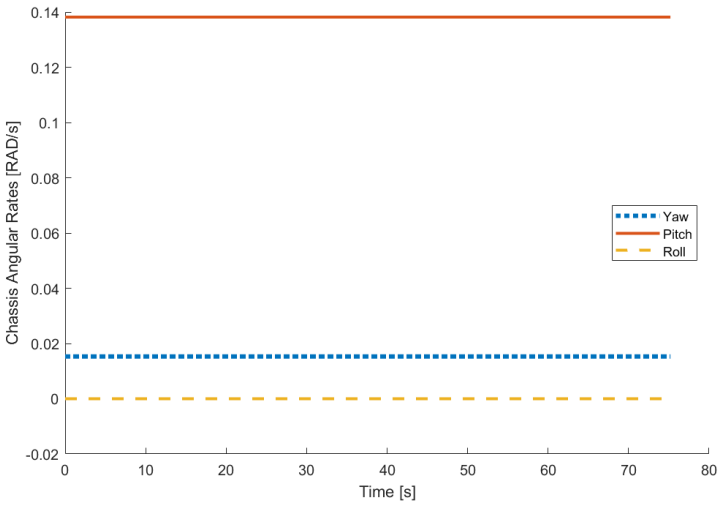


Figure C.89: Chassis angle rates vs time (side-slope 10º), for slip, i=0.5.

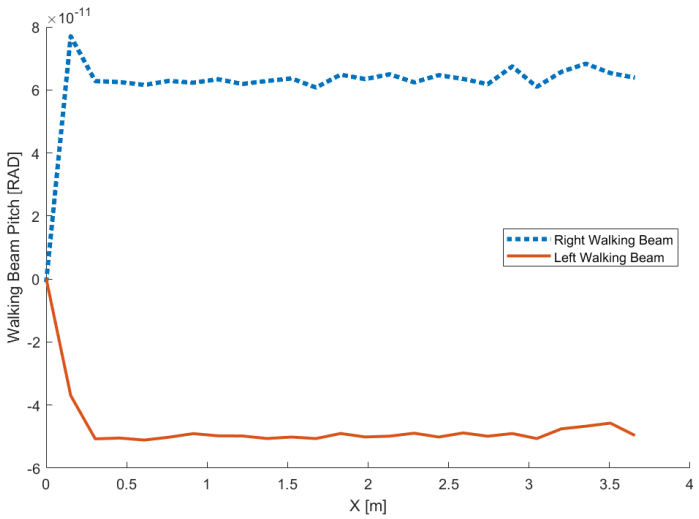


Figure C.90: Walking beam pitch vs distance (side-slope 10º), for slip, i=0.5.

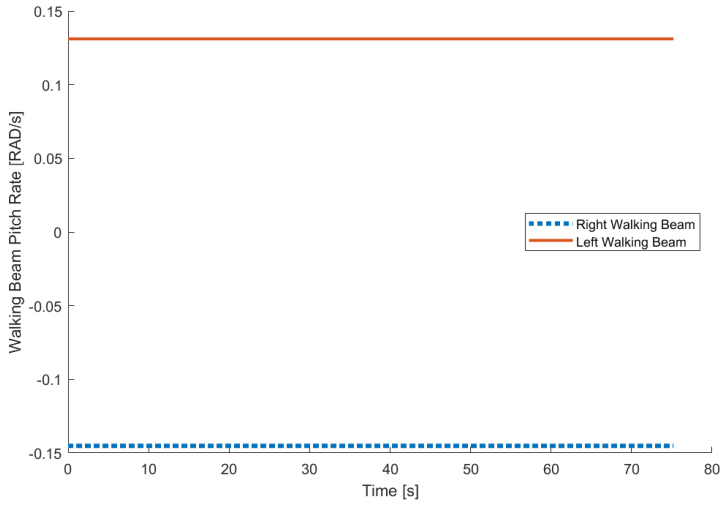


Figure C.91: Walking beam pitch rates vs time (side-slope 10º), for slip, i=0.5.

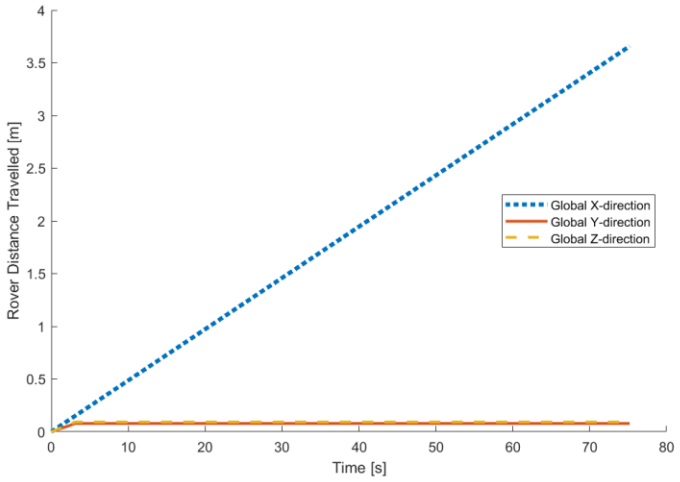


Figure C.92: Displacement vs time (side-slope 10°), for slip, i=0.5.

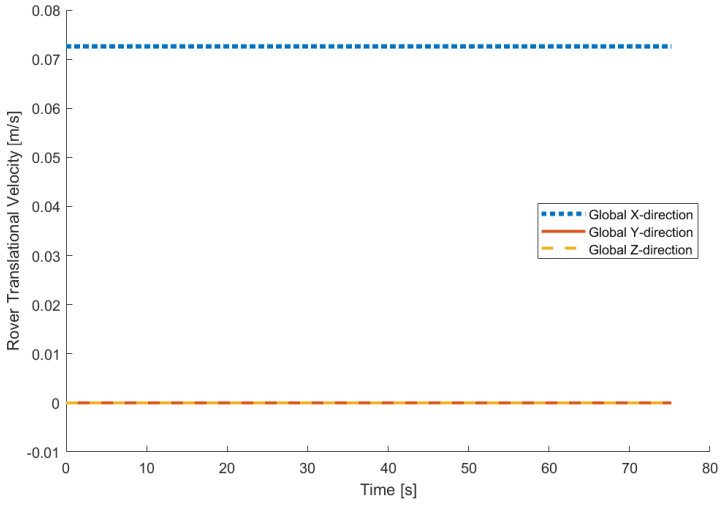


Figure C.93: Rover translational velocities vs time (side-slope 10°), for slip, i=0.5.

Case 4: Sinusoidal Terrain

Sine function: z=0.4sin(0.4x)

I. No-Slip

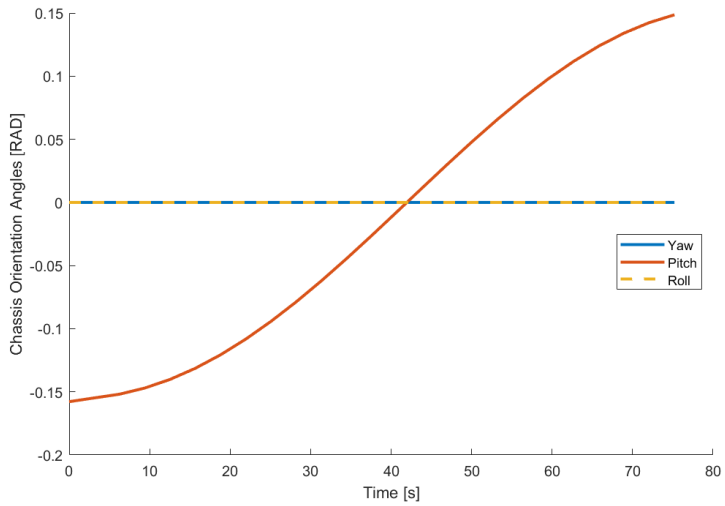


Figure C.94: Chassis angles vs time (sinusoidal terrain), for slip, i=0.

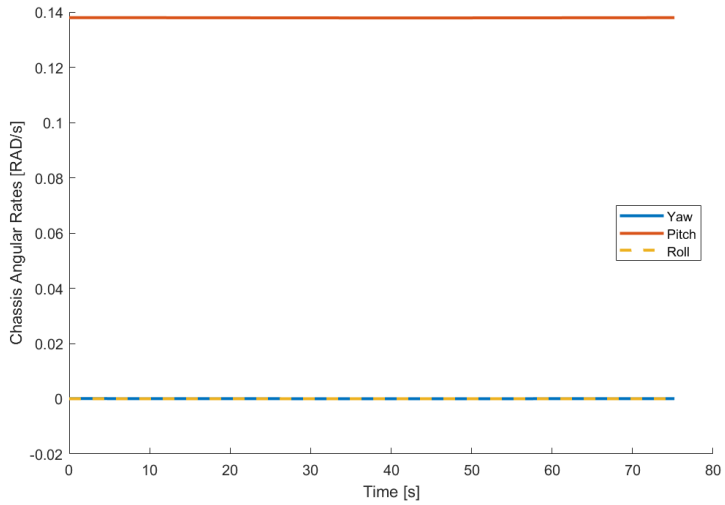


Figure C.95: Chassis angle rates vs time (sinusoidal terrain), for slip, i=0.

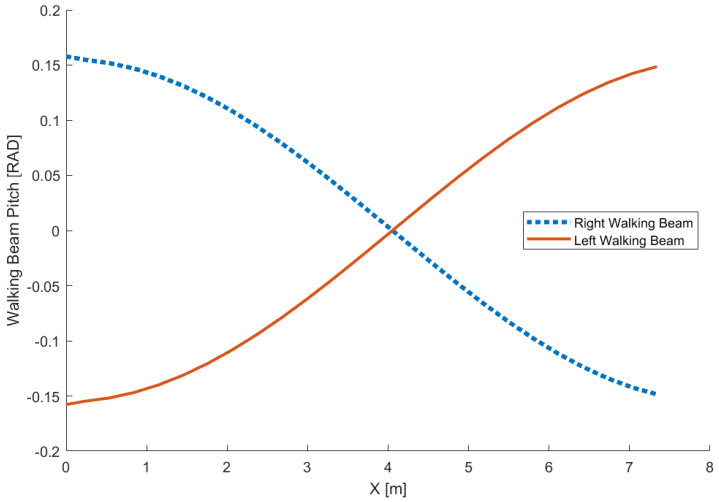


Figure C.96: Walking beam pitch vs distance (sinusoidal terrain), for slip, i=0.

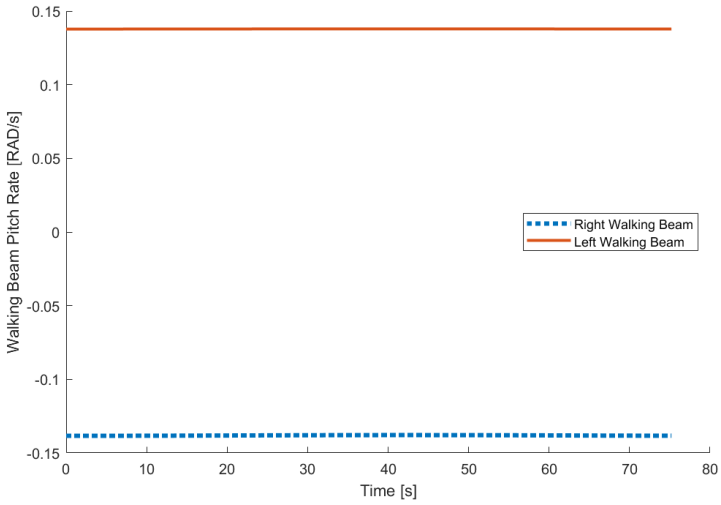


Figure C.97: Walking beam pitch rates vs time (sinusoidal terrain), for slip, i=0.

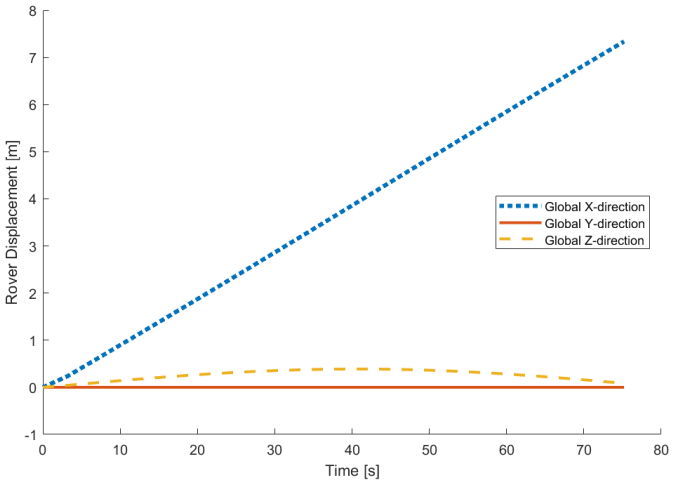


Figure C.98: Displacement vs time (sinusoidal terrain), for slip, i=0.

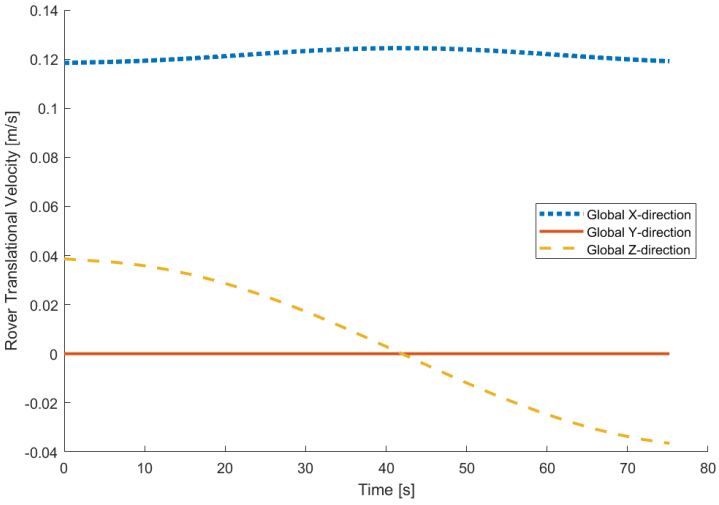


Figure C.99: Rover translational velocities vs time (sinusoidal terrain), for slip, i=0.

II. 0.05 Slip

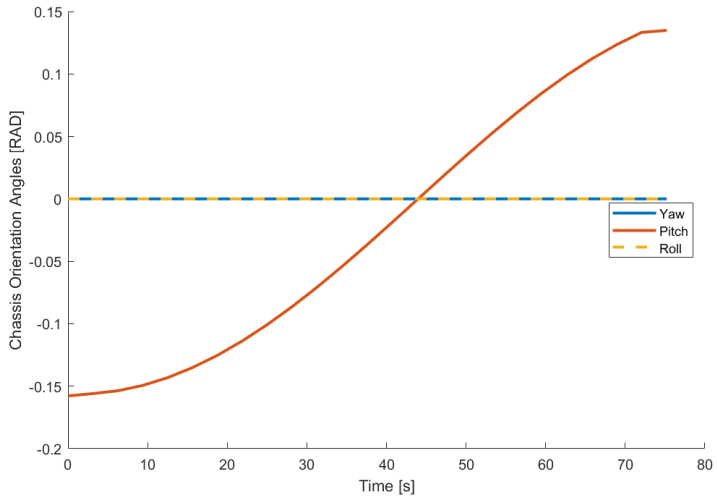


Figure C.100: Chassis angles vs time (sinusoidal terrain), for slip, i=0.05.

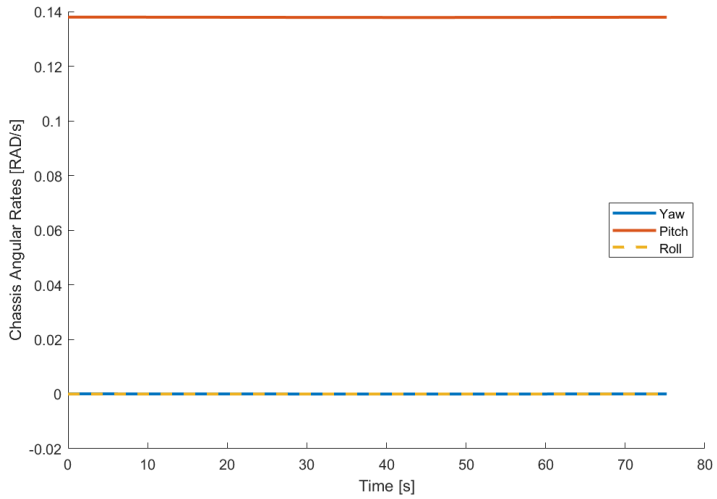


Figure C.101: Chassis angle rates vs time (sinusoidal terrain), for slip, i=0.05.

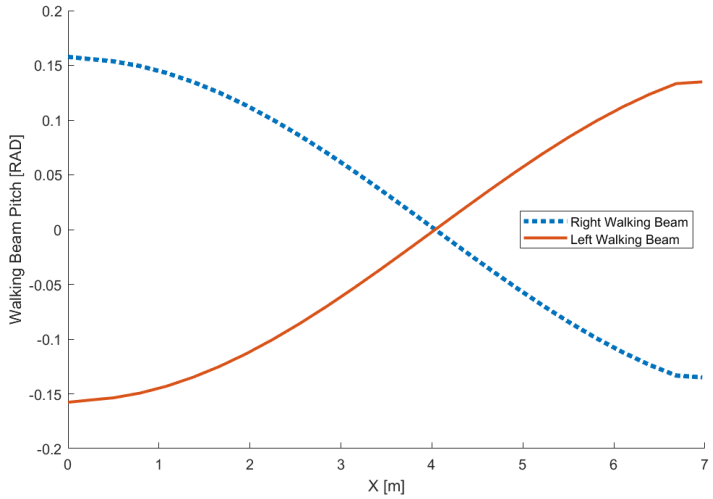


Figure C.102: Walking beam pitch vs distance (sinusoidal terrain), for slip, i=0.05.

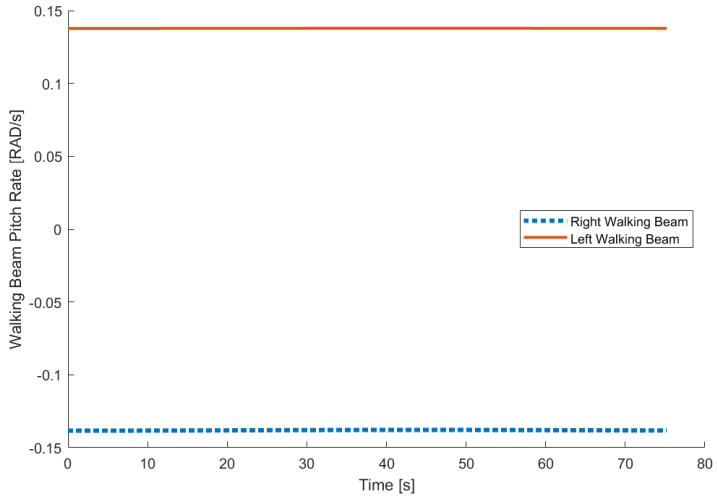


Figure C.103: Walking beam pitch rates vs time (sinusoidal terrain), for slip, i=0.05.

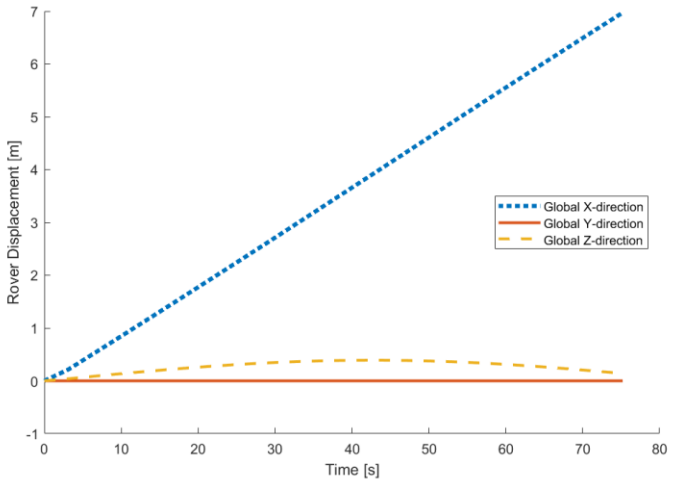


Figure C.104: Displacement vs time (sinusoidal terrain), for slip, i=0.05.

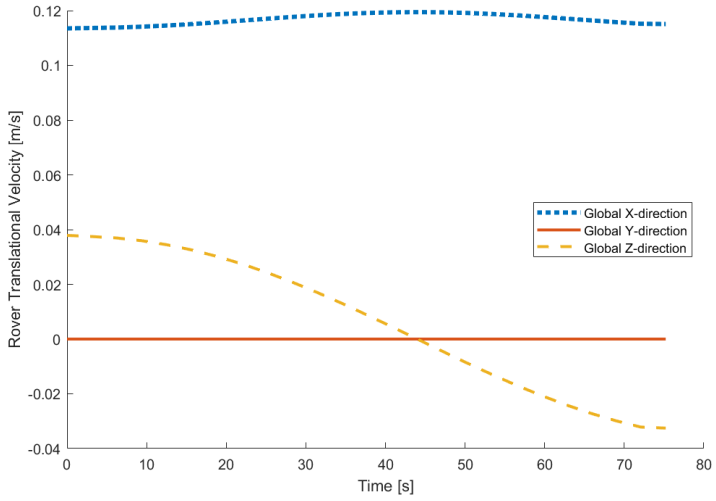


Figure C.105: Rover translational velocities vs time (sinusoidal terrain), for slip, i=0.05.

III. 0.1 Slip

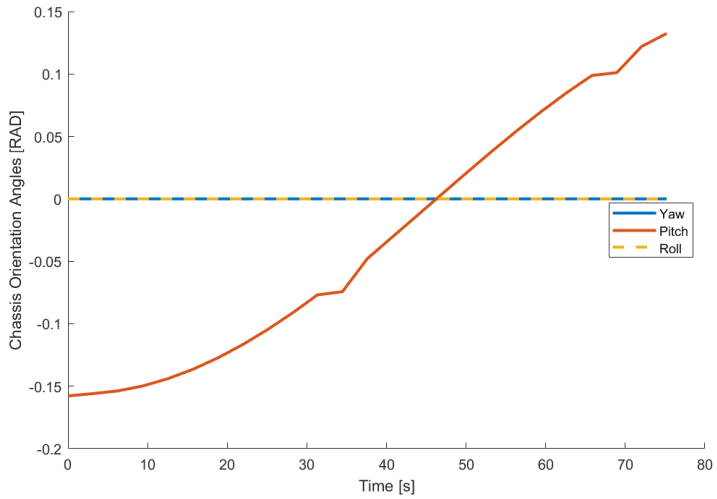


Figure C.106: Chassis angles vs time (sinusoidal terrain), for slip, i=0.1.

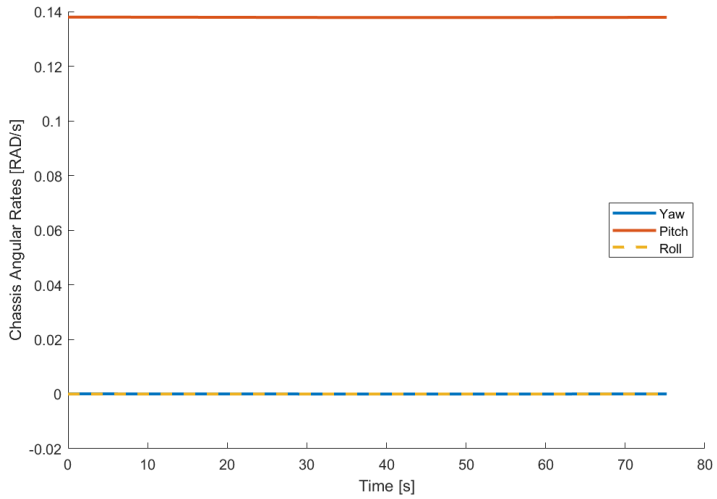


Figure C.107: Chassis angle rates vs time (sinusoidal terrain), for slip, i=0.1.

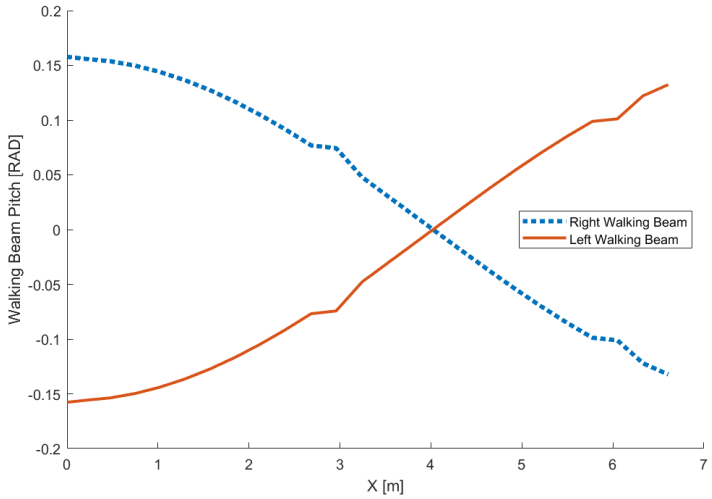


Figure C.108: Walking beam pitch vs distance (sinusoidal terrain), for slip, i=0.1.

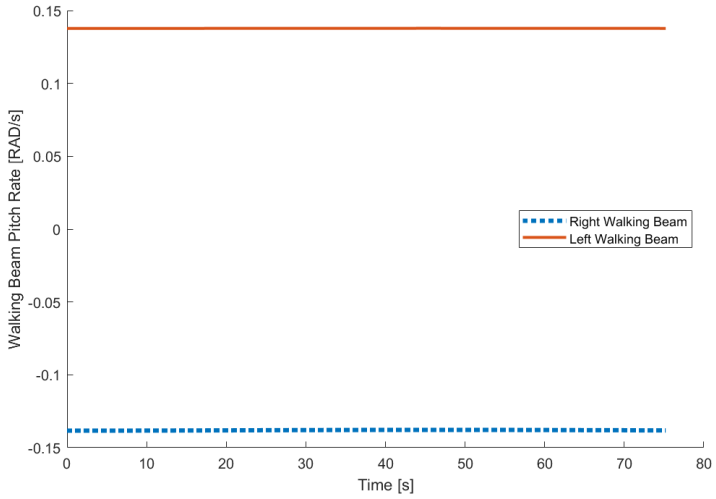


Figure C.109: Walking beam pitch rates vs time (sinusoidal terrain), for slip, i=0.1.

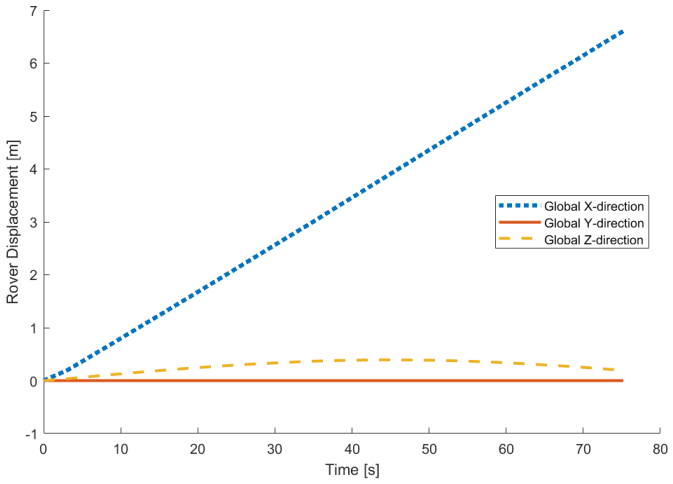


Figure C.110: Displacement vs time (sinusoidal terrain), for slip, i=0.1.

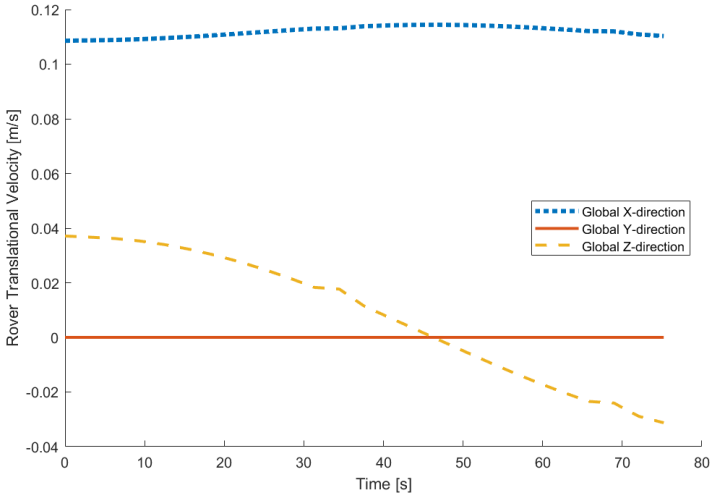


Figure C.111: Rover translational velocities vs time (sinusoidal terrain), for slip, i=0.1.

IV. 0.25 Slip

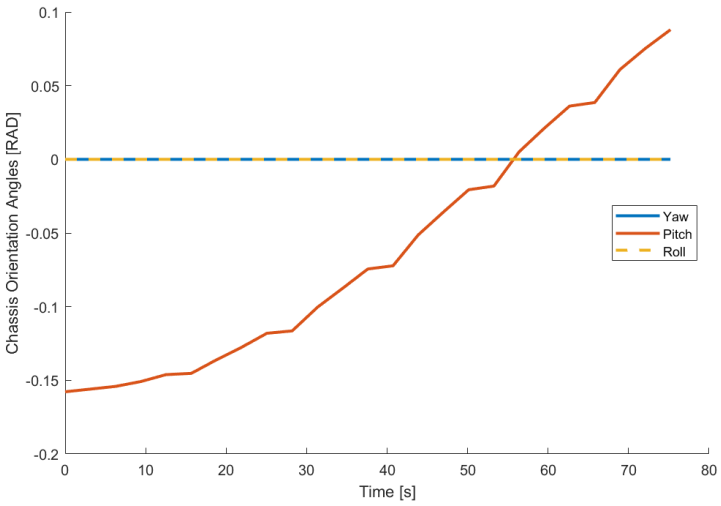


Figure C.112: Chassis angles vs time (sinusoidal terrain), for slip, i=0.25.

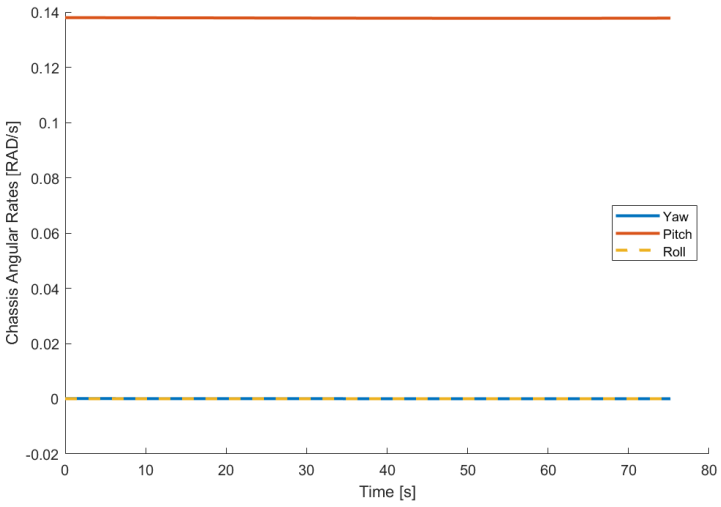


Figure C.113: Chassis angle rates vs time (sinusoidal terrain), for slip, i=0.25.

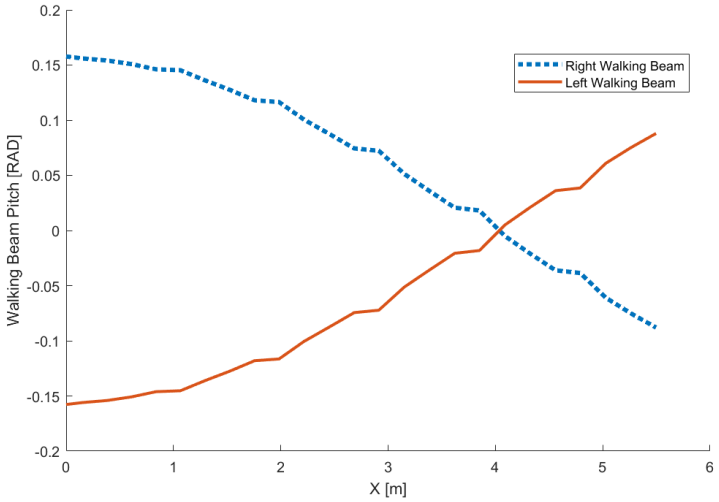


Figure C.114: Walking beam pitch vs distance (sinusoidal terrain), for slip, i=0.25.

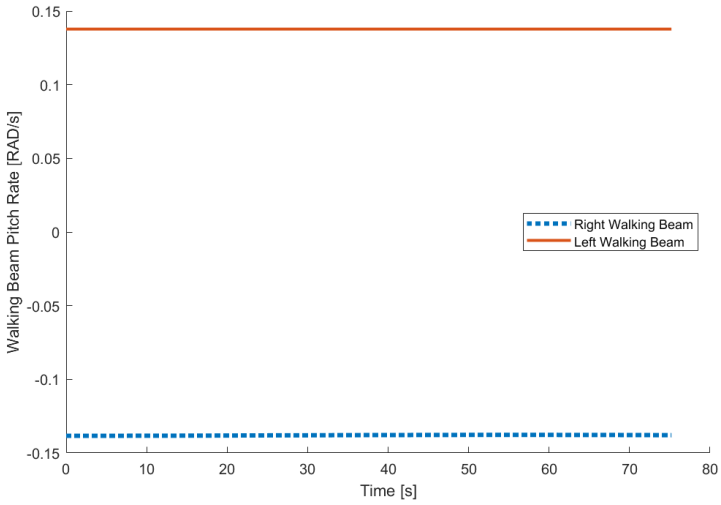


Figure C.115: Walking beam pitch rates vs time (sinusoidal terrain), for slip, i=0.25.

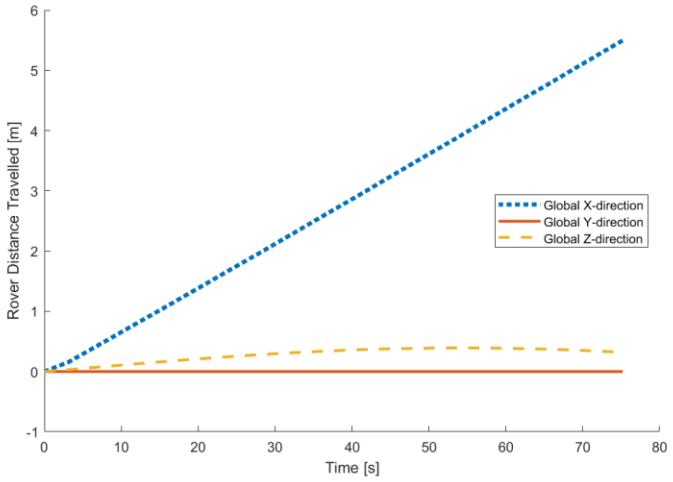


Figure C.116: Displacement vs time (sinusoidal terrain), for slip, i=0.25.

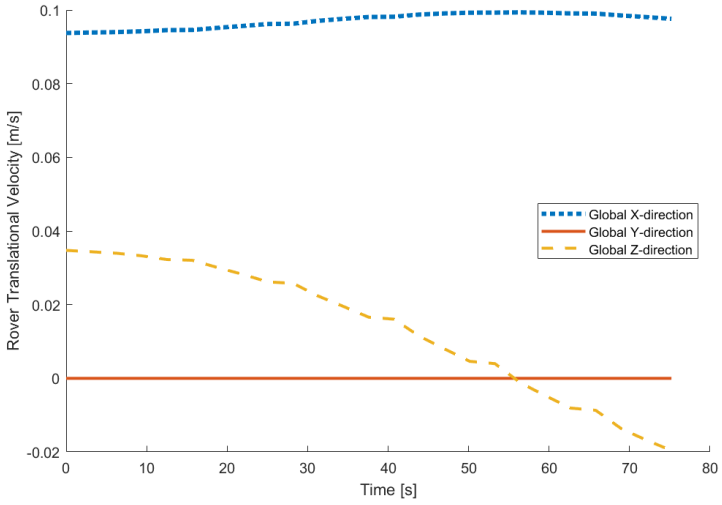


Figure C.117: Rover translational velocities vs time (sinusoidal terrain), for slip, i=0.25.

V. 0.5 Slip

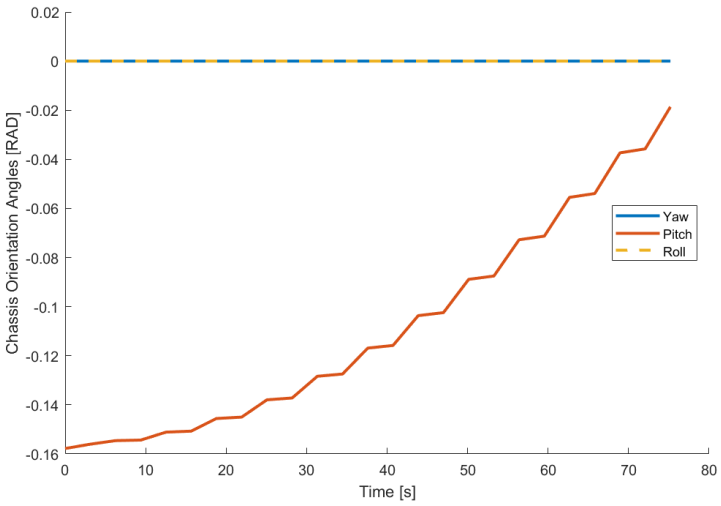


Figure C.118: Chassis angles vs time (sinusoidal terrain), for slip, i=0.5.

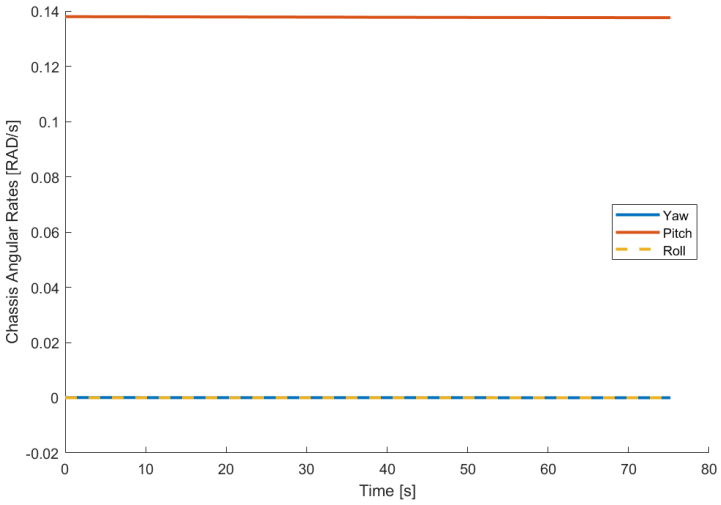


Figure C.119: Chassis angle rates vs time (sinusoidal terrain), for slip, i=0.5.

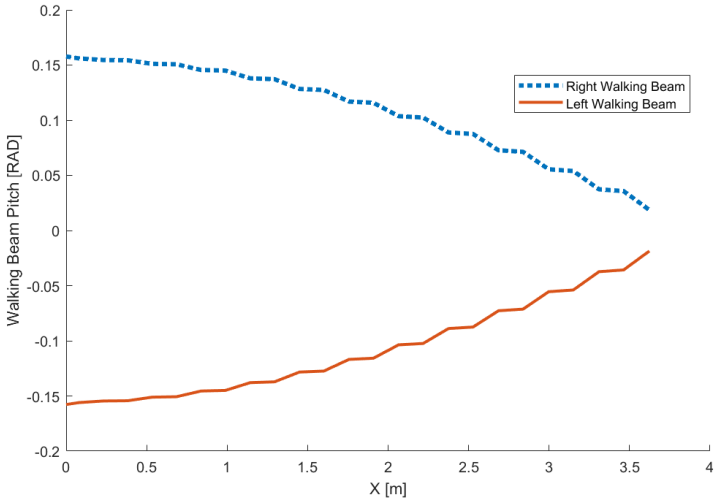


Figure C.120: Walking beam pitch vs distance (sinusoidal terrain), for slip, i=0.5.

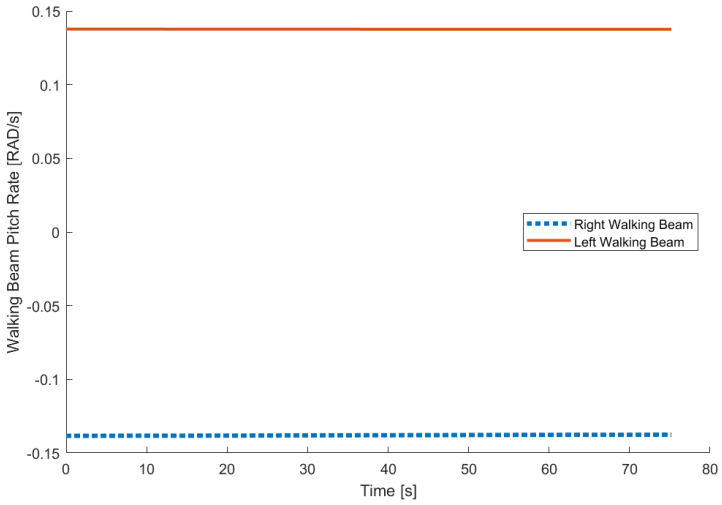


Figure C.121: Walking beam pitch rates vs time (sinusoidal terrain), for slip, i=0.5.

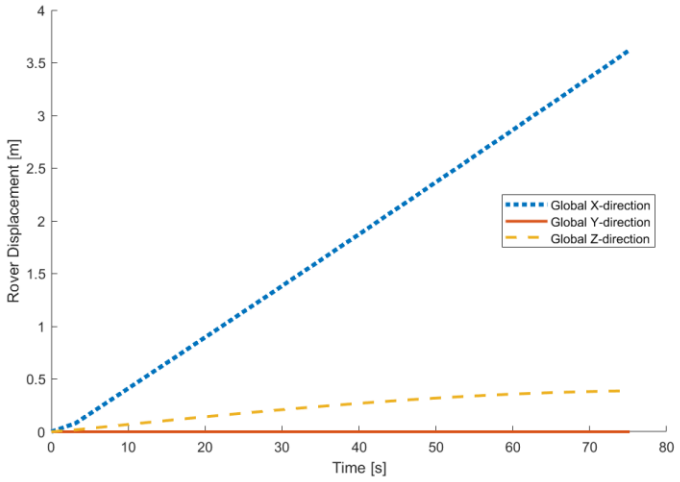


Figure C.122: Displacement vs time (sinusoidal terrain), for slip, i=0.5.

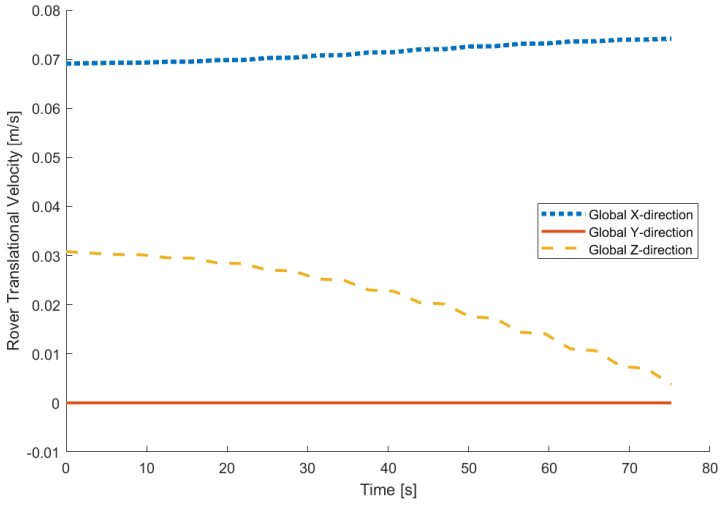


Figure C.123: Rover translational velocities vs time (sinusoidal terrain), for slip, i=0.5.

C.8 Additional Results – 3D Velocity Kinematic Analysis

A sample calculation for each percent deviation table is detailed below.

I. Percent deviation of rover velocities with respect to the flat case.

Using inclined (10°) terrain at 0.05 slip.

Velocity analysis produces rover velocities for inclined terrain at 0.05 slip of:

$$V_x = 0.1122\frac{m}{s}$$
$$V_z = 0.0418\frac{m}{s}$$

The corresponding flat terrain rover velocity at 0.05 slip is:

$$V_{x,flat} = 0.1195\frac{m}{s}$$

For comparison with the flat case, the resultant velocity for the inclined terrain is

$$V_r = \sqrt{(V_x^2 + V_z^2)}$$
$$= \sqrt{\left(0.1122\frac{m}{s}\right)^2 + \left(0.0418\frac{m}{s}\right)^2}$$
$$= 0.1197\frac{m}{s}$$

Comparing with the rover velocity for flat terrain with 0.05 slip, yields

$$\%Dev = \frac{V_r - V_{x,flat}}{V_{x,flat}} * 100$$
$$= \frac{0.1197\frac{m}{s} - 0.1195\frac{m}{s}}{0.1195\frac{m}{s}} * 100$$
$$= 0.20\%$$

II. Percent deviation of rover axle velocities with respect to the flat case.

Using inclined (10°) terrain at 0.05 slip.

Velocity analysis produces rover axle velocities for inclined terrain at 0.05 slip of:

$$V_{x,axle} = 0.0936\frac{m}{s}$$
$$V_{z,axle} = 0.0165\frac{m}{s}$$

The corresponding flat terrain rover velocity at 0.05 slip is:

$$V_{x,flat\ axle} = 0.0950\frac{m}{s}$$

For comparison with the flat case, the resultant velocity for the inclined terrain is

$$V_{r,axle} = \sqrt{\left(V_{x,axle}^2 + V_{z,axle}^2\right)}$$
$$= \sqrt{\left(0.0936\frac{m}{s}\right)^2 + \left(0.0165\frac{m}{s}\right)^2}$$
$$= 0.09504\frac{m}{s}$$

Comparing with the rover velocity for flat terrain with 0.05 slip, yields

$$\%Dev = \frac{V_{r,axle} - V_{x,flat\ axle}}{V_{x,flat\ axle}} * 100$$

$$= \frac{0.09504\frac{m}{s} - 0.0950\frac{m}{s}}{0.0950\frac{m}{s}} * 100$$

$$= 0.05\%$$

III. Percent deviation of rover displacement with respect to the flat case.

Using inclined (10°) terrain at 0.05 slip.

Velocity analysis produces rover displacement for inclined terrain at 0.05 slip of:

$$X = 6.8439\ m$$

$$Z = 1.2000\ m$$

The corresponding flat terrain rover displacement at 0.05 slip is:

$$X_{flat} = 6.9494\ m$$

For comparison with the flat case, the resultant velocity for the inclined terrain is

$$D_r = \sqrt{(X^2 + Z^2)}$$

$$= \sqrt{(6.8439\ m)^{\,2} + (1.2000\ m)^{\,2}}$$

$$= 6.9483\ m$$

Comparing with the rover velocity for flat terrain with 0.05 slip, yields

$$\%Dev = \frac{D_r - X_{flat}}{X_{flat}} * 100$$

$$= \frac{6.9483\ m - 6.9494\ m}{6.9494\ m} * 100$$

$$= -0.020\%$$

IV. Percent deviation of rover velocity from expected/commanded velocity.

For flat terrain (i = 0):

$$V_{x,flat} = 0.1245 \frac{m}{s}$$

Commanded velocity:

$$\omega r = 0.1000\ m/s$$

Percent Deviation:

$$\%Dev = 24.5\%$$

C.9 Sample Terrain Maps

Sample terrain maps are attached for each of the four basic terrain cases for a value of no-slip. It should be noted that to generate the maps for other slip values, one simply updates the value in the slip column at the appropriate locations for the given terrain. Also, these terrain maps include extra columns where the expected values were manually determined and inserted as a means for checking the first simulations.

Table C.5: Flat terrain map – no slip.

Terrain Coordinates — **Flat - No Slip**

Rover wheel base (left right) is 44.44" or 22.2188" (.5644 m) from cg

Rover wheel base (wheel to cg) is 18.0" (.4572 m) total front rear is .9144 m

	Left Rear Wheel				Right Rear Wheel				Left Front Wheel				Right Front Wheel				C of G			Angles in Degrees		
	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z	Pitch	Roll	Yaw
	x_lr	y_lr	z_lr	i_lr	x_rr	y_rr	z_rr	i_rr	x_lf	y_lf	z_lf	i_lf	x_rf	y_rf	z_rf	i_rf	x_cg	y_cg	z_cg			
1	-0.4572	0.5644	0.0000	0.0000	-0.4572	-0.5644	0.0000	0.0000	0.4572	0.5644	0.0000	0.0000	0.4572	-0.5644	0.0000	0.0000	0.0000	0.0000	0.4515	0.0000	0.0000	0.0000
	-0.1524	0.5644	0.0000	0.0000	-0.1524	-0.5644	0.0000	0.0000	0.7620	0.5644	0.0000	0.0000	0.7620	-0.5644	0.0000	0.0000	0.3048	0.0000	0.4515	0.0000	0.0000	0.0000
	0.1524	0.5644	0.0000	0.0000	0.1524	-0.5644	0.0000	0.0000	1.0668	0.5644	0.0000	0.0000	1.0668	-0.5644	0.0000	0.0000	0.6096	0.0000	0.4515	0.0000	0.0000	0.0000
2	0.4572	0.5644	0.0000	0.0000	0.4572	-0.5644	0.0000	0.0000	1.3716	0.5644	0.0000	0.0000	1.3716	-0.5644	0.0000	0.0000	0.9144	0.0000	0.4515	0.0000	0.0000	0.0000
	0.7620	0.5644	0.0000	0.0000	0.7620	-0.5644	0.0000	0.0000	1.6764	0.5644	0.0000	0.0000	1.6764	-0.5644	0.0000	0.0000	1.2192	0.0000	0.4515	0.0000	0.0000	0.0000
	1.0668	0.5644	0.0000	0.0000	1.0668	-0.5644	0.0000	0.0000	1.9812	0.5644	0.0000	0.0000	1.9812	-0.5644	0.0000	0.0000	1.5240	0.0000	0.4515	0.0000	0.0000	0.0000
3	1.3716	0.5644	0.0000	0.0000	1.3716	-0.5644	0.0000	0.0000	2.2860	0.5644	0.0000	0.0000	2.2860	-0.5644	0.0000	0.0000	1.8288	0.0000	0.4515	0.0000	0.0000	0.0000
	1.6764	0.5644	0.0000	0.0000	1.6764	-0.5644	0.0000	0.0000	2.5908	0.5644	0.0000	0.0000	2.5908	-0.5644	0.0000	0.0000	2.1336	0.0000	0.4515	0.0000	0.0000	0.0000
	1.9812	0.5644	0.0000	0.0000	1.9812	-0.5644	0.0000	0.0000	2.8956	0.5644	0.0000	0.0000	2.8956	-0.5644	0.0000	0.0000	2.4384	0.0000	0.4515	0.0000	0.0000	0.0000
4	2.2860	0.5644	0.0000	0.0000	2.2860	-0.5644	0.0000	0.0000	3.2004	0.5644	0.0000	0.0000	3.2004	-0.5644	0.0000	0.0000	2.7432	0.0000	0.4515	0.0000	0.0000	0.0000
	2.5908	0.5644	0.0000	0.0000	2.5908	-0.5644	0.0000	0.0000	3.5052	0.5644	0.0000	0.0000	3.5052	-0.5644	0.0000	0.0000	3.0480	0.0000	0.4515	0.0000	0.0000	0.0000
	2.8956	0.5644	0.0000	0.0000	2.8956	-0.5644	0.0000	0.0000	3.8100	0.5644	0.0000	0.0000	3.8100	-0.5644	0.0000	0.0000	3.3528	0.0000	0.4515	0.0000	0.0000	0.0000
5	3.2004	0.5644	0.0000	0.0000	3.2004	-0.5644	0.0000	0.0000	4.1148	0.5644	0.0000	0.0000	4.1148	-0.5644	0.0000	0.0000	3.6576	0.0000	0.4515	0.0000	0.0000	0.0000
	3.5052	0.5644	0.0000	0.0000	3.5052	-0.5644	0.0000	0.0000	4.4196	0.5644	0.0000	0.0000	4.4196	-0.5644	0.0000	0.0000	3.9624	0.0000	0.4515	0.0000	0.0000	0.0000
	3.8100	0.5644	0.0000	0.0000	3.8100	-0.5644	0.0000	0.0000	4.7244	0.5644	0.0000	0.0000	4.7244	-0.5644	0.0000	0.0000	4.2672	0.0000	0.4515	0.0000	0.0000	0.0000
6	4.1148	0.5644	0.0000	0.0000	4.1148	-0.5644	0.0000	0.0000	5.0292	0.5644	0.0000	0.0000	5.0292	-0.5644	0.0000	0.0000	4.5720	0.0000	0.4515	0.0000	0.0000	0.0000
	4.4196	0.5644	0.0000	0.0000	4.4196	-0.5644	0.0000	0.0000	5.3340	0.5644	0.0000	0.0000	5.3340	-0.5644	0.0000	0.0000	4.8768	0.0000	0.4515	0.0000	0.0000	0.0000
	4.7244	0.5644	0.0000	0.0000	4.7244	-0.5644	0.0000	0.0000	5.6388	0.5644	0.0000	0.0000	5.6388	-0.5644	0.0000	0.0000	5.1816	0.0000	0.4515	0.0000	0.0000	0.0000
7	5.0292	0.5644	0.0000	0.0000	5.0292	-0.5644	0.0000	0.0000	5.9436	0.5644	0.0000	0.0000	5.9436	-0.5644	0.0000	0.0000	5.4864	0.0000	0.4515	0.0000	0.0000	0.0000
	5.3340	0.5644	0.0000	0.0000	5.3340	-0.5644	0.0000	0.0000	6.2484	0.5644	0.0000	0.0000	6.2484	-0.5644	0.0000	0.0000	5.7912	0.0000	0.4515	0.0000	0.0000	0.0000
	5.6388	0.5644	0.0000	0.0000	5.6388	-0.5644	0.0000	0.0000	6.5532	0.5644	0.0000	0.0000	6.5532	-0.5644	0.0000	0.0000	6.0960	0.0000	0.4515	0.0000	0.0000	0.0000
8	5.9436	0.5644	0.0000	0.0000	5.9436	-0.5644	0.0000	0.0000	6.8580	0.5644	0.0000	0.0000	6.8580	-0.5644	0.0000	0.0000	6.4008	0.0000	0.4515	0.0000	0.0000	0.0000
	6.2484	0.5644	0.0000	0.0000	6.2484	-0.5644	0.0000	0.0000	7.1628	0.5644	0.0000	0.0000	7.1628	-0.5644	0.0000	0.0000	6.7056	0.0000	0.4515	0.0000	0.0000	0.0000
	6.5532	0.5644	0.0000	0.0000	6.5532	-0.5644	0.0000	0.0000	7.4676	0.5644	0.0000	0.0000	7.4676	-0.5644	0.0000	0.0000	7.0104	0.0000	0.4515	0.0000	0.0000	0.0000
9	6.8580	0.5644	0.0000	0.0000	6.8580	-0.5644	0.0000	0.0000	7.7724	0.5644	0.0000	0.0000	7.7724	-0.5644	0.0000	0.0000	7.3152	0.0000	0.4515	0.0000	0.0000	0.0000
	7.1628	0.5644	0.0000	0.0000	7.1628	-0.5644	0.0000	0.0000	8.0772	0.5644	0.0000	0.0000	8.0772	-0.5644	0.0000	0.0000	7.6200	0.0000	0.4515	0.0000	0.0000	0.0000
	7.4676	0.5644	0.0000	0.0000	7.4676	-0.5644	0.0000	0.0000	8.3820	0.5644	0.0000	0.0000	8.3820	-0.5644	0.0000	0.0000	7.9248	0.0000	0.4515	0.0000	0.0000	0.0000
10	7.7724	0.5644	0.0000	0.0000	7.7724	-0.5644	0.0000	0.0000	8.6868	0.5644	0.0000	0.0000	8.6868	-0.5644	0.0000	0.0000	8.2296	0.0000	0.4515	0.0000	0.0000	0.0000
	8.0772	0.5644	0.0000	0.0000	8.0772	-0.5644	0.0000	0.0000	8.9916	0.5644	0.0000	0.0000	8.9916	-0.5644	0.0000	0.0000	8.5344	0.0000	0.4515	0.0000	0.0000	0.0000
	8.3820	0.5644	0.0000	0.0000	8.3820	-0.5644	0.0000	0.0000	9.2964	0.5644	0.0000	0.0000	9.2964	-0.5644	0.0000	0.0000	8.8392	0.0000	0.4515	0.0000	0.0000	0.0000
11	8.6868	0.5644	0.0000	0.0000	8.6868	-0.5644	0.0000	0.0000	9.6012	0.5644	0.0000	0.0000	9.6012	-0.5644	0.0000	0.0000	9.1440	0.0000	0.4515	0.0000	0.0000	0.0000
	8.9916	0.5644	0.0000	0.0000	8.9916	-0.5644	0.0000	0.0000	9.9060	0.5644	0.0000	0.0000	9.9060	-0.5644	0.0000	0.0000	9.4488	0.0000	0.4515	0.0000	0.0000	0.0000

Table C.6: Inclined terrain (10°) map – no slip.

Terrain Coordinates — **Inclined 10° - No Slip**

(Front wheels upslope)

Rover wheel base (left right) is 44.44" or 22.2188" (.5644 m) from cg

Rover wheel base (wheel to cg) is 18.0" (.4572 m) total front rear is .9144 m

	Left Rear Wheel				Right Rear Wheel				Left Front Wheel				Right Front Wheel				C of G		
	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z
	x_lr	y_lr	z_lr	i_lr	x_rr	y_rr	z_rr	i_rr	x_lf	y_lf	z_lf	i_lf	x_rf	y_rf	z_rf	i_rf	x_cg	y_cg	z_cg
1	-0.37180	0.56440	-0.07940	0.00000	-0.37180	-0.56440	-0.07940	0.00000	0.52870	0.56440	0.07940	0.00000	0.52870	-0.56440	0.07940	0.00000	0.00000	0.00000	0.44460
	-0.07163	0.56440	-0.02647	0.00000	-0.07163	-0.56440	-0.02647	0.00000	0.82887	0.56440	0.13233	0.00000	0.82887	-0.56440	0.13233	0.00000	0.30017	0.00000	0.49753
	0.22854	0.56440	0.02646	0.00000	0.22854	-0.56440	0.02646	0.00000	1.12904	0.56440	0.18526	0.00000	1.12904	-0.56440	0.18526	0.00000	0.60034	0.00000	0.55046
2	0.52870	0.56440	0.07940	0.00000	0.52870	-0.56440	0.07940	0.00000	1.42920	0.56440	0.23820	0.00000	1.42920	-0.56440	0.23820	0.00000	0.90050	0.00000	0.60340
	0.82887	0.56440	0.13233	0.00000	0.82887	-0.56440	0.13233	0.00000	1.72937	0.56440	0.29113	0.00000	1.72937	-0.56440	0.29113	0.00000	1.20067	0.00000	0.65633
	1.12904	0.56440	0.18526	0.00000	1.12904	-0.56440	0.18526	0.00000	2.02954	0.56440	0.34406	0.00000	2.02954	-0.56440	0.34406	0.00000	1.50084	0.00000	0.70926
3	1.42920	0.56440	0.23820	0.00000	1.42920	-0.56440	0.23820	0.00000	2.32971	0.56440	0.39700	0.00000	2.32971	-0.56440	0.39700	0.00000	1.80100	0.00000	0.76220
	1.72937	0.56440	0.29113	0.00000	1.72937	-0.56440	0.29113	0.00000	2.62987	0.56440	0.44993	0.00000	2.62987	-0.56440	0.44993	0.00000	2.10117	0.00000	0.81513
	2.02954	0.56440	0.34406	0.00000	2.02954	-0.56440	0.34406	0.00000	2.93004	0.56440	0.50286	0.00000	2.93004	-0.56440	0.50286	0.00000	2.40134	0.00000	0.86806
4	2.32971	0.56440	0.39700	0.00000	2.32971	-0.56440	0.39700	0.00000	3.23021	0.56440	0.55579	0.00000	3.23021	-0.56440	0.55579	0.00000	2.70151	0.00000	0.92100
	2.62987	0.56440	0.44993	0.00000	2.62987	-0.56440	0.44993	0.00000	3.53037	0.56440	0.60873	0.00000	3.53037	-0.56440	0.60873	0.00000	3.00167	0.00000	0.97393
	2.93004	0.56440	0.50286	0.00000	2.93004	-0.56440	0.50286	0.00000	3.83054	0.56440	0.66166	0.00000	3.83054	-0.56440	0.66166	0.00000	3.30184	0.00000	1.02686
5	3.23021	0.56440	0.55579	0.00000	3.23021	-0.56440	0.55579	0.00000	4.13071	0.56440	0.71459	0.00000	4.13071	-0.56440	0.71459	0.00000	3.60201	0.00000	1.07979
	3.53037	0.56440	0.60873	0.00000	3.53037	-0.56440	0.60873	0.00000	4.43087	0.56440	0.76753	0.00000	4.43087	-0.56440	0.76753	0.00000	3.90217	0.00000	1.13273
	3.83054	0.56440	0.66166	0.00000	3.83054	-0.56440	0.66166	0.00000	4.73104	0.56440	0.82046	0.00000	4.73104	-0.56440	0.82046	0.00000	4.20234	0.00000	1.18566
6	4.13071	0.56440	0.71459	0.00000	4.13071	-0.56440	0.71459	0.00000	5.03121	0.56440	0.87339	0.00000	5.03121	-0.56440	0.87339	0.00000	4.50251	0.00000	1.23859
	4.43087	0.56440	0.76753	0.00000	4.43087	-0.56440	0.76753	0.00000	5.33138	0.56440	0.92633	0.00000	5.33138	-0.56440	0.92633	0.00000	4.80267	0.00000	1.29153
	4.73104	0.56440	0.82046	0.00000	4.73104	-0.56440	0.82046	0.00000	5.63154	0.56440	0.97926	0.00000	5.63154	-0.56440	0.97926	0.00000	5.10284	0.00000	1.34446
7	5.03121	0.56440	0.87339	0.00000	5.03121	-0.56440	0.87339	0.00000	5.93171	0.56440	1.03219	0.00000	5.93171	-0.56440	1.03219	0.00000	5.40301	0.00000	1.39739
	5.33138	0.56440	0.92633	0.00000	5.33138	-0.56440	0.92633	0.00000	6.23188	0.56440	1.08512	0.00000	6.23188	-0.56440	1.08512	0.00000	5.70318	0.00000	1.45033
	5.63154	0.56440	0.97926	0.00000	5.63154	-0.56440	0.97926	0.00000	6.53204	0.56440	1.13806	0.00000	6.53204	-0.56440	1.13806	0.00000	6.00334	0.00000	1.50326
8	5.93171	0.56440	1.03219	0.00000	5.93171	-0.56440	1.03219	0.00000	6.83221	0.56440	1.19099	0.00000	6.83221	-0.56440	1.19099	0.00000	6.30351	0.00000	1.55619
	6.23188	0.56440	1.08512	0.00000	6.23188	-0.56440	1.08512	0.00000	7.13238	0.56440	1.24392	0.00000	7.13238	-0.56440	1.24392	0.00000	6.60368	0.00000	1.60912
	6.53204	0.56440	1.13806	0.00000	6.53204	-0.56440	1.13806	0.00000	7.43254	0.56440	1.29686	0.00000	7.43254	-0.56440	1.29686	0.00000	6.90384	0.00000	1.66206
9	6.83221	0.56440	1.19099	0.00000	6.83221	-0.56440	1.19099	0.00000	7.73271	0.56440	1.34979	0.00000	7.73271	-0.56440	1.34979	0.00000	7.20401	0.00000	1.71499
	7.13238	0.56440	1.24392	0.00000	7.13238	-0.56440	1.24392	0.00000	8.03288	0.56440	1.40272	0.00000	8.03288	-0.56440	1.40272	0.00000	7.50418	0.00000	1.76792
	7.43254	0.56440	1.29686	0.00000	7.43254	-0.56440	1.29686	0.00000	8.33305	0.56440	1.45566	0.00000	8.33305	-0.56440	1.45566	0.00000	7.80434	0.00000	1.82086
10	7.73271	0.56440	1.34979	0.00000	7.73271	-0.56440	1.34979	0.00000	8.63321	0.56440	1.50859	0.00000	8.63321	-0.56440	1.50859	0.00000	8.10451	0.00000	1.87379
	8.03288	0.56440	1.40272	0.00000	8.03288	-0.56440	1.40272	0.00000	8.93338	0.56440	1.56152	0.00000	8.93338	-0.56440	1.56152	0.00000	8.40468	0.00000	1.92672
	8.33305	0.56440	1.45566	0.00000	8.33305	-0.56440	1.45566	0.00000	9.23355	0.56440	1.61445	0.00000	9.23355	-0.56440	1.61445	0.00000	8.70485	0.00000	1.97966
11	8.63321	0.56440	1.50859	0.00000	8.63321	-0.56440	1.50859	0.00000	9.53371	0.56440	1.66739	0.00000	9.53371	-0.56440	1.66739	0.00000	9.00501	0.00000	2.03259
	8.93338	0.56440	1.56152	0.00000	8.93338	-0.56440	1.56152	0.00000	9.83388	0.56440	1.72032	0.00000	9.83388	-0.56440	1.72032	0.00000	9.30518	0.00000	2.08552

Table C.7: Side slope terrain (10°) map – no slip.

Terrain Coordinates — **Level Side Slope (10°) - No Slip**

(Right side of rover is upslope)

Rover wheel base (left right) is 44.44" or 22.2188" (.5644 m) from cg

Rover wheel base (wheel to cg) is 18.0" (.4572 m) total front rear is .9144 m

Z coordinate includes an upward shift of .1m

	Left Rear Wheel				Right Rear Wheel				Left Front Wheel				Right Front Wheel				C of G		
	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z	Slip	X	Y	Z
	x_lr	y_lr	z_lr	i_lr	x_rr	y_rr	z_rr	i_rr	x_lf	y_lf	z_lf	i_lf	x_rf	y_rf	z_rf	i_rf	x_cg	y_cg	z_cg
1	-0.4572	0.5558	0.0020	0.0000	-0.4572	-0.5558	0.1980	0.0000	0.4572	0.5558	0.0020	0.0000	0.4572	-0.5558	0.1980	0.0000	0.0000	0.0000	0.5446
	-0.1524	0.5558	0.0020	0.0000	-0.1524	-0.5558	0.1980	0.0000	0.7620	0.5558	0.0020	0.0000	0.7620	-0.5558	0.1980	0.0000	0.3048	0.0000	0.5446
	0.1524	0.5558	0.0020	0.0000	0.1524	-0.5558	0.1980	0.0000	1.0668	0.5558	0.0020	0.0000	1.0668	-0.5558	0.1980	0.0000	0.6096	0.0000	0.5446
2	0.4572	0.5558	0.0020	0.0000	0.4572	-0.5558	0.1980	0.0000	1.3716	0.5558	0.0020	0.0000	1.3716	-0.5558	0.1980	0.0000	0.9144	0.0000	0.5446
	0.7620	0.5558	0.0020	0.0000	0.7620	-0.5558	0.1980	0.0000	1.6764	0.5558	0.0020	0.0000	1.6764	-0.5558	0.1980	0.0000	1.2192	0.0000	0.5446
	1.0668	0.5558	0.0020	0.0000	1.0668	-0.5558	0.1980	0.0000	1.9812	0.5558	0.0020	0.0000	1.9812	-0.5558	0.1980	0.0000	1.5240	0.0000	0.5446
3	1.3716	0.5558	0.0020	0.0000	1.3716	-0.5558	0.1980	0.0000	2.2860	0.5558	0.0020	0.0000	2.2860	-0.5558	0.1980	0.0000	1.8288	0.0000	0.5446
	1.6764	0.5558	0.0020	0.0000	1.6764	-0.5558	0.1980	0.0000	2.5908	0.5558	0.0020	0.0000	2.5908	-0.5558	0.1980	0.0000	2.1336	0.0000	0.5446
	1.9812	0.5558	0.0020	0.0000	1.9812	-0.5558	0.1980	0.0000	2.8956	0.5558	0.0020	0.0000	2.8956	-0.5558	0.1980	0.0000	2.4384	0.0000	0.5446
4	2.2860	0.5558	0.0020	0.0000	2.2860	-0.5558	0.1980	0.0000	3.2004	0.5558	0.0020	0.0000	3.2004	-0.5558	0.1980	0.0000	2.7432	0.0000	0.5446
	2.5908	0.5558	0.0020	0.0000	2.5908	-0.5558	0.1980	0.0000	3.5052	0.5558	0.0020	0.0000	3.5052	-0.5558	0.1980	0.0000	3.0480	0.0000	0.5446
	2.8956	0.5558	0.0020	0.0000	2.8956	-0.5558	0.1980	0.0000	3.8100	0.5558	0.0020	0.0000	3.8100	-0.5558	0.1980	0.0000	3.3528	0.0000	0.5446
5	3.2004	0.5558	0.0020	0.0000	3.2004	-0.5558	0.1980	0.0000	4.1148	0.5558	0.0020	0.0000	4.1148	-0.5558	0.1980	0.0000	3.6576	0.0000	0.5446
	3.5052	0.5558	0.0020	0.0000	3.5052	-0.5558	0.1980	0.0000	4.4196	0.5558	0.0020	0.0000	4.4196	-0.5558	0.1980	0.0000	3.9624	0.0000	0.5446
	3.8100	0.5558	0.0020	0.0000	3.8100	-0.5558	0.1980	0.0000	4.7244	0.5558	0.0020	0.0000	4.7244	-0.5558	0.1980	0.0000	4.2672	0.0000	0.5446
6	4.1148	0.5558	0.0020	0.0000	4.1148	-0.5558	0.1980	0.0000	5.0292	0.5558	0.0020	0.0000	5.0292	-0.5558	0.1980	0.0000	4.5720	0.0000	0.5446
	4.4196	0.5558	0.0020	0.0000	4.4196	-0.5558	0.1980	0.0000	5.3340	0.5558	0.0020	0.0000	5.3340	-0.5558	0.1980	0.0000	4.8768	0.0000	0.5446
	4.7244	0.5558	0.0020	0.0000	4.7244	-0.5558	0.1980	0.0000	5.6388	0.5558	0.0020	0.0000	5.6388	-0.5558	0.1980	0.0000	5.1816	0.0000	0.5446
7	5.0292	0.5558	0.0020	0.0000	5.0292	-0.5558	0.1980	0.0000	5.9436	0.5558	0.0020	0.0000	5.9436	-0.5558	0.1980	0.0000	5.4864	0.0000	0.5446
	5.3340	0.5558	0.0020	0.0000	5.3340	-0.5558	0.1980	0.0000	6.2484	0.5558	0.0020	0.0000	6.2484	-0.5558	0.1980	0.0000	5.7912	0.0000	0.5446
	5.6388	0.5558	0.0020	0.0000	5.6388	-0.5558	0.1980	0.0000	6.5532	0.5558	0.0020	0.0000	6.5532	-0.5558	0.1980	0.0000	6.0960	0.0000	0.5446
8	5.9436	0.5558	0.0020	0.0000	5.9436	-0.5558	0.1980	0.0000	6.8580	0.5558	0.0020	0.0000	6.8580	-0.5558	0.1980	0.0000	6.4008	0.0000	0.5446
	6.2484	0.5558	0.0020	0.0000	6.2484	-0.5558	0.1980	0.0000	7.1628	0.5558	0.0020	0.0000	7.1628	-0.5558	0.1980	0.0000	6.7056	0.0000	0.5446
	6.5532	0.5558	0.0020	0.0000	6.5532	-0.5558	0.1980	0.0000	7.4676	0.5558	0.0020	0.0000	7.4676	-0.5558	0.1980	0.0000	7.0104	0.0000	0.5446
9	6.8580	0.5558	0.0020	0.0000	6.8580	-0.5558	0.1980	0.0000	7.7724	0.5558	0.0020	0.0000	7.7724	-0.5558	0.1980	0.0000	7.3152	0.0000	0.5446
	7.1628	0.5558	0.0020	0.0000	7.1628	-0.5558	0.1980	0.0000	8.0772	0.5558	0.0020	0.0000	8.0772	-0.5558	0.1980	0.0000	7.6200	0.0000	0.5446
	7.4676	0.5558	0.0020	0.0000	7.4676	-0.5558	0.1980	0.0000	8.3820	0.5558	0.0020	0.0000	8.3820	-0.5558	0.1980	0.0000	7.9248	0.0000	0.5446
10	7.7724	0.5558	0.0020	0.0000	7.7724	-0.5558	0.1980	0.0000	8.6868	0.5558	0.0020	0.0000	8.6868	-0.5558	0.1980	0.0000	8.2296	0.0000	0.5446
	8.0772	0.5558	0.0020	0.0000	8.0772	-0.5558	0.1980	0.0000	8.9916	0.5558	0.0020	0.0000	8.9916	-0.5558	0.1980	0.0000	8.5344	0.0000	0.5446
	8.3820	0.5558	0.0020	0.0000	8.3820	-0.5558	0.1980	0.0000	9.2964	0.5558	0.0020	0.0000	9.2964	-0.5558	0.1980	0.0000	8.8392	0.0000	0.5446
11	8.6868	0.5558	0.0020	0.0000	8.6868	-0.5558	0.1980	0.0000	9.6012	0.5558	0.0020	0.0000	9.6012	-0.5558	0.1980	0.0000	9.1440	0.0000	0.5446
	8.9916	0.5558	0.0020	0.0000	8.9916	-0.5558	0.1980	0.0000	9.9060	0.5558	0.0020	0.0000	9.9060	-0.5558	0.1980	0.0000	9.4488	0.0000	0.5446

Table C.8: Sinusoidal terrain map – no slip.

Terrain Coordinates — **Sine - No Slip**

Z= .4 sin (.4 x) (radians)

Z= .4 sin (22.9183 x) (degrees)

Slope = 9.167 cos (22.9183 x)

Rover wheel base (left right) is 44.44" or 22.2188" (.5644 m) from cg

Rover wheel base (wheel to cg) is 18.0" (.4572 m) total front rear is .9144 m

	Left Rear Wheel					Right Rear Wheel					Left Front Wheel					Right Front Wheel							
	X	Y	Z	Slip	Slope	X	Y	Z	Slip	Slope	X	Y	Z	Slip	Slope	X	Y	Z	Slip	Slope	X	Y	Z
	x_lr	y_lr	z_lr	i_lr		x_rr	y_rr	z_rr	i_rr		x_lf	y_lf	z_lf	i_lf		x_rf	y_rf	z_rf	i_rf		x_cg	y_cg	z_cg
1	-0.4514	0.5644	-0.071830	0.0000	0.0000	-0.4514	-0.5644	-0.071830	0.0000	0.0000	0.4514	0.5644	0.071830	0.0000	0.0000	0.4514	-0.5644	0.071830	0.0000	0.0000	-0.0719	0.0000	-0.0115
	-0.1505	0.5644	-0.024070	0.0000	0.0000	-0.1505	-0.5644	-0.024070	0.0000	0.0000	0.7523	0.5644	0.118560	0.0000	0.0000	0.7523	-0.5644	0.118560	0.0000	0.0000	0.2290	0.0000	0.0366
	0.1504	0.5644	0.024050	0.0000	0.0000	0.1504	-0.5644	0.024050	0.0000	0.0000	1.0532	0.5644	0.163570	0.0000	0.0000	1.0532	-0.5644	0.163570	0.0000	0.0000	0.5299	0.0000	0.0842
2	0.4513	0.5644	0.071820	0.0000	0.0000	0.4513	-0.5644	0.071820	0.0000	0.0000	1.3541	0.5644	0.206220	0.0000	0.0000	1.3541	-0.5644	0.206220	0.0000	0.0000	0.8308	0.0000	0.1305
	0.7522	0.5644	0.118540	0.0000	0.0000	0.7522	-0.5644	0.118540	0.0000	0.0000	1.6550	0.5644	0.245880	0.0000	0.0000	1.6550	-0.5644	0.245880	0.0000	0.0000	1.1317	0.0000	0.1745
	1.0531	0.5644	0.163560	0.0000	0.0000	1.0531	-0.5644	0.163560	0.0000	0.0000	1.9559	0.5644	0.281970	0.0000	0.0000	1.9559	-0.5644	0.281970	0.0000	0.0000	1.4326	0.0000	0.2169
3	1.3540	0.5644	0.206200	0.0000	0.0000	1.3540	-0.5644	0.206200	0.0000	0.0000	2.2568	0.5644	0.314000	0.0000	0.0000	2.2568	-0.5644	0.314000	0.0000	0.0000	1.7335	0.0000	0.2557
	1.6549	0.5644	0.245870	0.0000	0.0000	1.6549	-0.5644	0.245870	0.0000	0.0000	2.5577	0.5644	0.341480	0.0000	0.0000	2.5577	-0.5644	0.341480	0.0000	0.0000	2.0344	0.0000	0.2907
	1.9558	0.5644	0.281970	0.0000	0.0000	1.9558	-0.5644	0.281970	0.0000	0.0000	2.8586	0.5644	0.364020	0.0000	0.0000	2.8586	-0.5644	0.364020	0.0000	0.0000	2.3353	0.0000	0.3216
4	2.2567	0.5644	0.314000	0.0000	0.0000	2.2567	-0.5644	0.314000	0.0000	0.0000	3.1595	0.5644	0.381290	0.0000	0.0000	3.1595	-0.5644	0.381290	0.0000	0.0000	2.6362	0.0000	0.3479
	2.5576	0.5644	0.341480	0.0000	0.0000	2.5576	-0.5644	0.341480	0.0000	0.0000	3.4604	0.5644	0.393050	0.0000	0.0000	3.4604	-0.5644	0.393050	0.0000	0.0000	2.9371	0.0000	0.3691
	2.8585	0.5644	0.364020	0.0000	0.0000	2.8585	-0.5644	0.364020	0.0000	0.0000	3.7613	0.5644	0.399120	0.0000	0.0000	3.7613	-0.5644	0.399120	0.0000	0.0000	3.2380	0.0000	0.3849
5	3.1594	0.5644	0.381290	0.0000	0.0000	3.1594	-0.5644	0.381290	0.0000	0.0000	4.0622	0.5644	0.399420	0.0000	0.0000	4.0622	-0.5644	0.399420	0.0000	0.0000	3.5389	0.0000	0.3952
	3.4603	0.5644	0.393050	0.0000	0.0000	3.4603	-0.5644	0.393050	0.0000	0.0000	4.3631	0.5644	0.393930	0.0000	0.0000	4.3631	-0.5644	0.393930	0.0000	0.0000	3.8398	0.0000	0.3998
	3.7612	0.5644	0.399120	0.0000	0.0000	3.7612	-0.5644	0.399120	0.0000	0.0000	4.6640	0.5644	0.382750	0.0000	0.0000	4.6640	-0.5644	0.382750	0.0000	0.0000	4.1407	0.0000	0.3985
6	4.0621	0.5644	0.399420	0.0000	0.0000	4.0621	-0.5644	0.399420	0.0000	0.0000	4.9649	0.5644	0.366030	0.0000	0.0000	4.9649	-0.5644	0.366030	0.0000	0.0000	4.4416	0.0000	0.3916
	4.3630	0.5644	0.393930	0.0000	0.0000	4.3630	-0.5644	0.393930	0.0000	0.0000	5.2658	0.5644	0.344010	0.0000	0.0000	5.2658	-0.5644	0.344010	0.0000	0.0000	4.7425	0.0000	0.3789
	4.6639	0.5644	0.382750	0.0000	0.0000	4.6639	-0.5644	0.382750	0.0000	0.0000	5.5667	0.5644	0.317010	0.0000	0.0000	5.5667	-0.5644	0.317010	0.0000	0.0000	5.0434	0.0000	0.3608
7	4.9648	0.5644	0.366030	0.0000	0.0000	4.9648	-0.5644	0.366030	0.0000	0.0000	5.8676	0.5644	0.285430	0.0000	0.0000	5.8676	-0.5644	0.285430	0.0000	0.0000	5.3443	0.0000	0.3374
	5.2657	0.5644	0.344010	0.0000	0.0000	5.2657	-0.5644	0.344010	0.0000	0.0000	6.1685	0.5644	0.249720	0.0000	0.0000	6.1685	-0.5644	0.249720	0.0000	0.0000	5.6452	0.0000	0.3092
	5.5666	0.5644	0.317010	0.0000	0.0000	5.5666	-0.5644	0.317010	0.0000	0.0000	6.4694	0.5644	0.210400	0.0000	0.0000	6.4694	-0.5644	0.210400	0.0000	0.0000	5.9461	0.0000	0.2765
8	5.8675	0.5644	0.285430	0.0000	0.0000	5.8675	-0.5644	0.285430	0.0000	0.0000	6.7703	0.5644	0.168020	0.0000	0.0000	6.7703	-0.5644	0.168020	0.0000	0.0000	6.2470	0.0000	0.2398
	6.1684	0.5644	0.249720	0.0000	0.0000	6.1684	-0.5644	0.249720	0.0000	0.0000	7.0712	0.5644	0.123220	0.0000	0.0000	7.0712	-0.5644	0.123220	0.0000	0.0000	6.5479	0.0000	0.1996
	6.4693	0.5644	0.210400	0.0000	0.0000	6.4693	-0.5644	0.210400	0.0000	0.0000	7.3721	0.5644	0.076640	0.0000	0.0000	7.3721	-0.5644	0.076640	0.0000	0.0000	6.8488	0.0000	0.1565
9	6.7702	0.5644	0.168020	0.0000	0.0000	6.7702	-0.5644	0.168020	0.0000	0.0000	7.6730	0.5644	0.028950	0.0000	0.0000	7.6730	-0.5644	0.028950	0.0000	0.0000	7.1497	0.0000	0.1112
	7.0711	0.5644	0.123220	0.0000	0.0000	7.0711	-0.5644	0.123220	0.0000	0.0000	7.9739	0.5644	-0.019160	0.0000	0.0000	7.9739	-0.5644	########	0.0000	0.0000	7.4506	0.0000	0.0643
	7.3720	0.5644	0.076640	0.0000	0.0000	7.3720	-0.5644	0.076640	0.0000	0.0000	8.2748	0.5644	-0.067000	0.0000	0.0000	8.2748	-0.5644	########	0.0000	0.0000	7.7515	0.0000	0.0164
10	7.6729	0.5644	0.028950	0.0000	0.0000	7.6729	-0.5644	0.028950	0.0000	0.0000	8.5757	0.5644	-0.113860	0.0000	0.0000	8.5757	-0.5644	########	0.0000	0.0000	8.0524	0.0000	-0.0317
	7.9738	0.5644	-0.019160	0.0000	0.0000	7.9738	-0.5644	-0.019160	0.0000	0.0000	8.8766	0.5644	-0.159080	0.0000	0.0000	8.8766	-0.5644	########	0.0000	0.0000	8.3533	0.0000	-0.0794
	8.2747	0.5644	-0.067000	0.0000	0.0000	8.2747	-0.5644	-0.067000	0.0000	0.0000	9.1775	0.5644	-0.202010	0.0000	0.0000	9.1775	-0.5644	########	0.0000	0.0000	8.6542	0.0000	-0.1259
11	8.5756	0.5644	-0.113860	0.0000	0.0000	8.5756	-0.5644	-0.113860	0.0000	0.0000	9.4784	0.5644	-0.242000	0.0000	0.0000	9.4784	-0.5644	########	0.0000	0.0000	8.9551	0.0000	-0.1705
	8.8765	0.5644	-0.159080	0.0000	0.0000	8.8765	-0.5644	-0.159080	0.0000	0.0000	9.7793	0.5644	-0.278490	0.0000	0.0000	9.7793	-0.5644	########	0.0000	0.0000	9.2560	0.0000	-0.2127

Appendix D - Dynamic Analysis and Data

This appendix contains the La Grange formulation along with additional results from preliminary analysis.

D.1 La Grange Formulation

The dynamics of the rover are developed in terms of the position and time derivatives of its joint angles as [65] illustrated by Equation 5.1 in its torque form.

$$\tau = J^T(q)\left[M_X(q)\ddot{X} \;+\; V_X(q,\dot{q}) + G_X(q)\right] \tag{5.1}$$

Examination of Equation 5.1 reveals the different components of a dynamic analysis, with $J^T(q)$ being the transpose of the Jacobian, which is dependent on the joint displacements and displacement rates, given here as q and $\dot{q}$. $M_X(q)$ represents the mass matrix with respect to the x coordinate in the world frame and is multiplied by the acceleration column vector, $\ddot{X}$. The Coriolis and centrifugal effects are jointly represented by $V_X(q,\dot{q})$, whereas the contributions from gravity are given by $G_X(q)$. However, it can also be used in the expanded version as Equation 5.2, with specific matrices for the Coriolis and Centrifugal coefficients, given by $B_X(q)$ and $C_X(\dot{q})$, respectively.

$$\tau = J^T(q)\,[J^{-T}(q)\,M(q)\,J^{-1}(q)]\ddot{X} \;+\; B_X(q)[\dot{q}\quad\dot{q}] \;+\; C_X(\dot{q})[\dot{q}]^2 \;+\; G(q) \tag{5.2}$$

The two new vectors, $[\dot{q}\quad\dot{q}]$ and $[\dot{q}]^2$ are vectors of joint velocity products and joint velocity, and are defined as follows:

$$[\dot{q}\quad\dot{q}] = [\dot{q}_1\dot{q}_2\quad\dot{q}_2\dot{q}_3\quad\dots\quad\dot{q}_{n-1}\dot{q}_n] \tag{5.3}$$

$$[\dot{q}]^2 = [\dot{q}_1^2 \quad \dot{q}_2^2 \quad \dots \quad \dot{q}_n^2] \tag{5.4}$$

The dynamic torque equation can be derived using either the La Grange or Newton-Euler method from which the above equations are formed by grouping the derived terms (ie. group the acceleration terms, group the displacement terms, etc). The La Grange method, as the name implies, requires that the Lagrangian of the system be determined and is often referred to as an energy method [67], the Lagrangian being the difference between the kinetic and potential energy of the system. Not only is this method built around the concept of energy, it is also advantageous since the inertia parameters have linearity, along with skew symmetry and passivity characteristics exhibited by the inertia matrix [67]. These features make this method more suited for application in feedback control, if one desires to take the work in that direction. It also allows for deformation to be modeled in this approach. The Newton-Euler method, on the other hand, has its basis in Newton's 2nd law and, as such the formulation is compiled through the forces and moments which dynamically describe the system. Unlike the La Grange method, the links in the Newton-Euler method are all individually analysed for force and torque balance, from which the resulting equations must be recursively solved. This involves first determining joint poses, initial velocities and accelerations which enable the forces and moments to be computed from the resulting angular velocity and accelerations plus the linear acceleration.

For the work presented in this thesis, the La Grange method was selected for its advantages and considering the complexity of the system (rover) to be analysed.

Additionally, the La Grange method is more flexible in allowing the user to choose the generalised coordinates and doesn't require all the constraint forces to be accounted for, which is especially convenient when the problem has a multitude of constraints [67]. As such, it tends to be more commonly used in multibody dynamic platforms.

In applying the La Grange method, the equation for torque becomes that of Equation 5.5, wherein the torque can be computed as the difference of the time derivative of the partial derivative of the Lagrangian with respect to joint velocity rates and the partial derivative of the Lagrangian with respect to joint displacements.

$$\tau_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \tag{5.5}$$

As shown by Equation 5.5, to obtain the desired equation for torque, the Lagrangian must be computed. It can also be observed that to obtain Equation 5.5, it is required to have the complete pose (joint displacements) along with the joint rates from an earlier kinematic analysis. For this work, both of those were derived and computed in Chapter 4. The next piece to compute is that of the Lagrangian, for which the formula is given by Equation 5.6.

$$L = \sum_{j=1}^{n}\left(K_j - P_j\right) \tag{5.6}$$

Equation 5.6 confirms the definition of the Lagrangian as the sum of the differences in kinetic (K) and potential energy (P) as one goes through the kinematic pairs of the system.

Beginning with the potential energy term, P, it can be defined in Equation 5.7.

$$P_j = m_j g z_j \tag{5.7}$$

Following the standard definition, potential energy is dependent on the vertical displacement, z, experienced by an object of weight $m_j g$. Note that the displacement variable depends upon the chosen world coordinate frame definition – whichever direction corresponds to the vertical displacement is chosen and care is taken to note the direction of gravity. For the system presented in Chapter 4, the component will remain to be z with the datum at the world origin frame. These z components were found by concatenating the homogeneous transform (T) matrices from the world reference frame to each particular link, and extracting the cell [3, 4] from the resultant matrix. For example, to obtain the z component of the chassis centre, the concatenation of matrices is as follows in Equation 5.8.

$$T_6^W = T_0^W T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 \tag{5.8}$$

$$= \begin{bmatrix} -\mathrm{c}\,\varphi_y\,\mathrm{s}\,\varphi_p\,c\varphi_r - \mathrm{s}\,\varphi_y\,\mathrm{s}\,\varphi_r & c\varphi_y c\varphi_p & -\mathrm{c}\,\varphi_y\,\mathrm{s}\,\varphi_p\,s\varphi_r + s\varphi_y c\varphi_r & X_{trans} \\ -s\varphi_y\,\mathrm{s}\,\varphi_p c\varphi_r + c\varphi_y\,\mathrm{s}\,\varphi_r & s\varphi_y c\varphi_p & -\mathrm{s}\,\varphi_y\,\mathrm{s}\,\varphi_p\,s\varphi_r - c\varphi_y c\varphi_r & Y_{trans} \\ -c\varphi_p c\varphi_r & -s\varphi_p & -c\varphi_p\,\mathrm{s}\,\varphi_r & Z_{trans} + h_{CoG} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

All symbolic multiplication was performed in Maple. Following the procedure mentioned above, the z component for the chassis centre is extracted as cell [3, 4] from Equation 5.8 and yields Equation 5.9.

$$z_{chas} = Z_{trans} + h_{CoG} \tag{5.9}$$

Therefore, the potential energy is

$$P_{chas} = m_{chas} g (Z_{trans} + h_{CoG}) \tag{5.10}$$

Having generated the kinematic models, the required z components and subsequent potential energy terms are fairly easily obtained.

Next, the kinetic energy of each link is determined. Similar to the potential energy, it follows the basic definition as Equation 5.11 [67].

$$K_j = \frac{1}{2} m_j (\vec{v}_1 \cdot \vec{v}_1)^2 = \frac{1}{2} m_j \left(\dot{x}_j^2 + \dot{y}_j^2 + \dot{z}_j^2 \right) \tag{5.11}$$

With Equation 5.11, mass is easily determined, with the velocity components of the end effector obtained from velocity Jacobians or computing the derivative of the position equations. Although Equation 5.11 appears in the majority of textbooks and sample

problems, its popularity is due to the majority of problems applied being those that can be treated as a point mass. While Equation 5.11 is applicable to simple mechanisms (ie. pendulums), one cannot justify applying it to a large four-wheel rover, where the inertia will have an impact on the motion of the system. As such, the kinetic energy equation is modified to account for inertia, as detailed in Equation 5.12.

$$K = \frac{1}{2}\dot{q}^T \sum_j^n \left[m_j J_{v,j}^T(q) J_{v,j}(q) + J_{\omega,j}^T(q) R_j(q) I_j R_j^T(q) J_{\omega,j}(q) \right] \dot{q} \tag{5.12}$$

$$K = \frac{1}{2}\dot{q}^T D(q) \dot{q}$$

$D(q)$ is also referred to as the inertia matrix. Attempts were made to obtain the kinetic energy terms, however the complexity of the rover meant that many assumptions were made to grossly simplify it, but the terms still were difficult to obtain. After some investigation, it was decided to drop this line of pursuit due to the level of effort required, and because the solution obtained would not be accurate. This rationale is why most companies opt to use some form of multibody software because it's not only accurate, but easier to use. Based upon recommendations, it was decided to use a different approach to generate a dynamic model. The decision made was to use SimMechanics as described in the next section; however, it is still important to go through La Grange derivation of the dynamic equation, as this is still applicable as the basis of the multibody physics platforms.

D.2 Additional Terramechanics Runs and Simulation Results

It should be noted that prior to running the single wheel terramechanics model of Irani et al [37], the model was run for the different cases and modifications using the original model parameters, to serve as a guideline for the trends and expected results.

D.1.1 Unmodified Terramechanics Model Results

Case: Single Wheel Mass (15 kg)

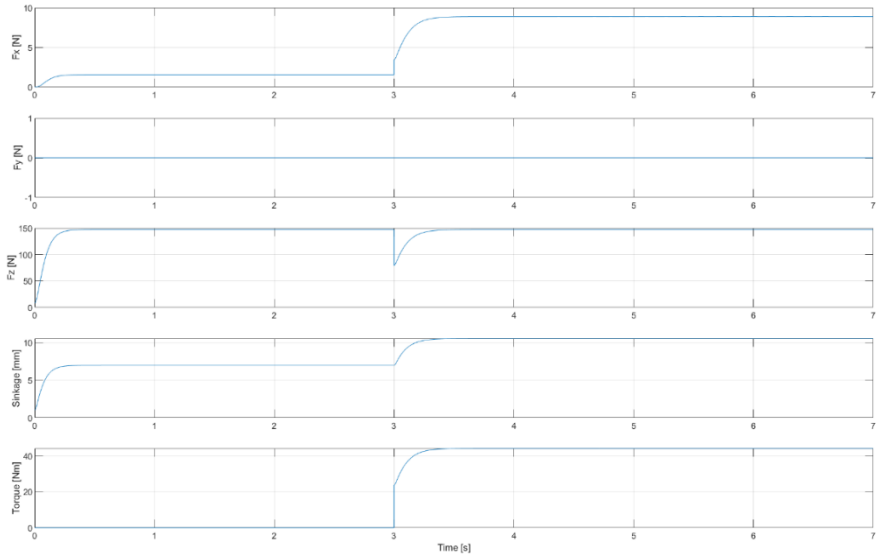


Figure D.1: Unmodified terramechanics single wheel model results for i =0.003.

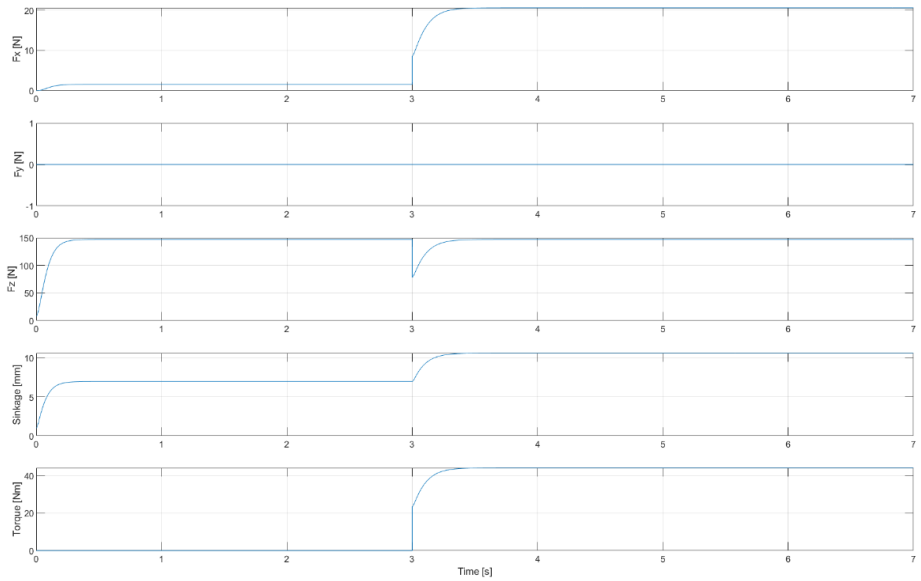


Figure D.2: Unmodified terramechanics single wheel model results for i =0.05.

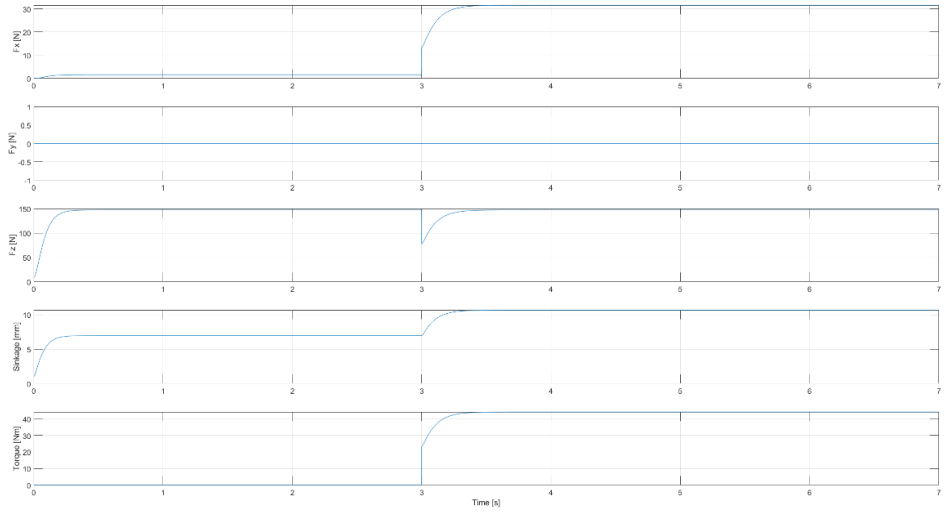


Figure D.3: Unmodified terramechanics single wheel model results for i =0.1.

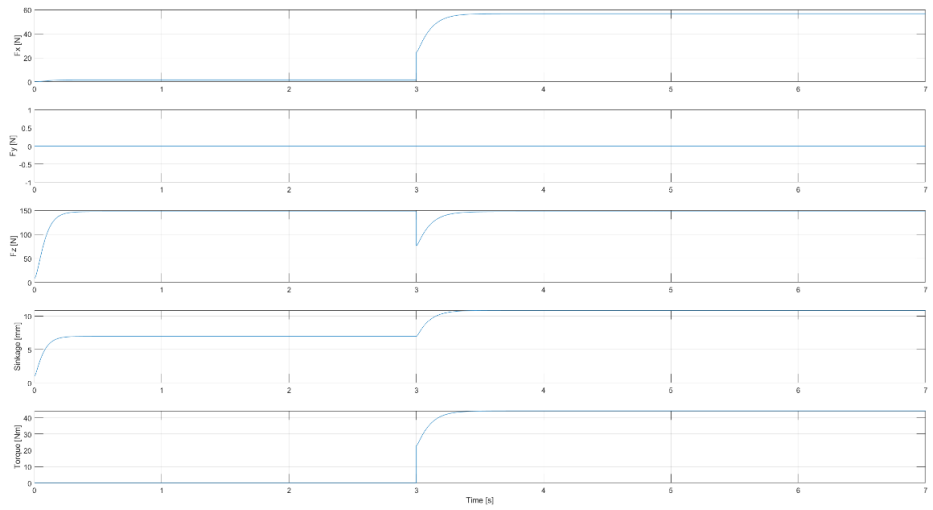


Figure D.4: Unmodified terramechanics single wheel model results for i =0.25.

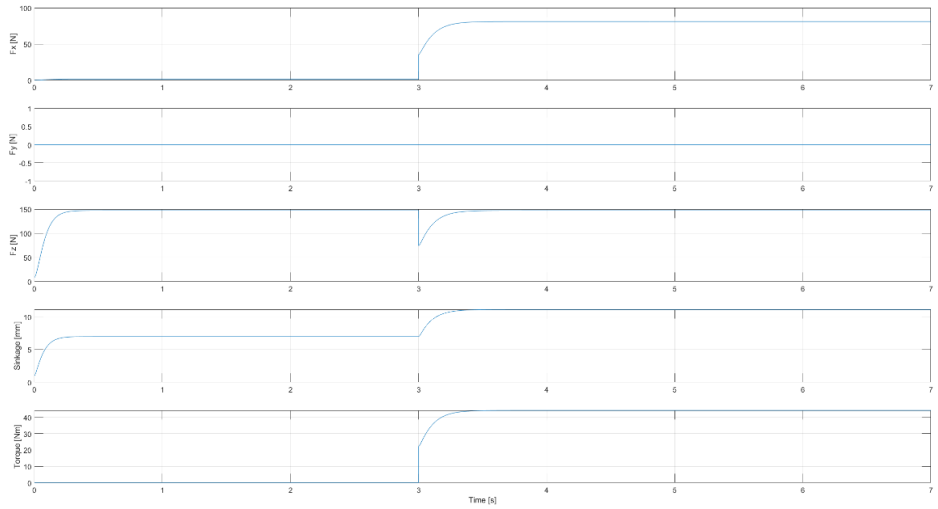


Figure D.5: Unmodified terramechanics single wheel model results for i =0.5

Case: Single Wheel with ¼ Rover Mass (~115 kg)

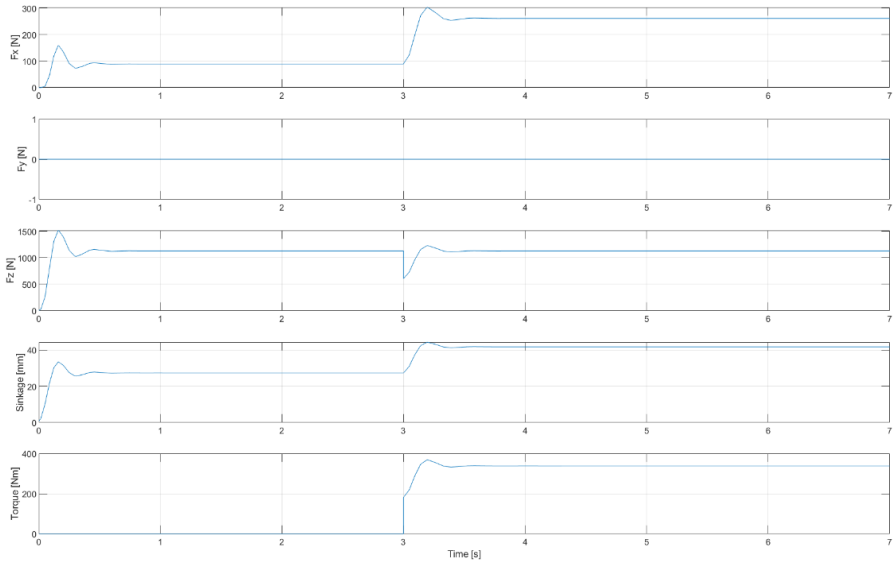


Figure D.6: Unmodified terramechanics single wheel model results for i =0.003.

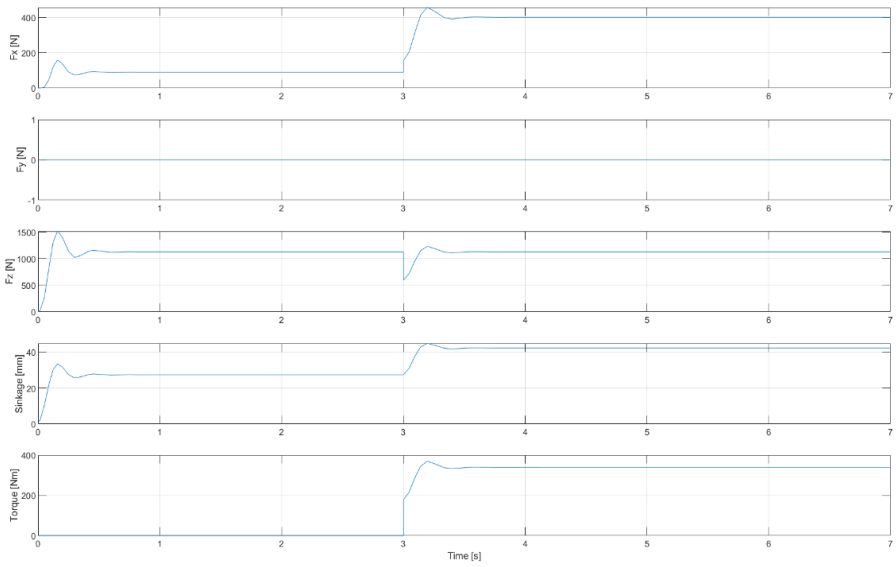


Figure D.7: Unmodified terramechanics single wheel model results for i =0.05.

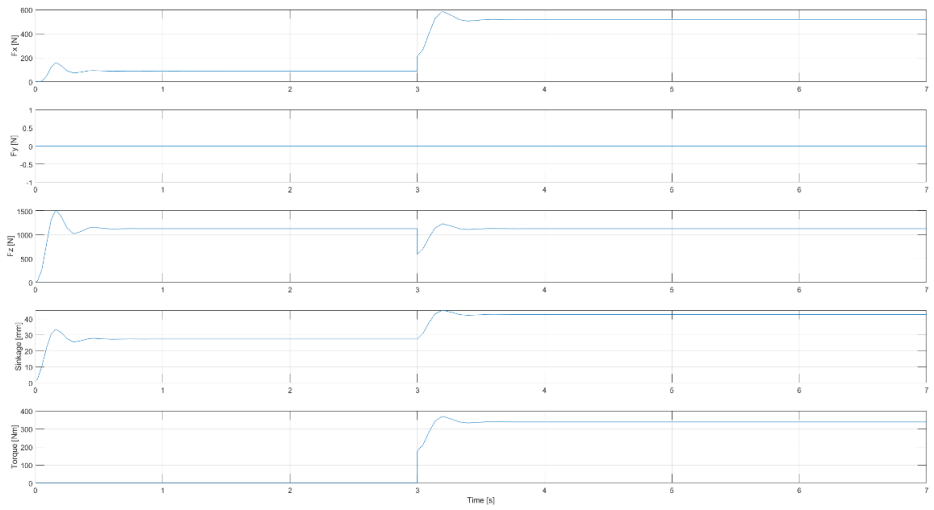


Figure D.8: Unmodified terramechanics single wheel model results for i =0.1.

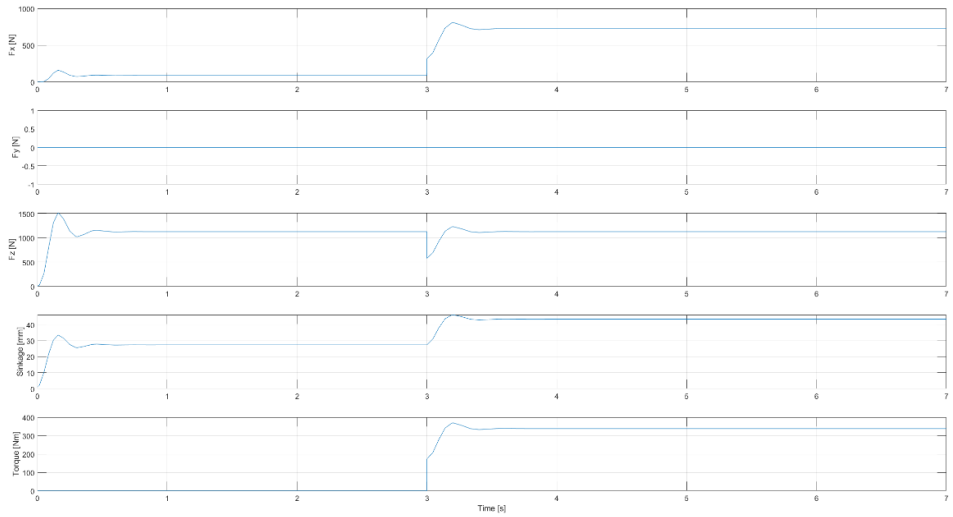


Figure D.9: Unmodified terramechanics single wheel model results for i =0.25.

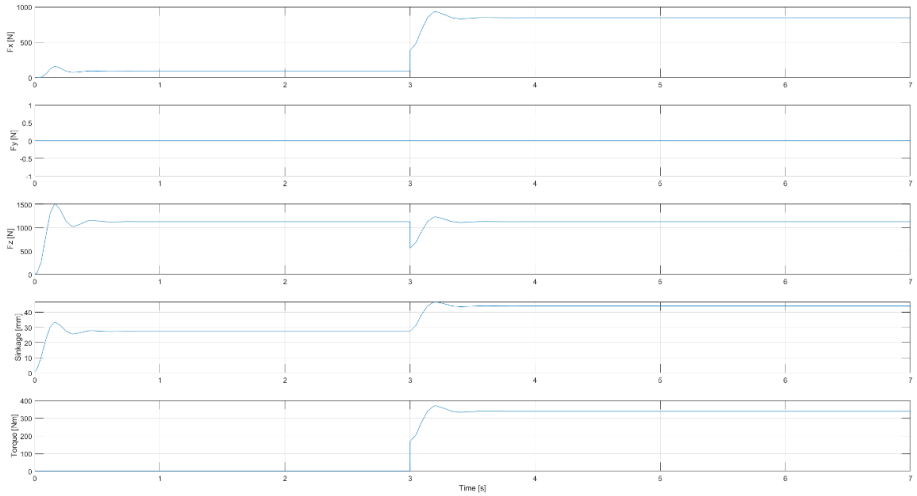


Figure D.10: Unmodified terramechanics single wheel model results for i =0.5

D.1.2 Modified Terramechanics Model Results

Case: Single Wheel Mass (15 kg)

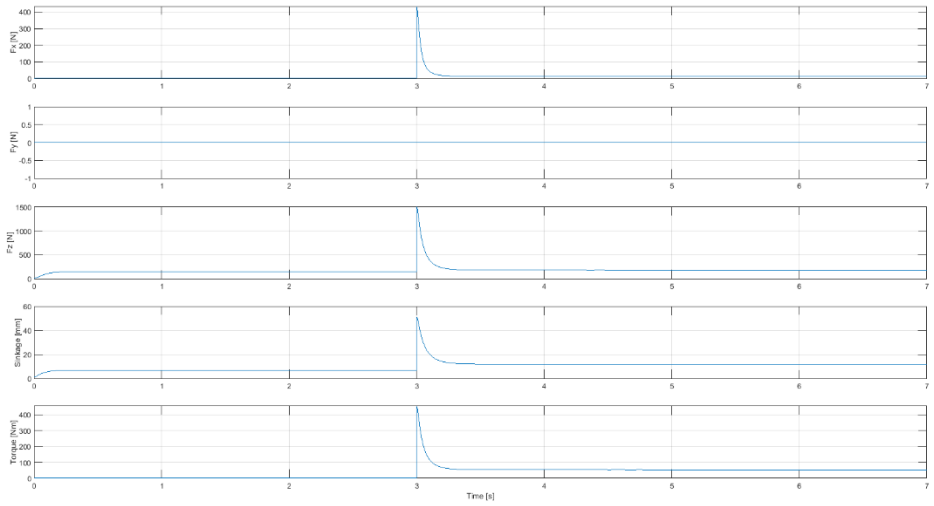


Figure D.11: Modified terramechanics single wheel model results for i =0.003.

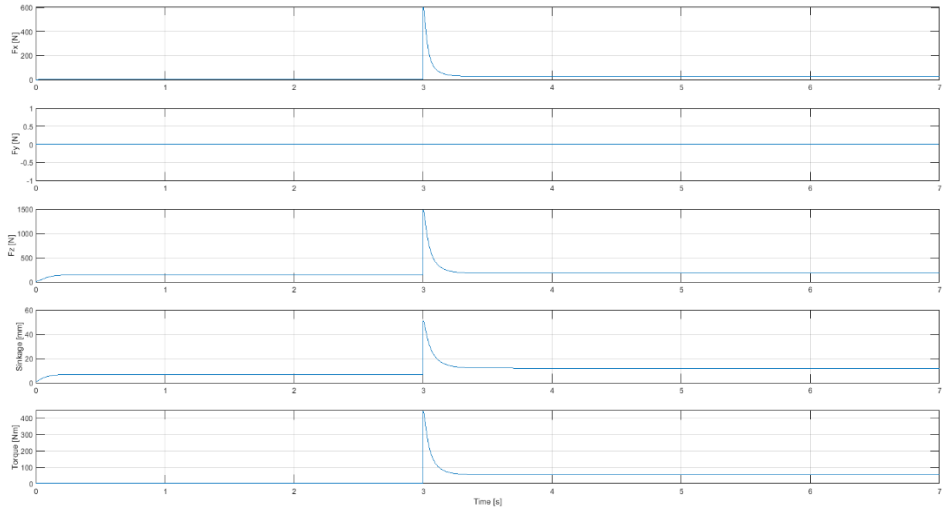


Figure D.12: Modified terramechanics single wheel model results for i =0.05.

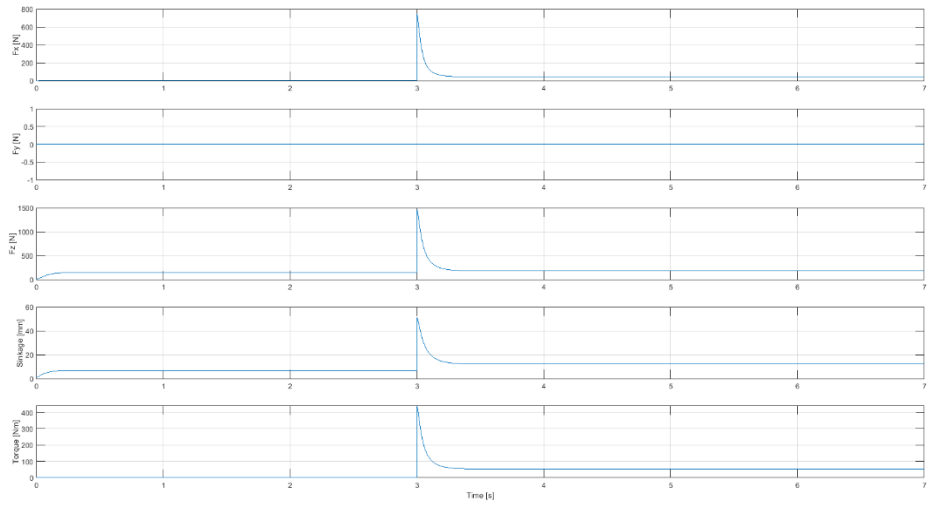


Figure D.13: Modified terramechanics single wheel model results for i =0.1.

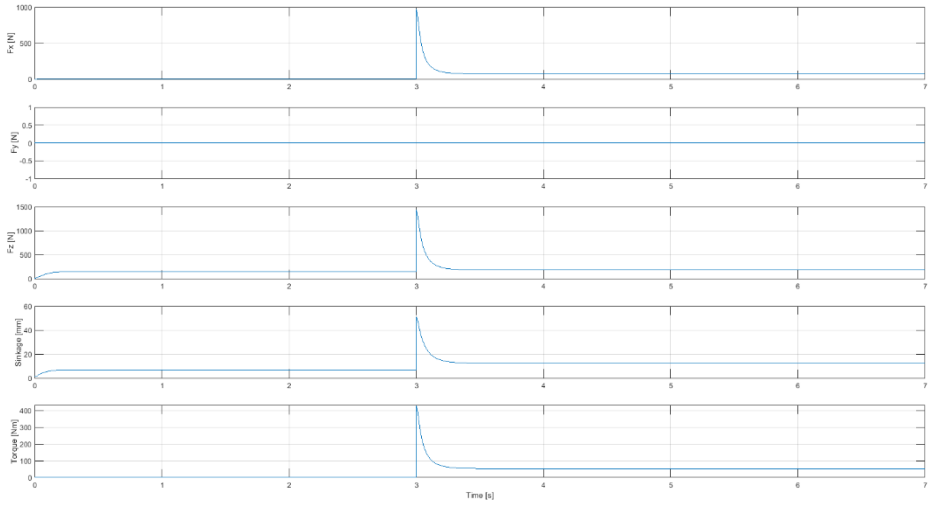


Figure D.14: Modified terramechanics single wheel model results for i =0.25.

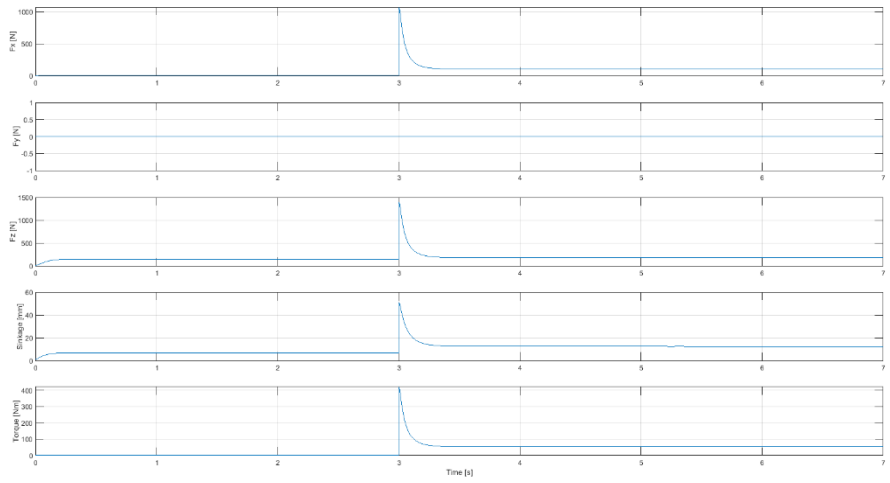


Figure D.15: Modified terramechanics single wheel model results for i =0.5

Case: Single Wheel with ¼ Rover Mass (~115 kg)

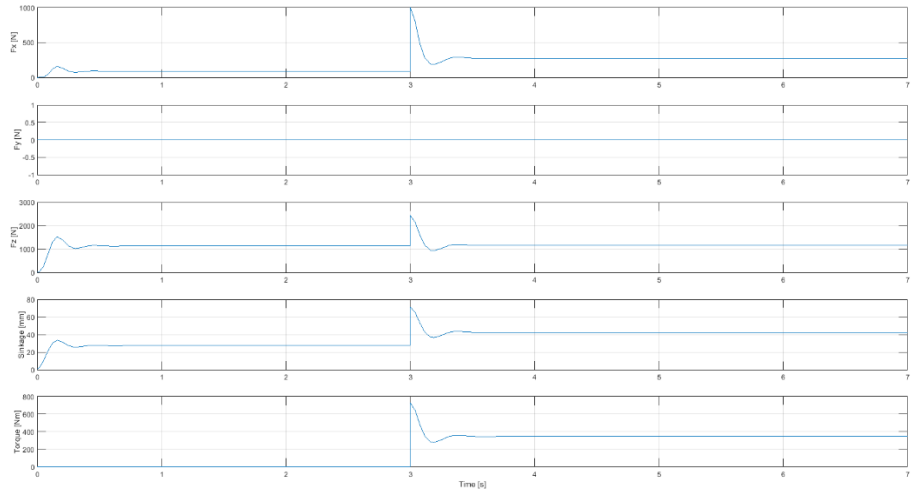


Figure D.16: Modified terramechanics single wheel model results for i =0.003.

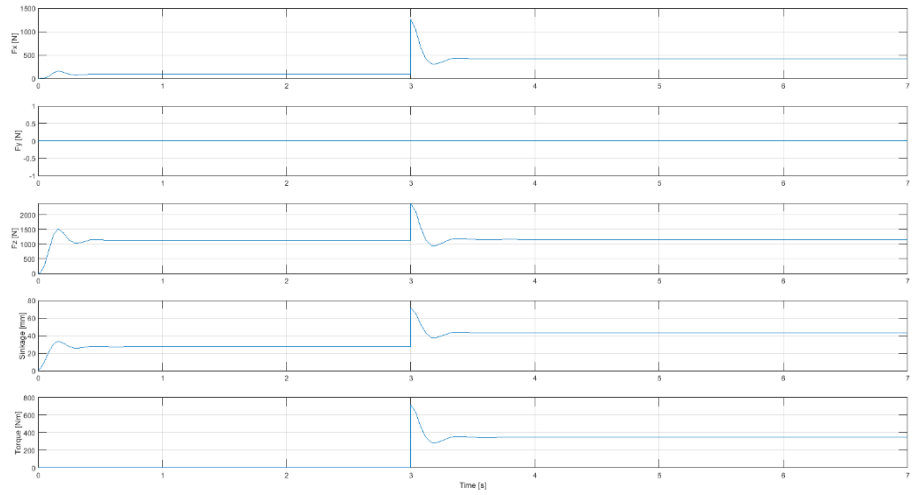


Figure D.17: Modified terramechanics single wheel model results for i =0.05.

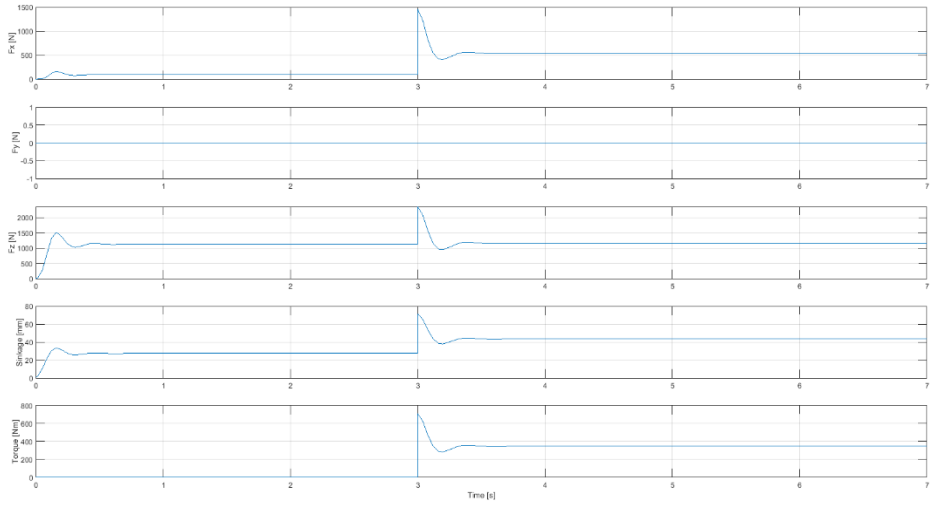


Figure D.18: Modified terramechanics single wheel model results for i =0.1.

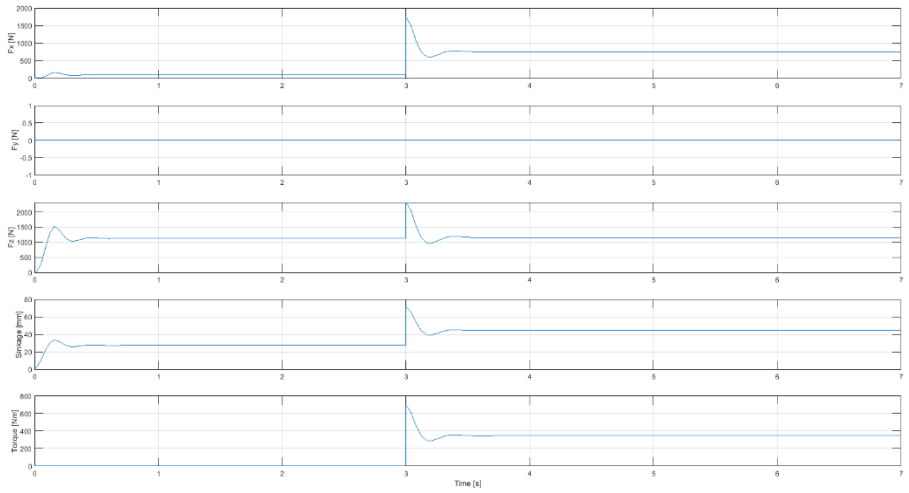


Figure D.19: Modified terramechanics single wheel model results for i =0.25.

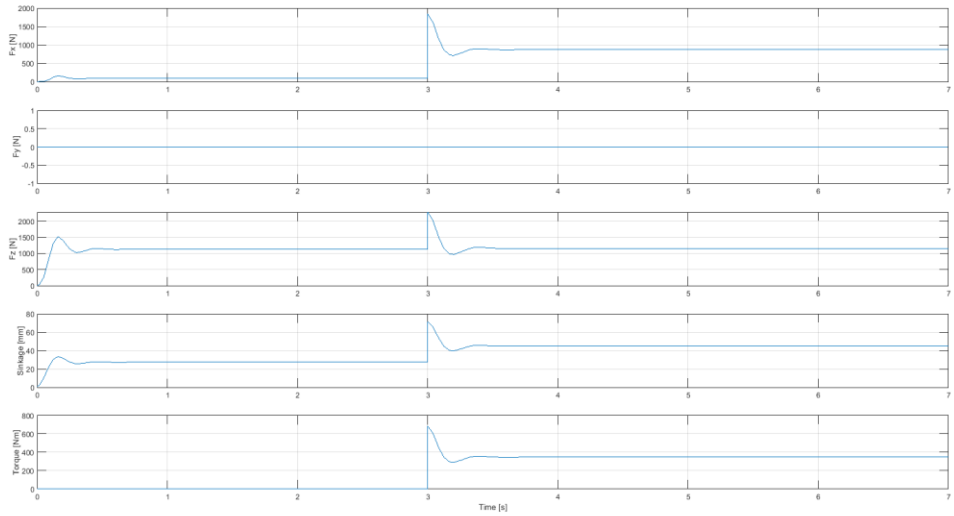


Figure D.20: Modified terramechanics single wheel model results for i =0.5